Comp 47480 Learning Journal – Assignment 1

1. Reflection and Account on Team

First, we split our team into the customer and the developer groups. Myself and Jia were the customers and Carl and Neha were the developers in the first iteration. Myself and Jia began to specify our user stories. Whilst we defined the user stories the developers took each card and estimated how long it would take to implement. In hindsight I feel that some of the user stories we defined could have been improved upon. The implementation of the user stories we left down to the developer which I think is a requirement of the process. However, for some features and user stories it may be advantageous to seek clarification on the location of the features within the fridge. I realised this after we defined that the fridge should have a light that comes on when the door opens. The developer implemented the light at the bottom of the fridge, but it may have been better to be located closer to eye level. Overall in the first phase all the stories that the customers asked for were implemented. In the next iteration we switched roles. In this iteration the customers decided to update small features for instance having a second door for the freezer area and a second light so there would be one in the fridge and one in the freezer. Again, on this iteration it was noted that the design that the customers had compared to the developer differed slightly. In that the developers implemented an internal freezer door, but the customer wanted an external freezer door. This highlights how important proper specification is on the part of the customer but also the onus is on the developer to think about the specification and all the issues, lack of clarity and different ideas each party can have on the design concept. Overall both the customers and developers were happy with the overall product made, the stories presented and the deadlines for the project were met. However, the most important aspects of the planning game are simplicity clarity and communication and as a group we could have designed a better product had communication been more open and more questions asked. It is also worth noting that this product was a simple design in that everyone knows what a fridge is and what it does however when you apply this to software development the product will not always be clear and more input from the customer would be needed to guide the development of the product.

2. Reflection on my Learning on the Planning Game

The planning game is a technique that can be used in the extreme programming model and is a significant element in mapping out the design of the product. It allows customers and developers to interact and communicate with each other in a way that is advantageous to the development process by allowing customers to provide an outline of the features they require and developers to estimate how long each part of the process will take. It accepts the fact that the design process is not finite, and features and functionalities are subject to

change. Subsequently, the idea is that the customer can prioritise functionalities and the developer can then correspond with the customers on how long each aspect will take and whether each is possible given the order the customer wants to have the design completed. In this way this process is iterative, and the design can start as a skeleton and very basic concept after each repetition of the cycle the progress and specification of the functionalities and features can be enhanced. This process iterates through many stage many times until the required product has been achieved.

The planning game plays an essential role in the creation of common goals by communicating ideas. It essential breaks the process into two main stages, the design stage and the iterative stage. As I observed the planning game, it became clearer that the design phase was the part of the planning game that was based from a collaborative point of view, to understand what the essential requirements of the project are, to ensure that you as a developer have an understanding the requests of the customer. This can be achieved by listening to the customers' requests and being informative and communicative about what is achievable in the timeframe of the iteration which usually lasts 2-3 weeks. An important element in this process is to ensure the customer specifies all the user stories. The developer only estimates the timeframe of each user story or reject user stories that are overly complex so the customer can split the story into smaller more manageable stories or ask for clarification on some aspects of the story. It is vital that the user stories consist of 1 clear sentence to eliminate confusion and complexity. During this the stories are prioritising by the customer and the developer and the customer commits to the design until the next meeting.

The iterative stage is the process that just involves the developer. This is the stage where the developer goes away for the build phase of the project and implements the stories that the customer has specified. They don't diverge from the customers design they implement the design as given to them. This may involve assigning tasks to persons in a team of programmers. When the build is achieved it is released and the next iteration occurs where aspects can be redesigned, new stories can be added and stories that aren't reaching the mark can be deleted.

This planning process is really all about cycles of development and communication and a willingness to keep the design flexible so changes in requirements can be embraced by adapting the design at the end of every cycle. Due to it being a part of the extreme programming model in which testing is seen as imperative to a project so it is one of the first things done errors can also be addresses in a timelier manner. The downfall for me in this model is how many iterations must be completed before the finished product is produced. It seems like that the customer could redefine and create more user stories to implement in the project indefinitely.

Comp 47480 Learning Journal – Assignment 2

1. Reflection and Account on Team

During the practical our team first started the assignment by addressing the use case model this was an important part to defining what the library would do. Whilst reading through the instruction we picked out that there were 2 types of member that would be able to access the system. The student and staff would be a generalisation of the model whilst every member could borrow and return books only staff could borrow journals and to return either then there is a dependency that a book would be borrowed in the first place. I found this one the most difficult as I wanted to be able to define more things the library could do. I found it difficult to hone in on just the basics. We then moved forward to the class diagram which we have used more and used to show that the library consisted of items which were either of type book or journal and members that were either of type student or staff in this diagram we were able to define attributes that each class might have for instance a time period due to the fact that we could take out a book for 4 weeks and others were on short term loan. This for the team was relatively straightforward. Subsequently the sequence diagram could be addressed. This diagram was to show an action that might happen within a class on a timeline. The first sequence we decided upon was to borrow an item it the first sequence the system checks that the member has not reached their limit of items they can borrow. The second sequence that we had to design was to send an alert to the user when their book was out of time and needed to be renewed. This functionality required an interface to send the requests to the correct class (as seen in the artefact). In this the system runs a query on all the items in the library and checks the time borrowed against the last borrowed date if it is over the time allowed it then finds the user who has the book out and alerts the member.

2. Reflection on my Learning on UML

After attending the lectures and the practical given about UML, it has highlighted to me my knowledge of UML and the its different models has been quite limited to this point. Previously, I had used UML to depict class diagrams and had little knowledge of the other diagrams that UML encompassed. I was aware that UML could be useful tool in many industries in providing a methodology of how to approach constructing a mock-up of the system that team is implementing and providing a tangible visualization of how the system will work or what the system design will do to senior management that may not understand the system fully if the team was to describe the system by using technical jargon alone. UML is a graphical language used to model systems using

relationships of the components and the dependencies between these components. I have learned that while I have only really used the class diagram. The unified modelling language consists of use case diagrams, class diagrams, sequence diagrams, collaboration diagrams, state chart diagrams, activity diagrams, component diagrams, deployment diagrams. The use case diagram is probably the simplest form of diagram as it just represents what the system can do. It doesn't provide implementation however it does consider dependency of features. It can help in deciding what features are necessary in the system. The domain model /class model is one that I have previously used this I find provides a useful first mock-up of the system by conceptualizing the essential classes and features. It makes use of relationship between components which can be described in tree ways association aggregation and generalization A generalization occurs when there is a class that inherits functionality from the super class. An association is when there is some affiliation between classes and aggregation is when a class belongs to a collection. Arrows provide and idea of the direction of the association and we can provide a way of showing the number of possible instances of the class by providing multiplicities like zero or one instance by using 0.1 notation. The sequence diagram shows how the model interacts a with the classes in terms of the operations. These are read top to bottom not left to right as they are depicted like a type of timeline. Collaboration diagrams also provide a representation of interactions however it is focuses mainly on the role of the objects. I feel this diagram is more useful as an interaction diagram and I would lean towards using this as I feel it's closer to the class/domain models that I'm comfortable illustrating, so this feels like a good way of providing some modelling of interactions.

Personally, I find some of the UML diagrams simplistic and find it difficult to depict some of the diagrams because I tend to lean towards putting too much of the implementation and complexity into the diagrams where simplicity is necessary. I would much prefer to mock up the bigger picture of the system than to simplify it, to the point that I feel the diagrams can become too simplistic and insignificant to warrant illustration. Although UML is not continually utilized in industry It remains a useful tool where the abstraction of the model is necessary to maintain common system goals and design concepts between programmers where systems can become complex and difficult to understand. It also provides a common language that the everyday person can understand due to its graphical nature. I feel through the course of this practical I have gained a better understanding of UML for demonstrating the model of a system.