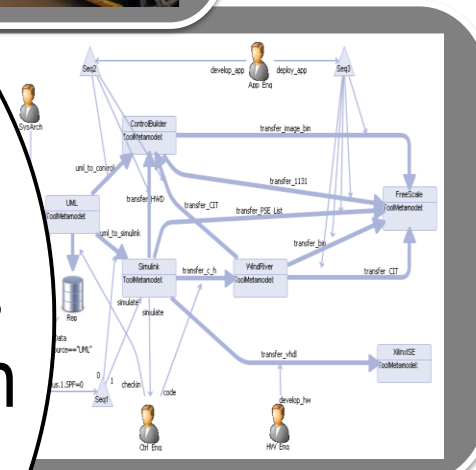
Integrating Viewpoints in the Development of CPS

Martin Törngren
Professor in Embedded Control Systems, ICES director
Division of Mechatronics, KTH – Royal Institute of Technology

Joint work with Matthias Biehl, Jad Elkhoury, Frederic Loiret, Magnus Persson and Ahsan Qamar, KTH, and Stavros Tripakis, Patricia Derler and Edward Lee, UC Berkeley

Research areas in the group



ECS Group:

Autonomy

System Architecture

Design and Optimization

Innovative Product Concepts

Model Based Design

Multiview Modeling

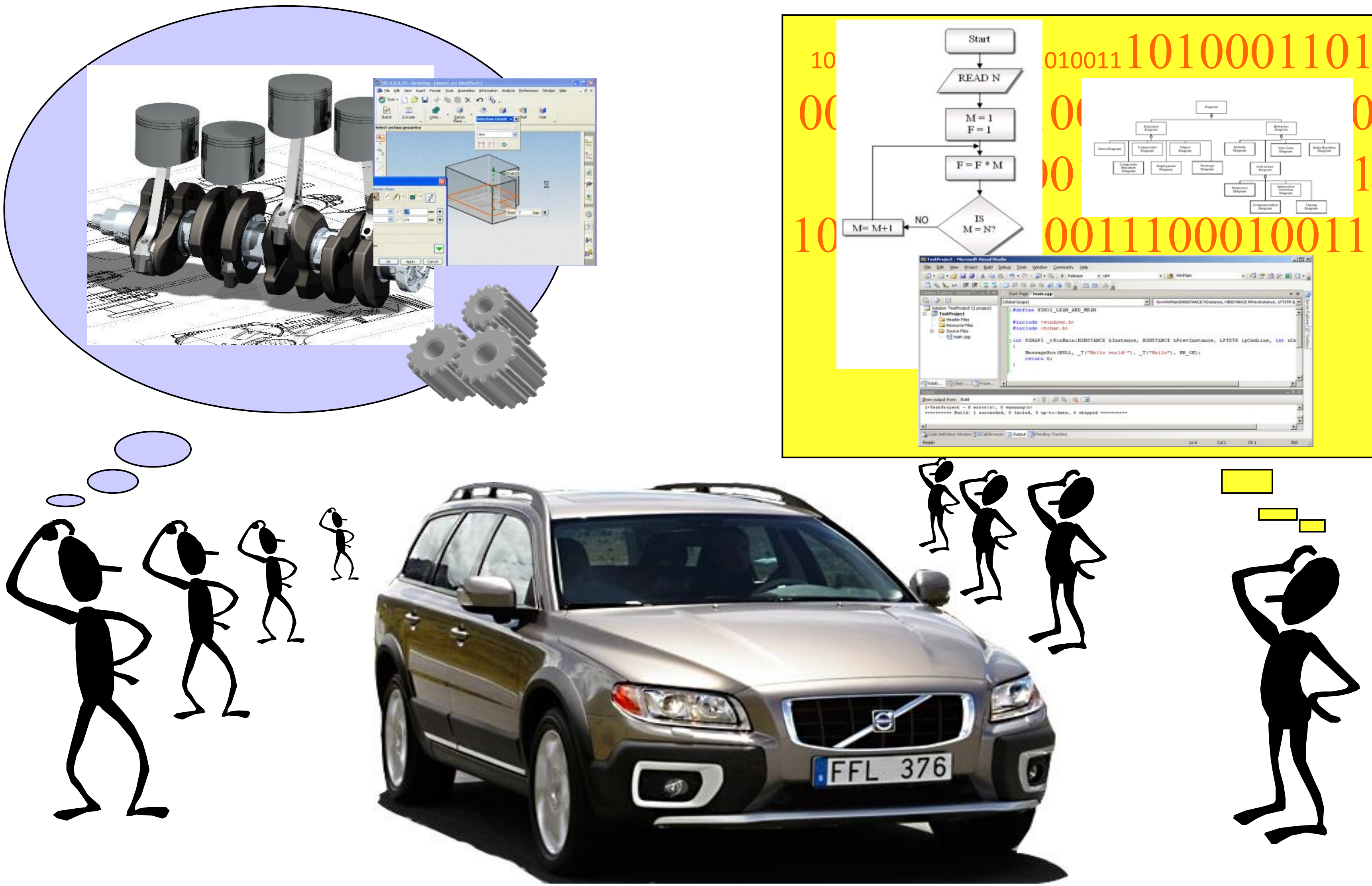
Tool Integration

Safety

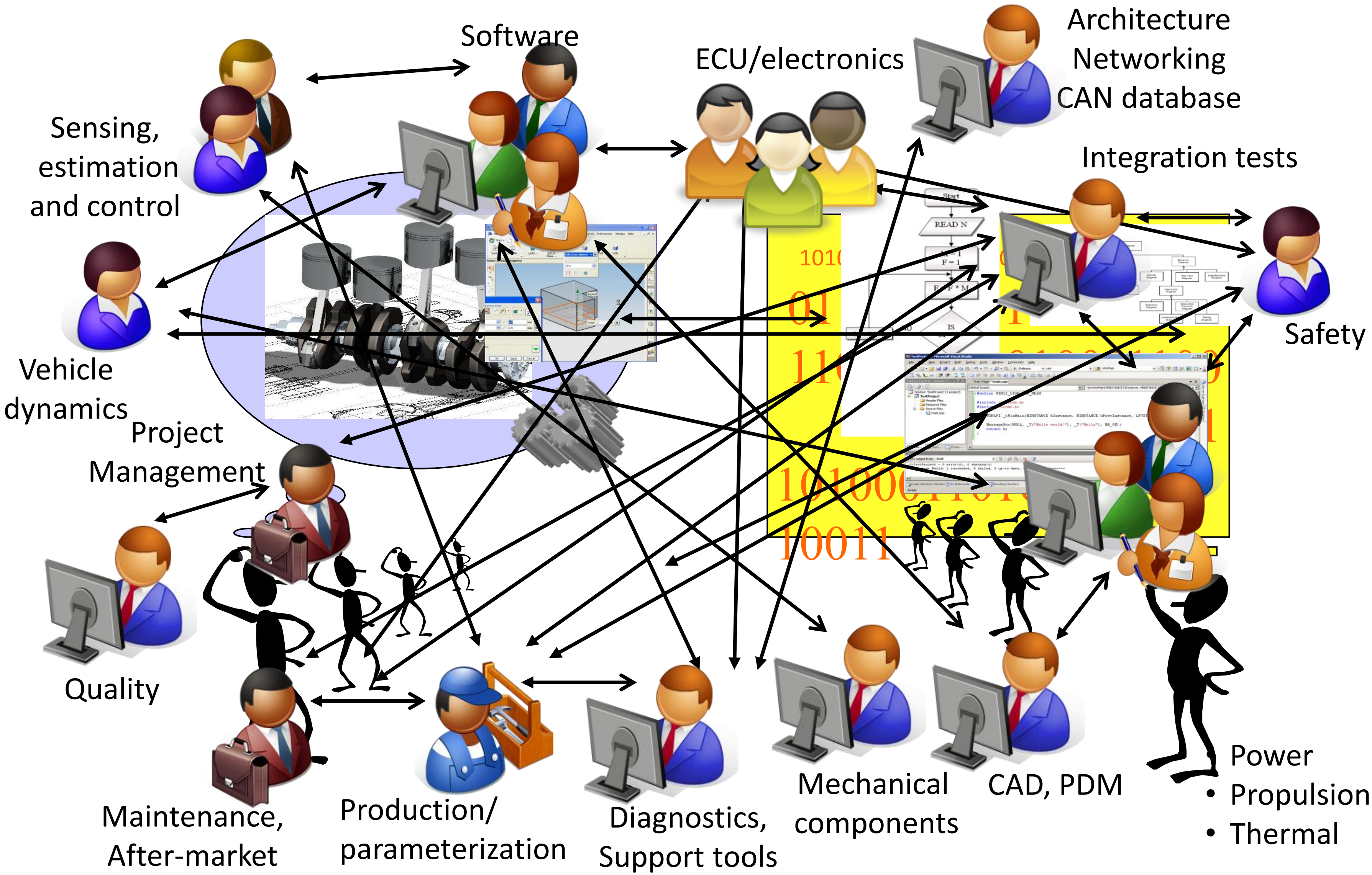
CPS like demo

Manifested by, and courtesy of, the Berkeley juggling club

Different viewpoints and concerns



Viewpoints and relationships

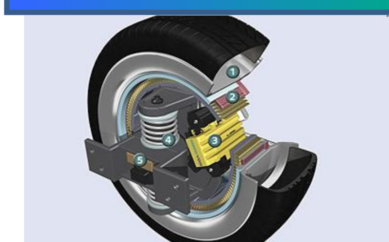


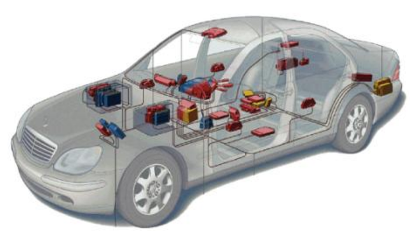
Main messages

CPS come with growing functionality, extent and concerns

- More advanced engineering environments:
  - **viewpoints** and **relations** between them
- We need to engineer and deal with **viewpoint systems**
  - **Viewpoint contracts**
  - **Dependency modeling**
  - **Systematic and efficient tool integration**

Beyond traditional stakeholders





Outline

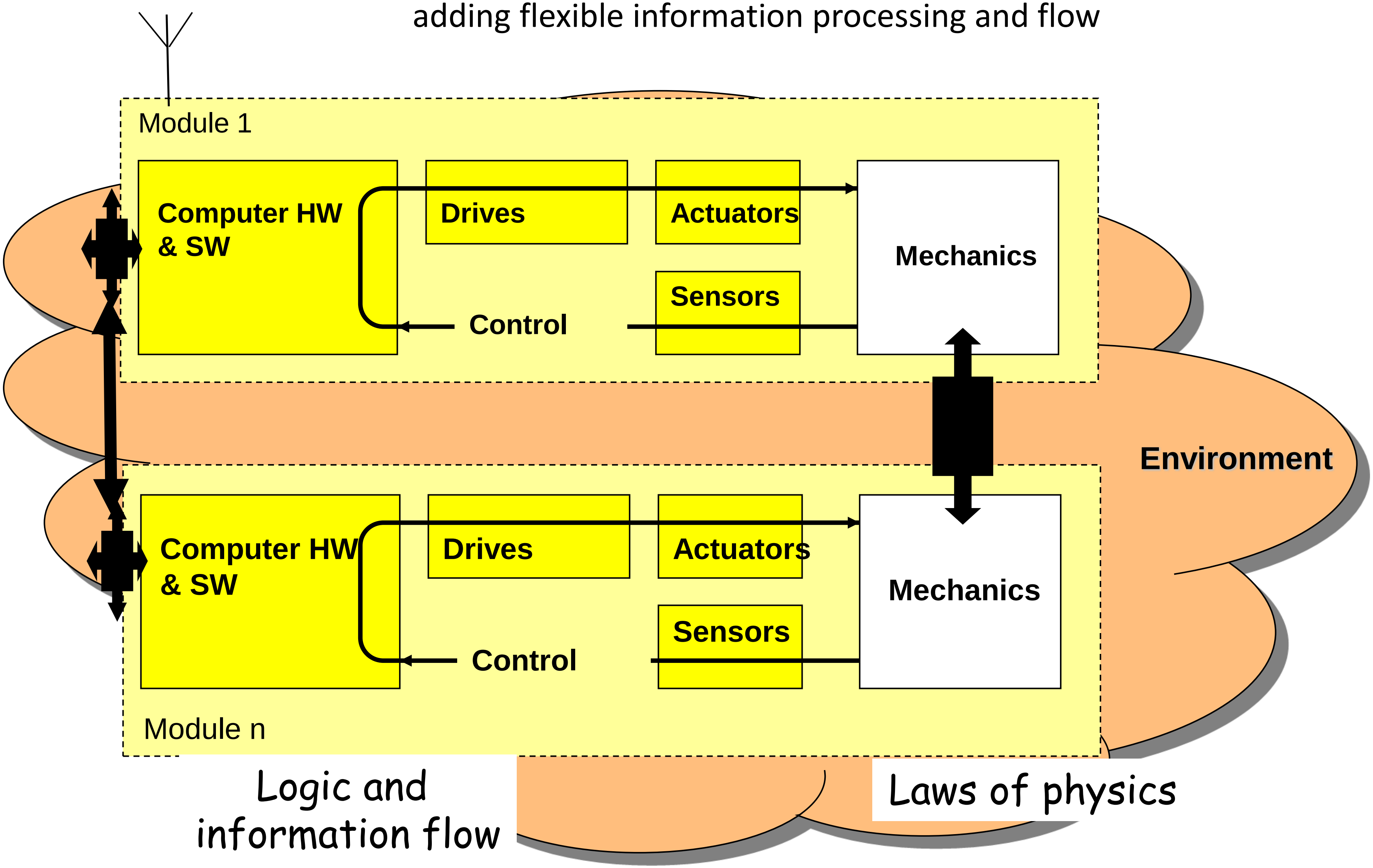
- CPS Characterization
- Engineering Environments
  – Problem analysis
  – Multiview modeling
- Integrating viewpoints
  – Contracts, Dependency modeling, Tool integration
  – Discussion
- Wrap-up

CPS characteristics

- Stringent requirements
- Heterogeneity
- Extent and scale (compared to ES)
  - Autonomy
- Non-technical challenges
  - Socio-technical systems

Mechanics vs. Mechatronics;

adding flexible information processing and flow



”Purely” mechanical vehicle

	Sus/C	Brake	Steer	Wheel	Diff	Trans	Clutch	Eng	Driver
Susp				X					X
Brake				X					X
Steer				X					X
Wheel	X	X	X		X				
Diff				X		X			
Trans					X		X		
Clutch						X		X	
Eng							X		
Driver		X	X				X		

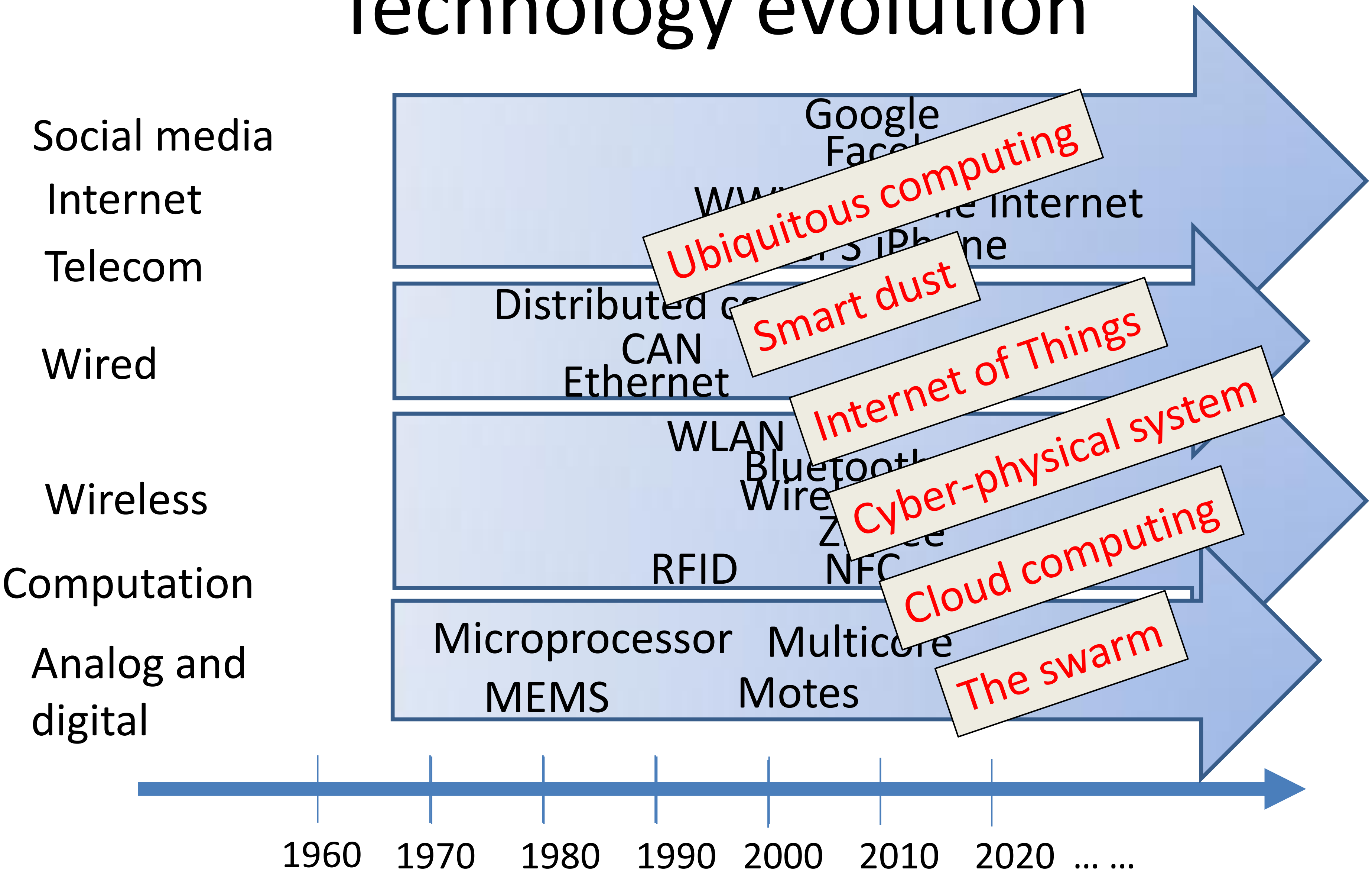
X – Mechanical relations

Fully programmable vehicle!

	Sus/C	Brake	Steer	Wheel	Diff	Trans	Clutch	Eng	Driver
Susp		P	P	X+P	P	P	P	P	X+P
Brake	P		P	*X*+P	P	P	P	P	*X*+P
Steer	P	P		*X*+P	P	P	P	P	*X*+P
Wheel	X	X	*X*+P		*X*				
Diff	P	P	P	*X*+P		*X*+P	P	P	
Trans	P	P	P	P	*X*+P		*X*+P	P	P
Clutch		P	P		P	*X*+P		*X*+P	P
Eng	P	P	P	P	P	P	*X*+P		P
Driver	P	*X*+P	*X*+P		P	P	*X*+P	P	

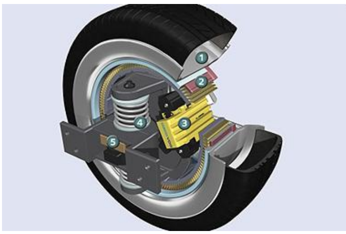
P – Programmable relations
X – Possible change

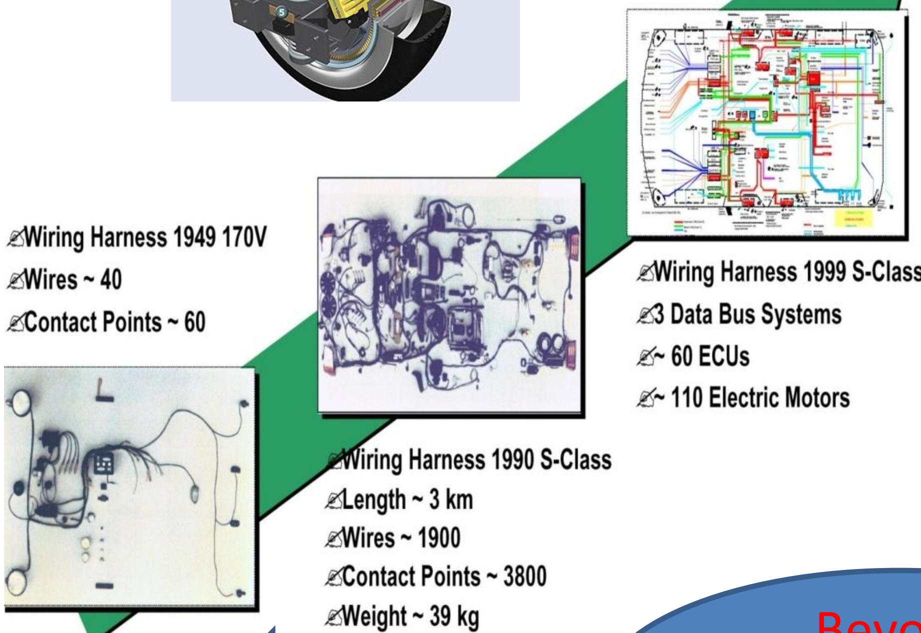
Technology evolution

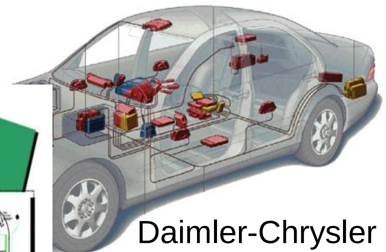


Autonomous vehicles

E-Corner, Siemens







Daimler-Chrysler

Platooning competition (Scania and KTH)

Local, global communication:
- V2V, V2I, Internet, positioning

New Opportunities and Challenges!

Components – Devices – Subsystems – Systems – SoS - Society

People – Processes - Technology

Organization
- Competence
- Responsibilities
- Communication

Process
- Analysis
- Modelling/tool use
- Synthesis/decisions

Social Integration

Model Based Integration

Technology: Products & Tools
- Function
- Implementation

Architecture Integration

Complexity

Here taken in relation to human abilities to deal with different aspects and their relations

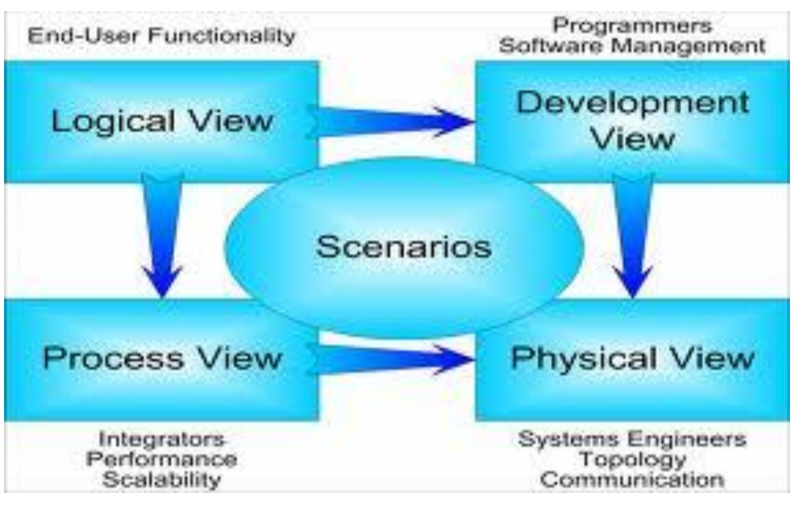
- Requirements, components and networks
- Behaviors and interactions
  - Digital state space
  - Hybrid systems
  - Product variants
- Life-cycle usages
- Potential faults, errors and failures

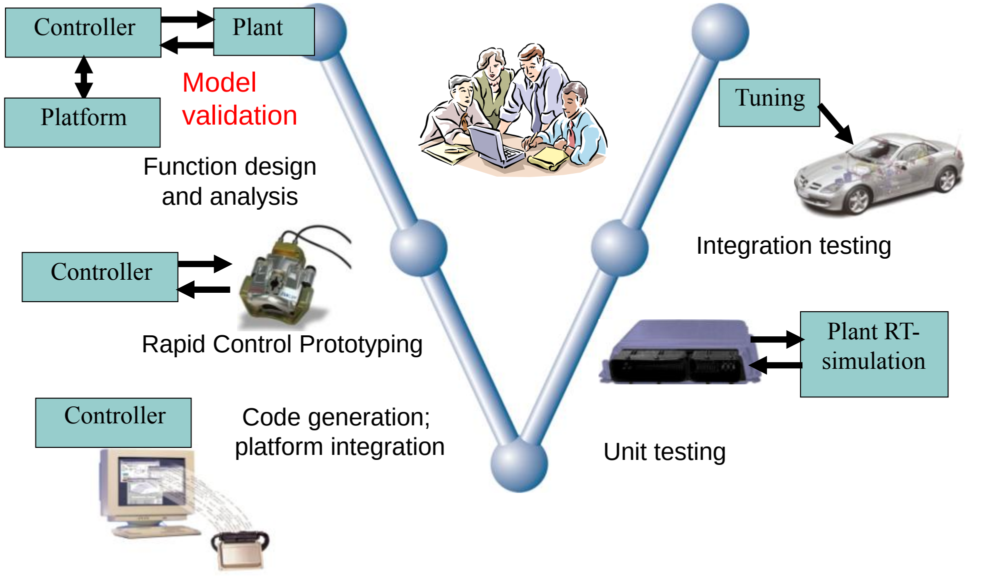
How do we manage complexity when developing technical systems?

- Divide and conquer
  - Applied to products and systems (supporting artefacts, organizations)
- Abstraction
  - Modeling, at the ”right” level of abstraction using the ”right” formalism
- Ensuring integration / composability
  - Interfaces, interrelationsships, assumptions
- By use of automation (”tools”)

Abstraction

- Modeling and simulation
- Multiple abstraction layers; Multiple views
- Platform abstractions: API's, services





Aspects of product & support tools

Concepts Dev. Prod.. Oper/Maint.

Technical engineering

Information management

Implemented by

+ Other functions!
+ Other requirements!

Scripts

Traces

Versions
Configurations

Application functionalities

Mapping

Architecture

models
code
scripts
configs

Test cases

Additional models

Analysis results

Change requests

Reports

Integration scenarios

Mapping

Architecture
(Tools & integration technology)

Often forgotten:
- Model management
- Support models and tools

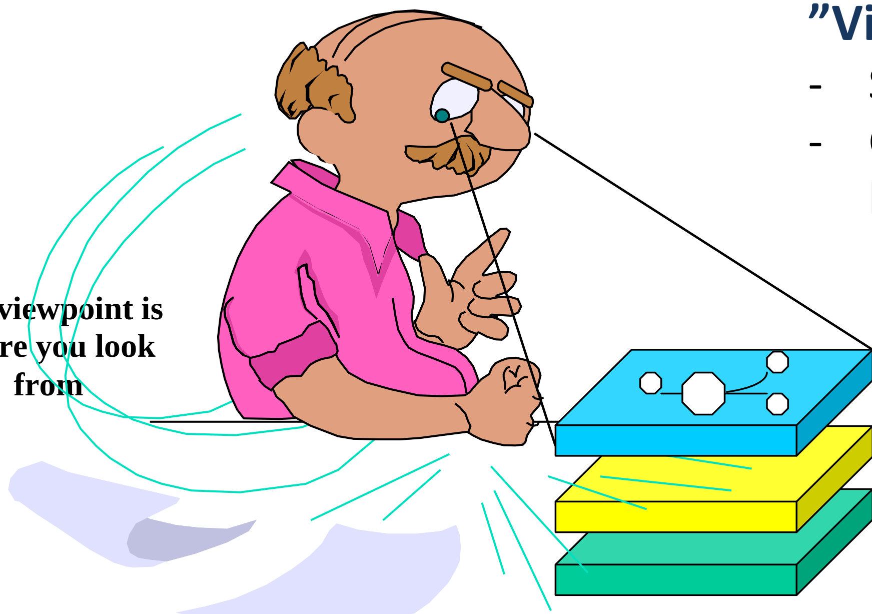
Outline

- CPS Characterization
- **Engineering Environments**
  – Problem analysis
  – Multiview modeling
- Integrating viewpoints
  – Contracts, Dependency modeling, Tool integration
  – Discussion
- Wrap-up

Viewpoints and views

"Viewpoints"

- Stakeholder concerns
- Concepts, theories, frameworks, languages



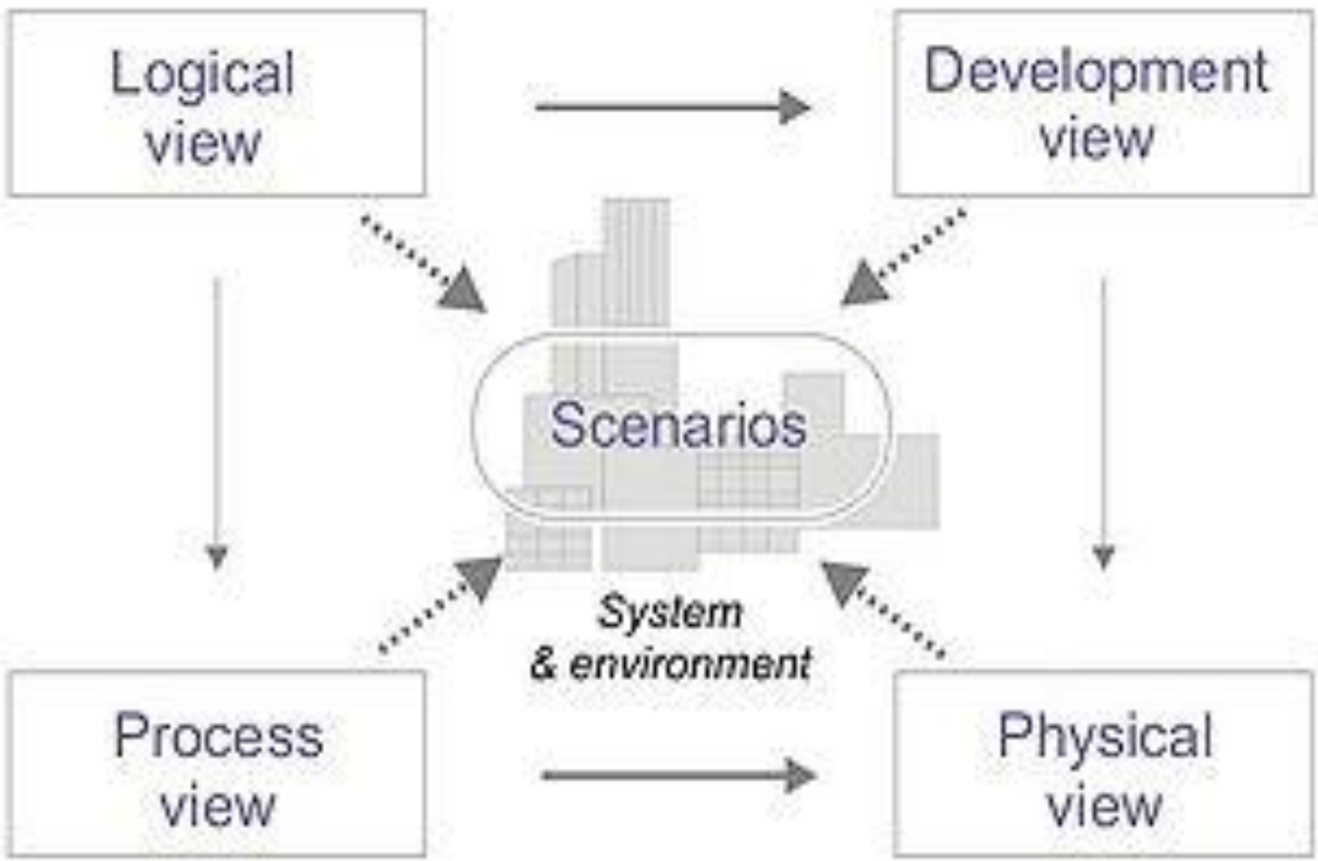
Models/Artefacts – "Views"

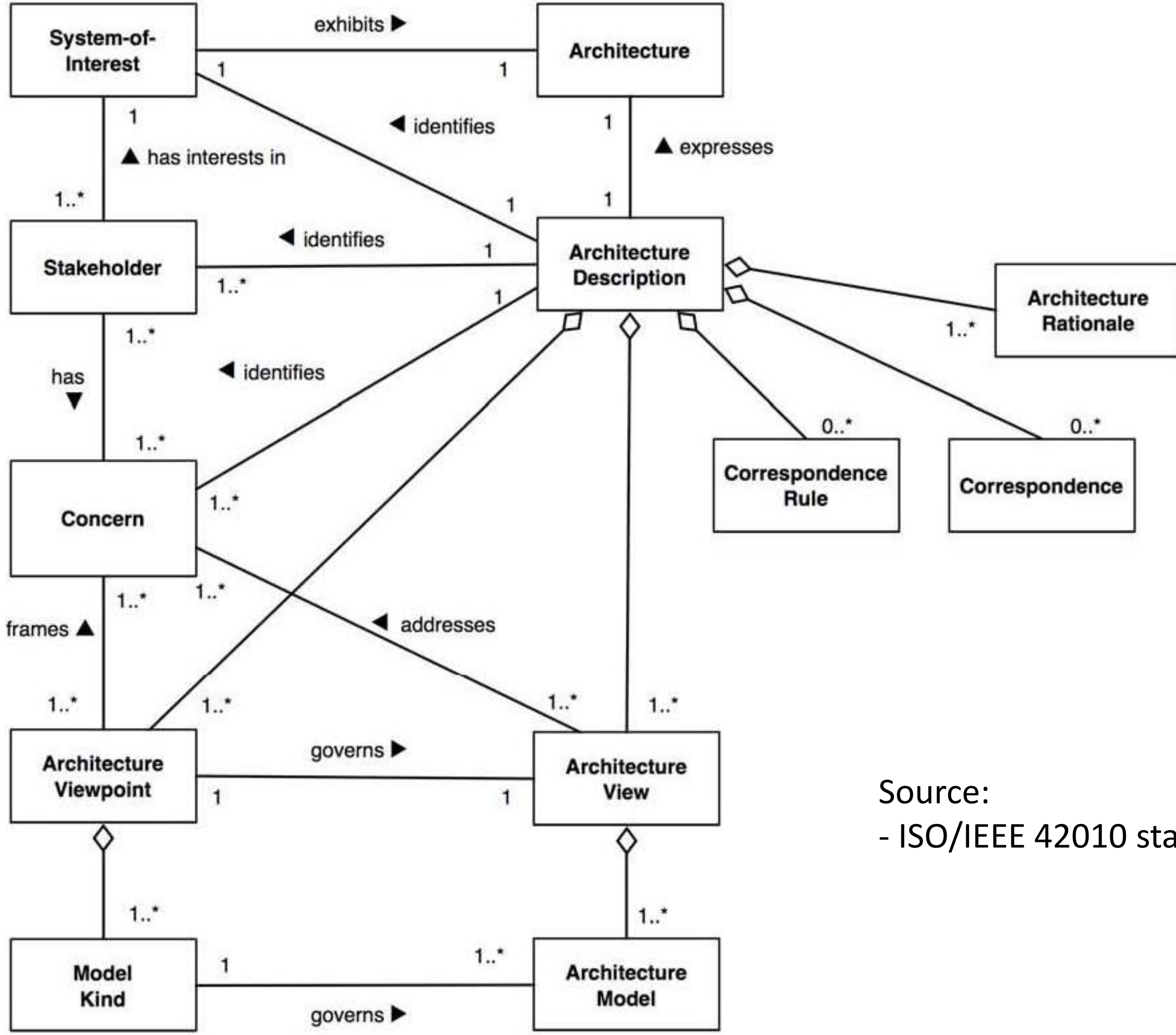
Product properties

Tools – using models and providing services

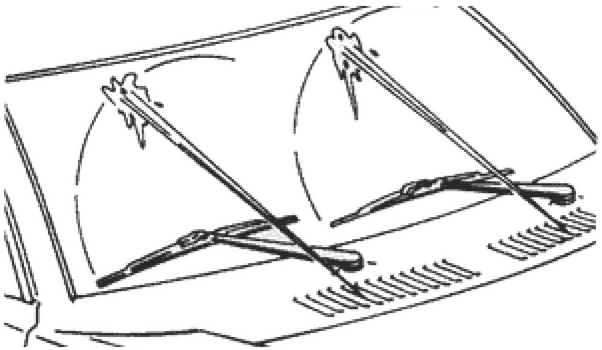
Terminology partly from the ISO/IEEE 42010 standard: Systems and software engineering — Architecture description

One example multi-view reference model: 4+1 view model





Source:
- ISO/IEEE 42010 standard

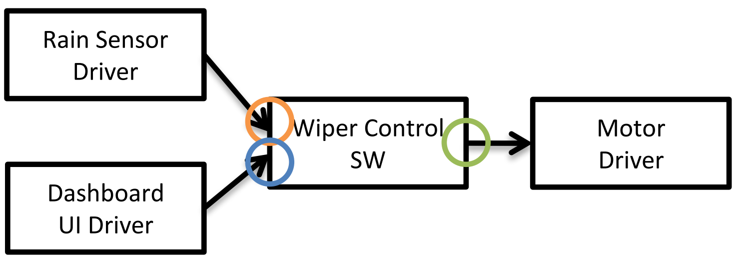


Multiple views - Example

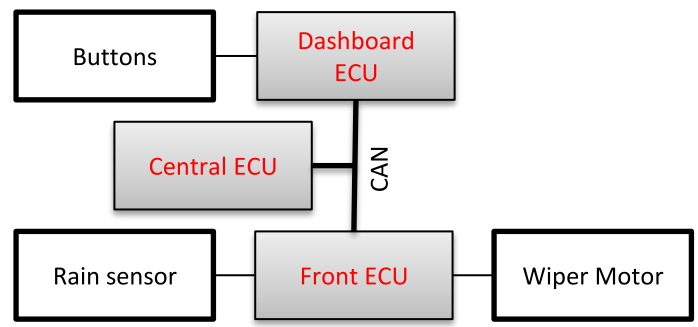
Allocation

Rain Sensor Driver - F ECU
Dashboard UI Driver - DB ECU
Wiper Control SW - C ECU
Motor Driver - DB ECU

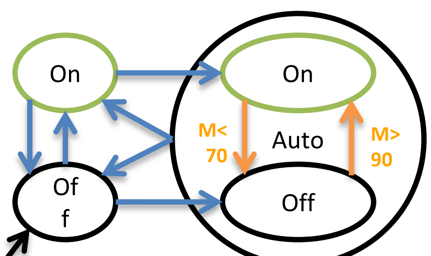
Software structure



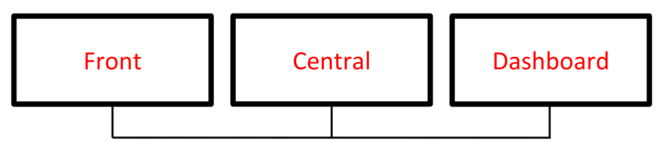
Hardware structure



Software mode description

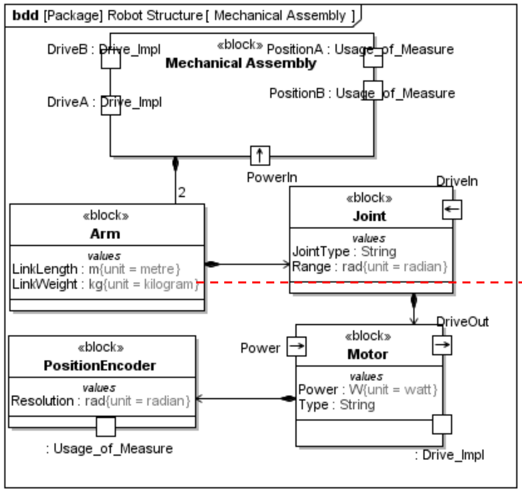


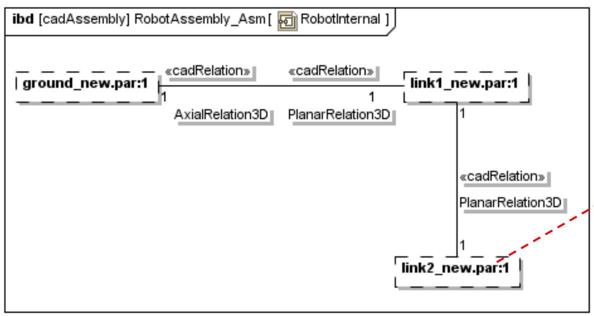
Network diagram

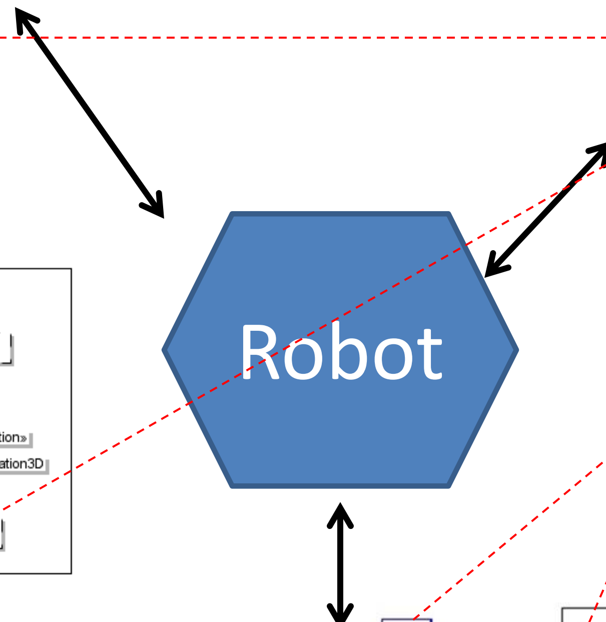


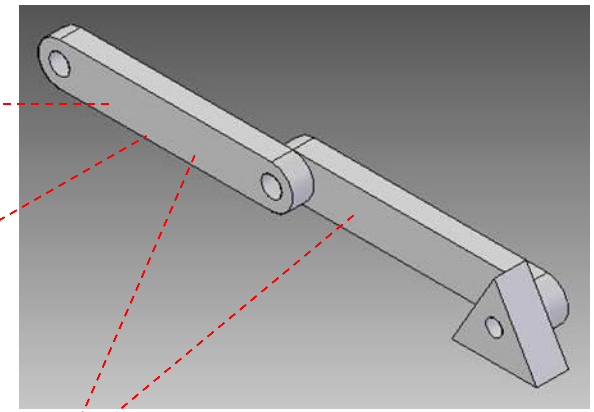
Example models by Magnus Persson

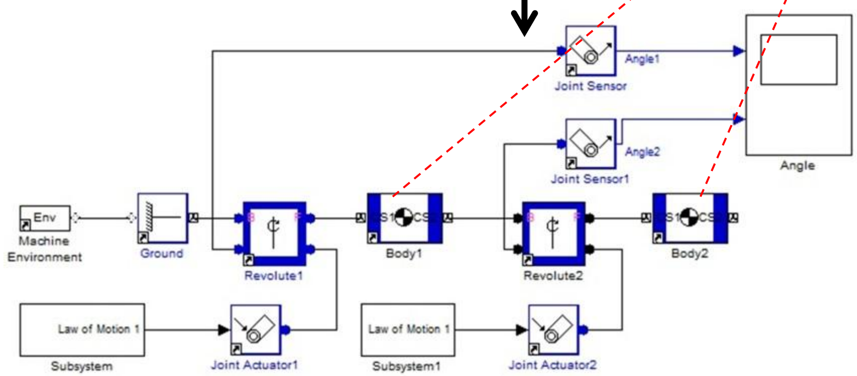
Multiple views - Example



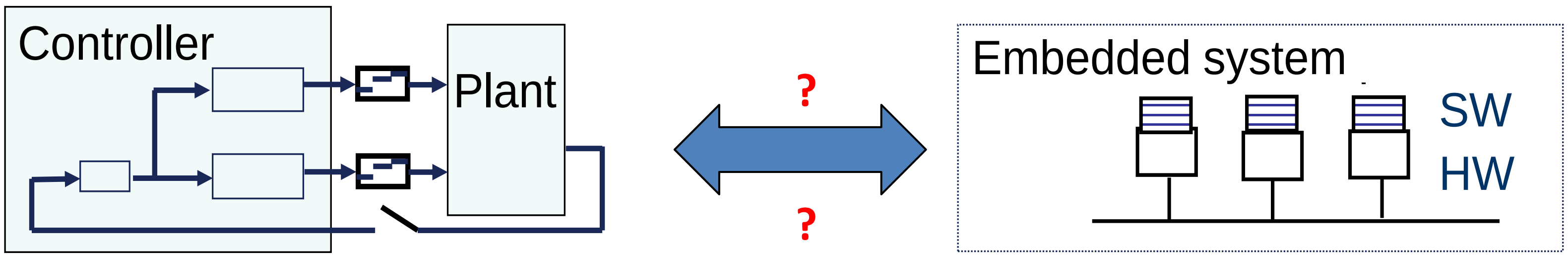








Controller and Embedded system – gaps

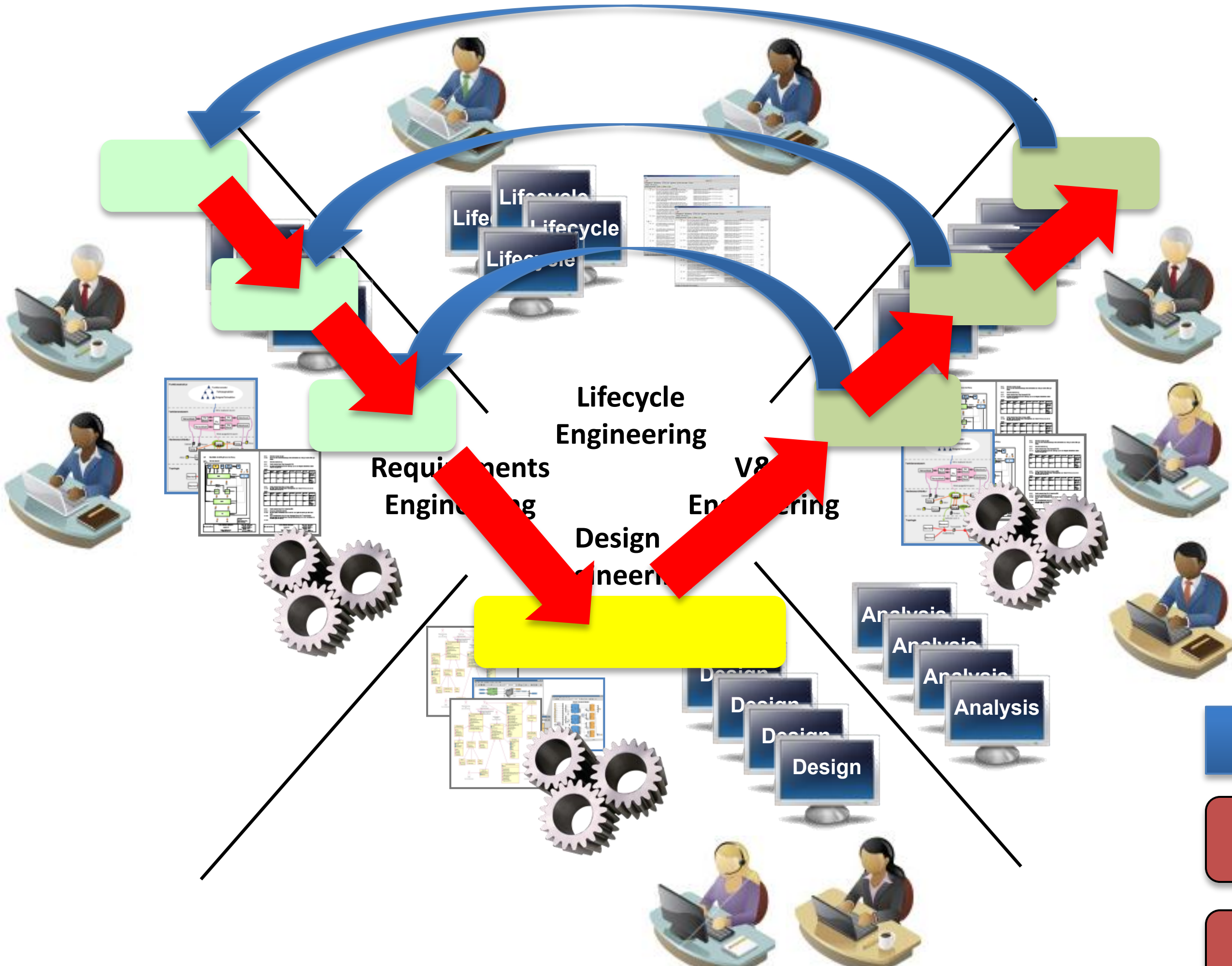


- Little emphasis on effects due to embedded systems realization
- Delays, quantization, partial failures, resource sharing

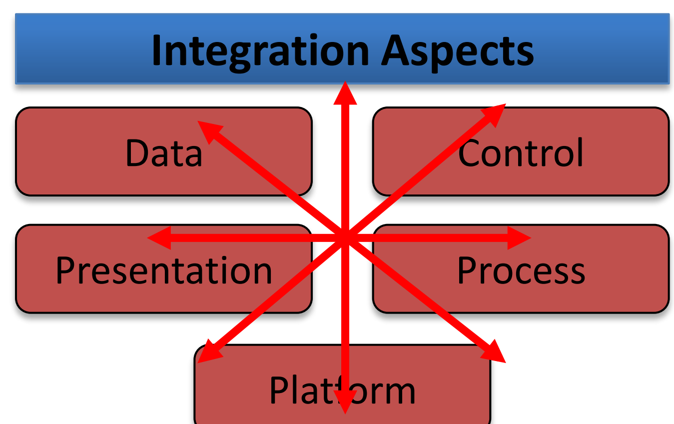
- Little emphasis on control specific requirements
- Assumed behavior, error handling

Concepts, theory and tools mismatches
- e.g. Discrete-time control theory vs.
Fixed priority scheduling /rate monotonic

Tool Integration Challenge



Multiple
- Concerns & Domains
- Stakeholders
- Isolated Tools & Models
- Processes and Tasks
- Integration Aspects
- Dependencies



Lack of methodologies for tool integration
Lack of mature standards

iFEST – Desired tool integration capabilities

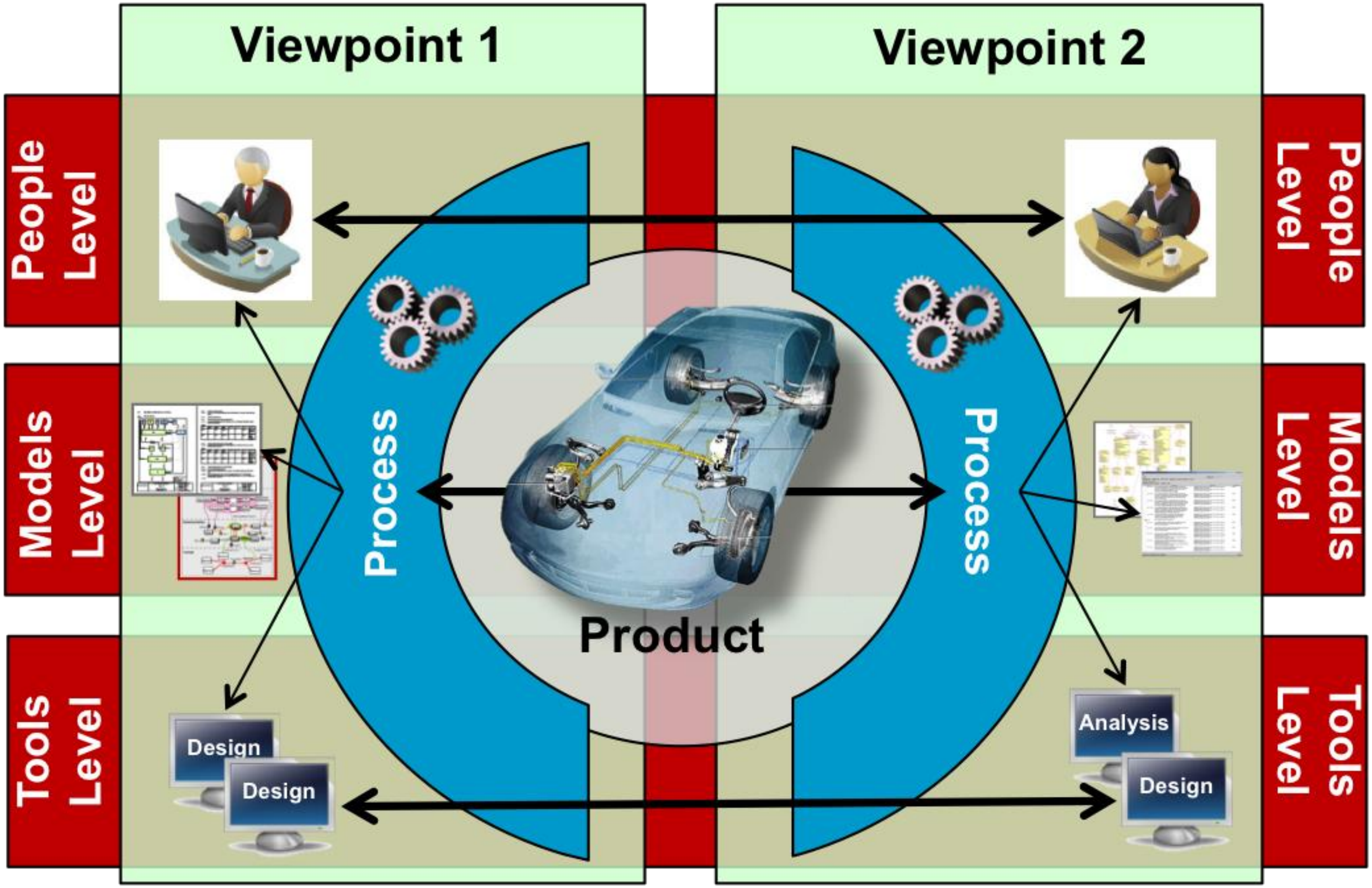
- A uniform approach to tool integration
  - Dealing with complexity
  - Tailorability, Evolvability and Scalability

- Lower threshhold to integrate tools
  - Ability to define information and functionalities
  - Access to information and functionalities of tools
  - Manage information
  - Data, control, presentation, process
  - Support relevant standards

Outline

- CPS Characterization
- Engineering Environments
  - Problem analysis
  - Multiview modeling
- **Integrating viewpoints**
  - Contracts, Dependency modeling, Tool integration
  - Discussion
- Wrap-up

Towards integrating viewpoints



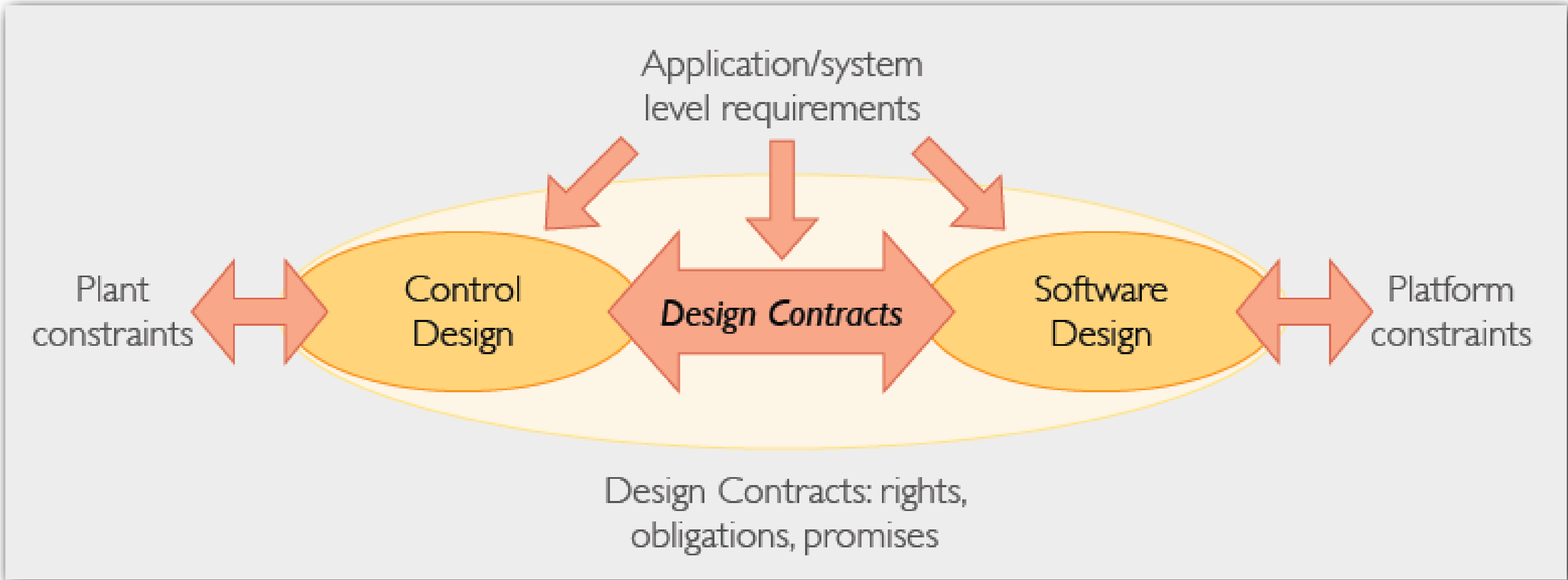
Techniques

- Viewpoint contracts
  – ”people level”
  – Establishing shared concepts, assumptions and constraints
- Dependency models
  – Explicitly captures dependencies among properties of models
- Tool integration models
  – Explicitly describes tools and their interactions

Viewpoint contracts

- Bridging conceptual and communication gaps
- Capturing
  - Common concepts shared among two or more viewpoints – Minimalistic
  - Assumptions
  - Constraints
- Currently dealt with informally

Example: Control-Embedded SW contracts



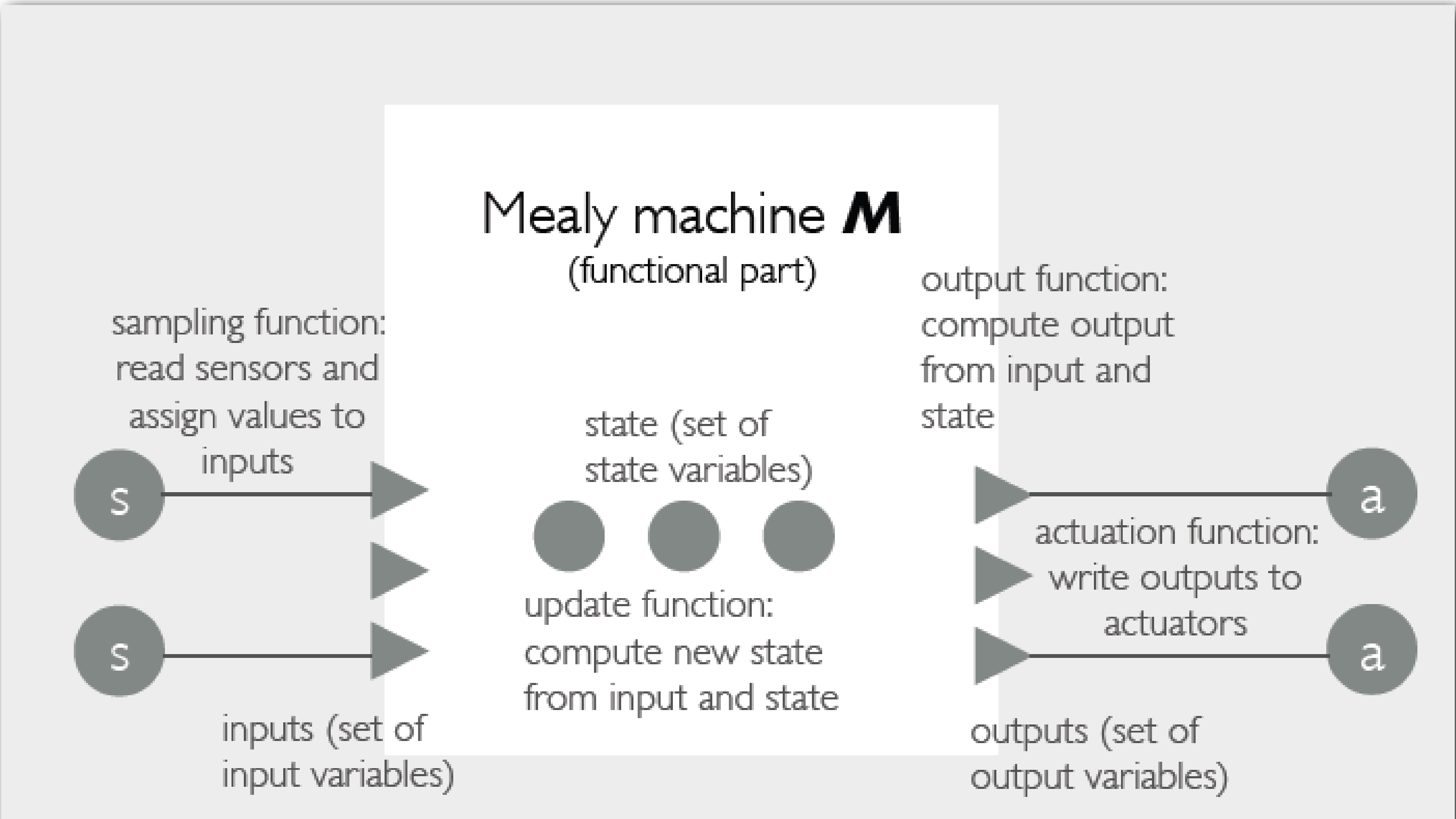
Courtesy of P. Derler, Design contracts presentation at CPS conference 2013

Control vs. Embedded software

Concepts/Domain	*Control*	*Embedded software*
Metrics, Constraints	Robustness, noise sensitivity, bandwidth, overshoot, settling time	Utilization, response time, memory footprint, WCET, slack, power consumption
Design parameters	Choice of strategy (PID, optimal, adaptive, etc.), noise/robustness trade-off	Task partitioning, scheduling, inter-process communication
Formalisms, Theory	Continuous time (ODEs), discrete time, sampled data control theory	C code, synchronous languages, scheduling theory, model-checking, task models, UML/SysML

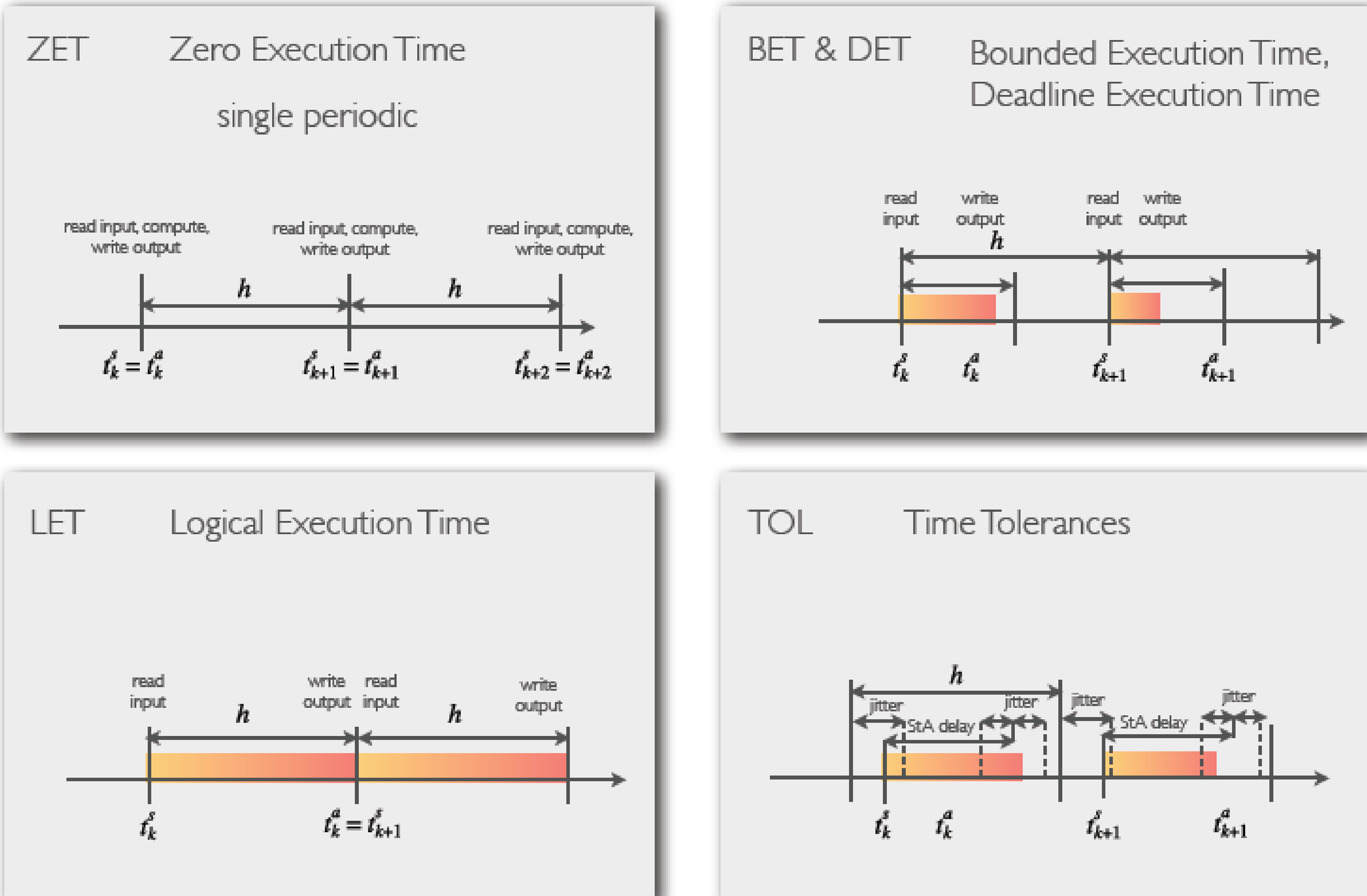
Courtesy of P. Derler, Design contracts presentation at CPS conference 2013

Example: Control-Embedded SW contracts



Courtesy of P. Derler, Design contracts presentation at CPS conference 2013

Example contracts - Control-Embedded SW with timing constraints



Courtesy of P. Derler, Design contracts presentation at CPS conference 2013

Example contracts continued;
Control-Embedded SW with timing constraints

Agreement and obligations regarding functionalities and timing properties

- SW engineers: execute functions; meet timing requirements
- Control engineers: ensure correct closed-loop behavior

Example contracts:

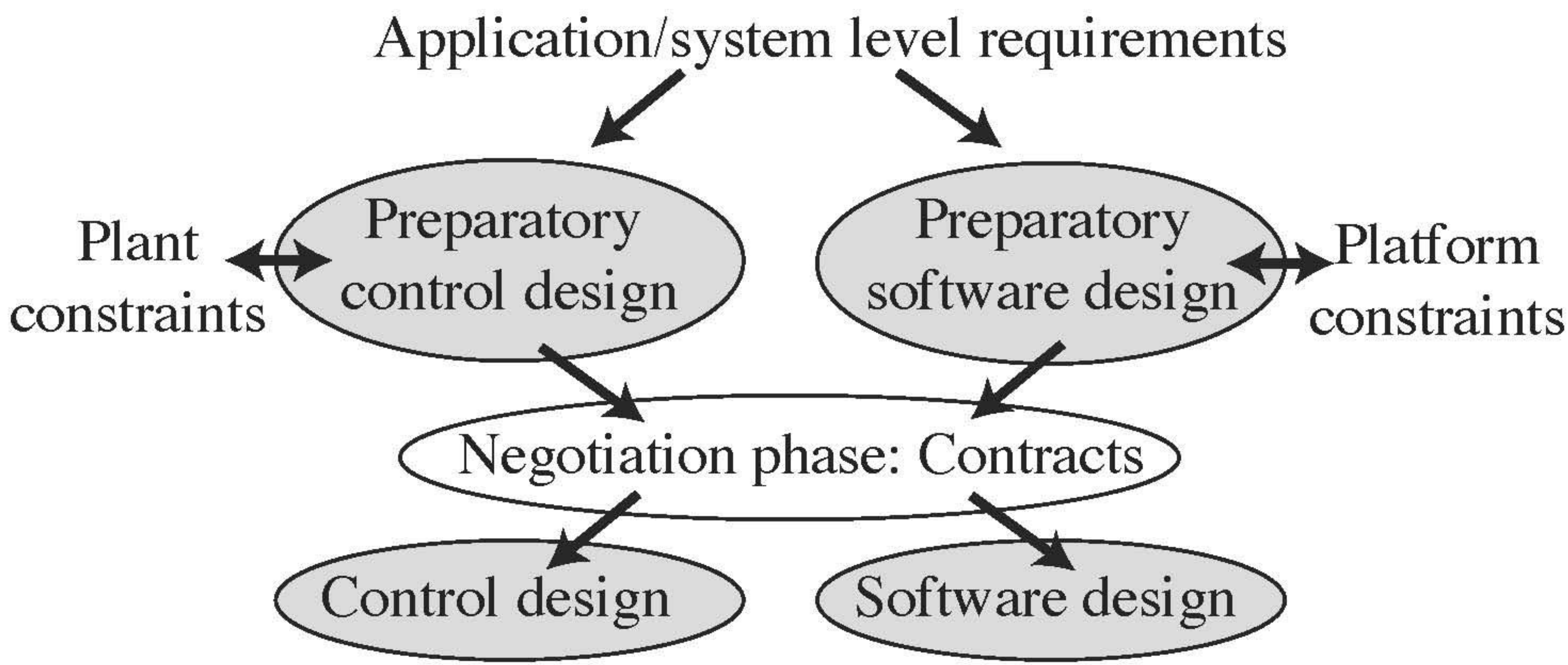
- ’ZET’ ~ the synchronous approach
- ’LET’ ~ the PLC / Giotto approach
- ’BET’ ~ interpretation of FPS
- ’DET’ ~ deadline monotonic scheduling
- ’TOL’ ~ Tolerances on time variations

Basis for discussions and agreements

Baseline for implementation

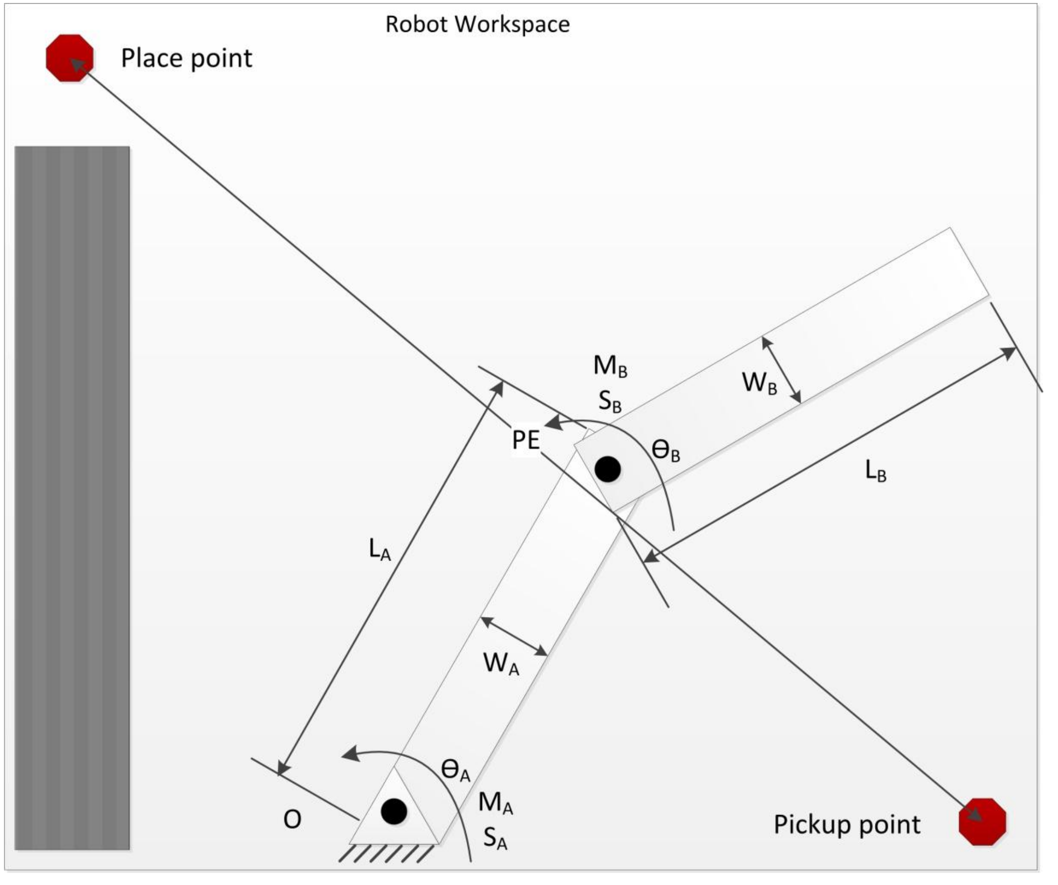
Support for modeling and simulation

Support for Contract based design-methodology outline

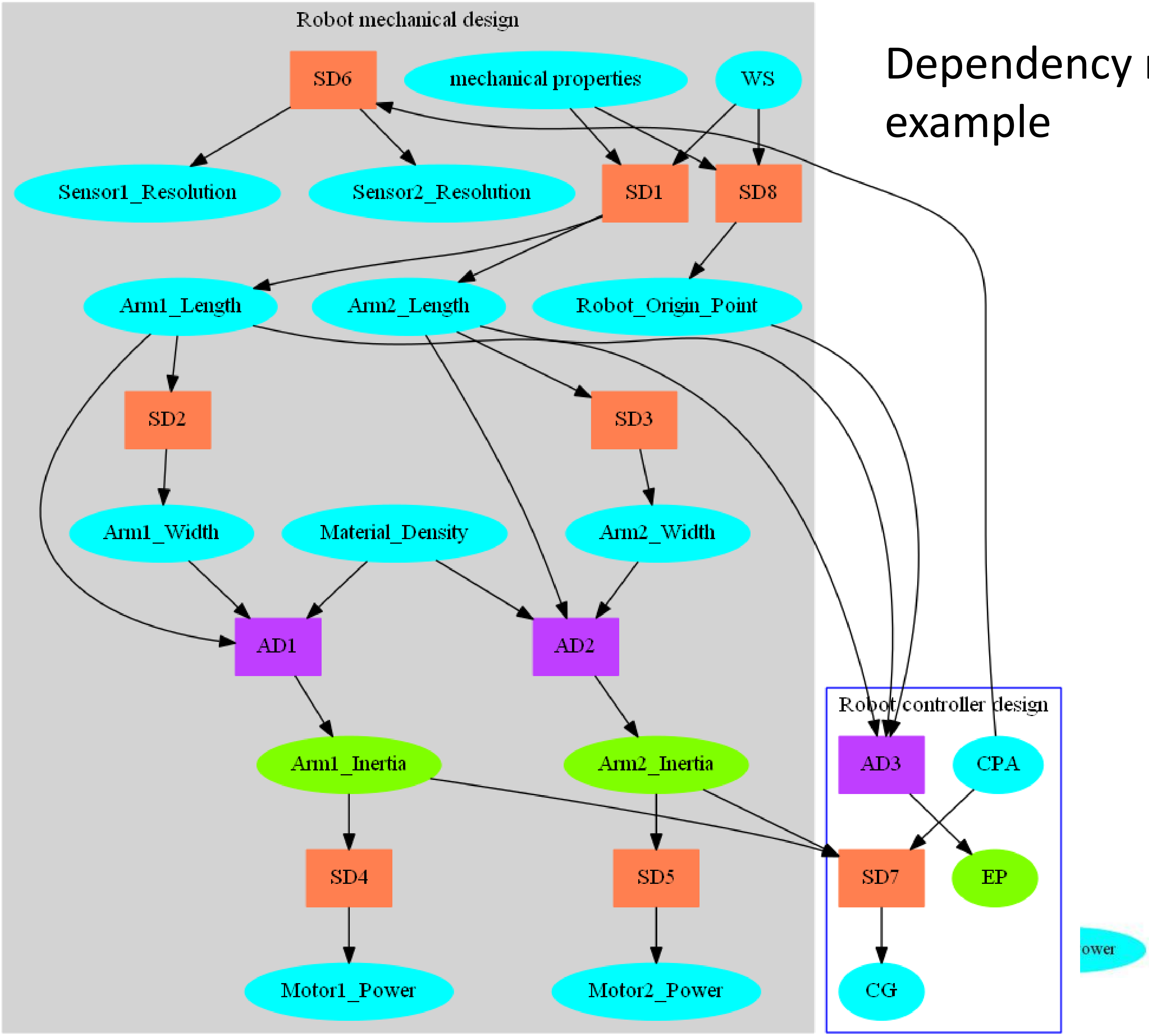


Two Degree-Of-Freedom Robot

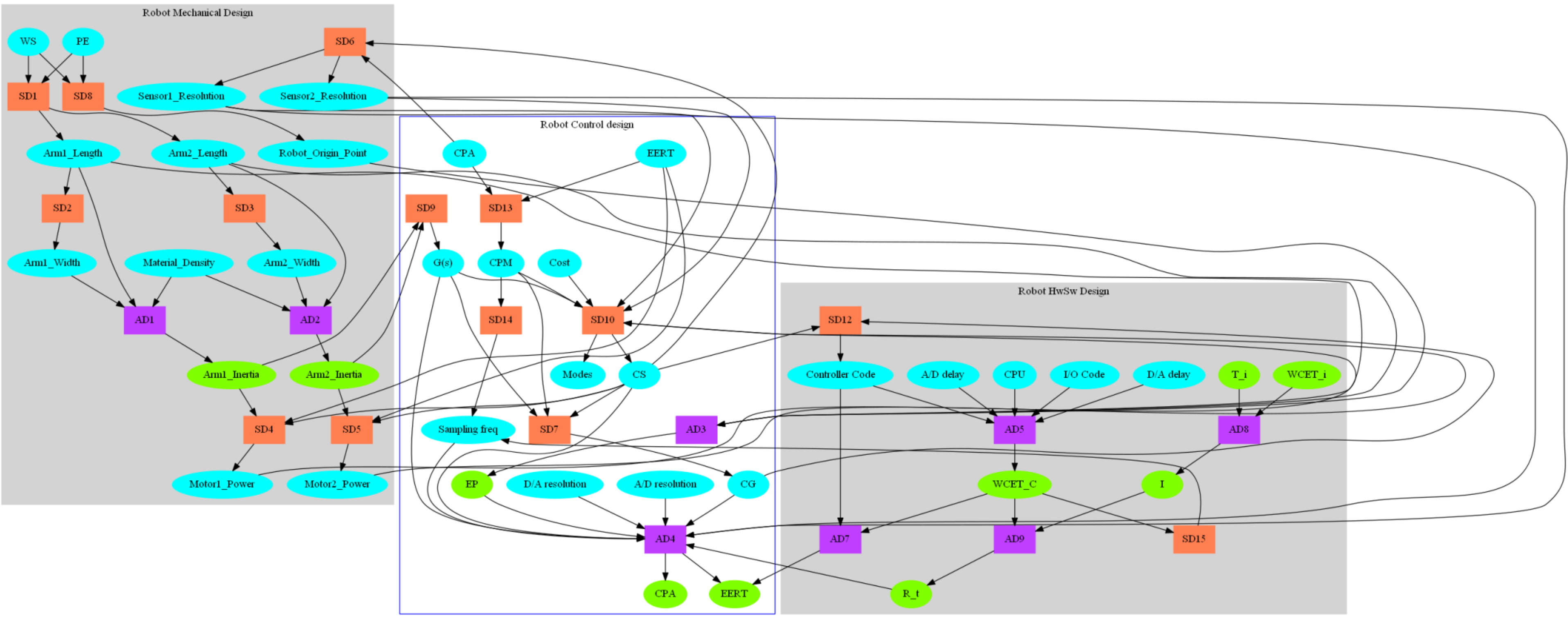
- Control the position of the robot with accuracy (CPA) and avoid obstacle within the specified workspace (WS)
- Two design domains
  - Mechanical design
  - Control design
  - Hw/Sw design
- Design variables
  - L_A, L_B
  - W_A, W_B
  - Θ_A, Θ_B
  - ϱ
  - S_A, S_B
  - M_A, M_B
  - Origin ‘O’
  - PE

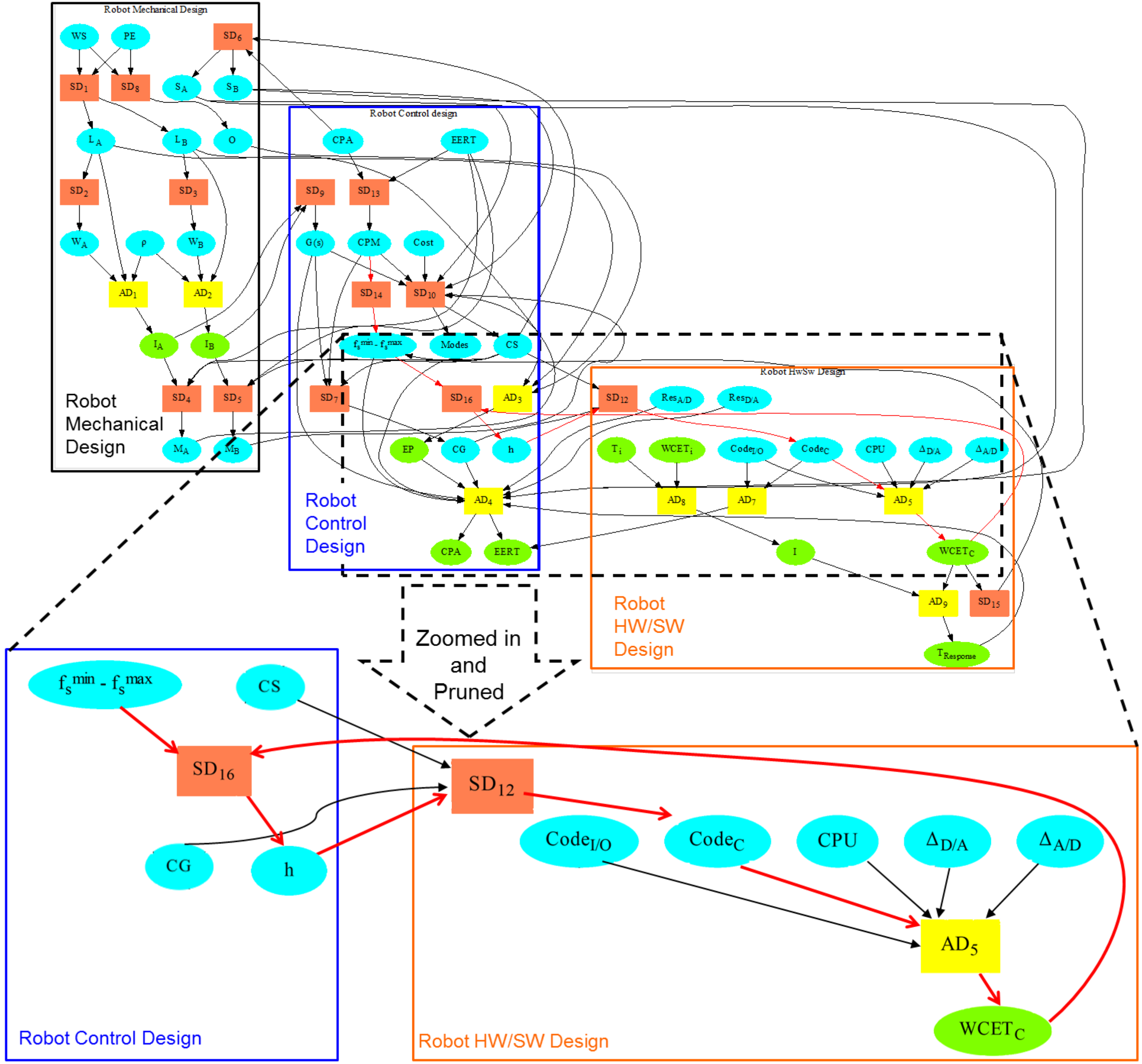


Dependency model example

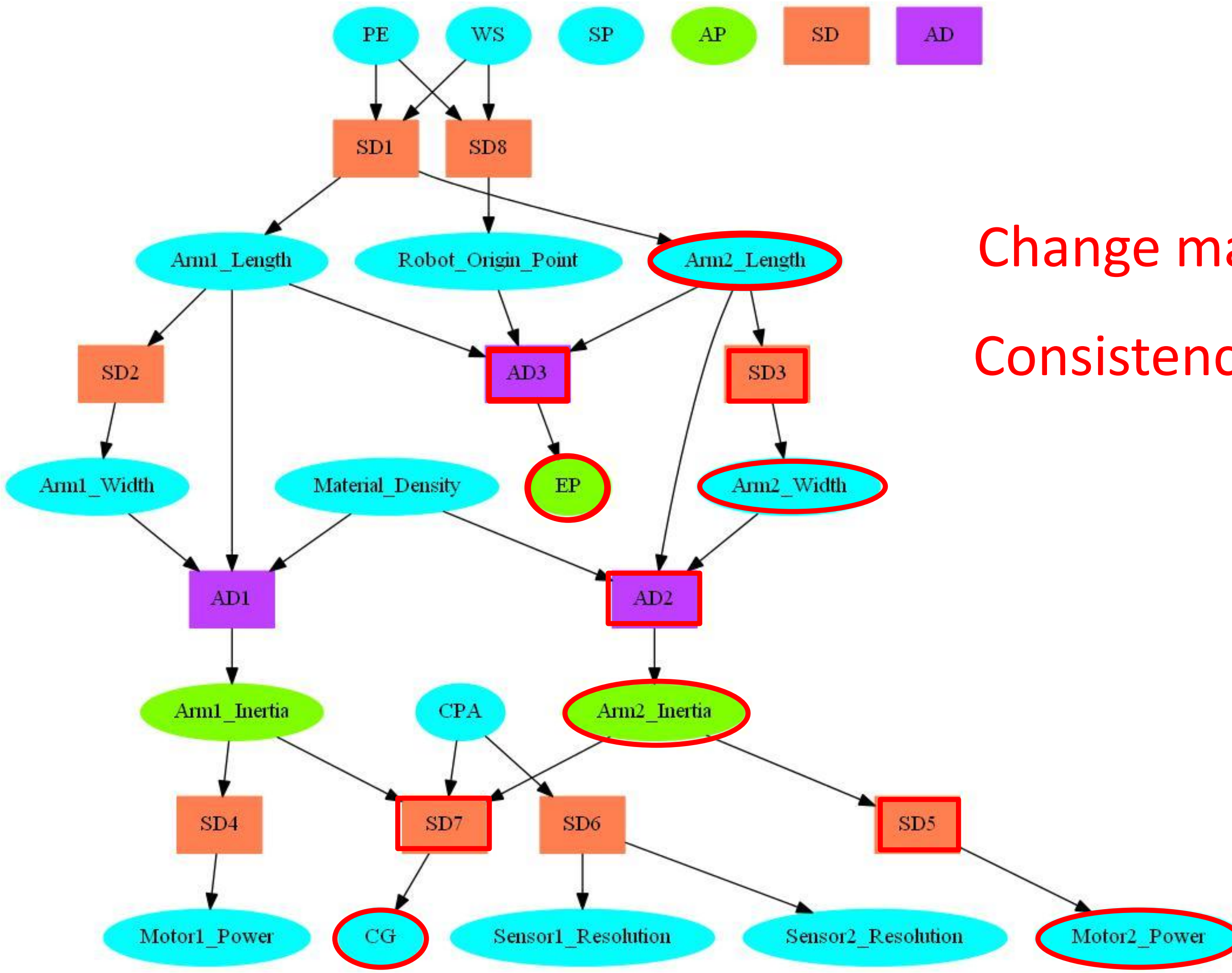


Snapshot of the dependency graph for Mechanical, Control and Hw/Sw design of the robot




Robot Mechanical Design
WS
PE
SD6
SD1
SD8
SA
SB
LA
LB
O
SD2
SD3
WA
ρ
WB
AD1
AD2
IA
IB
SD4
SD5
MA
MB
Robot
Mechanical
Design
Robot Control design
CPA
EERT
SD9
SD13
G(s)
CPM
Cost
SD14
SD10
fs min - fs max
Modes
CS
SD7
SD16
AD3
EP
CG
h
AD4
CPA
EERT
Robot
Control
Design
Robot HwSw Design
SD12
ResA/D
ResD/A
Ti
WCETi
CodeI/O
CodeC
CPU
ΔD/A
ΔA/D
AD8
AD7
AD5
I
WCETC
AD9
SD15
TResponse
Robot
HW/SW
Design
Zoomed in
and
Pruned
fs min - fs max
CS
SD16
SD12
CG
h
CodeI/O
CodeC
CPU
ΔD/A
ΔA/D
AD5
WCETC
Robot Control Design
Robot HW/SW Design

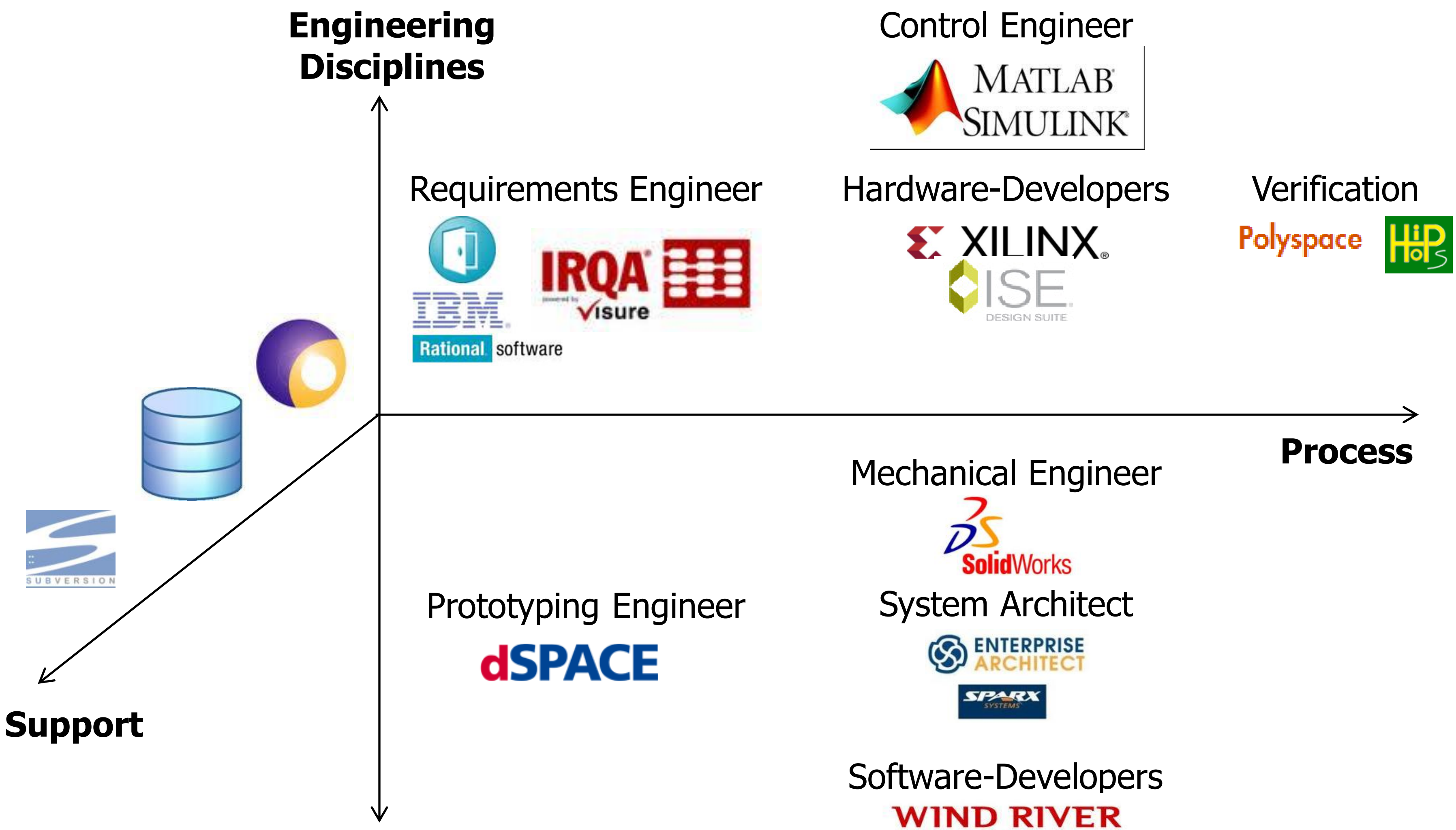
Analysis made possible



Change management

Consistency checking

Development of Mechatronic products



Challenges of Tool Integration

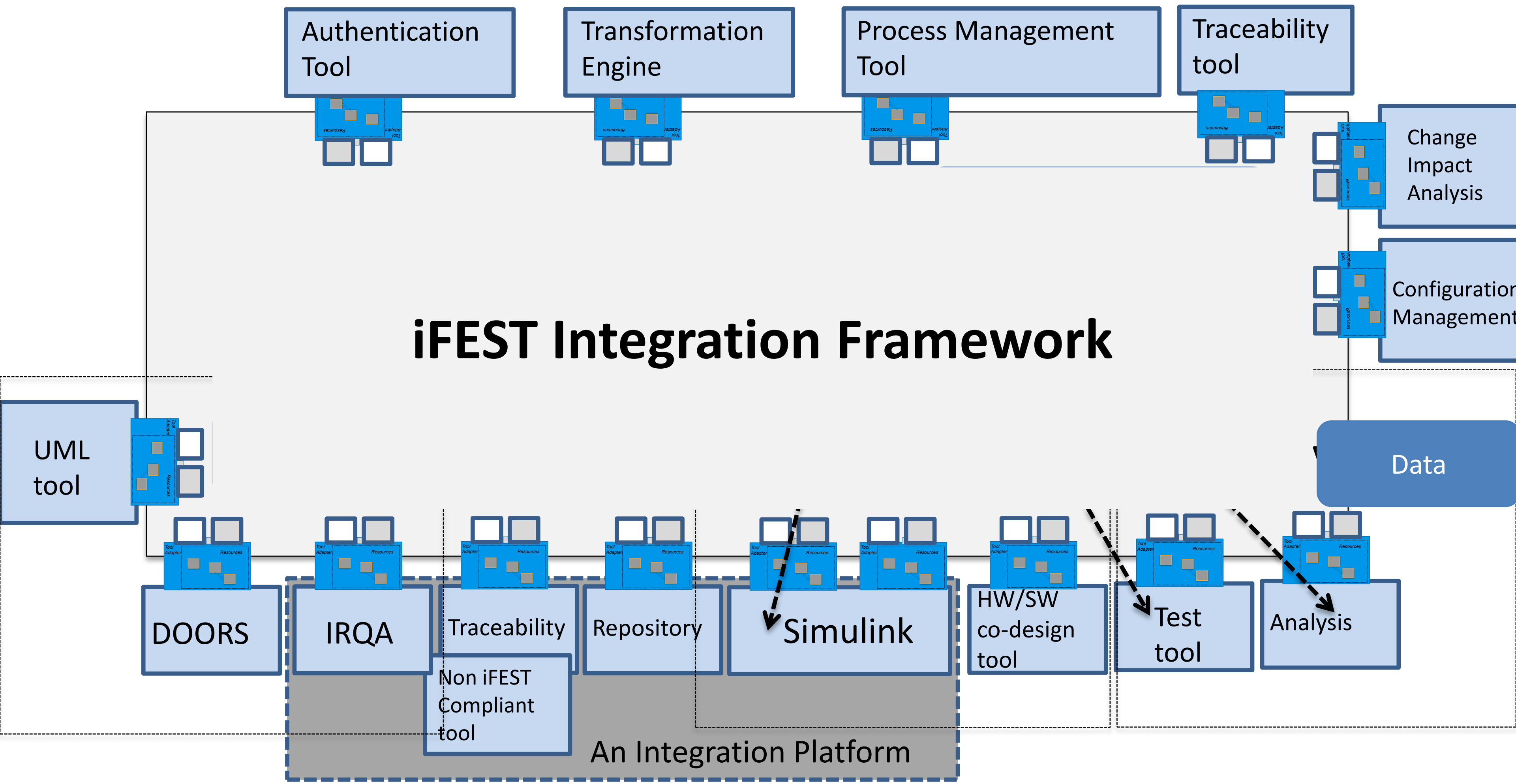
Tools provide limited native integration

- **island** solutions
- **difficult** to connect tools
  - to **transfer** data,
  - to create **traces**,
  - to create **scripts** involving multiple tools
- Hard to push workable tool integration standards

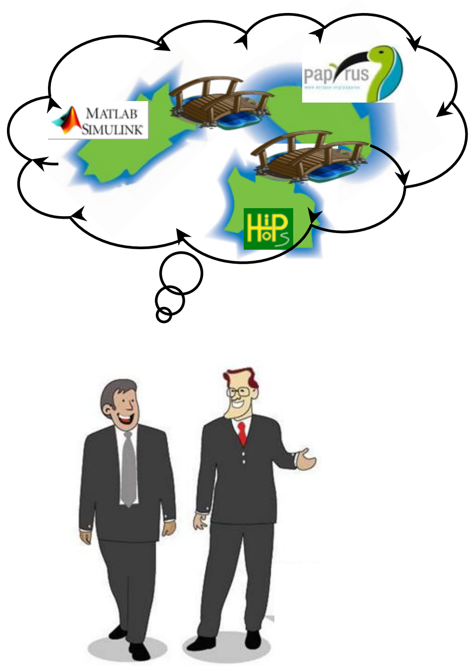
Integration Concerns

- Semantic
- Structural
- Technical

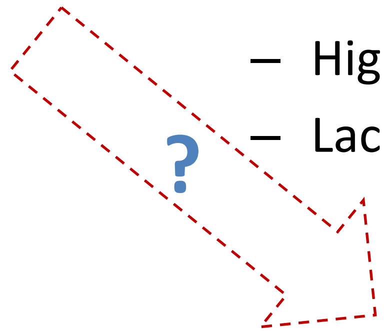
Goal: Tool Chains Instantiation



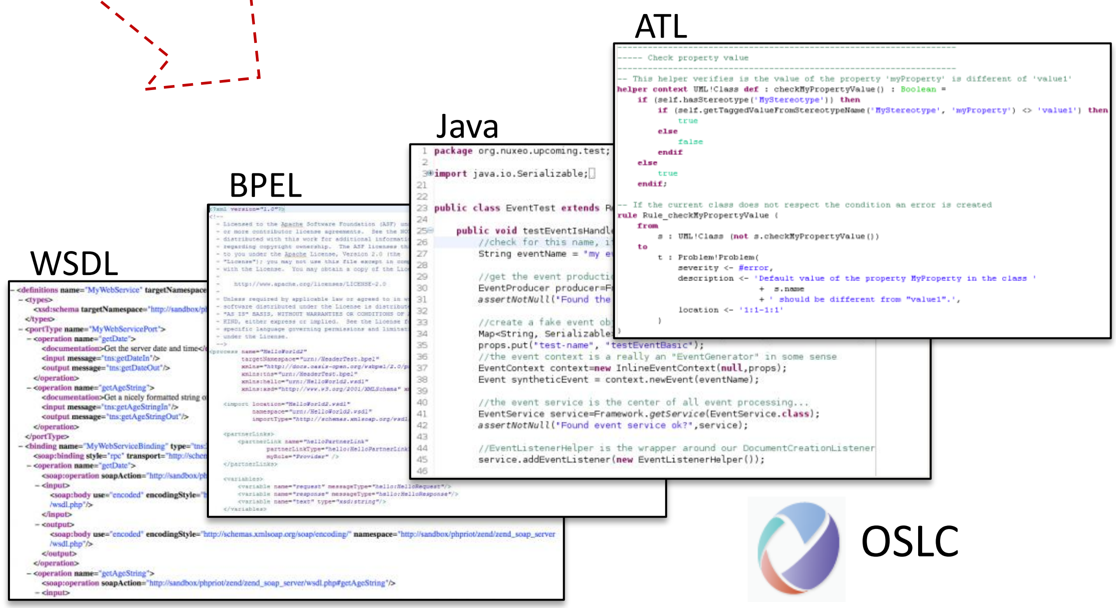
Challenge: Lack of Methodologies and Tools for Building Tool Chains



Stakeholders

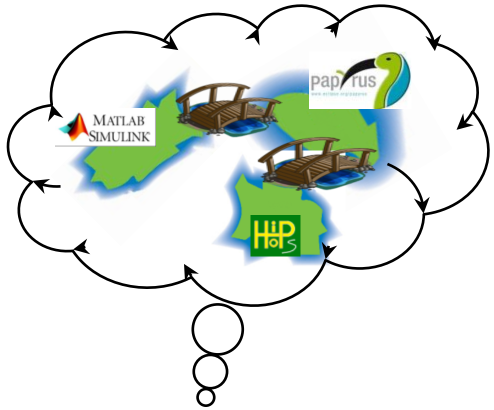


- Limited development support
- Mostly built manually
- Low levels of reuse
- High effort to build
- Lack of support tools



OSLC

Approach



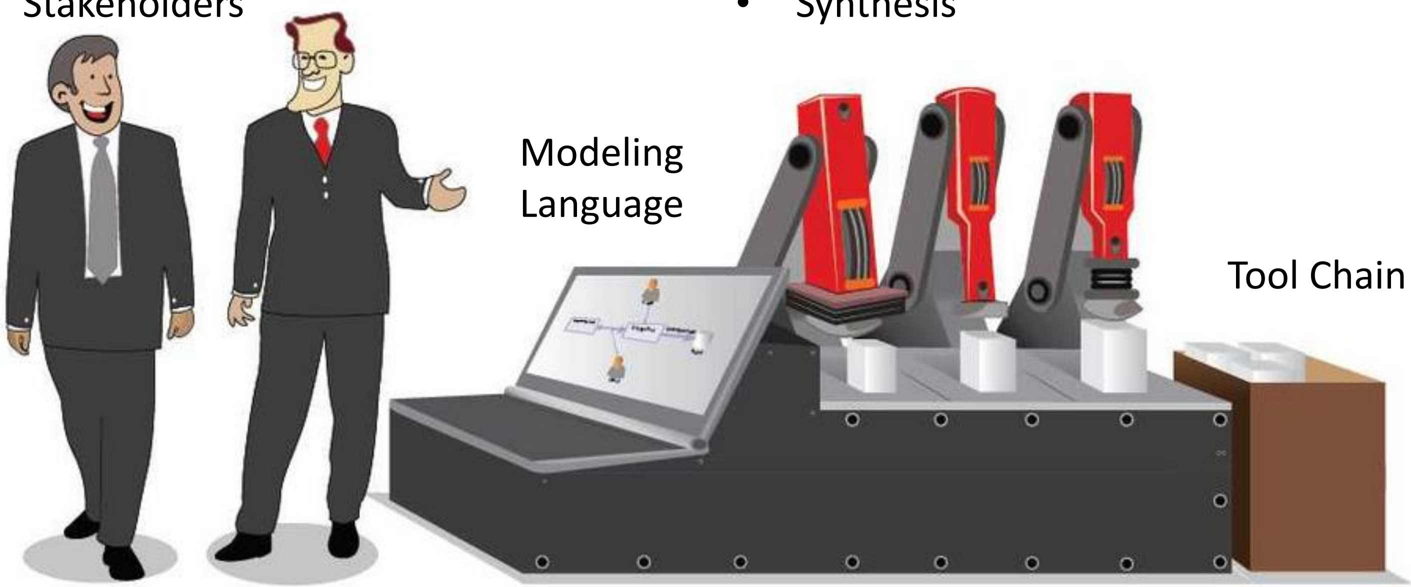
Development Support:

- Design/Refinement
- Analysis
- Synthesis

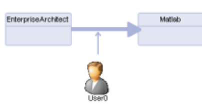
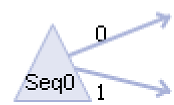
Stakeholders

Modeling Language

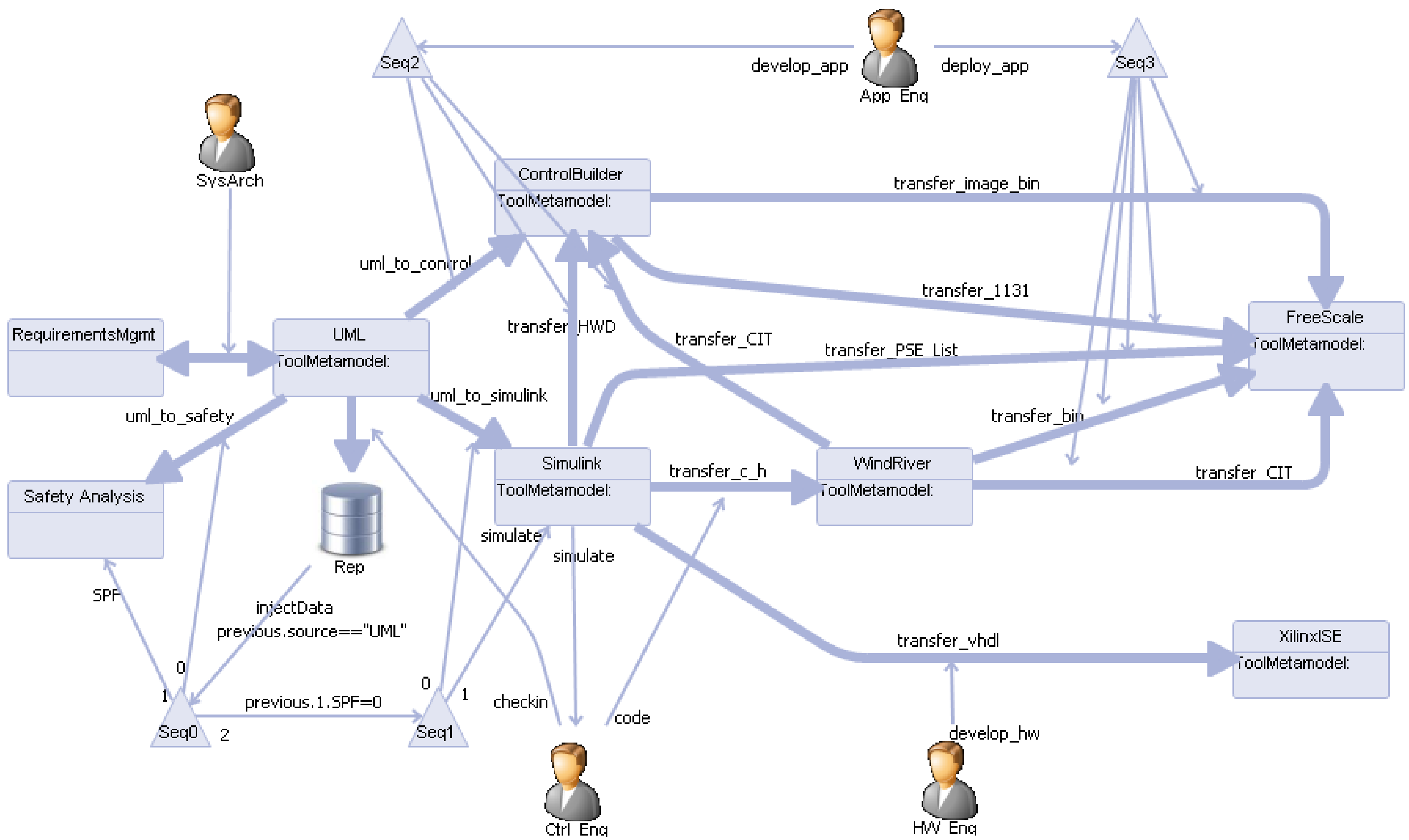
Tool Chain Impl.



Tool Integration Language (TIL) Concrete Syntax

	ToolAdapter		DataChannel
	User		TraceChannel
	Repository		ControlChannel
	ToolChain		Sequencer

Case Study: A TIL Model for an industrial embedded systems tool chain



49

Status

- Several papers, reports, case studies and PhD thesis on the presented viewpoint integration techniques
- Dependency modeling and Tool integration implemented as Domain specific modeling languages
- Viewpoint contracts so far applied to Control-Embedded systems and Control-Mechanical design

50

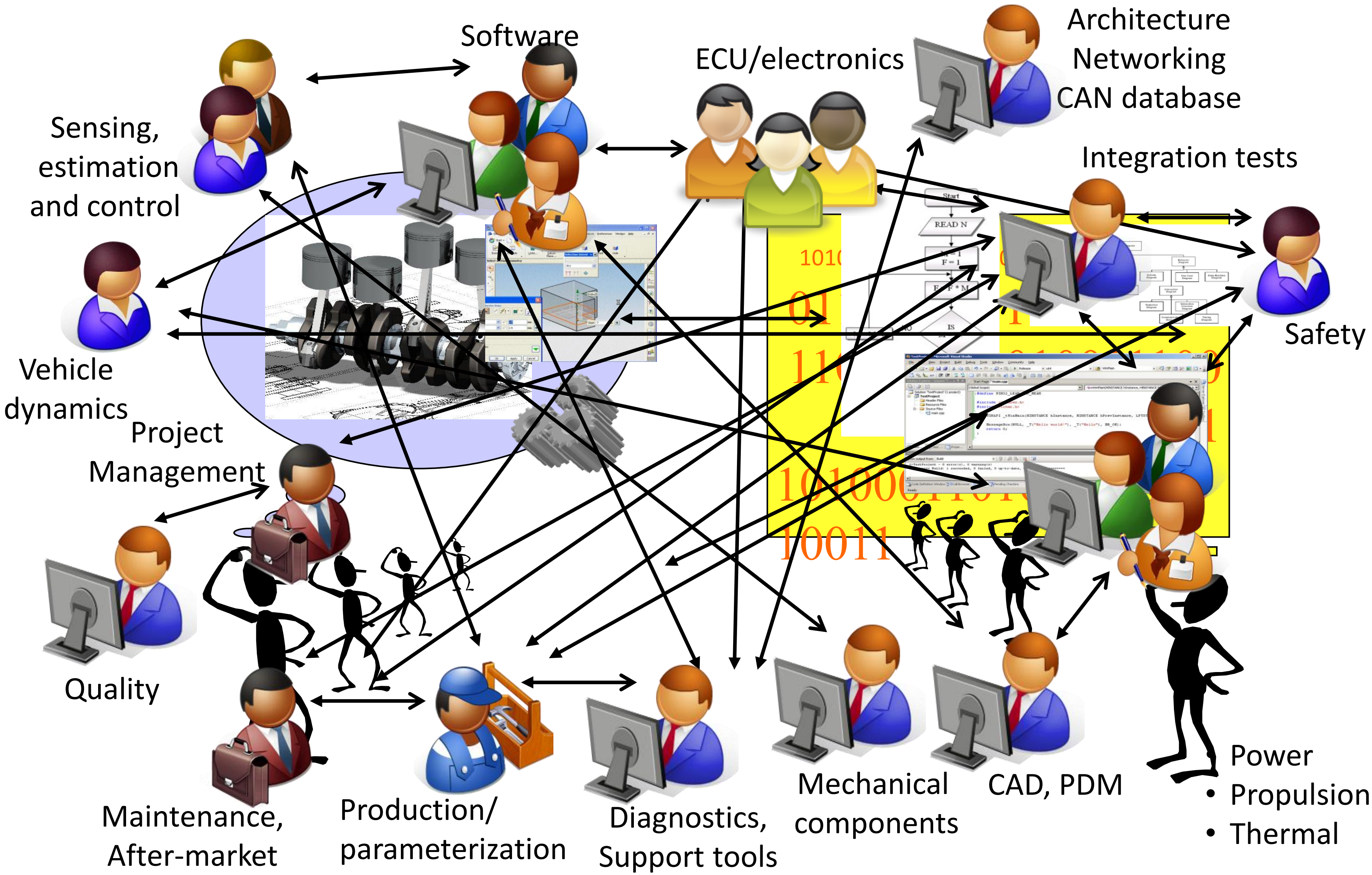
Discussion

- Viewpoint integration techniques can be applied in different ways
  - Informally / Formally
  - Individually / Together
  - Top-down / Bottom-up
- Complements other techniques such as
  - Co-simulation
  - Component contracts
  - Integration specific views (e.g. ADLs, function models)

Outline

- CPS Characterization
- Engineering Environments
  - Problem analysis
  - Multiview modeling
- Integrating viewpoints
  - Contracts, Dependency modeling, Tool integration
  - Discussion
- **Wrap-up**

Viewpoints and relationships



Summary and directions

- Trends: Functional growth, connectivity, scale, complexity
- Engineering of increasing no. of viewpoints and dependencies
  - Make viewpoint, dependencies and tool interactions explicit
- Dealing with and designing dependencies
  - Design contracts – towards viewpoint contracts
  - Dependency modeling
  - Systematic tool integration

Contact MartinT@kth.se

References

Fredrik Asplund, Lic. Thesis, Mechatronics, KTH (Oct. 2012): “Tool Integration and Safety: A Foundation for Analyzing the Impact of Tool Integration on Non-functional Properties”.

Matthias Biehl, PhD thesis, Mechatronics KTH (Jan. 2013): “A Modeling Language for the Description and Development of Tool Chains for Embedded Systems”.

Ahsan Qamar, PhD thesis, Mechatronics, KTH (March, 2013): “Model and Dependency Management in Mechatronic Design”.

Magnus Persson, PhD thesis, Mechatronics, KTH (June, 2013): “A formalized approach to multi-view components for embedded systems”.

Martin Törngren, Matthias Biehl, Ahsan Qamar, Jad Elkhoury, and Frederic Loiret. “Multiview Modeling and Integration for Mechatronics Engineering”. Proc. of the ACCM-Workshop on Mechatronic Design, Linz, Austria, November, 2012.

Patricia Derler, Edward Lee, Martin Törngren, Stavros Tripakis. Cyber-Physical System Design Contracts, ICCPS '13: ACM/IEEE 4th International Conference on Cyber-Physical Systems, 2013.