TSW – Introduction

Alexandre David

1.2.05

AALBORG UNIVERSITY DENMARK

Credits: some slides by Alan Burns & Andy Wellings.

Teachers

- Main teachers
  - Alexandre David ← SW5 coordinator
  - René Rydhof Hansen
- Guest lecturers
  - Brian Nielsen
  - Jens Alsted

Course overview

- Introduction to RTS – 1
- Fault tolerance – 2
- NXT sensors & actuators
- RT facilities – 9,10
- RT analysis – 11
- OSEK on NXT, case-study
- Project presentations
- UPPAAL
- Times tool
- Concurrent programming – 4
- Synchronization – 6,5
- Atomicity, deadlocks – 7,8
- Programming RTS – 10,12
- Timing faults – 13
- Exceptions – 3
- Low-level programming – 14
- Project presentations

Basics, tight schedule early for projects. Project: define, analyze, experiment with sensors.

Deadline

2nd main part in parallel with projects.

Last non vital concepts.

Deadline

Goals of the course

- Understanding of real-time systems
  - focus on the software side
    → requirements on languages and OS
  - concepts of scheduling, timing, concurrency, and correctness
    → how to fullfil those requirements
  - practice through the projects.

What is a real-time system?

- Let’s discuss these terms:
  - **real**-time
  - response time
  - sensors/actuators
  - reactive system
  - embedded systems
  - safety-critical systems
- Which systems are RT?
- Where are they?

What is a real-time system?

- A real-time system is an information processing system which has to respond to externally generated input stimuli within a finite and specified period of time.
  - Respond to external stimuli → reactive system.
  - Correctness depends on
    - the logical result - right result - and ✓
    - the time of delivery - right time.
  - This system is part of a larger system → embedded computer system.
    - Note: 99% of all processors are for embedded systems.

Terminology ✓

- Hard real-time systems: responses **must** occur before the specified deadline otherwise the system does not work and (usually) breaks.
  - Braking system, air traffic...
- Soft real-time systems: responses **should** occur before the specified deadline but may still work, possibly in a degraded mode, if occasional deadlines are missed.
  - Video conference, data acquisition...
- Firm real-time systems: have timing requirements typical of hard real-time systems with service requirements typical of soft real-time systems.
  - Allow RT and non RT tasks to co-exist.
  - *No benefit in late delivery of service.*
- Abstraction: associate a cost function to missing deadlines.

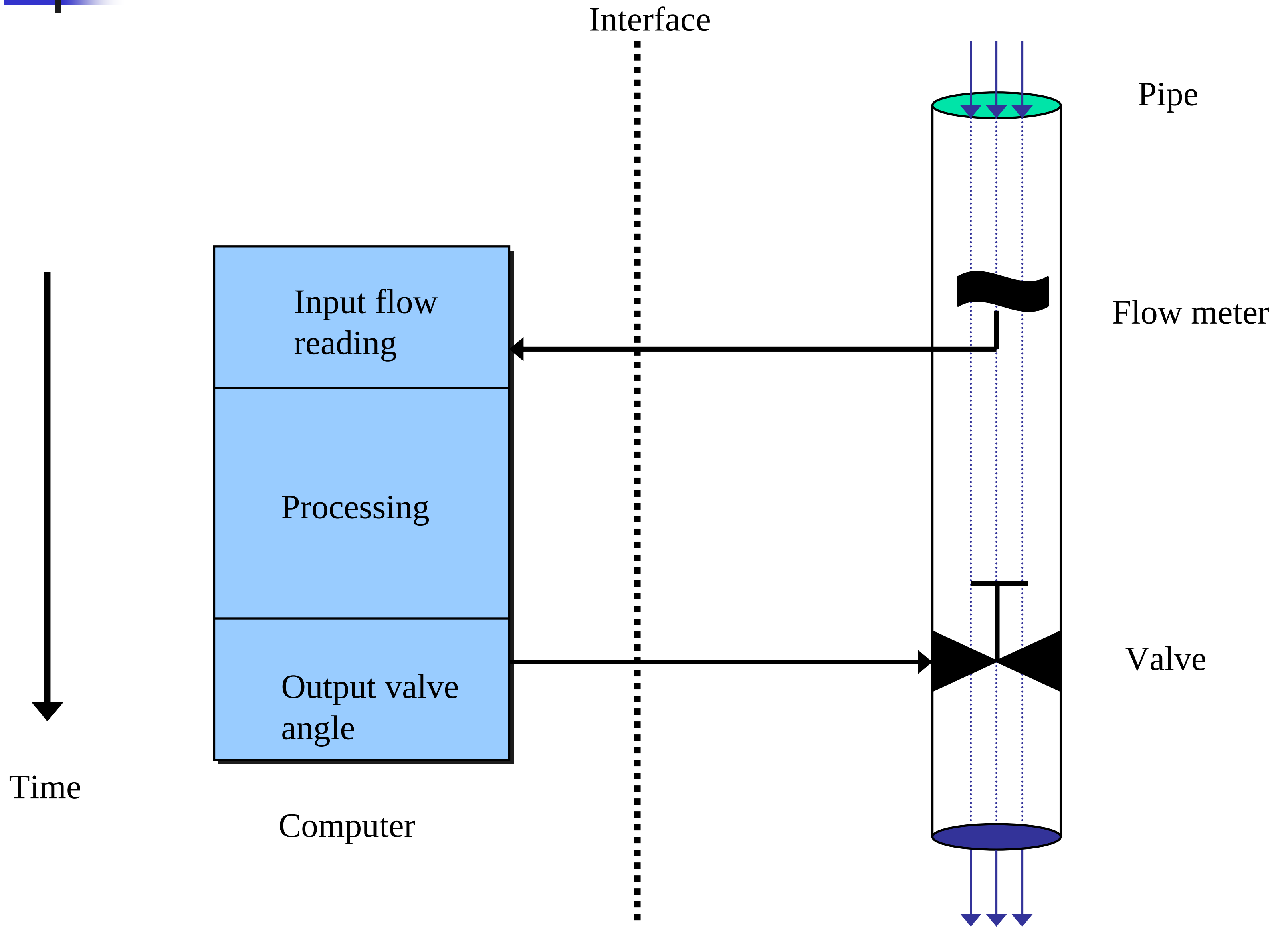
Terminology

- Time aware: explicit reference to time.
  - Real-time = wall clock.
- Reactive systems: must produce outputs as response of inputs.
  - Control systems.
- Jitter: delays, may be non-deterministic. Input/output jitter.
- Feedback loop: combine the outputs with the inputs to control the system – compensate jitters and other uncontrollable effect, auto-adjustment.

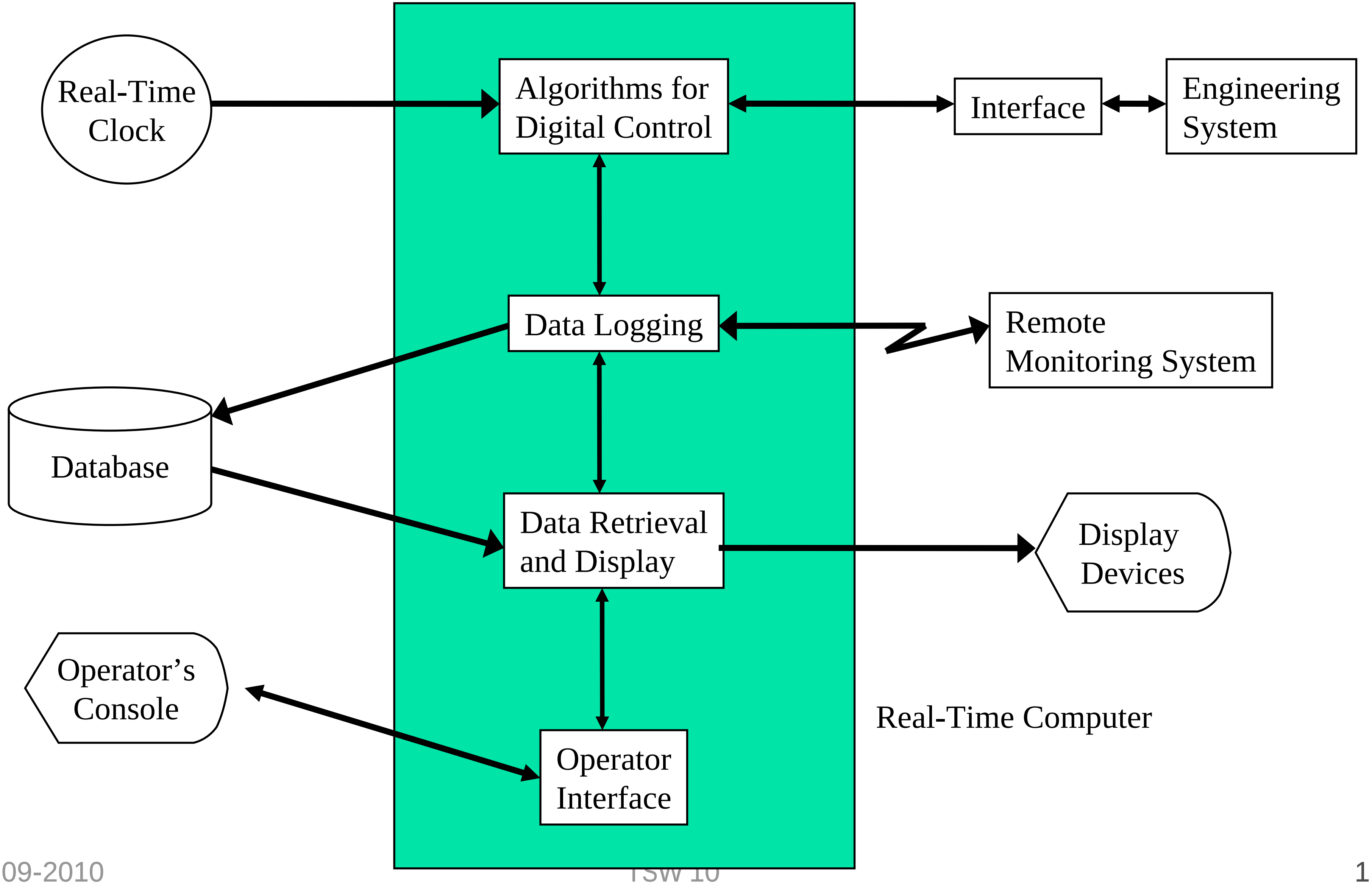
Terminology ✓

- Time-triggered: computations are triggered by passage of time.
  - Periodic activity: polling, USB 1 & 2.
- Event-triggered: computations are triggered by events.
  - Sporadic (occurrence bounded) or aperiodic (unbounded) activity: alarm, USB 3.

A simple fluid control system



A typical embedded system



Real-Time Clock

Algorithms for Digital Control

Interface

Engineering System

Data Logging

Remote Monitoring System

Database

Data Retrieval and Display

Display Devices

Operator's Console

Operator Interface

Real-Time Computer

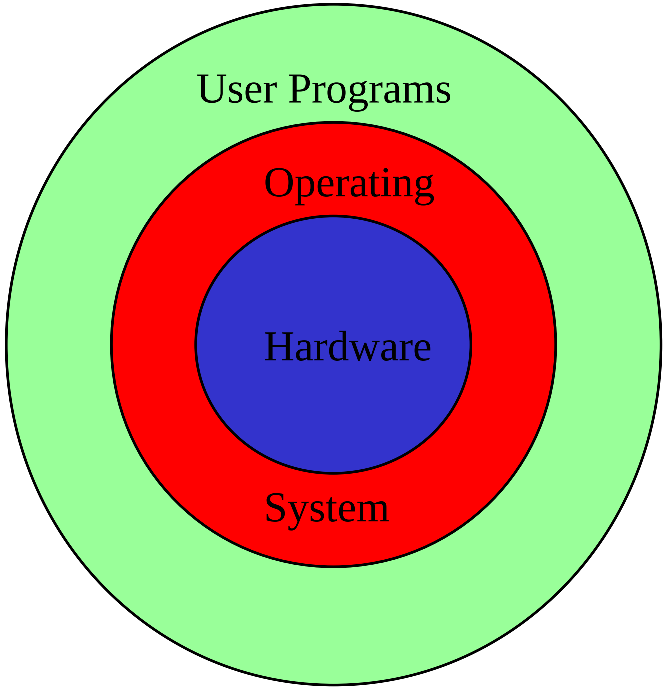
Characteristics of RTS

- Predictability: guarantee response times, worst-case response time analysis.
  - Predictability is more important than efficiency.
- Concurrency: control of real-world devices, several components operating in parallel.
- Interaction: sensors, actuators, special hardware → special programming needs.
- Digital: sample inputs (ADC), numerical computations, send outputs (DAC).
- Scale: large and small, few and numerous.
- Safety-critical: failure means loss of lives.

RT programming languages

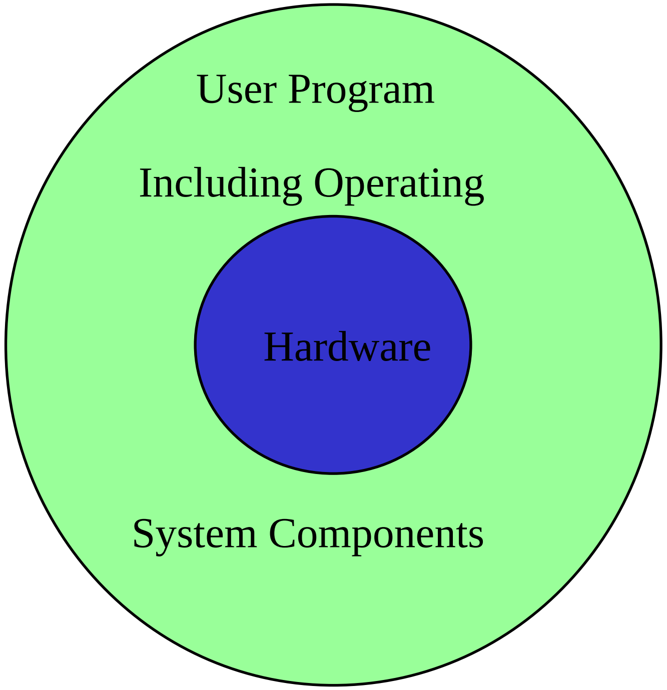
- Assembly languages
- Sequential systems implementation languages — e.g. RTL/2, Coral 66, Jovial, C.
- Both normally require operating system support.
- High-level concurrent languages. Impetus from the software crisis. e.g. Ada, Chill, Modula-2, Mesa, Java.
- No operating system support!
- We will consider:
  - Java/Real-Time Java
  - C and Real-Time POSIX (not in detail)
  - Ada 2005

Real-time languages and OSs



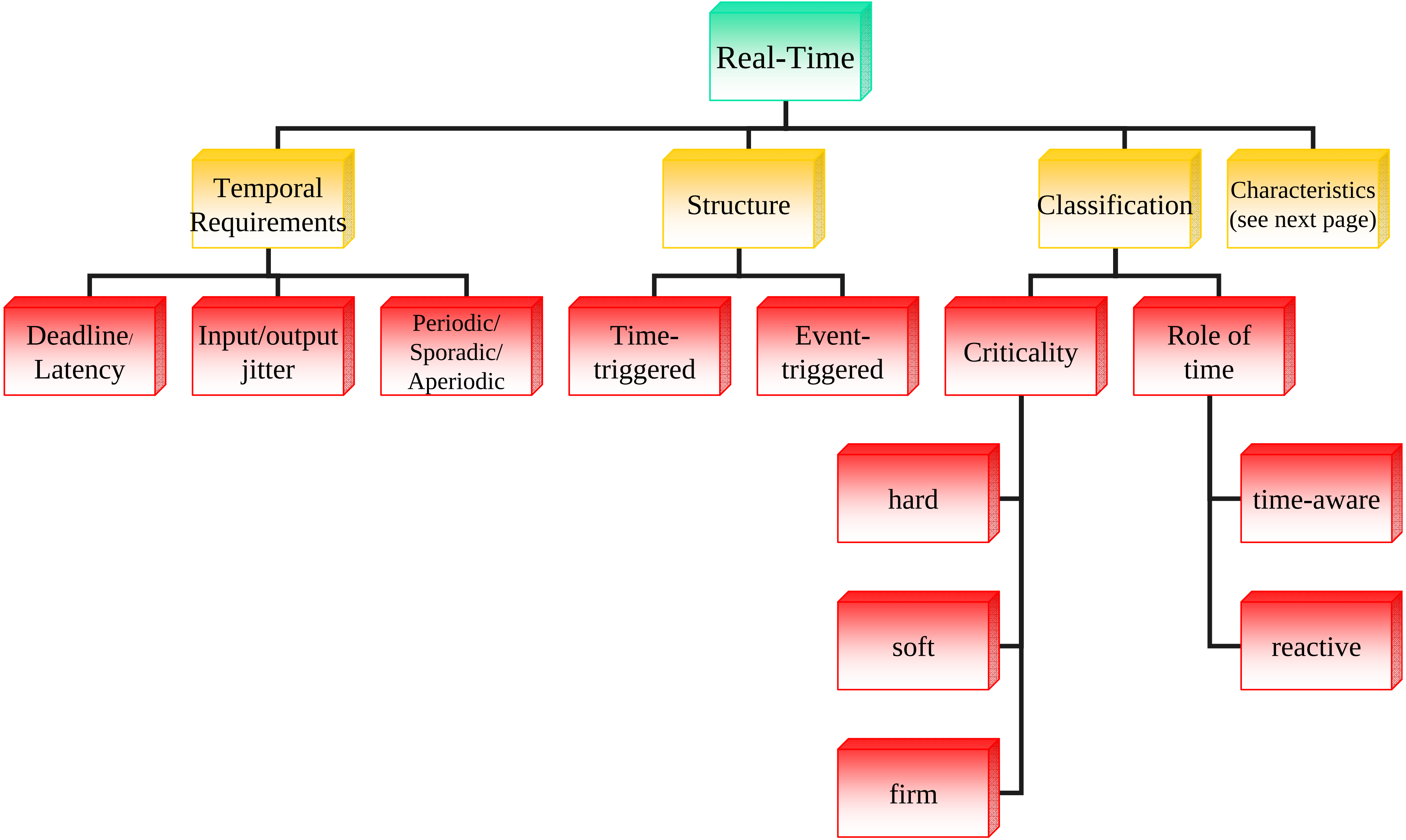
Typical OS Configuration

✓



Typical Embedded Configuration

Aspects of RTS



Real-Time

Temporal Requirements

Structure

Classification

Characteristics (see next page)

Deadline/ Latency

Input/output jitter

Periodic/ Sporadic/ Aperiodic

Time-triggered

Event-triggered

Criticality

Role of time

hard

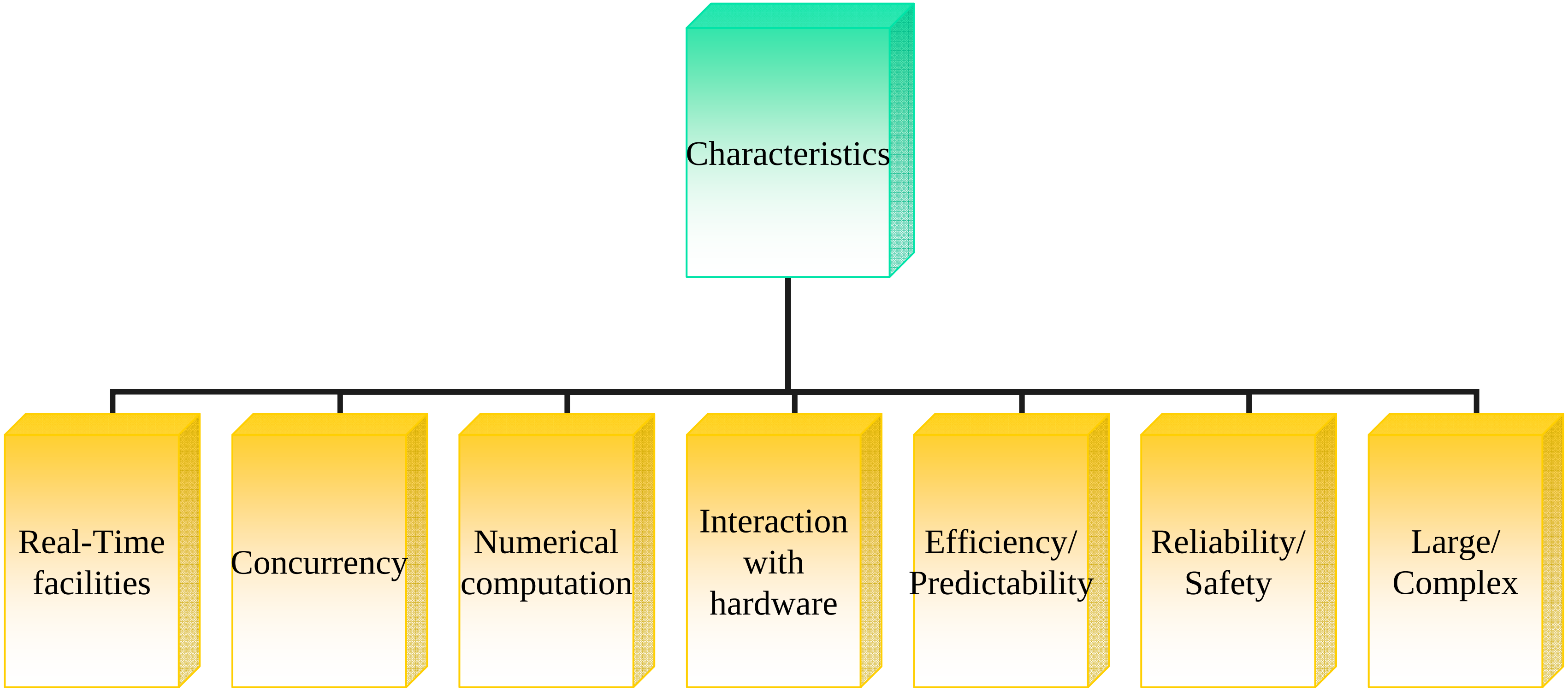
soft

firm

time-aware

reactive

Aspects of RTS


Characteristics
Real-Time facilities
Concurrency
Numerical computation
Interaction with hardware
Efficiency/ Predictability
Reliability/ Safety
Large/ Complex