Reading 6 -- Sequence Diagrams x Use-cases

Goal: To verify that sequence diagrams describe an appropriate combination of objects and messages that work to capture the functionality described by the use case.

Inputs to process:

1. A use case that describes important concepts of the system (which may eventually be represented as objects, classes, or attributes) and the services it provides.
2. One or more sequence diagrams that describe the objects of a system and the services it provides. There may be multiple sequence diagrams for a given use case since a use case will typically describe multiple "execution paths" through the system functionality. The correct set of sequence diagrams for a use case must be selected by using traceability information, or by someone with semantic knowledge about the system. Finding the correct set of sequence diagrams without traceability information or knowledge of the system will be hard.
3. The class descriptions of all classes in the sequence diagram.

1) **Identify the functionality described by a use case, and important concepts of the system that are necessary to achieve that functionality.**

   INPUTS: Use case (UC)

   OUTPUTS: System concepts (marked in blue on UC);
   Services provided by system (marked in green on UC);
   Data necessary for achieving services (marked in yellow on UC).

   - ☞ Read over the use case to understand the functionality that it describes.
   - ☞ Find the nouns included in the use case; they describe concepts of the system. Underline and number each unique noun with a blue pen as it is found. (That is, if a particular noun appears several times, label the noun with the same number each time.)
   - ☞ For each noun identify the verbs that describe actions applied *to* or *by* the nouns. Underline the identified services and number them (in the order they must be performed) with a green pen. Look for the constraints and conditions that are necessary in order for this set of actions to be performed. As an example, consider Example 1, in which constraints and conditions have been highlighted. In this use case, there is an example of both a constraint ("The Customer can only wait for 30 seconds for the authorization process") and a condition ("time of payment is the same as the purchase time").
   - ☞ Also identify any information or data that is required to be sent or received in order to perform the actions. Label the data in yellow as "Di,j" where subscripts i and j are the numbers given to the nouns between which the information is exchanged.

2) **Identify and inspect the related sequence diagrams, to identify if the corresponding functionality is described accurately and whether behaviors and data are represented in the right order.**

   INPUTS: Use case, with concepts, services, and data marked;
   Sequence diagram (SD).

   OUTPUTS: System concepts (marked in blue on SD);
   Services provided by system (marked in green on SD);
   Data exchanged between objects (marked in yellow on SD).

   - ☞ For each sequence diagram, underline the system objects with a blue pen. Number them with the corresponding number from the use case.

- ☞ Identify the *services* described by the sequence diagrams. To do this, you will need to examine the information exchanged between objects and classes on the sequence diagrams (the horizontal arrows). If the information exchanged is very detailed, at the level of messages, you may need to abstract several messages together to understand the services they work to provide. Underline the identified services and number them (in the order they occur in the diagram) with a green pen. Look for the condition that activates the actions.
- ☞ Identify the information (or data) that is exchanged between system classes. Label the data in yellow as "Di,j" where subscripts i and j are the numbers given to the objects between which the information is exchanged.

3) **Compare the marked-up diagrams to determine whether they represent the same domain concepts.**

INPUTS: Use case, with concepts, services, and data marked;
Sequence diagram, with objects, services, and data marked.

OUTPUTS: Discrepancy reports.

- ☞ For each of the blue-marked nouns on the use case, search the sequence diagram to see if the same noun is represented. Mark the noun on the use case with a blue star (*) if it can be found on the sequence diagram.
  Is there an unstarred noun on the use case? If yes, it means that a concept was used to describe functionality on the use case but was not represented on the sequence diagram. For each of the nouns on the sequence diagram, find the corresponding class on the class description and check whether the unstarred noun is an attribute. If the unstarred noun does not appear as an attribute of any of these classes, you have found an omission. A concept was described on the use case but has not appeared in the system design. Fill in a discrepancy report to describe the problem.
- ☞ Mark the nouns on the sequence diagram with a blue star (*) if they appear only on the sequence diagram.
  Is there a starred (*) noun on the sequence diagram? If so, you have found an extraneous noun, or a noun describing a lower-level concept, on the sequence diagram. Think about whether the concept is necessary for the high-level design, and whether it represents a level of detail that is appropriate at this time. If necessary, fill in a discrepancy report describing the problem.
- ☞ Identify the services described by the sequence diagram, and compare them with the description used on the use case. Are the classes/objects exchanging messages in the same order specified on the use case? Were the data that appear on messages on the sequence diagram correctly described on the use case? Is it possible for you to understand the expected functionality just by reading the sequence diagram?
  Do the classes exchange messages in the same specified order? If not think about whether this represents a defect. Usually, switching the order of messages may have an effect on the functionality. But sometimes messages can be switched without affecting the outcome; other times, messages can be performed in parallel, or conditions may ensure that only one or the other message is executed anyway. If a defect has been found, fill in a discrepancy report describing the problem.
  Are all transported data in the right message? Is data being sent between the right two classes (i.e. do the labels "Di,j" for the data match between diagrams)? Do the messages make sense for the objects sending and receiving them, and do they make sense for achieving the relevant services? If not, it means that the sequence diagram is using information incorrectly. Fill in a discrepancy report describing the problem.
- ☞ Are all the constraints and conditions from the use case being observed in this sequence diagram? Is some detail from the use case missing here?

  Were the constraints observed? Was all of the behavior and data on the sequence diagram directly concerned with the use-case? If no, it means that the sequence diagram is using information incorrectly. Fill in a discrepancy report on this matter.