COP 2535: Data Structures, Term Project

Step 01, Iteration 2, Requrements

1 Introduction

The COP 2535 Term Project will give you the opportunity to build an application using one of the data structures we will learn about. You may in fact use several. This project is not intended to be a heavy-weight project, and I don't expect you to make an extraordinary on it. However, this could be part of your portfolio, so you could decide to make more of an effort if you choose to do so.

This project follows an iterative, incremental development model. It is *iterative* in that you will repeat the same series of steps four times through the term. It is *incremental* in that you will add to the functionality of your project during each of the four iterations. Each iteration will consist of the same four phases.

Requirements Analysis In this phase, you will determine the functional requirements to be implemented in this iteration. *Functional requirements* refers to what the software will actually do. This is probably the most important phase, because you cannot build anything unless you know what it is that you are building. Your deliverable will typically be one or more Use Cases, or a Functional Requirements Specification.

Design In this phase, you will design the software to be implemented. In some respects, this is the hardest phase. Thhis is where you will make decisions as to *how* to implement the requirements. Your deliverable will typically be in the form of UML diagrams. We will mostly be using Activty Diagrams, which are similar to flow charts.

Implementation In this phase, you will implement your design in code. The deliverable is your source code.

Testing In this phase, you will test your implementation against the requirements. That is, you will answer the question, "Does the code do what the requirements expect?" Your deliverable will typically be a text document containing the output of the code when run.

2 Instructions

During iteration 1, you completed four phases: (1) requirements analysis, (2) design, (3) implementation, and (4) testing. In iteration 2, you will complete the same series of steps. The difference is that you will increment your application by either (a) adding a new set of requirements, or (b) re-implementing the old requirements.

The idea behind adding a new set of requirements is that you build software one step at a time. As the old joke goes, "How do you eat an elephant?" The answer is — one step at a time. So as iterations continue, you add additional features and corrrect bugs.

The idea behind refining the old requirements is that requirements change. Either the needs have changed, the customer may have changed hhis mind, or the initial requirements may have been mistaken. In a sense, requirements are a moving target. You will not have to work as a software engineer long before you hear a customer say, "That's not what I wanted." You think to yourself — I implemented exactly what you asked for. Some of us have decided that customers never know what they want.

3 Deliverable

Your deliverable for Iteration 2, Requirement Analysis will be a Use Case. *Please make sure you give your Use Case a title!* Eventually, you may have many use cases,and you need to have some way of telling them apart.

Actors anyone or anything that performs a behavior (who is using the system)

Stakeholder someone or something with vested interests in the behavior of the system under discussion (SUD)

Primary Actor stakeholder who initiates an interaction with the system to achieve a goal

Preconditions what must be true or happen before and after the use case runs.

Triggers this is the event that causes the use case to be initiated.

Main success scenarios (Basic Flow) use case in which nothing goes wrong.

Alternative paths (Alternative Flow) these paths are a variation on the main theme. These exceptions are what happen when things go wrong at the system level.

A Use Case is a text document. No particular formality is required. The purpose is to (1) describe the function of the software so that it can be tested, and (2) to pass on to the design team so that they can design the software. Here are some useful links.

- `https://www.usability.gov/how-to-and-tools/methods/use-cases.html`
- `https://www.wrike.com/blog/what-is-a-use-case/`
- `https://www.indeed.com/career-advice/career-development/list-of-use-cases-examples`