Moises Fernandes

Applied Programming Languages

Assignment 2

May 28th, 2022

**Waterfall Model Diagram**

**Description of each step (Waterfall Model)**

**Requirement Definition:** This is the step where we establish the services required, possible constraints and overall goals.

**System and software design**: The next step is to establish the overall system architecture; we also need to define the software abstractions and relationships. This will be the step where we brainstorm what type of classes/objects are needed, potential methods and databases.

**Implementation and unit testing**: This step is where we write the code and test the components.

**Integration and system testing**: This step is where we integrate the components into a system and test the system.

**Operation and maintenance**: This are the installation phase; we also need to give the system outside usage (potentially a control group) and fix bugs and update any issues that we come across.

**Waterfall Model Detailed Outline (LeopardWeb System)**

**Task**

We will be designing a scheduling system for a university; it will be like LeopardWeb, and it will have various functions. This system will allow multiple different users to do various things such as search for courses, add courses, drop courses, search and print class list and class rosters etc. Of course, certain functions will be locked behind a specific user.

**Requirement Definitions**

Feasibility study: Looking at the criteria and understanding what needs to be done, I conclude that this project is possible and can be satisfied with current technology. For this project the best approach will most likely be using Object Orientated methods such as classes, inheritance, polymorphism etc.

Requirements elicitation and analysis: Understanding that we are emulating a university course system we can look at other models that are used for out project to lessen the workload, doing so we can also make necessary changes.

Requirement Specifications:

* Data base of users: The system needs to operate for 100 students, 10 instructors and 1 admin
* Data base of courses: This is going to contain all the information needed for course related searches such as, CRN codes, course names, time, and the instructor per course
* Three different types of users:

1. Student – can register, see available courses and their own schedule
2. Instructor – can see available courses, and their own course roster
3. Admin – can see everything, can edit courses/ users/ schedules

* The system should include multiple semesters, print out of schedules, schedule preferences.
* The system needs to be thoroughly testing before being released
* The base class is the User, which shall accommodate these specifications:

1. Attributes: First Name, Last Name, User ID
2. Methods: Set function for each individual attribute, and a corresponding function to print all the information of a desired user.

* Derived Classes:
* All derived classes must contain any additional attributes appropriate to the specific class and needs to have “set/get” methods per specific attribute.
* **Student:** The students will be able to search for courses, add/drop courses, and print out their schedule
* **Instructor:** The instructors should be able to print their schedules and print their class list and search for courses
* **Admin:** The admins should be able to add courses to the system, remove courses from the system, add/remove users, add/remove students from a course, search, and print both rosters and courses.

Requirement Validation: These requirements make sense, and they also match with what the user wants the system to do.

**AFTER this has been thoroughly looked at then we can move on to the next step.**

**System Software and Design**

**Software validation**

* Components: Looking at what needs to be done a lot of the functions are similar in nature (add/drop, search, and print) so many of the components can be reused throughout the project.
* Program testing: Using made up course names and CRN codes to do an initial test of the system.
* Component testing: Frequent testing of all the stated components.
* System testing: Integrate databases, code, and user interface to test the entire system
* Acceptance testing: Using actual course names and CRN codes in our program

**AFTER this has been thoroughly looked at then we can move on to the next step.**

**Implementation and Unit testing**

Architectural Design: What components – Functions, classes and objects, data base   
Interface design: (1) Interfacing sub-components (2) the user interface (GUI will most likely be used) Data base design: how many tables are needed? One for all users; possibly a table per species, etc.? What should be in each table.

What is needed

Students: All derived attributes from user class

Instructor: All derived attributes from user class

Admin: All derived attributes from user class

Information will be stored: Class times, Instructors linked Courses,

Component design: Looking at what needs to be done a lot of the functions are similar in nature (add/drop, search, and print) so many of the components can be reused throughout the project.

Implement: Code, databases, user interface

**AFTER this has been thoroughly looked at then we can move on to the next step**

**Integration and System Testing**

* Integrate all components and data bases and test them in a system.
* Testing all the different types of users, functions etc.

**AFTER this has been thoroughly looked at then we can move on to the next step.**

**Operation and Maintenance**

**Software Evolution**

Once we have reached a completed version of the system; Update the system based on the user request or bugs found while using the system

**Repeat if necessary**