

INDIVIDUAL REPORT

COMP8117-1-R-2021S

21-08-2021



SUBMITTED TO:

Dr. Aznam Yacoub

SUBMITTED BY:

ANUBHA SHARMA

110037181

Contents

[Introduction 3](#_Toc80444883)

[What’s That Knowledge? 4](#_Toc80444884)

[So, Was It Worth It? 7](#_Toc80444885)

[Was It a Smooth Ride? 9](#_Toc80444886)

[What Did I Learn & What Else I Want to Learn? 11](#_Toc80444887)

[References 14](#_Toc80444888)

## Introduction

According to the definition, software engineering is the process of evaluating user needs and then designing, developing and testing software applications that will satisfy the requirements specified by the client. Specifically, it is concerned with the processes and procedures that are used in the industry to produce an enterprise project. It is concerned with increasing the reliability and dependability of the systems. Projects at the enterprise-level are complex and have a large range of requirements; therefore, neglecting to follow software engineering processes when building them can be devastating, resulting in low-quality projects and the loss of life or cancellation of the project.



[1]Figure: 1

Taking a look at the history of software engineering and its origins, we can see that it all started as a result of a software-crisis issue [2]. Many projects were running late and over budget, and this was causing concern. A few projects resulted in the deaths of civilians, and everything was traced back to poor code quality and poor project management techniques. As the scale of the projects increased, the code grew cluttered and teams were unable to manage it. As a result, researchers began looking for procedures that would result in high-quality code that could be relied on to deliver results. Because of this, various lifecycle models and techniques have been developed by the researchers, which has resulted in an improvement in the overall quality of the projects over this period. Because of this necessity for responsible coders, the profession of software engineer was created. Although software engineering standards impose some limitations on developers, they must be strictly adhered to in order to deliver high-quality projects that do not end in disastrous outcomes.

Obtaining customer requirements, designing, developing, and testing a specific piece of code or functionality can be accomplished through a number of different approaches. To complete all of these steps, however, depending just on our best guesses is insufficient, and does not always result in a high-quality solution. In this case, software engineering techniques and software engineers are absolutely vital to success. Although these procedures and processes should be, they should also be also be customized to the specific requirements of the project, in my opinion. Every domain is different, every client is different, and every project has human involvement in the form of team hence, adhering to these standards and practices can result in projects that are high-quality, maintainable, and scalable.

Thinking about the project we completed as part of our coursework: the initial vision and features that we had in mind did not align with what our project managers and customer desired from us. However, because we followed Agile principles, we were able to incorporate practically all of the customer's requests in our project, and all of this was made feasible because we were adhering to a predetermined standard. With regard to the others, we actually anticipated whether or not this was the best practice, and we were able to do so since we had researched the procedures beforehand. We gained information from the training, and it was only with the support of that knowledge that we were able to persuade the customer that either the request was not in accordance with best practices or was not practicable and we were able to complete our project on time.

## What’s That Knowledge?

Software engineering methods have been developed with the understanding that, in addition to solving the real-time concerns of the client, the code must also be able to do the following:

* Readable
* Testable
* Maintainable
* Extendable
* Reliable
* Reusable
* Secure
* Portable
* Available

Today, the business requirements of customers are continuously changing, and as a result, the software that they use must change in order to keep up with those requirements. Making certain that the code produced to meet the current requirements has the features listed above will assist in future upgrades.

Despite the fact that these objectives are basic, achieving them is quite difficult. As a result, software engineering practices are required across all domains. Because every domain is different, each demands a particular set of practices. For example, in domains where requirements change infrequently, a waterfall model can be used, whereas in domains where requirements change frequently, an agile model can be used. Regardless of the domain in which we are working, software engineering practices make the lives of developers and stakeholders easier because the seemingly huge task is divided into small parts and can be solved independently.

When we were doing our coursework, we came up with the idea of creating a community project called Nurture Community. In the course of presenting an initial proposal to the customer, we discovered that the customer and project manager had a number of distinct ideas. We were overjoyed by this development. It was a fantastic opportunity to include the client and your manager in providing you with their thoughts on how we could improve our services. However, this also implies that, despite the fact that we had fixed the domain of giving food to the needy, there may be many other recommendations, requirements could change, and we could not put everything in stone. Also, we believed that incrementing the projects over simple small functionalities and involving the customer in providing us with their valuable feedback would help us build a better project and keep the customer satisfied because they would be aware of what has been accomplished and what the team is currently working on.

Keeping all of this in mind, we evaluated all of the existing options available on the market and determined that Scrum is the most effective approach. The project progresses during the course of the several sprints, which follow the scrum approach. The following are some of the terminologies [3]:

* SPRINT:

A sprint is a timed iteration in the development lifecycle. At the beginning of a sprint team members along with the product owner and scrum master meet and decide on the features that will be completed during this iteration and the rest of the features/functionalities are added to the sprint backlog. The selected features are expected to be completed in terms of coding, review, testing, and integration into the evolving product.

* SCRUM CALL:

During a sprint, a daily scrum call is held during which all team members meet and discuss

* the work done on the previous day,
* the work they will pick up next
* if they are facing any challenges while the development and need help.

Daily Scrum, is a mechanism for team members to coordinate their work. For our web application development scrum call will be held daily over MS-Teams and generally will not be more than 15 mins long.

* SPRINT PLANNING

During Sprint Planning we identify how and what will be accomplished during a sprint. The discussion happens at the beginning of each sprint. During this meeting, sprint goals are defined.

* SPRINT REVIEW

It is conducted at the end of each sprint and identifies what has been accomplished during this sprint and how it contributed to the development of the product.

Apart from these we also divided these roles of Scrum Master, Product owner, Scrum team, QA Lead and QA team among ourselves. The definitions and responsibilities of these roles are defined below

* SCRUM MASTER:

He is responsible for ensuring that team follows the values and practices that they said they would. He also acts as a guide clearing their obstacles and laying down a clear path towards the goals.

* PRODUCT OWNER:

Product owner is responsible for defining requirements, diving them into stories and manages the backlog for the team.

* SCRUM TEAM:

These are the team of the developers who work on their assigned task.

* QA LEAD:

He guides the team, in preparing the testcases and oversees that the team does an efficient job. He clarifies the doubts of the QA team.

* QA TEAM:

They are responsible for ensuring that the functionality developed is according to the needs of the customer. They are also responsible for testing the non-functional requirements.

There are other processes which we followed, all of these can found in QAP, we have submitted.

## So, Was It Worth It?

Industrial software is difficult to develop and is quite complicated. Furthermore, software engineering aids in the division of large problems into smaller ones. It is thus possible for individual members to take on little challenges, so reducing complexity and making the problems more manageable. The use of software engineering methodologies to proceed through the project helps to streamline the project's progress. It reduces the time required for development.

Consider the situation of scrum connect, as an illustration of this. It facilitates team collaboration by ensuring that everyone is on the same page and that everyone understands what each member of the team is responsible for completing. Team members can also share information about problems they are encountering while working, and if someone in the team has prior experience with the problem, they can collaborate to solve it together. This may appear to be a little aspect, yet it is quite beneficial. It decreases the overall development time, and the team is able to benefit from one another's knowledge and expertise. This has all been made possible as a result of following the process.

One of the situations that occurred during our assignment brought home to me the necessity of these principles for the first time. Due to a communication breakdown, two of our team members ended up doing the same thing during the initial setup of the project, which caused a confusion. Despite the fact that we only lost half a day, we learned an essential lesson: it is critical to stay on the same page with the team in order to increase productivity while saving time.

Additionally, some of our team members had prior experience, while others lacked it. We were able to assist our colleagues and find a rapid solution because we were aware they were having difficulties. It fosters interpersonal relationships among team members, which is critical for breaking the ice with new team members, and ultimately, it benefited us in completing the project on time and within budget.

In addition, I've learned to appreciate the importance of unit testing in the context of a codebase. The project will be more costly if a bug in the system is discovered during the integration testing or manual testing phases. This is because the developer may have moved on to a different issue or may not be available at that time, and when you return to the code after a period of time, you are likely to forget why certain pieces of the code were written and what the intention of a particular piece of code was. Unit tests come in handy in situations like this. It aids in describing the purpose of a piece of code, which, in turn, aids in the reduction of resolution time. Furthermore, the earlier a bug is identified and fixed, the lower the overall cost of the project will be in the long term. In addition, it permits you to remain focused on the current task at hand, rather that understand the prior work done.

Having an understanding of the many difficulties that an engineer has to deal with also assisted us in deciding on the technological stack that we would utilize. For example, we opted to use GIT software in order to keep track of the modifications that were made to the code. Keeping up to date with each other's code is especially critical when working with a distributed team since otherwise some of us can end up coding our tasks on redundant code, which would be quite inconvenient. Additionally, not utilizing a versioning system may result in an additional effort of merging the entire code base at the conclusion of each iteration. I am not claiming that using GIT will totally liberate you from the need to integrate your code; you will still need to integrate sometimes; nevertheless, taking regular checkouts from the remote saves a significant amount of time and work and increases productivity.

Then there was the situation with the web API and Android. The primary goal of our community-based initiative was to feed food to those in need while also reaching as many people as possible, which we accomplished. Despite the fact that we did not conduct a survey, we wanted everyone to be able to use the application, regardless of what platform their phone was running on or whether they had a phone at all. We assumed they would be able to access our application through an internet café. Apart from that, we wanted to evaluate the initial response of the audience before deciding whether or not to invest in a separate Android and iOS application. It is unlikely that we would have reached these conclusions if we had not asked the fundamental questions that software engineers should be asking themselves during the project initiation process. Who are we attempting to assist, and what are our objectives? As a result, the Web API was victorious in this competition.

Following that, there was a battle between Vue.js and Angular. Even though Vue.js contains all of the features of Angular, it is a new framework that has learned from the faults of its predecessor and is far lighter and faster. In addition, we learned during our course that while making decisions, we should always keep the team dynamics in mind, and only a few of us had prior experience with frontend technologies, out of a total of seven people. So , we chose Vue.js because the learning curve for this technology is far less difficult.

To summaries’, if the time and effort spent setting up all of the processes and procedures, creating all of the documentation, keeping the customer informed, deciding on the architecture, data flow, and spending time deciding on the technological stack was worth it, my answer would be a resounding "YES." These processes and procedures have increased our productivity by streamlining many time-consuming tasks, they have assisted us in developing a high-quality project that is maintainable and easily extendable, and the regular feedback from the customer and project manager has motivated the team to maintain the high standards that are expected of any software engineer working for the company.

## Was It a Smooth Ride?

Well, the response would be a resounding NO. Implementing engineering practices did assist us in streamlining the project, but it was a bumpy ride. On the bright side, it provided us with valuable insights into a real-world project, and we gained a great deal of knowledge and experience.

For-example, The GIT that was chosen to assist us in tracking the changes resulted in a major difficulty for us. Due to inexperience one of our peer committing some changes into the GIT without looking at the current architecture of the code, and as a result, the entire build became corrupted, and no one was able to execute anything. After conducting extensive study, we were able to find a solution; nonetheless, there was a brief moment of panic. After we resolved the issue, the team discussed it and came to the agreement that no one would commit code to the GIT repository before it has been peer reviewed by others. We had established this as a part of our practice previously as well, but no one took it seriously until this incident occurred. After that, everyone took it seriously.

Then there was JIRA, which was used so that everyone could keep track of the development of the stories. However, no one filed any work during our first sprint, although the work was completed. It became increasingly impossible for us to justify our hours to the project manager at the end of this sprint, and we gave up. It would not have been a problem if we had logged our effort in JIRA instead.

In addition, we encountered a difficulty with the JPA implementation. JPA was supposed to make our lives easier by partially automating a data layer, but it ended up corrupting our data instead. In reality, we were all working on a single database in order to save money on administrative costs. We had to enter fresh information into the system in the middle of the project. Because it was the project beginning, we were able to complete it in a short period of time; nevertheless, in the context of larger projects, this would be devastating. The problem stemmed from the coding technique used. Although the issue was rectified, it served as a reminder of how important testing and code reviews are. Furthermore, every real-world project necessitates the creation of a backup database.

In addition, we agreed that we wanted to alternate our positions during the duration of the project. Despite the fact that we were asked to recreate the working of a real-world project and that this would never happen in the real world, we decided that since we were in the learning stage, we should give everyone the opportunity to participate in leadership roles. It wasn't a particularly smooth ride. Because of their inexperience, some of the team members were unable to provide leadership to the group, while others encountered difficulties in splitting the task and others encountered difficulties in estimating. However, we were able to settle the situation as a team. This may not appear to be a lesson in software engineering, but it was a lesson in the significance of having strong and consistent leadership in a team. Even though I have had some past experience working on real-world projects, I had never appreciated the necessity of managers' duties until this project, and assumed developers were capable of completing everything on their own.

## What Did I Learn & What Else I Want to Learn?

I've discovered that being a Software Engineer entails more than just writing code. Being a software engineer entails adhering to ethical standards. Software can be found everywhere; we utilize a variety of gadgets from dawn to evening, every day. In order to provide our customers with the most reliable solution possible, we as software engineers must keep our knowledge and abilities up to date on an ongoing basis.

I've gained a great deal of knowledge about the processes and procedures that are used in the industry. From the fundamentals, such as UML diagrams, to the art of communicating with customers, from the different components of leadership required to operate a team to the gathering of needs from the end users, from doing cost analysis to market analysis, there was a lot to learn. Because of the necessity of adhering to these standards and the ethics that must be upheld in the software engineering field, this project, I believe, has placed me on the route to becoming a successful software engineer in the future.

During the course of our project work as engineers, I believe that we were all involved in the design of the project, decided how the communication process should look like, how data should flow, what technology stack would be most beneficial in the long run, what non-functional requirements are a priority for the customer, and what documents would be useful to us in the long run. As an engineers, we have to negotiated with the customer about our requirements, essentially advising them and persuading them that something was not feasible in the long term. We divided the workload amongst ourselves and then focused our efforts on meeting a single goal to contribute towards the project while following all of the processes and procedures and customizing our processes in need with the dynamics of our team. We also kept in mind the privacy and security of the data of the customer and for the future development our major goal is to work towards improving the performance of the project and incorporating some other essential features that we think will add value to the lives of our customers.

I've also learned that there isn't a one practice that can solve all of your difficulties all by itself. Software engineering is a process of continuous improvement process and although, the we must take advantage of the newest solutions available on the market, we must also keep an eye on what is working in terms of the dynamic of our team. We should learn from any problems that we encounter, consult with our superiors, and discover solutions as soon as possible. We should teach this new procedure to our fellow teammates and we will only be able to construct a reliable solution and run like a well-oiled machine if we are constantly learning new things and improving our current processes.

Furthermore, as software engineers, we should not be afraid to admit when we do not understand something. We should acknowledge this and learn about the requirements for the project. Asking questions is always beneficial and it is usually preferable to invest some time inquiring about the doubts rather that building something that doesn’t fit the needs to our customer. Also, as software engineers we must recognize the importance of a second perspective and we should be always willing to listen to our colleagues. As engineers we also have to acknowledge when something is not working and should declare it among the known bugs. This increases the trust among the customer and the team and also improves the efficiency of the team

Although learning about the various software engineering methods was beneficial, I would like to learn more about architectures and how they are implemented in the real world. Which architecture should be used where, which modal should be used when, and which problems do each of these solutions solve? I also learned a great deal about myself by undertaking this project. In each sprint planning session, I asked for the most interesting problem to be assigned to me, and I hope to have the opportunity to solve even more interesting problems in the future. I learnt that I am a coder at heart but I want to transition from being a software developer to a software engineer, the one who considers the big picture, is responsible for deciding the blueprint of the projects, considers how to make code reliable, extendable and secure rather than simply giving a solution.

Also, I'd like to gain experience in a variety of fields. I've worked in the retail industry. But before I make a decision regarding a domain I want to settle in, I'd like to experience a variety of domains and collaborate with a variety of different people.

# References

|  |  |
| --- | --- |
| [1] | "https://www.fzi.de/," [Online]. Available: https://www.fzi.de/en/about-us/organisation/research-divisions/se-software-engineering/. |
| [2] | "https://en.wikipedia.org/," [Online]. Available: https://en.wikipedia.org/wiki/History\_of\_software\_engineering. |
| [3] | "https://www.tutorialspoint.com/," [Online]. Available: https://www.tutorialspoint.com/scrum/index.htm. |
| [4] | "https://www.lucidchart.com," [Online]. Available: https://www.lucidchart.com/blog/tips-for-a-perfect-business-requirements-document. |
| [5] | "http://web.cse.ohio-state.edu/," [Online]. Available: http://web.cse.ohio-state.edu/~bair.41/616/Project/Example\_Document/Req\_Doc\_Example.html. [Accessed 26 05 2021]. |
| [6] | "https://dl.acm.org/," [Online]. Available: https://dl.acm.org/doi/10.1145/2393347.2393433. [Accessed 28 05 2021]. |
| [7] | "What is Software Engineering? Definition, Basics, Characteristics," [Online]. Available: https://www.guru99.com/what-is-software-engineering.html. [Accessed 13 09 2021]. |

