Week 2 Assignment – UML Design Modeling

Maria Reyes

The University of Arizona Global Campus

CST 499 Capstone for Computer Software Technology

Professor Charmelia Butler

October 2, 2023

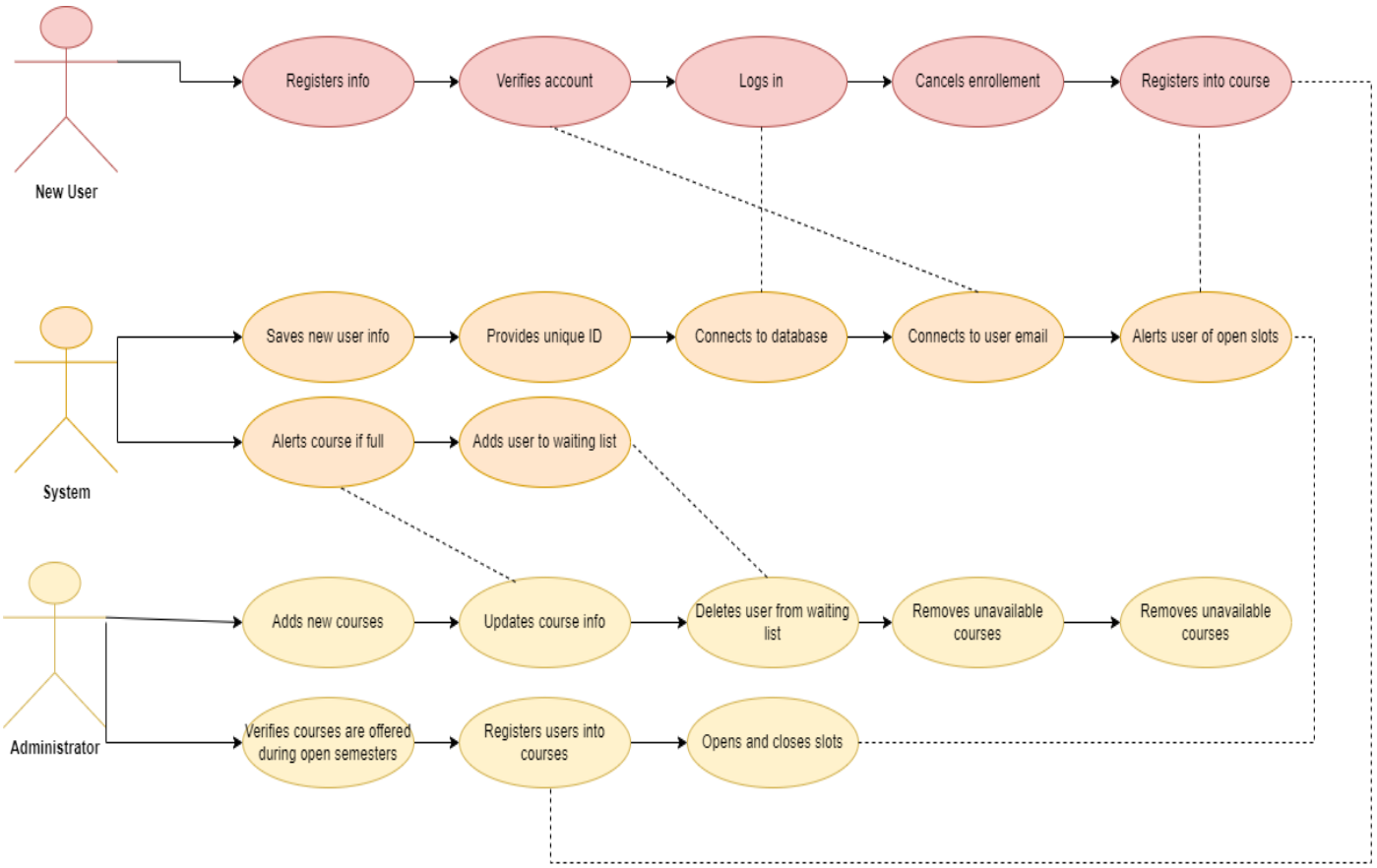
UML Design Modeling

An important process in software design requirements is creating UML design modeling. These can be compared to ideas being brainstormed into structured and well-informed diagrams. Instead of connecting bubbles, these diagrams offer a more organized foundation and decipher all the necessary components that a software system build should have. In accordance, the project that will be translated into UML design modeling diagrams from requirements is the enrollment online system build. In this paper, the modeling diagrams will be explained in detail and the different levels of testing will be discussed that software testers can apply from the UML diagrams such as: component testing (or unit testing), integration testing, system testing, and acceptance testing.

The first UML design model that will be explained is the Use-Case diagram below. Within this software build, the three active actors that will interact with the system will be the new user, the system, and the administrator. After the new user enters their information and verifies their account, they are registered into the system. Once they are registered, its job is to log in, register into courses, and cancel enrollment when desired. On the other hand, the system records all the new users' actions by saving their information into the database and connecting to their email in order to send notifications about open slots for courses and know their status in the waiting list. Accordingly, the administrator will perform actions that the system will also record such as updating any course information such as their availability. This, in turn allows the course list to be up-to-date. With that information, the new user (or student) can register into a course and the administrator will do the job of signing them in.

Figure 1

Use-Case Diagram



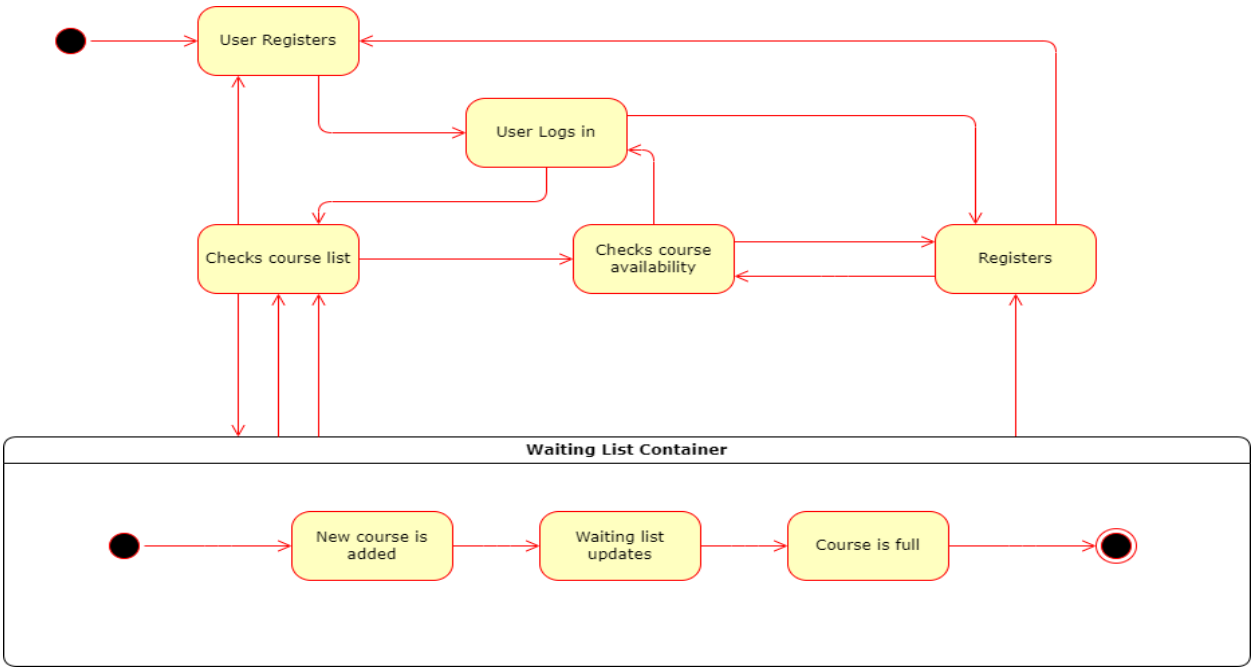
Note. The three actors in this system would be the new user, the system, and the administrator. Each of them performs different use cases and it is documented according to their appropriate actions.

The second UML design model that shows further detail for this software build is a State diagram, which is shown below. Tsui et al., (2018) explains this type of modeling: “In many cases, objects can be in different states, and it is important to model those states and how they are allowed to change” (p. 155). When applying it to the enrollment online system, it demonstrated how each state of objects is affected by different actions or triggers. The object in this case is the

new user that registers and logs in. In the waiting list container, their actions are further emphasized by interacting with the users' actions. If the course if full, then the user's actions ends. From there, the process starts again when the student logs in. Depending on the course availability, the student sign in or they will be added to the waiting list. In all, the states are constantly be changed when the states are being triggered by the waiting list actions.

Figure 2

State Diagram



Note. The user interacts with the system by registering and checking if the courses are available. If they are available, then the user registers. If they are not, then they are transferred to the waiting list container. The waiting list adds the new user. Since it is full, the process ends there.

The last UML model to further explain for this system is the Class diagram illustrated in Figure 3. The specific classes that should be documented are the new user, log in feature, online

course list, and the waiting list. The purpose of each class is to provide “group objects with similar structure and also a set of similar objects” (Tsui et al. 2018, p. 152). This is very important to illustrate because how the different classes relate to each other will a provide a clear picture of their functions and attributes. What the new user will enter into the system is their information, as well as create a password. The functions will assimilate to the attributes, along with providing a unique ID that only the user will possess. The password and the unique ID will be used for the log in feature. In accordance, the log in feature enables the user to enter a verification code when entering their credentials as a way to secure their information. To check, the log in feature will connect to the database to make sure that their information is correct. If it is, then the will be moved on to the online course list, which is the next class. This class will perform preserving all the courses information and updating them. The last class for this software build is the waiting list. This is an important feature because it records all the students who are waiting for courses to open up. It also notifies the student about recent changes about their status in connection to the course. Administrators can also use this feature to take note of all the students who are added to the list and record any audits for school purposes.

Now, these models are essential in software testing. They can be used to check all the requirements and detect the presence of defects within the system build. Software testers apply the following levels of testing: “unit [component], integration, system, and acceptance tests, where unit tests are performed on a lower level of system abstraction, e.g. on system components, and acceptance tests on a higher level of abstraction, e.g. the graphical user interface (GUI)” (Liebel et al., 2019, p. 17). Hence, it is important to start from the first level to the last level, which will identify any errors independent from others or together as a whole. Depending on the severity of the errors located, then the software build requirements might have

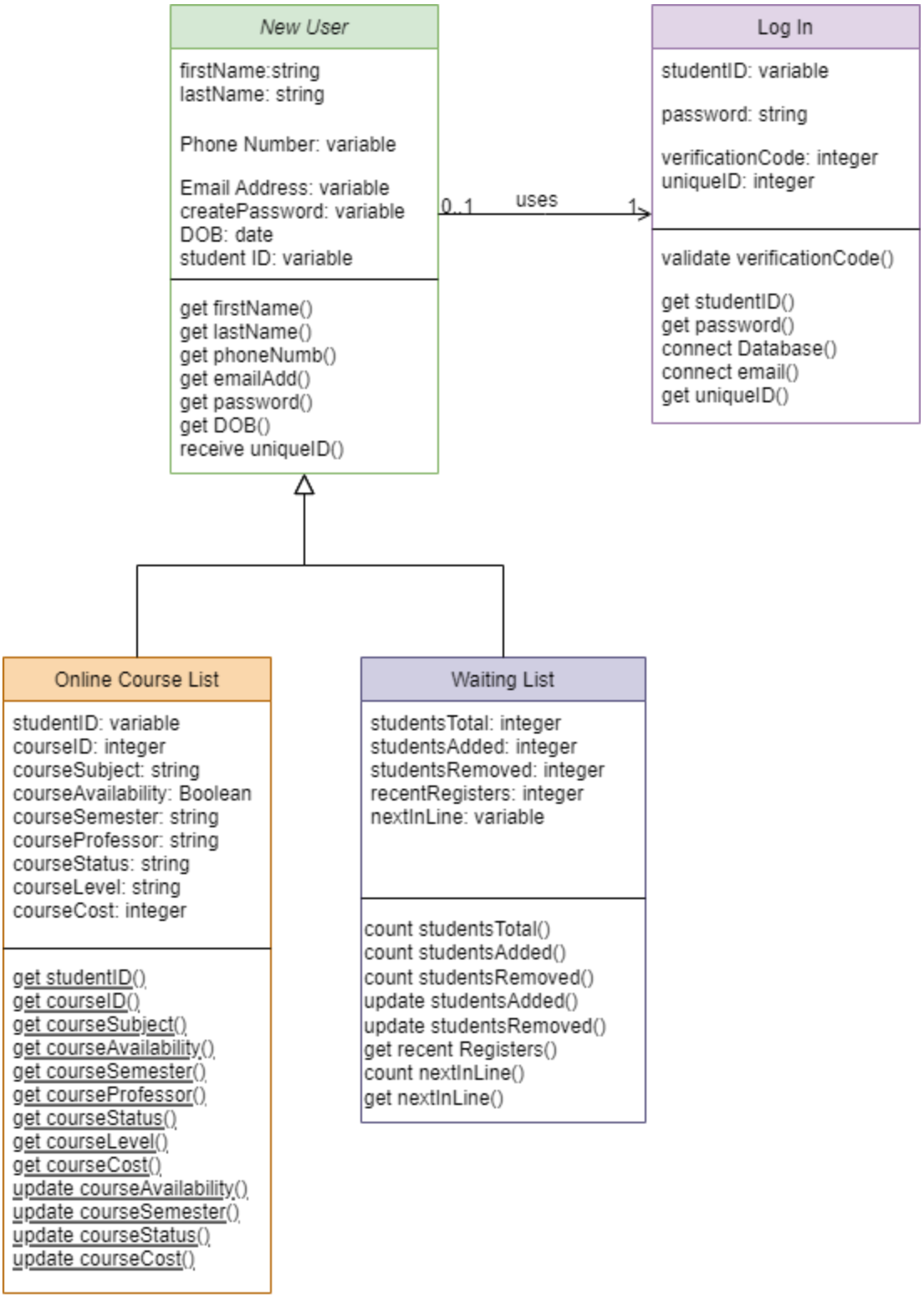
to revised. Unit testing or component testing is typically done by software developers. What they look for is the indication that errors are present within the inner components individually. In integration testing, on the other hand, certifies that "an error-free flow of control and data (such as consistency of parameters, file format, and so on) among combined units and their overall correct design and integration" (Mubarak, 2020, p. 3). In other words, the components are now combined in order to verify that their integration is running smoothly or not. This is when testers intervene. Something to note is that unit testing is done "by a technician who has the full knowledge of the code (usually the coder himself) whereas in Integration testing, the testing can be done by the tester with knowledge of the system's architecture or design or it can be done by the coder" (Brar & Kaur, 2015, p. 798). That is one difference that demonstrates who performs the testing. The third level of testing is system testing. For system testing, a team is now formed to test the whole integrated software system, such as its quality, its fulfillment in requirements, and other functional and non-functional requirements. The last level of testing is acceptance testing. Alpha testing and Beta testing take place in this level, which allow the future user to take part in to provide feedback. In essence, it "is done to ensure that the software does what the customer wants it to do and check the acceptability of the system" (Mubarak, 2020, p. 2020). When no errors were present in the system from the public's point of view, then the software build in headed to the right direction.

To summarize, there are a variety of UML design modeling diagrams that exist that can illustrate all the requirements in any software build. In the case of the enrollment online system, there were three UML design modeling diagrams that were used are a Use-Case diagram, State diagram, and Class diagram. They were utilized to show the actors involved, the classes and their attributes, and the state interactions. In connection, the different levels of testing were discussed.

By using all four (component, integration, system, and acceptance), the software testers can test if there are any errors present that will reflect the success of the enrollment online system. As a whole, it is the aim to fulfill the requirements and demonstrate the software build can be an effective program in the educational community.

Figure 3

Class Diagram



Note. The different classes that are depicted are the new user, the log in feature, the online course list, and the waiting list. Each receive their own attributes along with their functions.

References:

Brar, H. K., & Kaur, P. J. (2015). Differentiating Integration Testing and unit testing. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 796–798.

Liebel, G., Alegroth, E., & Feldt, R. (2013). State-of-Practice in GUI-based System and Acceptance Testing: An Industrial Multiple-Case Study. *2013 39th Euromicro Conference on Software Engineering & Advanced Applications*, 17–24. https://doi.org/10.1109/SEAA.2013.29

Mubarak A, U. (2020). Comprehensive study of software testing: Categories, levels, techniques, and types. *Research Gate. 1-10.* DOI:10.36227/techrxiv.12578714.v2

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Jones & Bartlett Learning.