

INDIVIDUAL REPORT

COMP8117-1-R-2021S

13-08-2021



SUBMITTED TO:

Dr. Aznam Yacoub

SUBMITTED BY:

ANUBHA SHARMA

110037181

##### What is software engineering and why is software engineers are important?

Solution:

Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements [1].

Enterprise-level projects are complex, and failing to follow software engineering processes when developing them can be disastrous, resulting in low-quality projects or even cancellation

We all know that there are a variety of methods to obtain customer requirements, create, and test a given piece of code. However, relying solely on our best guess to complete all of these processes is insufficient and does not always guarantee us a high-quality outcome. Software engineering processes and software engineers are essential in this situation. These procedures and processes, in my opinion, should not be blindly followed; rather, they should be tailored to the specific needs of the project. Every domain is different, every client is different, and every project will always have a human touch, but adopting these standards and procedures can lead to high-quality, manageable and scalable projects.

If I think about the project we made during our coursework, the initial vision and features that we had in mind were not in line with what our project managers and customer wanted from us. But since we employees’ Agile practices, we were able to accommodate almost all the ask from the customer in our project and all this could be made possible we were following a set standard. For other which we could not, we actually anticipated if this is the best practice and we could do that because we had studied about the practices. We had the knowledge by studying the course and only with the help of that knowledge we could convince the customer that either it was not the best practice or is not a feasible ask.

##### Is software engineering really needed to produce (industrial) software?

Solution:

Industrial software is complex and difficult to produce. And software engineering helps to divide big problem into small ones. The small problems can then be taken up my individual teams, which helps reduce the complexity and makes the problems manageable.

Employing software engineering techniques to progress over the project helps streamline the project. It reduces the development time. For example, consider the case of scrum connect. It helps the team connect so that everyone is on the same page and it is clear what has to be done by each of the member in the team. Team members can also tell about the issues they are facing while working and if anyone in the team has prior experience to the problem, they can help each other. This might seem like a small thing, but is a huge help. It reduces the overall development time and team is able to leverage each other’s experience and all has been made possible because of the following the process.

I realized the importance of these practices with one of the incidents that happened during our project. During the initial setup of the project two of our team members ended up doing the same thing, this was because of the communication gap. Although we lost only half a day, but we learnt an important lesson that it is important to be on the same page with the team to boost productivity, save time. Also, some of the members in our team were experienced, and some were not. Only by knowing that our peers are facing an issue, we could help them and reached the resolution swiftly which helped us immensely in completing the committed project on time.

Apart from these, the lifecycle helps in build quality projects. Be it any lifecycle, waterfall, Agile, iterative. Different lifecycles can be employed in different kinds of domain and they have been proven effective to solve the problem. The different phases of lifecycle ensures that team analyses, develops, unit test, walkthroughs and test the code which reduces the chances of bugs and makes the code more maintainable. I have realized the importance of unit testing among the code. After sometime, when you come back you are bound to forget why you wrote what you wrote in code, the unit test are boon. It helps clarifying the purpose of a piece of code which again helps saving the time and lets you focus on the current problem.

##### What are the advantages and drawbacks of software engineering?

Solution:

Advantages:

* As already discussed, it helps in improving the productivity by streamline the set of procedures to follow and reduces confusion among peers.
* Helps in making quality projects by reducing the chances of bugs in the software.
* The different practices followed by the team members makes the code maintainable in long run and easy to adapt to the new requirements.
* The constant feedback from the peers and customers holds the developer to high standards and reduces the chances of project cancellation

Disadvantages:

* The only disadvantage that I can think of is that there is no one model that fits all project. Different lifecycle work with different projects and has to be customized according to the needs and requirement of project.
* It does put some restriction on the developers, but I think its still a good thing. It better to have some restrictions and produce quality code, which doesn’t lead to loss of privacy, security and lives.

##### What is the goal of Software Engineering? In which domain Software Engineering might be useful? Do you think that Software Engineering should really exist?

Solution:

Software engineering practices have been designed keeping in mind that the code developed to solve the problem must be

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

Today, the business needs of the customer are constantly evolving, and hence the software they employ have to evolve with those needs. Making sure that the code written to fulfill the current needs has above characteristics helps in the future enhancements.

Although these goals are straightforward these are very hard to obtain. And hence we need software engineering practices in every domain. Every domain is unique and requires different kind of practices. For example, the domains in which requirements seldom changes can use waterfall model and the domain in which the requirements are frequently changing can employ an agile mode but irrespective of the domain we are working on, software engineering practices makes the lives of the developers and stakeholders easier as the seemingly huge task gets divided in small parts and can be solved independent of the other problems. Hence in my opinion software engineering is a need and requirement of all the domains.

If we look at the software engineering history and its origin, we can find that it began as a result of software-crisis. Many projects were running over time and over budget. A few projects led to the death of civilians and everything was linked back to the quality of the code. Moreover, as the size of the projects increases, the code became messy and teams were not able to maintain and hence began the research to find the practices which can lead to the quality code that were reliable. And since then, the researchers have developed many lifecycle models and practices, following which we have seen the improvement in the quality of the projects and hence, I think, the software engineering practices put some restricts on the developers and though the rules and regulations are enemy of the developers they should still absolutely exist to produce quality projects.

1. **What are the different possible roles in theory of software engineers? In practice?**

Solution

The names of the roles and positions might be different in different organizations and in different lifecycles. But the major roles in software engineering are:

1. Business Analyst: These are the engineers that talk to customer and gather their requirements, they also make an educated guess if the requirements given by the customer are feasible or not.
2. Team Leads: They are responsible for establishing good practices among the team members and provides them guidance and coaching and clears all the doubts. They are responsible to upload the quality and standards as laid by the organization.
3. Developers: They are responsible for implementing the code of the problem they have been assigned. They also resolve the bugs in case any of them are reported by the testing team.
4. Testers: They are responsible for testing the implemented solution against the requirement given to them. In case they find any issues
5. Deployment Team: they are responsible to deploy the project in the real environment to be used by the real users of the application.
6. **What is the difference between software engineers and software developers?**

Solution:

Software engineers think about the big picture. They follow engineering processes and procedures to analyses, design, test, develop and maintain a software. Software engineers realize the value of second opinion and they always listen to the team members. They solve the issues of the big picture systematically rather than just solving the small problem. They have to think about the security, privacy and that the non-functional requirements of the customer are also met. In other words, they design the blueprint of software, they divide the bigger problem into smaller problems so that the project becomes manageable.

On the other hand, software developers are responsible for developing and solving a small problem they are assigned. They are required to do everything that an engineer does but they apply the same principles on a much smaller scale.

If we look at the project that we designed during our term, I think we all were engineers, when we were designing the architecture and deciding how the communication process should look like, how data should flow, what technology stack would be most beneficial in longer run, what non-functional requirements are a priority for the customer and what documents would help us in longer run. We were engineers when we negotiated with customer regarding our requirements, convincing him why implementing something was not viable in the longer run. And we became developers, when we all divided the task among ourselves and worked towards fulfilling just one requirement. But we did employ all the process and procedures, kept in mind the privacy and security of the data. But in that moment we were only thinking about the one task not the whole picture.