Reading 1 -- Sequence x Class Diagrams

Goal: To verify that the class diagram for the system describes classes and their relationships in such a way that the behaviors specified in the sequence diagrams are correctly captured. To do this, you will first check that the classes and objects specified in the sequence diagram appear in the class diagram. Then you will check that the class diagram describes relationships, behaviors, and conditions that capture the dynamic services as described on the sequence diagram.

Inputs to process:

1. A class diagram (possibly divided into packages) that describes the classes of a system and how they are associated.
2. Sequence diagrams that describe the classes, objects, and possibly actors of a system and how they collaborate to capture services of the system.

1) **Take a sequence diagram and read it to understand the system services described and how the system should implement those services.**

INPUTS: Sequence diagram (SD).

OUTPUTS: System objects (marked in blue on SD);
Services of the system (marked in green on SD);
Conditions on the services (marked in yellow on SD).

- ☞ For each sequence diagram, underline the system objects and classes, and any actors, with a blue pen.

- ☞ Underline the information exchanged between objects (the horizontal arrows) with a green pen. Consider whether this information represents *messages* or *services* of the system. If the information exchanged is very detailed, at the level of messages, you should abstract several messages together to understand the services they work to provide. Example 2 provides an illustration of messages being abstracted into services. Annotate the sequence diagram by writing down these services, and underline them in green also.

- ☞ Circle any of the following constraints on the messages and services with a yellow pen: restrictions on the number of classes/objects to which a message can be sent, restrictions on the global values of an attribute, dependencies between data, or time constraints that can affect the state of the object. Also circle any conditions that determine under what circumstances a message will be sent. The sequence diagram in Example 2 contains several examples of constraints and conditions on messages. The conditions concerning payment type and payment time determine when messages authorize_payment and new_payment_type_request will be sent , while the restrictions on response_time for message authorize_payment represent time constraints.

2) **Identify and inspect the related class diagrams, to identify if the corresponding system objects are described accurately.**

INPUTS: Sequence diagrams, with objects, services, and constraints marked;
Class diagrams.

OUTPUTS: Discrepancy reports.

- Verify that every object, class, and actor used in the sequence diagram is represented by a concrete class in a class diagram. For classes and actors, simply find the name on the class diagram. For objects, find the name of the class from which the object is instantiated.

  If a class or object cannot be found on the class diagram, it means that the information is inconsistent between both documents. If an actor cannot be found, it may also mean that there is inconsistent information; you need to consider whether the actor must be represented as a class in the system in order to provide necessary behavior. Fill in a discrepancy report describing these problems.

- Verify that for every green-marked service or message on the sequence diagram, there is a corresponding behavior on the class diagram. Verify that there are class behaviors in the class diagram that encapsulate the higher-level services provided by the sequence diagram. To do this, make sure that the class or object that *receives* the message on the sequence diagram, or should be responsible for the service, possesses an associated behavior on the class diagram. Also make sure that there exists some kind of association (on the class diagram) between the two classes that the message connects (on the sequence diagram). Remember that in both cases, you may need to trace upwards through any inheritance trees in which the class belongs to find the necessary features. Finally, verify that for each service, the messages described by the sequence diagram are sufficient to achieve that service.

  Is there a message on the sequence diagram for which the receiving class does not contain an appropriate behavior on the class diagram? If yes, it means that there is an inconsistency between the diagrams. The sequence diagram implies that a behavior must exist, but no such behavior is recorded for the appropriate class on the class diagram. Fill in a discrepancy report describing the problem.
  Are there appropriate behaviors for the system services? If not, it means that no class assumes responsibility for a particular service. Fill in a discrepancy report describing the problem.
  Is there an association on the class diagram between the two classes between which the message is sent? If not, necessary information has been omitted from the class diagram. If a message is exchanged between two classes they must be associated in some way. Fill in a discrepancy report describing the problem.

  Are there any missing behaviors, without which the system service cannot be achieved? If so, it means that there is an omission from the sequence diagram. Fill in a discrepancy report describing the problem.

- Verify that the constraints identified in the sequence diagram can be fulfilled according to the class diagram. Consider the following cases: 1) If the sequence diagram places restrictions on the number of objects that can receive a message, make sure that constraint appears as cardinality information for the appropriate association in the class diagram. 2) If the sequence diagram specifies a range of permissible values for data, make sure that constraint appears as a value range on an attribute in the class diagram. 3) If the sequence diagram contains information concerning the dependencies between data or objects (e.g. “a ‘Bill’ object cannot exist unless at least one ‘Purchase’ object exists”) make sure that this information is included as a constraint on a class or relation on the class diagram. Dependencies between objects may also be represented by

cardinality constraints on relationships. 4) If the sequence diagram contains timing constraints that could affect the state of an object (e.g. "if no input is received within 5 minutes then the window should be closed") make sure that this information is included as a constraint on a class or relation on the class diagram. For example, the class diagram in Example 3 contains a timing constraint for the class "Credit_Card_System" since it applies to all instantiations of this class. The conditional expressions from Example 2 should not appear in the class diagram because they do not affect the state of a class.

Could you find the data but they do not conform to the behavior arguments? Or, could you find the constraints but they do not completely agree in both documents? If yes, it means that the documents are inconsistent. Fill in a discrepancy report to describe the problem.

☞ Finally, for each class, message, and data identified above, think about whether, *based on your previous experience*, it results in a reasonable design. For example, think about quality attributes of the design such as cohesion (do all the behaviors and attributes of a class really belong together?) and coupling (are the relations between classes appropriate?).

Does it make sense for the class to receive this message with these data? Could you verify if the constraints are feasible? Are all of the necessary attributes defined? If not, the diagrams may contain incorrect facts. Fill in a discrepancy report to describe the problem.
For the classes specified in the sequence diagram, do the behaviors and attributes specified for them on the class diagram make sense? Is the *name* of the class appropriate for the domain, and for its attributes and behaviors? Are the relationships with other classes appropriate? Are the relationships of the right *type*? (For example, has a composition relationship been used where an association makes sense?) If not, you have found an incorrect fact because something in the design contradicts your knowledge of the domain. Fill in a discrepancy report to describe the problem.