ELEC3225 Applied Programming Concepts

Assignment #2 Process Models

Robbie Cullerton

Partner: Nick DosRemedios

5/13/2025

**Requirements:**

1. Feasible Study: The project is feasible to complete, and it will align with the budget we have (which is none).
2. Requirements Elicitation: We will be basing this project by the LeopardWEB system that is already created, so we have created a list of classes, attributes, methods, and specific code needed to create a good working system.
3. Requirements Specification:
   1. Users should be able to see that they are in the LeopardWEB system. They should see the class that they are (student, instructor, and admin), they should be able to see first/last name, and ID number.
   2. All Students should be able to register for classes, and see available courses, and see their own schedule.
   3. All Instructors should be able to see available courses, and their own schedule.
   4. All Administrators should be able to see everything that the other two classes can see, and they also should be able to edit courses/users/schedules.
4. Requirements Validation: Since other users will not be able to test our system, we will need to do simulation-based testing for each class to figure out if our code works with proficiency. After testing, we can see if we need to change what we have completed from the previous steps.

**The requirements should take a week to get finished in total.**

**System and Software Design:**

1. Architectural Design:
   1. User will be the base class which will have all the attributes and print attribute methods that will be inherited by the following classes.
   2. Students will be an inherited class from the base class user where it will have a method to see its attributes, a method that allows them to search courses, a method to add/drop courses, and a method to print their schedule.
   3. Instructors will be an inherited class that will have a method to allow them to print their schedule, a method to print their class list, and a method to search for courses.
   4. Administrators will also be an inherited class from user that will have a method to allow them to add courses to the system, a method to remove courses from the system, a method to add and remove users, a method to add/remove student from a course, and a method to search and print rosters and courses.
2. Interface Design: All the previously mentioned “high level components” from the Architectural Design process will be connected through header files and inheritance. All the files will share the same MAIN which means that the classes can be used all in the same location through importation.
3. Database Design: This is the point where we will implement a database that will have two sections. The first database would be the Database of users where the system should work for 100 students, 10 instructors, and 1 admin. Then the Database of Courses which will contain the information such as the CRN, course name, times, and instructors.
4. Component Design: Here we will implement a User Interface either graphical or text based that will align with all the code that we have written and tested from this point.

**The design/analysis should take about 5-6 weeks in total to finish, maybe even longer**

**Implementation and Unit Testing and Integration and System Testing:** at this point we will go back and test all the above implementations to get a working prototype. This will include all the class components and functions, the database integration, and the interface integration. If everything goes as plan, we can proceed with the final coded system.The reset of the time left should be spent doing the implementation and testing, so about 3-4 weeks.

**Operation and Maintenance:** If this were a real project for a customer this would be the part that we would do bug fixes and further testing. This will help add additional pieces that the customer might want, or it might help us find bugs in the development of the code. This will take no time because we do not need this for the LeapordWEB project.

A diagram of software development

Description automatically generated

**Note: With the Waterfall Model, each step will follow one another like the chart above.**

**Incremental Model:**

1st Version: Students, Admin, and Instructors should be able to see that they are in the LeopardWEB system (database). They can see first/last name, ID, and what class they fall into (Instructor, Admin or Student). Attributes for all the different classes are in place and print methods work with the attributes. This should take about a week to finish (already done).

2nd Version: Students can register and see available courses, Instructor can see available courses, Admin can see both the Student’s registrations and the available courses offered. There will be scheduling preferences added at this point as well (with multiple semesters). This will take approximately 4 weeks to finish.

3rd Version: Students can see their schedule, Instructors can see their course roster, and Admins can see both. Students and Instructors can print schedules and Admin can print either of the two. This will take about 4 weeks to finish this portion.

4th Version: Admin can edit courses/users/schedules. Both the Instructor and the Student should be able to see the changes after they are made from the Admin. This will take 4 weeks to finish, and the rest of the time will be spent doing validation.

**Integrate and Configure Model:**

**Requirements Specifications:** We need to find a website or system, similar to what we would like to create, and base our system around it. In this project we will be designing a scheduling system for a university similar to LeopardWeb. The system will allow students, faculty, and an admin (such as the registrar) to add courses, search for courses, print schedules, etc.

**Component Analysis:** The existing software that we can base our project off would be the current LeopardWEB or any other college’s scheduling system with the same purpose. Since LeopardWEB is an already existing platform, we will be able to match some of the functionalities with our project.

**Requirements Modification:** The things that we would need to revise when basing our project off the current LeopardWEB system would only stem from our project. For example, we only need 100 students, 10 instructors, and 1 administrator, whereas the “real” LeopardWEB has a lot more students, instructors, and administrators. Also, the complexity of our project will most likely not be close to the level that the actual site is on (quality). This should only take about a day to get done.

**System Design with Reuse:** The system design with reuse will be like the LeopardWEB system but not totally the same. The database, and GUI will be different but will be modeled after the existing website. The functions and or methods will be similar, which is where most of the reuse will be from.

**Development and Integration:** This part will be strictly built from us. This is because although we want to sort of re-use the design of LeopardWEB, we still want to add our flair so there is no copyright infringement. It sounds silly now because we go to the school, and use LeapordWEB almost every day, but if we were out at a real company or doing a solo project, we must create our own system even though we are using an existing system for reference.

**System Validation:** With this model, the main way we are going to validate our system would be to compare our system with the “real” LeopardWEB system. Although it will not be the same, we can check out all the key similarities to make sure we did a good job to reuse the design. The good thing about this model is that it is easy to reuse a design and to base a system off it. Validation with this model comes easy, because we will know that if it is like the previous system, ours will work.

**Link/Path for Integrate and Configure:**

[my.wit.edu](https://my.wit.edu/campusm/home#menu) 🡪 Leopard WEB (Path to Reused Site)