COMP47480 Contemporary Software Development

Lab Journal

Shourya Khujneri (22204653)

MSc. in Computer Science (Negotiated Learning)

UCD School of Computer Science

University College Dublin

Table of Contents

1 Agile Methods

This practical involved a group activity wherein students were supposed to get into groups of two or three and reflect on the topics covered during the lecture sessions. The theme of this particular practical was to come up with answers to the question “What criteria help a software project succeed?”.

1.1 Work Done

The activity was divided into 3 phases. The first phase was the Brainstorming phase where I along with 2 other group members discussed various criteria that help a software project succeed. We came up with 13 criteria that we thought were imperative for a software project to succeed. The second Phase involved choosing the top eight criteria from the initial list that we thought were the most important. The third Phase involved a detailed discussion with other team members on whether the criteria chosen by us were Agile compliant or not and if they were then we needed to justify it by writing a short note for all the criteria.

1.2 Reflections

I found the activity quite interesting as it involved a discussion with other students and even though I was aware of various software development methodologies such as Agile and Waterfall, I still got to learn something new about them. For example, I didn’t know much about the code review process but during the discussion, I understood that it is a process in software development where one or more developers review the code written by another developer before it is integrated into the main codebase with the goal of making the software high quality, reliable, maintainable and free of bugs. Code reviews are a great way to improve the quality of the codebase while also learning and improving your skills from fellow developers.

While we were discussing different criteria based on different software development approaches, we also discussed a bit about DevOps. DevOps is a software development process that emphasizes collaboration and communication between development and operations teams. The goal of DevOps is to improve the speed and reliability of software delivery by automating key processes and breaking down the traditional silos between development and operations. Even though DevOps wasn’t covered yet during the lectures but the concept was quite fascinating to me so I did read about it before the practical. Being aware of another software development methodology turned out in my favor as I was able to suggest a few different criteria during our discussion.

During our discussion, I also realized that choosing different criteria for the software development project to succeed depends a lot on a number of factors such as the size and complexity of the project, the team's experience and skill level, the desired speed of delivery, and the organizational culture. A lot of thought is given to deciding the best criteria using the right methodology. We came to the conclusion that while there are a lot of software development methodologies out there, we cannot apply the same criteria or software methodology to all projects. We must adapt to different approaches to developing software and even modify the existing approaches if needed to our own advantage because, at the core, we all share the same goal which is to develop state-of-art quality software.

I believe that this practical session was an eye-opener for me as I learnt a lot about the thought processes that are required while working on projects to deliver high-quality software. The session also helped in providing a simulation of how different development teams go about developing software for

their clients. The key, in my opinion, is effective communication and collaboration among the team members developing the software and the stakeholders who will be using the software.

2 UML Modelling

This practical involved a group activity wherein students were supposed to get into groups of two or three and reflect on the topics covered during the lecture sessions. The theme of this particular practical was based on UML Modelling.

2.1 Work Done

The activity involved students to perform some simple modelling using the three main models that were discussed during the lectures namely the use case model, class diagrams and sequence diagrams. All the students were provided with three problem statements that each involved using and developing the above-discussed models.

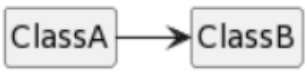
2.2 Reflections

The activity was quite insightful as it involved working with different UML models. Usually, I like to study the topic of the practical before performing it and during my research, I found out that UML models can be categorized into two main types namely static and dynamic. I understood that Classes, objects, interfaces, and their connections are displayed in static modelling. Both the class diagram and the object diagram are UML-based static modelling techniques. On the other hand, Dynamic modelling shows the behaviour of the static aspects of a system. This type of modelling can also be used to show a workflow, the passage of time, and different states. In this type of modelling, something will be changing, whereas in static nothing is changing.

I also got to know about structural and behavioural UML models. Structural diagrams depict a static view or structure of a system. It is widely used in the documentation of software architecture while presenting the outline for the system. Whereas Behavioural diagrams portray a dynamic view of a system or the behaviour of a system, which describes the functioning of the system. It defines the interaction within the system. Understanding these models enhanced my thinking ability while I was building different models during the activity.

In Addition to that, while we were working on developing the models, I got a little curious regarding the different types of lines that were used in the class diagram models and with some help from the teaching assistant (T.A) and StackOverflow I understood the following: -

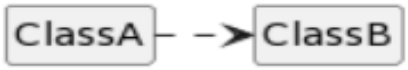
1) The solid lines show an association



This would be like Class A storing a reference to Class B as an attribute. The code snippet below explains the above point clearly.

```
public class ClassA {
    ClassB theClassB = ...
    ...
}
```

2) The dotted lines show Dependency

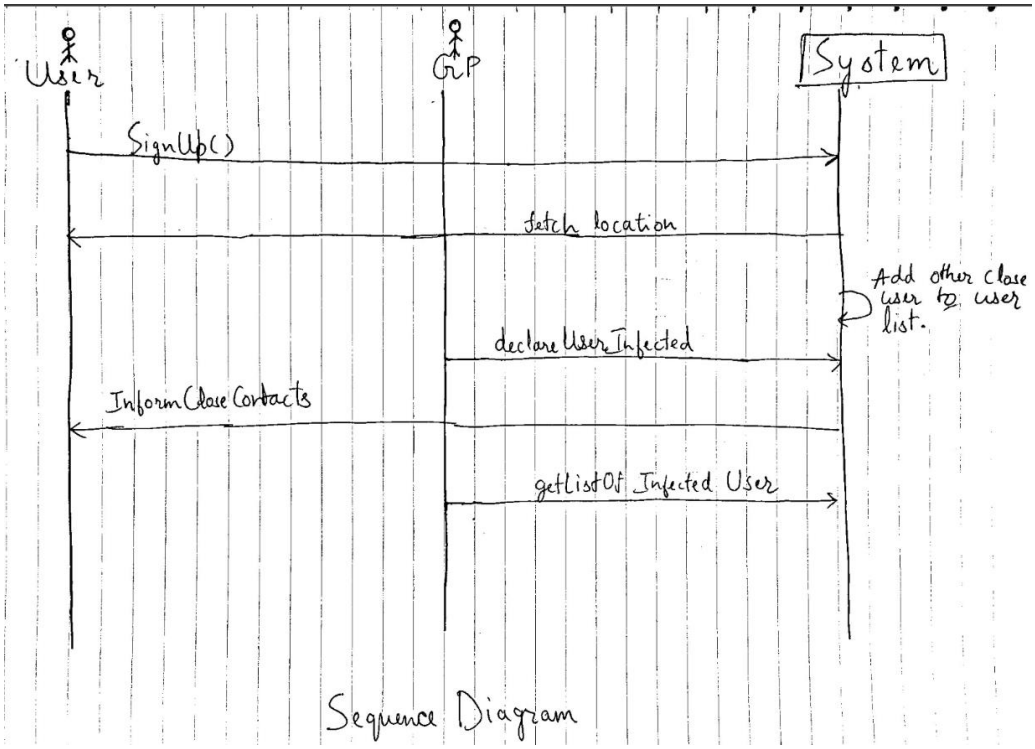


Dependency is much weaker than an association. They exist for a variety of reasons such as a class could depend on another class instance variables to perform some operation for instance the salary class can depend on the Bonus class instance variables in order to calculate the final salary of the employee. The below code snippet can help in making the above point clear.

```
public class ClassA {
    ...
    public void someMethod(ClassB arg1) {...}
    ...
    public void someOtherMethod() {
        ClassB localReferenceToClassB = ...
    }
    ...
}
```

The above code snippets have been referenced from StackOverflow.

Another highlight for me was the sequence diagrams as I didn't have any prior understanding of them but during the practical, I understood that sequence diagrams are used to model interactions between objects in a system. They visually represent the sequence of messages or method calls exchanged between the objects in a specific scenario or use case.



In the sequence diagram mentioned above, objects are represented as vertical lines, also known as lifelines, and messages are represented as horizontal arrows. The arrows indicate the flow of messages between objects in the sequence of steps or events of the scenario being modelled.

Sequence diagrams can also show the order of message exchanges and the duration of each message, including its start and end points. They are often used to model the behaviour of a system from the perspective of a user or actor and can be helpful for designing, documenting, and understanding complex systems and their interactions.

Overall, sequence diagrams are a valuable tool for software developers and designers, as they help to visually represent the interactions between objects and illustrate the behaviour of a system in a specific scenario.

The above activity helped in providing a simulation of how software developers go about developing a software system which in this case was a covid tracker system. This activity helped me in applying the UML Modelling skills learnt during lectures to develop real-world applications. After the completion of the activity, I feel that I am in a much better position to approach the problems encountered while developing a software system in a structured and systematic way. To sum up, This activity also helped me in understanding the crucial role that UML Diagrams play while developing software which is to have clarity among various team members regarding the software system.