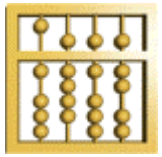
Complex Systems Modeling and Engineering and CPS research

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany



Topics

- Cyber Physical Systems - a slightly more general view
  - steps of their history
  - their characteristics - how they are different
- Modeling Cyber Physical Systems
  - Abstraction
  - Structuring
- Engineering CPS
  - Modeling as a basis of engineering
  - Requirements engineering
    - Functional Requirements
    - Quality
  - Architecture
- The German acatech Project agendaCPS
  - A holistic view onto CPS

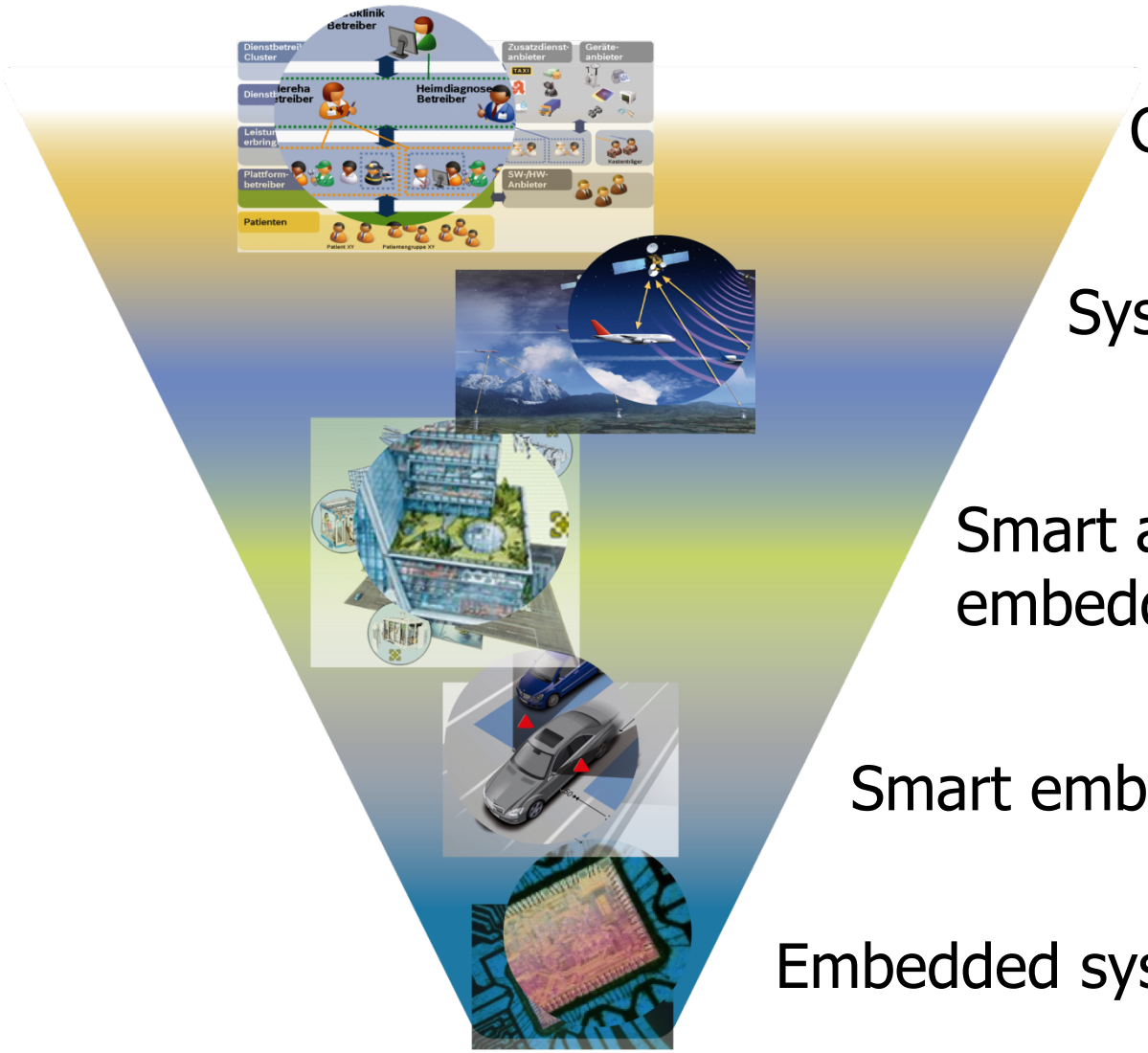
Key areas of innovation in ICT ...

Convergence of major fields of innovation in IT:

- IT infrastructure
  - Advanced software applications
  - Devices: PC, laptop, smart phone, …, a sea of sensors
- Embedded digital hardware & software systems
  - embedded control - real time
  - adapted automation
  - augmented reality
  - advanced assistance
- Cyber space: internet and world wide web – the "cloud"
  - data mining - customized search
  - social networks – human factors
  - knowledge engineering

The synergy: cyber-physical systems

Evolution: from Embedded zu Cyber-Physical Systems



Cyber-Physical Systems

Systems-of-systems

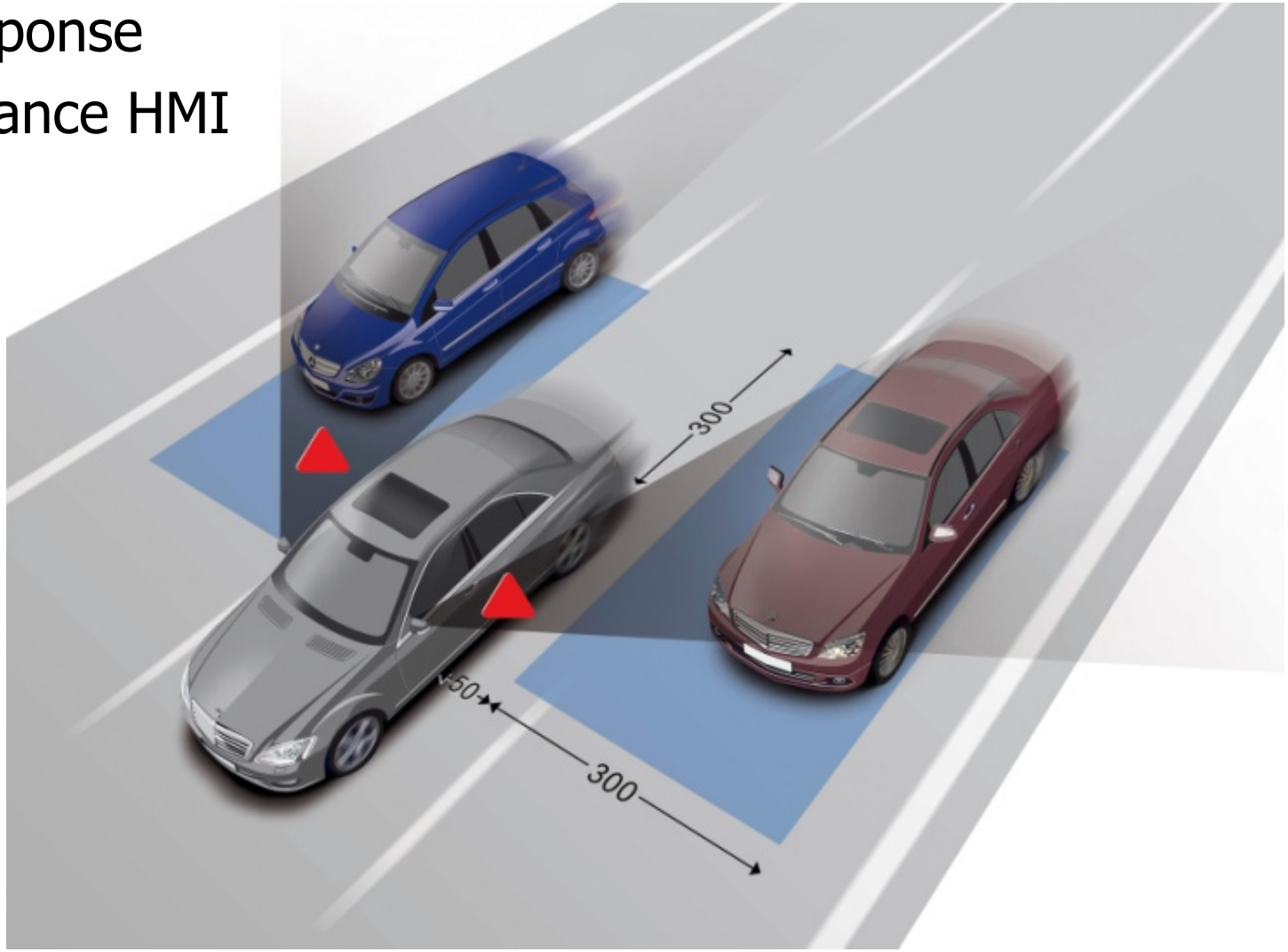
Smart and cooperative embedded systems

Smart embedded systems

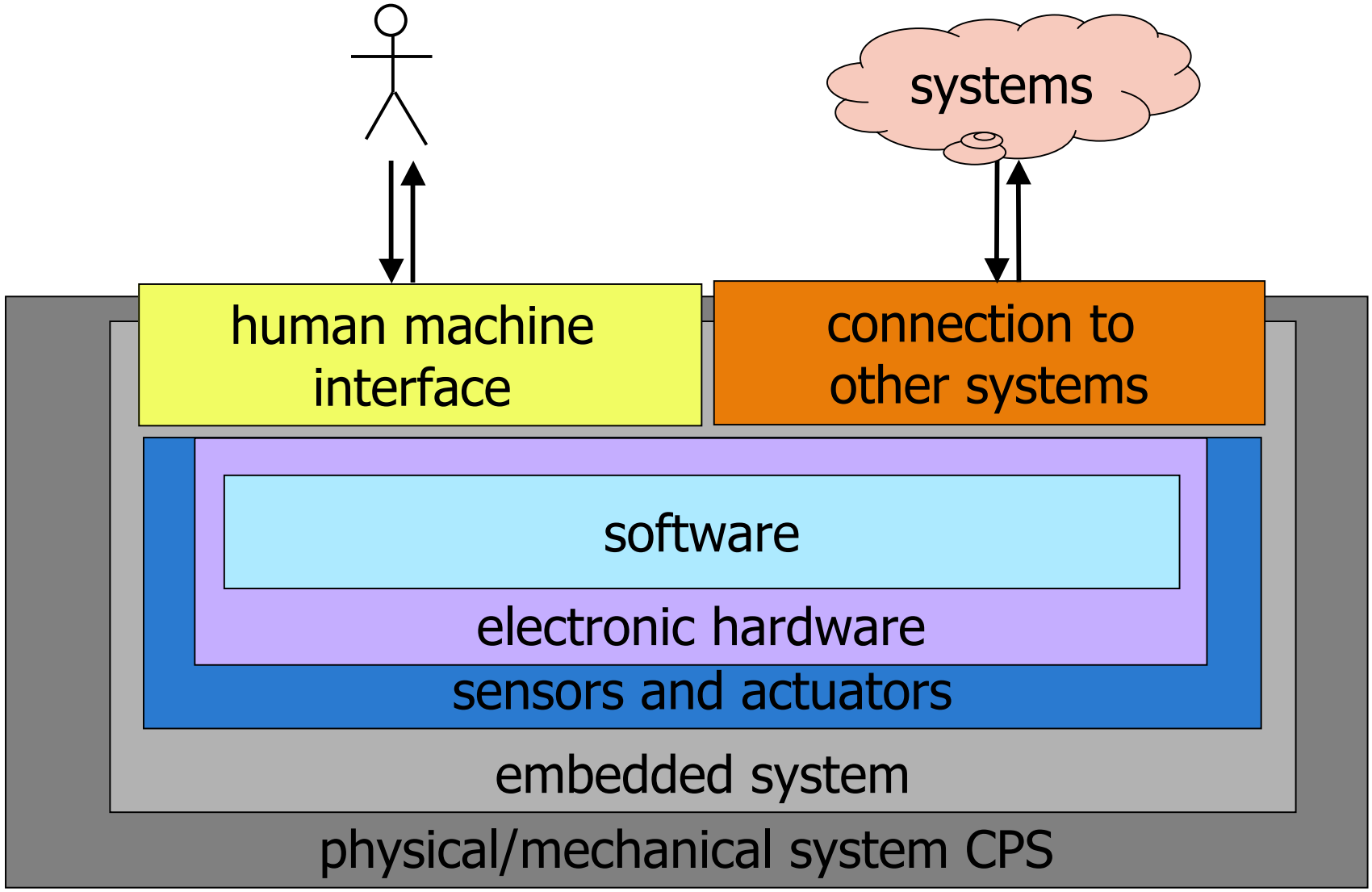
Embedded systems

Smart Embedded Systems

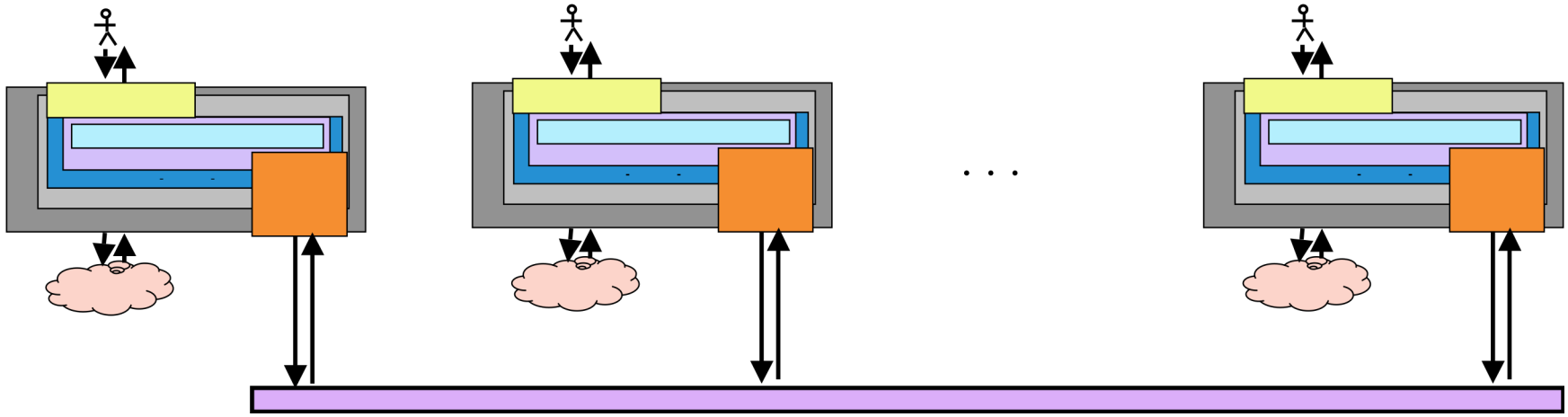
- Analyze environment – situational model
- control response
- user assistance HMI



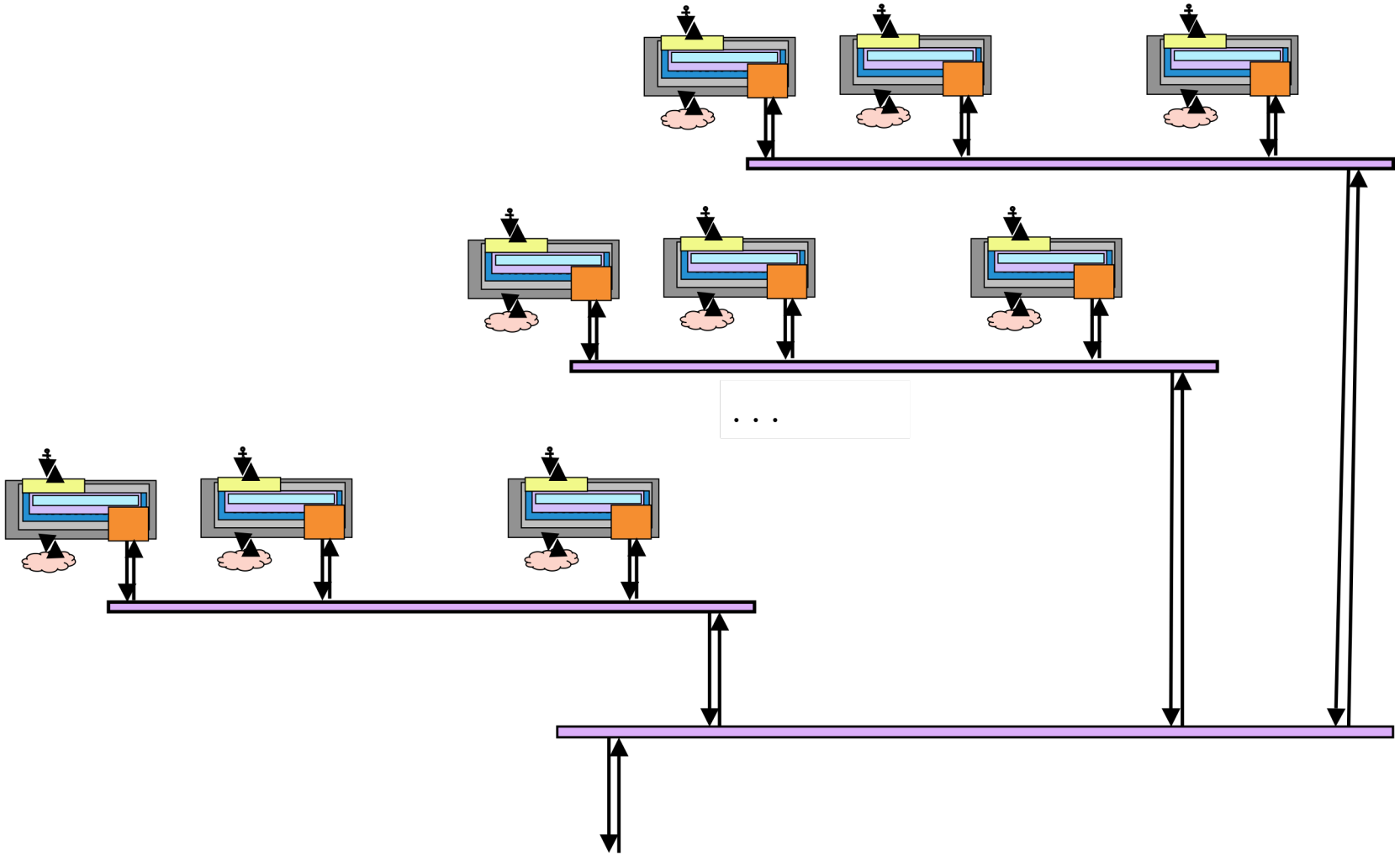
Onion ring like structure of CPS



System of systems



System of systems of systems

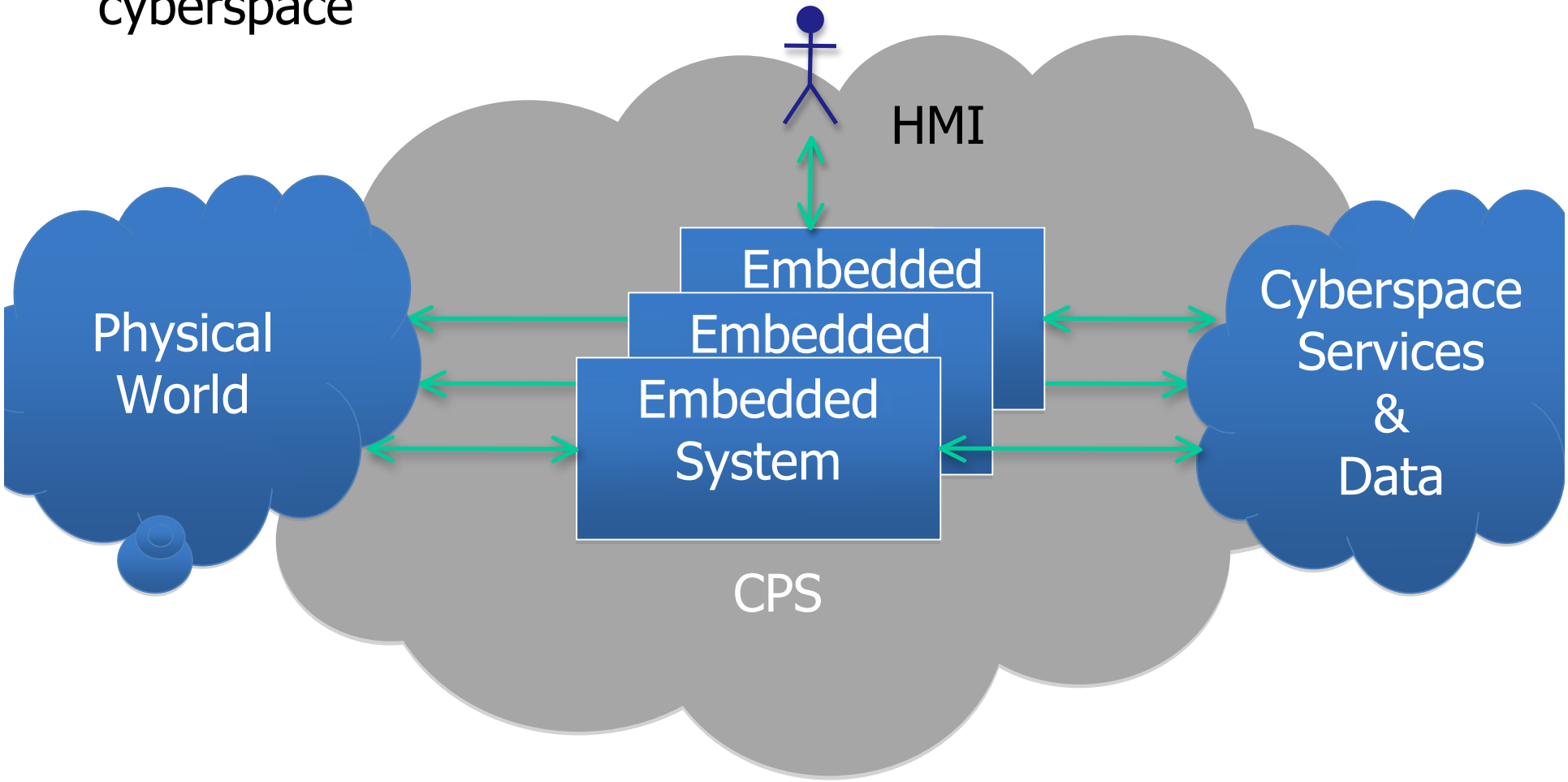


Prominent example for CPS: smart grid

- Embedded control
  - ◇ stable provision of electrical energy
  - ◇ net management
  - ◇ management of energy generation
- New requirements and conditions
  - ◇ prognosis
    - energy production
    - energy consumption
  - ◇ distributed decentralized generation of energy (wind, sun, water, ...)
    - availability depends on weather etc.
  - ◇ consumption oriented pricing
    - consumption depends on social and economical events
    - smart meter
  - ◇ e-mobility
    - relationship to traffic
    - ...

Real World Awareness in the Web

Interfaces to users, to the physical world and to cyberspace



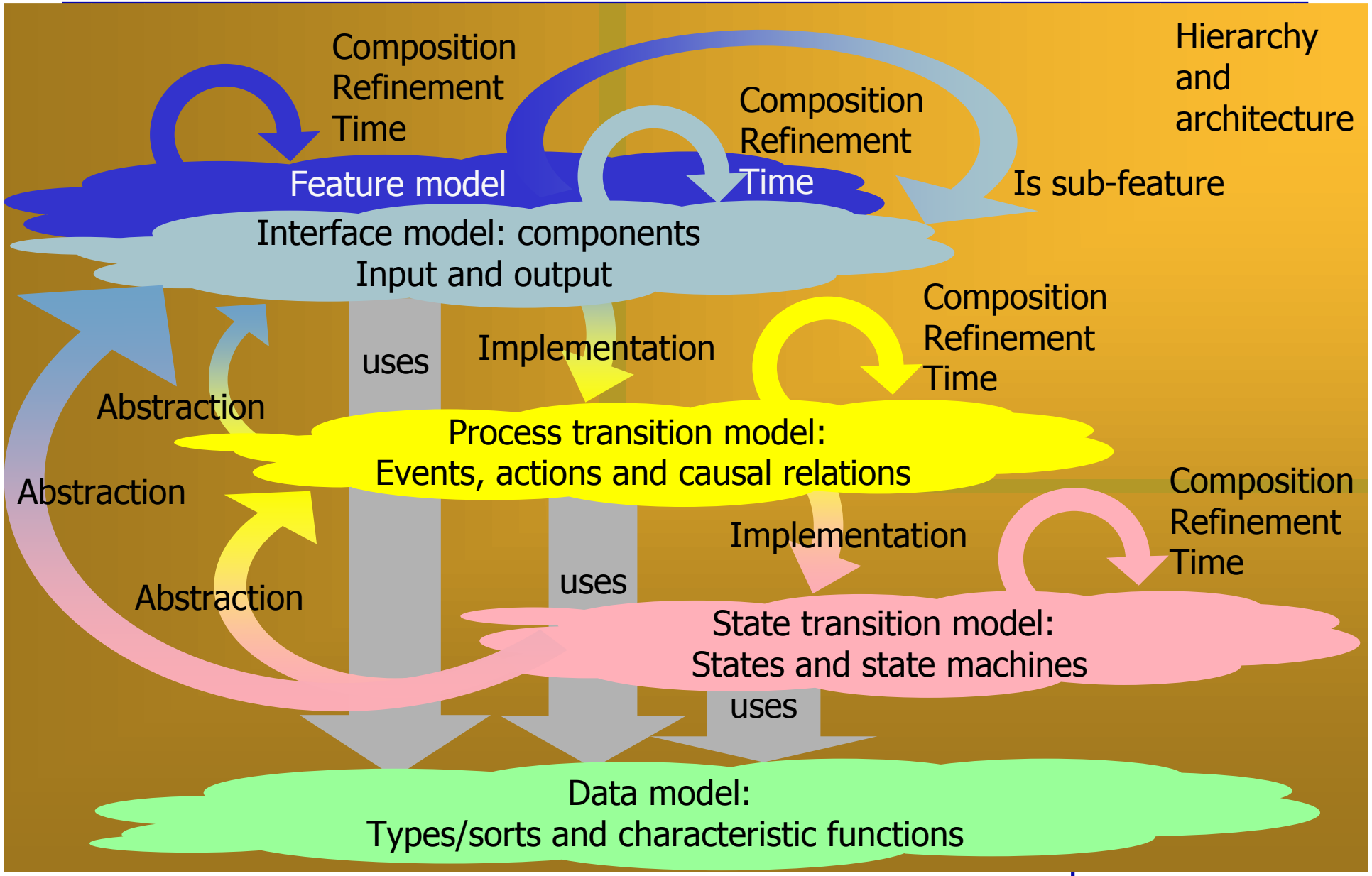
Modelling CPS: Motivation

- Why modelling? – A basis for engineering
  - ◇ Abstraction
    - complexity reduction
    - implementation unbiased
  - ◇ Structuring
  - ◇ Automation
    - advanced tool support
    - reuse
- Why seamless modelling?
  - ◇ optimized integrated use of models over all phases of development
  - ◇ integrated tool support
- Why formal modelling?
  - ◇ precision
  - ◇ automation

What is seamless modeling

- Integrated model framework
  - model theory
  - description techniques
  - tool support
- Seamless usage of models in the development process
  - requirements engineering
    - data models
    - functional specification model
    - quality model
  - architecture design
    - component hierarchy
    - component interfaces
  - state (machine) views
  - refinement and tracing

Towards a comprehensive theory of system modelling: meta model



What is a hybrid (discrete and continuous) CP system?

A system

- has a scope - boundary
- a behaviour functional view: an interface and an interface behaviour
  - ◇ input and output via ports, channels, events, messages, signals
  - ◇ discrete and continuous time
  - ◇ histories – discrete and continuous
  - ◇ functional is what we can observe at the interface
- a structure and distribution: a glass/white box view (including differential equation models)
  - ◇ architecture
  - ◇ state and state transition
- quality profile

Change of paradigm in engineering CPSs

There is a high degree of

- innovation in functionality

in CPSs but also

- increasing costs and complexity

in the design of CP systems, which asks for new approaches and paradigms for engineering and development:

- Systems Engineering
  - ◇ instead of assembling components - integration of subsystems

  requires emphasis on
  - ◇ requirements engineering
  - ◇ architecture and integration
  - ◇ comprehensive quality assurance
- Function orientation
  - ◇ instead of developing components developing functions
  - ◇ functional view part of architecture

Paradigm shift development CPS

… and new development principles:

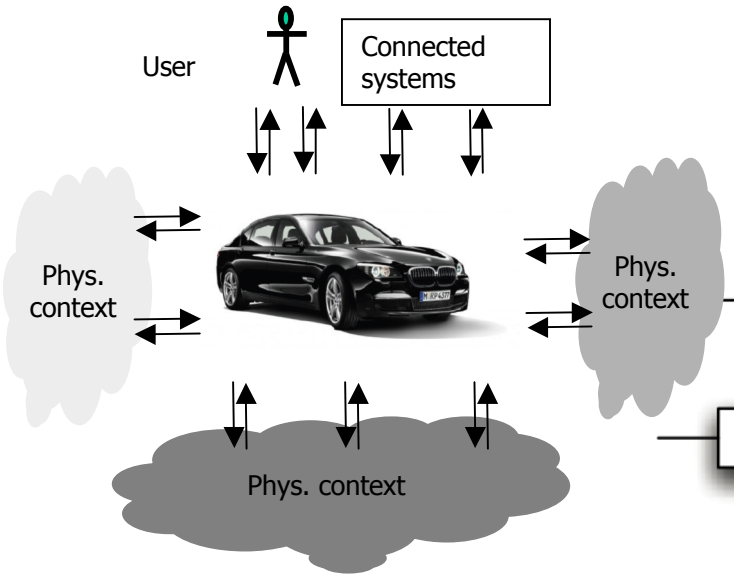
- Front loading
  - shift in expenditure on early phases
  - instead of eliminating errors in the integration error prevention
- Model based development
  - structuring
  - automation
  - seamless use of all models
    Example: Functional models for testing, diagnostics, maintenance
- Artefact orientation - PLM E/E
  - archiving of all development results in databases
- Product lines
  - modular function construction kit
  - mastering variability
  - systematic reuse at all levels

Comprehensive architecture - what is it

Views onto structure - structuring views of a system

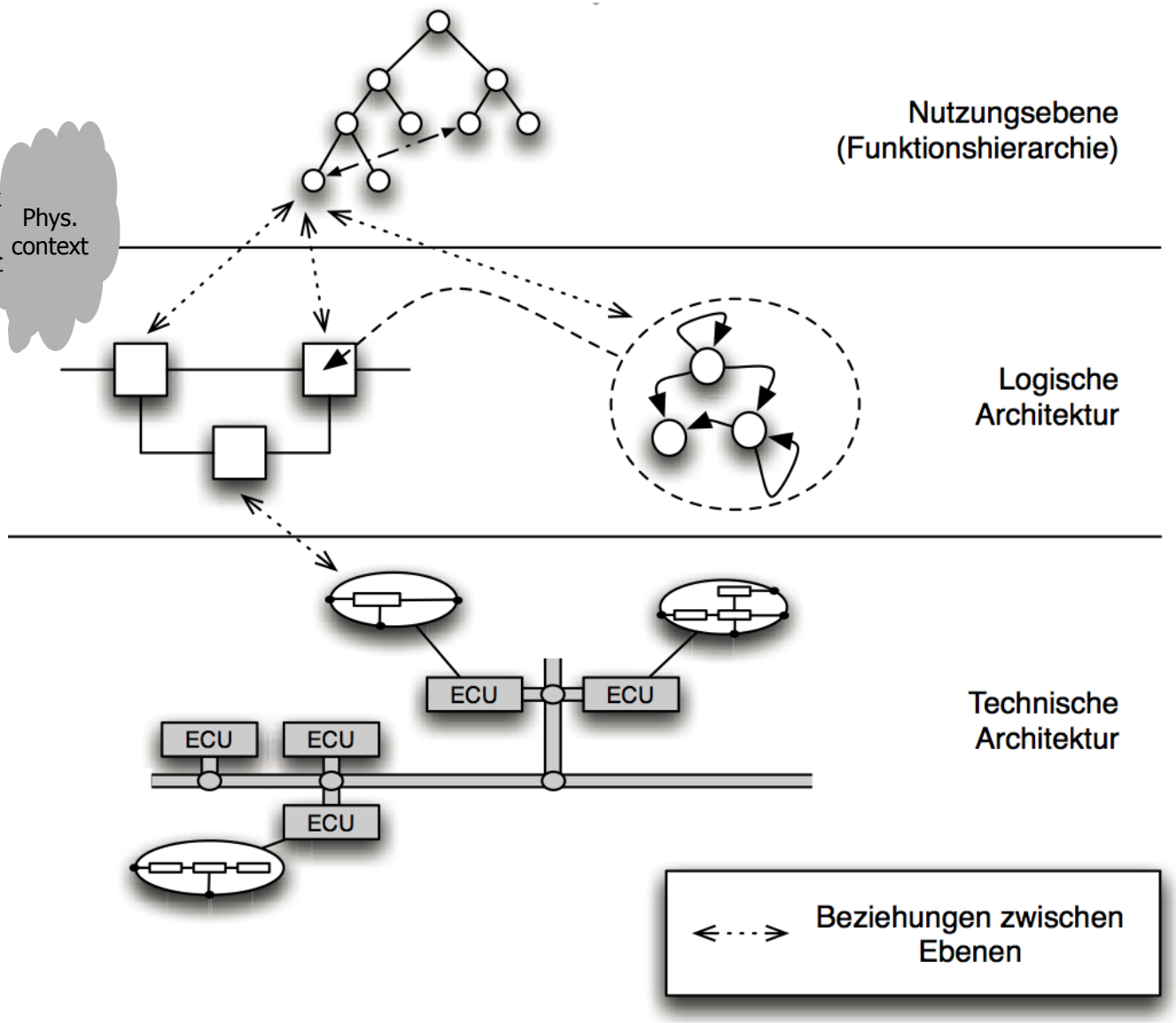
- context - domain model
  - ◇ relevant properties of the system environment
- functional view - system level interface
  - ◇ functionality - function hierarchy
  - ◇ dependencies
  - ◇ non-functional requirements (quality: safety, reliability, performance, ...)
- logical sub-system view
  - ◇ architecture of components - component hierarchy
  - ◇ Logic of the signal / message flow between components
- technical view
  - ◇ deployment, scheduling

Structured architecture views: abstraction levels

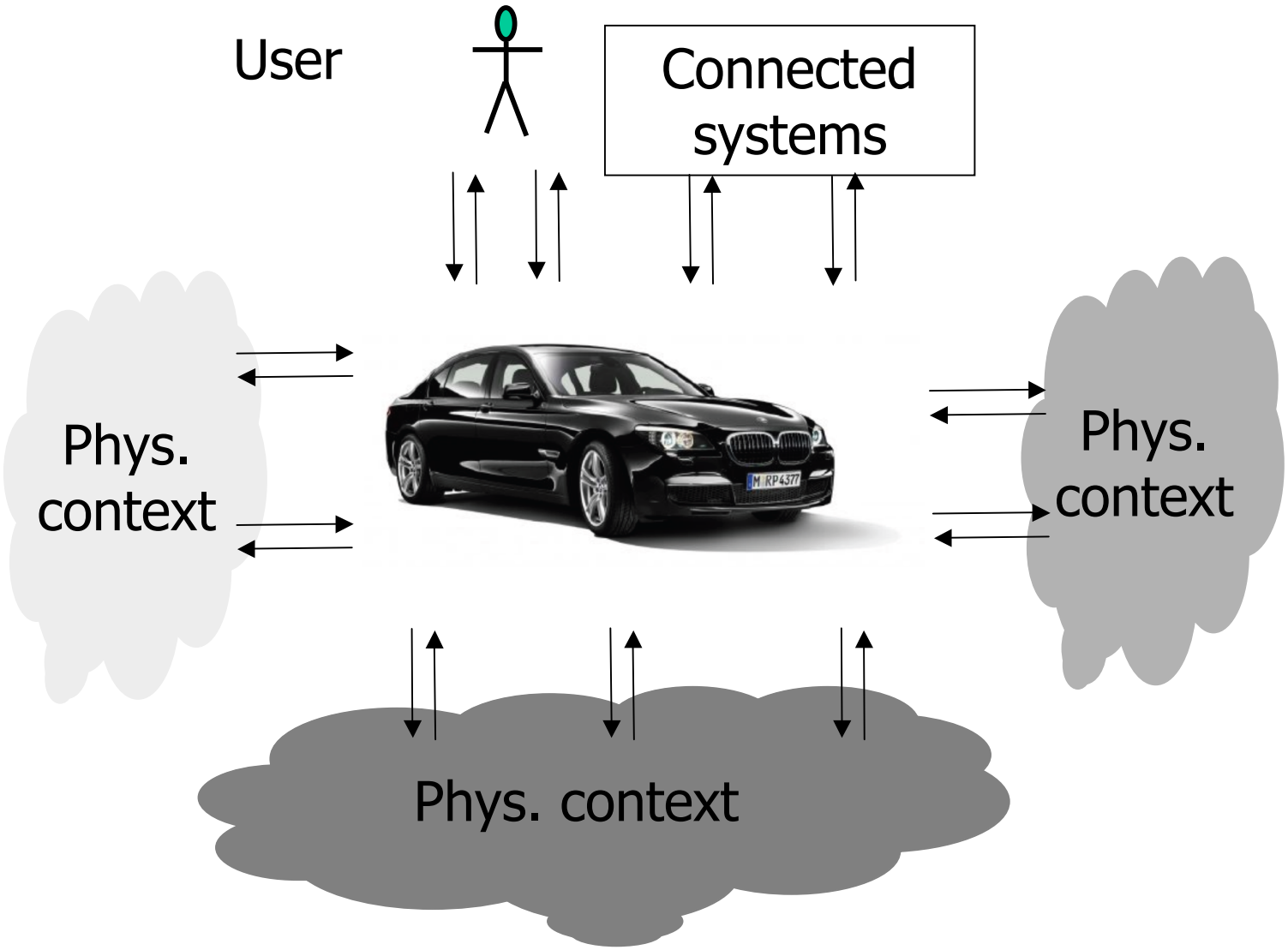


Requirements

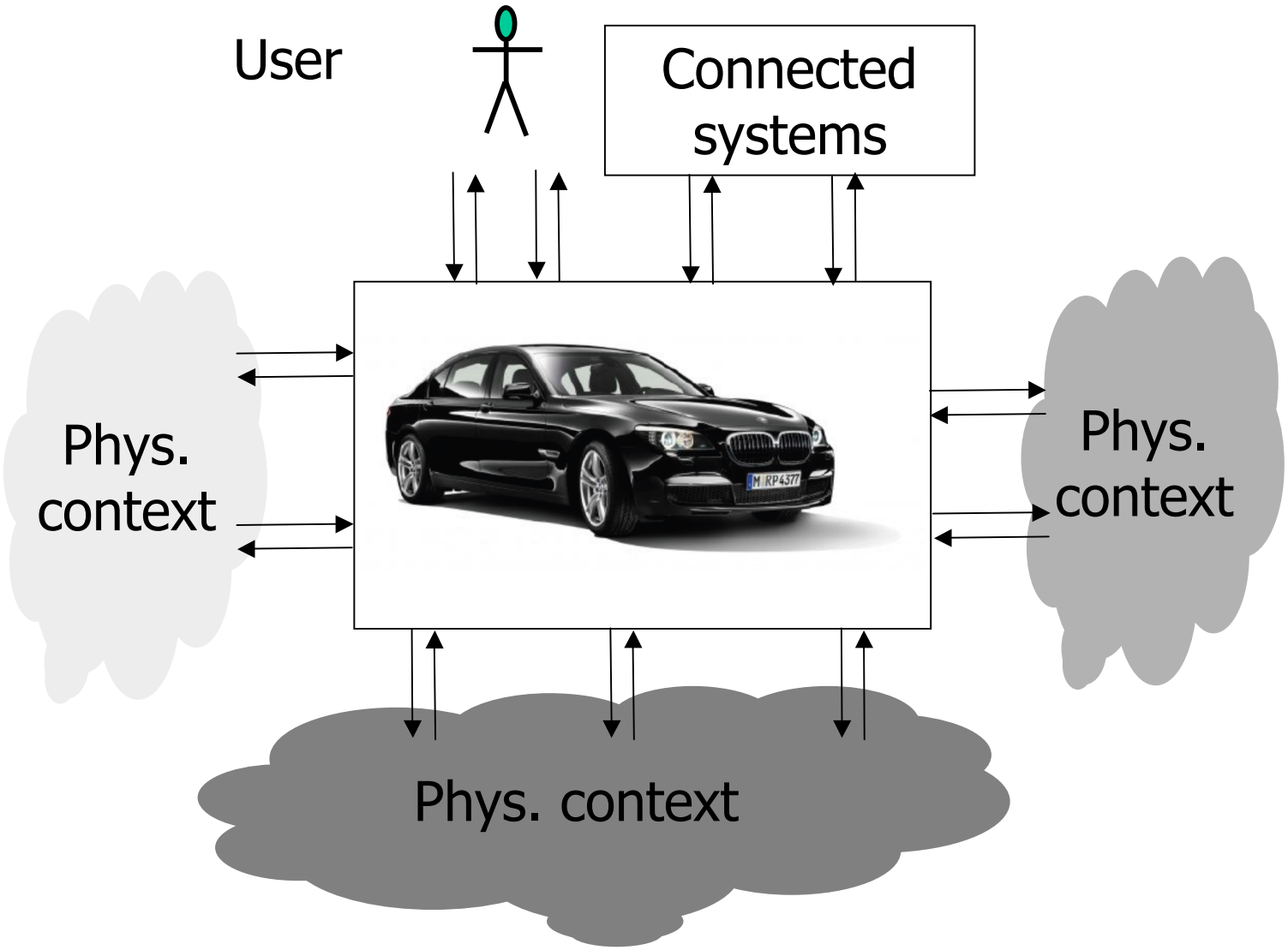
	Interface assertion	Safety	Priority	Component	Function
R_1	...	Yes	high		
R_2	...	No	medium		
R_n	...	no	low		



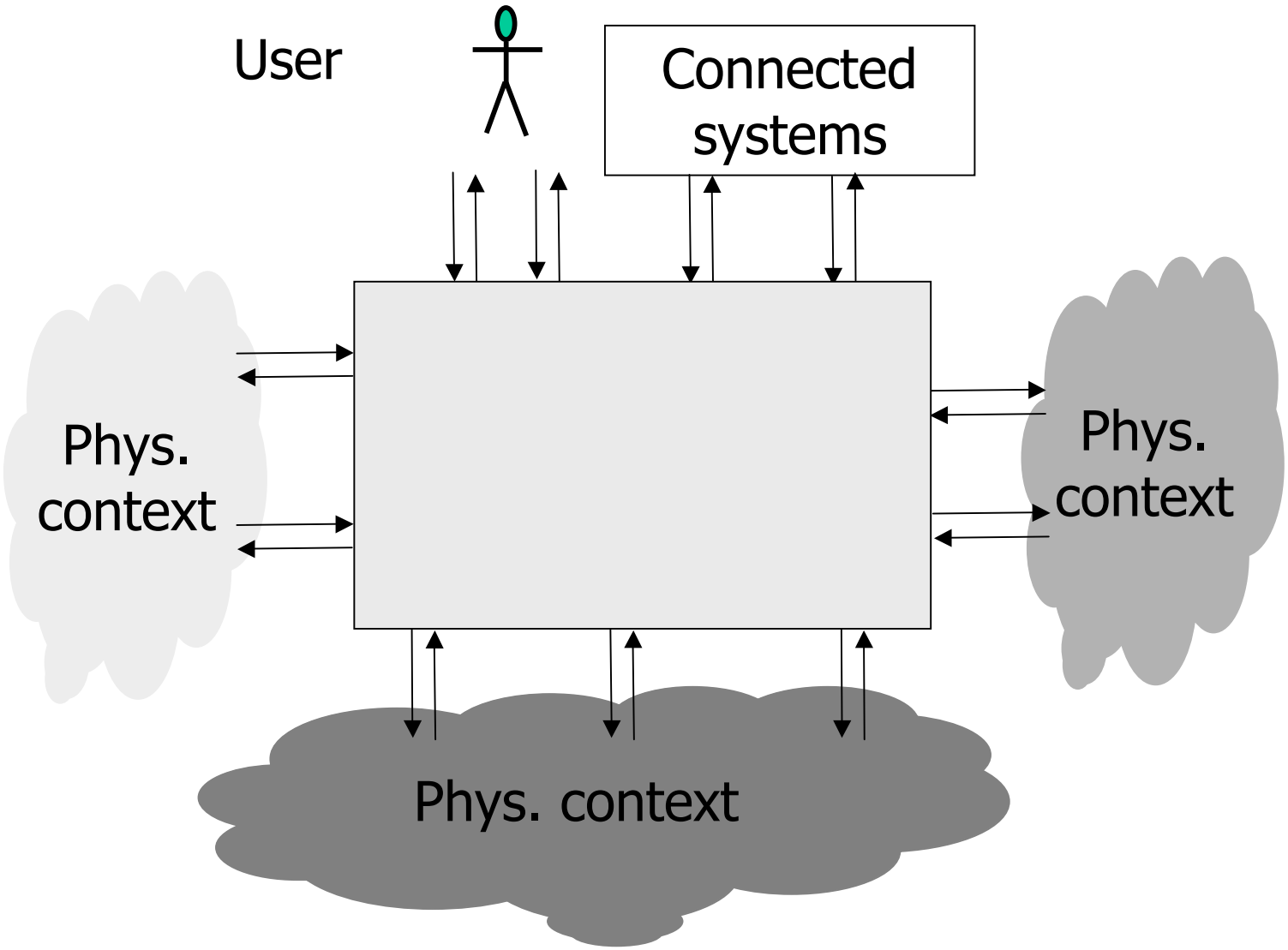
Structuring functionality



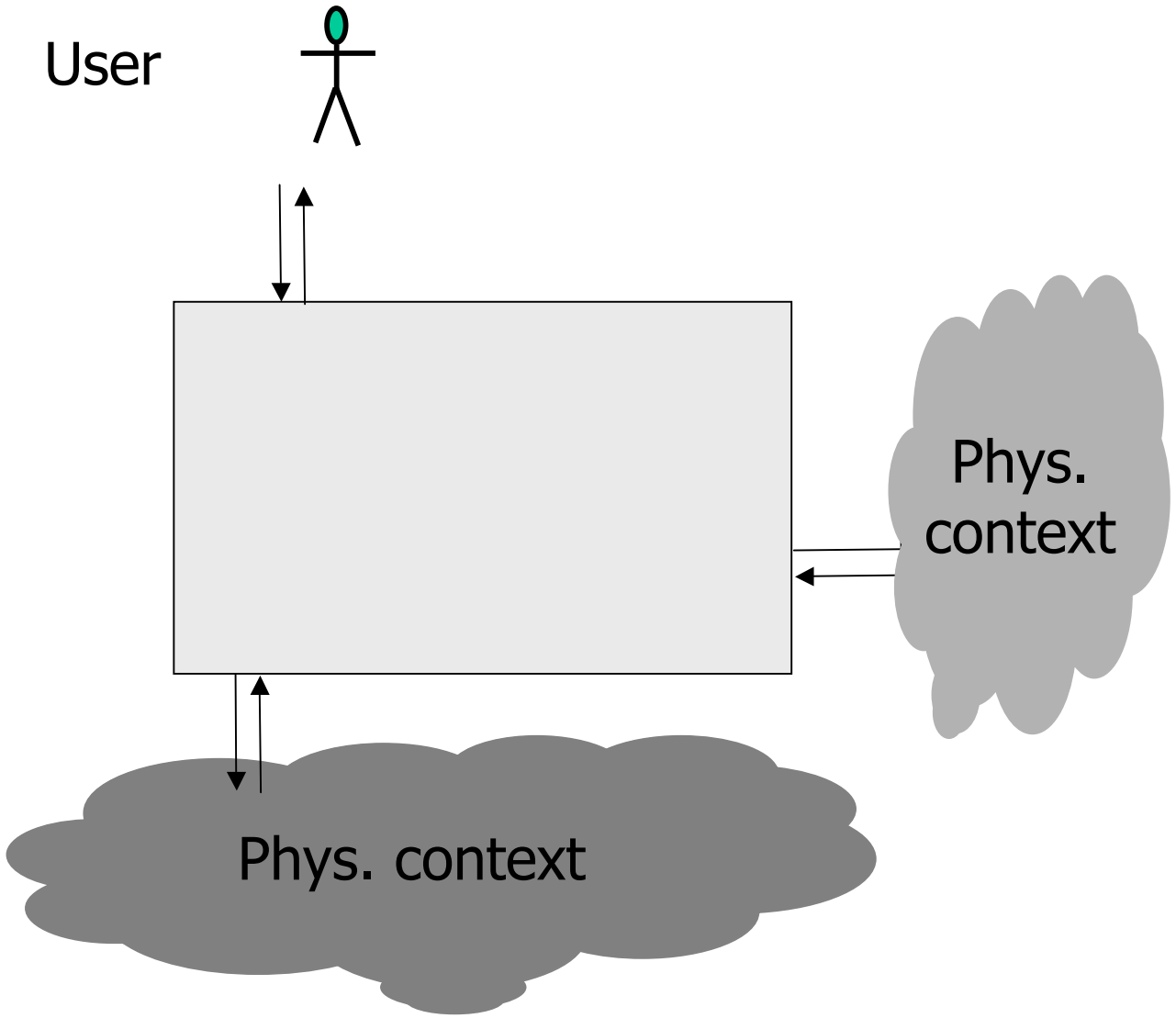
Structuring functionality



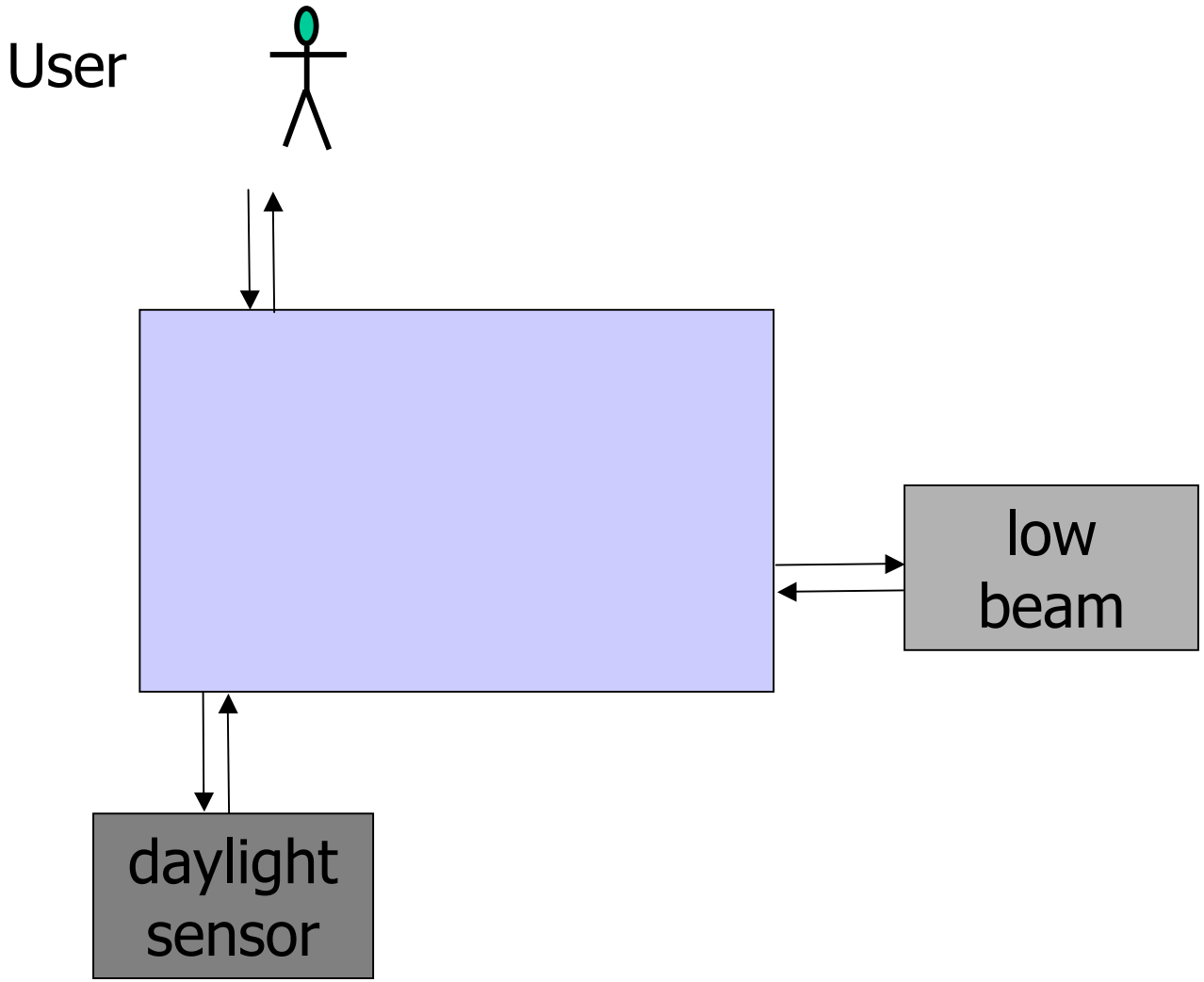
Structuring functionality



Structuring functionality



Structuring functionality in single functions



The role of the conceptional architecture in development

- Function hierarchy/service taxonomy:

  The function hierarchy is to be specified in the requirements engineering

  It comprises (models) all functional requirements

- Logical architecture

  The has to be worked out in the design phase

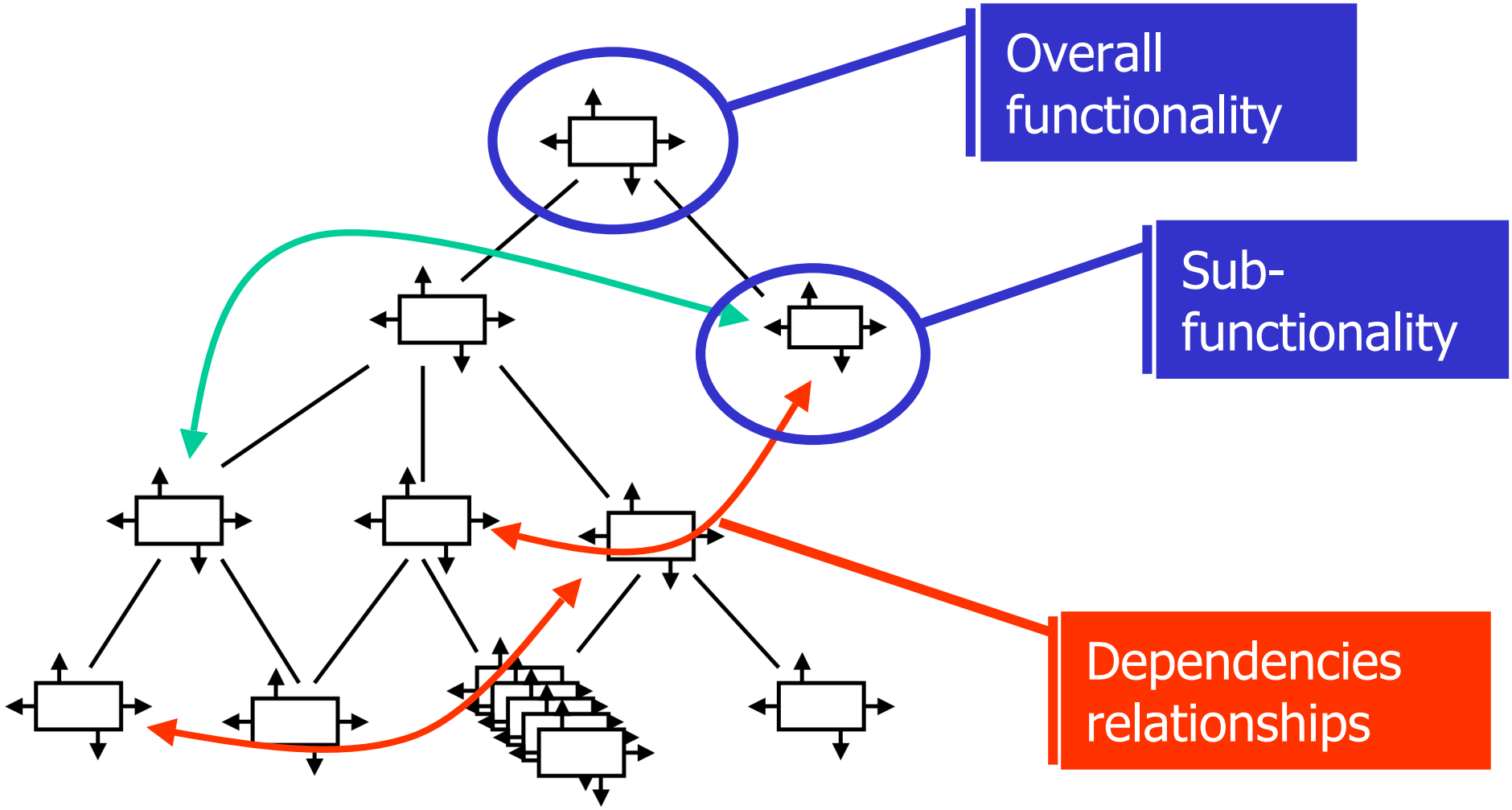
  It comprises the decomposition of the systems in a hierarchy of sub-systems (logical components) fixing their logical roles

What is a function?

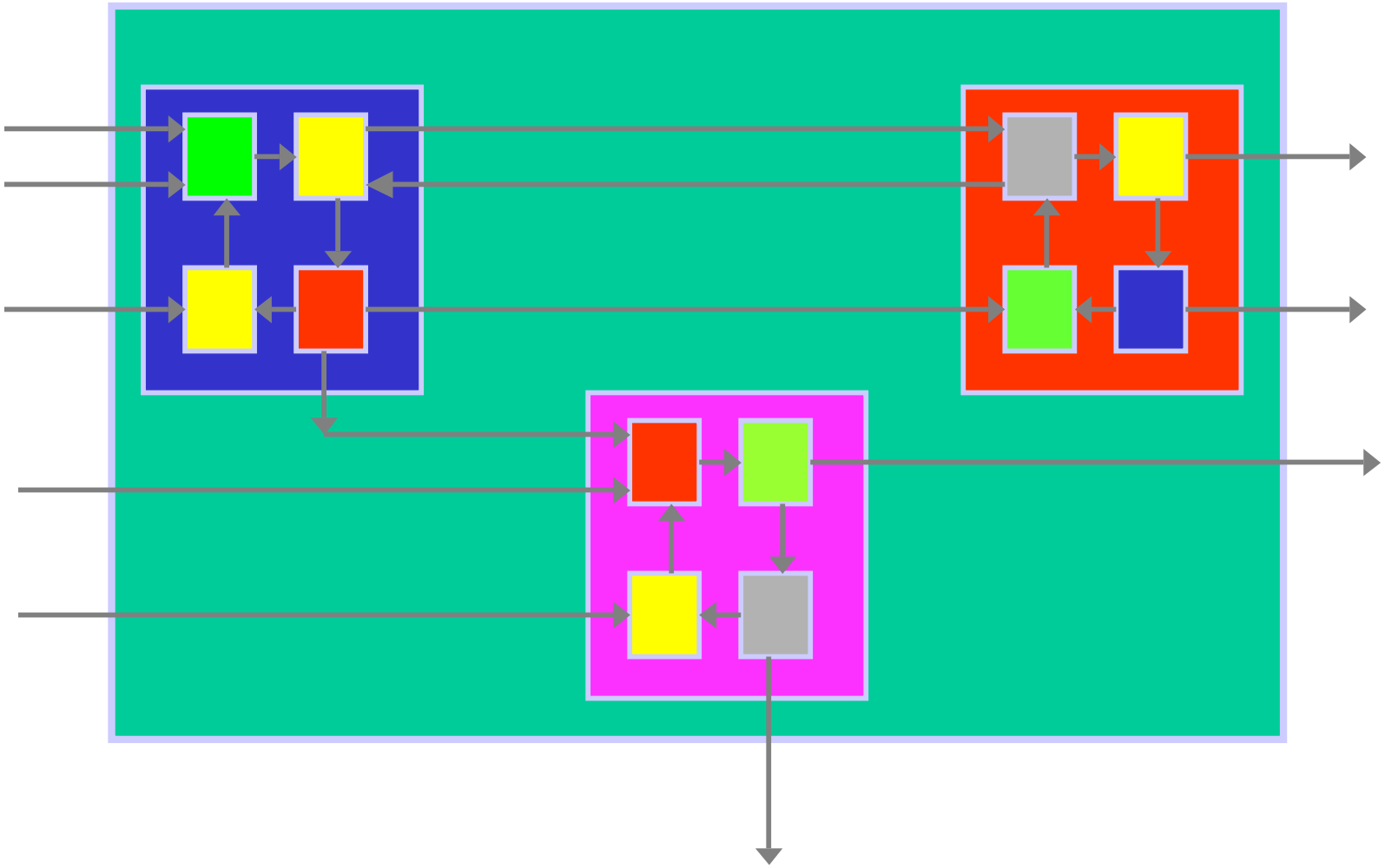
What is a sub-system?

Sub-function ≠ Sub-system

Hierarchy of usage functions („services“)



A logical component architecture: sub-system hierarchy



Key question for system design: modularity and hierarchies

- Note: The principle of hierarchical decomposition
  - A function is a sub-function is a function
  - A system is a component/sub-system is a system
- What does it mean that
  - a system (component) S offers a function F?
  - The the projection of the interface behavior of S to the syntactic interface of function F is (a refinement of) the function F!
- Can we understand the behaviour of a multi-functional system as the hierarchy of the functions it offers?
- How can we capture the dependencies between the functions?

Modes - operating conditions as a missing link

- The individual functions of a vehicle are not logically / functionally independent
  - feature interactions
  - desirable / undesirable
- Collection and presentation of the modes
  - modes: logical operational states of a vehicle
  - example: locking, motor, driving conditions, etc.
  - allows for inclusion of adaptive elements - MMI
- Modular modelling of functions
  - primary in/output of the function
  - modes as input/output to represent the dependencies
  - behaviour as
    - state machine
    - interface representation

Comprehensive Architecture Views: Levels

The structure of software-intensive systems:

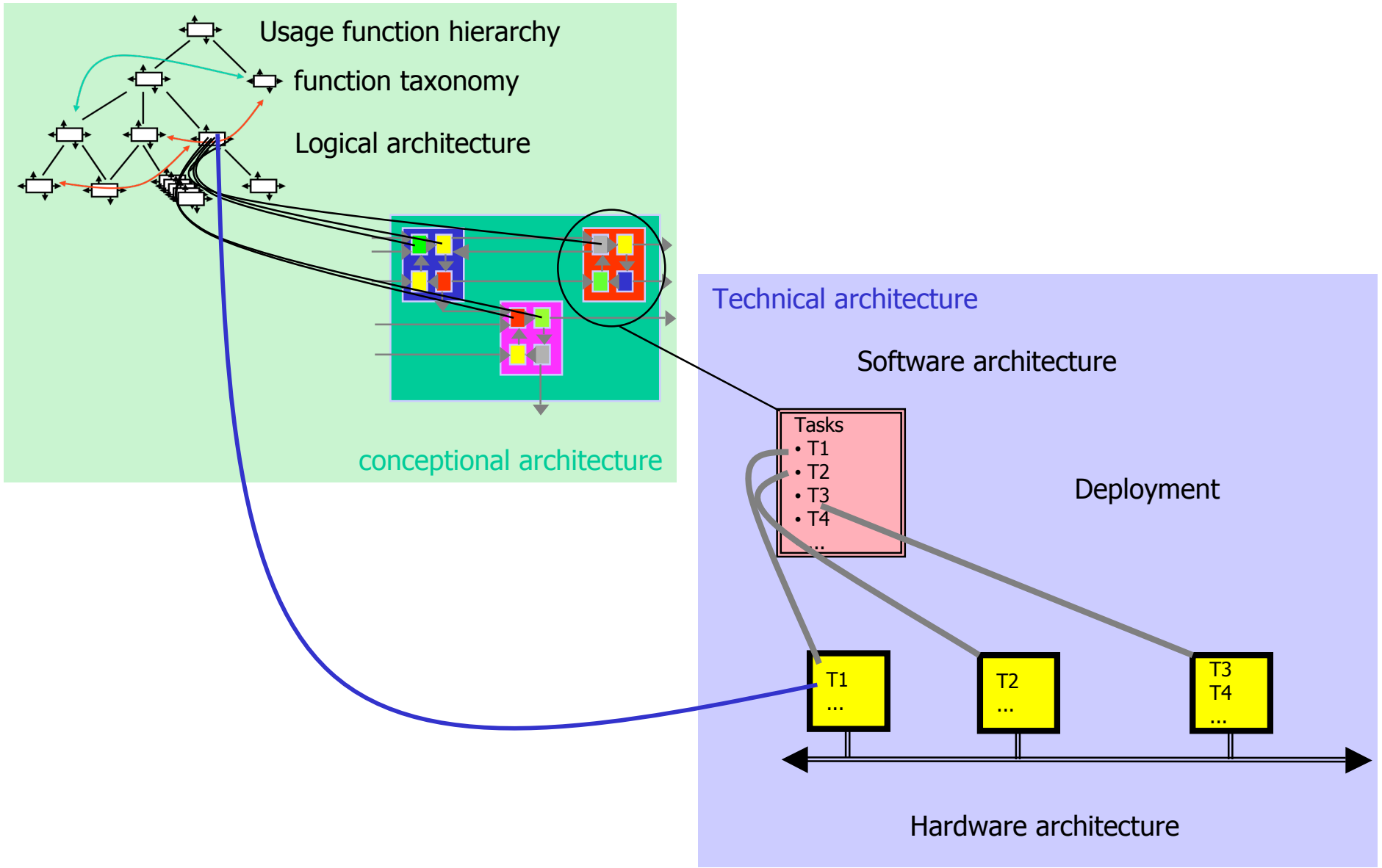
- Functionality: usage view
  - ◇ Multi-functional systems: feature hierarchies
  - ◇ Feature interaction
- Logical component architecture

Conceptional Architecture

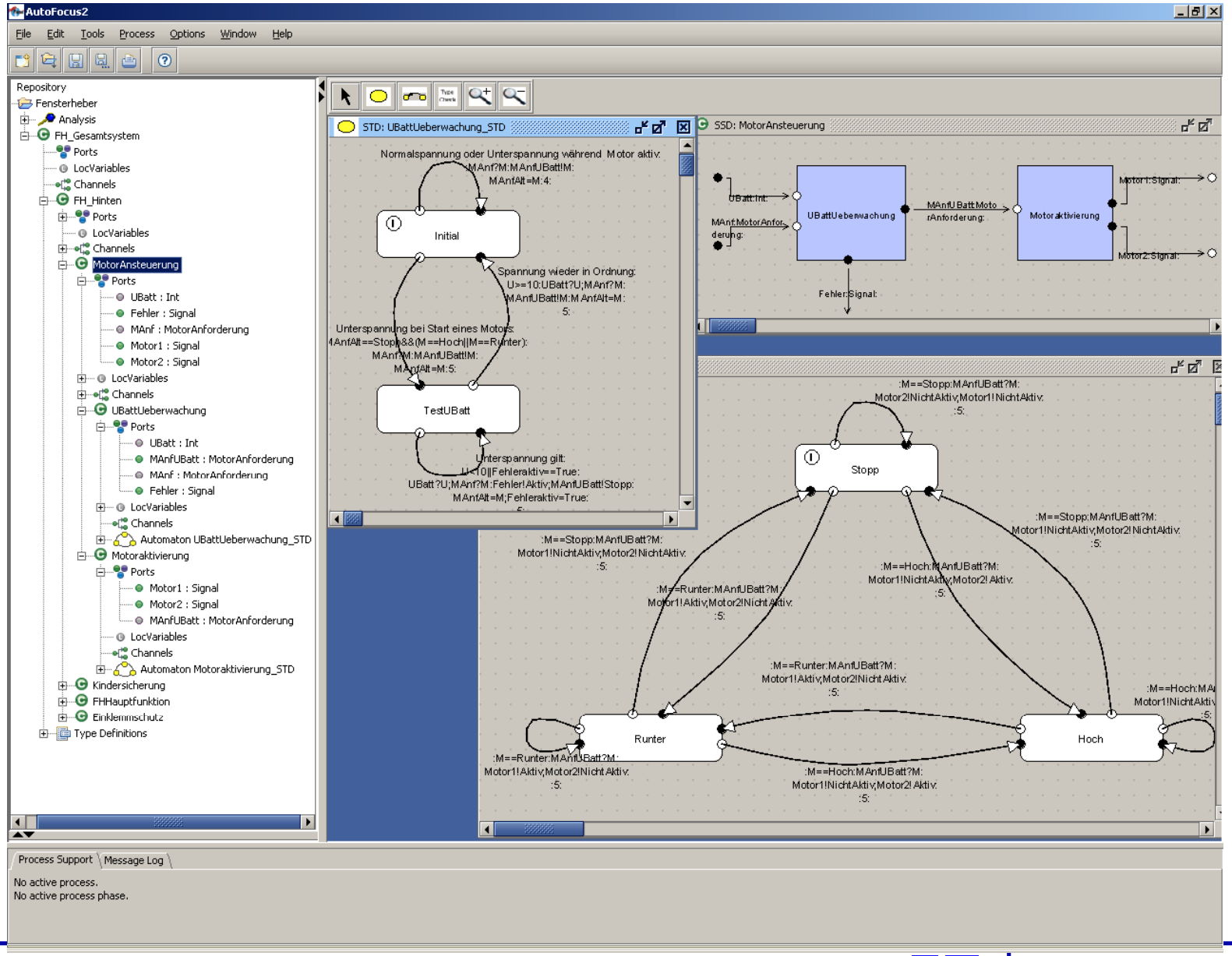
- Software Architecture
  - ◇ Design time software architecture
    - Application software
    - Software platform (OSEK, bus systems)
  - ◇ Run time software architecture
    - Tasks
    - Scheduling
- Hardware Architecture
  - ◇ Controllers
  - ◇ Communication devices
  - ◇ Sensor and actuators
- Deployment

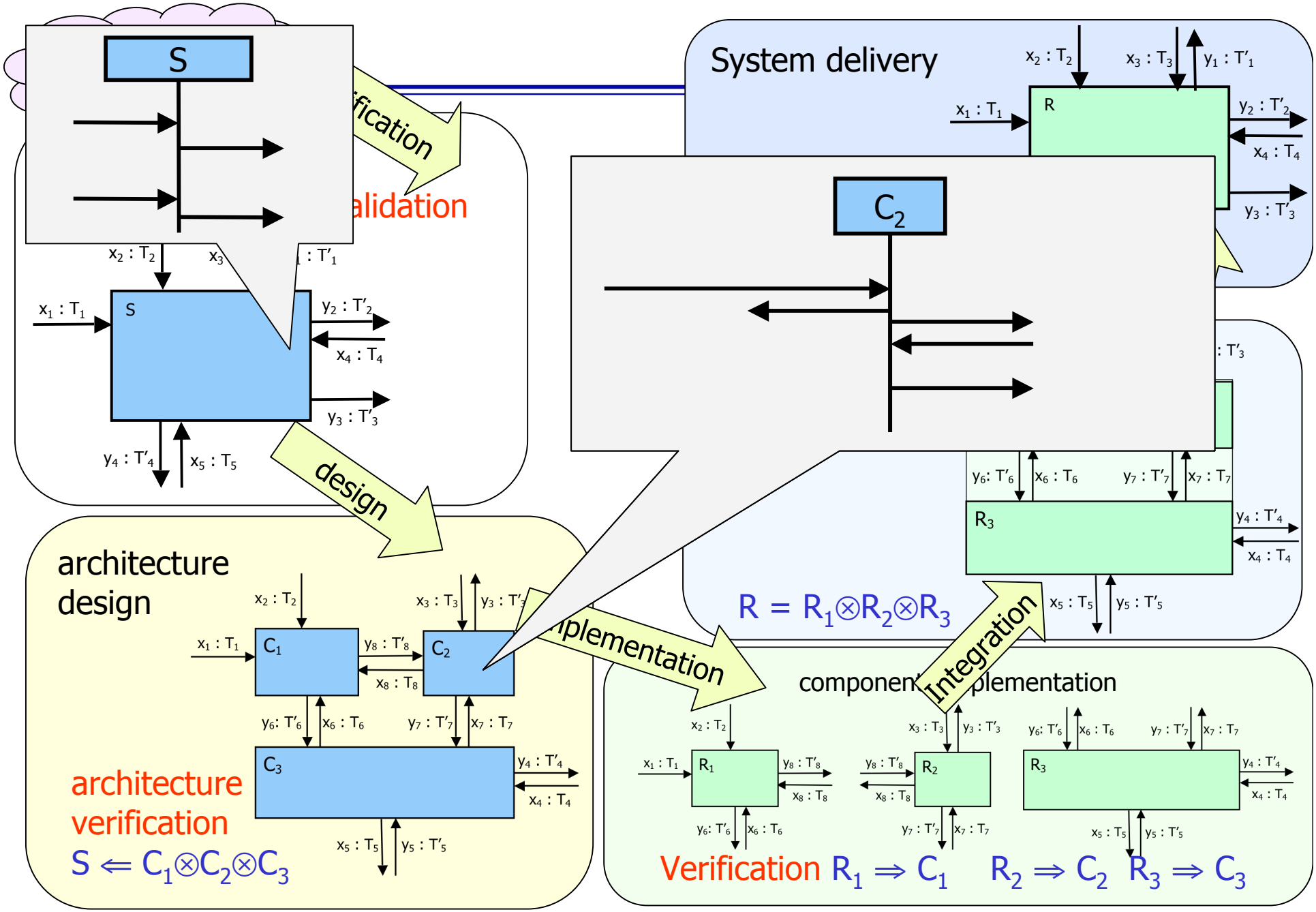
Technical Architecture

The comprehensive model



A screen shot from AutoFocus




S
ification
alidation
System delivery
R
C2
design
architecture
design
C1
C2
C3
architecture
verification
S ⇐ C1⊗C2⊗C3
R = R1⊗R2⊗R3
R3
plementation
Integration
component
R1
R2
R3
Verification R1 ⇒ C1 R2 ⇒ C2 R3 ⇒ C3

Hybrid systems: an interface model

Sets of typed channels

$I = \{x_1 : T_1, x_2 : T_2, \ldots\}$

$O = \{y_1 : T'_1, y_2 : T'_2, \ldots\}$

syntactic interface

$(I \blacktriangleright O)$

data stream of type T

$STREAM[T] = \{IN \to T^*\}$ discrete T – discrete stream

$STREAM[T] = \{IR_+ \to T\}$ dense T – continuous stream

valuation of channel set C

$IH[C] = \{C \to STREAM[T]\}$

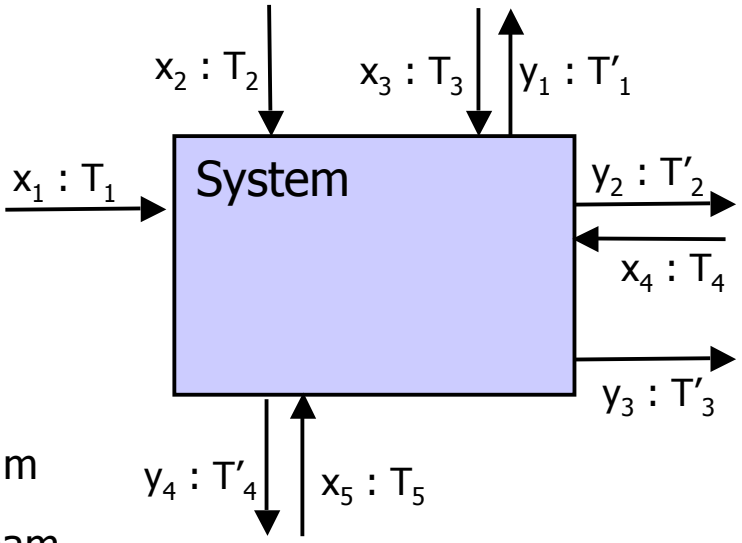
interface behavior for syn. interface $(I \blacktriangleright O)$

$[I \blacktriangleright O] = \{IH[I] \to \wp(IH[O])\}$

interface specification

$p: I \cup O \to IB$

represented by an interface assertion S
a logical formula with channel names as variables for streams



Result: function based structuring/architecture of systems

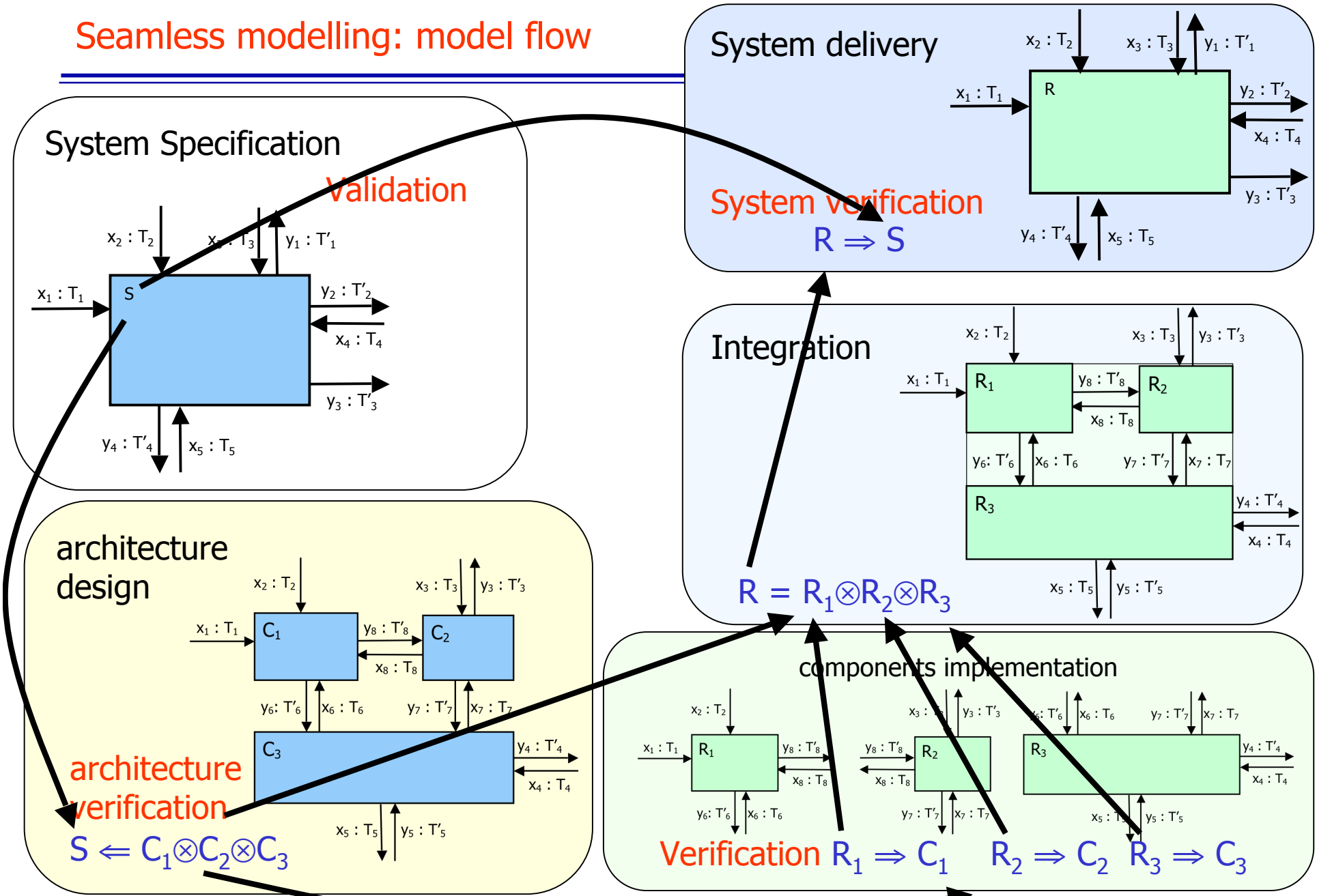
Modeling:

- Function hierarchy
  - ◇ Structured list of all functions
    - user functions
    - system functions
  - ◇ Mode view
  - ◇ Modular specification of each function
    - dependencies by modes
- Logical components (sub-systems)
  - ◇ Tracing: understanding which of the sub-systems and which of their properties contribute to which function
- Technical level
  - ◇ Automatic generation of code
    - Parameterized by technical architecture

Seamless usage:

- Analysis
  - ◇ feature interactions
  - ◇ completeness of specification
- Validation
- Simulation
- Generation of system test cases
- Configuration planning
  - ◇ when is which function available
- Impact analysis
- Generation of integration test cases

Seamless modelling: model flow



CPS - a new engineering paradigm

Not software - systems in the first place – an integrated view

- What is a CP system
  - ◇ a unit of
    - software
    - electronic hardware
    - mechanical parts
- we need a more integrated holistic view onto systems: a theory of CP system modeling: hybrid system theory
  - ◇ interfaces
  - ◇ architectures
    - composition of CP systems
  - ◇ states
- The theory of programming
  - ◇ specification and verification, interfaces, composition, modularity and compatibility, refinement, state, architecture

  is a perfect starting point for such a theory of systems

An algebraic view onto modeling cyber-electromechanical systems

HW:	electronic programmable hardware including sensors, actuators, HMI devices
SW:	software
ITS:	hardware and software integrated (example CPU)
CN:	communication devices – bus systems
MD:	mechanical systems
CPS:	cyber physical systems

$\otimes$ composition

$\otimes$: SW × SW → SW

$\otimes$: HW × HW → HW

...

$\otimes$: HW × SW → ITS

$\otimes$: ITS × ... × ITS × CN → ITS

$\otimes$: ITS × MD → CPS

...

Laws:

$$[md_1 \otimes md_2] \otimes [hw_1 \otimes hw_2] \otimes [sw_1 \otimes sw_2]$$

$$=?=$$

$$[md_1 \otimes hw_1 \otimes sw_1] \otimes [md_2 \otimes hw_2 \otimes sw_2]$$

Re-thinking the role of time

- Ed Lee's structure of an CPS is essentially an embedded system
  - ◇ Observation: a C program sw does not say anything about timing – we need the platform to understand the timing

Observation

$$timing[sw]$$

$$\neq$$

$$timing[hw \otimes sw]$$

Re-thinking the notion of "functional requirements"

- Time should be part of behavior – but there is a difference
  - ◇ specification and implementation
    - timing as requirement – hard real time
    - timing as property of execution
  - ◇ between hard and soft real time
- What is functional is in the eye of the beholder:
  - ◇ wide range of observations (temperature, weight, speed, …)
  - ◇ time – discrete and continuous
  - ◇ today is tomorrow: timing as a build in property of models of programs and systems
- What is called "functional" is what is modeled by the functional view by the interface behavior including
  - ◇ qualitative views: classical concepts of correctness including time
  - ◇ quantitative views: probability, performance, safety, ...
- What is "non-functional" is what cannot be seen in the functional view – modeled by the interface behavior of a CPS

acatech
DEUTSCHE AKADEMIE DER TECHNIKWISSENSCHAFTEN

acatech – GERMAN ACADEMY OF SCIENCE AND ENGINEERING

Research Agenda CYBER-PHYSICAL SYSTEMS

Manfred Broy
Technisch Universität München
acatech, Germany

acatech project "Research Agenda Cyber-Physical Systems"
Project Data

Funding by:	Federal Ministry for Education & Research (BMBF)
Duration:	18 months (May 1, 2010 - October 31, 2011)
Funding amount:	0,7 Mio. Euro + in kind funding by German companies
Project lead:	Prof. Dr. Dr. h.c. Manfred Broy, Technical University Munich
Technical lead:	fortiss – Innovation Center for Software-intensive Systems
Coordination:	acatech
Project Partners:	Intel Deutschland GmbH (supporting) Robert Bosch GmbH (supporting) BMW AG (supporting) DTAG (supporting) Siemens AG EADS Deutschland GmbH Festo AG & Co. KG ESG Elektroniksystem- und Logistik GmbH SAP AG Software AG BITKOM VDMA ZVEI

The acatech Project agenda CPS

- Organisation
  - Based on German Road Map Embedded Systems
  - Sponsored by German BMBF, Intel, BMW, Bosch, …
  - In cooperation with Siemens, EADS, ESG
- Goals
  - Future scenarios of CPSs
  - Needed capabilities
  - Core technologies
  - Research agenda
- Schedule
  - Deliver results in autumn/winter 2011

Aspects beyond technology …

CPS as drivers of change …

- Law
- Politics
- The human factor
  - HMI
  - Social networks and CPS
  - User acceptance issues
    - privacy
    - complexity
    - …

Concluding remarks: the bottom line ...

- CPSs are more than embedded systems
  - integrated cyber-mechanical systems consisting of mechanics/hardware/software

- Connecting cyber-mechanical systems to the internet and www brings in a new dimension of
  - Research questions
    - interoperability
  - Innovative application opportunities