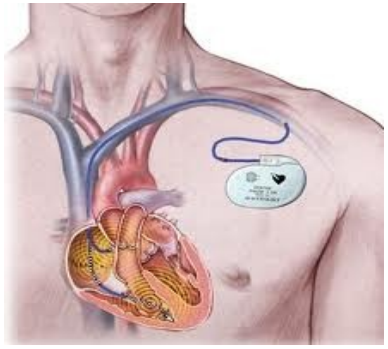
Cyber-Physical Systems
The Challenge for Rigorous Design

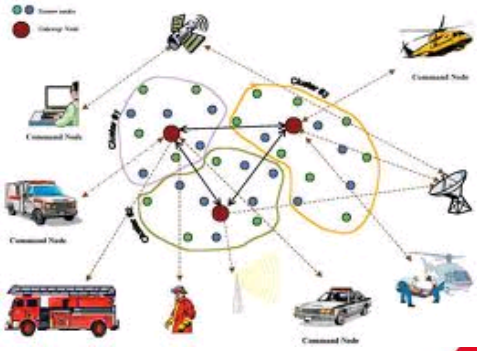
Axel Legay

Inria Rennes (France)

Cyber-Physical Systems

- Complex systems that tightly integrate multiple, networked computing elements (hardware and software) with …
- non-computing physical elements such as electrical or mechanical components, and that
- Are increasingly connected to the internet



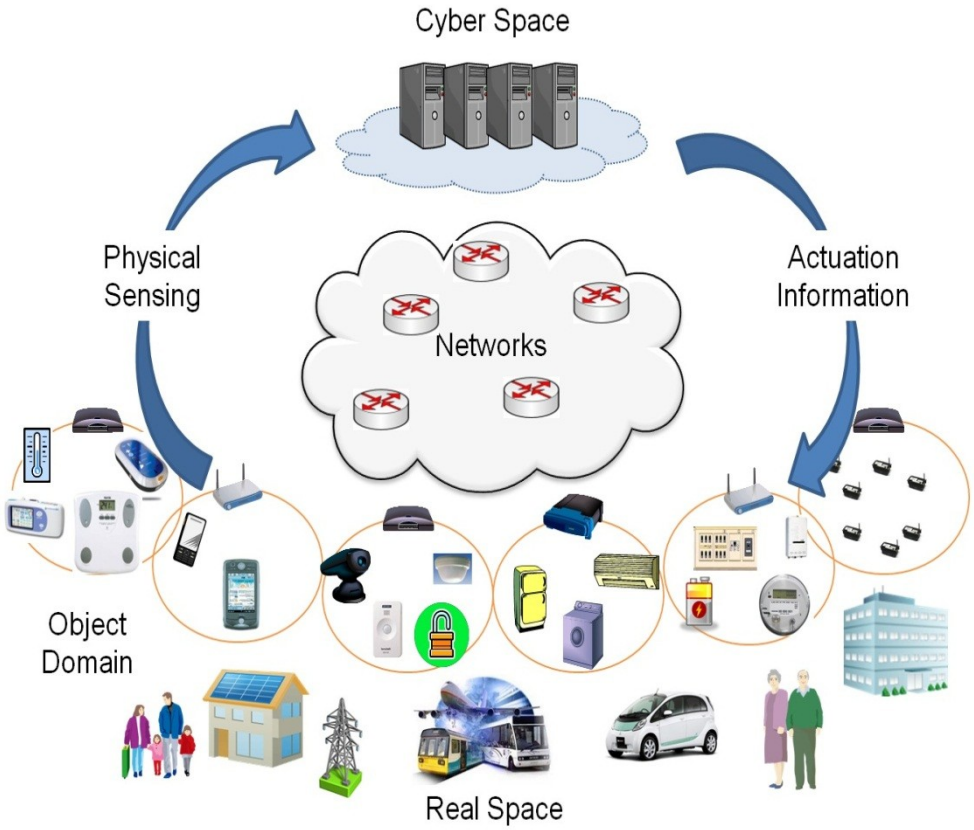


Main Features?

- Decentralized control
- Availability, Dynamicity
- Intensive communication

  Big Data
- Heterogeneity
- Quantities (time,

  ressources, autonomy,…)
- Marked friendly
- …



What really matters?

Adopted by humans

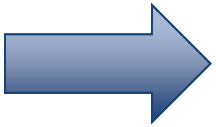
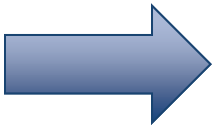
Trustworthy: Safety, Security, and Privacy

Energy constraints

State of the art

- Ad-Hoc design: components are plugged « here and there ». No methodology!
- Security/safety/privacy via existing techniques (limits/no combination)
- Reasoning on energy, coordination, heterogeneity, and dynamicity is poorly understood
- ....

What are we after?
My vision for 15 years and more

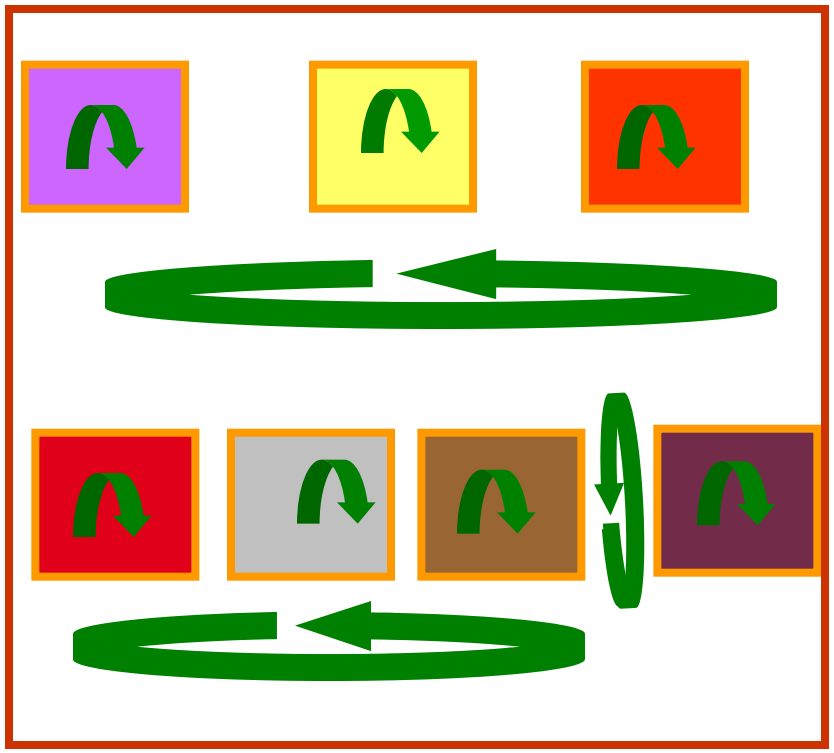
- A new workflow for the rigorous design of CPS
- Language for components, interactions, properties
-  New mathematical foundations :
  - Quantities, datas, Heterogeneity, dynamicity
  - Distributed design and Hardware (systems!)
-  New Powerful verification techniques
  - Exploit Incremental Design and synthesis
  - Correct by construction.

Component Heterogeneity – Programming Styles

Thread-based programming

Software Engineering

Actor-based programming



Systems Engineering

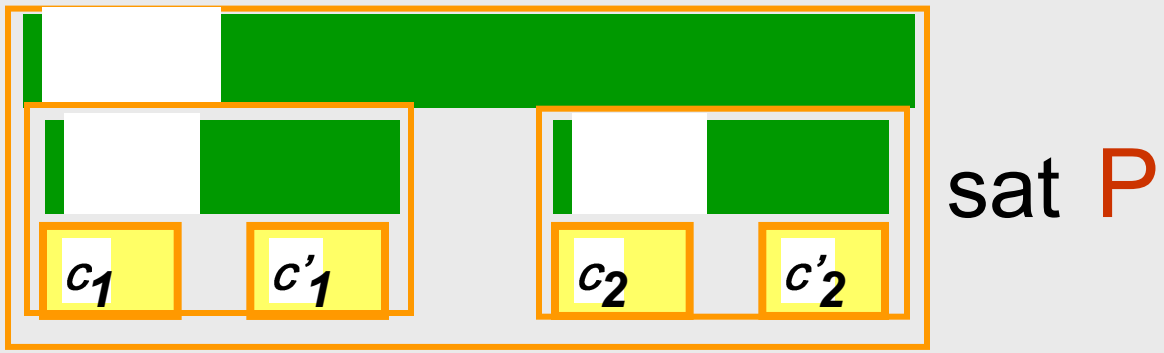
Component Heterogeneity – Composition (Sifakis)

- How to express component coordination in terms of composition operators?

- Orthogonality: clear separation between behavior and coordination constraints
  - Minimality: uses a minimal set of primitives
  - Expressiveness: achievement of a given functionality with a minimum of mechanism and a maximum of clarity

- Almost nothing:
  - Some are formal such as process algebras e.g. CCS, CSP, pi-Calculus
  - Other are ad hoc such as most frameworks used in software engineering e.g. ADL, or systems engineering e.g. SystemC

Joseph Sifakis: The Concept of Glue



The glue is used to synchronize (heterogeneous) components and to generate code!

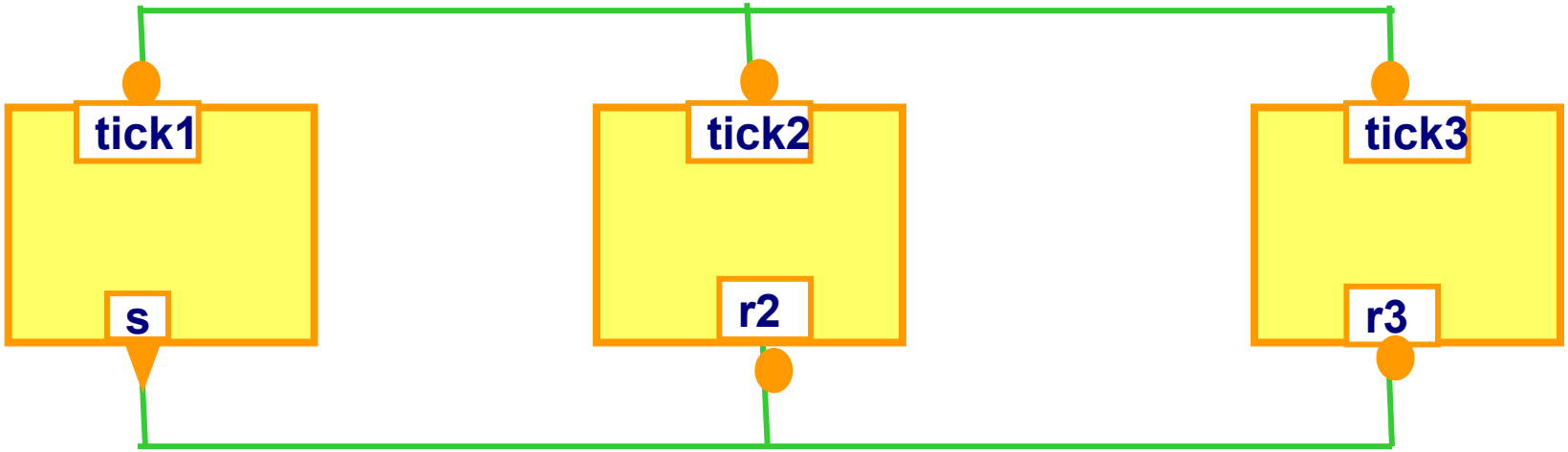
Questions:

1. How to handle (data) communication?
2. How to enforce global properties (priorities)?
3. How to deal with composability?

Modeling Interactions – Connectors

Express interactions by combining two protocols: rendezvous and broadcast

- A ***connector*** is a set of ports that can be involved in an interaction
- Port attributes (***trigger*** ▼, ***synchron*** ●) are used to model rendezvous and broadcast.
- An ***interaction*** of a connector is a set of ports such that: either it contains some trigger or it is maximal.

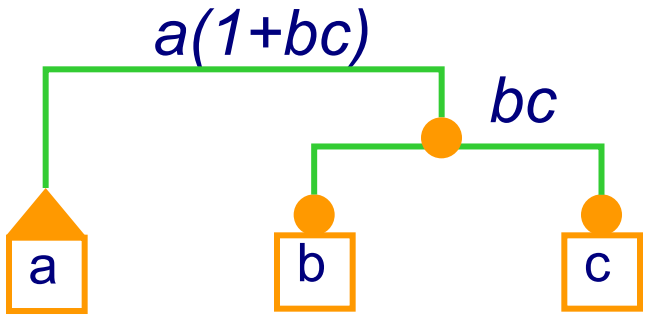


s + sr2 + sr3 +sr2r3

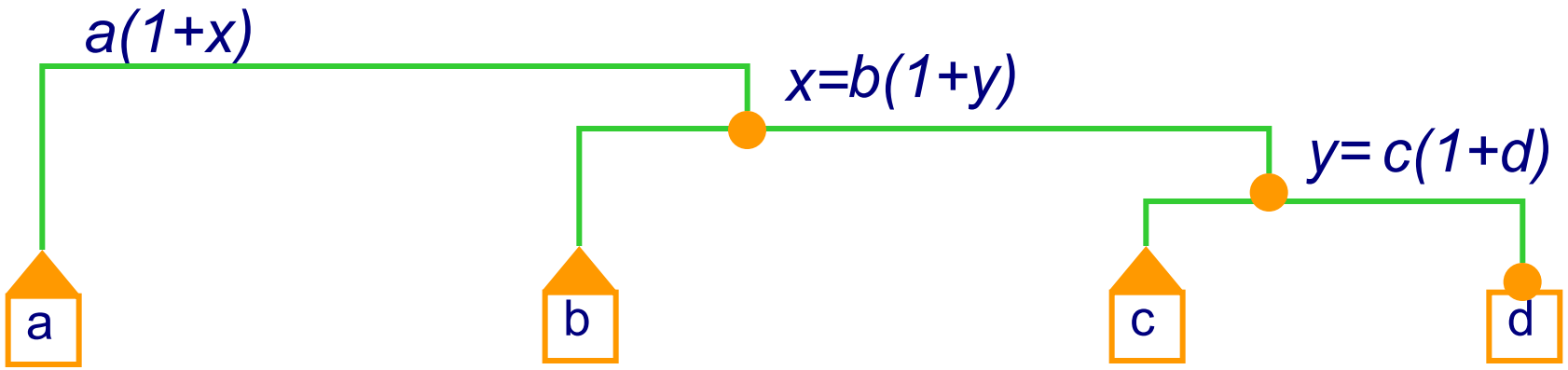
Modeling Interactions – Connectors

Atomic Broadcast:

a+abc

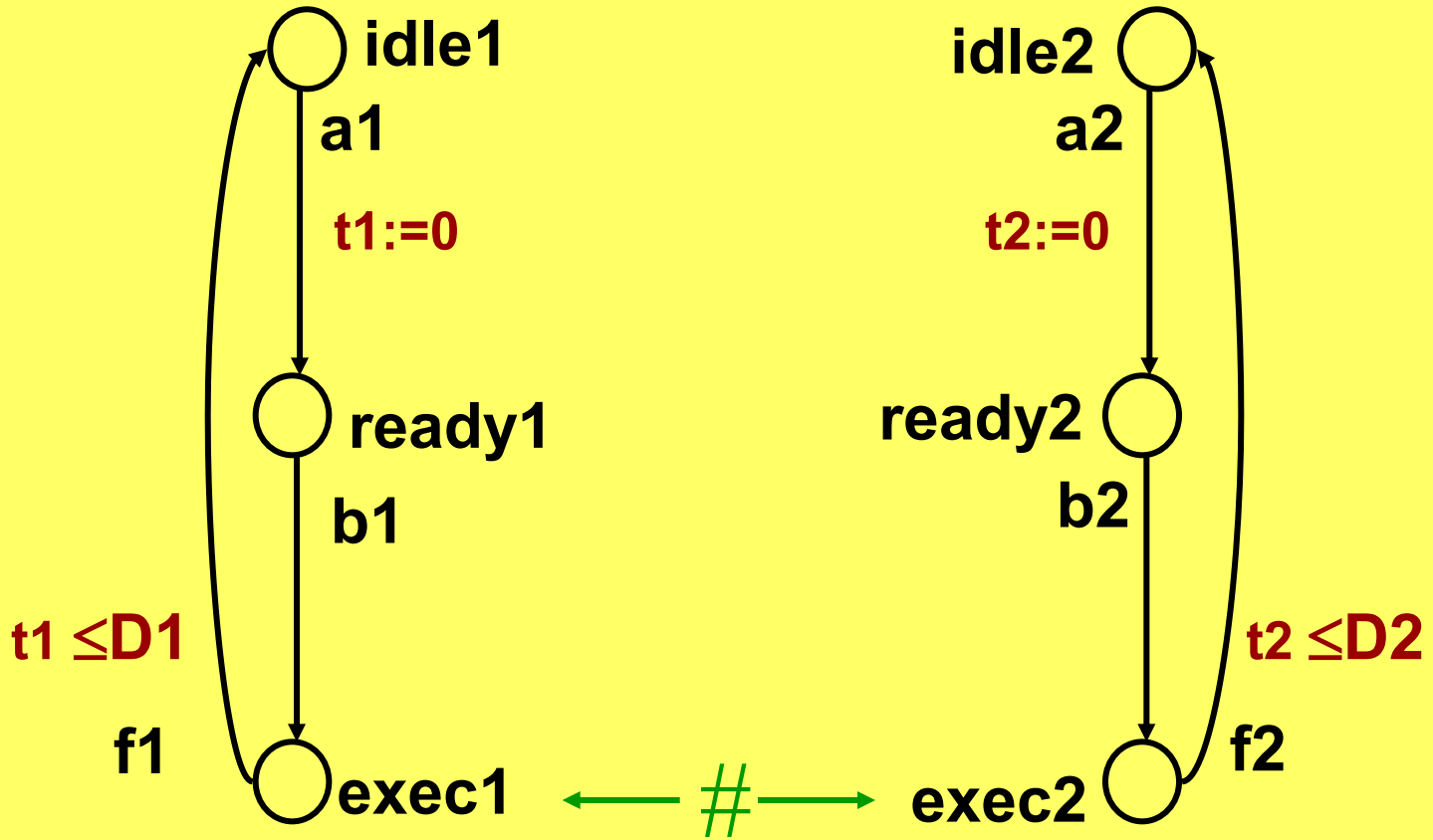


Causality chain: a+ab+abc+abcd



Example: Modeling Priorities – EDF policy

PR: D1-t1≤ D2- t2 → b2⟨ b1 D2-t2< D1-t1 → b1⟨ b2



Correct-by-Construction – Architectures

Architectures

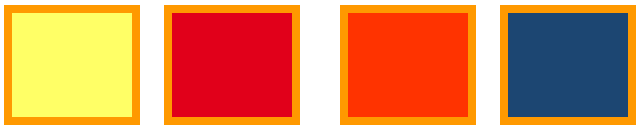
1. depict design principles, paradigms that can be understood by all, allow thinking on a higher plane and avoiding low-level mistakes (Abstraction – glue again)
2. a means for ensuring global properties characterizing the coordination between components – (correctness for free)
3. Using architectures is key to ensuring trustworthiness and optimization in networks, OS, middleware, HW devices etc.

System developers extensively use libraries of reference architectures ensuring both functional and non functional properties e.g.

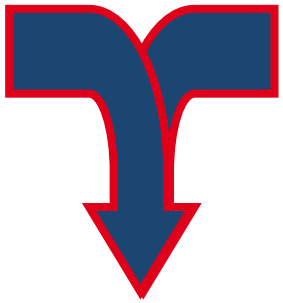
1. Fault-tolerant architectures
2. Resource management and QoS control
3. Time-triggered architectures
4. Security architectures
5. Adaptive Architectures

Rule1: Property Enforcement

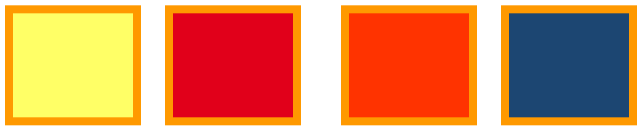
Architecture
for Mutual Exclusion



Components



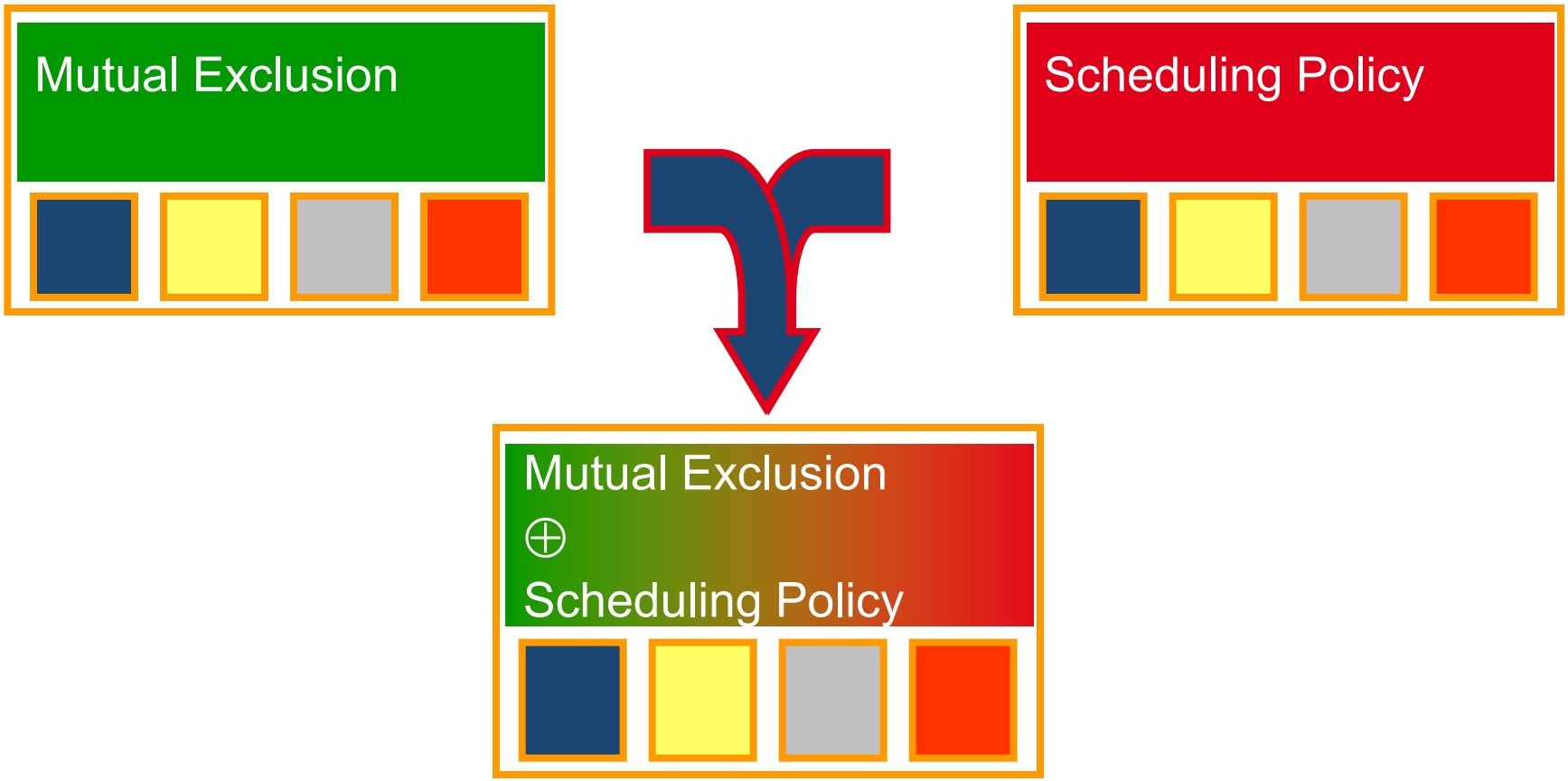
Architecture
for Mutual Exclusion



satisfies Mutex

Correct-by-Construction – Architectures: Composability

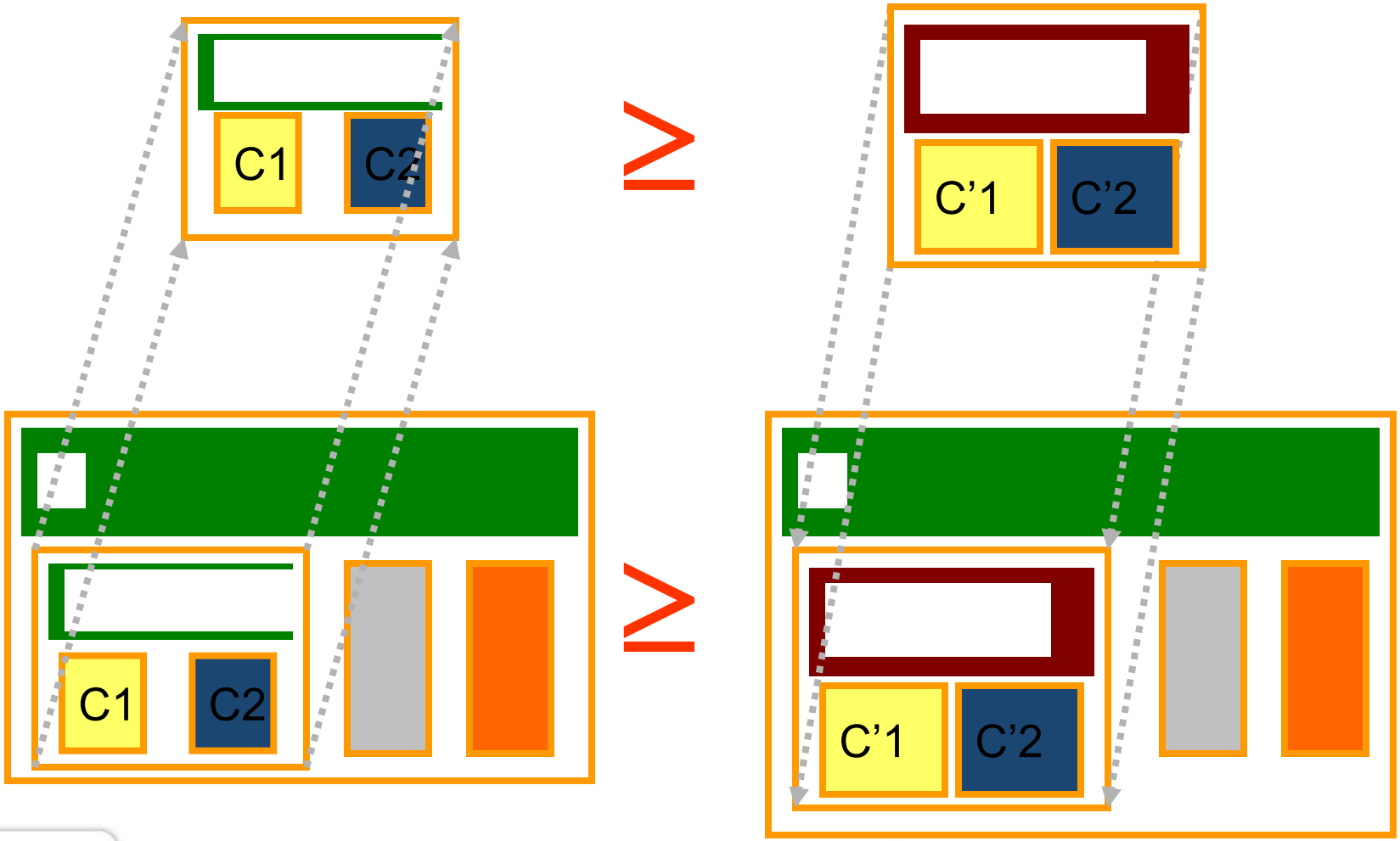
Rule2: Property Composability



Feature interaction in telecommunication systems, interference among web services and interference in aspect programming are all manifestations of a lack of composability

Correct-by-Construction – Refinement Preservation

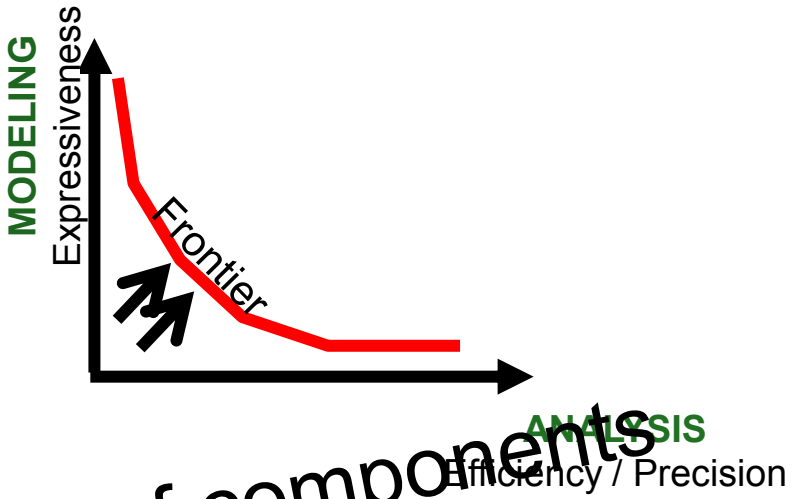
Preservation of ≥ by substitution



Model for components

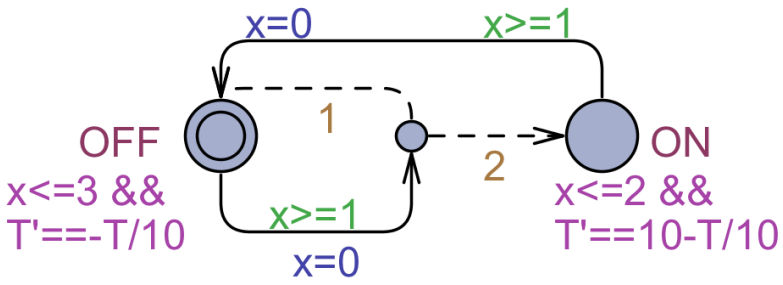
The need to describe:

- Behaviors
- Quantitative informations
- Interface with external world
- heterogeneity
- Hardware constraints
- ….



A finite library of types of components

And to conform to standards!



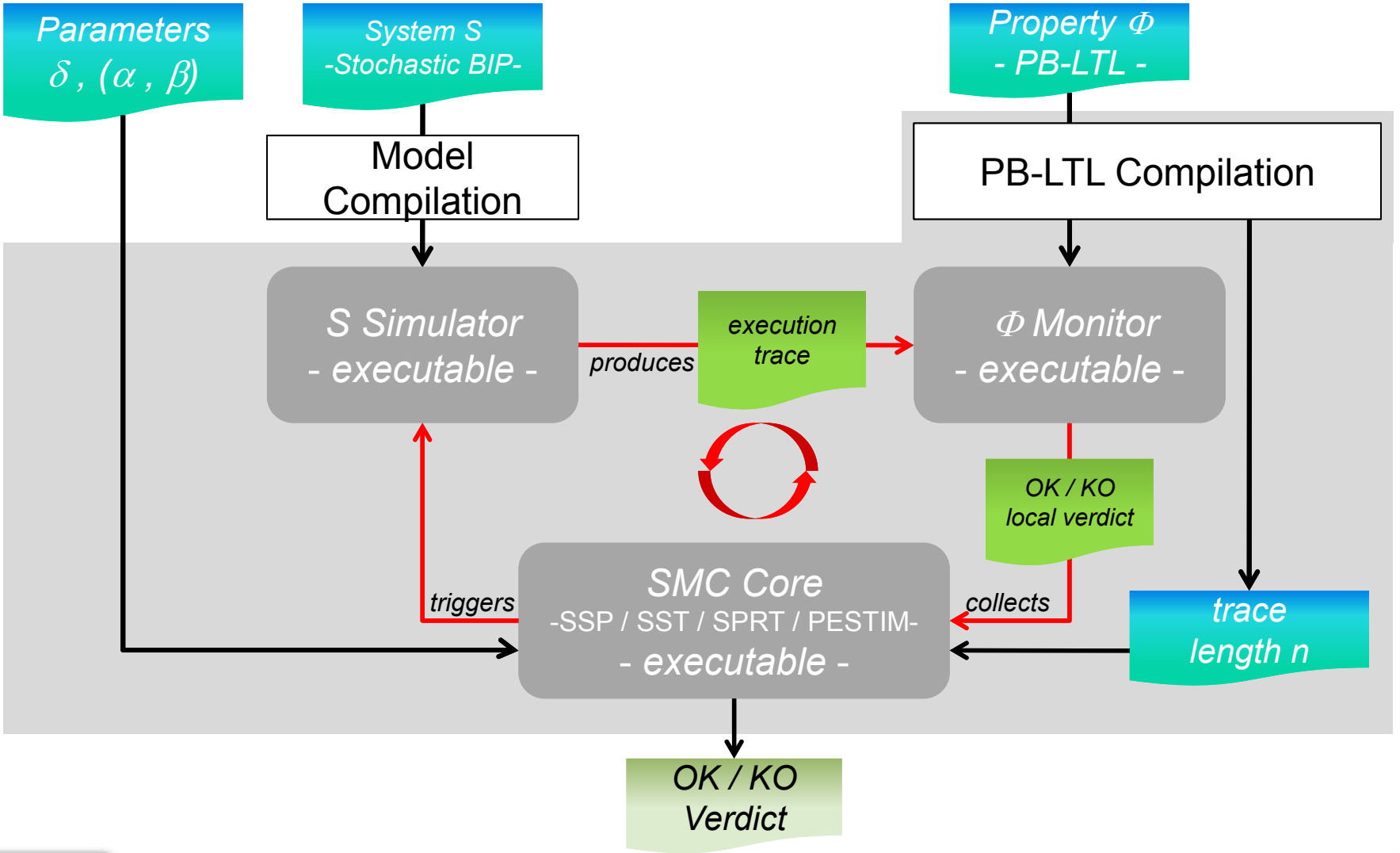
Verifying properties on (sets of) component(s)

- State-space explosion problems (+ Big Datas)
- Undecidability problems
- Expressivity problems
- …..

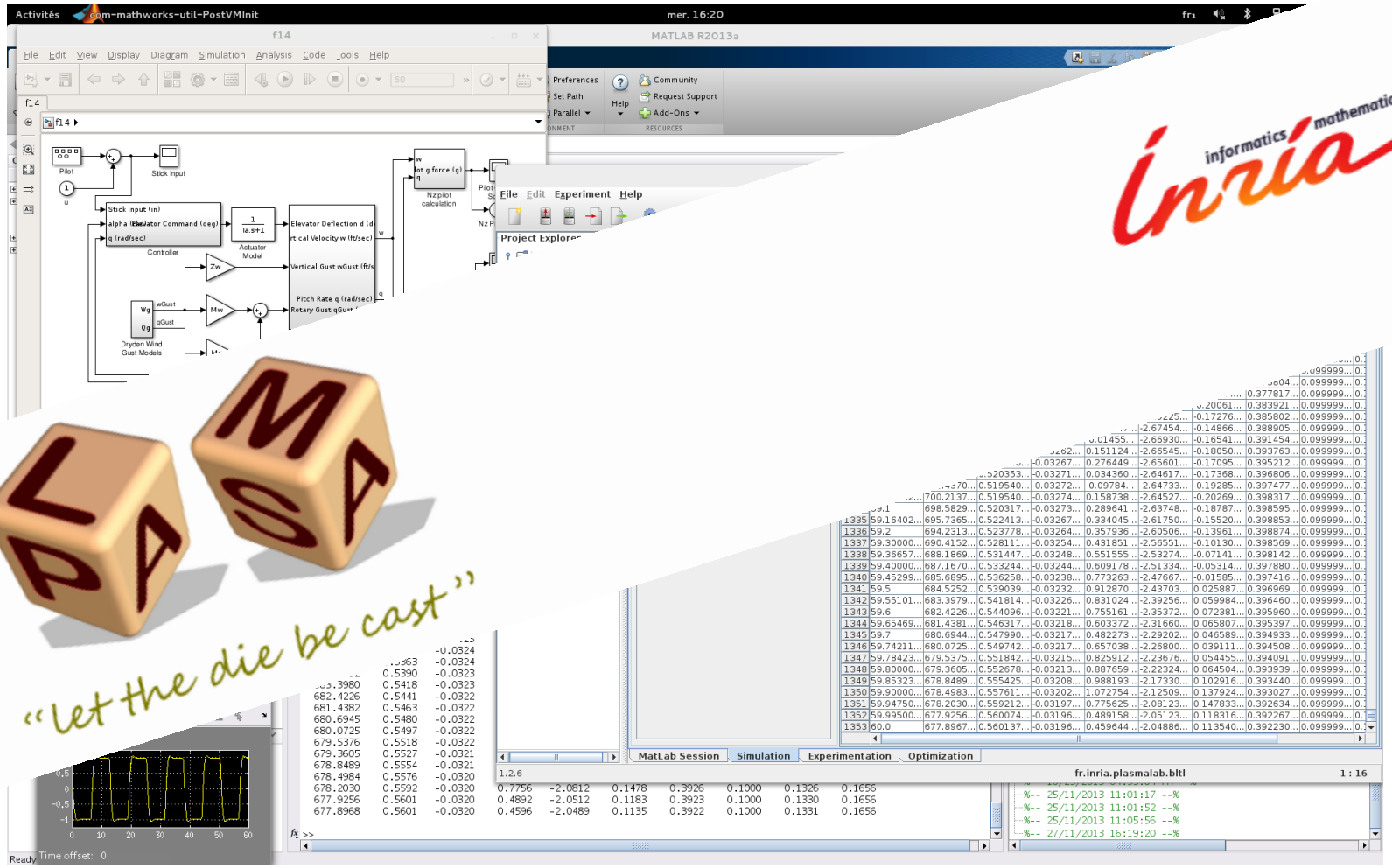
Exploits new types are highly needed

New ligthweight technologies!

Statistical Model Checking



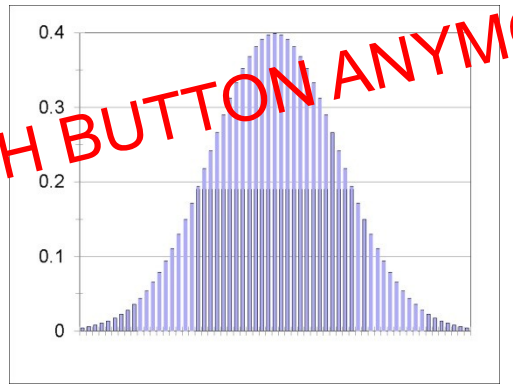
Statistical Model Checking



Modeling Uncertainty

CPS evolve in uncertain environment:

- How to capture and represent human behaviors ?
- How to capture potential additions/removals?



Design will be an iterative process: NOT PUSH BUTTON ANYMORE!

From Humans to Statistics?

Research priorities: summary

- New modeling approaches
  - Composition (correct by construction) is central
  - Learning is needed
- New efficient verification techniques
  - The Boolean world is obsolete
- Interleaving engineering approaches with formal ones
- Crossing knowledge and interdisciplinarity
- Tools and interactions with industry (chist-era?)
- Understanding the social aspects of engineering

You should not try to walk alone