

Towards a Specification Technique for Component-Based Measurement and Control Software for Embedded Systems

Walter Maydl

Bernhard Sick

Werner Grass

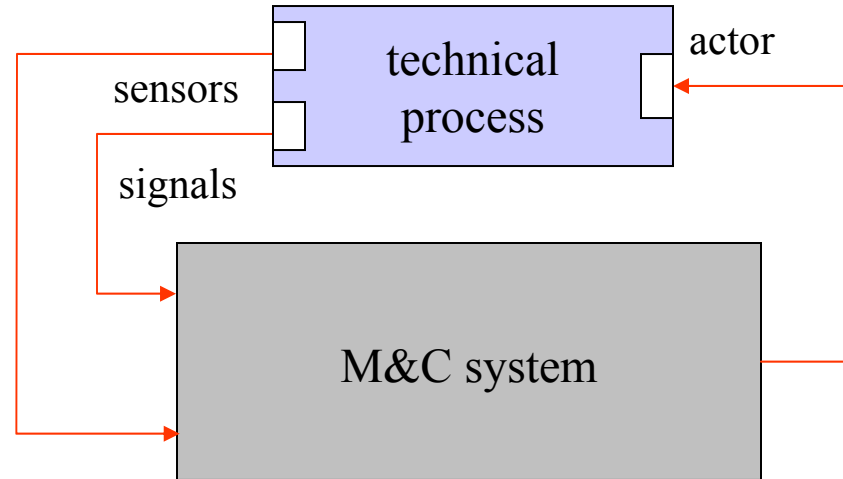
University of Passau

Institute for Computer Architectures

Innstr. 33

94032 Passau

Motivation (1)



Application area: Measurement, signal processing, and control (M&C)

- supervision of thermal conductivity of gas concrete bricks
- thickness control in the production of flat glass for notebook displays
- thickness control of thin metal foils and synthetic films
- control of a motor test stand with sensors for torque, temperature, pressure, etc.

Software development = combination of various parameterized algorithms:

- resampling methods
- digital filters
- Fourier transforms
- PID controllers

Motivation (2)

Szyperski's definition of a component:

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A Software component can be deployed independently and is subject to composition by third parties.

Significant advantages:

- improved quality of the application due to software reuse at the component level
- support of rapid prototyping and rapid development (shorter time-to-market)
- easy adaptation to changing requirements and improved configurability
- enhanced maintainability
- multiplication of investment and innovation

Motivation (3)

Differences to other approaches:

- Detailed data model
 - definition of signal including time domain and value range
 - exact definition of the interface types of signal processing algorithms (constraints between their incoming and outgoing interfaces)
 - ➡ avoidance of compatibility errors
 - more precise calculations
- Component model based on the separation of basic signal processing and adaptation algorithms, explicit control structures
 - hierarchy of dataflow paradigms with increasing modeling power/ decreasing analyzability
 - application engineer has control over complexity/analyzability
 - ➡ avoidance of execution errors

Motivation (4)

Focus of this presentation:

The derivation of properties of a component-system from given properties of components and rules for their interaction

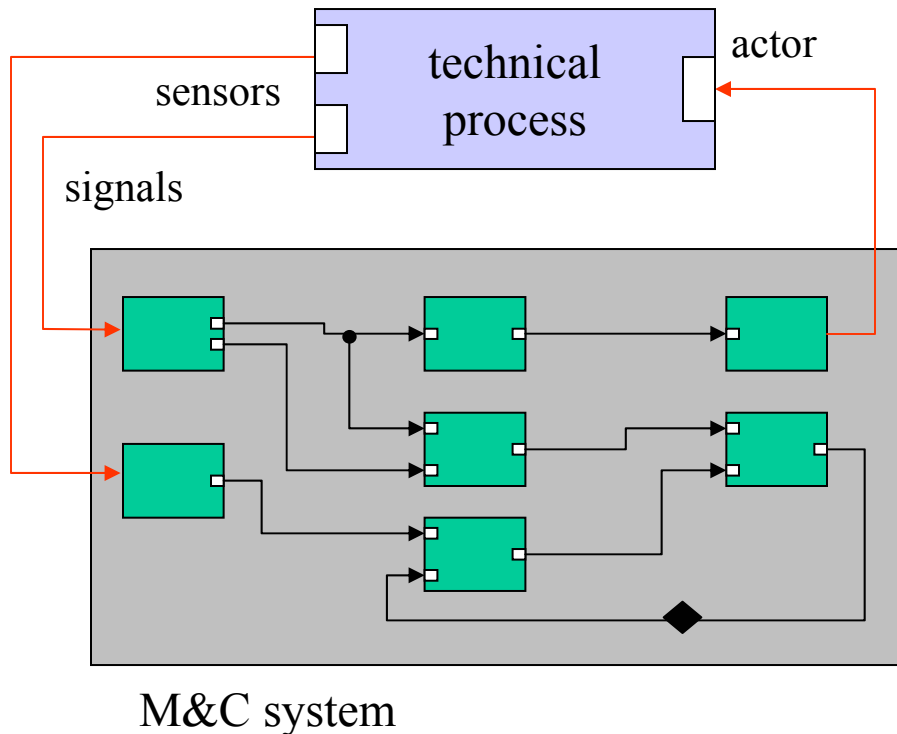
Interesting questions:

- Is the overall component system deadlock free?
- What is the overall amount of memory needed by this system?
- Does a cyclic schedule of the component instances exist and can it be calculated statically?
- Has the overall component system a deterministic behavior?

Overview

- A novel concept for M&C systems
 - Structure of a complex application
 - Data model
- Dataflow model
 - Dataflow paradigms
 - Component models
- Conclusion & Outlook

A Novel Concept for M&C Systems (1)



Signal graph (dataflow graph)

= **component system**

- accepted formalism for engineers
- constructed by visual programming

Nodes = **component instances:**

- basic signal processing algorithms
- adaptation algorithms
- access points
- shared libraries

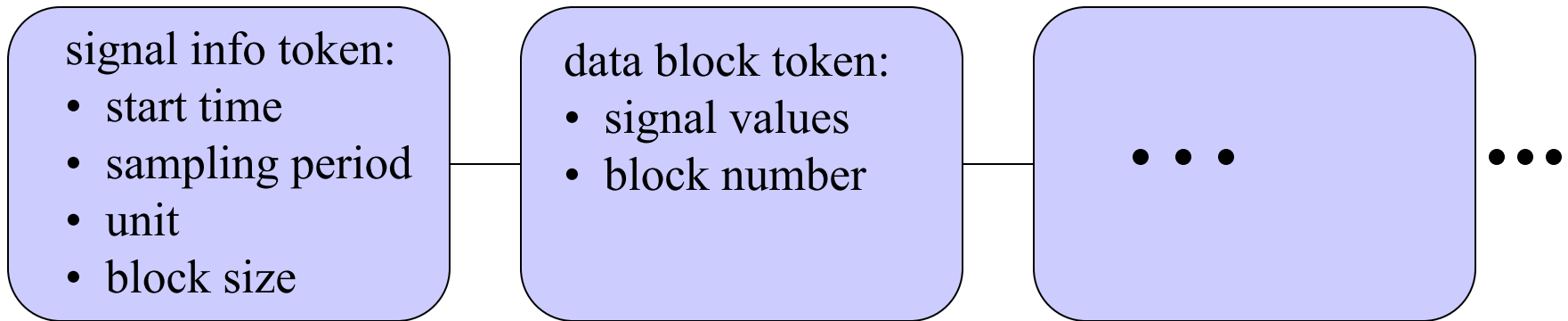
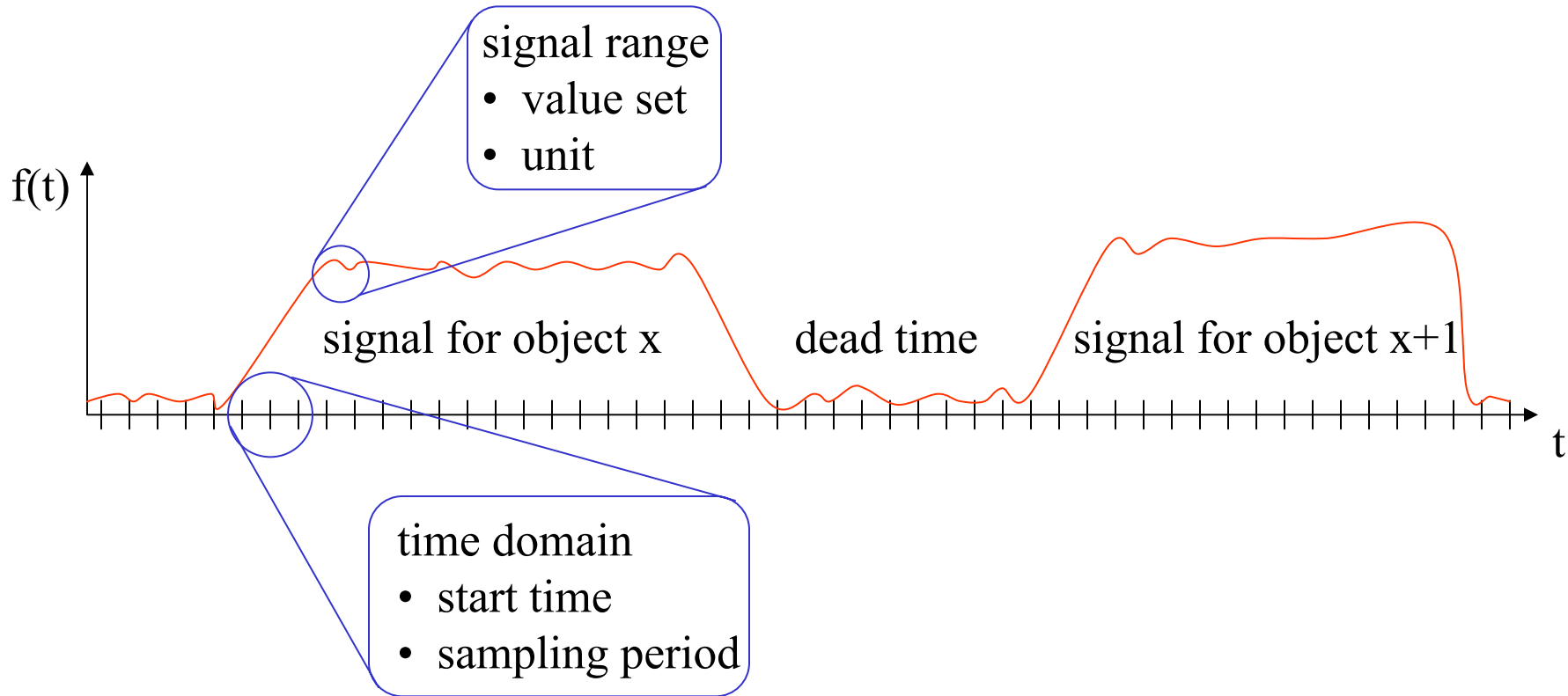
Directed edges = **fifo channels**

- asynchronous communication
- TCP/IP or Shared Memory

Scheduler = **part of component framework**

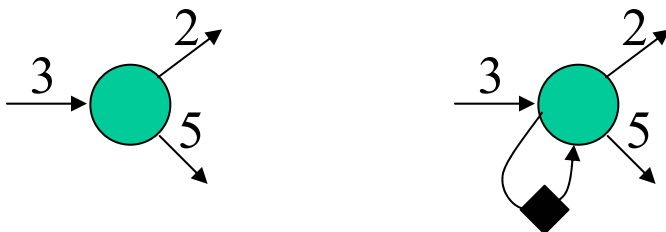
- regulation of interoperation aspects
- up to several hundred component instances
- partial order of execution given by edges
- distributed execution feasible
- in general data-dependent execution (dynamic scheduling)

A Novel Concept for M&C Systems (2)



Dataflow Model (1)

SDF (Synchronous Dataflow)



- actors with fixed token consumption/production
- edges = fifo channels
- memory of actors = self-loop with initialization token
- no modeling of time, but includes causality
- firing of enabled actor depends on scheduler
- no data-dependent execution (static scheduling feasible)

Computational Power \neq Turing machine



Decidable/Computable:

- presence of deadlocks
- cyclic schedule
- amount of memory needed

Dataflow Model (2)

BDF (Boolean Controlled Dataflow)

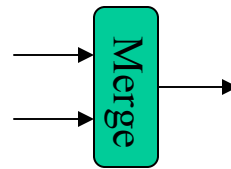


- extension of SDF
- switch/select have conditional input/output edges
- data-dependent execution (dynamic scheduling)

Computational Power = Turing machine
deterministic because of validity of
Kahn-condition

Dataflow Model (3)

DDF (Dynamic Dataflow)



- extension of BDF
- nondeterminism: edges do not reflect the execution order of the dataflow graph
- data-dependent execution (dynamic scheduling)

Computational Power = Turing machine
nondeterministic

Dataflow Model (4)

SDF components

- correspond directly to SDF actors
- encapsulate any basic signal processing algorithm
- contain any adaptation algorithm which can be modeled by SDF
- storage of component status in self-loops

Advantages:

- pure SDF component systems
 - ➔ decidable/computable:
 - presence of deadlocks
 - cyclic schedule
 - amount of memory needed
- separation of code and data
(execution of subsequent task instances on different processors)

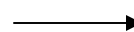
Dataflow Model (5)

BDF components

- Switch
- Select
- For-next-loop
- Collector
- Divider
- Trashcan
- Filler
- Cutter

} control
structures

} adaptation
algorithms



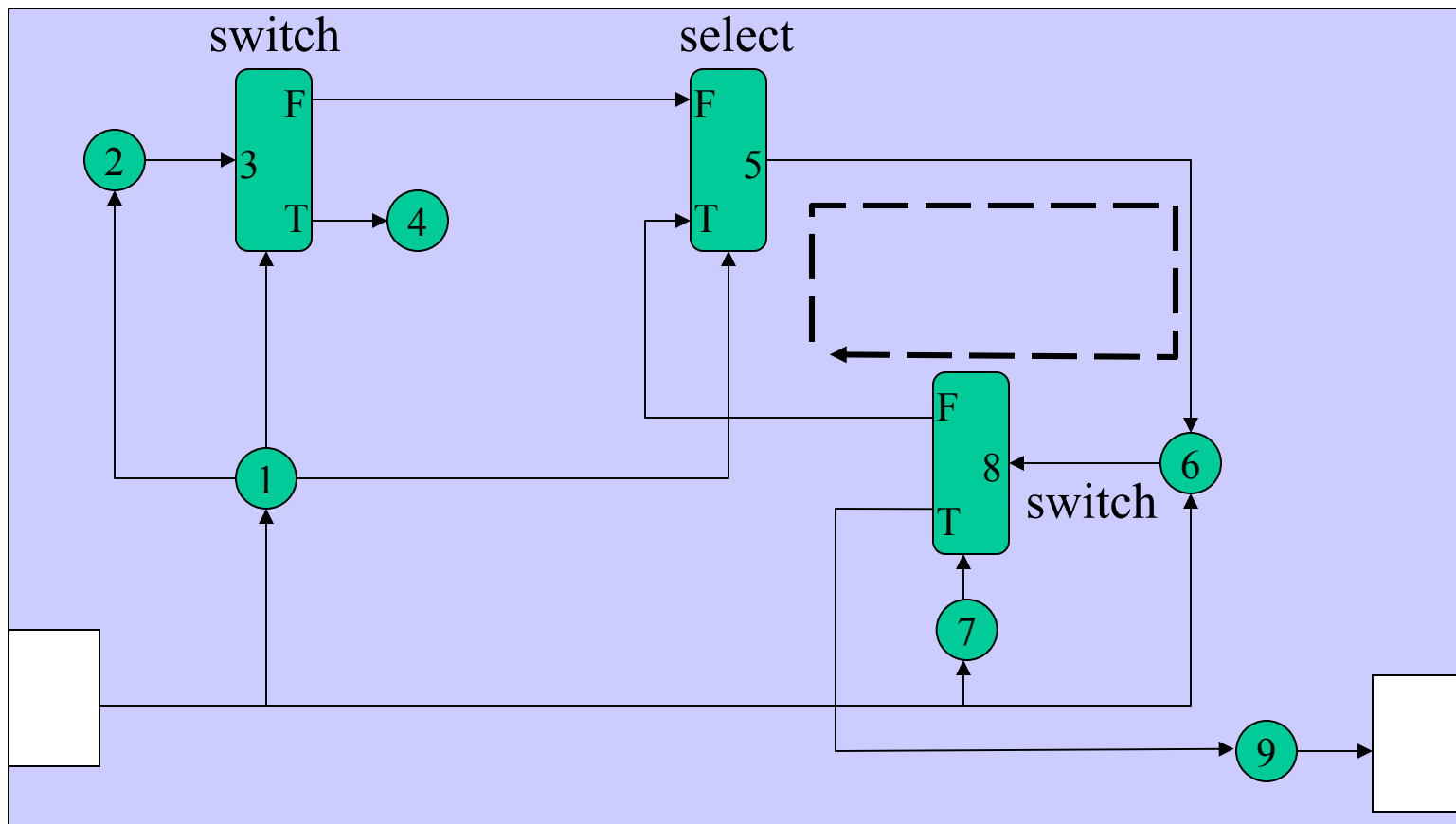
Design Patterns:

If-then-else
Case
While

Decision about usage of more powerful BDF components with loss of analyzability is up to the application engineer

Dataflow Model (6)

Name: Collector



Task: Collects all the datablock tokens of a signal and fuses them to one data block token

Advantage: Successor component working on a whole signal can be modelled as SDF component

Dataflow Model (7)

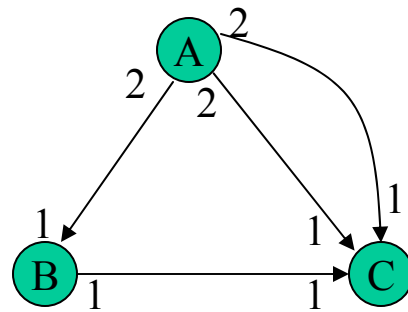
DDF components

- Merge (control structure)
- Decoupler (adaptation component)

Decision about usage of DDF components with loss of analyzability is up to the application engineer

Dataflow Model (8)

Analysis of SDF graphs



Possible Schedule: ABCBC

Partial Analysis of BDF/DDF graphs

1. Static schedules for SDF clusters
2. Dynamic scheduling of the rest

Conclusion and Outlook

Specification technique for M&C systems based on

- interface types (avoidance of compatibility errors)
- separation of basic signal processing and adaptation algorithms
 1. encapsulation into components
 2. modelling of these components with the well-founded dataflow paradigms SDF, BDF, and DDF
 3. transfer of general techniques for these paradigms to the component models and the resulting component system
 4. important questions decidable/computable (for graphs containing SDF components only):
 - presence of deadlocks
 - calculation of cyclic schedules
 - calculation of the amount of memory needed

Correctness by Construction, not by Testing!