Contents

- Introduction
  - Context: Robots and software
  - Model-driven design
    - From both worlds
- Approach
  - Categorization of the tools
- The overview
  - Highlights from each category
- Conclusion
  - Compatibility and reuse via components
Context, MD SD, Motivation

- Model-driven design: models of
  - Embedded control software
  - Behavior of the robot mechanism
- The combination is relevant
  - Early integration of models
  - Early testing: virtual prototyping
    - Simulation / Co-simulation
  - Systematic design steps
- => SW for Robots can really be engineered
  - Integration phase better supported
Embedded Control Systems

- Essential Properties Embedded Control Software
  - **Dynamic** behavior of the physical system essential for SW
  - Real-time constraints with low-latency requirement
  - Dependability: Safety, Reliability

- Layered Software structure

![Diagram showing layered software structure with Embedded software, I/O hardware, and Process with Actuators and Sensors]

- Model-driven Design
  - Heterogeneous modeling
  - Multiple Models of Computation
  - Multiple Modeling formalisms
Models & Views

- Continuous time / Discrete time
  - Dynamic behavior & control
    - Block diagrams / Bond graphs / FEM models
- Signal processing
- Discrete event
  - Software framework
    - Process / Block diagrams,
    - UML diagrams, …
- Signal processing
- Construction
  - Mechanical Structure
  - Kinematics

Many different types, different Models of Computation (MoC):
Translation not always effective, Co-operation easier

9-5'-11
MD SD for Robotics: an overview
Early Integration

- Approach
  - Heterogeneous models
  - Simulation as verification means

- Modeling
  - Start from overall model
    - Refine – more detail
    - Basis for model repository
  - Combine models of parts
    - Bottom up
    - Co-simulation

- Benefit
  - More effective way of working
Co-design using models (MDD)

- Way of Working
  - Abstraction
    - Hierarchy
    - Split into Subsystems
    - Cope with complexity
  - Model-driven design
    - Design Space Exploration
    - Aspect models
    - Make choices
    - Limit solution space
  - Step-wise refinement
    - Add detail
    - Lower abstraction
  - Implementation
  - Realization
- Concurrent design trajectory
- Early Integration where possible
Concurrent Design Flow

1. Architecture and Dynamic behavior
2. Model-based control law design
3. Software: co-sim, and real-time sim
4. First-time right realization
Implication for tools

- Different nature of models
  - Relevant modeling approaches be covered
    - So, not necessary everything on previous slide
  - Co-operating tools that cover it
    - One simulation model / Co-simulation of more models

- Iterative design
  - Support for hierarchical models
  - Model administration / management
    - Also for experiments and simulation results

- MDD from different / both worlds
  - Application-specific versus Generic
- MDD with different completeness
  - Complete tool, chain of tools / filters, library, framework
Approach: A categorization of tools

- Generic Design tools from Control / Mechatronics
  - Matlab, Simmechanics, Labview

- Generic Design tools / approaches from Software Engineering
  - Mostly UML / SysML based
  - IBM Rational Rhapsody, and – Rose

- Multidisciplinary model-based design tools
  - Dynamics and control in one tool
  - 20-sim, Simulink

- Co-operating model-based design tools
  - Co-simulation is used, combining different models of computation
  - PtolemyII, DESTECS

- Generic / Specific robot software design tools
  - Mostly for a specific brand of robots
  - SmartSoft
Matlab, Simmechanics
- Market leader as control engineering and signal processing tool

Tool specifics
- Signal level
- Mix model and experiment
- Lot of toolboxes

For Robot SW
- Algorithm -> code
- Dedicated HW
Generic from Control / Mechatronics

- Labview
  - Emerged from measurement company

- Tool Specifics
  - Testing, Rtsim
  - Only graphs

- For Robot SW
  - Dedicated HW
  - Various HIL sim
**Generic from Control / Mechatronics**

- **Labview**
  - Emerged from measurement company
- **Tool Specifics**
  - Testing, Rtsim
  - Only graphs
- **For Robot SW**
  - Dedicated HW
  - Various HIL sim

9-5-'11

MD SD for Robotics: an overview
Mostly UML / SysML based
- IBM: Rhapsody, Rose
- Eclipse: GEF, MDF

Rhapsody
- UML IDE for real-time software
- Animation facilities

For Robot SW
- Code gen -> framework
- Adaptable
- Quite heavy
Multidisciplinary MBD design tools

- Dynamics and control in one tool
  - 20-sim, Simulink, Modelica

- 20sim
  - Physics based
  - Mix types
  - Robot models
  - Simulation
  - Animation

- For Robot SW
  - Code gen templates
Co-operating co-modeling

- Co-operating model-based design tools
  - Co-simulation: combining different models of computation
  - PtolemyII, DESTECS
- PtolemyII
  - Research tool
  - Dedicated drawing symbols
- DESTECS
  - Research tool
  - Starting phase
  - 20sim + Overture (VMD++)
- **Generic / Specific robot software design tools**
  - Specific for a specific brand of robots
    - So, not open, not usable for other robots
    - Often not adaptable
    - See ICRA-2010 workshop on Robot Control Architectures: How to Modify and Enhance Commercial Controllers

- **SmartSoft**
  - Communication patterns as core of the robot component model
  - Code-driven to real model-driven
  - Strict component-based approach separates concerns:
    - Algorithms / middleware
    - (see presentation later today)
Frameworks / libraries etc

- Quite a few are around
  - YARP, RoboFrame, OROCOS, ROS
- OROCOS
  - Component: ports are categorized
  - Supports hard-real time
  - Separating normal activity from configuration
  - Can use other frameworks like ROS
- ROS – Willow Garage
  - Open source framework for service robots
  - Oriented from SW engineering
  - Very active, also here at ICRA
Conclusion

- Demands for tools is diverse
  - Different contexts -> different goals / questions
  - -> Different models -> different tools

- Trends
  - Towards model-driven design (and thus support)
    - From document-based; From code-based; From model-based
  - Towards **components** to support reusability
    - **Tools** can interface to each other
      - Better: need to interface, i.e. be compatible
      - End users can choose the best tool they like
    - **Models / Software solutions as components**
      - To effectively reuse existing results
      - To follow / stimulate new developments
Model-Driven Software Development for Robotics: an overview

IEEE-ICRA2011 Workshop on Software Development and Integration in Robotics

Jan F. Broenink, Maarten M. Bezemer

Control Engineering, University of Twente, The Netherlands