Kevin Shields

5/26/22

ELEC3225 Professor Rawlins

Leopard Web Assignment 2 – Process Models

**Incremental Model**

**Block Diagram:**

Diagram

Description automatically generated

**Incremental model:**

The incremental model is a software development process that adds a lot of flexibility to development of a system. By splitting development into smaller, more incomplete versions before combining them into the final product means less commitment to the whole system at one time and being able to focus more on the finer details of each component.

**End product specification:**

Goal: This program is modeled after a Leopard Web type service. Students, instructors, and admins will be able to perform operations to create and manipulate course schedules over not just one, but several semesters. This program will utilize object-oriented programming to create a hierarchy of classes and objects, as well as databases to store and access data. This program will also implement a simple and easy to use UI (User interface) as its front end for ease of use.

**Initial Version:**

**Specification:**

The initial version of our incremental model will be a very bare-bones, terminal-based iteration of the scheduling program that simply demonstrates the functionality of the class hierarchy. A user should be able to create instances of our classes (Student, instructor, admin) and interact with them, testing their functions. These functions will not necessarily do anything meaningful just yet, but this is just a very low-level version of the final product.

**Development:**

This version of the software only requires one component to be programmed, that being a class hierarchy with a “User” base class with subclasses of “Student”, “Instructor”, and “Admin”.

* User
  + This is the base class of the system; the user class defines attributes needed for all 3 user types
    - First name, last name, and ID number
  + Any user should be able to view their attributes at anytime
* Students
  + Students should be able to perform a myriad of operations such as being able to add/drop classes from a list of available courses, as well as see their own schedule
* Instructors
  + Instructors should be able to see available courses, view their own rosters, as well as search for courses
* Administrators
  + Admins should be able to have an overview of the whole system , admins can add/remove courses and users from the system, as well as add/remove students to a specific course. Admins should also be able to search and print rosters and courses

As previously stated, functions in this iteration of these classes will not achieve much, most will simply print a message to the terminal stating that they were called properly and invoked on the proper object with the correct parameters.

**Validation:**

After all of the programming of the classes is completed. Several tests should be done in order to ensure the operationality of this version. Testing with a variety of inputs and seeing how the program reacts, calling each function and seeing through that the correct output is produced. This can include the developer themselves as well as the user who can then provide feedback to the developers for future iterations.

**Intermediate Version(s):**

**Specifications:**

With the class hierarchy programmed, the next iteration will implement the user database:

This system should be able to handle a database of 100 users, 10 instructors, and 1 administrator

* Testing of the software will be done with fewer users before it is ramped up to a full scale.
* The database will contain all the attributes of the users

**Development:**

The first step in the development process for this version is creating the database with our given restraints. Once this is done, Proper implementation of the database into the class hierarchy can be done. This will populate the class methods with meaningful blocks of code that will now perform functions relevant to our software goals. Our functions will now properly query the database for the information needed to preform each task.

**Validation:**

The validation for each version will be very similar. This step just requires a lot involved testing of the components to make sure everything is running as it should. For this iteration, testing should be done by the user as well as the developer in order to make sure the database integration is running as it should, querying attributes properly, functions in the class files are pulling from the right places in the database. This will still be a terminal/text-based software, lacking a UI, this is what will be addressed in the 3rd and final iteration.

**Final Version:**

**Specifications:**

Now that both the class hierarchy and the Database are implemented that leaves one last component to program and integrate, the User Interface.

The User interface should be dynamic and adaptive, depending on what type of user is logged in at any given time. For example, an instructor will have access to special menus that students do not, and administrators will have access to special menus that students and instructors alike do not have access to. Through this user interface, all users should not have to jump through any hoops to get where they need to be or achieve certain functions. In other words, the user interface will be simple and intuitive enough for the user to jump right in with no issues, but also have no shortage in complexity of tasks that can be completed. Students should be able to search for courses and add/drop them with ease, instructors should be able to print their class rosters with no trouble, and admins, who have the greatest number of actions they are able to take, should have everything they need in as few menus as possible.

**Development:**

Nowadays, most user interface development environments are fairly easy to use, after the user interface is designed and styled, the rest of the components can all be put together. In the second iteration of the software, the database was integrated into the class hierarchy, so all that’s left now is programming the class hierarchy into the user interface. This is done by the association of different methods with elements in the interface, such as buttons, text boxes, and things of that sort. This is the final development step of the entire system before it gets rolled out to the user!

**Validation:**

This is the final testing stage of the software, one final chance for any software-breaking bugs to be worked out before it is shipped. In this step, the developers and users through the user interface with a fine-tooth comb, making sure that every element is doing exactly what it should be doing, nothing more and nothing less. It does not hurt to also go back and retest the other components as well (Database and class hierarchy) just in case there were any mishaps with the interface integration.

Once the user is satisfied with the state of the final version, it can be released!