Document saved at 13:41 on 26. October 2011

Midterm Exam
for
Fall 2011 URI CSC 305/505

1 SCOPE

This document provides the Midterm Exam for the Fall 2011 URI CSC 305/505.

1.1 Document Overview

This document provides a consistent means for receiving answers to the Midterm Exam for the Fall 2011 URI CSC 305/505.

To use this document, perform the following steps:

1. Enter your name information in section 1.2.
2. Provide any general comments under the text in major section 2.
3. Answer each question in the space following the question. Feel free to include figures, tables, etc. but make sure that they get included properly in the submitted PDF file.
4. DO NOT CHANGE general formatting of this document.
5. Save your answers and convert the document to a PDF file that is named LastName_FirstName_CSC305_Midterm_20111026.pdf.
6. Email the completed PDF file to wlcollier@cs.uri.edu no later than 04:00 pm on Wednesday 26 October 2011. Make sure the subject includes your last name and CSC 305 or 505 Midterm.
7. Email me with any questions regarding this exam before 11:59pm on Saturday 22 Oct 2011. So that everyone benefits from any questions asked, I will respond to non-personal questions to ALL members of the class.
8. Exam will be graded as noted in the class syllabus.

1.2 Examinee Information

This section provides information regarding the examinee and exam completion date here. By submitting this exam, you certify, in accordance with the URI honor code, that the answers provided herein are your own product.

Examinee : Andrew T. Poirier

Submission Date: October 26, 2011

2 QUESTIONS

This section are my answers to the exams.

2.1 Answer the following questions regarding Design, Development, and Analysis.

2.1.1 Discuss the relationship between design and analysis.

The design of the of the system is the "how" the system where the analysis is more of "what" the system will do. The design of the system may include how the system will perform a certain action, how the system will achieve it's performance goals and how the interfaces will work together. The analysis of the system is what is the system going to do, what the performance of the system and what are the system of the system interfaces like keyboards, screens, card readers, etc. They are related in the way is that the analysis is going to connect the interfaces to the system and the design will determine how to use the resources in the system.

2.1.2 List 5 software development process types and 5 software engineering analysis methods. (Yes, you may have to do some research to find them.)

The five software development process types are waterfall model, unified process, incremental process, evolutionary process, and spiral model. All of these processes are analyzed differently. Five software engineering analysis methods are consisted of the data flow analysis, Yourden structured method, the structured design method, object oriented method and use cases.

2.1.3 Discuss the relationship between development processes and analysis methods. (You may actually need to read about some of the answers listed in 2.1.1 to form an opinion here!)

The development process is the step by step way of doing something. These are the steps that you can do to get to the end result. The most common process is inception, elaboration, construction, and transition. The analysis methods are the approaches you are going to take to solve the problem. So overall the process are the basic steps in development and the analysis is the overall system with different flows from piece to piece.

2.2 Answer the following questions regarding Software Engineering:

2.2.1 Discuss how context and hierarchy are important in the design of a software system.

Hierarchy are important to the design of the software system because it breaks the system down into different levels. This is found mostly in the documentation for the system but also used for the design of the system starting with the highest level the "brain of the system" which is then shown how it is communicating with the inputs and out puts of the system. Once the top level is created you keep on moving your way down to lover levels with the inputs and outputs as the center of the system and what parameters they will receive for the system. The context is important to the design because it relates the context of the document to the system itself.

2.2.2 Discuss how hierarchy is important to the models of software systems.

The hierarchy is important to the models of the software system because it relates the document to the models in a numerical approach. So the section x.y.z of the document will relate to section x.y.z in the diagram and its components.

2.3 Answer the following questions about the Documentation Templates Discussed in Class:

2.3.1 Define the "read-forward" structure and explain why this helps (or hurts) these documents.

The read-forward structure is a structure that only tells the reader the information in a numerical order. It starts with the scope and overview with the document then goes onto describing words or abbreviations that might be found in the context of the document then goes onto the system. With the system it starts with the existing system and the talks about why there is a need for modifications and the current interfaces. It then goes into information about the new system, requirements and interfaces that will be used. After both systems are described then there is a section describing both systems old and new and compares and contrasts them.

2.3.2 Describe the relationship between the OCD sections and the sections of the ensuing specification documents (SSS/SRS).

The relationship of the OCD and the SSS documents is that the OCD has a section just outlining the requirements for the system. The SSS document takes these requirements and adds details to them like constraints and performances that they are going to

perform. The SSS only describes the new system and not the old system like the OCD does.

2.3.3 Suggest changes that may make the OCD and SSS/SRS documents function better together.

I believe that the requirement section numbers in the SSS document should correspond with the ones in the OCD. This would make it easier when looking at both documents. They should keep the details of the requirement in the SSS but just number it the same number as the OCD but with greater depth like it is with the X.Y.Z format.

2.4 Answer the following questions regarding Use Cases and Context Diagrams:

2.4.1 What aspects of a system are identified in UML use cases.

UML use cases consist of four major elements, actors, system, use cases and relationship. The Actor interacts with the system then the use case is the way that the system will respond to the actors action and the relationship is the lines that connect all of these elements together to create a working system.

2.4.2 What aspects of a system are identified in data flow context diagrams.

In the data flow context diagram the main process or system are identified inside a circle in the middle of the diagram. The external interfaces are are in boxes around the center process with arrows representing the direction data is flowing from the process to the interfaces.

2.4.3 Discuss the similarities and differences between use cases and context diagrams.

The similarities between the use cases and the context is that they both have the external interfaces and main processes in them. In the use cases tho the arrows represent the connection in time from each to device and then to end user. In the context diagrams the interfaces are still there but the arrows are just represented once with the direction the data is flowing and not every time a interfaces talks to the main system or another process.

2.5 Provide a state machine (state transition diagram) following the methodology presented in class to meet the following requirements for an automatic door controller (like you find at a supermarket). Name all states according to the action being performed. Show all transitions with conditions and actions as appropriate. Provide a "data dictionary" of all signals used. Assume that external, but distinct, timers are available as needed.

- The system shall have two doors; one "inside" and one "outside".
- The system shall provide sensors to denote a person is "inside", "middle", or "outside" the doors. (*These sensors just produce signals to the state machine*).
- The inside door shall receive an OPEN command when a person is detected inside and the middle does NOT have a person. (We don't want to open the door into the person in the middle:)
- The outside door shall receive an OPEN command when a person is detected in the middle and the outside does NOT have a person.(We don't want to open the door into the person standing just outside the door, though maybe we should:)
- All doors shall close 5 seconds after the person has left the entry and exit locations for each door (inside and middle for inside door; middle and outside for outside door). (*Use generic timer signals, DO NOT IMPLEMENT THE TIMER*).