COMDES-II

Formal Definition, Software Engineering and Experimental Validation

Case Study: Production Cell Case Study

Xu Ke, Yu Guo

Supervisor: Prof. Christo Angelov

Presentation Schedule

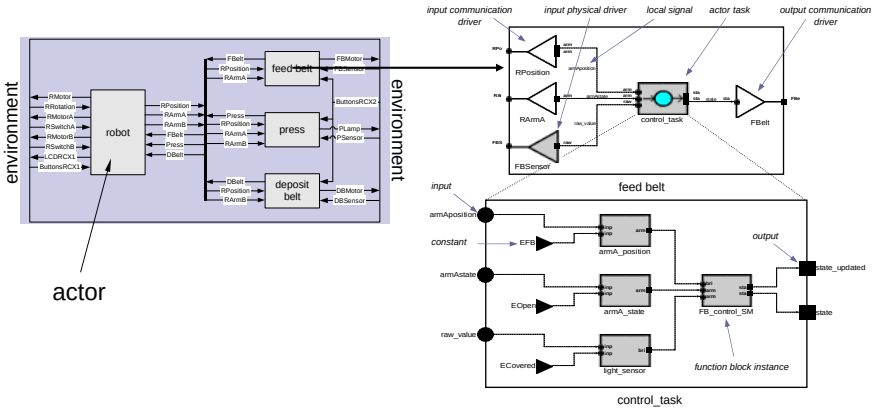
- Introduction of COMDES-II
- Formal Definition of COMDES-II Components
- Software Engineering of COMDES-II
- Future Work
- Experimental Validation of COMDES-II

COMDES-II Framework

- COMDES-II: Component-based Design of Software for Distributed Embedded Systems
- A domain-specific software framework providing modelling concepts and constraints for different issues in the problem space of a distributed real-time embedded system, such as:
  - *Component structures, interaction, hierarchy*
  - *System architecture, concurrency*
  - *Environmental physicality (external events, message exchange etc.)*
  - *Time*

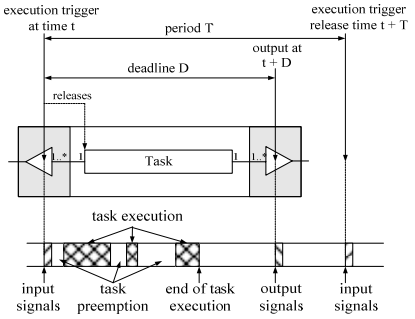
COMDES-II Framework

- Two-level specification of the system architecture

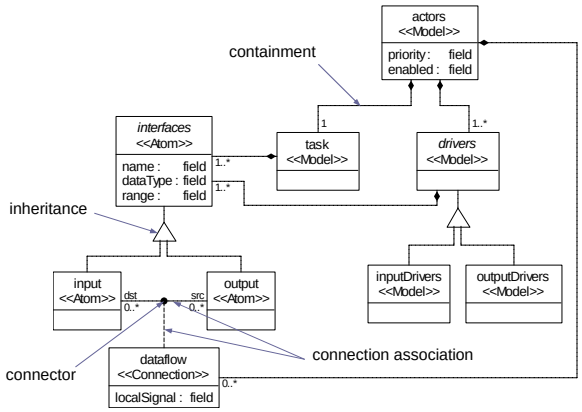


Actor Scheduling and Execution

- Actors are activated by execution triggers that also release the actor task, which is executed in a priority-based timed multitasking environment provided by *HARTEX*$_{TM.}$



Formalization of Actors



Function Blocks

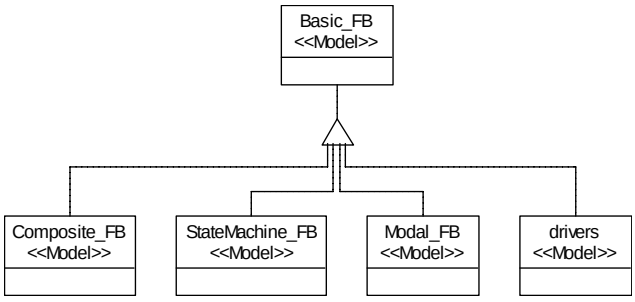
- *Function Blocks* (FBs) are the executable components implementing concrete computation algorithms, which can be used to specify the system functional behaviour.

Continuous behaviour (data flow)	A composition of ***basic*** and/or ***composite*** FBs.
Sequential behaviour (control flow)	A composition of ***state machine*** and ***modal*** FBs.
Modal continuous behaviour	Hierarchical composition of a FB diagram (data flow) into an operation mode of the host modal FB, which is controlled by a supervisory state machine FB.

FB kinds -> FB types (e.g. PID) -> FB instances (e. g. PID_1)

FB instances are used perform the computations.

Meta-type Relationships of FBs

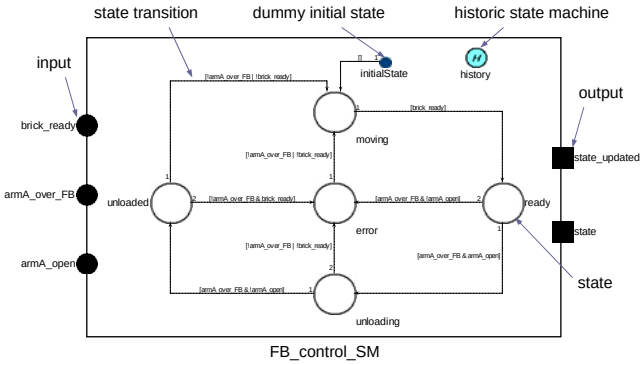


Formal Definitions of FBs

- Meta-models formally specifying the syntax and static semantics for each kind of FB.

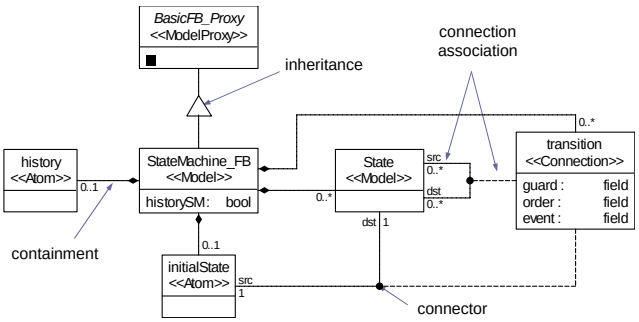
State Machine FBs

- Adopt a clocked event-driven state machine model



Meta-modeling SMFBs

Class diagram specification (FB syntax):



Meta-modeling SMFBs

Constraints in OCL (FB static semantics) :

Syntactic Constraints	OCL Expressions	Applying Object	Checking on Event
The state machine is reactive.	`self.models("State")->forAll(s | s.connectedFCOs("dst")->size >= 1)`	StateMachine_FB	CLOSE_MODEL
The state machine is deterministic.	`self.connectionPoint("src").target().attachingConnections("src","transition")->select(c : transition | c.event = self.event and c.guard = self.guard)->size = 1`	transition	CONNECT
All states are reachable.	`self.models("State")->forAll(s | s.connectedFCOs("src")->size >= 1)`	StateMachine_FB	CLOSE_MODEL

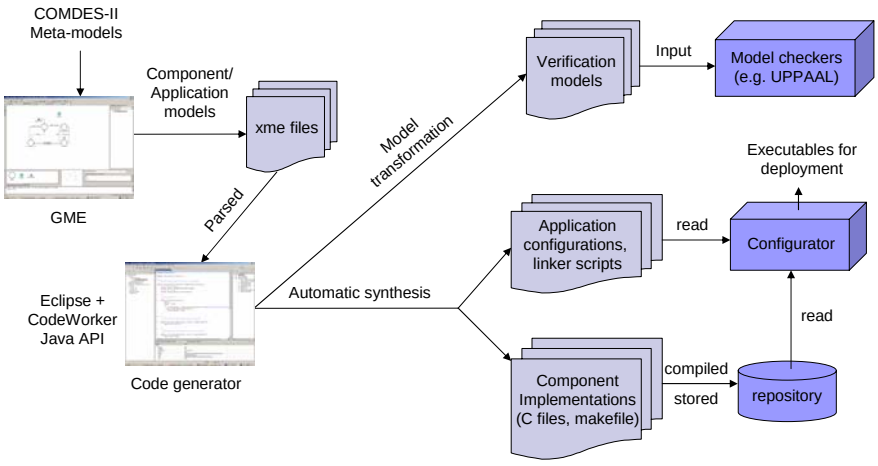
Software Engineering for COMDES-II

- Definition of a domain-specific modeling (DSM) language (i.e. ***meta-model***)
- A DSM ***toolset*** (workbench) including:
  - A ***graphical editor*** supporting graphical design of component and/or system models using the defined modeling language
  - A ***constraint language*** for checking model syntax and static semantics
  - An approach enabling ***formal verification*** (e.g. model checking) of established models
  - A ***code generator*** supporting automatic synthesis of system implementations from system models

Language Workbenches

- Language Workbench: a tool enabling the definition of a DSM language and the development of the associated DSM tools
- Popular language workbenches:
  - Eclipse ***EMF***, ***GMF*** and ***GEF*** (www.eclipse.org)
  - Generic Modeling Environment (***GME***, http://www.isis.vanderbilt.edu/projects/gme/)
  - MetaCASE ***MetaEdit+*** (http://www.metacase.com/)

COMDES-II Toolset

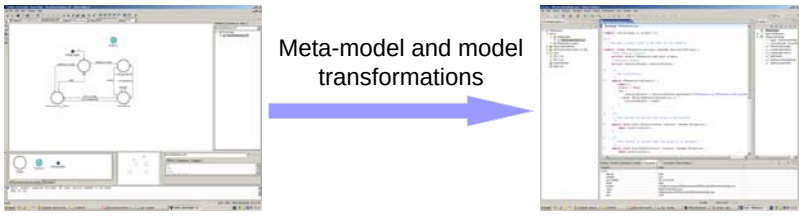


Future work

- Implementing an integrated framework toolset in Eclipse:
  - Graphical editor
  - Syntax & static semantics checker
  - Code generator
  - Integrated model checker

Future work

- Meta-model and model transformations:



Meta-models and models in GME

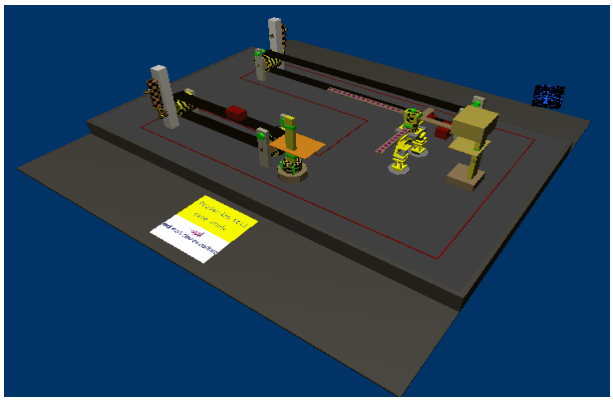
Meta-models in EMF and models in Eclipse

Experimental Validation of COMDES-II

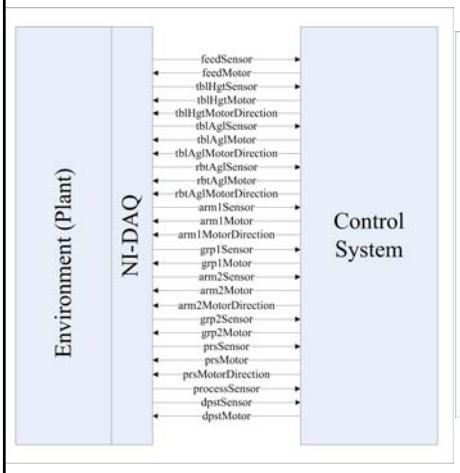
- Case studies:
  - Production cell case study
  - Steam boiler control problem
- Hardware-in-the-loop simulation
- Distributed computer control systems
- Systematic development in a top-down fashion:
  - Control system and plant-system interaction
  - Control subsystems
  - Actors and actor interaction
  - Internal structure of actors (function block diagrams)
  - Uniform implementation (reusable design template and FBs)

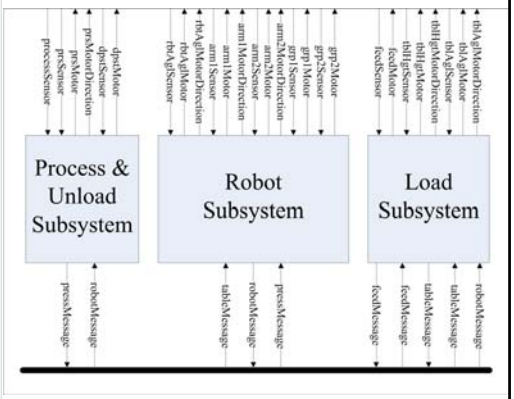
Experimental Validation of COMDES-II

Production Cell Case Study: A safety-critical reactive system with five mechanical units



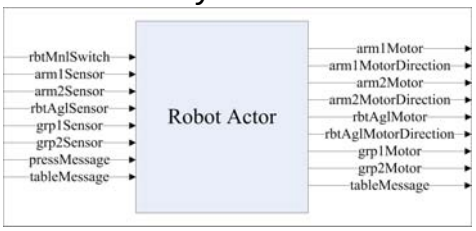
Production Cell Case Study - Environment & Control System



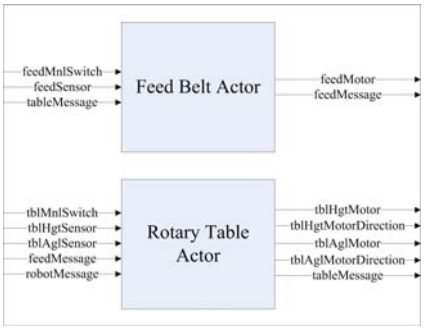


Subsystem specification

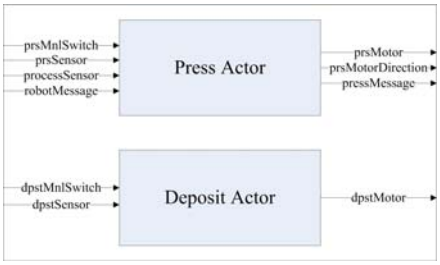
Robot Subsystem



Load Subsystem



Process & Unload Subsystem



Actor interaction

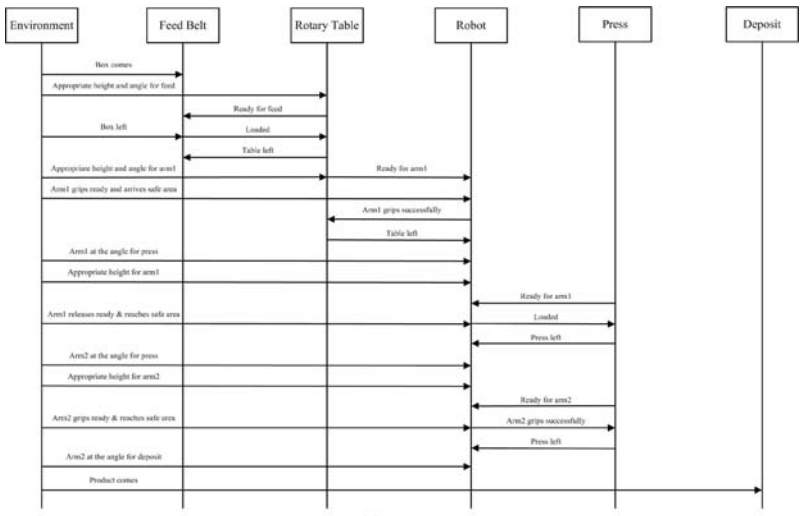
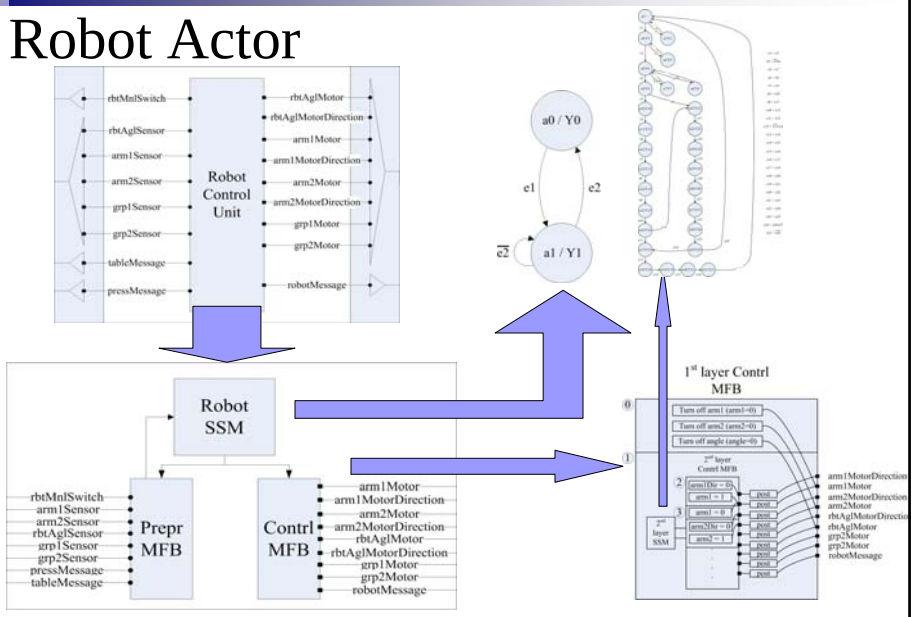
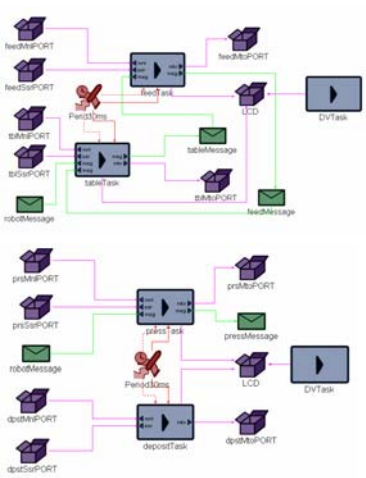


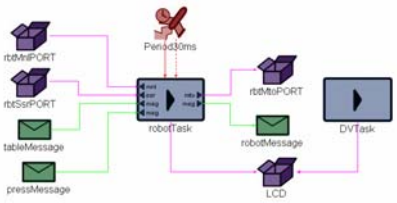
Figure 6.6 Actors interaction diagram

Robot Actor

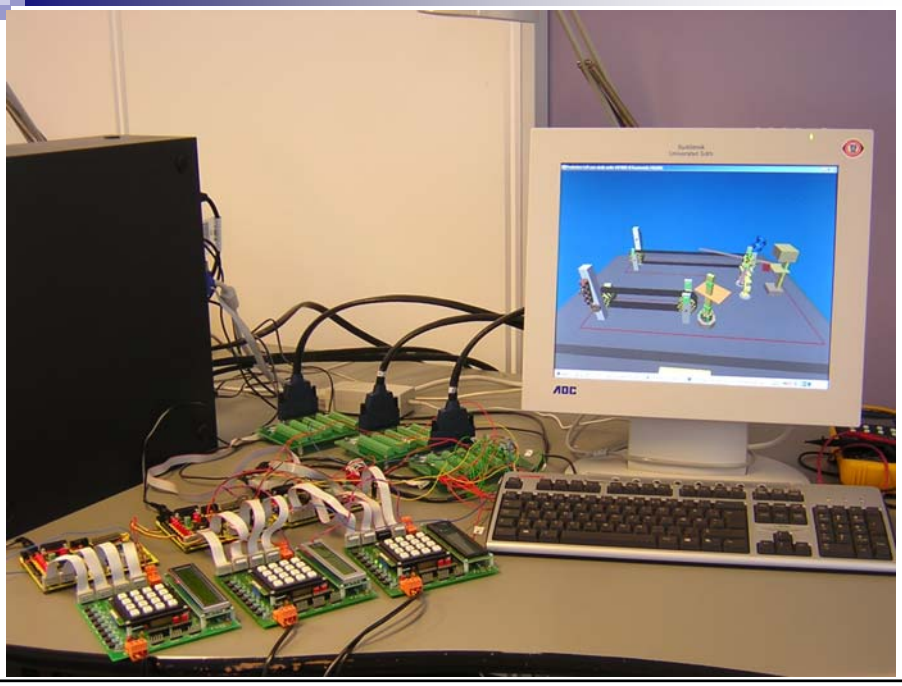


System Configuration in *HARTEX*$_{TM}$





- Load Subsystem
- Robot Subsystem
- Process&Unload Subsystem



Demonstration