System Design: From Requirements to Implementation

A.Ferrari
O.Ferrante, L.Mangeruca

Advanced Laboratory on Embedded Systems S.r.l.
A Research and Innovation Company

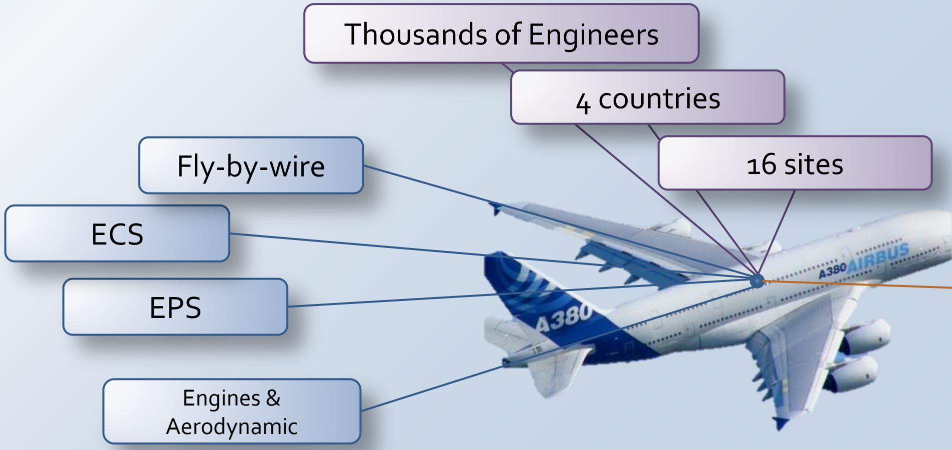
Outline

- ✓ Motivations
- ✓ Design using successive refinement
  - —Design flow description
  - —From requirements to sub-systems
  - —From sub-systems to functional decomposition
  - —From functional decomposition to physical implementation
- ✓ Overview of existing design languages
- ✓ Conclusions

System Engineering Challenges

Large systems

- **Heterogeneous**
- **Distributed**
- **Complex**
- Described using **natural language requirements**
- Developed **concurrently**
- **High** Costs
- **Long time-to-market**




Forbes.com
Home Page for the World's Business
U.S. EUROPE ASIA
Home | Lists | Business | Tech | Markets | Personal Finance | Entrepre
Bonds Commodities Currencies Economic Calendar Economy Emer
E-Mail | Print | Comments | Request Reprints | E-Mail Newsletters | RSS

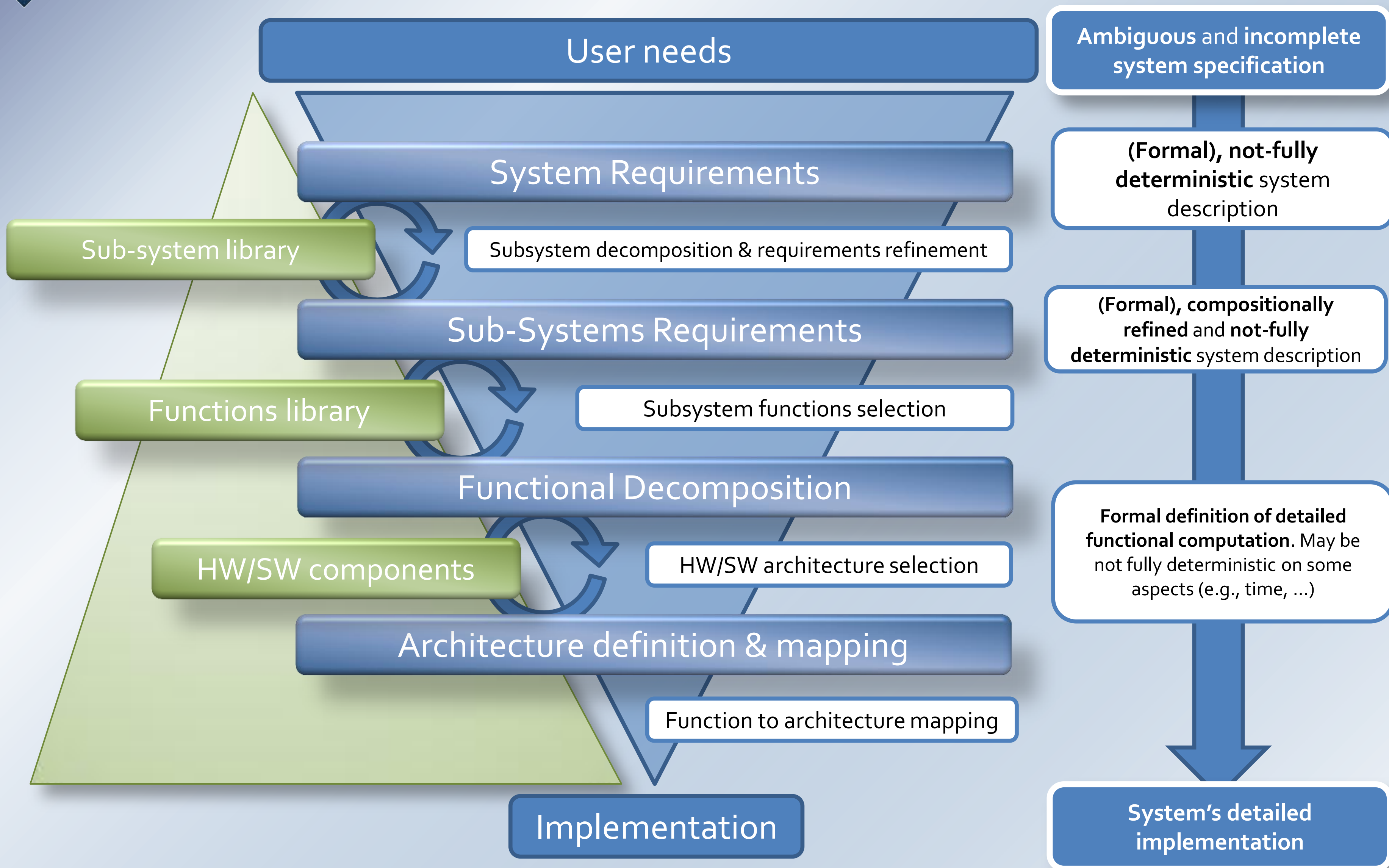
Market Scan
Major Turbulence For EADS On A380 Delay
Mary Crane, 06.14.06, 12:05 PM ET

Source: New York Times (http://www.nytimes.com/2006/12/11/business/worldbusiness/11iht-airbus.3860198.html?pagewanted=2&_r=2)

Outline

- ✓ Motivations
- ✓ **Design using successive refinement**
  - — Design flow description
  - — From requirements to sub-systems
  - — From sub-systems to functional decomposition
  - — From functional decomposition to physical implementation
- ✓ Overview of existing design languages
- ✓ Conclusions

Design using successive refinement



From System Requirements to Sub-system Requirements

Design step **actions**

- **Requirements** decomposition
- Sub-systems **interfaces** identification
- **Responsibility** identification (concurrent development)

Library elements

- **Sub-system** elements
- **Sub-system interface** elements

System Requirements

Sub-system library

Sub-Systems Requirements

Design step verification

Requirement Analysis

The sub-system requirements are a refinement of the system's requirements?

Target level model verification

Feasibility Analysis

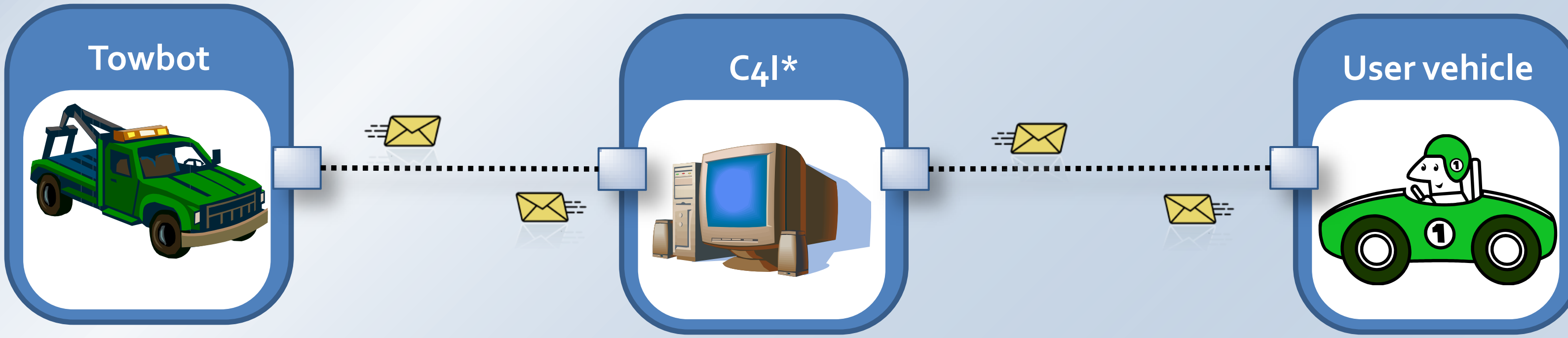
Does an implementation exist? (no conflicting requirements)

(Non-)Determinism Analysis

Are multiple behaviors (and implementations) expected?

SPRINT ATS use case

Automated Towing System (ATS)



Receives dispatch commands and autonomously moves to the requested location

Uses an automatic **cruise control** system

Centralized control for the **dispatching** of the towbot

Should **coordinate** user vehicles requests and the towbot dispatching

In case of emergency requests a towbot

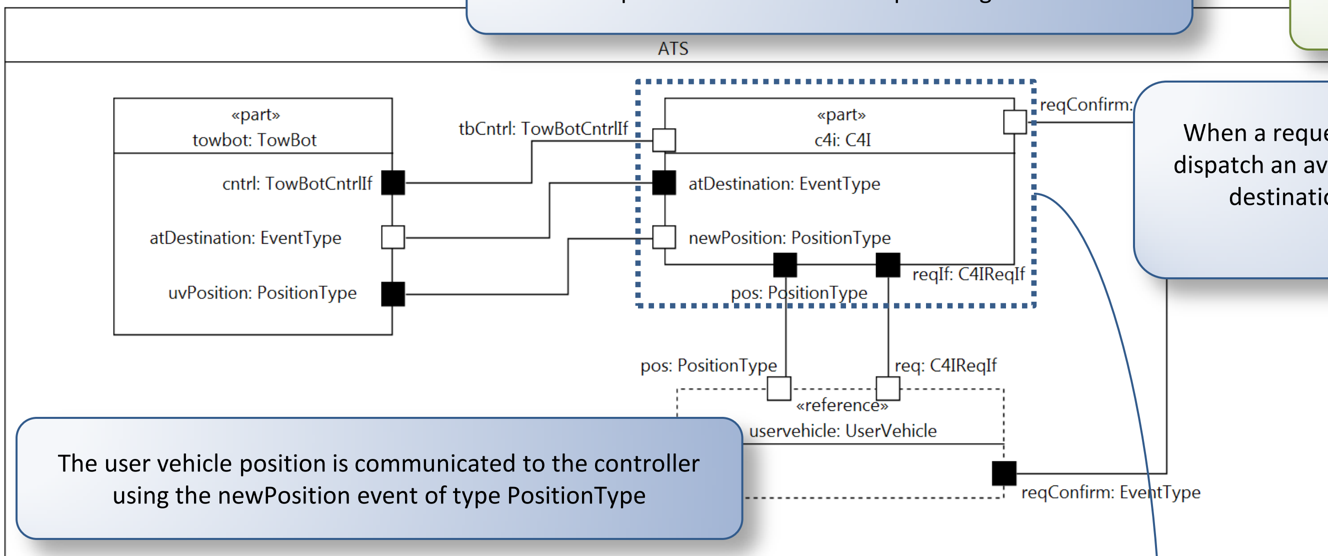
Waits until a towbot arrives

***Command Control Communications Computers, and Intelligence**

From requirements to sub-system requirements - Example

The C4I handles the communication with the tow bots and implements the tow-bot dispatch algorithm

C4I Sub-system **decomposition** and requirements **formalization**



When a request arrives (**req event)** to the controller, it dispatch an available tow-bot and waits until it arrives at destination (input **atDestination**) generating a **reqConfirm** event

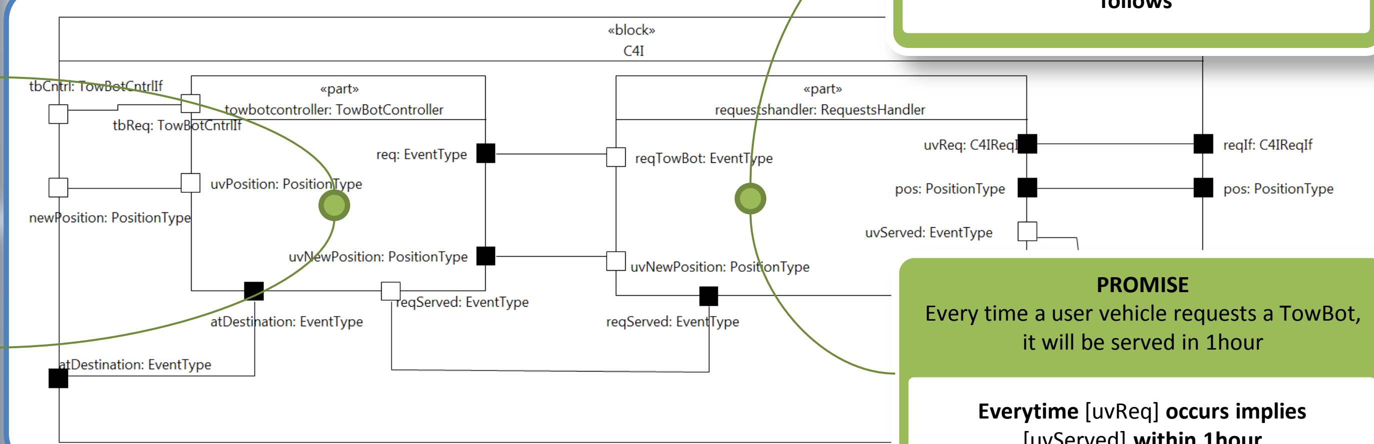
The user vehicle position is communicated to the controller using the newPosition event of type PositionType

ASSUMPTION
Every time a TowBot is requested , a TowBot dispach confirmation follows

Everytime [reqTowBot] **then** [reqServed] **follows**

ASSUMPTION
Every time a TowBot dispatch request is send, the TowBot arrives at destination

Everytime [tbReq] **then** [atDestination] **eventually**



PROMISE
Every time the TowBot controller receives a dispatch requests, it sends a request to the TowBot in 5ms

Everytime [req] **then** [tbReq] **happens within 5ms**

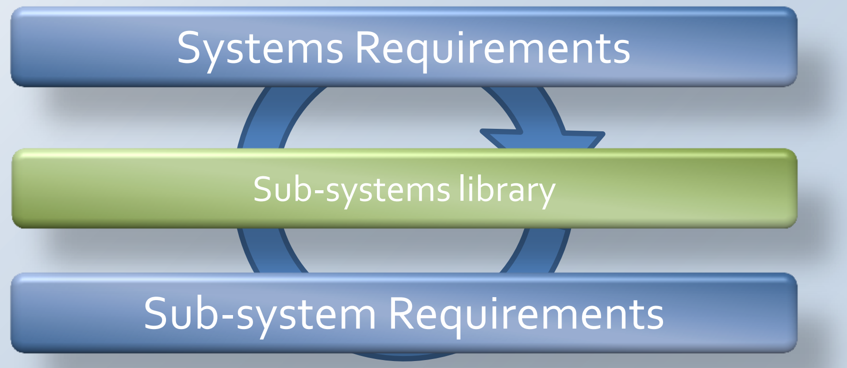
PROMISE
Every time a user vehicle requests a TowBot, it will be served in 1hour

Everytime [uvReq] **occurs implies** [uvServed] **within 1hour**

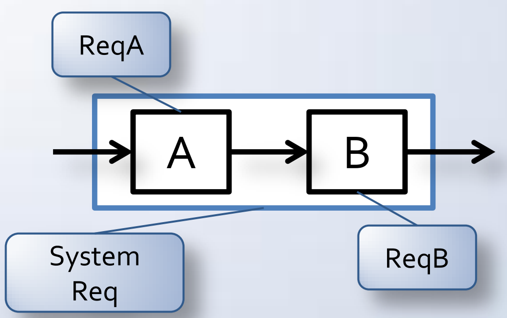
Design using successive refinement

Definition and **selection** of system logical components

- **Requirements** decomposition
- Sub-systems **interfaces** identification
- **Responsibility** identification (concurrent development)



The need of a **compositional approach** and **component-based modeling languages**



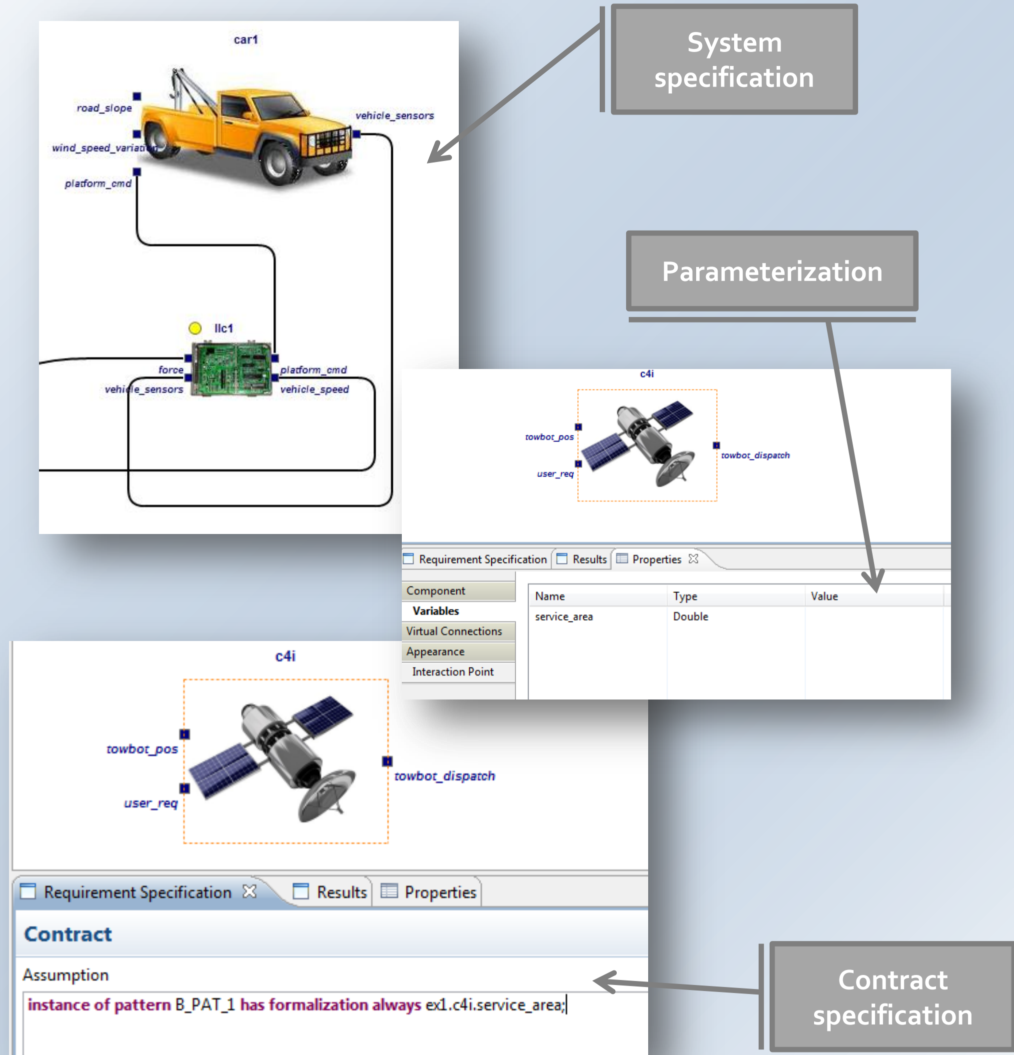
Are ReqA and ReqB a valid decomposition of System Requirements?

If A satisfies its requirements and B satisfies its requirements, is this true also for their composition?

If A satisfies its requirements and B satisfies its requirements, is it true that they also satisfies the System Requirements?

ALES Experience – Requirements formalization using the Contract Editor tool

- ✓ **Graphical editor** for system specification
  - — Graph basic semantics
  - — Native concepts: component, port, connection, parameter, variable
  - — Eclipse & EMF underlying technologies
    - Unique formalized model for capturing design
- ✓ **Multiple DSL**s support
  - — E.g., system structure, distributed simulation structure, etc…
- ✓ **Visual representation of state**
  - — Textual (displays/scopes) or change of image/shape/color of components/lines
- ✓ Dedicated **parameterization** view
- ✓ **Contracts specification**
  - — **Pattern based** specification
  - — Textual language
- ✓ Plug-in based framework
  - — New functionality can be built using the eclipse mechanism



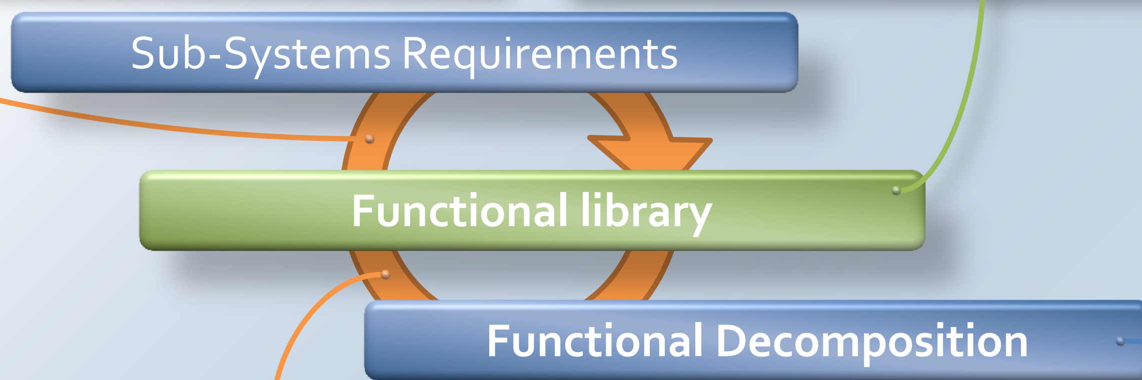
From Requirements to Functional Architecture

Design step **actions**

- Functional elements **data-types & interfaces** definition
- **Communication primitives** selection
- Identification of **appropriate MoC** for each functional component

Library elements

- **Data type & Interface** elements
- **Communication primitives** library (FIFO, rendez-vous,, …)
- MoC library (DE, CT, DT, …)



Design step verification

- For each subsystem, is the functional network **satisfying** the subsystem's requirements?

Target level model verification

- Are the interfaces of connected functional elements compatible?
- Are data types well-defined?
- Are chosen MoCs and communication primitives compatible?

From Requirements to Functional Architecture – Example

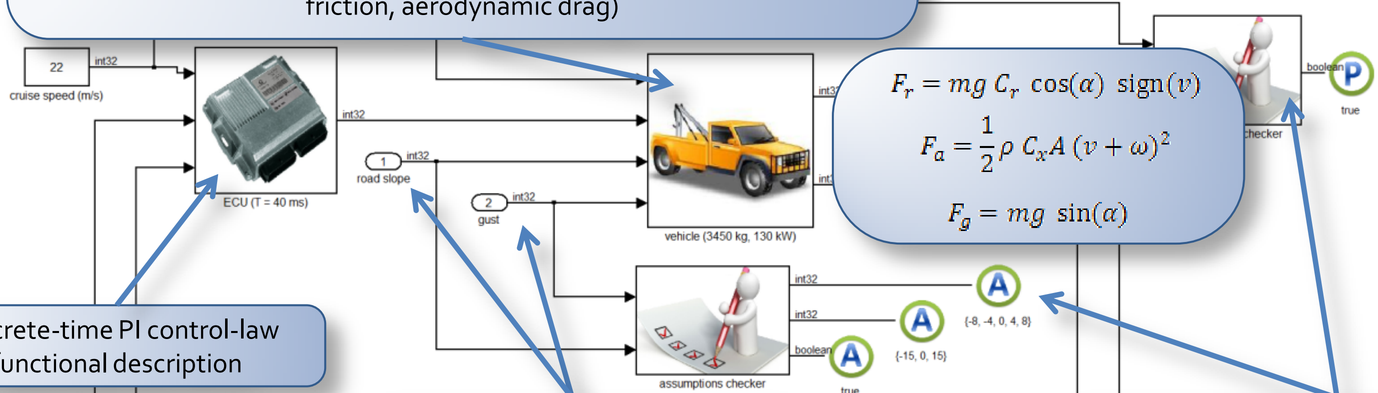
Towbot cruise control subsystem requirements

If the road slope changes in the range -8% and 8%, the cruise speed is equals to the reference speed with a maximum error of 5.5%

The cruise control shall tolerate variations of the wind speed between -15 m/s (headwind) and +15 m/s (tailwind) with a maximum variation of 5 m/s every sampling period (T=40 ms).

Functional model

Discretized model of the cruise control subsystem (road slope, rolling friction, aerodynamic drag)



$$F_r = mg\ C_r\ \cos(\alpha)\ \mathrm{sign}(v)$$

$$F_a = \frac{1}{2}\rho\ C_x A\ (v+\omega)^2$$

$$F_g = mg\ \sin(\alpha)$$

Discrete-time PI control-law functional description

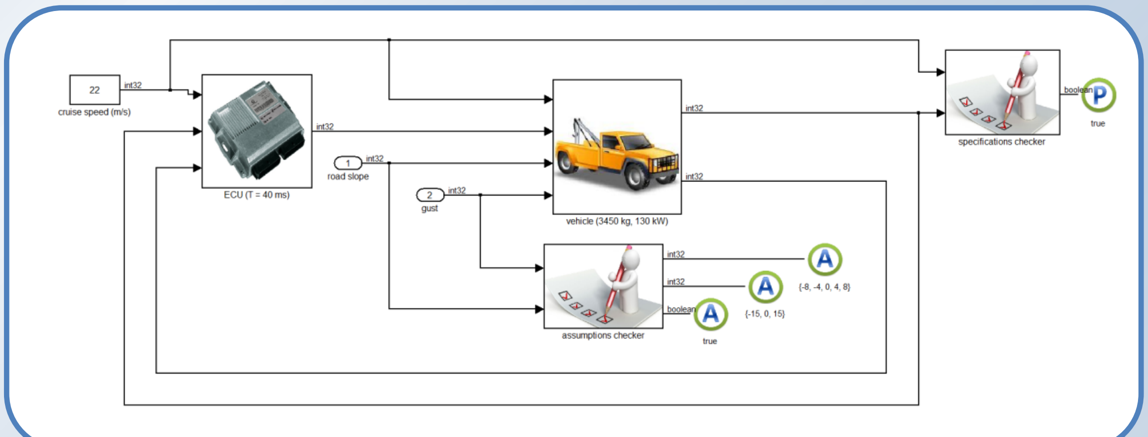
Sub-system free inputs

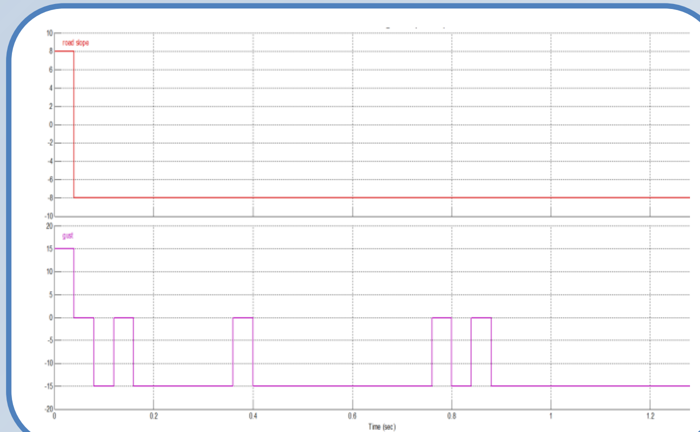
Formal sub-system requirements

Verification of the functional network

Cruise control contracts

- Contract specification
  - **Assumption** is the conjunction of three **assertions**
    - The slope value (slope percentage) is in {-8, -4, 0, 4, 8}
    - The wind gust value is in {-15, 0, 15} m/s
    - The wind gust, every 40 ms, can change of a maximum absolute value of 15 m/s
  - **Promise**:
    - the actual speed value is ±5.5% of the reference speed value





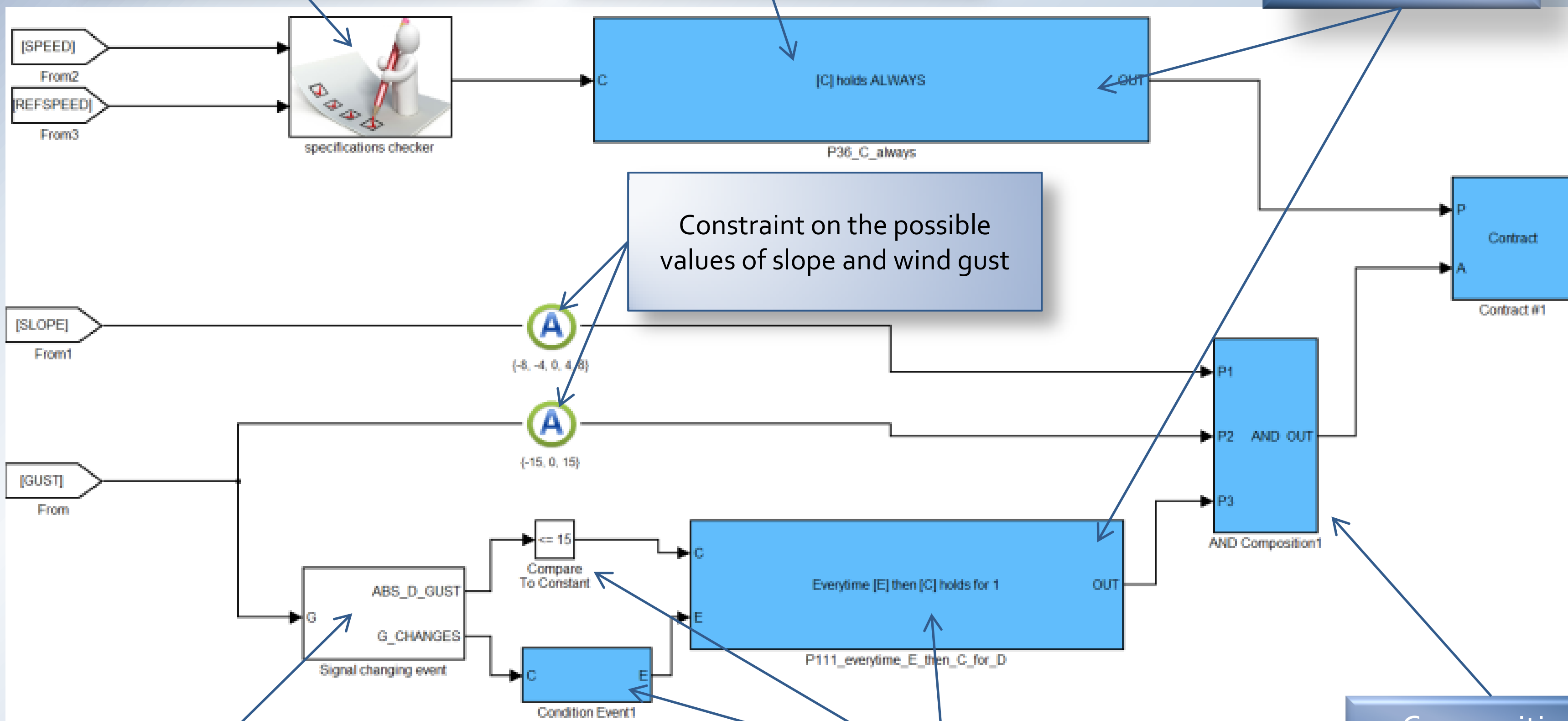
From requirements to functional architecture – Example

This block constraints current speed value to be ±5.5% of the reference value

This pattern constraint the input condition to be true at every step

Atomic patterns

Constraint on the possible values of slope and wind gust

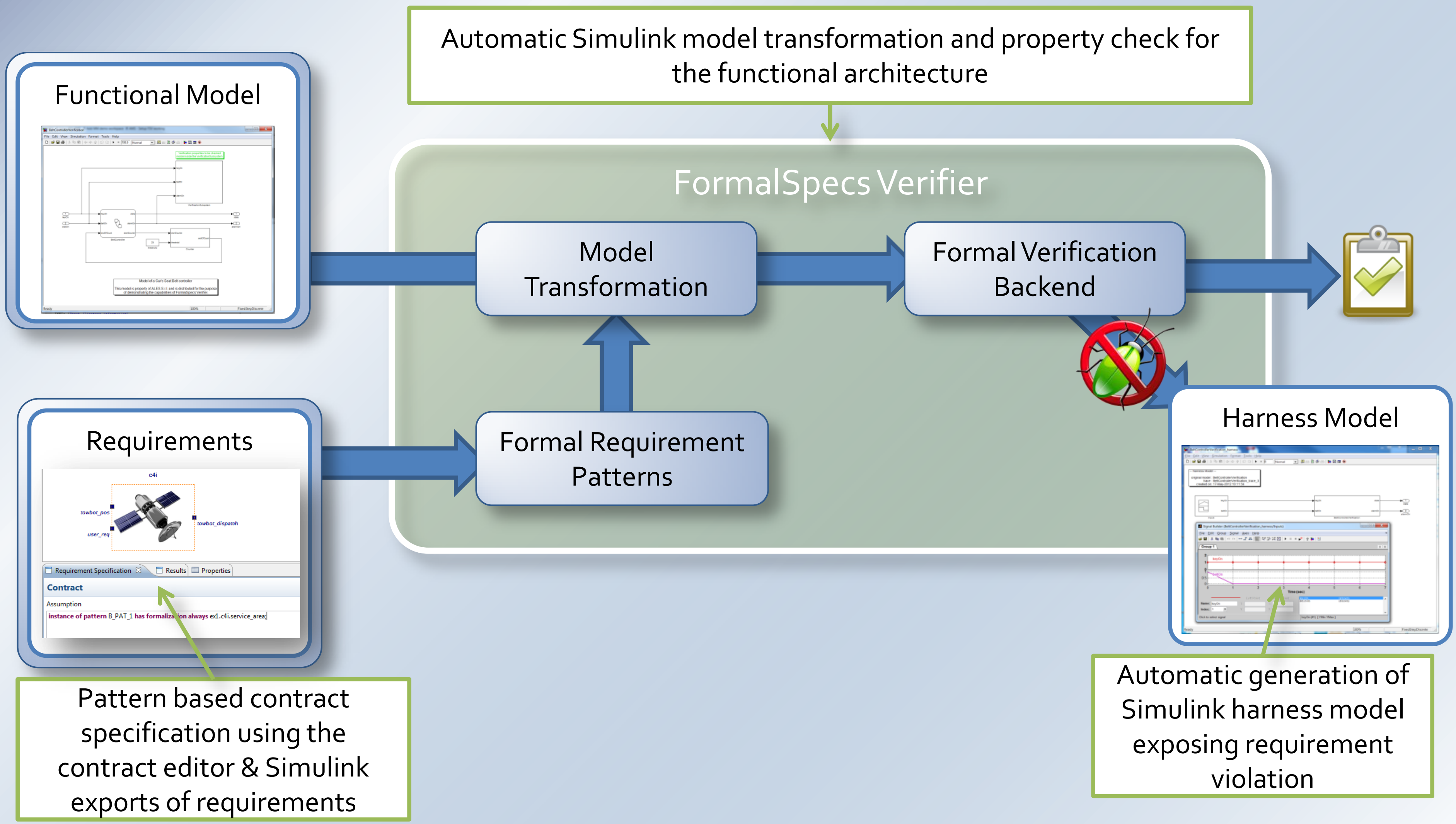


This block computes the derivative of the gust (D_GUST) and it assert a Boolean signal if the slope gust value has changed

The blocks assert that every-time the gust value changes, the derivative is less than ± 15 m/s

Composition blocks

ALES Experience – Requirement & Functional architecture description & formal verification



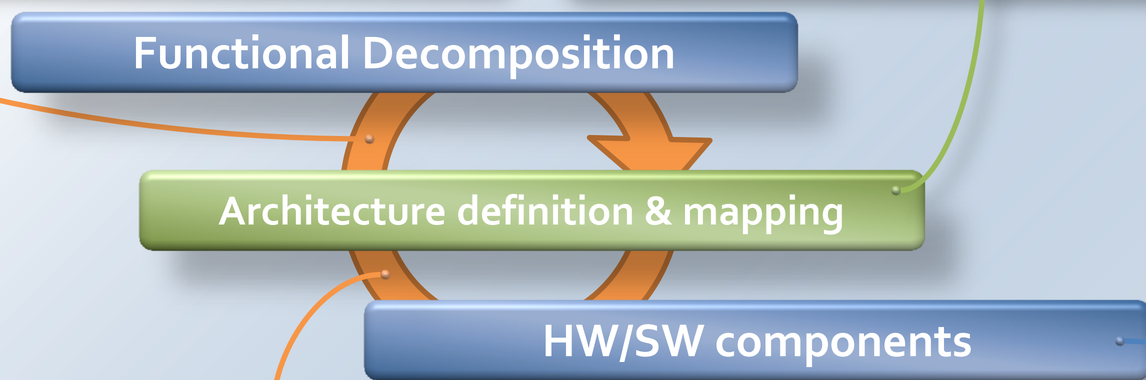
From functional architecture to physical implementation

Design step **actions**

- Selection of a set of physical components and architecture with predictable performance
- Identification of physical architecture
- Identification of legacy code and architectural COTS
- Automatic Code Generation for CUs

Library elements

- Non-functional constraints & Measures of Effectiveness
- **Computation** and **Communication** elements
- **Legacy** code
- **COTS**



Design step verification

- Are safety requirements met?
- For each component and subsystem, is its **architectural** representation **satisfying the requirement**s taking into account also **non functional constraints** (timing, performance, power consumption, weight, …)?
- Are sampling requirements met?
- Are chosen MoCs and communication primitives preserved?

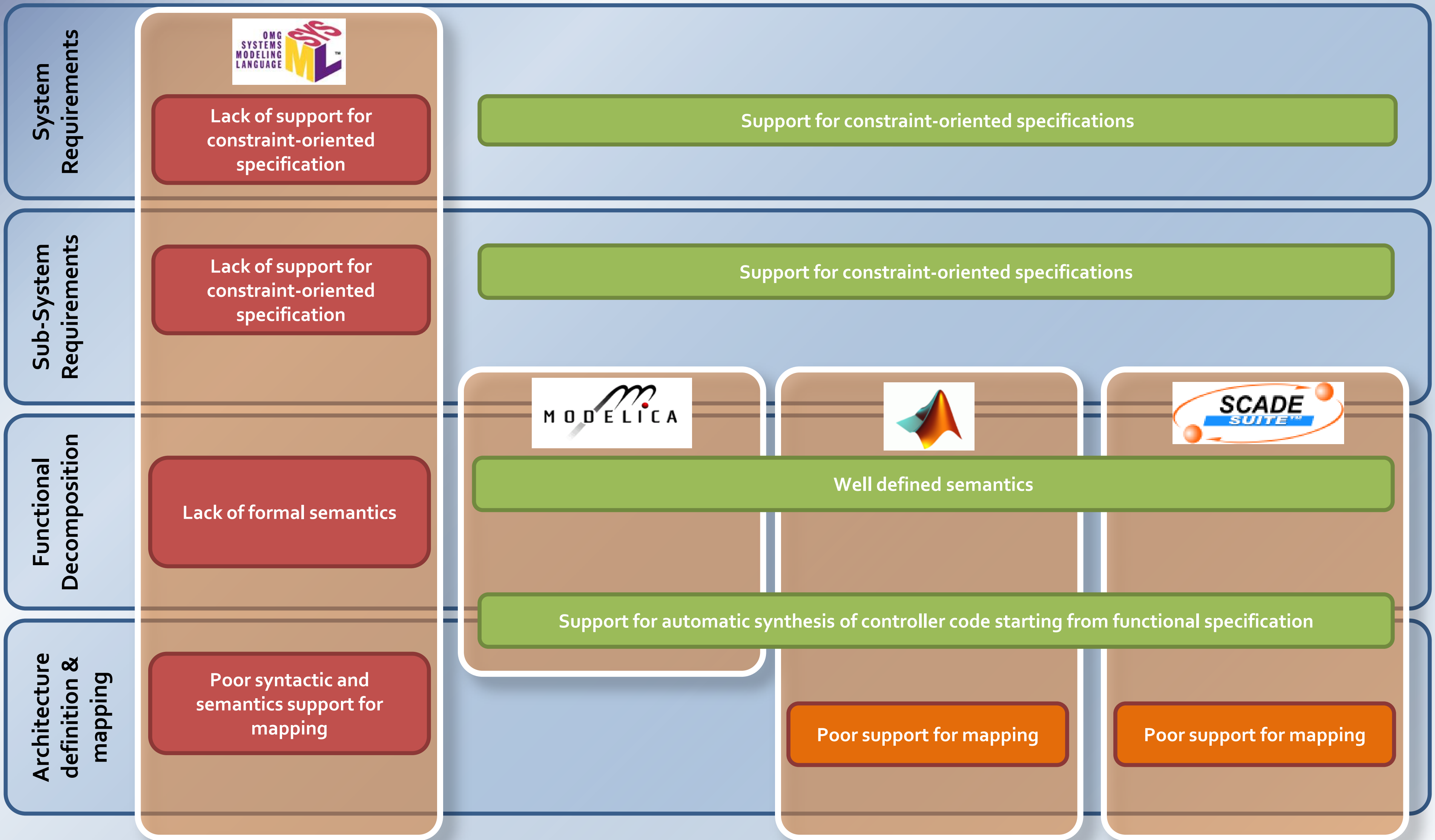
Target level model verification

- Is software code buggy? (over-/underflow, division by zero, null pointers, …)
- Are the interfaces of HW/SW components compatible?

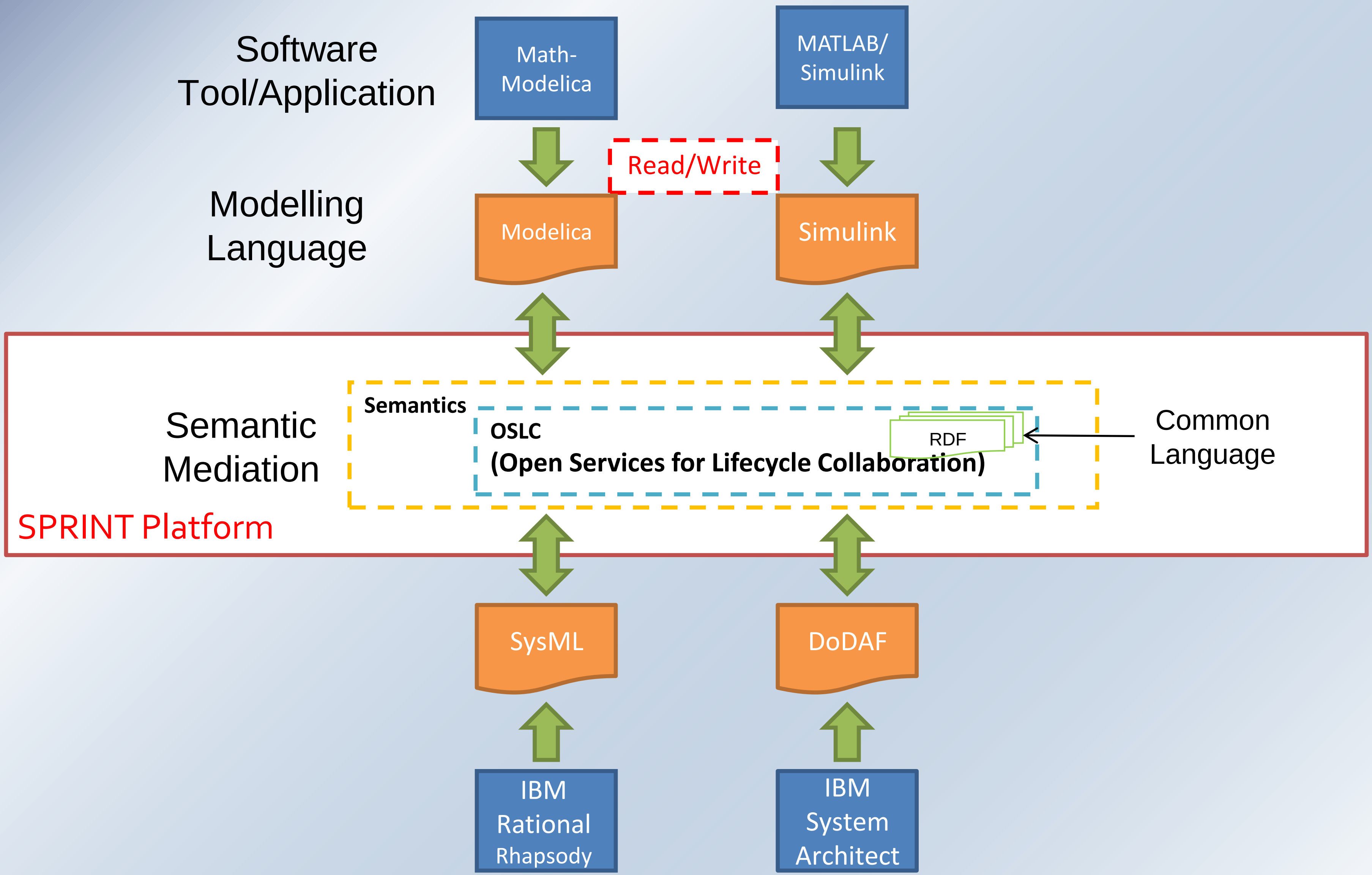
Outline

- ✓ Motivations
- ✓ Design using successive refinement
  - Design flow description
  - From requirements to sub-systems
  - From sub-systems to functional decomposition
  - From functional decomposition to physical implementation
- ✓ **Overview of existing design languages**
- ✓ Conclusions

Where Languages Map ?



SPRINT Approach!



Design using successive refinement – the ideal language

?

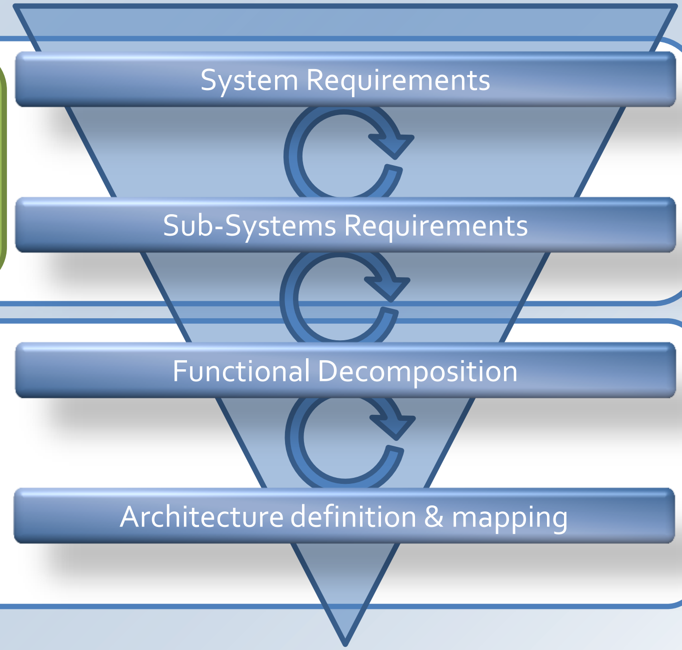
Language X

Graphical and textual representation

Used in academy and industry

Support for the compositional description of different model of computations

- Support for operational description of requirements
- Support for constraint-based description of requirements
- Well defined semantics
- Composable integration of heterogeneous MoCCs
- Support for non-deterministic specification
- Capability of capturing safety/timing constraints



- Formal and compositional description of structure and behaviors of functional and architectural network
- Support for efficient and formal description of mapping between function and architectural elements
- Automatic synthesis of controller code staring from functional specification & architectural constraints

Conclusion

- ✓ Summary
  - — Design flow using successive refinement
    - From requirements to sub-system
    - From sub-system to functional architecture
    - From functional architecture to physical implementation
  - — Equation-based language
    - Overview
    - Limitations