UNIVERSITY OF EDUCATION, WINNEBA

GROWTH OF SOFTWARE FRAMEWORK USABILITY AND ITS EFFECTS ON SOFTWARE ENGINEERING

GIDEON AGYEMAN OWUSU

MASTER OF SCIENCE DISSERTATION

2021

UNIVERSITY OF EDUCATION, WINNEBA

GROWTH OF SOFTWARE FRAMEWORK USABILITY AND ITS EFFECTS ON SOFTWARE ENGINEERING

GIDEON AGYEMAN OWUSU

A dissertation in the Department of Information Technology Education, Faculty of Applied Sciences and Mathematics Education, submitted to the School of Graduate Studies in partial fulfilment of the requirements for the award of the degree of Master of Science (Information Technology Education) in the University of Education, Winneba

MAY, 2021

DECLARATION

STUDENT'S DECLARATION

I, **GIDEON AGYEMAN OWUSU**, declare that this dissertation, with exception of quotations and references contain in publish works which have all been identified and acknowledge. I hereby declare that this dissertation is the result of my own original research and that no part of it has been presented for another degree in this University or elsewhere.

SIGNATURE: …………..…………………………

DATE: ………………………………………………

SUPERVISOR'S DECLARATION

I hereby declare that the preparation and presentation of the dissertation was supervised in accordance with the guidelines for supervision of dissertations laid down by the University of Education, Winneba.

NAME OF SUPERVISOR: **DR. SAMUEL ADU GYAMFI**

SIGNATURE: …………..…………………………

DATE: ………………………………………………

DEDICATION

The project is dedicated to my family, friends and to all sundry who worked effortlessly to help make this dissertation a success.

ACKNOWLEDGEMENT

I am deeply grateful to my supervisor, Dr. Samuel Adu Gyamfi for all the encouragement, inspiration and guidance. To all the faculty members at the Department of ICT Education, College of Technology Education, UEW. I am very grateful for your wonderful support and guidance. Also, I express my deepest gratitude to my family. I would have never come this far without their encouragement and financial support. Again, my deepest gratitude to all my colleagues, Adutwumwaa, Ephraim, Andy for the constant motivation and assistance.

I would like to thank Mr Bashirudeen Saeed Nabie, who supported me very well in this dissertation writings and report analysis.

Lastly, I would like to thank my wife, Paulina Nyarko Gyimah, who has supported me and helped me persevere through my graduate career.

I will ever treasure the help from both mentioned and non -mentioned well-wishers deeply in my heart.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE **PAGE**

ABBREVIATIONS

UDS	University for Development Studies
UEW	University for Education, Winneba
UNER	University of Energy and Natural Resources-Sunyani
KNUST	Kwame Nkrumah University for Science and Technology

ABSTRACT

The researcher found that few study measures the growth of software usability and its effects, so we want to measure it and then try to improve on our software development industry. The purpose of the study is to find out the growth of software framework . usability and its effects software engineering or development with respect to computer programming. A descriptive survey study was employed. The targeted population (277) for this study covers the final years computer science or information technology masters' students in five public universities in Ghana and two software companies (Gracecoms – programmers and Kologsoft – programmers). The sample size was 249 students' software developers. The questionnaire tool was used to collect data. The data had been gathered through MS Excel then exported to the Statistical Packages for Social Sciences (SPSS) for analyses. The analyses measure the association between socio-demographic and other related variables in the survey by using frequencies, percentage mean and standard deviation. The study found that there is very high/ high

60% (n= 149) level of growth of software framework usability in software engineering. And the popular software frameworks use was bootstrap 32.1% (n=80) and flutter 26.1% (n=65). Secondly, the study found that the most leading cause students and software developer identified is*:* It is faster to develop a system with framework than pure coding with the highest mean score of (means = 2.67 std = 0.64) respectively. Lastly, the study found that even though effect of Software Framework usability contributed significantly to the statements, there is a positive relationship between effect of Software Framework usability and the software development to a large extent.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

A framework is an integrated collection of components that collaborate to produce a reusable architecture for a family of related applications (Njeru, 2014). Again, software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software (Christain, Tomasz, & Jaroslaw, 2010). A software framework provides a standard way to build and deploy applications. A software framework is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products and solutions. Software frameworks may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or system (Christain, Tomasz, & Jaroslaw, 2010). Software frameworks have many powerful features that separate or make them different from a pure coded application or libraries and other normal user application that are in the system. Some of these features includes (1) Default behaviour – A framework has its own default behavior, which must actually be some useful behavior for new programs it is going to be used for and not a series of noops; (2) Inversion of control – The overall program's flow of control and its algorithm are not dictated by the caller, but by the framework itself; (3) Non-modifiable framework code – The framework code in general, is not allowed to be modified. Users can extend the framework, but not modify its code; (4) Extensibility – A framework can be extended by the user/programmer usually by selective overriding or specialized by user code providing specific functionality. Design patterns represent solutions to problems that arise when developing software within a particular context, capture the static and dynamic structure and collaboration in software designs thus facilitate reuse of successful software architectures and designs. Software systems can be very complex constructions and can span into years of development (Ragnarsson, 2014). Overtime software engineering processes have sought to reduce time to market, reduce the cost of development, standardize software development, improve quality, improve reliability and reduce the complexity in process management (Riehle, 2000). Also, many software developers use software framework for developing their software to beat time, cost and improve the quality of software they are developing (Ragnarsson, 2014). Frameworks help increase the performance of software, increase the capabilities of software and also provide a bank of codes for programmers to use so that they do not make traditional programming. Aside making development of software easy, they also make room for both front-end and back-end development meaning the stress of going through hard coding to get logics of a particular algorithm working for a particular system is shorten and now the focus is on how to make good use of a particular software framework in a way that you can use in your software development. It is also known that each programming language currently in existence has at least one universal, reusable framework supporting it users or programmers (Upworks, 2018). A software framework provides an abstraction where generic functionality can be selectively overridden or specialized by user code. The overall development time will be cut into minimum as it concentrates on the low-level details of a working system (Mary & Rodriguez, 2012). Frameworks are like jet packs for development languages: They help increase performance, extend capabilities, and offer libraries of coding shortcuts so developers aren’t hand-coding web applications from the ground up or from scratch.

Frameworks aren't just bundled snippets of code; they offer many more features like models, APIs, and other elements to streamline development of dynamic, rich web applications. And while some frameworks offer a more rigid approach to development, others allow for more fluidity in the process—developers can pick and choose based on project needs or their own work styles. (Upworks, 2018). Most software systems though implement in part what has already been built and tend to follow known or nearly known architectures. It is notable that complexity of software development can be quick to increase due to its nature such as flexibility, extensibility, hidden constraints and the lack of visualization to the owners. Thus, among these methods that are the most important in re-use of the known is the use of architectural and design patterns and software frameworks. It's notable though frameworks are usually domain-specific and applicable only to families of applications (Riehle, 2000). Most of the users of software framework know how to use it for providing quick solutions but few have much knowledge about the effects of using software framework for software development. This study is to find out the knows and unknows about the use of software framework for software development.

1.2 Statement of the Problem

Software systems can be among the most complex constructions in engineering disciplines and can span into years of development. Most software systems though implement in part what has already been built and tend to follow known or nearly known architectures (Njeru, 2014). Although most software systems are not of the size of say Microsoft Windows 8, complexity of software development can be quick to increase. Thus, among these methods that are the most important is the use of architectural and design patterns and software frameworks (Njeru, 2014). Patterns provide known solutions to re-occurring problems that developers are facing. There are eminent changes in software engineering about the importance of software framework in software development. Quality and cost delivery of software continue to be the key aspects of software engineering whiles time also continue to be main factor affecting software development. By using well-known patterns reusable components can be built in frameworks. Software frameworks provide developers with powerful tools to develop more flexible and less error-prone applications in a more effective way. Software frameworks often help expedite the development process by providing necessary functionality “out of the box”. Providing frameworks for reusability and separation of concerns is key to software development today. Novice or beginners of programming sometimes becomes confuse as to whether software frameworks are the new programming language or not as stated by Wayner (2015) in his article about the seven reasons why frameworks are the new programming language. The decission to start with software framework or go by traditional programming of coding from scratch is still a big problem for many novice/beginners of programming to decide. Advanced and experienced software developers are moving towards the software frameworks because of it flexibility and easy to use approach indicating that less programmers now do traditional programming but the question now is are all users of framework aware of the consequnces of using software frameworks? The future of software engineering in terms of pure coding or traditional programming is nothing to talk about with this rapid growth of softaware framework usabilities because more powerful and robust softwares are being designed from software framework as a results of this certain softwares are not being able to be manipulated well to meet its’ customer/user satisfaction due to the limitaion of the framework given onto the programmer or due to inadequate knowlegde gained by the user of that particular software framework

(Gamma, 2004). In this study I take a look at the growth of software framework usability and its effects on software engineering. This was the motive why the researcher took it as a challenge to research into the growth of software framework usability and to determine whether it has positive or negative effect on software engineering.

1.3 Purpose of Study

The purpose of the study is to investigate the reason for selection of software frameworks, key things taken into consideration during the selection and to find out the effects that are associated with those selected software frameworks.

1.4 Objectives of Study

The objectives of the study were to:

1. Identify the types of Software Framework use in Software development.
2. Examine the causes of rapid growth of software framework usability in software development.
3. Examine the effects of software framework usability on software development

1.5 Research Questions

The following research questions were raised to guide study:

1. What types of Software Framework do you use in Software development?
2. What are the causes of rapid growth of software framework usability in software development?
3. What are the effects of software framework usability to software development?

1.6 Scope of the Study

The study will be conducted with three different tertiary institutions in Ghana offering computer related programs (Information Technology and Computer Science). Questionnaires and survey will be used to find out the perception of students about software framework, programmers who use both frameworks and traditional programming for solving problems too will be questioned to find out their motivations and challenges of using software framework over traditional programming. This will help in finding out the differences, similarities, advantages and disadvantages of using the two different approaches.

1.7 Significance of Study

The significance attributed to this study are outlined as follows:

Firstly, results of the study might highlight on the main reasons for the selection of software framework over traditional programming.

Secondly, this research may help predict the future of software framework in software engineering.

Lastly, this research may accumulate the knowledge for use by future researchers about the effects of software framework in software engineering.

1.8 Limitations of the Study

Financial problem is the foremost limitation ascribed to the research. This is due to the financial commitment involved in the drafting of the study, setting, and administration of survey questionnaires before the actual presentation. Also, the study was only limited to sample of students and software developers, instead of generalizing it throughout the entire polytechnics and universities in Ghana that is offering computer science or

Information Technology.

1.9 Delimitations of the Study

The research focuses essentially on students and software developers of UEW, UDS, Kumasi Technical University, KNUST, UNER and Gracecoms, Kologsoft. The outcome of the research could not cover entirely the students within the universities and software companies. Nevertheless, to draw an exact outcome and for critical study, some students were sampled from each of the five universities and two software companies in the country for the study. Hence, this research intends to help students and stakeholders to realize the growth of software framework usability and its effects on software engineering.

1.10 Organization of the Study

The research encompasses five chapters and is summarily listed below:

The First Chapter is an introduction which consist of the background of the study, statement of the problem, the purpose of the study, objectives of the study, research questions, the significance of the study, the limitation of the study, delimitation of the study, definition of terms and the organization of the study. The Second Chapter involves the literature review which deals with other personalities view about the problem under study. The Third Chapter focuses on a methodology which talked of the method employed in doing the study. It deals with the research design, population and sampling techniques used in the study. It also consists of the data gathering instrument, validity and reliability, data collection procedure and data analysis as well as ethical considerations. The Fourth Chapter deals with the presentation of the results or findings of the study. Finally, the Last Chapter summarizes, concludes and gives recommendations for the study.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter focuses on the literature review that deals with the review of books and other written resource materials that have made sense on the study topic. The chapter also reviewed under the following headings.

2.2 Software Frameworks

A software framework is a concrete or conceptual platform where common code with generic functionality can be selectively specialized or overridden by developers or users. Frameworks take the form of libraries, where a well-defined application program interface (API) is reusable anywhere within the software under development (Janssen, 2021).

Certain features make a framework different from other library forms, including the following:

- *Default Behavior*: Before customization, a framework behaves in a manner specific to the user's action.
- *Inversion of Control*: Unlike other libraries, the global flow of control within a framework is employed by the framework rather than the caller.
- *Extensibility*: A user can extend the framework by selectively replacing default code with user code.
- *Non-modifiable Framework Code*: A user can extend the framework but not modify the code. The purpose of software framework is to simplify the development environment, allowing developers to dedicate their efforts to the

project requirements, rather than dealing with the framework's mundane, repetitive functions and libraries (Janssen, 2021).

For example, rather than creating a VoIP application from scratch, a developer using a prepared framework can concentrate on adding user-friendly buttons and menus, or integrating VoIP with other functions. Software frameworks consist of frozen spots and hot spots. Frozen spots define the overall architecture of a software system, that is to say its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. Hot spots represent those parts where the programmers using the framework add their own code to add the functionality specific to their own project. In an object-oriented environment, a framework consists of abstract and concrete classes. Instantiation of such a framework consists of composing and subclassing the existing classes. When developing a concrete software system with a software framework, developers utilize the hot spots according to the specific needs and requirements of the system. Software frameworks rely on the

Hollywood Principle: "Don't call us, we'll call you". This means that the user-defined classes (for example, new subclasses), receive messages from the predefined framework classes. Developers usually handle this by implementing superclass abstract methods. While frameworks generally refer to broad software development platforms, the term can also be used to describe a specific framework within a larger programming environment. For example, multiple Java frameworks, such as Spring, ZK, and the Java Collections Framework (JCF) can be used to create Java programs. Additionally, Apple has created several specific frameworks that can be accessed by OS X programs. These frameworks are saved with a .FRAMEWORK file extension and are installed in the

/System/Library/Frameworks directory. Examples of OS X frameworks include

AddressBook.framework, CoreAudio.framework, CoreText.framework, and QuickTime.framework (Techterms, 2013). There are frameworks that cover specific areas of application development such as JavaScript/CSS frameworks that target the presentation (view) layer of the application, and there are others that handle more of the dynamic aspects of the application. Some include both! Examples of frameworks that are currently used or offered by standards bodies or companies include:

- *Resource Description Framework*, a set of rules from the World Wide Web Consortium for how to describe any Internet resource such as a Web site and its content.
- *Internet Business Framework*, a group of programs that form the technological basis for the mySAP product from SAP, the German company that markets an enterprise resource management line of products.
- *Sender Policy Framework*, a defined approach and programming for making email more secure.
- *Zachman framework*, a logical structure intended to provide a comprehensive representation of an information technology enterprise that is independent of the tools and methods used in any particular IT business (Rouse, 2005). Examples of frameworks used in development today include: Zend framework for PHP, Spring framework for Java, .NET Framework (ASP.NET MVC), Django for python, Java Server Faces, Java apache cocoon etc.

2.3 Advantages and Disadvantages of Software Frameworks

Software frameworks have both advantages and disadvantage and these play major roles in software engineering alongside measuring the qualities of software. Even though frameworks help in delivering software on time due to already coded functionalities within it.

2.3.1 Advantages of framework programming

Software frameworks offer a variety of advantages as indicated by Christain, Tomasz and Jaroslaw (2010), including the following:

- Use of code which has already been built, tested and used by other programmers
- No need to learn a specific API nor use any low-level programming techniques, allowing to focus on business logic and implementation of project's guidelines
- Access to standardized, reusable codes
- Access to large library of useful data structures and algorithms in order to manipulate them
- All advantages of object-oriented programming
- Clean coding with using MVC (model, view, controller) methodology and design pattern
- Database access abstraction layer allowing to write database-independent code and change data storing system during project development with little or no code change
- For some frameworks (i.e. .NET) multi-language support or language independency
- Automated scripts to create an application basic skeleton, thus reducing starttime in new projects
- Simplified, standardized configuration separated in to specific file or database table, possibly with tools ready for manipulating configuration and thus allowing to centralize configuration of even large application in one place
- For some frameworks (i.e. -NET) and under some systems (i.e. Windows), problem with DLL libraries access

- For most web development frameworks – Hall URL management making developed webpages SEO friendly
- Object oriented access to database (for example DAO — Database Access Object), allowing users to manipulate on data without knowing of SQL or other data-manipulation language, treating database like an ordinary object
- Strong code modularity, allowing to program blocks of which final application is build and that can be later reused in other projects
- Ready out-of-the-box libraries, classes or solutions for common issues like user authorization or authentication, security features, database connectivity issues, caching
- Platform, system or browser independency
- Noticeable application performance increase with introduction of layered caching scheme or other caching techniques
- Good prepared for easy building of front-end user interface with build-in support for templates, themes and skins as good as strong focus on logic and look separation
- Very well documented with many resources available freely in the Internet, with many books published about and often strong community behind
- Very well prepared for internationalization and localization
- Event-driven programming
- Extensible with many extensions introducing developer to many standardized protocols or solutions in many development accesses like authorization, verification, user management, user interface, etc.
- User-entered and database-retrieved data validation
- Error handling and logging, good testing support

- Reuse code that has been pre-built and pre-tested. Increase the reliability of the new application and reduce the programming and testing effort, and time to market.
- A framework can help establish better programming practices and appropriate use of design patterns and new programming tools.
- A framework can provide new functionality, improved performance, or improved quality without additional programming by the framework user.
- By definition, a framework provides you with the means to extend its behaviour.

2.3.2 Disadvantages of software framework

Similarly, the use of software framework programming offers some disadvantages. Among them are the underlisted:

- Platform, system or browser dependency
- Large overhead of framework code, in some situations noticeable decreasing application performance
- Bugs and security wholes discovered in framework code can (and probably will) affect
- Every application built using it,
- For many frameworks: not suitable for development of large-scale applications or whole systems,
- Sometimes a high learning curve — i.e. often require a significant education to use efficiently and correctly,
- Need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques,

- ❖ Hard to introduce two or more frameworks in the same project, and some frameworks may not contain all necessary libraries or classes forcing developer to write on or take an attempt to introduce more than one framework in a project,
- ❖ Not supported by many popular IDE designers and in some cases (Zend Framework) framework supporting IDES are expensive programmers, forcing developer to pay for licensee to use them, (Christain, Tomasz, & Jaroslaw, 2010).
- ❖ Creating a framework is difficult and time-consuming (i.e. expensive).
- ❖ The learning curve for a new framework can be steep.
- ❖ Over time, a framework can become increasingly complex.
- ❖ Frameworks often add to the size of programs, a phenomenon termed "code bloat" (Riehle, 2000).

As it can be noticed from the above advantages and disadvantages of software framework it is clearly seen that framework advantages strongly more than that of disadvantages and this shows clearly the more reasons why frameworks are so popular and being chosen for implementation of a lot of recent or modern IT projects. (Christain, Tomasz, & Jaroslaw, 2010). Even though the disadvantages are small in number but the concern is much on the effects and seriousness of those disadvantages on software engineering since security issues and cost of license for some frameworks was mentioned as disadvantages.

2.4 Architectural and Design Patterns

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations. Object-oriented design patterns typically show relationships and interactions between classes or objects, without specifying the final application classes or objects that are involved (wikibooks). Design patterns reside in the domain of modules and interconnections. At a higher level there are architectural patterns that are larger in scope, usually describing an overall pattern followed by an entire system. There are many types of design patterns:

Structural patterns address concerns related to the high-level structure of an application being developed.

Computational patterns address concerns related to the identification of key computations.

Algorithm strategy patterns address concerns related to high level strategies that describe how to exploit application characteristic on a computation platform. *Implementation strategy patterns* address concerns related to the realization of the source code to support how the program itself is organized and the common data structures specific to parallel programming.

Execution patterns address concerns related to the support of the execution of an application, including the strategies in executing streams of tasks and building blocks to support the synchronization between tasks.

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems, and it also improves code readability for coders and architects who are familiar with the patterns. In addition to this, patterns allow developers to communicate using well-known, well understood names for software interactions. In order to achieve flexibility, design patterns usually introduce additional levels of indirection, which in some cases may complicate the resulting designs and hurt application performance. By definition, a pattern must be programmed a new into each application that uses it (Ragnarsson, 2014).

2.5 Classification and List of Patterns

Design patterns were originally grouped into the categories: creational patterns, structural patterns, and behavioral patterns, and described using the concepts of delegation, aggregation, and consultation (Gamma, 2004). In Design Patterns, an aggregate is not a design pattern but rather refers to an object such as a list, vector, or generator which provides an interface for creating iterators (Gamma, 2004). In software engineering, delegation can be thought of as passing execution to another object while retaining the identity of the calling object. Thus, if the object delegated to calls another method of its "self", that self is the original object, not the object delegated to. If, on the other hand, we hand off processing to another object, and the object acts on its own, with its own "self", and no implicit reference to the original message receiver, that is

2.5.1 Creational Patterns

❖ Abstract factory: Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

❖

❖ Builder: Separate the construction of a complex object from its representation allowing the same construction process to create various representations.

Factory method: Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

❖ Lazy initialization: Tactic of delaying the creation of an object, the calculation of a value, or some other expensive process until the first time it is needed.

❖ Multiton: Ensure a class has only named instances, and provide global point of access to them.

❖ Object pool: Avoid expensive acquisition and release of resources by recycling objects that are no longer in use. Can be considered a generalization of connection pool and thread pool patterns.

❖ Prototype: Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.

❖ Resource acquisition is initialization: Ensure that resources are properly released by tying them to the lifespan of suitable objects.

❖ Singleton: Ensure a class has only one instance, and provide a global point of access to it.

❖

2.5.2 Structural Patterns

- Adapter or Wrapper: Convert the interface of a class into another interface clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces.
- Bridge: Decouple an abstraction from its implementation allowing the two to vary independently.

  Composite: Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

  Decorator: Attach additional responsibilities to an object dynamically keeping the same interface. Decorators provide a flexible alternative to subclassing for extending functionality.
- Facade: Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
- Front Controller: Provide a unified interface to a set of interfaces in a subsystem. Front Controller defines a higher-level interface that makes the subsystem easier to use.
- Flyweight: Use sharing to support large numbers of fine-grained objects efficiently.
- Proxy: Provide a surrogate or placeholder for another object to control access to it.

❖

2.5.3 Behavioral Patterns

- ❖ Blackboard: Generalized observer, which allows multiple readers and writers. Communicates information system-wide.
- ❖ Chain of responsibility: Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.
- ❖ Command: Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

  Interpreter: Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.

- Iterator: Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
- Mediator: Define an object that encapsulates how a set of objects interact.
  Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
- Memento: Without violating encapsulation, capture and externalize an object's internal state allowing the object to be restored to this state later.
- Null object: Avoid null references by providing a default object.
- Observer or Publish/subscribe: Define a one-to-many dependency between objects where a state change in one object results with all its dependents being notified and updated automatically.
- Servant: Define common functionality for a group of classes.
- Specification: Recombinable business logic in a Boolean fashion.
- State: Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.
- Strategy: Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.
- Template method: Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.
- Visitor: Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.

2.5.4 Concurrency Patterns

- Active Object: Decouples method execution from method invocation that reside in their own thread of control. The goal is to introduce concurrency, by using asynchronous method invocation and a scheduler for handling requests.
- Balking: Only execute an action on an object when the object is in a particular state.
- Binding Properties: Combining multiple observers to force properties in different objects to be synchronized or coordinated in some way.
- Messaging pattern: The messaging design pattern (MDP) allows the interchange of information (i.e. messages) between components and applications.
- Double-checked locking: Reduce the overhead of acquiring a lock by first testing the locking criterion (the “lock hint”) in an unsafe manner; only if that succeeds does the actual lock proceed. Can be unsafe when implemented in some language/hardware combinations. It can therefore sometimes be considered an anti-pattern.
- Event-based asynchronous: Addresses problems with the Asynchronous pattern that occur in multithreaded programs.
- Guarded suspension: Manages operations that require both a lock to be acquired and a precondition to be satisfied before the operation can be executed.
- Lock: One thread puts a “lock” on a resource, preventing other threads from accessing or modifying it.
- Monitor object: An object whose methods are subject to mutual exclusion, thus preventing multiple objects from erroneously trying to use it at the same time.
- Reactor: A reactor object provides an asynchronous interface to resources that must be handled synchronously.

- Read-write lock: Allows concurrent read access to an object but requires exclusive access for write operations.
- Scheduler: Explicitly control when threads may execute single-threaded code.
- Thread pool: A number of threads are created to perform a number of tasks, which are usually organized in a queue. Typically, there are many more tasks than threads. Can be considered a special case of the object pool pattern.
- Thread-specific storage: Static or "global" memory local to a thread.

2.5.5 Data Access Patterns

Another interesting area where patterns have a wide application is the area of data access patterns.

- ORM Patterns: Domain Object Factory, Object/Relational Map, Update Factory.
- Resource Management Patterns: Resource Pool, Resource Timer, Retryer, Paging Iterator.
- Cache Patterns: Cache Accessor, Demand Cache, Primed Cache, Cache Collector, Cache Replicator. Concurrency Patterns: Transaction, Optimistic Lock, Pessimistic Lock.

2.5.6 Enterprise Patterns

If you deal with J2EE or with .Net Enterprise applications, the problems that occur and the solutions to them are similar. These solutions are the Enterprise patterns.

- Presentation Tier Patterns: Intercepting Filter, Front Controller, View Helper, Composite View, Service to Worker, Dispatcher View.
- Business Tier Patterns: Business Delegate, Value Object, Session Facade, Composite Entity, Value Object Assembler, Value List Handler, Service

Locator.

- Integration Tier Patterns: Data Access Object, Service Activator.

2.5.7 Real-Time Patterns

Finally, in the area of real-time and embedded software development a vast number of patterns have been identified.

- Architecture Patterns: Layered Pattern, Channel Architecture Pattern, Component-Based Architecture, Recursive Containment Pattern and Hierarchical Control Pattern, Microkernel Architecture Pattern, Virtual Machine Pattern.
- Concurrency Patterns: Message Queuing Pattern, Interrupt Pattern, Guarded Call Pattern, Rendezvous Pattern, Cyclic Executive Pattern, Round Robin Pattern.
- Memory Patterns: Static Allocation Pattern, Pool Allocation Pattern, Fixed Sized Buffer Pattern, Smart Pointer Pattern, Garbage Collection Pattern, Garbage Compactor Pattern.
- Resource Patterns: Critical Section Pattern, Priority Inheritance Pattern, Priority Ceiling Pattern, Simultaneous Locking Pattern, Ordered Locking Pattern.
- Distribution Patterns: Shared Memory Pattern, Remote Method Call Pattern, Observer Pattern, Data Bus Pattern, Proxy Pattern, Broker Pattern.
- Safety and Reliability Patterns: Monitor-Actuator Pattern, Sanity Check Pattern, Watchdog Pattern, Safety Executive Pattern, Protected Single Channel Pattern, Homogeneous Redundancy Pattern, Triple Modular Redundancy Pattern, Heterogeneous Redundancy Pattern (Janssen, 2021).

2.6 Software Frameworks and Software Performance

There are several factors that impact software and applications performance. The following are some of the industry's factors that impact application performance (Sanna, 2013).

2.6.1 Application Complexity

Application complexity is one of the biggest factors impacting application performance. Today's applications and services, particularly those delivered via the Web, are a mosaic of components sourced from multiple places: data center, cloud, third-party, etc. While the customer or employee looking at a browser window sees a single application, multiple moving parts must execute in the expected manner to deliver a great end-user experience. Maybe the Web server and app server are running fine, but if the database is faltering, user experience will suffer. As the saying goes, "The more moving parts, the more that can go wrong". Frameworks can add to this complexity while reducing the developers time to production. The interdependency in APIs, platforms and implementations within the frameworks directly affect application development complexity.

2.6.2 Application Design

One of the biggest factors that impacts application performance is design. Performance must be designed in. When applications are specified, performance goals need to be delineated along with the details of the environment the applications will run in. Often development is left out of this and applications are monitored, analyzed and "fixed" after they are released into production. The only way to prevent poor app performance is to expose your app development to the rigorous quality controls and processes early on in the application lifecycle—and actually fix them early in the cycle. This is a critical part in deciding the implementation phase. Frameworks make the implementation phase easy if studied well but they should be considered in the design stage and analyzed for impact on software systems performance. Architectural and design patterns can if not well implemented impact heavily on delivered applications performance.

2.6.3 Application Testing

Today's applications are often developed in components and include lots of interfaces and integrations. Plugins are assumed tested and most code is simulated without testing performance on real-world networks. Before applications are deployed, transport across today's highly distributed network architectures should be monitored and optimized. Insufficient testing of the application in the actual production environment and under varying conditions impacts performance. Developers and testers need to have a clear understanding of the non-functional performance criteria. Frameworks provide some built-in tools for testing code, APIs and plugins but strict quality controls and thorough testing must be done to achieve desired performance index.

2.6.4 The Butterfly Effect

The environmental variants need to be minimized and closely monitored to prevent the anomalous events in a software implementation successful—it's the choices you make in how you put them together to support the multiple environments within IT. Frameworks are built for specific families of applications and thus may improve performance.

2.6.5 The Infrastructure and Components of the Application Service

Application performance is impacted by components used to deliver the service to the user-frameworks and design patterns included, the user's interfaces with the application, and the connectivity between these components. The variance and complexity is what makes the problem hard to solve, and often causes approaches to fail on given architectures. One of the most critical factors that affect application performance, and often the hardest to identify and track, are application dependencies—on supporting applications, as well as the underlying system and network components that connect them all together. With the advent of virtualized servers and networks, the complexity of the application delivery infrastructure has increased significantly, and so the challenge is finding an application performance monitoring solution that can automatically discover and monitor the network and server topologies for the entire application service.

2.6.6 The Dynamic IT Environment: Virtualization and the Cloud

Applications today are an intricate mesh of multi-tier software running on servers, networks, and storage. In addition, there is a good chance they are running on virtualized hardware that is shared with other applications. It is very challenging in this dynamic environment to understand what will impact your application performance as it requires intimate knowledge of your ever-changing application structure at any given moment. Many IT organizations are very advanced on the application side but unfortunately still struggle to move beyond managing applications via a silo approach to the different technology tiers—application, server, network, storage, etc.

2.6.7 Mobility

One of the biggest factors we see is the acceleration of mobility and IT consumerization, which will propel the ongoing shift in application architectures required to deliver the most dynamic, modern mobile end user interfaces. Software frameworks are to be designed based on either elegance or just a problem solver. Software elegance implies clarity, conciseness, and little waste. For example, "elegance" to a code generating framework would imply the creation of code that is clean and comprehensible to a reasonably knowledgeable programmer (and which is therefore readily modifiable), versus one that merely generates correct code (Sanna, 2013).

2.7 Analysis and Suggestions

There is software in every applicable area of real world. As software systems have grown in complexity ranging from needs, integrations, collaborations, globalizations and security, software engineering has to meet the same. This can be achieved by all software construction methodologies but it's easier and cheaper to deliver using frameworks and design patterns. Due to the complexity of their APIs, the intended reduction in overall development time may not be achieved due to the need to spend additional time learning to use the framework; this criticism is clearly valid when a special or new framework is first encountered by developers. If such a framework is not used in subsequent tasks, the time invested in learning the framework can cost more than purpose-written code familiar to the project's staff; many programmers keep copies of useful boilerplate for common needs. Thus, frameworks are useful if their learning is re-used. As a framework is an application that is complete except for the actual functionality, you plug in the functionality and you have an application, they are very useful to developers. Software systems architects and designers can use helps guide the process and even produce designs already used. Frameworks though add to the size of programs, a phenomenon termed "code bloat". Due to customer demand driven applications needs, both competing and complementary frameworks sometimes end up in a product. Frameworks if incorrectly applied could lead to loss of performance . due to cross reference and non-useful calls to APIs functions. Consider, say, a GUI framework. The framework contains everything you need to make an application. Indeed, you can often trivially make a minimal application with very few lines of source that does absolutely nothing-but it does give you window management, sub-window management, menus, button bars, etc. That's the framework side of things. By adding your application functionality and "plugging it in" to the right places in the framework you turn this empty app that does nothing more than window management, etc. into a real, fullblown application. This is the core use of frameworks kind of provide an environment with skeleton for meat to be put on. There are similar types of frameworks for web apps, for server-side apps, etc. In each case the framework provides the bulk of the tedious, repetitive code (hopefully) while you provide the actual problem domain functionality. This is the ideal. In reality, of course, the success of the framework is highly variable and will be growing rapidly as we further the component-based development, iterative and incremental object-oriented development techniques. Why not traditional libraries then? When you invoke a traditional library, you are still in control: you make the library calls that you want to make, and deal with the consequences. A framework inverts the flow of control: you hand over to it, and wait for it to invoke the various call-back functions that you provide. You put your program's life in its hands. That has consequences: one of the most important ones is that, while your program can use as many libraries as it likes, it can only use—or, rather, be used by—one framework. Frameworks are jealous. They don't share. But some frameworks allow importing plugins or libraries into their repository as may be required. A design pattern is a description or template for how to solve a problem that can be used in many different situations. Design patterns provide the much-needed reuse of a solution. It doesn't provide any code that can be used in application development. There is such a brighter future for design patterns especially as software gets more standardized and as environments evolve into an ecosystem of frameworks, designs and components. The key is to build frameworks that can generate only useful code based on user needs. Elegant frameworks can be useful tools in ensuring great performance. A framework can help establish better programming practices and appropriate use of design patterns and new programming tools. A framework can provide enhanced functionality, improved performance, or improved quality without additional programming by the framework user especially extensible frameworks.

2.8 Technology Acceptance Model Theory

There have been many theories that are relevant about growth of software framework usability and its effect on software engineering. The study linked to Technology Acceptance Model (TAM). According to (Davis, Bagozzi, & Warshaw, 1989) Technology Acceptance Model was developed to predict the likelihood of an individual to accept new technology in a workplace environment. The model talks about the fact that users will make an adoption decision based on the outcome of their evaluation of using the technology (Perceived Ease of Use), and they have a belief that using the new technology will increase their job or coding performance (Perceived Usefulness).

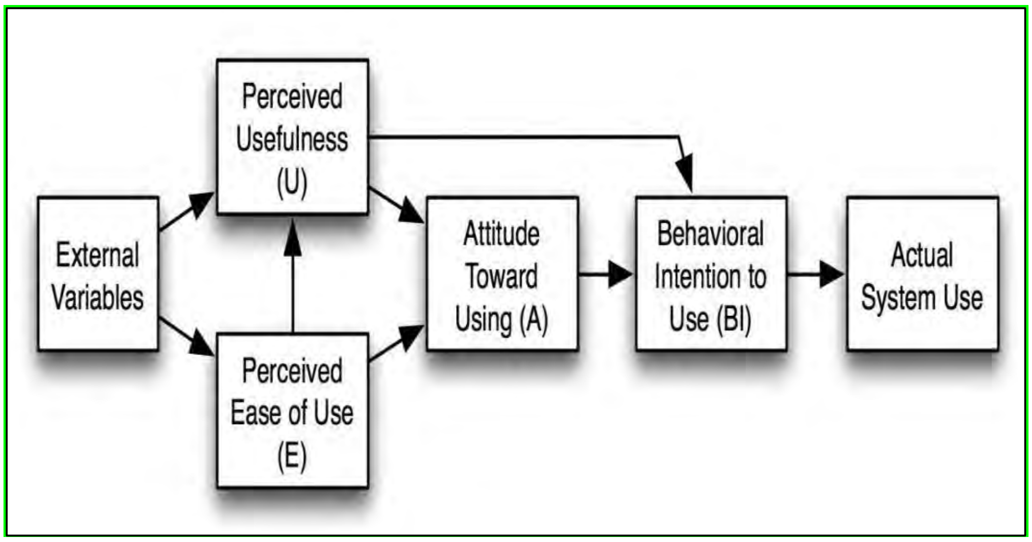
Perceived Usefulness (U)
External Variables
Attitude Toward Using (A)
Behavioral Intention to Use (BI)
Actual System Use
Perceived Ease of Use (E)

Figure 1: Technology Acceptance Model

Source: https://www.researchgate.net/figure/Technology-Acceptance-Model-

TAM_fig1_278676371

2.8.1 Implication of the Theory to the Study

Using the Technology Acceptance Model, this study was placed within a context that made it possible to explore the contextual factors that relate to growth of software framework usability and its effect on software engineering. The model supported the indication that the adoption of a particular software framework is underpinned by key variables – perceived usefulness, perceived ease of use, and attitude – which are also influenced by other external variables. The combination of these factors thereby results in an individual's behavioral intention to adopt a particular software framework, which in turn results in actual use. This understanding informed the direction of the study by ensuring that the various factors that relate to a particular software framework adoption were not considered as independent entities, but rather as components of a system that combine to predict how individuals decide to adopt software framework. Furthermore, the theory provided a framework for the discussion of the findings and recommendations on how decisions are made about software framework adoption. This was expressed by the proposition that, at all times, the decision to implement policies on a particular software framework adoption for a particular project should be based on several factors which relate to ensure smooth adoption.

2.9 Contextual Factors that Relate to ICT tools Adoption by Teachers, Developers and Institutions

There are several factors the relate to the adoption of software framework for a particular project or as a tool for software development by teachers, developers and institutions, according to TAM. Among these are the attitudes individuals have towards a particular software framework, the perception of how easy it is to learn and use the software framework (Perceived Ease of Use), and the perception of the usefulness of the software framework (Perceived Usefulness). It is important to explore the interdependence of the factors, and how they relate to the adoption and use of software frameworks and its effects on software engineering.

2.10 Classification of Frameworks

According to Mary and Rodrigues (2012), classification is the problem of identifying which of a set of categories a new observation belongs to. This is a problem, because as frameworks are developed under the interests of different developers, they individual perspectives influence their classification. Software frameworks are therefore classified as being developed by standard bodies or individual interests or by private agencies. Frameworks developed by standard bodies fall under the standard category and others fall under the nonstandard category. These categories are subcategorized into smaller groupings based on their usage for commercial or government purposes. Software framework classification in critical because it invokes the user to choose the right framework for their industry, organization and business based on their requirement (Mary & Rodrigues, 2020). The classification also helps users to easily identify the supporting tools available for their frameworks.

. It is important to note that developers with little experience in the use of traditional programming languages encounter a number of challenges when they start software development with software frameworks. However, comparing the use of traditional programming techniques to the use of software frameworks, it is evident that software framework usage is much easier, faster and more comfortable to use. Also, problems are seamlessly solved because a lot of syntax errors are corrected by the software frameworks. Christian, Tomasz, and Jaroslaw (2010) noted that some novice programmers admit that they are starting to learn an art of software development as compared to the traditional programming.

According to Christain, Tomasz, & Jaroslaw (2010), even though there are more powerful software frameworks that can perform a lot of jobs for programmers, programmers are to note that there are advantages and disadvantages for using such software framework. Also, even though they can be treated as fast, modern and agile software development technology, they should not be understood as a kind solution for every problem they may come across and also should not be considered as the best or the only way to implement any project.

2.11 Factors that Promote the Adoption of Software Frameworks among Application Developers

Although there are strong disadvantages for the use of software frameworks, there still exist some factors that support the usage of such software frameworks. In a study by . Varvana, Sari, Raatikainen, and Pile (2018), which focused on the Qt Software framework, respondents were practioners from companies using Qt framewoork and some students who use the Qt framework. The study found out the following factors to be supporting the usage of software frameworks.

1. API capabilities are the key justification when developers select the framework: they want to make sure the API is cross-platform development and also for embedded software development.
2. When developers adopt a framework, the need to learn a new programming language adds to the initial learning curve: because some software frameworks demand additional learning before developers can use them this actually prevent a lot of them from using such software frameworks as it was noticed that Qt frameworks demands additional learning curve.
3. Specializing APIs to serve different kinds of application developers and purposes improves productivity, provided that the specialized APIs do not lack the necessary features.
4. Having the framework implementation available as open-source code improves the visibility and continuity of the framework: a framework that have it source codes available for developers make it easy for them to debug their software whenever there is internal error and this gives them more power whenever they need to edit something within the framework source code.
5. If the development tools are not easy to install, or it is difficult to initialize the first project, adoption is hindered: frameworks that have difficulties in installing always hinder developers from using it.
6. If the installed development tools include unnecessary libraries, adoption is hindered: at times software frameworks are too heavy after installations hence creating much load on the program there by reducing its' speed due to unnecessary libraries that have been installed unintentionally during the installations process.

7. If the development tools do not support the deployment of applications, initial use is hindered: Software frameworks that are not complete and needs additional third-party system or software for deployment also hinder developers from using it.
8. Good editor capabilities and documentation embedded in the editor improve development efficiency: if a particular software framework editor is not user friendly enough to correct some basic syntax error or to alert error at the initial stage for editing but always pop-up during run-time of the program such frameworks are always not used by developers.
9. When selecting the framework, personal recommendation and peer experiences are valued highly: developers want to use framework that is mostly known to their friends so that they may find it easy in terms of difficulties in debugging.
10. When selecting the framework, active and long-lived community is valued.
11. If there is insufficient information on getting started, adoption is hindered.
12. If the information is not targeted at and organized to meet developers' needs, the use of the framework becomes less satisfactory.
13. Realistic enough code examples and how-to videos give practical, hands-on guidance in using the framework; developers believe in real codes that are working as an example in order to have much trust in the particular framework.
14. Peer help, either online or face-to-face, is valuable for solving problems in using the framework.

2.12 Kinds of Frameworks

Fundamentally, there are two kinds of software frameworks. These are full-stack frameworks, and Glue frameworks.

- A full-stack framework provides a single set of components, covering the entire spectrum of features needed to build the application or offers an all-in-one solution including URI Routing, Caching, Hooks and many more features for the development of a new system. This mean with the Full-Stack Framework the developer is going to go by their rules in making the development but not his own, this framework is good for security and rapid application development just that it does not give endless flexibility. They ensure that the components work well with one another. Examples are Android, Django, CakePHP Symfony and Zend frameworks.
- A Glue framework provides a set of adapters and interface code that gives you total control of your codes and allow you to implement their classes, libraries and helpers in the way you want it to be. The idea is not to tie you down and make your code conform to their standards by force but to give you the flexibility to make your own suggestion. They are supposed to handle many different components at any part of the software stack. They ensure that the adapters will work well with any possible combinations of components. Examples are Codeigniter, FLASK, GFAC, TurboGears, Twisted and Pylons.

According to Pree (1994), software frameworks consist of frozen spots and hot spots. Frozen spots define the overall architecture of a software system, that is to say its basic components and the relationships between them. These remain unchanged (frozen) in any instantiation of the application framework. Hot spots represent those parts where the programmers using the framework add their own code to add the functionality specific to their own project.

2.13 Types of Frameworks

Frameworks can be used to create most applications on the back end, including web services, web applications, and software.

2.13.1 Desktop Software frameworks

A software framework is a reusable environment that's part of a larger software platform. They're specifically geared toward facilitating the development of software applications and include components, such as libraries of code, support programs, compilers, tool sets, and specific APIs that facilitate the flow of data.

2.13.2 Web application frameworks

These are software frameworks used to streamline web app and website development, web services, and web resources. A popular type of web app framework is the ModelView Controller (MVC) architecture, named for the way it separates the code for each application component into modules. (Upworks, 2018). Table 1 below shows some of the most popular frameworks, broken down by the programming languages in which they are written.

Table 1: Popular Software frameworks

Programming Language	**Framework**
Python	Django framework, Flask, Pyramid, Tornado, Bottle
Ruby	Ruby on Rails, Sinatra
PHP	CodeIgniter, Zend framework, CakePHP, FuelPHP, Laravel framework, Drupal, Joomla
Perl5	Catalys, Symfony, Interchnage, Maypole
JavaScript	AngularJs, jQuery, EmberJS, Node.JS, Backbone.js, MeteorJS, ExpressJS, Koa.js, ReactJS
Java	Apache Click, Grails, Oracle ADF, Java Web Services
Cold Fusion Markup Language	Wirebox, Fusebox, Mach II
CSS	Pure.css, LESS & Sass
C	Saetta Web Server
C++	Boost, Platinum
C#	VB.NET, ASP.NET
Objective-C & Swift	Cocoa, Apple's API
Mobile frameworks	Bootstrap, Sencha Touch, Cocoa + Cocoa Touch, jQuery Mobile + Backbone.js, Kendo, AngularJS + Ionic, React Native

2.14 Architectural Problem of Software Frameworks

Architectural problems occur as a result of the addition of unintended design decisions in software development that violate either the original, intended architecture of a system or the general software modularity principles (Perry & Wolf, 1992). According to Hochstein and Lindvall (2005), architectural problems have caused the discontinuation or reengineering of several software projects. Frank, Regine, Hans, Peter, and Michael, (2001) stated that patterns help in building on the collective experience of skilled software engineers by capturing existing well-proven experience in software development and help to promote good design practice.

Framework revolution also result in a lot of issues if the framework is not well structed .and tested. According to Mattsson (2000), Object-oriented frameworks, as other software, evolve over time. Controlled and predictable evolution of framework is of importance since the reusable framework not only evolves as a framework but that it only causes the evolution of the applications developed using the framework. To reduce the costs of updating the applications developed and the cost of evolving the framework, controlled and predictable evolution of the framework is necessary. Four major kinds of framework evolution are usually caused by a new or changed requirements in the software development process. They include:

1. **Internal reorganization:** Internal reorganization led to Hot-spot introduction, a place where framework extension takes place. It also results in Restructuring, mostly for better maintainability of the framework, and Changing control flow, where only the internal behavior of the framework is affected but not the

structure Also, internal reorganization affects Refining framework concepts, where the new requirement may point out flaws in the previous one.

2. **Changing functionality**: Changes in the functionality of a framework may be initiated as a result of new underlying hardware and the way the framework will interact with the hardware components. This change sometimes leads to changes in internal class behavior.
3. **Extending functionality**: Extending the functionality of a framework may lead to functionality introduction, where new requirements demand the extension of the framework with new functionality. If the requirement only affects one framework, the situation is similar to the traditional monolithic framework development process, which implies iterating the framework design for achieving a new framework version. Also, there is concept introduction, where new concepts in the framework are introduced. This introduction causes the introduction of new functionality. New concepts require the reorganization of the framework, since new concepts have relations to the existing concepts.
4. **Reducing functionality**: In some instances, the functionality that was developed as an integral part of the framework has to be removed from the framework. Such action may lead to requirements that require the introduction of an enable/disable mechanism in the framework. Also, if new requirements require the use of an additional framework that has to collaborate with the existing framework component, conflicts due to of domain overlap or implicit assumptions about ownership of the thread of control may occur. This will make it necessary to cut functionality out of the existing framework since it may not be possible to make necessary changes in the new framework, for instance, due to lack of access to source code or the complexity of the framework.

2.14.1 Other problems and experiences

According to Mattsson (2000), Problems and experiences were categorized into three phases, which are framework development, framework usage and framework evolution. The development of a framework is somewhat different from the development of a standard application. The important distinction is that the framework has to cover all relevant concepts in a domain, whereas an application is concerned only with those concepts mentioned in the application requirements.

Framework usage also poses another problem for developers who have no prior knowledge in any of the programming languages, and want to use it from scratch. Nonetheless, there are some frameworks with drag and drop features which make it easy, whiles others do not support drag and drop, meaning the developer has to learn . all the classes and flow of the framework before they can make good use of it.

Again, framework evolution is another problem that affects product that have been developed by older versions of a particular software framework that is currently having a newer version for installations since this can lead to a lot of problems. Once the framework has been deployed, new versions of a framework cause high maintenance cost for the products built with the framework (Mattsson, 2000).

2.15 The Future of Software Framework

Wayner (2018) explain the seven reasons why frameworks are the new programming languages by relating the traditional way of coding to the current trend of coding based on the following:

1. Most coding is stringing together APIs: According to Wayner (2018), there is no need to learn all then pointers and syntax of a particular programming

langauge but rather getting more knowledge about how to use the API of a particular selected software framework in order to know when and where to apply or use them really matter most. APIs are already coded in software frameworks to function exactly the same as a programmer coding them from scratch or using traditional programming. The most important thing that matters now is no more about the syntax but understanding the API is what matters.

2. The shoulders of giant are worth standing on: Just imagine creating something new, the time it will take you to finish it depending on the complexity of the system and your level of knowledge in the selected programming language. Sometimes it will take you a lot of time testing newly created codes to be sure that they are working the way you intended for it to work or the safety aspect of it. But software frameworks are tested codes that have been reviewed and tested by many programmers and have been proven to be more secured for complex systems. Standing on th shoulders of other advnaced programmers help you to speedup your software development and makes coding easy for novice of programmers.
3. Kowing the achitecture matters is what matters not the syntax: Wayner (2018) stated that "Getting the details of the languagecorrect can help but knowing what's going on in the libraries can pay off dramtically". What actually matters is knowing the flow and the achitecture of the framework but not the syntax of the programming language.
4. Algorithm dominate: Already ther are tested and most frequently used algorithms for many problems a programmer will like to solve hence software frameworks have captured all this algorithms for easy implememtation in systems hence there is no need knowing how to implement a particular

algorithmin a particular progrmming language. APIs can be called from the framework libraries to implement such algorithms in a more perfect way.

5. Compiler and smart IDEs correct your syntax: Assumming you are writing a code in a pariticular language like Java that needs to terminate each line with a semicolon and you forget about it in a traditional programming way, it likey for the program not to run at all or prompt error at the running-time of the program but with the current compilers for software frameworks it is highly possible for such mistakes to be corrected by the compiler or the IDE without the prrogrammer being aware of such errors.
6. Syntax is disappering with visual languages: traditional programming involves writing and editing of thousands lines of codes before running or compiling . codes to view the results of such codes but with the recent frameworks annd their editors there is drag and drop functionalities that enable widgets to be use in developing user interface in a more advance way where the developer determins where to display them without writing any lines of codes and is also able to view the results whiles building it with or without running the codes.
7. Code is law: Wayner (2018) stated that "Computer languages are largely agnostic. They're designed to be open, accepting, and almost infinitely malleable. They're meant to do whatever you want. Sure, sometimes you need to use a few extra characters because of the syntax, but those are merely keystrokes. After that, it's mainly if-then-else, plus occasional clever bits. All of the language will still help you get the results you want the way you want to get them. If there are strictures, they're designed to keep your code as bug-free as possible, not limit what you can do." Frameworks are where the power lies. This is where architects can decide what is allowed and what is inherently forbidden.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter focuses on the methodology that deal with methods and procedures which were used in guiding the research under study. This chapter include research design, population, sampling strategies, data collection methods and instruments, data collection procedure and analysis techniques and ethical consideration.

3.2 Research Design

Research design occupies a key position in the research work. It is based on the research purpose, like exploration, description, diagnosis and experimentation. Kerlinger (1986) refers to a research design as a plan, structure and strategy of investigation so conceived as to obtain answers to research questions and to control variance. Also, Hassan (1995) views Research design as a blue print of activities or specification of procedures and strategies to follow so as to obtain the most value answers to research question or attain the objectives of study with optimal control of variables. The approach used in this study is quantitative in nature. The descriptive survey design was taken for the study. The descriptive survey research design entails a critical observation of events, objects, subjects and ideas without attempt to control the condition of such phenomenal. It is a description of a given state of affairs that exist at a particular time which required a direct contact with individual whose characteristic, behaviors and attitudes are relevant to the investigation (Jongbo, 2014). This research design was used because it provided information useful to the solution of the problems. It employs application of scientific method by critically analysing and examining the source materials, by analysing and interpreting data, and arriving at generalisation and prediction (Salaria, 2012). The advantage of descriptive research is that of allowing for the research to be conducted in the natural environment of the respondent and this ensures that high-quality and honest data is collected. And the disadvantage is that of respondents aren't always truthful if questions are too personal or they feel that they are being "watched". This may negate the validity of the data (Salaria, 2012).

3.3 Population

Population refers to the complete set of individuals (subjects), objects or events having common observable characteristics in which the researcher is interested in studying (Agyedu, Donkor, & Obeng, 2010). The total population for the research was 277 and was selected from different areas. The selection was based on those that have more or little knowledge in software development, hence not all the students were selected from a particular class. The targeted population for this study covers the final years computer science or information technology masters' students in five public universities (UEW50, UDS-40, Kumasi Technical University-32, KNUST – 40 and UNER - 50) in Ghana and two software companies (Gracecoms – programmers 35 and Kologsoft – programmers 30).

3.4 Sample and Sampling Technique

Saunders, Lewis, & Thornhill (2012) define sample population as a group of subjects identified from a larger population as a representation of the entire group. According to Creswell (2012) the sampling method adopted helps in identifying safe generalizations that can be deduced from a larger target population. The total population of the sample

SHSs totaling 277 masters' students consist of 194 males and 83 females**.** These masters Students were selected randomly by the used of simple random sampling method. The following formula was used to compute the sample size:

$$n = \frac{N}{[1 + N(e)^2}$$

Where n represents required sample size, N indicates sample frame, α represents significance level or error margin. For the sample size to be fairly represented, the sample size is determined at a 98% confidence level (at a 0.02 significance level) for the study.

$$\frac{277}{[1 + 277\ (0.02)^2]}$$

$$n = \frac{277}{[1 + 277\ * 0.0004]}$$

$$n = \frac{277}{1.1108}$$

$$n = 249.369\ \approx\ 249$$

The simple random sampling method was used to randomly pick 249 masters students and software developers from the population for the data collection.

3.5 Data Collection Instrument

The instruments used to collect data for the study were questionnaires, and observations guide, respectively.

3.5.1 Questionnaires

The instrument used to collect data for the study was questionnaires. A questionnaire is defined as a document containing questions and other types of items designed to solicit information appropriate to analysis (Babbie, 1990). Questionnaire is equally used in survey research, experiments and other modes of observation. Indeed, people ask different questions in their daily life to satisfy their queries. The two most common types of questionnaires are close-ended questions and open-ended questions (Creswell & Plano, 2007). It was chosen because of the nature of this study to get the opinions and views of the respondents. Respondents replied to them on their own free will without any influence from another person; they were easy to be administered within a short time and from the relatively larger groups of people who were scattered geographically. Questionnaires were prepared and administered in the five universities and two software companies by the researcher. The questionnaires for the respondents were sectioned into three segments; the first part solicited on the social-demographic characteristics of respondents. The second part talked about the type of software framework they use in software development and third group indicated the causes of rapid growth of software framework useability and the final part solicits the effects of software framework useability to software engineering.

3.5.2 Observation

I made observation about the type of software framework that is mostly used and the end product of those systems being produced. I used observation method because I wanted to know the type of product that are being produce after using software framework. Observation was focused on the end product of software frameworks.

3.6 Validity of Instrument

Content validity is the measure of the degree to which data collected using a particular instrument represents a specific domain of indicators of a particular concept (Muganda & Mugenda 2000). Validity was conducted to ensure that the degree to which result obtained from the analysis of the data actually represents the phenomenon Robinson (2002). Validity of an instrument is demonstrated when an instrument is seen to be asking the right question framed in the least ambiguous way the face validity of the questionnaire was established with the help of experts. These experts helped to correct any elements of ambiguity in the instruments before it was used in the pilot test. They deemed it suitable for gathering information on respondents view on growth of software framework usability and its effects on software engineering. Content validity of the instrument was determined with the help of the researcher's supervisor who was an expert in the field of Research. After the examination of the instruments by the supervisors and other research experts' changes were affected as a result of comments and suggestions from them. These changes were in the form of the deletion of incorrect items, addition of new items and modification of existing ones. This helped to improve the content validity of the instrument, because their collective judgments were used to establish congruence between all of them.

3.7 Reliability of Instrument

Reliability refers to a measure of the degree to which research instruments yield consistent results (Mugenda & Mugenda, 2003). The Statistical Package for Social Sciences (SPSS) was used to determine the Cronbach alpha coefficient value for the questionnaire, which was found to be 0.885. According to Leech, Barrett and Morgan (2005), Cronbach alpha coefficient value of 0.70 and above indicates a reasonable internal consistency and that alpha value between 0.60 and 0.69 indicate minimal adequate reliability. According to Ary, Jacob and Razavieh (2002), where results are used to make decisions about a group, reliability coefficient of 0.50 to 0.60 is acceptable. The questionnaire items were therefore reliable as the Cronbach alpha coefficient value was above 0.70.

3.8 Data Collection Procedures

The students were given the questionnaires to fill on their own. This practice was found to be consistent with Kumekpor (2002) who said that self- administered questionnaires are good for respondents whose literacy rates and educational levels are high and can . complete the questionnaire on their own without any special assistance from others. Filling the questionnaire in front of the researcher aided him to address all questions that respondents raised in the course of completing the questionnaire. The researcher also ensure that all questions were answered correctly and orderly. The anonymity of questionnaires encouraged the students to tick choices of their answers and fill in. The questions produced their response, which help in the overall success of the study. The 249 students and developers responded and the entire questionnaires were retrieved and compiled. The questions are attached in Appendix A of this piece of work, respectively.

3.9 Data Analysis

Data quantitatively gathered from the questionnaires were coded, scrutinized and analyzed with statistical tool called Statistical Packages for Social Sciences (SPSS) version 25 application. In this study, both descriptive and inferential statistics were used. Descriptive statistics was used to analyze the demographics characteristics of the students. In other for the researcher to examine the effects of software framework usability to software engineering, regression analyses were used as an effective inferential statistical tool. Tables were also used to present the descriptive aspects of the study to enable the researcher achieved the stated objectives.

3.10 Ethical Considerations

Research ethics plays an important part in forming a research design. Research ethics involves in the planning of research, requesting access to organizations or individuals and while reporting the data as well (Saunders, Lewis, & Thornhill, 2012). I explained the research purpose to respondents' and I asked for their consent before involving them in the research. I involve the respondents by briefing them about the research topic, objectives and roles of the respondents and how they would benefit from the research study. The researcher informed participants about what their participation in the research entailed, the requirements of the study and its importance so as to get their consent before proceeding with data collection. Informed consent is central in social research and it is up to the participants to weigh the benefits and risks associated with participating in the research and deciding whether to take part or not (Burgess, 2016). By explaining to the respondents, the purpose of the study, the researcher did not force them to participate in any way but allowed individuals to decide whether or not to participate in the study. In this study, I assured the respondents high degree of confidentiality in matters pertaining their privacy and sensitive information that will be provided in the cause of the research. This was done by ensuring that the principles governing research participants were followed. Great care was taken to assure respondents that all information was treated with a lot of confidentiality. The researcher informed the respondents that no information was shared to third party. Also, their information was not identified and was used for research purposes only. Finally, the researcher made sure that there was no plagiarism in the study by acknowledging other people's work. The findings have been reported as per the respondents' answers and not otherwise.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

This part presents the analysis of the data gathered from respondents from selected universities in Ghana. This chapter also sectioned into three parts. The first section deals with the descriptive analysis of the socio-demographic background characteristics of respondents and the second part presents results and discussion based on the selected objectives.

4.2 Response Rate

In the study, all the 249 questionnaires sent to the field were returned, resulting in 100% returned rate. "As a rule of thumb, a 30% return [of the questionnaire survey] has to be seen as fairly satisfactory and more than 50% is good" (Abdul Samad, 2005).

4.3 Background Information of Respondents

This sector indicated the gender, age, years of experience as well as a student or software developer. The background of respondents was very necessary to enable the researcher describe the peculiar characteristics of the respondents.

4.4 Socio-Demographic Background Characteristics of Respondents

In order to investigate on the growth of Software Framework Usability and its effects on software engineering, demographic background characteristics were conducted. This includes the gender, age, years of experience as well as a student or software developer of respondents as indicated in Table 2 below.

Table 2: Demographics characteristics of the students and software developers

Demographics	Demographics	Freq	Percentage
Gender	Male	174	69.9
	Female	75	30.1
	Total	249	100
Age	18- 25 years	95	38.2
	26- 30 years	105	42.2
	More than 31 years	49	19.7%
	Total	249	100
Are you a	Student	174	69.8
	Software Developer	75	30.2
	Total	249	100
How many years of experience do you have in your Software Framework Usability	1- 5 years	199	79.9
	6- 10 years	27	10.8
	11-15 years	23	9.2
	More than 15 years	0	0.0
	Total	249	100.

Source: Field work, 2019

The result as shown in table 2 illustrate that 69.9% (n=174) of the respondent are male while 30.1% (n=75) of the respondent are female. It can be concluded that majority of the students and software developers used in the study were male. Again, the table also shows that 38.2% (n=95) of the respondents are within 18- 25 years, 42.2% (n=105) of the respondents are within 26- 30 years, 19.7% (n=49) of the respondents are within 31 and above. It can be concluded that majority of the students and software developers used in the study were between the ages of 26 years to 30years.The results also proved that 100% of the respondents were matured and therefore could be taken in an academic sphere such as this. Again, the result also indicates that 69.8% (n=174) of the respondent were students, 30.2% (n=75) of the respondents were software developers. From the above table 1, it can be concluded that majority of the respondents used in the study were students. Again, with regard to respondents' experience on software framework usability, the result also indicates that 79.9% (n=199) of the respondents had 1-5 years' experience, 10.8% (n=27) of the respondents had 6- 10 years' experience, 9.2% (n=23) of the respondents had 11- 15 years' experience while 0.0% (n=0.0) of the respondents are more than 15 years' experience. An inference from the above table indicates that mainstream of the respondents with high level of experience are 1- 5 years category.

4.5 Types of Software Framework respondents use in Software Development

The results as shown in table 2 indicates frequencies and percentages of students' and software developers with various types of software framework that they use in software development. It indicates that 4.4% (n=11) of the respondents use Django for software development, 2.0% (n=5) use Laravel framework for software development, 0.8%

(n=2) use mobirise for software development, 6.8% (n=17) use JQuery for software development, 2.8% (n=7) use VB.NET for software development, 10.8% (n=27) use ASP.NET for software development, 32.1% (n=80) use Bootstrap for software development, 26.1% (n=65) use Flutter for software development and 14.1% (n=35) use AngularJS for software development. It can be concluded that more students and software developers use bootstrap and flutter for software development.

Table 3: The types of software framework respondents use in software development

		Responses	
	Frameworks	**N**	**Percent**
TYPES OF SOFTWARE FRAMEWORKS	Django	11	4.4%
	Laravel Framework	5	2.0%
	Mobirise	2	0.8%
	JQuery	17	6.8%
	VB,NET	7	2.8%
	ASP.NET	27	10.8%
	Bootstrap	80	32.1%
	Flutter	65	26.1%
	AngularJS	35	14.1%
Total		**249**	**100.0%**

Source: Field Work, 2019

And in trying to find out the level of growth of software framework usability in software engineering, the respondents who are students and software developers were asked to rate the level of growth of software framework usability in software engineering. The following represents the responses gathered.

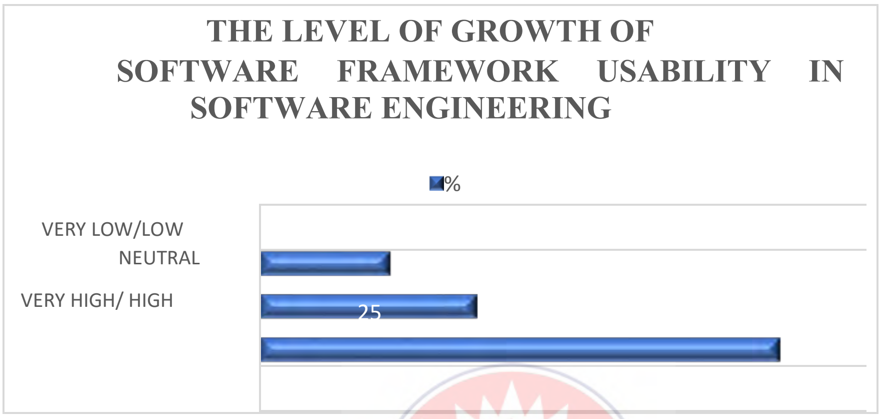

Figure 2: Level of growth of software framework usability in software engineering The results as shown in figure 2 indicates bars of students' and software developers with various rate level of growth of software framework usability in software engineering. It shows that 60% (n=149) of the respondents indicated that there is very high and high level of growth of software framework usability in software engineering,

.25% (n=62) of the respondents were not certain with software usability while 15% (n=38) indicated that there is very low and low level of growth of software framework usability in software engineering.

Overall, one can conclude from the results that there is very high/ high 60% (n= 149) level of growth of software framework usability in software engineering. This is line with Varvana M., Sari, Mikko, & Piia (2018) proposition that Software frameworks are nowadays extensively used to develop different kinds of software applications efficiently. And the popular software frameworks use was bootstrap 32.1% (n=80) and flutter 26.1% (n=65).

4.6 The Causes of Rapid Growth of Software Framework Useability in Software Development

In order to pinpoint the causes of rapid growth of software framework useability in software engineering, seven imaginable questions were raised by the researcher to solicit information using Likert scale method as presented in table 4.

Table 4: The causes of rapid growth of software Framework useability in Software Engineering

CAUSES	SD/D		N		SA/A				
	N	%	N	%	N	%	Mean	Std D	Rank
1. Reduces Code Length	32	7.5	32	7.5	185	43.2	2.614	.704	3
2. Bootstraps the development process	7	1.6	17	4.0	225	52.6	2.876	.407	1
3. Reinforces Security	12	2.8	174	40.7	63	14.7	2.205	.510	5
4. Framework system are more efficient than pure coded systems	192	44.9	45	10.5	12	2.8	1.277	.546	6
5. It is faster to develop a system with framework than pure coding	25	5.8	30	7.0	194	45.3	2.679	.649	2
6. Improves database proficiency	12	2.8	144	33.6	93	21.7	2.325	.563	4
7. Eases debugging and application maintenance	224	52.3	25	5.8	0	0.0	1.100	.301	7
AVERAGE	**72**	**29%**	**67**	**27%**	**110**	**44%**	**2**	**1**	

Key: SD = Strongly Disagree, D = Disagree, N= Neutral, A = Agree, SA = Strongly Agree

Source: Field Work, 2019

The results shown in table 4 indicates that on Cause 1: Reduces Code Length, 43.2% (n = 185) of the respondents were both strongly agree/ agree with the statement, 7.5% (n = 32) of students or software developer uncertain with the issue at hand while 7.5% (n = 32) of the respondents were both strongly disagree/disagree to the issue of reduces code length. On Cause 2: Bootstraps the development process, 52.6% (n = 225) of the respondents strongly agree/ agree with the statement, 12.5% (n = 45) respondents were undefined with the issue while 37.7% (n = 136) of the respondents were both strongly disagree/disagree to the issue of bootstraps the development process. On Cause 3: Reinforces Security, 14.7% (n = 63) of the respondents strongly agree/ agree with the statement, 40.7% (n = 174) respondents were neutral to the statement while 2.8% (n = 12) of the respondents were both strongly disagree/disagree to the issue of reinforces security. On Cause 4: framework system is more efficient than pure coded systems, 2.8% (n = 12) of the respondents were both strongly agree/ agree with the statement, 10.5% (n = 45) respondents uncertain with the issue at hand while 44.9% (n = 192) 152 of the respondents were both strongly disagree/disagree to the issue. On Cause 5: It is faster to develop a system with framework than pure coding, 45.3% (n = 194) of the respondents strongly agree/ agree with the statement, 7.0% (n = 30) students were neutral while 5.8% (n = 25) of the respondents were both strongly disagree/disagree to the issue. On Cause 6: Improves database proficiency, 21.7% (n = 93) of the respondents strongly agree/agree with the statement, 33.6% (n = 144) respondents were neutral while 2.8% (n = 12) of the respondents were both strongly disagree/disagree to the issue. On Cause 7: Eases debugging and application maintenance, 0.0% (n = 0) of the respondents strongly agree/ agree with the statement, 5.8% (n = 25) respondents were neutral while 52.3% (n = 224) of the respondents were both strongly disagree/disagree to the issue of eases debugging and application maintenance.

Overall, we understood that widely held of the students and software developers 44% (n = 110) were agreement with acknowledged causes and minority of them 29% (n = 72) were in disagreement with the identified causes students and software developers identified as causes of rapid growth software Framework useability in Software Engineering. Again, the mean scores in the table 3 suggest that the most leading causes students and software developer identified are cause 2*:* Bootstraps the development process and cause *5:* It is faster to develop a system with framework than pure coding with the highest mean score of (means = 2.87, 2.67) with standard deviation of (std = *0.47*, 0.64***)*** respectively**.** In comparison with similar work done by Manger, Trejderowski, & Paduch (2010) indicated that Bootstraps the development procedure is a major cause of software framework usability.

4.7 The Effects of Software Framework usability on Software Development The table below (Table 5) shows respondents view on the effects of Software Framework usability on the variables which has been explained below, the predictors: (Constant) variables were it deters learning the actual language, provides an inflexible programming paradigm, hampers website load-time, bugs and security wholes discovered in framework code can (and probably will) affect every application built using it, need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques, hard to introduce two or more frameworks in the same project, and some frameworks may not contain all necessary libraries or classes forcing developer to write on or take an attempt to introduce more than one framework in a project, and large overhead of framework code, in some situations noticeable decreasing application performance whilst the dependent variable is the effects of Software Framework usability. The results were based on the regression analysis with a margin of error of 0.05 and a confidence level of 95%. It can be deduced from the table that out of the seven independent variables used in the predictions, only four representing provides an inflexible programming paradigm, Bugs and security wholes discovered in framework code can (and probably will) affect every application built using it, Need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques and Hampers Website Load-time were affected by the dependent variable which is indicated as effects of Software Framework usability on software engineering. This means that effect of Software Framework usability on software engineering had significant contribution on the statements, however the extent of significance depended on the co-efficient variable. It should be noted that a negative coefficient signifies that there is no direct relationship between the variables whilst a positive co-efficient variable indicates that there is a positive relationship amongst the variables. This therefore suggest that even though effect of Software Framework usability contributed significantly to the statements there is a positive relationship between effect of Software Framework usability and the software engineering to a large extent. This is line with Folmer, Van, & Bosch (2003) proposition that Software framework has a positive effect on usability but are difficult to retrofit into applications because they have architectural impact.

Table 5: The Effects of Software Framework usability on Software Engineering

Independent variable	Coefficients			95.0 ıfidence Interval for B	
		t	P-value	Lower Bound	Upper Bound
(Constant)		-3.370	.001	-.842	-.221
It deters learning the actual nguage	-.068	-.717	.474	-.314	.146
Provides an inflexible programming paradigm	.258	3.843	.000	.126	.390
Hampers Website Load-time	-.193	-3.342	.001	-.419	-.108
Bugs and security wholes discovered in framework code can (and probably will) affect every application built using it	.306	6.264	.000	.307	.588
Need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques	.503	11.905	.000	.346	.484
Hard to introduce two or more frameworks in the same project, and some frameworks may not contain all necessary libraries or classes forcing developer to write on or take an attempt to introduce more than one framework in a project	.134	2.627	.009	.064	.449
Large overhead of framework code, in some situations noticeable decreasing application performance	.126	1.333	.184	-.069	.358

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This chapter involves three parts. The first subdivision presents a summary of the .research findings. The second and third sections present respectively the conclusions drawn from the research and recommendations put forward.

5.2 Summary of the Study

This study sought to examine the Growth of Software Framework usability and its effects on software engineering. This study was set up to achieved the following objectives; to identify the types of Software Framework use in Software development, examine the causes of rapid growth of software framework usability in software development and examine the effects of software framework usability on software development. Base on the study objectives, the researcher set up the following Research Questions (What types of Software Framework do you use in Software development? , What are the causes of rapid growth of software framework usability in software development?, What are the effects of software framework usability to software development?

The study employed both descriptive study designs to help the researcher achiev+e the various objectives. The targeted population (277) for this study covers the final years computer science or information technology masters' students in five public universities (UEW-50, UDS-40, Kumasi Technical University-32, KNUST – 40 and UNER - 50) in Ghana and two software companies (Gracecoms – programmers 35 and Kologsoft – programmers 30). The study adopted the Yamane's formula for estimating the sample size required samples for the study. The require sample size estimated for this study was (249). The study employed the following random sampling techniques in selecting the students and software developers for the study. The researcher relied solely on primary data in conducting the study and the instruments used for the data collection was structured questionnaire. In this study, both descriptive and inferential statistics were used. For the descriptive statistics, table and graphs were used to simplify the results and interpretation of the findings.

The study found that there is very high/ high 60% (n= 149) level of growth of software framework usability in software engineering. This is line with Varvana M., Sari, Mikko, & Piia (2018) proposition that Software frameworks are nowadays extensively used to develop different kinds of software applications efficiently. And the popular software frameworks use was bootstrap 32.1% (n=80) and flutter 26.1% (n=65).

Secondly, the study found that widely held views of the students and software developers 44% (n = 110) were agreement with acknowledged causes and minority of them 29% (n = 72) were in disagreement with the identified causes students and software developers identified as causes of rapid growth software Framework useability in Software Engineering. Again, the mean scores in the table 3 suggest that the most leading causes students and software developer identified are cause 2*:* Bootstraps the development process and cause *5:* It is faster to develop a system with framework than pure coding with the highest mean score of (means = 2.87, 2.67) with standard deviation of (std = *0.47*, 0.64***)*** respectively**.** In comparison with similar work done by Manger, Trejderowski, & Paduch (2010) indicated that Bootstraps the development procedure is a major cause of software framework usability.

Lastly, the study found that even though effect of Software Framework usability contributed significantly to the statements, there is a positive relationship between effect of Software Framework usability and the software development to a large extent. This is line with Folmer, Van, & Bosch (2003) proposition that Software framework .has a positive effect on usability but are difficult to retrofit into applications because they have architectural impact.

5.3 Conclusion

In this study, the researcher investigates the growth software usability and its effect on software development and find answer to the research question stated in this study. Results of this study exhibits that, there is very high/ high 60% (n= 149) level of growth of software framework usability in software development and most developers use bootstrap and flutter framework.

Empirical results of this study indicated that the major reason why developer use of software framework usability is because it is faster to develop a system with framework than pure coding with the highest mean and standard deviation score of (means= 2.67, std = 0.64)*)* respectively.

However, researcher could find significant statistical support for software framework usability when there exists a positive relationship between effect of Software Framework usability and the software development to a large extent.

5.4 Recommendations

The ends with following recommendations;

5.4.1 Suggestion for Feather Research

As future work, a holistic view of different kinds of boundary resources throughout the developer journey could benefit from more research. The study ends with the suggestion that feathers study should be directed toward the importance of software framework Usability in Agile Software Development.

REFERENCES

Abdul, S. K. (2005). *Knowledge Management in the Construction Industry*: A Socio technical Perspective. London: Idea Group Inc (IGI.

Agyedu, G., Donkor, F., & Obeng, S. (2010). *Teach Yourself Research Methods, . .* Kumasi: University Press, KNUST.

Bhatti, R. (2013). Impact of ICT on social science faculty members' information usage pattern at Bahauddin Zakariya University, Multan. *Library Philosophy and Practice (e-journal)*, 928.

Burgess, R. (2016). *The Ethics of Educational Research Volume 8 of Social Research and Educational Studies Series* (Vol. 24). Colorado: Taylor & Francis Group. Retrieved August 5, 2019, from https://pdfs.semanticscholar.org/3d72/1907de7e64a0084952e41255359d586d1e2c.pdf.

Christain, M., Tomasz, T., & Jaroslaw, P. (2010). Advantages and disadvantages of framework programming with reference to Yii php framework, gideon .net framework and other modern frameworks. *Studia Informatica, 31*(4A), 119-137. Retrieved 09 20, 2018, from http://studiainformatica.polsl.pl/index.php/SI/article/view/343/342

Creswell, J., & Plano, C. (2007). *Designing and Conducting Mixed Methods.* London: Sage Publication LTD.

Crewell, J. W. (2002). *Education Research: Planning, Conducting and Evaluating Quantitative and Qualitative Research. Upper Saddle River,.* NJ: Merill/Prentice Hall.

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management Science*, 982–1003.

Folmer, E., Van, G. J., & Bosch, J. (2003). A framework for capturing the relationship between usability and software architecture. *Software Process: Improvement and Practice, 8*(2), 67-87.

Frank, B., Regine, M., Hans, R., Peter, S., & Michael, S. (2001). *Pattern-orinted Software Architecture.* John Wiley & Sons Ltd.

Gall, M. D., Borg, W. R., & Gall, J. P. (1996). . *Educational research: An introduction* (6th ed.). White Plains, NY: Longman.

Hochstein, L., & Lindvall, M. (2005). Combating architectural degeneration: a survey. *Information and Software Technology, 47*(10), 643-656.

Jenny, P. (1999). Identifying the best research design to fit the question. Part 2: qualitative designs. *BMJ, 2*(2), 36-37. Retrieved 10 01, 201, from https://ebn.bmj.com/content/2/2/36

Jongbo, O. (2014). The role of research design in a purpose driven enquiry. *Review of Public Administration and Management, 3*(6), 90. Retrieved from : www.arabianjbmr.com/RPAM_index.php

Kahn, O. &. (1998). Spin-transition polymers: from molecular materials toward memory devices. *Science*, 44-48.

Kerlinger, F. (1986). *Foundation of behavioural research* (3rd ed.) . New York: Holt, Rinehart, and Winston.

Leopoldo, R. Y., Marcio, C. O., Fontana, C. F., Sakurai, C. A., & Yano, E. T. (2014). Framework for designing automotive embedded systems based on reuse approach. *International Journal of Systems Applications, Engineering & Development, 8*, 9-17.

Manger, C., Trejderowski, T., & Paduch, J. (2010). Advantages and disadvantages of framework programming with reference to Yii php framework, gideon. net framework and other modern frameworks. *Studia Informatica, 31*(4A), 119137.

Mary, S. R., & Rodriguez, P. (2012). Software Architecture- Evolution and Evaluation. *International Journal of Advanced Computer Science and Applications*, 82-88.

Mattsson, M. (2000). *Evolution and Composition of Object-Oriented Frameworks.* Retrieved 09 20, 2018, from http://www.divaportal.org/smash/get/diva2:836468/FULLTEXT01.pdf

Motroc, G. (2017, 04 04). *What's next for frameworks?* Retrieved 09 19, 2018, from Jaxenter: https://jaxenter.com/new-jax-mag-issue-frameworks-133767.html

Oliveira, T. C., Alencar, P. S., Lucena, C. J., & Cowan, D. D. (2007). RDL: A language for framework instantiation representatio. *The Journal of Systems and Software*, 1902–1929. Retrieved 09 20, 2018, from https://www.cos.ufrj.br/~toacy/material/jss.pdf

Perry, D. E., & Wolf, A. L. (1992). Foundations for the Study of Software Architecture. *Software Engineering Notes, 17*(17), 40-52. Retrieved 09 10, 2018, from http://users.ece.utexas.edu/~perry/work/papers/swa-sen.pdf.

Pree, W. (1994). Meta Patterns: A Means for Capturing the Essentials of Reusable Object-Oriented Design. *8th European Conference* (pp. 150–162). Bologna: Springer-Verlag. Retrieved 07 25, 2019, from https://www.springer.com/gp/book/9783540582021

Sarah, M. S. (2012). Retrieved from Explorable.com: https://explorable.com/types-of-survey

Saunders, M., Lewis, P., & Thornhill, A. (2012). *Research methods for business students, 6th Ed.* Harlow: Harlow : Financial Times Prentice Hall.

Upworks (2018). Retrieved 09 10, 2018, from https://www.upwork.com/hiring/development/understanding-softwareframeworks/

USC Libraries Research Guides (2018). Retrieved from https://libguides.usc.edu/writingguide/researchdesigns

Varvana, M., Sari, K., Mikko, R., & Piia, S. (2018). Development as a journey: factors supporting the adoption and use of software frameworks. *Journal of Software Engineering Research and Development*. Retrieved 10 26, 2020, from https://jserd.springeropen.com/articles/10.1186/s40411-018-0050-8

Varvana, M., Sari, K., Raatikainen, M., & Pile, S. (2018, 05 15). Development as a journey: factors supporting the adoption and use of software frameworks. *Journal of Software Engineering Research and Development*, 1-22. doi:https://doi.org/10.1186/s40411-018-0050-8

Wayner, P. (2015). *7 reasons why frameworks are the new programming languages*. Retrieved 09 15, 2018, from Infoworld: https://www.infoworld.com/article/2902242/application-development/7reasons-why-frameworks-are-the-new-programming-languages.html Wikipedia. (2018). Retrieved 10 12, 2018, from https://en.wikipedia.org/wiki/Research_design

Wikipedia. (2018). Retrieved 09 20, 2018, from https://en.wikipedia.org/wiki/Software_framework

. Wikipedia. (2018, 09 19). *Programming paradigm*. Retrieved 09 22, 2018, from Wikipedia: https://en.wikipedia.org/wiki/Programming_paradigm

APPENDIX A

UNIVERSITY OF EDUCATION, WINNEBA

DEPARTMENT OF INFORMATION TECHNOLOGY EDUCATION

Dear Respondent,

.I would be pleased if you could willingly devote your valued **20 minutes** and vigor to fill my twenty (20) questionnaires. The purpose of this questionnaire is to examine the growth of Software Framework Usability and its effects on software engineering or development. Kindly provide answers to the questions by ticking the appropriate boxes. Your confidentiality and anonymity are assured. This is exclusively for academic purposes.

I count on your cooperation and may God richly bless you.

Thank You

INSTRUCTIONS: Please tick appropriate boxes or write short sentences where necessary

SECTION A: DEMOGRAPHIC INFORMATION

1. Gender: Male [ ] Female [ ]
2. Age: 18- 25 years [ ] 26- 30 years [ ] More than 31 years [ ]
3. Are you a ….? Student [ ] Software Developer [ ] Others [ ]
4. How many years of experience do you have in your Software Framework Usability?

   1- 5 years [ ]

   6- 10 years [ ]

   11-15 years [ ]

   More than 15 years [ ]

SECTION B

WHAT TYPES OF SOFTWARE FRAMEWORK DO YOU USE IN

SOFTWARE DEVELOPMENT

5. What types of software framework do you use in software development? (tick as many items as possible)

   Django [ ] Flask [ ] Ruby on Rails [ ] Laravel framework [ ] Drupal [ ]

   Joomla [ ] CodeIgniter [ ] Symfony [ ] AngularJs [ ] jQuery [ ] Sass [ ]

   EmberJS [ ] VNode.JS [ ] ReactJS [ ] LESS [ ] VB.NET [ ] ASP.NET [

   ] Bootstrap [ ] Cocoa + Cocoa Touch [ ] jQuery Mobile [ ] Flutter [ ] AngularJS +

   [ ] Ionic [ ] React Native [ ]

6. How will you rate the level of growth of software framework usability in software engineering?

   Very High [ ] High [ ] Neutral [ ] Very Low [ ] Low [ ]

SECTION C: WHAT CAUSE THE RAPID GROWTH OF SOFTWARE FRAMWORK USERBILITY IN SOFTWARE ENGINEERING

SN	DESCRIPTION	SD	D	N	A	SA
7	Reduces Code Length					
8	Bootstraps the development process.					
9	Reinforces Security					
10	Framework system are more efficient than pure coded systems.					
11	It is faster to develop a system with framework than pure coding					
12	Improves database proficiency					
13	Eases debugging and application maintenance					

Key: SD = Strongly Disagree, D = Disagree, N= Neutral, A = Agree, SA = Strongly Agree

SECTION D: WHAT ARE THE EFFECTS OF SOFTWARE FRAMEWORK

USERBILITY TO SOFTWARE ENGINEERING

SN	DESCRIPTION	1	2	3	4	5
14	It deters Learning the Actual Language					
15	Provides an Inflexible Programming Paradigm					
16	Hampers Website Load-time					
17	Bugs and security wholes discovered in framework code can (and probably will) affect Every application built using it					
18	Need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques					
19	Hard to introduce two or more frameworks in the same project, and some frameworks may not contain all necessary libraries or classes forcing developer to write on or take an attempt to introduce more than one framework in a project					
20	Large overhead of framework code, in some situations noticeable decreasing application performance					

Key: 1 = Strongly Disagree, 2 = Disagree, 3= Neutral, 4 = Agree, 5 = Strongly Agree

APPENDIX B

RELIABILITY TEST FOR QUESTIONNAIRES

Reliability Statistics

Cronbach's Alpha	N of Items
.885	29

Cronbach's Alpha = 0.885 indicate Excellent

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
Gender	41.64659	48.939	.697	.876
Age	41.13253	44.623	.857	.868
Are you a	41.44578	46.313	.750	.872
How many years of experience do you have in your Software Framework	41.65462	47.953	.608	.876
Usability				
Django	42.90361	55.031	-.472	.891
Laravel Framework	42.92771	54.237	-.302	.888
Mobirise	42.93976	53.863	-.184	.887
JQuery	42.87952	55.711	-.569	.893
VB,NET	42.91968	54.502	-.367	.889
ASP.NET	42.83936	56.756	-.687	.896
Boostrap	42.62651	59.299	-.817	.904
Flutter	42.68675	58.926	-.812	.903
AngularJS	42.80723	57.463	-.749	.898
Codelgniter	42.94779	53.630	.000	.886
How will you rate the level of growth of software framework usability in software engineering	40.49398	44.574	.861	.868
Reduces Code Length	40.33333	45.570	.796	.870
Bootstraps the development process	40.07229	50.180	.569	.879

Reinforces Security	40.74297	47.885	.777	.873
Framework system are more efficient than pure coded systems	41.67068	48.496	.636	.876
It is faster to develop a system with framework than pure coding	40.26908	46.310	.782	.871
Improves database proficiency	40.62249	47.107	.803	.872
Eases debugging and application maintenance	41.84739	51.412	.493	.881
It deters Learning the Actual Language	40.22892	46.959	.762	.872
Provides an Inflexible Programming Paradigm	40.41365	44.856	.829	.869
Hampers Website Load-time	40.21285	47.588	.771	.873
Bugs and security wholes discovered in framework code can (and probably will) affect Every application built using it	40.14458	48.503	.692	.875
Need to follow framework coding convention, which might differ much between different frameworks, switching between frameworks may require to learn a completely new approach to coding techniques	40.75100	43.276	.810	.869
Hard to introduce two or more frameworks in the same project, and some frameworks may not contain all necessary libraries or classes forcing developer to write on or take an attempt to introduce more than one framework in a project	40.12851	49.379	.757	.876
Large overhead of framework code, in some situations noticeable decreasing application performance	40.24900	46.494	.767	.872