# **Entry Two*:*** Modelling With UML

Student Name: Gregory Sloggett

Student Number: 14522247

## **Work Done:**

We first established our use case model for the system. We identified the student members and the staff members as the actors of the system, as a library member is too vague to be considered an actor. We also assume that students and staff make up the population of library members given that it is a university library. We used the actor generalisation notation to formalise the use cases for the two main actors, in terms of the library member. We also determined simple cases to show the how "includes" and "extends may be used within the system. This took some time to complete this first stage of the practical, as our understanding of the terms was limited.

We moved onto the domain model and immediately set about using the noun identification technique. We determined the following nouns: University, library, book, journal, library member, student member, staff member, system, copy (of book). We concluded that the following would determine our classes within the domain model: book, journal, copy (of book), student member & staff member. We determined the relationships between these five classes and depicted a domain model as such.

We finally dealt with the interaction model, by which we agreed that there were several sequence diagrams that we needed to draw. The obvious two were the cases whereby a library member would borrow a copy of a book, and a staff member would borrow a journal. We also agreed we needed to have a distinctive sequence diagram for returning books and journals to the library.

Finally, we refactored each of the steps to include the new requirement as specified in part four of the question. This was tricky to find the appropriate spot in each of the diagrams we had done previous, and we were uncertain about how drastically the change could and should affect model.

## **Reflection:**

I had very little experience and knowledge of the principle UML diagram types before taking on this exercise, so I consider it to have been a very steep learning curve, and somewhat beneficial. Obviously, as Mel mentioned, UML is very rarely used in practice, but from the past couple of weeks spent on the topic, I view it as quite a useful method to show the workings of a software system. Sometimes it can be tricky to grasp the overall architecture of a system, especially if you're working with an extremely large application or piece of software with thousands of lines of code. The concept of UML is one I believe would help a new employee or someone viewing the software code, to gain a quicker and more rounded understanding of the system structure, behaviour and overall architecture.

When we started the use case modelling in part one of the practical, we decided that the actor generalisation notation was the best way to formalise the use cases for the actors in the system. In this way, we were able to give a clear depiction of how both the student and staff members inherit all functionality to that of the library member, with the staff members having the added ability of being able to borrow a journal. This added functionality stems directly off the staff member, thus showing that it is a capability of the staff, but an action that eludes the students.

I found the concepts of includes and extends difficult to understand. For includes, I found the example of the flow of events that occurs at the beginning of every interaction with an ATM as a good example of how it works. The user places in their ATM card, enters their PIN, and is shown the main menu. The ATM card requires the entry of a PIN on input to the ATM, meaning it is not complete without the “included” PIN, although it can standalone. The “included” PIN cannot standalone however, as it is a result of the ATM card. This depicts a perfect and obvious real-world example of “includes” and helped me to understand how the includes term is interpreted and used.

My thought process upon writing this would then assume - given that the use case prior to an extends call can stand alone – that the touch debit cards used for paying at a till would have an “extends” PIN in its use case diagram, meaning that the ability to pay using a PIN is there under the extends link, but this is not a requirement, and the debit card can act alone without use of the PIN.

The modelling of the classes and their relationships I found perhaps the easiest concept to grasp and understand. The noun identification technique was quite straightforward and gave us an easy and direct route to establishing the main classes. From there by following the notation in the notes and using common sense to interpret the relationships, we established the class diagrams.

I felt the sequence diagrams were somewhat already depicted in a sense by the use case diagrams. The main difference is that the sequence diagram adds that sense of time to the concept. Perhaps it’s just the simplicity of the example that we were dealing with here that gave me the feeling of the sequence diagram being of less importance than to the use case, or that both aren’t required as long as one gives coverage, but it did feel as though the sequence diagram was easily interpreted by the use case. However, as I said, the added sense of time and route of the functions is important and in a more complex system this would probably be more evident to me.

Overall, this was a very informative introduction to the world of UML, and hopefully it will stand to me as things progress in my career. I know that some of my peers went for interviews in WorkDay, and one of the largest parts of the technical interview was associated with UML and how it is used. Even despite it’s infrequent use in the workplace, I have no doubt that it will pop up and become a useful piece of knowledge to be equipped with later down the line.