Programming Refrigerators with Eclipse Xtext

EclipseCon Europe 2011

November 3, 2011
Klaus Dörlfer
Markus Völter
The Setup

• Refrigerators

• One of the most important goals: Energy Efficiency

• Huge number of variants
  • Differing in size
  • Differing in components

• Cooling Algorithms
  • Straight-forward
  • But with a big number of exceptions and interactions
  • Different for almost all product variants
The Candidates

B/S/H/

B/S/H/

B/S/H/

B/S/H/
The Development Situation

- Basic cooling algorithms already well known
- Experiments (trial and error, experience) for even better energy efficiency
- Testing of new ideas in appliances is time-consuming
  - Round-Trip-Times for testing new ideas: >3 weeks
- Building up new variants by using existing algorithms
The New Approach

- A set of cooperating DSLs:
  - DSL for defining appliance variants
  - DSL for defining cooling algorithms
  - DSL for testing the cooling the refrigerators

- Expected to be used by (and developed together with the) domain experts

- An interpreter-based simulator for testing and experimenting with cooling algo’s

- … and a C code generator for the target device
Technologies

Xtext

Xtext/TS
The Benefits

- Allow Domain Experts to get involved in the software development process directly
  - shorter development/turn-around times

- Automatically generate “standardized” C code from models

- Create list of used parameters for appliance variant

- Create additional pieces of documentation
  - State-Charts
  - Flow-Charts
  - List of relevant requirements for variant
Simple Appliance

```java
appliance KIR {
    compressor compartment cc {
        static compressor cl
        fan ccfan
    }
    ambient tempSensor at
    cooling compartment RC {
        light relight
        superCoolingMode
        door rcdoor
        fan rcfan
        evaporator tempSensor rceva
    }
}
```
Appliance + Algorithm

```c
appliance KIR {
    compressor compartment cc {
        static compressor c1
        fan ccfan
    }
    ambient tempsensor at
    cooling compartment RC {
        light rclight
        superCoolingMode
        door rcdoor
        fan rcfan
        evaporator tempsensor rceva
    }
}

start:
    entry {
        if ( RC.rceva->evaTemp < 10 ) {
            state blockCompressorFor10Minutes
        } else {
            state noCooling
        }
    }

state blockCompressorFor10Minutes:
    entry {
        perform after 10 {
            state noCooling
        }
    }

state noCooling with tueroeffnen:
    check ( (RC->needsCooling) && (cc.c1->stehzeit > 333) ) {
        state rccooling
    }

on isDown ( RC.rcdoor->open ) {
    state rclight->active = false
    perform rcfanausschalten after 10 {
        set RC.rcfan->active = false
    }
}
```
**Algorithm + Test**

```java
start:
  entry {
    if ( RC.cheva->evaTemp < 10 ) {
      state blockCompressorFor10Minutes
    } else {
      state noCooling
    }
  }
  state blockCompressorFor10Minutes
  entry {
    perform after 10 {
      state noCooling
    }
  }

state noCooling with trueoffen:
  check ( RC->needsCooling ) && ( cc.c1->stehzeit > 333 ) {
    state rccooling
  } on isDown ( RC.rcdoor->open ) {
    set RC.rclight->active = false
    perform rcfanausschalten after 10 {
      set RC.rcfan->active = false
    }
}

```
DEMO