Way of Working for Embedded Control Software Using Model-Driven Development Techniques

Maarten M. Bezemer
Marcel A. Groothuis
Jan F. Broenink

Control Engineering, University of Twente, The Netherlands
Outline

- Context and Introduction
- Way of Working
- Example
- Conclusions
Need to control embedded (lab) setups:
- Low on resources
- Interaction with outside world (actuators and sensors)
- No / few debugging tools

Writing software becomes complex

Model Driven Development (MDD) helps:
- Validation / debugging on model
- First time right code generation for control software
Controlling actual setups requires real-time levels

- Hard real-time - must meet deadlines
- Soft real-time - should meet deadlines
- Non real-time - everything else
Introduction

- Why do we need a Way of Working (WoW)
  - Even though MDD is helping, it is only the start of the development trajectory
  - Automation of tedious steps
  - Need for a 'known standard' when working in groups
Way of Working for Embedded Control SW Using MDD Techniques

1. Code generation
2. Code processing
3. Cross compilation
4. Deployment
Way of Working for Embedded Control SW using MDD Techniques

1. Code generation
2. Code processing
3. Cross compilation
4. Deployment
Layered approach

- Architecture modelling tool needs support for layers
  - eg for the real-time levels

- Fairly easy to generate code
  - Need a good, supporting framework

- More complex for multiple cores or distributed systems
Decentralised control

- Loop controller for each joint
  - Running of cheap and low power consuming CPUs

- Supervisory control on centralised controller
  - More complex computing unit
  - Provides loop controllers with task related information

- Needs support of the architecture modelling tool
Model optimisation

- Model optimisation needs to be automated
  - Designer has a design point of view
  - Generated code has an execution point of view

- Optimize complex (distributed) models
  - Load balancing
  - Communication
  - ...

- The optimisation algorithm has some extra constraints
  - Input/Output (I/O) is fixed
  - Efficiency of code may vary on execution location
Example

- Simple 2 DOF pan-tilt robotic setup
- Used for educational purposes
  - Practical assignments
    - Easy platform for experimenting
    - Vision-in-the-loop
    - Spot tracking
  - Courses
    - Real-time software development
    - Hardware/Software trade-offs
Example – loop controller

[Diagram showing a loop controller with components such as Joystick, PositionControllerPan, Servo mode, IO, MotorPan, and a PID control system.]
Example - architecture

- Included 20-sim loop control software
- Possible model optimisations / mappings
Conclusions

- Developing complex embedded control software requires less effort with a good tool chain

- Way of Working can be used with own tool chains

- Model optimisation algorithms are useful for complex models running on distributed setups

- Good integration between architectural modelling tool and model optimisation is need