

# Separation of Concerns in the Formal Design of Real-Time Shared Data-Space Systems

M. Mousavi, M. Reniers, T. Basten, M. Chaudron

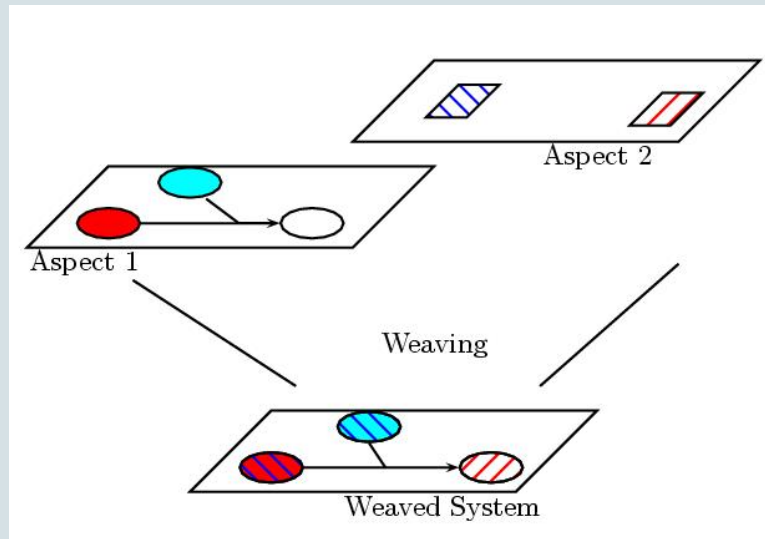
Eindhoven University of Technology

## Overview

1. Design Philosophy
2. Case-Study: Steam-Boiler
3. Functionality and Timing
4. Coordination
5. Conclusion and Future Works

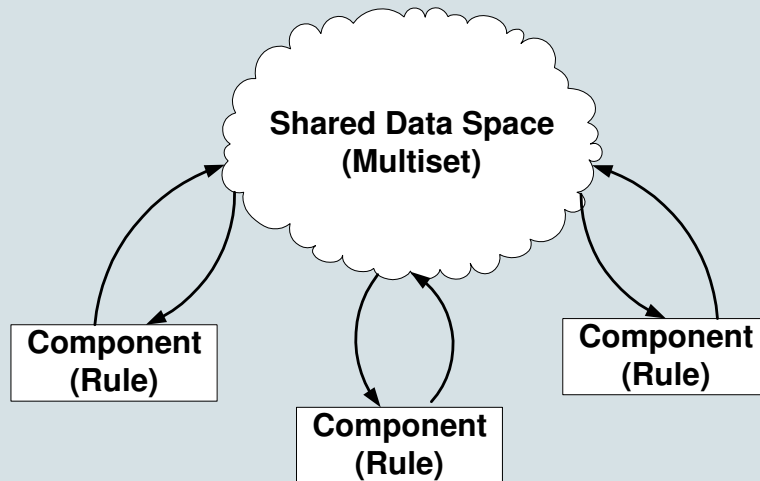
# I. Design Philosophy

## Separation of Concerns in the Design of Embedded Systems



## Separation of Concerns in the Design of Embedded Systems

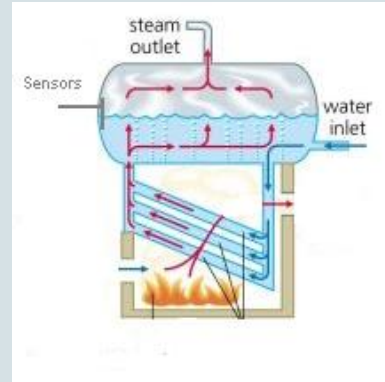
- Functionality, Timing and Behavior (Coordination): Basic Aspects of Real-Time Design
- Shared Data Space framework as a means, providing:
  - Spatial and temporal decoupling of components
  - Ideal model for pure functionality



## 2. Case-Study: Steam-Boiler Control

Timed Functionality:

- Initializing the system
- Pump controller
- Communication channel
- Sensors, sampling dynamics
- Fault detection and phase change
- Estimating dynamics of the system



## 2. Case-Study: Steam-Boiler Control

Coordination:

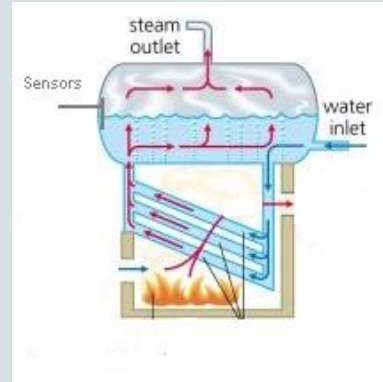
- Composition of environment actions:

1. Clock

2. Sensors system

3. Communication channel

- Design of control strategy



### 3. Functionality and Timing

(Simplified) Syntax

---

$$RuleName = ReadExpr? : TakeExpr \mapsto PutExpr \Leftarrow Cond$$
$$T_{RuleName} = Interval$$

---

### 3. Functionality and Timing

A Few examples from the case study:

$$\begin{aligned} PumpSensorError &= PumpState?, PumpThroughput?, \\ &PumpSensorStatus \mapsto defect \\ &\Leftarrow (PumpState = closed \\ &\wedge PumpThroughput \neq 0) \end{aligned}$$

...

$$T_{PumpSensorError} = [minErrorDet, maxErrorDet]$$

$$\begin{aligned} PourWater &= WaterPoured?, SystemTime? : \\ &EstWaterLevel, LastPoured \mapsto \\ &EstWaterLevel + WaterPoured, SystemTime \end{aligned}$$

### 3. Functionality and Timing

#### Semantics

- Multiset ( $M : M$ ) : Shared Data Space
- Substitution ( $\alpha = [M/N]$ ) : Single Computation
- Computation ( $\sigma = [M_0/N_0, M_1/N_1, \dots]$ ) : Concurrent computations
- Independence ( $M \models \sigma_1 \bowtie \sigma_2$ ):  
 $\sigma_1$  and  $\sigma_2$  find in  $M$  enough:
  - shared data copies to read
  - distinct data copies to take

### 3. Functionality and Timing

#### Single Rule Semantics

State:  $[Rule, Data Multiset, Task Multiset]$

Task:  $Substitution@Active\ Time : Interval$

$$\text{(Scheduling)} \frac{M, \bar{v} \propto r}{\langle r, M, \boxplus \rangle \xrightarrow{0}_1 \langle r, M, [\alpha@0 : r.I] \rangle}$$

where for a rule  $r = m_0 \mapsto m_1 \leftarrow c$ ,  $\alpha = \bar{v}[m_1]/\bar{v}[m_0]$

$$\text{(Time Pass)} \frac{t + t' \in I \vee t + t' \leq lb(I) \quad t' > 0}{\langle r, M, [\alpha@t : I] \rangle \xrightarrow{t'}_1 \langle r, M, [\alpha@t + t' : I] \rangle}$$

$$\text{(Commitment)} \frac{t \in I}{\langle r, M, [\alpha@t : I] \rangle \xrightarrow{[\alpha]}_1 \langle r, M([\alpha]), \boxplus \rangle}$$

### 3. Functionality and Timing

#### Part of Abstract Semantics of Programs

$$\text{(Time Pass o)} \frac{r \in R \quad \langle r, M, [\alpha@t : I] \rangle \xrightarrow{t'}_1 \langle r, M, [\alpha@t + t' : I] \rangle \quad t' > 0}{\langle R, M, [\alpha@t : I] + T \rangle \xrightarrow{t'} \langle R, M, [\alpha@t + t' : I] + T \rangle}$$

$$\text{(Time Pass i)} \frac{\langle R, M, T_0 \rangle \xrightarrow{t} \langle r, M, T'_0 \rangle \quad \langle R, M, T_1 \rangle \xrightarrow{t} \langle R, M, T'_1 \rangle \quad t > 0}{\langle R, M, T_0 + T_1 \rangle \xrightarrow{t} \langle R, M, T'_0 + T'_1 \rangle}$$

$$\text{(Scheduling)} \frac{r \in R \quad \langle r, M, \sqcap \rangle \xrightarrow{0}_1 \langle r, M, [\alpha@0 : I] \rangle \quad M \models [\alpha@0 : I] \bowtie T}{\langle R, M, T \rangle \xrightarrow{0} \langle R, M, [\alpha@0 : I] + T \rangle}$$

## 4. Coordination

### (Simplified) Syntax

*Schedule* =

*RuleName*

Rule: Building Block of Schedules

*Schedule* { ; , || , ||| } *Schedule*

Sequential Composition

Abstract and Strict Parallel Compositions

*RuleName*  $\rightarrow$  *Schedule* [*Schedule*] Conditional

$\mu$  *Recursion Var.* *Schedule*

Recursion

## 4. Coordination

### A Few Examples from the Case-Study

$$\textit{ChangePhase} = \textit{PumpSensorError} \rightarrow (\textit{RescuePhase})[\textit{NormalPhase}]$$
$$\begin{aligned} \textit{SteamBoiler} = & \textit{Initialize} ; \\ & (\textit{Environment} \parallel \\ & (\mu X. (\textit{DangerCheck} \parallel \\ & (\textit{ChangePhase} ; \textit{Control})) ; X))) \end{aligned}$$

## 4. Coordination

### Semantics

The first step: Extending basic semantics of timed-functionality ( $\rightarrow_1$ ) to coordination ( $\rightarrow$ ), for example:

$$\text{(Coordination Scheduling)} \frac{\langle r, M, \square \rangle \xrightarrow{0} \langle r, M, [\alpha@0 : I] \rangle}{\langle r, M, T \rangle \xrightarrow{0} \langle r[\alpha@0 : I], M, [\alpha@0 : I] \rangle}$$

## 4. Coordination

### Semantics

Abstract Parallel Composition:

$$\text{(Po, Pi)} \frac{\langle s_0, M, T_0 \rangle \xrightarrow{\chi} \langle s'_0, M', T'_0 \rangle \quad M' \models T'_0 \bowtie T_1}{\begin{array}{l} \langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{\chi} \langle s'_0 \parallel s_1, M', T'_0 + T_1 \rangle \\ \langle s_1 \parallel s_0, M, T_1 + T_0 \rangle \xrightarrow{\chi} \langle s_1 \parallel s'_0, M', T_1 + T'_0 \rangle \end{array}}$$

$$\text{(P2)} \frac{\langle s_0, M, T_0 \rangle \xrightarrow{t} \langle s'_0, M, T'_0 \rangle \quad \langle s_1, M, T_1 \rangle \xrightarrow{t} \langle s'_1, M, T'_1 \rangle \quad t > 0}{\langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{t} \langle s'_0 \parallel s'_1, M, T'_0 + T'_1 \rangle}$$

$$\text{(P3)} \frac{\langle s_0, M, T_0 \rangle \xrightarrow{\sigma_0} \langle s'_0, M_0, T'_0 \rangle \quad \langle s_1, M, T_1 \rangle \xrightarrow{\sigma_1} \langle s'_1, M_1, T'_1 \rangle}{\langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{\sigma_0 + \sigma_1} \langle s'_0 \parallel s'_1, M(\sigma_0 + \sigma_1), T'_0 + T'_1 \rangle}$$

## 4. Coordination

### Semantics

Strict Parallel Composition:

$$(\text{SPo}) \frac{\langle s_0, M, T_0 \rangle \xrightarrow{0} \langle s'_0, M, T'_0 \rangle \quad M \models T'_0 \bowtie T_1}{\langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{0} \langle s'_0 \parallel s_1, M, T'_0 + T_1 \rangle}$$

$$(\text{SPi}) \frac{\langle s_0, M, T_0 \rangle \xrightarrow{\sigma} \langle s'_0, M', T'_0 \rangle}{\langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{\sigma} \langle s'_0 \parallel s_1, M', T'_0 + T_1 \rangle}$$

$$(\text{SP2}) \frac{\langle s_0, M, T_0 \rangle \xrightarrow{t} \langle s'_0, M, T'_0 \rangle \text{ disabled}(\langle s_1, M, T_1 \rangle, T_0)}{\langle s_0 \parallel s_1, M, T_0 + T_1 \rangle \xrightarrow{t} \langle s'_0 \parallel s_1, M, T'_0 + T_1 \rangle}$$

## 5. Conclusion and Future Works

### General achievements:

1. Proposing a model for separation of concerns for real-time distributed design
2. Defining a formal semantics for it
3. Specifying a number of academic case-studies in this model

## 5. Conclusion and Future Works

### Specific (technical) contributions:

1. Defining and formalizing *read* parallelism in GAMMA semantics using computations and an independence relation
2. Extension of GAMMA and schedule semantics to timed semantics
3. Reflection of three phases of basic transactions (scheduling, computing, and commitment) in the formal semantics of Timed-GAMMA and schedules

## 5. Conclusion and Future Works

### Ongoing research:

1. Mechanizing reasoning about specification
2. Implementing an aspect-oriented middle-ware for translating specification into implementation
3. Extending the approach to other aspects of real-time distributed system design:  
PARS: A Process Algebra with Resources and Schedulers

For more information, see:

<http://www.win.tue.nl/~mousavi/sacc/>