UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SOFTWARE ENGINEERING

HANDBOOK

Name :	**Matrix No :**	**Subgroup :**
1. Muhammad Kasyfi Bin Kamarul Hamidi	A20EC0093	1
2. Muhammad Yusri Bin Yusoff	A20EC0102	
3. Mohamad Haziq Zikry Bin Mohammad Razak	A20EC0079	
4. Dzakirin Asyraff Bin Zamsari	A20EC0030	2
5. Muhammad Aniq Aqil Bin Azrai Fahmi	A20EC0083	(In Charge)
6. Nur Afikah Binti Mohd Hayazi	A20EC0220	
7. Shady Nabeel Y Hamza	A20EC0267	3
8. Abdulrahman Ahmed Rafat Abdelhamid	A20EC0253	
9. Muhammad Naim Bin Abdul Jalil	A20EC0096	

ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to our lecturer, Dr Norhaida Binti Mohd Suaib. Who helped us in succeeding in this project. Without guidance from her, we couldn't have accomplished this. This opportunity also enabled us to understand about our future careers as Software Engineering Students.

We would also like to express our gratitude to everyone in the Main group of 40.1, who has helped us directly or indirectly.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 What is Software Engineering?

In this world of advanced technology, these courses are dominant in the technology industry. Almost everything needs a software application to execute any kind of task from as simple as ordering foods to make complex microchips for computers. Software Engineering consists of many kinds of fields such as Software Requirements, Software Design, Software Development, Software Testing, and Software Maintenance. Software Engineers work on many types of projects because they hold a responsibility to turn ideas into the final product that's why software engineers are vital to the modern world nowadays.

In today's advanced world, software engineering holds the key to ensure the improvement in software quality to all industries using these as their core in their business. Software Engineering also has a different relationship with other Computer Science, Management Science, Economics and System Engineering principles. Here they play different principles in providing their best to the clients or their company (Rungta, 2020). All Software Engineers have the basics of all other disciplines stated and that makes them more reliable for the job.

Aside from these advanced engineering software technologies from History, we also can learn that there are many failures in the past software development project. For example, a scene that happened in the 1968 NATO Conference. Many companies declared software engineering failures after facing bankruptcy, abandoned projects, and the programs failed to achieve the objectives in the past days. From this, we can conclude that Software Engineering has gone through many kinds of ups and downs in the industry, and until today it finally became a vital part of this Industrial revolution 4.0 (Ewusi-Mensah, 2003).

1.2 The Objectives

- To develop a set of skills that stands out together with the needs of current global computing-based.

- To develop the skill to advance software engineering's career and continuously upgrade for professional and technical skills.

- To improve the quality of human needs to a better standard of lives, making new technologies that dynamic global can adapt to the new technologies.

- To develop technical and soft skills; be able to work in groups and organizations to solve computing-based problems.

1.3 Aspects of Software Engineering

Methodology in Software Engineering requires 4 conditions that the information technologist needs to require, where software engineering had a lot of branches. However, Methodology means the methods used in a particular area of study. As approaching to be a software engineer, the information technologist needs to have 4 conditions. Case 1; the information technologist has to be in apprenticeship mode, Apprenticeship mode is a stage the information technologist need training of profession on-job-training and also gain a license in a regulated profession.

Case 2; the information technologist needs to be independent and capable of working on their own with their minimal supervision. Case 3, the information technologist needs to accomplish the project with their own where they can hands-on their efforts to lead and inspire others. Lastly, Case 4 is the information technologist have to create their executive role and be concerned about guiding the organization as a whole and let the others IT people can grow, learn and fix together for the software project.

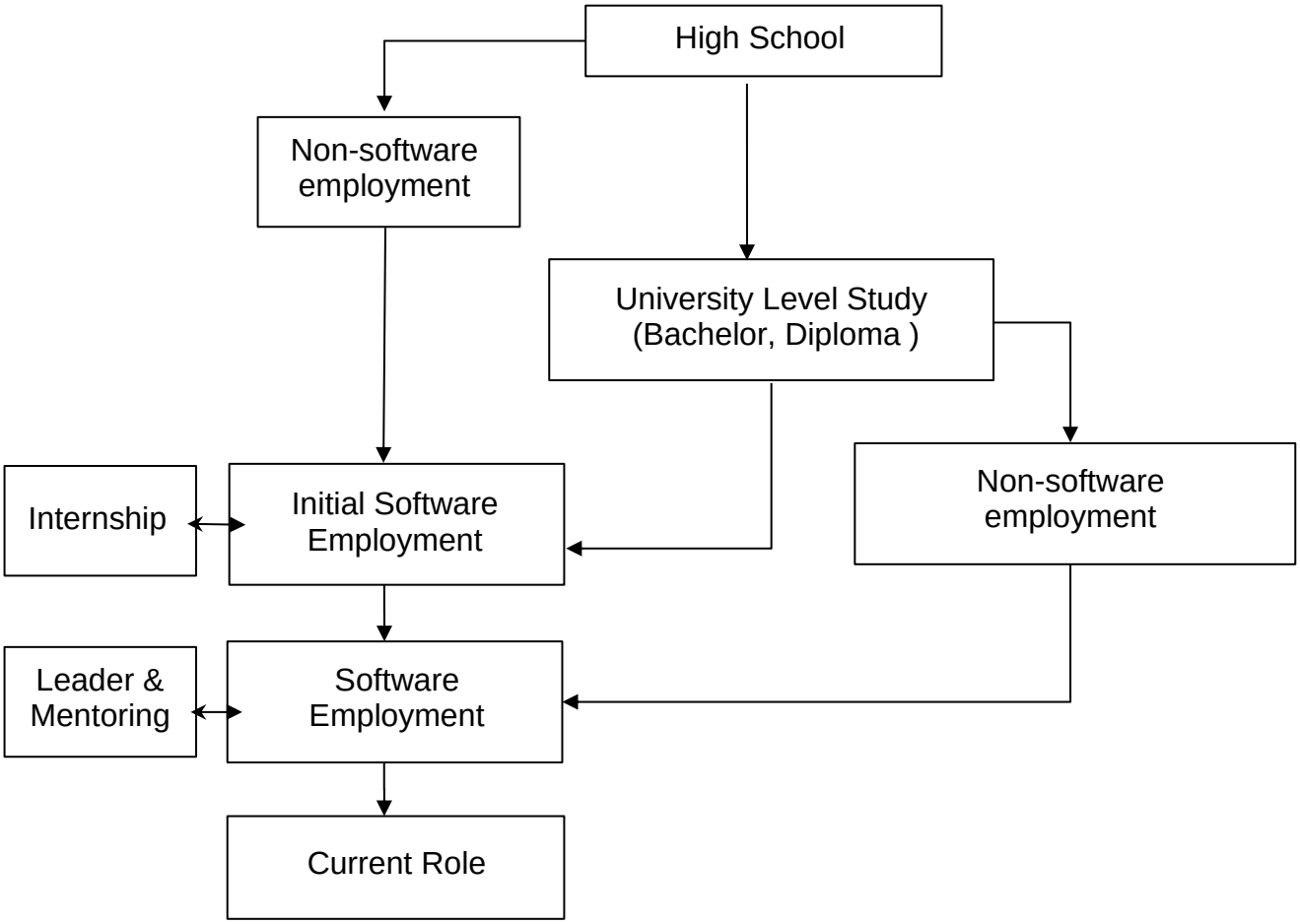

Figure 1: The Origin path of Information Technologist

Based on figure 1, the result can be shown commonly in our world software flow diagram that if a person has a lack of university qualification, they don't have any barrier to enter the IT profession. Logically, whether people are in IT courses or not, the IT profession is not a double standard, as someone from outside the IT studies can seriously jump into the IT area within the knowledge they have such that the IT profession does not have any barrier to enter and exit the role of software employment. Even, a person does not have any background and experience they can still manage to enter the IT profession. Anyone can be an information technologist without a doubt.

2 REQUIREMENTS IN SOFTWARE ENGINEERING

2.1 Introduction

Every field of study has to have suitable requirements for people to get involved in that specific field of study. Software engineering is not excluded, as it is one of the most growing fields and industries in the 21st century. There are ultimate skills that are required such as soft skill, logic thinking, and advanced knowledge of information technology. The checklist done for the ultimate skills will bear as a great software engineer. A great software engineer surely will be good in soft skills as soft skills are not easy to achieve in a short time, it takes a long time to gain. However, the skill and knowledge requirements of a software engineer are evolving and changing for the future so the software engineering needs more updates on skill and knowledge to catch up with the current trend and future IT.

2.2 Skill and Knowledge

2.2.1 Soft Skill

Soft skill is a skill that is unable to be touched as it has no physical presence. Just a thought, every software engineer surely has the hard skills for the work and also some experiences. However, in any job hard skills are the most important but for hiring people to work together as a whole in a company or organization the employers are looking for the ultimate skills like soft skills. Soft skills mean a personal attribute such as skills in communication, teamwork, enthusiasm, and leadership. Soft skills are indeed the critical situation to success in any business. If making a bad or wrong decision, any person's skills and knowledge can lead bad causes to the organization due to a problem such as attitude, management and engagement (Dean & East, 2019).

Soft skills examples of personal attributes and interpersonal attributes:

Personal Attributes

Communication

Communication is looked at from a process and a message is transferred from the sender to the receiver and vice versa. Communication inquires two categories; verbal communication and non-verbal communication. Verbal communication is a need to communicate appropriately in working to avoid any sort of misunderstanding and different assumptions in the working zone. While non-verbal communication is a body language where it presents what we feel and thinks about the matter being talked about.

Good communication from a good language used where you communicate a message from one to one another in the team. Language needs to use manners grammar through all the conversation, before talking thinks first about the matter, avoiding a lot of signal messages and complex sentences can be used by applying simple sentences to convey the messages. Good communication did not use verbal and nonverbal fillers during the conversation. Not only for verbal communication skills but we also have written communication skill and presentation skill.

Figure 2: A Flow of Good Communication

Teamwork

Teamwork is the effort for the team to collaborate to achieve a task efficiently. To be true, teamwork means either gender, different age groups, status, skills and knowledge work together as a whole in a team to complete the task. From teamwork, people can have exposure from having strong work ethics, good personal chemistry, positive attitude and interpersonal skills. As current work mostly needs work done in groups to achieve the objective of the project.

Teamwork is needed because the success of a company is not dependent on only one person doing all the task. The truth of the company's success can be shown in the result of many people working toward the same target. When all the employees manage to get done their task together and propose the project. It means everyone wins. Instead, teamwork can boost job satisfaction so the employees will be all equality and motivate themselves to get done the planned project together.

However, in teamwork, we will manage to have a positive attitude when all the team's members spread healthy manners as in case healthy manners also are contagious where it can spread good vibes to each of the team's members. In a working network, it is really important to have that energy and behaviour. As it can keep all the team's members going forward when they are under stress, underperforming, have difficulty so it can make the work environment more fun.

Interpersonal Attributes

Leadership

Leadership is an interpersonal skill where it can motivate, communicate, build the organization into the positive vibes. Leadership is also a linkage between the ability to inspire, communicate and leadership effectiveness. As looking at the top pyramid of management, leadership is the highest rank one. The leadership will effectively focus on how the organization will progress and change.

The linkage between communication and leadership will bring success in changing of organization. If the leader having a lack of management skill it will bring failure thus making an un-able to influence others team's clique. Therefore, if a leader can implement the success of change including coaching, motivate, build the growth of the group, and rewarding the top worker. So, if having a lack of leadership skill it will be ineffective not be able to make the organization expands and bigger (Dr Vasanthakumari S, 2019).

Figure 3: What leadership can do to give success to the organization.

2.2.2 Logic Thinking

What is Logic Thinking?

Anyone with a well-developed mind would have at least a basic ability to reason, and reasoning is the basic requirement for logical thinking. Logical thinking is involved in making a conclusion from relating facts and situations, solving problems and so on (Plessis, 2012). Logical thinking promotes problem-solving skills, mathematical reasoning, strategic thinking, etc. It is generally crucial because the correlation of events and objects can easily be made, and rational or logical people are likely more successful at establishing cause and effect relation as well as developing the right strategies with minimal mistakes (Cole, 2017).

Importance of Logic Thinking

Logical thinking is specifically important in this tech and software industry and needs to be mastered and improved over time. As the technological world advances rapidly, IT professionals have to foster a critical thinking approach. The organization in which we function is complex, and it is informed by the nature of individual thought processes as well as current technology and market pressures. Any adjustments will have causes and effects that could have a much broader impact. Solving an issue will alter things, which may lead to other issues. Too often, interconnections are overlooked, a single cause can be inferred, or an individual is easily blamed. This is not unique to IT, we see this all the time in broader society; it is easier to blame particular offenders for crime than to contend with the many nuanced social variables that led some to criminality. The other mistake focuses on results, i.e. how many suspects we can prosecute rather than how many crimes we can stop. To prevent these failures, by thinking about the processes that impact the challenge or opportunity, issues should be addressed. This is more complicated than isolating and solving a problem, but more likely to yield a better solution eventually (Bateman, 2015).

Logic Thinking in Software Engineering

In a niche software engineering industry, logical thinking is very important. Software often mediates contact between the company and the external environment, and the company has a duty to its larger society that can be served or threatened by this software. By designing or adapting software, managing projects and sales, analyzing results and customer data, and automating tasks, professional software engineers may add a lot of value. All of these happen in a dynamic real world, where humans respond in various ways to change. Every new framework must consider how it will be reacted to by users or clients. The ability is not one of understanding what to do, it is one of knowing how to model the interactions between the software, the company it represents, and its wider setting. In development, roll-out, updates and maintenance, this method should be used as it is a changing process. Logical thinking doesn't mean sidelining technology. By understanding various software engineering tools that can help them simulate, control and track, the process can be further advanced. Using these successfully is part of the ability to plan well for IT.

How to Improve Logic Thinking

Logical thinking skills can easily be improved with the right habits and routines. One of the habits is establishing the habit of questioning. Questioning skills must be developed and used everywhere. Whatever data that have been gathered, whether facts and numbers or just hunches, all of it has to be verified. Checking information sources and researching any piece of data that are even slightly questionable. Before beginning to determine the importance of any such information received, everything must be verified for its validity. Next is changing the perspective. Understanding the prejudices and biases that we humans might have by deciding precisely how they can influence the way data are handled. Be versatile enough, even though they contradict our long-held convictions, to look at a problem from multiple perspectives. Accept and entertain all new knowledge with an open mind, without any personal prejudices that we might have (Logical Thinking, 2016).

2.2.3 Advanced Knowledge of Information Technology

Advance Knowledge is a knowledge that needs a deep study to master it and use it in a software engineering career some of this knowledge is vital and need a good understanding of the concept to apply it in the industry

Programming Languages

Programming Languages is essential for a programmer to have at least one programming language that their masters use in their work or task. Every developer must have an interest in a particular area, and they must feel comfortable when working with the languages their use to develop any kind of software. Different languages have their pros and cons. For example, python is excellent and famous for its Web Development and Data Science, including machine learning, data analysis, and data visualization scripting. Many programmers in this work area used to use python as their work program.

Figure 4: Most wanted languages in the industry by stack overflow survey 2020

Databases

As a programmer, they need to have a good understanding of databases. Every developer should master operations like storing records, creating, insert, updating and deleting specific data or essential information that must be stored. Databases are useful for the developers to store any crucial data about their users. Without databases creating any practical application is almost impossible nowadays. In this generation of information technology, the developer needs to manage all the security issues using databases so they can store backup files in case something might happen to their servers (Bortz, 2010).

Figure 5: Example of databases used by developers

Data Structure and Algorithm

This skill helps developer lookout for new recruitments as it tests the programmer's problem solving and coding skill. A good software engineer knows how to organize data and use it for solving real-life problems (Bortz, 2010). Developers learn Data Structure and Algorithm to help them write optimized and scalable code in a specific condition. Next, its effective use of time and memory. This experience and knowledge help run a program efficiently and not use much memory when developers work on a project. Initially, most developers do not realize its importance, but when developers start their software development career, they will find their code to take too much time or take too much space (Learn Data Structures and Algorithms, 2020).

Software Quality Assurance

Software Quality Assurance (SQA) also known as a process to make sure that all the standard software engineering processes, methods, activities and work items are following the user-defined standards. The knowledge is useful from the development defining requirements until the coding is released for public usage or the clients (What is Software Quality Assurance (SQA): A Guide for Beginners, 2020).

Figure 6: Software Quality Assurance Plan

This SQA plan is used for planning the procedures, techniques and tools are all aligned with requirements defined.

Software Design & Architecture

This knowledge is very vital as it can bring up many factors for business, quality human dynamics, and IT environment. This part can be separated by two distinct phases which are Software Architecture, which is for nonfunctional decision while Software Design is a functional requirement that has been accomplished.

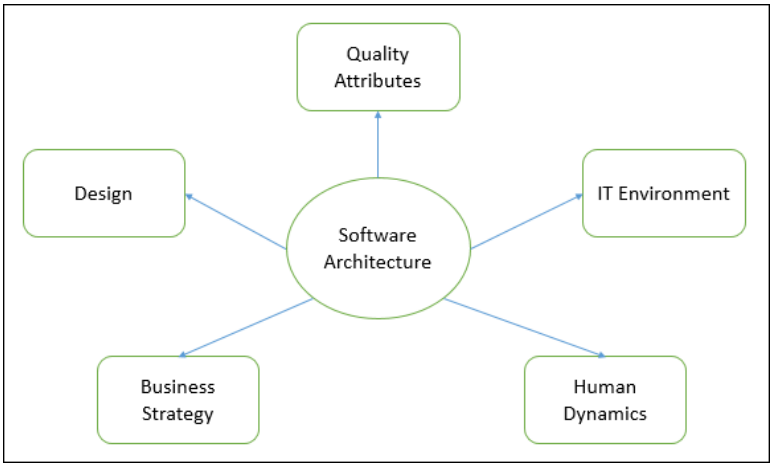

Figure 7: Factor of Software Architecture

Software Architecture provides a system blueprint and gives the fundamentals on how to manage the system complexity. It involves a set of decision making and brings an impact on the quality and performance of the system. Finally the overall success of the final product (Software Architecture & Design Introduction, 2020). Software Design gives a design plan the elements of a system so that they can achieve the objective of a design plan which is to negotiate system requirements and give an expectation with customers, marketing and others. It is also a guide for the implementation task (Software Architecture & Design Introduction, 2020).

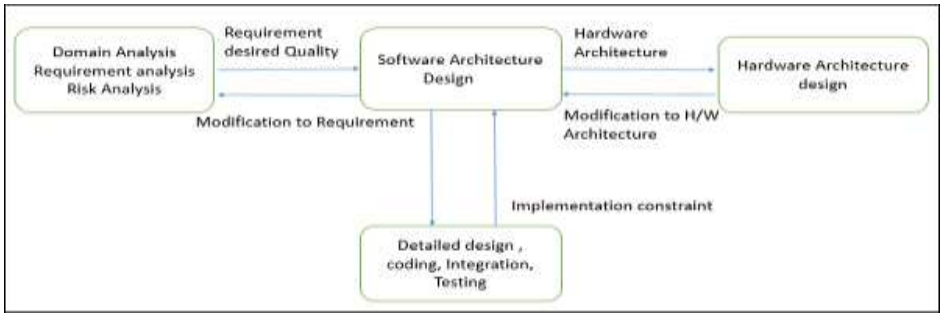

Figure 8: The Software Architecture Design flow

Artificial Intelligence

Artificial Intelligence has the capabilities to do tasks smarter than any human ability. Artificial Intelligence used AI applications used for problem-solving and making decisions for the system. Today on this Industrial Revolution 4.0, Artificial Intelligence began to be essential for any system to use in their project because the output can be more precise and satisfies the client.

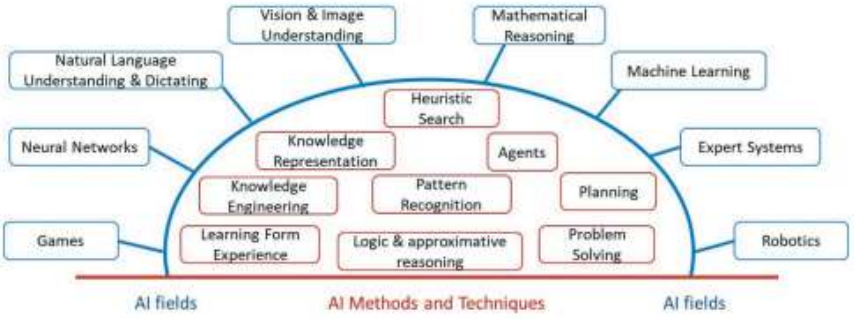

Figure 9: AI Fields and techniques

Artificial Intelligence uses many mathematical calculations to get the exact values that the system desires to make a decision. AI also always improves their efficiency by learning from experiences to solve future problems. This method is really useful and helps human beings so that they shouldn't have to worry about the decision of their system but just by monitoring their behaviour would help sustain the developed programs (Shehab, Abualigah, Jarrah, & Daoud, 2020).

2.3 Summary

All in all, software engineering might not have the most rigorous requirements relative to other industries or sectors such as healthcare or academia, but it does have specific requirements that need to be mastered to be included in the software engineering industry. The crucial skills are discussed above, i.e. advanced knowledge of IT, soft skills and logical thinking. Advanced knowledge of IT is the main key in software engineering, with specialisation towards certain subjects such as mastery of programming language software quality assurance. Soft skills and logical thinking are not less important as these are the qualities that make a successful IT professional in the ever-changing tech industry. All of these go hand in hand with each other, complementing one another to provide IT professionals with the best value.

3 OUTLINE OF SOFTWARE ENGINEERING

3.1 Introduction

In this Industry of advanced technology, software engineering principles have this many departments depending on which section they are masters at or many experiences they have in creating successful software. Some of them are experienced in managing a project, or experts in certain fields.

3.2 Outlines

3.2.1 Software Requirements

This is one of the fields that needed to collect any software requirement from the clients, analyses, and documents each of them (Software Requirements, 2020). This software requirement runs with four main step processes: a Feasibility Study, Requirement Gathering, Software Requirement Specification, and Software Requirement Validation.

Feasibility Study

This is important when the client wants to approach to get their desired product. Clients will develop a rough idea for the product and expect all features to perform in the software. From this, the analyst will study the detailed process and its functionality is feasible enough to develop the software.

This study will determine the software product can be practically materialised in every term including the implementation, contribution, cost and the objectives. All the technical aspects are taken into account (Software Requirements, 2020).

After this phase, the feasibility study report must have all the recommendations and comment for the management to deal with the project whether they should take it or not

Requirement Gathering

After the report from the feasibility study is good to take the project, it's time to gather all the requirements from the clients. The engineers must communicate with their end-user on what software features needed to include.

Software Requirement Specification

Software Requirement Gathering is also known as SRS created by the system analyst after requirements are collected. SRS should have the features to use a natural language, written in pseudo-language and format of the GUI screen prints (Software Requirements, 2020).

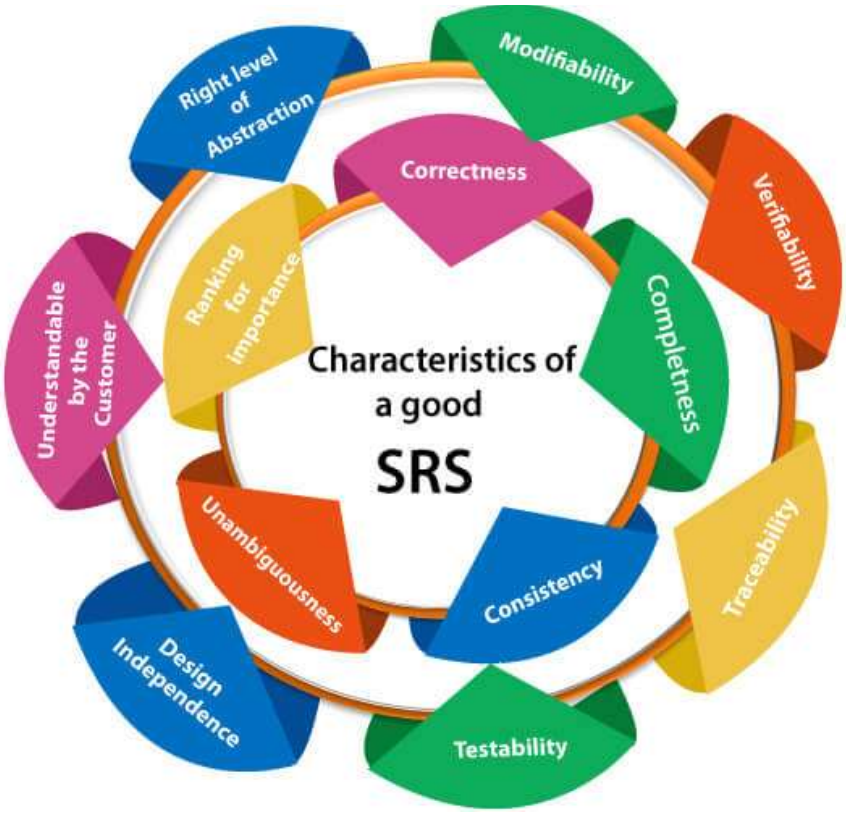

Figure 10: Characteristic of a good SRS

To have a good SRS a developer must at least have all of these characteristics because these will make the projects run smoothly. Any conflict between the clients can be solved because the SRS is complete and would make the developer have a full understanding of what project they've been assigned (Software Engineering | Software Requirement Specifications, 2011).

3.2.2 Software Design

Software designers are in the field where they are really on duties for problem-solving and planning for a solution of software. Software designers are creating new ideas and designs for pre-packaged and customizing computer software. Software design is a procedure of transforming user requirements into a convenient form, where it helps the programmer in coding and development. A software design will become more specialized and detailing all requirements in the software regulation. Although, we can say that the software design is the process directly used into implementations in programming languages like C++, C, Java, Python, etc. However, software design moves the major point from a problem to the solution of the domain. It tries to be more specialized to fulfil the software requirement (Software Design Basics, 2020).

Figure 11: Programming Languages

When the market identifies the needs, software designers come out with a program by deciding what type of program they will make it. Then, write those specifications from programmers' code computer instructions to perform the given functions. The software designer writes the program specifications, and the applications programmers write the coding in the programming language. Software designs can be categorized into three levels of results, such are architectural design, high-level design, and detailed design (Software Design Basics, 2020).

Table 1: Type levels of design

Levels of Design	***Descriptions***
Architectural Design	● The top category in the system can identify many elements interacting with one another in a software system. At this time, designers get to generate ideas to propose a solution.
High-Level Design	● Returns the function component concept of architectural design into less-complicated subsystems and modules and interaction with each other.
Detailed Design	● Development part as seen in system and sub-systems in the previous two designs above. More up to standard and specialized towards modules and their implementation.

Software Design Process & Structured

As mentioned above, other than software design results there are also software design approaches. Firstly, structured design is mostly a "dividing and filling" strategy in which the problem is returned to several small problems and the small problem is slowly solved individually until the whole problem has been solved.

The small problems mean a solution of modules. The modules are organized in a hierarchy system and communicate with one another. To be the truth, a well-structured design always has some communication rules among modules which namely cohesion and coupling. Cohesion means all the functions are in groups related to elements while coupling means two different modules that communicate with each other. A desired structured design has very high cohesion and very low coupling structured (Software Design Strategies, 2020).

Software Design Verification

The final product of the software design process is design documentation, pseudocode, diagrams with detailed logic, and the descriptions of all functional or non-functional software requirements. The next procedure is the software developing where all of the implementations depend on all design structures and processes mentioned above.

However, it is a must before proceeding to the next step of development of software to detect any earlier mistake before testing the product. In case the output of the design structure is informal form, the verification tools will be used or either a design review can be used for design validation and verification.

Parts for validation and verification approaches, reviewers can detect the mistakes that might be overlooked. The desired design review is important for having a good software design process, structures, and accuracy. Progressing to the next phase, which is software development when all the review of defects is good (Software Design Strategies, 2020).

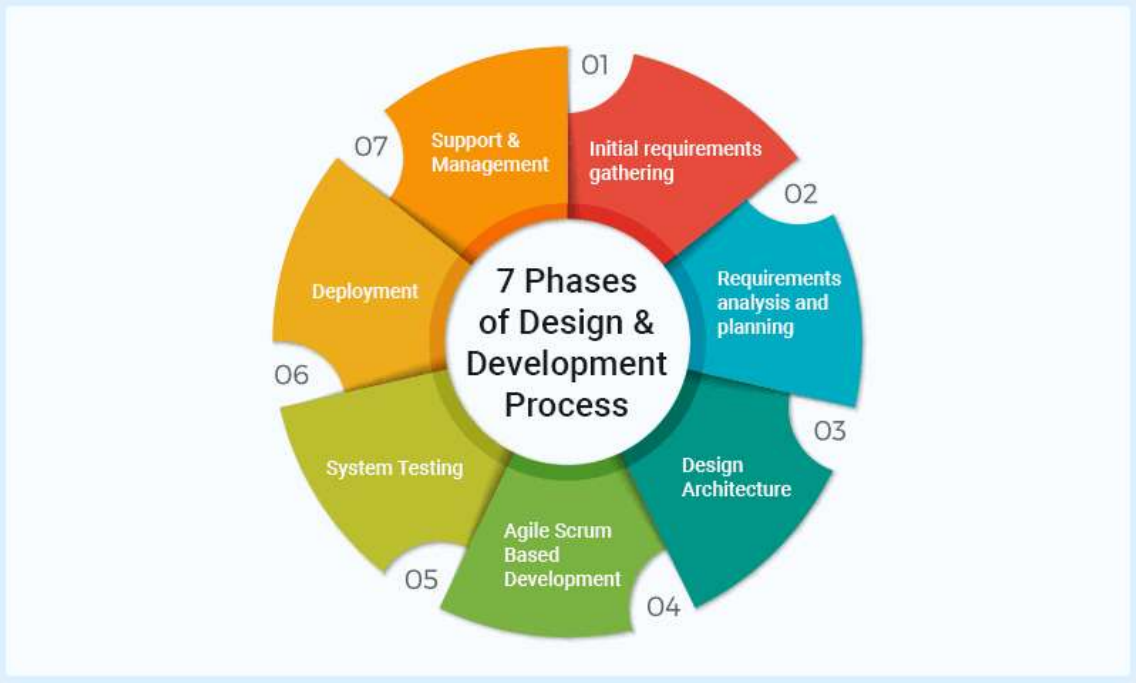

Figure 12: Software Design & Development Process

The Job, Salary of a Software Designer

The job of a software designer is recommended to have a certified professional license before you come into the industry or freelance. The salary range of software designers is a minimum of $50,000. Instead of the job, the minimum education level is only for bachelor's degree and people who have certified professional license even if they are out of the IT circle. To have a person to be hired in the company look at how much faster the person can be compared to the average. The only personality traits that software designers need to have is a creative person, knowledge on problem-solving and being more scientific. Software designer career ladder is from a computer programmer, then a software designer projects team ladder and software manager (Software Designers, 2015)

3.2.3 Software Development

Software development is really important to be used in the process for programmers to build computer new programs. The process is known as the Software Development Life Cycle (SDLC) where this is one of the methods on how the software can reach the user requirements. This is structured that can follow up from design, creating and maintenance of top quality software. The purpose of software development is building products efficiently in the desired budget and timeline ahead (Software Development Life Cycle, 2020).

Figure 13: The Software Development Life Cycle

There are six main steps in the software development life cycle:

I. In need to identify the problems
II. User requirements analysis
III. Designing
IV. Development
V. Testing
VI. Maintenance

Table 2: Steps in SDLC

Main Steps	***Descriptions***
1. Identify Problems	● Process of brainstorming before building software. ● Identifying functions and all the services needed for the software.
2. User requirements analysis	● Providing a detailed outline for every component including tasks, scope and parameters for the high-quality product.
3. Designing	● Software architects will propose a solution up to the specifications for user requirements.
4. Development	● Coding will be developed within the user requirement in the previous stages.
5. Testing	● Checking for bugs and the performance before producing it.
6. Maintenance	● No, detect a defect, the product is sent to customers and manages to create a team to encounter issues with clients.

Careers in Software Development

- Computer Programmer
- Quality Assurance Engineers
- Database Administrator
- System Analyst
- Software Engineering

3.2.4 Software Testing

What is Software Testing

The next step towards a successful project of software engineering is the testing phase. Software testing is an investigation that is undertaken to provide information on the quality of the software product or service under evaluation to stakeholders (Kaner, 2006). Software testing is an important part of the software engineering process for a few main reasons, mainly because of cost-effectiveness, customer satisfaction, security, and product quality (Rajkumar, 2015).

Software Testing Approaches

There are many testing approaches that can be done to test a program. The main three approaches are static, dynamic, and passive testing. Static testing requires no program execution such as code review, software peer review and proofreading; meanwhile, dynamic testing is any type of testing activity that involves running the code itself (Oberkampf & Roy, 2010). Passive testing looks for pattern and behaviour and is related to log analysis to find this pattern (Lee, Netravali, Sabnani, Sugla, & John, 1997). There is also exploratory approach, which is coined by Cem Kaner in 1984 and defined as “a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project” (Kaner, 2008).

Other than that, there is also a “box” approach, which can be divided into white-box testing and black-box testing. White-box testing verifies the internal workings of a program, instead of the functionality exposed to the end-user. To design test cases, an internal perspective of the system which is the source code, as well as programming skills, are used in white-box testing. The tester will choose inputs to exercise paths through the code and decide the suitable outputs (Limaye, 2009). Figure 13 shows the white box testing diagram.

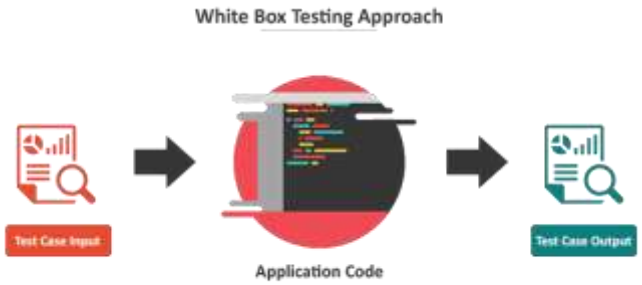

Figure 14: White Box Testing Approach Diagram

Black-box testing, on the other hand, considers the entire program/software as a black box. The tester does not need any prior knowledge of programming as they are now aware of how the software does something, but instead, they only know what the software is supposed to do (Saleh, 2009). Figure 14 shows the black box testing diagram.

Figure 15: Black Box Testing Approach Diagram

3.2.5 Software Maintenance

What is Software Maintenance

Last but not least, the final process in a software engineering project is software maintenance. Software maintenance is a vast task that requires optimization, correction of errors, removal of discarded features and improvement of existing features. A system for calculating, monitoring and making changes must be developed because these changes are required. During the development cycle, the critical part of software maintenance involves the preparation of an accurate schedule. Maintenance normally takes up about 40-80 percent of the cost of the project, generally closer to the higher pole. An emphasis on maintenance, therefore, undoubtedly helps to keep costs down (The Economic Times, 2020).

Importance of Software Maintenance

There are a few reasons why software maintenance is important and required in the whole software development cycle. One of the main reasons is market conditions in which policies that change over time, such as taxes and newly imposed restrictions, including how book curation can be maintained, can cause the need for change (Software Maintenance Overview , 2020). Secondly, maintenance is important due to the customer’s requirements and needs. Clients can ask for new features or functions in the software over time. Host modifications could also be the reason for maintenance because if any of the target host's hardware or platform such as operating system changes, software updates are necessary to retain adaptability. Lastly, there is also a possibility for organizations shifts. If there is some shift in the corporate level at the consumer end, such as decreasing the strength of the organization, purchasing or merging with another company, venturing into a new business, there might be a need to alter the original software. Simply put, software maintenance is required to correct any faults in the software, to greatly improve the design and to facilitate the program so that different hardware, software, or system can be used and accommodated (Software Engineering | Software Maintenance, 2018).

Categories of Software Maintenance

Software maintenance can be classified into 4 distinct categories which are preventive maintenance, corrective maintenance, adaptive maintenance, and perfective maintenance. Preventive maintenance applies to improvements to the software carried out to protect the product for the future. As they plan for any future changes ahead, software maintenance changes are preventive. This means making it easier to scale or retain the source code and handle the legacy content but it also involves identifying and repairing latent defects in your brand, before they turn into operational defects. Corrective changes in the maintenance of software are those that fix software glitches, bugs and defects. On a semi-regular basis, it often comes in the form of quick, small updates. For consumers, corrective software maintenance is unlikely to trigger negative opinion. No one would be unhappy about getting irritating bugs, glitches or problems fixed. This type of improvement helps make the user experience better as well as more consistent immediately and obviously. Adaptive maintenance of software tackles the issue of the constantly shifting technological environment. New hardware, comprehension and cybersecurity threats mean that software becomes obsolete quickly. Adaptive changes concentrate on the device infrastructure. To retain continuity with the software, they are made in response to new operating systems, new hardware, and new platforms. Lastly, perfective maintenance may be the most significant and most important type of software maintenance. The software's accessibility and functionality are solved by ideal perfective software maintenance. Perfect maintenance includes refining, removing, or adding new features to improve current product functionality. As well as altering the way a product works, perfect modifications can also alter the way it looks. The ideal maintenance category also involves user interface tweaks, redesigns, or in-app user travel changes (Parker Software, 2018).

3.3 Summary

That is all of the Outlines of Software Engineering that all Software Engineering graduates can master consists of many fields. These fields are essential for any organisation that wants to recruit for new software engineering. It shows the person's speciality when operating some complex system with a particular method to succeed in each project. As described Software Requirements, Software Design, Software Development, Software Testing and Software Maintenance possess a piece of excellent knowledge and experience to conduct by following ethical procedures. Lastly, every graduated Software Engineers need to find the perfect career path to boost their competency in the industry to stay relevance

4 OBSTACLES IN SOFTWARE ENGINEERING

4.1 Introduction

There are several obstacles that come with having a career in software engineering. These include, but not limited to, unestablished project environments, changing developmental expectations, time limitations, and rapid technology changes.

4.2 Obstacles

4.2.1 Unestablished project environments

Having an unestablished environment to work makes it difficult for workers to effectively produce results, hence it will severely impact project delivery. Without a proper environment, it will be difficult to continue with the project and finish on time and with the current budget

4.2.2 Changing developmental expectations

Sometimes, when deciding on projects, the ideas will come without a specific set of requirements. For example, during the initial planning, the group has decided on a certain design of a product. However, before moving on with the project, the upper management decides that a few features should be added to make things more interesting, without specifying what the features should be. This leads to confusion and the developers will struggle to meet the ever-changing demands of the ones in charge.

4.2.3 Time limitation

A career in software engineering always plays with time. Software developers work under pressured environments and strive to fulfil project requirements within strict and time-limited timeless conditions, especially when having an international client at multiple time-zones. Lack of time can cause the downgrade of the developer’s quality products.

4.2.4 Rapid technology changes

Every IT company is doing the best for the development of technology. But at the same time, it will cause more pressure for the professional software developer to adapt to the new trends of technology to ensure they would be able to survive and compete in the new market. Sometimes rapid technology might cause trouble to developers as their ideas or software development will be called outdated.

5 CONCLUSION

As addressed in our analysis and research, software engineering is crucial towards the advancements towards technology worldwide. From the requirements of studying software engineering to having a career related to software engineering, it is clear that it takes a lot of effort to master the theory and understanding of it. Having skills such as soft skills, logical thinking, and professionalism are just some qualities that make a person successful in this field. With that being said, there are also many barriers or obstacles to become successful in the software engineering field. Therefore, these obstacles are good for people to adapt and adjust accordingly so that they can succeed even further in their respective field of software engineering. Thus, it can be drawn that software engineering will be an industrial shift that will change how we as a society function and how it could be adapted and implemented in the real world.

REFERENCES

Bateman, K. (2015). *The growing importance of critical thinking in IT education*. Retrieved from Computerweekly.com: https://www.computerweekly.com/blog/ITWorks/The-growing-importance-of-critical-thinking-in-IT-education

Bortz, D. (2010). *Top software engineer skills for today's job market*. Retrieved from Monster Career Advice: https://www.monster.com/career-advice/article/software-engineer-skills

Cole, L. (23 October, 2017). *What Is Logical Thinking? 8 Tips to Improve Logic | MentalUP*. Retrieved from MentalUP.co: https://www.mentalup.co/blog/what-is-logic-how-to-develop

Dean, S., & East, J. (2019). Soft Skills Needed for the 21st-Century Workforce. *International Journal of Applied Management and Technology*, 19. Retrieved from https://scholarworks.waldenu.edu/cgi/viewcontent.cgi?article=1260&context=ijamt

Dr Vasanthakumari S. (30 September, 2019). *Soft skills and its application in work place*. Retrieved from ResearchGate: https://www.researchgate.net/publication/337181806_Soft_skills_and_its_application_in_work_place

Ewusi-Mensah, K. (2003). *Software development failures.* Mit Press.

Kaner, C. (2006). Exploratory Testing. *QAI*, (p. 11). Retrieved from http://www.kaner.com/pdfs/ETatQAI.pdf

Kaner, C. (2008). A Tutorial in Exploratory Testing. *QAI*, (p. 36). Chicago. Retrieved from http://www.kaner.com/pdfs/QAIExploring.pdf

Learn Data Structures and Algorithms. (2020). Retrieved from Programiz.com: https://www.programiz.com/dsa

Lee, D., Netravali, A. N., Sabnani, K. K., Sugla, B., & John, A. (1997). Passive testing and applications to network management. *Proceedings 1997 International Conference on Network Protocols*, (pp. 113-122). Atlanta. doi:10.1109/ICNP.1997.643699

Limaye, M. G. (2009). *Software Testing.* Tata McGraw-Hill Education. Retrieved from https://books.google.com.my/books?id=zUm8My7SiakC&pg=PA108&redir_esc=y#v=onepage&q&f=false

Logical Thinking. (12 August, 2016). Retrieved from Cleverism: https://www.cleverism.com/skills-and-tools/logical-thinking/

Oberkampf, W. L., & Roy, C. J. (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press. Retrieved from https://books.google.com.my/books?id=7d26zLEJ1FUC&pg=PA155&redir_esc=y#v=onepage&q&f=false

Parker Software. (2 October, 2018). *The 4 software maintenance categories and what they mean for your users*. Retrieved from Parker Software: https://www.parkersoftware.com/blog/the-4-software-maintenance-categories-and-what-they-mean-for-your-users/

Plessis, S. d. (21 December, 2012). *Logical Thinking: A Learned Mental Process*. Retrieved from Edublox Online Tutor | Development, Reading, Writing, and Math Solutions: https://www.edubloxtutor.com/logical-thinking/

Rajkumar. (6 December, 2015). *What Is Software Testing | Everything You Should Know*. Retrieved from Software Testing Material: https://www.softwaretestingmaterial.com/software-testing/

Rungta, K. (2020). *What is Software Engineering? Definition, Basics, Characteristics*. Retrieved from Guru99.com: https://www.guru99.com/what-is-software-engineering.html

Saleh, K. A. (2009). *Software Engineering*. J. Ross Publishing. Retrieved from https://books.google.com.my/books?id=N69KPjBEWygC&pg=PA224&redir_esc=y#v=onepage&q&f=false

Shehab, M., Abualigah, L., Jarrah, M. I., & Daoud, M. (25 June, 2020). *Artificial intelligence in software engineering and inverse: review*. Retrieved from ResearchGate: https://www.researchgate.net/publication/342457338_Artificial_intelligence_in_software_engineering_and_inverse_review

Software Architecture & Design Introduction. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_architecture_design/introduction.htm

Software Design Basics. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_engineering/software_design_basics.htm

Software Design Strategies. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_engineering/software_design_strategies.htm

Software Designers. (2015). Retrieved from Vault: https://www.vault.com/industries-professions/professions/s/software-designers

Software Development Life Cycle. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm

Software Engineering | Software Maintenance. (11 October, 2018). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/software-engineering-software-maintenance/

Software Engineering | Software Requirement Specifications. (2011). Retrieved from javatpoint: https://www.javatpoint.com/software-requirement-specifications

Software Maintenance Overview. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_engineering/software_maintenance_overview.htm

Software Requirements. (2020). Retrieved from Tutorialspoint.com: https://www.tutorialspoint.com/software_engineering/software_requirements.htm

The Economic Times. (2020). *What is Software Maintenance?* Retrieved from The Economic Times: https://economictimes.indiatimes.com/definition/software-maintenance

What is Software Quality Assurance (SQA): A Guide for Beginners. (13 November, 2020). Retrieved from Softwaretestinghelp.com: https://www.softwaretestinghelp.com/software-quality-assurance/