CSCE 742 - Assignment 3

Due Date: Sunday, December 9, 11:59 PM (in PDF format, via Dropbox)

Question 1 (20 Points)

Consider the student accounts database at the University of South Carolina.

1. Identify two kinds of sensitive data or other resources in the system.
2. Determine the possible attackers, their motivations, and their resources.
3. Develop an attack tree for possible attacks on one of the identified sensitive resources. Be sure to include both technical and 'social engineering' risks.
4. Define mitigation strategies for each leaf-level risk

Question 2 (9 Points)

Describe three benefits and three drawbacks of using a dedicated architecture description language (ADL) such as AADL rather than a general purpose UML modeling tool for describing architectures.

Question 3 (6 Points)

"Every system has real-time performance constraints."

Do you agree with this statement? Discuss why or why not. If you disagree - provide a counterexample.

Question 4 (7 Points)

Web-based systems often use proxy servers, which are the first element of the system to receive a request from a client (such as your browser). Proxy servers are able to serve up often-requested web pages, such as a company's home page, without bothering the real application servers that carry out transactions. There may be many proxy servers, and they are often located geographically close to large user communities, to decrease response time for routine requests.

What performance tactics do you see at work here?

Question 5 (8 Points)

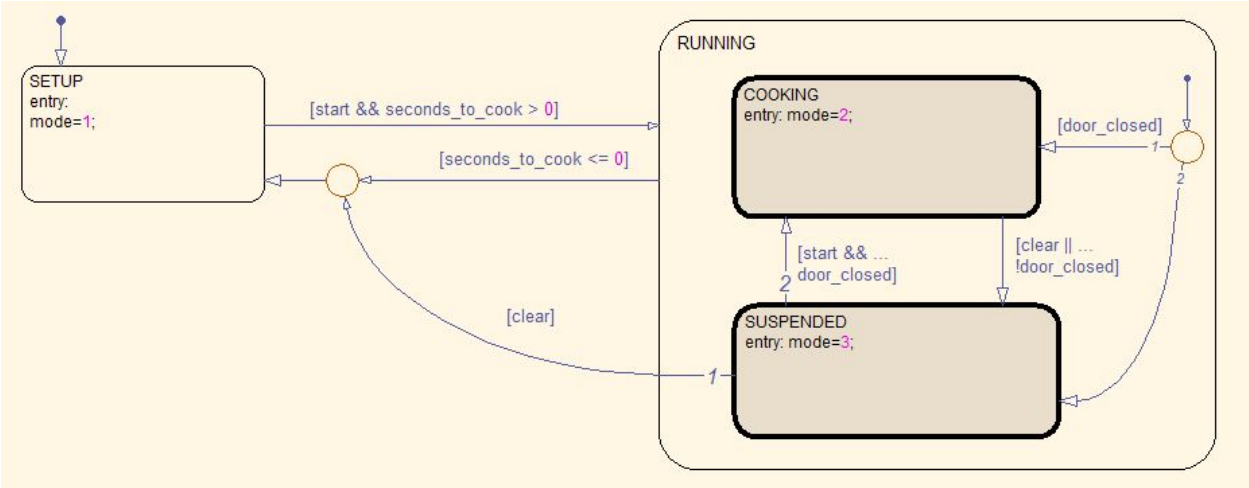
How does availability trade off against modifiability? How would you make a change to a system that is required to have "24/7" availability (no scheduled or unscheduled downtime, ever)?

Question 6 (50 Points)

In the assignment materials, there is an AGREE User's Guide, which contains installation instructions for the OSATE tool suite for AADL and the AGREE plug-in. The website also contains three .zip files containing AADL/AGREE projects. Once AGREE is installed, to use the steps in Section 4.2 of the AGREE User's Guide to import the .zip files into projects.

The first project is "GPCA Controller" and contains a medical device infusion pump example. This project is just there as an example and for reference in terms of property writing. The second project is the "Toy_Verification" component example that we discussed in class and I used to demonstrate the AGREE tools.

The third project is the "Microwave" example that I would like you to explore. This example defines the behavior for a very simple microwave controller in terms of two subsystems that control the operating mode of the microwave (Mode_Control) and the display panel (Display_Control), respectively. The Mode_Control uses a simple state machine to control its behavior:



The mode controller starts in the SETUP mode, and transitions to the RUNNING mode when the start button is pressed (if the seconds_to_cook parameter is > 0). In the RUNNING mode, the microwave can either be in COOKING or in SUSPENDED mode. If the microwave door is opened when in RUNNING mode or the clear button is pressed, then the microwave is in SUSPENDED mode, where the user can either press the start button to go back to COOKING, or press the clear button again to go back to SETUP.

Similarly, the Display_Control model processes keypad inputs and displays the time to cook (in terms of three digits that represent minutes, tens of seconds, and seconds, respectively). If the microwave is RUNNING, then the numeric keypad buttons are locked out; the user can't change the time when the microwave is running (try this on your microwave at home!). While the microwave is COOKING, the digits should decrease (along with the total seconds to cook); when it is in SUSPENDED, the display digits should be frozen. In SETUP mode, the user can program in the desired time to cook.

When the microwave is assembled from these two subsystems, we should be able to prove three safety properties about the overall microwave behavior:

```
guarantee "The heating element is on only when door_closed_sensor is true"
guarantee "When the heating element is on the time to cook shall decrease"
guarantee "When time to cook is zero, the heating element shall be off"
```

I have provided the skeleton of the microwave example and a set of property skeletons (currently all set to ‘true’) that you must fill in to verify the system level safety properties.

1. Fill in the definitions of the guarantees in the Microwave example and prove that the components are consistent and that the system properties are satisfied. Include the .aadl file that you create as a separate file in the ZIP file for the submission.
2. Which (if any) of the three system properties actually require guarantees from all of the subcomponents {mode_control, display_control, output_processing}, and which require guarantees on only a subset of the components? Why? Another way of saying this is the following: if we removed all the guarantees from one of the subcomponents, would any of the system-level properties be provable?
3. Is it possible to write a single property of the form: “The microwave will eventually stop cooking” to replace some of the properties we currently have in the top-level system component? Why or why not?