A Software Architecture for Model-Based Programming of Robot Systems

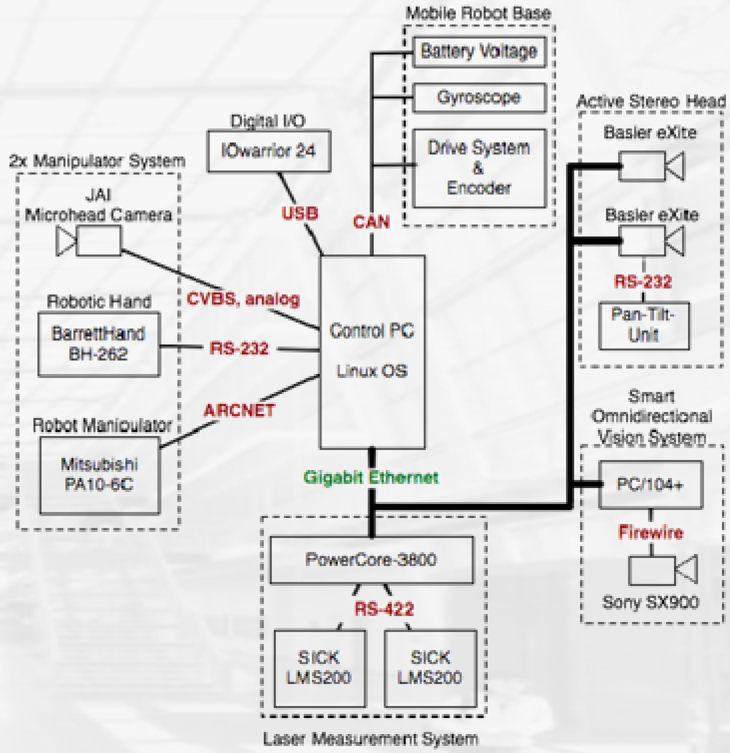
Michael Geisinger

**Joint work with
Simon Barner, Martin Wojtczyk, and Alois Knoll**

German Workshop on Robotics (GWR09)
June 9 and 10, 2009, Braunschweig, Germany

Motivation

- Robot systems consist of highly heterogeneous sub-components
  - Microcontroller platforms
  - Processing power
  - With/without operating system
- Need for tool support
  - Model-driven design at a high level of abstraction
  - Equal design process for components on all levels



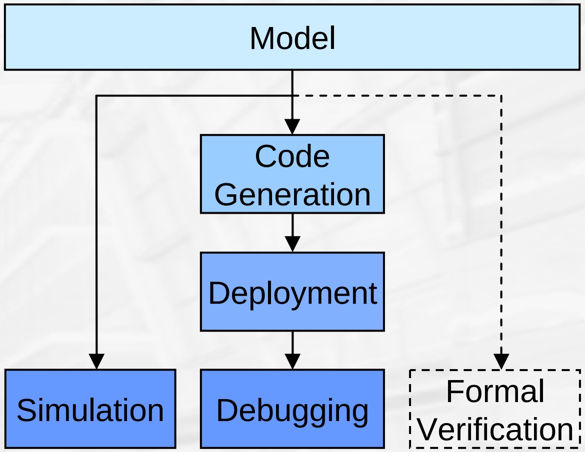
Example: TAMS Service Robot System Architecture (Westhoff, Zhang; 2007)

- Required model semantics
  - Different levels of “model execution” (code generation, …)
  - Use single tool to program many components of a concrete robot system

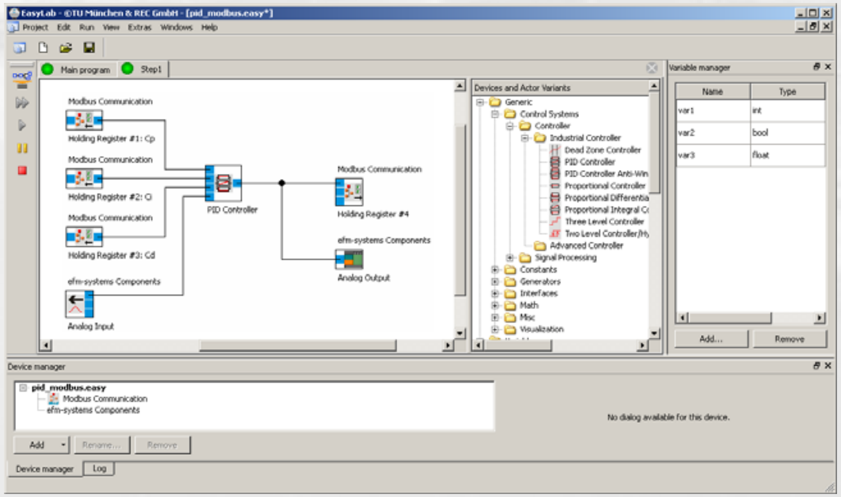
Outline

1. Model-Driven Development Tool EasyLab
2. Modes of Model Execution
3. Conclusion and Outlook

EasyLab: Model-Driven Development Tool



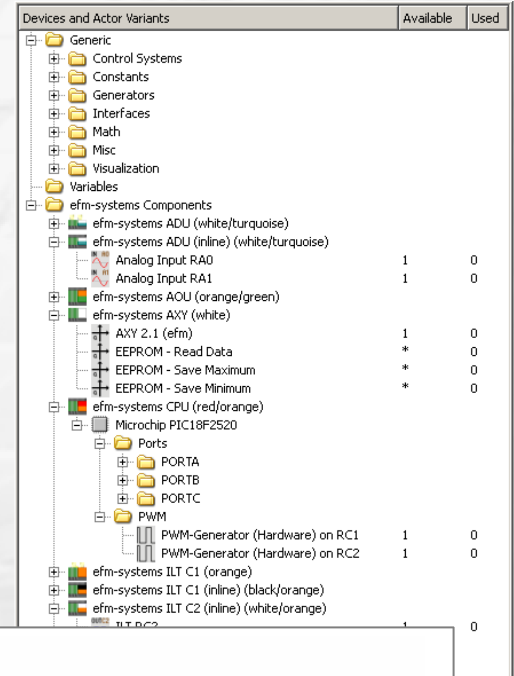
- Software tool for modeling, simulation, code generation, debugging
- Primary focus: mechatronic systems (i.e., small units with “local intelligence”)
- Supports different microcontroller platforms
- Currently used to program smart sensors and actuators

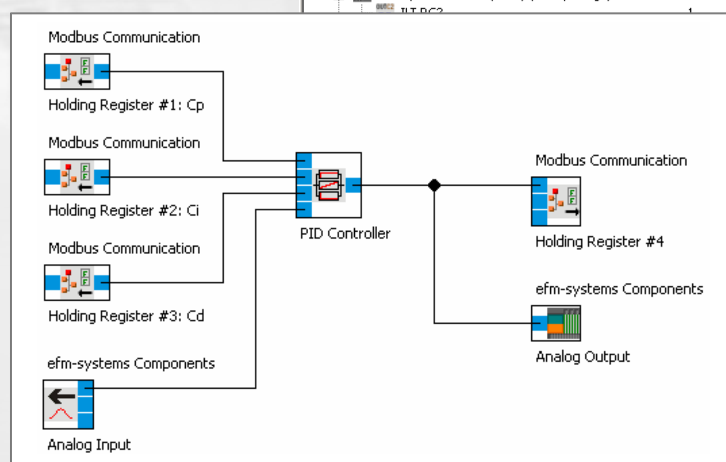


EasyLab: Models

- Device model
  - Specification of (controller) hardware
  - Resource management
  - Easily extensible device library

- Application model
  - System behavior
  - Two visual modeling languages

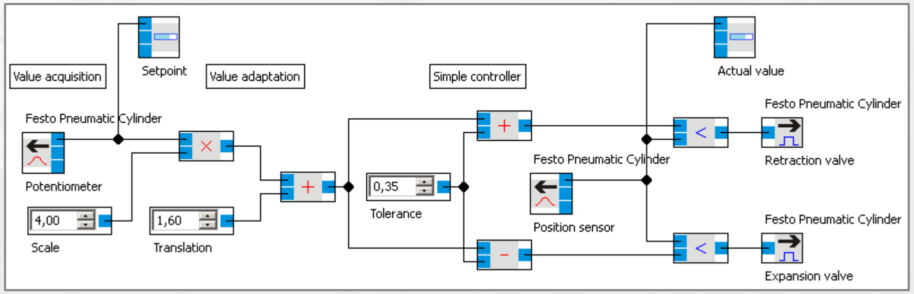


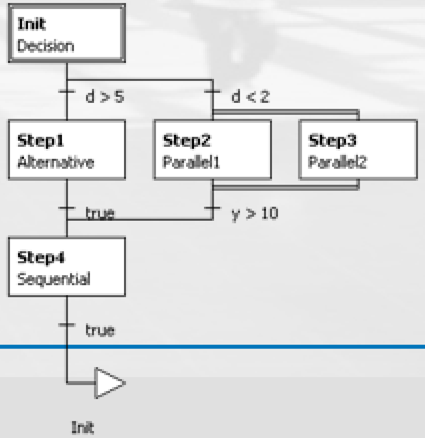


EasyLab: Modeling Languages

- *Structured Data Flow* (SDF)
  - Function blocks as primitives with associated code templates
- Advantages:
  - Widely accepted in application domain (“black boxes“)
  - Static scheduling and memory allocation

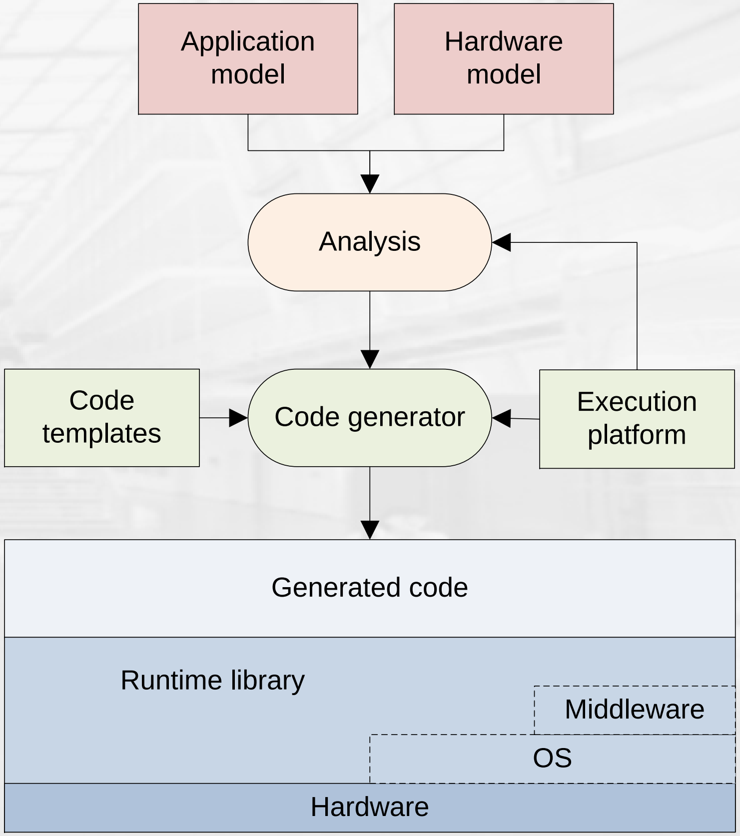
- *State Flow Chart* (SFC)
  - State sequences with Boolean transition conditions
  - Alternative/parallel branches, jumps
  - State as reference to SDF program
- Advantages:
  - Automaton-like semantics
  - Explicit representation of parallelism





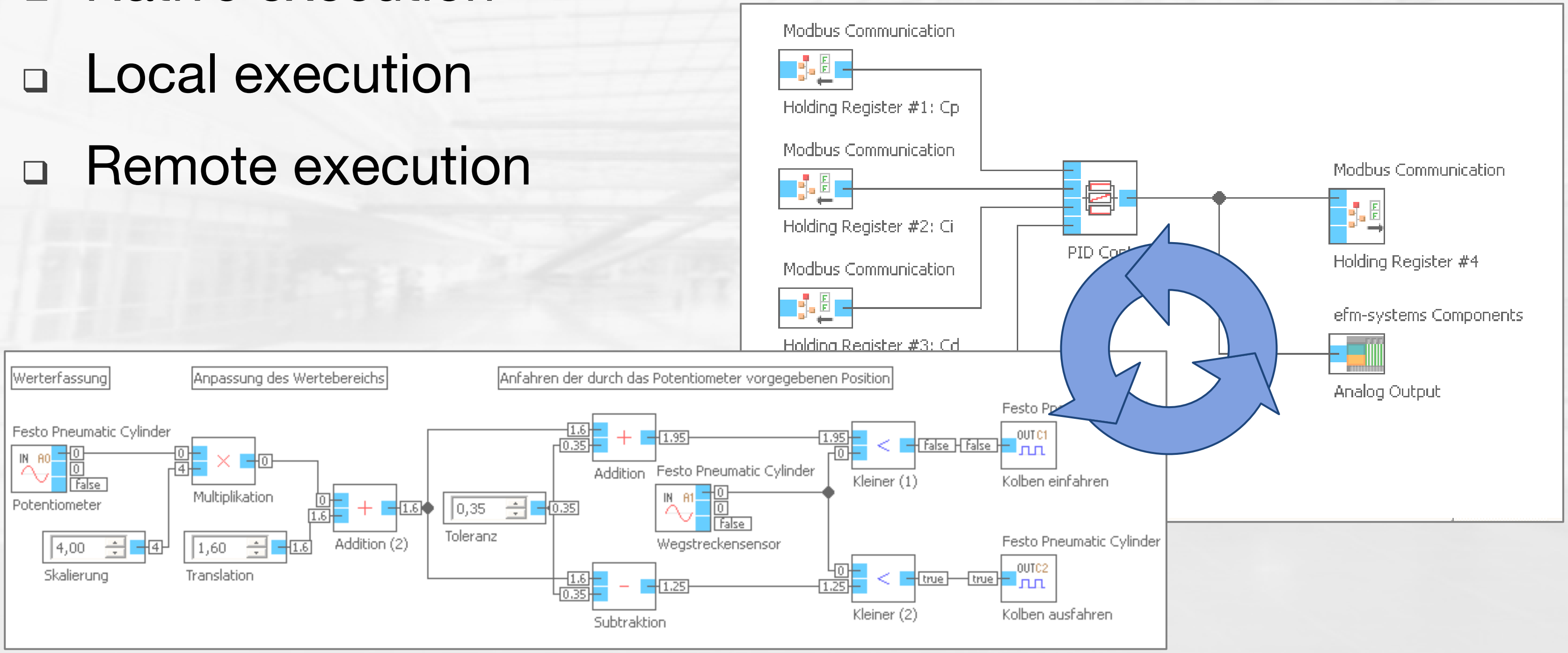
EasyLab: Code Generation

- Efficient and robust implementation on resource-constrained systems
- Approach:
  - **Code templates** for primitive model elements
  - Templates are based on platform-specific **runtime library** (abstraction of low-level hardware features)
  - **Execution platform information** influence code generator

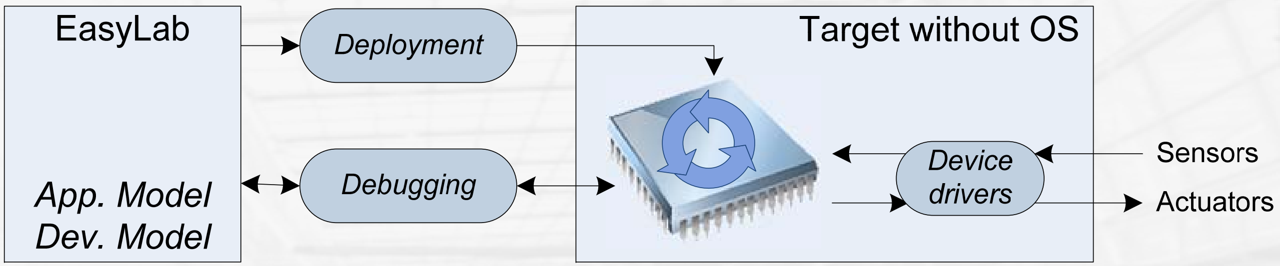


Modes of Model Execution

- Different ways to actually run the modeled application
  - Native execution
  - Local execution
  - Remote execution



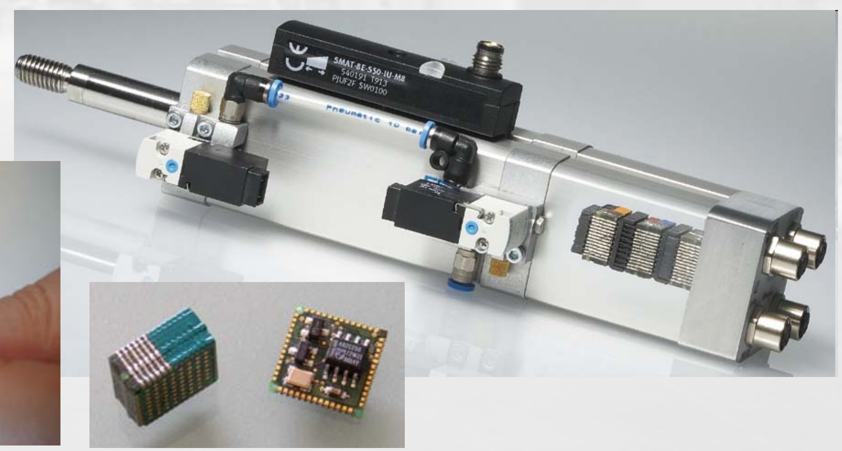
Native Execution

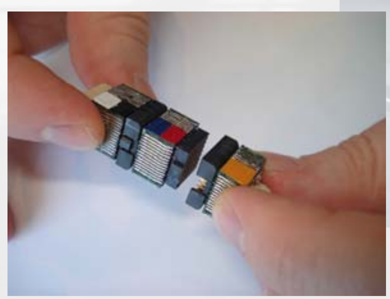


- **Code generation**
- Application executed natively on target hardware
- No operating system required
- Application scenario: smart sensors and actuators without operating system

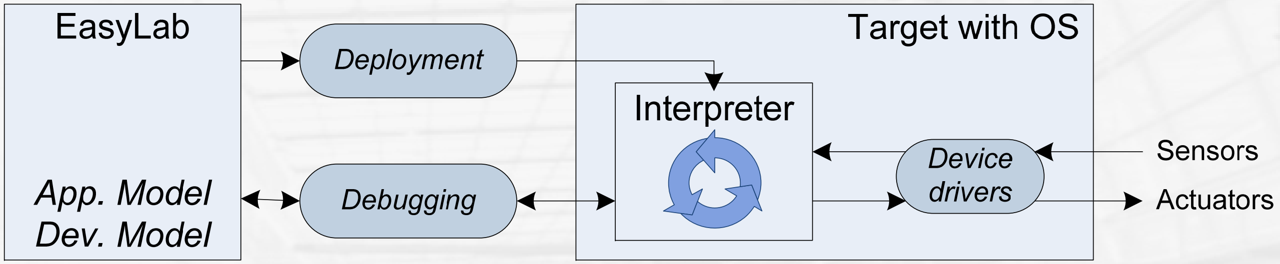
Native Execution Example

- Intelligent pneumatic cylinder: piston positioning
  - Scope: Demonstration of simple controller tasks, “local intelligence”
  - Hardware: cylinder, 2 valves, position sensor
  - Controller: Match-X construction kit (modular micro system) with PIC18F 8-bit CPU





Local Execution



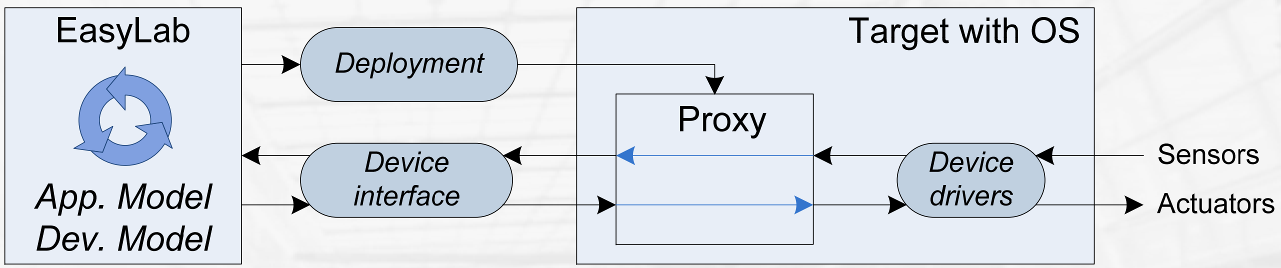
- Application executed by **interpreting application model** on target
- Requires OS and command-line interpreter for target platform
- No code generation required, platform independent application model may be directly modified

- Application scenario: main controllers of robot systems

Local Execution Example

- F5 platform/Leonardo:
  image processing, localization, mapping, path planning
  - Scope: advanced service tasks in industrial or research environments, education
  - Hardware:
    - Laser range scanner, motors
    - Extensions:
      - Camera/force-torque based mobile manipulator
      - Box transportation system
  - Controller: Powerful standard PC

Remote Execution

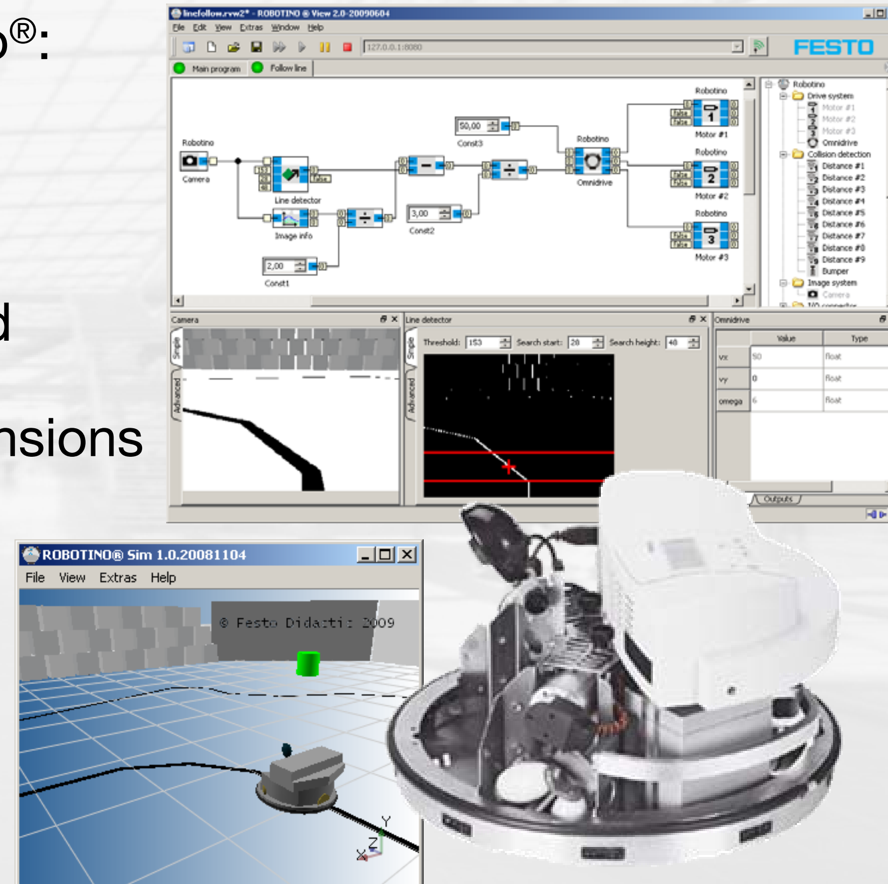


- Application **simulated on a remote machine**
- Control and sensor signals redirected to target
- Target has proxy application that relays between simulator/hardware

- Application scenarios: “slow” target systems, during development (adaptation of model during runtime possible)

Remote Execution Example

- Mobile robot platform Robotino®: camera-based navigation, path planning, odometry
  - Scope: educational area
  - Hardware: 3 motors, infrared distance sensors, camera, bumper, I/O for adding extensions
  - Controller: PC104 (not suitable for computation intensive tasks)



Conclusion and Outlook

- EasyLab is suitable for model-driven development of components of robot systems at different levels
- *Native execution* (code generation) on resource-constrained embedded targets
- *Local execution* (interpretation) on targets with operating system if easy reconfiguration is required
- *Remote execution* (simulation) during development/debugging and on “slow” target systems with OS
- Future work:
  - Addition of distribution model
  - Interfaces for service oriented architecture
  - Optimized support for multi-core architectures