**Lecture "Software Engineering** SoSe 2023

Free University of Berlin, Institute of Computer Science, Software Engineering Group Lutz Prechelt, Linus Ververs, Oskar Besler, Tom-Hendrik Lübke, Nina Matthias

Exercise sheet 6 **Static and dynamic analysis** for 2023-05-29

---

## Task 6-1: Terminology & Static Object Model

*Learning Objectives: Be able to define core concepts. Understand how a UML diagram type can be used differently according to target group and purpose.*

**a)** Contrast the following terms:

*Application domain* classes vs. solution *domain classes*

**b)** For each term from subtask **a),** name a concrete example from the area of your own software project idea.

The *static object model* describes the static properties of the system, e.g. through class and object diagrams. These can be used in different phases of the development process.

**c)** Research and describe the difference between class diagrams as used in the *analysis*, *design,* and *implementation* phases.

Differentiate according to the following aspects: Purpose or intended use, terminology used, semantics of the modeled classes and their properties, semantics of the associations, level of detail of the representation, target group of the diagram.

| | **Development phase** | | |
|---|---|---|---|
| **Aspect** | **Analysis** | **Draft** | **Implementation** |
| *Intended use* | | | |
| *Terminology* | | | |
| *Class semantics* | | | |
| *Association semantics* | | | |
| *Detail level* | | | |
| *Target group* | | | |

Explicitly use the terms from subtask **a)** when describing the respective class semantics.

## Task 6-2: Dynamic object model (1): activity diagram

*Learning Goal: Be able to model a natural language use case as an activity diagram.*

The keywords in parentheses refer to the "Dictionary of Terms" of the book *"The Unified Modeling Language Reference Manual. Second Edition"* by Rumbaugh, Ja- cobson and Booch (starting on page 129). Refer to it if necessary.

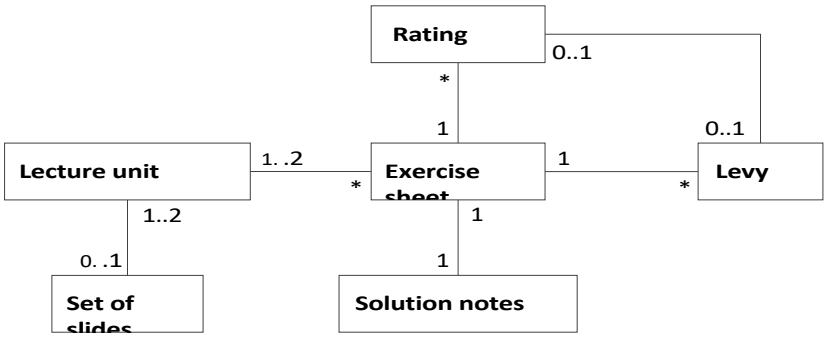
**Task:** Model the following use case (next page) as a UML activity diagram (keyword "activity").

- Use partitions to separate the activities according to the respective executing actors ("activity partition").
- Model both the control and the data flows. The latter can either be written as explicit "pins" at the activity nodes, or as object nodes between the activity nodes ("data flow").

**Overview use case**

1. The lecturer of the course takes an unprepared lecture unit and plans this unit by creating a set of slides. After that, the lecture unit is considered prepared.
2. The lecturer uses the slide set to give the lecture later. He also uploads the slide set to a web server.
3. If a lecture unit has been prepared, it goes to the instructor, who plans a new exercise if a new slip is needed for this lecture unit. (Exercise notes appear weekly, while there are usually two lecture units in a week).
4. The exercise planning results in a new exercise sheet on the one hand and the corresponding solution instructions on the other hand.
5. Students receive the exercise sheet, and work on them, and hand them in to their tutors.
6. Tutors grade the submissions using the solution notes. They then upload the assessments to the KVV.

A colleague has already created a class diagram that specifies the relationships between the domain concepts. Make sure that all objects involved (i.e., ex- emplaces of these domain classes) are included as object flows in the activity diagram.

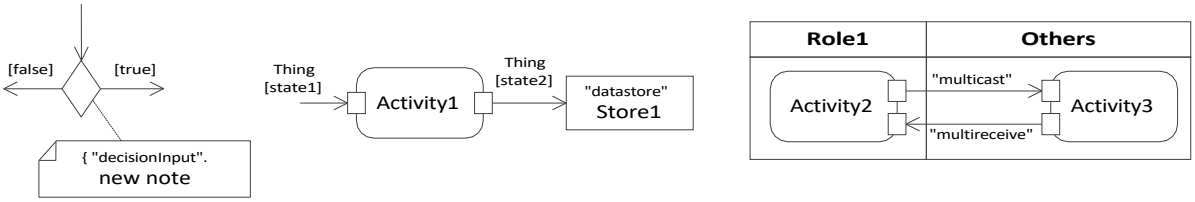



Supplementary explanations of the domain concepts:

- An assessment refers to a specific student; even if the student does not have a submission for a particular exercise sheet, there should be an assessment in the KVV.
- Lecture units have two states: `unprepared` and `prepared`. The following UML

model elements might be helpful for you (notation: see below):

- *Left:* Decisions in the control and data flow can be made by a so-called `decisionInput`, which do not have to be modeled out. ("decision node")
- *Centered:* States of objects can be specified in square brackets. ("ob- ject node")
  You can model the web server and KVV as a `datastore.` ("data store node")
- *Right:* Object flows that go to several addressees can be split up with `multicast` and combined with `multireceive.` ( object flow )



## Task 6-3Æ: Dynamic object model (2): state diagram

*Learning Objective: To be able to model a simple state machine in UML.*

*State diagrams are* used to represent all possible progressions of the states of objects of a class. In software engineering language: A state machine describes the life cycle of an object.

**a)** In *statechart diagrams,* actions/activities can be bound to state *transitions on the* one hand and to *states* on the other hand. How do these two alternatives differ in terms of their notation? What is the semantic difference?

**b)** Create a state diagram for an electric spin dryer (as depicted for drying laundry).

The spin dryer knows three signals (*signal events*) that you can use in your state automates:

- `to make`: Close lid
- `open`: Open lid
- `start`: spin start button is pressed as well

as two sensors (conditions):

- `loaded`: Drum is loaded
- `dry`: Content is dry and

three actions:

- `beep()`: Error display
- `on()`: Engine starts
- `off()`: Motor stops

It should behave as follows:

**V1** The lid of the spin dryer is initially closed.

**V2** The motor starts when the start button is pressed, but only when the lid is closed, the drum is loaded and the laundry is damp. Otherwise, the button is acknowledged with beep.

**V3** The motor stops when the laundry is dry or the lid is opened.

**Reminder**: There are several *events that* can *trigger* a state transition. You will need to make use of the above signals as well as *change events that* listen for a Boolean expression and trigger when it becomes true.

Signal events are simply noted by specifying their name, change events as "`when(<boolean expression>)`".