Document saved at 21:19 on 19. December 2011

Final Exam
for
Fall 2011 URI CSC 305/505

1 SCOPE

This document provides the Final Exam for the Fall 2011 URI CSC 305/505 class. This document also provides a "re-do" mid-term.

1.1 Document Overview

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

To use this document, perform the following steps:

1. *Enter your name and submission date 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 the general formatting of this document.
5. Save your answers and convert the document to a PDF file that is named LastName,FirstName_CSCx05_Final.pdf. OpenOffice can convert directly to PDF!
6. Email the completed PDF file to wlcollier@cs.uri.edu no later than **10:00pm on Monday 19 Dec 2011**.
7. Email me with any questions regarding this exam before **11:59pm on Monday 12 Dec 2011**. So that everyone benefits from any questions asked, I will respond to non-personal questions to ALL members of the class.
8. **Final exam will be graded 25% for each question in section 2.**

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: December 19,2011

2 FINAL EXAM QUESTIONS

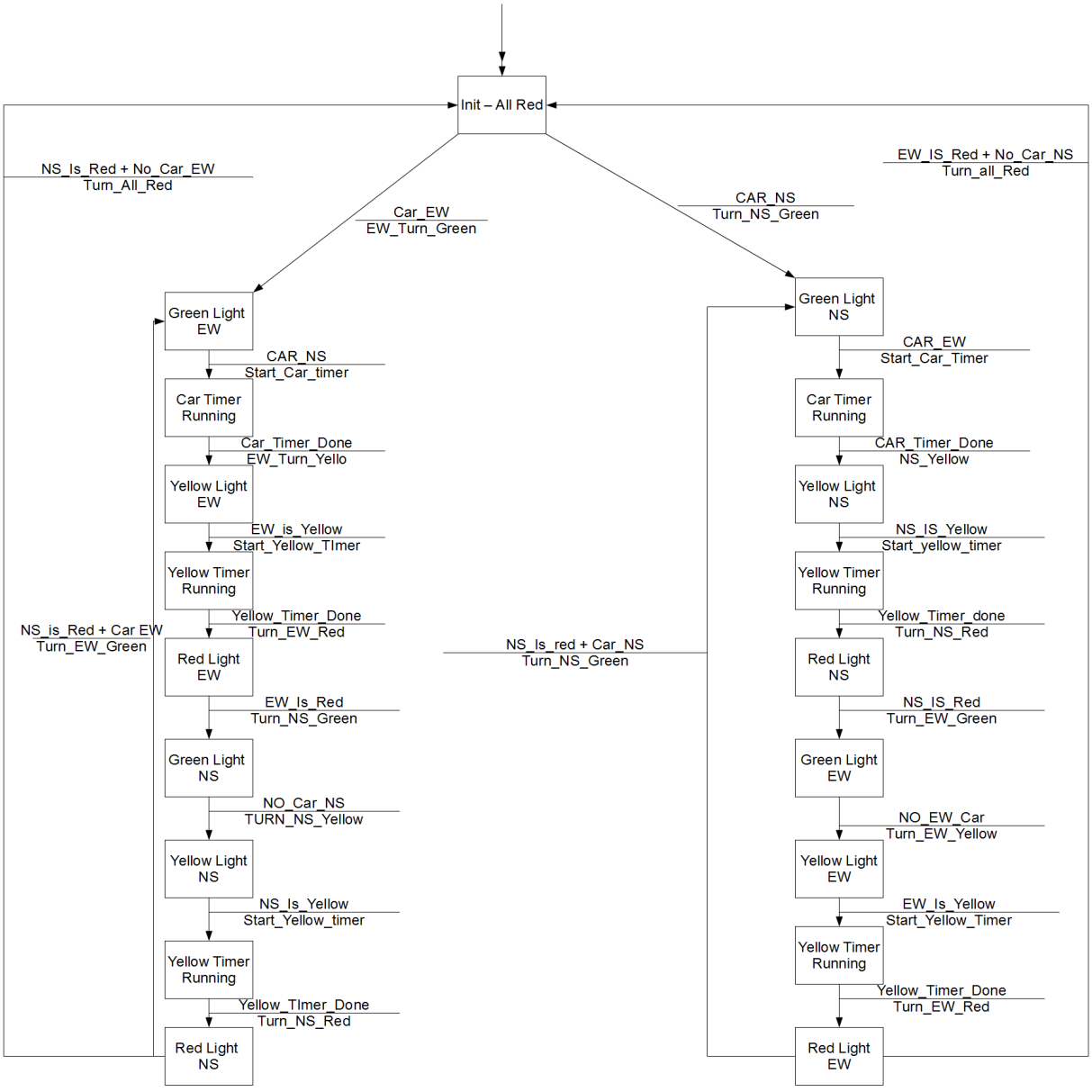
Replace this text with any general comments regarding the exam..

2.1 List the levels of CMMI and provide a brief description with examples (*in your own words*) of what each level means. You can find information about this in the Pressman text, at http://www.sei.cmu.edu/cmmi/, or lots of other places on the web..

In CMMI there are five different levels that make it up. The five levels are Initial, Managed, Defined, Quantitatively Managed, and optimizing. In the first level, initial or incomplete level, there is very low quality and it has the highest risk. There is very little documentation, processes, scope and procedures which means that you can be successful only by the other actions you and your team members. The second level known as Managed there is a little better success rate then in Initial level. This level consists of basic project management which means there is project planning, project monitoring, product quality assurance and configuration management. In this level there is still a low quality and a high risk. In the managed level there is enough documentation so it can be used again for a new version or product upgrade. The third level which is the defined level there is a medium quality wit ha medium risk for the result. In this level there main focus is on process and standardization with developed requirements, technical solutions along with verification and validation to back up the solutions. In this level there is also organized training with integrated teaming which creates organized process focus and definitions. In the quantitatively managed level the main focus is on quantitatively management which consists of an organizational process performance and a quantitative project management. In this level there is a higher quality along with a lower risk. In the fifth and final level, optimizing there is a main focus on continuous process improvement. In this level the main process area is in organization in innovation and deployment with casual analysis and resolution.

2.2 Provide a state machine (state transition diagram) following the methodology presented in class to meet the following requirements for a traffic light. 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 for timing each cycle.

- The system shall have two sets of signal lights; one in the north-south direction (NS) and one in the east-west direction (EW).
- The system shall provide sensors to denote a car is at the EW or NS direction. (*These sensors just produce signals to the state machine*).
- Both lights (EW and NS) shall start in the red state.
- When a car arrives at the light, the signal shall give that direction a green light and the other direction shall have a red light. (Assume that there are no ties so you do not have to worry about priority arbitration.)
- The signal shall follow the normal green-yellow-red progression as expected for a traffic signal when a car approaches from a direction that is being shown a red light.
- The signal shall remain in the green state for no longer than X seconds when a car is waiting in the other direction. (*Use generic timer signals, DO NOT IMPLEMENT THE TIMERS*).



See attached PDF For full state machine

Data Dictionary	
Term	Meaning
EW	East West Direction
NS	North South Direction
Car_EW	Car Sensed in the East West direction
Start_Car_timer	Starts the timer for the length the car has to wait in the opposite direction
Car_Timer_done	Car timer has finished
EW_Turn_Yellow	EW lights are sent signal to turn yellow
EW_IS_Yellow	Signal saying the EW light is now yellow
Start_Yellow_Time	Starts the timer for the length of the yellow light to stay on
Yellow_Timer_Don	The Yellow Light has been on for a certin amount of time and its time to turn red
Turn_EW_Red	Signal to turn EW lights to red
EW_Is_red	Signal saying the EW Light is now red
Turn_NS_Green	signal to turn NS lights to green
NO_Car_NS	Signal saying there is no cars in the NS direction
Turn_NS_Yellow	Turns the NS lights to yellow
NS_Is_Yellow	signal saying NS lights are now yellow
Turn_NS_Red	turns the NS light to red
No_Car_EW	No car ditected in the EW direction
Turn_All_Red	Turns both lights to red
CAR_NS	Car in sensed in the NS direction
NS_Yellow	Turns the NS lights to yellow
NS_is_Red	Signal saying the NS Light is red
Green Light NS	The NS lights are in the green state
Car Timer Running	The Car timer is running
Yellow Light NS	the NS lights are yellow
Yellow Timer Runn	the state saying the yellow timer is running
Red Light NS	the NS lights are in the red state
Green Light EW	EW lights are in the green state
Yellow light EW	EW lights are in the yellow state
Red Light EW	EW lights are in the red state

2.3 List 5 metrics, describe each metric, and discuss how each can be used in the software engineering process. You can find many metrics in the class slides or in the Pressman text.

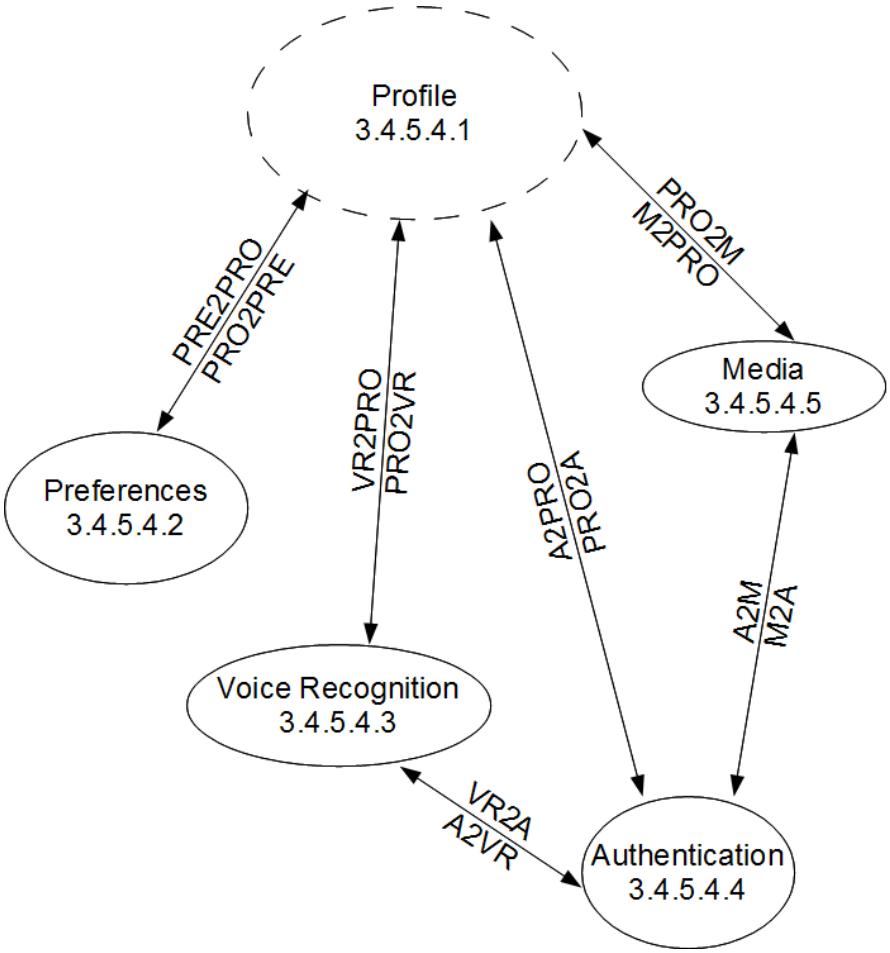
Thee following are five metrics and how they can be used in in software engineering

1. Architectural Design metrics focuses on the program architecture of the software and how it is structured and how effective it is with the modules or components within the architecture. This metric is refereed to the "black box". The reason why it is called the black box is that they do not need any prior knowledge of the inner workings of a certain software component. This is helpful in software engineering because you can develop one part of the software and test the architecture of it with out designing or incorporating any other parts of the system when testing the metric.
2. Class oriented metrics are all measurements and metrics for individual classes with the object oriented software system. This metric provides a class hierarchy and class collaborations. The problem with these collaborations is that they are not useful to the software engineer who accesses the design quality of the software. The reason being is because of the inheritance that is used with the classes and the parent class and the children classes. On the other hand it is important to software engineers because it is a metric that describes the classes and how they will all work together.
3. Cohesion metrics are metrics that were made by Bieman and Ott and they are a collection of metrics that provide an indication of the cohesiveness of a module in the software. The cohesion metrics have five different concepts and measurements which are data slice, data token, glue tokens superglue tokens and stickiness. The data slice is basically a backward tour through a modules that will be looking for data values that affects the state of the module. The data tokens are just variables for a module defined by the glue tokens for that particular module. The glue tokens for the module are the sets of data tokens that are laying on one or more data slices. Superglue tokens are data tokes that are very common to every data slice in the modules. Stickiness is the relations and how close the relations are between the number of data slices that are bounded.
4. User interface design metrics are metrics that are used for the design of the human/computer interfaces and the quality and usability of the interfaces with the end user. There is a metric that will measure the direct usage of the UI interactions and the time it takes to achieve the particular requirement and the

time it takes to respond to errors. This also counts the specific tasks that are required to complete a a use case. This metric is important when engineering software because touch screens are becoming more popular these days and the metrics for these devises are really good especially when it comes to the response time and positioning of the touch from the end user.

5. Halstread metrics applied to testing is a metric that tests the effort of the programing volume, programing level and the percentage of the overall testing efforts that are allocated. The sum of of this metric is used across all of the modules on the system. This is important to software engineering because it is the metric that works across all of the testing procedures that are used to test a piece of software.

2.4 Take one of the data flow diagrams that you generated in your project design and describe how it represents an object. (Remember that an object is defined by its name, its attributes, and methods. The attributes include data and state information.) Include Any applicable UML artifacts that help describe the object.



This DFD is representing the object of the Profiling system of the smart house. The object will send information to the preferences object, voice recognition object, authentication object, and the media object. It will also receive data from all of those objects. The voice recognition object will send and receive data from the authentication object so authenticate the voice that is being spoken. The media object will also talk to the authentication object to get the users information on the media to be displayed. All of the objects are referring to the different states that they are in. So with preference it

starts in the profile state then will go onto preference state when preference are requested. All of the other objects have a similar state. When voice recognition is in the state requesting authentication then it is sitting in a state requesting authentication for the certain user speaking. When media requests authentication it is doing the same but it is requesting for the current user logged in so it knows what media to display for the user that is in the house.