Prof. Dr. A. Podelski, Dr. B. Westphal
E. Plaku

Wintersemester 2011/12

Software Design, Modeling, and Analysis in UML

http://swt.informatik.uni-freiburg.de/teaching/winter-term-2011-2012/sdmauml/sdmauml

Exercise Sheet 3

Early submission: Monday, 2011-12-12, 12:00 Regular submission: Tuesday, 2011-12-13, 12:00

Exercise 1 (5/20 Points)

Consider the class diagram in Figure 1. Note that some characteristics of associations/association ends are omitted. We want to distinguish three "severity levels" for omissions:

- Serious: different conventions would lead to different consistent system states.
- Annoying: could lead to difficulties when working with the diagram, for instance, when adding constraints.
- Not severe: none of the two.

For each severity level, give an example in the diagram. Explain your example in your particular understanding of "severity level".
For level "serious", use object diagrams to illustrate the most interesting case where a different choice of defaults makes a difference for system states. Except for level "not severe", propose a fix, which could be concrete for this diagram, a modelling guideline, or a convention.

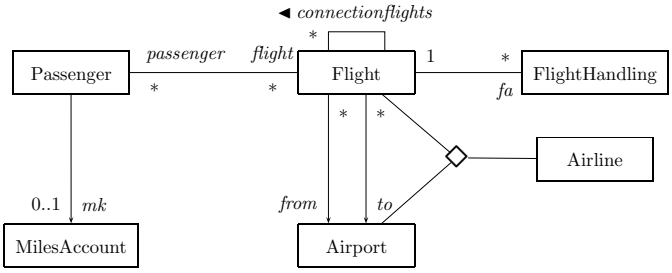

Figure 1: Travelling Domain

Exercise 2 (5/20 Points)

Consider the class diagram in Figure 2.
The class diagram seems to capture information about teams in a league. Assume that the diagram models a particular league where there is the (admittedly strange) rule saying that a player is not allowed to be the goalkeeper ("goalie") in more than 2 years. In order to reflect these rules in the diagram, a modeler proposes to change the multiplicity of the unnamed association accordingly.
Please carry out the change and discuss whether the change has the desired effect, that is, that exactly those system states induced by the class diagram in Figure 2 are consistent that are also respecting the rule.

If you think that the proposal doesn't have the desired effect: do you have an own proposal to fix the issue?

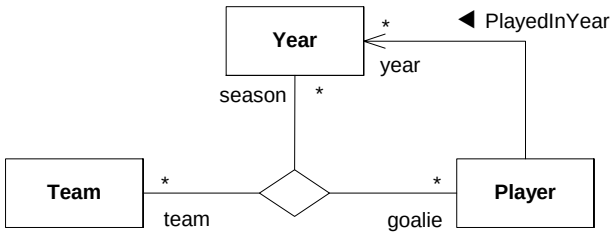

Figure 7.21 - Binary and ternary associations

Figure 2: League [OMG, 2007, 44].

Exercise 3 (10/20 Points)

Choose two (in numbers: 2):

(i) In some older UML textbooks one finds the claim that the two class diagrams shown in Figure 3(a) Figure 3(b) are equivalent. Discuss this claim.

Hint: Discuss semantical equivalence in the course's formal framework with the extended system states from Lecture 07/08. Does, for instance, the choice of defaults for missing characteristics make a difference? If the figures are equivalent formally*, can you imagine differences in pragmatics?*

(ii) In addition to the decorations of association ends discussed in the lecture, UML admits at most one end to be decorated with a hollow or solid "diamond", to indicate "aggregation" and "composition".

Integrate the semantics of aggregation and composition into the course's formal framework.

Hint: That is, first assess what has to be covered (name it, cite it from the standard) and briefly explain its informal semantics as given by the standard.

Recall that we've seen that different "UML things" are of different semantical relevance: it ranges from, e.g., attribute types with prominent semantical relevance, over activeness which we postponed, to reading direction of association names which we don't even represent in the abstract syntax because of its weak semantical relevance.

Discuss into which class of relevance you think aggregation/composition belongs and treat it accordingly. For example, if you opt for prominent semantical relevance, you may have to extend the definition of signatures, to define the mapping from diagram to your extended signature, think about the impact on OCL and well-typedness etc.

(iii) Propose a UML model corresponding to the Java program [Stevens, 2002] in Figure 4.

Hint: As always, explain your model, discuss your choices, etc. Please in particular address the fact that one can identify different "qualities" of relations between objects at run-time. For instance, compare the one between `C` *and* `B`*, or the one between* `B` *and* `A` *when* `doA()` *is called.*

(a) Bidirectional. (b) Unidirectional.

Figure 3: Bidirectional vs. unidirectional links.

```
class A {
  public void doA(B b) {b.doB();}
}

class B {
  public void doB() {;}
}
```

```
class C {
  private A itsA;
  private B itsB;
  public void doC() {itsA.doA(itsB);}
}
```

Figure 4: Java program.

References

[OMG, 2007] OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

[Stevens, 2002] Stevens, P. (2002). On the interpretation of binary associations in the Unified Modelling Language. *Journal of Software and Systems Modeling*, 1(1).