Model-Driven Engineering, VT2018

Thorsten Berger, thorsten.berger@chalmers.se
Regina Hebig, hebig@chalmers.se
Sergio Garcia, gsergio@chalmers.se

General remarks

The course Model-Driven Engineering uses a common theme for most assignments and for all groups. Consequently, the assignments build on each other and also the delivered reports will be extended by each assignment.

Remarks for all assignment deliverables

- *State the authors of the deliverable (the group members)*
- *Correct language use, no grammar or spelling errors*
- *Reference and describe all figures/tables in the text*
- *Figures and graphs should be readable from a quality perspective*
- *Reference literature in your text where appropriate*
- *Define non-obvious acronyms*
- *The deliverable should be easily readable, understandable and complete*
- *Give arguments for your decisions (also using references)*
- *Show critical thinking*
- *Be prepared to get frustrated if something does not work as you think it should.*

Assignment 2 – Meta-Modeling

Hard deadline for Handin via PingPong: 1.2. 23:55 (CET)

A. Define a Meta-Model for Testing Mobile Robots

You are the CTO of a company producing simple mobile robots, similar to the TurtleBot[1] (which is essentially a Roomba, but without vacuum cleaner). These models differ in terms of the sensors and actuators they have. To automate the testing of the different robot models, instead of laboriously creating complex control programs for the different models and different test tracks, you decide to

[1] http://www.turtlebot.com/

develop a simple language that your test engineers (who are not good programmers) can use to automate the testing.

Below, you find a description of the language that we expect you to model. Your meta-model should at least be able to support the description. Any additions beyond can make your report stronger if they are reasonable. This might be useful in case your model has other weaknesses!

A Test Specification has a name and contains a specification of at least one Robot, of at least one Test Area, and of at least one Test Mission.

A Robot has a name and various Sensors and Actuators, which themselves have a name and an ID of type string, so that these sensors can be referenced in other parts of the model.

A Test Area Specification has an ID (string) and a description, and declares the area in which the turtle-bot should move, including:

- the area's size (length and width in natural numbers);
- the types of waypoints that can be found within the area. These types should not be predefined in the meta-model, so a test engineer creating the model can say that the area has, for instance, a Charging Station, GraspingPlace, MeetingPoint, or Obstacle;
- the concrete waypoints in the area. A waypoint has an ID (string) and a type (a waypoint can have multiple types, e.g. can be a ChargingStation and a MeetingPoint) and the position (x and y position relative to the area, e.g. x=0; y=0 is on corner of the area and x=length; y=width is the opposite corner)

A Mission Specification has a name, and it:

- has a scope definition defining which types of robots (based on what sensors and actuators they have) should do the mission. The scope definition is a Boolean expression over sensor and actuator IDs; see below for hints how to realize it;
- references one Test Area Specification on which the mission should be executed;
- defines the starting point of the robot in this area by referencing a concrete waypoint;
- has a sequence of tasks (see below) that shall be executed

There can be multiple kinds of tasks. Your meta-model should pre-define the following ones, but also allow to extend it later with more tasks. The tasks include:

- "Line" movement, specifies a sequence of waypoints that should be visited by the robot in the given order;
- "Free-order" (you can also call them "shortest-path") movement, specifies a set of waypoints that should be approached by the robots, but the order in which they are approached is not specified. In these cases, the robot should choose the shortest path through the waypoints;

- "Return to start" movement, provides no further waypoints, but specifies that the robot should return to its defined starting point.

Hint for the scope definition, which is a Boolean expression: This expression can just be the ID of a sensor or actuator or a more complex logical expression with the operators AND, OR, and ! (NOT). For instance, if there exist sensors/actuators with the IDs S1, S2, S3, A4, A5, one should be able to specify expressions such as:

(S1 AND !S3) OR (A4) OR (S2 AND A4 AND S3)

This expression is shown in concrete syntax; you need to understand its abstract syntax and model it. It is a bit tricky, but you can represent such expressions with just three classes, with one of them being abstract.

For this assignment, we expect:

1. A meta-model defined in EMF. The meta-model shall contain all relevant classes, associations (including role names and multiplicities), and attributes.
2. A textual description of the meta-model describing what it supports and how to use it. This description should explain the structure of your meta-model and the core design decisions.
3. An example of a test specification, following your meta-model modeled in the generated EMF Editor (from your meta-model). The example shall make use of all the classes and associations in your meta-model. The example does not have to be large, but we expect that all the meta-classes and associations are used.
4. All team members individually attempt to solve the assignment, but of course collaborate and discuss. One solution should be selected as the hand-in, but we expect that everyone attempts on her/his own. Only the selected solution will be graded and get feedback. The other attempts do not have to be correct or complete; only the attempt matters.
5. A report describing the results for assignment 2 as described above, including screenshots/images of your meta-model and the example model. There is no need to do this individually.

Deliverables

- Document (pdf) with the report (maximum 5 pages, font size 12).
- A ZIP archive of all plugins you developed, including your example model.
- The ZIP archive should also come with the attempted/non-selected solutions of all team members (just create a folder for each other attempt)