THE UNIVERSITY OF BRITISH COLUMBIA

Department of Electrical & Computer Engineering

CPEN 333 – Systems Software Engineering

Assignment 2 – Using UML - November 2019

Part A – Capturing requirements with a USE-Case Diagram

In Lecture 20/21, you were introduced to the concept of Use Case diagrams as a means to graphically capture important functional requirements of a system and document them from the perspective of the user and what functionality they need from the system,

Imagine we have been given the task of writing software to computerise the traditional paper-records approach of running a University. Think about what things a user (e.g. Prof, Admin person or student) would want to be able to do. Some suggestions are listed below, but you could add/amend these.

1. Enrole in the university and register for a program
2. Registers for courses,
3. Submit a set of course grades.
4. Submit a change of grade for a student
5. Graduate a student from year 1 to year 2 status etc
6. Graduate at the end of the program
7. Download your transcript

Think about the university rules and regulations that determine the success or otherwise of these requirements. Who would you interview to discover how each of these processes work (who are the domain experts).

Think about who the stakeholder are:- Students, Profs, Senate, Faculty Admin and think about what they want from the system. Now draw as complete, detailed and thorough a use-case diagram as possible using Visual Paradigm to capture these use-cases and identify any APPROPRIATE relationships between them. You should fully document your use-cases along with any interesting scenarios and full descriptions of any items of data. For example

[Student ID = Number, 8 digits]
[Name = String, 24 chars]

Also document any algorithms or equations etc. relevant to the business operations of the University, e.g. students can enrole and register in APSC program if they have an average of 94% in Grade 12 English, Math, Physics, Chemistry. Other faculties may set different requirements.

Think about who the primary uses are i.e. those who initiate the action, and secondary users who happen to be involved in it, i.e. there is some interaction required of them.

Remember your focus at this stage is about documenting "**what**" will happen, rather than "**how**" it will happen in the finished system. There should be no technical stuff in here, you should be able to show your use-case model (including descriptions) to someone at the university and he/she should be able to understand it and sign off on it.

Marks will be awarded for the completeness of use-case descriptions, data, any relevant business operational rules and the correct and appropriate use of the UML diagram syntax and relationships – there should be clear evidence that each of your use-cases passes the 3 tests required of any use-case.

1) It documents a step by step interaction between the user and the system.
2) It documents the benefit to the user and
3) It documents the effect on the system of that interaction **(20 marks)**

Part B – Capturing Use-case behaviour using a sequence diagram

Pick any 4 interesting and non-trivial use cases (trivial ones will most likely earn you less marks) from Part A above and using the approaches identified in Lecture 22 and 22a (Architectural Modelling and Object Identification Techniques) identify a set of objects (using nouns analysis) from which your system will be composed.

Now think about each objects roles and responsibilities. What is its purpose in the system? What actions can other objects ask it to perform? For example in the case of a linked list object, its role might in the system be to store and retrieve data on behalf of other objects. To realise this role, the designer of the linked list might provide insert(), delete() and find() functions which are actions other objects can ask it to perform.

Now, using the material of lecture 23, draw a separate sequence for each of the 4 use cases. These should clearly show the user (i.e. who initiates the use case), a set of collaborating objects involved in realising the use-case and a set of messages that will propagate between these objects as they attempt to implement the use-case. A message implies one object asking another to perform a task or carry out an action related to its roles or responsibilities in the system. It follows from this that messages MUST therefore have a **verb** in their name.

You should also document the sequence diagram with "**stick it**" notes, appropriate "**alt**", "**opt**" type frames and/or **comments** down the left hand side of the diagram to show if/then/else type decisions.

Some messages require parameters, others don't, so document appropriately. In addition consider the type of message that is appropriate, e.g. **synchronous** or **asynchronous** and draw appropriately.

(20 marks)

Part C – Classes, Class diagrams and class relationships

In the last two slides of lecture 23 you were shown an elaborate sequence diagram that captures the behaviour of a set of collaborating objects as they attempt to realise a use-case description of what happens to a car when it is serviced.

Collaboration between objects means that one object sends messages to others, which in turn implies that an **association** relationship exists between sender and receiver objects, that is, they work together and rely on each other.

For this part of your assignment, you are asked to identify and capture a set of classes on a UML class diagram in Visual Paradigm (derived from the objects on the sequence diagram). For each class, identify the member functions. These will come from the messages, parameters and returned data that are sent to/from the object (see example on say page 11/12 of lecture 24/25).

Secondly identify from that sequence diagrams, the relationships between these objects and also the multiplicities involved and capture those on your class diagram. Think about other potential object relationships such as 'aggregation', 'composition' and 'kind-of/inheritance' that might be present in your model (I'm not saying you have to use them all, but see if you can identify some where it makes sense, but only where it makes sense)

(20 marks)

Part D – Mapping to Code

Finally turn the class and sequence diagrams of Part C) above into a C++ application by writing some 'harness' code to create instances of the relevant classes, (e.g. technicians, receptionists, etc.) and initialise at run-time the associations between those objects and thus simulate a customer walking into a garage to have their car serviced

For each class function that implements a message on your sequence diagram it will be helpful if you can also include some form of print statement inside to put out some suitable 'text' on the screen, so that at run-time it is easy to trace the execution of the program and, where it is required, get the function to be as realistic as possible (for example calculate an appropriate formula, or implement some suitable algorithm etc) and return appropriate data back to the source of the message.

Obviously the receptionists and technicians are not real people, they are objects captured in classes and behave ONLY as per your sequence diagram meaning that they don't take coffee breaks, go on strike or claim overtime etc. Use some creative latitude in translating the messages and concepts shown on your sequence diagrams into functions and code. Don't use active objects or implement any form of concurrency into the solution unless you feel confident.

(20 marks)

Part E

Automated Test Suites and use of GitHib repository and Version Control

There are two parts to this. Firstly, read Topic 5 lecture notes on Testing located on Canvas. Now complete the following

1. For Part D of the assignment above pick on the ServiceCar() function that is part of the Technician class. Write a set of automated test case (using your own test data) to check the correct operating of this. Use Assertions in your code to check for valid/invalid data and also to produce pass/fail criteria for ServiceCar().Of course this assumes that the classes and functions used by ServiceCar() also exist.

   Show your test cases to the TA and demo them running. **(10%)**

2. In addition, demonstrate your use of GitHub for maintaining version control of the software between you and your partner (if you didn't have a partner for the assignment – create a second GitHub user and simulate having a partner – i.e. checking out some code, making changes and committing those changes. **(10%)**

Deadline: Demo and submit in your Normal lab section in the last week of term (See Canvas Calendar). You should use the remaining lab periods to work on your assignment

Group Work:

This assignment may be done in groups of NO MORE THAN 2, or, if you prefer, individually. Do not ask to work in a group of 3 or more. All students who wish to be awarded marks for their work must attend the demonstration and expect to be asked questions about their design. If you cannot find a partner, then ask and I will announce it in the class and try to hook you up with someone.

All students within a group will be awarded the same mark i.e. that awarded for the assignment itself, unless there is clear disagreement about the amount of work each person did. If this cannot be resolved amongst the students themselves, then the TA will interview the students to determine their knowledge of the work they are submitting and award individual marks on that basis alone. The TAs decision is **final**. If you don't turn up you don't get a mark regardless of whether or not your name is on a submission.

SUBMISSION OF WORK and Assessment Strategy

YOUR DESIGN AND ALL SUPPORTING WORK MUST BE SUBMITTED VIA THE CANVAS DROP BOX.

YOUR NAME(S) AND STUDENT IDS MUST BE CLEARLY INDICATED ON YOUR SUBMISSION.

THE FINAL DECISION ABOUT WHAT IS A BETTER DESIGN RESTS WITH THE INSTRUCTOR AND/OR TA. THEIR WORD AND DISCRETION ARE FINAL. IT SHOULD BE NOTED THAT DESIGN IS NOT AN EXACT SCIENCE AND EVERYONES SOLUTION WILL BE SOMEWHAT DIFFERENT. IT FOLLOWS FROM THIS THAT NEITHER IS MARKING SUCH A DESIGN AN EXACT SCIENCE.

THE MARKING PROCESS WILL THEREFORE NOT BE AN ABSOLUTE MEASURE OF ANYTHING, RATHER DEMONSTRATIONS WILL BE AWARDED MARKS RELATIVE TO THE PERFORMANCE OF OTHERS IN THE GROUP, IN SIMPLE TERMS, BETTER ASSIGNMENTS WILL ATTRACT BETTER MARKS, IF YOUR ASSIGNMENT IS INCOMPLETE, CONTAINS BUGS OR IS NOT EFFICIENT, THEN DO NOT EXPECT TO GET THE SAME MARK AS SOMEONE ELSE'S PROGRAM THAT DOES NOT SUFFER FROM THESE LIMITATIONS.