Design of Embedded Systems (DES)

Jozef Hooman

http://www.cs.ru.nl/~hooman/DES/

Course 9
12 November 2014
Planning

• 12 Nov: discuss Rover results + explanation Domain Specific Language (DSL)

• 19 Nov: apply DSL to small Lego Rover

• 26 Nov: discuss DSL proposals + requirements Mars Rover

• 3 Dec: start development Mars Rover using DSL

• 10 & 17 Dec: Mars Rover development

• 7 Jan: final demos (?) - development

• 14 Jan: final demos - afternoon
Bluetooth identity – info Harco

• When two nodes are paired name of slave is stored in devices list of master in the lejos OS
  (note: the slave doesn't store a name in its devices list)

• When master program is trying to connect to some named device it will first try matching ones in this list
  Connection will only succeed if pincode is correct

• If you change name of slave but not in master program then it still works because old name is still in devices list of master
  (I guess it is only an alias for the real Bluetooth address)

• If you would remove old slave from list, it won't work anymore; you must pair again from master to slave to get new name
  (and change name of slave in master program)
Subsumption architecture

• See examples:
  Hover over takeControl(), suppress() and action() to see explanation / contract
  ➔ make sure “action()” can be suppressed
    – See Arbitrator
Productivity

Difficult to deliver new software with sufficient quality on time
Model-Based Development

High-level models

switch (currentState) {
case Drive:
    if (lightSensor >= 750) { currentState = Edge; }
    else { speedLeft = UltrasonicSensor - 25;
            speedRight = UltrasonicSensor - 25; }
}

mov i2c_wArgs.Port, PORT_US_SENSOR
set i2c_wArgs.ReturnLen, 1
mov i2c_wArgs.Buffer, i2c_cmd_bufLSWrite1
syscall CommLSWrite, i2c_wArgs
brst EQ, lbl_i2cwOk, i2c_wArgs.Result
add i2cFailedWrite, i2cFailedWrite, 1

01110100100101010100001111011

...
Aims of Model-Based Development

• Detect errors earlier, e.g., using
  – model simulation
  – model checking

• Full code generation from models makes development more efficient

• Improves maintenance & communication
  (model = documentation)
Notes on options for model-based development

- Generation of code from models is essential to avoid that model and code get out of sync.

- Often models are close to programming level; they hardly raise the level of abstraction.

- Most approaches are very generic; applicable to many application domains.
Using a Domain Specific Language

```plaintext
switch (currentState) {
    case Drive:
        if (lightSensor >= 750) { currentState = Edge; }
        else { speedLeft = UltrasonicSensor - 25;
             speedRight = UltrasonicSensor - 25; }
}
```
DSL technology

- Based on Eclipse tooling:
  - Eclipse Modeling Framework (EMF)  
  - Abstract syntax (= Meta-model / Domain model)
  - Xtext  
  - Concrete syntax (= Context-free grammar)
  - Xtend  
  - Validation (= Constraints)
  - Code generation (= Model-to-text transformation)
**Meta level**, for developing the general infrastructure

**Concrete syntax** (.xtext) → **Abstract syntax** (.ecore) → **Code generation** (.xtend)

**Textual input** (.mydsl) → **Model** (.xmi) → **Generated code** (.*)

**Instance level**, for developing a specific system
Assignment 19 November 2014


- **Section 2**: if you already have version for DSL developers: only install Ecore Diagram Editor and Antlr generator library (point 3)
- Perform all actions of **section 3** up to and including 3.7; section 3.8 is optional or can be done later
- See other sections for background info
- Make a first **proposal for a DSL for our first NXT robot** with light, touch and ultrasonic sensors
  → only grammar + typical instance

Send short description of this proposal before Tuesday 18 November 18:00