Appendix A

Rules for Consistency Checking

A.0.4 Intra-Model consistency checking rules

The following rules, including those defined in [uml07] are taken into consideration when checking intra-model consistency. OCL equivalent of some rules are shown here.

*Use case Model*

Rule 1: An actor must have a name.

\[ \text{self.name } \rightarrow \text{notEmpty()} \]

Rule 2: A use case must have a name.

\[ \text{self.name } \rightarrow \text{notEmpty()} \]

Rule 3: An actor can only have associations to use cases, components, and classes. Furthermore these associations must be binary.

Rule 4: An actor must be related to one or more use cases.

Rule 5: Each use case must be used by(associated to) one or more actor.

Rule 6: Use cases can only be involved in binary associations.

Rule 7: Use cases cannot have associations to use cases specifying the same subject.

Rule 8: A use case cannot include use cases that directly or indirectly include it.
not self.allIncludedUseCases() \rightarrow \text{includes}(\text{self})

Rule 9: An actor and use case can be the source or target of an association.

self.source \rightarrow \text{forAll}(p \mid p \rightarrow \text{oclIsKindOf}(\text{Actor}) \text{ or } \text{oclIsKindOf}(\text{UseCase}))

self.target \rightarrow \text{forAll}(p \mid p \rightarrow \text{oclIsKindOf}(\text{Actor}) \text{ or } \text{oclIsKindOf}(\text{UseCase}))

Rule 10: No two use cases in a use case diagram can have the same name, unless it refers to the same requirement.

Rule 11: A use case cannot extend use cases that directly or indirectly extend it.

Rule 12: A use case cannot inherit use cases that directly or indirectly inherit it.

Rule 13: A use case cannot inherit another use case if the corresponding actors inherit from one or the other.

Rule 14: An actor cannot be associated to use cases that have an association re-
lation between themselves.

**Activity Model**

Rule 15: No two activities in an activity diagram can have the same name.
Rule 16: An activity diagram must have an initial node and at least one final node.
Rule 17: Each activity must have an incoming transition and outgoing transition.
Rule 18: Each activity must belong to a class.
Rule 19: Each activity must conform to signature of method corresponding to the activity.
Rule 20: Each activity must be reachable from the initial node.
Rule 21: From each activity, at least one final node must be reachable.
Rule 22: An activity node can be the source or target of information.

context ActivityNode:

\( (\text{self.source} \rightarrow \forall \,(p \mid p \rightarrow \text{oclIsKindOf(ActivityNode)))} \) or \( (\text{self.target} \rightarrow \forall \,(p \mid p \rightarrow \text{oclIsKindOf(ActivityNode)))} \)

### A.0.5 Inter-Model consistency checking rules

For inter-model consistency checking, we have taken into consideration five diagrams, namely, use case, sequence, activity, class and state diagrams. The rules are:

**Use Case - Activity:**

Rule 23: For each actor in a use case diagram there must exist a matching class in at least one activity diagram.

The corresponding OCL constraint is given below:

package BehavioralElements
context classifier

invariant rule-23:

let actor = self.feature → select(f | f.oclIsKindof(Actor)) in
let class = self.feature → select(f | f.oclIsKindof(Class)) in
→ exists(g | g.name = f.name)

Here, f is the set of all actors and g is the set of all classes. f and g are compared to check if they are equal. If equal, then each actor has a corresponding class, indicating that the model is consistent. If not, then the models are inconsistent.

Use Case - Class:

Rule 24: Each actor in a use case diagram must find a corresponding class in the class diagram.

inv rule 24:

let actor = self.feature → select(f | f.oclIsKindof(Actor)) in
let class = self.feature → select(g | g.oclIsKindof(Class))
!exists (g | g.name = f.name)

Here, f is the set of all actors and g is the set of all classes. f and g are compared to check if they are equal. If equal, then each actor has a corresponding class, indicating that the model is consistent. If not, then the models are inconsistent.

Use case - Sequence:

Rule 25: For each actor associated to a use case diagram there must exist a matching actor at least one sequence diagram.

Activity - Sequence:

Rule 26: Each class in an activity diagram must correspond to an object in a sequence diagram.
Rule 27: Each activity in an activity diagram must correspond to a message in a sequence diagram.

**Sequence - Class:**
Rule 28: Each object and message in a sequence diagram must have a corresponding class and method in the class diagram.
Rule 29: Each object in a sequence diagram must reference a valid class in the class diagram.
Rule 30: Every message received by an object must correspond to a method of the object’s class.
Rule 31: For every message between sender and receiver pertaining to different objects(belong to different classes), there should be a relationship between the sender and receiver class. Also, the association should be navigable from the sender to the receiver.

**Activity - Class:**
Rule 32: Each class and activity in an activity diagram must have a corresponding class and method in the class diagram.
Rule 33: For every method, if the sender and receiver pertain to different objects(belong to different classes), then there should be a relationship between the sender and receiver class. Also, the association should be navigable from the sender to the receiver.
Rule 34: For every activity between sender and receiver pertaining to different objects(belonging to different classes), there should be a relationship between the sender and receiver class. Also, the association should be navigable from the sender to the receiver.

**Class - Sequence:**
Rule 35: The public methods of a class in a class diagram must be called in at least one sequence diagram to depict the interaction of classes.

Class - Activity:
Rule 35: The public methods of a class in a class diagram must be called in at least one diagram to depict the interaction of classes.

Class - State:
Rule 36: Each class in a class diagram must have a corresponding state.
Rule 37: Each state in a state diagram must be related to one and only one class.

Activity - State:
Rule 38: Each object (class) and corresponding activity in an activity diagram must match with a state (message) of the class in the state diagram.

Sequence - State:
Rule 39: Each message of an object must match with a state (message) of the object’s class in the corresponding statechart diagram.