Invited Talk at SBCCI 2012

A Revolutionary Change in Embedded System Design

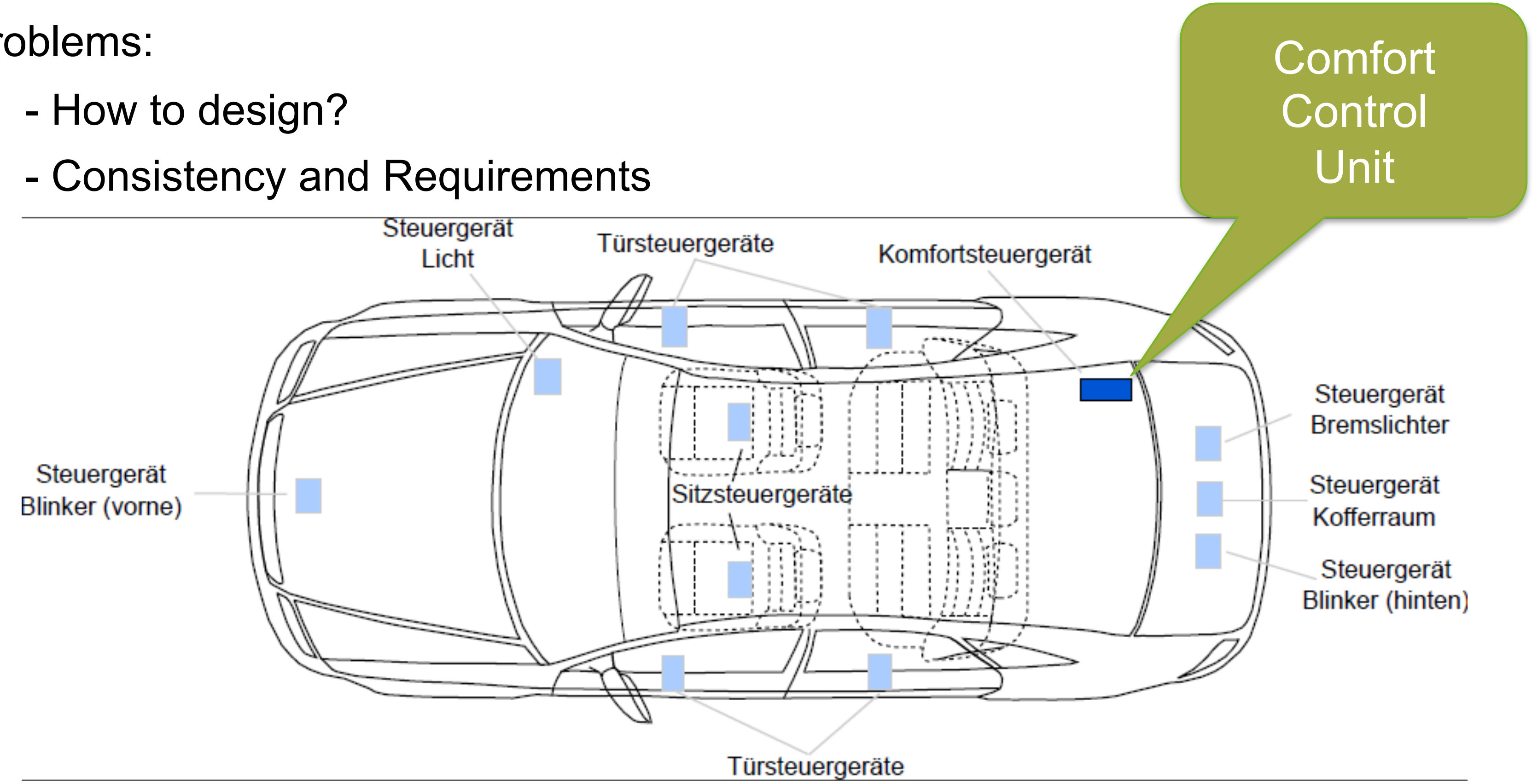
Prof. Dr. rer. nat. Achim Rettberg

Verkehr
Transportation

2 Example – Complex System

Comfort Control Unit

- System control lights, signals and doors of car
- Interfaces to other domain networks in car, like Powertrain
- Problems:
  - How to design?
  - Consistency and Requirements



3 Convergence of Two Trends

Model-Based Development ➡  ⬅ **Automated Analysis**

A Revolutionary Change in How We Design and Build Systems

4 Model-Based Development Examples

Company	*Product*	*Tools*	*Specified & Autocoded*	*Benefits Claimed*
Airbus	A340	SCADE With Code Generator	• 70% Fly-by-wire Controls • 70% Automatic Flight Controls • 50% Display Computer • 40% Warning & Maint Computer	• 20X Reduction in Errors • Reduced Time to Market
Eurocopter	EC-155/135 Autopilot	SCADE With Code Generator	• 90 % of Autopilot	• 50% Reduction in Cycle Time
GE & Lockheed Martin	FADEDC Engine Controls	ADI Beacon	• Not Stated	• Reduction in Errors • 50% Reduction in Cycle Time • Decreased Cost
Schneider Electric	Nuclear Power Plant Safety Control	SCADE With Code Generator	• 200,000 SLOC Auto Generated from 1,200 Design Views	• 8X Reduction in Errors while Complexity Increased 4x
US Spaceware	DCX Rocket	MATRIXx	• Not Stated	• 50-75% Reduction in Cost • Reduced Schedule & Risk
PSA	Electrical Management System	SCADE With Code Generator	• 50% SLOC Auto Generated	• 60% Reduction in Cycle Time • 5X Reduction in Errors
CSEE Transport	Subway Signaling System	SCADE With Code Generator	• 80,000 C SLOC Auto Generated	• Improved Productivity from 20 to 300 SLOC/day
Honeywell	Primus Epic	MATLAB	• 60% Automatic Flight Controls	• 5X Increase in Productivity

5 Does Model-Based Development Scale?

Airbus A380

Length	239 ft 6 in
Wingspan	261 ft 10 in
Maximum Takeoff Weight	1,235,000 lbs
Passengers	Up to 840
Range	9,383 miles

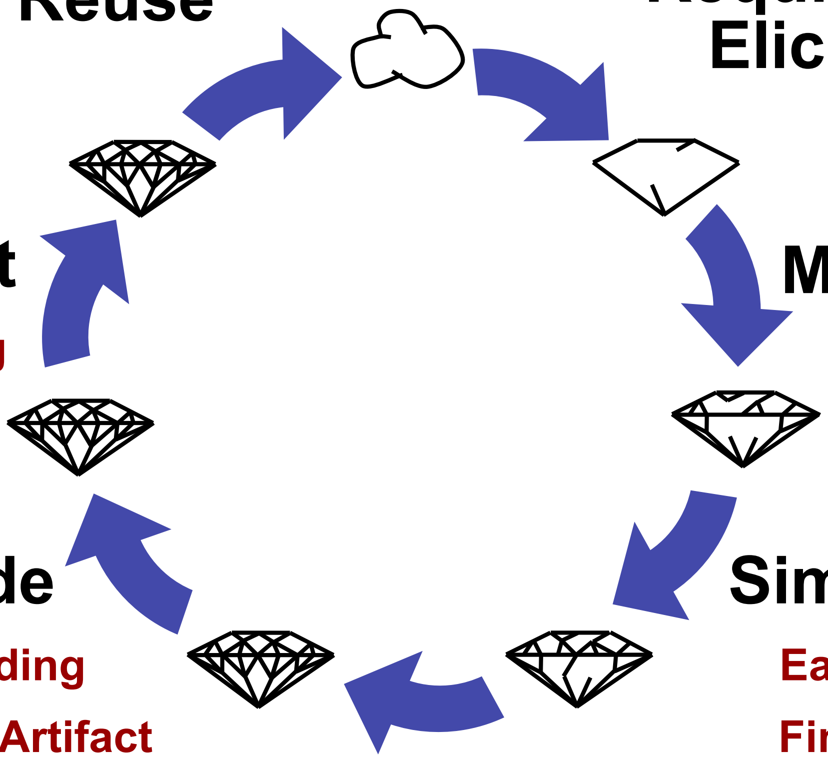
Systems Developed Using MBD

- Flight Control
- Auto Pilot
- Fight Warning
- Cockpit Display
- Fuel Management
- Landing Gear
- Braking
- Steering
- Anti-Icing
- Electrical Load Management

6 How we can Reduce Costs *and* Improve Quality?

Reuse

Requirements Elicitation



Modeling 10%

Clear Specifications
Improves Communication

Simulation 10%

Easy Validation
Finds Errors Early

Automated Analysis 10% - 20%

Cheaper Than Manual Analysis
Finds the Really Hard Errors

5% Autocode

Eliminates Manual Coding
Makes Model Primary Artifact

15% Autotest

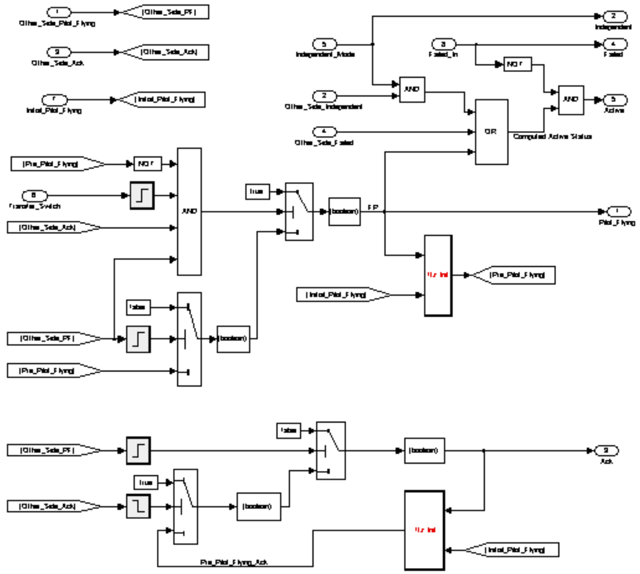
Reduces Cost of Testing
Enables More Testing

7 Example – ADGS-2100 Adaptive Display & Guidance System

883 Subsystems

9,772 Simulink Blocks

2.9 x 10^{52} Reachable States



©Rockwell Collins

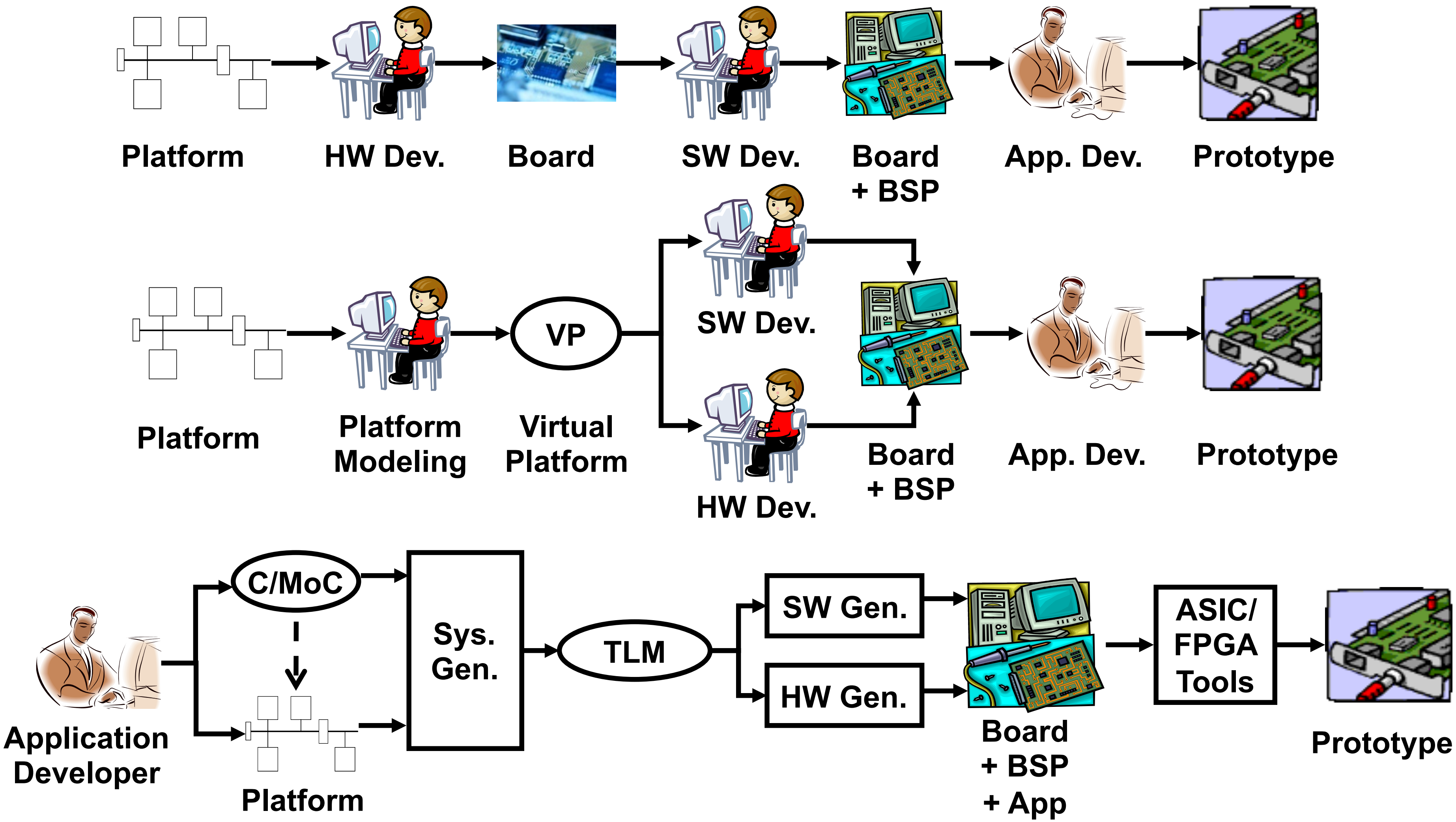
Requirement

Drive the Maximum Number of Display Units Given the Available Graphics Processors

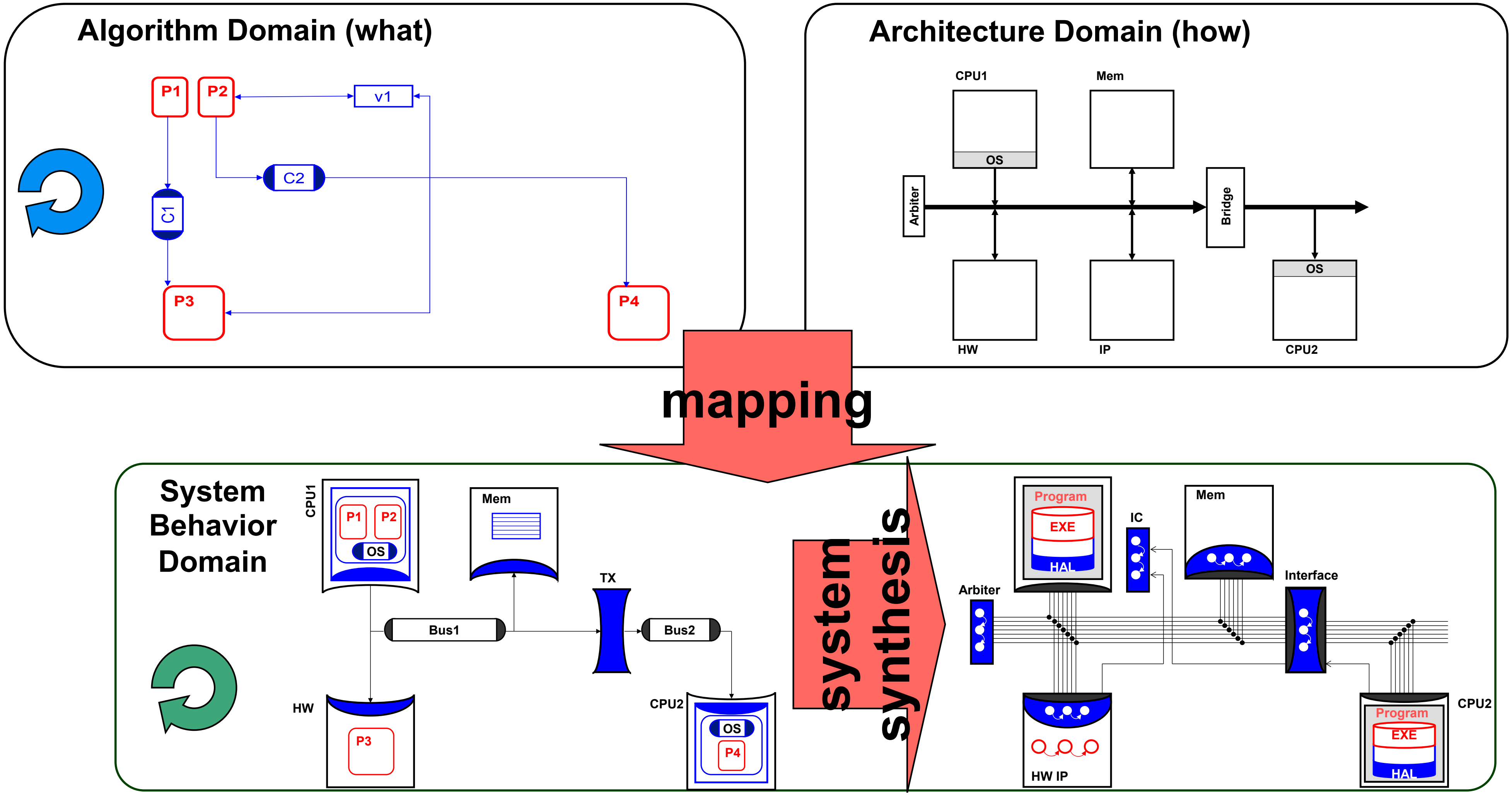
Counterexample Found in 5 Seconds!

Checking 373 Properties
Found Over 60 Errors

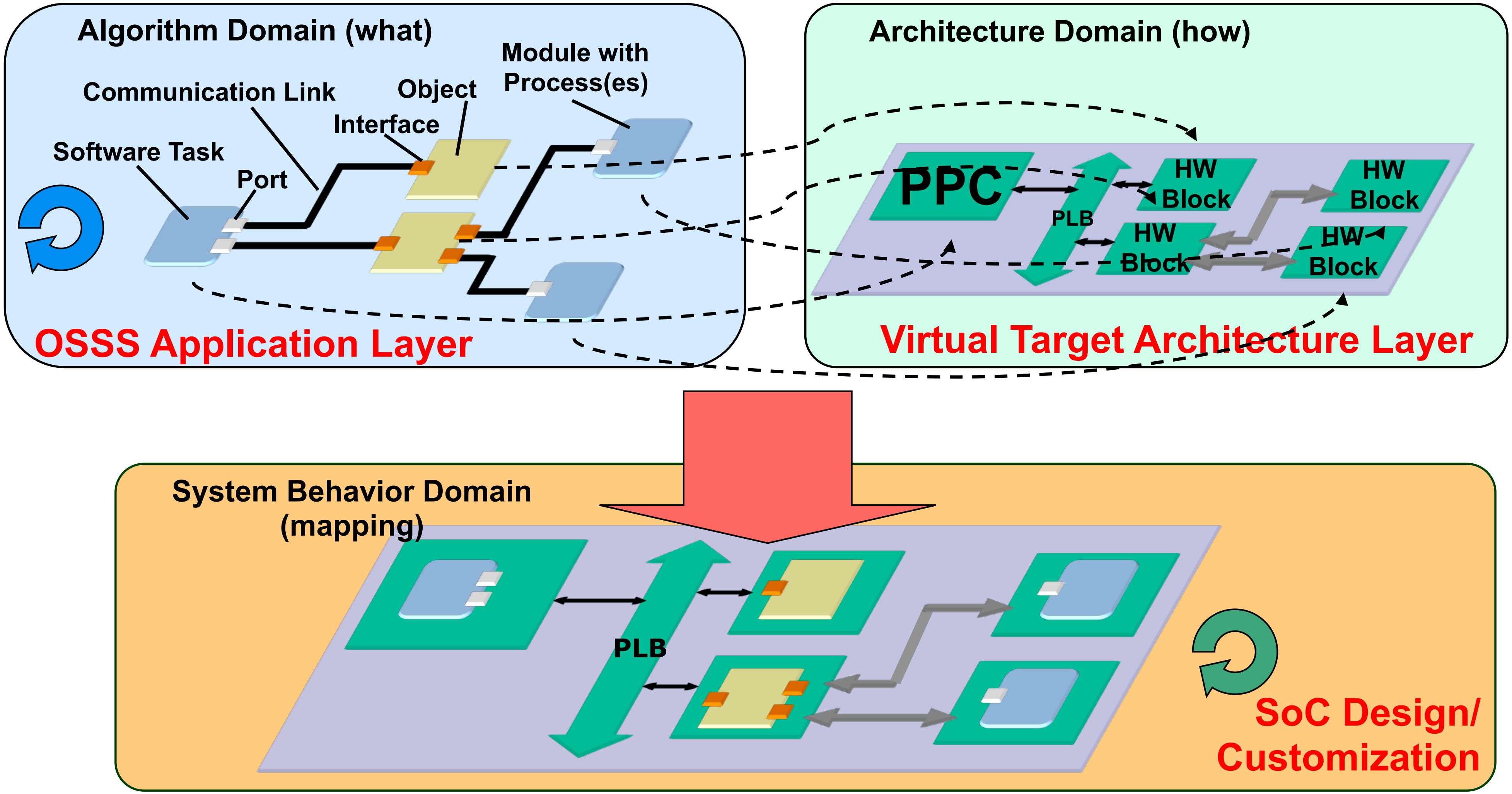
8 System Design Process



9 Overview of Model-Based Design



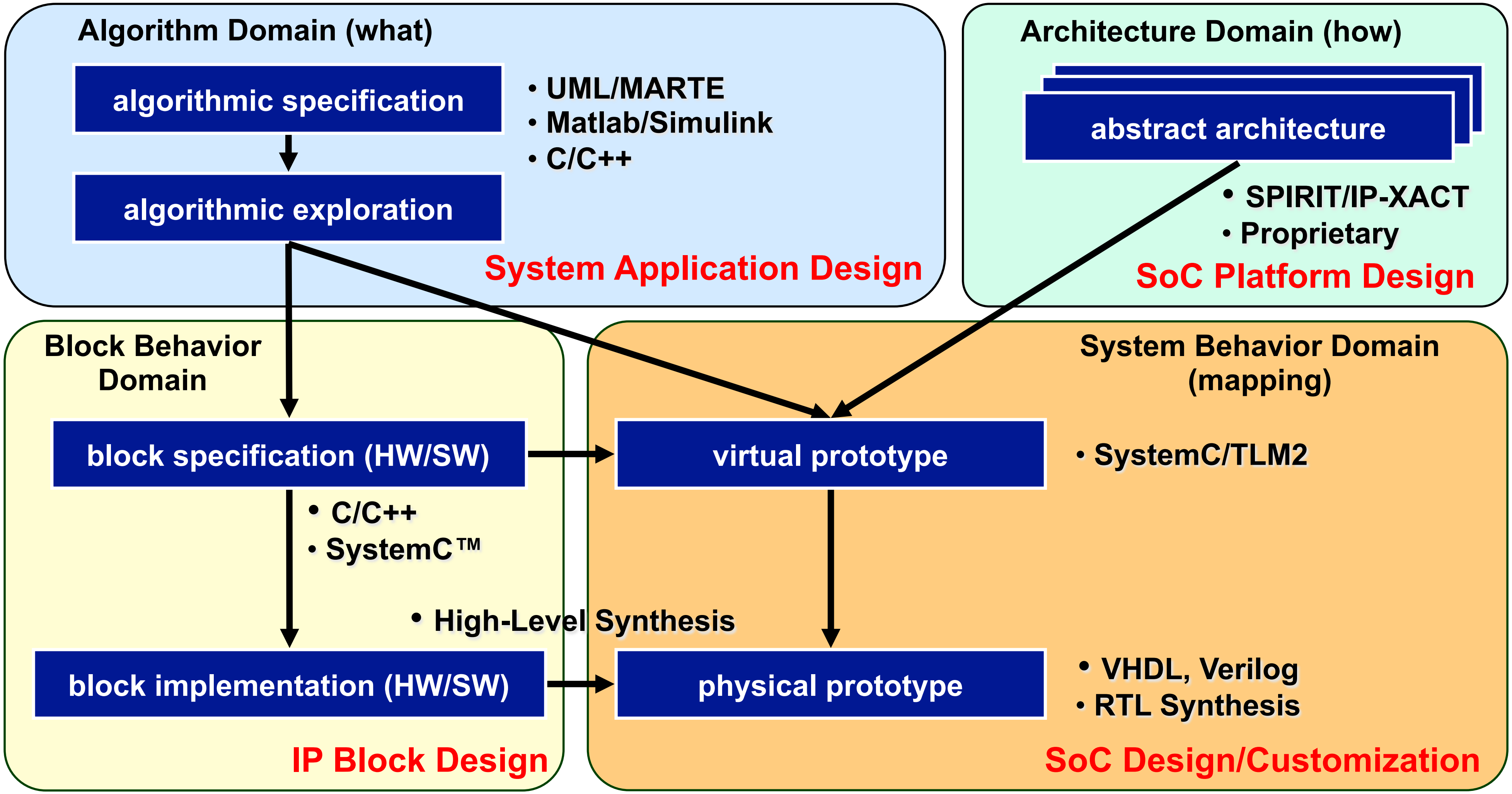
▸10 Model-Based Design with OSSS (Oldenburg System Synthesis Subset) & SystemC



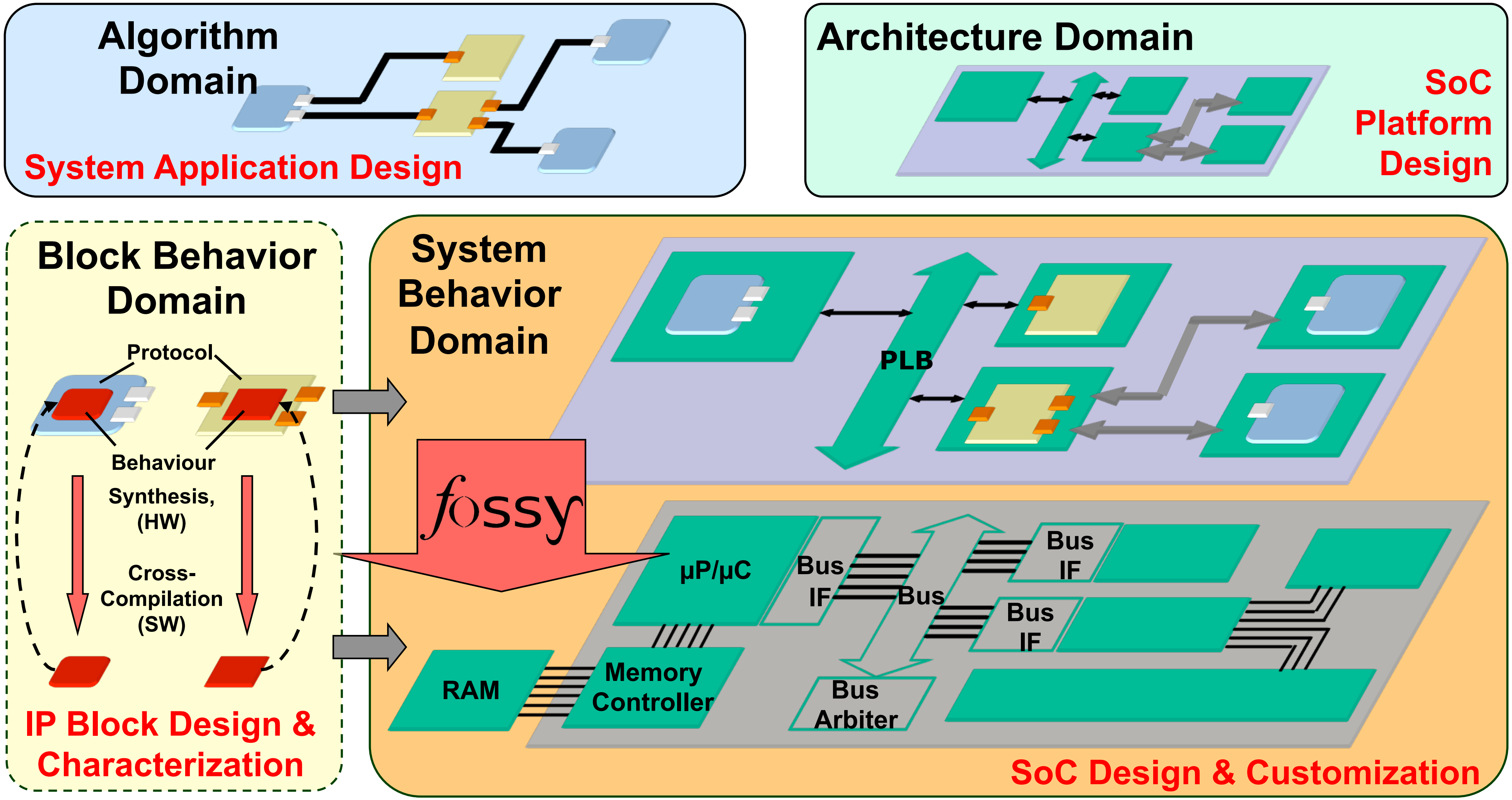
▸ 11 Model-Based Design with OSSS (Oldenburg System Synthesis Subset) & SystemC

- Using Model-Based Design and Object-Oriented (OO) techniques
- Application (*what*)
  - Executable and parallel application model (*a SW Designer understands*)
  - Separation of behavior and communication (computation and protocol)
  - Communication via application-specific method calls…
  - … on Communication Objects
  - A way of expressing implementation alternatives (e.g. HW or SW)
- Target Platform (*how*)
  - Static (non executable) structural model (*a system architect understands*)
  - PEs (CPU, DSP, dedicated HW), Memory, Communication Channels
- Retargetable TLM Synthesis (*mapping*)
  - Executable Application Description +
    Target Platform Description +
    Mapping Constraints = Executable System Prototype
- Synthesis of Executable System Prototype for Platform-FPGA

12 Model-Based Design: Languages & Tools



13 Model-Based Design with OSSS (Oldenburg System Synthesis Subset) & SystemC

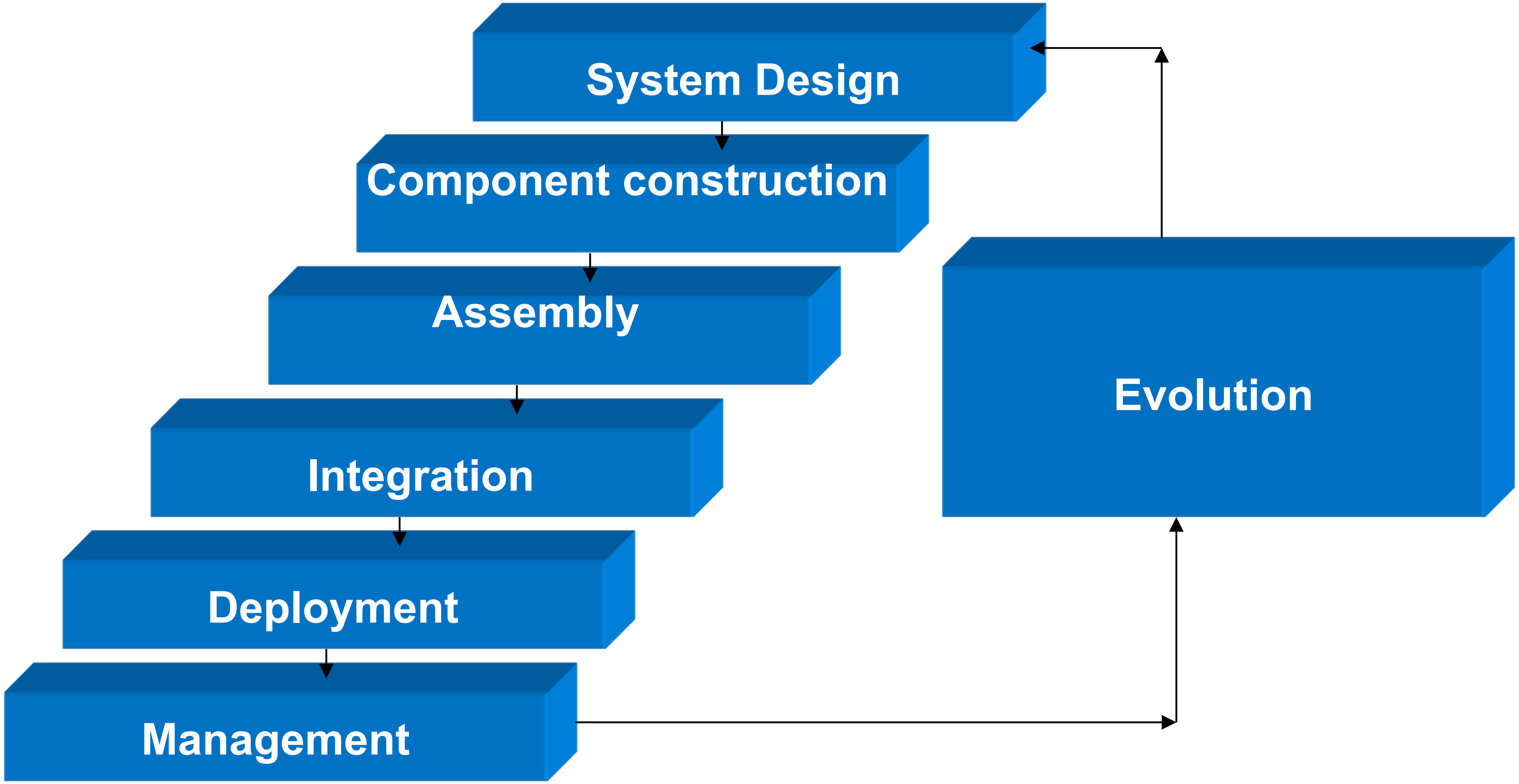


Model Driven Architecture

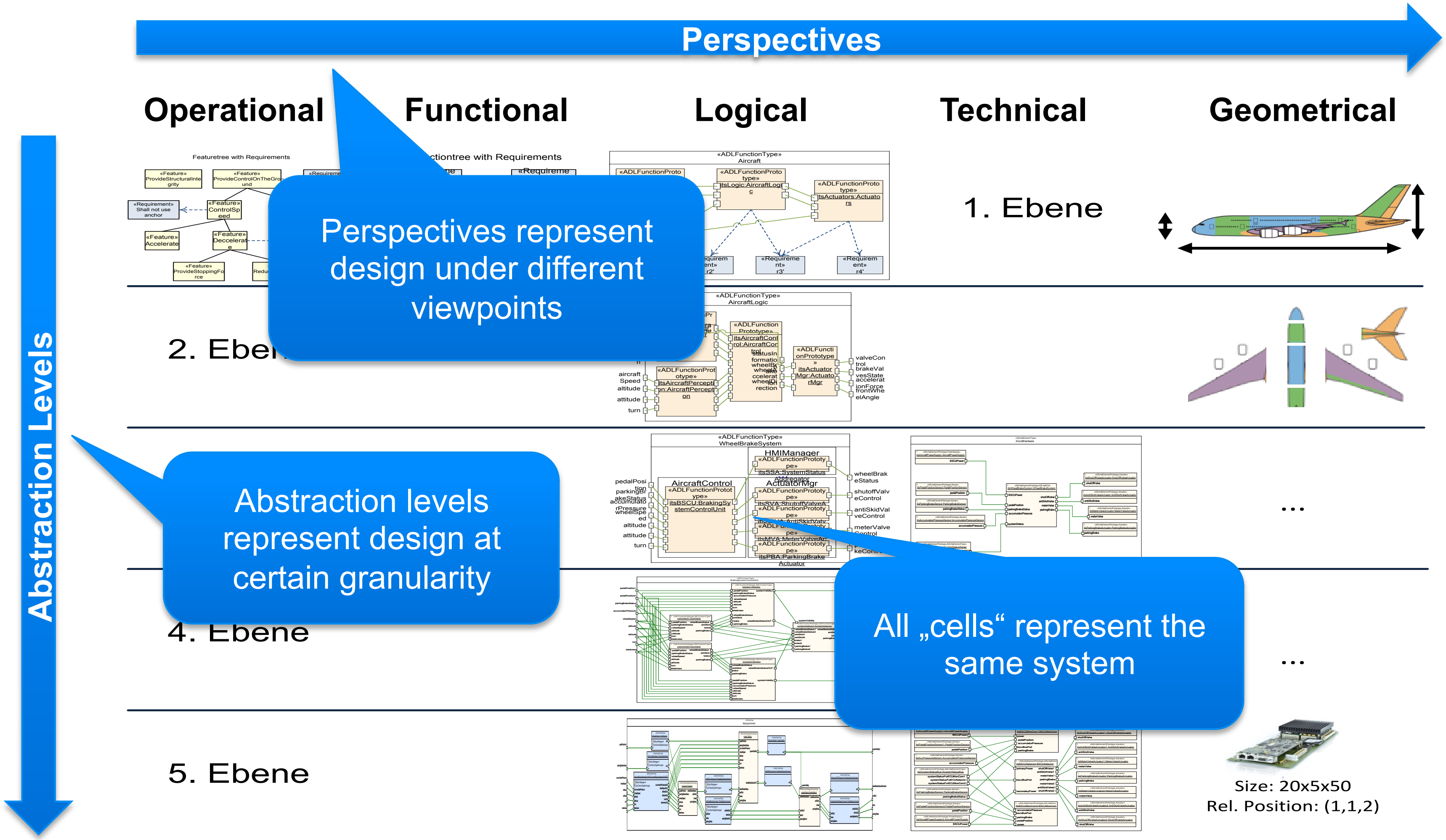
Provides an open vendor neutral approach to the challenges of business and technology change. It separates business and application logic from underlying platform technology.

15 Development Process

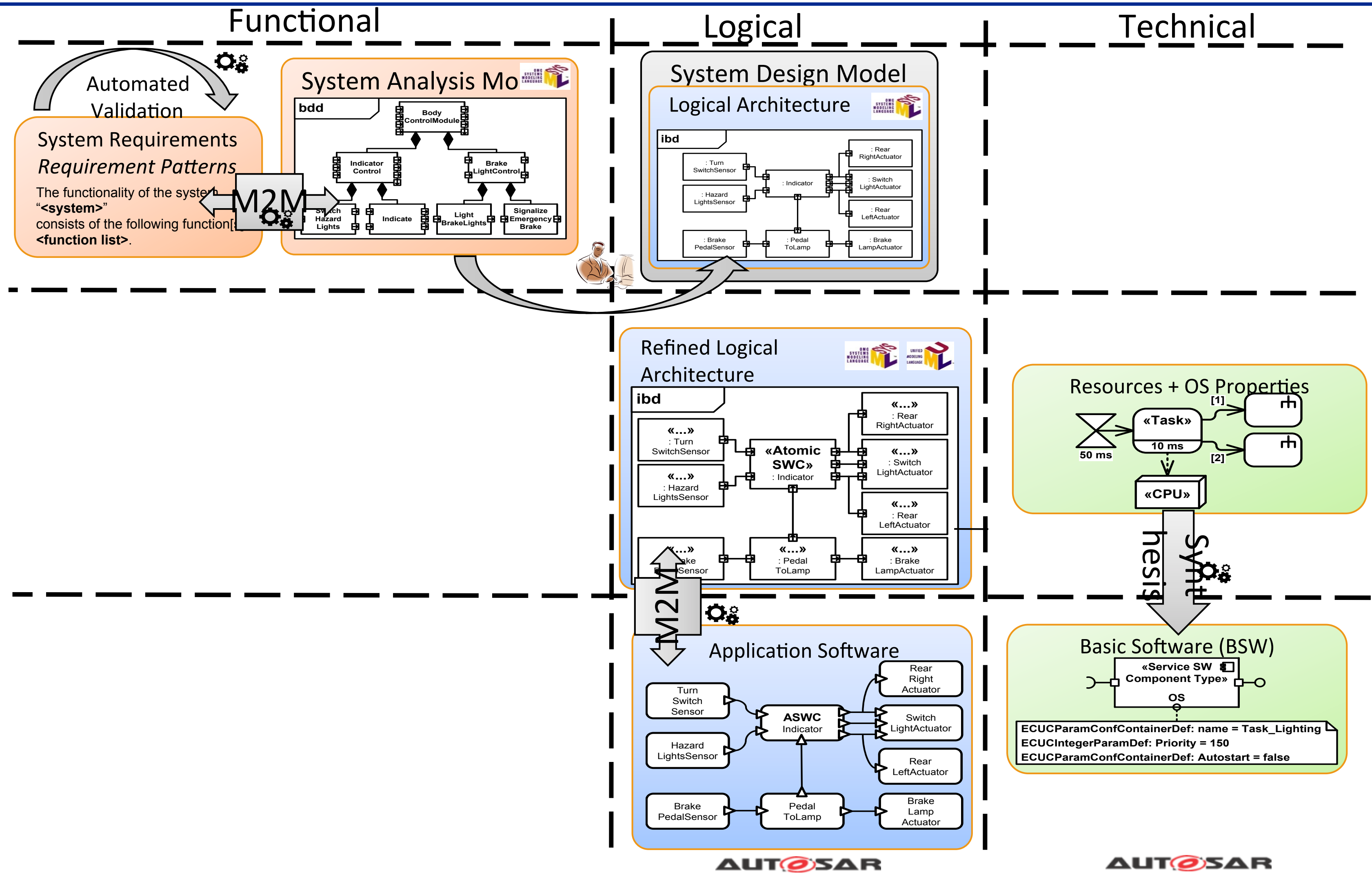
During the development process of any system it is necessary to support interoperability with specifications that address integration through entire lifetime of the system:



▸16 SPES Architecture Meta-Model Abstraction Levels and Perspectives



▸17 Example – Comfort Control Unit



▸19 Model-based Design along SPESMM

- SPESMM provides for component-based design:
  - Components and their interfaces (HRC) constitute the design architecture
  - Aspects of a component may be defined by
    - specifications,
    - implementation
- In SPESMM, HRC are the common modeling artifacts:
  - functions, logical components, software tasks, … are modeled in terms of HRC.
- Contracts are used to formalize *specification* of aspects
  - Well suited to explicate responsibilities between different actors of a design
    - What behavior is expected (guaranteed) by a component,
    - In which contexts (assumption)

20 Requirement Specification Language (RSL)

- Contracts have formal underpinning due to (trace) semantics
- Various formalisms thinkable for specification of contracts:
  - Automaton based
  - Logical formulas
  - Formalisms developed in ZP-AP1
- RSL provides user friendly and easy to understand natural language like formalism:
  - Based on patterns, with attributes to be instantiated
  - Patterns for different aspects:
    - Functional, real-time, safety, …
- Example for functional pattern:

  whenever *request* **occurs** *response* **occurs within** *[10ms,20ms]*
- Different attribute types:
  - Events, conditions, Intervals, time values, components
- In this talk, we will see some further examples

21 Traversing the Meta-Model in a Design Process

- Typically, a design process along the SPESMM involves many different design steps.
  For example:
  - Identification of use cases and initial requirements in the operational perspective,
  - Functional decomposition,
  - Partitioning of functions into logical components,
  - Allocation of logical components to a technical system that distinguishes software, processing and communication hardware, mechanical, hydraulic, and electrical components.
  - Allocation of technical system to geometrical space.
- Design steps are performed at different levels of abstraction, representing different refinements of the initial design.
  - SPESMM does not define which “cells” are used in a particular design process

- This talk does not cope with a particular design process
- This talk does not define which types of artifacts are required at particular perspectives

22 Key Design Steps

- Each design step identified in *any* process supported by the SPESMM can be represented by a sequence of *key design steps*:
  - **Decomposition** Most component types defined in the SPESMM can be decomposed into smaller parts, or sub-components.

    Examples: Functions may be decomposed into sub-functions, decomposition of logical components.
  - **Allocation** Components within different perspectives are entangled in that they represent (partly) the same system entities.

    Functions for example are allocated to logical components.
  - **Realization** The same system may be represented at different levels of abstraction. We say, the system at a certain abstraction level (and the same perspective) *realizes* the system at the higher levels.
  - **Implementation** Finally, components get implemented.

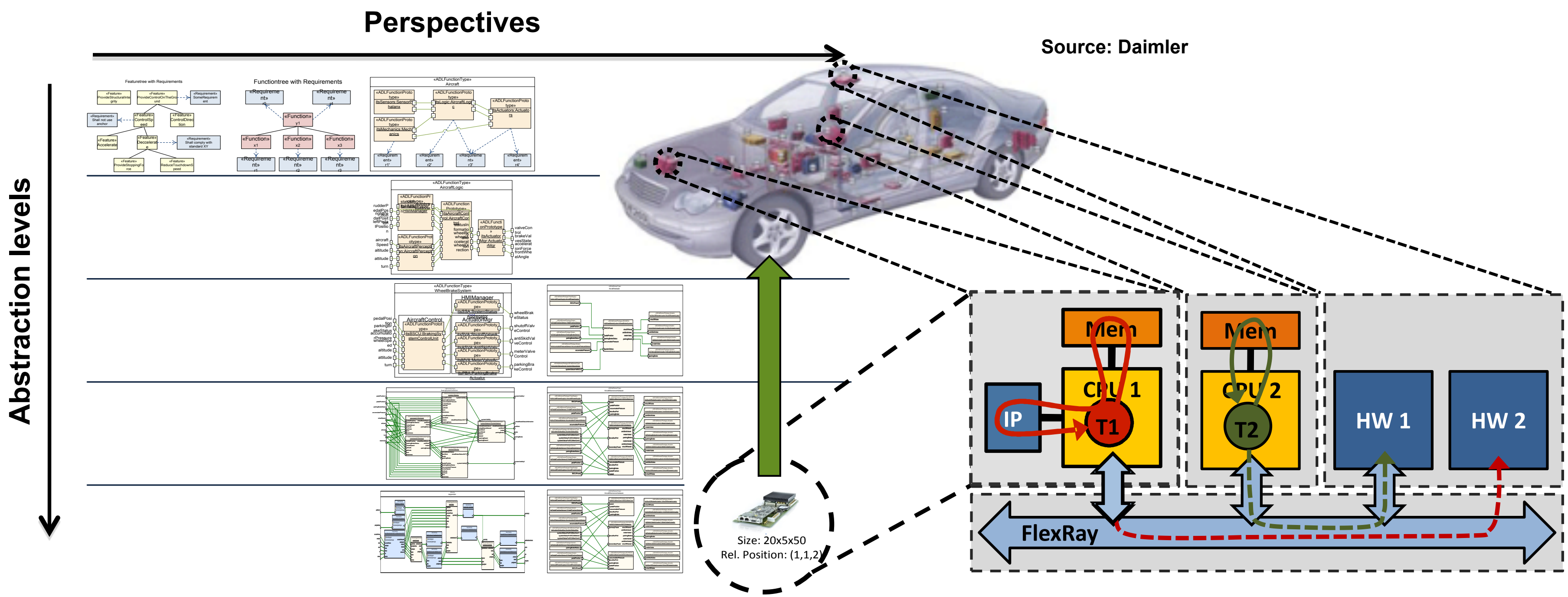
    Implementations may be automata models, StateCharts, MatLab models, and even C-Code.

23 Design Steps and Traceability

- Performing a design process thus means (in arbitrary order):
  - Component-based design in each “cell”:
    - Definition of requirements,
    - Definition of components, their interfaces, and communication structure,
    - Specification of component aspects,
    - Decomposition of components into parts.
  - Traversal between “cells” in the matrix:
    - Shifting the viewpoint at a certain abstraction level (moving horizontally)
    - Shifting the abstraction level (moving vertically)
      - Refinement of the design
  - Evaluating responsibilities within a cell
    - Do all components satisfy their responsibilities (guarantees)?
    - Are assumptions of all components satisfied?
  - Evaluating responsibilities between cells
    - Do components satisfy their responsibilities with respect to different perspectives?
    - Does a model satisfy all responsibilities if the model at a lower abstraction level does?

▸24 ARAMiS project

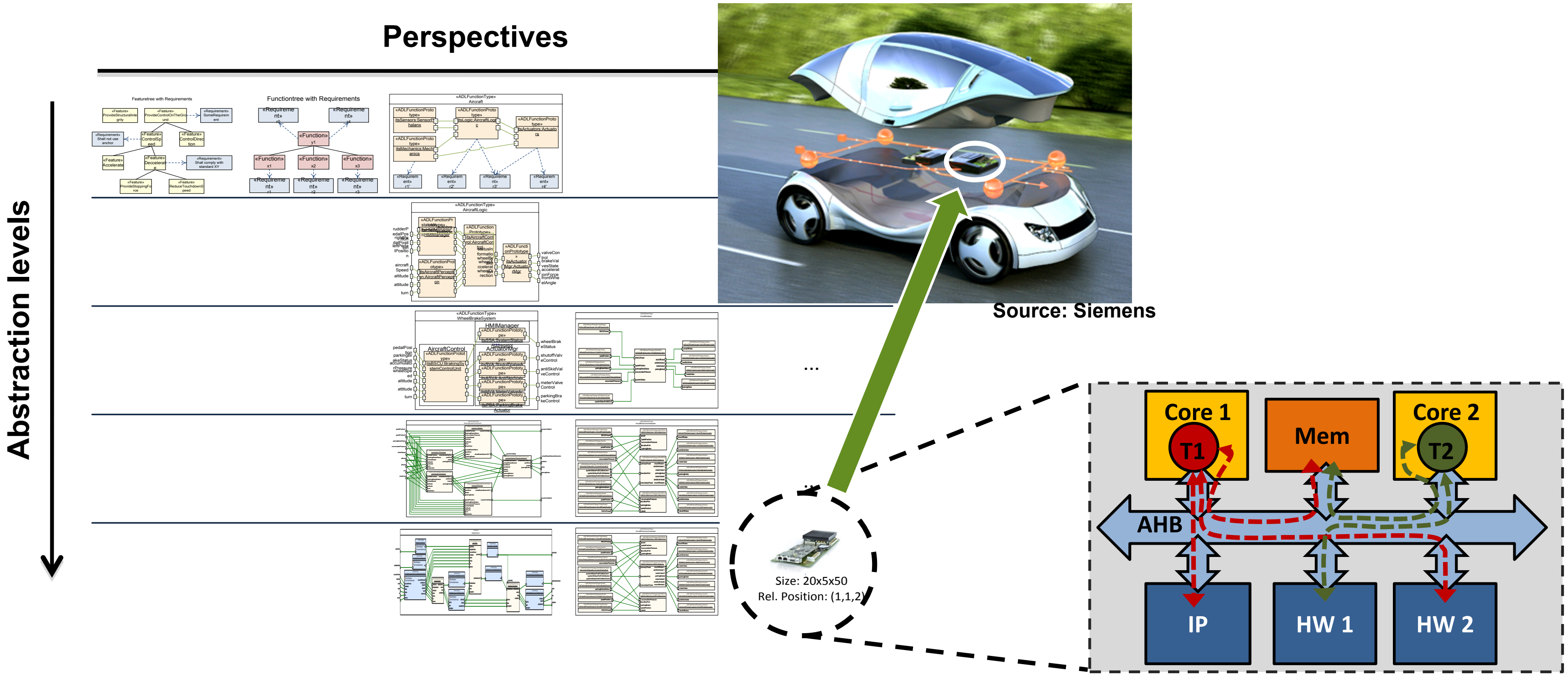
Automotive Railway And Avionics Multicore Systems



▸25 ARAMiS project

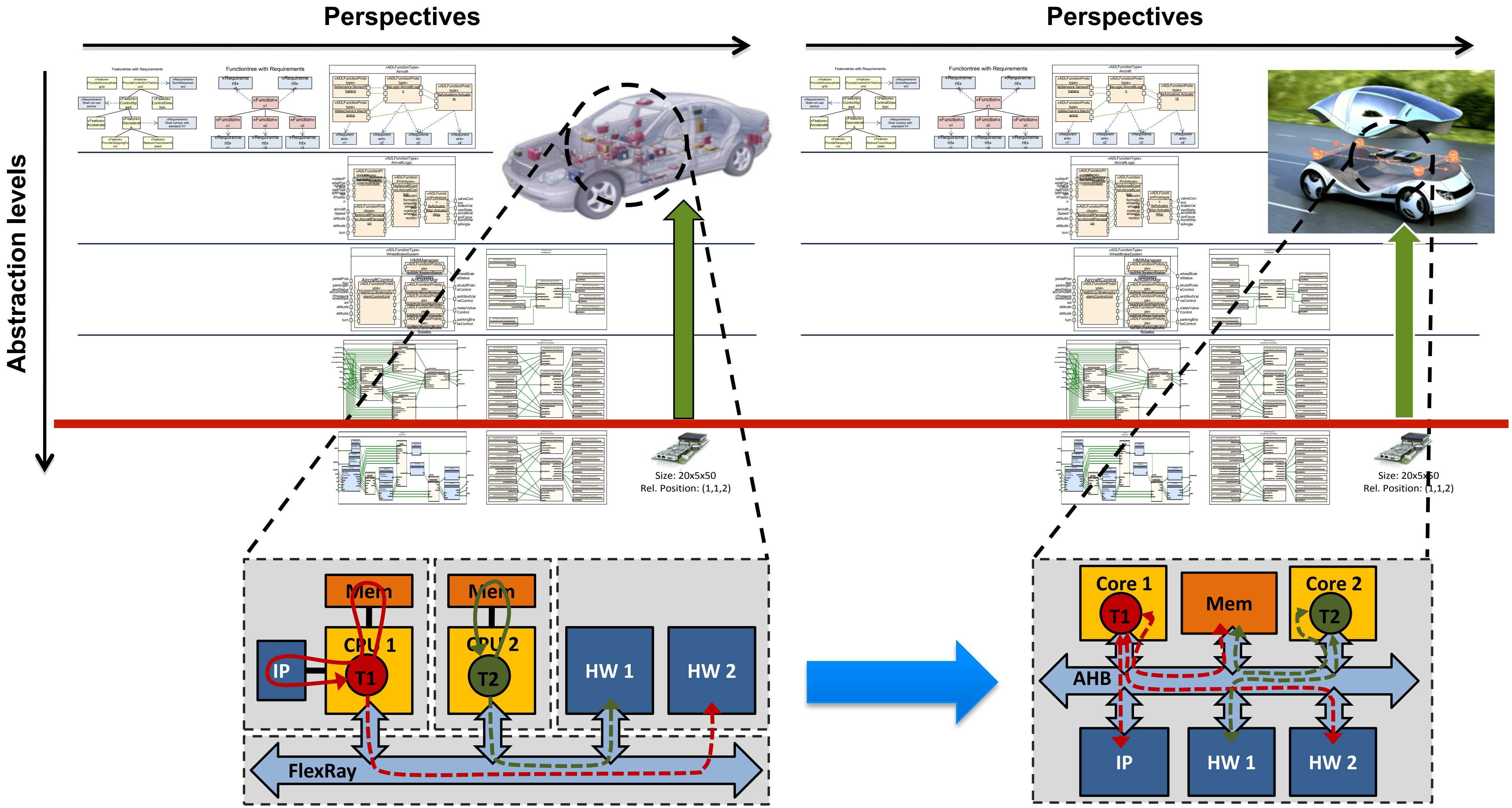
Automotive Railway And Avionics Multicore Systems

Bundesministerium für Bildung und Forschung



▸26 ARAMiS project

Automotive Railway And Avionics Multicore Systems



27 Conclusion

Model-Based Development

Automated Analysis

A Revolutionary Change in How We Design and Build Systems Nowadays with Model-Based Design