Embedded Systems

Research Challenges and Work Directions

40-ième Anniversaire du LAAS

28 mai 2008

Joseph Sifakis
VERIMAG & ARTIST2 NoE

Embedded Systems: Scope

An Embedded System integrates **software and hardware** jointly and specifically designed to provide given functionalities, which are often **critical**.

Embedded Systems: Economic Stakes

Embedded Systems are of strategic economic importance

- ❖ Factor for innovation and differentiation:
  - ➢ new functionalities and services in existing products
  - ➢ new products and services
- ❖ Principal source of added value: particularly for embedded software
- ❖ Increased competitivity
- ❖ This is the fastest-growing sector in IT

Europe has leading positions in sectors where embedded technologies are central to growth

- ❖ Currently: Industry (avionics, automotive, space, consumer electronics, telecom devices, energy distribution, rail transport, …)
- ❖ Anticipated: Services (e-Health, e-Education)

Embedded Technologies are of strategic importance

for modern economies

Embedded Systems: Trends

- An exploding number of embedded reactive heterogeneous components in mass-market products

- Massive seamless integration of heterogeneous components in a real-world environment (conflicts/competition, confidentiality, responsibility)

- Technical and Economic Constraints
  - Dependability (safety, security, availability)
  - Autonomy (no humans in the loop)
  - Low resource consumption (memory, power, energy)
  - Physical constraints (weight, size, heat dissipation, …)
  - Market positioning (optimal cost/quality, time to market)

Building systems of <u>guaranteed functionality and quality</u>,

at an acceptable cost,

is a major technological and scientific challenge.

Embedded Systems: The State of the Art

Today, we master – at a high cost :

- Critical control systems

  ↳ ***Automated aircraft landing systems***

  High reactivity + High Dependability

- Complex “best effort” systems

  ↳ ***mobile telecommunications***

  Distribution + Good reactivity + Good dependability

Tomorrow, the vision we’re aiming for are

Distributed, Heterogeneous **Systems of Systems**

- ***Automated freeways***
- ***Next generation air traffic control***
- ***« Ambient Intelligence »***

Air Traffic Control – the Next Generation
Is it … attainable ?

1984

Start of the project

April 25th, 1994

Air traffic takes another turn

FAA weighs fixes for automation project

By Gary H. Anthes - April 25th, 1994

The Federal Aviation Administration is considering major changes in its troubled Advanced Automation System (AAS) project, now billions of dollars over budget and years behind schedule.

FAA administrator David R. Hinson told a congressional panel that the agency might **scrap the project entirely**, although he said some scaling back was more likely. The FAA has spent about $1.5 billion to date on the estimated $6.9 billion air traffic control system.

April 9th, 2002

The FAA's Course Correction

The Ugly History of Tool Development at the FAA

By David Carr and Edward Cone April 9th, 2002

Online exclusive: The agency wrote off $1.5 billion of its $2.6 billion investment to overhaul the nation's air traffic control computer systems. What went wrong? (Just about everything.)

One participant says, "**It may have been the greatest failure in the history of organized work.**"

Certainly the Federal Aviation Administration's Advanced Automation System (AAS) project dwarfs even the largest corporate information technology fiascoes in terms of dollars wasted. Kmart's $130 million write-off last year on its supply chain systems is chump change compared to the AAS. **The FAA ultimately declared that $1.5 billion worth of hardware and software out of the $2.6 billion spent was useless.**

The Challenges

Technological Challenge:

Building systems of
guaranteed functionality and quality
(performance and robustness),
at acceptable costs.

This **Technological Challenge**
hides an underlying **Scientific Challenge**

Scientific Challenge:

The emergence of Embedded Systems
*as a **scientific and engineering discipline***
*enabling **system design predictability**,*
as is already the case for the physical sciences.

Proposed Vision

By their nature, Embedded Systems need results and paradigms from both
Computing Systems and Physical Systems Engineering

We need a new formal foundation for Embedded Systems, which systematically and even-handedly marries
computation and physicality
performance and robustness

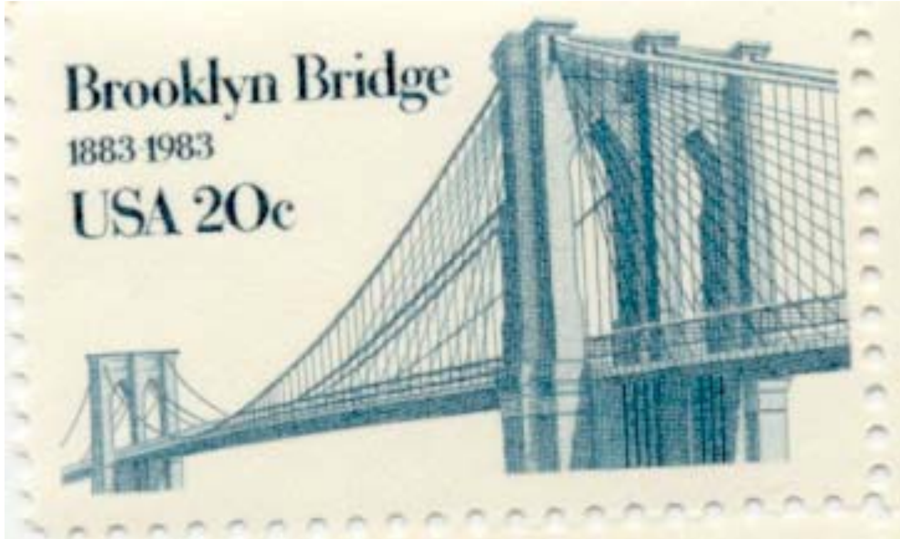
What is being computed? At what cost?

How does the performance change under disturbances? (change of context; change of resources; failures; attacks)

The Challenges

Physical Systems Engineering

Computing Systems Engineering



Uptime: 125 years

The Challenges

Physical Systems Engineering – Analytical Models

Differential Equations
Linear Algebra
Probability Theory

Synthesis

Theories of estimation
Theories of robustness

Mature

Computing Systems Engineering – Computational Models

Logic
Discrete Structures
Automata Theory

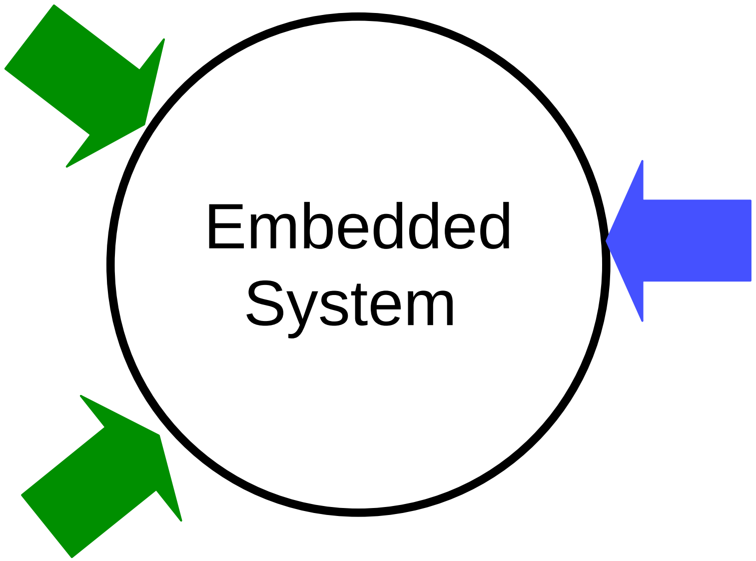
Theories of correctness
Verification

Promising

Proposed Vision: Multidisciplinary Integration

Execution constraints

CPU speed
memory
power
failure rates



Computing

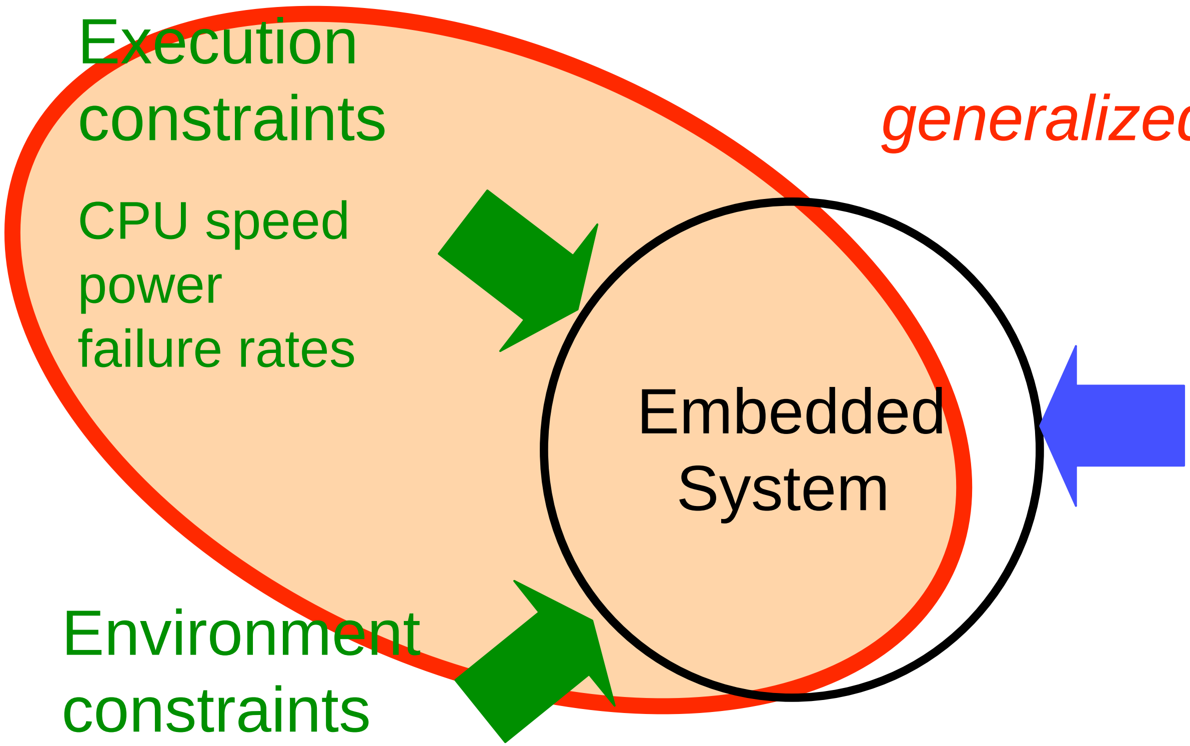
algorithms
protocols
architectures

Environment constraints

- Performance (deadlines, jitter, throughput)
- Robustness (security, safety, availability)

Proposed Vision: Multidisciplinary Integration

Embedded System Design
is
generalized hardware design



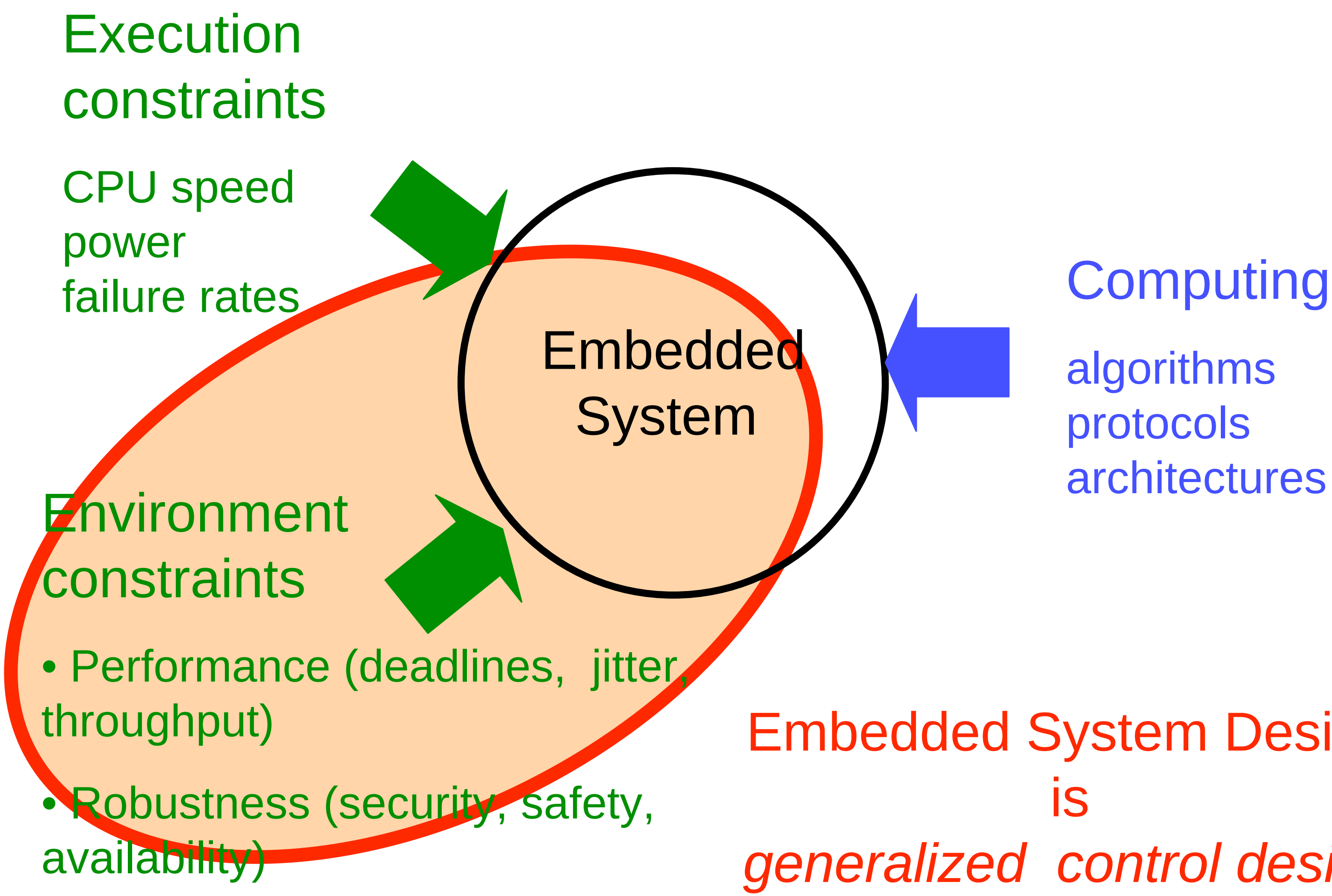
Computing

algorithms
protocols
reuse

• Performance (deadlines, jitter, throughput)

• Robustness (security, safety, availability)

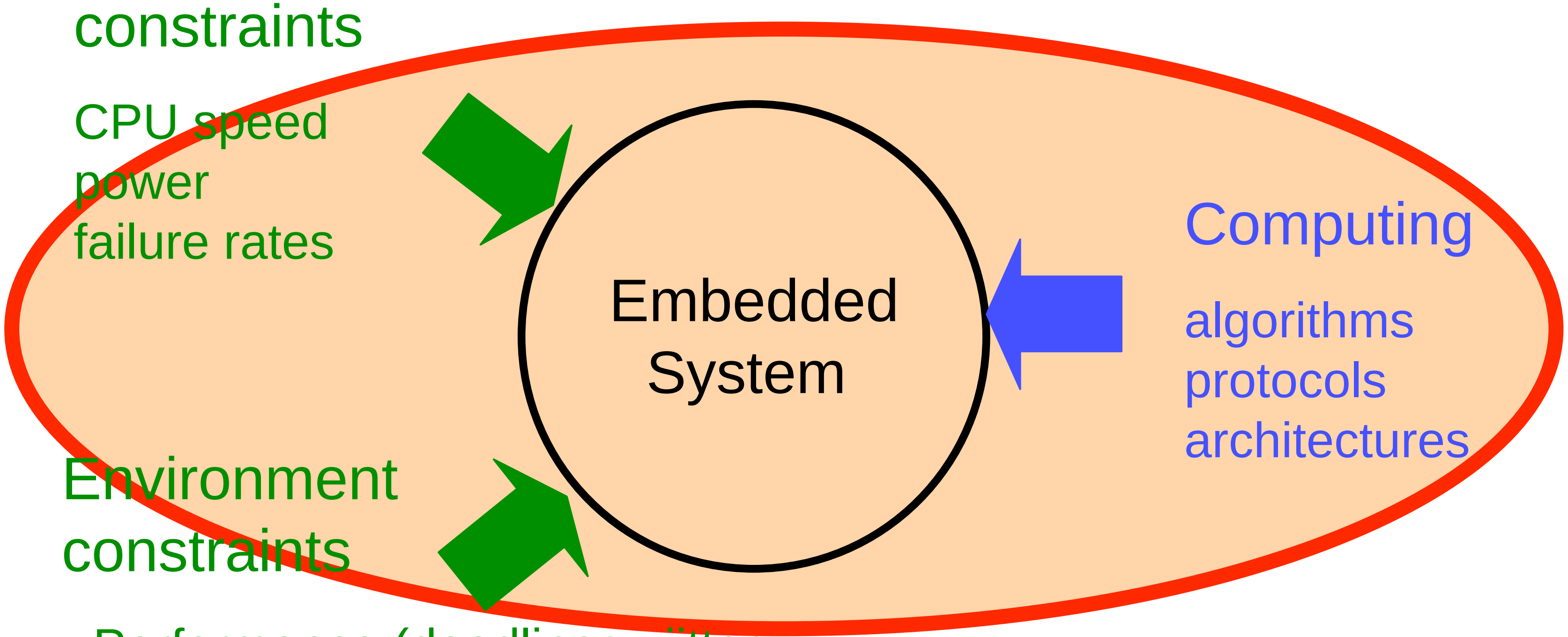
Proposed Vision: Multidisciplinary Integration



Embedded System Design
is
generalized control design

Proposed Vision: Multidisciplinary Integration

Embedded System Design
coherently integrates all these



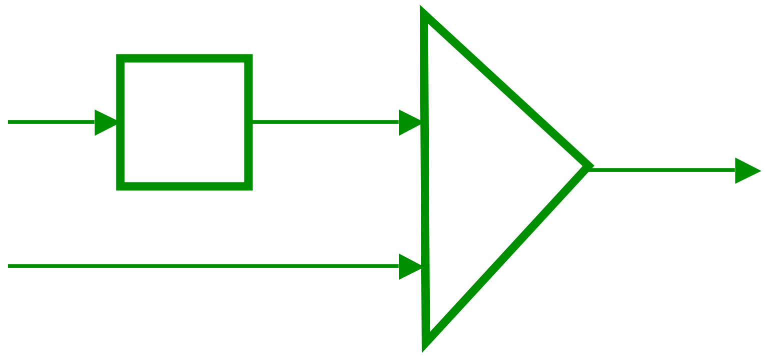
- Performance (deadlines, jitter, throughput)
- Robustness (security, safety, availability)

We need to revisit and revise the most basic computing paradigms to include methods from EE and Control

Sub-challenge 1:
Integrate Analytical and Computational Modeling

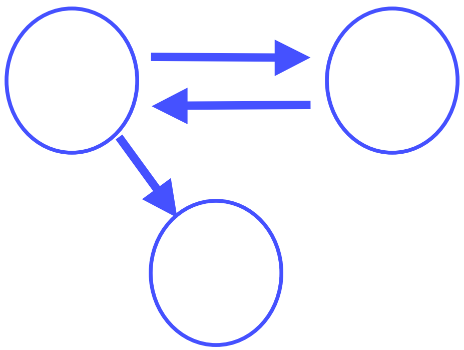
Physical Systems Engineering

Component model: transfer function
Composition: parallel
Connection: data flow



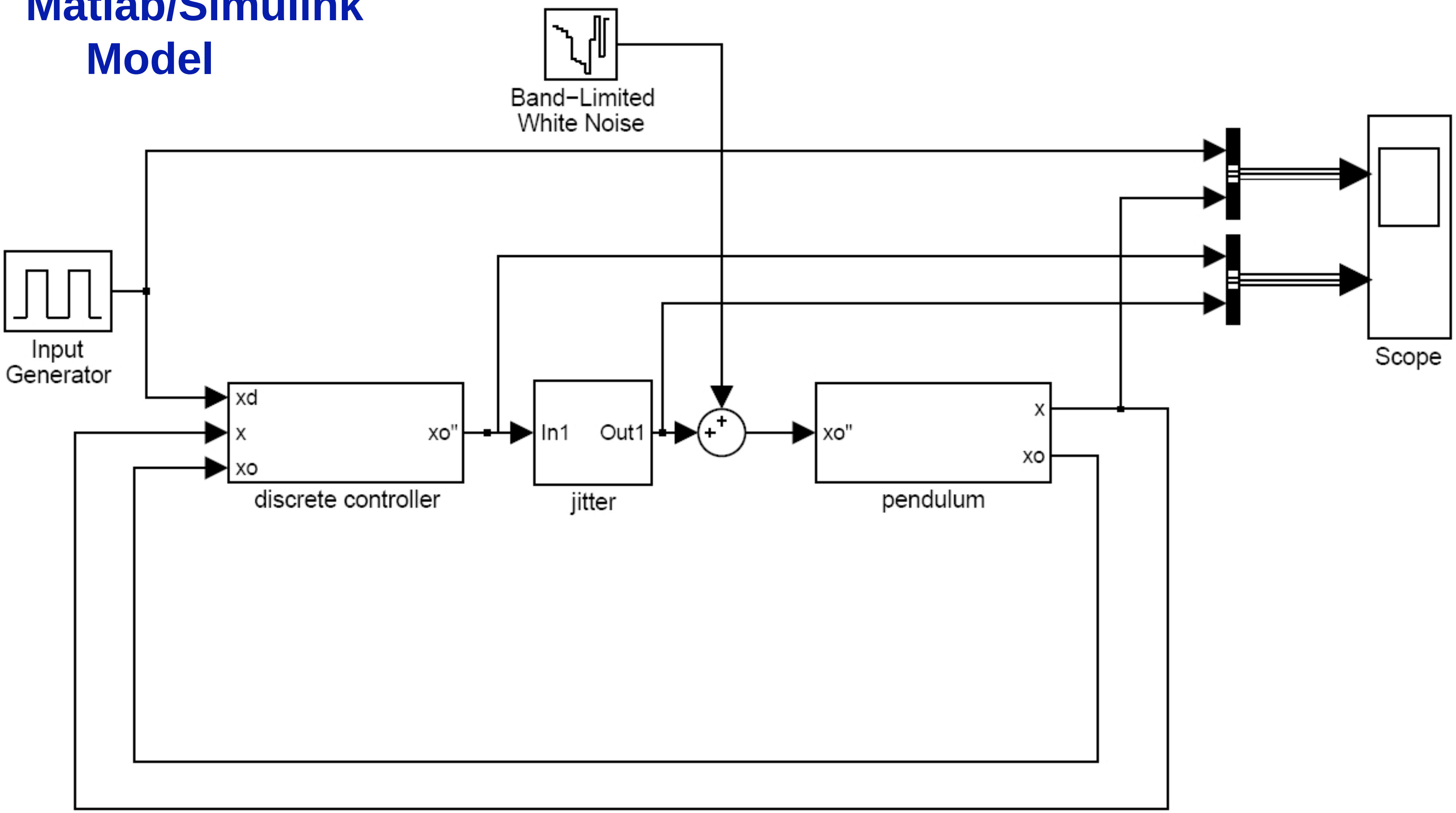
Computing Systems Engineering

Component model: subroutine
Composition: sequential
Connection: control flow



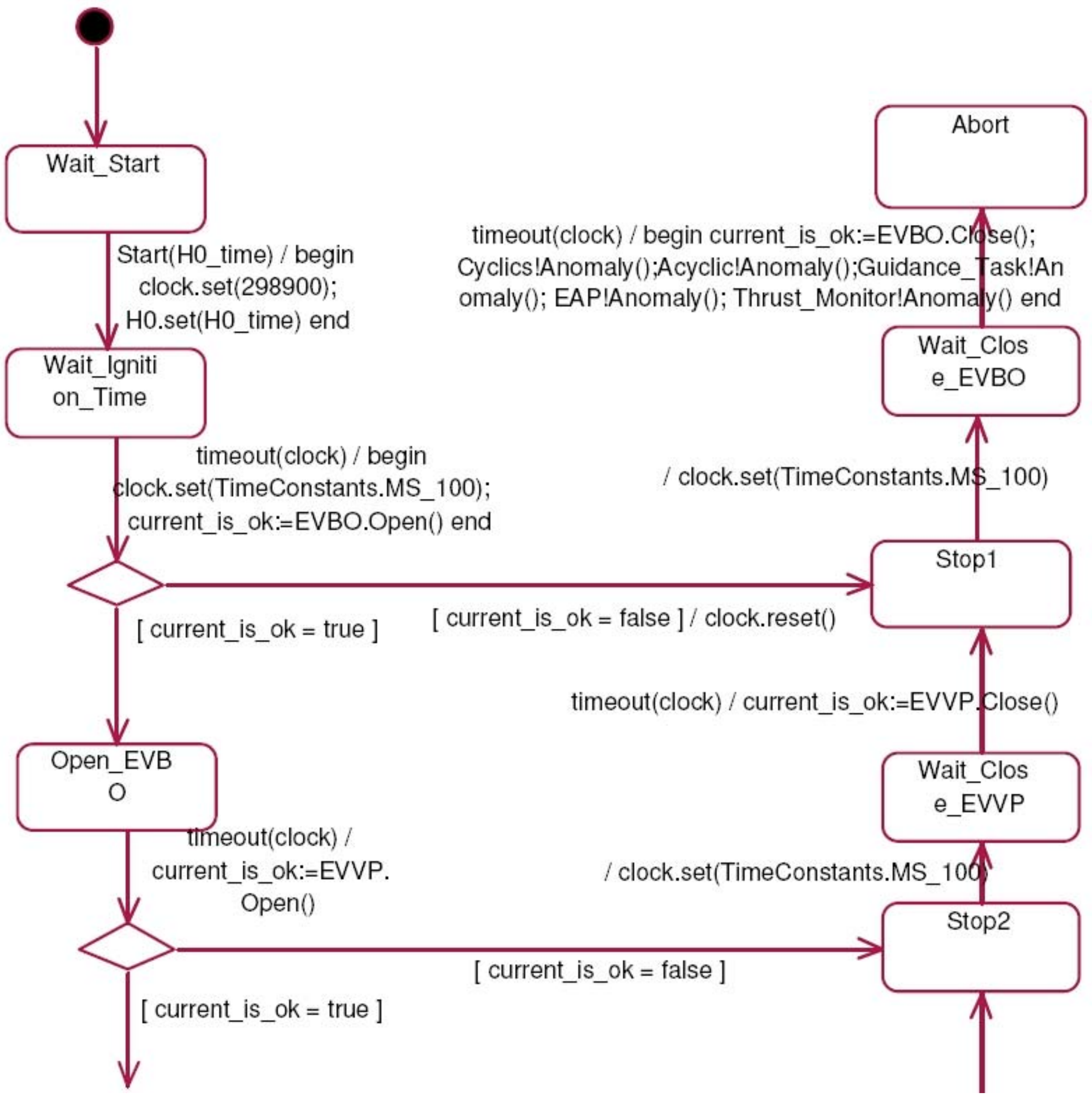
Sub-challenge 1:
Integrate Analytical and Computational Modeling

Matlab/Simulink Model



Sub-challenge 1:
Integrate Analytical and Computational Modeling

UML Model
(Rational Rose)

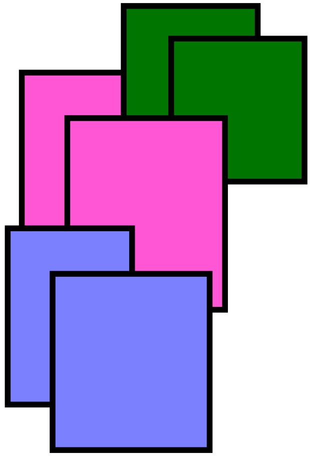


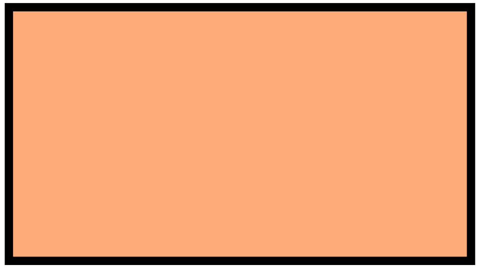
Sub-challenge 1:
Integrate Analytical and Computational Modeling

Analytical Models	Computational Models
Defined by equations Deterministic or probabilistic	Defined by programs Executable by abstract machines
Strengths:	
Concurrency Real time Quantitative constraints (power, QoS, mean-time-to-failure)	Dynamic change Complexity theory Nondeterminism (abstraction hierarchies, partial specifications)
Tool support:	
Average-case analysis Optimization Continuous mathematics (differential equations, stochastic processes)	Worst-case analysis Compilers Discrete mathematics (logic, combinatorics)
Main paradigm:	
Synthesis	Verification

Sub-challenge 2:
Component-based Engineering

Component-based Design: Build from a given set of components a system meeting given requirements





***Key issues**:*

- *Encompassing **Heterogeneity**:*
  *We need a unified framework for the meaningful **composition** of heterogeneous components*
- *Achieving **Constructivity**:*
  *We need **architectures** and **rules** for correctness by construction wrt essential properties*
- *The two demands for **heterogeneity** and **constructivity** pull in different directions.*

Sub-Challenge 2:
Encompassing Heterogeneity

Embedded systems are built from components with different characteristics.

We distinguish 3 main sources of heterogeneity:

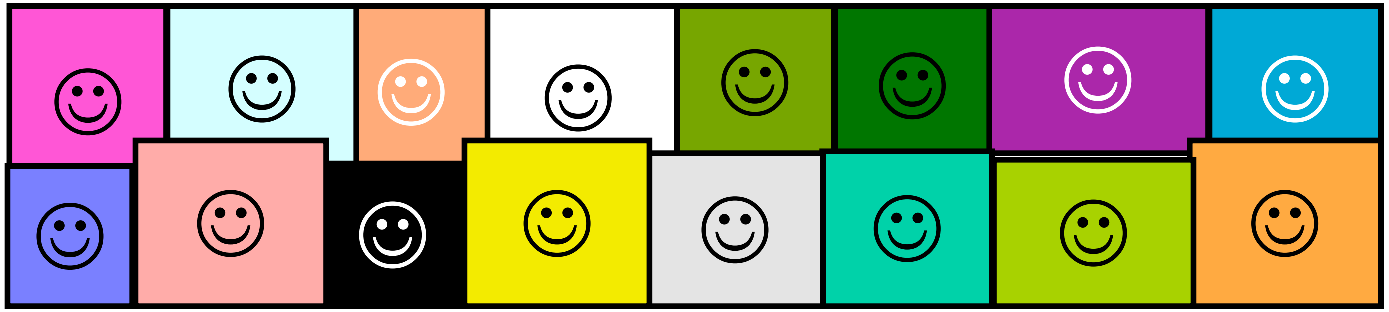
- **Execution**: synchronous and asynchronous components
- **Interaction**: function call, broadcast, rendezvous, monitors
- **Abstraction levels**: hardware, execution platform, application software

*We need a **unified composition paradigm** for describing and analyzing the coordination between components.*

*Such a paradigm would allow system designers and implementers to formulate their solutions in terms of **tangible, well-founded and organized concepts** instead of using dispersed low-level coordination mechanisms including semaphores, monitors, message passing, remote call, protocols etc.*

Sub-challenge 2:
Constructivity - Compositionality

Rules for proving global properties of compound components from properties of individual components.



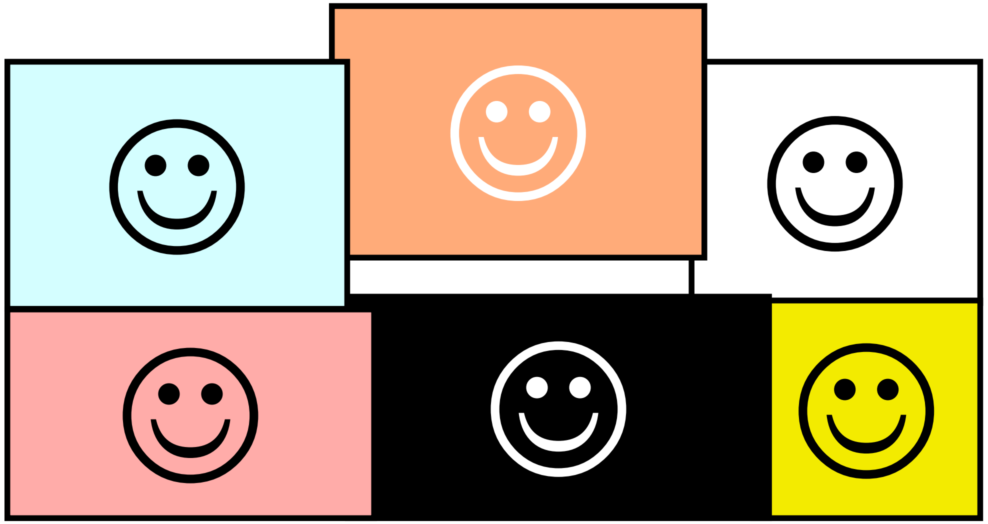
Sub-challenge 2:
Constructivity - Compositionality

Rules for proving global properties of compound components from properties of individual components.

We need compositionality results for progress properties and extra-functional properties

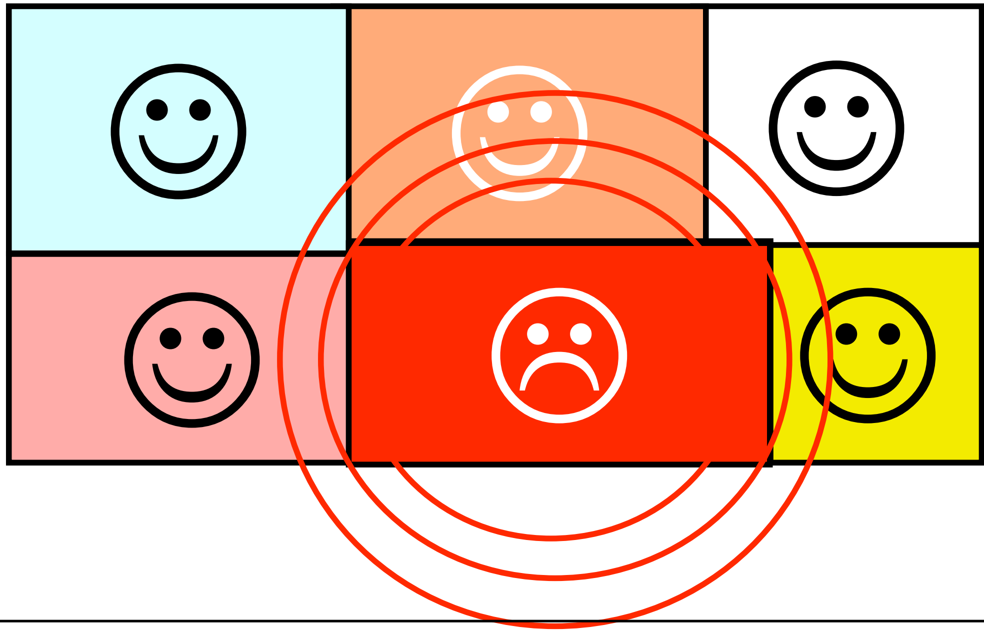
Sub-challenge 2:
Constructivity - Composability

Rules guaranteeing that essential properties of individual components are preserved across composition.



Sub-challenge 2:
Constructivity - Composability

Rules guaranteeing that essential properties of individual components are preserved across composition.



Property stability phenomena are poorly understood.
We need composability results e.g.
- *feature interaction in middleware*
- *composability of scheduling algorithms*
- *theory for reconfigurable systems*

Sub-challenge 3: 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.

Adaptivity is a means for enforcing predictability in the presence of uncertainty

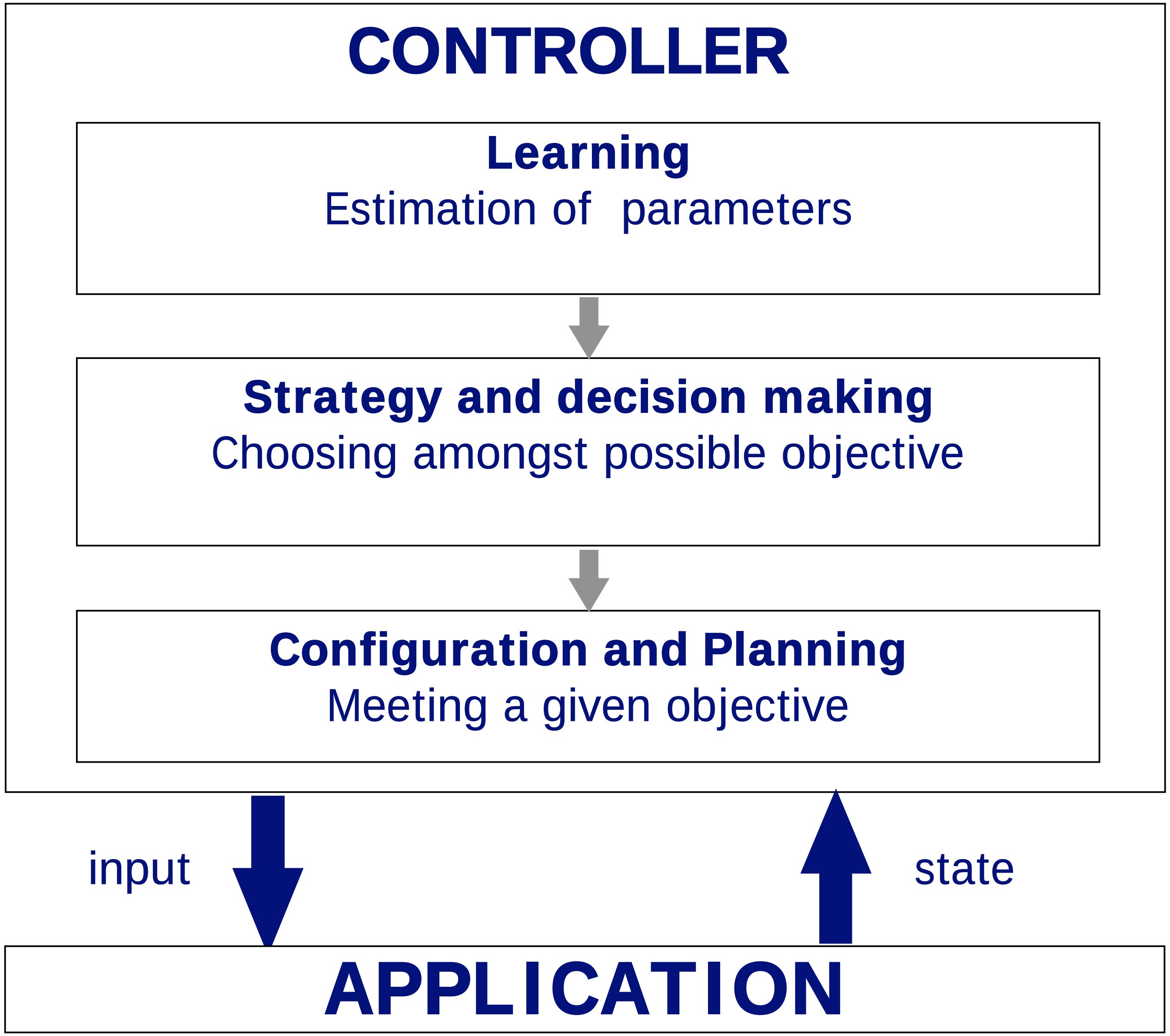
- Uncertainty is characterized as the difference between average and worst-case behavior of a system's environment. The trend is towards drastically increasing uncertainty, due to:
  - Connectivity with complex, non-deterministic, possibly hostile external environments
  - Execution platforms with sophisticated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

Sub-challenge 3: Adaptive Systems - Critical vs. Best effort

- Increasing 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.

Challenge: 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).

Sub-challenge 3: Adaptive Systems - Architecture



The central problem: Rigorous System Design

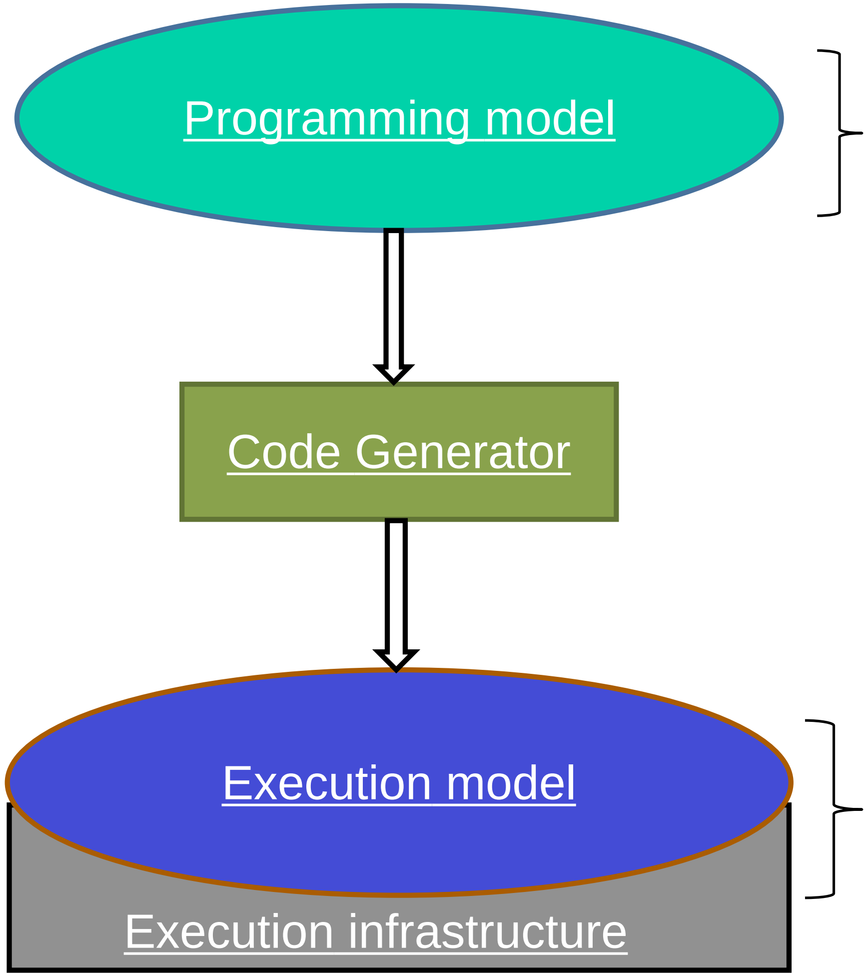
Rigorous system design methods rely on the implicit or explicit use of a pair (programming model, execution model), e.g.

- Synchronous languages have reactive execution models
- Real-time languages such as ADA rely on « event driven » execution (fixed priorities and preemption)
- Time triggered languages and architectures (TTA, Oasis, Giotto)

This allows :

- correctness-by-construction for certain essential properties, the correspondence between programs and their implementation is established once and for all
- automatic code generation becomes possible

The central problem: Rigorous System Design



Extension of an existing language with concepts and primitives for concurrency and resource management

Abstract machine encompassing execution mechanisms needed for efficient and dependable execution

Model-based Development – the idea

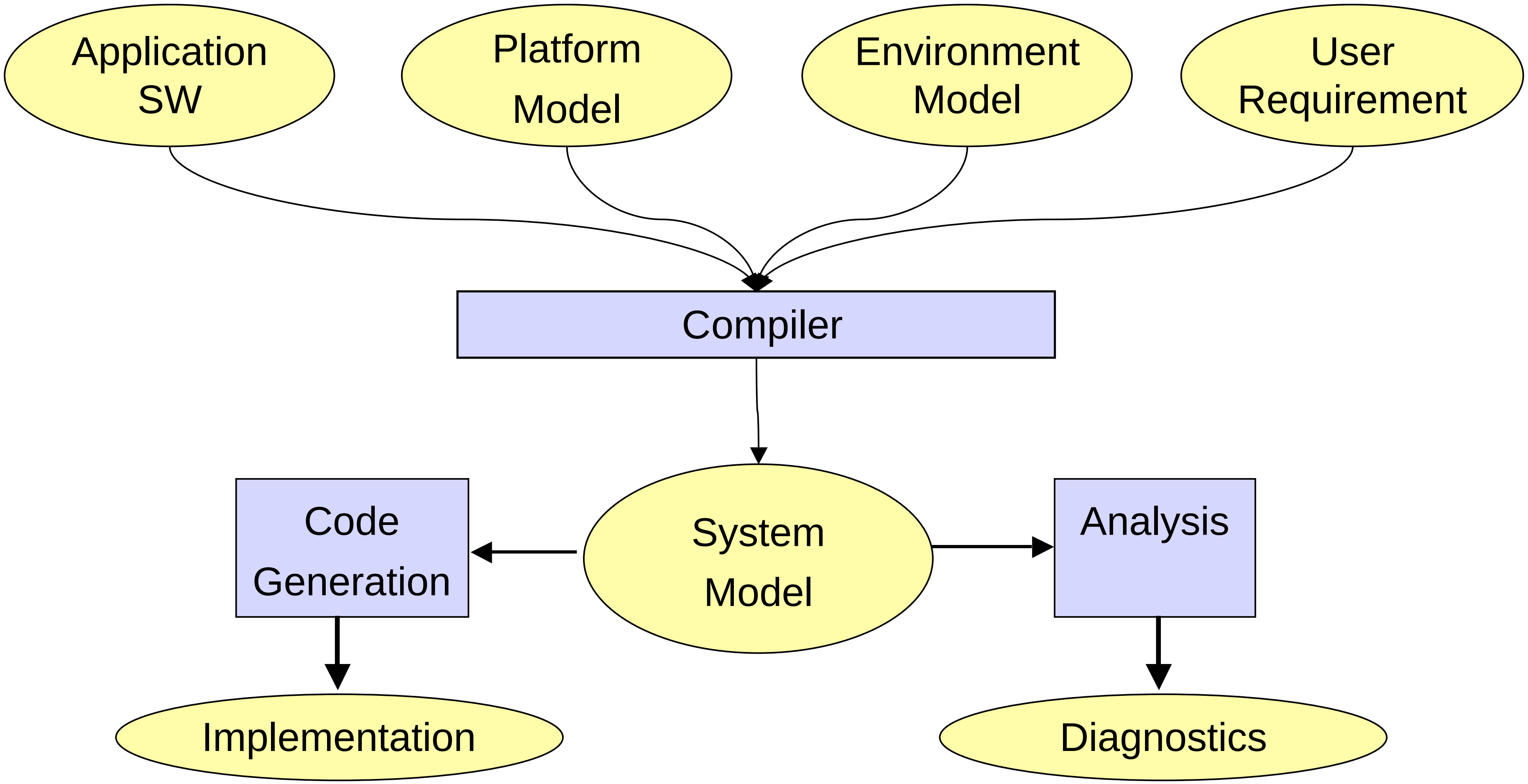
Move from physical prototypes to virtual prototypes (models) with obvious advantages : minimize costs, flexibility, genericity, formal validation is a possibility

Modeling and validation environments for complex real-time systems

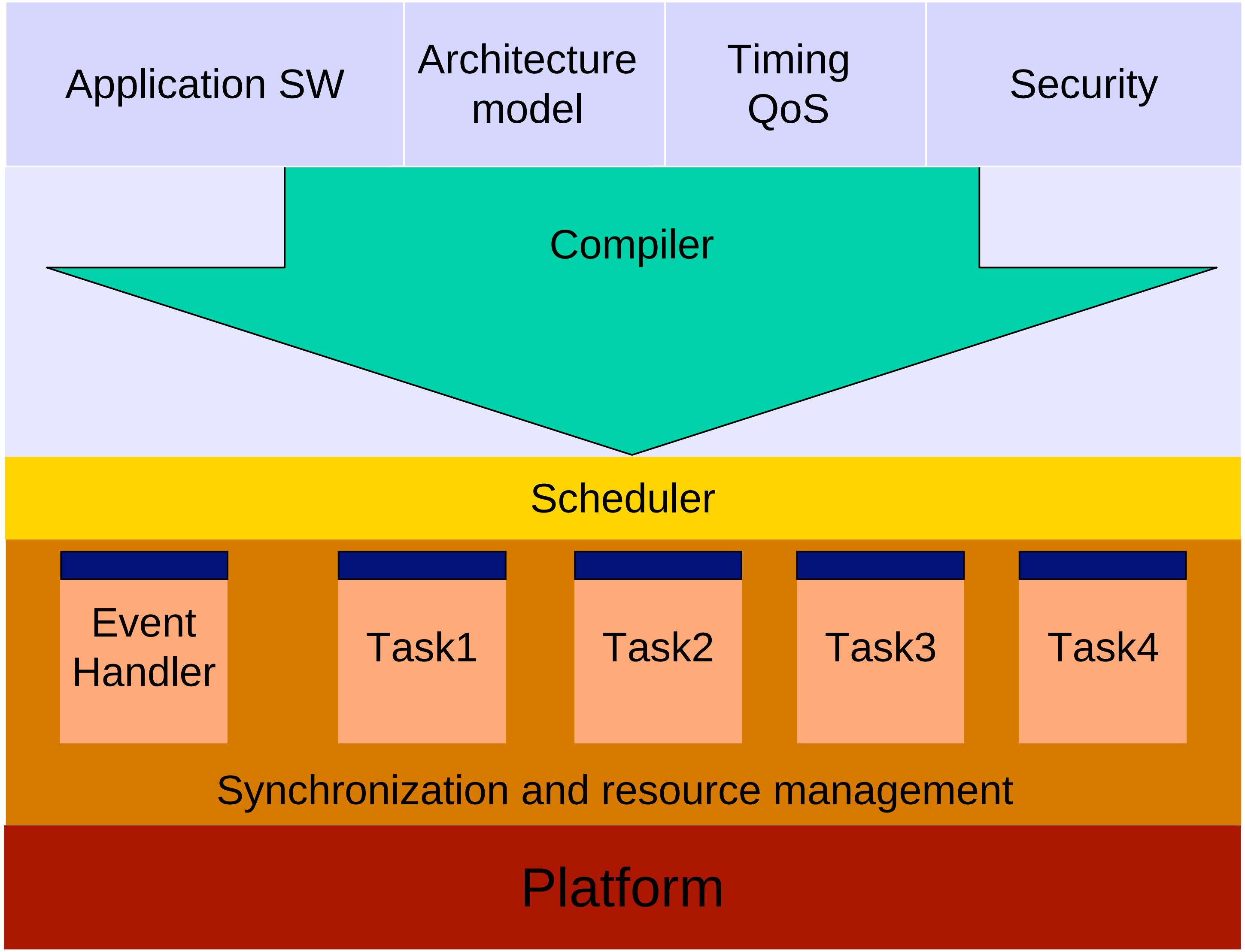
- Libraries of Components
  ex. HW, SW, Models of continuous dynamic systems
- Languages and tools for assembling components

Synthesize embedded software from domain-specific models
ex. Matlab, SystemC, UML, SDL.

Model-based Development – the principle



Resource-aware Compilation



Operating Systems

Operating systems are often:

- Far more complex than necessary
- Undependable
- With hidden functionality
- Difficult to manage and use efficiently

We should move towards lightweight operating systems, each dedicated to a particular application domain ***ex. OSEK, ARINC, JavaCard, TinyOS***

- Minimal architectures, reconfigurable, adaptive, with features for **safety** and **security**
- Give up control to the application –
  move resource management outside the kernel
- Supply and allow adaptive scheduling policies which take into account the environmental context (ex: availability of critical resources such as energy).

Control for Embedded Systems

Automation applications are of paramount importance –
their design and implementation raise difficult problems

Hybrid Systems – active research area

- Combination of continuous and discrete control techniques
- Multi-disciplinary integration aspects (control, numerical analysis, computer science)
- Modeling and Verification
- Distributed and fault-tolerant implementations (influence communication delays, clock drift, aperiodic sampling)

↳ Use of control-based techniques for adaptivity

Dependability (Security, Safety, Availability …)

- *Traditional techniques based on massive redundancy are of limited value*
- *Dependability should be a guiding concern from the very start of system development. This applies to programming style, traceability, validation techniques, fault-tolerance mechanisms, …*

<u>Work Directions :</u>

- Methodologies for domain-specific standards, such as :
  - DO178B Process Control Software Safety Certification
  - Integrated Modular Avionics; Autosar
  - Common Criteria for Information Technology Security Evaluation
- Verification Technology (verify resistance to certain classes of errors and attacks) – **certification**
- Architectures**,** protocols and algorithms for fault-tolerance and security taking into account QoS requirements (real-time, availbability)

Networked Embedded Systems : Wireless Sensor Networks

Nodes

- sensors + actuators + CPU+ Memory (~100 KB) + radio

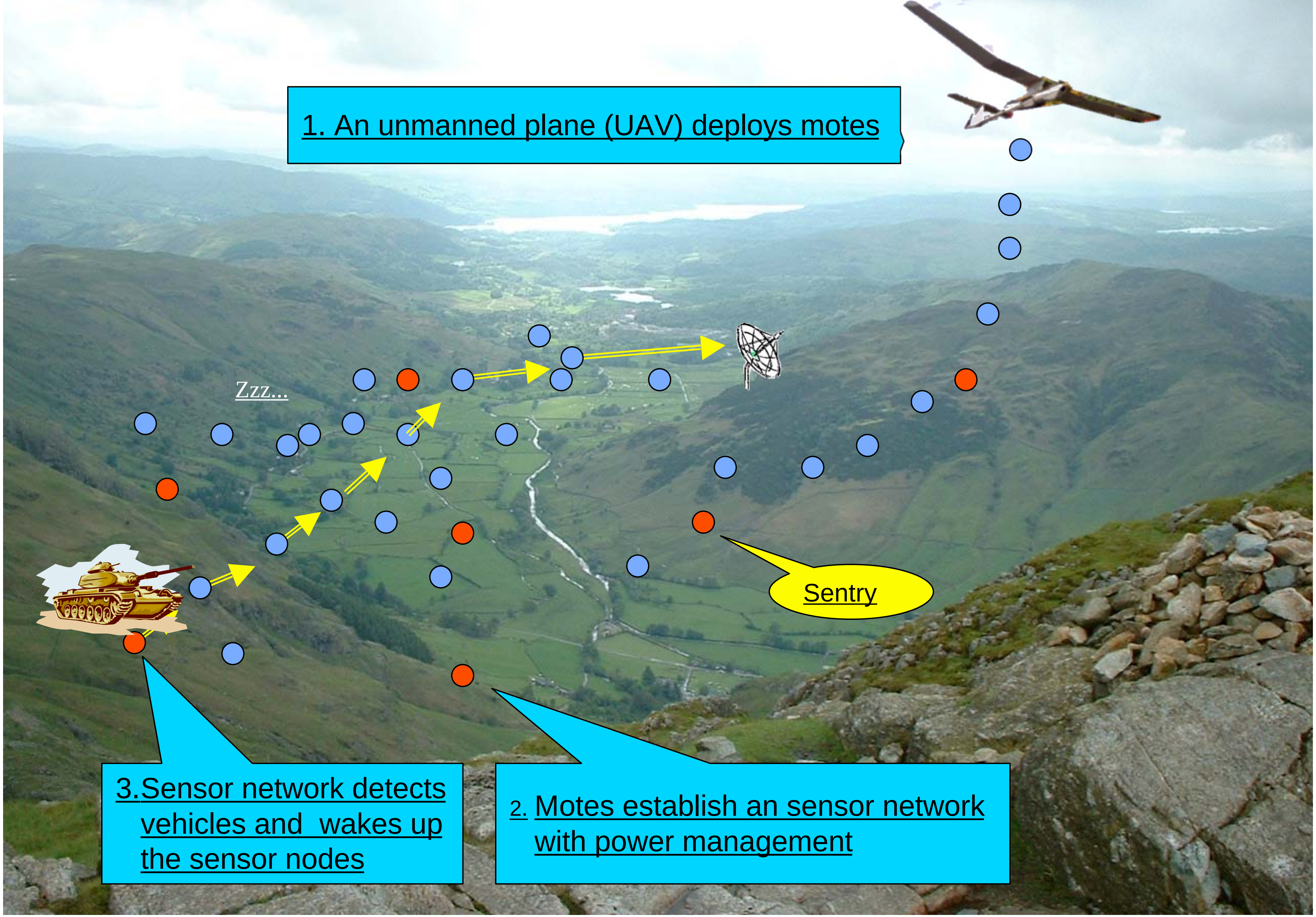
Technical characteristics

- Real-time
- Scarce power
- Dynamically changing resources
- Self-organization, adaptive aggregate behavior is important

Applications

- Military: surveillance and warfare
- Monitoring : environmental, biological, medical
- Smart environments, ubiquitous computing

Wireless Sensor Networks



Adaptive real-time behavior

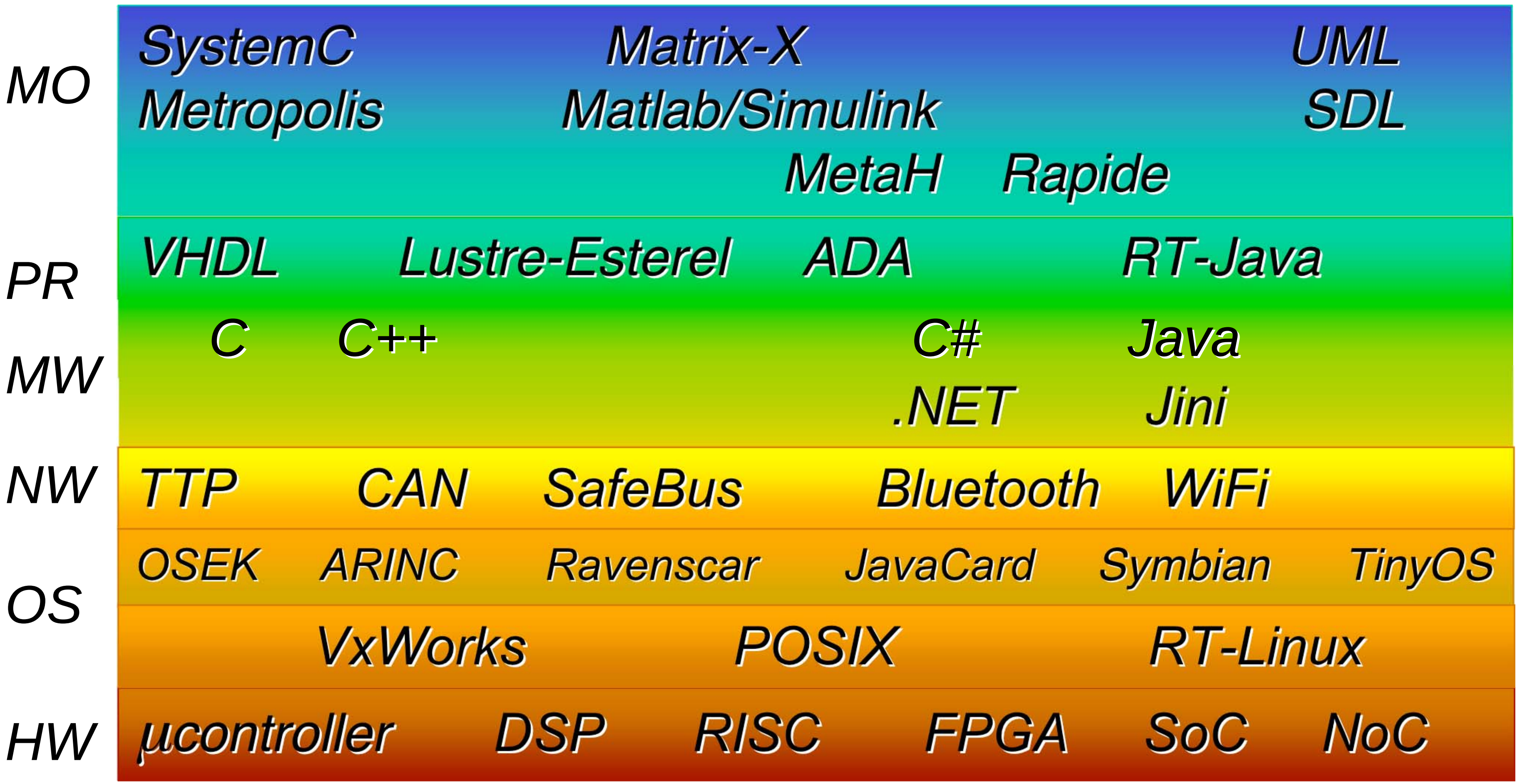
Inherently dynamic, must adapt to accommodate workload changes and to counter uncertainties in the system and its environment

- Clock synchronization, parameter settings
- Specific routing algorithms
- Location discovery, neighbor discovery
- Group management (dormant, active-role assignment)
- Self-organization : Backbone creation, leader election, collaboration to provide a service

Power management :

- turn-off of dormant nodes
- periodical rotation of active nodes to balance energy

Integration of Methods and Tools



Conclusion

Research: Embedded Systems offer a unique opportunity for creating a new discipline marrying computation and physicality. The challenge spans the spectrum from theoretical foundations to engineering practice.

Education: In order to adequately train new generations of engineers and researchers, institutions need to focus on embedded systems as a scientific discipline and as a specialization area within existing curricula. This requires taking down the cultural wall that exists between many Computer Science and Electrical Engineering departments.

Industry: Industry tends to stay with available technologies, optimizing existing investments and competencies. Nonetheless, the inherent limits of ad-hoc approaches to manage system complexity, and the resulting explosion in costs, provide strong incentives for industry to look for alternatives. It is important to seize this opportunity and develop new technologies through joint academic-industrial pilot projects.

MERCI