Real-time systems
“Introduction to real-time systems”

Mathieu Delalandre
University of Tours, Tours city, France
mathieu.delalandre@univ-tours.fr

Lecture available at http://mathieu.delalandre.free.fr/teachings/realtime.html

Introduction to real-time systems

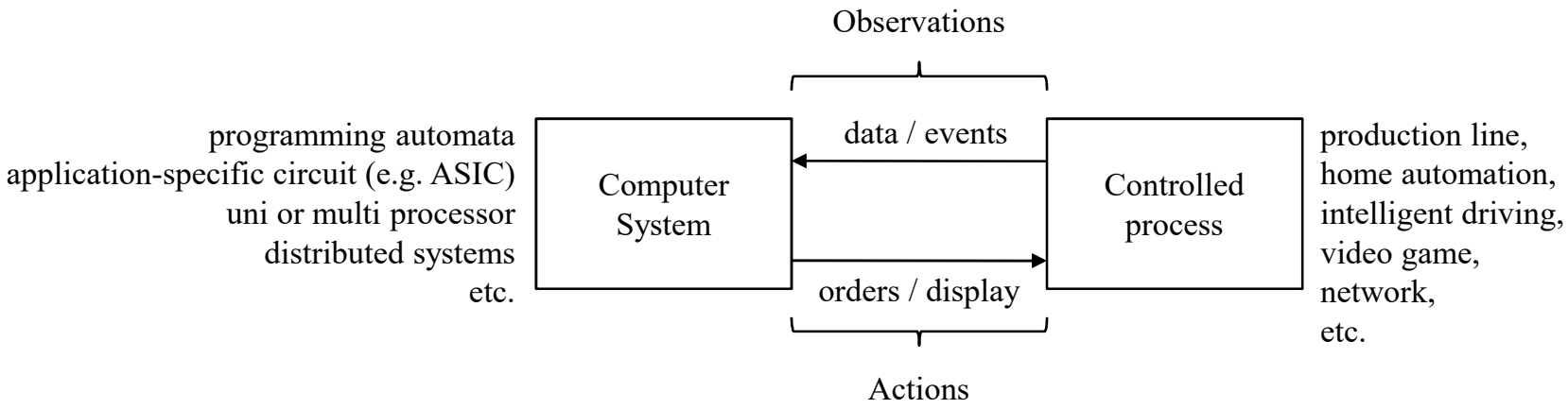
1. Definition
2. Application use-case
3. Concept of time
4. Design issues

Definition

Real-time systems are hardware and software components that are subject to real-time constraints (i.e. operational deadlines from events to system responses). They must execute within strict constraints on response time. By contrast, a non-real-time system is one for which there is no deadline. They can be defined considering two aspects:

- ✓ **Time** means that the correctness of the system depends not only on the logical results of the computation but also on the time at which the results are produced.

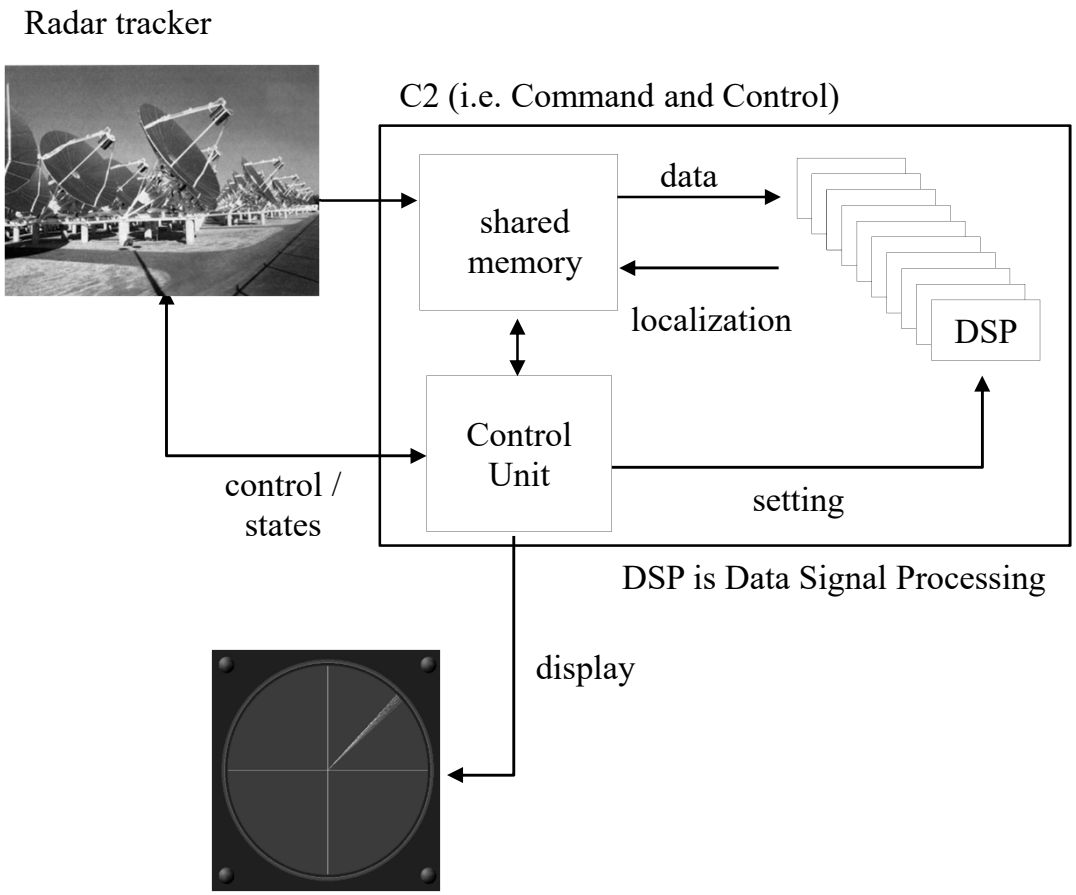
- ✓ **Real** indicates that the reaction of the system to external events must occur during their evolution. As a consequence, the system time must be measured using the same time scale used in the controlled environment.



Introduction to real-time systems

1. Definition
2. Application use-case
3. Concept of time
4. Design issues

Application use-case



Tracking steps in C2.

1. **Track initiation:** to create a new radar track from a radar plot.
2. **Track smoothing:** the latest track prediction is combined with the associated plot to provide a new, improved estimate of the target state. A basic algorithm is the Kalman filter.
3. **Track maintenance:** to end the life of a track.

Other applications are at the corner:

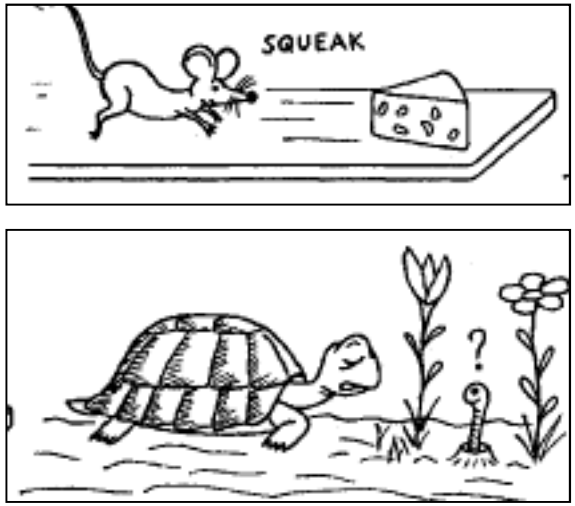
chemical and nuclear plant control,
railway,
flight control systems,
telecommunication systems,
industrial automation,
robotics,
military systems,
etc.

Introduction to real-time systems

1. Definition
2. Application use-case
3. Concept of time
4. Design issues

Concept of time (1)

Time and environment: a generally held opinion defines a system as working in real-time, if it is able to quickly react. The concept of time is not an intrinsic property of the system, but strictly related to the environment in which the system operates.



Application	Time scale
Aircraft	millisecond
Machine Man Interface	second
Production line	minute
Chemical reaction	hour
Meteorological prediction	day
Pay slip	month

Concept of time (2)

Fast vs. predictability: some people erroneously believe that it is worth investing in real-time because advances in computer hardware will take care of any real-time requirements (i.e. the Moore law). In fact, the objective of fast computing is to minimize the average response time of a given set of tasks, whereas the objective of real-time computing is to meet the individual timing requirement of each task. Hence, rather than being fast, a real-time computing system should be predictable.

Soft vs. hard real-time: the main difference between a real-time and a non real-time system is that a task in a real-time system is characterized by a deadline, which is the maximum time within which it must complete its execution. Therefore, in a real-time system, a result provided after a deadline is not only in late but wrong. Depending of the consequence of a missed deadline, the real-time systems could be considered in two categories:

Type	**Description**	**Examples**
Hard real-time systems	A missing deadline may cause catastrophic consequences on the environment under control.	Aircraft command, nuclear process control, intelligent transportation, medical equipment, etc.
Soft real-time systems	A meat deadline is desirable for performance reasons, but a missed deadline does not cause serious damages to the environment and does not jeopardize correct system behavior.	Handling input data from keyboard, display message on screen, handling input from pad for video game, data format conversion, etc.

Introduction to real-time systems

1. Definition
2. Application use-case
3. Concept of time
4. Design issues

Design issues (1)

There are important properties that real-time systems must have to support for critical applications.

	System	
Features	**no real-time**	**real-time**
Scalability	++	+
Maintainability	+	+
Fault tolerance	+	++
Design for peak load	+	++
Timeliness	no	yes
Predictability	no	yes

Scalability: a system is described as scalable if will remain effective when there is a significant increase in the number of resources and users.

Maintainability: the architecture of a system should be designed according to a modular structure to ensure that possible system modifications are easy to perform.

Fault tolerance: single hardware and software failures should not cause the system to crash. Therefore, critical components of the system have to be designed to be fault tolerant.

Design for peak load: systems must not collapse when they are subject to peak-load conditions, so they must be designed to manage all anticipated scenarios.

Timeliness: results have to be correct not only in their values but also in the time domain. As a consequence, the system must provide specific kernel mechanisms for time management and for handling tasks with explicit time constraints and different criticality.

Predictability: to guarantee a minimum level of performance, the system must be able to predict the consequences of any processing decision. If some tasks cannot be guaranteed within time constraints, the system notify this fact in advance so that alternative actions can be planned.

Design issues (2)

Ad hoc vs. design methodologies: most of the real-time systems are still designed with ad hoc techniques (e.g. large portions of code in assembly language, programming timers, handling tasks and interrupt priorities). Although the code produced by these techniques can be optimized, this approach has the following disadvantages:

- ✓tedious programming,
- ✓difficult code understanding,
- ✓difficult software maintainability,
- ✓difficult verification of time constraint,
- ✓etc.

e.g. Patriot system, 25th of february 1991 (Dhahran city, Saudi Arabia)

During the golf war, a scud has been missed by the patriot system, and crashed on the Dhahran city (Saudi Arabia). A bug on real-time clock management was accumulating a delay of 57 μs / mm. The day of the accident, a 100 hours execution has accumulated 343 ms of delay corresponding to a distance approximation error of 687 meters. The bug was corrected the day after (26th of February).

This is mainly due to the fact, in real-time control applications, the program flow depends of input sensory data and environmental conditions, which cannot be fully replicated during the testing phase.

Design issues (3)

Ad hoc vs. design methodologies: a more robust guaranty of the real-time system performances under all possible operating conditions can be achieved only by using more sophisticated design methodologies (operating system mechanisms, static analysis of the source code, etc.).

Design Levels	**No real-time features**		**How to make it working in real-time systems**
	Components	**Problems**	
Hardware	Direct memory access	Bus access restriction	disable optimization & interrupt, device handler rewriting
	Cache memory	Random reading access	
	Interrupt	Processor resource restriction	
	CPU Optimization	Carry-look ahead, booth encoding, etc.	
	Etc.		
Operating system	Scheduling	No real time scheduling	Design of real-time systems (VxWorks, RT Linux, etc.)
	IPC and synchronization	Priority inversion	
	System call	Processor resource restriction	
	Memory management	Expended memory	
	Etc.		
Application	Programming language	Garbage collector, recursive programming, etc …	Real-time programming language (C/C++, RUST, Ass…)