Assignment 2: Developing a Model Transformation for Visualizing Hardware Configurations

2IMP20 – DSL Design

Introduction

The goal of the second assignment is to develop a metamodel and a model transformation using QVT Operational (QVTo) language. The required model transformation shall transform models expressed in the language from the first assignment (Hardware Configuration Language - HCL) to a graph structure that can be later translated to a format suitable for visualization, for example, Dot Graphviz. This assignment uses Modelware technologies for metamodeling and model transformation.

Ecore and QVTo

Consult the *Tool Guide for DSL Design course Modelware* (available on Canvas, Files/code), sections on Eclipse Installation, Metamodeling and QVTo, in order to install the tools. Furthermore, the same guide gives information about creating a new QVTo project and how to execute transformations on one or more input models.

Assignment

The assignment contains two parts: creation of a metamodel of the source language (HCL) and performing a model transformation from HCL models to a given target language.

Metamodeling

Create a metamodel of the HCL language. The metamodel has to capture all the information that can be expressed in the Rascal based programs. A model in HCL contains a number of computational resources. Each resource describes storage and processing units, and displays (as described in Assignment 1).

Deliverables:

You need to deliver an Eclipse Ecore project that contains the metamodel of the HCL language.

Tips:

- Create a root class in your metamodel that will contain directly or indirectly all other classes by following composition references
- It is **not required** to define any validity (well-formedness) constraints in OCL

Model Transformation

The purpose of this part is to practice with model transformations, in particular transformations written in QVTo. You are provided with an Eclipse project that contains the target metamodel: *nl.tue.dsldesign.graph.metamodel*.

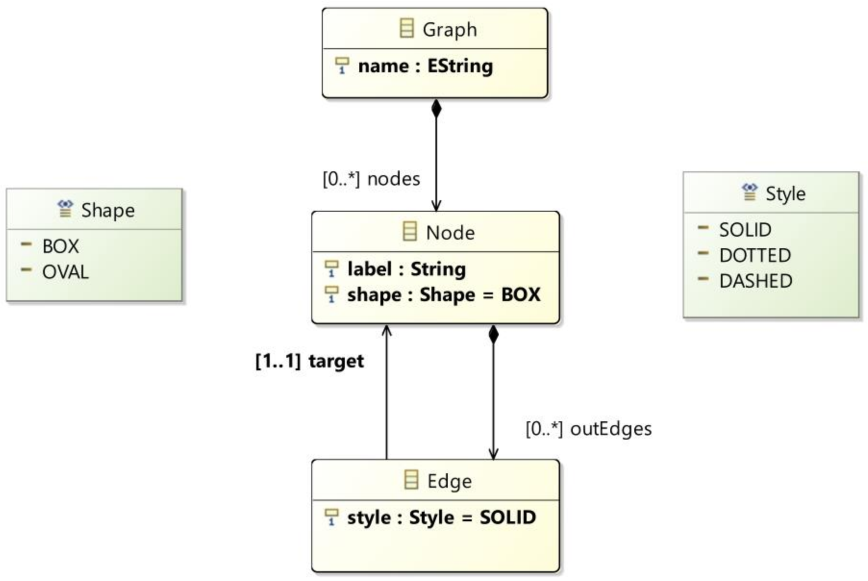


Figure 1Target Metamodel

The target metamodel defines a language for creating graph structures. A graph contains a set of *labeled nodes.* Nodes have a number of outgoing *directed edges*. Nodes have an attribute that indicates the shape used to visualize them (as a box or an oval). Similarly, edges have attribute that indicate the style of visualizing them: solid, dotted, or dashed.

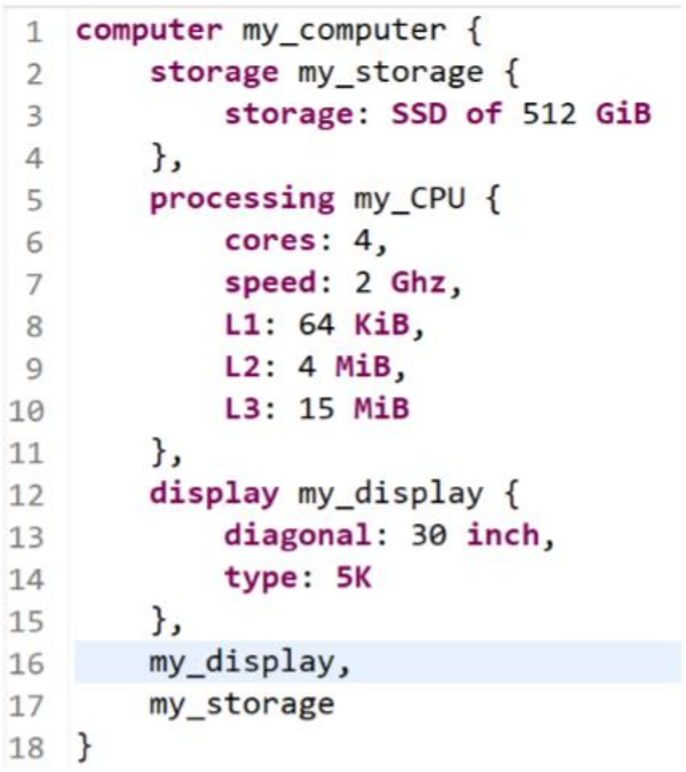
Transformation Requirements

For a given HCL model that contains a description of hardware resources, a graph model shall be created that gives an overview of the resources. The resources are grouped according to the processor speed and the size of the display diagonal. The purpose of the graph model is to be further translated to a format for which visualization tools exist (not part of this assignment).

1. The result graph contains two nodes with box shape labelled 'CPU' and 'Display'
2. For each value of processor speed present in the source model, a box node is created labeled with the speed. There is a solid arrow from the 'CPU' node to each such speed node. For example, if the resources contain two processing units of 2 GHz and 3 units with speed 3,66 GHz, then two nodes labeled '2 GHz' and '3,66 GHz' will be created
3. From every processing unit in the input model, a box node is created labeled with the unit's name. There are dashed arrows from the box node to two oval nodes labeled with the processing unit number of cores and the size of L1 in KiB
4. Every speed node created from requirement (2) has one or more outgoing solid edges to the boxes created from the processing units with the given speed (these boxes are the ones from requirement 3)
5. For every value of the display diagonal size a box node is created labeled with the size. There is a solid arrow from the 'Display' box to the boxes with the sizes. For example, if there are two displays with diagonal 25 inches, a single node labeled '25 Inch' is created
6. From each display in the source model a box node is created labeled with the display name. There is a dashed arrow pointing from the box to an oval node labeled with the type of the display, e.g. '5K'
7. Every node created in (5) has one or more outgoing solid edges to the boxes created from the displays with the given size (display boxes are created in requirement 6)

8. From each computer in the source model, a box node is created labeled with the computer’s name. There are outgoing solid edges from this box to the nodes created from the computer’s processing unit and display (according to (3) and (6))

The following are examples of an input model (in textual syntax defined according to assignment 1) and the expected output model (shown as a graph). Be aware that the real models to be used in the transformation will be in XMI format. The input models can be created as dynamic model instances (see the next section)



```
computer my_computer {
    storage my_storage {
        storage: SSD of 512 GiB
    },
    processing my_CPU {
        cores: 4,
        speed: 2 Ghz,
        L1: 64 KiB,
        L2: 4 MiB,
        L3: 15 MiB
    },
    display my_display {
        diagonal: 30 inch,
        type: 5K
    },
    my_display,
    my_storage
}
```

Figure 2 Example input model (in textual syntax)

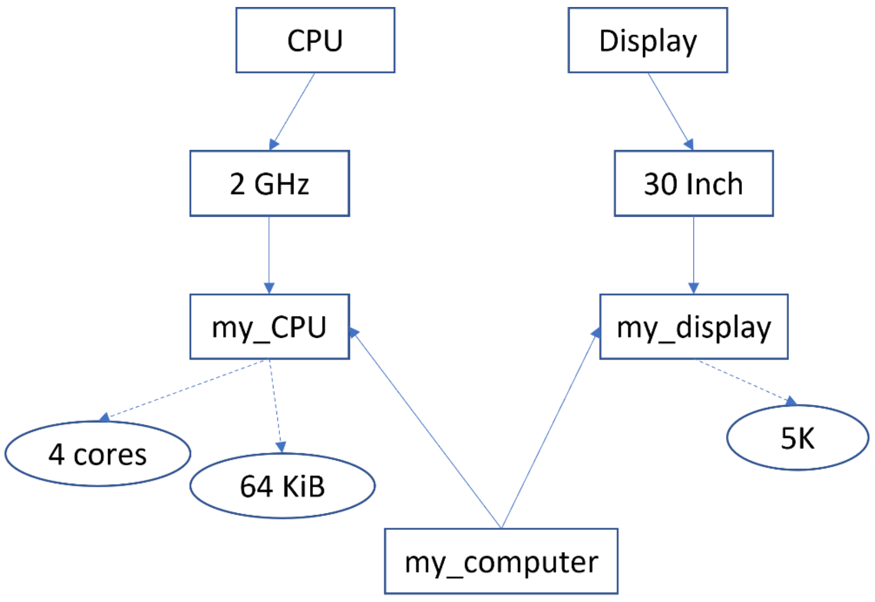


Figure 3 Expected result model

Creating Input Models

For this assignment there will be no concrete syntax for the input and output models. Input models can be created as dynamic instances of the HCL metamodel. Creation of dynamic model instances is described in the Tool Guide, section *Creating Instance Models*. Use this mechanism for creating test models while you develop the transformation and as part of the deliverables.

Deliverables

In this part of the assignment, you are asked to deliver:

- A QVTo Eclipse project that contains a transformation in QVTo implementing the previously formulated requirements
- At least one input model in XMI format that conforms to the HCL metamodel (see the previous section Creating Input Models). The input model(s) shall contain:
  - at least two computers
  - processing units with at least two different speeds
  - displays with at least two different diagonal sizes
- For each delivered input model, the result model obtained after applying the requested transformation

Submission

Each group of two students (as created on Canvas) has to submit a zip file containing the requested deliverables. Submission is expected not later than 23:59 on 17 June via Canvas.