Six Seminars on System Design

Joseph Sifakis
RiSD Laboratory, EPFL
About the Seminars – Aims

- Explain why system design differs from pure software design as it should jointly take into account both functional and extra-functional requirements.

- Discuss theoretical problems raised by the formalization of system as a rigorous process leading from application software to correct implementations.

- Explain why awareness on centrality of system design is a chance to reinvigorate research in Computer Science and build novel scientific foundations matching the needs for increasing system integration and new applications.

- Occasionally discuss some more foundational issues about the nature of computing and its relations to other scientific disciplines.
About the Seminars – Program

Seminar 1: System Design and the Concept of Correctness, Friday October 5

Seminar 2: System Modeling, Friday October 19

Seminar 3: Correctness-by-checking, Friday November 9

Seminar 4: Correctness-by-construction, Friday November 23

Seminar 5: Coping with Uncertainty – Adaptivity, Friday December 7

Seminar 6: Putting System Design into Practice, Friday November 21
Seminar 1: System Design and the Concept of Correctness

Joseph Sifakis
RiSD Laboratory, EPFL
The Evolution of IST

- Scientific Computing – Defence Applications
  - Convergence between Computing and Telecommunications
  - Graphic Interfaces, Mouse

- WEB – Information Society
  - Seamless revolution
    - 95% of chips are embedded

- Embedded Systems: Computing + Physicality
  - 95% of chips are embedded

- Information Systems: Commercial Applications
  - Integrated circuits

- Multi-core Systems

- Cloud Computing

- The Internet of Things: Convergence between Embedded Systems and the Internet

Evolution driven by exponential progress in technology and explosion of applications
Systems integrating software and hardware jointly and specifically designed to provide given functionalities, which are often critical.
Moore’s Law: the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years.
The Embedded Systems Explosion

Exploding annual microprocessor* shipments

the fast growing market for embedded systems

Year

Source: Royal Philips Electronics
System technologies

- have a tremendous economic and societal impact
- they allow addressing global challenges for growth, health, services and innovation

IBM’s view for a Smarter Planet

Systems are

- **Instrumented**
  The state of almost everything can be measured, sensed and monitored
- **Interconnected**
  People and objects can communicate and interact with each other in entirely new ways
- **Intelligent**
  Enhanced predictability of events and optimal use of resources
## Reactive Systems – The Hardest and the Most Important

### Significant differences

<table>
<thead>
<tr>
<th>Transformational Software</th>
<th>Reactive System</th>
</tr>
</thead>
<tbody>
<tr>
<td>terminating</td>
<td>non-terminating</td>
</tr>
<tr>
<td>deterministic</td>
<td>non-predictable</td>
</tr>
<tr>
<td>platform-independent</td>
<td>platform-dependent</td>
</tr>
<tr>
<td>Theory</td>
<td>No theory!</td>
</tr>
</tbody>
</table>

### Reactive systems

- are hard to design due to unpredictable and subtle interactions with the environment, emergent behaviors, and occasional catastrophic cascading failures rather than to complex data and algorithms

- are increasingly important in modern computing systems: embedded systems, cyber-physical systems, mobile systems, web-services
From Programs to Systems

SW+HW

Real-Time Control

Resources
Buildings
Transport
Communications
Health
System Design

Basic Technologies

Future Applications

A System-centric Vision for CS
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 – Status and Vision

**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 in areas such as transport, health, energy management

- Successful integration of heterogeneous mixed criticality systems
  - *Internet of Things*
  - *Intelligent Transport Systems*
  - *Smart Grids*
  - « Ambient Intelligence»

This vision is currently unattainable – Trustworthiness should be a guiding concern throughout the design process
System Design – Next generation Air Traffic Control

Start of the project

1984

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.

The FAA's Course Correction

April 9th, 2002

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 fiascos 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.
Traditional systems engineering disciplines are based on solid theory for building artefacts with predictable behaviour over their life-time.

Computing systems engineering lacks similar constructivity results

- only partial answers to particular design problems
- predictability is hard to guarantee at design time
- *a posteriori* validation remains essential for ensuring correctness
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.
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.

One of the central aims of the Programming Research Group as a teaching and research group has been to set up an atmosphere in which this separation cannot happen.

Christopher Strachey (1916-1975)
Design is a **universal concept**, a par excellence intellectual activity leading to artifacts meeting given requirements.

**Easy Apple Pie**

- Put apples in pie plate;
- Sprinkle with cinnamon and 1 tablespoon sugar;
- In a bowl mix 1 cup sugar, flour and butter;
- Blend in unbeaten egg, pinch of salt and the nuts;
- Mix well and pour over apples;
- Bake at 350 degrees for 45 minutes

**INGREDIENTS (Resources)**
- 1 pie plate buttered
- 5 or 6 apples, cut up
- ¾ c. butter, melted
- 1 c. flour
- ½ c. chopped nuts
- 1 tsp cinnamon
- 1 tbsp sugar
- 1 c. sugar
System Design – Two Main Gaps

Correctness?

Requirements (declarative)

Application SW (executable)

System (HW+SW)
System Design – Productivity vs. Correctness

System designers strive to reconcile two often conflicting demands:

- **Productivity** characterizes the efficiency of the design process.
- **Correctness** means compliance to requirements.

As flawless system design is not attainable, owing to both theoretical limitations and cost-effectiveness considerations, system designers target levels of criticality.

**Levels of criticality**

- Correspond to tradeoffs between correctness and productivity.
- Determine which types of requirements are relevant and to what extent these requirements should be met e.g. probability of failure or disparity between nominal and observed values of significant parameters.
**System Design – Levels of Criticality**

- **Safety critical**: a failure may be a catastrophic threat to human lives.
- **Security critical**: harmful unauthorized access.
- **Mission critical**: system availability is essential for the proper running of an organization or of a larger system.
- **Best-effort**: optimized use of resources for an acceptable level of trustworthiness.
System Design – Productivity

Efficiency of the design process

Skills

Tools

Components
System Design – Productivity vs. Correctness

Design process 1

Design process 2

Productivity vs. Correctness
System Design – Requirements

Functional requirements characterize the services ensured for potential users. These are independent of the resources of the execution platform, e.g.

- The program terminates and computes a given function
- The temperature is always between 18° and 20°
- When train is crossing the gate is down
- A sent message is eventually received

Extra-functional requirements characterize the quality of the services. These take into account the resources of the execution platform:

- Performance
  - Throughput is not less than 100 Mb/s (for a network)
  - Power needed is less than 2mW (for a circuit)
  - Image quality is optimal (for an encoder)
- Security: Resistance to attacks (for a server)
- Safety: Resistance to failures (for a flight controller)
Trustworthiness requirements express assurance that the designed system can be trusted that it will perform as expected despite

- HW failures
- Design Errors
- Environment Disturbances
- Malevolent Actions

Optimization requirements dealing with optimization of functions subject to quantitative constraints on

1) **Performance**: how well the system does wrt user demands e.g. throughput, jitter, latency, quality.
2) **Cost**: how well resources are used wrt economic demands e.g. storage efficiency, processor utilizability, energy efficiency.

Usually they determine tradeoffs between performance and cost.
System Design – Trustworthiness vs. Optimization

- Trustworthiness requirements characterize qualitative correctness – a state is either trustworthy or not
- Optimization requirements characterize execution sequences

Trustworthiness vs. Optimization
The two types of requirements are often antagonistic: trustworthy designs give rise to non optimized solutions and conversely
System Design

Basic Technologies

Future Applications

A System-centric Vision for CS
A multicore system integrates many cores (processors) on a chip.

- The switch to multicore systems
  - 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.
Gartner, Research Note [09]: “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."

Intel [06]: “Multi-core processing is taking the industry on a fast moving and exciting ride into profoundly new territory. The defining paradigm in computing performance has shifted inexorably from raw clock speed to parallel operations and energy efficiency.”

Microsoft Research [07]: “Multicore processors represent one of the largest technology transitions in the computing industry today, with deep implications for how we develop software.”
Basic technologies – Sensors

- Accelerometer
- Gyroscope
- Magnetometer
- Proximity sensor
- Digital Microphone
- Keypad
- Image Sensor
- Touch Screen
- GPS
- Fingerprint Sensor
- Pressure Sensor
- Light Sensor
- Temperature Sensor
Basic technologies – Sensor Networks

- Microphone
- Sounder
- Magnetometer
- Temperature Sensor
- Light Sensor
- Accelerometer

Dimensions:
- 1.25 in
- 2.25 in
**Internet-based Systems – Security**

**Security**: protection of information from unauthorized access, use, disclosure, disruption, modification or destruction.

**Key requirements:**

- **Confidentiality** is the property of preventing disclosure of information to unauthorized individuals or systems
e.g. for credit card transactions

- **Integrity** means that data cannot be modified without authorization
e.g. resistance to viruses

- **Availability**, that is information must be available when it is needed
e.g. preventing denial-of-service attacks.

- **Authenticity**, that is ensuring that the transactions, communications and data are genuine.

- **Non-repudiation** that is ensuring that a party in a dispute cannot repudiate, or refute the validity of a statement or contract
e.g. electronic check or contract.
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 ?
- System Design
- Basic Technologies
- Future Applications
- A System-centric Vision for CS
Applications – Smart Transportation Systems

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.
Applications – e-Health

- Brain monitoring
- Robot-assisted surgery
- Blood pressure meter
- ECG
- Thermometer
- Remote Medical Assistant
- System Design
- Basic Technologies
- Future Applications
- A System-centric Vision for CS
Computer Science

- Is a young and rapidly evolving discipline due to exponential progress of technology and applications

- Focuses on building systems and thus system design is central to the discipline

- Existing models of computation should be extended to encompass physicality – physical resources e.g. memory, time, energy should become first class concepts

- Complements and enriches our knowledge with theory and models that allow a deeper understanding of discrete dynamic systems

- Proposes a constructive and operational view of the world which complements the classic declarative approach adopted by Physics
A System-Centric Vision for CS

System Design

- Raises a multitude of deep theoretical problems such as the conceptualization of needs in a given area and their effective transformation into correct artifacts.

- has attracted little attention from scientific communities and is relegated to second class status
  - design is by its nature multi-disciplinary and requires consistent integration of heterogeneous system models supporting different levels of abstraction including logics, algorithms and programs as well as physical system models.

- is central to CS. Awareness on its centrality is a chance to reinvigorate CS research and build new scientific foundations matching the needs for increasing system integration and new applications.
A System-Centric Vision for CS – The Frontiers

Mathematics

Logic

Physics

Computer Science

Biology
We need theory, methods and tools for climbing up-and-down the abstraction hierarchy
**A System-Centric Vision for CS for CS – The Frontiers**

**Physics**
- Deals with phenomena of the « real » physical world (transformations of matter and energy)
- Focuses mainly on the discovery of physical laws.
- Physical systems – Analytic models
- Continuous mathematics
- Differential equations – robustness
- Predictability for classical Physics
- Mature discipline

**Computer Sc.**
- Deals with the representation and transformation information
- Focuses mainly on the construction of systems
- Computing systems – Machines
- Discrete mathematics – Logic
- Automata, Algorithms, Complexity Theory
- Verification, Testing,
- Young fast evolving discipline

---

**Deals with phenomena of the « real » physical world (transformations of matter and energy)**

**Focuses mainly on the discovery of physical laws.**

**Physical systems – Analytic models**

**Continuous mathematics**

**Differential equations – robustness**

**Predictability for classical Physics**

**Mature discipline**

**Deals with the representation and transformation information**

**Focuses mainly on the construction of systems**

**Computing systems – Machines**

**Discrete mathematics – Logic**

**Automata, Algorithms, Complexity Theory**

**Verification, Testing,**

**Young fast evolving discipline**
Artificial vs. Natural Intelligence
Living organisms intimately combine interacting physical and computational phenomena that have a deep impact on their development and evolution

- Shared characteristics with computing systems
  - use of memory
  - distinction between hardware and software
  - use of languages

- Remarkable differences:
  - robustness of computation
  - built-in mechanisms for adaptivity
  - emergence of abstractions – concepts

Interactions and cross-fertilization
- Non von Neumann computing ↔ Neuromorphic, Cognitive Computing
- CAD methods&tools ⇒ Synthetic Biology
Church-Turing Thesis: Every effectively calculable function is a computable function (by a Turing machine or an equivalent model)

The models of computation on which CS is founded
- Come from a priori knowledge (mathematics, logic)
- Are discrete and sequential, can only approximate physical computing processes which are intrinsically continuous and parallel
- They can model only slow, deliberate, analytical and consciously effortful human reasoning but not our fast, automatic, intuitive and largely unconscious thinking (Daniel Kahneman’s “Thinking Fast and Slow”)

There exist probably other models of computation to discover
- Using directly the computational capabilities of physical or biological systems e.g. analog computing, quantum computing, bio-molecular computing
- which are robust, parallel and are not subject to current limitations
Theoretical research in CS often focuses on “nice theory” that is not always practically relevant.

“Make everything as simple as possible, but not simpler”

“..... in the academic world the theories that are more likely to attract a devoted following are those that best allow a clever but not very original young man to demonstrate his cleverness.”

.... while practitioners propose ad hoc frameworks hardly amenable to formalization e.g. non-orthogonal concepts, ambiguous semantics

“Perfection is reached not when there is no longer anything to add, but when there is no longer anything to take away”
A System-Centric Vision for CS – Looking for Foundations

Is it possible to find a mathematically elegant and still practicable theoretical framework for system design?

Physics and Biology study a given “reality”  The key issue is discovering laws governing phenomena

"The most incomprehensible thing about the world is that it is at all comprehensible."
“The Book of Nature is written in the language of mathematics”

Computer Science mainly deals with building systems (artifacts)  The key issue is building correct systems cost-effectively
The physical world is part of our conditions of existence.

We mold the conditions of existence of the cyber-world.

*Is everything for the best in the best of all possible cyber-worlds?*