Towards A Domain-specific Language For Pick-And-Place Applications

Thomas Buchmann, Johannes Baumgartl,
Dominik Henrich and Bernhard Westfechtel

UNIVERSITÄT
BAYREUTH

Content

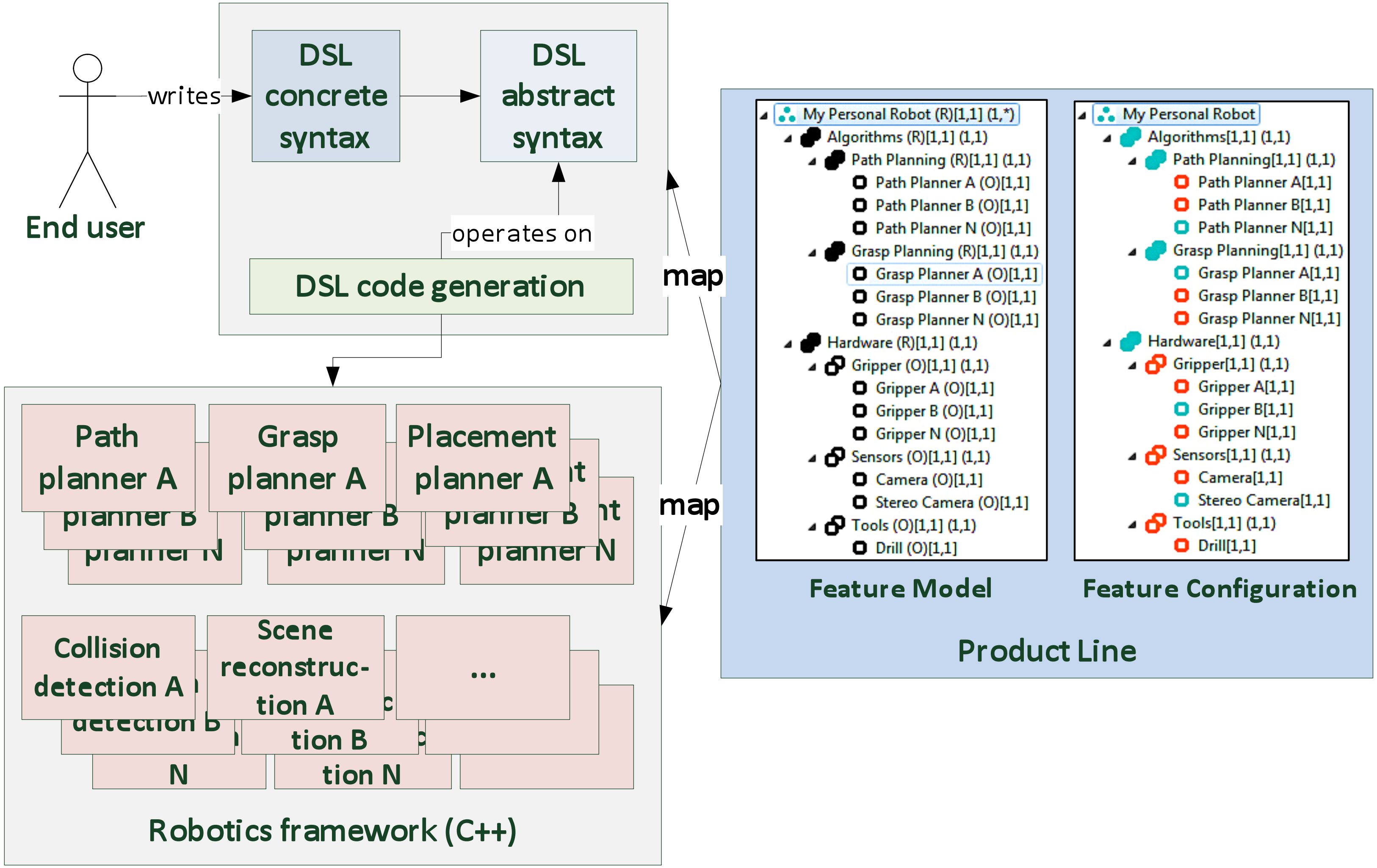
- Motivation
- Current status of the DSL
- Current status of the Codegenerator
- Future work
- Discussion

Motivation

Motivation (2)

- Variability in hardware →
  - almost every robot program has to be written from scratch
  - even if the problem description is similar
- Variability in software →
  - different algorithms for the same problem exist
  - variability in terms of accuracy, performance, sensor input, etc.

Our approach



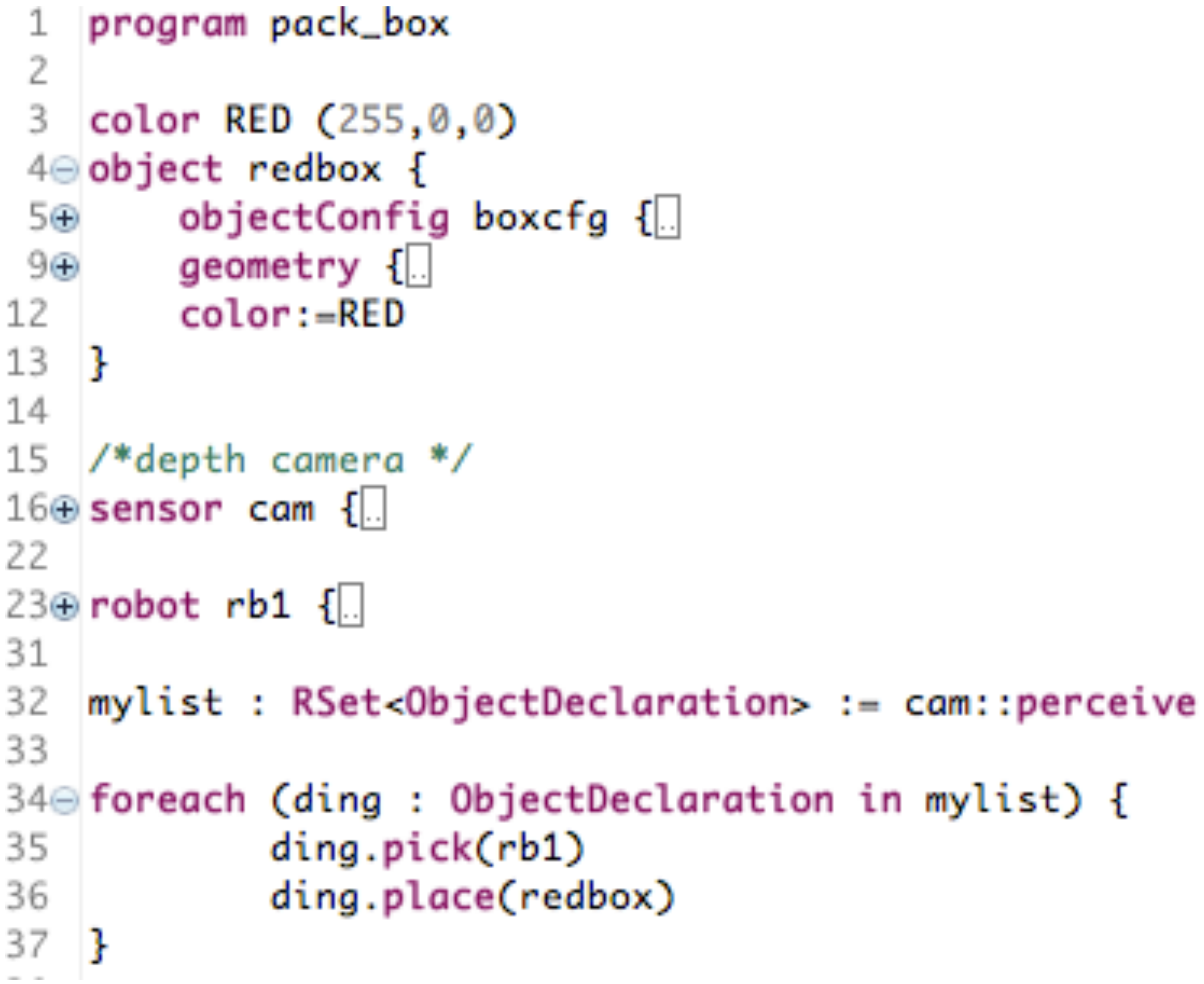
- My Personal Robot (R)[1,1] (1,*)
  - Algorithms (R)[1,1] (1,1)
    - Path Planning (R)[1,1] (1,1)
      - Path Planner A (O)[1,1]
      - Path Planner B (O)[1,1]
      - Path Planner N (O)[1,1]
    - Grasp Planning (R)[1,1] (1,1)
      - Grasp Planner A (O)[1,1]
      - Grasp Planner B (O)[1,1]
      - Grasp Planner N (O)[1,1]
  - Hardware (R)[1,1] (1,1)
    - Gripper (O)[1,1] (1,1)
      - Gripper A (O)[1,1]
      - Gripper B (O)[1,1]
      - Gripper N (O)[1,1]
    - Sensors (O)[1,1] (1,1)
      - Camera (O)[1,1]
      - Stereo Camera (O)[1,1]
    - Tools (O)[1,1] (1,1)
      - Drill (O)[1,1]

- My Personal Robot
  - Algorithms[1,1] (1,1)
    - Path Planning[1,1] (1,1)
      - Path Planner A[1,1]
      - Path Planner B[1,1]
      - Path Planner N[1,1]
    - Grasp Planning[1,1] (1,1)
      - Grasp Planner A[1,1]
      - Grasp Planner B[1,1]
      - Grasp Planner N[1,1]
  - Hardware[1,1] (1,1)
    - Gripper[1,1] (1,1)
      - Gripper A[1,1]
      - Gripper B[1,1]
      - Gripper N[1,1]
    - Sensors[1,1] (1,1)
      - Camera[1,1]
      - Stereo Camera[1,1]
    - Tools[1,1] (1,1)
      - Drill[1,1]

Development approach

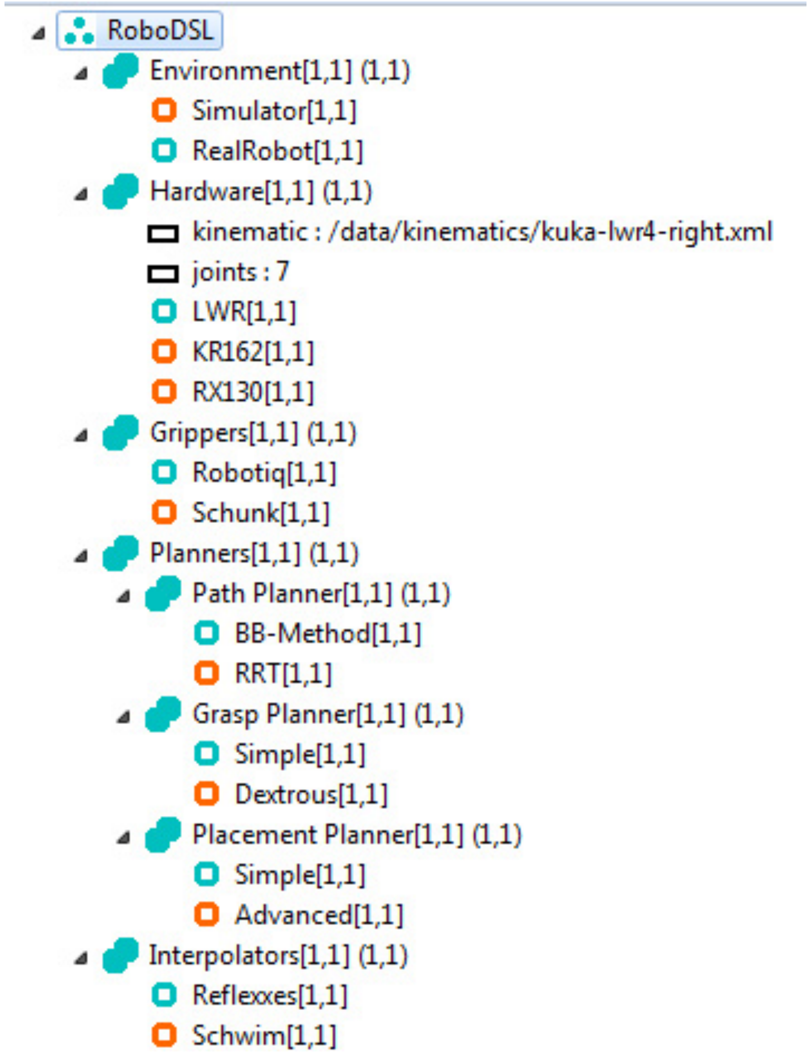
- Model-driven approach using Eclipse Modeling Tools:
  - Xtext for the DSL
  - Acceleo for the Code generator
  - FAMILE for Feature modeling and Feature mapping [Buchmann2012]

DSL for Pick-And-Place Applications



```
program pack_box

color RED (255,0,0)
object redbox {
    objectConfig boxcfg {..
    geometry {..
    color:=RED
}

/*depth camera */
sensor cam {..

robot rb1 {..

mylist : RSet<ObjectDeclaration> := cam::perceive

foreach (ding : ObjectDeclaration in mylist) {
        ding.pick(rb1)
        ding.place(redbox)
}
```

Configurable Code Generator

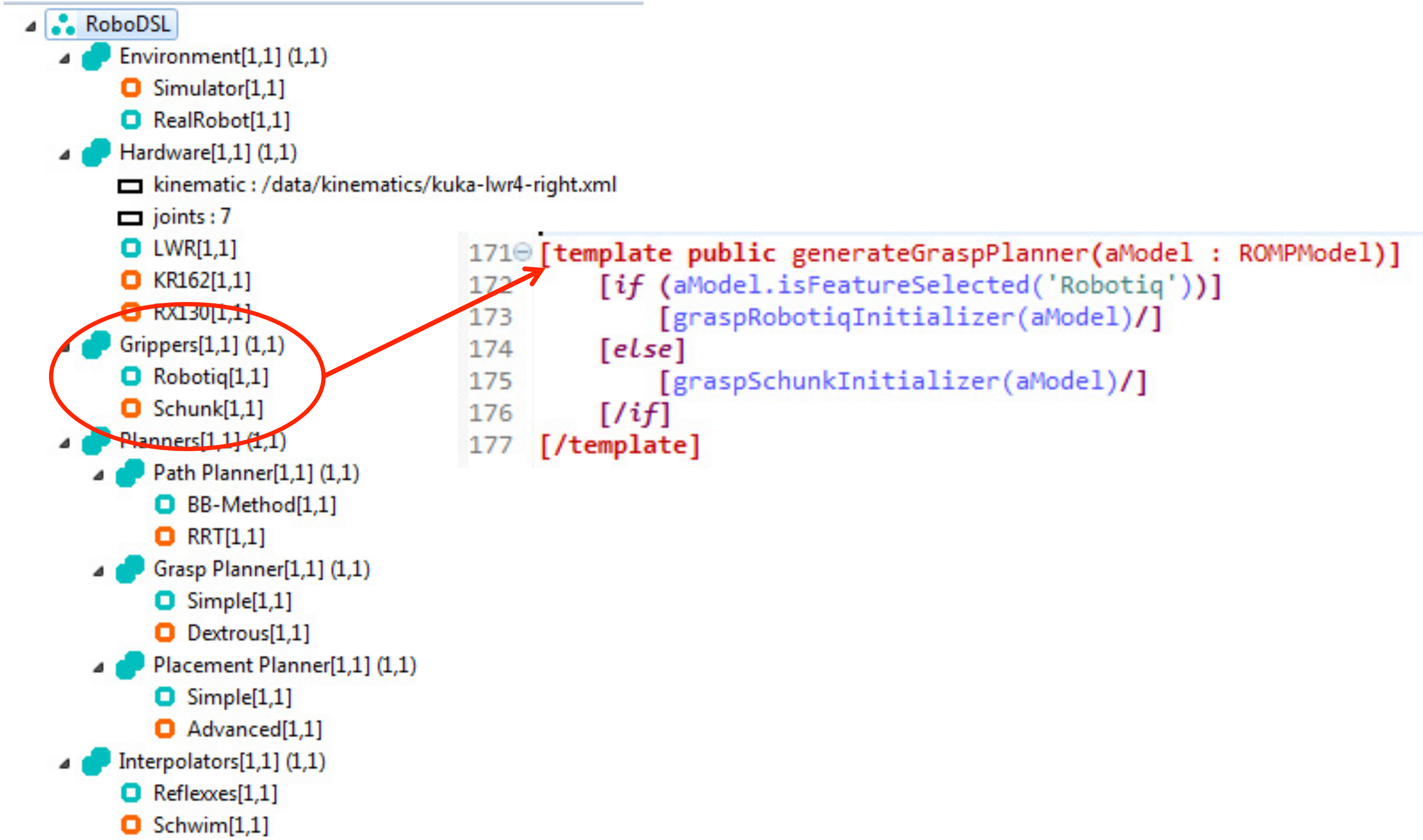


- RoboDSL
  - Environment[1,1] (1,1)
    - Simulator[1,1]
    - RealRobot[1,1]
  - Hardware[1,1] (1,1)
    - kinematic : /data/kinematics/kuka-lwr4-right.xml
    - joints : 7
    - LWR[1,1]
    - KR162[1,1]
    - RX130[1,1]
  - Grippers[1,1] (1,1)
    - Robotiq[1,1]
    - Schunk[1,1]
  - Planners[1,1] (1,1)
    - Path Planner[1,1] (1,1)
      - BB-Method[1,1]
      - RRT[1,1]
    - Grasp Planner[1,1] (1,1)
      - Simple[1,1]
      - Dextrous[1,1]
    - Placement Planner[1,1] (1,1)
      - Simple[1,1]
      - Advanced[1,1]
  - Interpolators[1,1] (1,1)
    - Reflexxes[1,1]
    - Schwim[1,1]

- Acceleo Code generation templates
  - target language C++

- Implementing variability
  - Acceleo queries accessing the current feature configuration

Example



```
RoboDSL
    Environment[1,1] (1,1)
        Simulator[1,1]
        RealRobot[1,1]
    Hardware[1,1] (1,1)
        kinematic : /data/kinematics/kuka-lwr4-right.xml
        joints : 7
        LWR[1,1]
        KR162[1,1]
        RX130[1,1]
    Grippers[1,1] (1,1)
        Robotiq[1,1]
        Schunk[1,1]
    Planners[1,1] (1,1)
        Path Planner[1,1] (1,1)
            BB-Method[1,1]
            RRT[1,1]
        Grasp Planner[1,1] (1,1)
            Simple[1,1]
            Dextrous[1,1]
        Placement Planner[1,1] (1,1)
            Simple[1,1]
            Advanced[1,1]
    Interpolators[1,1] (1,1)
        Reflexxes[1,1]
        Schwim[1,1]
```

```
[template public generateGraspPlanner(aModel : ROMPModel)]
    [if (aModel.isFeatureSelected('Robotiq'))]
        [graspRobotiqInitializer(aModel)/]
    [else]
        [graspSchunkInitializer(aModel)/]
    [/if]
[/template]
```

Future Work (DSL)

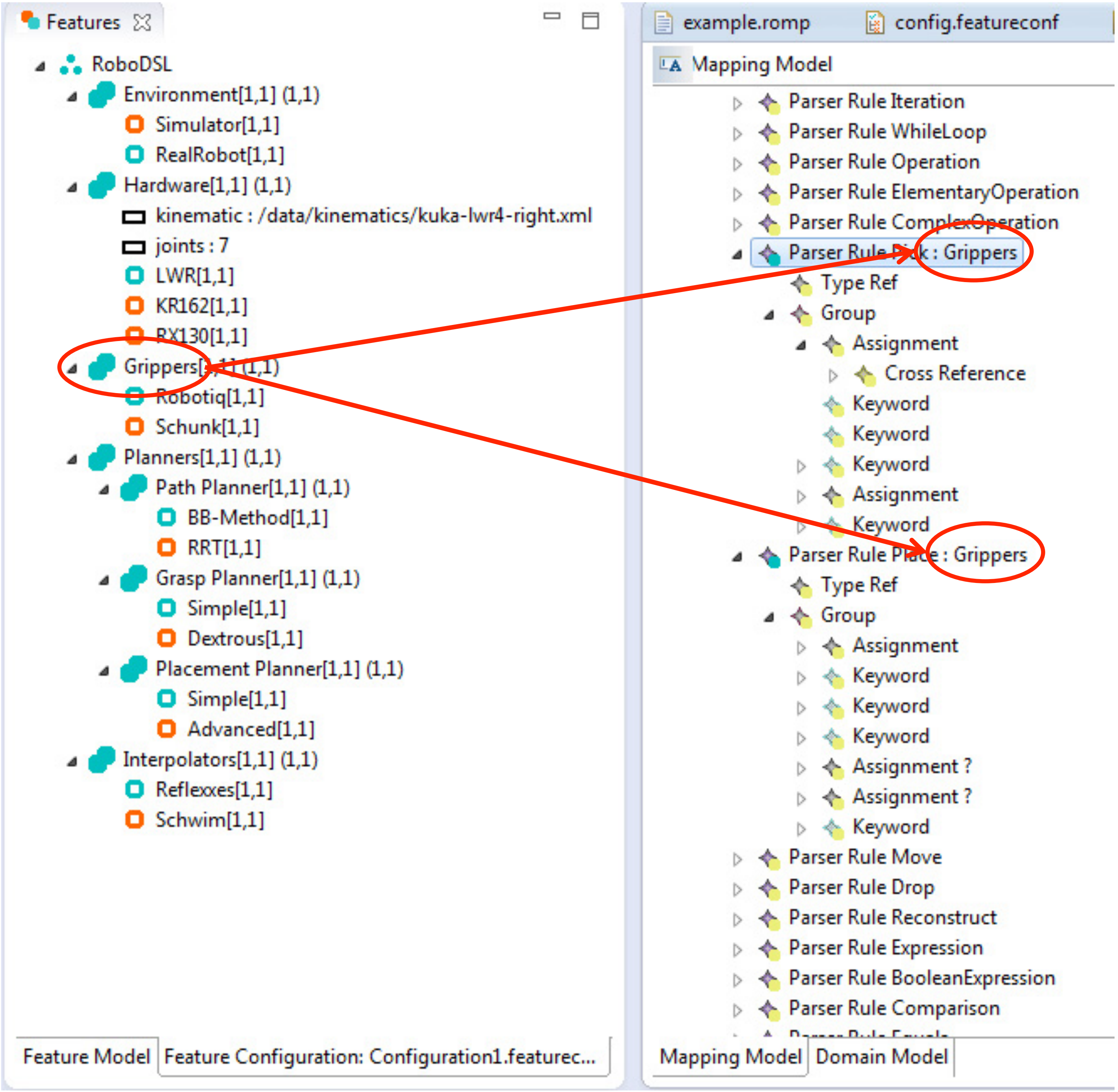
- Formal definition of dynamic semantics
- Implementation of a type system
- Adding language constructs for error handling
- Detecting objects:
  - defining the exact representation of detected objects
  - based upon commonly used features like color, shape, etc.

Future work (Product Line)

- FAMILE allows for mappings between feature models and arbitrary Ecore-based domain models
- Our whole toolchain is based upon Ecore
  - Xtext
  - Acceleo
  - MoDisco (current work in progress for a C++ discoverer)

- Goal: using FAMILE to map features between the different implementation fragments (Xtext grammar files, Acceleo templates and even C++ fragments)

- → Derive a customized toolchain for endusers depending on feature configuration
- → feature configuration is retrieved (semi-)automatically from the connected hardware (Plug & Play)



```
RoboDSL
  Environment[1,1] (1,1)
    Simulator[1,1]
    RealRobot[1,1]
  Hardware[1,1] (1,1)
    kinematic : /data/kinematics/kuka-lwr4-right.xml
    joints : 7
    LWR[1,1]
    KR162[1,1]
    RX130[1,1]
  Grippers[1,1] (1,1)
    Robotiq[1,1]
    Schunk[1,1]
  Planners[1,1] (1,1)
    Path Planner[1,1] (1,1)
      BB-Method[1,1]
      RRT[1,1]
    Grasp Planner[1,1] (1,1)
      Simple[1,1]
      Dextrous[1,1]
    Placement Planner[1,1] (1,1)
      Simple[1,1]
      Advanced[1,1]
  Interpolators[1,1] (1,1)
    Reflexxes[1,1]
    Schwim[1,1]
```

```
Parser Rule Iteration
Parser Rule WhileLoop
Parser Rule Operation
Parser Rule ElementaryOperation
Parser Rule ComplexOperation
Parser Rule Pick : Grippers
  Type Ref
  Group
    Assignment
      Cross Reference
    Keyword
    Keyword
    Keyword
    Assignment
    Keyword
Parser Rule Place : Grippers
  Type Ref
  Group
    Assignment
    Keyword
    Keyword
    Keyword
    Assignment ?
    Assignment ?
    Keyword
Parser Rule Move
Parser Rule Drop
Parser Rule Reconstruct
Parser Rule Expression
Parser Rule BooleanExpression
Parser Rule Comparison
Parser Rule Equals
```

Questions?

- Thank you for your attention!