1 Tasks

1.1 Extend the existing DSL

Extend the existing DSL with the following elements: the *motion detector*, *alarm*, *smoke detector* and *controller* elements are inherited from Task while the *server* and *microcontroller* elements are Nodes. Additionally, a *controller* must *controls* other tasks.

1.2 Create an instance model for the DSL

In the runtime Eclipse application, initiate an instance model for your extended DSL. It should contain at least 10 model elements.

1.3 Provide additional validation rules

Create the following validation rules from the following list:

- Every Computer instance should contain only one *critical* task.
- Every Task instance should have the correct severity level.
  - *motion detector*: low
  - *alarm*: medium
  - *smoke detector*: high
  - *controller*: critical
- A *controller* controls tasks in a different node.
- The *availableSlots* attribute of every Computer should be calculated as follows:

$$\mathsf{defaultSlots} - \sum \mathsf{tasks.reqSlots}$$

  where the tasks are the allocated Task instances on the actual Computer.

Validate your instance model (see Section 1.2).

1.4 Create an Acceleo code generator

This task has more sub-tasks to provide some refinement steps but it is possible to implement the whole generator in a single step. The requirements for the generator are the following: (1) generate enum and class signatures, (2) add properties and references with translated multiplicities, and finally (3) generate the run command with the proper number of the existing classes in your model. A simple output is shown in Listing 1.

```
enum MyEnum {
        literal1, literal2
}

abstract sig AbstractMyClass {
        attr1 : one MyEnum,
        attr2 : one Int,
        ref1  : many AbstractMyClass
}

sig MyClass extends AbstractMyClass {}

run {} for 2 MyClass
```

Listing 1: Simple example output

The generator has to be well-commented and well-formatted. It will be awarded with some extra points if you use multiple *templates* and/or *queries*.

Sub-tasks:

- Generate enumerations with their literals
- Generate classes (abstract/not abstract, define ancestor)
- Generate an additional abstract Object type, which is the supertype of all other types
- Generate properties, references (EStructuralFeature) with respected to types (EInt $\rightarrow$ Int)
- Generate run command for 10 Object

Additional task for extra point:

- Order the generation as follows: (1) enumerations, (2) classes with previously defined ancestors

1.5 Create an Acceleo UI Launcher

Generate Alloy code from the previously created instance model in a runtime eclipse.

1.6 Alloy Analyzer

Download the Alloy Analyzer and try it with your generated als code.