Reading Techniques for OO High-Level Design

Horizontal Reading

Draft

1) Reading 1 -- Sequence x Class diagrams

1) Take a **sequence diagram** and read it to identify its functionality, all objects and actors involved in, messages and data exchanged, and the constraints associated for each one of the messages and data exchanged. As you read, follow these guidelines to assist subsequent actions:

☞ Underline and number the name of each identified object or actor with a blue pen

☞ Underline and number the names of each identified message (action) with a green pen.

☞ Circle and number each identified constraints with a yellow pen.

2) Now, you should read the **class diagram** to verify if:

- There is some definition for each object or actor used in the sequence diagram

  Is it possible to identify a concrete class for the object or actor?

  ☞ Underline the name of the class and write in the same number given to the objects or actors on the sequence diagram with a blue pen

  ☞ Mark found objects with a blue symbol (*) on the sequence diagram

- There is a behavior for each received message and the constraints can be verified using the object data or attributes

  Does the representation of the class or class hierarchy on the class diagram encapsulate a behavior with the same message name? Is the object receiving some information with this message? Do they conform to the behavior arguments? Does it make sense for the object to receive this message with these data? Could you verify if the constraints are feasible for this object?

  ☞ Underline the identified behavior on the class diagram and write in the same number given to the messages (actions) on the sequence diagram with a green pen

  ☞ Mark behavior found in this way with a green symbol (*) on the sequence diagram

  ☞ Mark data and constraints found in this way with a yellow symbol (*) on the sequence diagram

  ☞ Write down, if possible, how the constraints were verified. You may use a ball pen, but mark it with a yellow pen.

3) Read the same **sequence diagram** again, to look for defects.

   Is there some non (*) marked object or actor? If yes, it means that probably this object or actor is represented in the sequence but not on the class diagram. This information is inconsistent between both documents. Fill in a defect record describing the problem.

   Can you see some non (*) marked message or data? If yes, it means that there is an inconsistency between the diagrams. Therefore, this information could not be found on the class diagram. Fill in a defect record describing the problem.

Guilherme H. Travassos travasso@cs.umd.edu

Could you find the data but they do not conform to the behavior arguments? Could you find the constraints but they do not completely agree in both documents? If yes, it means that probably the documents are ambiguous. Fill in a defect record to describe the problem.

Sequence x Class Diagrams Defects Report

Fill in one record for each defect

Document: Sequence Diagram

Object/Class Name:

Granularity: (actor, object, message, data, condition)

Concept Name:

Type of defect: (inconsistency, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

Document: Class Diagram

Class Name:

Granularity: (attributes, behavior, condition, inheritance, relationship, cardinalities, roles)

Concept Name:

Type of defect: (inconsistency, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

2) Reading 3 -- State x Class diagrams

1) Take a **state diagram** and read it to identify the object, its possible states and the events (possible actions) that drive the object for the states, observing the constraints regarding each one of them.

Could you identify the type of the object that this state machine is modeling? Are all events described? Is it possible to understand what is going with the object just reading this states machine? Are the events associated to object behaviors? ? Have the actions some constraints (pre-conditions, pos-conditions, or invariant) to be accepted? Is it possible to identify whether this object depends on other object state to be driven by events?

☞ Underline and number the names of the identified states with a yellow pen.

☞ Mark the identified events with (E_1, E_2, E_3 …) using a green pen.

☞ Mark the identified constraints with (C_1, C_2, C_3 …) using a blue pen.

2) Read the **class diagram** and look for a concrete class or class hierarchy representing the modeled object, and that encapsulates some attributes to represent the object' states. It's important to observe the presence of behaviors that will change the object' states.

Did you find the corresponding class? Have the class attributes captured the idea of the modeled states? Does the class encapsulate behaviors to deal with the modeled events? Can you identify the right object data used to verify the constraints?

☞ Underline the attributes on the class diagram with a yellow pen and write in the same numbers given to each state on the state diagram that the attribute is capturing. Mark found states with a yellow symbol (*) on the state diagram.

☞ Underline the identified behavior identification on the class diagram with a green pen and write in the same E_i numbers given to the events on the state diagram associated to this behavior. Mark found events and behaviors with a green symbol (*) on the state diagram.

☞ Write down how a condition was verified if that was done. You may use a ball pen, but mark it with an appropriate C_i using a blue pen. Mark found constraints with a blue symbol (*) on the state diagram.

3) Read the same **state diagram** to look for possible detected defects. Fill in a defect record for each one found.

Is there some non (*) marked state? Could you evaluate the importance of this non-marked state? Does it really describe an essential object state? Is the state feasible considering all events 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.

Is there some non (*) marked event? Has this event a behavior association? If yes, fill in a defect record showing the inconsistency between the two diagrams. Is there some non (*) marked condition? Is the condition directly concerned with some object data? If yes, there is an inconsistency between the two diagrams. Fill in a defect record to report the problem.

State x Class Diagrams Defects Report

Fill in one record for each defect

Document: State Diagram

Object/Class Name:

Granularity: (state, event, behavior, condition)

Concept Name:

Type of defect: (inconsistency, extraneous, miscellaneous)

Description:

Document: Class Diagram

Class Name:

Granularity: (attribute, behavior, condition)

Concept Name:

Type of defect: (inconsistency, extraneous, miscellaneous)

Description:

3) Reading 2 -- Sequence x State diagrams

1) Take a **state diagram** and read it to identify the object, its possible states and the events (possible actions) that drive the object for the states, observing the constraints regarding each one of them.

Could you identify the type of the object that this state machine is modeling? Are all events described? Is it possible to understand what is going with the object just reading this states machine? Are the events associated to actions? Have the actions some constraints (pre-conditions, pos-conditions, or invariant) to be accepted? Is it possible to identify whether this object depends on other object state to be driven by events? Could you identify external events driven this object' states?

☞ Underline and number the names of the identified states with a blue pen.

☞ Mark the identified events with (E_1, E_2, E_3 …) using a green pen.

☞ Mark the identified constraints with (C_1, C_2, C_3 …) using a yellow pen.

2) Take the sequence diagrams and choose that ones that use the object modeled by the state diagram. Read each one of them to identify the wrapped up object and the messages (actions) that it receives. Observe if there are constraints associated to the actions.

Is some actor sending message to this object? Could you identify the corresponding event on the state diagram? Is it possible for this event constraint be satisfied? Could you map directly these actions and constraints to those one marked on the state diagram? Is the object driven by the actions for some one possible state?

☞ Underline the identified actions (messages) on the sequence diagram and write in the same E_i number with a green pen. Mark the events with a green (*) symbol on the state diagram. Mark the state that this event is driven the object with a blue (*) symbol.

☞ Underline the identified constraints on the sequence diagram and write in the same C_i number with a yellow pen. Mark the constraints with a yellow (*) symbol on the state diagram

3) Read the **state diagram** to look for possible detected defects. Fill in a defect record for each one found.

Is there some non (*) marked state? Could you evaluate the importance of this non-marked state? Does it really describe an essential object state? Is the state feasible considering all events 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.

Is there some non (*) marked event? Has this event a behavior association? If yes, fill in a defect record showing the inconsistency between the two diagrams. Is there some non (*) marked condition? Is the condition directly concerned with some object data? If yes, there is an inconsistency between the two diagrams. Fill in a defect record to report the problem.

State x Sequence Diagrams Defects Report

Fill in one record for each defect

Document: State Diagram

Object/Class Name:

Granularity: (state, event, behavior, condition)

Concept Name:

Type of defect: (inconsistency, extraneous, miscellaneous)

Description:

Document: Sequence Diagram

Class Name:

Granularity: (behavior, condition)

Concept Name:

Type of defect: (inconsistency, extraneous, miscellaneous)

Description:

4) Reading 4 -- Class diagrams x Class descriptions

1) Read the **class diagram** and find all classes concerned with the problem and their relationships.

☞ Underline and number the name of the class on the class diagram with a blue pen

2) Read the **class descriptions** to find an associated description to each class on the class diagram.

Take a look at the class description. Can this description be used to describe the class that you are considering at this time? Is it using an adequate abstraction level?

☞ Box the class name in the class description with a blue pen and write in the same number given to the class on the class diagram.

☞ Mark found classes with a blue symbol (*) on the class diagram

Make good use of this time and verify if:

- All the attributes are described along with basic types

  Can this class encapsulate all these attributes? Are the attributes associated to feasible/possible basic types? Does it make sense to have these attributes in the class description? Is some attribute missing between the two documents or sounding extraneous?

  ☞ Write down missing and extraneous attributes with a yellow pen

- All the behaviors and constraints are described

  Can this class encapsulate all these behaviors? Is some behavior missing between the two documents or sounding extraneous? Are the behavior descriptions using the same abstraction level? Do the behaviors use the class attributes to accomplish the procedure? Are the constraints reachable using the attributes and behaviors of the class? Do the constraints make sense for this class? Are the constraints depending upon some class attributes? Were they described?

  ☞ Write down missing/extraneous behaviors with a green pen

  ☞ Write down missing/extraneous constraints with a yellow pen

- All the class inheritance relationships are described

  Is there some inheritance mechanism applied to this class? If yes, take a look at the class hierarchy looking for the parents of this class. Is it possible to answer “yes” when asking, “Is the <class name> a type of <parent name>?”? Does it make sense to have this class at this point of the hierarchy?

  ☞ Record any missing/extraneous inheritance relationships with a yellow pen

- All the class relationships (association, aggregation and composition) are described with multiplicity indication

  Take a look at the relationships for this class. Do the compositions seem like a real “whole-part” structure? Does the quantities of objects enrolled seem enough? Are there some differences between the relationship descriptions and their corresponding graphical notation? Were the object roles captured? Does they make sense for the relationship and the objects? Were the cardinalities described into the class description? Is there some attribute grabbing the relationship? Does it use a possible basic type?

☞ Record any missing/extraneous class relationships with a yellow pen

3) Read the **class descriptions** and the **class diagram** to look for detected defects.

Is there some no blue boxed class name on the class description? If yes, it means a defect. This class was defined but not represented on the class diagram. Fill in a defect record for this. Is there some non (*) blue marked on the class diagram? If yes, it means a defect. This class was represented but not described on the class diagram. Fill in a defect record for this.

Are there some attributes or constraints in yellow? It means that attributes or constraints are not truly described. It could be an inconsistency or an incorrect fact between the documents. Fill in a defect record describing this situation and showing which document didn't capture the information.

Are there some behaviors in green? If you found the behaviors on the class diagrams and not on the class description it means an inconsistency between the documents. Fill in a defect record to report this matter.

Is there some inheritance or relationships in yellow? It means that the class hierarchy was not captured. Fill in a defect record reporting this situation.

Class Diagram x Class Description Defects Report

Fill in one record for each defect

Document: Class Descriptions

Class Name:

Granularity: (attributes, behavior, condition, inheritance, relationship)

Concept Name:

Type of defect: (inconsistency, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

Document: Class Diagram

Class Name:

Granularity: (attributes, behavior, condition, inheritance, relationship, cardinalities, roles)

Concept Name:

Type of defect: (inconsistency, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

Reading Techniques for OO High-Level Design
Vertical Reading
Draft

5) Reading 5 -- Class Descriptions x Requirements Description

1) Take the requirements description and read each functional requirement to identify the nouns, which may be candidates for class/objects or attributes, and the verbs or some descriptions of actions, which may be candidates for behaviors.

   Is it possible to identify the relationships between the concepts? Are there some explicit constraints definitions or limitations? Is the requirement describing quantity of anything?

   ☞ Underline the nouns with a blue pen

   ☞ Underline the verbs or actions descriptions with a green pen

   ☞ Underline the constraints with a yellow pen

2) Read the class descriptions to verify if the requirements were captured

   Is the requirement defining an expected functionality? Do the class descriptions contain enough information regarding this functionality? Can you identify classes relating to the requirement? Have these classes names some linkage with the nouns you had marked? Are the classes using ambiguous or contradictory information to describe the concepts?

   ☞ Mark the name of the class in the class description with a blue symbol (*) if there is a corresponding noun in the requirement descriptions

   ☞ Mark the noun in the requirements description with a blue symbol (*) if there is a class definition using the same name.

   Do these classes encapsulate attributes concerned with the nouns you had marked? Are they using compatible types to represent information? Were the constraints and limitations regarding these attributes observed into their definition? Were all identified relationships for these classes regarding this requirement described?

   ☞ Mark the name of the attribute in the class description with a yellow symbol (*) if there is a corresponding noun in the requirement descriptions

   ☞ Mark the noun in the requirements description with a yellow symbol (*) if there is an attribute definition using the same name.

   Do these classes encapsulate behaviors concerned with the verbs or actions descriptions you had marked? Can you understand their description? Can the behaviors receive the right information for accomplish the procedure? Are they specified to return feasible results? Were all identified requirement constraints for these classes described?

   ☞ Mark the name of the behavior in the class description with a green symbol (*) if there is a corresponding verb or action description in the requirement descriptions

☞ Mark the verb or action description in the requirements description with a green symbol (*) if the class template is used.

3) Read the marked requirement description and the class description to look for what is missing or imprecise in the class descriptions.

   Is there some non (*) marked description? If yes, it means that probably something was not captured. Therefore, it should be identified as a defect. Show what is missing filling in a defect record for each not marked noun.

   Could you found some ambiguous or contradictory concept definition? If yes, it means a defect. Show what must be precisely defined filling in a defect record for each ambiguous concept.

4) Read the marked class descriptions to look for what is superfluous in the class descriptions

   Is there some no (*) marked description? If yes, it means that probably something extraneous was inserted. Therefore, it should be identified as a defect. Show what is extraneous filling in a defect record for each no marked noun.

Class Description x Requirement Description Defects Report

Fill in one record for each defect

Document: Class Descriptions

Class Name:

Granularity: (class/objects, attributes, behavior, condition, inheritance, relationship)

Concept name:

Type of defect: (omission, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

6) Reading 6 -- Sequence Diagrams x Use-cases

1) Take a **use-case** and read it to identify the functionality and the nouns that it includes. For each noun identify actions (behaviors) and possible information (data) exchanged with the other nouns. Mark the order (sequence) of the actions. Look for the condition that activates behaviors or actions.

   ☞ Underline and number the nouns with a blue pen as they are found

   ☞ Underline and number the identified actions (behaviors) in the order of actions with a green pen

   ☞ Label the information (data) in yellow as "Dij" where subscripts i and j are the numbers given to the nouns between which the information is exchanged.

2) Read the **sequence diagram** to identify if the corresponding functionality is represented and whether behaviors and data were represented in the right order. Number the actors and classes you have found using the same order you used on the use-case. Apply the same approach you used to mark the behaviors and data exchanging.

   Could you find the same nouns in this sequence diagram? Are they showing up in the same order you had marked on the use case? Is this sequence diagram representing completely the use-case? Are some of the nouns missing here? How these nouns were represented (actors, classes, or attributes)?

   ☞ Mark the nouns on the use-case with a blue symbol (*) that are represented on the sequence diagram.

   ☞ Mark the nouns on the sequence diagram with a blue symbol (*) if they are represented in different order on the use-case or they appear only on the sequence diagram.

   Can you identify the specified behaviors in this sequence diagram? Are the classes/objects exchanging messages in the same specified order that you observed on the use-case? Were the data that messages are carrying forward quoted by the use-case? Are the constraints being observed in this sequence diagram? Is it possible for you to understand the expected functionality just reading the sequence diagram? Is some detail from the use-case missing here?

   ☞ Mark the behaviors with a green symbol (*) and data with a blue symbol (*) on the sequence diagram if they are not represented in the same use-case order.

   ☞ Mark the constraints on the use-case with a yellow symbol (*) if they are not represented in the sequence diagram

3) Read the use-case and sequence diagram to look for what is wrong between the two documents

   Is there some non-marked noun on the use case? If yes, it means that a concept was used to describe functionality but not represented on the sequence diagram. Probably you found an omission between the two documents. Fill in a record to describe the problem.

   Is there some marked noun on the sequence diagram? Does it not appear on the use-case? If yes, it means that a concept was used to describe functionality but not in the same way that was used on the use-case. Probably you have found an incorrect fact between the two documents. Fill in a defect record to describe the problem. Otherwise, you have found a extraneous noun on the sequence diagram. Fill in a defect record describing the problem.

Is there some marked behavior or data on the use-case? Do the classes exchange messages in the same specified order? All transported data are in the right message? Were the constraints observed? Could you find some behavior or data on the sequence diagram not concerned with the use-case? If yes, it means that the sequence diagram is using information incorrectly. Fill in a record reporting this matter.

Sequence Diagrams x Use Cases Defect Report

Fill in one record for each defect

Document: Sequence Diagrams

Class Name:

Granularity: (class/objects, data, message exchange, condition)

Concept name:

Type of defect: (omission, ambiguity, incorrect fact, extraneous, miscellaneous)

Description:

7) Reading 7 -- State Diagrams x Requirements Description and Use-cases

1) Take a **state diagram** and read it to identify the object, its possible states and the events (possible actions) that drive the object for the states, observing the constraints regarding each one of them. Sort the states using the time as the sorting criteria. Identify the initial and end state of the object. Identify for each state the next possible one (adjacent states) when some event has been occurred. Use the events and constraints to decide the next state. Mark the states that have self-transitions.

   ☞ Mark the initial state with a letter I and the final state with a letter F. From the first state mark the next one with I+1 and so on. Use a blue pen.

   ☞ Mark the identified events with (E_1, E_2, E_3 …) using a green pen.

   ☞ Mark the identified constraints with (C_1, C_2, C_3 …) using a yellow pen.

2) Read the **requirements description** to look for some information concerned with this object. Try to find the definitions, if possible, for the object attribute values. Name each combination of attribute values (state). Try to identify which kind of action (event) and under which condition object attributes assume those values.

   Could you find the functional requirements that use this object? Is there some mention for the possible attribute values or some constraint applied for the object? Does the requirement describe some action that affects the object? Is it possible to assume how the object changes its attribute values? And, of course, does the requirement mention that the object must have some specific condition (state) to participate in the functionality?

   ☞ Underline and number the identified state on the requirement description. Mark found state with a blue (*) symbol on the state diagram

   ☞ Mark the identified attributes with a yellow I_i number where the subscript is the number given to the state.

   ☞ Underline the identified actions (events) with a green pen and attach the same I_i number given to the state with which the action is associated. Mark with a green (*) symbol on the state diagram

   ☞ Mark the condition with a yellow C_i where the subscript is the number given to the state with which the condition is associated. Mark with a yellow (*) symbol on the state diagram

3) Look for the use-cases that this object takes part in. Read these **use cases** and try to identify the actions (events) applicable to it.

   Could you find on the use-case descriptions the actions that are applied on the object? Is there some constraint associated with each one of them? Is it possible to identify reading the use-case whether when the object carrying on an action its attributes change the values? Is there some mention about the conditions that the object must have to participate in the use-case?

   ☞ Underline the identified actions (events) with a green pen. Mark with a green (*) symbol on the state diagram

   ☞ Mark the condition with a yellow C_i where the subscript is the number given to the state with which the condition is associated. Mark with a yellow (*) symbol on the state diagram

Guilherme H. Travassos travasso@cs.umd.edu

5) Read the **state diagram** to look for possible detected defects. Fill in a defect record for each one found.

Is there some non (*) marked state? Could you evaluate the importance of this non-marked state? Does it really describe an essential object state? Is the state feasible considering all events and constraints surrounding it? If yes, probably something was forgotten when reading the requirements or use-case. Otherwise, an extraneous fact should be reported.

Is there some non (*) marked event? Has this event a behavior association? If yes, fill in a defect record showing that there is an incorrect fact or extraneous. Is there some non (*) marked condition? Is the condition directly concerned with some object data? If yes, there is an incorrect fact or extraneous here. Fill in a defect record to report the problem.

State x Sequence Diagrams and Use-Cases Defects Report

Fill in one record for each defect

Document: State Diagram

Object/Class Name:

Granularity: (state, event, behavior, condition)

Concept Name:

Type of defect: (inconsistency, extraneous, miscellaneous)

Description: