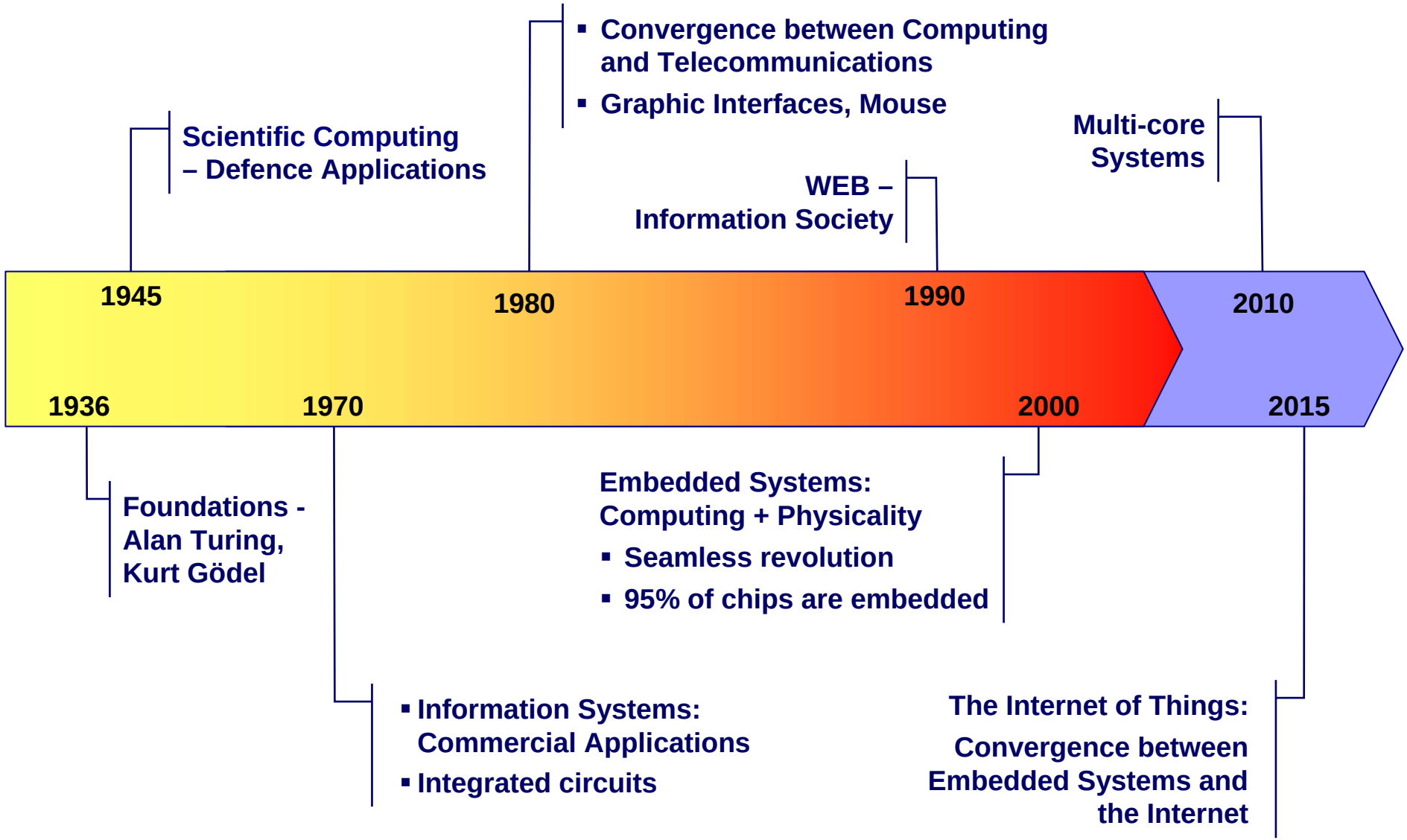
Embedded Systems Design – Challenges and Work Directions

NECSYS 2010
Annecy
September 14, 2010

Joseph Sifakis
VERIMAG Laboratory

The Evolution of Informatics



Informatics is a young discipline, driven by exponential growth of components and their applications.

Embeded Systems

Electronic components integrate **software and hardware**
jointly and specifically designed to provide given functionalities, which are often **critical**.

OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

System Design – New Trends

New trends break with traditional Computing Systems Engineering.
It is hard to jointly meet **technical requirements** such as:

- **Reactivity**: responding within known and guaranteed delay
  Ex : flight controller
- **Autonomy**: provide continuous service without human intervention
  Ex : no manual start, optimal power management
- **Dependability:** guaranteed minimal service in any case
  Ex : attacks, hardware failures, software execution errors
- **Scalability:** at runtime or evolutionary growth (linear performance increase with resources)
  Ex : reconfiguration, scalable services

...and also take into account **economic requirements** for optimal cost/quality

Technological challenge :
Capacity to build systems of guaranteed functionality and quality,
at an acceptable cost.

System Design – State of the Art

TODAY

We master – at a high cost – two types of systems which are difficult to integrate:

- Safety and/or security critical systems of low complexity
  - ***Flight controller, smart card***
- Complex « best effort » systems
  - ***Telecommunication systems, web-based applications***

TOMORROW

We need:

- Affordable critical systems
  Ex : transport, health, energy management
- Successful integration of heterogeneous systems of systems
  - ***Convergence internet/embedded systems (internet of things)***
  - ***Automated Highways***
  - ***New generation air traffic control***
  - ***« Ambient Intelligence»***

System Design – Still a long way to go

Physics

Theory for building artifacts with predictable behavior

Informatics

Lack of results allowing constructivity

Suggested by T. Henzinger: T. Henzinger, J. Sifakis "The Embedded Systems Design Challenge" FM06

System Design – Still a long way to go

Design of large IT systems is a risky undertaking, mobilizing hundreds of engineers over several years.

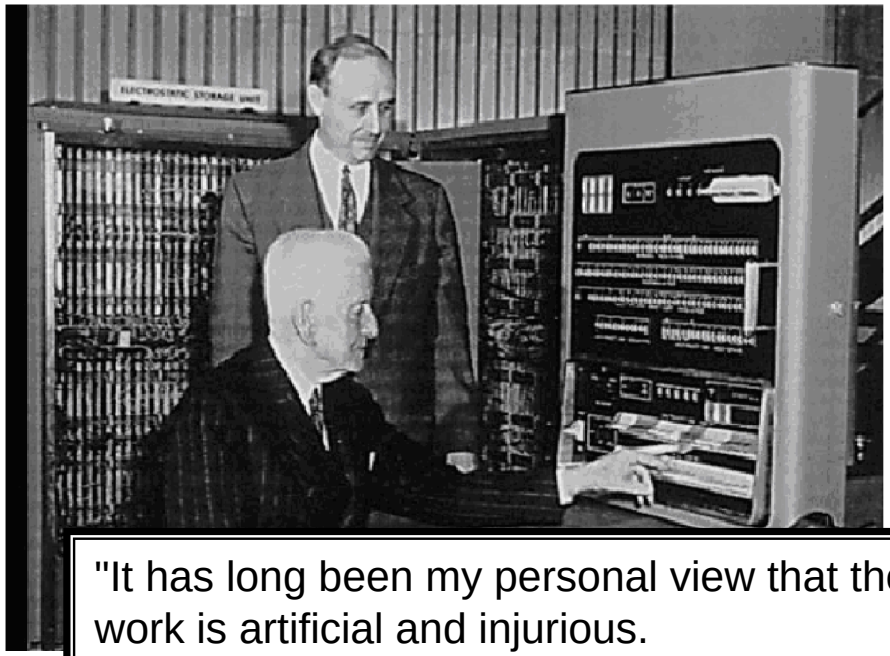
Difficulties

- Complexity – mainly for building systems by reusing existing components
- Requirements are often incomplete, and ambiguous (specified in natural language)
- Design approaches are empirical and based on the expertise and experience of teams

Consequences

- Large IT projects are often over budget, over time, and deliver poor quality.
- Of these, 40% fail, 30% partially succeed, 30% succeed.

System Design – a Long Way to go



There is an increasing gap between:

- Our technological capabilities for treating and transmitting information
- Our know-how in computing systems engineering

"It has long been my personal view that the separation of practical and theoretical work is artificial and injurious.

Much of the practical work done in computing, both in software and in hardware design, is unsound and clumsy because the people who do it have not any clear understanding of the fundamental design principles of their work.

Most of the abstract mathematical and theoretical work is sterile because it has no point of contact with real computing.

Christopher Strachey (1916-1975)

OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Basic Technologies – Multicore Systems

- The switch to multicore architectures
  - is not at all the consequence of a scientific breakthrough,
  - is primarily due to technology walls that prevent from pushing forward the efficient implementation of traditional uniprocessor designs in silicon

- Increasing application's efficiency, simply by upgrading the hardware without significantly changing the software, is not anymore possible for multicore systems.

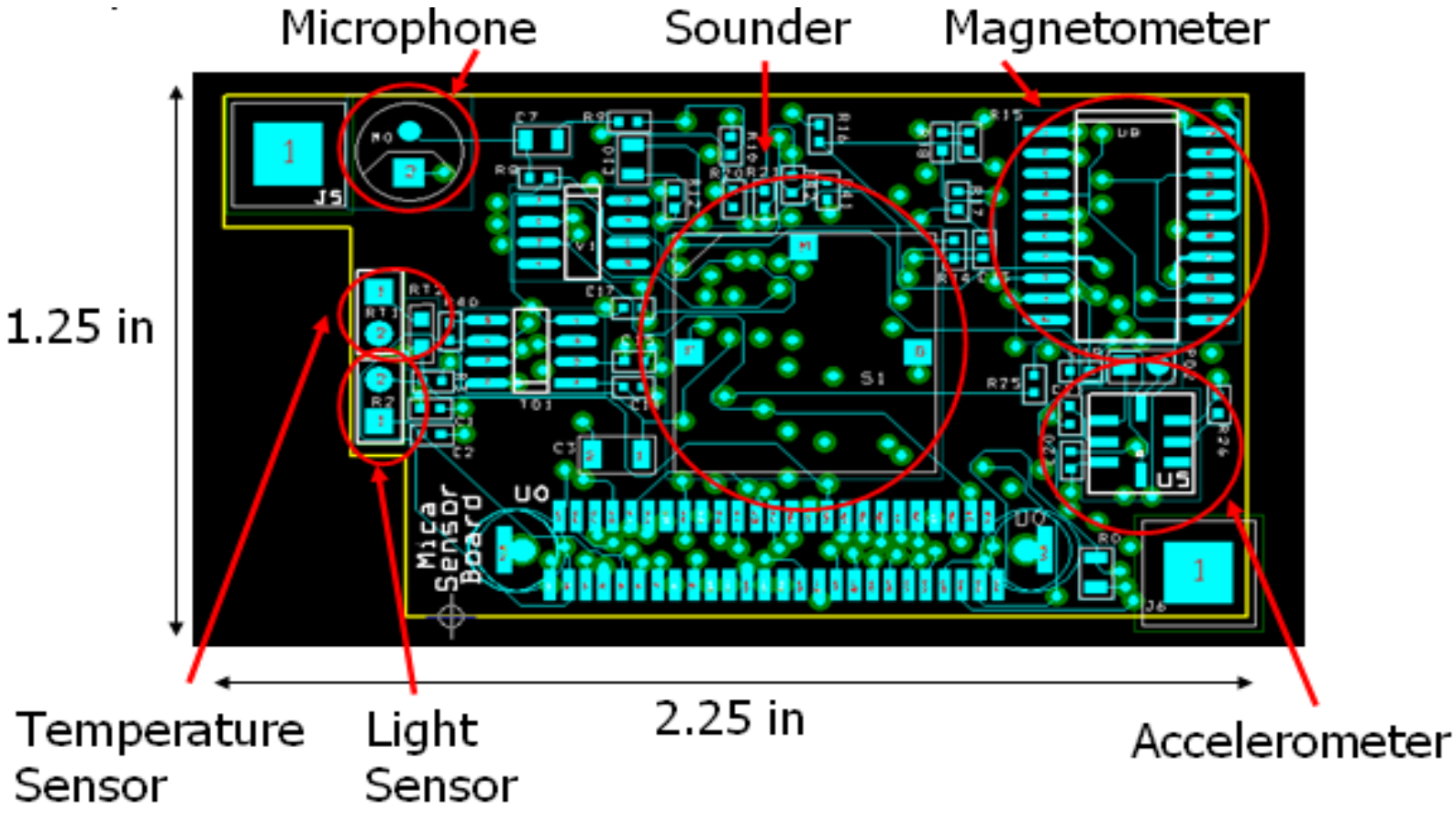
- The promise of parallel machines delivering almost unlimited computing power is not new. For decades there have been numerous industrial attempts in this direction, but almost all failed. Why?

Basic Technologies – Multicore Systems

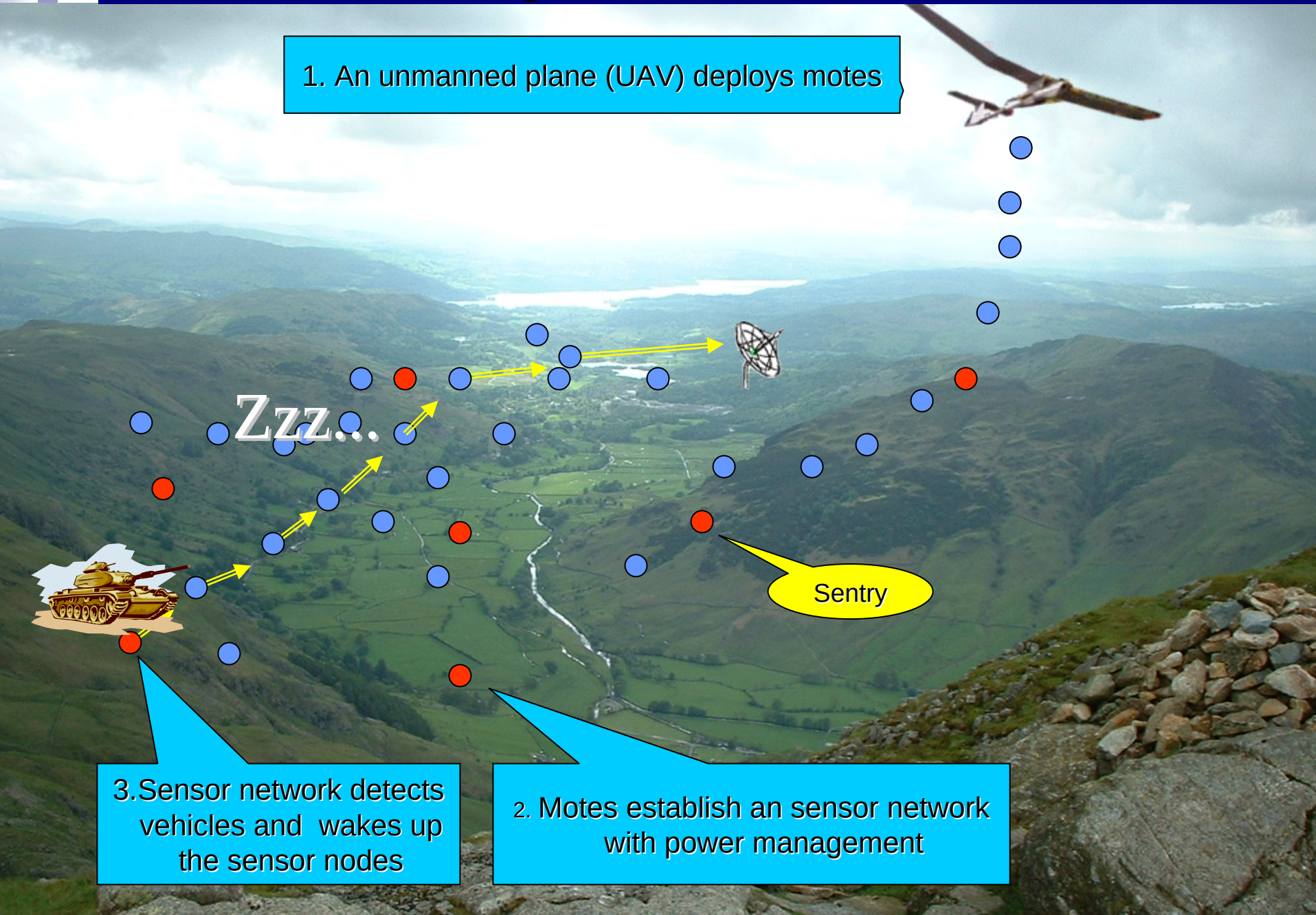
- “Software is struggling to keep pace with the fast growth of multicore processors”
- “Running advanced multicore machines with today's software is like "putting a Ferrari engine in a go-cart,“
- "Many of the software configurations in use today will be challenged to support the hardware configurations possible, and those will be accelerating in the future."

Gartner, Research Note, January 2009

Basic Technologies – Sensor Networks



Basic Technologies – Sensor Networks



Internet-based Systems – The Internet of Things

The “Internet of Things” will be the result of the convergence between Embedded Systems and the Internet.

Basic idea: Use the Internet Protocol Suite for Human/ES or ES/ES interaction

This is not as easy as it seems.

Some major breakthroughs needed:

- Wireless sensor networks and RFID technologies.
- Advances in miniaturization and nanotechnology mean that smaller and smaller things will have the ability to interact and connect.
- Getting IP down to small devices and implementing features for QoS control and responsiveness.
- Standards e.g. for naming objects, tools and platforms.
- Improving overall security, and reliability of the internet.

Evolution towards specific “critical” internets ?

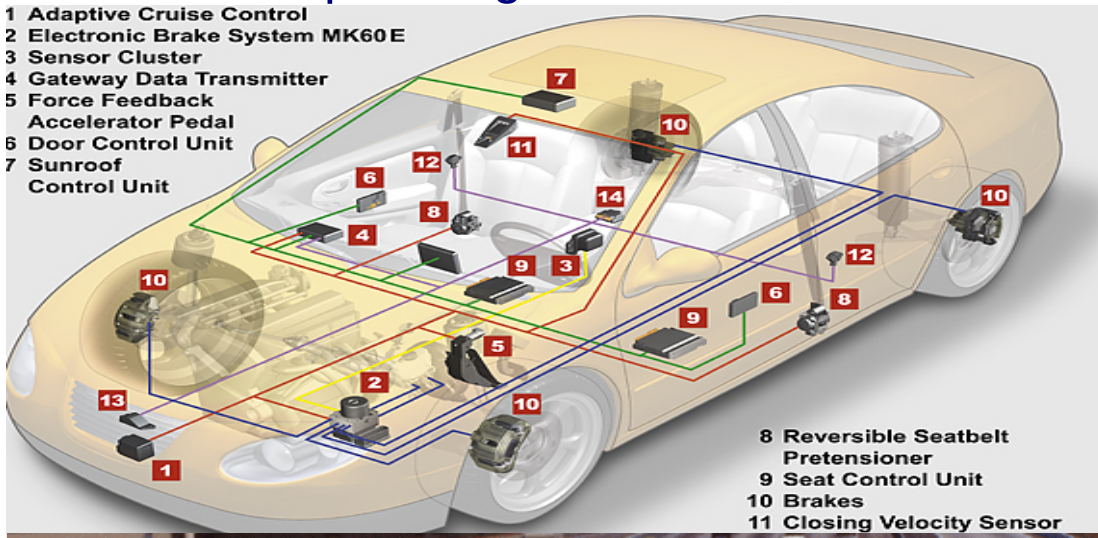
OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Applications – Transportation

Make transportation safer, more efficient, less polluting

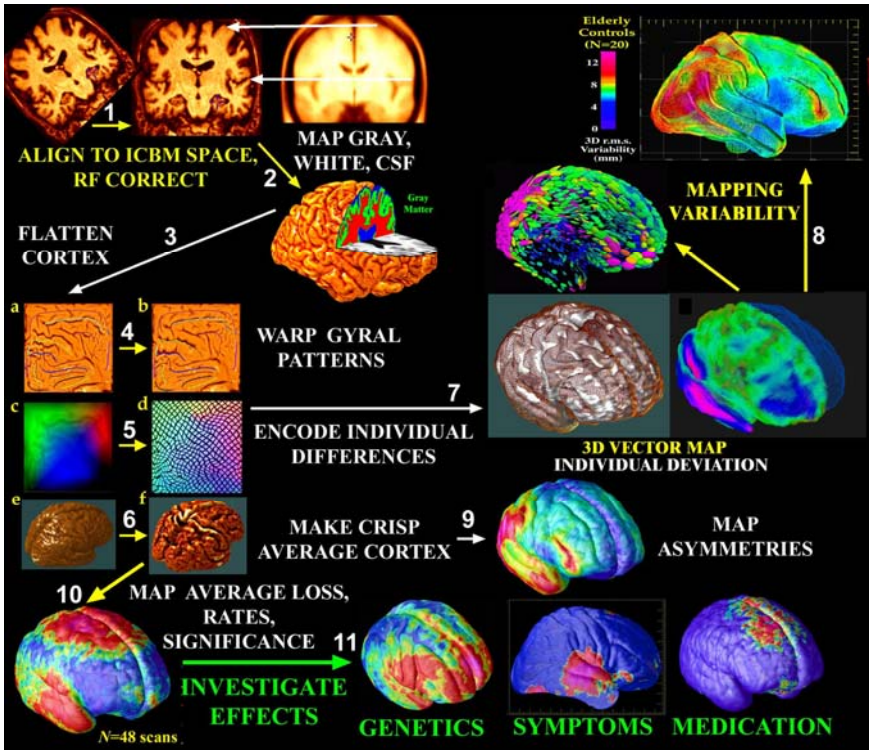
Active safety:
assist/protect the driver by using drive-by-wire and brake-by-wire technology



Automated Highways:
intelligent transportation system technology designed to provide for driverless cars on specific rights-of-way.

Applications – Health

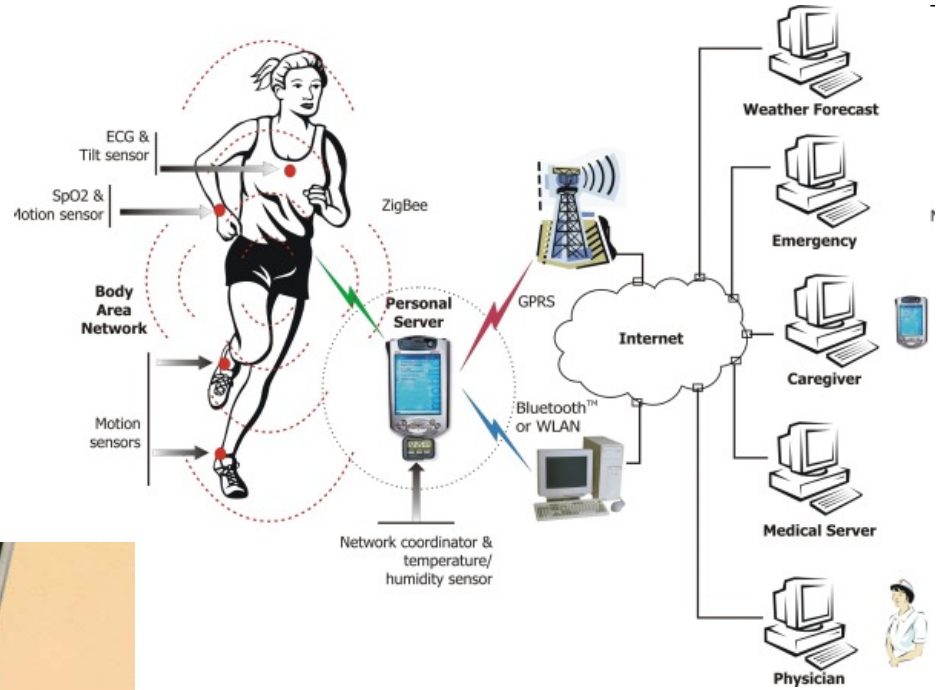
Robot-assisted surgery

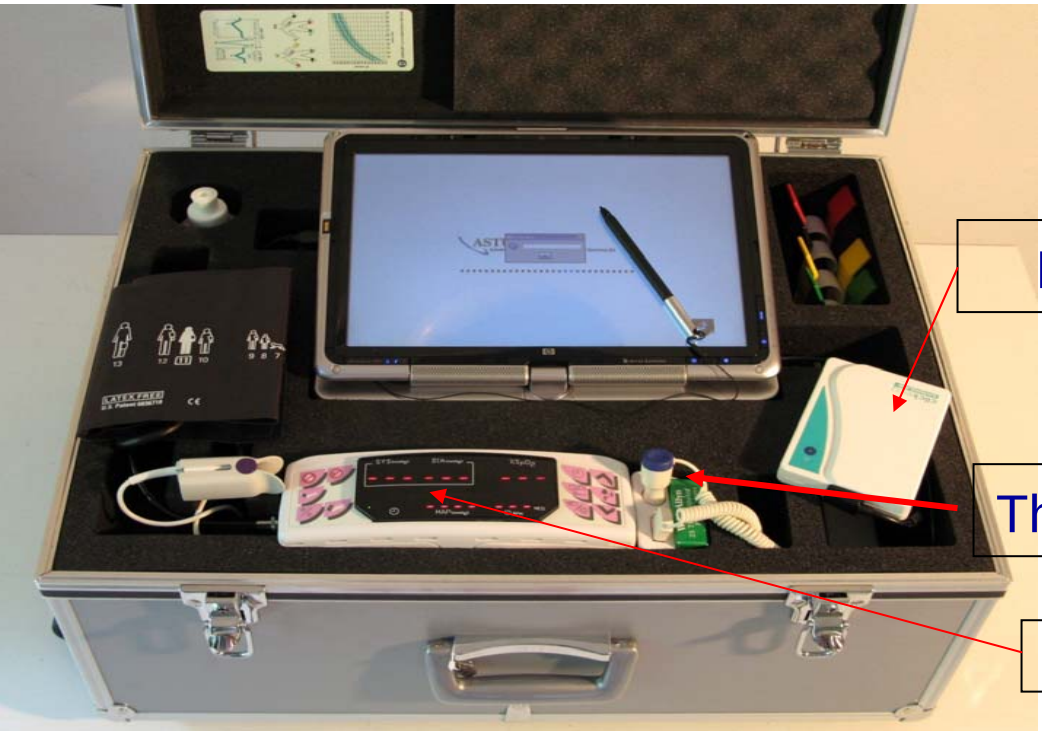


Brain image analysis

Applications – Health

Health monitoring



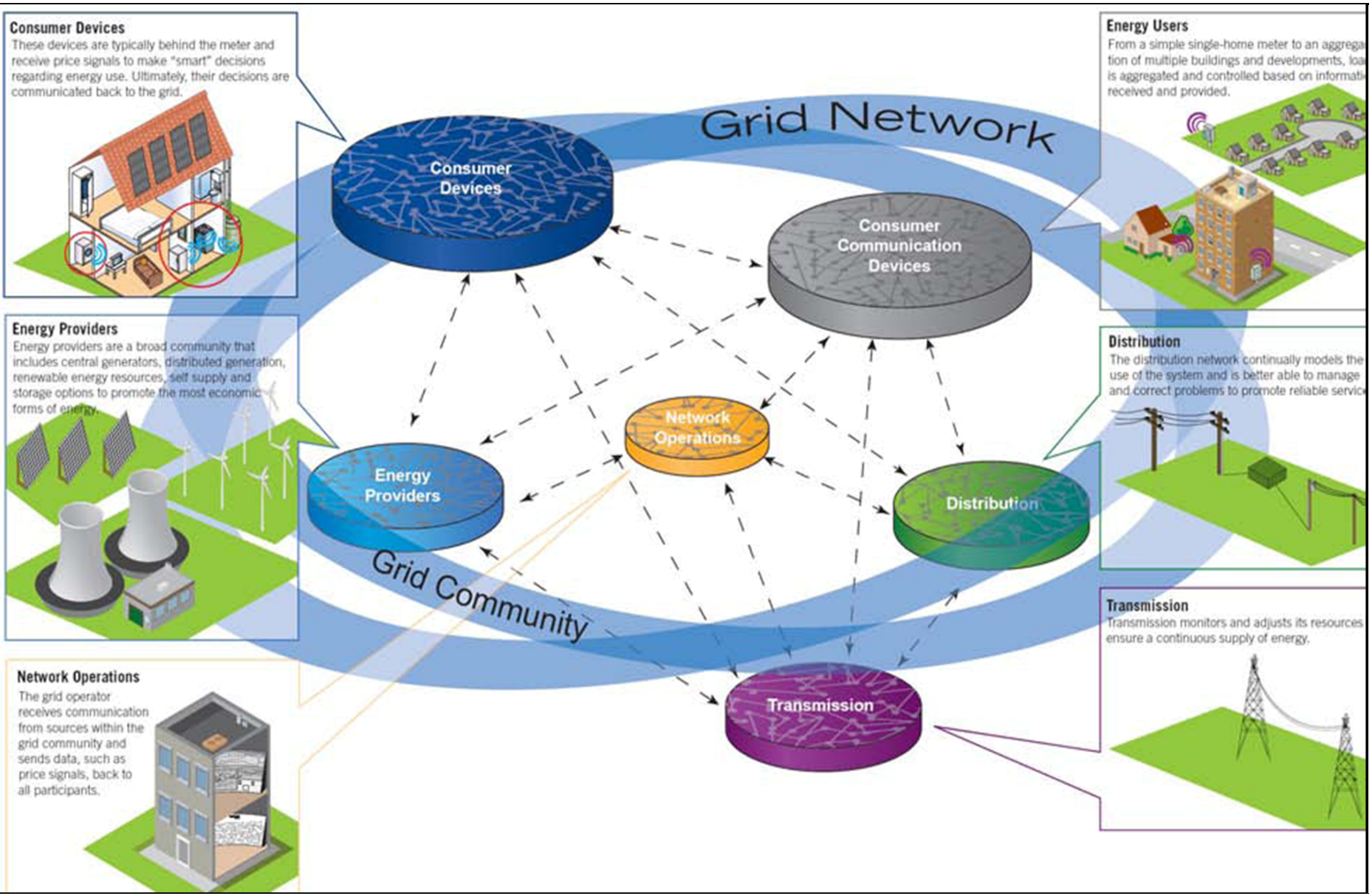


ECG

Thermometer

Blood pressure meter

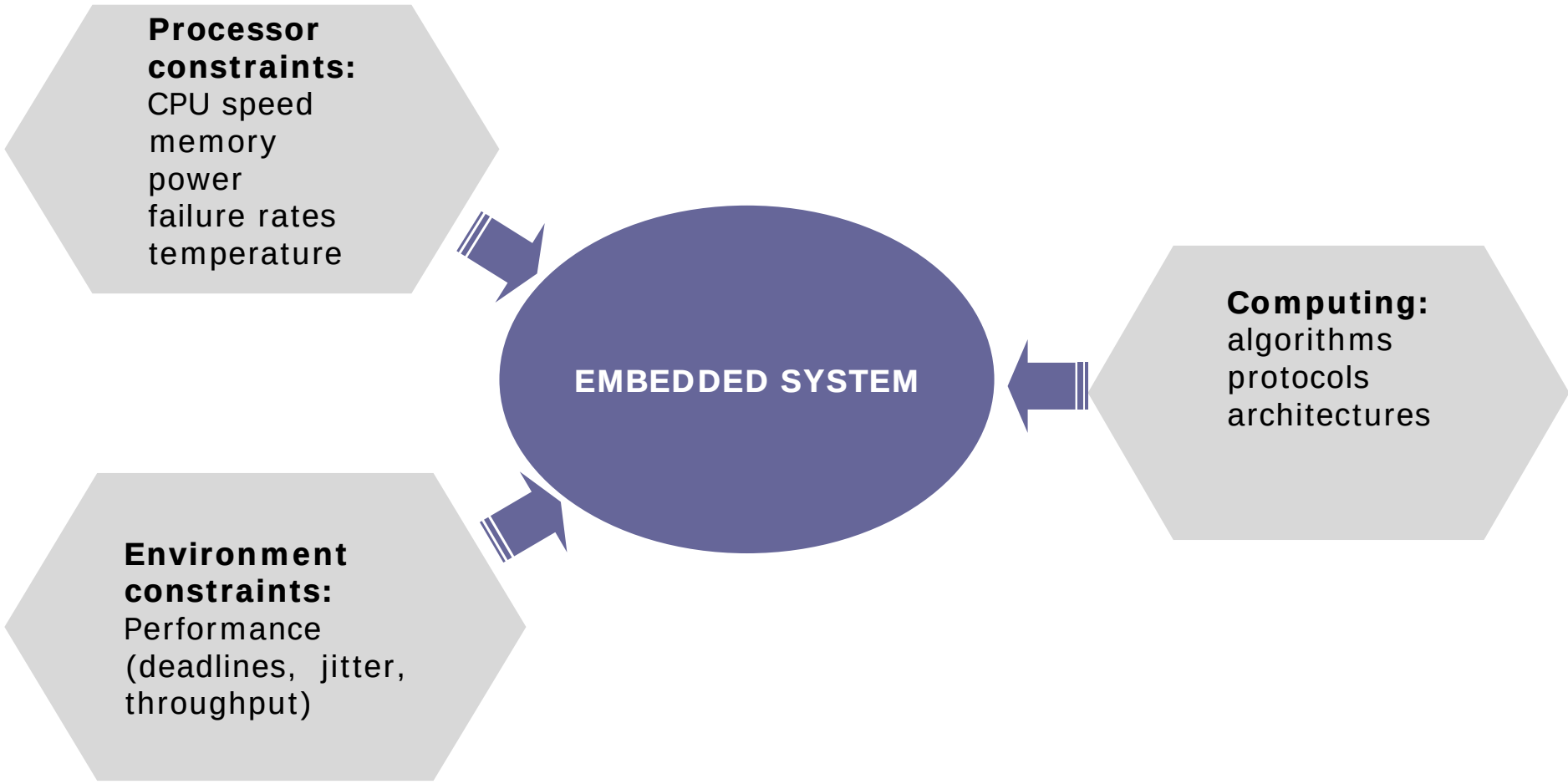
Applications – Smart Grids



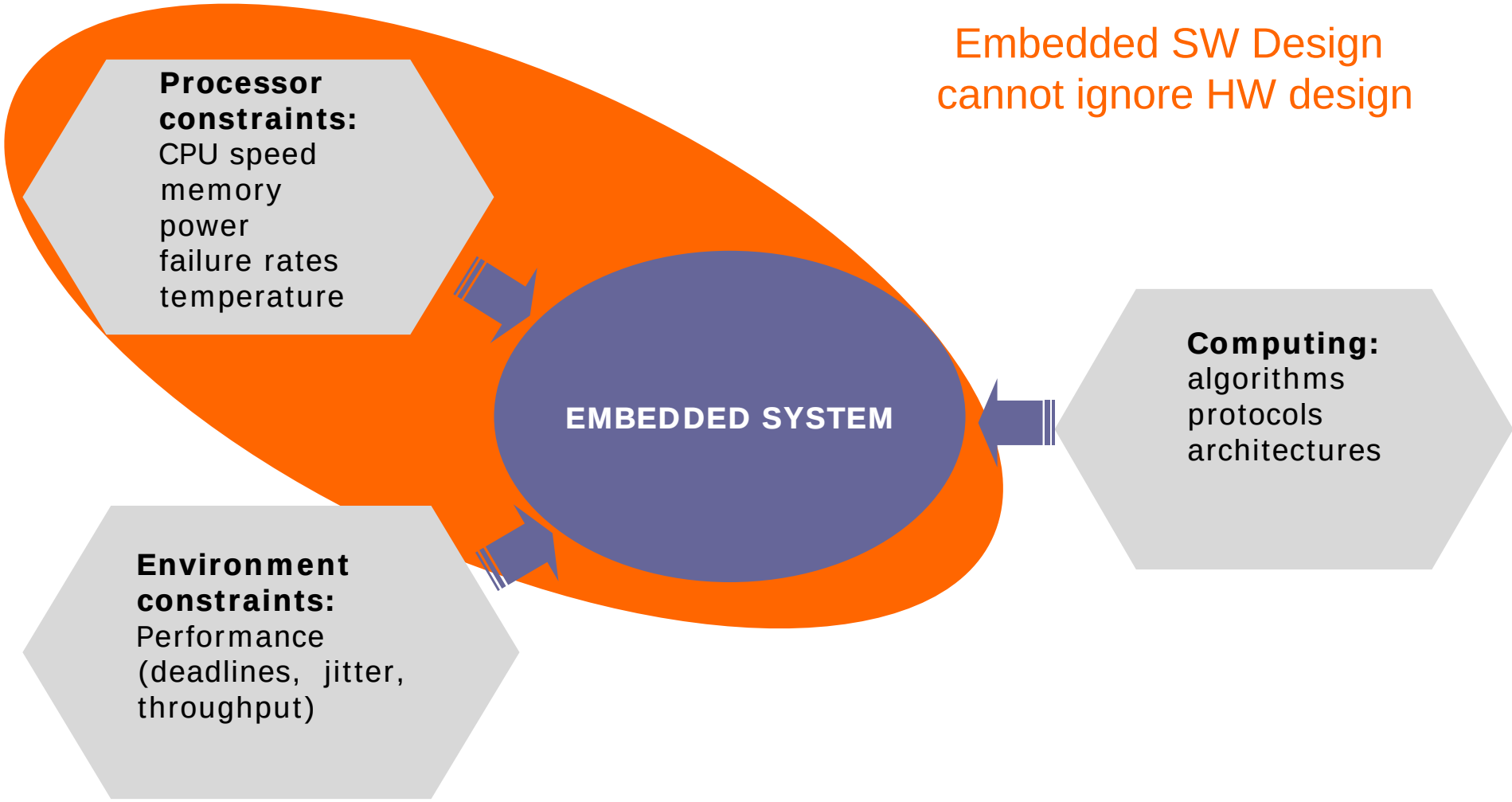
OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

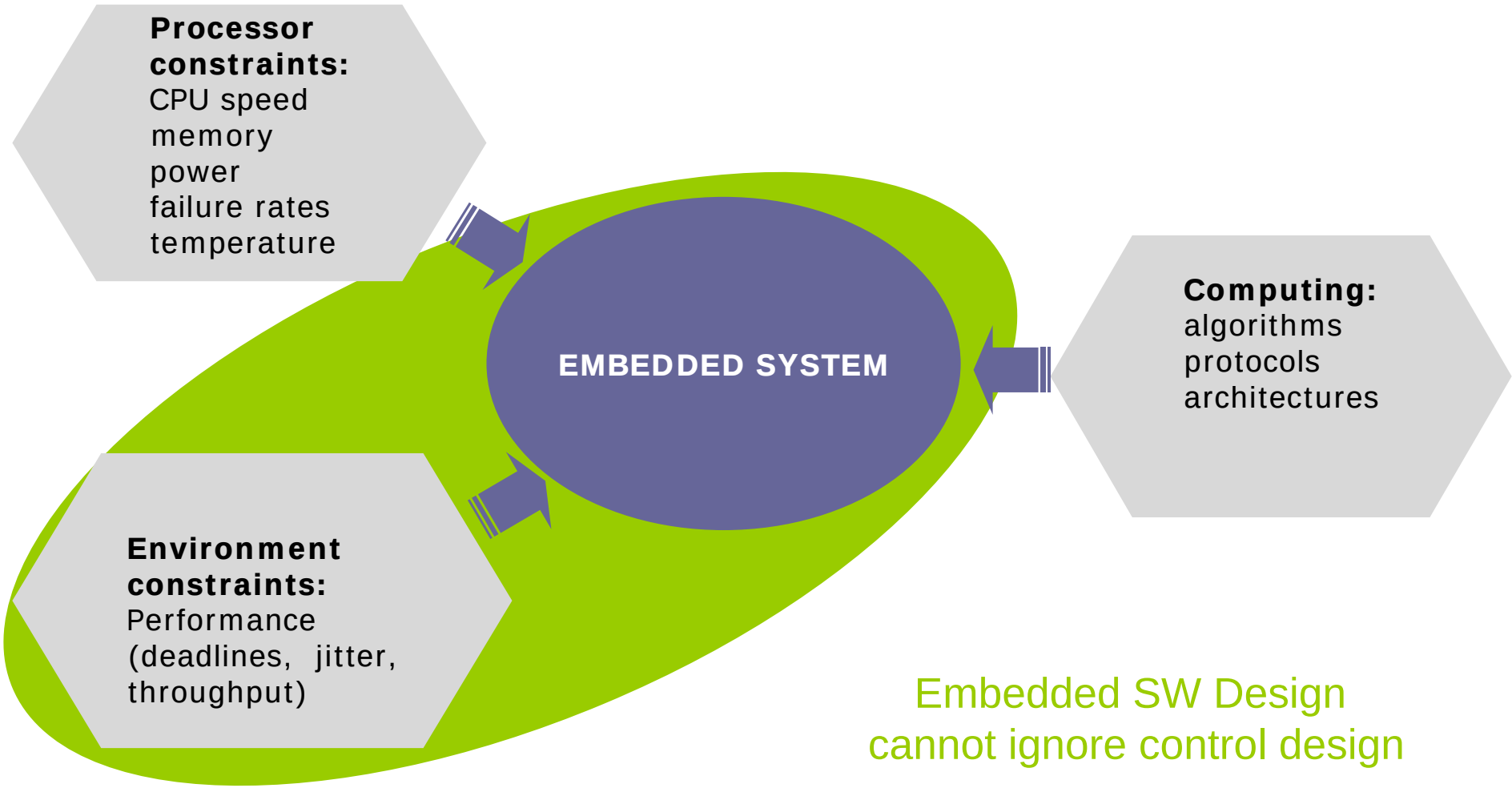
Marry Physicality and Computation



Marry Physicality and Computation



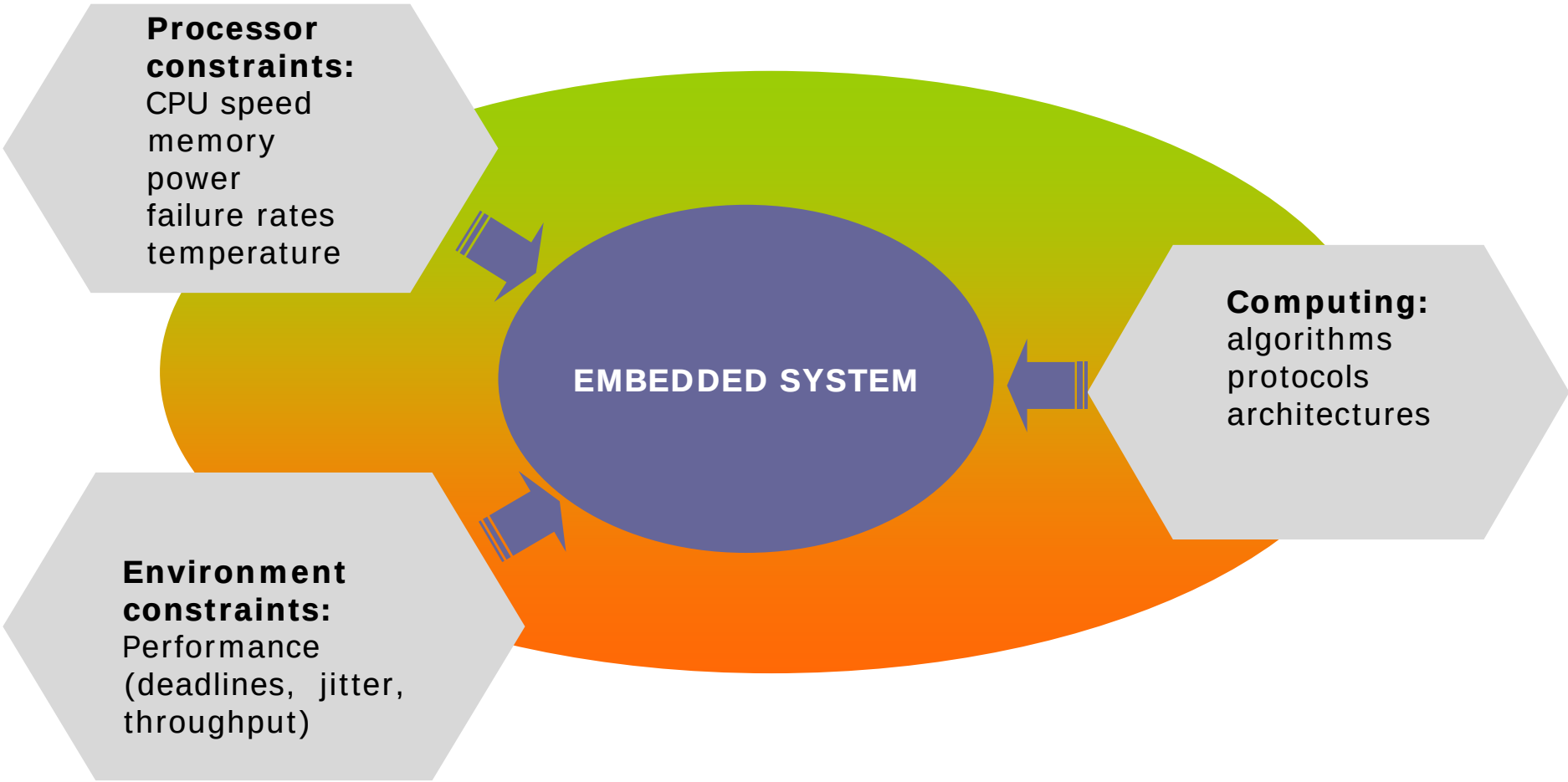
Marry Physicality and Computation



Embedded SW Design
cannot ignore control design

Marry Physicality and Computation

Embedded SW Design coherently integrates all these



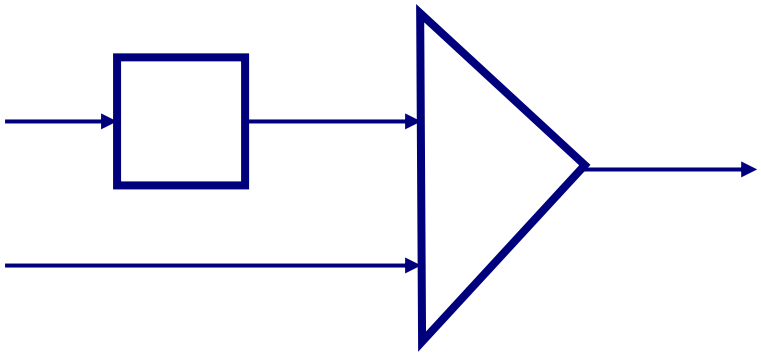
We need to revisit and revise computing to integrate methods from EE and Control

Marry Physicality and Computation

Physical Systems Engineering

Analytic Models

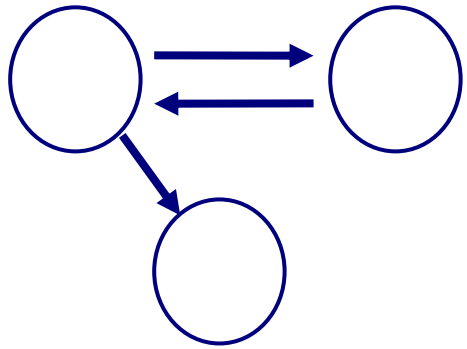
Component: transfer function
Composition: parallel
Connection: data flow



Computing Systems Engineering

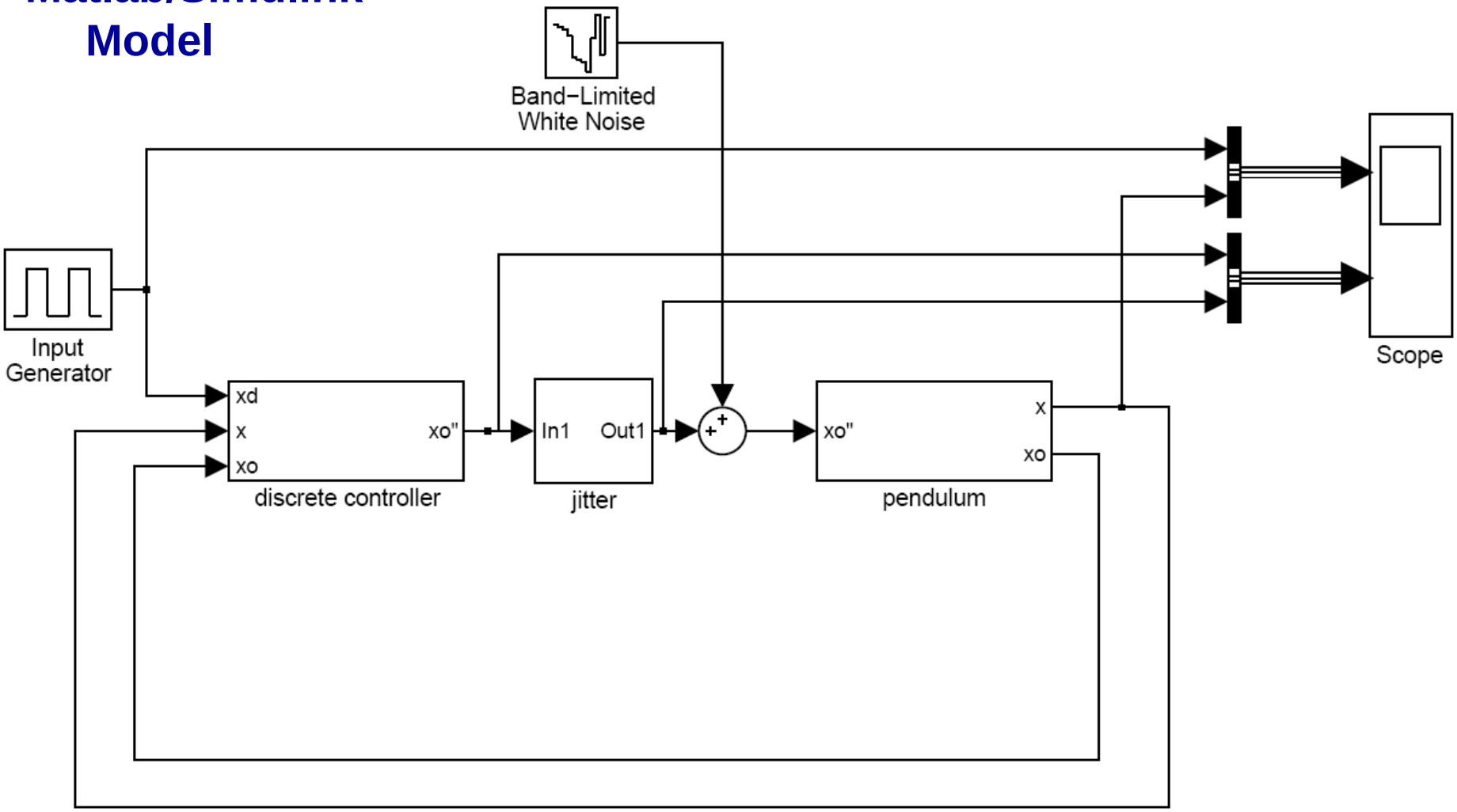
Computational Models

Component: subroutine
Composition: sequential
Connection: control flow



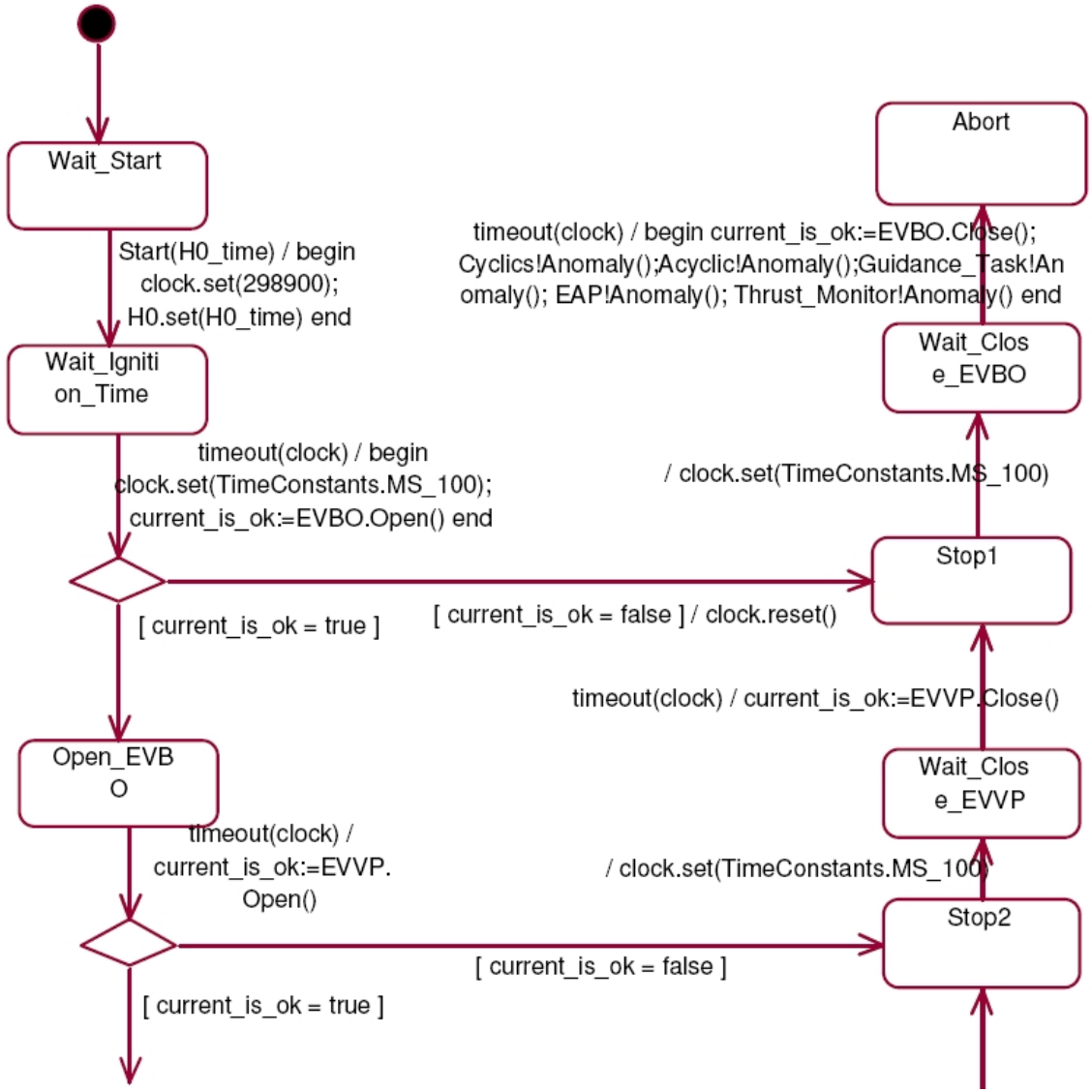
Marry Physicality and Computation

Matlab/Simulink Model



Marry Physicality and Computation

UML Model
(Rational Rose)



OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Encompass Heterogeneity – Unified Composition Paradigm

Composition lies at the heart of the parallel computing challenge.
Without sane composition, there can be no reuse.

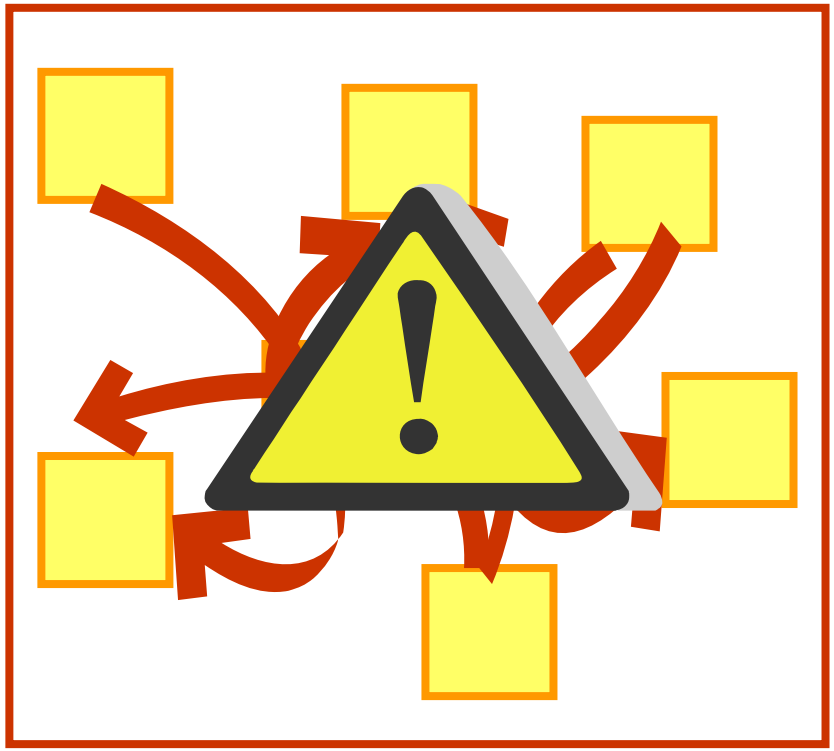
- SW Component frameworks:
  - Coordination languages extensions of programming languages e.g. BPEL, Javaspaces, TSpaces, Concurrent Fortran, NesC
  - Middleware e.g. Corba, Javabeans, .NET
  - Software development environments: PCTE, SWbus, Softbench, Eclipse
- System modeling languages: Statecharts, SysML, Matlab/Simulink, AADL, Ptolemy
- Hardware description languages: Verilog, VHDL, SystemC

Heterogeneity: Embedded systems are built from components with different characteristics

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

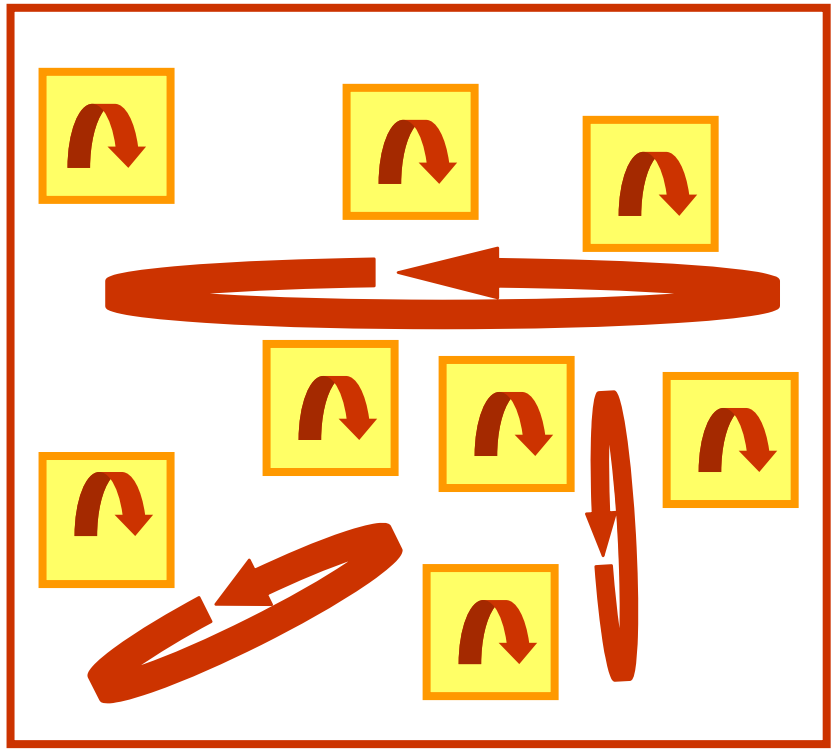
Encompass Heterogeneity – Unified Composition Paradigm

Thread-based programming



Software Engineering

Actor-based programming

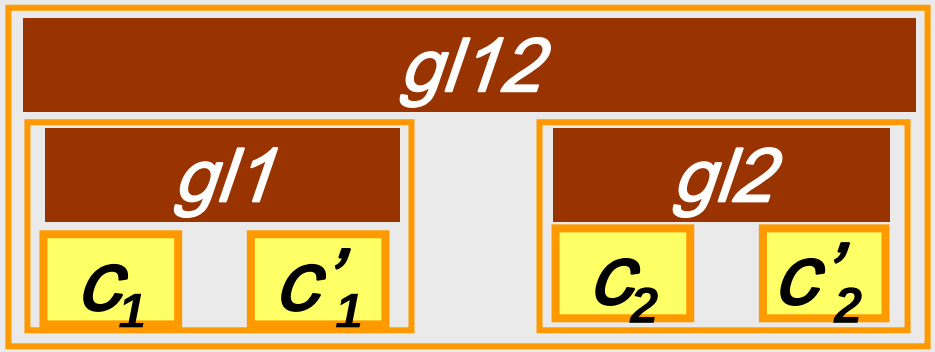


Systems Engineering

Encompass Heterogeneity – Unified Composition Paradigm

Build a component *C* satisfying given requirements *f, from*

- C_0 a set of **atomic** components described by their behavior
- *GL* $=\{gl_1, \ldots, gl_i, \ldots\}$ a set of **glue *operators*** on components



satisfies f

- Move from single low-level composition operators e.g. automata-based to families of high-level composition operators e.g. protocols, controllers

- We need a unified composition paradigm for describing and analyzing the coordination between components to formulate system designs in terms of tangible, well-founded and organized concepts

OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Cope with Complexity – Constructivity

- Today, *a posteriori* verification at high development costs limited to medium complexity systems
- Tomorrow, correct-by-construction results should advantageously take into account structuring of components and their features.

There is a large space to be explored, between full constructivity and *a posteriori* verification. Develop correct-by-construction results

- For particular
  - architectures (e.g. client-server, star-like, time-triggered)
  - programming models (e.g. synchronous, data-flow)
  - execution models (e.g. event triggered preemptable tasks)
- For specific classes of properties such as deadlock-freedom, mutual exclusion, timeliness

Constructivity – Compositionality

Build correct systems from correct components: rules for proving global properties from properties of individual components

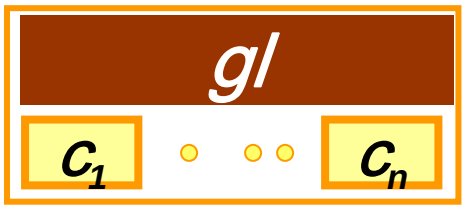
c_i sat P_i implies $\forall gl\ \exists \widetilde{gl}$

sat $\widetilde{gl}(P_1, .., P_n)$

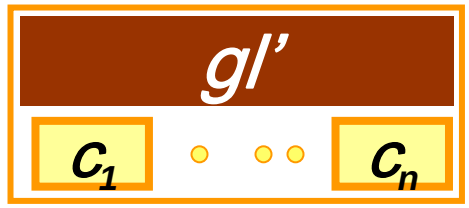
We need compositionality results for the preservation of progress properties such as deadlock-freedom and liveness as well as extra-functional properties

Constructivity – Composability

Essential properties of components are preserved when they are integrated



sat P and



sat P'

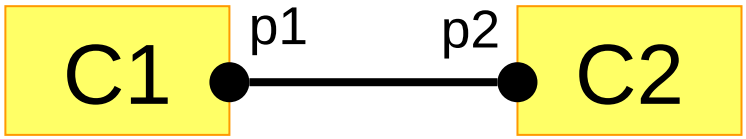
implies

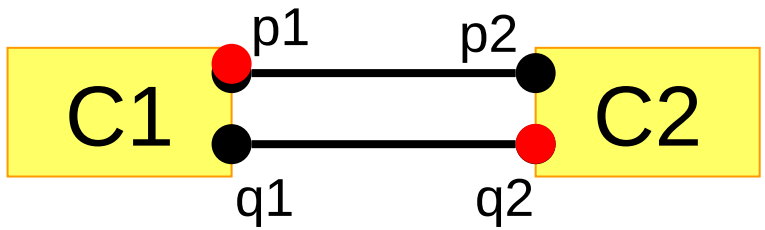
sat P∧P'

Property stability phenomena are poorly understood.
We need composability results e.g. non interaction of features in middleware, composability of scheduling algorithms, of Web services, of aspects

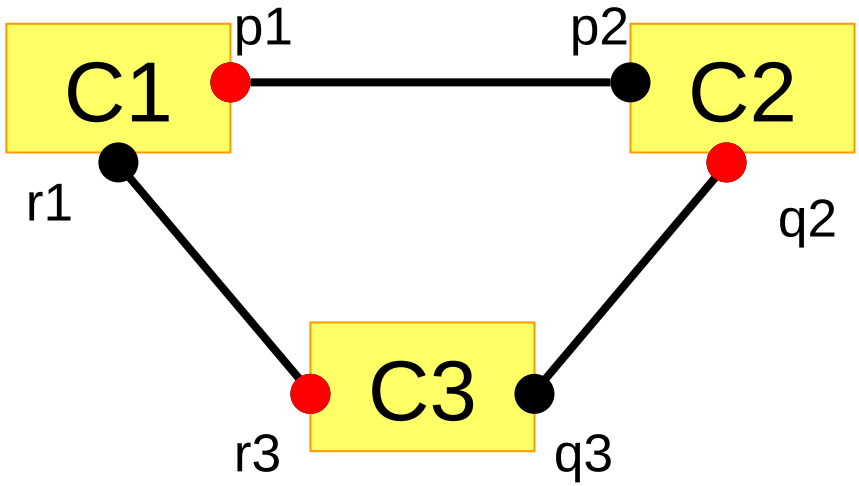
Constructivity – Checking for Deadlock-freedom

Checking global deadlock-freedom of a system
built from deadlock-free components,
by separately analyzing the components and the architecture.





Potential deadlock
D = en(p1) ∧ ¬ en(p2) ∧
en(q2) ∧ ¬ en(q1)



Potential deadlock
D = en(p1) ∧ ¬ en(p2) ∧
en(q2) ∧ ¬ en(q3) ∧
en(r3) ∧ ¬ en(r1)

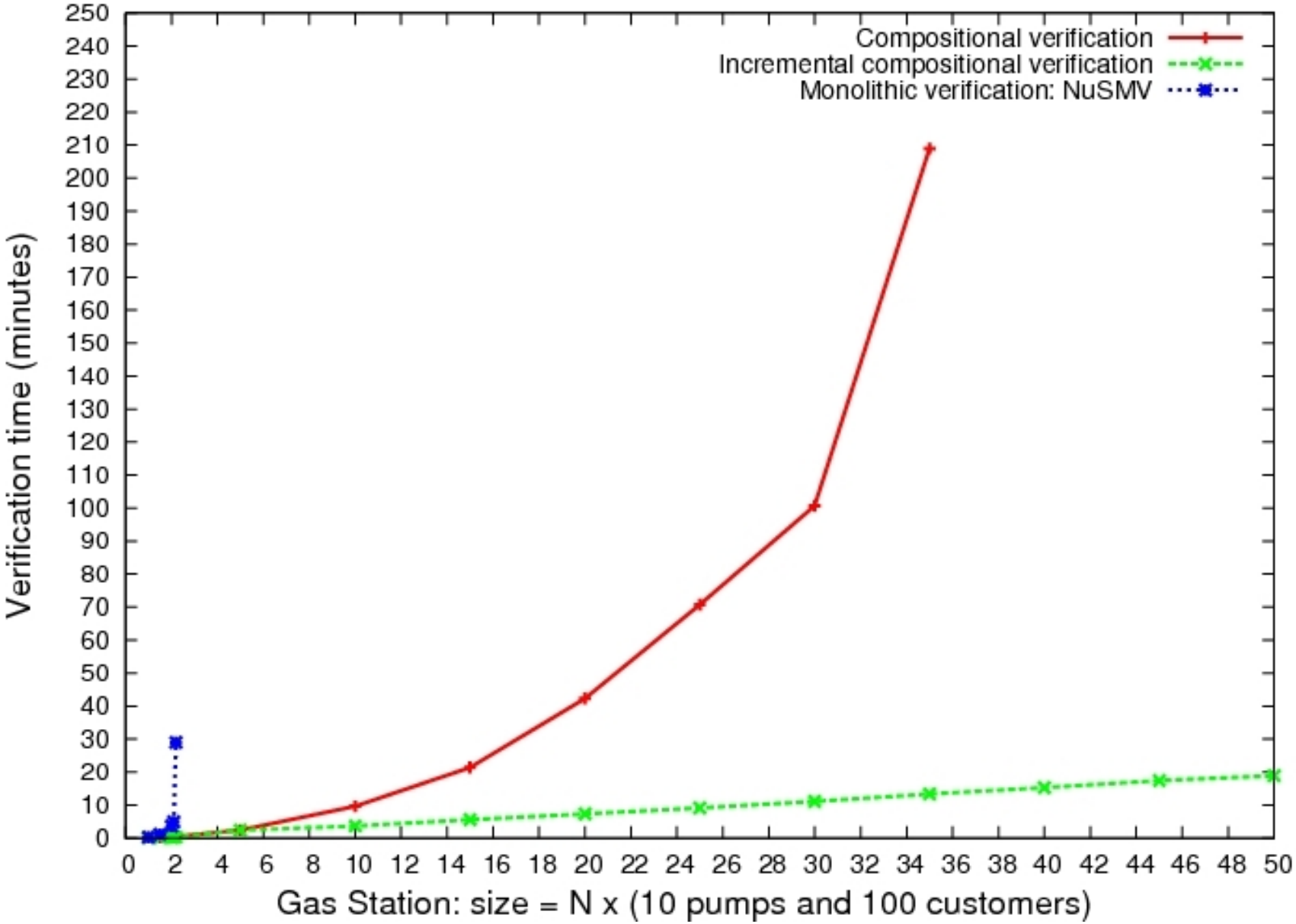
Constructivity – Checking for Deadlock-freedom

Eliminate potential deadlocks D by checking that
$I \wedge D = false$
where I is a global invariant computed compositionally

Example	Nb Comp	Nb Ctrl St	Nb Bool Var	Nb Int Var	Nb Pot Deadl	Nb Rem Deadl	time
Temperature Control (2 rods)	3	6	0	3	8	3	3s
Temperature Control (4 rods)	5	10	0	5	32	15	1m05s
UTOPAR (300 cars, 900 CUs)	1201	3903	300	1204	??	0	7m48s
UTOPAR (200 cars,1600 CUs)	1801	4603	200	804	??	0	17m46s
Philos (9000 philos)	18000	53000	0	0	??	1	24m16s
ATM (600 atms)	1202	13204	0	1200	??	0	24m37s
Gas Station (700 pumps+7000 customers)	7701	30102	0	0	??	0	7m14s

Results obtained by using the D-Finder tool: http://www-verimag.imag.fr/~thnguyen/tool/

Constructivity – Checking for Deadlock-freedom



OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Cope with Uncertainty - Adaptivity

Systems must provide a service meeting given requirements in interaction with uncertain and unpredictable environments

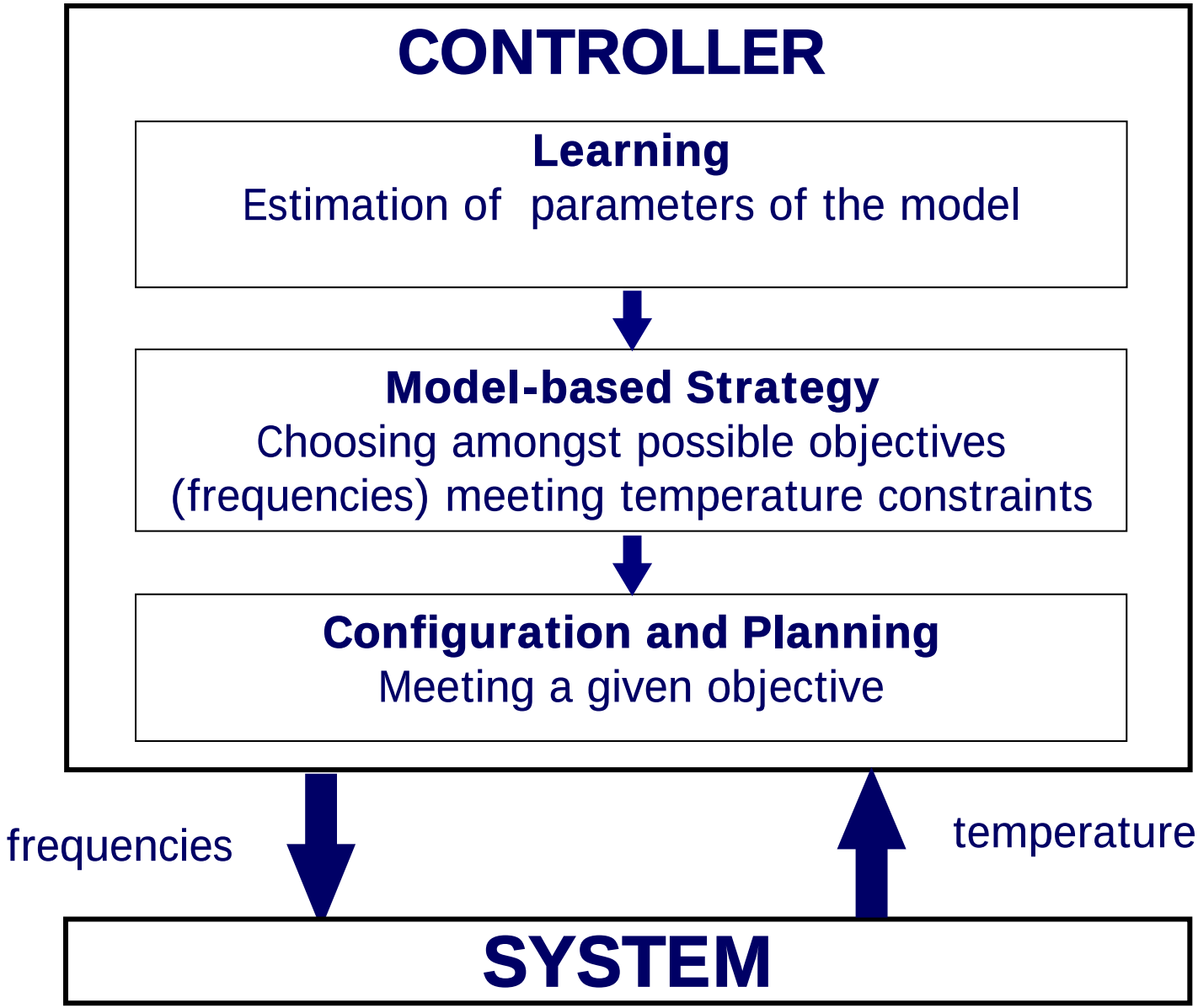
Uncertainty is characterized as the difference between average and worst-case system behavior. It is drastically increasing due to:

- Interaction with complex, non-deterministic, possibly hostile external environments
- Execution platforms with sophisticated HW/SW architectures (layering, caches, speculative execution, …)

For complex systems it is impossible to foresee at design time by a case-by-case analysis all the potentially critical situations

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.

Cope with Uncertainty – Adaptivity



Cope with Uncertainty – Adaptivity

Learning

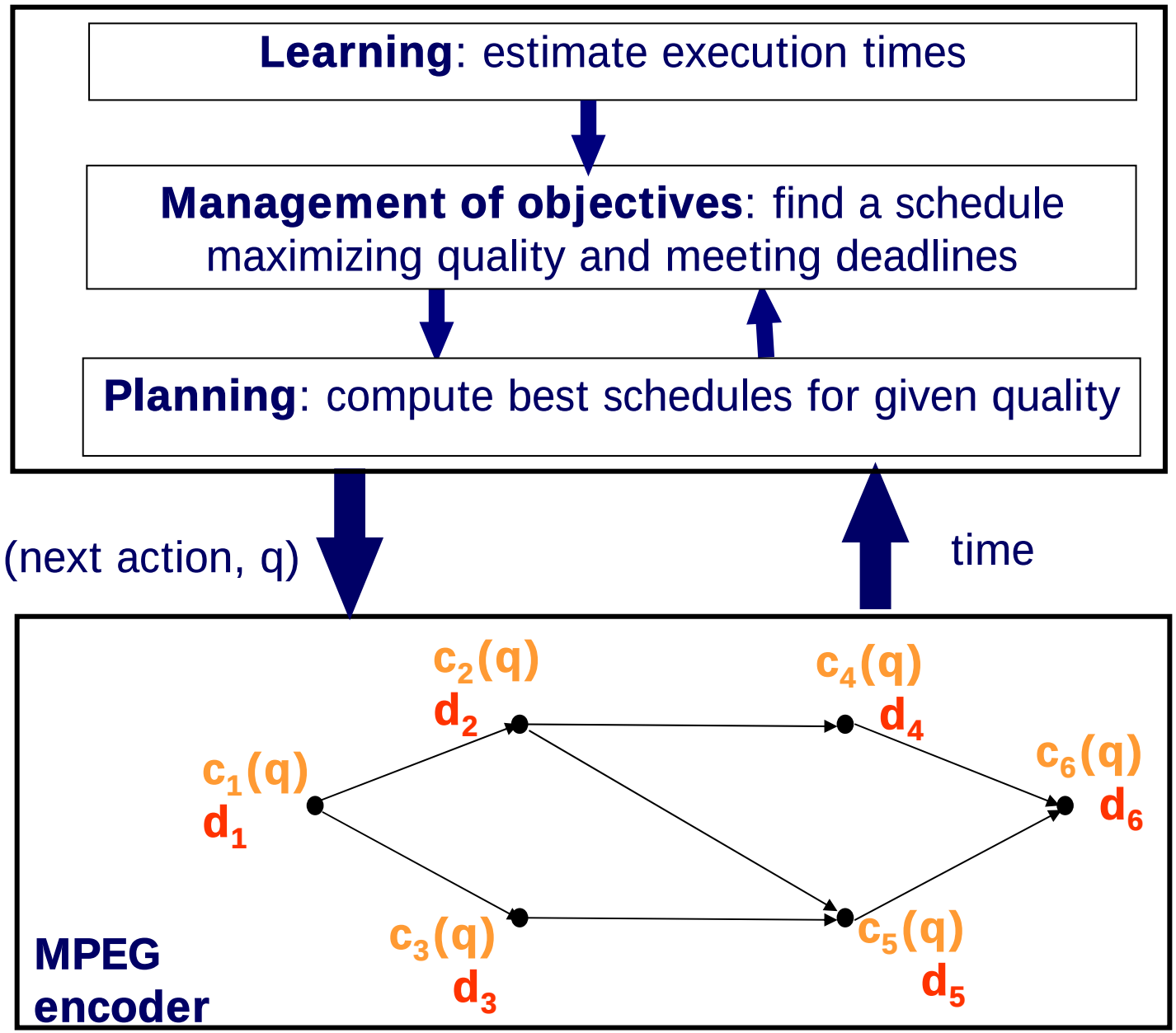
Movie would have been better …

Strategy and decision making

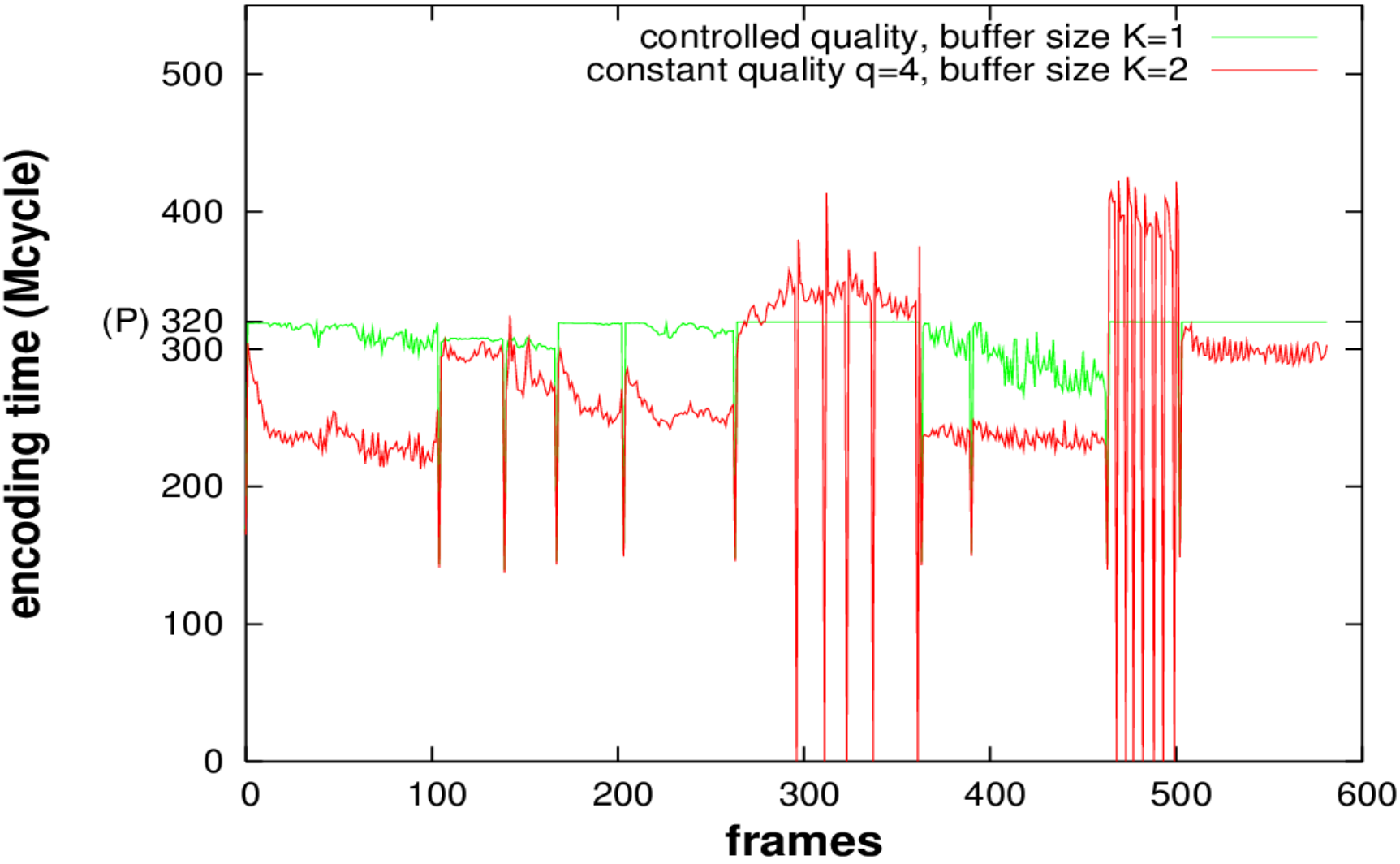
Go to: 1) Stadium 2) Movie 3) Restaurant

Planning

Cope with Uncertainty – Adaptivity



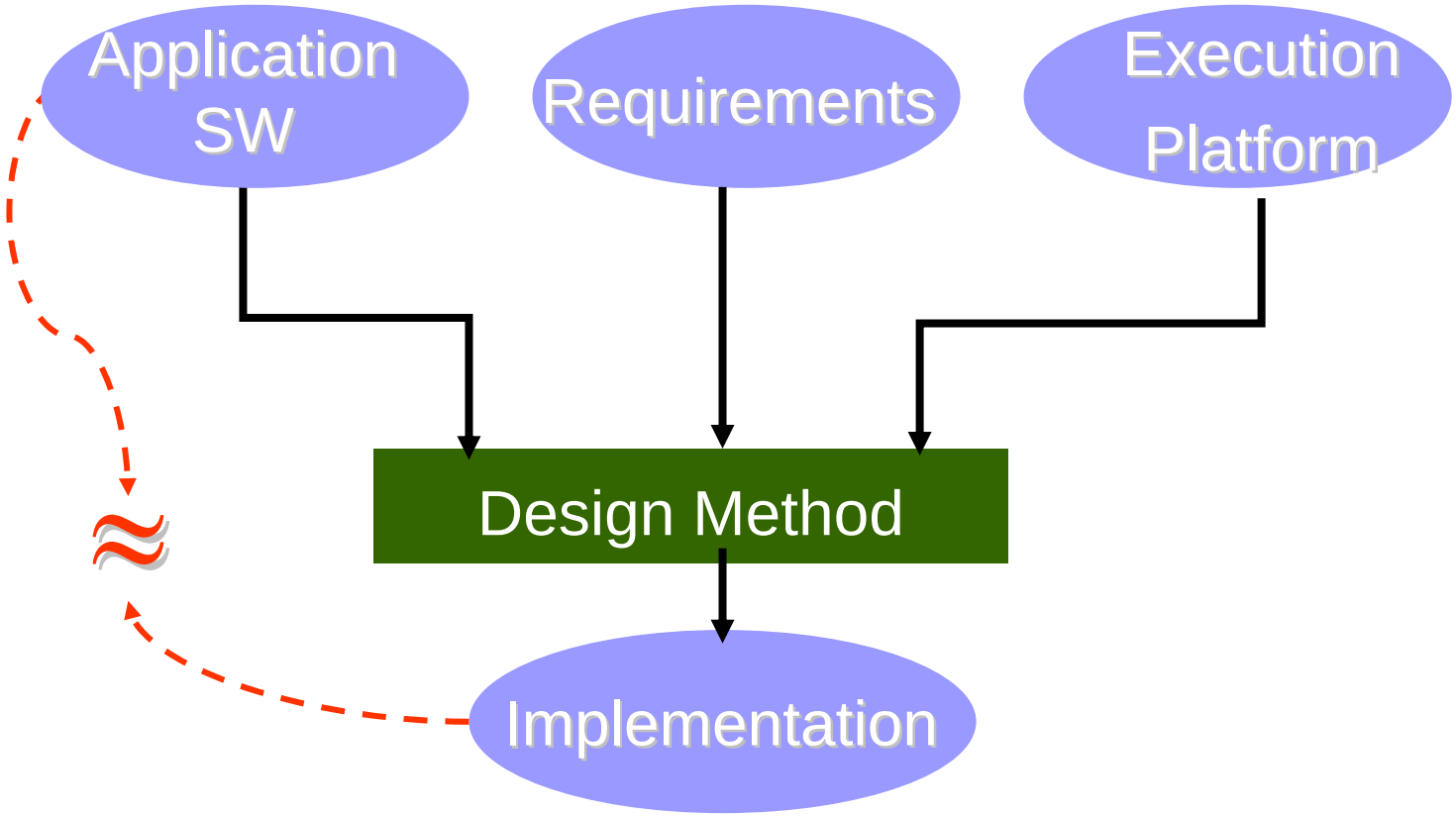
Cope with Uncertainty - Adaptivity



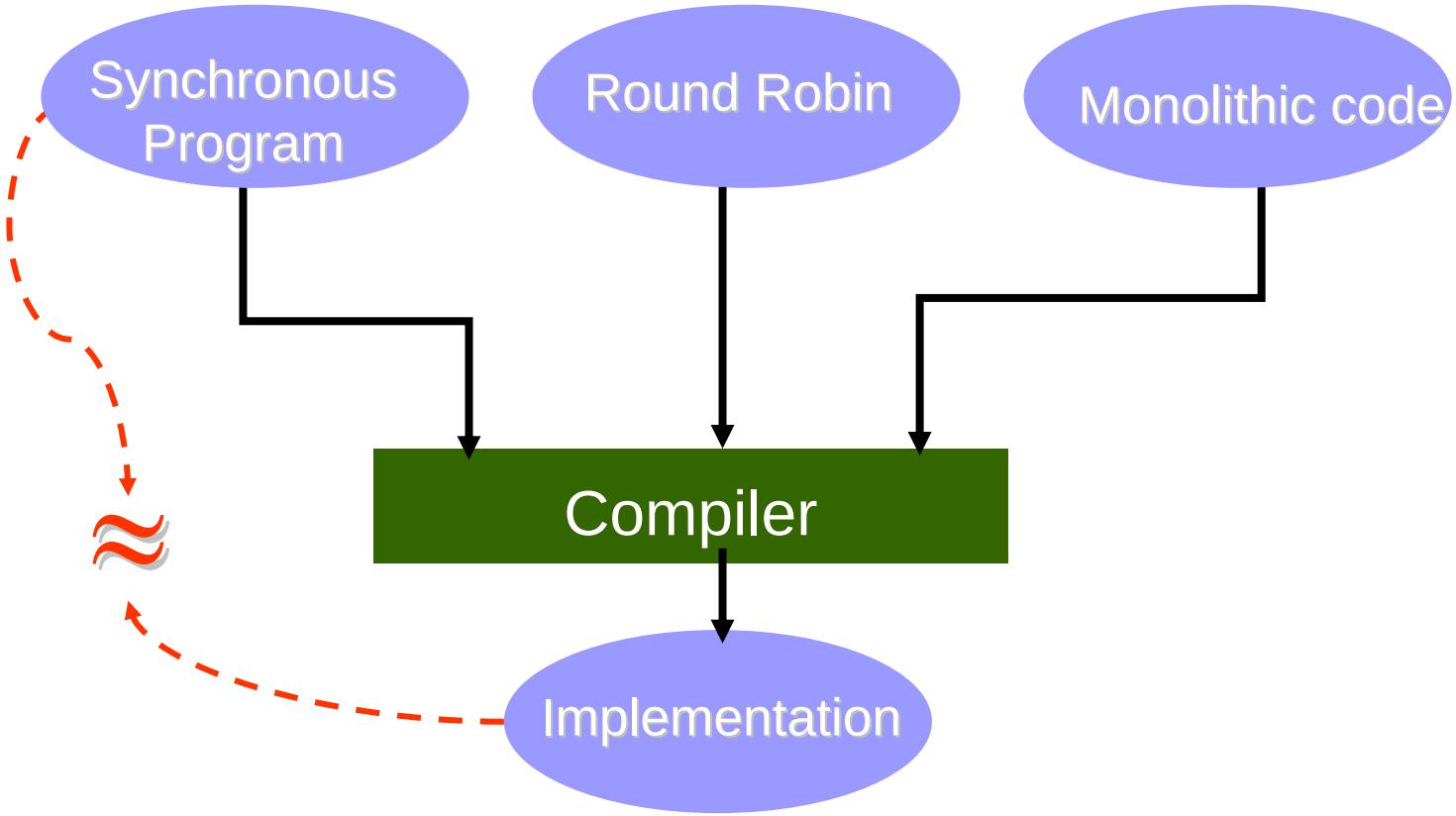
OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

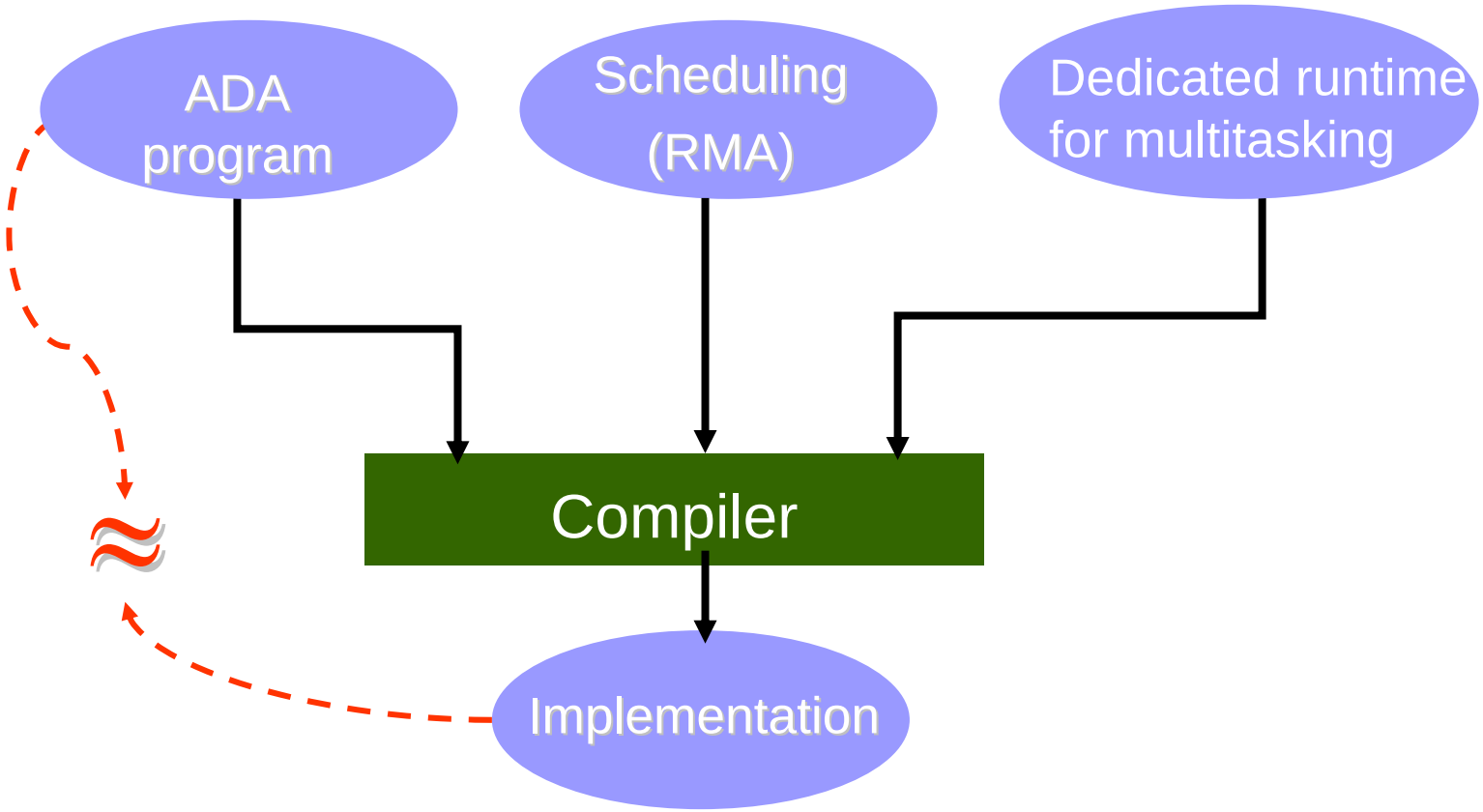
Model-based Design - Principle



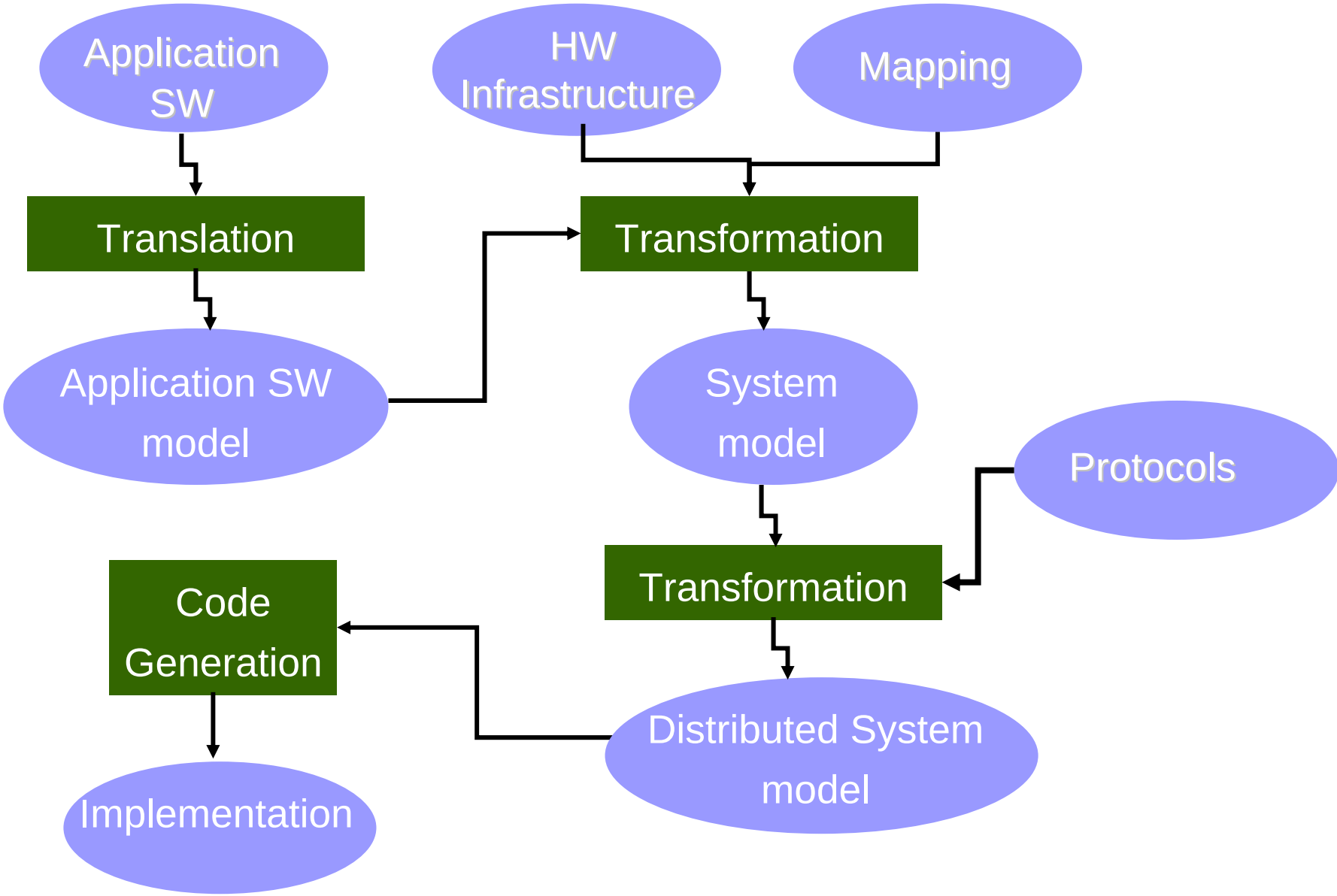
Model-based Design - Example



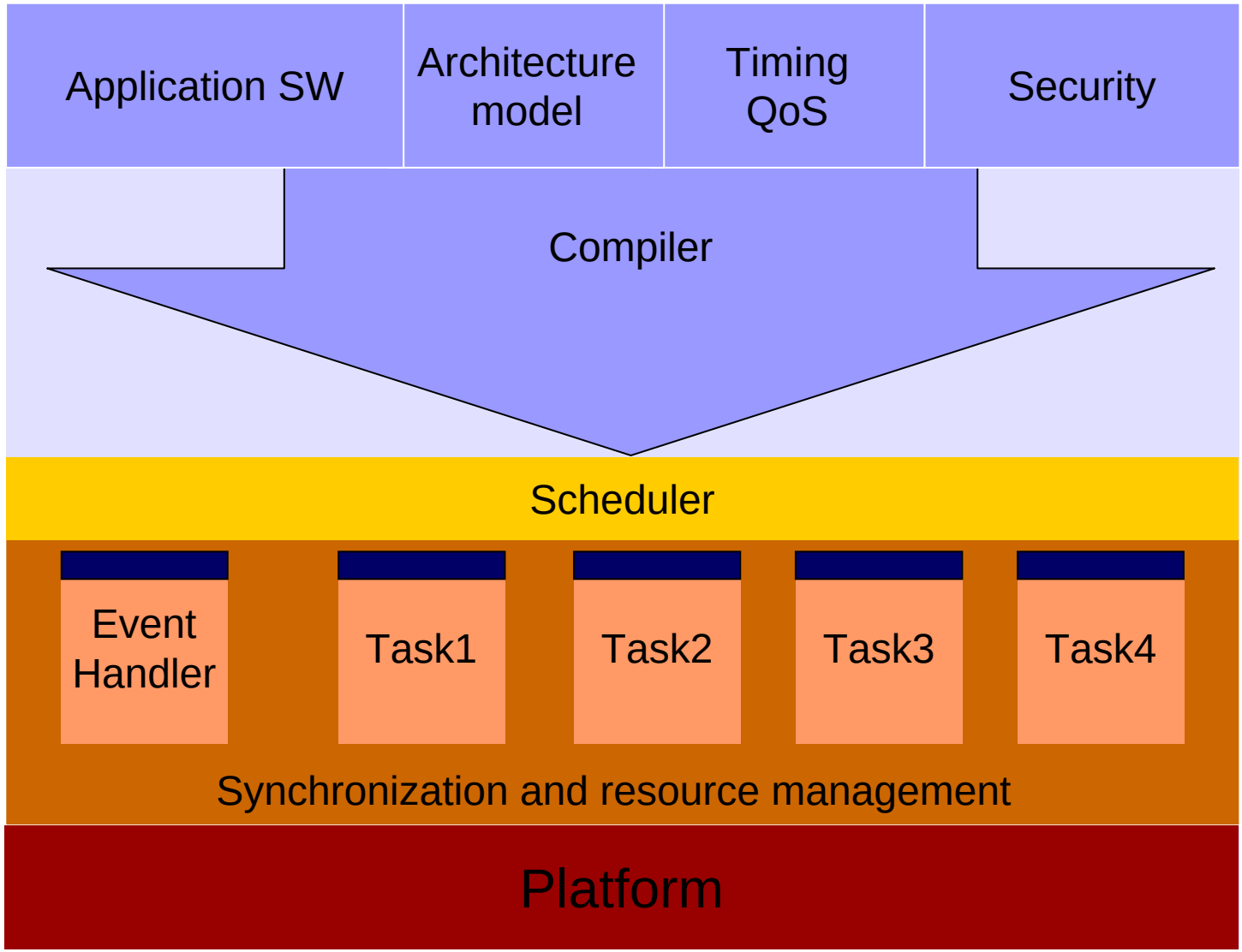
Model-based Design - Example



Model-based Design Flow



Resource-aware Compilation



Operating Systems

Operating systems are often:

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

Move towards standards dedicated to specific domains
Ex: OSEK, ARINC, JavaCard, TinyOS, Symbian

- 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, computing)
- Modeling and Verification
- Distributed and fault-tolerant implementations (influence communication delays, clock drift, aperiodic sampling)

↳ Use of control-based techniques for adaptivity

Dependability

- 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, ...

Work Directions :

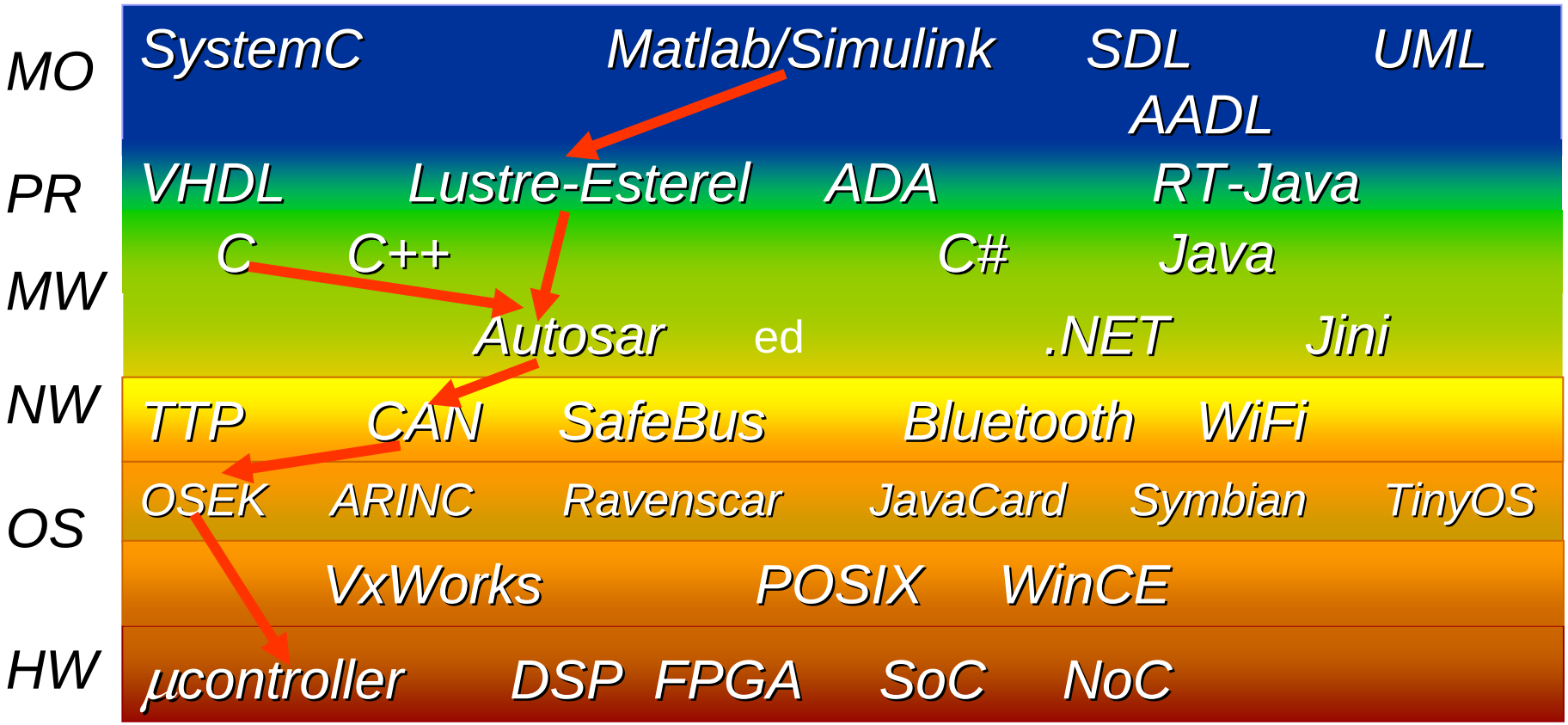
- Methodologies for domain-specific standards, such as :
  - DO-178B Process Control Software Safety Certification
  - Integrated Modular Avionics; Autosar
  - Common Criteria for Information Technology Security Evaluation
- Certification methods and tools
- Architectures, protocols and algorithms for fault-tolerance and security taking into account QoS requirements (real-time, availability)

Networked Embedded Systems

Adaptive distributed real-time systems, 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



OVERVIEW

- System Design Today
- Basic Technologies
- Applications
- Research Challenges
  - Marry Physicality and Computation
  - Encompass Heterogeneity – Unified Composition Paradigm
  - Cope with Complexity - Constructivity
  - Cope with Uncertainty – Adaptivity
- Work Directions
- Discussion

Discussion

Embedded Systems

- ❑ break with traditional Systems Engineering. They need new design techniques guaranteeing both functionality and quality (performance and dependability) and taking into account market constraints
- ❑ are an opportunity for reinvigorating and extending Informatics with new paradigms from Electrical Engineering and Control Theory

We need methods and supporting unified model-based design flows for **productivity**, **correctness** and **performance**

- Programmability: Parallel programming supporting heterogeneous component-based programming models
- Adaptivity: SW should take into account HW variability
- Performance: choose SW partitioning, data management and interaction model so as to reduce communication overhead
- "Magic Compilation Chain": automatic SW deployment techniques

THANK YOU