Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Embedded and Real-time Systems

RT design

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Tomáš Bureš

<bures@d3s.mff.cuni.cz>

CHARLES UNIVERSITY IN PRAGUE

Faculty of Mathematics and Physics

Software Substitute

- Today mechanical and electrical control systems are replaced by computer based solutions.
- Contributing causes are:
  - It is possible to improve already existing technologies, e.g., brakes in cars
  - It is possible to do things previously seemed impossible, e.g., drive-by wire, electronic stability program in cars, etc..
- But …
  - Stress on reliability and safety

Department of Distributed and Dependable Systems

Accidents

- Ariane 5
  - Exploded on June 4, 1996
    - only 39 seconds after launch
    - loss of about US$ 370 million
  - A 64-bit float was truncated to 16-bit integer in a “non-critical software component”
  - This caused unhandled hardware exception
  - The erroneous component (a method) was inherited/reused from Ariane 4 and had no practical use in Ariane 5

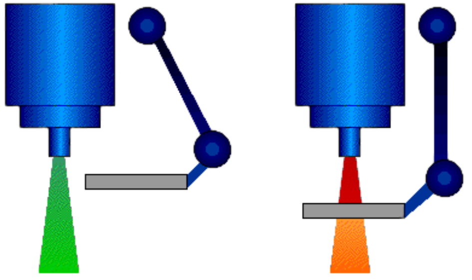
Department of Distributed and Dependable Systems

Accidents

- Patriot – Failure at Dhahran
  - February 25, 1991, an Iraqi Scud hit the barracks in Dhahran killing 28 soldiers
  - The area was protected by Patriot aerial interceptor missiles
  - Due to drift of system's internal
    - by one third of a second in 100 hours
    - amounted to miss distance of 600 meters
    - The system detected the missile but due to the time skew, it disregarded it as spurious

Accidents

- Therac-25
  - Computer controller radiation therapy machine
  - 6 accidents 1985-1987
    - three people died as the direct consequence of radiation burns
  - Race condition as the primary cause
  - Other causes included
    - Poor design, no review of the software
    - Bad man-machine interface
    - Overconfidence in the software
    - Not understanding safety
      - The software was in use previously, but different hardware design covered its flaws




```
PATIENT NAME    : JOHN DOE
TREATMENT MODE : FIX  BEAM TYPE: X    ENERGY (MeV): 25

                      ACTUAL        PRESCRIBED
    UNIT RATE/MINUTE    0              200
    MONITOR UNITS      50 50           200
    TIME (MIN)         0.27            1.00

GANTRY ROTATION (DEG)     0.0            0.0      VERIFIED
COLLIMATOR ROTATION (DEG) 349.2          359      VERIFIED
COLLIMATOR X (CM)         13.2           14.3     VERIFIED
COLLIMATOR Y (CM)         21.2           27.3     VERIFIED
WEDGE NUMBER                1              1      VERIFIED
ACCESSORY NUMBER            0              0      VERIFIED

DATE    : 84-DEC-27   SYSTEM : BEAM READY    OP. MODE : TREAT AUTO
TIME    : 12:55: 8    TREAT  : TREAT PAUSE              X-RAY 173777
OPR ID  : T25V02-R03  REASON : OPERATOR      COMMAND:
```

This lecture

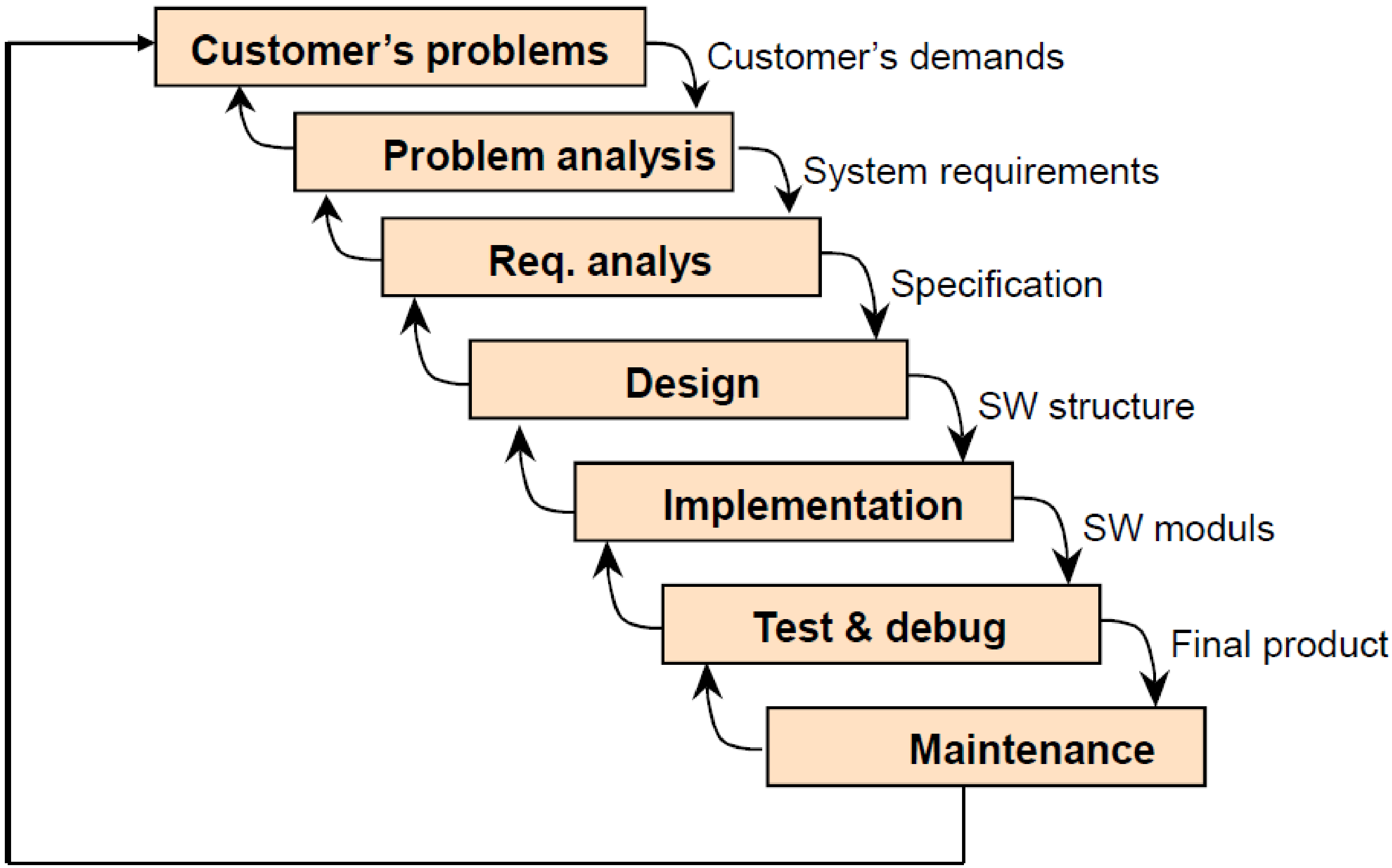
- How to obtain task attributes from a given specification
- How to design a systems that has timing requirements
- More specific: Real-time Talk (RTT) design method

Design

- What is design?
  - A high-level description of the system
  - You don't start building your house without a plan (drawing) …
- Why design?
  - A tool for the system constructor
  - Documentation
  - Parallel development
  - Evaluation on an early stage
  - Easier testing and verification
  - Automatization

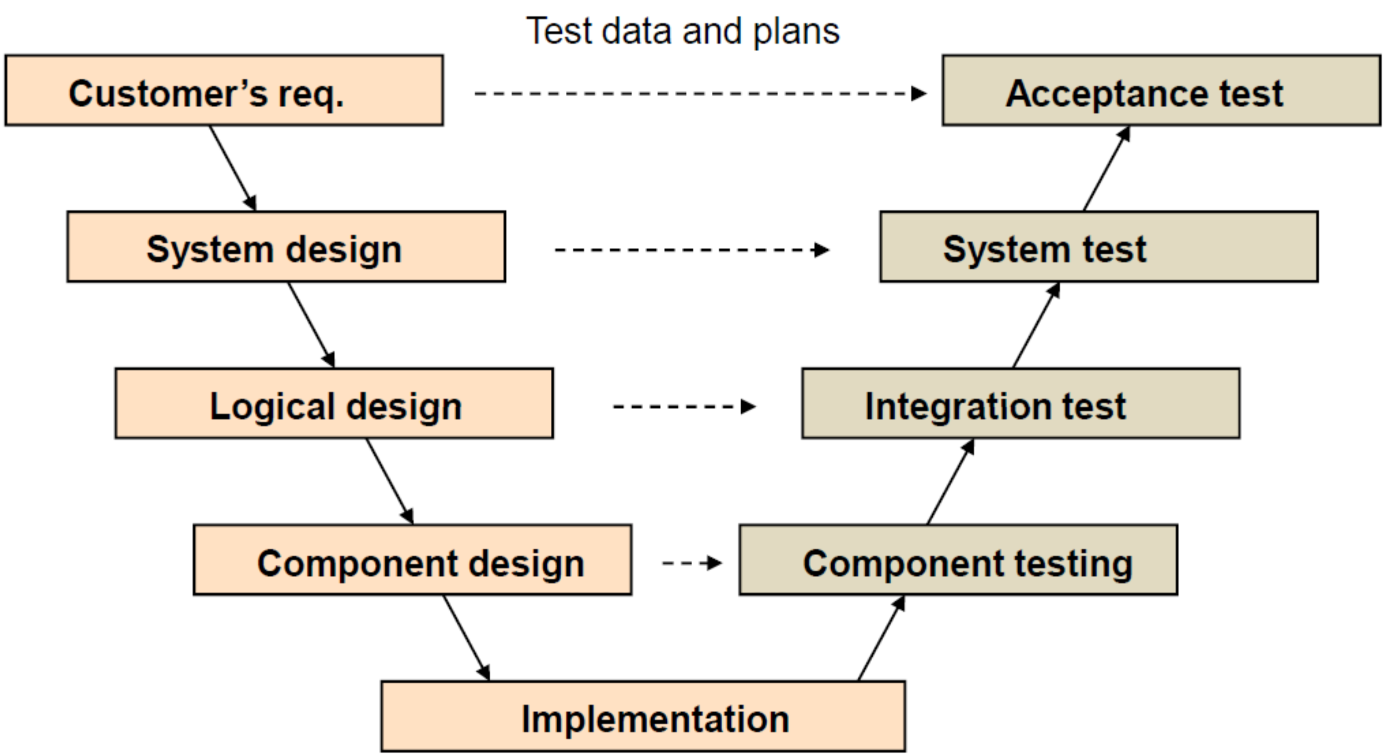
Design methods

- The waterfall model is the one of the oldest and the most famous design method



Design methods

- V-model introduces early testing in design process
- Test plans and test data produced while designing the system the time needed to test the system is decreased



Design methods

- Agile methods
  - XP, RUP, Scrum, …
  - characterized by
    - regular rapid cycles which create executable deliverables
    - focus on coding rather than planning or documentation
    - refactor continually to improve code
    - communicate continually and extensively within the development team
    - communicate continually and extensively with customers
    - continually measure project progress, extrapolate projections, adjust long-term project goals (project end date and features set), set short-term goals (work elements for the next iteration)
    - use test-driven development to verify that code is initially correct, and emphasize regression testing to ensure that the code stays correct

    [Doug Dahlby: Applying Agile Methods to Embedded Systems Development]

Design methods – Agile methods

- Agile methods must be adapted for development of embedded RT-systems – details below

- Rapid fixed-length cycles each delivering a new version of the product – not always possible
  - substantial initial effort needed to set up the development, debugging and testing infrastructure (simulator, etc.)
  - embedded systems are often monolithic (i.e. not easily separable to independent features)
  - the required features are often more clear upfront than in enterprise systems

Design methods – Agile methods

- Focus on coding rather than planning or documentation – not completely possible
  - embedded and real-time systems are often optimized – thus lacking on the self-documentability of the code
  - need for objective proof that the software works correctly
    - requires additional artifacts such as test/requirements traceability, test records, test coverage records, …
  - ability to keep the software working correctly in the future (even for decades) needed
    - may require additional artifacts such as architecture documentation (high-level structure of the system) and design documentation (trade-offs, alternatives, design decisions),
  - user documentation needed
    - continuously maintained user instruction manual, installation guide, change log, feature summary, errata list

Design methods – Agile methods

- Refactor continually to improve code – beneficial
  - more difficult because code is optimized
  - dependence on hardware makes refactoring more costly
    - if hardware has to be upgraded
    - necessity of having a good “initial guess”
  - impractical to have universal code ownership because many software specializations are combined
    - operating systems, control theory, signal processing, communication protocols, user interfaces

Design methods – Agile methods

- Communicate continually and extensively with customers – not completely possible
  - design and optimizations in embedded systems often require deep understanding to electronics, physics and mathematics
  - a story like “replace Gaussian elimination with L-U decomposition” is not something that you can consult with a customer
  - requires engineering management that has technical and business skills to supplement the interaction with customers

Design methods – Agile methods

- Communicate continually and extensively withing the development team – beneficial
  - is beneficial if the informal communication does not result in scarcity of formal written documents
- Continual measurements, planning, projections, and adjustments by management – beneficial
  - helps with hard to predict development time
    - due to coupling with hardware, more difficult debugging, etc.
- Test-driven development and regression testing – beneficial
  - is more difficult due to timing constraints and coupling with HW – use of simulators
  - difficulties to test real systems due to problems in probing into the system

Design of real-time systems

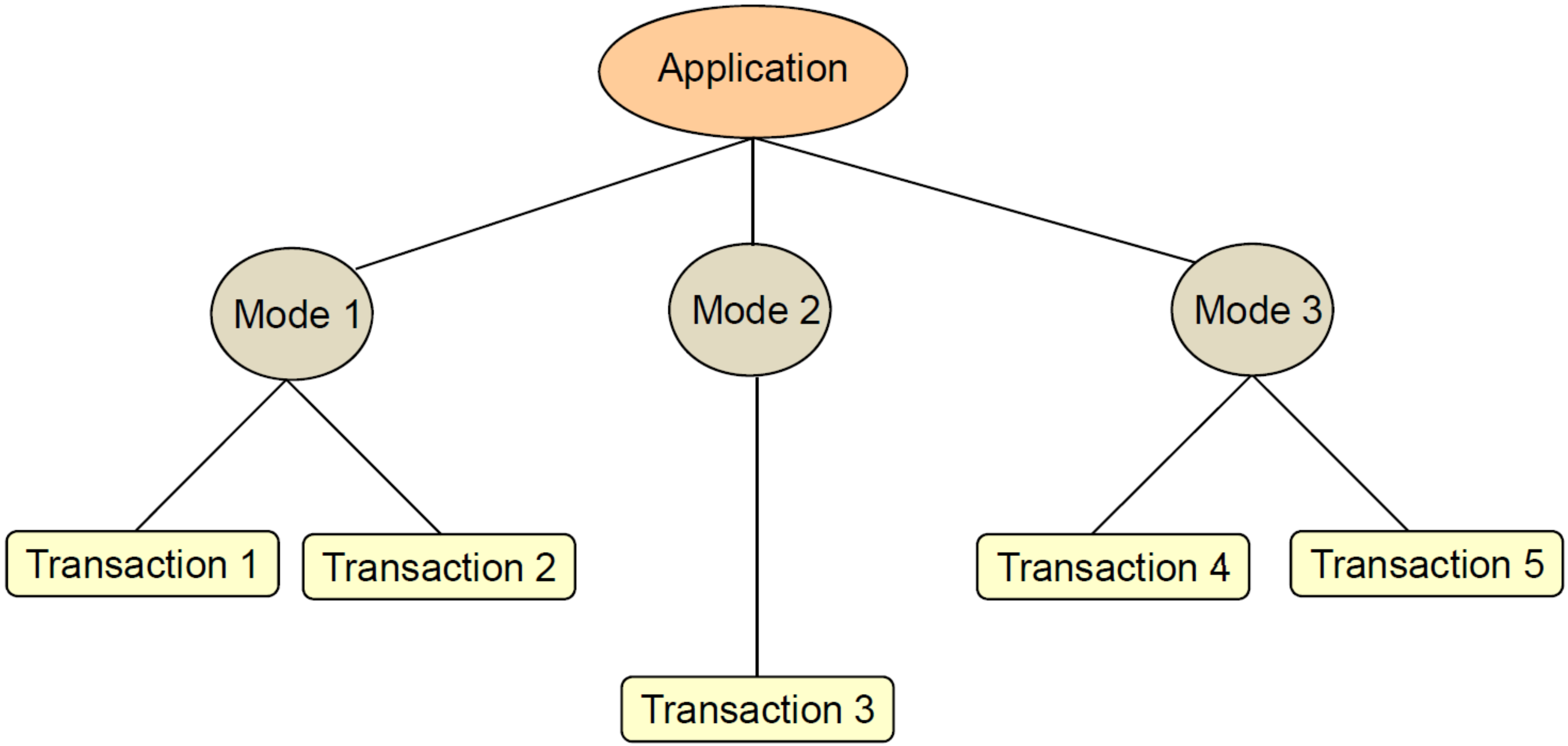
- Design of real -time systems
  - An extra design parameter: TIME!
  - It makes the design more difficult
  - We need design methods!
- Very few design methods for real-time systems
  - No support for the temporal domain
  - Existing methods focus on the structure (data and control flow)
  - Usually general methods with a “patch” for time

RTT design model

- Developed at MDH university
- Used in industry
  - Volvo Construction Equipment
  - Design of control systems of wheel loaders
- Simple model
  - Can be combined with “standard” development processes (e.g. V-model, agile)
  - Few powerful constructions suited for RT systems, not a general purpose design method that suits “everything”
  - Break down a huge problem into smaller problems that are easier to manage

RTT application model

- Pre-defined design objects with strict semantics



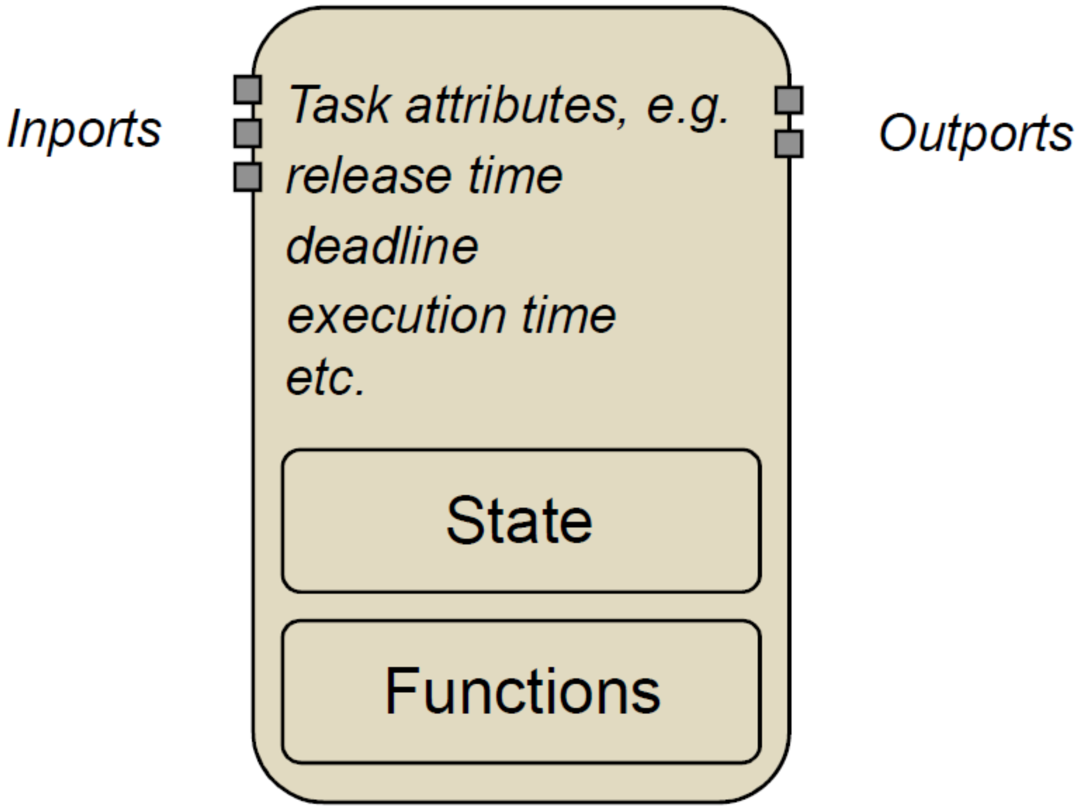
  - Mode: describes functionality in a certain system state
  - Transaction: A set of tasks that provides a certain function

Department of Distributed and Dependable Systems

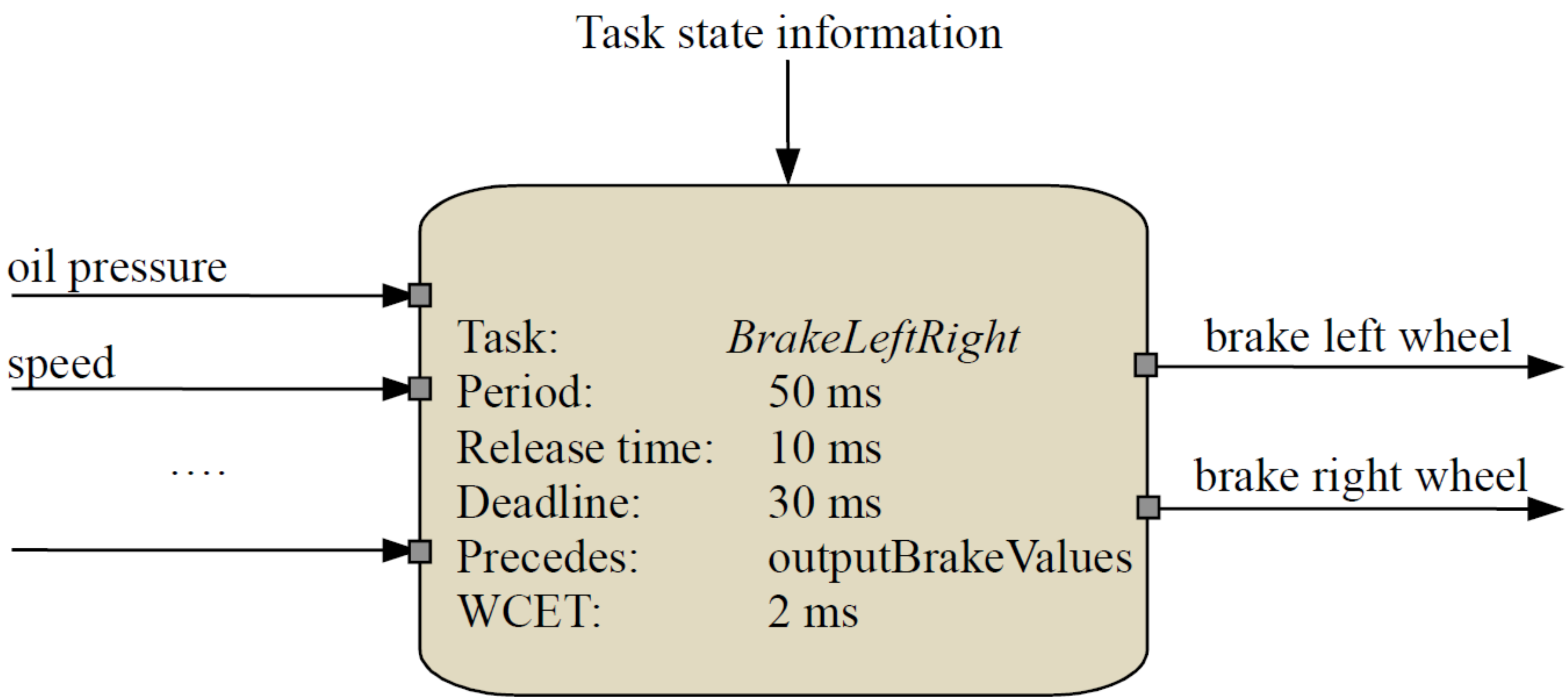
Figure from Issovic, D.:Real-time systems, basic course

Tasks in RTT

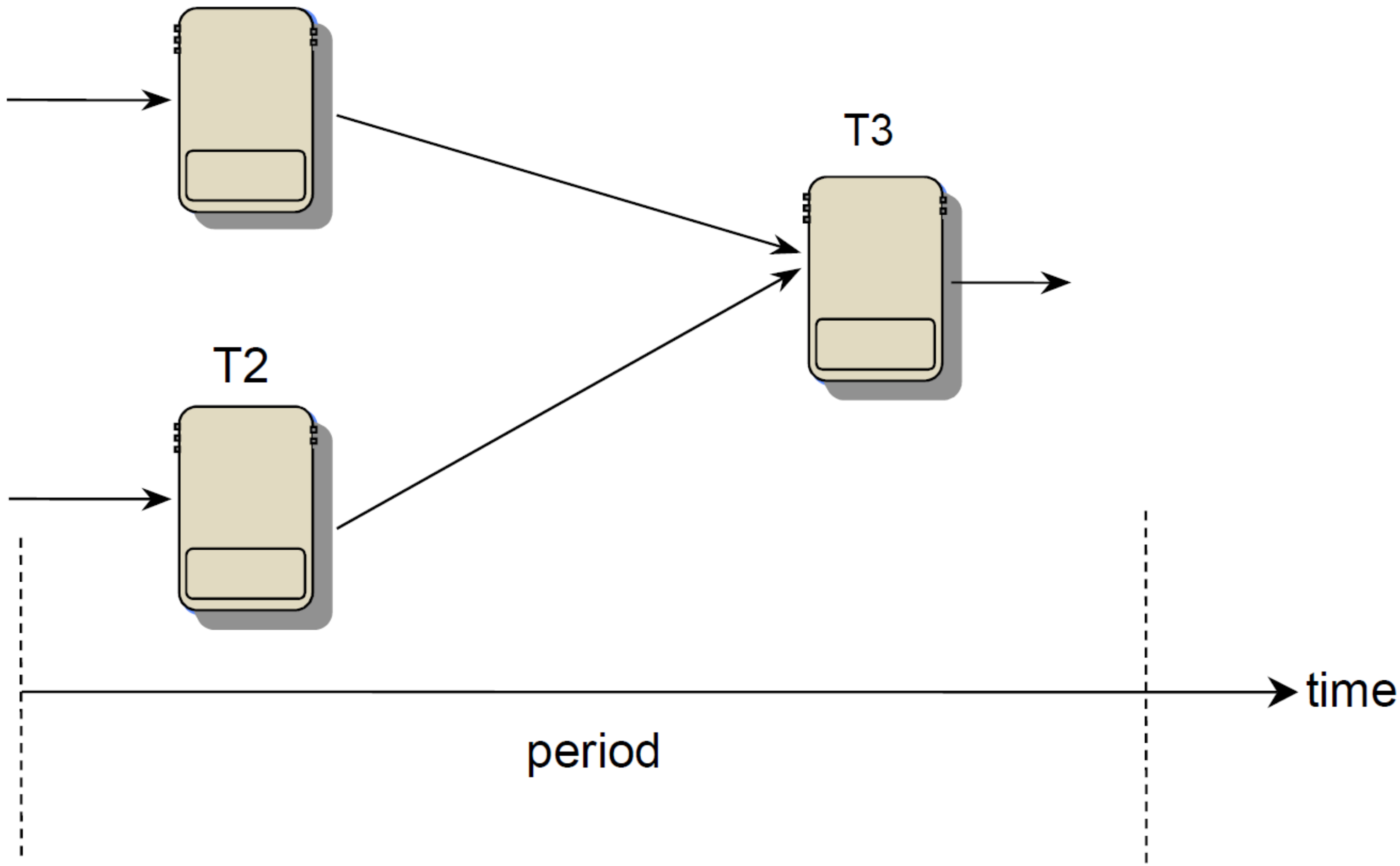
- A task is defined by its temporal behavior, states and function



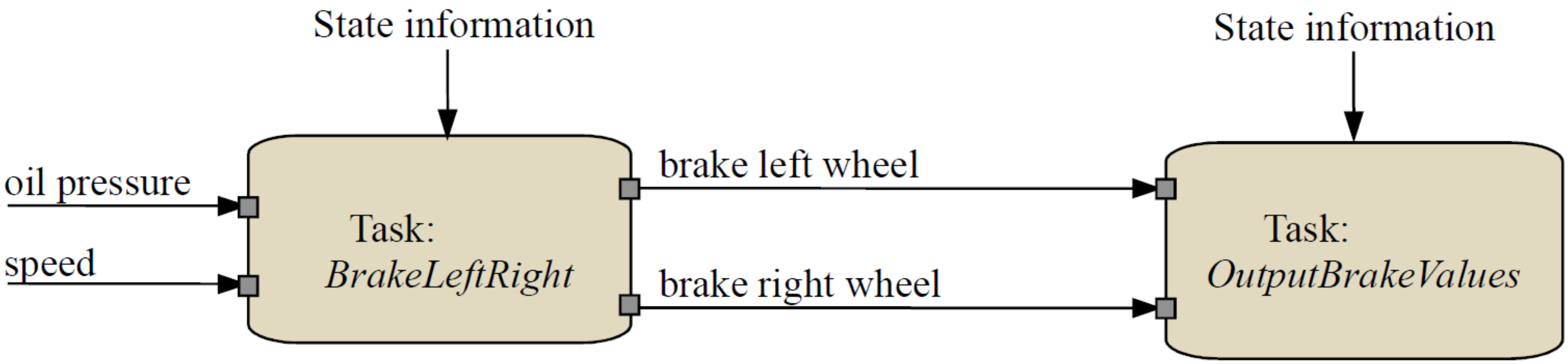
Example task and its interfaces



Precedence graph

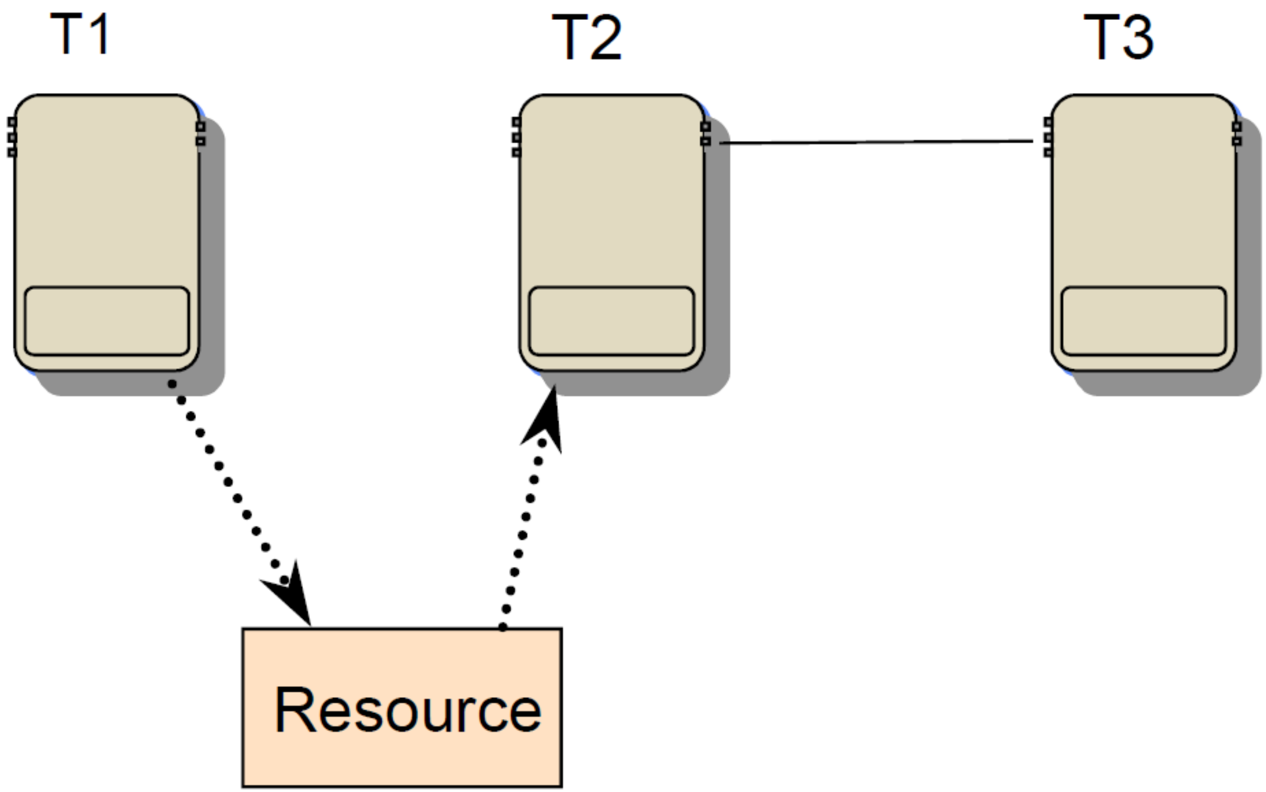


Example composed system with precedence



Interaction graph

- Communication between tasks in a mode
- Shared resources (mutual exclusion)



RTT design methodology

- Iterative method
- Hierarchical decomposition
- Early integration
- Comm. and synch. separated
- WCET estimation
- Automatic temporal
- verification (scheduling)

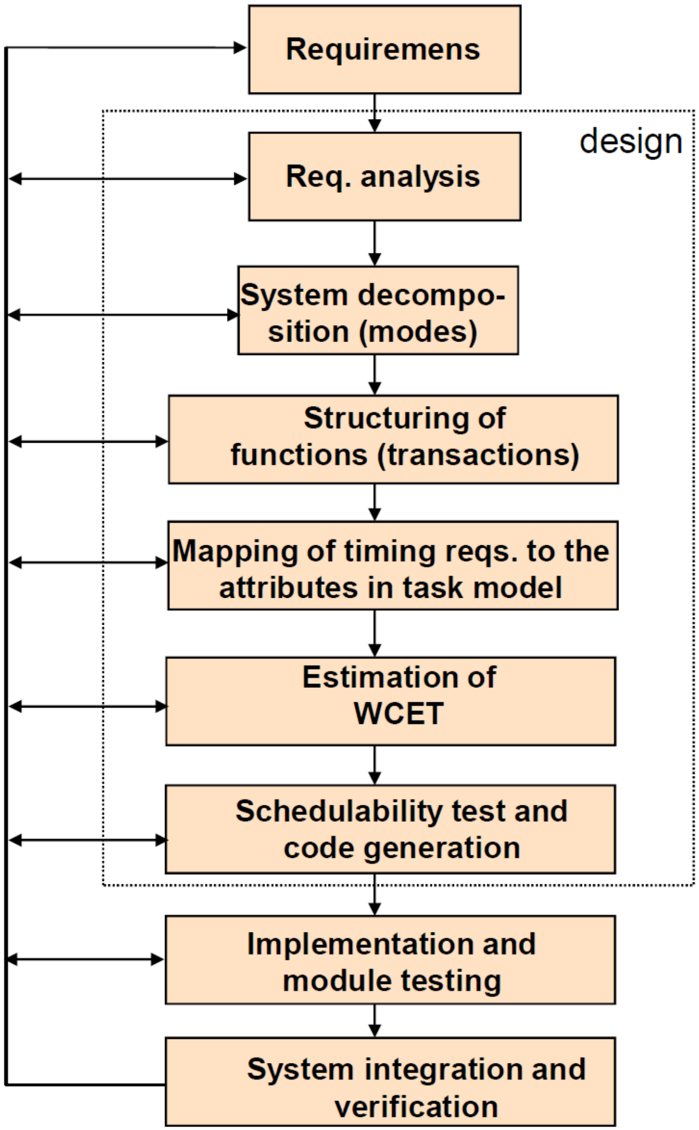


Figure from Issovic, D.:Real-time systems, basic course

Example: Control of a truck bed

- Design of a simple control system by using RTT

- Assignment:
  - To control the truck bed with a motor and some sensors, without damaging the truck bed
  - The driver must be warned if (s)he starts driving with the truck bed up

Department of Distributed and Dependable Systems

Information about the system

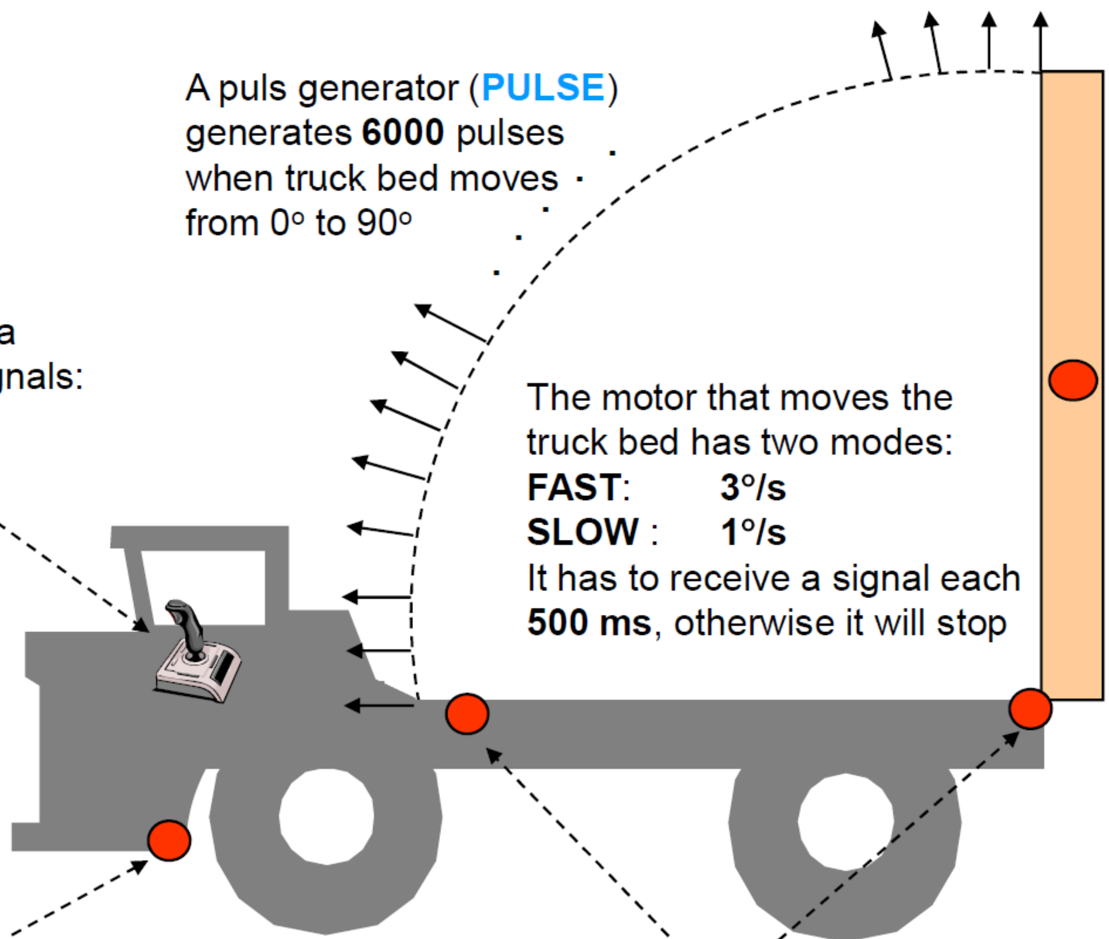
Truck bed can be raised from **0° to 90°**

A puls generator (**PULSE**) generates **6000** pulses when truck bed moves from 0° to 90°

The driver has a joystick with signals:
UP: **+1**
STOP: **0**
DOWN: **-1**
to control the truck bed

The motor that moves the truck bed has two modes:
FAST: **3°/s**
SLOW : **1°/s**
It has to receive a signal each **500 ms**, otherwise it will stop

A position sensor (**POSITION**) indicates the movement of the truck bed (up or down)

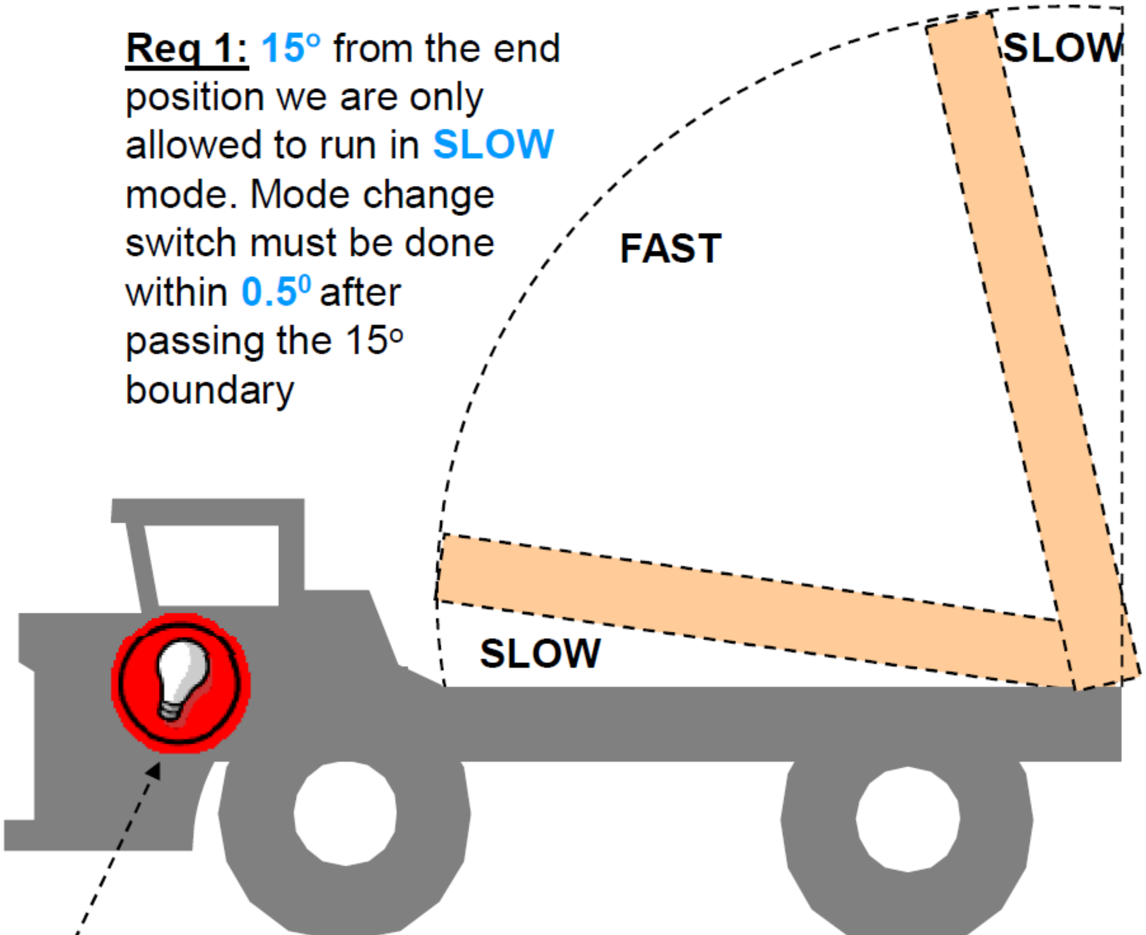


A movement sensor (**MOVE**) to detect the movement of the truck

Two end-sensors **UP** (90°±0.3°) and **DOWN** (0°±0.3°) to detect end positions

Requirements from the customer

Req 1: 15° from the end position we are only allowed to run in SLOW mode. Mode change switch must be done within 0.5⁰ after passing the 15° boundary



Req 2: The motor must not be running in the end-positions longer than 0.4s, otherwise it can get damaged

Req 3: If the driver starts driving with the truck bed up, a warning signal must be issued and the truck bed motor must be stopped within 0.5s. No joystick command allowed when the vehicle is moving.

Timing requirements

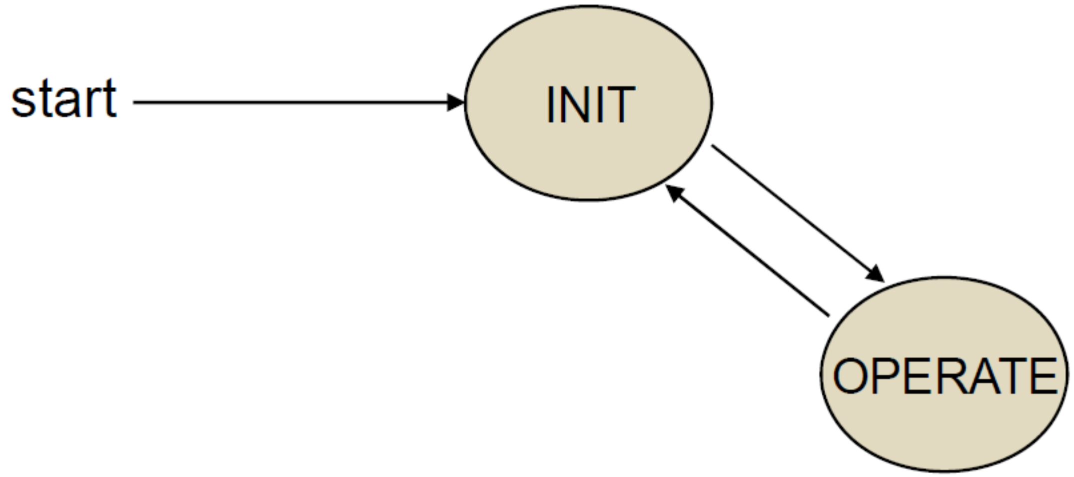
- Requirements for motor control
- switch from FAST to SLOW within 0.5º → 167 ms to switch
- [speed = 3º/s, distance = 0.5º ⇒ t = distance/speed = 0.5º / (3º/s) = 167 ms]
- must get a new control signal at least every 500 ms
- must not run the motor in the end-position longer than 400 ms

- Requirements for keeping track of the current position of
- the truck bed:
- no explicit timing requirements, but we must detect ALL pulses

- Requirements to warn and stop the truck bed motor if the
- vehicle is moving with the truck bed up:
- We have 500 ms to stop and warn

Decomposition into modes

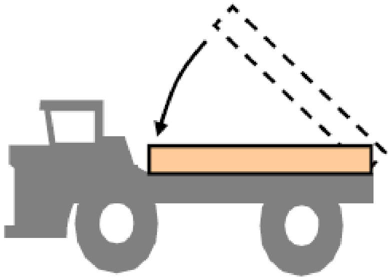
- OPERATE
  - An obvious mode to control the truck bed
  - Is it enough?
    - No, since we don’t know the initial position of the truck bed (when we start to use it) Hence we need an INIT mode
- INIT
  - If something goes wrong, we recalibrate the system by entering the INIT mode



Department of Distributed and Dependable Systems

INIT mode

- Transactions
  - Transaction 1: control the truck bed into a defined position (start position) and then switch to OPERATE
- Tasks
  - INIT – run to the start position and switch to OPERATE mode



- Interaction graph

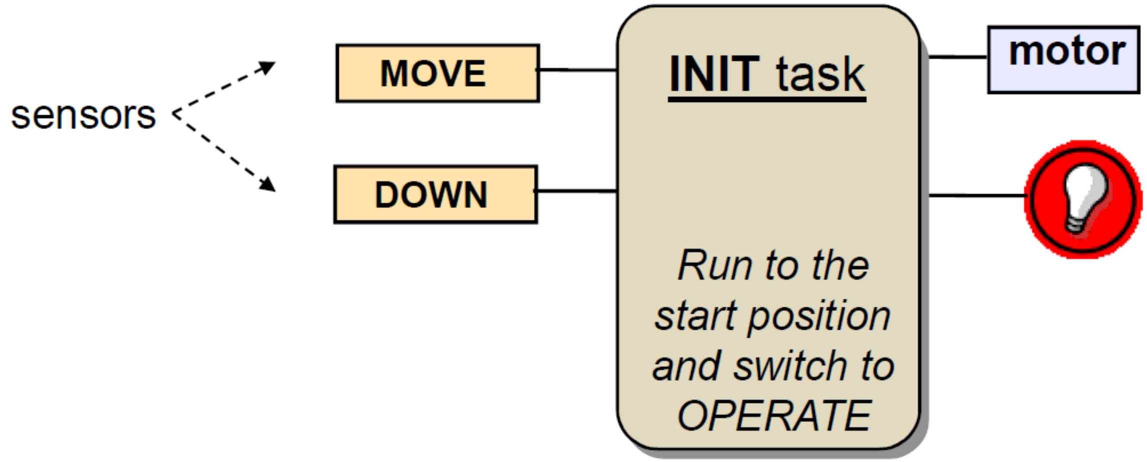
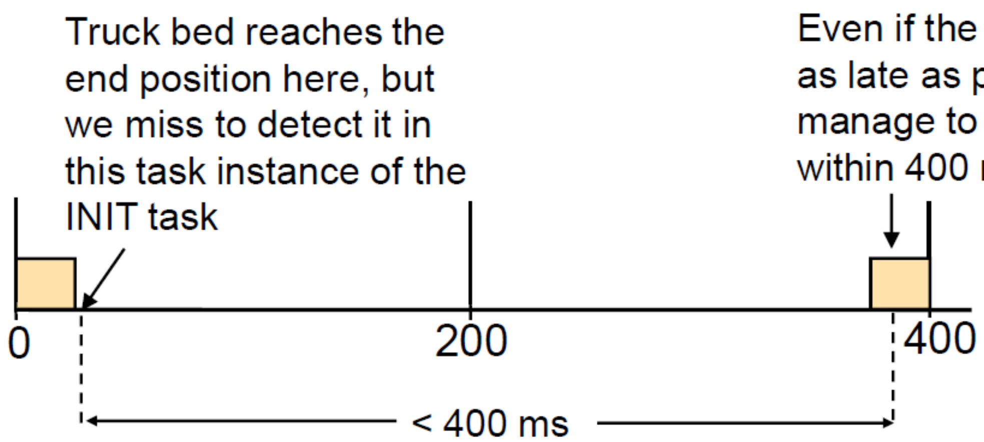


Figure from Issovic, D.:Real-time systems, basic course

INIT mode - timing constraints

- Requirement 1: The motor must not run in end-position longer than 400 ms

Worst-case scenario:



$T_{INIT} = 200$ ms

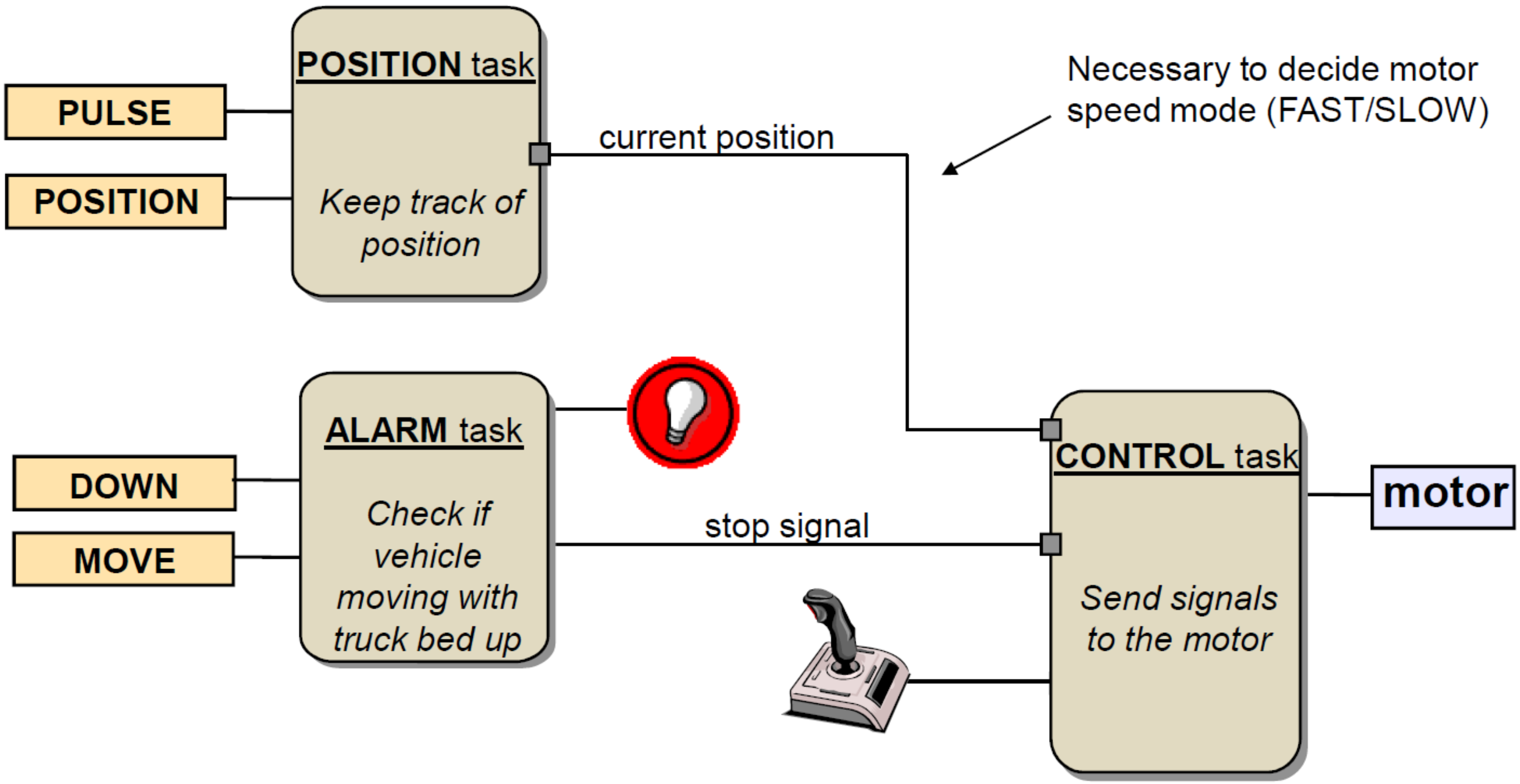
$D_{INIT} = 200$ ms

- Requirement 2: Warn and stop within 500 ms
- Previous requirement is more strict than this one, hence T=200 ms is good enough to fulfill this requirement
- Requirement 3: The motor must get a signal each 500 ms
- The same reasoning as above, T=200 ms is enough

Figure from Issovic, D.:Real-time systems, basic course

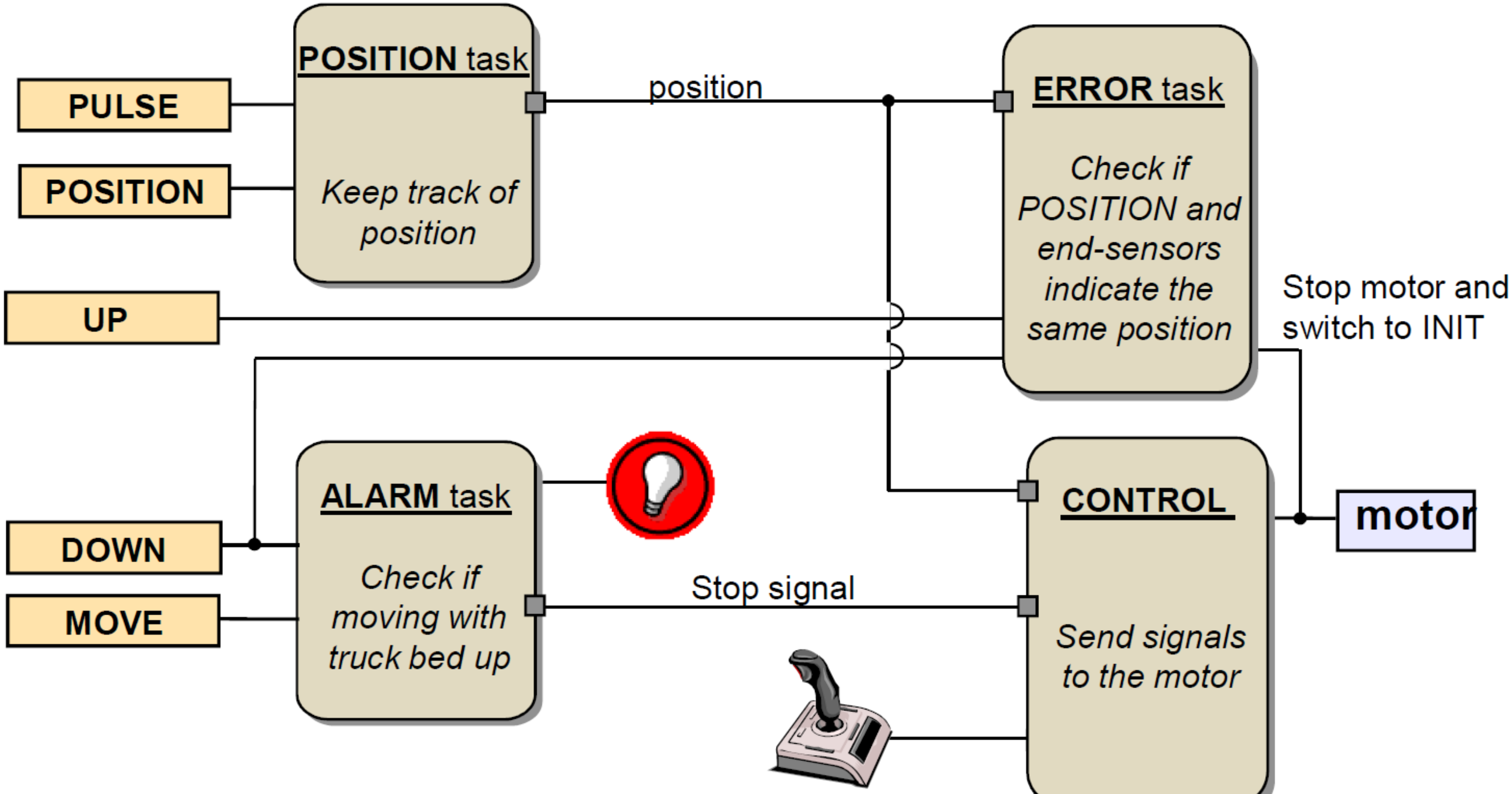
OPERATE mode

- Transactions and tasks
  - Transaction 1: keep track on position and control the motor (tasks: POSITION, CONTROL)
  - Transaction 2: detect vehicle movement, stop and warn (tasks: ALARM)



OPERATE with error handling

- Transaction for error handling
  - Transaction 3: check if position task and end sensors indicate the same position (task: ERROR)



OPERATE mode – timing requirements

- Requirement 1: We must detect all pulses
- How often do we need to invoke POSITION task in order not to miss OPERATE mode – timing constraints any of the pulses, i.e., what is the period of POSITION task?

- 6000 pulses/90° and 3°/s → 200 pulses/s

- Nyquist-Shannon sampling theorem: The sampling frequency of a signal must be strictly greater than twice the highest frequency of the input signal

- To detect all pulses, we must sample at least 400 times/s
- We can easily manage that with a period of 2 ms

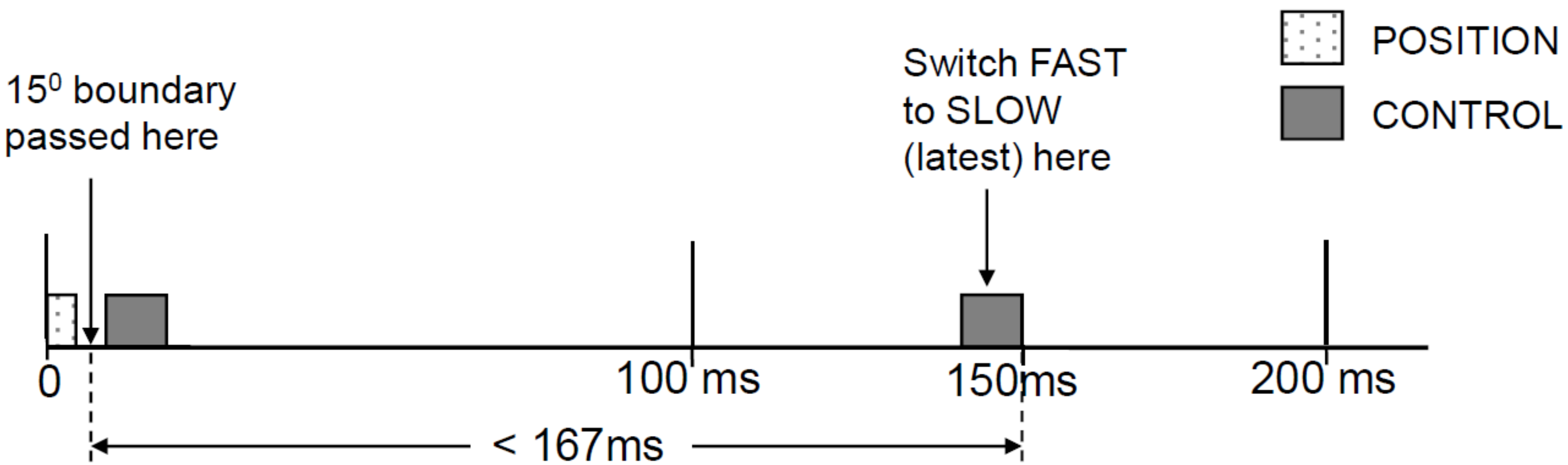
$$T_{position} = D_{position} = 2\,ms$$

Department of Distributed and Dependable Systems

OPERATE mode – timing requirements

- Requirement 2: We must switch from FAST to SLOW within 0.5°
  - It takes 167 ms for the motor to move the truck bed 0.5°.
  - How often do we need to invoke the CONTROL task?
    - Alternative 1: period is less or equal to 167/2 ms (as in previous cases)
    - Alternative 2: set longer period and shorter deadline to enforce earlier execution (scheduling)

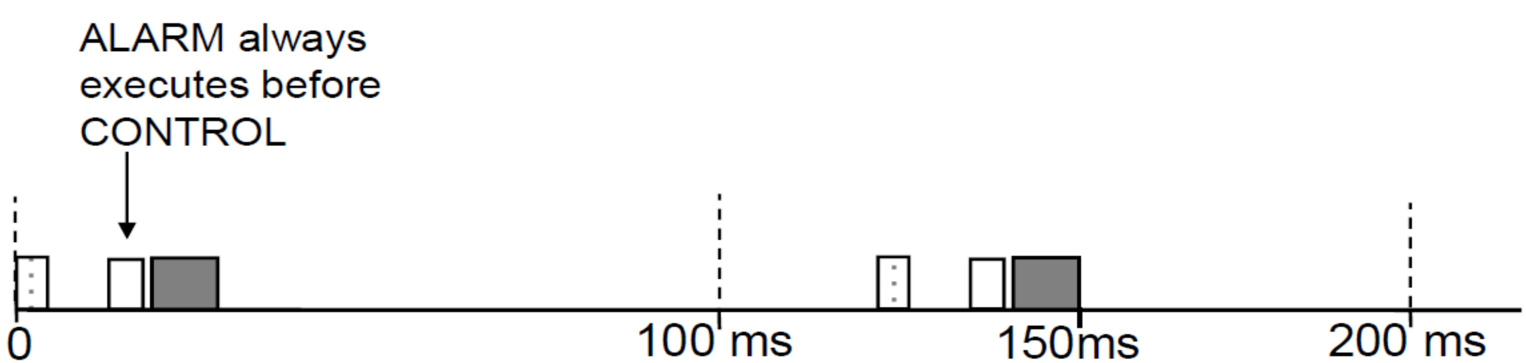
- Worst case scenario:



$T_{CONTROL}$=100 ms and $D_{CONTROL}$=50 ms

OPERATE mode – timing requirements

- Requirement 3: We must detect the movement and warn the driver within 0.5 s
  - What requirements do we have on the ALARM task?
  - If we make sure that ALARM is always executed before CONTROL, then CONTROL will get the latest info → we put precedence relation between ALARM and CONTROL
  - Hence, the timing attributes for the ALARM task:
    - T =100 ms
    - D = 50 ms
    - ALARM precedes CONTROL



OPERATE mode – timing requirements

- Requirement 4: The motor must not run in an end-position longer than 0.4 s
  - Can we meet this requirement with T=100 and D=50 for the CONTROL task (derived from requirement 2)?
    - **Yes!** It takes max 150 ms from detecting the end-position to stopping the motor (see illustration for requirement 2)
  - What happens if POSITION misses a pulse?
    - The ERROR task must detect the end-position and stop the motor within 400 ms
    - This is achieved if we set a period for ERROR to T=200 ms

- Worst case scenario:

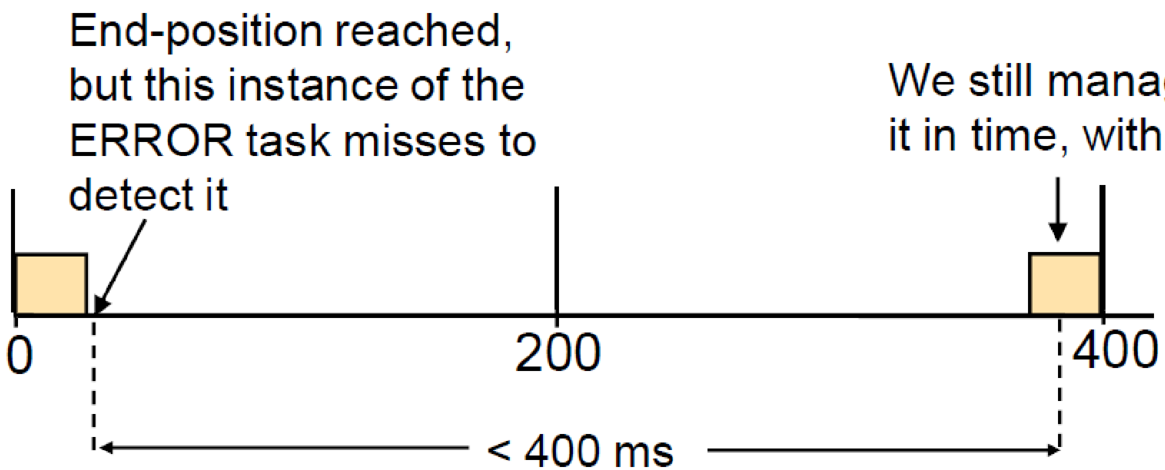
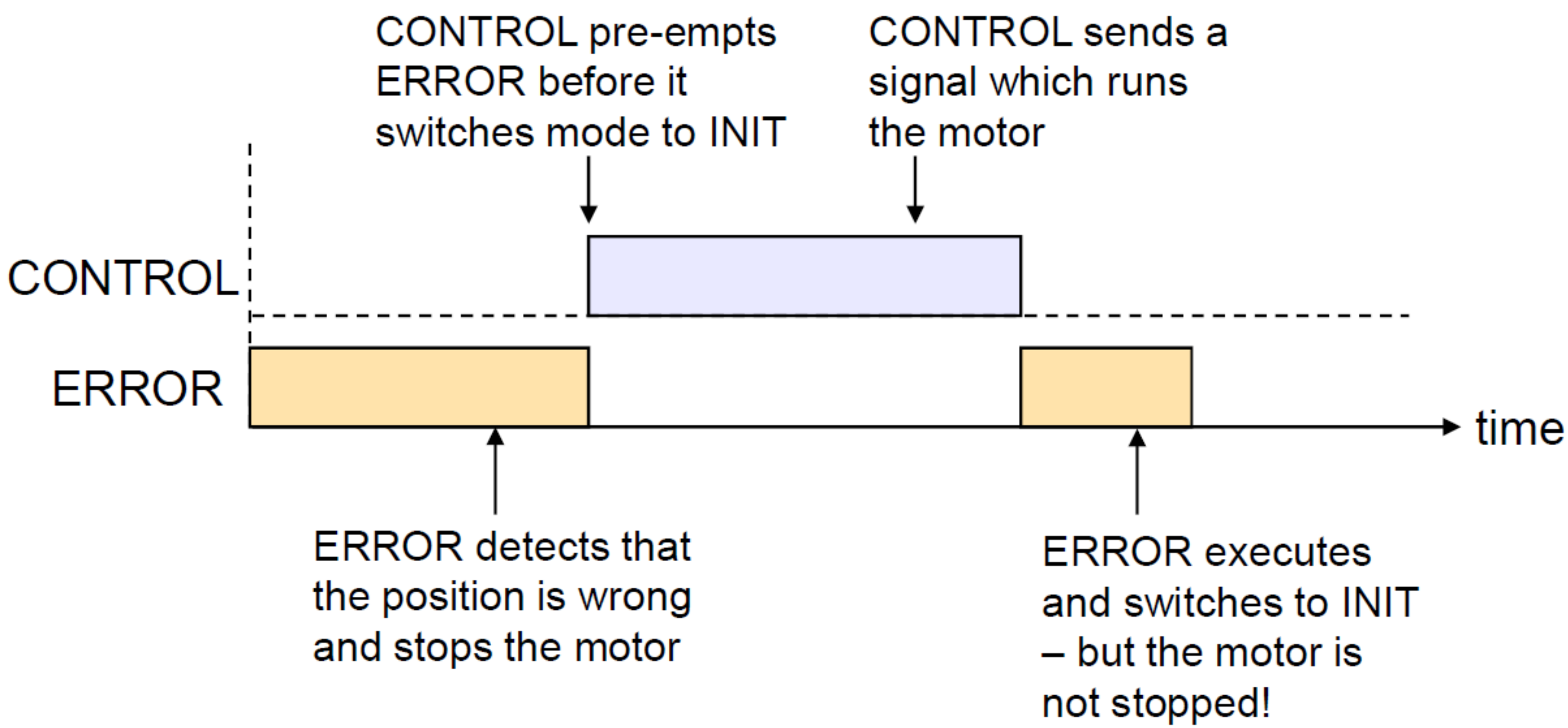


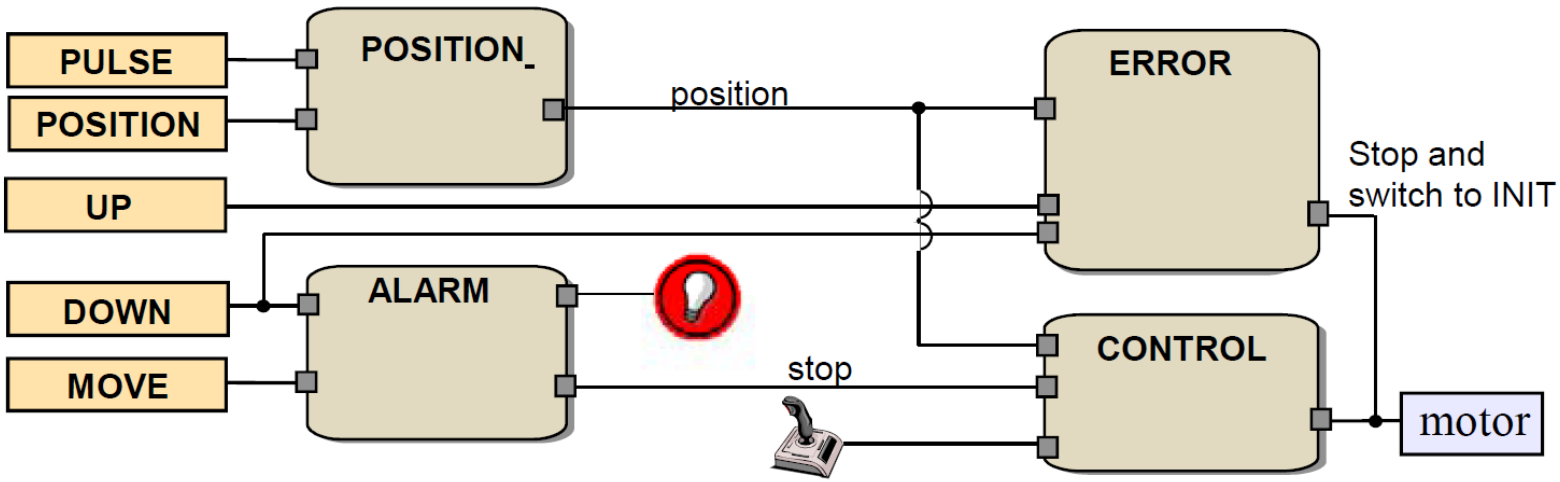
Figure from Issovic, D.:Real-time systems, basic course

OPERATE mode – Error handling

- Are we done?
- What happens if ERROR gets pre-empted by CONTROL?
  - The problem is a common resource: the motor
  - Solution: mutual exclusion between CONTROL and ERROR



Truck bed – Final design



INIT mode

Period: T=200 ms

INIT task
RT=0 ms
D=200 ms

OPERATE mode

Period: T=2 ms

POSITION task
RT=0 ms
D=2 ms

Period: T=100 ms

ALARM task
RT=0 ms
D=50 ms

→

CONTROL task
RT=0 ms
D=50 ms

Period: T=200 ms

ERROR task
RT=0 ms
D=200 ms

MUTEX: CON, ERR

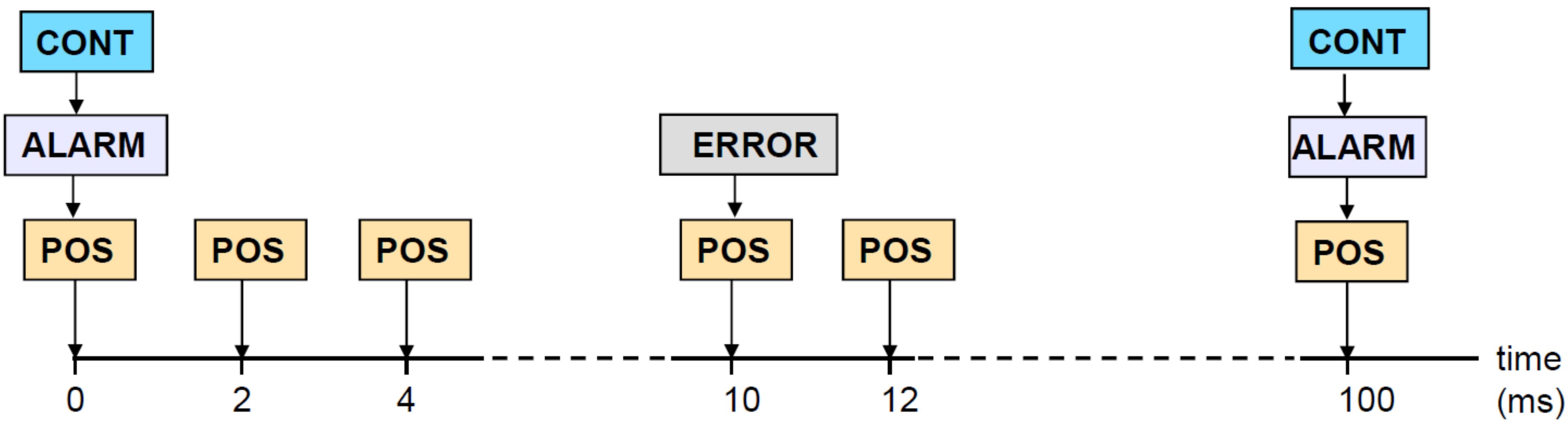
Estimated WCET:

INIT:	1 ms
POSITION:	1 ms
ALARM:	2 ms
CONTROL:	2 ms
ERROR:	1 ms

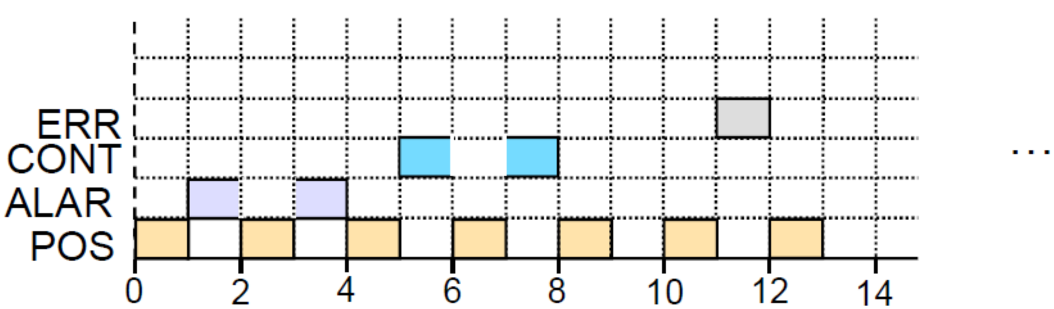
Figure from Issovic, D.:Real-time systems, basic course

Example impl. – Offline scheduling

- A possible offline schedule that fulfills the specification



A possible trace according to the schedule above:



Example impl. – Online scheduling

- CONTROL and ERROR are sharing a resource (motor)
  - We need a resource access protocol, e.g., Priority Inheritance Protocol (PIP) or Priority Ceiling Protocol (PCP)
- Precedence relation between ALARM and CONTROL
  - This can be achieved by setting the right priorities
- We can use Response Time Analysis to check schedulability

Task	C	RT	dl	T	priority	R
POS	1	0	2	2	1 (high)	?
ALARM	2	0	50	100	2	?
CONT	2	0	50	100	3	?
ERR	1	0	200	200	4 (low)	?

$$R_i = C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Department of Distributed and Dependable Systems

Summary

- Design is a very important part of SW development
  - Higher abstraction
- Design of real-time systems
  - More complex due to the timing requirements
- RTT-model
  - Specific for development of real-time systems
  - Can be combined (on the logical design level) with other standard design methodologies
- Design should be separated from the implementation
  - We do not need to make decision about the implementation when making design (truck bed example with offline and FPS)