REAL-TIME SYSTEMS
Introduction

Prof. J.-D. Decotignie
CSEM Centre Suisse d'Electronique et de Microtechnique SA
Jaquet-Droz 1, 2007 Neuchâtel
jean-dominique.decotignie@csem.ch

Outline

- Taxonomy of computer systems
- Definition
- Domain
- Realizations
- Real-time system features
- Architecture
- Models and properties

csem

Taxonomy of computer systems

- Transformational systems
- Reactive systems
  - Interactive systems
  - Real-time systems

few systems lie in a single category

csem

Transformational systems

- Have a limited operation duration
- Execution time is not critical
- Results are independent of the time at which they are produced
- Typical examples
  - Mathematica
  - Compilers
  - Salary management, etc.

csem

Interactive systems

- In constant dialog with one or more users
- They do nothing except under user request
- Response time does not really impact the result
- Typical examples
  - CAD systems
  - Minitel / Videotext / WWW browsers
  - Text processors / editors / spreadsheet editors

csem

Real-time systems

- Are connected to the external world (process)
- Must react in bounded time to external system solicitations
- Results (functions) depend on the time at which they are produced
- In general, they never stop operating
- Should not endanger users or process
- Typical examples: PSTN, CNC, robot control,

csem

Definitions

- Response time

  Interval between the instant at which one or more inputs are changed and the instant at which the matching response at outputs can be perceived

- Real-time system

  « A computer system for which correctness of calculations not only depends on the logical behavior but also on the instant at which the result is produced. If the temporal constraints are not fulfilled, it is said that a failure has occurred », real-time systems FAQ

csem

Definitions (2)

« hence, it is essential that the timing constraints of the system are guaranteed to be met. Guaranteeing timing behavior requires that the system be predictable. It is also desirable that the system attain a high degree of utilization while satisfying the timing constraints of the system », Real-time systems FAQ

csem

Hard vs soft real-time

- Hard real-time
  in case of violation of one or more temporal constraints, there is full loss of functionality
- Soft real-time
  the functionality is not lost but its quality is degraded
- Firm real-time

NIST Study of failures in PSTN

- CM = #customers x outage duration (min)

Sources	#failures	customers affected	Average duration	CM (millions)
Human mistakes	150 (49%)	165'996	255 min	4766 (28%)
HW	56 (19%)	95'690	159.8	1211 (7%)
SW	44 (14%)	118'200	119.3	356 (2%)
Act of nature	32 (11%)	159'000	828.2	3124 (18%)
Overloads	18 (6%)	276'760	1123.7	7527 (44%)
vandalism	3 (1%)	85'930	456	111 (1%)

NIST study conclusions

- Contrary to widely held beliefs
  - « an unexpected finding, given the complexity of the PSTN and its heavy reliance on SW, was that SW errors caused less downtime (2%) than any other source of failure except vandalism »
  - "HW and SW failures were similar in terms of average # of customer affected and outage duration"
  - "SW is not the weak link in the PSTN system's dependability"

NIST study conclusions (2)

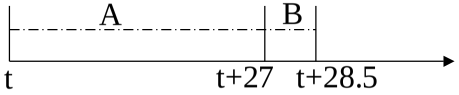
- « overloads caused nearly half of all down time in terms of customer minutes »
  - « overloads are accounted separately because they represents failures accepted as an engineering trade-off between dependability and cost »

Too many overloads = poor / faulty engineering decisions (dimensioning)

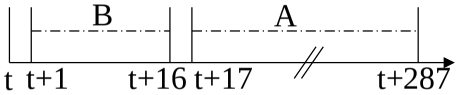
Too many overloads = poor or faulty schedulability analysis

Real-time is not equal to fast processing

- Hare
  - Power = 10
  - Policy: FIFO, no preemption
  - Switching and scheduling penalty = 0



- Tortoise
  - Power = 1
  - Policy: EDF, no preemption
  - Switching and scheduling penalty = 1



Real-time is not equal to fast processing

- High speed execution and fast context switch
- To obtain quick preemption, simple scheduling algorithms (or none) are often used

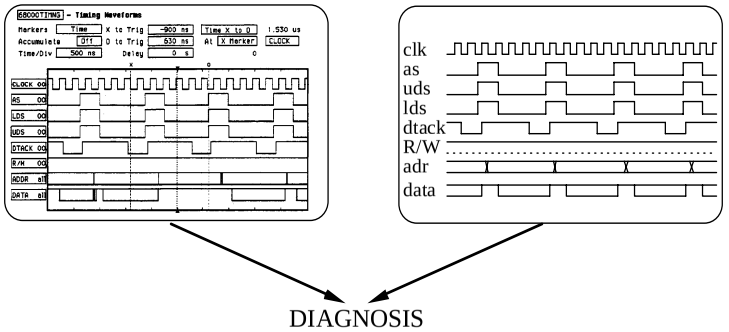
« It does not matter so much whether wrong scheduling decisions or, at best poor scheduling decisions, are made quickly or not », G. Le Lann (1990)

Use of real-time systems

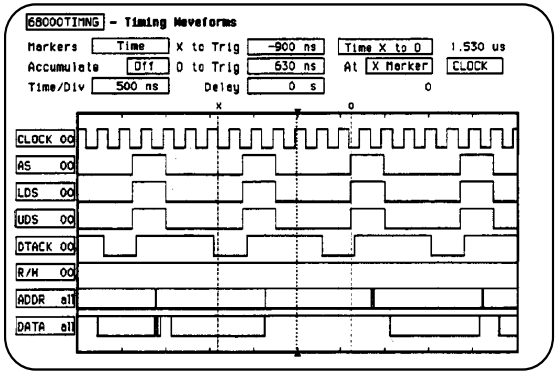
- Telecommunications / computer comm.
- Industrial process control
- Signal processing / multimedia
- Acquisition and measurement systems
- Embedded applications (transport, spatial, etc.)
- Appliances
- Military applications

Controlling a process

- The primary function of the lowest layers is to esta-blish a « diagnosis » by comparing the measurements with the process model. They will act accordingly.



Measurement



- $AS_i = AS(t_i)$ after measurement AS == seq of $\{AS_i\}$
- $AS = \{ev_i \in \{\uparrow,\downarrow\}; t_i \in \Re \mid (ev_i , t_i)\}$

csem

Physical time, logical time

- Logical time
  - Process only "watched" at pre-defined instants
  - Simultaneity and synchronization may be defined
  - Characteritic of sampled systems
- Physical time
  - The system follows continuously the evolution of the process in order to determine significant changes
  - Order and simultaneity based on time stamps

csem

Temporal notions

- Reaction time
  - Function of internal dynamics of external system
  - Production rate
- Time as seen from the control system
  - Logical time : Time-Triggered (TT) systems
  - Physical time: Event-Triggered (ET) systems

csem

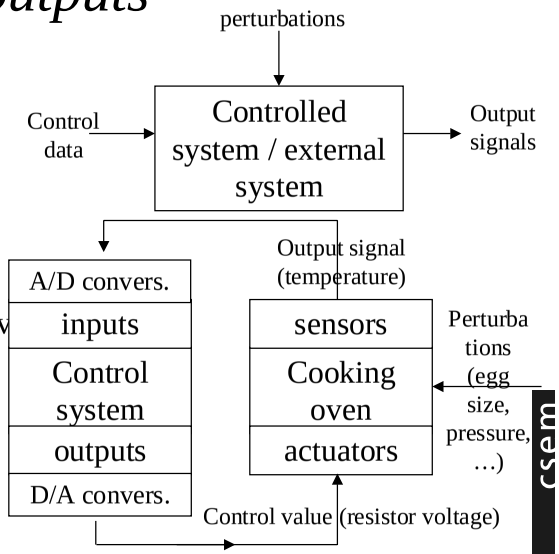
Real-time systems - Features

- Inputs and outputs to the external world
- Asynchronous vis à vis the control system
- More control than data
- Constraints (RAS, size ,consumption, environment)
- A number of parallel activities
- Behavior should be predictable
- Data consistency and validity duration
- Complexity and size

csem

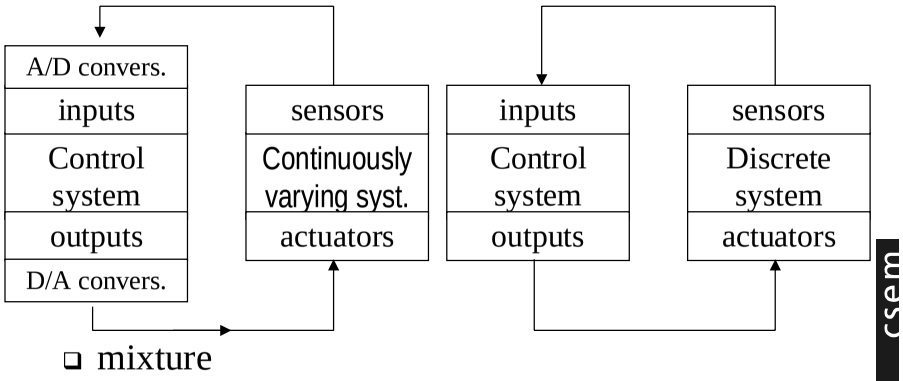
Inputs and outputs

- Controlled external element
- Characterized by a sequence of operations
- Input data (control v & perturbations)
- Output signals



Types of controlled systems

- Continuously varying
- discrete



- mixture

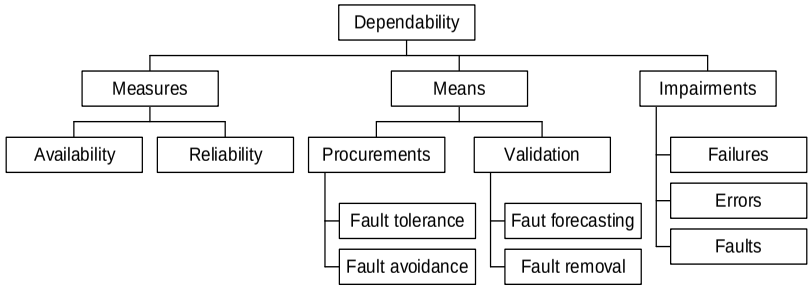
Input and output interfaces

- Purpose of interfaces
  - Analog to digital conversion and vice versa
  - Galvanic isolation
  - Conversions (current-voltage, frequency voltage, resistance-voltage, etc.)
  - Multiplexing
  - Signal conditioning (filtering, linearization, etc.)
  - Protection (EMI, ESD)

Reliability, Availability, Security

- Stop is disastrous
- Error in calculations not always severe
  $\Rightarrow$ few failures, degraded modes
  - Reliability = probability that a system fulfills the required function, under given conditions, during a given duration
  - Safety = ability to avoid given erroneous commands
  - Security = absence of circumstances that may cause human or material damages

Dependability



csem

Concurrent activities

- Task =
  a part of software that may be executed to fulfill a given function
- Process == task
- Does not imply (neither preclude) any manner to implement the task

csem

Task Taxonomy

- Continuous or permanent tasks
  $PM_i\left(T_i, C_i^{av}, C_i^{\min}, D_i = T_i, r_i = 0\right)$
- Cyclic tasks
  $TC_i\left(C_i, T_i^{av}, T_i^{\max}, D_i = T_i, r_i = 0\right)$
- Periodic tasks
  $P_i\left(T_i, C_i, D_i, r_i\right)$
- Sporadic tasks
  $S_i\left(T_i, C_i, D_i, r_i = 0\right)$
- Aperiodic tasks

csem

Predictability

- It is necessary to be able to determine in advance
  - the behavior when hypotheses are violated
  - the behavior when the temporal constraints are violated
  - the access time / response time
  - the behavior in case of incomplete information
  - how the result quality will vary

csem

Predictability - access/response time

- Optimization of processor usage
- Optimization of memory usage
- Worst case calculations
- Concurrent access

Predictability - incomplete or wrong information

- Environment is hardly predictable

- Liveness: « something good will eventually happen »
- Safety: « a bad thing will never happen »
- Fairness: « something will happen after another one »

Consistency

- Temporal consistency
  - A set of values available to a task corresponds to samples acquired at the same logical instant
  - Applies to sampled data systems
- Spatial consistency
  - Copies of the same sample on several applications (or tasks) are identical
  - Applies to both kinds (TT and ET) of systems

Validity duration - freshness

- information that is not used within a given delay is no longer useful to the control system
- Also called « absolute temporal consistency »

Validity duration is the delay after production during which the information is meaningful to its users

Freshness is a logical flag. If true, it tells that the validity duration has not elapsed.

- The same information may have different validity durations

Efficiency

- Time versus space compromise
- Not a programming language issue per se
- Power consumption is more and more an additional issue

csem

Environmental issues

- Electromagnetic and electrostatic interference
- Dust
- Liquids
- Vibrations
- Moisture
- Temperature
- Explosive or flammable environment (intrinsic safety)

csem

Size and complexity

- Size
- Complexity
  - Related to size, performances and heterogeneity
  - not easy to manage
  - Ex: ticket booking system
  - Solutions
    - ✓ mask
    - ✓ divide

csem

Architecture

- Centralized systems
- De-centralized systems
- Distributed systems

> « A computing system whose behavior is determined by algorithms explicitly designed to work with multiple loci of control », G. Le Lann (1990)

- Any combination of last two

csem

Models and properties

- Dominance
- Models
  - Structural, computational, data, task, event arrival, failure
- Properties
  - Safety, liveness, timeliness, dependability, complexity

Dominance

- Many classes may be hierarchically structured
- When a element E precedes element F in the hierarchy considered, this is noted as E $\supseteq$ F (E dominates F that is E is stronger than or equal to F)
- Examples
  - Arbitrary arrival $\supseteq$ periodic arrival
  - Timeliness $\supseteq$ liveness

Structural or architectural models

- Shared memory multiprocessors
- Meshed networks
- Broadcast bus
- Group of replicated processors

Computational models

- Synchronous
  - $\exists$ Upper bounds, values are known
- Partially synchronous
  - $\exists$ Upper bounds, values are unknown
- Asynchronous
  - Delays cannot be bounded

AM $\supseteq$ PSM $\supseteq$ SM

Task models

- Individual structure
  - Sequence (sq), star (st), tree (t), finite graph (fg)

$$fg \supseteq t \supseteq st \supseteq sq$$

- Individual attributes
  - Triggering (internal, external), shared persistent variables (external, internal), longest path of execution
- Collective attributes (Independent, causally dependent, chronologically dependant)

csem

Event arrival models

- Periodic (pr): period, arrival time known (concrete) or not (non-concrete)
- Sporadic (sr): sporadicity interval, always non concrete
- Aperiodic (apr): only once, concrete or not
- Arbitrary (uarr)
  - sliding time window of size w
  - a = highest number of arrivals within any window of size w

$$uarr \supseteq sr \supseteq pr,\ uarr \supseteq apr$$

csem

Failure models

- Time domain
  - Crash (cf): correct until halt
  - Omission (of): correct except correct steps not taken from time to time
  - Timing (tf): correct steps taken but too early / late
- Value domain
  - Stop (sf): correct values or crash
  - Byzantine (bf): any behavior in time and value

$$bf \supseteq tf \supseteq of \supseteq cf \supseteq sf$$

csem

Property classes

- Safety
- Liveness
- Timeliness
  - Latest termination deadline (ld)
  - Earliest / latest termination deadline, bounded jitter (bj)
  - Earliest start time (st)

$$st/bj \supseteq st/ld \supseteq bj \supseteq ld$$

csem

Real-Time Computing Techniques

- « about the best use of the machine »
  - Specification and design
    - ✓ Methods that the check the logical behavior and timing constraints compatibility
  - Architectural Design
    - ✓ Techniques to check the fulfillment of temporal constraints
    - ✓ Different architectural approaches
  - Implementation
    - ✓ Languages with adequate primitives
    - ✓ Run time support (from OS or Kernel) for these primitives

csem

Languages for real-time programming

- Capability to handle inputs and outputs
- Task management primitives
- Interrupt handling capability
- Good typing with access to bit level
- Capability to insert code instead of procedure call (INLINE)
- Primitives to handle time

csem

Course Objective

- Acquire basic knowledge in real-time computer engineering
- Be able to design the software of a real-time computing system

The scope will be limited to a single processor system

csem

Course Content

- Introduction
- Scheduling
- Performance evaluation
- Real-time system modeling
- GRAFCET and Petri Nets
- Implementation techniques
- Real-Time kernels

csem

General references

- A. Agrawala, S. Levi, "Real-Time Systems Design", Mc Graw Hill, ISBN 0-07-100797-0, 1990.
- G. Buttazzo, « Hard Real-Time Computing Systems », Kluwer, Boston, ISBN 0-7923-9994-3, 1997.
- R. David, H. Alla, "Petri Nets and Grafcet", Prentice Hall, ISBN 013327537X, 1993.
- D. Gajski, "Specification and Design of Embedded Systems", Prentice Hall; ISBN: 0131507311, 1994
- W. Halang, K. Sasha, "Real-Time Systems - Implementation of Industrial Computerised Process Automation". World Scientific, 1992
- W. Halang, D. Stoyenko, "Constructing predictable real-time systems", Kluwer, Boston, ISBN 0-7923-9202-7, 1991.
- M. Klein et al. , "A practioner's handbook for real-time analysis", Kluwer, Boston, ISBN 0-7923-9361-9, 1993.
- H. Kopetz, "Real-Time Systems : Design Principles for Distributed Embedded Applications", Kluwer Academic Publishers; ISBN: 0792398947, 1997
- P. Laplante, "Real-Time Systems Design & Analysis: An Engineer's Handbook", IEEE, ISBN 0780334000, 1997
- S. Mullender, "Distributed Systems", ACM Press, 1989
- S. Son, "Advances in real-time systems", Prentice Hall, ISBN 0-13-083348-7, 1995
- J. Stankovic, K. Ramamritham, "Hard Real-Time Systems", IEEE Computer Soc. Press, ISBN 0-8186-0819-6, 1988
- A. Wellings, A. Burns, "Real-Time Systems and Their Programming Languages", Addison Wesley, ISBN: 020140365X„1996

csem

