7.GRAZER SYMPOSIUM VIRTUELLES FAHRZEUG
May 27-28, 2014 - Graz, Austria

Semantic Model Integration for System Specification

- Meaningful, Consistent and Viable

Dr.-Ing. Oskar v. Dungern
enso managers gmbh, Berlin

Summary: Motivation and aim

Quality Assurance of Requirement Specifications is difficult. A hierarchical list of hundreds or even thousands of requirements, as provided by most Requirement Management Tools, does not lend itself for checking consistency or feasibility.

Putting the requirements in the context of system architecture and user scenarios, thus pursuing semantic integration, gives much better insight, allows for quality assurance and early estimation of development effort.

Summary: Presented results

An approach using system models in the specification phase is presented and illustrated with an example. While many modeling techniques offer a multitude of graphical and conceptual elements, the ‘Fundamental Modelling Concept’ (originally developed by Prof. Siegfried Wendt and his team at the University of Kaiserslautern and the Hasso-Plattner-Institute Potsdam) is based on three elementary model elements ‘Actor’, ‘State’ and ‘Event’, plus ‘Requirement’ and ‘Feature’.

Different model types are integrated by sharing model elements. For example, system composition diagrams and activity diagrams are expected to share ‘functions’. Semantic value is added by logic relationships between model elements, such as ‘Module A contains Module B’ or ‘Module B satisfies Requirement X’. The former relationships are automatically extracted from model diagrams (as many others), whereas the latter must be created manually.

The resulting collection of all model elements and relations allows for some formal checking and effective audits. Insufficiently covered areas and open issues are more easily identified as well as contradictions; loose ends can be consolidated and so forth.

This approach is more abstract and formal than others. Practical experience shows that it is concrete enough to be understood. Sharing and interrelating model elements adds a lot of value. Collaboration between disciplines is enhanced and so is the consistency of the system specification. It is perfectly suited for top-level modeling, before more detail is added in domain-specific design environments.

Please note that no new methods are introduced: The disciplines continue to use their preferred modeling techniques and tools. Added value is created through a common context for all model views.

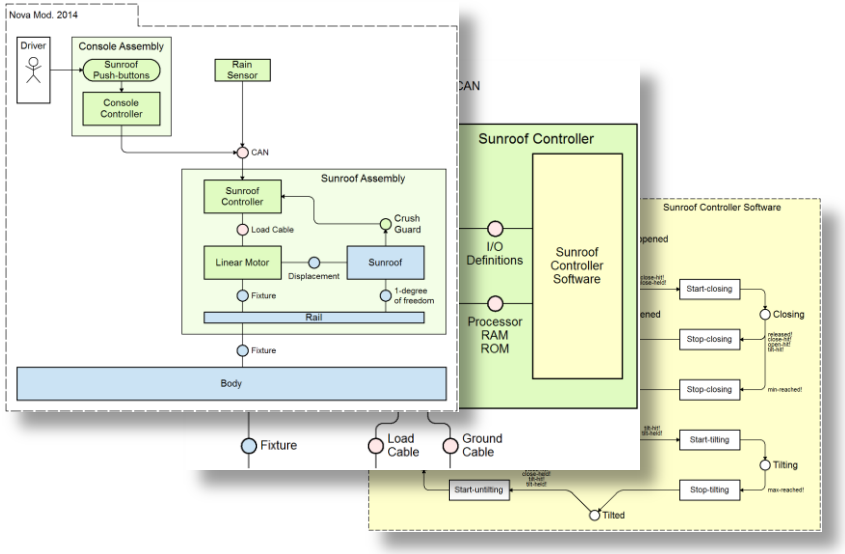
Summary: Originality, news value and benefit

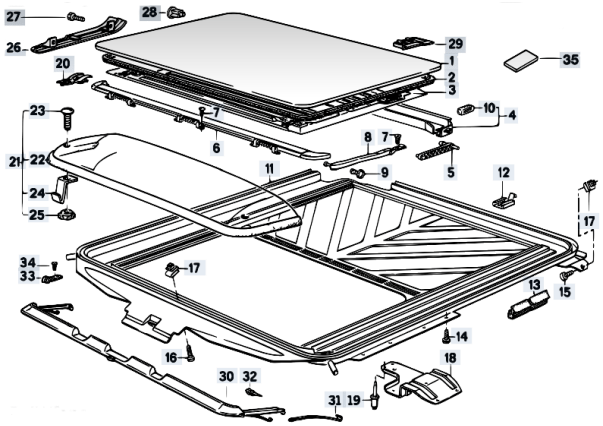
With a carefully selected abstraction level, system specifications can be effectively audited and validated. An unprecedented quality of system specification is achieved. Inconsistencies are detected earlier, resulting in reduced development time and cost.

This approach overcomes the limitations of list-based requirements management and has shown its value in industry projects.

Outline

1. Analysis
2. The Potential of System Modelling
3. Fundamental Modelling Concept
4. Example: Sunroof
5. Why ?





Quotes from everyday‘s life …

- An international producer of construction machinery (2012):

  „Today, stake-holders store important information in distributed, unlinked files. Documents have been repetitively discussed and updated, while others involved in the development don‘t even know they exist.“

  „Product requirements for the vehicle, for subassemblies and components are usually communicated rather informally. Changing requirements is risky because it is unknown who needs to be informed. On several occasions engineers have been working based on requirements which had already changed two months back.“

- A supplier of self-service terminals (2013):

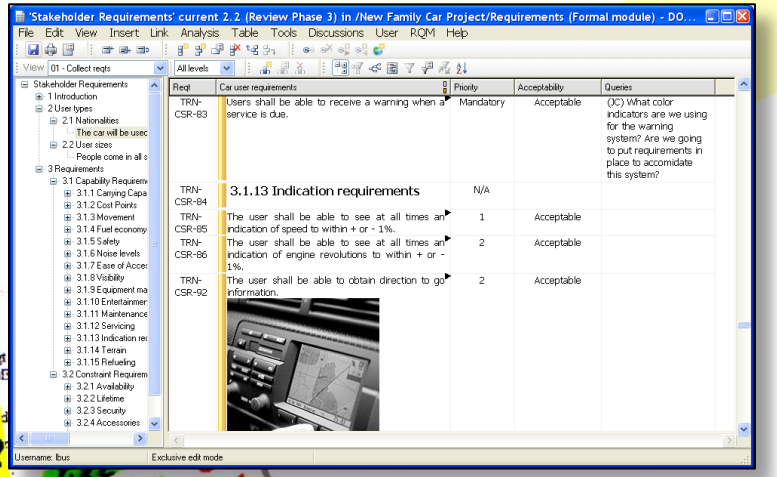
  „Der Definierer schreibt ein Word-Dokument. Die relevanten Teile werden als Anforderungen ins Enterprise Architect (EA) kopiert und dort modelliert. Dann ins Jira kopiert und dort entwickelt. Dann ins Testtool kopiert und dort getestet. Inzwischen wird am Word-Dokument weitergearbeitet. Über Änderungsverfolgung werden die Änderungen im EA nachgezogen. Der Tester wird vergessen zu informieren .... Wo ist nun die aktuelle und gültige Spezifikation zu finden??“

1 Analysis

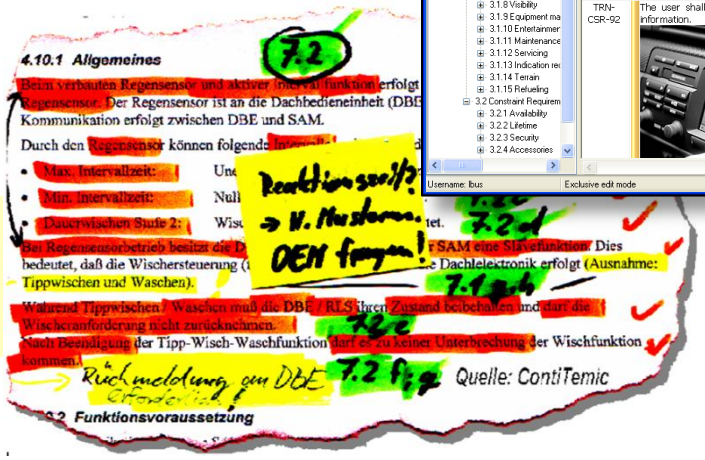
- Requirement lists as such cannot be checked for quality.
- Most system modelling ist by far too detailed and formally too vague to be useful for system conception and specification.

The challenges of list-based requirements management

2007: List-based
Requirement Management



2000: Text Document



?

Looking at a list of >1000 requirements:

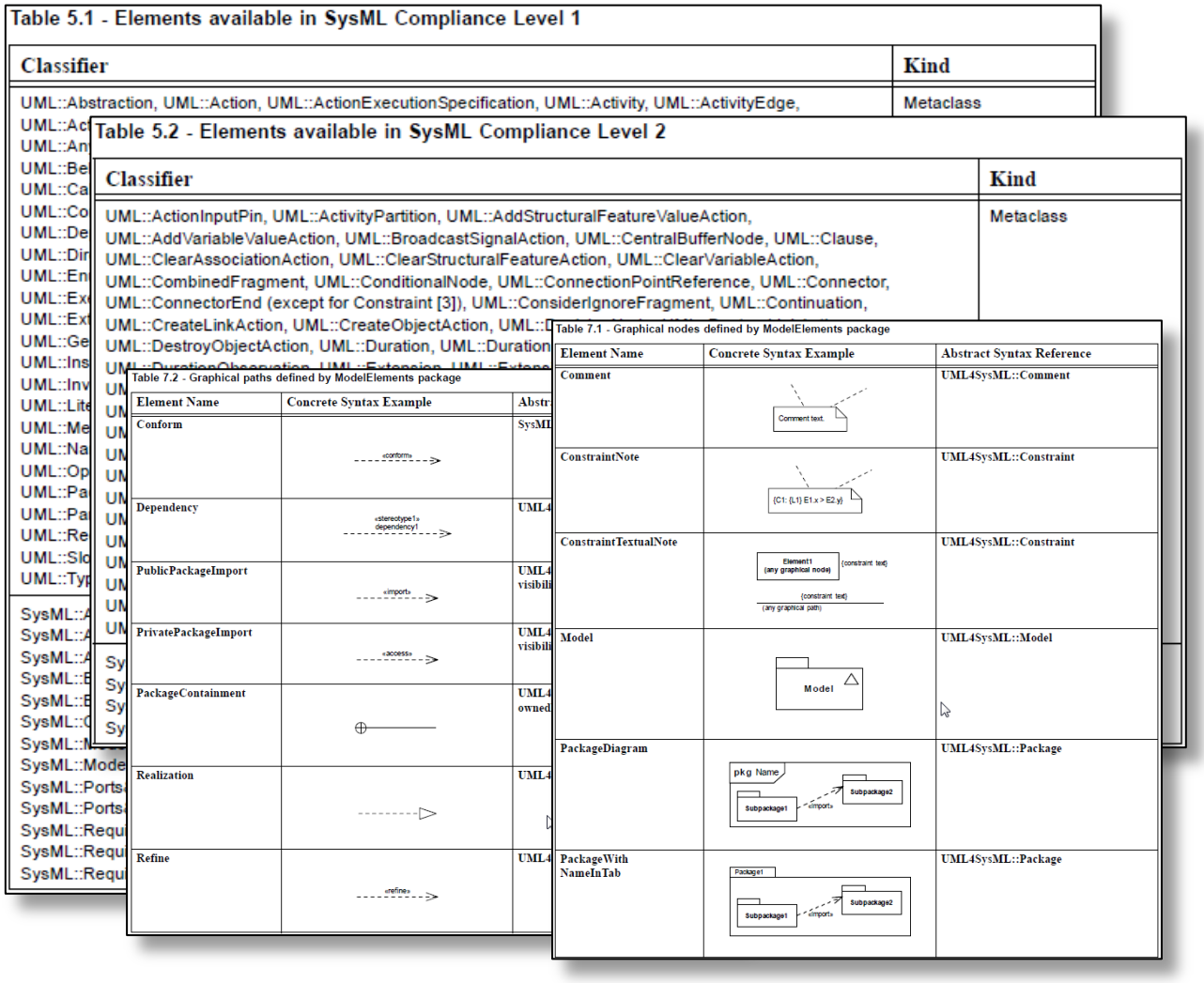
Are they
- Consistent ??
- Complete ??
- Feasible ??

The challenges of system modelling

SysML defines:

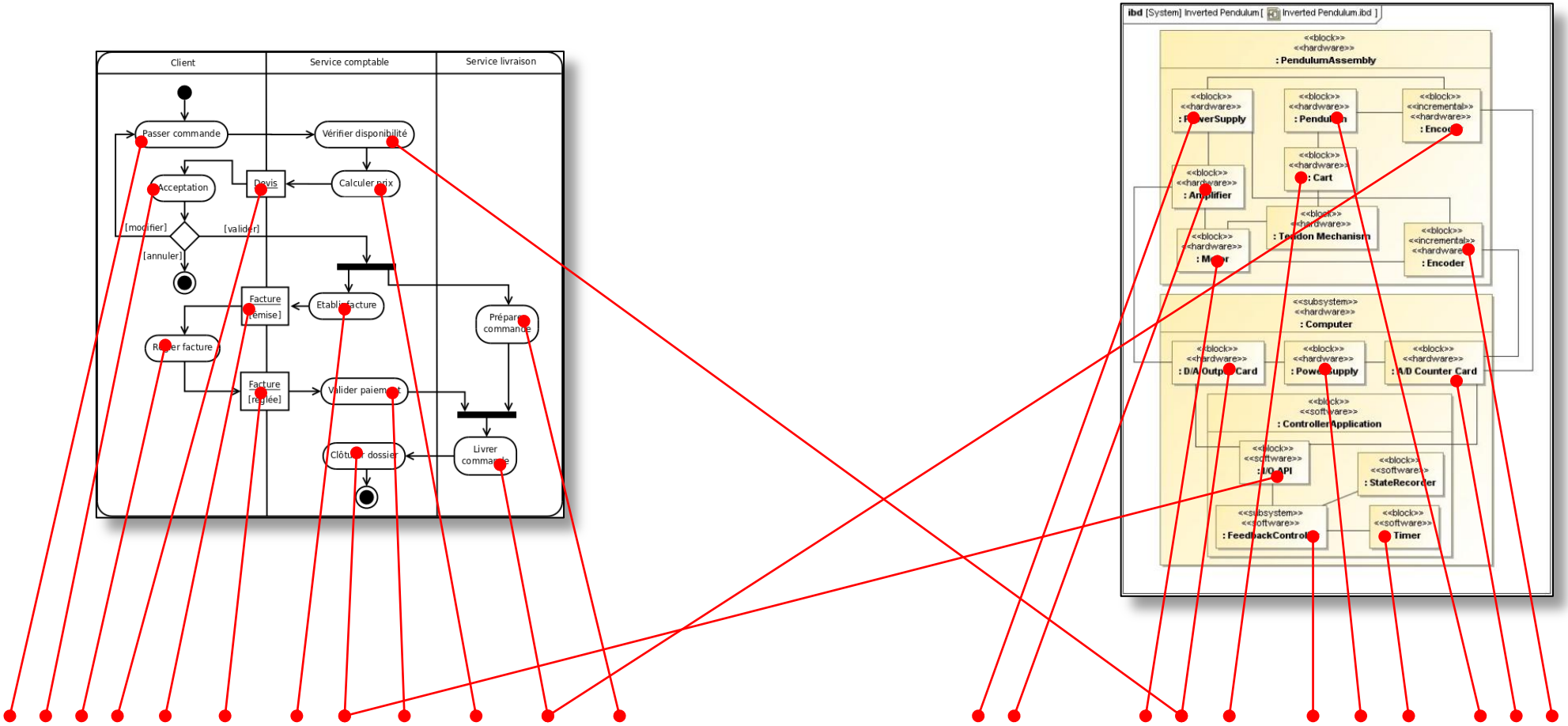
- 163 graphical node types,
- 22 data types,
- 211 metaclasses,
- 25 stereotypes.

The authors state: “The use of more formal constraints and semantics may be applied in future versions to further increase the precision of the language”



Source: OMG Systems Modeling Language (OMG SysML™), Version 1.3, http://www.omg.org/spec/SysML/1.3/

System modelling in practice



… why are there so many ‚open ends‘ ??

2 The Potential of System Modelling

- Interrelate ‚model views' semantically.
- Derive many logical dependencies from model views.
- Check semantics automatically.

→ Improve quality significantly !

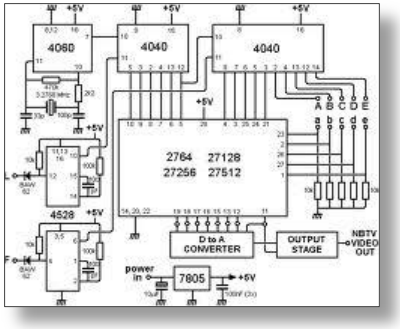
Rationale

Standards and requirements on products are getting more demanding

- SPICE Requirement Management (ENG 1-4) and Test Management (ENG 7-10)
- ISO 26262 Functional Safety in Road Vehicles
- ….

Looking at other domains and industries

- DO-254 in Avionics → „Design Assurance“
- Hardware Design Language (HDL) for Electronics




```
library ieee;                           -- top level circuit
use ieee.std_logic_1164.all;
use work.all;

entity comb_ckt is
port(   input1: in std_logic;
        input2: in std_logic;
        input3: in std_logic;
        output: out std_logic
);
end comb_ckt;

architecture struct of comb_ckt is

    component AND_GATE is               -- as entity of AND_GATE
    port(   A:  in std_logic;
            B:  in std_logic;
            F1: out std_logic
    );
    end component;

    component OR_GATE is                -- as entity of OR_GATE
    port(   X:  in std_logic;
            Y:  in std_logic;
            F2: out std_logic
    );
    end component;

    signal wire: std_logic;             -- signal just like wire
```

Design-rules OK!

- Software Mining/Tomography

Frequent changes!

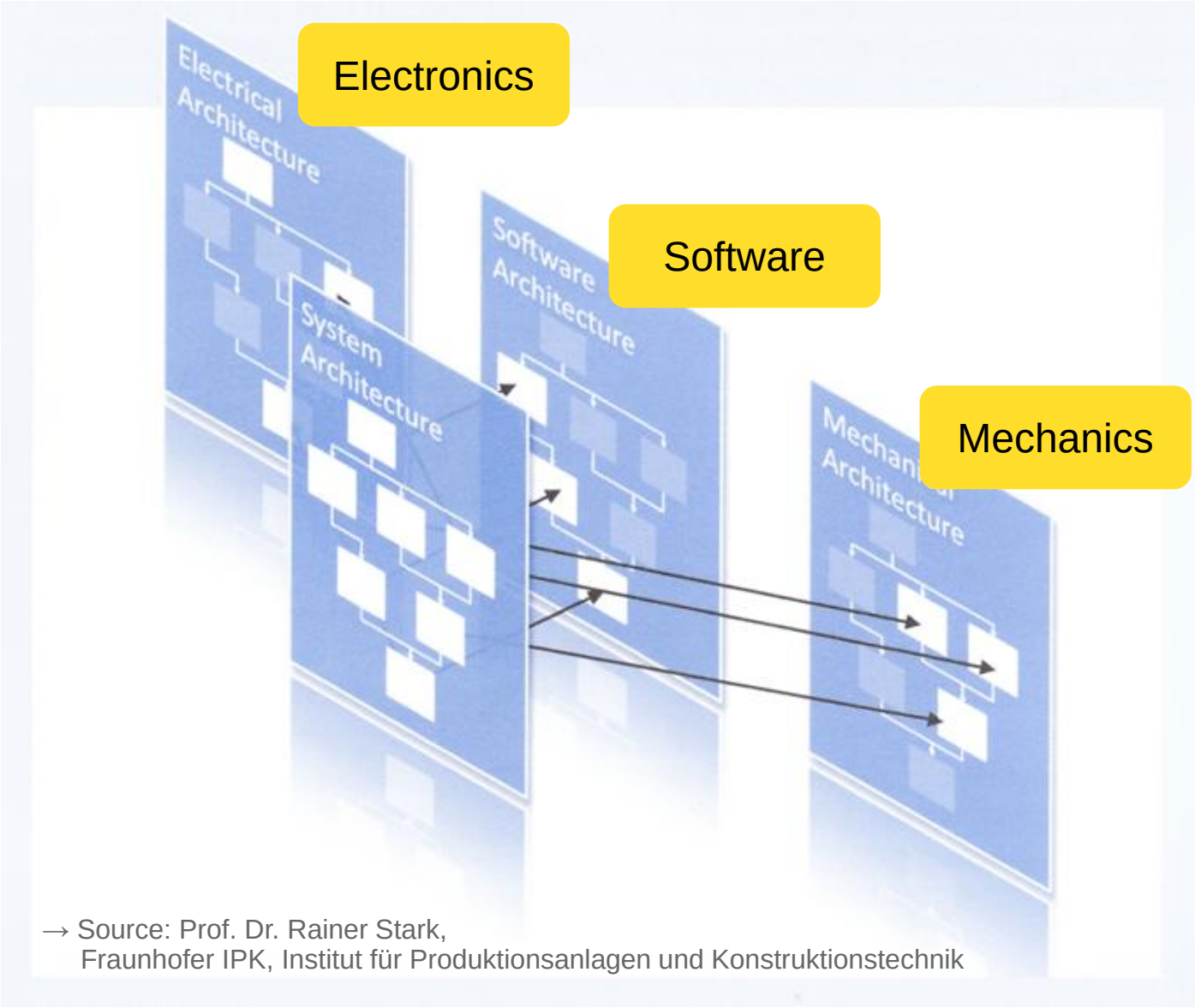
Goal: Common context for mechanics, electronics and software

Topics

- Interfaces between the work products of the disciplines
- Relations between artefacts in different models (tools)

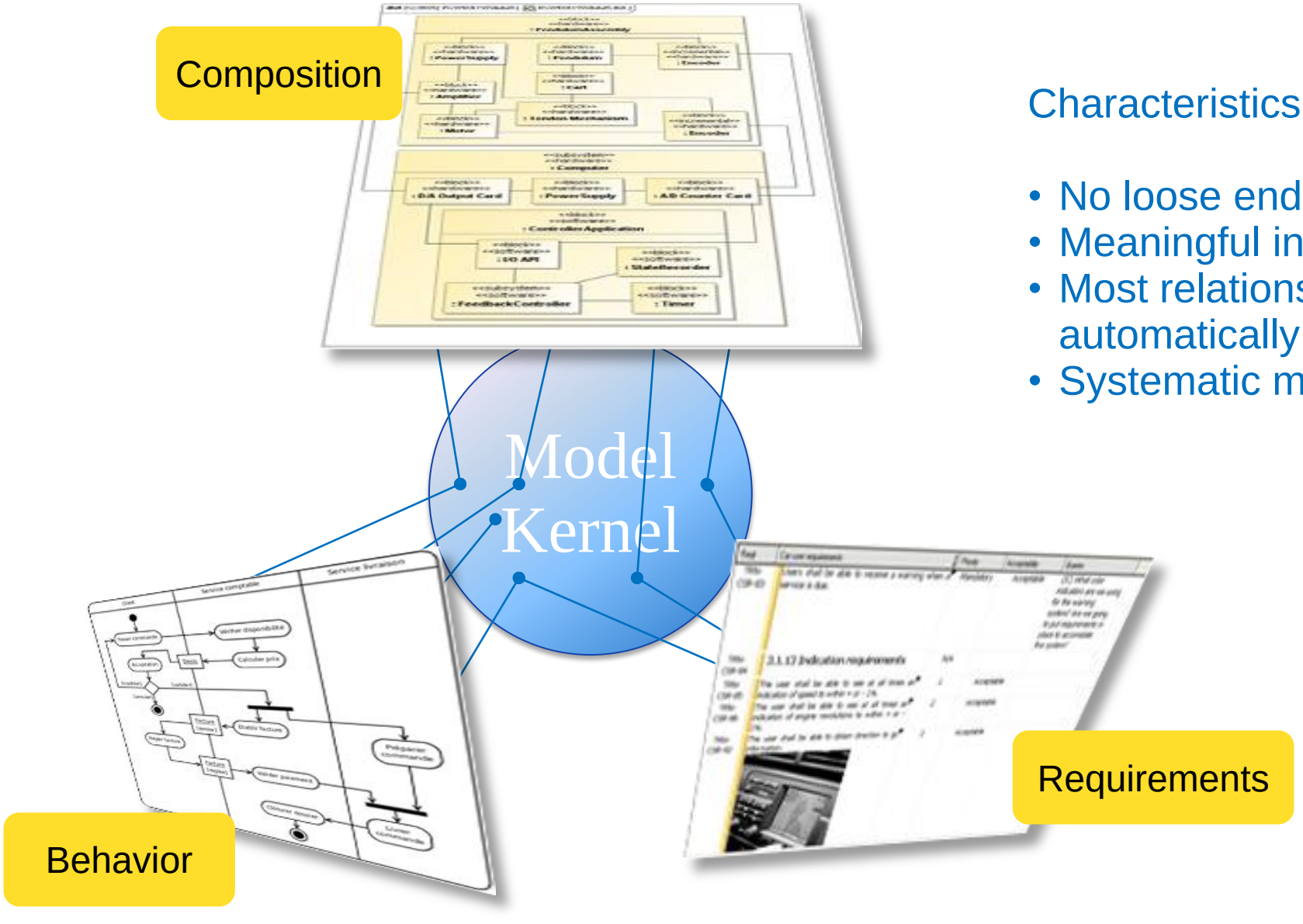
Showcases:

- Machine Tools
- Automotive supplier parts (subsystems)
- Home Automation



→ Source: Prof. Dr. Rainer Stark,
Fraunhofer IPK, Institut für Produktionsanlagen und Konstruktionstechnik

Goal: Diagrams are views of a logical model kernel



Characteristics:

- No loose ends !
- Meaningful in all views !
- Most relations created automatically !
- Systematic model checking !

3 Fundamental Modelling Concept *

- ‚Plans' and just 5 element types: ■ ● ◆ ✶ ϟ
- Some formal rigor improves semantic value and insight
- Shared and related model elements tie a semantic net

* Prof.Dr. Siegfried Wendt, Founding Director of the Hasso-Plattner-Institute, Potsdam

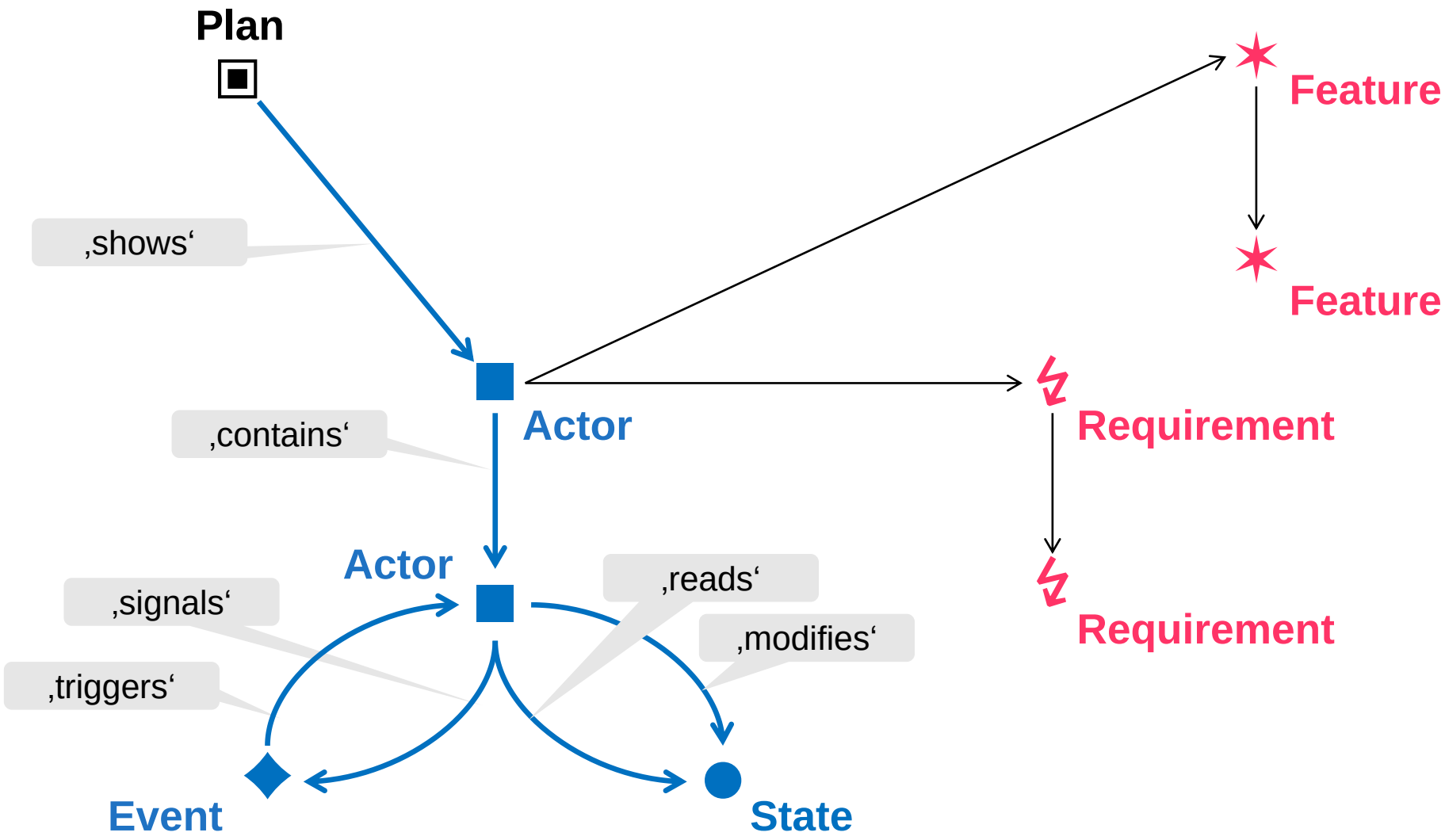
‚Fundamental Modelling‘ offers plans and 5 element types

▣	Plan	A 'Plan' is a **model diagram**, i.e. a view on the system model with a specific communication purpose.
■	Actor	An 'Actor' represents an **active entity**, be it an activity, a process step, a function, a system component or a role.
●	State	A 'State' represents a **passive entity**, be it a value, a condition, an information storage or even a physical shape.
◆	Event	An 'Event' represents a **time reference**, a change in condition/value or more generally a synchronisation primitive.
✶	Feature	A 'Feature' is an intentional distinguishing **characteristic** of a system, often a unique selling proposition.
ϟ	Requirement	A 'Requirement' is a singular documented physical and functional **need** that a particular design, product or process must be able to perform.

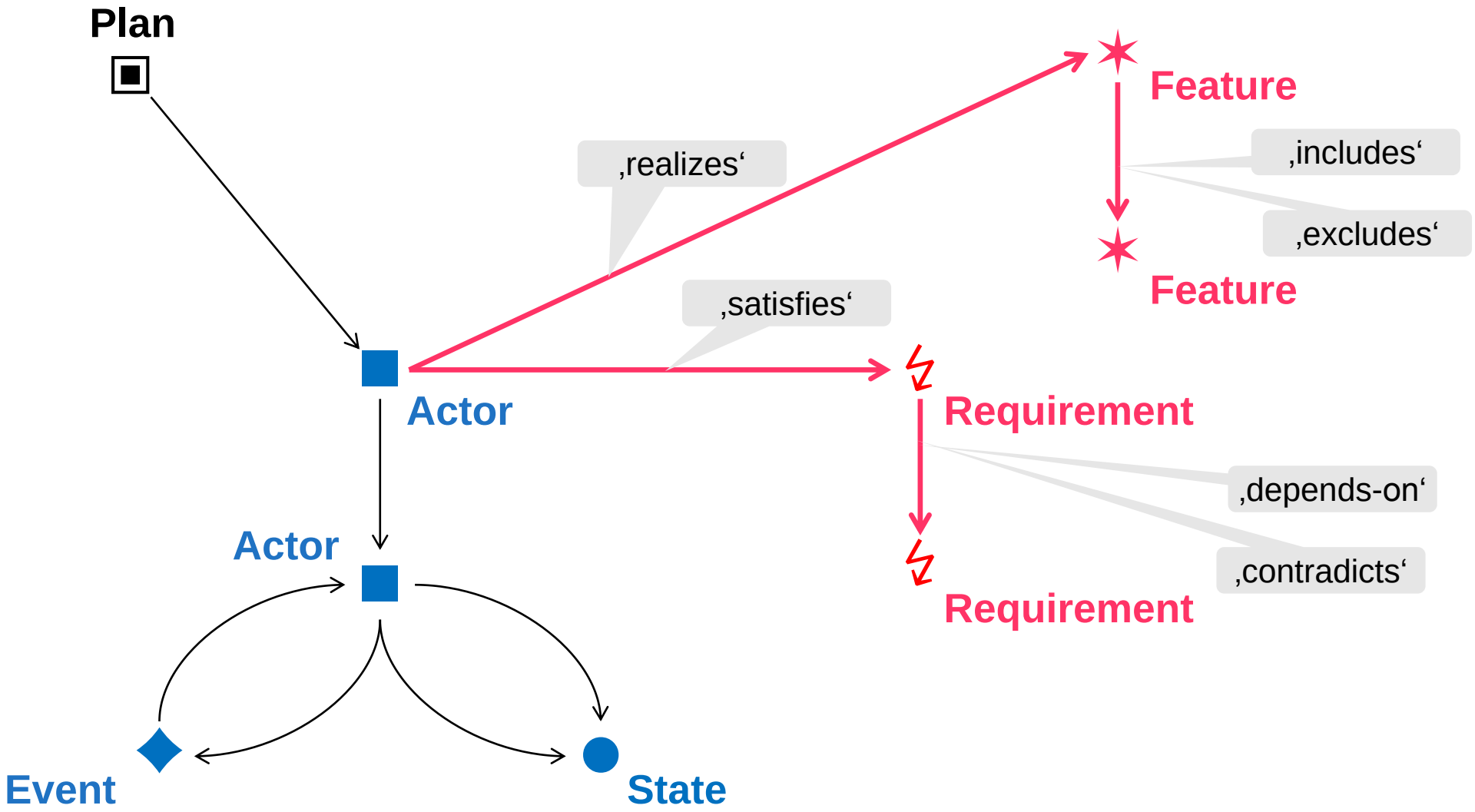
Plans share model elements

▣	Plan	Tree (Outline)	Process	State-Machine	System Composition	Organization Chart	Data Structure
■	Actor	X	X	X	X	X	
●	State	X	(X)	X	X		X
◆	Event	X	X	X			
✶	Feature	X					
↯	Requirement	X					

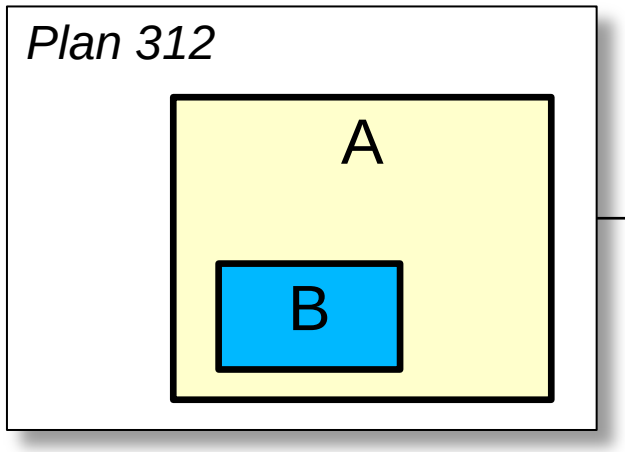
Many semantic relations are derived automatically from plans



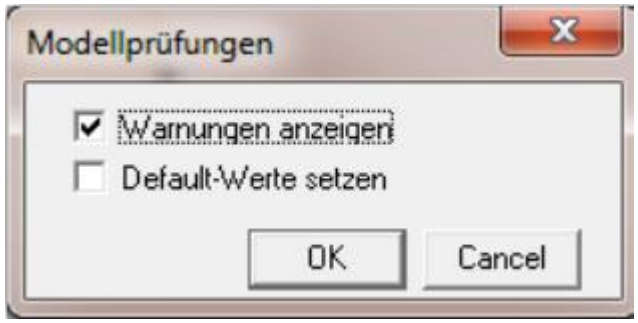
Others are maintained manually



Check models automatically



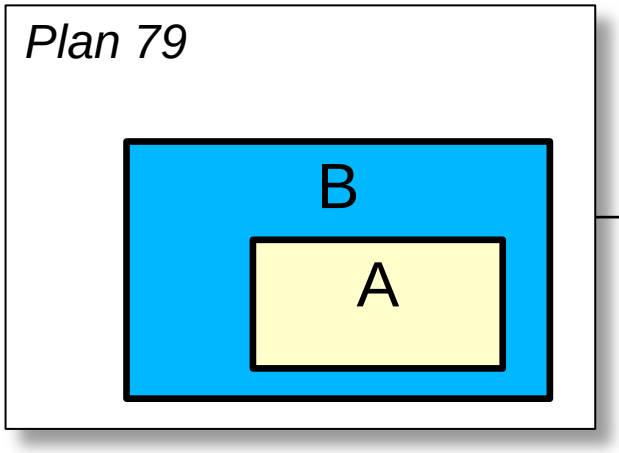
„A contains B“



Inconsistent !

Warnung - View "Projekt" im Package "Projec
Warnung - Package Projectmanagement beg
Warnung in Projectmanagement - Project - A
Warnung in Projectmanagement - Employee
Warnung - Package Datatypes beginnt nicht
ng in Datatypes - ProjectState: Project
ng in Datatypes - Skill - Attribute : subj
ng in Datatypes - KnowledgeLevel - At

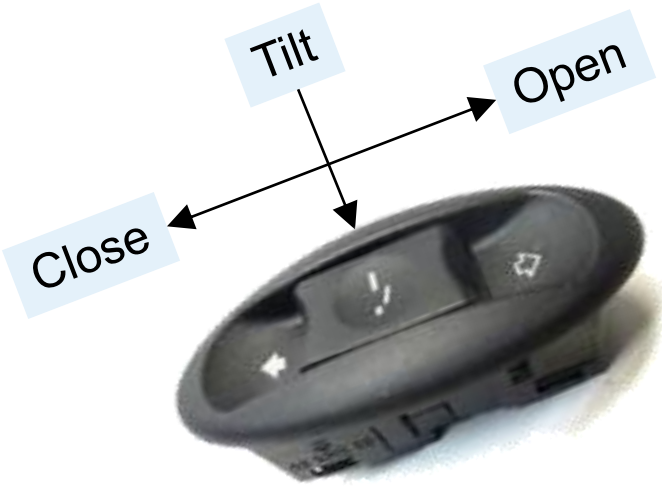
Warnungen:	8
Fehler:	0
Packages:	4
Klassen:	8
Interfaces:	0
Datentypen:	4
Operationen:	12
Abstrakte Operationen:	0
Attribute:	26

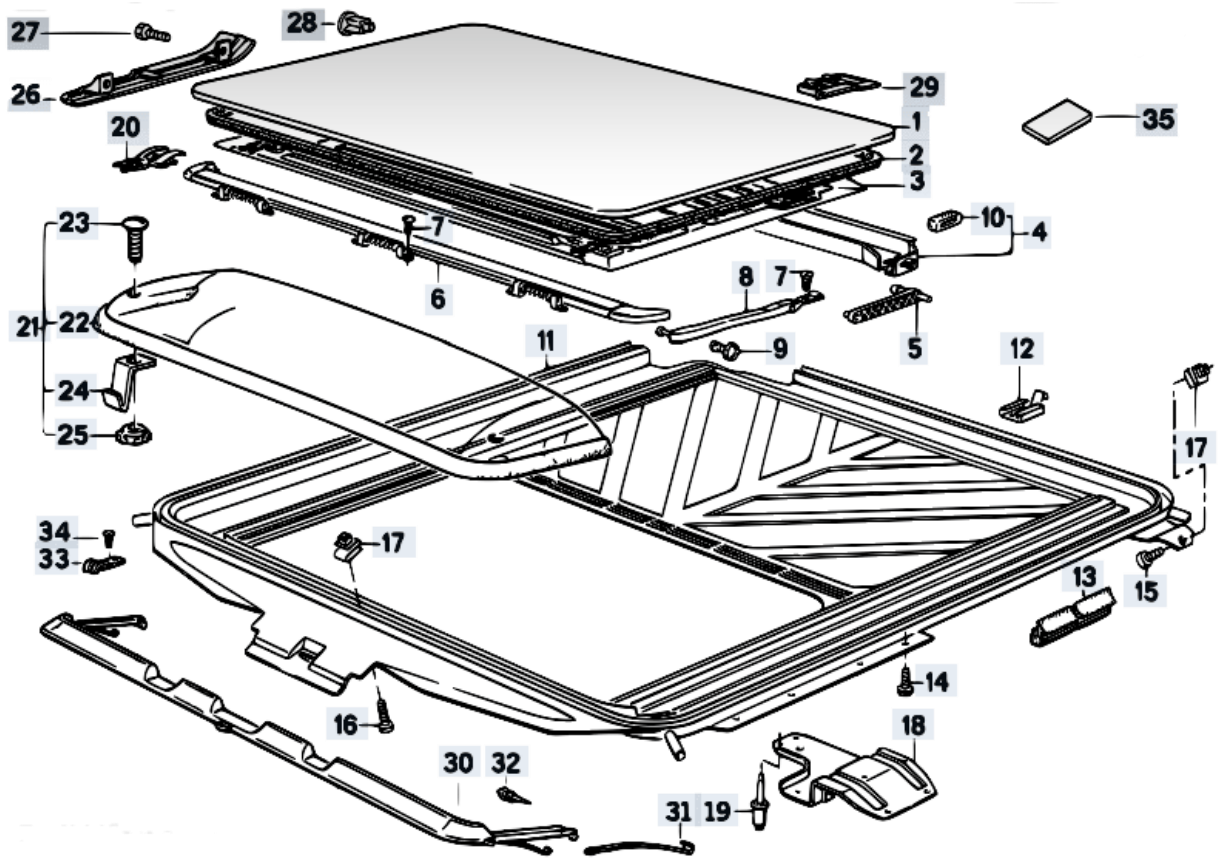


„B contains A“

… and other automated checks!

4 Example: Sunroof

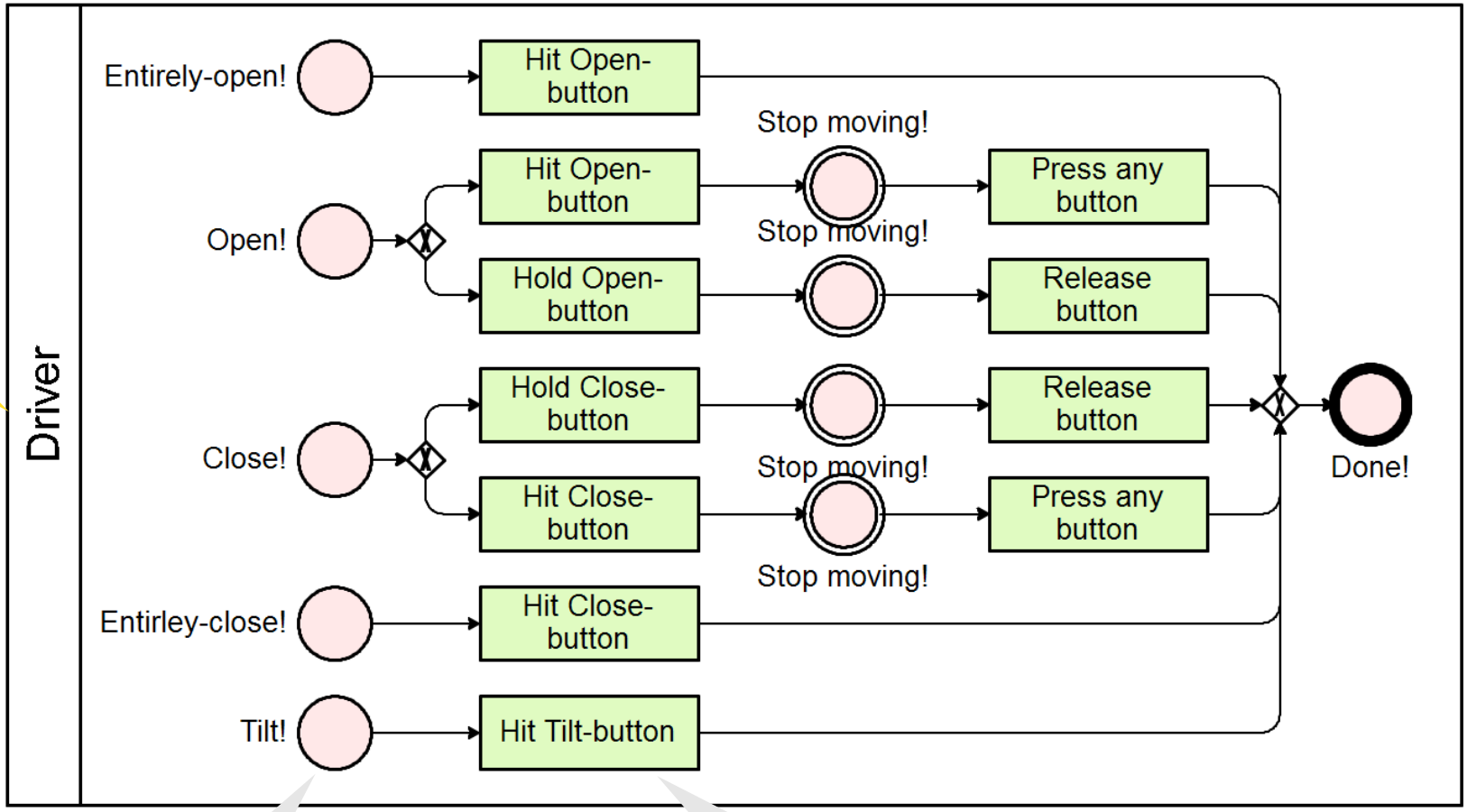




a) Use-cases

▣ Process Diagram

(Notation: BPNM)



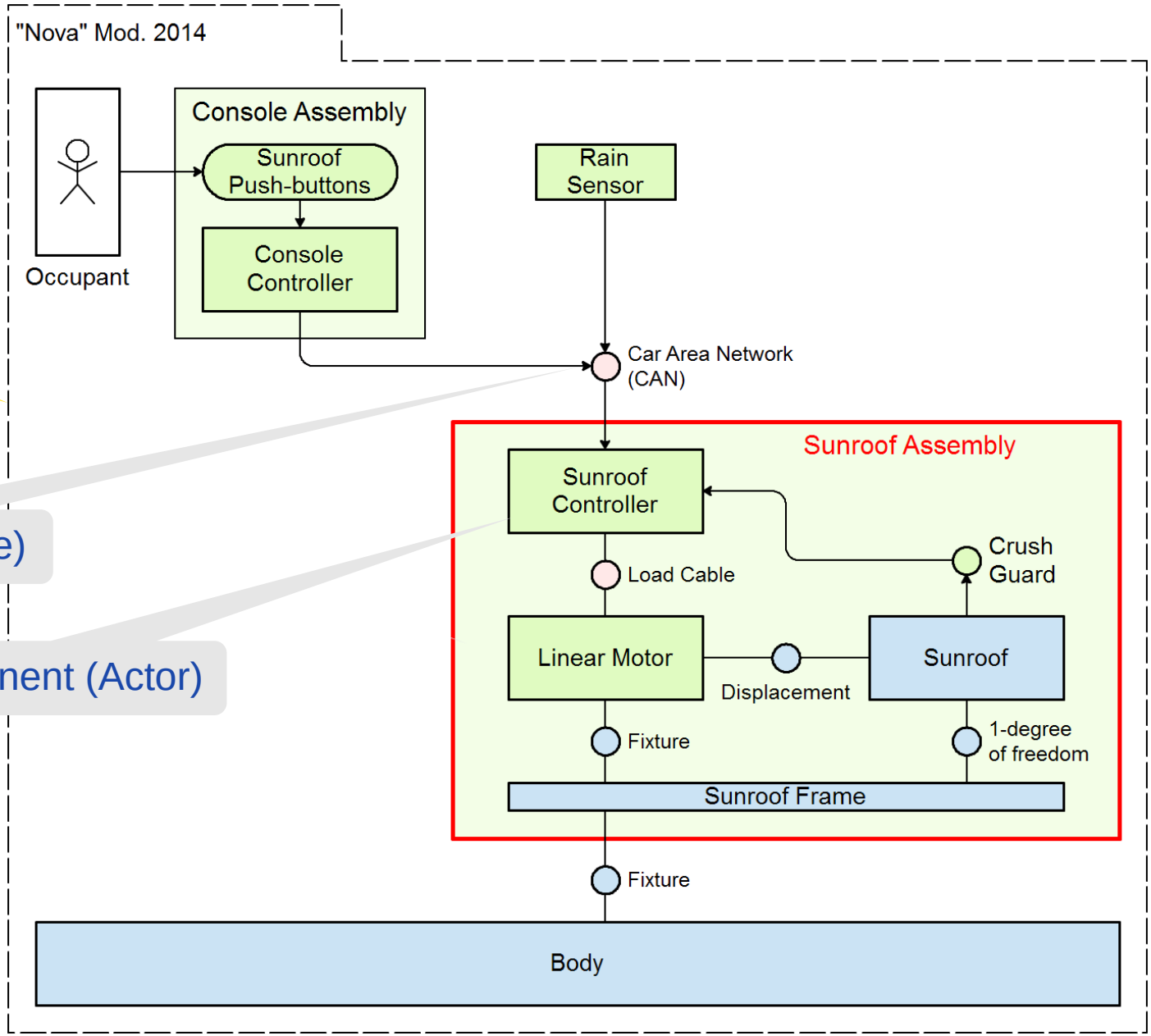
◆ Event

■ Process Step (Actor)

b) System composition

▣ Composition Diagram

(Notation: FMC Block Diagram)



● Information (State)

■ Component (Actor)

Every view serves a communication purpose!

c) Interfaces between mechanics, electronics and software

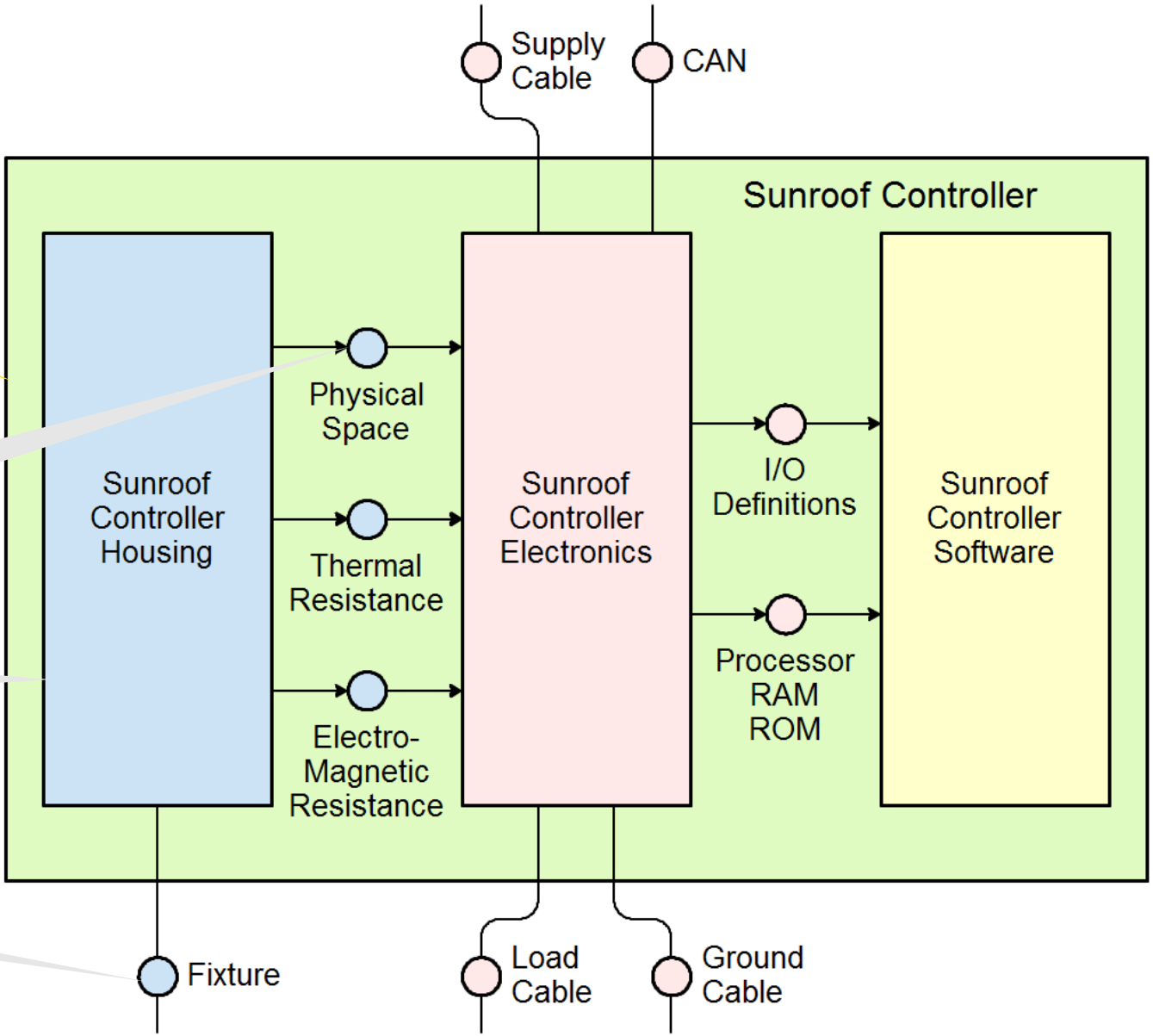
▣ Composition Diagram

(Notation: FMC Block Diagram)

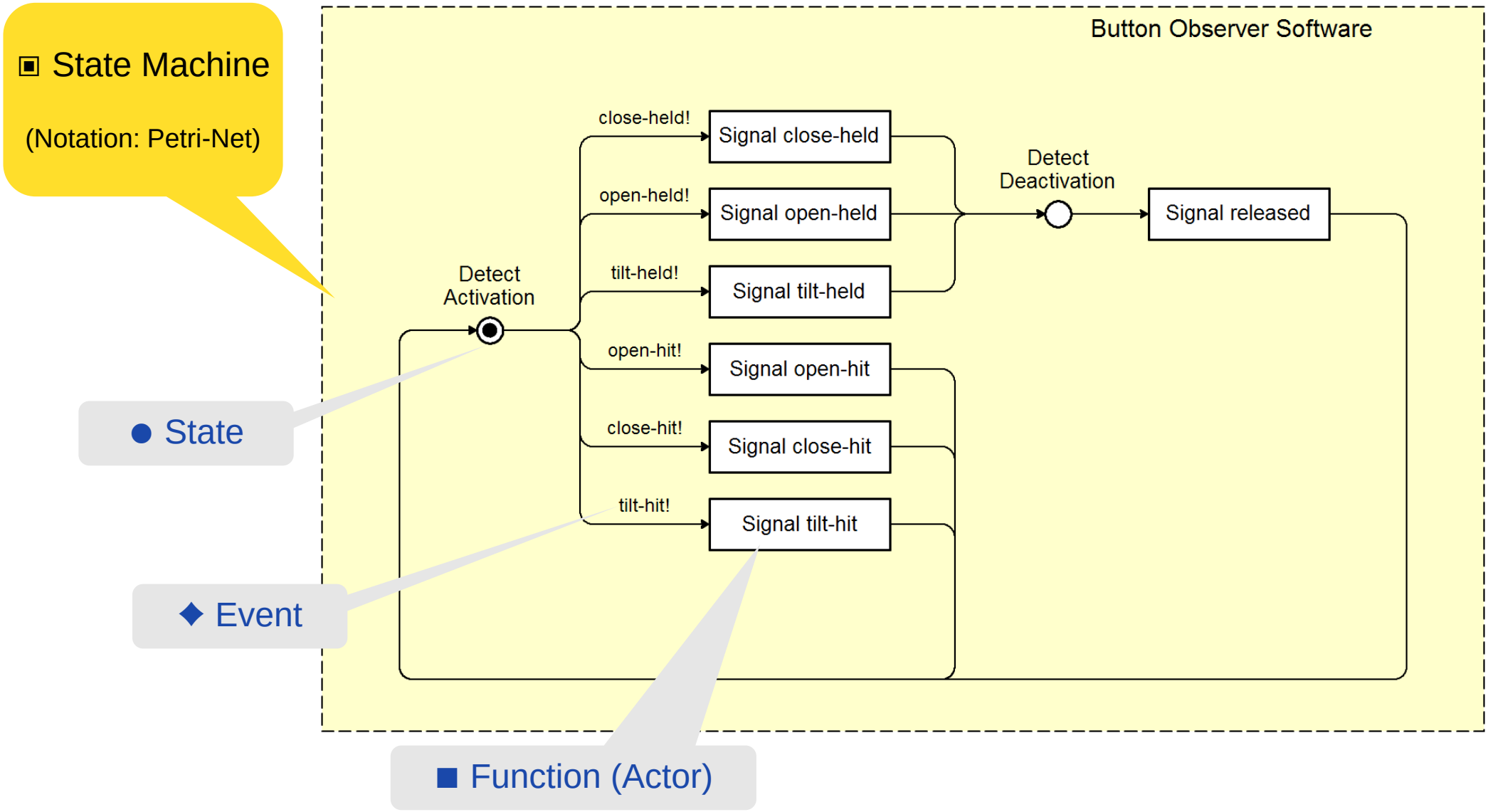
● Information (State)

■ Component (Actor)

● Form (State)



d) Behavior of the controllers, here button observer



e) Behavior of the sunroof controller

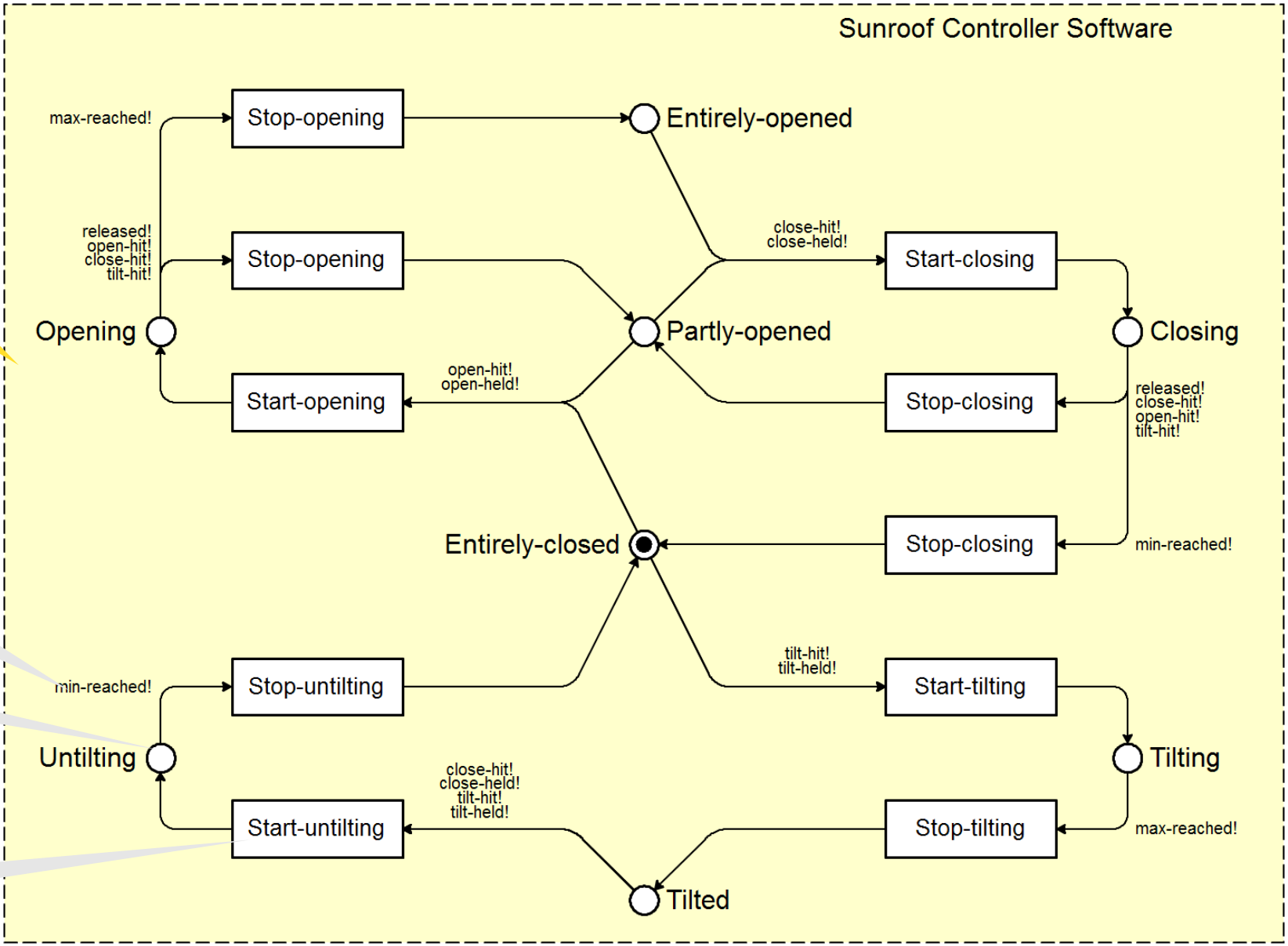
▣ State Machine

(Notation: Petri-Net)

◆ Event

● State

■ Function (Actor)

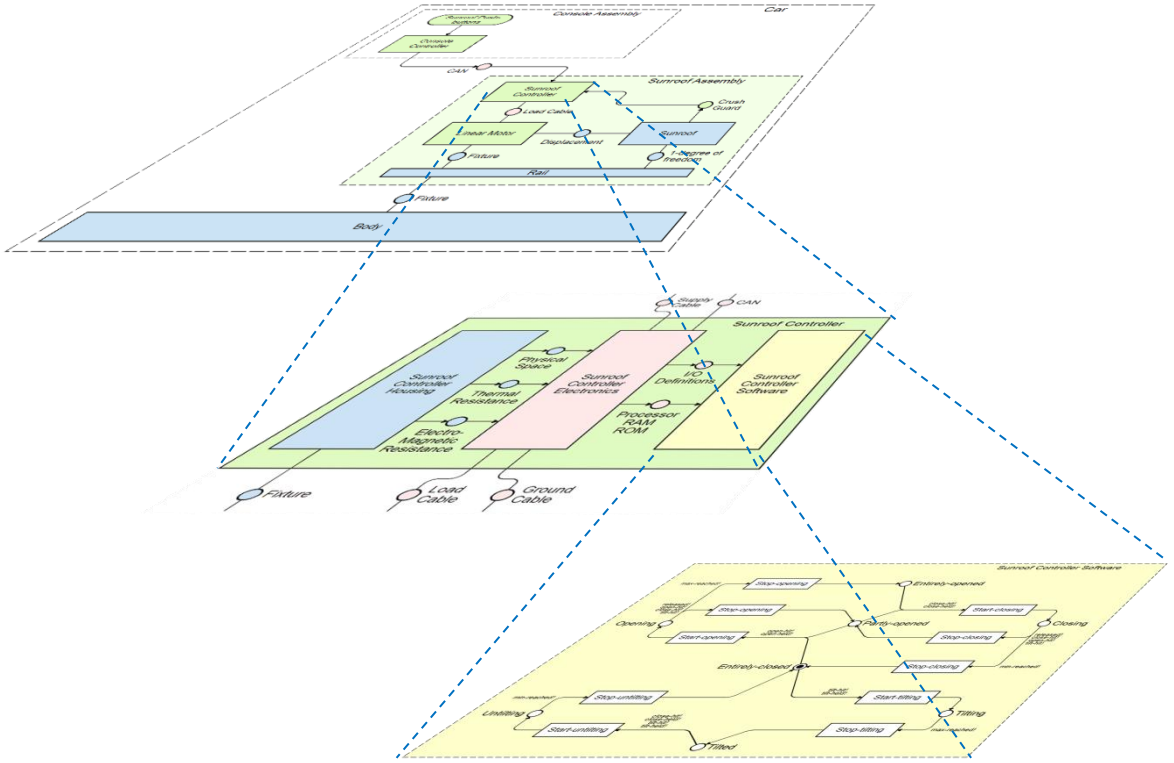


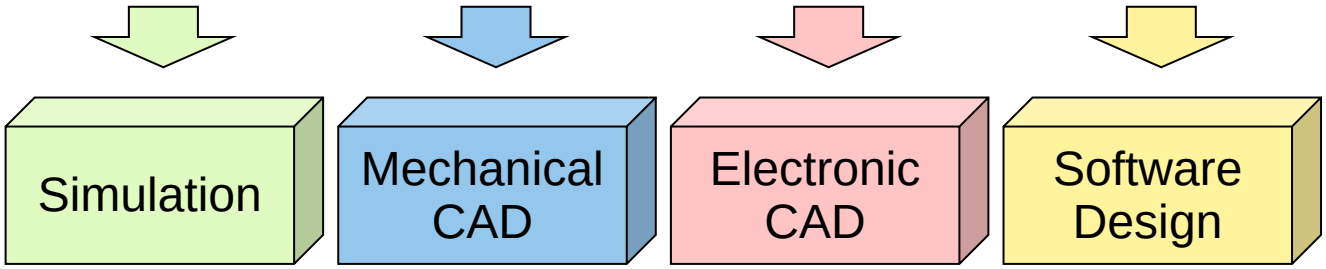
Refine stepwise …

.. and hand over to the domain specific design tools:

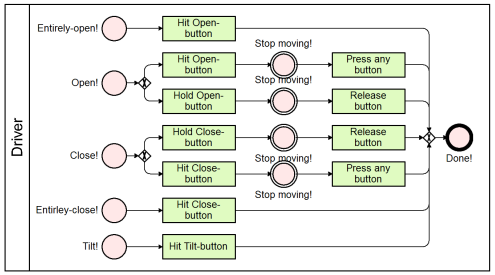
- Mechanical CAD
- Electronic CAD
- Software Design
- Simulation
- …

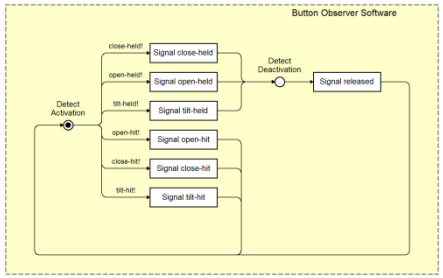
as soon as possible.

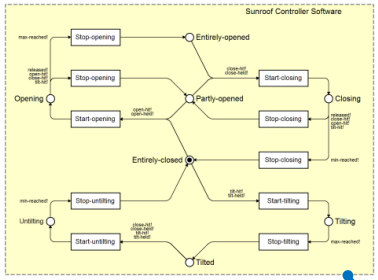


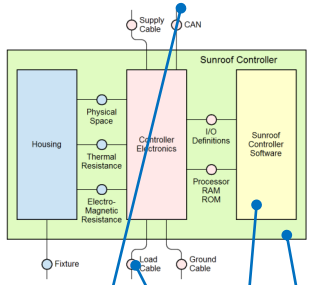


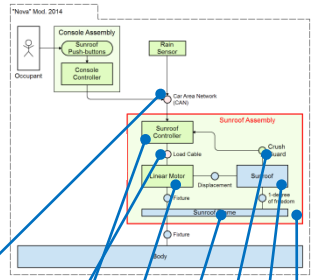
Share model elements







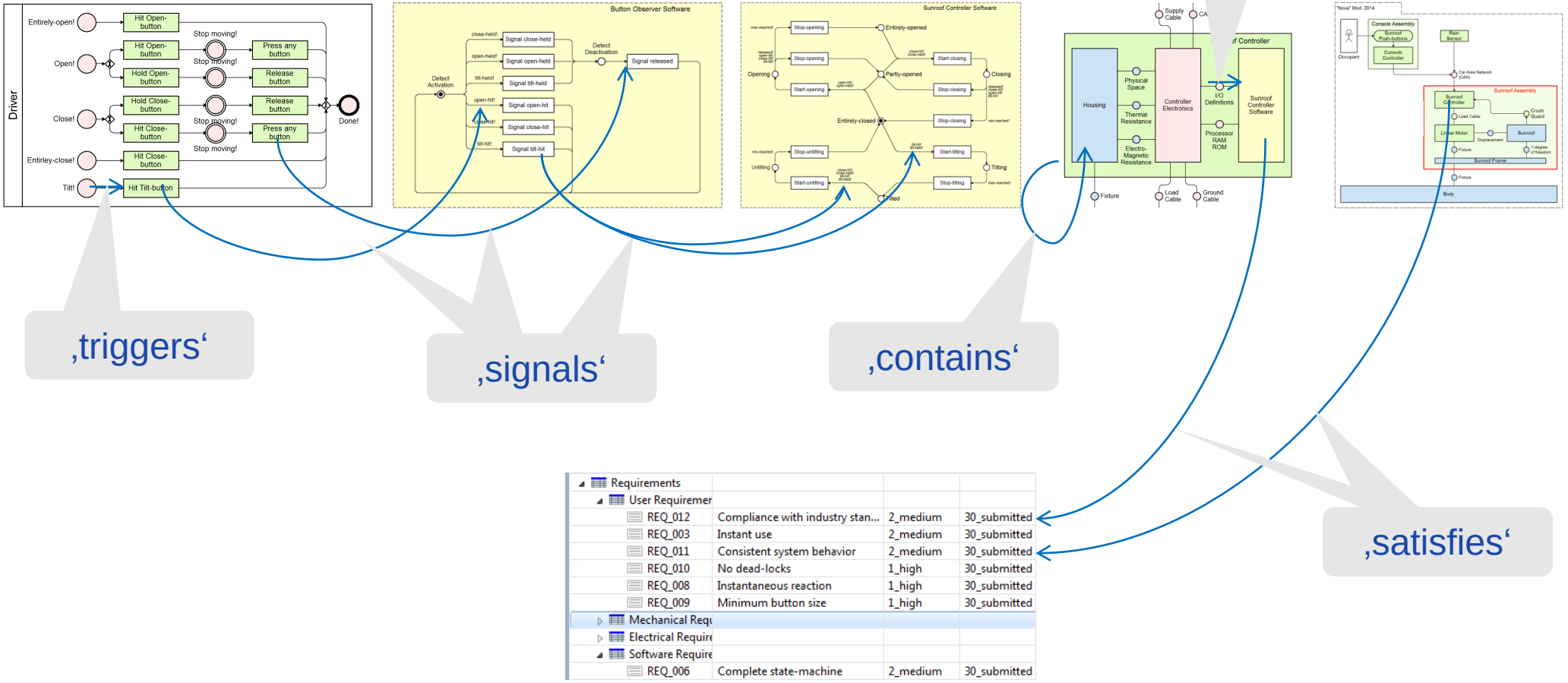




- CAN
- Sunroof Controller Software
- Load Cable
- Sunroof Controller
- Linear Motor
- Sunroof Frame
- Crush Guard
- Sunroof
- Sunroof Assembly

„Parts list“

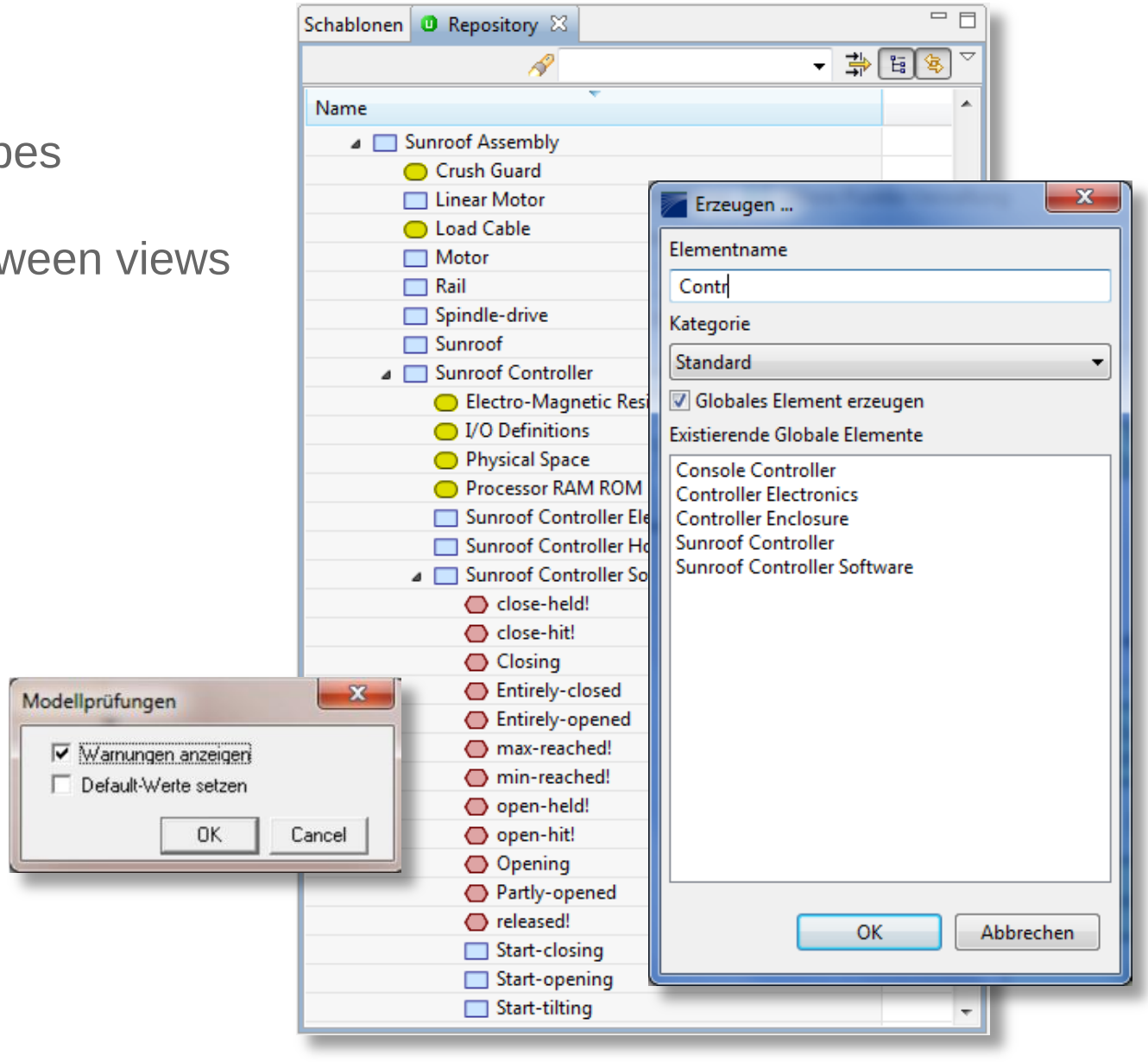
Interrelate model elements



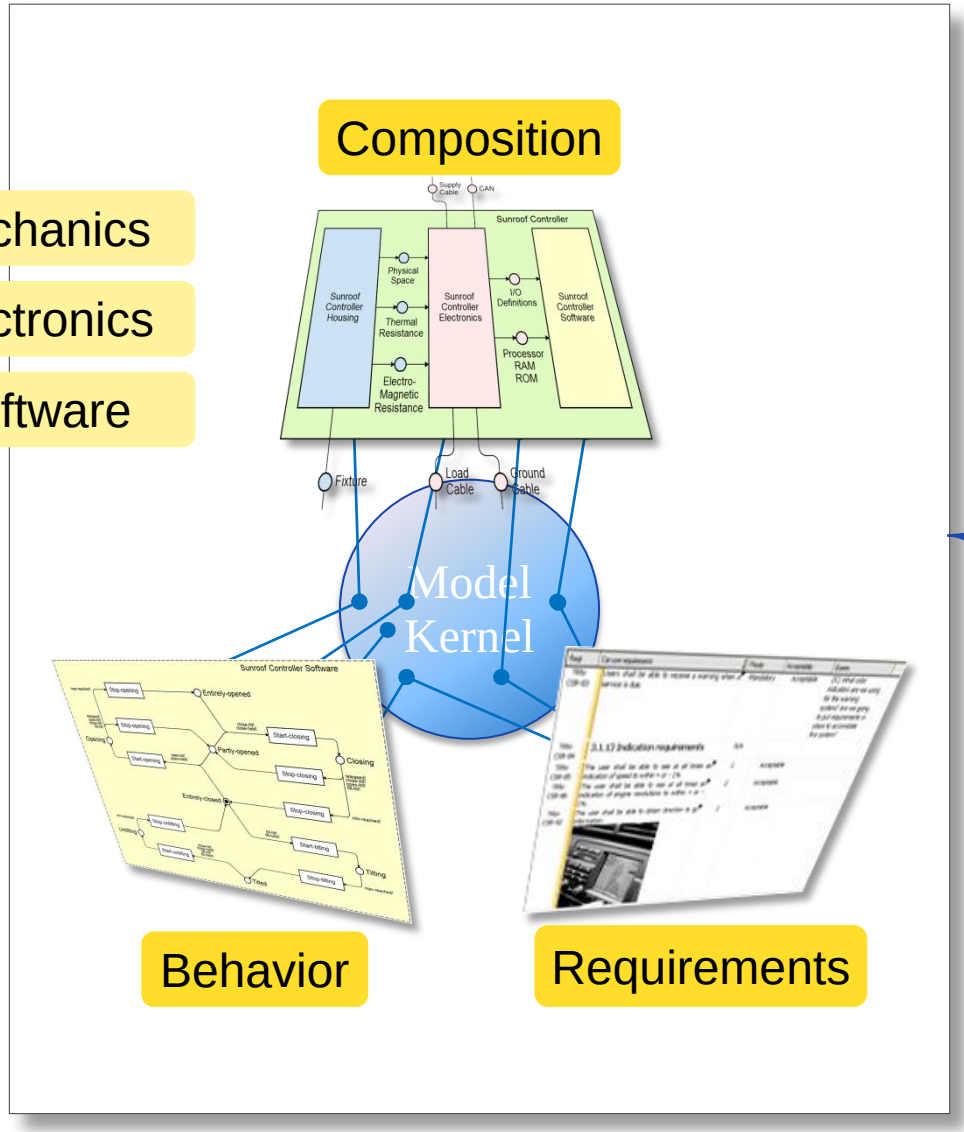
… and many more relationship types!

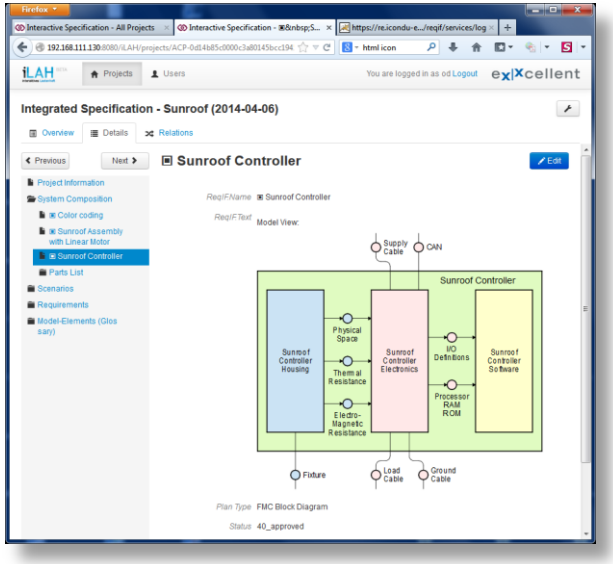
Take advantage of semantic integration

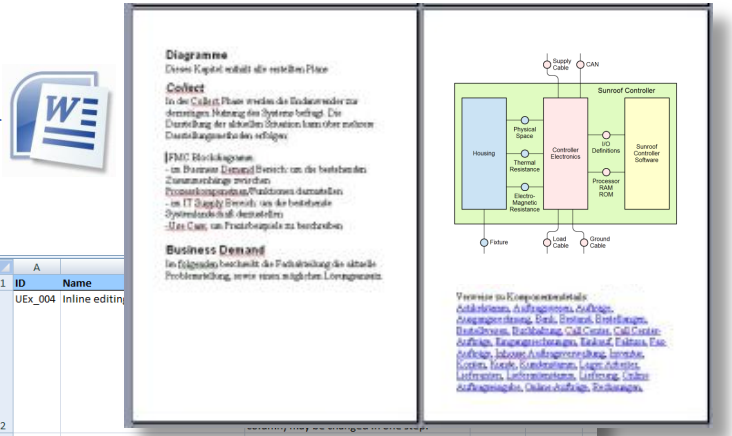
1. Separate views and model
2. Abstract model element types
3. Share Model elements between views
   - a priori: re-use
   - a posteriori: consolidate
4. Interrelate model elements
   - automatically
   - manually
5. Check the model
   - automatically
   - manually
6. Let a system do the repetitive work

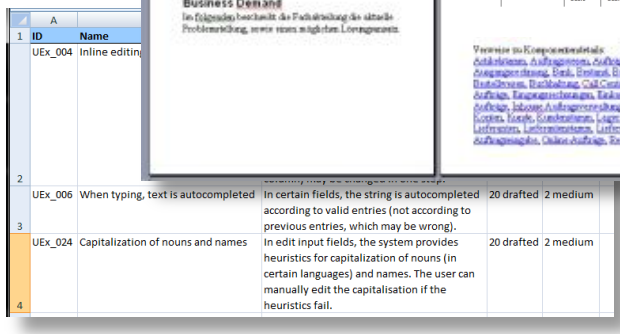


Generate documents, tailored for the audience, and invite for review









ID	Name			
UEx_004	Inline editing			
UEx_006	When typing, text is autocompleted	In certain fields, the string is autocompleted according to valid entries (not according to previous entries, which may be wrong).	20 drafted	2 medium
UEx_024	Capitalization of nouns and names	In edit input fields, the system provides heuristics for capitalization of nouns (in certain languages) and names. The user can manually edit the capitalisation if the heuristics fail.	20 drafted	2 medium

5 Why ?

- Create a common context for mechanics, electronics and software
- Communicate more clearly using a semantically interconnected system model
  - Share model elements with multiple diagrams
  - Harvest logical content from diagrams
  - Collect the bill-of-material automatically etc.
  - Associate features and requirements with the components
  - Use relations for navigation and filtering
- Let the system do the repetitive tasks, e.g. automatic checking
- Explore the semantic net in audits

- Detect inconsistencies earlier → reduce development time and cost
- Improve consistency and viability → quality of the system spec

Build on standards

Automotive SPICE

- Support processes ENG.1-4 (Requirement-Management)
- Provide references for ENG.7-10 (Test-Management)

HIS Agreement Process for Requirements

- 2 Attributes ‚Status‘ and ‚Comment‘ each for OEM and Supplier
- Role-specific access rigths to create, read, update and delete

OMG ReqIF (Requirements Interchange Format)

- Exchange data with established RM-Tools (DOORS, Integrity, …)
- Work with ReqIF-Data – don‘t use it just for transport

International Web-Standards

- (X)HTML, CSS, PNG, SVG, PDF, Unicode, HTTP(S), ….
- Web-Services

Codex of PLM-Openness (ProSTEP iViP e.V.)

- Documented Interfaces, Data Accessibility
- Open for Third-Party Add-Ons and Service

Literature

- Wendt, S.: Ein grundlegender Begriffsrahmen für das Wissensmanagement im Software-Engineering. In Proceedings „Knowtech“ Dresden 2001, www.fmc-modeling.org/download/publications/wendt_2001-grundlegender_begriffsrahmen_fuer_wissensmanagement.pdf.
- Knöpfel, A.; Gröne, B.; Tabeling, P.: Fundamental Modelling Concepts – Effective Communication of IT Systems. ISBN-13: 978-0-470-02710-3. John Wiley & Sons, Chichester, 2005.
- Diverse Papers on the Fundamental Modelling Concept 1995-2014, http://www.fmc-modeling.org/publications.
- Dungern, O.v.: Semantic Model Integration for System Specification - Creating a Common Context for Different Model Types. http://reqif.de/tl_files/ReqIF/resources/documents-en/Semantic-Model-Integration.pdf.
- Object Management Group: OMG Systems Modeling Language (OMG SysML™), Version 1.3, http://www.omg.org/spec/SysML/1.3/, June 2012.
- Screenshots were taken from ARCWAY Cockpit and BitPlan UML Checker.

Is it interesting to You?

Let us exchange ideas!

Kontakt
Dr.-Ing. Oskar v. Dungern
+49 173 670 9958
od@enso-managers.com

Anschrift
enso managers gmbh
Charlottenstrasse 68
10117 Berlin
Deutschland

Information
www.enso-managers.com
www.reqif.de