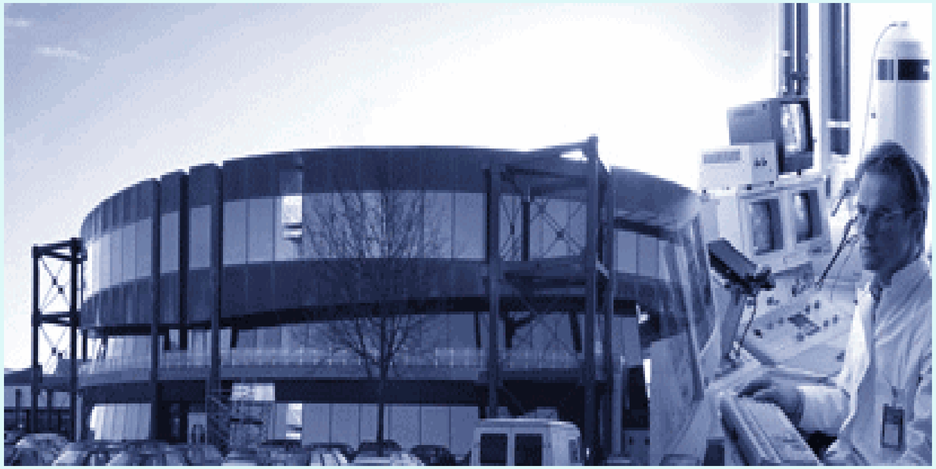
Océ Development Processes

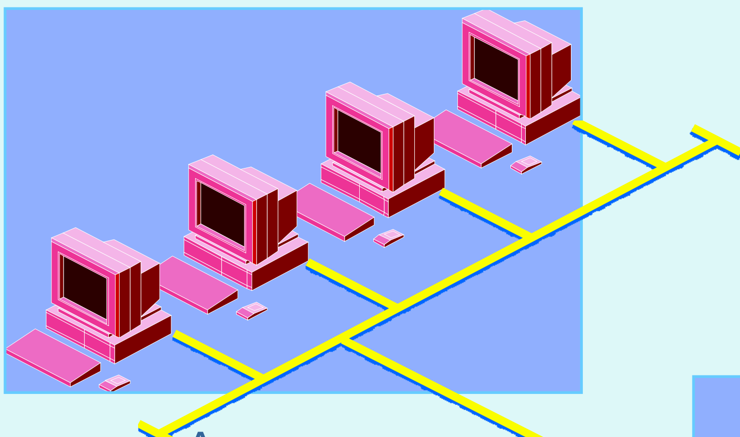


Ed Brinksma, ESI

presentation sources:

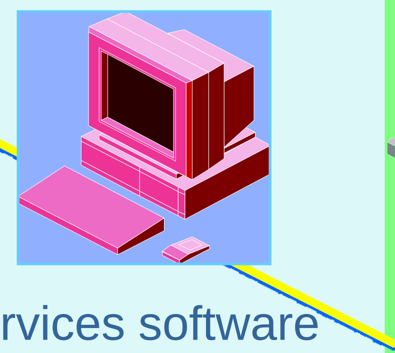
- Ron Notermans (VP R&D Océ)
- Peter van den Bosch (developer Océ)
- Lou Dohmen (architect Océ)
- Roelof Hamberg (formerly Océ)
- Gerrit Muller (BoDERC project)

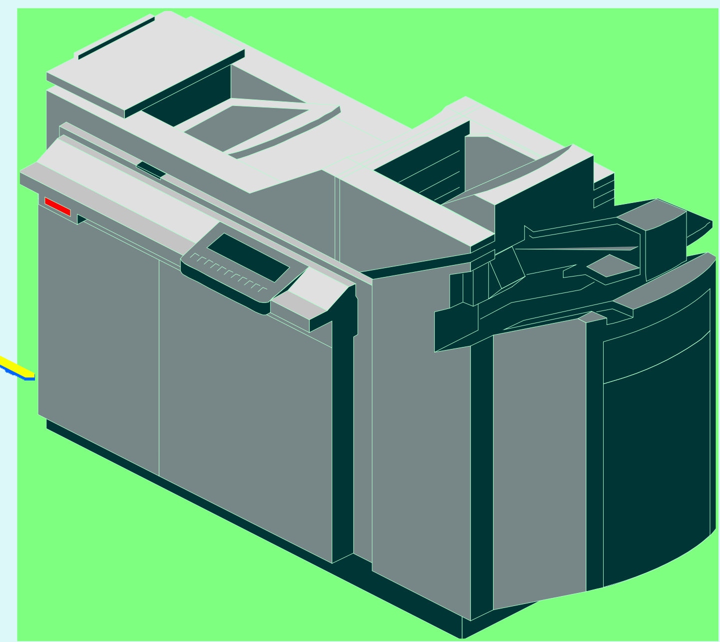
Océ - Document Systems



- **Consultancy**
- **Facility Management**
- **Maintenance**

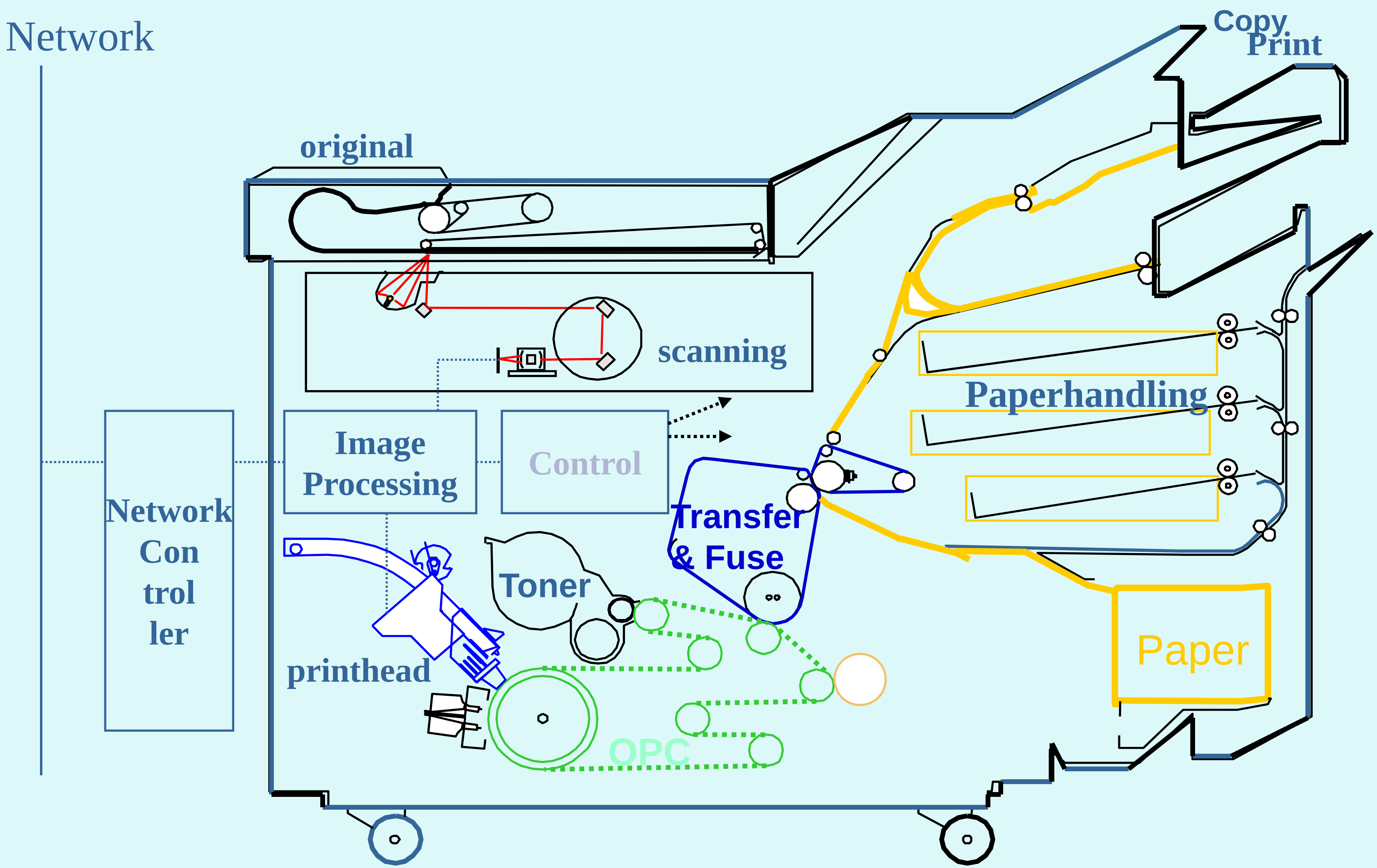
Océ enables its customers to manage their documents efficiently and effectively by offering innovative print and document management products and services for professional environments





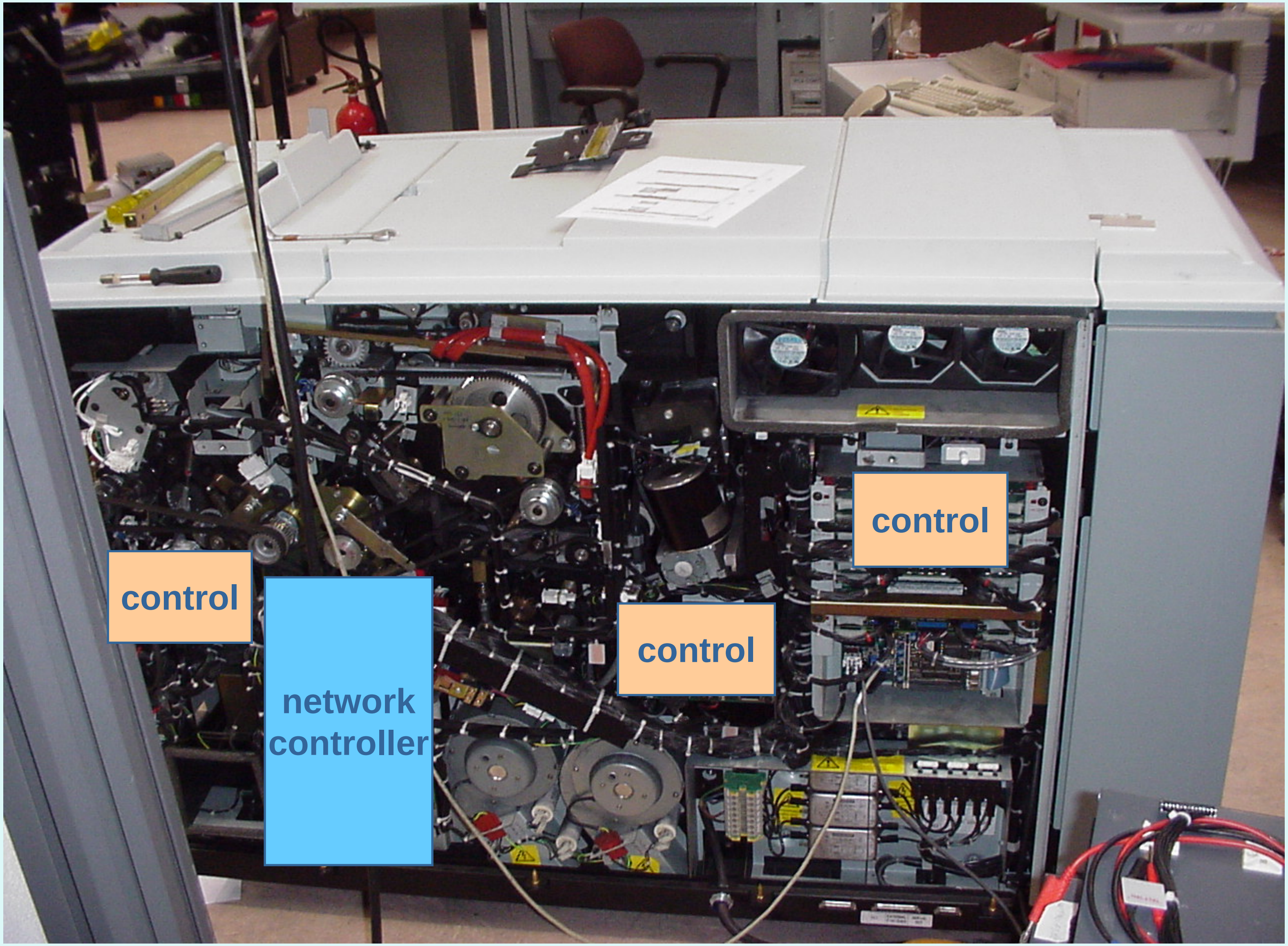
Application Software + Machines

Principle of working



Mechanics, Electronics, Procédé
Embedded Software

Embedded Systems INSTITUTE



Current multidisciplinary approach

Embedded Systems INSTITUTE

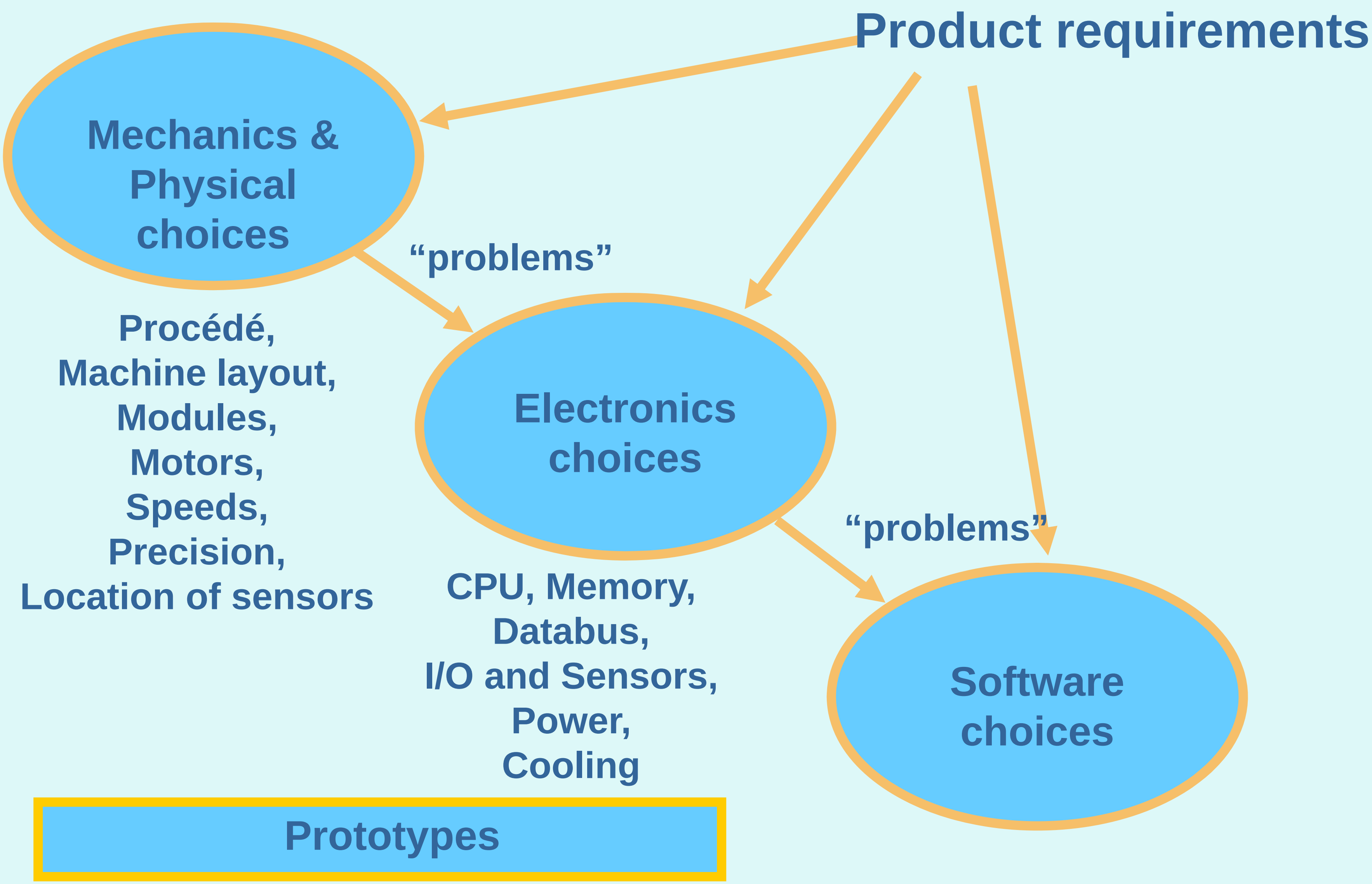
- **Multidisciplinary project teams (M, E, I, ID, Φ, C, …)**
- **Common project goal**
- **Dedication** (1 project per person at a time)
- **Co-location** (1 location per project team)
- **Project organisation processes**
  - Groups per function, following constructional decomposition
  - TC decides (function leaders, architect(s), project leader)
- **Lot of prototyping & testing**
  - **FunctionTest → Labmodel → EngineeringProtoType**
- **Years of experience**

,but…

Mechanics	Electronics	Embedded Software

• Agenda
– Cost price
– Reliability
– Performance
– Precision
• Models
– 3D CAD
– Total construction drawings
– Some local Matlab models
– Specification of time/position tables in Excel
	• Agenda
	– Cost price
	– Energy consumption
	– Life-cycle-management of components
	– Safety and EMC regulations
	• Models
	– System architecture diagrams
	– Circuit simulations (VHDL, PSPICE, Matlab)
		• Agenda
		– Specifications for normal and exceptional behaviour
		– Maximum performance for all jobs
		– Architecture
		  • Modular
		  • Maintainable
		  • Re-use of components
		• Models
		– Use Cases
		– UML
		– Text documents

De facto decision process



Problem characteristics

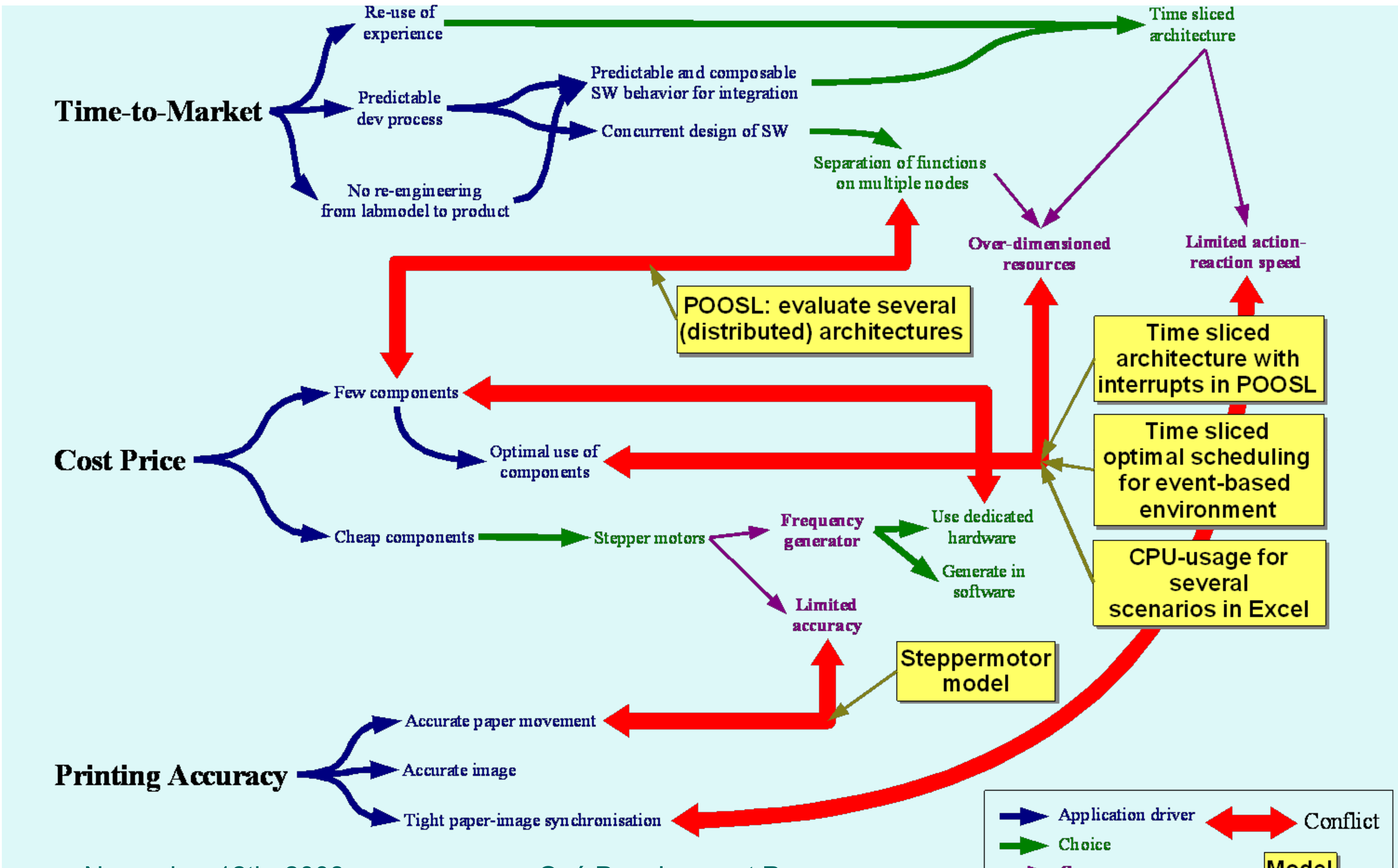
Embedded Systems INSTITUTE

- **Development of complex products with**
  - highly interactive sub-domains,
  - where only few people master both ends
  - of the physical and the software world
- **Long lead times**
  - Sequential order of work
  - Coincidental “just-in-time” delivery by hard work
  - Need for more predictability and shorter lead times
- **Software is often on the critical path**
  - Impact of software in earlier phases
  - Integral system approach is required

High level design method

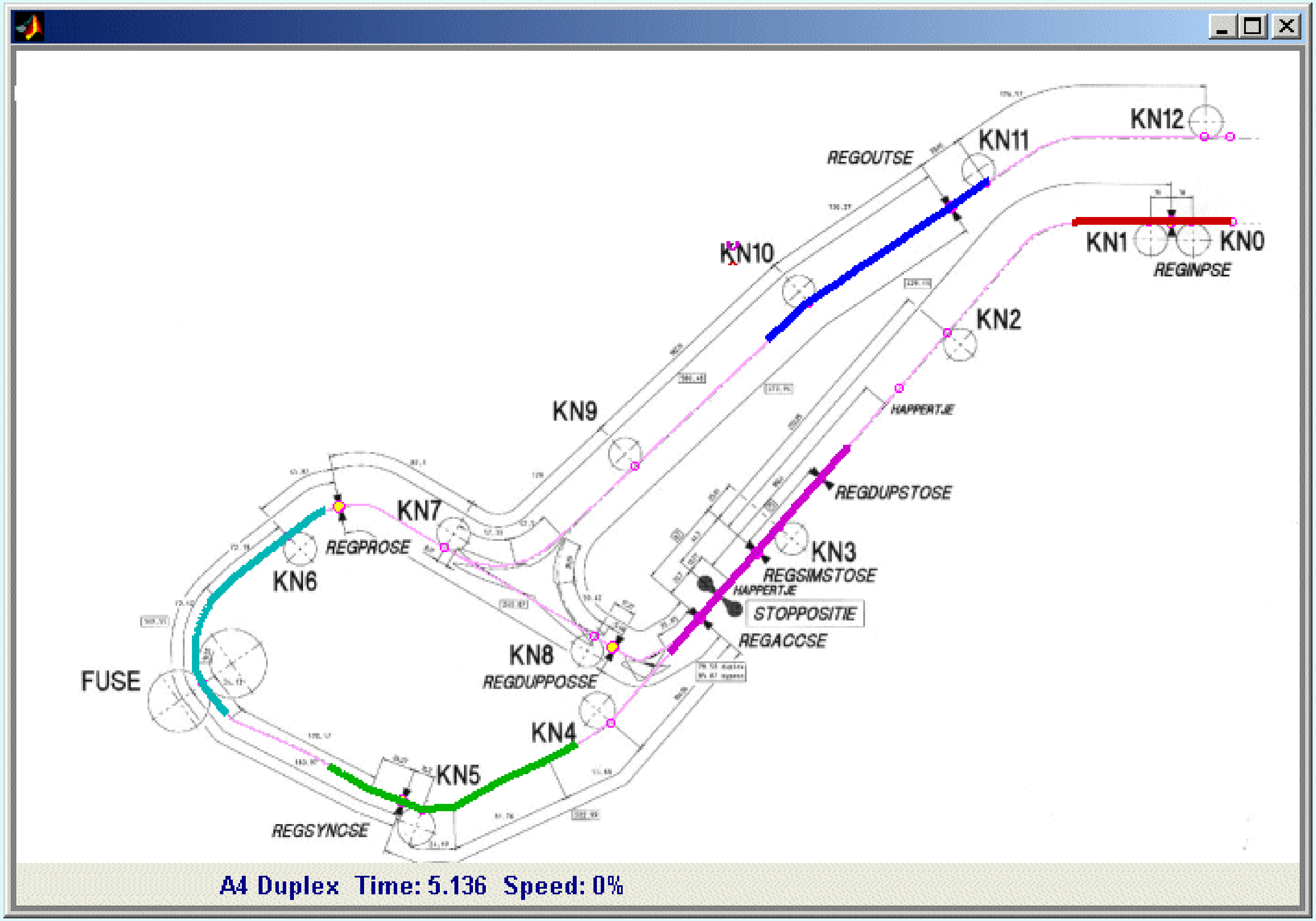
1. **Preparation of the design**
    a) Identify (customer) key drivers and requirements
    b) Identify realization aspects of concern
    c) Consolidate core domain knowledge
2. **Selection of critical design aspects**
    a) Identify tensions and conflicts (qualitative)
    b) Gather facts and identify uncertainties to quantify tensions and conflicts
3. **Evaluation of design aspects**
    a) Build small models
    (small = hours to 4 weeks of effort)
    b) Perform measurements

Example step 2a)
Identify tensions & conflicts (qualitative)



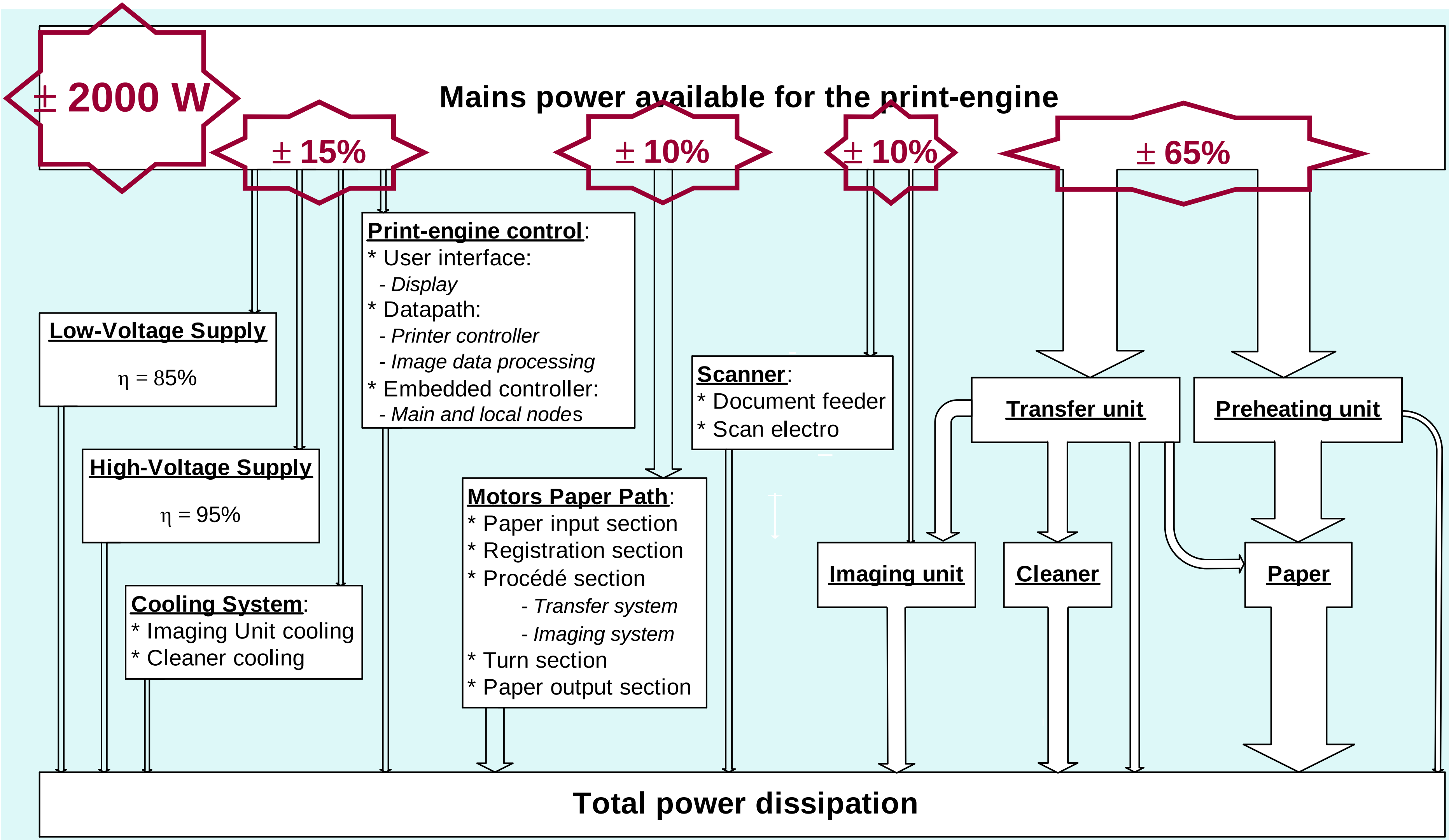
Example step 3a)
Build small models: Printer topology & sheet flow

Embedded Systems INSTITUTE



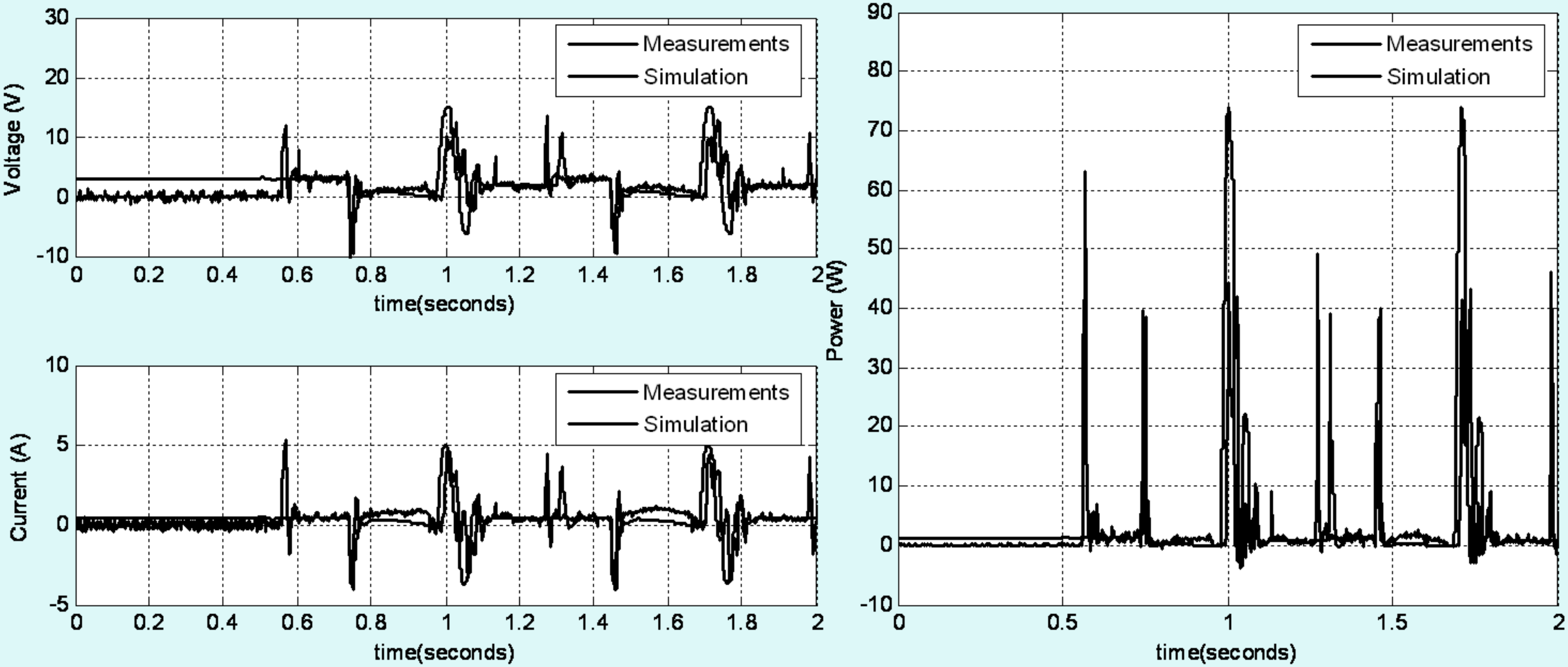
Example step 3a)
Build small models: Semi-Static Power Decomposition

Embedded Systems INSTITUTE



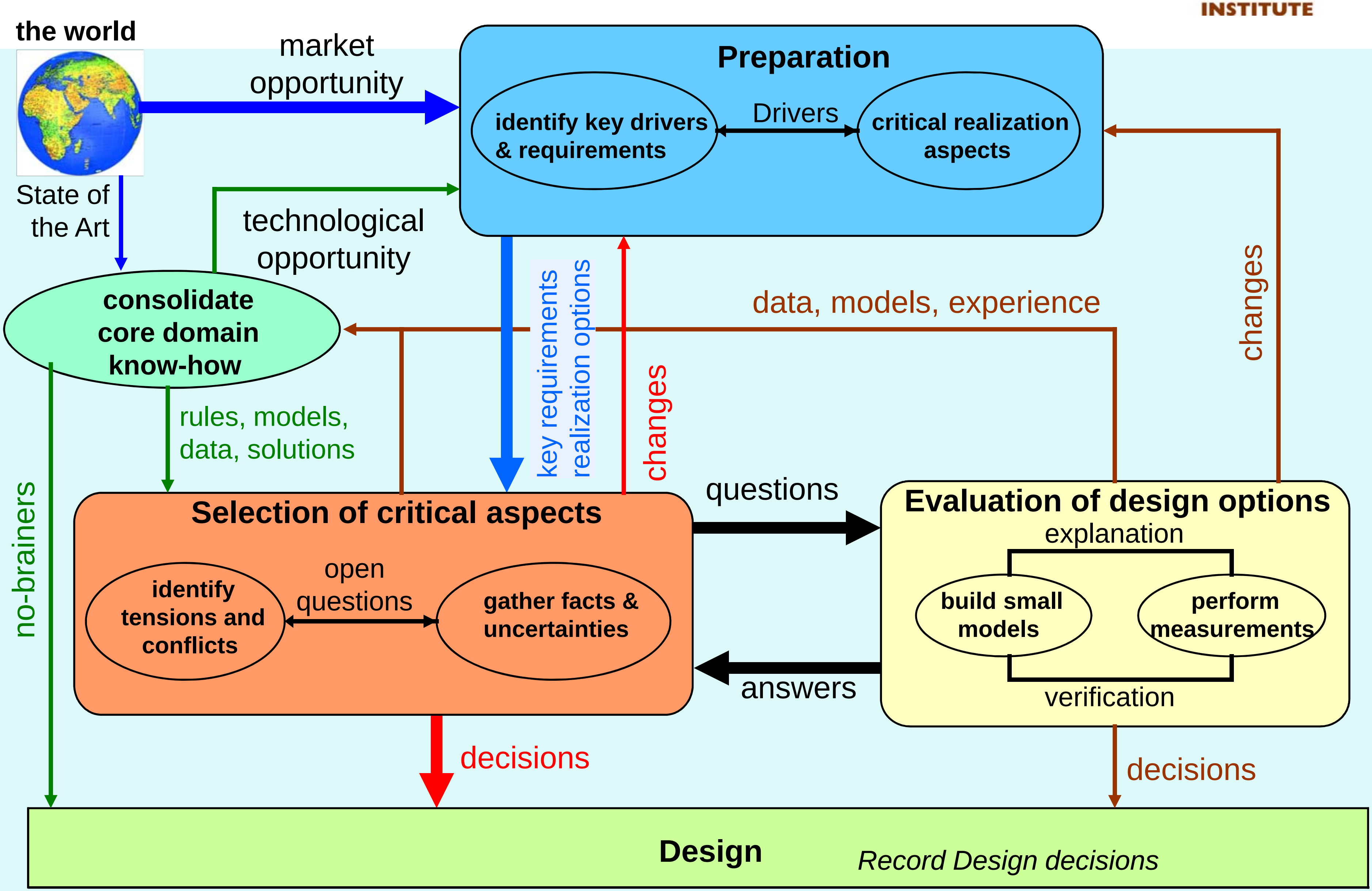
Example step 3b)
Measure: Dynamic behavior motors paper path

Embedded Systems
INSTITUTE



Iteration: dynamic workflow model

Embedded Systems INSTITUTE



A successor project:
Octopus (2007 – 2012)

Embedded Systems INSTITUTE

Challenges:

- Current systems are developed to operate reliably within a **very well specified range** of conditions (e.g. temperature, energy, humidity, etc)
- Such systems often do **not easily adapt** to environmental or product usage fluctuations

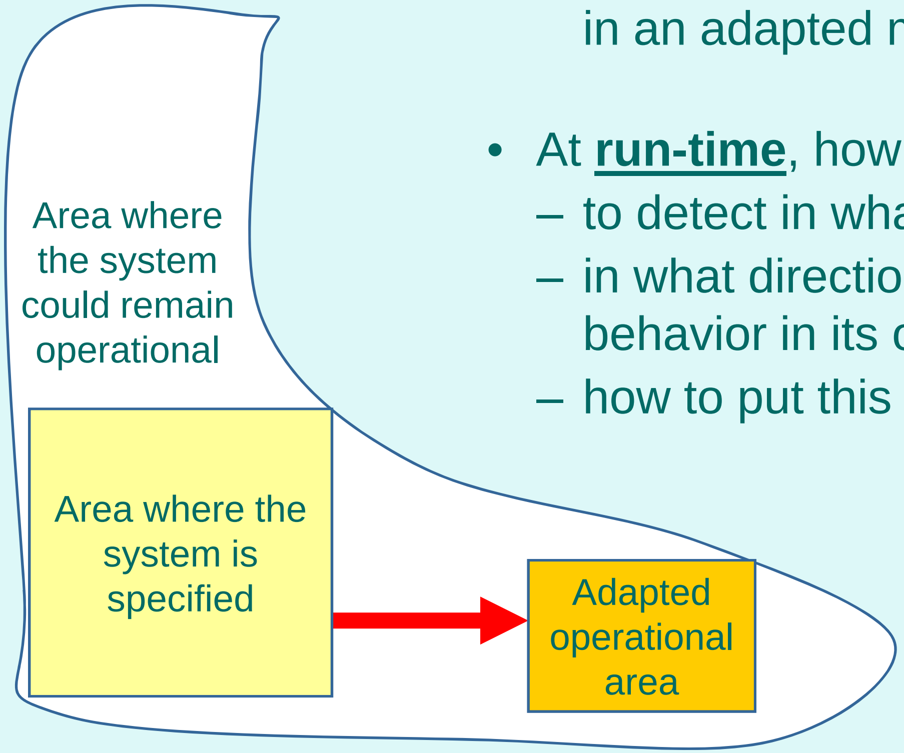
- An approach is to design and implement rapidly; equivalent to “next release”
  - Essential to **understand** the effects of design changes
- Another approach is **system’s adaptability**
  - Adapt the actual use of the product at run-time

Octopus objectives

Embedded Systems INSTITUTE

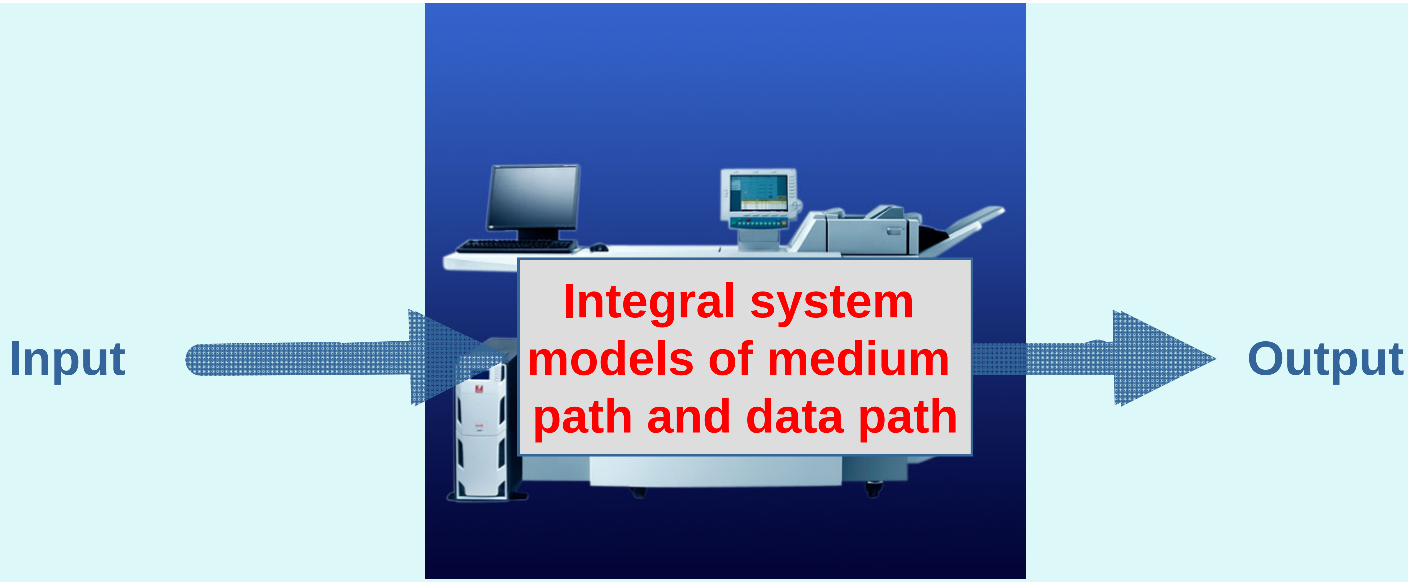
- At **design-time**, how
  - to build systems that are kept in a predictable, reliable, and operational mode,
  - while performing their task in an adapted manner?

- At **run-time**, how
  - to detect in what state the system is,
  - in what direction to adapt the product behavior in its operational space, and,
  - how to put this adaptability into effect?



Research approach Octopus (1)
1. Virtual prototypes

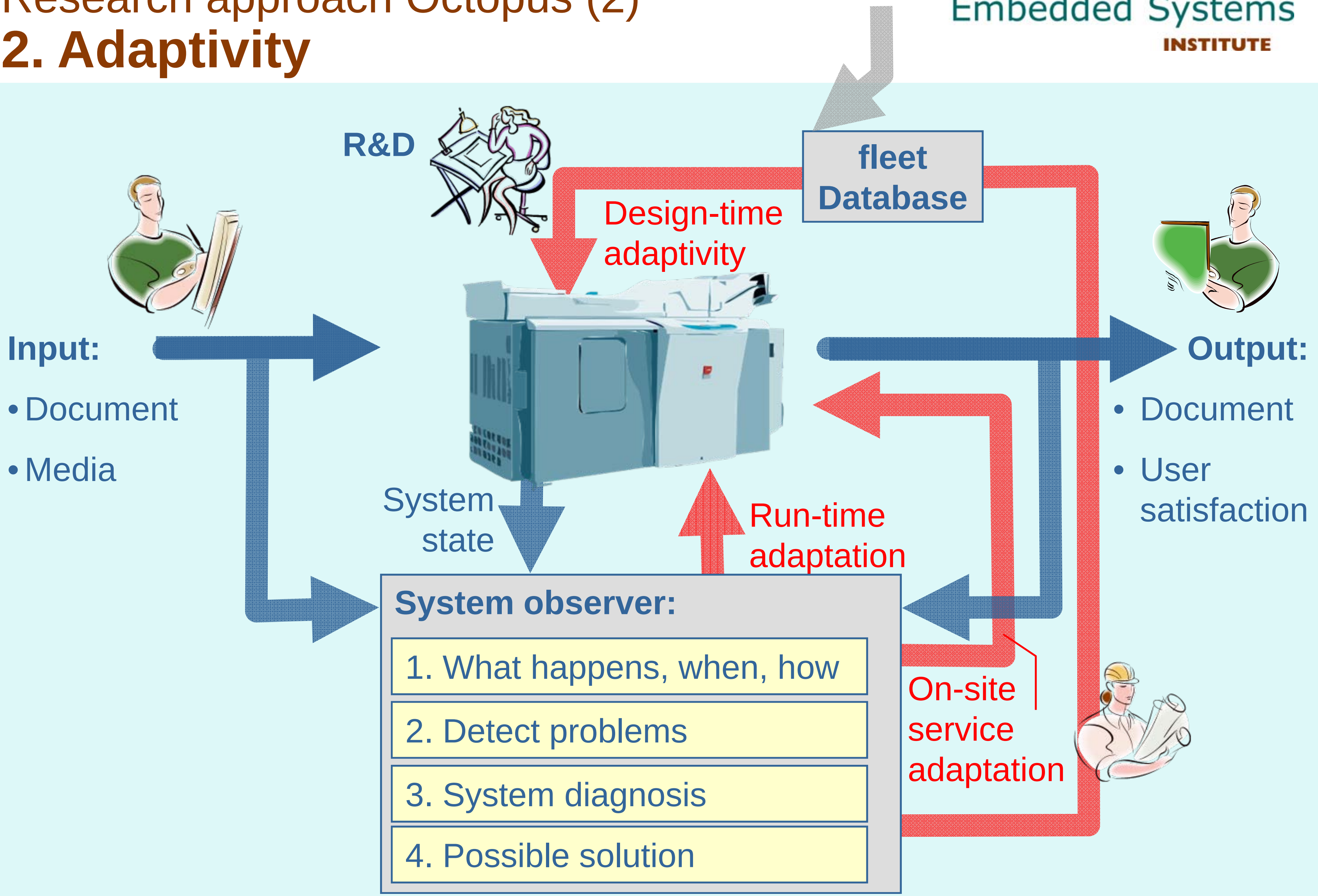
Embedded Systems INSTITUTE



1. Make good predictions early in the design phase
2. Develop integral models that cover medium and paper path, mechanics, black-box view of print process, etc.
   – including performance, resource usage, energy, space, …
3. Experiment with model without expensive (e.g. time-consuming) prototypes
4. Gradually replace models with real components
5. Tightly link architectural and design trade-offs
6. Validate integral models

Research approach Octopus (2)
2. Adaptivity

Embedded Systems INSTITUTE



Research approach Octopus (3)
several practical cases

Embedded Systems INSTITUTE

Bayesian networks

Pareto optimality

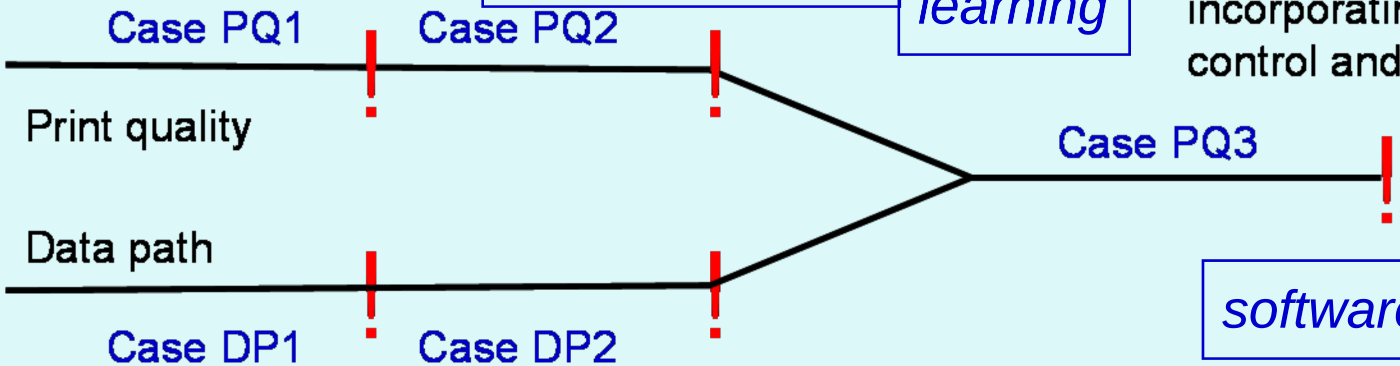
Demonstrator PQ 1
Trade off power consumption and productivity

Demonstrator PQ 2
Multiple trade-offs Including heat and glossiness

adaptive control

learning

Demonstrator PQ3
Full print quality control incorporating physical control and data-path



software architectures

Demonstrator DP1
High performance multiple scenario data path without adaptability

Demonstrator DP2
Data path without adaptability

SDF

Uppaal

PetriNets