A Model-Driven Approach to Embedded Control System Implementation

Jan F. Broenink,
Marcel A. Groothuis, Peter M. Visser, Bojan Orlic

Control Engineering,
CTIT, Faculty EE-M-CS,
University of Twente, Enschede, Netherlands

e-mail: J.F.Broenink@utwente.nl, web: www.ce.utwente.nl/bnk
University of Twente, Enschede, Netherlands

University of Twente, Enschede, Netherlands
Electrical Engineering

• University of Twente
  – Enschede
  – 2500 personnel; 6500 students
  – 5 Faculties:
    » Behavioral Sci; Management & Governance
    » Engineering Techn. Science & technology; EEMathCS

• EE
  – Microsystems; Biomedical Syst; Telecom;
  – Control & Mechatronics; Embedded Systems
  – CE
    » 2.2 full; 2.2 assoc; 10 PhD; 15-20 MSc
    » www.ce.utwente.nl
A Model-Driven Approach to Embedded Control System Implementation

Jan F. Broenink,
Marcel A. Groothuis, Peter M. Visser, Bojan Orlic

Control Engineering,
CTIT, Faculty EE-M-CS,
University of Twente, Enschede, Netherlands

e-mail: J.F.Broenink@utwente.nl, web: www.ce.utwente.nl/bnk
Overview

• Introduction
  – Embedded Control Systems
  – Goal and Benefit

• Design Approach
  – Mechatronic Systems, Embedded Control Software
  – Model-based design; Verification by Simulation

• Modeling formalisms
  – Port-based (bond) graphs – plant
  – CSP diagrams – software

• Embedded Control System Implementation
  – Controller Models, Workflow
  – Stepwise Refinement
  – Tools, Distributed Simulation Framework

• Case
  – Try-outs at lab
Embedded Control Systems

• **Essential properties**
  – Loop controllers **hard** real time
  – **Dynamic** behavior of plant essential
    » **Latency small** compared to time constants plant
    » **Whole system** must be considered
  – Intrinsically **concurrent**

• **Software**
  – User Interfacing, Data processing, **Plant control** (20-30% of code)
  – Reliable, safe, **timing** guaranteed
  – Triggering: **bounded jitter** (isochronous)

• **Hardware**
  – Computer hardware & I/O interfacing
  – Programmable devices
  – Distributed and heterogeneous

• **Plant**
  – Machine, Sensors, Actuators, Power Amplifiers
Layered Structure of Controllers

- **Timing**
  - **Hard** real time
    Safety, Loop Controllers, Sequence Controllers (set point generators)
  - **Soft** real time
    Sequence Controllers, Supervisory Controller, User Interface
Goal, Benefit, Approach

• **Problem**
  - Developing **Reliable** and **Robust** Embedded Control Software is **too** costly and **too** time consuming.

• **Reasons**
  - Complexity
  - Heterogeneity
  - Lack of Predictability
  - Late Integration

• **Approach**
  - Virtual prototyping = **Simulation**
  - Model-level integration discipline-specific issues
  - Property-preserving code generation
  
Shorten design time
Better quality product
Design Approach, Tools

• **Tools needed**
  – Extendable / updatable software
  – Total system (embedded + embedding!)

• **Embedded Control Systems**
  – Dynamic behavior of plant
  – Layered structure of controllers
  – Stepwise refinement
    » Physical Systems modeling
    » Control law Design
    » **Embedded Control System Implementation**
      • Gradually enhance laws to code
    » Realization
  – Verification by Simulation & Formal Checking
Used Model Formalisms

• Demands
  – Overview - Hierarchy
  – Reusability - Interfaces,
    - Implementation independent of connection
  – Simulate-ability - Total model!

• Essential solution
  – Object Orientation, Components
  – Port-based Interfaces

• Choices
  – CSP Diagrams
    » Software structure, CSP-based, DFD, compositional
  – VHDL
    » Configurable: design I/O functionality as if it were software
  – Port-based (Bond) Graphs
    » Object-oriented physical systems modeling
Port-based (Bond) Graphs – Plant

- **Bond Graphs**
  - Relevant dynamic behavior as diagram
    - Directed graph: submodels & ideal connections
  - Domain-independent
    - Analogies between physical domains
  - Restricted number of elements
    - Per physical basic concept 1 bond-graph element

- **Encapsulation of contents**
  - Interface: ports with 2 variables
    - (u, i): voltage & current; (F, v): force & velocity;
  - Equations as equalities (math. Equations)
    - Not as algorithm: \( u = i \times R \rightarrow u := i \times R \) of \( i := u / R \)

- **Simulation (tool)**
  - Compile to differential equations (statements)
  - Simulation = repeatedly execution of statements
CSP Diagrams – Software Structure

• **Dataflow diagrams - CSP**
  – Kind of block diagram
  – Communicating Processes
  – Connections (= channels) transport only
  – Formally verify-able
  – Theory: CSP (Hoare)
  – Checkers – FDR2

• **Encapsulation**
  – Implementation *independent* of communication
    » Channel connections as ports
  – Scheduling at rendezvous: in application

• **Process operators**
  – PAR, ALT, SEQ
  – PRI-PAR, PRI-ALT, EXC
  – Compositional semantics

• **Events**
  – Atomic
  – *Instantaneous*: no duration
  – Variable v over channel c: c.v
  – Direction specific: in?x and out!x
Embedded Control System Implementation

• Models
  – Controller (CSP) -> code on target
  – Plant (bond graphs) -> simulation

• Co-simulation
  – Discrete Event & Continuous Time

• Steps in the method
  1. Physical Systems modeling
  2. Control law Design
  3. Embedded Control System Implementation
  4. Realization
Embedded Control System Implementation II

• Step 3 in the method
  – Plant model OK; Control laws OK
  – Gradually enhance laws to code
    » Integrate control laws
    » Safety, error & maintenance facilities
    » Capture non-ideal components

Working Order
1. Internal checks
2. Formal Check process logic
3. Include (control) algorithms
4. Check target code
Tools

- **20-sim**
  - Bond Graphs, Block Diagrams
  - Continuous Time Simulator
- **gCSP**
  - CSP Diagrams
  - Software Structures
  - FDR2
- **CTC++**
  - CSP concurrency
  - Middleware
- **NWsim**
  - Discrete Event / Network Simulator
  - CTC++ based
- **ForSee (4C)**
  - Configure, Compile
  - Command, Control
20-Sim

Modeling & simulation

- Bond graphs
- Ideal Physical Models
- Block Diagrams
- Continuous Time Simulation
- Animation
- Basic Controller Design
- Code Generation
  » C (CTC++)
  » Token Replacement
    • %NUMBER_OUTPUTS%

- Commercially Available
Parallel Software Modeling

- Software Structure
  » Composition
  » Communication
- Code Generation
  » CSPm
  » CTC++
  » occam

- Prototype
CT-library: CSP-based Software Framework

- **CSP Process**
  - Active object: One thread of control

- **CTC++ software library**
  - Implements as **building blocks-components**
  - Connections as **channels** (synchronous, rendezvous)
    » Link Drivers
  - **Scheduler** included (kernel-like)
  - Runs on Windows, DOS, RTAI (linux), ADSP

- **Prototype**
  - Data-flow model

  Data -> rendezvous
  Process A  Process B

  System 1  System 2
  distributed / heterogeneous
NWsim: Distributed Simulation Framework

• CSP approach
  – All parts are CT processes
  – Remote Channels couple to Fieldbus
  – Network Simulator based on TrueTime

• Time Synchronization
  – Prioritized Parallel
  – Rendezvous in Timer Channels
  – SimTimer advances time

• Interaction continuous – discrete parts
  – Basic integration methods (continuous part)
  – Plant model calculations on demand

  – 1st try out
Details / Limitations

• Assumptions
  – Software execution time ignored
  – Execution should be fast compared to communication

• Adaptations to TrueTime
  – Change of environment
    » CT Process, Interface with Timer,
    » Remote Channel driver
  – Functionality
    » Node numbering and checking
    » Input queue sorting on priority

• Towards real-time
  – NDD of Remote channel & SimTimer -> Real versions
Network Simulator - Case

- **Simulator OK**
  - compared with traditional
- **Network parameters**
  - Influence behavior
  - Optimal via simulation
ForSee: Connect, Compile, Configure, Command

Code -> Target
- Connect
  » Model variables to target signals
    • Token replacement
  » Keeps model free of target anomalies
- Guides
  » Compilation, Configuration, Logging specification
- Prototype

Target template:
- HW descriptions
- Target options
- Logging capabilities

20-Sim

Target Connector

Compiler assistant

Deployment manager

Real-time logger

Target(s)

Send to target(s)

Start/stop

Modify parameters

Token replacement
- Keeps model free of target anomalies

- Connect

- Guides

- Prototype
ForSee: Connect, Compile, Configure, Control

Model → Code generator → Sources → Code compiler → Executable → Deployment → Running task

**Code generation**

Connecting model inputs to I/O hardware

Compiling the generated code

ADSP board hardware

and outputs to I/O hardware
ForSee: **Connect, Compile, Configure, Control**

**Deployment manager**
- Check target (status)
- Upload
  - Executable
  - Configuration (parameters/hardware)
- Task control
  - Start/Stop
  - Data logging control
- Retrieve log data
- Modify/inspect parameters
ForSee: Connect, Compile, Configure, Control

- Real-time logging
- Work in progress
Further tool integration

• **Current tools cover design flow**
  – Cooperate smoothly
  – Still separated per discipline

• **Real cooperative design**
  – One-model – One-tool approach not feasible
    » Multiple view modeling (& tools)
  – Tightly coupling needed also on model level
    » Check cross-domain aspect relations
  – Co-simulation on execution level
Design methodology

Automatic Control View

Dynamic View

Hardware View

Discrete Event View

Robustness View

Reliability View

Multidisciplinary Core models

Co-simulation
Verification
Consistency check

20-sim/Matlab/Simulink

Pspice?

Shesim/gCSP/…

2

AADL/SysML

C/C++/POOSL/Other

Embedded Target Platform

3
Multi-view Integration Framework

- Co-simulation engine
- Consistency checking
- Translators
- Dependency coupling
- Combined code generation
- Tools
- Tool integration layer
- Tool neutral format: core model
- Inter-view Data storage

Coremodel A

Translation/coupling

Continuous time
Software related
Hardware related
Discrete event
dependencies requirements Parameters Version control test patterns

Data repository

Tools, viewpoints & views
Data exchange

Tool 1

extension/plugin
Data mapping Model update

Tool 2

extension/plugin
Data mapping Model update

Data bus

Computer & Software
I/O
Appliance

Motor

Motor

Motor

I/O

Controller

Controller

Safety

Motor

Motor

Motor

Sensor

Sensor

Sensor

Motor

Motor

Motor

Tool 1

extension/plugin
Data mapping Model update

Tool 2

extension/plugin
Data mapping Model update

Data bus

Co-simulation engine
Consistency checking
Translators
Dependency coupling
Combined code generation
Tools

Tool integration layer
Tool neutral format: core model
Inter-view Data storage

University of Twente

Broenink et al, 2007
UCB Chess Seminar
Example: Co-Simulation

Co-simulation engine

Data bus

Time synchronization

Data flow coupling

Coremodel A

Translation/coupling

Coremodel B

Continuous time

dependencies

requirements

parameters

version control

test patterns

Software related: interfaces

Hardware related: interfaces

Discrete event

Data repository

Tools, viewpoints & views

Data exchange

Tool integration layer

Tool neutral format: core model

Inter-view

Data storage

Broenink et al, 2007

University of Twente
Example: Code Generation

Tool 1
- Computer & Software
- I/O
- Appliance

Tool 2
- TOOL 1
- TOOL 2

Data mapping
Model update
Data bus

Combined code generator
Coremodel A
Coremodel B

Translation/coupling

Continuous time
Software related: interfaces
Hardware related: interfaces
Discrete event

Dependencies
Requirements
Parameters
Version control
Test patterns

Data repository

Tools, viewpoints & views
Data exchange
Tool integration layer
Tool neutral format: core model
Inter-view
Data storage

30 Broenink et al, 2007
UCB Chess Seminar
University of Twente
Case: twin-axes device JIWY

• **Characteristics**
  - 2 motor encoder pairs
  - Timed belt driven

**Toplevel model (step 1 & 2)**

```
Joystick -> controller -> io -> Plant
```

**Configuration**

```
Joystick Setpoints
  ↓
PC104 RT-linux
  ↓
Control Software
  ↓
FPGA
  ↓
I/O signals
  ↓
Amplifier
  ↓
JIWY appliance
     Motors
     Encoders
     Camera
```

PC
  ↓
Development Camera Images
JIWY: SW models
Cases: Observations

• JIWy & Tools
  – 4th Yr EE elective course on Real-Time Software
    » Preparation exercises to follow the workflow
    » “Doing right first time” on real setup succeeded

• Tools
  – Shorting of design time observed
  – 2nd Yr EE students, mechatronics project
    » ECS completely hidden (only 20-sim, 4C)
  – MSc projects
    » Robotics / Mechatronics: effective use
    » ECS: stress testing parts of the chain
Conclusions

• Prototype tool chain functions rather smoothly
• Shortening design time not (yet) significant
• Continue working on the tools
• Use larger cases in cooperation with Industry