

**itemis**

**B/S/H/**



## Programming Refrigerators with Eclipse Xtext

EclipseCon Europe 2011

November 3, 2011

Klaus Dörfler

Markus Völter



BOSCH AND SIEMENS HOME APPLIANCES GROUP

## 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

B/S/H/

The Candidates



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

B/S/H/

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

```
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  
  }  
}
```



## Appliance + Algorithm

```

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 ) {
    set RC.rcfan->active = true
    set RC.rclight->active = false
    perform rcfanausschalten after 10 {
      set RC.rcfan->active = false
    }
  }
}

```

## Algorithm + Test

```

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 ) {
    set RC.rcfan->active = true
    set RC.rclight->active = false
    perform rcfanausschalten after 10 {
      set RC.rcfan->active = false
    }
  }
}

```

```

test KIRTest1 for KIR extends KIRSetup {
  step 3

  assert-currentstate-is noCooling
  assert-value RC->needsCooling is false
  assert-value cc.c1->active is false
  assert-value RC.rcfan->active is false

  mock: set RC->needsCooling = true
  step 20

  assert-currentstate-is rccooling
  assert-value cc.c1->active is true
  assert-value cc.ccfan->active is true
  assert-value RC.rcfan->active is true

  step 5
  assert-value cc.ccfan->active is true
}

```

B/S/H/

**DEMO**