Reading 2 -- State diagrams x Class description

Goal: To verify that the classes are defined in a way that can capture the functionality specified by the state diagrams.

Inputs to Process:

1. A set of class descriptions that lists the classes of a system along with their attributes and behaviors.
2. A state diagram that describes the internal states in which an object may exist, and the possible transitions between states.

For **each state diagram,** perform the following steps:

1) **Read the state diagram to understand the possible states of the object and the actions that trigger transitions between them.**

   INPUTS: State diagram (SD).

   OUTPUTS: Object States (marked in blue on SD);
   Transition Actions (marked in green on SD);
   Discrepancy reports.

   ☞ Determine which class is being modeled by this state diagram.

   Could you identify the type of the object that this state machine is modeling? If you can't find this information, you have found a defect of omission. Fill in a discrepancy report for this.

   ☞ Trace the sequence of states and the *transition actions* (system changes during the lifetime of the object, which trigger a transition from one state to another) through the state diagram. Begin at the start state (filled circle) and follow the transitions until you reach an end state (double circle). Make sure you have covered all transitions.

   ☞ Underline the name of each state, as you come to it, with a blue pen.

   ☞ Highlight transition actions (represented by arrows) as you come to them using a green pen. For example, the state diagram provided in Example 5 contains seven transition actions. The arrow leading from the state labeled "authorizing" back to itself represents an action that does not actually change the state of the object.

   ☞ Think about the states and actions you have just identified, and how they fit together.

   Is it possible to understand and describe what is going on with the object just by reading this state machine? If not, you may have discovered a defect of ambiguity; the behavior of this class over its lifetime is not well described.

2) **Find the class or class hierarchy, attributes, and behaviors on the class description that correspond to the concepts on the state diagram.**

   INPUTS: Class description (CD);
   Object States (marked in blue on SD);
   Transition Actions (marked in green on SD).

   OUTPUTS: Relevant object attributes (marked in blue on CD);
   Relevant object behaviors (marked in green on CD);
   Discrepancy reports.

   ☞ Use the class description to find the class or class hierarchy that corresponds to this state diagram. **Did you find the corresponding class? If not, you have found a defect of inconsistency. The state machine describes a class that has not been described on the class description.**

   ☞ Find how the responsible class encapsulates the blue-underlined states described on the state diagram. States may be encapsulated:

- 1 attribute explicitly. (An attribute exists whose possible values correspond to system states, e.g. attribute “mode” with possible values “on”, “off”.)
- 1 attribute implicitly. (An object is considered to be in a specific state depending on the value of some attribute, but the state is not recorded explicitly. E.g. if a>5 the object behaves one way, for other values of a another behavior is appropriate, but nothing explicitly records the current state.)
- a combination of attributes.
- class type. (E.g. subclasses “fixed rate loan” and “variable rate loan” can be considered states of parent class “loan”.) Remember to check the corresponding class and all parents in its inheritance hierarchy.
  Mark each blue-underlined state with a star (*) when it is found.

  Has the class captured the idea of the modeled states? If not, you may have found a defect of inconsistency; the state diagram specifies certain states for an object that cannot be captured by the class as it is described. Or, you may have found an ambiguity; it is not clear how the modeled states can be captured by a class.

☞ For each green-highlighted transition action on the state diagram, verify that there are class behaviors capable of achieving that transition. Remember to look both in the currently selected class and any classes higher in the inheritance hierarchy.

(Keep in mind the following possible exceptions: 1) The transition depends on a global attribute, outside of the class hierarchy. 2) In instances of poor design, i.e. high coupling and public class attributes, behaviors in associated classes can modify the value of a variable in the class directly.)
If the transition action is an *event* (i.e. a transition occurs when something happens) look for a behavior or set of class behaviors that achieve that event.

If the transition action is a *constraint* (i.e. a transition occurs when some expression becomes true or false) look for behaviors that can change the value of the constraint expression. For example, note the constraints “[payment ok]” and “[payment not ok]” in example 5. These describe when the actions they describe can happen, based on the status of payment.

Does the class encapsulate behaviors to deal with the modeled actions? If not, you have found a defect of inconsistency. The state diagram specifies certain actions that no class behavior is capable of implementing.
Could you identify the object data used to verify the constraints? Are they consistent between the state diagram and the class description? If not, you have found an inconsistency. The state diagram describes certain constraints that must be checked but the class description as described may not support this check.

3) Compare the class diagram to the state diagram to make sure that the class, as described, can capture the appropriate functionality.

INPUTS:	Object States (marked in blue on SD); Transition Actions (marked in green on SD).
OUTPUTS:	Discrepancy reports.

☞ Consider the system functionality in which this class participates, as described by the class description, and the states in which it may exist, as described by the state diagram.

Using your semantic knowledge of this class and the behaviors it should encapsulate, are all states described? If not, you have uncovered a defect of incorrect fact, that is, the class as described cannot behave as it should.

☞ Review the state diagram, looking for any unmarked states.

Is there some unstarred state? Could you evaluate the importance of this state? Does it really describe an essential object state? Is the state feasible considering all actions and constraints surrounding it? If yes, probably something is missing on the class diagram and there is an inconsistency between the diagrams. Otherwise, an extraneous fact should be reported.

☞ Review the state diagram, looking for any unmarked actions.

Is there some unstarred event? If yes, fill in a defect record showing the inconsistency between the class description and state diagram.

Is there some unstarred constraint? Is the constraint directly concerned with some object data? If yes, fill in a defect record showing the information that has been omitted from the class description.