- Hardware has evolved ⇒ Applications have evolved.

What are the differences between applications from the past and current applications?

Which requirements became important when real-time systems were considered?

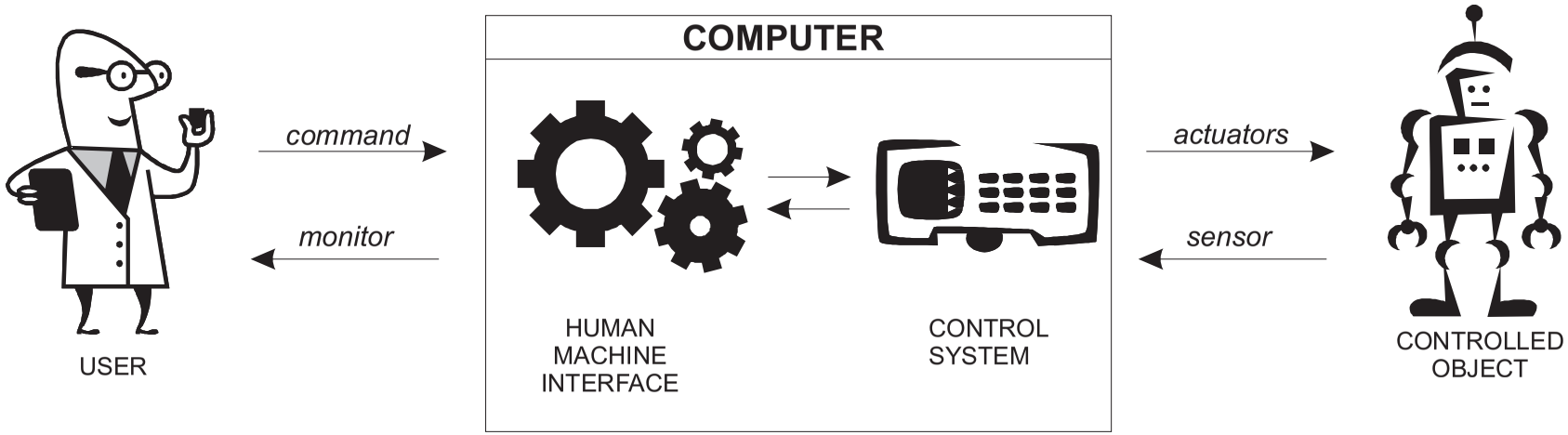
- Traditional computing systems need to be logically correct.
- Others, need to deal with logic and temporal requirements.

Definition

*In **real-time computing** the correctness of the system depends not only on the logical results but also on the time at which such results are produced.*

- Chemical nuclear plan control
- Control of complex production processes
- Railway switching systems
- Automotive applications
- Flight control systems
- Robotics
- Multimedia Systems
- Smart Toys
- Consumer Electronic Devices
- Virtual Reality

General Scheme of a Real-Time System



Is real-time really important?

Real-time computing plays an important role in society since an increasing number of of complex systems partially rely on computer control

- There is no science in real-time system design: most real-time design is ad-hoc, which does not mean that scietific approach is not possible.
- Advances in supercomputer hardware will take care of real-time requirements: this improves system throughput, but does not mean meeting timing requirements;
- Real-time computing is equivalent to fast computing: fast computing aims at minimizing average response time. Real-time computing focus on meeting individual timing requirements of each task;

- The problems in real-time-system design have all been solved in other areas of computer science or operations research: there are especific problems in real-time which has not been solved yet (as for example assessing if stringent deadlines can be met);
- It is not meaningful to talk about real-time performance because we cannot guarantee that hardware will not fail and the software is bug free: the relevant question is how to build systems in such a way that we can have as much confidence as possible that they will meet specifications at acceptable costs.

- Not all real-time applications have the same time requirements

- Not all real-time applications have the same time requirements
- **Hard** Real-Time Systems: a system is said to be *hard* when producing results **after** the deadline, may cause "catastrophic" consequences

- Not all real-time applications have the same time requirements
- **Hard** Real-Time Systems: a system is said to be *hard* when producing results **after** the deadline, may cause "catastrophic" consequences
- **Firm**: a real-time is said to be *firm* when producing results **after** the deadline is useless for the system, but does not cause any damage.

- Not all real-time applications have the same time requirements
- **Hard** Real-Time Systems: a system is said to be *hard* when producing results **after** the deadline, may cause "catastrophic" consequences
- **Firm**: a real-time is said to be *firm* when producing results **after** the deadline is useless for the system, but does not cause any damage.
- **Soft**: a real-time is said to be *soft* when producing results **after** still have some utility for the system, although causing performance degradation.

- Timeliness: results have also to be correct in time domain (besides logical domain)
- Predictability: consequences of scheduling decisions must be predictable
- Efficiency: related to the efficiency in managing available resources, specially in embedded devices (space, weight, energy, memory and computational power)
- Robustness: systems must be able to support eventual overloads
- Fault Tolerance: single failures must not cause system crashes.

Ubiquitous Real-Time Systems

"Softwares in mobile and ubiquitous systems has both traditional features of real-time embedded systems and also contemporary features"

- Adaptive resource management
- Proactive service discovery
- Context-aware coordination
- Multi-agents
- Models for heterogeneous platforms
- Interoperability
- Mobility
- Survivability
- Security, adaptability,
- Ability of self-organization

Predictability Definition

Definition

"(...) the system should be able to predict the evolution of tasks and guarantee in advance that all critical timing constraints will be met."

What does affect predictability?

- components failure or malfunctioning
- **processor scheduling**
- synchronization mechanism
- memory management policy
- ... other internal characteristics of real-time kernel