

# Component-based Design

## EU-US Workshop

on Future Challenges in Embedded Systems Design

July 8th, 2005 - Paris

Joseph Sifakis

VERIMAG & ARTIST2 NoE

# Component-based engineering - Motivation

Building complex systems from simpler ones is universally the basis for any system theory and practice.

- Raises hard problems about concepts, languages and their semantics e.g. What is an architecture? What is a scheduler? How synchronous and asynchronous systems are related?
- Requires a deep understanding of basic system design issues such as development methodologies (combination of techniques and tools, refinement) and design principles

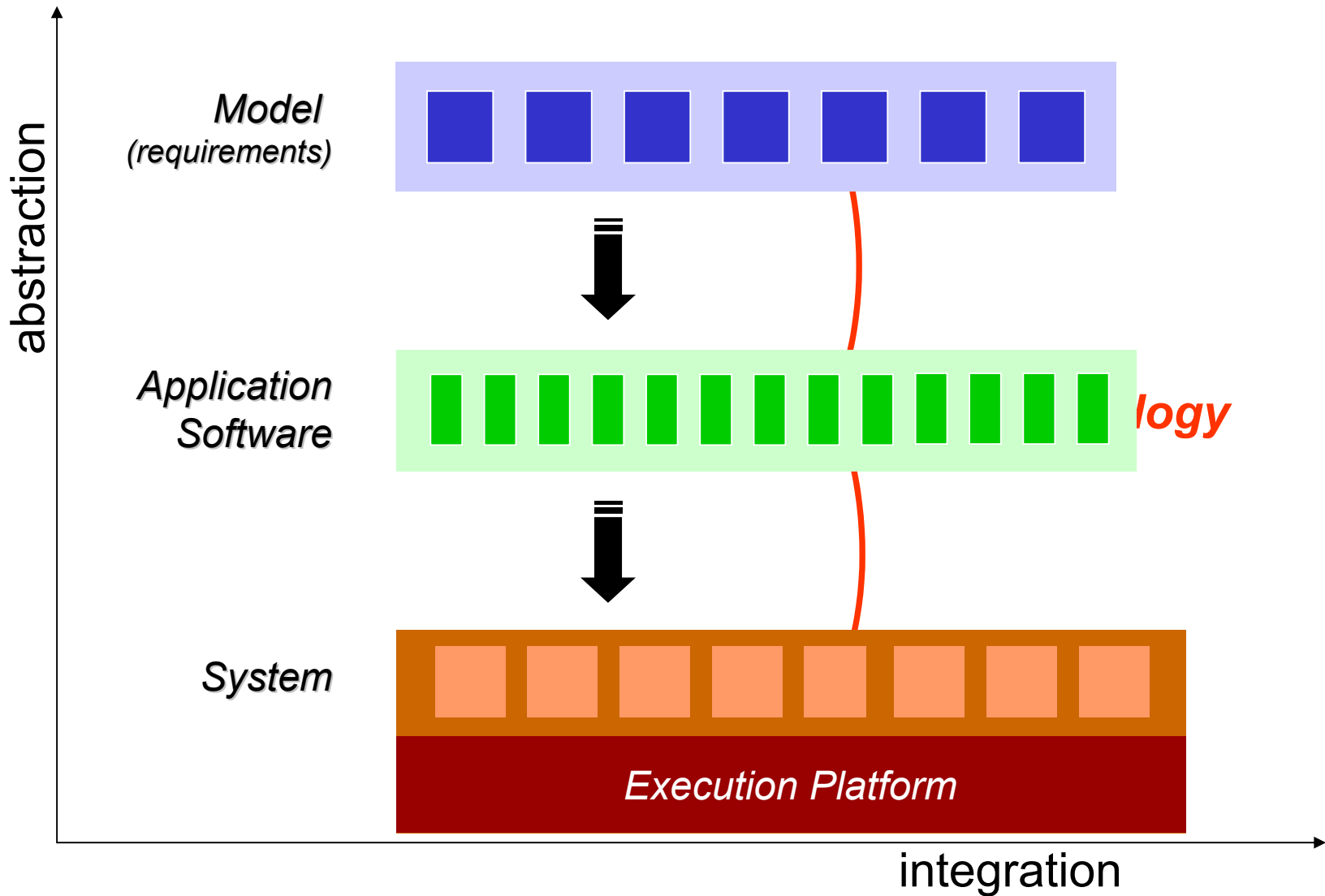
*It's not just playing with syntax and graphical tools ....*

# Component-based engineering - Related stuff

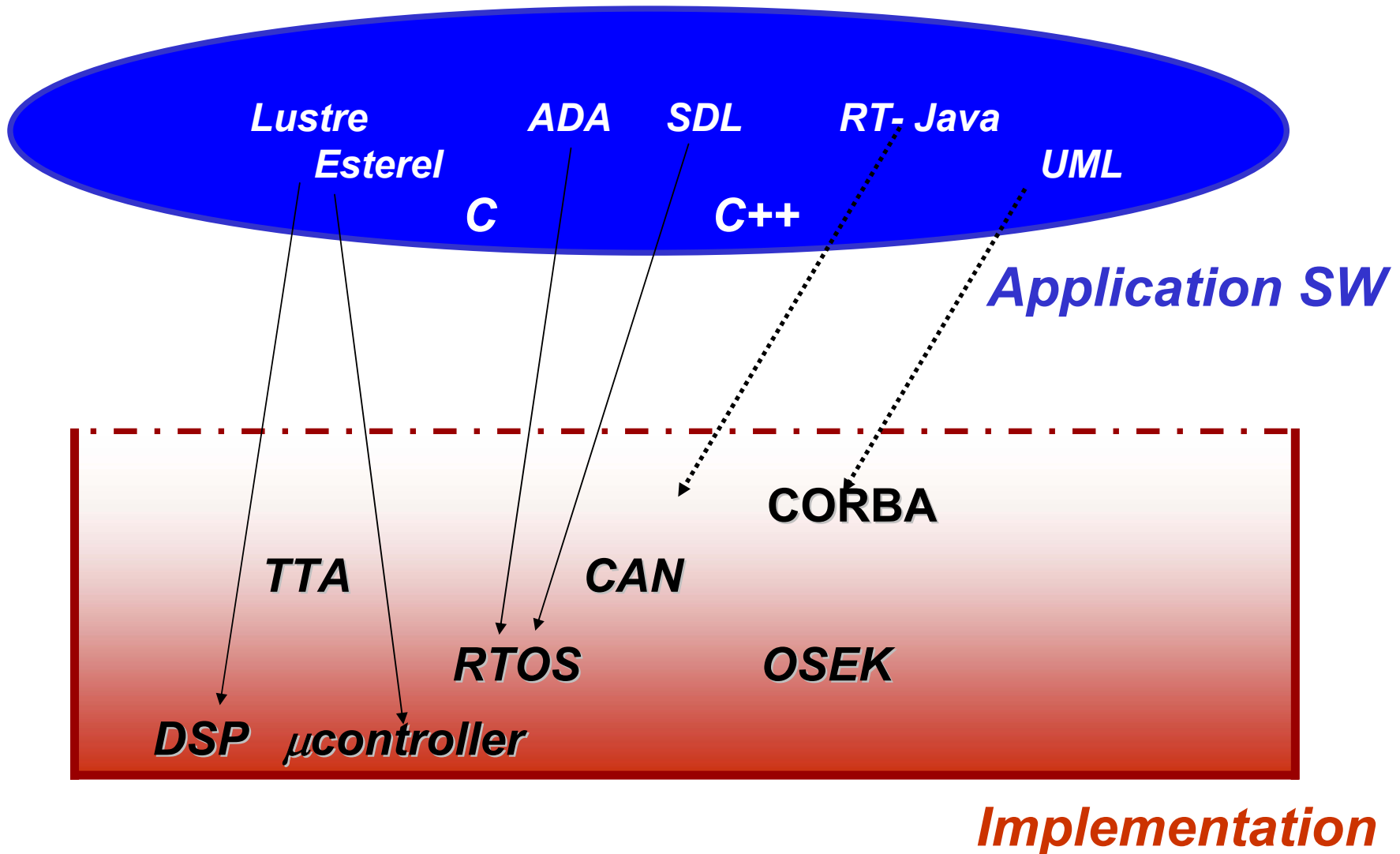
The following deal with issues related to component-based engineering:

- Architecture Description Languages focusing on non-functional aspects or SW Design Description Languages
- Modeling languages: Statecharts, UML, Simulink/Stateflow, Synchronous languages, SystemC, Metropolis, Ptolemy
- Coordination languages (extensions of programming languages): Linda, Javaspace, TSpaces, Concurrent Fortran, ...
- Middleware standards: IDL, Corba, Javabeans, .NET
- Software development environments: PCTE, SWbuses, Softbench, Eclipse
- Process algebras and automata : Pi-Calculus, I/O automata

# System Design – Abstraction Levels



# System Design: From application SW to implementations



# System Design: From application SW to implementations

*Logical abstract time*

*High level structuring constructs and primitives*

*Simplifying synchrony assumptions wrt environment*

***Application SW***



*Physical, Non functional properties*

*Execution times, interaction delays, latency*

*Task coordination, resource management, scheduling*

***Implementation***

# System Design: From Application SW to Implementations

*synchronous vs. asynchronous*

*Application SW*

**Synchronous**  
**Lustre, Esterel**  
**Statecharts**

**Asynchronous**  
**ADA, SDL**

**Adaptive**  
**approaches**

- **Non interruptible execution steps**
- **Usually, single task, single processor**

• **«Everybody gets something »**

- **Event triggered**
- **Multi-tasking - RTOS**
- **Usually, static Priorities**

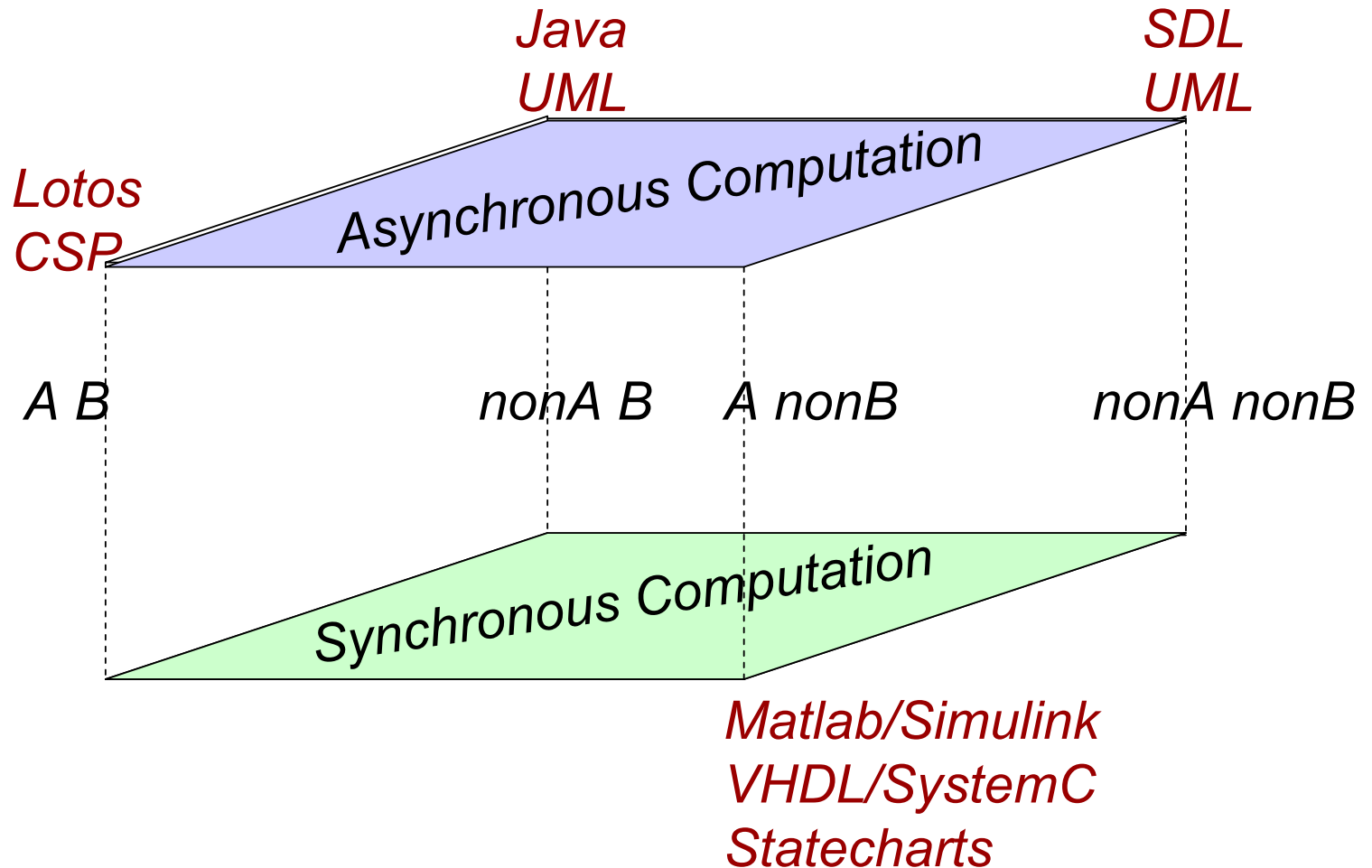
• **«Winner takes all »**

**Implementation**

# Heterogeneity

*A: Atomic interaction*

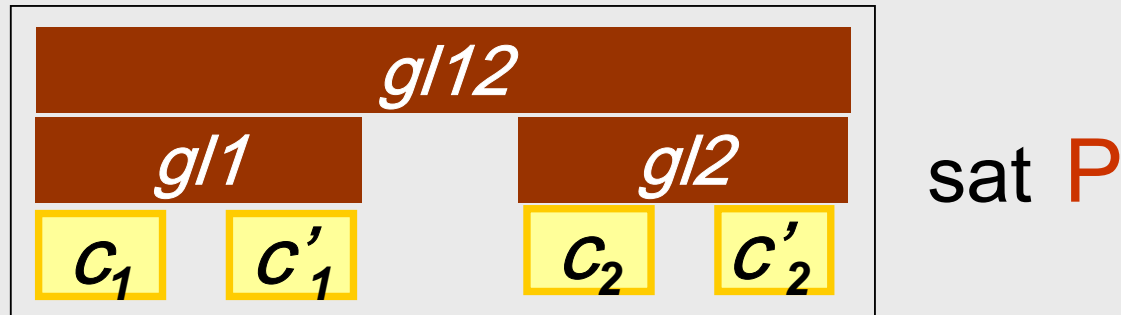
*B: Blocking interaction*



## Component-based design

Build a component **C** meeting a given property **P** from

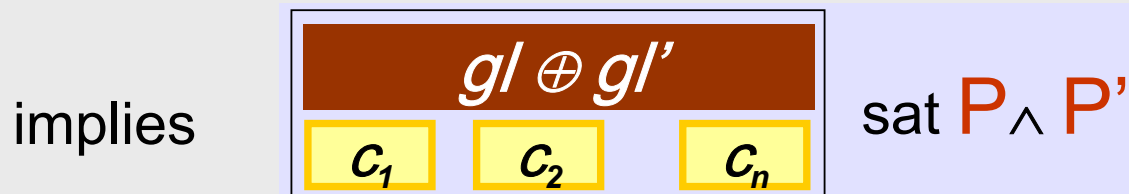
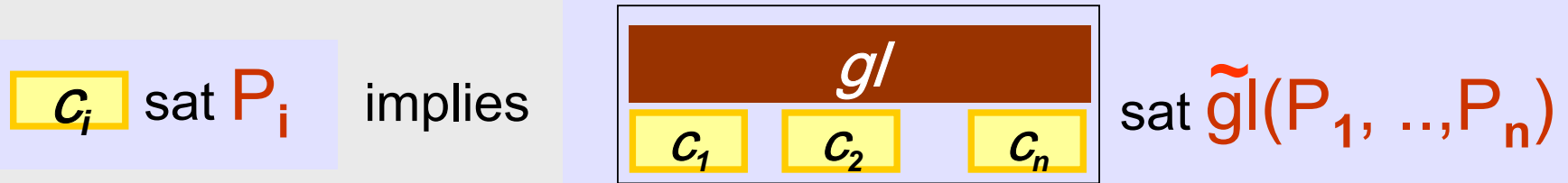
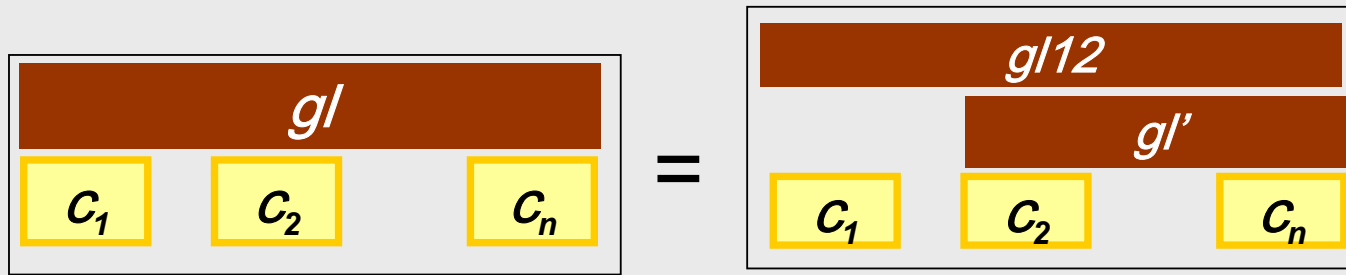
- $C_0$  a set of atomic components
- $GL$  a set of operators on components



Glue can be any mechanism used for communication and control such as protocols, controllers, HW architectures.

*Problem: Find a «minimal» set of operators with rules for component-based construction*

# Component-based design – Requirements



# Architectures

Provide a rigorous and general basis for architecture modeling, design and implementation encompassing

- A general concept of architecture as a means to organize computation (behavior, interaction, control)
- Heterogeneity and specific styles and paradigms, e.g.
  - synchronous and asynchronous execution
  - heterogeneous interaction (strong, weak, event-driven, state-driven)
  - architecture styles e.g., client-server, blackboard architecture
- Correctness-by-construction results for generic properties such as deadlock-freedom, liveness, safety.
- Automated support for component integration and generation of glue code meeting given requirements

# Adaptive Systems

- Adaptivity is the capacity of a system to meet given requirements including safety, security, and performance, in the presence of uncertainty in its external or execution environment.

*It is a means for enforcing predictability in the presence of uncertainty*

- Uncertainty is characterised as the difference between average and worst-case behavior of a system's environment. The trend is towards drastically increasing uncertainty, due to:
  - Connectivity with complex, non-deterministic, possibly hostile external environments
  - Execution platforms with sophisticated HW/SW architectures (layering, caches, speculative execution, ...)

# Adaptive systems

- The increase in uncertainty gives rise to 2 diverging approaches and technologies:
  - **Critical systems engineering** based on worst-case analysis and static resource reservation e.g. hard real-time approaches, massive redundancy.
  - **Best effort engineering** based on average case analysis e.g., soft real-time for optimization of speed, memory, bandwidth, power,
- This leads to a physical separation between critical and non critical parts of a system running on dedicated physical units, which implies increasing costs and reduced hardware reliability, e.g.: an increasing numbers of ECUs in automotive systems.
- *It is essential to develop holistic adaptive design techniques combining the advantages of the two approaches: guaranteed satisfaction of critical properties and efficiency by making best possible use of available resources (processor, memory, power).*

# Adaptive systems

