Design Tools for
Distributed Embedded Mechatronic Control Systems

Jan F. Broenink
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
Projects Involved

• **TES.5224 Design Tools**
  – Methods & software tools using our CSP-library
  – Dusko Jovanovic, Gerald Hilderink, Geert Liet, Marcel Groothuis, Job v Amerongen

• **TES.5410, Fieldbuses**
  – Hard real-time control over fieldbus-interconnected processors
  – Bojan Orlic, Matthijs ten Berge, Job v Amerongen

• **TES.7020 (with TU/e), ViewCorrect**
  – Methods & tools for predictable design of distributed hard real-time embedded control systems
  – Marcel Groothuis, Jinfeng Huang
  – Jeroen Voeten, Henk Corporaal, Job v Amerongen
Overview

• Introduction
  – Embedded Control Systems

• Design Approach
  – Mechatronic Systems Embedded Control Software
  – Design Tools

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

• Project results
  – CTC++ library, gCSP tool
  – Exception handling, I/O patterns
  – Distributed Simulation Framework
  – Multi-view integration

• Outlook
  – Further research topics
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
Design 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
Focus on Embedded Control Software

- Model based
- Refinement

20-Sim / Matlab

gCSP / FDR / POOSL

CTC++ / CCE

Physical System Modeling

Control Law Design

Computer System Implementation

Realization

Verification by Simulation

Verification by Simulation / Model Checking

Validation and Testing

• Model based
• Refinement
Design 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
    - Realization
  - Verification by Simulation & Formal Checking
**ECS Implementation**

- **Stepwise refinement**
  - Gradually enhance laws to code

- **Working order**
  - Integrate control laws
    - Loop control laws + Sequencers + Supervisors
    - Reaction to external commands
    - Implementation assumed ideal
  - Capture non-ideal components
    - Non-idealness added:
    - Relevant dynamic behavior
    - Signal processing: Estimators
  - Safety, error & maintenance facilities
    - External events on safety (emergency stops)
    - Central or component wise
  - Non-idealness Computer hardware
    - HW + SW architecture, Timing aspects
    - Optimization, scheduling

**Mix of**
- Process Structures
- Control Algorithms
Realization

- **Stepwise refinement**
  - **Hardware-In-the-Loop Simulation**

- **Working order**
  - Part wise towards realization
    - Others still simulated

- **Concurrent engineering**
  - Plant simulated
    - SW in time, Plant late
  - Code simulated
    - SW in time, ASIC late

- **Test setup**
  - Plant simulated
  - For training purposes
Used Model Formalisms

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

• Essential solution
  – Object Orientation
  – Component based

• 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 – FDR

- **Encapsulation**
  - Implementation from communication
  - Scheduling 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: \( \text{in?x} \) and \( \text{out!x} \)
Overview

• Introduction
  – Embedded Control Systems

• Design Approach
  – Mechatronic Systems Embedded Control Software
  – Design Tools

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

• Project results
  – CTC++ library, gCSP tool
  – Exception handling, I/O patterns
  – Distributed Simulation Framework
  – Multi-view integration

• Outlook
  – Further research topics
(5224): CSP-based Software Framework: CT-lib

- **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

Data-flow model:

Process A \(\rightarrow\) Process B

System 1 \(\leftrightarrow\) System 2

distributed / heterogeneous
5224: Graphical CSP – gCSP tool

- **Editing CSP diagrams**
  - Modeling
  - Designing
  - Documenting

- **Code generation:**
  - CSPm for formal checks with FDR/ProBE
  - CT executable code (CTC++)

- **Developed in Java**
Editing CSP diagrams

- Full CSP view
- Communication view
- Compositional view
CSPm code generation

datatype Double = b_val

channel DataChannel1 : Double
cannel DataChannel2 : Double

Par1 = PROCESS1 || { || DataChannel1, DataChannel2 || } || (PROCESS2 ||| PROCESS3)

PROCESS1 = Seq_P1 ; Seq_P2
Seq_P1 = SKIP
Seq_P2 = SKIP

PROCESS2 = Alt_P1 || Alt_P2
Alt_P1 = SKIP
Alt_P2 = SKIP

PROCESS3 = DataChannel1?a -> DataChannel2!b_val -> PROCESS3
CTC++ code generation

---

//-- Channel Allocations
Channel<double> *DataChannel1 = new Channel<double>();
Channel<double> *DataChannel2 = new Channel<double>();

//-- Process Allocations
PROCESS1 *PROCESS1_1 = new PROCESS1(DataChannel1, DataChannel2);
PROCESS2 *PROCESS2_1 = new PROCESS2();
PROCESS3 *PROCESS3_1 = new PROCESS3(DataChannel1, DataChannel2);

//-- Network Builder
Parallel *Par1 = new Parallel(
    PROCESS1_1,
    PROCESS2_1,
    PROCESS3_1,
    NULL);
Par1->run();

---

//-- Channel Allocations
Channel<double> *DataChannel1 = new Channel<double>();
Channel<double> *DataChannel2 = new Channel<double>();

//-- Process Allocations
PROCESS1 *PROCESS1_1 = new PROCESS1(DataChannel1, DataChannel2);
PROCESS2 *PROCESS2_1 = new PROCESS2();
PROCESS3 *PROCESS3_1 = new PROCESS3(DataChannel1, DataChannel2);

//-- Network Builder
Parallel *Par1 = new Parallel(
    PROCESS1_1,
    PROCESS2_1,
    PROCESS3_1,
    NULL);
Par1->run();

---

//-- Process Allocations
Alt_P1_1 = new Alt_P1();
Alt_P2_1 = new Alt_P2();

//-- Network Builder
Alternative *Alt1 = new Alternative(
    Alt_P1_1,
    Alt_P2_1,
    NULL);

//-- Run method
void PROCESS3::run() {
    while(true) {
        DataChannel9->read(&a);
        b = f(a);
        DataChannel14->write(&b);
    }
}

---

//-- Process Allocations
READER1_1 = new Reader<double>(DataChannel9, &a);
WRITER1_1 = new Writer<double>(DataChannel14, &b);
CodeBlock_1 = new CodeBlock(a, b);

//-- Network Builder
Sequential *Seq7 = new Sequential(
    READER1_1,
    CodeBlock_1,
    WRITER1_1,
    NULL);
Seq7->run();

//-- Run method
void PROCESS3::run() {
    while(true) {
        Seq7->run();
    }
}

---

//-- Process Allocations
Seq_P1_1 = new Seq_P1(DataChannel26);
Seq_P2_1 = new Seq_P2(DataChannel127);

//-- Process Allocations
READER1_1 = new Reader<double>(DataChannel9, &a);
WRITER1_1 = new Writer<double>(DataChannel14, &b);
CodeBlock_1 = new CodeBlock(a, b);

//-- Network Builder
Sequential *Seq6 = new Sequential(
    READER1_1,
    CodeBlock_1,
    WRITER1_1,
    NULL);
Seq6->run();

//-- Run method
void PROCESS3::run() {
    while(true) {
        DataChannel9->read(&a);
        b = f(a);
        DataChannel14->write(&b);
    }
}

---

//-- Process Allocations
Alt_P1_1 = new Alt_P1();
Alt_P2_1 = new Alt_P2();

//-- Network Builder
Alternative *Alt1 = new Alternative(
    Alt_P1_1,
    Alt_P2_1,
    NULL);

//-- Run method
void PROCESS2::run() {
    while(true) {
        Seq7->run();
    }
}
Exception handling

gCSP example!
Exception Handling – Examples

Safety: Motor overload

Safety & Fault Tolerance: Soft End Stops
I/O composition

• Cycles
  – Easy deadlock

• Preferred order
  – Read – Compute – Write
  – Easy deadlocks (example next slides)

• Solution
  – I/O SEQ
    » Read – Compute – Write still SEQ
    » I/O in itself parallel
  – I/O PAR
    » Read – Compute – write PAR
Deadlock checking and code generation

```c
//-- Network Builder
Seq0 = new Sequential(
    ehReader_1,
    jhReader_1,
    XXPositionControllerHorizontal_1,
    dacWriter_1,
    corrWriter_1,
    NULL);

//-- Network Builder
Parallel *Par0 = new Parallel(
    Vertical_1,
    Horizontal_1,
    SanityCheck_1,
    NULL);

Par0->run();
```
Deadlock checking and code generation

I/O-SEQ

PositionControllerHorizontal

PositionControllerVertical

EncoderH

EncoderV

Jh

Jv

I/O-SEQ

Repeat_H

ehReader

hor_reader

position

hor_san

encoderH

hor_output

dacH

DacH

I/O-SEQ

Repeat_V

evReader

vReader

position

v_out

vOut

DacV

I/O-SEQ

Check:Hor

Check:Ver

dacH

DacH

I/O-SEQ

[true]
dacReader

corrReader

corr

corrWriter

dacWriter

output

output

[true]

[true]

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DAC

DacH

DacV

I/O-SEQ
5410: Distributed Simulation Framework

- **CSP approach**
  - Remote Channels couple to Fieldbus
  - Time increment via Timer Channel
  - Packet Simulator based on TrueTime

- **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

[Diagram showing network components and data flow]

[Graph showing motor position over time with different network configurations.
Legend: setpoint, 20sim [rad], no network, network 1MBps, network 100KBps, network 10KBps, 100KBps, p20, 100KBps, p10]
7020: ViewCorrect

- **Multiple views**
  - Each discipline its own

- **2 Core models**
  - Continuous, Discrete
  - Co-Simulation
  - Interview-dependent Coupling
    » parameters

- **Code generation**
  - Correctness preserving -> PC
Multi-view Integration Layer, 1st idea

Data mapping \rightarrow Model update

Co-simulation engine
Consistency checking

Translators
Dependency coupling

Test tools
Combined code generation

Tools

Tool neutral format:
core model

Inter-view
Data storage

Data bus

Coremodel A \leftrightarrow Coremodel B
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 integration layer
Overview = ‘Summary’

• **Introduction**
  – Embedded Control Systems

• **Design Approach**
  – Mechatronic Systems Embedded Control Software
  – Design Tools

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

• **Project results**
  – 5224: CTC++ library, gCSP tool
  – 5224: Exception handling, I/O patterns
  – 5410: Distributed Simulation Framework
  – 7020: Multi-view integration

• **Outlook**
  – Further research topics
Outlook

• **Modeling**
  – Separation ‘happy flow’ from exceptional situations
  – Structuring support: Controller Agents
  – Multiple-view integration (ViewCorrect)

• **Analysis / Transformation / Tools**
  – Symbiosis Formal methods – Simulation
  – Stepwise Refinement
  – Methods -> Prototype Tools

• **Hardware**
  – Exploit flexibility / programmable hardware

• **Usage**
  – Use methods and tools in courses
  – Cases with Industry  
    » Check our Design Approach