CST315 Software Enterprise: UML Statecharts and Class Diagrams (Fall 2015 Revision)

UML Statecharts focus on object behaviors and how events impact the state of the object. While activity diagrams take an “outside in” approach, meaning objects are *black boxes* and we focus on flow, statecharts take an “inside out” or *white box* approach, where we focus on the internals of object behaviors and categorize the various events that are incident on the object. UML class diagrams create structural representations between *classes*, the finest-grained detail modeling element we will use this semester. Class diagrams may feel the most familiar to due their direct language representation in Object-Oriented Programming, but from a modeling perspective they are actually not as useful – provide as much abstraction - as Component diagrams, which we also mentioned in this module.

Objectives:

1. Understand and apply behavioral modeling via statecharts to model an object lifecycle.
2. Create structural diagrams using class and common UML modeling element relationships
3. Practice UML syntax and semantic “tools” in the context of a modern tool.

Tools:

1. *Visual Paradigm, Community Edition*
   or
2. *Microsoft Visio*

Task 1 (individual): Creating a UML Statechart for an ATM [40 total points]
Create a UML Statechart in VisualParadigm or Visio to model the lifecycle of an ATM (from the ATM’s perspective). Consider the workflows for withdrawing money, depositing money, and printing a receipt. What are the events incident on the ATM? What states does it go through? Does it have a final state? This model should consider all of the complexity of an ATM and use the full power of a Statechart – guard conditions, different types of events, and so on. As a starting place, think about including:

- An event indicating a debit card was inserted and event indicating the user requested a withdrawal, a deposit, or a print receipt.
- A transition from the initial state to some state indicating “waiting”. The transition should not have a label on it at all.

a) The diagram should attempt to model an ATM – multiple interpretations are fine. [10 points]
b) The diagram should be complex enough to be useful – at *least* four states with appropriate transitions. [15 points]
c) UML notation rules must be obeyed. [15 points].

Task 2 (individual): Creating a UML Class Diagram for Computer Internals [40 total points]
Create a UML Class diagram in VisualParadigm or Visio that represents typical workstation computers (i.e. generic laptops or desktops, not servers or mobile/tablets). Your diagram will be created in two stages of increasing detail – submit both. This is an analysis and design question – there are multiple correct approaches.

a) Create a class diagram that describes the structure between common parts of a computer. [20 points]
   - Be sure to consider what parts are required/owned (*Composition*), parts-of (*Aggregation*), or just *Associations*.
   - What level of detail should you include (CPUs, RAM, hard drive, etc.)? What are the multiplicities of the internal parts?
   - Consider if there are Generalization relationships to include. For example, is there more than one type of hard drive?
   - This class diagram should only show structural information, not attributes or operations.

b) Copy the diagram you just created and add: [20 points]
   - Attributes (or properties) describing each internal part.
   - Behaviors each internal part may exhibit.
   - Consider what attributes & operations go on what classes. For example, a hard drive might have a *seek time* attribute and a *read* operation. A specialization *mechanical hard drive* may add attribute *RPM* and operation *read sector*.

Submission

- Submit a Word document containing exported images of each of the three diagrams you created.
- Name the file <asurite>_classandstatechart.docx.
- Make sure that the diagrams you produce are syntactically and semantically correct. Your UML tool will help, but do not just manipulate the tool to get a picture, realize that the symbols mean something.