HAWASSA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

A THESIS SUBMITTED TO THE PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF MASTERS OF SCIENCE (MSc) IN COMPUTER SCIENCE

DESSALEGN MENGESHA

HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

March, 2023

RESEARCH TITLE

COLLABORATIVE APPROACH OF AGILE AND DEVOPS FOR CONTINUOUS DELIVERY OF QUALITY SOFTWARE

By DESSALEGN MENGESHA

MAJOR ADVISOR: DEGIF TEKA (PhD)

CO-ADVISOR: DANIEL TESFAYE

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
HAWASSA UNIVERSITY INSTITUTE OF TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

March, 2023

ADVISOR'S APPROVAL SHEET

This is to certify that this thesis entitled *Collaborative Approach of Agile and DevOps for Continuous Delivery of Quality Software* submitted in partial fulfillment of the requirements for the award of the degree of Masters in Computer Science was done by Dessalegn Mengesha under my guidance. Hence, I recommend that the student has fulfilled the requirements and so hereby submit the thesis to the department.

Name of the Advisor	**Signature**	**Date**
_____	_____	_____

APPROVAL SHEET-II

We, the undersigned, members of the Board of Examiners of the final open defense by Dessalegn Mengesha have read and evaluated his/her thesis entitled “Collaborative Approach of Agile and DevOps for Continuous Delivery of Quality Software”, and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

_____	_____	_____
Name of Major Advisor	Signature	Date
_____ _	_____	_____
Name of Internal Examiner-I	Signature	Date
_____	_____	_____
Name of Internal Examiner-II	Signature	Date
_____	_____	_____
Name of External examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the School of Graduate Studies (SGS) through the Department/School Graduate Committee (DGC/SGC) of the candidate's department.

Stamp of SGS Date: _____

Remark

- Use this form to submit the thesis with ***MINOR CORRECTION*** suggested by the examining board
- 3 copies

ACKNOWLEDGEMENTS

First of all, I would like to give abundant thanks to the almighty God for his holistic precaution and fatherhood throughout my study. Secondly, words cannot express my gratitude to my advisor Dr. Degif Teka for his commitment to advising and for his continuous support during the research work. He advised me politely and respectfully. In addition to academic knowledge that gives me a great lesson in my personal life.

My lovely family did unforgettable support in my academic life. I did nothing without their support and any achievements of me are due to their unbreakable payer and support. So I would like a great thanks to my family specially my lovely wife Seble Tafesse.

I would also like to give appreciation to my instructors who equipped me with untarnished knowledge and wisdom during my graduate class study. Some of them are instructed me during undergraduate classes and have their great contribution to my today's professional life. Specially, Dr. Tesfaye Bayu shows me a better roadmap for software development with a rapid application development approach.

Finally but not least place, I would like to give great respect to Hawassa University application development Team members and the team leader Mr. Getachew Kumara for their support during my research work.

STATEMENT OF THE AUTHOUR

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis / have been duly acknowledged.

Name: ……..……. Signature: …….……

Place: Institute of Technology, Hawassa University, Hawassa

Date of Submission: …….

List of Abbreviations and Acronyms

DSDM----------Dynamic Systems Development Method

DAD----------Disciplined Agile Delivery

LeSS----------Large Scale Scrum

SAFe----------Scaled Agile Framework

HU----------Hawassa University

ICT----------Information Communication Technology

Dev----------Development

Ops----------Operations

CRUD----------Create, Read, Update, Delete

OSAS----------Online Student Advisory System

EMS----------Ekub Management System

ER----------Entity Relation

List of Tables

List of Figures

Table of Contents

Abstract

We are in the era of high demand for quality software in many organizations in order to achieve their organizational goals. Many organizations around the globe have shown great interest in the automation of their business processes. This in turn causes emerging and improvement of different software development methodologies and the way of service provision dramatically. Among those methodologies, Agile Software Development Methodologies and DevOps culture/tool have become more popular due to their capability on supporting rapid software development, continuous integration, and continuous delivery. Even though the two methodologies are complementary and have their own significant role in the software development lifecycle, using the two approaches independently will not bring development process improvement to the optimum level. Contextualizing the software development process enables the practitioners to improve their development process and for better productivity. The objective of this thesis work is to integrate the two approaches together with minor modifications to the DevOps team structure by extending the role of the DevOps team to the development environment. The research is conducted as experimental research and the evaluation was done by using two working projects, one using classical Agile as a control group and the other by integrated approach of Agile and DevOps as an experimental group. The number of changes accepted and developed and the number of deliveries in a specific period of time are used as measurement parameters. The experiment was done using students who joined Hawassa University Application Development Team for practical attachments. The findings of the experiment demonstrate that the experimental group project, which utilized agile methodologies in conjunction with DevOps practices, achieved superior outcomes compared to the control group project, which relied on the department's standard Agile/Scrum approach. This improvement was evident in metrics such as accepted changes and committed deliveries. Furthermore, the guideline applied to the experimental group project was refined and is included in this paper to serve as a valuable resource for future researchers and developers.

Chapter One

Introduction

1.1 Background

The term “Software Development” subsumes all types of software development that can be categorized based on different criteria including criticality level of development, development technology, deployment platform, size and complexity of the development, and the number and skills of the developers. Concerning the criticality level of developing software, some software is error-tolerable but others never tolerate any error and incompleteness. Error-tolerable software can be improved over time but critical software needs immediate solutions for any errors occurred on the software. Besides the criticality of the software, some customers demand software to be developed in a short period of time due to challenges faced in their business area. Others need solutions for their business activities through extended time. Based on the size and complexity of development, we can categorize software as small-size software, medium-sized software, and large-sized software. The size of the software is indicated either by the number of modules of the software or by the size of the modules of the software. When the number and size of modules increase the complexity of the software also increases. This is due to interconnection and sharing functionalities between some modules of the software. The large-sized software needs well-organized project management and development teams with high-level experts. When the software size and complexity become smaller it is not difficult to manage the development process with small-sized team members and minimum project management skills. The size and experience of developers also have a significant impact on the success of software development projects. In order to handle software development projects in the desired manner and to be productive; we have to follow some software development methodology. Software development methodologies are created and improved from time to time due to emerging of large and sophisticated software development projects. Developers always look for an approach that improves the development processes, minimizing development time and cost with available manpower while improving the quality of the developing software. When the software development project size becomes larger and larger it is difficult to handle all activities without declaring formal procedures and principles. As a result, different software development

approaches emerged and evaluated in the last decades. The researcher categorizes the methodologies into two main categories based on the agility of the methodologies either as flexible or as rigged. The earlier methodologies like waterfall methodologies do not support agility whereas the modern ones specifically, Agile and DevOps are suitable for projects with a very dynamic environment where continuous development, continuous integration, and continuous delivery is demanded. Agility in developing software is required in either of the three cases. The first one is customer requirement change, sometimes customers change their requirements based on business logic amendments while the software is in the development process. The second one is based on design changes by developers due to either code optimization or security issue. The third reason why the changes happen in developing software is due to its outdated development technology. When the size or scope of development becomes large with a limited number of developers and a poor project-managed environment there will be a probability of becoming obsolete for both developing software and development technology before the development process is completed. For example, in the past few years, desktop applications are primarily developed and used at the national level widely but nowadays web-based applications and mobile applications are becoming more familiar due to their accessibility and ease of use. For example, in the case of Hawassa University Student Information Management System development (where the researcher working), the first registrar system was developed by using a desktop application. Today the system can provide service via different channels including web-based applications, Telegram boot, and short SMS codes. It is in such applications that agile software development is popularly applied due to small-sized and medium projects that fit the agile requirement of small and collocated teams. In order to have a clear understanding of software development methodologies some of the methodologies are discussed in the following sub-topics.

1.1.1 Waterfall

The waterfall model follows the classic SDLC model which is a heavyweight process and consists of different sequential phases including planning, analysis, design, development or coding, testing, implementation, and maintenance. All these steps come one after the other. Before completing the prior phase it is impossible to work on the next phase. Any changes that are requested in the middle of the SDLC either by the customer or by developers can’t be accepted until the life is completed. The waterfall model is document intensive. Each step and

activity in the development life cycle is documented either electronically or on paper by using different descriptive diagrams (like Entity Relation Diagrams and Data Flow Diagrams) and formal narrations. Any change made to the developing software should be adjusted on all related diagrams and narrations. In general waterfall model is not suitable for projects that are planned to be delivered in a short period with less degree of certainty.

Waterfall methodology takes the first place in the history of software development methodologies. Before the waterfall model developers follow structured development due to small-size software development projects. Figure 1.1 illustrates the sequential phases of the waterfall process.

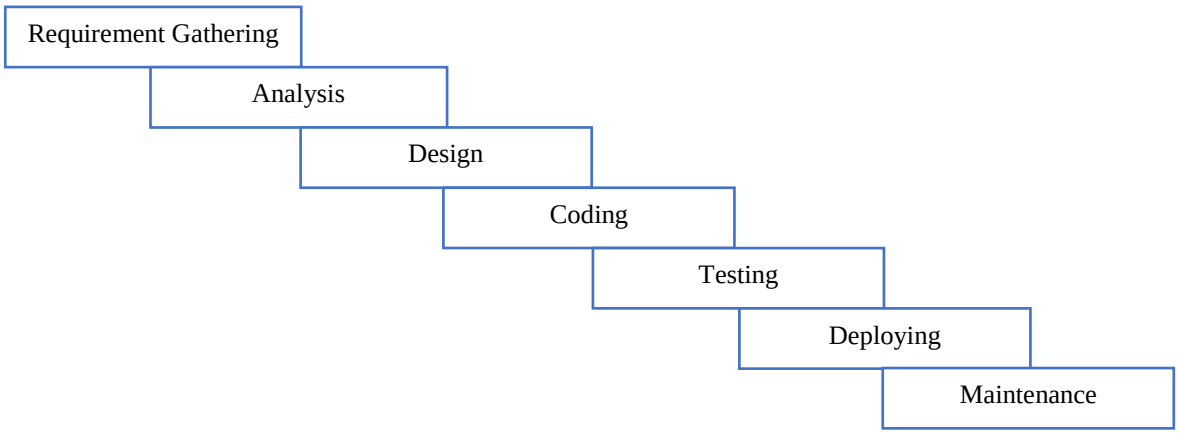


Figure 1.1: Waterfall development methodology

1.1.2 Prototyping

Prototyping is a software development model in which a sample software is designed and presented to the customer in order to get feedback from customer evaluation and modified based on the feedback until the acceptable prototype is reached. Prototyping methodology consists of six successive phases namely, requirements gathering and analysis, quick design, building a prototype, user evaluation, refining the prototype, and implementation. From the product perspective prototype software is not complete software but primarily designed and developed to get customers' requirements amply. Prototyping methodology is best when user requirements are not clear. Prototyping methodology enables controlling software projects under certain planned samples. As a disadvantage, in the prototyping approach the number of prototypes to be developed is not clearly known. As a result, it may take an excessive amount of time to get the

final desired prototype, which in turn causes customer dissatisfaction and may lead to the failure of software projects.

1.1.3 Spiral Model

The spiral model is risk handling driven model which consists of four successive steps in one development phase including planning, risk analysis, coding and testing, and project evaluation. As its name indicated it looks like a spiral with many loops in which each loop starts after the end previous loop as indicated in Figure 1.2. The radius of the spiral at any point represents the cost in terms of time and manpower, in general, the cost of the project and the angular dimension represents the progress made so far in the current phase. In spiral mode, the exact number of loops is not known and can vary from project to project. The number of loops can be determined based on the risk analysis of the project. In spiral mode, the project manager has an important role to develop a product using the model.

In addition to risk handling capability, the spiral model is good for customer participation during software development. Customers can see the product at the early stage of development. In this model, it is possible to change existing requirements and add new functionalities at a later phase of the development. The spiral model is suitable for large and high-risk software projects where requirements are not stable. On the other hand, this model is not suitable for small-sized projects, in which the degree of risk is low. In comparison with the other SDLS models spiral model is much more complex and dependability on risk analysis is high. It is difficult to know the number of phases to be procced and that in turn makes it hard to estimate the end of the project in terms of time, cost and so far.

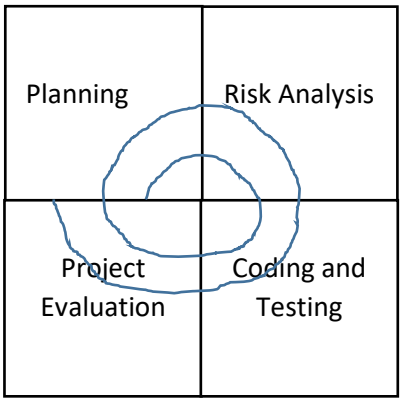


Figure 1.2 Spiral Development Methodology

1.1.4 Incremental Model

In the Incremental Model, requirements are gathered to the maximum level and prioritized according to dependency and customers' needs. In the first stage of the development, a small working system will be implemented with basic features only as an initial version. Other successive versions of the system will be developed at the top of the first initial version until the desired product is realized as indicated on Figure 1.3. Iterative waterfall model is used to develop all the incremental versions. Based on the number of modules developed in a single development period, Incremental Model is divided into two basic categories. The first category is named as Staged Delivery Model, in which only one part of the project will be accomplished at a time and the second one is named as Parallel Development Model, in which different subsystems are developed at the same time. This can reduce development time and the selection of the approach depends on team size, expiry level, and availability of resources.

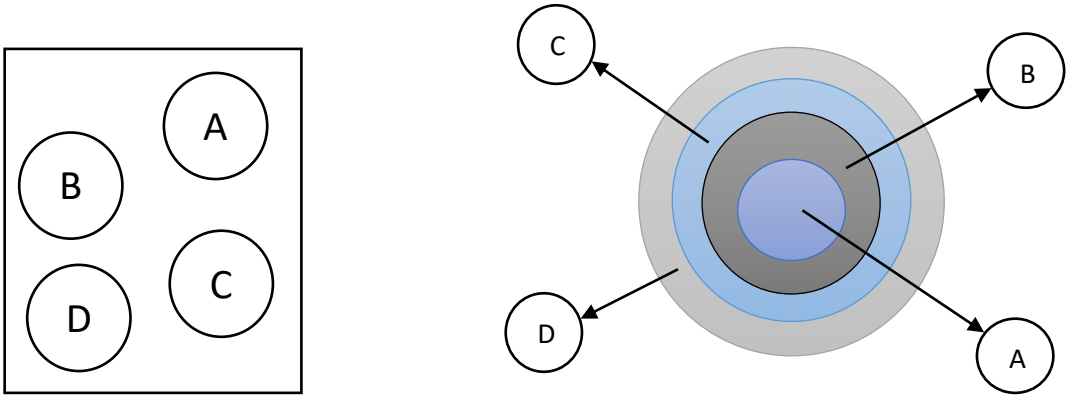


Figure 1.3 Incremental ((a) parallel development, (b) staged delivery) model representations

Incremental Model is best for the projects which have clear requirement specifications and lengthy development schedules. Using Incremental model enable project managers to identify and manage risk and to control over program complexity. On customer side, customers will have better image on over all projects and can get desired product at appropriate time. This approach need well organized project management and development team to accomplish the project work with in the schedule. Due to its iterative work the cost of the project will rise with respect to formal waterfall approach.

1.1.5 Agile Methodology

Agile is an umbrella or family of process with similar philosophies and principles and practiced in different organizations and companies including software companies, IT organizations and other organizations. February 11-13, 2001 was starting point for Agile Software Development Methodology when seventeen people working on different organizations and institutions met to talk, ski, relax and try to find common ground for the betterment of software development and project management. The agile manifesto emerged at that time with four values and twelve principles [15]. Agile methods are iterative and incremental methods, focus on time to market of working software. Figure 1.4 shows a general form of agile iteration lifecycle. Representatives from different software development practices like Extreme Programming, SCRUM, Dynamic Systems Development Method(DSDM), Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development (waterfall) processes attended on this assembly. Issues like customer satisfaction, requirement change entertaining, continuous delivery, cooperation of customer and developers, motivated individuals, a face-to-face conversation of the development team, and working software are among the twelve principles of the agile manifesto. After the Snowbird meeting, the alliance team posted the manifesto document online and invite others to add their names as supporters. As a result the agile movement spread out rapidly in software development industry. Most members of the alliance team, joined and discuss on ways to disseminate agile principles for software development industry.

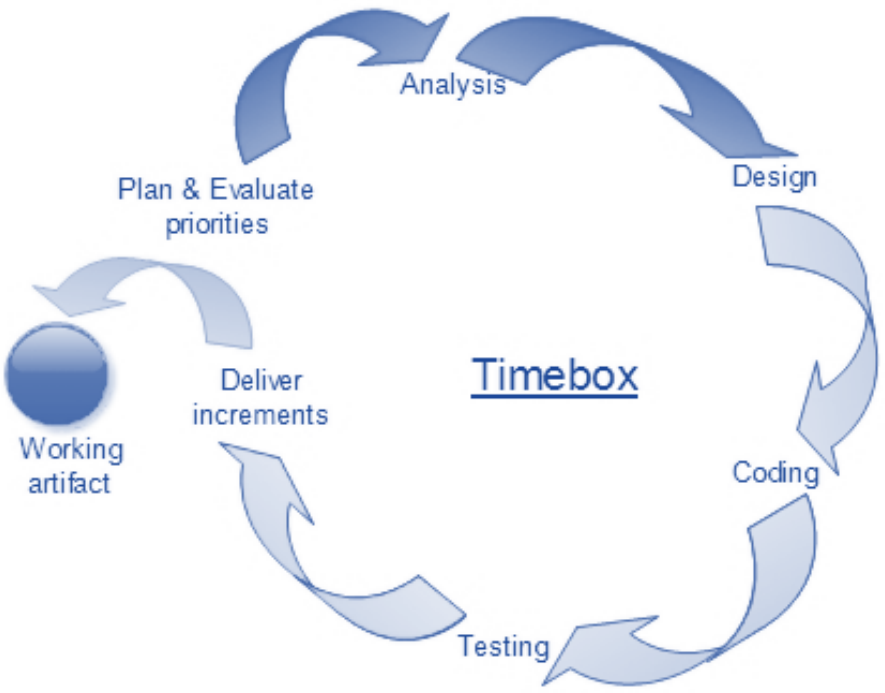


Figure 1.4 Agile iteration lifecycle

Agile methods are mainly based on four values and twelve principles [15]

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

1. Individuals and interactions over processes and tools.
    - Refers to the fact that it is the people involved and how they communicate that typically has the largest bearing on the success (or failure) of a software project.
    - Software development processes, methodologies, tools, etc., can help but they are still not the overriding influence
2. Working software over comprehensive documentation

- At the end of the day it is the software produced by a development project that will be used by a user and not the documentation.
- Documentation should not be a major goal in and of itself.
- Documentation should be a supporting medium for the actual product i.e. the software.

3. Customer collaboration over contract negotiation.
    - Time should be spent on working with customers and in getting them involved in the software development rather than on detailed contract negotiations
    - Challenges: although your direct clients may buy into this philosophy, their legal and financial departments may not
    - Legal and financial departments want something to hit us with, if things go wrong
4. Responding to change over following a plan.
    - Agile software development embraces change rather than saying "it's not in the requirements or the plan, so we can't do it."
    - I.e. Development progresses in response to user feedback, rather than as a reaction to a fixed plan.
    - Note that does not mean that there is no plan and that planning isn't important
    - Planning is very important but the project adapts itself to its environment

The 12 Principles behind Agile manifesto are:

1. Highest priority is to satisfy the customer.
2. Welcome changing requirements, even late in development
3. Deliver working software frequently.
4. Business people and developers must work together daily.
5. Build projects around motivated individuals.

6. Face-to-face communication is best.
7. Working software is the primary measure of progress.
8. Promote sustainable development.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements and design emerge from self-organizing teams.
12. Introspection – teams should regularly review itself and its processes to try and improve.

When we come to agile characteristic, practices and principles, agile plays a very essential role in rapid software development and delivery processes. The first nature of agile is the ability to create and respond to change at any development stage of the project based on its value in the philosophy. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment [10]. Agile Software Development refers to software development methodologies centered on the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile Software Development is that it enables teams to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change. Scrum and Kanban are two of the most widely used Agile Methodologies. [9]

The Agile Software Development Methodology is the leading methodology due to its contribution to rapid application development and enhancement of communication between customers and software developers. In Agile Development Methodology, both developers and customers will have common understanding about the developing software. Agile is flexible for adding, removing, and changing the requirement based on the findings during communication of the two parties. Many large application development projects today follow Agile Software Development Methodology to deliver software with required functionalities and quality within a short period of time. Agile Software Development Methodology is suitable for software projects which require CI and CD. [8, 14, 15].

Agile Development Methodology consists different lightweight development methodologies within it. Among the methodologies, Scrum, Kanban, Extreme Programming (XP), Crystal and Lean Development are the prior ones. Some of the methodologies are discussed in the following sub tops for the sake of clarity.

1.1.5.1 Scrum

Scrum methodology is the most used project management framework among methodologies under an agile umbrella and characterized by cycles or stages of development, known as sprints as shown in figure 1.5. Developers or scrum teams can prioritize, manage, and execute works based on the sprints. The Scrum team have a regular meeting every morning for fifteen (15) minute known as daily scrum. During the meeting scrum team share activities statuses and finding, and plan out the working day. Scrum is good for team motivation in such programmers work vigorously to finish every spring before the deadline. In scrum, there is transparency that enables members to follow the status of the projects on which the team participated or even other projects throughout the organization. Even though the scrum methodology has many supportive features for rapid application development, it is criticized in terms of lack of specialization and developers have a less clear image of general projects rather they focus on sprints only.

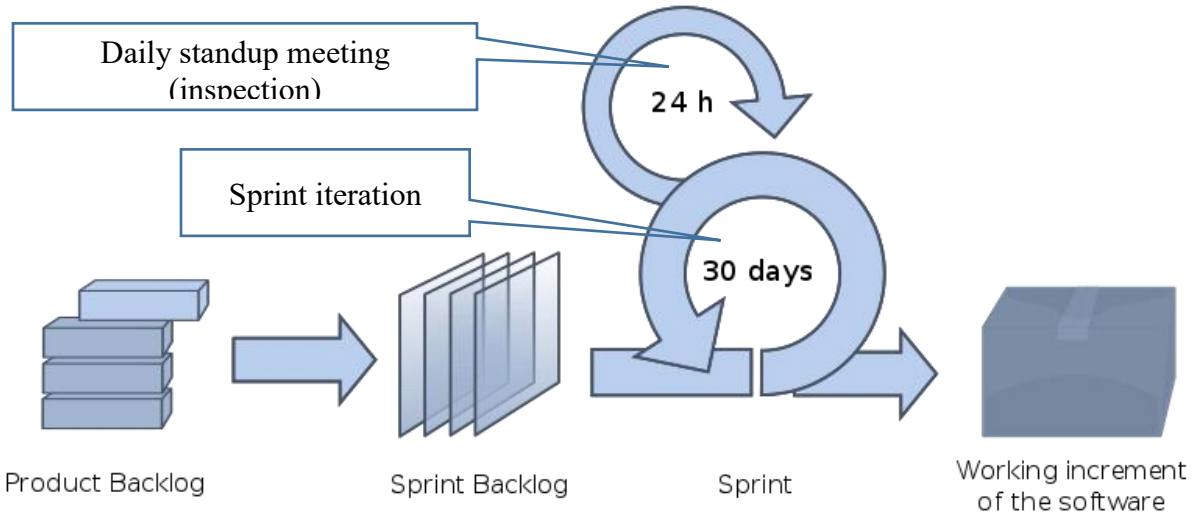


Figure 1.5 Scrum skeleton

1.1.5.2 Kanban

Kanban is among the methodologies under an agile umbrella with a distinct ability to fit an organization's settings. The word Kanban originated from Japanese and the meaning of the word is linked to the concept of "just in time". Kanban focuses on visualizing the entire project on boards to increase project transparency and collaboration between team members. Tasks of the project categorize into "to do", "doing", and "done" statuses. Kanban is based on the ideology of starting from a small working system, continuous improvement, flexibility in task management, and enhanced workflow. In Kanban methodology, project managers can limit the number of tasks in the "doing" list. In a Kanban methodology, developers focused on a given take rather than a deadline. That enables developers to deliver quality work in optimum time. The unlimited amount of time to complete a given task may sometimes lead to time-related problems.

1.1.5.3 Extreme Programming (XP)

Extreme Programming is among the lightweight methodologies under the agile umbrella. Extreme Programming works based on the five basic rules or principles. Planning, managing, designing, coding and testing are the basic rules that extreme programming follows. Under the planning rule writing user stories, preparing the release schedule, diving the project into small iterations, and prioritizing pieces to be developed at the initial place are the basic ones. Preparing a dedicated working environment, arranging standup meetings, and measuring the velocity of the project are some of the management rules. In the designing phase of extreme programming, developers use Class Responsibility Collaborator (CRC) card for the sake of visualization of the work and start from a very small part of the system (system metaphor) and increase iteratively until the desired system is complete. In extreme programming refactoring part of the system is allowed anytime and anywhere. Coding must follow a pre-prepared standard and unit tests should be performed before an integration. Collective ownership should be followed by developers to enable modifying any part of the system by any of the developers. This, in turn, create a trusted and motivated environment for developers, managers and customers.

1.1.5.4 Crystal

The crystal method is a lightweight color-coded agile framework that focuses on individuals and their interactions. Crystal methodology gives the first priority to interactions, people, skills, talents, and communication over processes and tools. The color code is given based on the number of participants in a single team.

No.	Color code	Team Size
1	Crystal clear	Teams of 6 or fewer employees
2	Crystal yellow	Teams between 7 and 20 employees
3	Crystal orange	Teams between 21 and 40 employees
4	Crystal red	Teams between 41 and 80 employees
5	Crystal maroon	Teams between 81 and 200 employees
6	Crystal diamond and Crystal sapphire	For larger projects

Table 1.1 Crystal method color code and team size associations

Even if Crystal has no formal structure, it's still based on a set of seven basic properties that teams should follow in order to maximize its capacity. Frequent delivery, Reflective improvement, Consistent communication, Personal safety, Focus, Easy access to expert users, and Technical tooling are the basic properties of Crystal development methodology. In Crystal methodology, teams should release new features frequently. Reflective workshops should be arranged every few weeks to make discussions on the improvement of development processes and tools. A safe environment for all team members should be prepared in order to get their opinion and suggestions without any fear of rebuke or retaliation. In Crystal methodology, all team members should have a clear and common understanding of the overall project progress. An adequate amount of tools should be available for the development environment as well as the hosting environment.

1.1.5.5 Lean development

In the middle of the 20th century, the lean manufacturing concept emerged out of a production process in the Toyota manufacturing industry in Japan. The main goals of lean manufacturing are reducing defects, cutting out waste, increasing productivity, and encouraging accountability and innovation. Due to its productivity in Toyota manufacturing it spread out to other industries dramatically. Lean software development becomes part of agile development methodology after

the first agile assembly due to the open invitation for contribution by agile alliances. With respect to software development lean software development emphasizes optimizing efficiency and minimizing waste in the software development process. Lean have also seven basic principles which are eliminating waste, building quality products, amplifying learning, delay commitment as long as possible, delivering fast, respecting people, and optimizing the whole processes.

1.1.6 DevOps

Successful software development is not only measured by developing working software. But it also measured by hosting environment infrastructures and management capacity. Almost all high scaled software deployed and hosted in running environment in order to provide desired functionalities. Poor infrastructures and management on software running environment has significant contribution for failure of the developed software. In order to facilitate running environment activities collaboration of developers and operational teams is very indispensable. The idea of develops comes with the cooperating Developers and Operational Team (System Administrator) together in order to realize Continuous Development(CD), Continuous Integration (CI) and Continuous Delivery (CD). The idea of DevOps started at 2007, the then an IT expert Patrick Debois was interested in learning IT from every perspective, he began working on a large data center migration where he was in charge of testing. During this project, he realized that the frustrations experienced in projects such as these are from the constant switching back and forth between the development side and the silo of operations on the other side of the fence. He recognized that a lot of time and effort was wasted navigating the project between these two worlds, but the divide between them seemed too wide to link. DevOps is a newly emerged culture/tool to enhance the communication between software developers and software operators. For software projects that demand continuous development, continuous integration and continuous delivery, developers and operators should work together in some regular and formal manner. As a solution DevOps come up with a mission of supporting the communication between the two entities. On software project with rapid development, issues like version control, operational environment upgrading, security enhancement and performance monitoring are the main areas that need more attention. System administrators or IT operators are the more responsible for such kind of activities [13].

1.1.6.1 DevOps Lifecycle

The DevOps lifecycle is a series of development processes or workflows within an iterative development lifecycle. It follows a continuous approach as indicated in figure [1.6] blow. The lifecycle is symbolized in the form of an infinity loop which indicated the collaborative and iterative approach throughout the application lifecycle, consisting of different tools and technology stacks for each stage. The left part of the figure deals with software development and testing. And in contrast, the right side of the figure represents the deployment and operations cycle.

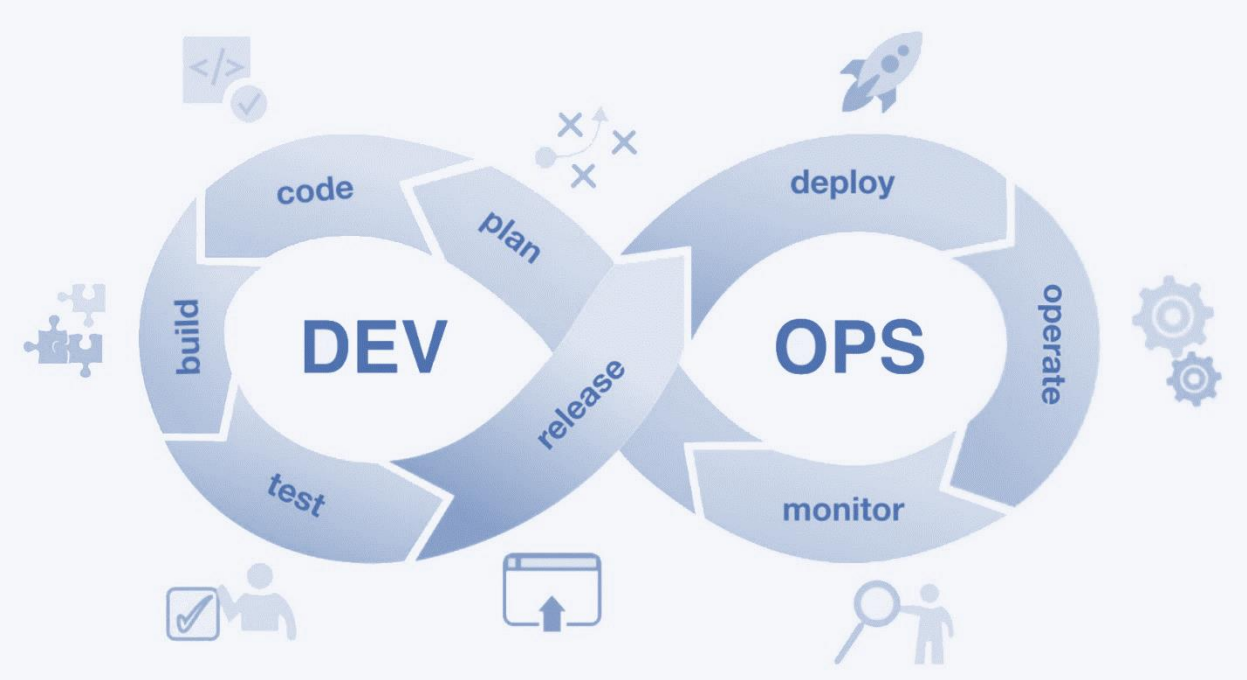


Figure 1.6 DevOps Lifecycle

Plan – Planning is the first step in the DevOps lifecycle and developers determine the customers' needs and gather end-user opinions throughout this stage. In this step, they design a project plan to optimize business impact and produce the intended result.

Code – Coding comes after planning and the business logic changes to algorisms and then to coding. To simplify the design process, the developer should use different process automation tools and frameworks.

Build – After coding activities are accomplished the program should be assembled and ready for running hosting. All necessary resources and dependencies should be compiled and included in the build for testing.

Test –To assure software integrity, the product is first delivered to the test platform to execute various sorts of screening such as user acceptability testing, safety testing, integration checking, speed testing, and so on, automations and tools can be used for testing purposes alternatively.

Release – Release is simply sending a tested software to a running environment. At this point, the build is prepared to be deployed in the operational environment. The DevOps Team prepares updates or sends several versions to production when the build satisfies all checks based on the organizational demands.

Deploy – Hosting the built and tested software on a running environment to make it available for end users. DevOps Team is responsible for preparing and managing all necessary infrastructures.

Operate – After deploying software user and system administrator should operate it in order to get desired functionalities. The operation needs the support of the DevOps team and end-user training in order to make the operational environment smooth and enjoyable.

Monitor - The DevOps Team's heavy burden is observed at this level expressed by collecting user experience, application performance monitoring, security issue follow-up, and performing other running environment activities.

1.1.7 Reasons behind selection of Agile and DevOps

Among the discussed methodologies, Agile and DevOps are selected due to their ability for supporting rapid software development such as continuous development (CD), continuous integration (CI) and continuous delivery (CD). The two methodologies aim to achieve rapid development and delivery of quality software but have their own role and practices, working independently. In this research an attempt is made on creating an integrated framework that brings the two technologies together with minor modification on DevOps team role and structure.

Even though the two mentioned approaches are the most appropriate approaches for software projects that need Rapid Software Development, the two have their own drawback when used independently. That Agile, specifically Scrum focuses mainly on project management whereas DevOps focuses mainly on communications between production and hosing environments. Project management without product management will not be successful and vice versa. From the approaches under agile umbrella, scrum development methodology is selected for this research work due to its capability of limiting development time for a given single sprint.

The researcher was motivated to conduct the research due to challenges faced in the working environment where there is more need for automation but with the limited amount of experts, resources, processes, and awareness problems on software development from the own practice and observation while the researcher working at Hawassa University as a system analyst for more than a decade. Here at Hawassa University, Information and Communications Technology (ICT) department, there were many software projects running by the Application Development Team as well as developers from faculty of informatics. The projects are developed for internal use, community services, and undergraduate degree fulfillment purposes. In the case of the Application Development Team, participating in different projects increase from time to time, and become difficult to handle all the projects in a desired manner. Later on, the developers decided to use organization-tailored methodology by combining Agile-Scrum and DevOps together. Even if the two methodologies are used in integrated form there is a limitation in preparing the standard of usage for the tailored methodology. Also, it could not be denied that from the experience of the researcher as well as research works conducted on the national level [16] it is difficult to get well-organized development teams that follow formal software development methodologies at the national level. However, there are some organizations that practice standard methodologies like Scrum, and Kanban, and most of these are located in the capital Addis Ababa.

1.2 Statement of the Problem

Flexible, dynamic and rapid software development approaches become more encouraging for early delivery and quality software. Among these approaches Agile and DevOps are widely used approaches for their capability of supporting requirement changes during development, continuous integration and continuous delivery. But the two approaches used for different

purposes in software development processes. Agile focuses on project management while DevOps focuses on product management. If the development follows the approach that allows requirement changes while developing, there should be a mechanism to accept the change, integrate the change and deliver a new product for the customer within a short period of time. Even if the two approaches have their own significant impact in software development industry, integrating the two can bring improvements in the software development and delivery processes. Furthermore, in most cases the development processes are not used as they are prescribed in the book. Instead most organizations apply the methodologies based on their applicability to the context. Therefore, this research tries to look into the local practice, challenges and context in proposing the integration of agile and DevOps. The primary goal of this research is to get the advantage of both approaches by proposing an integrated framework that incorporates the two approaches or part of the two approaches together. Generally, this research is going to answer the following questions:

RQ1: How to enhance the communication between customers, developers and operators?
RQ2: To what extent the concept of separation of concern become clear between developers and IT operators?
RQ3: How to improve the software development process by integrating Agile and DevOps?

1.3 Objectives

1.3.1 General objective

The general objective of the research is to propose an integrated conceptual framework of Agile and DevOps in order to get the advantage of the two approaches at single spot and eliminating the limitations of the two approaches when used independently.

1.3.2 Specific objectives

In order to attain the main objective the following are the specific objectives:

- Reviewing related literature to understand about the two methodologies in depth and the research design
- Identifying agile practices that can work closer to DevOps
- Identifying DevOps practices that can work in line with Agile practices

- Working in closer collaboration with development teams to examine the effect of integrating the different practices in projects
- Developing conceptual integrated framework of Agile and DevOps in suitable manner for end-to end development process important and knowledge transfer.
- Evaluating the framework with case projects
- Prepare guideline for the developed framework

1.4 Scope of the study

The scope to the study is limited to proposing conceptual framework by combining Agile and DevOps in a way suitable for adaption and evaluation by developers, researchers and student at national level. This include identifying useful principles and practices from both approach, framing new conceptual framework , evaluating the framework by conducting experiment with minimum requirements in terms of team size and professional skills, identifying common assumptions to be included in the integrated framework, preparing guideline for developer, researchers and students, and illustrating the framework by using different diagrams and charts. The framework here to be developed is conceptual framework and have the following characteristics:-

- Tool independent
- Easily adoptable by developer and researcher
- Open for further modification for the betterment of development process
- Used for end-to-end development process improvement
- Have room for knowledge transfer during development process
- Supportive for development environment management

1.5 Significance of the study

The researcher was interested to conduct this thesis in the area due to challenges he faced during his participation in software development projects at *Hawassa University Application Development Team* where he worked as a senior system analyst. The challenge he faced happened due to the lack of formal development methodology followed by the team. At the national level also, many software development companies follow ad hoc development methodologies [16] and that is the main factor for the failure of in-house software development projects. As a result, the ultimate significance of this research work is improving the software

development process at the national level by drafting an easily understandable and improvable framework. In addition to this, the following benefits will be gained by conducting this research:-

- The proposed framework can be used in the teaching-learning process to add value to the educational sector.
- The research work is will contribute visible value to Science and technology.
- Since the framework has room for knowledge transfer, it enables developers to share what they know.

1.6 Thesis Organization

This research paper is organized in a way suitable to demonstrate the flow of the research work by categorizing the paper into five (5) successive chapters. *Chapter One* is the introduction part and addresses the background of software development and software development processes. In addition statement of the problem, the research questionnaire, and the objective of the research are included in this chapter. The conceptual and related literature review included in *Chapter Two* and research conducted on software development process improvement were reviewed in an adequate manner. In Chapter Three, materials, methodology, and procedures used for experimental work are discussed wildly. The findings of the experiment are discussed in *Chapter Four*. In this chapter, discussions are made on the findings of the research and the conceptual framework illustrated by using different diagrams and charts and also the guideline for the usage of the integrated framework included in this chapter. The conclusion of the research work and recommendations for future work on the area is indicated in *Chapter Five*.

Chapter Two

Literature Review

2.1 Conceptual Literature Review

Both Agile and DevOps are latest approaches and used for rapid software development, continuous integration and continuous delivery of quality software. DevOps is a set of practices that combines software development (Dev) and information-technology operations (Ops) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. [13]

Agile development used to be front and center in the conversation about software development. Now, DevOps has taken over the conversation. How do agile and DevOps relate? Both ideas began as ways to improve different aspects of software development. Agile embraced the changing nature of requirements and prioritized working software over rigid processes. DevOps collapsed the development and operations silos to improve both development and production operations. Each share some fundamental ideas, but each target different stakeholders and set different business goals. [10]

2.1.1 Similarity and Difference between the Two Approaches

Mohammad discussed similarities and difference between the two approaches [10]. In this discussion he tried to show two wrong assumptions that many people believe in about the similarities and difference between the two approaches. The first assumption is that many people think as DevOps and Agile are similar and one can replace the other. The second assumption is that DevOps and Agile are completely different and they are adversary. These two assumptions raise due to the lack of detailed understanding about the two approaches.

Similarity Between Agile and DevOps

- Both help teams to work faster and more efficiently, where quality work is delivered.
- Both worked aiming to satisfy customers by allowing changes during software development and even latter after final delivery.
- Both approaches are based on team formation, which creates good team spirit to produce quality software in short period of time.
- Both support rapid software development approach.

Difference Between Agile and DevOps

- DevOps is a software development method which focuses on communication, integration, and collaboration among IT professionals namely developers and operators.
- Agile software development method emphasis on iterative, incremental, and evolutionary development.
- Agile refers to an iterative approach which focuses on collaboration, customer feedback, and small, rapid releases.
- DevOps consider as a practice of bringing development and operations teams together.
- Agile method is to give priority to the working system over complete documentation. It is ideal when you're flexible and responsive.
- In the DevOps, process documentation is foremost because it will send the software to the operational team for deployment.
- Agile methodology is a project management strategy, whereas DevOps works on pipeline optimization.
- Agile concentrates on flexibility and the advancement of functions, whereas DevOps emphasizes on continuous integration and installation of software developed [12].
- Agile is commonly aligned with systems like Scrum, Disciplined Agile Delivery (DAD)**,** Large Scale Scrum(LeSS) and Scaled Agile Framework (SAFe), but DevOps does not necessarily apply to specific frameworks.
- Agile emphasizes on operations and DevOps on operational efficiency and automation.

2.1.2 Advantage and Disadvantage of Using the Two Approaches Independently

Disadvantage of Using the Two Approaches Independently.

Consider the factory which have two processing units namely A and B, the first process starts from unit A and then the second processing unit B gets the input from Unit A. The end product will be available after the second processing unit B. If processing unit A processes faster than processing unit B, there will be excess input available for processing unit B. This will lead wastage of resources used for the production. If the processing unit B processes faster than processing unit A, then possessing unit B will have ideal time and this will cause wastage of energy and time at processing unit B.

Based on the above example, Agile Software Development Method represented by processing unit A and DevOps represented by processing unit B. If Agile Development Methodology used without DevOps there will be many updates on developing software. Due to the Agile capability to entertain change, even latter the software delivered, based on customer requests, this update may not arrive at running environment on time. This will reduce customer satisfaction and also there will be probability of jumping or missing of some important updates that take time and energy of developers. On the other hand, if DevOps is used in other software development approaches like waterfall, it is wasting manpower by building operational team for low frequent releases of software updates. The other disadvantage of using Agile without DevOps is that in agile methodology, separation of responsibility is not clearly defined between developers and operators and that leads developers to expend time and energy on operators' work.

Benefit of Using the Two Approaches Collaboratively.

The benefit of using both Agile and DevOps collaboratively is like a factory with the two processing units mentioned above which work in harmony in such, processing unit A processing semi-products for processing unit B and processing unit B produces final product. In general collaborating the two approaches facilitates the development, integration and delivery of software. The integration of DevOps and Agile helps in:

- Streamlines the release process and improves product offerings
- Allows for better collaboration
- More value and fewer risks in each release
- Fewer bugs and faster fixes
- Increased visibility

2.1.3 Software development trend at the national level

At the national level, demand for business automation increases from time to time. Nowadays, governmental and private organizations at the national level show great interest in the automation of their business activities. In contrast, the supply of software is not adequate to fulfill the demand in the area and is mainly characterized by failure. The main cause for the failure of a software project is the gap between demand and supply [16]. As Biru indicated in his research work, software development companies at the national level are not mature enough and are characterized by a failure history due to the following reasons:-

- Inexperienced and small
- Follow ad hoc processes and methods
- Lack of competence in project management and soft-skills
- Affected by very high staff turnovers
- Inadequate educational and training support infrastructure
- Absence of home-grown or contextualized methods
- Absence of national standards or guidelines

On the demand side, the software development industry is characterized by the:-

- Outsourcing very large projects
- Involve the development of multiple applications systems for specific organizations
- Involve business process redesign as a front-end process to the software development
- Operate in unstable organizational environments

In overcame the gap, Biru proposed a context-sensitive methodical approach that related to tailoring complementary aspects of suitable software development approaches (methods and processes) with the dual purpose of enriching the approaches themselves and developing a context-sensitive methodical approach that would contribute to the improvement of the software development situation in Ethiopia.

From my professional experience in the area, governmental and private organizations are in confusion between outsourcing and in-house development due to challenges faced related to software development. In the case of outsourcing, the cost of software becomes higher from time to time, and support and maintenance cannot be accessible on time. Support and maintenance charges additional cost which is nearly all equal to the development cost. Outsourcing big software also creates dependence on others in terms of technology; which in turn creates a long-term negative impact on sovereignty. On the other hand, in-house development is characterized by a full of failure history, and trust in in-house development decreases from time to time. Even though in-house development is characterized by many problems it is better to strengthen it by proposing solutions for the problems and by building the skill of professionals in the area. From the solution, developing a software development process management framework in the national context, which can be easily adoptable and understandable by developers, is the prior one. This will benefit the country in the long term in terms of independence and sovereignty.

2.2 Related Literature Review

Some of the papers reviewed in this research work are not directly related to combining Agile and DevOps together due to the insufficient work on the area. Most research works in this area focus on challenges of combining Agile and DevOps, Moving from Agile to DevOps and related issues. The papers selected and reviewed for this research work are due to their being closest in some extent. Hemon et al [4] conducted case study research under title "From Agile to DevOps: Smart Skills and Collaborations" to address two research questions. The first question is "How do individuals collaborate in teams and what skills do they need?" and the second is "Do patterns of skills and collaboration vary across teams with different degrees of automation?" The researchers followed the case study method combining interviews, observations and documentation to explore and analyze a new phenomenon. They conduct the research on 12 teams who develop applications in the same context of a large European IT services firm (more than 100 000 employees) with 15000 of IT Staff. On the study they reason out as they move towards DevOps, automation will affect the human skills that may be required and collaboration patterns (how and with whom job roles will collaborate). They divided software development process automation into three parts to conduct the research work.

The first level is Agile titled as "Automation Level 1"

Compared to the traditional V-model, agile can be considered as automation level 1 in the transition to DevOps, because it facilitates more frequent releases as development becomes more iterative. However, DevOps is not realized because Developers and Operators continue to work in silos with limited sharing, nor automation of releases.

The second level is Continuous Integration titled as "Automation level 2"

The Operators function needs to be aligned with the Developers function, and both begin to perform various tests (unit and non-regression tests), which are synchronized with code development, and also automated as far as possible.

The third level is Continuous Delivery Stage leveled as "Automation level 3"

Integration tests with other components, end-to-end tests, performance tests, user acceptance tests are then co-designed, performed and preferably automated by Operators in conjunction with the Developers function. The greater the level of automation, the more it requires sharing and

mutual discussion about measurement, so that the right indicators can be contextually designed to monitor the software development activity. They noticed that the level of automation has effect on improving software development process in addition to skill of developers. They also mention that software development process maturity progression will also depend on level of automation.

This research mainly focused on progression towards to DevOps from Agile but not combining the two approaches in order to get the advantages of them. The limitation to be considered here is that moving from Agile to DevOps by omitting the useful practices in Agile Software Development process have also negative impact on success of project and product management. The study also mention that the better collaborations between professionals reflecting better understandings among members about the project as well as working environment. But by missing agile practices like Scrum Ceremonies, the team will not collaborate to the desired extent. So that in order to maximize collaboration among skills, agile development practices are more important. Finally, integrating Agile and DevOps is essential for success of project and product management rather than moving from Agile to DevOps by replacing agile practices by automations.

Raj and Sinha, write a paper under a title “Project Management in Era of Agile and DevOps Methodologies” [1] that try to show the impact of Agile and DevOps on project management practices and team structure in Software development projects. They mention that project management practices determine the success of the project and both methodologies have the ability to provide more flexibility and faster-delivering projects and organizations are adopting the practices for their perceived benefits. Project management practices to an extent depend upon the software development methodology used and hence would be impacted. They propose a Conceptual Framework for using Agile and DevOps methodologies in software development which includes two main parts namely enhance project management practices including Scope management, Quality Management, and estimation (time and cost) and the other is impacted team structure which includes shared responsibilities (Role), automation tools and communication feedback. They discussed that there are many research works on agile methodology but limited works on DevOps practices. In their paper, the researchers try to show the impact of using the two methods in project management but they did not propose how to use

the two methods collaboratively. They mention that "a combined effect of Agile & DevOps development methodologies impact and relation to project management are not sufficiently represented" [1].

Raj and Sinha also discussed the impact of Agile and DevOps methodologies on Project Management Practices and Team structure. They aimed the research will provide new insights for the future researchers to understand the Agile and DevOps and how it significantly impacts on software development industry. Their research has both theoretical as well as practical implications. The theoretical implications of the research from the detailed review of existing study indicates that, in future the research can be extended by hypothesized and tested empirically by collecting and analyzing the primary data through quantitative research. The practical implications of this research indicate that, software sectors can implement Agile and DevOps methodologies for their software development process which can enhances their project management practices and also organizes team structure. This framework was designed and conceptual but not implemented for project management usage. The researchers did not describe each component of the framework in detail and also recommended for further research work. Their conceptual framework is shown in figure 2.1 as follows.

Figure 2.1: The conceptual framework for the impact of Agile, DevOps methodologies on Project Management Practices and Team structure (Ref: Raj and Sinha [1])

This paper mainly discussed about the two approaches in detail and try to show their impact on project management and team structure. But the paper did not formulate or show the way how to use the two approaches collaboratively. Since the primary goal of this thesis is proposing collaborate framework for project and product management, these reviewed papers have encouraging ideas.

Fernando conducted a qualitative study under the title “Exploring the Benefits of Combining DevOps and Agile” [17]. They conduct the research by including twelve (12) case studies from different international software companies and gathered secondary data from the companies’ news presses and blogs. They use thematic analysis in identifying the benefits of the combined adoption of agile and DevOps. They mention that, even if Agile and DevOps are different concepts, they can be combined and offer relevant benefits to software development companies. From the research work, the researcher identify the existence of twelve benefits, highlighting the automation of processes, improved communication between teams, reduced development time, and shorter software delivery cycles. The findings of the study made it evident that the two models are not incompatible, but when combined they can maximize their impacts on software development processes. As a limitation, the researchers indicate that the research was conducted by including multiple case studies from secondary sources, which does not allow them to dig out the knowledge on the themes with the use of interviews that may evidence the application of both paradigms. In addition to the limitation they indicate in their research, there are some gaps in their work. The first one is that they did not indicate the way to combine the two approaches and secondly, the research work did not describe the type of agile methodology appropriate to be used with DevOps and the issues about the team structure are not raised in their research work.

Samarawickrma [18] conduct a quantitative study under the tile “Continuous Scrum: A Framework to Enhance Scrum with DevOps” which focuses on building a software development methodology by bringing Scrum and DevOps together where Scrum is used to cover the managerial aspect of the software development lifecycle and DevOps is used to support the rapid delivery requirement. He selected Scrum from the agile family due to it being the best agile practice. As he indicates in this paper most software developments fail due to less control over the total flow and in Scrum, the loopholes have been closed and the adaptation proved that Scrum is the best framework in Agile. On the other hand, the demand raid rapid delivery is

booming due to the computation in the business domain having accelerated delivery as the key to success. Companies believe that the solution to this issue is DevOps. Most companies adapt DevOps without knowing that DevOps is a cultural movement. He proposed a new framework with the combination of Scrum, continuous integration, and continuous delivery by modifying Scrum ceremonies, and rules. The modifications are done to support continuous nature like development environment setup, planning, execute, inspect and adapt, and operation and support. Evaluation is done on data from a different source by using tools capable to achieve the target framework. Matrixes like User stories planned versus user stories delivered, User stories delivered versus user stories accepted, Defect-removal efficiency (DRE), Deployment (or Change) Frequency, and Velocity are used as measurement parameters. The tools used for data collection extract data from a different points and show them in a meaningful manner and have the following capabilities.

- Extract the data from the sprint management tool
- Extract data from the local repository and central repository
- Extract data on local build
- Extract data on the main line builds
- Extract data on test case execution
- Extract data from test, staging, and production environments
- Collect data from code-quality software

Task and respective tool indicated in table [2.1]

No	Task	Tool
1	Scrum management tool	JIRA
2	Source Management	GIT
3	Source code quality management	SonarQube
4	Unit testing	JUnit
5	Test automation	Selenium
6	Build tool	Maven
7	Automation tool	Jenkins
8	Log analyzer	The tool itself read the logs

		from a file path
9	Capture Help Desk Data	Web UI to insert the data

Table 2.1 Task and respective tool used for data collection by Samarawickrma.

[Samarawickrma] proves that the new framework is better than the traditional one by comparing the traditional scrum sprint and sprint from the proposed framework. He mentioned in the paper that frameworks should contain guidelines for developers. But he did not include the guideline for development. The research work also depends on tools in terms of evaluation data and that goes in contrary to the agile value of "choosing communication over tools". Finally, the nature of the proposed framework is not included in the paper and it is difficult to adopt the framework easily.

The reviewed literature here proves that combining Agile Methodology and DevOps has a significant impact on improving the software development process by conducting quantitative and case study researches. However, the reviewed research lack to frame the combination of the two methodologies in a way that can be understandable and adoptable by developers, researchers, and students with minimum effort. Additionally that the studies lack room in their framework for knowledge transfer and development environment management for process enhancement. Based on these, the researcher indented to conduct experimental research on the integration of Agile Methodology and DevOps in a way that is suitable for an end-to-end development process improvement by extending the role of the DevOps team to the Development environment and by enabling knowledge transfer platform in order to capacitate to the skill of developer as well as DevOps teams.

Chapter Three

Research Methodology

3.1 Materials and Methods

The method of the research is designed in a way to address the research problems and answer the research questions formulated. Accordingly in this thesis, experimental research design is followed to answer the questions. The experimental approach might be appropriate when it is impossible to take up a longitudinal study due to time limitations and project availability. Furthermore, the experimental approach can result in quick feedback on the design by considering a control group and experimental group. On the way of conducting the research experiment the researcher have passed some serious of steps. The experiment was done based on the trades of Hawassa university Application development team where the researcher currently working on. Hawassa University Application Development Team engaged in many large internal and external software development projects for the last ten years. At the beginning of the project work the team did not follow any formal development methodology for the development as well as integration activities. Later on, due to the increasing number of projects, the team decided to adopt agile development methodology for development purposes without specifying methodologies under the agile umbrella. After conducting successive meetings the team choose Scrum Methodology by considering the behavior of the projects run by the team. The standup meeting is held every Friday per week for reporting activities of the week and planning activities for next week. The researcher act as a scrum master in some project and a full-stack developer in many projects run by the team. Using Scrum Methodology increase the productivity of the team in terms of quality and efficiency. Although the adoption of Scrum makes the team happy, there is still limitation with respect to Continuous Integration and Continuous Delivery. As the result, the team dedicates some team members to integration and delivery work per project. The dedicated team members integrate the developed and tested part of the system and release it for customer evaluation and use. Later on, the integration work was replaced by GitLab and the productivity increases dramatically. This thesis experiment was conducted by following the trends of the Hawassa University Application Development team.

In the experiment conducting process, the first step is selecting experiment subjects, to do that the researcher selected students from Department of Electrical and Computer Engineering those joined Hawassa University Application Development Team (where the researcher is currently working at) for Practical Attachment. There were seven students joining the Hawassa University application development team for three (3) months practical attachment. The students given the necessary trainings with regard to the software development for both technical, project management and soft skills. The second step is forming two teams, with three (3) team members, one work on classical Agile Development and the other work on integrated framework of Agile and DevOps. The third activity in the experimental process is identifying project areas. The researcher let the two teams to identify projects and the assignment was given for one week time length. This is done to help the team members to develop problem identification skill. At the end of the week, the two team members discussed with the researcher and two projects were selected. These projects are "Online Student Advisory System" for the control group and "Ekub Management System" for the experimental group. Both projects were developed in web based version. The backlog of the two projects described under sub title functional and non-functional requirements below. The two projects have almost the same scope and complexity. The project which used as control group follow known scrum practices and the measurement done on number of requests per week and number integration and deliver per specific period of time. For experimental group follows the scrum practices and measurements; but the difference is that for an experimental group, DevOps practices are applied with DevOps team role modification.

3.2 Automation of Development, Integration and Delivery Processes

Nowadays automations are used in order to facilitate the process of development, integration and delivery. Using automation increases quality of the software, helps to keep coding standards, enhance security and to optimize the software.

Automation will reduce dependency on experts for some extent and used to perform routine tasks in the development process. Automation is used in two main software development processes namely, Development and Integration and Delivery processes.

Most software development frameworks have capability of generating model classes and basic CRUD functionality from relational database as well as generation database from model classes. But they can't meet developer's requirement adequately just scaffold basic structure of the

developing software. In this research we use automation for development but not for integration and delivery purpose because of the skill gap of using the automation tools and the scope of developing software.

Training for students on GitLab usage is time consuming, and as a result the researcher with one of the trainee decided to handle integration and delivery task manually. However, due attention is given to handle subjectivity and biasedness. This is not a problem, expending much more time and cost on automating for projects with small complexity and scope is not feasible.

3.3 Functional and Non-functional Requirements of the Projects

Online Student Advisory System (Control Group Project)

Sample functional requirements for Student Advisory System are listed below.

- Creating Organizational Structure
    i. Creating department
    ii. Creating programs and admission types
    iii. Creating batch
- Student Registration
    i. Registering student basic information under specific batch
    ii. Creating account for student
- Instructors Registration
    i. Registering Instructor basic information
    ii. Creating account for instructor
    iii. Assigning role for instructor (as Instructor and Department Head)
- Projects Management
    i. Creating project under certain batch
    ii. Assigning students for project
    iii. Assigning instructors for project as advisor and co-advisor
- Advisor Student Communication
    i. Both group member students and assigned instructors share ideas in form of test message and files.

Ekub Management System (Experimental Group Project)

- User Control Module
    i. User registration
    ii. Role management
- Ekub Management Module
- Member Management Module
- Lottery Management Module
- Payment Management Module

Non-functional requirements for both projects

- Security Management
- Layout and look and fill management
- Backup and restore
- Localization
- Coding and Document Standard Preparation
- Integration
- Deployment
- Hosting

For experimental group project non-functional requirements were managed by DevOps Team and functional requirements are managed using Agile specifically, Scrum development methodology. In this work the researcher have participated in DevOps team in order to cover some professional skill gap with in the team members. The backlog of the experimental group project and the control group project indicated Figures [3.1] and [3.2] respectively.

3.4 Product backlog of the two projects

Table 3.1: Experimental group project backlog (EKUB Management System backlog)

No.	Log	Module	Priority
1	As a gust I want to register to the system	Profile Management	2
2	As a registered member I went to login to the system	Profile Management	4
3	As a registered member I want to manage my profile/account	Profile Management	5
4	As a registered member I want to Search for "Ekub" and join "Ekub"	Ekub Member Management	7
5	As a registered member I want to create my own Ekub	Ekub Member Management	6
6	As a Ekub owner I want to approve joined members	Ekub Member Management	8
7	As a Ekub owner I want to reject joined members	Ekub Member Management	9
8	As a Ekub owner I want manage lottery	Lottery Management	10
9	As a Ekub owner I want collect payments	Lottery Management	11
10	As a Ekub owner I want pay winners	Lottery Management	12
11	As a system administrator I want to preform system settings	System Setting	1
12	As a system administrator I want to activate user account	User Management	3
13	As a system administrator I want to block user account	User Management	14
14	As a system administrator I want to view created Ekubs	Ekub Management	13
15	As a system administrator I want to limit number of Ekubs created by one user	Ekub Management	15

Table 3.2: Control group project backlog (Online Student Advisory System backlog)

No.	Backlog	Module	Priority
1	As a System Administrator I want to create, view, edit, and delete organization department	System configuration	3
2	As a System Administrator I want to create, view, edit, and delete organization Admission types and programs	System configuration	4
3	As a System Administrator I want to create, view, edit, and delete organization Student Batches	System configuration	5
4	As a System User I want to create an account on the system and mange my profile	User profile Management	1
5	As an Instructor I want to access assigned projects.	Project management	8
6	As an Instructor I want to send message and files to student(s) assigned to the project	Project management	9
7	As an Instructor I want to get message and files from student(s) assigned to the project	Project management	13
8	As a student I want create and manage my account	User profile Management	10
9	As a student I want to access messages and files from advisor	Project management	11
10	As a student I want to send messages and files to my advisor(s)	Project management	12
11	As a system administrator I want activate and block user account	User Management	2
12	As a department head I went to assign advisor and co advisor	Advisor Management	6
13	As a department Head I want cancel advisors assignment	Advisor Management	7

3.5 Trainings for team members of both projects

The researcher provided trainings for team members in order to have common ground on software development. The trainings are given just in introductory level considering the time and resource constraints that hindered to give detailed trainings on all mentioned aspects of software development. The followings are list of trainings given for students before the experimental work starts.

- Basics of Software Development
- Database Development
- C#
- Web Development
  - HTML
  - CSS
  - JavaScript (jQuery)
- Microsoft MVC 5
- Deployment and Hosting

In addition to end user training integrated framework also recommend training for low level and middle level experts in order to sharpen them on development activities. The type of training given for developers and other experts will be based on selected technology in which the project intended to be developed. The students have no such experience on developing projects in the same size and complexity as projects selected for an experimental purpose. All the team members participated in experimental work have a comparable experience on software development. The researcher selected average skilled (medium) students for experimental work intentionally in order to meet the goal of experiment by proving that developing quality software with minimum amount of time and with less skilled developers at student level using Agile and DevOps in the integrated manner. After forming the teams, trainings given for two consecutive weeks to all team members on areas related to software development in order to have common ground on software development and to minimize skill gap of the students. This is done mainly based on the intention to include knowledge transfer platform within the integrated conceptual framework.

3.6 Measurements

Measurable parameters are identified and the experiment is conducted based on the parameters. These parameters include the number of changes made to developing software, the Number of Deliveries, Responding to changes, and the quality of developed software in terms of reliability and clarity of written codes and procedures. In this thesis work, requirement changes are classified as major changes and minor changes based on the impact of the change on developing software and the size of the changes. Major changes are changes that have a big impact on developing software in terms of time and manpower to complete the changed part. Major changes may include adding, removing, or modifying some parts of developing software. Most of the time major changes are induced due to scope changes and design changes. Minor changes are changes that have a minor impact on developing software, which includes adding or removing database columns, fixing the datatype of database columns, layout change, and navigation change and indicated by time to complete the change and manpower used to work on the change.

When the scope of the project changes based on the request of the customer, developing software is enforced to be changed in terms of volume, and that in turn may lead to the design change. Scope change is considered to be a major change in this research work. Sometimes customers request business process changes on some parts of developing software before project work is completed. In this situation developer imposed to change the design of developing software based on the request of the customers. On the other hand, developers decide to change the design of developing software for a number of reasons. Among the reasons performance of the application, security issues, ease of use, and look and feel are prior ones that lead to a design change in developing software. Some bug fixing requires big changes in developing software and others can be done in a running environment easily. After the first delivery of the software bug fixing task will be performed continuously based on the feedback from the end users. Such practice is common especially for beginner software developer companies and less experienced companies which may characterize most of the development organizations in Ethiopia.

Chapter Four

Results and Discussion

4.1 Experimentation and Result

The measurement is performed for both control group and experimental group projects independently with respect to the changes made to the projects and number of deliveries committed. For the experimental group project the total number of identified sprints at the requirement gathering session is 15 and the total number of identified springs for the control project is 13 as indicated table [3.1] and [3.2] respectively. In the first iteration of the two projects, the two groups started to work on the User Management subsystem which consists of user registration and authentication. This subsystem didn't consider authorization at the first requirement specification session but later on the researcher (as the customer) ask to add authentication in the subsystem. Developers accomplished the requested change in different time duration with different quality of work. All the other major changes indicated in Figures [4.1] and [4.2] are done in the same manner. The two teams and the researcher have a weekly scrum meeting every Friday morning and the two teams provide reports of the week. In the report, they include the accomplished number of sprints, the number of changes made, product delivery if any, and challenges they have faced during the project work. After the report provision, a discussion has been made on the direction of next week's project activities. And technical challenges they face were solved after the meeting and new deliveries were accepted and integrated for the next iteration of the development.

In terms of changes, for the control group project, four (4) major changes and seven (7) minor changes are recorded. Two (2) of the major changes are caused due to scope change and the other two (2) are based on developers' decisions on design change. The recorded number of changes for the experimental group project is twelve (12). Among the changes made, seven (7) of the changes are major and the remaining five (5) are minor changes. Five (5) of the major changes are recorded due to requirement change and the other two (2) are recorded based on developers' decisions on design change. The minor changes made on both projects are caused by database column addition, database column removal, database column data type fixation, navigation and layout change, and optimizations of some codes and procedures. Furthermore, the

issues related to the quality of both projects are observed and evaluated by comparing the works of the projects with respect to reliability, workflow neatness, and readability of codes and used procedures.

Table 4.1: List of changes recorded for experimental group project (Changes Made on Ekub Management System)

No	Change	Change Type	Date
1	Role based authentication	Major	Oct 12, 2021
2	Ekub round and term management	Major	Oct 18, 2021
3	Ekub member back account management	Major	Oct 24, 2021
4	Invitation and request features	Major	Nov 6, 2021
5	Message and notifications	Major	Nov 10, 2021
6	Menu order arrangement	Minor	Nov 3, 2021
7	Layout adjustment	Major	Nov 4, 2021
8	Eligible user identification	Major	Oct 18, 2021
9	Data type change	Minor	Oct 24, 2021
10	Bug fixing	Minor	Nov 6, 2021
11	Bug fixing	Minor	Nov 10, 2021
12	Datatype change	Minor	Oct 18, 2021

Table 4.2: List of changes recorded for control group project (Changes Made on Online Student Advisory System)

No	Change	Change Type	Date
1	Role based authentication	Major	Oct 25, 2021
2	Template Change	Major	Oct 27, 2021
3	Advisor Type Addition	Major	Nov 5, 2021
4	Student profile image addition	Major	Nov 11, 2021
5	Bug fixing	Minor	Oct 25, 2021
6	Navigation rearrangement	Minor	Oct 27, 2021
7	Database column addition	Minor	Nov 5, 2021
8	Data type change	Minor	Nov 11, 2021

9	Bug fixing	Minor	Oct 27, 2021
10	Database column removal	Minor	Nov 5, 2021
11	Bug fixing	Minor	Nov 11, 2021

In terms of deliveries, greater number of deliveries are committed for the experimental group project than the control group project. From a measurement of an experiment on delivery, three deliveries are registered by control group project and five deliveries are registered for experimental group project as indicated in tables [4.3] and [4.4] respectively.

Deliveries for Ekub Management System

Table 4.3: Number of deliveries registered for experimental group experiment

No.	Delivery	Date	Description
1	Delivery 1	Oct 10,2021	First time delivery by developers
2	Delivery 2	Oct 22,2021	After first three major changes applied
3	Delivery 3	Oct 29,2021	After fifth major changes applied
4	Delivery 4	Nov 12,2021	After seventh major change applied
5	Delivery 5	Nov 19,2021	Final delivery before project closer

Deliveries for Online Student Advisory System

Table 4.4: Number of deliveries registered for control group experiment

No.	Delivery	Date	Description
1	Delivery 1	Oct 19,2021	First time delivery by developers
2	Delivery 2	Oct 29,2021	After first two major changes applied
3	Delivery 3	Nov 12,2021	Final delivery before project closer

Summarized Information of both Projects is presented in table 4.5 below.

Table 4.5: Summary information about control group and experimental group projects

Control Group Project	Experimental Group Project
Online Student Advisory System	EKUB Management System
Run using classical Agile	Run using Agile in Collaboration with DevOps
Have five small sized modules	Have five small sized modules
Three experts participated in development process	Three experts participated in development process
No Operators team participated	Operators team participated with two team members including the researcher
Two months of development time	Two months of development time
Have about 13 product backlog in number	Have about 15 product backlog in number
All team members get basic trainings for two weeks	All team members get basic trainings for two weeks
Automation used for development in small scale (Default scaffolding of Microsoft MVC 6 framework is used)	Automation used for development in better scale (modified scaffolding of Microsoft MVC 6 framework is used)
Automation for integration and delivery is not used	Automation for integration and delivery is not used
Integration performed by developers	Integration performed by DevOps Team
Three deliveries recorded	Five deliveries recorded
Four major change recorded	Seven major changes recorded
Seven minor changes recoded	Five minor change are recoded
DevOps team not facilitate development processes	DevOps team facilitate development processes
Around 65 % of the project accomplished	Around 80 % of the project accomplished

4.2 Discussions

Here in this section the observations from the experiment are presented. For the control group recorded number of major changes are four (4) and recorded number of minor changes are seven (7). Among the major changes two (2) of them raised based on requirement change by customer and the other two (2) are due to design change by developers. The minor changes recorded are due to minor design changes, database column addition, database column datatype fixation, navigation rearranging and look and feel redesigning. The major changes recorded for the experimental group is seven (7) and recovered number of minor changes is five (5). Five (5) of the major changes are resulted due to requirement changes by customer and two (2) of them are by developers' decision on design change. The minor changes recorded under this group are due to database column addition, database column datatype fixation, navigation rearranging and look and feel redesigning.

From the result of the experiment, the researcher observed that the number of major changes recorded for the experimental group is greater than the number of major changes recorded for the control group. On the other hand, the number of minor changes recorded in the control group is greater than the number of minor changes made in the experimental group. The reason for the higher number of major changes and lower number of minor changes recorded for the experimental group is that the DevOps team make the development environment suitable for developers and the developers focused more on the functional requirements. This in turn gives them enough to accept new requirements from the customer. The higher number of minor changes and the lower number of major changes made in the control group is due to the shortage of time to accept new requirements from the customer. In the control group, developers waste a lot of time preparing the development environment, collecting necessary software, and designing the development framework. Generally speaking, developers expend much of their time on non-functional requirements activities. For the experimental group, all these activities are done by the DevOps team and the developers get a significant amount of time to complete their concerning activities with respect to control group project developers.

In terms of delivery, the number of deliveries registered for experimental group project is greater than that of control group project. This is an indication that, the experimental group developers

accomplish sprint backlog in minimal period of time than developers working on the control group. In addition to the improvement on the number of change made on projects and deliveries, the quality of work in terms of reliability, workflow elegance and readability of codes and procedures are observed to be better in experimental group project. From the result of an experiment the researcher interested to work more on the area and prepared developers guideline and described some basic assumption that developers and researchers should consider when using the collaborative approach. This assumptions and guidelines used in order to have detailed understandings about the integrated approach are not closed for further additions and modifications.

4.3 Integrated Framework

Basic assumptions for the integrated framework proposed in this thesis are described in the following four activities to provide better insight for developers as well as researchers.

1. Understanding Basic Building Blocks for Success of Software Development Project

To be successful in software development project, developers and customers must be aware of important building blocks that leads to success. Among the building blocks, project management, product management, supply of resource for development and running environment, automation and manpower are the leading ones.

A. Resource and Development Environment

All necessary resources should be available for developers as well as for operators. The development and operational areas should be equipped with all necessary infrastructure, hardware, and software. During the experimental period of this research, the researcher observed that more challenges faced by the teams due to absence of adequate resources especially the computers they used to run the experimental project takes a lot of time even to run a single program change. The researcher also faces the same thing when he tried to configure GitLab on a local machine and then he decided to handle integration and delivery processes manually.

B. Development Process Automation

Nowadays automation is use for development, testing, integration and delivery processes in software development companies. Automation increases speed of development processes, used to keep working standards, eliminate minor mistakes committed when the development processes handled manually and decrease manpower used for routine activities.

In the integrated approach automation proposed for replacing man power used for routine activities but not to replace skill. Specialization strongly supported in collaborative approached in order to maximize skills in varieties but not in number and distribution of skills across different development activities.

C. Project and Product Management

In software development processes, project management and product management should be clearly identified and treated independently. Project management is holistic phrase that concern with project end to end activities whereas product management is indicates directly related activities to developing products. Project management task related to Agile Development approach and product management tasks related to DevOps approach. Therefore, to be successful in software development project both project management and product management activities should be handled properly.

D. Professional Skills

Even if automations emphasized in collaborative approach, professional skills should not be given lower consideration. All development processes need skill in order to archive activities under certain development process successfully. It is important to increase skills in kind and distribute it thought out the development phases and processes.

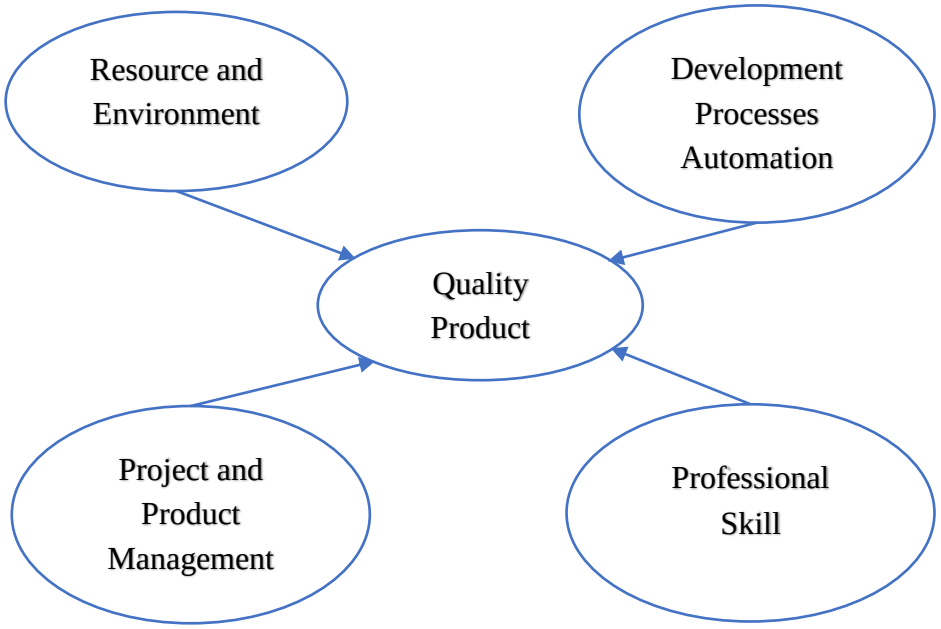


Figure 4.1: Important things for success of software project and quality product

2. Understanding Distribution of Development Activities Thought Timeline.

We can say end to end development ensured when all activities (from project agreement to implementation and maintenance) done in acceptable manner and customer satisfaction is adequate. As indicated in the figure 6.2 below, activities of developers overlap with customers from point A to point B and activities of operators overlap with developers from C to D. The small free space from point B to point C is a time that developers work on the first analysis and design phase. At this time operators have responsibility to prepare development and running environment by collecting and arranging all necessary resources and infrastructures.

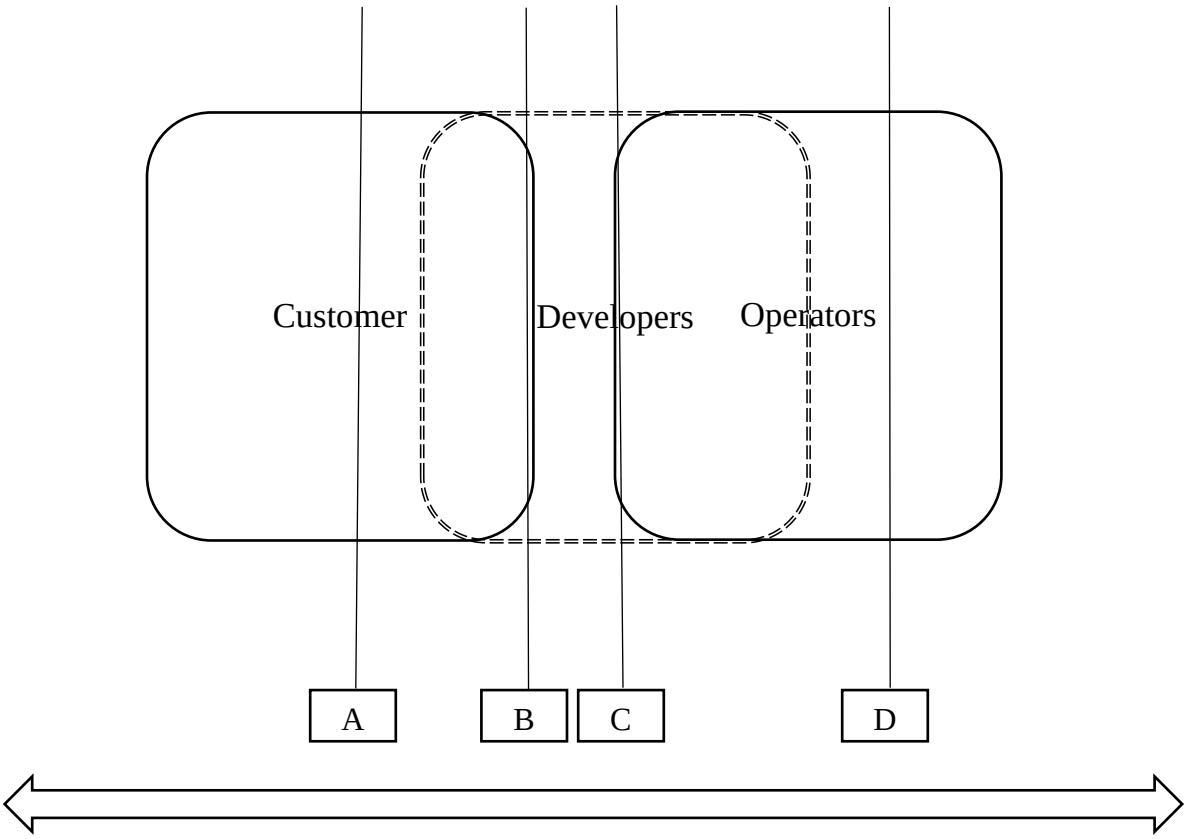


Figure 4.2: Distribution of activities on software project thought time line

3. Defining Well Organized Communication Channels Between Customer, Developers And Operators

In the integrated approach the researcher drafted different communications channels between customers, developers and operators. These communication channels used to communicate in different format including, virtual and face to face meetings, email and letter based communication and file sharing.

Customer ←→ Developers: Issues like term of reference, contract agreement and requirement gathering will be communicated through **customer-developer** channel.

Developers ←→ Operators: Software deploying and hosting, system optimization and continues deliver releases will be communicated through this channel

Customer ←→ Operators: Issues related with running environment like security, performance and stability will be communicated through this channel

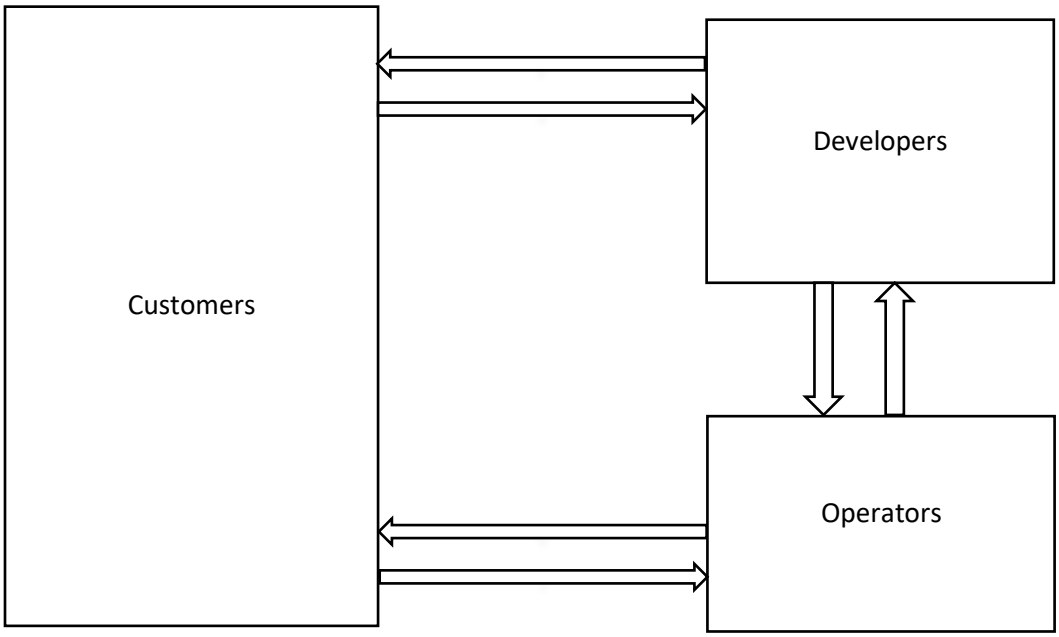


Figure 4.3: Communication channels between customer, Developers and operators

4. Identifying Relations That The Three Entities Have With Product

All the tree entities have tied relation with developing software. They contribute for development as well as get significant benefits from developing software. The benefits and contribution of the three entities with respect to developing software indicated as follows.

A. Customers

Customers contribute to developing software by initiating the project, by providing requirement, by funding the project and by giving feedback. The benefit that the customer get from developing software in handing their business activities in the manner they desired and increase productivities of their businesses by using developing software.

B. Developers

Career, personal and organizational profile and skills are among the benefits that the developers acquire from developing software project. Developers are the main role player in software development projects. They are responsible for providing working software for customers based on their requirements. They are expected to work even more than customers' requirements, since customers focus only on their business processes while developers take care of all aspects that related to developing software like security, performance, ease of use and all other non-functional requirements.

C. Operators

In this integrated approach responsibility of operators extended from operational environment to development environment. They are responsible for facilitating development and operational environment in software development project.

The benefits that operators get from developing software is the same as that of the developers, but since they participate in both development and running environment, the experience they get will be more adequate than that of the developers acquired. The relationship between the three entities with product is illustrated in figure 4.4 below.

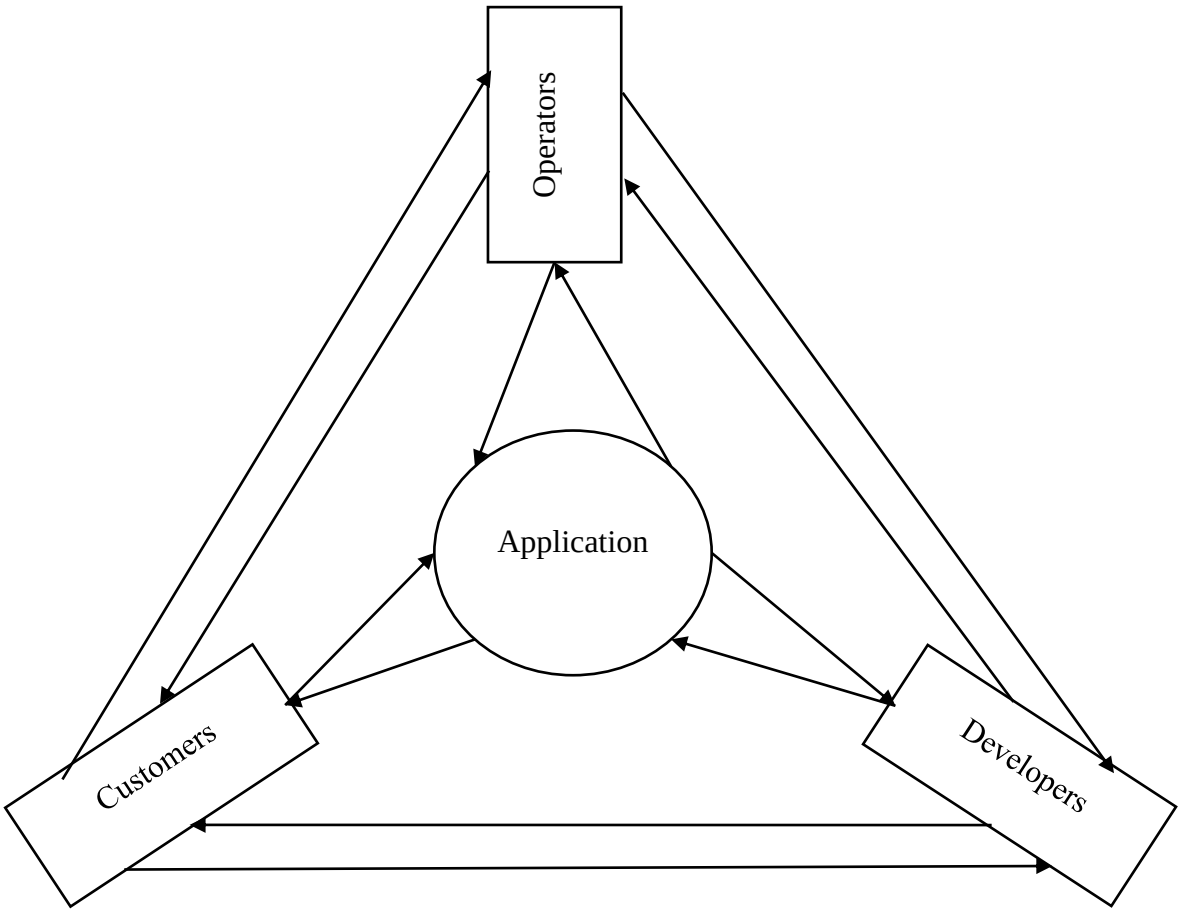


Figure 4.4: The relation of product with customers, developers and operators

Illustration of Integration of agile and DevOps

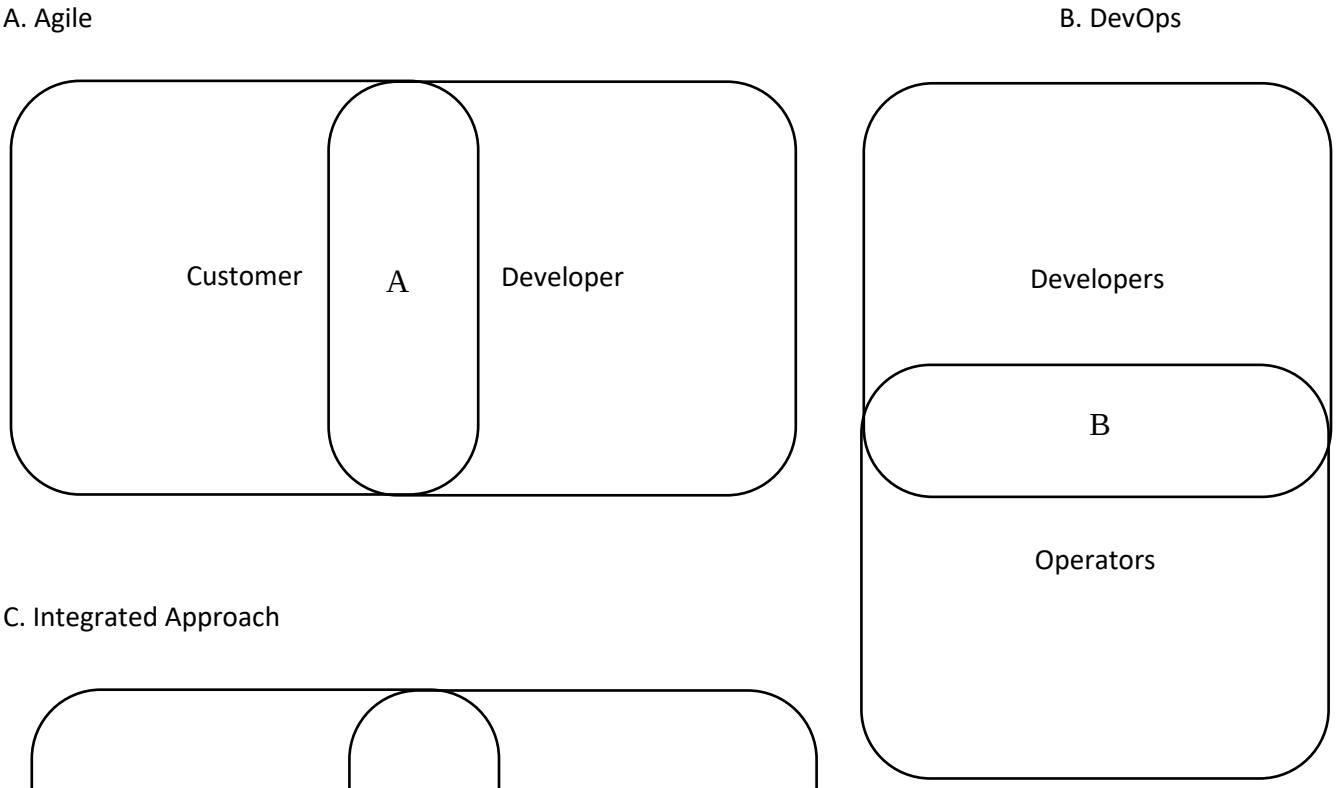


Figure 4.5: Simple illustration of integration of Agile and DevOps.

Descriptions

A: - Intersection between Customer and Developer

B: - Intersection between Developer and Operators

C: - Intersection between Customer and Developer in an integrated approach

D: - Intersection between Developer and Operators in an integrated approach

E: - Intersection between Customer, Developer and Operators in an integrated approach

Development and integration process illustration using flow chart

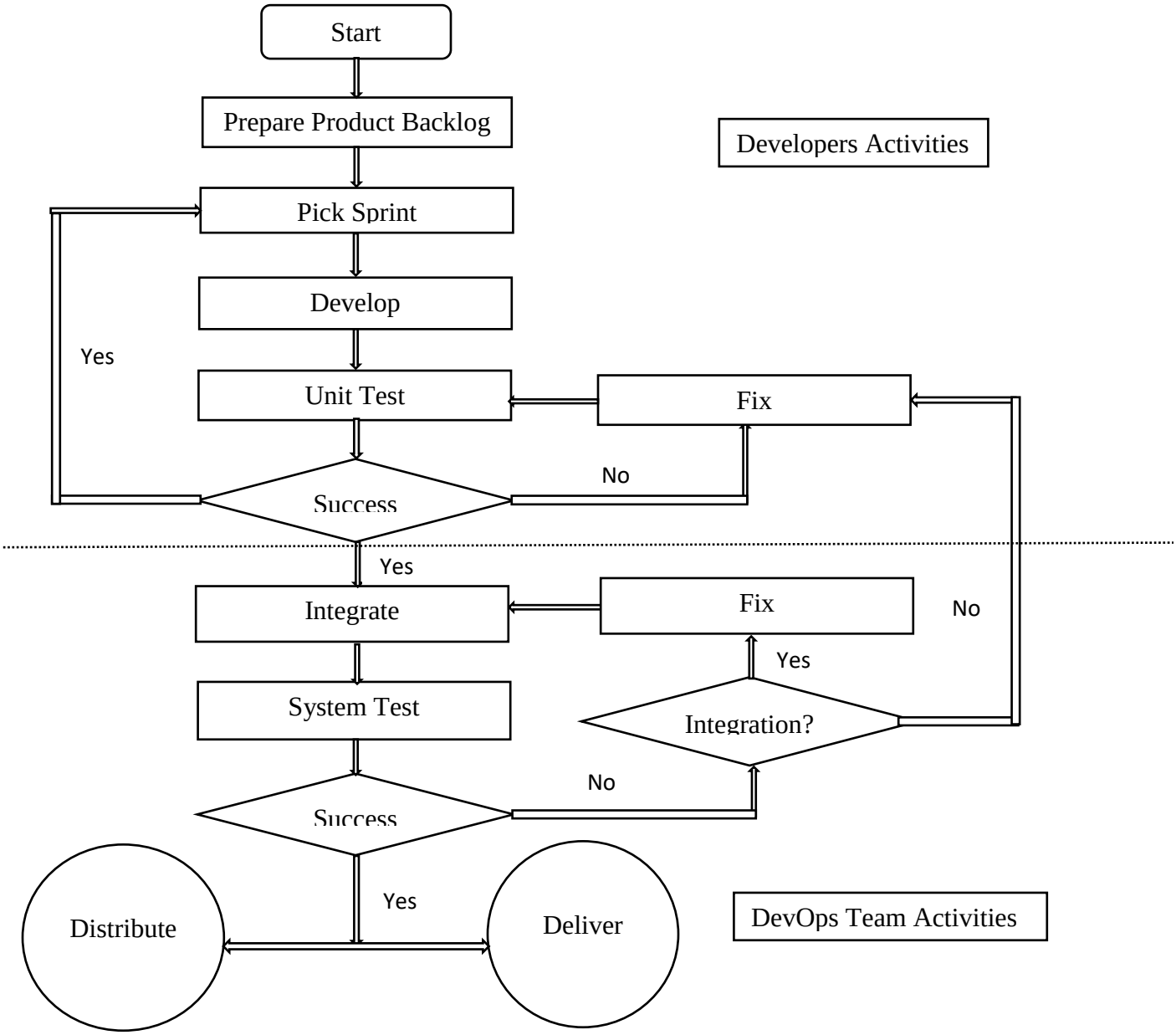


Figure 4.6: Integrated approach workflow representation

Workload of Developers and Operators

Workload representation of Developers and Operators team through development life cycle.

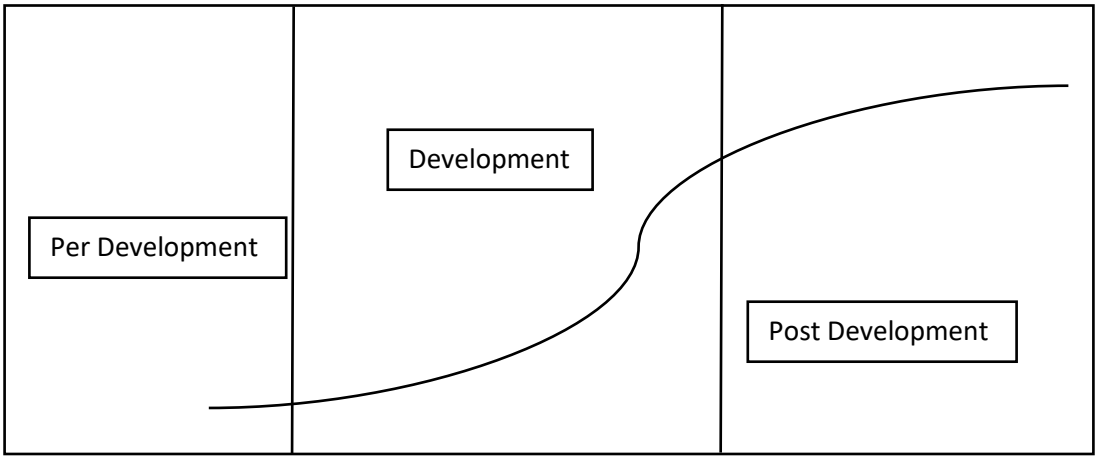


Figure 4.7: DevOps team workload illustration

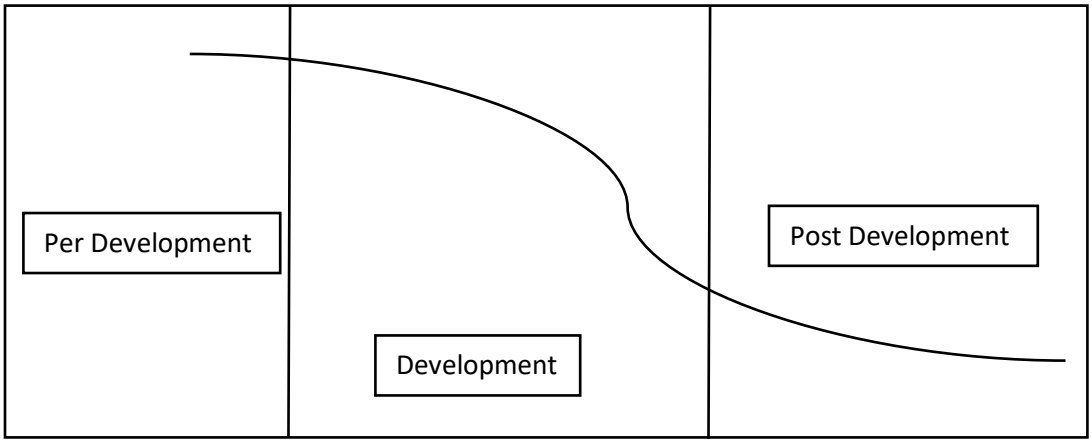


Figure 4.8: Developers team workload illustration

4.4 Guidelines for Using Integrated Approach

This guideline is used in this research work for the experimental group project. Some modifications are made to the guideline after conducting the experimental work. The guideline prepared to be used and improved by developers and researcher who interested evaluating the integrated framework. The guideline covers all the basic activities which are done in end to end software development processes. In addition to this guideline, developers and researches who use the integrated approach should follow all useful practices from both Agile and DevOps development approaches according to their project characteristics. This guideline is not limited for further additions and modification. All the steps and procedures that must be follow by developers and researchers are discussed in the following subtitles.

Initiate the Project

Most of the time project initiation comes from customers but sometime developers also motivate customers to look for automations. The initiation of project is the first step in software development project. Project initiation have significant impact on success of the software project depending on the aim of initiation and initiator.

Identify Project Scope and Complexity

Identify project scope and complexity is used to determine the budget of the project, time to finish the project and human resource and skill required in development processes. A software project that follow collaborative approach starts with optimum size of scope and increases through project time. Too small and too large scope are not recommended in collaborative approach. That is way if project scope is too small it is difficult to get general picture of the project and if the project scope is too large it is difficult to identify pivot of the project.

Making Agreement

Agreement between customer and developer is the second step in software development project. The agreement will be made in terms of time, cost and quality of software. The agreement used to bind developers and customers with regard to issues raised on the agreement documents.

Developers and Operators Team Formation

Good team formation is one of the factors that leads to success of software development project. Development companies with well-organized teams and good project management skill will have better achievement than the companies that have poor team structure and poor project management skill. Agile development have different teams that work in different areas of specializations and different parts of the development processes. In DevOps there are operator

teams that works on running environment activities. In collaborative approach, the role of operational team going to be extended to development environment in order to facilitate development environment for developers. So that the collaborative approach should have two teams one development team that work on development processes and operational team that will work on development environment as well as operational environment. Basic activities that going to be accomplished by DevOps team are indicated as follow:

A. Development Environment Activities

The first responsibility of DevOps team is choosing appropriate platform for development, deployment and hosting. Appropriate platform should be selected by considering the Experience of the developer Company, Availability of supporting technology, previous history of organization behavior of the developing software and size and complexity of developing software. In the second place, choosing development technology should be done by DevOps team. Even if Developers work in development environment, DevOps team should participate in preparing development environment in order to reduce burden from developers. Development technology should be selected based on platform on which the desired software to be run or hosted. Simplicity, security and maturity of development technology are also among the parameters to be considered.

Preparing standard for coding (classes, functions and variables) is one of the activities done by developers. But in this integrated framework DevOps team should participate in preparing standards for development environment works. Programmers and document writers should follow the standards to perform their respective activities.

Preparing code snippets and code generators.

> Code snippets and code generators are used to facilitate developer's task by reducing time to complete the task and increase quality of code in significant amount.

Preparing project framework

> All necessary resources should be arranged in appropriated way for further accessing and development.

B. Preparing Common Components and Tools

This common components and tools are used in developing software and they have significant role to facilitate development process. These components are not directly related to functional requirements of the project. They will be designed independently from main business processes of the developing software. Some examples of this components and tools are listed below.

User Management

User management is used almost in all software application and used to determine the identity of system user, to identify user specific resource and to manage user profile. User management is not completely distinguished from main business process. But it is similar in almost all application. So that DevOps team can handle activities related with user management and then developer integrate it with main business process of the software accordingly.

Localization

Localization is one of the essential issues in software development, some software need more than one human languages to operate it. For that kind of software developing localization framework independent of the main business processes developing software will speedup overall software development process.

Layout and Look and Feel

Software appearance is among one of the most important things those determines quality of the software. Before staring implementing desired software, the look and feel of the software must be designed, that includes which color to be user, how the logo of the system placed, what the list looks like, where and how controls placed on CRUD pages. Navigation is also among one of the most important aspects that makes the system easier to use by end users. Navigation from one part of the software to another

Back Up and Restore

All data intensive application should have back and restore functionality. In absence of backup system it is difficult to insure security of the software. In many cases system damage will be caused and at that time there should be a mechanism restoring latest backup data. DevOps Team is responsible to develop and manage backup system in development as well as operational environment.

C. Prepare Running Environment

The primary task of the DevOps team is managing the running environment of the application. The running environment should be ready to accept continuous deliveries from developers. All necessary infrastructures, hardware, and software should be ready in the running environment. Security and performance issues are also among the activities done by operators in the software hosting environment. In addition to the scram ceremony, the Developer-Operators meeting should be started at the beginning of the project launch. This meeting is crucial to have a common ground on developing software among team members. After the first meeting, the two teams (Agile and DevOps) go to their respective activities and the second general meeting will be held when they are ready to starting development.

Prepare Product Backlog

List of activities done during development process by developers should be listed described according to their priority. We can identify priorities ether based on order of dependably or based on customers demand. Dependency enforces developers to give higher priority for particular part of the developing software. Developers should arrange system components based on dependency level. This helps developers to achieve their tasks with minimal effort smoothly. Sometimes customer want some part of software to be developed and delivered first based on their business need. In such case developers imposed to give higher priority for that particular part of the product. Customers always look for a solutions based on their argent problems they faced. That area should be focus area for developers in order to meet customers need.

Gather Requirements Based on Priority

After a pivot of the project identified development team have responsibility to collect initial user requirements. These requirements use as start point for project and the incremental processes starts from this point. All gathered requirements should managed preparedly and verified by customers before the design and development processes proceed. Requirement changes and new requirements will be entertained during development even after development process completed since the collaborative approach inherits agile as its basic building block.

Developers Operators General Meeting

Discussion of developers and operators on different aspects of project area is essential to have common ground on developing project and development processes. Collaborative approach also used by arranging Agile and DevOps meetings accordingly.

Design and Development

System design and development the two core phases in software development life cycle.

Design starts after the first insight requirement gathering and system analysis. Requirement gathering, designing and development process will be continued until all maturity of the system confirmed by end uses according to the agreement made between developers and customers.

Continuous Integration

Integration is required because the software is developed by more than one developer based on project backlog. Different developers work on different parts of the software and that needs integration in order to place the software on the running environment. Continuous integration is the result of continuous development and continuous development is also the result of frequent changes of requirements by customers. DevOps's main mission is managing activities of Continuous Integration and Continuous Delivery in software development projects. In an integrated approach managing development environment will also be the responsibilities of DevOps team.

Feedback and Support

We can divided the end to end software development into three major parts; pre-development, development and post-development. Pre-development is a part that includes processes like project initiation, contract agreement, team formation, resources allocation and environment arrangement. This part have significant impact on overall success of the software project. During this phase customer and the developing company have more responsibility and the work burden is also significant. Development part seems to be the most challenging part in development process, but from experience of the researcher, post development part is equally challenging as development part in development process. That is way all the activities done in post-development part are so tedious. This activities includes implementation, data migration, data

mitigation, maintenance, feedback collection, user training and support. So that the weight that given for development part should also be given for post- development part. Feedback and support are essentials for software maturity. The two activities start when users start using the system. There should be proper way to collect feedback from end users and dedicated body to provide support for users when they face difficulties. Feedback is useful for improving the quality of the software. Without user feedback, it is difficult to identify the needs of the customers. Software development approaches should have a well-known channel in order to collect feedbacks from end users. The integrated approach define a well-organized channel to collect feedbacks from system end uses. Support is one of the post-development part that increases customer satisfaction when performed properly and that in turn increases the trust on Developer Company.

End Users Trainings

End user should be trained until they internalized the system. Lack of sufficient training leads to commit mistakes by end users. Training should be role specific, that mean each user should be trained on areas that he/she assigned for. Flow of system implementation and data migration have to be clearly indicate in the training process. Poor training sometimes raises negative feedback on the software even if the software developed properly according to customer requirements.

Document Preparation

In the software development projects that follows collaborative approach, some documents are mandatory due to their necessity during development as well as implementation. The collaborative framework have room for a team that works on documentation. From the beginning of the project to the end of the project, the documentation team should work with developer team and operator team collaboratively.

List of Documents in Integrated Approach

History of the Project

From project initiation to implementation all processes, challenges, good practices and participations should be registered. Recording history used for further improvement of the product, understanding challenges faced and the way the challenges solved. This documents should be available for customers as well as for developers.

Project Backlog

All functional and non-functional requirements should be recorded as a user history. Early and late requirements from customer should be managed properly. Arranging requirements in a suitable way for development pipeline by listing, describing, categorizing and prioritizing have significant role on success of software project.

Development Processes

Writing descriptive documentation about developing system's functionalities, components, used procedures and development and running environment technologies. This is useful for others to understand the system for future scalability.

Integration and Delivery Plane

After development the software should be integrated and delivered to running environment. This task can be accomplished either manually or by automated way. Integration and delivery plan should be prepared before the software project start. The plan should include method of integration and delivery, frequency and priority of tasks in integration and delivery.

Implementation Plane

Developing a working software with given requirements by itself is not the final goal of a software project. After the software is developed, there should be a plan to running environment to give desired services properly. The implementation plan document should contain a list of required infrastructures, a list of required software and tools, a data migration and mitigation plan, end-user training documents, a security insurance plan, feedback collection and maintenance format, and backup and restore procedures.

User Manuals

The user manuals are very important documents that should be included in deliverable. User manuals enable end users to use the system without difficulties. This in turn decreases the cost of training and complains that comes from the end users. User manuals should be written in a way the user can easily understand the system and should be role specific. The user manuals should contains terms of use of the system, general system descriptions, responsibilities of each role, detailed explanations of each components of the system, usage flow of the system and duties and responsibilities of end users.

In general documents can be categories in to two main categories in integrated approach.

1. Project Management Documentation
    - Terms of reference
    - Contract Agreement
    - Official Letters
    - Progress Reports
    - Meeting agenda and minute
    - Different Plane Documents
    -
2. Product Management Documentation
    - SRS Documents
    - Product Backlog
    - Sprint backlog
    - Design documents
    - Technology specification document
    - Integration and delivery plane
    - User manuals

Chapter Five

Conclusions and Future Works

5.1 Conclusion

In this thesis, the integrated approach of Agile and DevOps on the software development process which needs continuous development, continuous integration, and continuous delivery is presented. The research was conducted as experimental research by identifying two small-sized projects with a minimum level of expertise who joined Hawassa University Application Development Team for practical attachments. Different pieces of training are given to developers in order to overcome the skill gaps of the developers. Combining the two approaches (Scrum and DevOps) is done by minor modifications to the DevOps team role and merging the activities and practices from both approaches. The number of changes accepted and developed and the number of deliveries in a specific period of time are used as measurement parameters. The result of the experiment indicates that the experimental group project has a better score than the control group project in terms of accepted changes and committed deliveries. Developers of the experimental projects focus mainly on development activities and other activities of the development environment handled by the DevOps team and that in turn makes the experimental group developers more productive and efficient in their work. On the other hand, developers of the control group project waste more time by preparing the development environment, and that results in a negative impact on the development of the core business area. The communication between developers and operators is well managed for the project that follows an integrated approach and weekly scrum meetings include the DevOps team in the communication. As a result, the members of the DevOps team have a better awareness of the experimental project activities and progress than the control group project. Students who participated in this excrement work acquired significant knowledge of software development and prefer an integrated approach for their future development activities. In this research work, automation is used for only development activities but not for integration and delivery activities. Since the experiment is done in a small team size and time duration it is difficult to say the integrated approach gives a maximum level of improvement in software development processes. As a result, the researcher recommends more research to be conducted in this research area. Finally,

the guideline used for an experimental group project work was improved and included in this paper for researchers and developers.

5.2 Future work

This research was conducted to indicate the impact of using the integrated approach on software development processes over using Agile and DevOps independently. The two approaches are applied in many large software development projects with a large number of teams and team sizes. On the other hand, the integrated approach was evaluated using small-sized projects and a small number of teams for a short period of time. It is difficult to say the integrated approach is better than the two approaches used independently unless it is evaluated by applying to different projects of different sizes, with different team sizes and time duration. As a result, the researcher recommends that the research should be evaluated by applying to different projects with different team sizes and different project time duration. In addition to this, future work should be focused on the automation of development processes and tuning team structure to optimize software development processes. In this research work, automation is used for development activities but not for integration and delivery activities. This will have a significant impact on the result of the experiment. As a result, further evaluation should be conducted on the integrated approach by using full automation for both development, integration, and delivery processes. Since the number of teams and team sizes that participated in this research work is small in size, it is difficult to make different team arrangements for evaluation purposes from such small teams. So evaluation should be conducted on different projects with different teams which have different skill patterns. Finally, the researcher asks developers and researchers to contribute supportive ideas to improve the software development process from their experience and also by researching the area.

References

1. P. Raj and P. Sinha, “Project Management in Era of Agile and DevOps Methodologies” on In International Journal of Scientific & Technology Research Volume 9, Issue 01, January 2020
2. Sikender Mohsienuddin Mohammad Sr. Associate, KPMG LLP, Department of Information Technology, Wilmington University, “DevOps Automation and Agile Methodology’ on International journal of creativity research thoughts (IJCRT), an International open access, peer-reviewed, refereed journal.
3. Mali Senapathi, Jim Buchan, Auckland, Hady Osman, University of Technology Auckland, New Zealand, “DevOps Capabilities, Practices, and Challenges: Insights from a Case Study”
4. Hemon, Barbara Lyonnet, Frantz Rowe and Brian Fitzgerald “From Agile to DevOps: Smart Skills and Collaborations”
5. Leonardo Leite, Carla Rocha, Fabio Kon And Dejan Milojicic “Survey of DevOps Concepts and Challenges”
6. F.M.A. Erich, C. Amrit & M. Daneva “Qualitative Study of DevOps Usage in Practice”
7. Cheng Wang Changling Liu “Adopting DevOps in Agile Challenges and Solutions”
8. Gaurav Kumar, Pradeep Kumar Bhati “Impact of Agile Methodology on Software Development Process”
9. https://www.cprime.com/resources/what-is-agile-what-is-scrum, “What Is Agile and What Is Scrum?”
10. https://www.agilealliance.org/agile101, “What is Agile?”
11. https://en.wikipedia.org/wiki/DevOps, “DevOps”
12. https://hygger.io/blog/a-brief-history-of-agile-methodologyrs
13. https://www.bmc.com/blogs/devops-history
14. https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development
15. https://agilemanifesto.org
16. Tesfaye Biru “A Collaborative Learning Oriented Approach to Software Development and Process Improvement”
17. Fernando, JF, SF “Exploring the Benefits of Combining DevOps and Agile”

18. Saliya Samarawickrma “Continuous Scrum: A Framework to Enhance Scrum with DevOps”
19. Aditi Jain “DevOps Lifecycle” https://www.knowledgehut.com/blog/devops/devops-lifecycle

Appendices

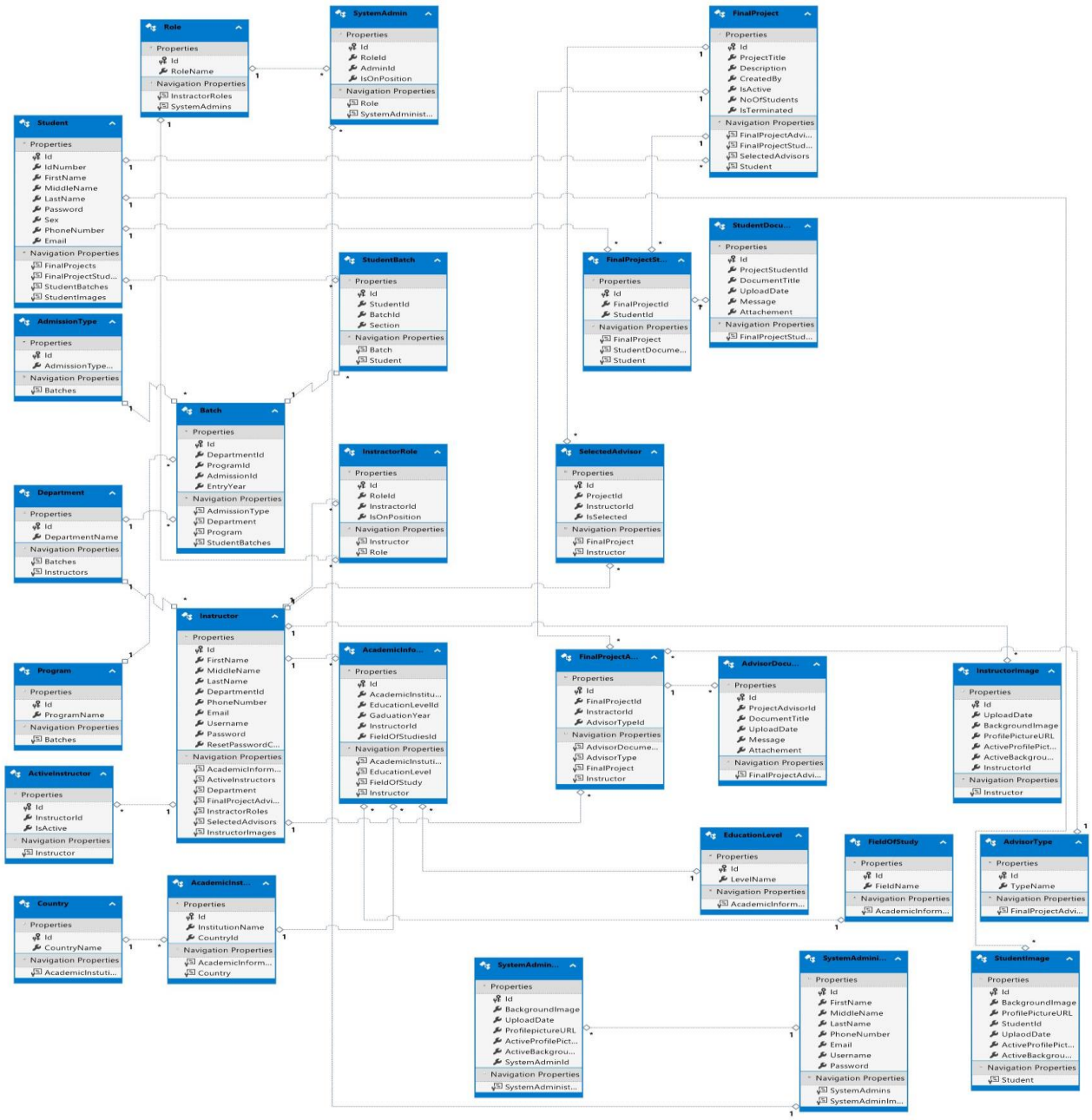
Appendix A: - Entities of Ekub Management System

- ☑. Users
- ☑. Ekubs
- ☑. Banks
- ☑. User Address
- ☑. User Bank Accounts
- ☑. Ekub Bank Accounts
- ☑. Ekub Types
- ☑. Address Type
- ☑. Member Types
- ☑. User Images
- ☑. Winner Payments
- ☑. Ekub Members
- ☑. Ekub Payments
- ☑. Messages
- ☑. Invitations
- ☑. Winners
- ☑. Requests
- ☑. Roles
- ☑. Ekub Terms
- ☑. Message Receivers
- ☑. User Roles
- ☑. Roles

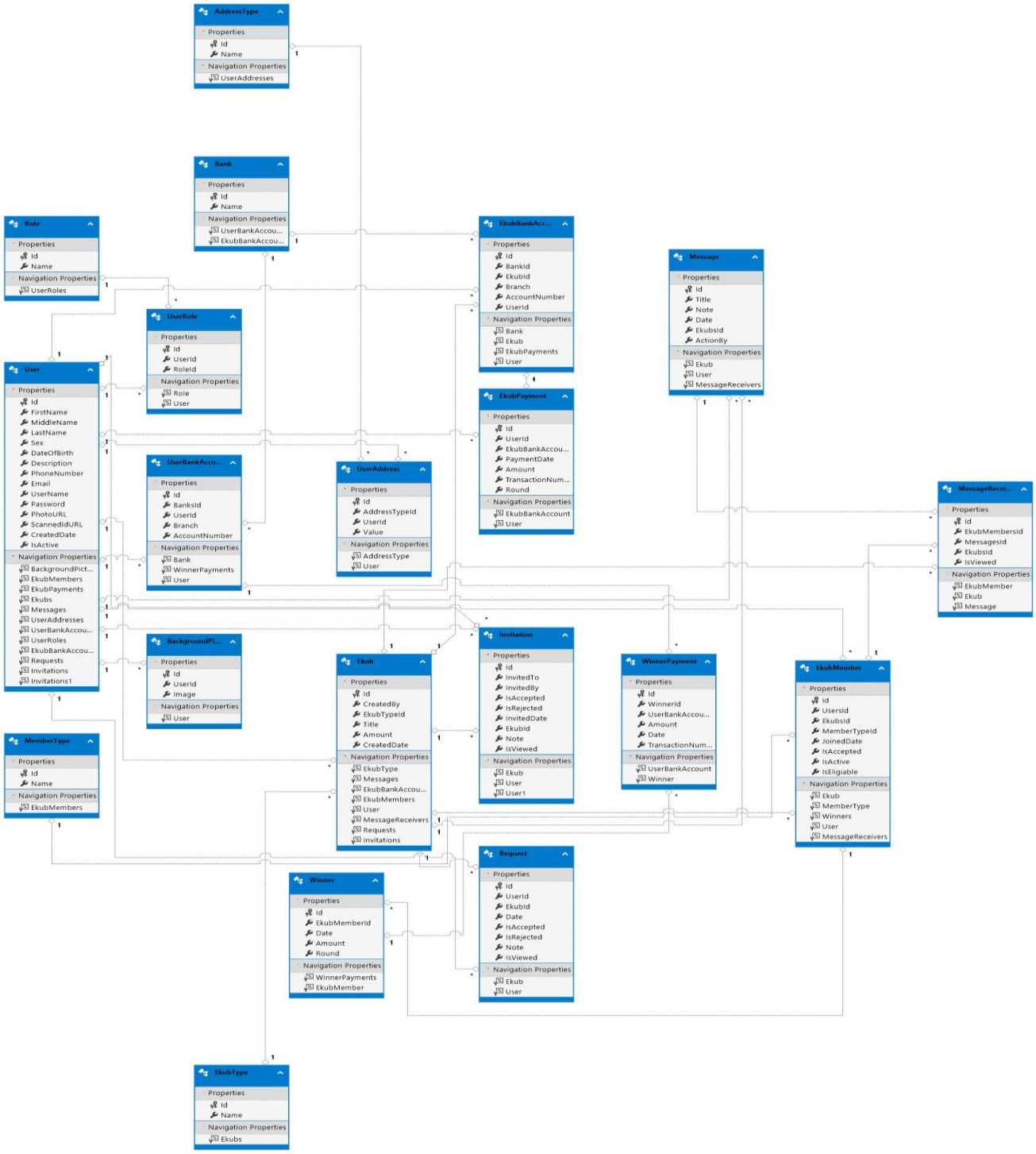
Appendix B: - Entities of Online Student Advisory System

☑. Academic Information
☑. Academic Institutions
☑. Active Instructors
☑. Admission Types
☑. Advisor Documents
☑. Advisor Types
☑. Batches
☑. Countries
☑. Departments
☑. Department Information
☑. Education Levels
☑. Field Of Studies
☑. Final Projects
☑. Final Project Advisors
☑. Final Project Students
☑. Instructor Roles
☑. Instructors
☑. Instructor Feedbacks
☑. Instructor Images
☑. Programs
☑. Roles
☑. Selected Advisor
☑. Students
☑. Student Batches
☑. Student Documents
☑.Student Feedbacks
☑.Student Images
☑.System Admin
☑.System Admin Images
☑.System Administration Information

Appendix C: - ER Diagram of Online Student Advisory System

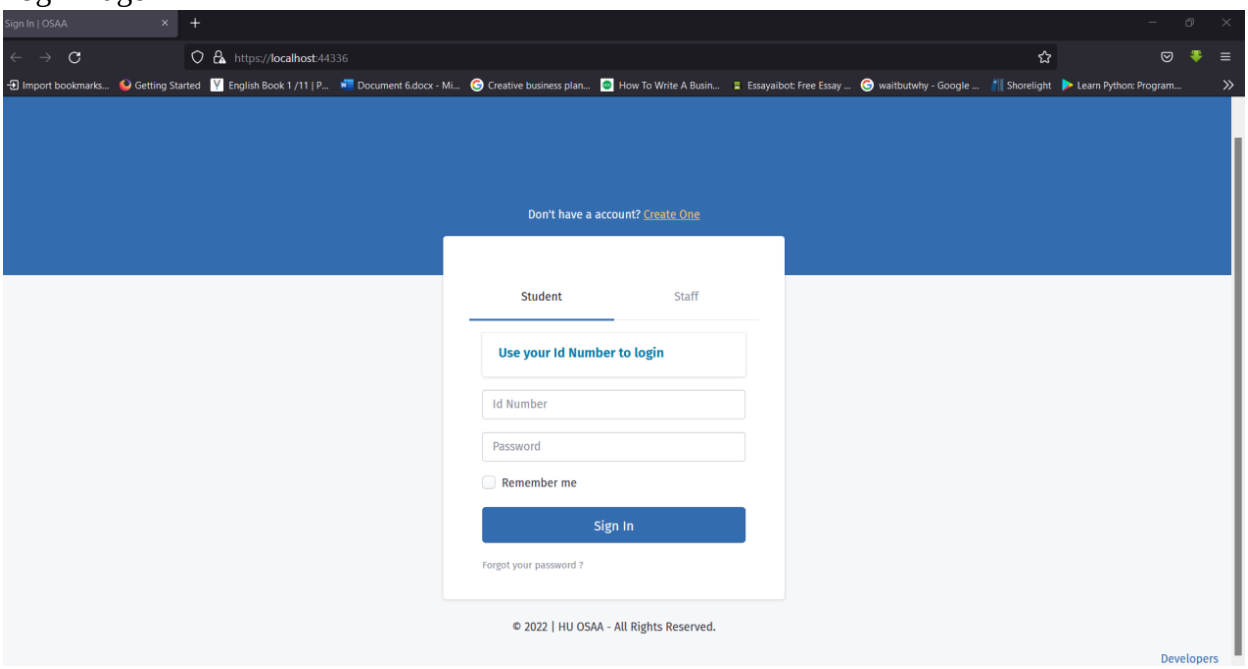


Appendix D: - ER Diagram of Ekub Management System

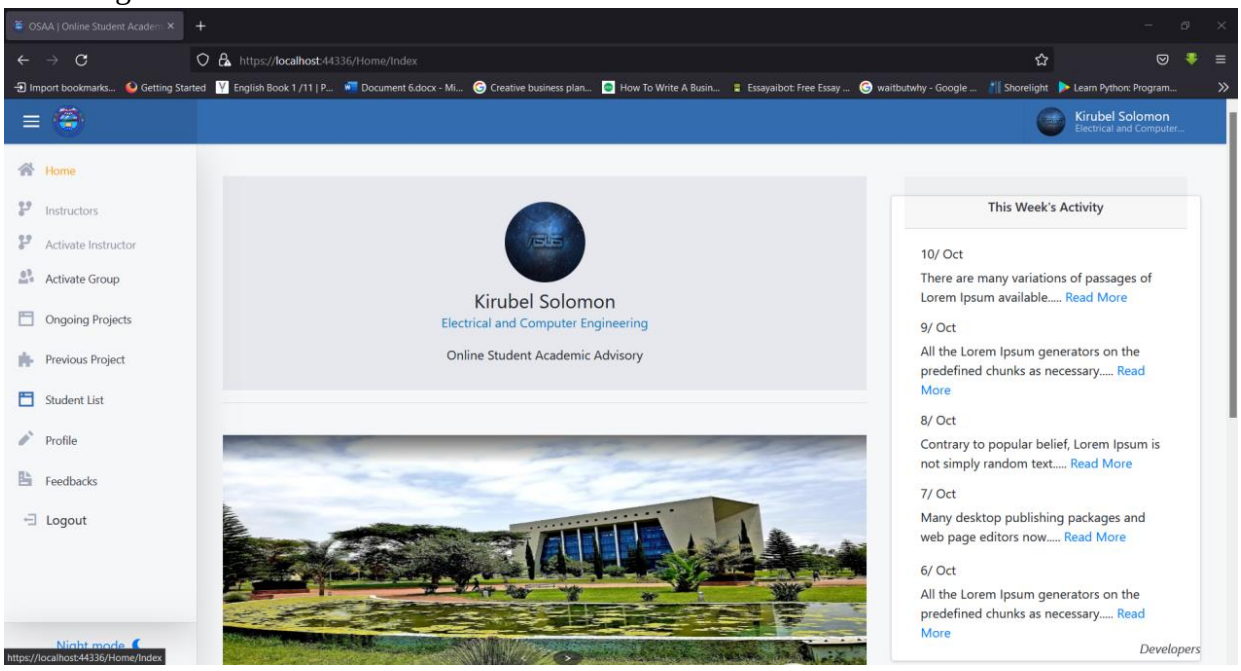


Appendix E: - User Interface of Online Student Advisory System

Login Page

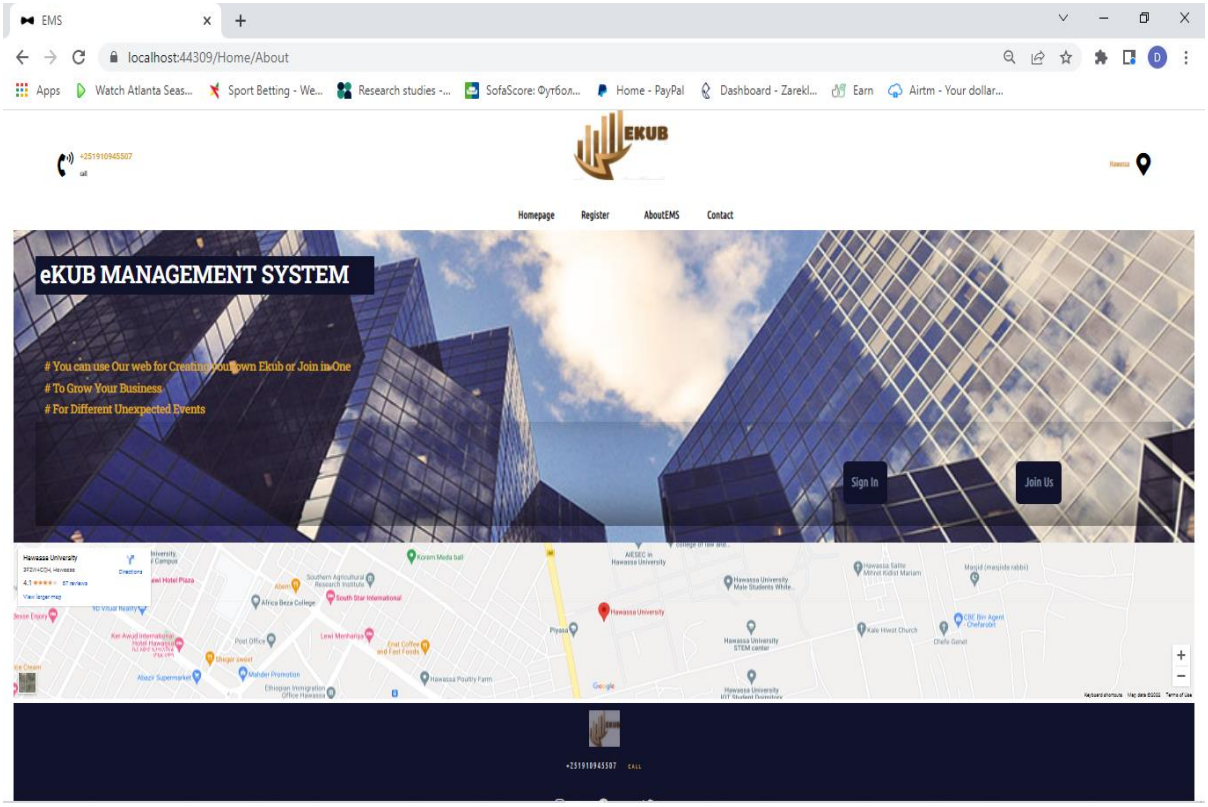


Home Page

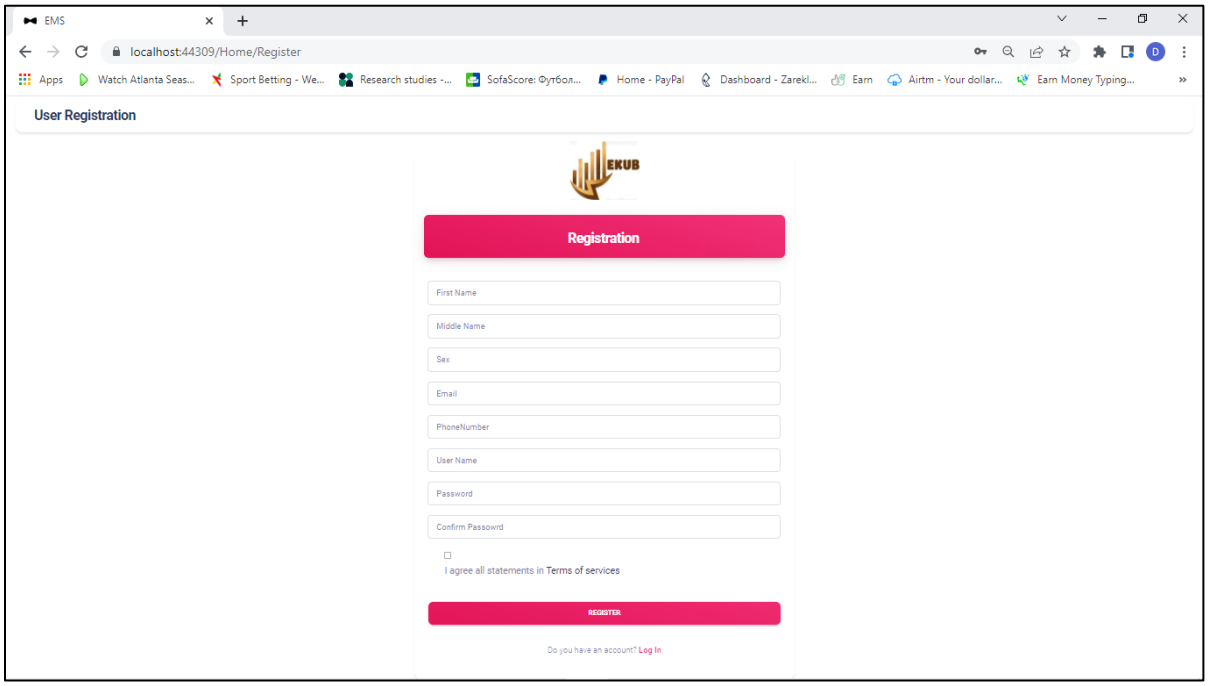


Appendix F: - User Interface of Ekub Management System

Home Page



Registration Page



Login Page