Approaches to Design of Real-time Systems

02224 Real-time Systems

20 April 2022

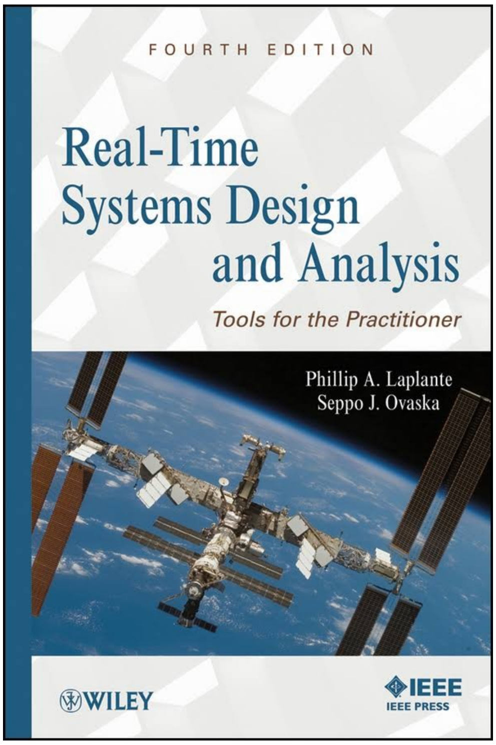
Hans Henrik Løvengreen

DTU Compute

Contents

- What is design?
- Which design goals?
- Some approaches
- Discussion of assignment

Partly based on:



What is design?

Given a system requirement specification:

- Delineation of system (interface, context diagram)
- Desired system behaviour (functional requirements)
- Constraints (resources)

Design is the process of determining an internal structure:

- Major components
- Their means of interaction

Such that

- The requirements are satisfied
- Desired system qualities are optimized

Design goals

System qualities [Laplante,Ovaska]

- Reliability
- Correctness
- Performance
- Usabililty
- Interoperability
- Maintainability
- Portability
- Verifiability

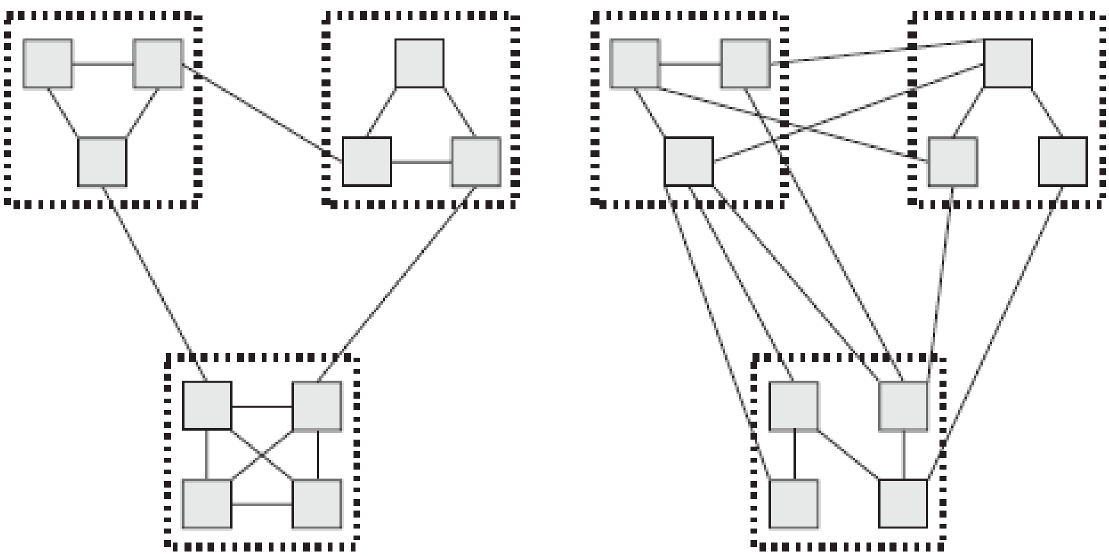
Software Engineering Principles

[Laplante,Ovaska]

- Rigor and formality
- Separation of concerns
- Modularity (hierarchies)
- Anticipation of change
- Generality
- Incrementality
- Traceability

DTU

Coupling and cohesion [Parnas]



(a) (b)

DTU

The V-model

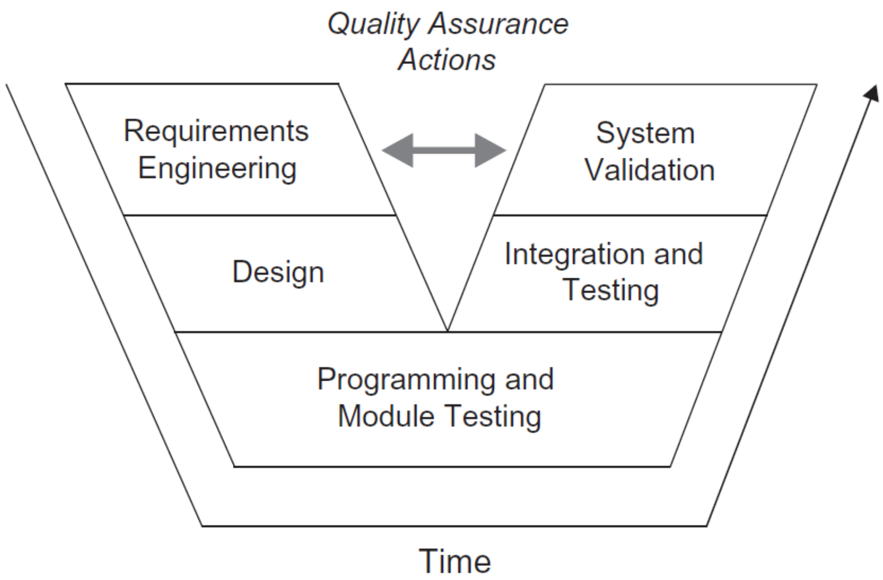


Figure 6.14. V-model with quality assurance activities.

Identifying components

Main Approaches

- Function oriented
  - Focus on data transformation/flow
  - Verbs become activities (procedures/threads)
  - Necessary state is added
- State orientered
  - Focus on data representation
  - Nouns become objects/classes
  - Necessary operations are added
  - UML, SysML

Example: Structured Analysis/Design

Data Flow Diagram:

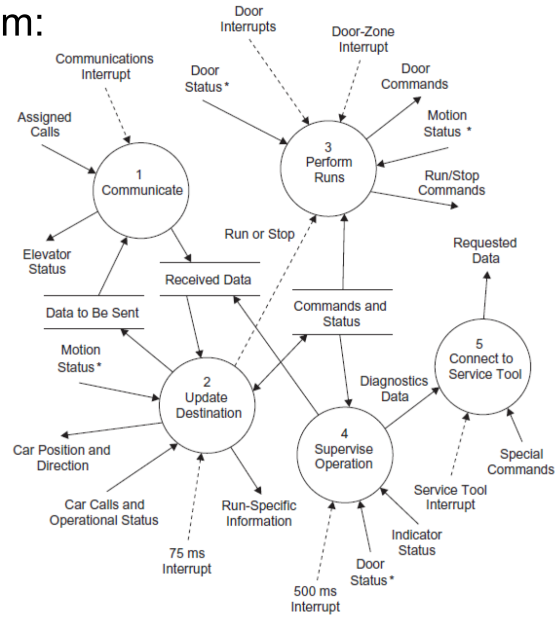


Figure 6.8. Level 0 DFD for the elevator control system. * This incoming data flow is connected to two processes.

DTU

Example: Design of (small) reactive systems

- Simple design method developed at DTU ~ 2000
- Function oriented
- Uses interaction patterns for systems structure
- Design may be subject to real-time analysis
- Limited scope

DTU

Example: Taximeter

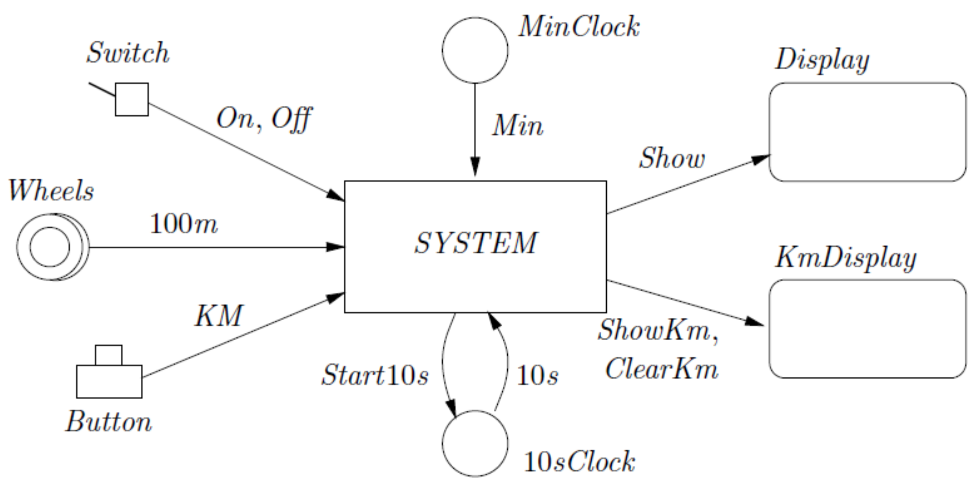
Given a requirements specification of a taximeter:

The taximeter can be switched on and off. When being switched on, the amount due shown on a display must be reset. While the taximeter is switched on, the amount should be incremented each minute as well as for each 100 meter signal received from the wheels. When switched off, the final amount remains on the display. When a "KM"-button is pressed, the total distance covered with the taximeter switched on must be shown for 10 seconds on a separate display.

DTU

Example: Taximeter

System diagram (context diagram):



DTU

Example: Taximeter

Event list:

Input event	Data type	Legend
On	*VOID*	Taximeter is switched on.
Off	*VOID*	Taximeter is switched off.
100m	*VOID*	Signal for each 100 meter.
KM	*VOID*	KM-button pressed.
Min	*VOID*	Regular signal from minute-clock.
10s	*VOID*	Signal 10 sec. after $Start10sec$.

Output event	Data type	Legend
Show	*NUMBER*	Show number on amount display.
ShowKm	*NUMBER*	Show number on distance display.
ClearKm	*VOID*	Clears distance display.
Start10s	*VOID*	Request for 10 sec. signal.

Example: Taximeter

Event patterns:

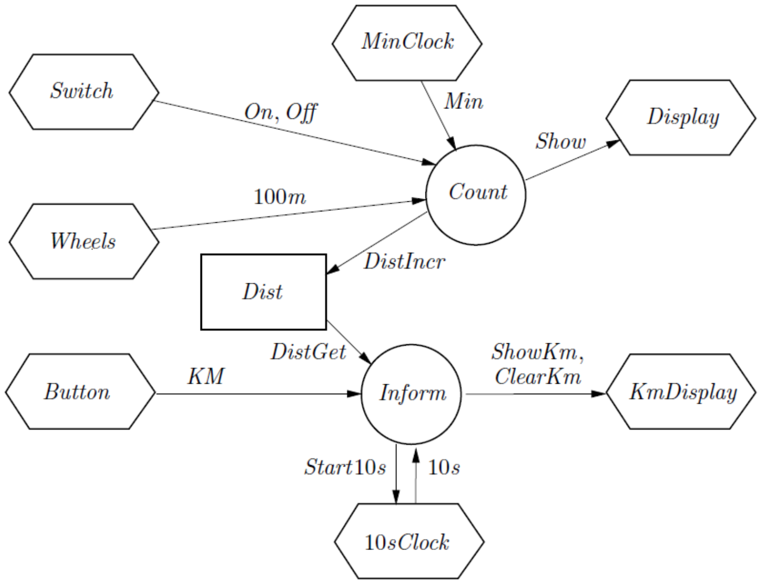
$$\begin{aligned} count &= (On;\ Show;\ ((Min\ [\!]\ 100m);\ Show)^*;\ Off)^* \\ inform &= (KM;\ (Start10s\ \|\ ShowKm);\ 10s;\ Clear)^* \end{aligned}$$

System:

$$taximeter = count \parallel inform$$

Example: Taximeter

Structure diagram:



DTU

Example: Taximeter

Processes:

```
Count =
  var amount : Real;
  do
    On? ;
    amount := StartFee;
    Show ! amount;
    do
      Off? → exit
    []
      [Min? → skip [] 100m? →  DistIncr ! ];
      amount := amount + DeltaAmount;
      Show ! amount
    od
  od
```

```
Inform =
  do
    KM? ;
    DistGet ? km;
    ShowKm ! km;
    Start10s ! ;
    10s? ;
  od
```

DTU

Exercise: Baggage Sorting

- Which events?
- How related?

Limitations of simple design method

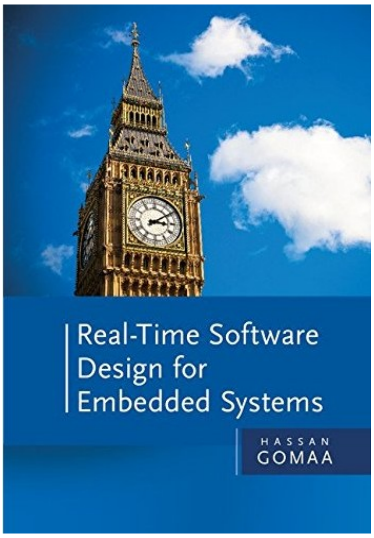
- Flat design – no hierarchies
- Lack of internal events
- No tool support
- Limited scope

Designing with Reactors

- A kind of data flow model
- All activity is event driven
- Internal events are the normal
- Shared state must be managed by a reactor
- Design method still lacking

A Real-Time Design Method: COMET/RTE

- **C**oncurrent **O**bject **M**odeling and Architectural Design **Met**hod for **R**eal-**T**ime **E**mbedded Systems
- By Hassan Gomaa (2016)
- Uses SysML/UML notions
- Well-defined steps
- Many case studies



COMET/RTE Method

Phases:

- Structural modelling (hw, sw, people) [block diagrams]
- Requirements modelling [use cases]
- Analysis modelling [classes, state machines]
- Design modelling [subsystems, active objects]
- Incremental software construction
- Incremental software integration
- System testing

COMET/RTE: Pump case – Analysis modelling

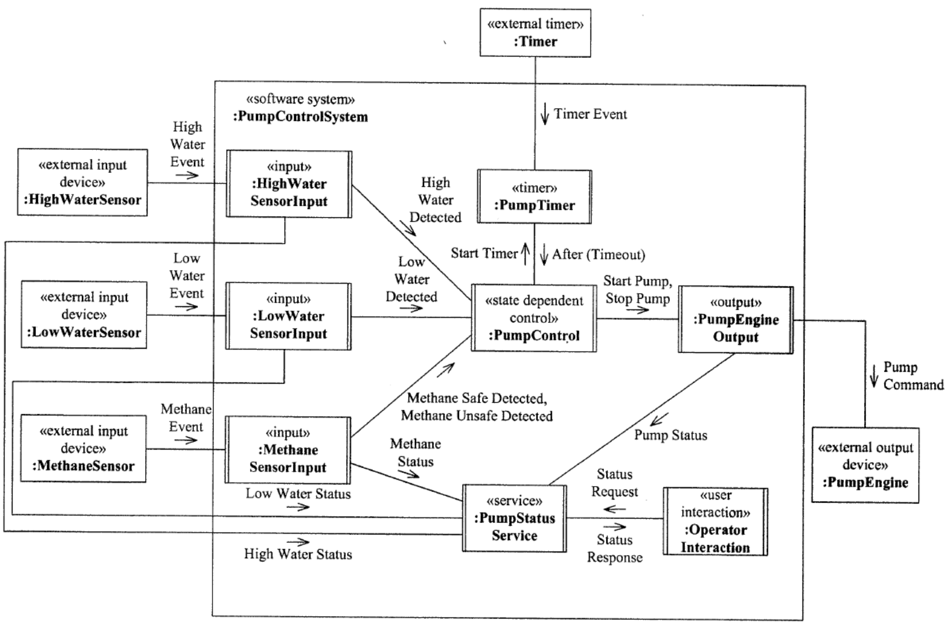
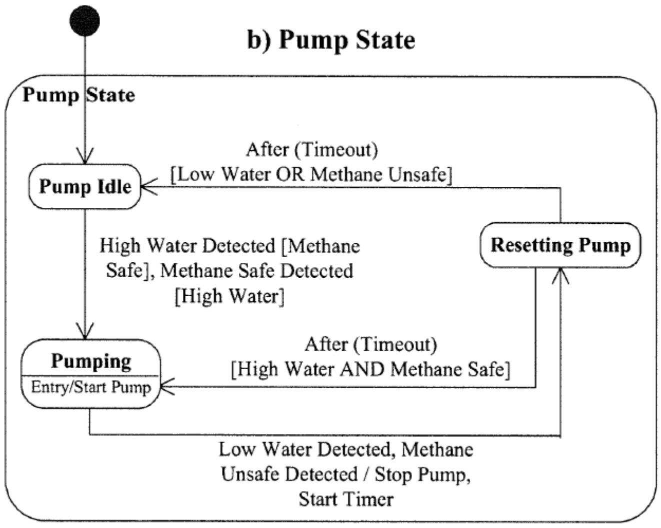


Figure 22.5. Integrated Communication diagram for Pump Control System.

COMET/RTE: Pump case – Analysis modelling



COMET/RTE: Pump case – Design modelling

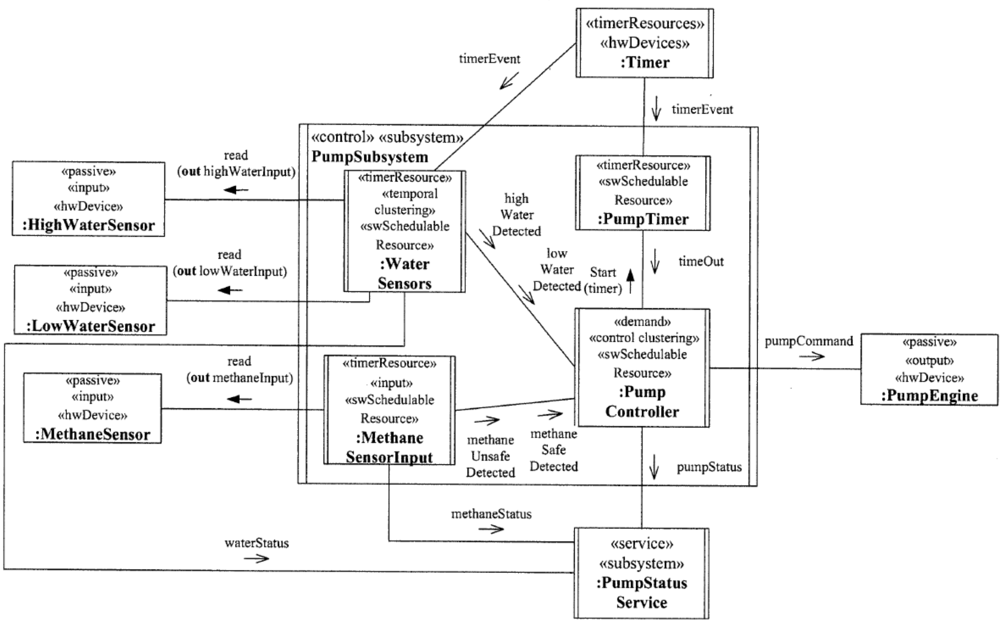


Figure 22.8. Pump Subsystem – task architecture.

COMET/RTE: Pump case – Sensor subsystem

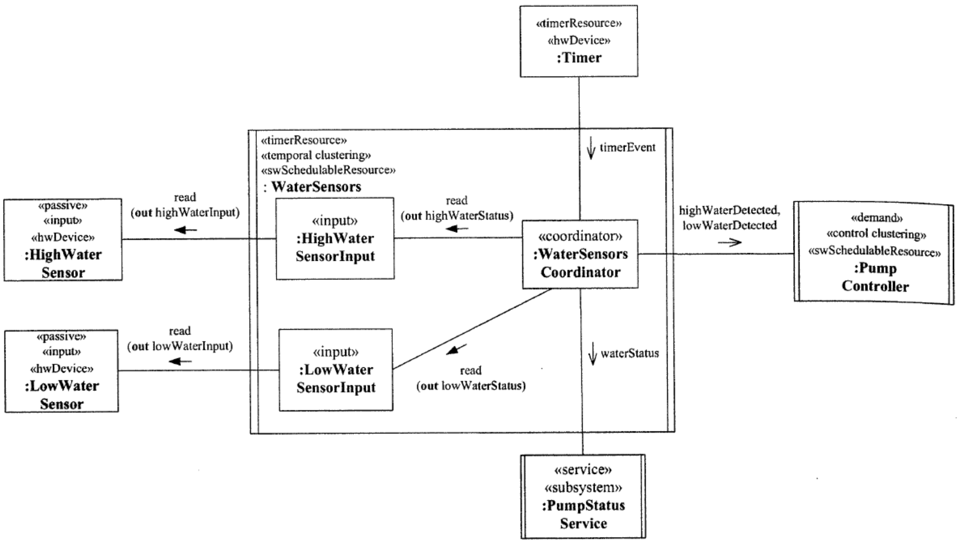


Figure 22.9. Water Sensors – temporal clustering with nested passive objects.

COMET/RTE: Pump case – Control subsystem

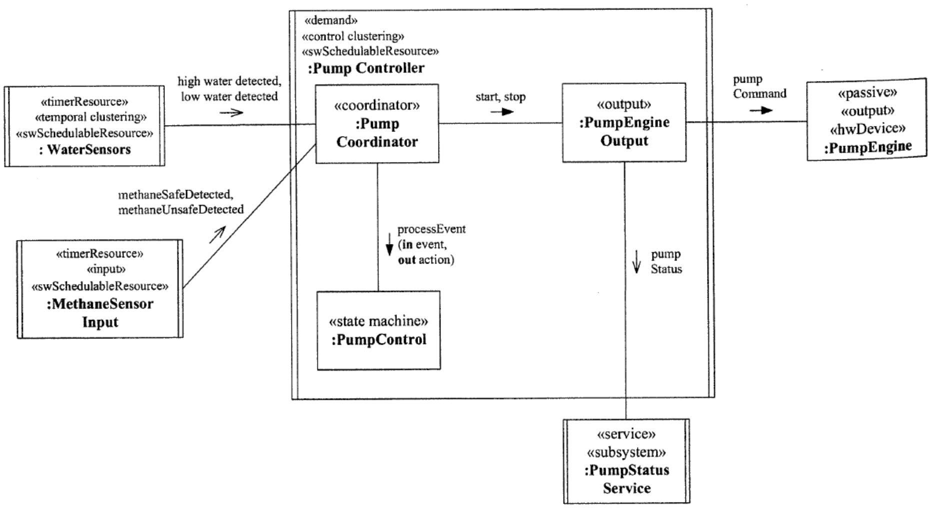


Figure 22.10. Pump Controller – control clustering with nested passive objects.

Design for Real-Time?

Typical approach (separation of concern):

- Design with focus on modularity
- Check RT properties

Final design should be amendable for RT analysis

- Triggering events must be identifiable
- Computation tasks must be quantifiable

Design may have to be twisted to fulfil RT requirements