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
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

gCSP
CT-library: CSP-based Software Framework

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

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

- Prototype

  ![Diagram](image)

  - Data-flow model
    - Process A
    - Process B
    - Data -> rendezvous

  - Distributed / heterogeneous
    - System 1
    - System 2
    - Process A
    - Process B
    - write
    - read
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
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

Diagram:

1. 20-Sim
2. gCSP
3. Other
4. Target Connector
5. Compiler assistant
6. Deployment manager
7. Real-time logger

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

Start/stop parameters
Send to target(s)
Modify parameters
Log data
Code generation
Code processing
(cross) compilation
ForSee: Connect, Compile, Configure, Control

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

- Code generation
- Connecting model inputs to I/O hardware
- ADSP board hardware
- Compiling the generated code
- and outputs to I/O hardware
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

3D model follows real setup

PositionPID \( (m) \)
PositionLQG \( (m) \)
Further tool integration

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

- **Real cooperative design**
  - One-model – One-tool approach too optimistic yet
    » Multiple view modeling (& tools)
  - Tightly coupling needed also on model level
    » Check cross-domain aspect relations
  - Co-simulation on execution level
Multi-view Integration Framework

Tool 1

- **I/O**
  - Computer
  - Software

- **Appliance**

  - Motor
  - Motor

  - Tool 1

  - **extension/plugin**
  - **Data mapping**
  - **Model update**

Tool 2

- **Tool 2**

  - **extension/plugin**
  - **Data mapping**
  - **Model update**

**Data bus**

- **Data exchange**

**Tools**

- **Co-simulation engine**
- **Consistency checking**
  
**Translations**

- **Dependency coupling**
- **Combined code generation**

**Coremodel A**

- Continuous time
- Software related
- Hardware related
- Discrete event
- Dependencies
- Requirements
- Parameters
- Version control
- Test patterns

**Coremodel B**

- Translation/coupling

**Tools, viewpoints & views**

- Tool integration layer

**Tool neutral format: core model**

- Inter-view
- Data storage
Example: Co-Simulation

Coremodel A

Coremodel B

Continuous time

Software related: interfaces

Hardware related: interfaces

Discrete event

Dependencies

Requirements

Parameters

Version control

Test patterns

Data repository

Translation/coupling

Time synchronization

Data flow coupling

Co-simulation engine

Data mapping

Model update

Data exchange

Tools, viewpoints & views

Tool integration layer

Tool neutral format: core model

Inter-view

Data storage

Tools, viewpoints & views

Example: Co-Simulation

Tool 1

Tool 2

Extension/plugin

I/O

Computer & Software

Appliance

Data mapping

Model update

Data bus

Co-simulation engine

Time synchronization

Data flow coupling

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

Hardware

Motor

Motor

Motor

Motor

Controller

Controller

Controller

Sensor

Sensor

Sensor

MOTOR1

MOTOR2

MOTOR3

Safety

Controller3

Controller2

Controller1

SENSOR1

SENSOR2

SENSOR3
Case: twin-axes device JIWF

• Characteristics
  – 2 motor encoder pairs
  – Timed belt driven
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