



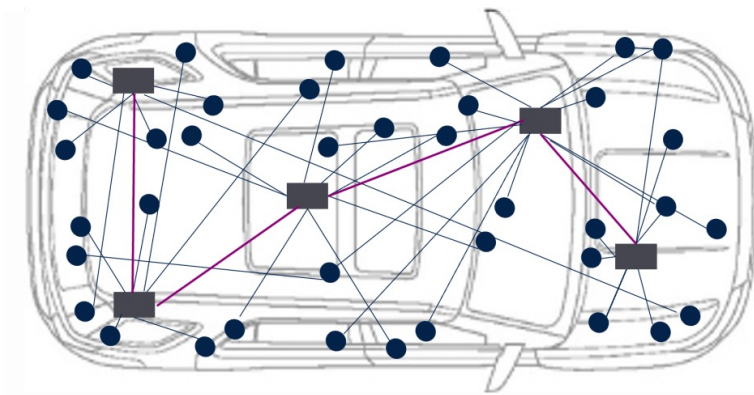
Co-Design of Automotive Boardnet Topology and Architecture

Sebastian Post, Christoph Grimm



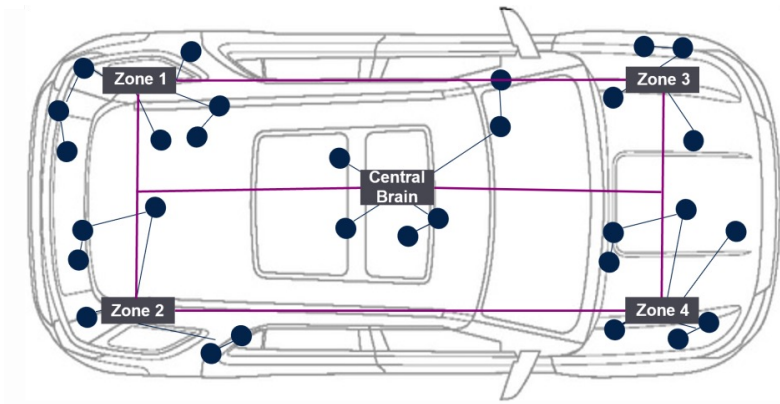
Introduction

- High and growing number of software and AI functions in modern cars
→ connection via boardnet needed



Domain Architecture

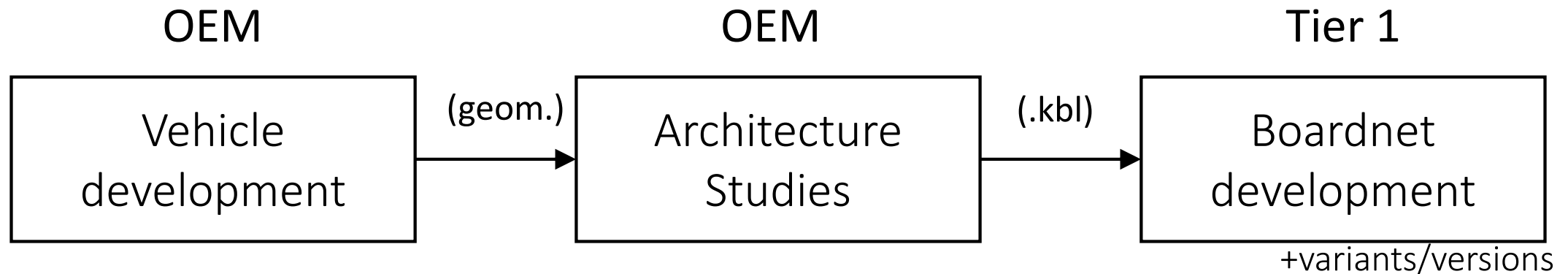
transforms to



Zonal Architecture

- Alternatives: centralized architecture and a mixture of different type

Introduction



- Problems

- Variations and branches of vehicles not considered efficiently
- No Specific solutions for different variants

State of the Art and Related Work

- Analyze and/ or optimize given boardnet architecture [1]
- Focusing on improving collaboration in the value chain [2]
- Do so with standardization or tool interoperability like AutoSAR [3]
- SysML v2 standard with standardized REST API for change management
 - No usage for this purpose known

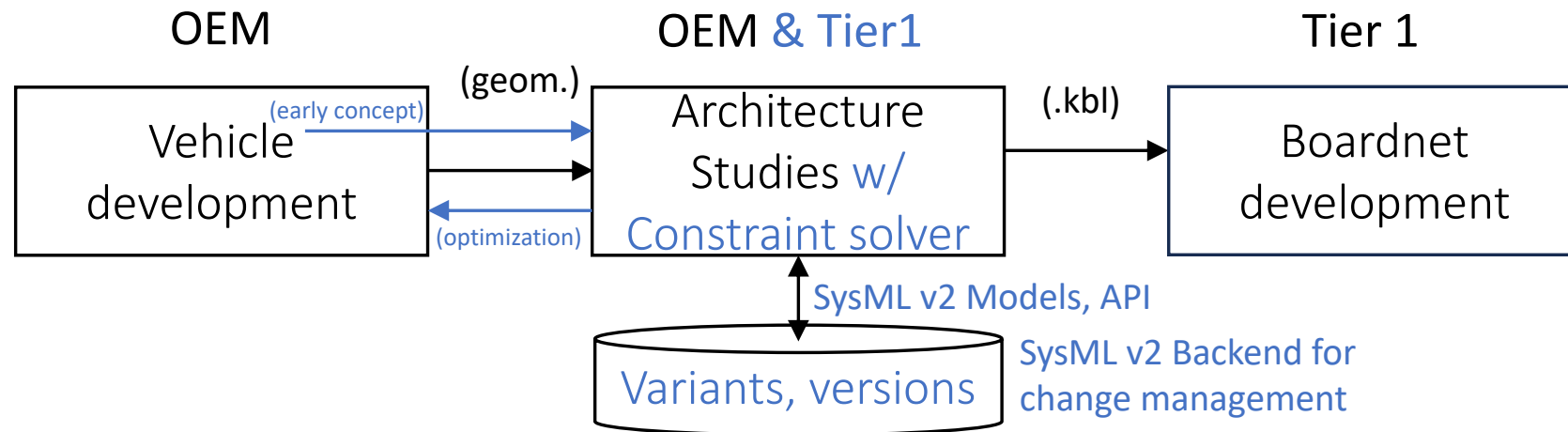
[1] Bowen Zheng, Hengyi Liang, Qi Zhu, Huafeng Yu, and Chung-Wei Lin. Next generation automotive architecture modeling and exploration for autonomous driving. In 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pages 53–58, 2016.

[2] Pelliccione, Knauss, Heldal. Auto- motive architecture framework: The experience of volvo cars. *Journal of Systems Architecture*, 77:83–100, 2017.

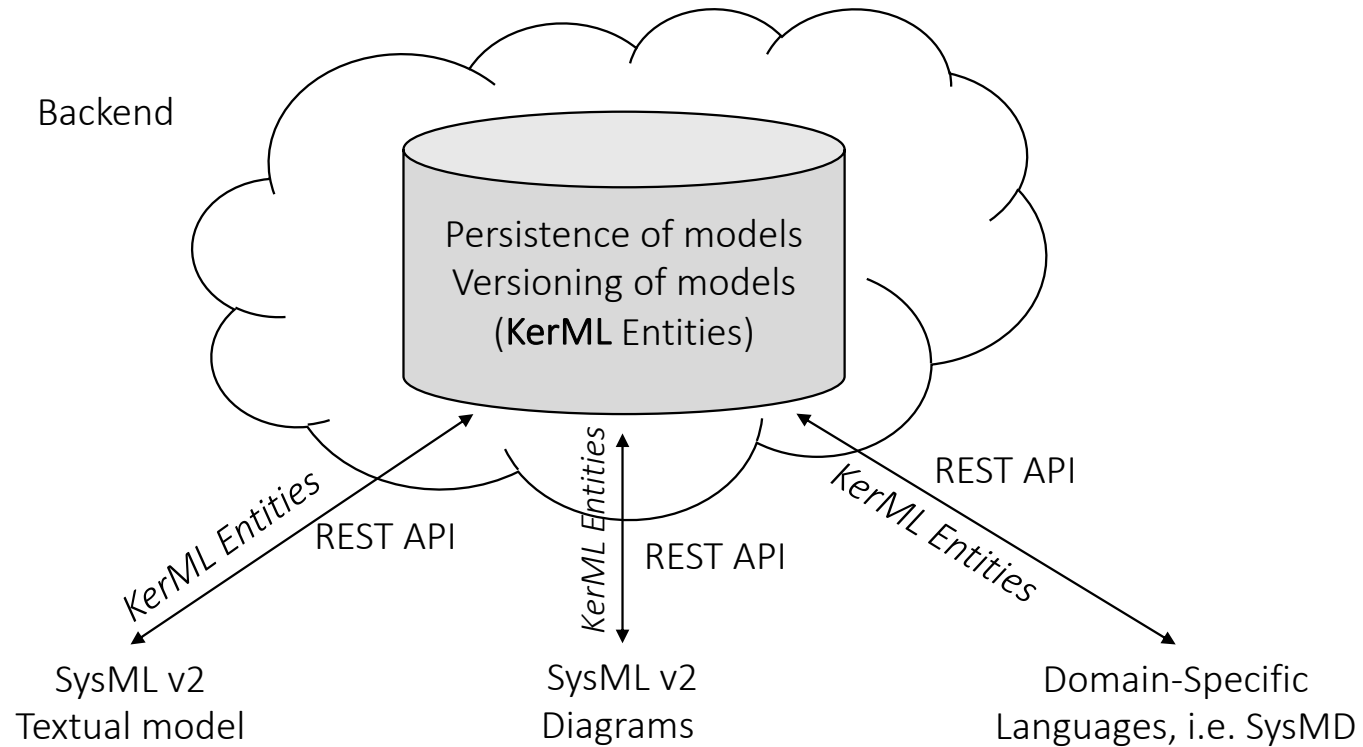
[3] Mozhan Soltani and Eric Knauss. Challenges of requirements engineering in autosar ecosystems. In 2015 IEEE 23rd International Requirements Engineering Conference (RE), pages 294–295, 2015.

Processes

- Target: Cooptimize of vehicle body and boardnet (building spaces, bending radii, ...)



Tool Ecosystem



SysMD Notebook

The screenshot displays the SysMD Notebook application. The main window shows a document titled "SysMD-Kickstart" with the following content:

Note that SysMD Notebook also renders LaTeX equations like $\alpha - \sum_{x=0}^{100} x$.

The feature is still experimental.

The Language SysMD

Dependencies of values

One of the main features of SysMD is that it propagates and checks the consistency of values and units. Below some simple examples!

Example 1: Real values and its dependencies

Below an example for SysMD code. Note, that all values are constraint to some ranges in different units. Also note, that there are dependencies between all the values:

- From the height, width, length to the volume
- From the volume to the height, width, length
- Also in-between height, width, etc.

*To calculate consistent values for all of the above quantities considering the dependency $\text{volume} = \text{height} * \text{width} * \text{length}$, click on the calculator symbol left. To display the values, click on the *i* in a circle left of the cell.*

```
1 Document uses ScalarValues, SI, ISO26262;  
2 import ScalarValues, SI, ISO26262;  
3 PartWithVolume isA Component;  
4 PartWithVolume hasA  
5   Value height: Length(10 .. 100)[cm];  
6   Value width: Length(1 .. 1.1) [m];  
7   Value length: Length(1 .. 1.1) [m];  
8   Value volume: Volume(1000 .. 2000) [l] = height * width * length.
```

The right sidebar shows the "Agenda" with a "Summary: 9 Errors" section. The visible errors are:

- SysMD error** using a project via imports is deprecated. Use 'Document uses ...' [More information here!](#)
- Line: 7, Syntax error** at Requirement: expected [DEF, CLASS, CONNECTOR, CONSTRAINT, ASSERT, ASSOC, ATTRIBUTE, DATATYPE, FEATURE, PART, IMPORT, INTERFACE, PACKAGE, STATE, TRANSITION, ;]
- Line: 24, Syntax error** at Requirement: expected [DEF, CLASS, CONNECTOR, CONSTRAINT, ASSERT, ASSOC, ATTRIBUTE, DATATYPE, FEATURE, PART, IMPORT, INTERFACE, PACKAGE, STATE, TRANSITION, ;]
- Line: 31, Syntax error** at Requirement: expected [DEF, CLASS, CONNECTOR, CONSTRAINT, ASSERT, ASSOC, ATTRIBUTE, ...]

SysMD

- Based on SysML v2 with some extensions and constraint propagation

```
Language: SysMD Package:
1 Package CarParts.
2 CarParts defines
3   Body isA Part;
4   Engine isA Part;
5   Wheel isA Part.
6
7 CarParts::Engine hasA Value mass: Real(200.0) [kg].
8 CarParts::Wheel hasA Value mass: Real(50.0) [kg].
9 CarParts::Body hasA Value mass: Real(100.0) [kg].
10
11 Vehicles::Car hasA
12   Part body: CarParts::Body;
13   Part wheels: [4 .. 4] CarParts::Wheel;
14   Part engine: [1 .. 2] CarParts::Engine;
15   Value mass: SI::Mass (100 .. 2000) [kg] = sumOverParts(mass).
```


Constraint Propagation

- Implemented constraint propagation to all dependent values and expressions
- Supports intervals and units for reals and intervals for integers

```
AreaCalculation isA Component;  
AreaCalculation hasA  
  Value width: Length(1 .. 10) [m];  
  Value length: Length(200 .. 200) [cm];  
  Value area: Area(6 .. 8) [m^2] = width * length.
```

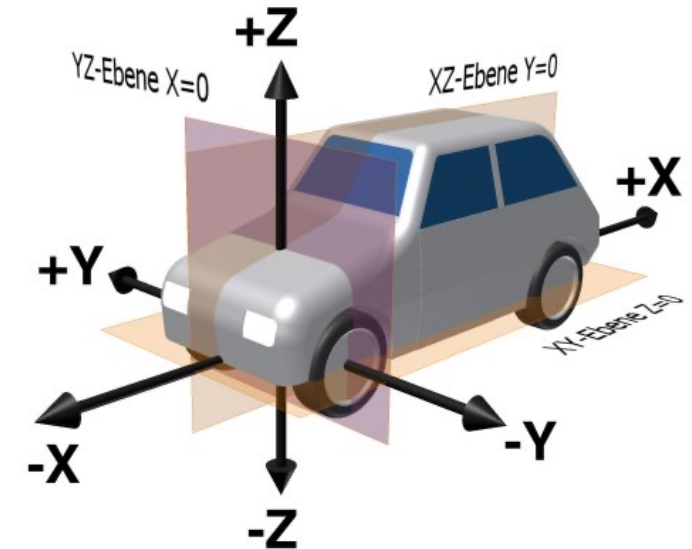
➔ Result: width = 2..3 m

```
Package HybridExample.  
HybridExample hasA  
  Value a: Real(1.0 .. 2.0);  
  Value b: Real(1.1 .. 2.1) = a + 0.1;  
  Value c: Boolean(true) = a > b.
```

➔ Result: not satisfiable

SysMD package for analysis of boardnets

- Analysis of wires with the following aspects:
 - wire length
 - wire costs
 - the data rate of the wire
 - the type of the wire, which influences the latency
- Geometric model of the car:
 - Positions stored as Vectors
e.g. (3.0,- 0.5, 2.0) m



Libraries

```
SysMD Package: Car::CarComponents
CarComponent hasA
  Value position: Length(-1.5..6, -1.25..1.25,-0.5..4.0) [m],
  Value plugCosts: Currency(1..2) [€].

Gateway hasA
  Value position: Length(0..6, -0.5..0.5,-0.5..2.0) [m],
  Value plugCosts: Currency(1..2) [€] = 1.0 [€].

CentralController hasA
  Value position: Length(0..6, -0.5..0.5,-0.5..2.0) [m],
  Value plugCosts: Currency(1..2) [€] = 1.0 [€].
```

CarComponents Library

```
SysMD Package: Car::WireTypes
Wiretype hasA
  Value specificWeight: Quantity(1..100) [g/m],
  Value CostsPerMeter: Quantity(0.0..1.0) [€/m],
  Value DataRate: Quantity(0.001 .. 1000.0) [MB/s],
  Value DataSizePerPackage: Quantity(0..10000) [B],
  Value OverheadPerPackage: Quantity(0..1000) [B].

Ethernet hasA
  Value specificWeight: Quantity(15..25) [g/m],
  Value CostsPerMeter: Quantity [€/m] = 0.8 [€/m],
  Value DataRate: Quantity [MB/s] = 10.0 [MB/s],
  Value DataSizePerPackage: Quantity(42..1500) [B],
  Value OverheadPerPackage: Quantity(26..30) [B].
```

WireTypes Library including Protocols for wires

Estimation of performances

- Estimation of wire length:

$$\text{wireLength} = \text{CityBlockDistance}(\text{start.position}, \text{end.position}) * \text{factor}$$

- Estimation of the latency of a wire:

$$\text{latency} = \frac{\text{dataPerFrame} + \text{overheadPerFrame}}{\text{dataRate}}$$

- Total Length of all wires

$$\text{totalLength} = \sum_{i=1}^N i.\text{wireLength}$$

Estimation of performances

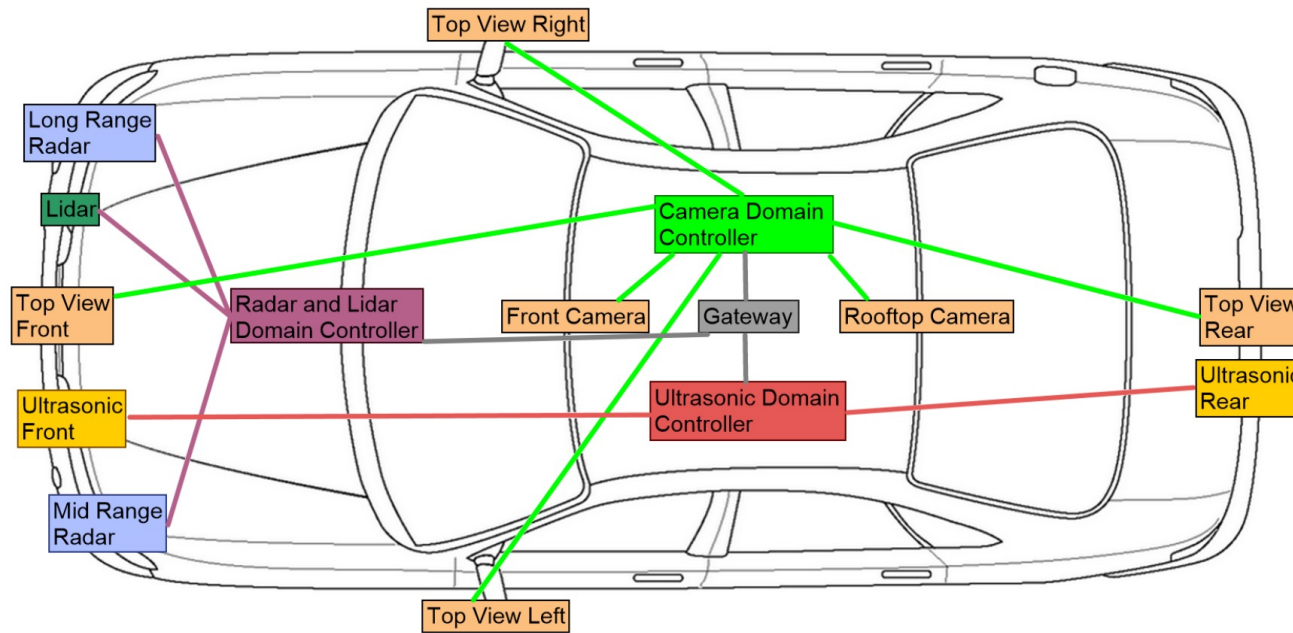
- total weight of all wires:

$$totalWeight = \sum_{i=1}^N i.WireType.specificWeight * i.wireLength$$

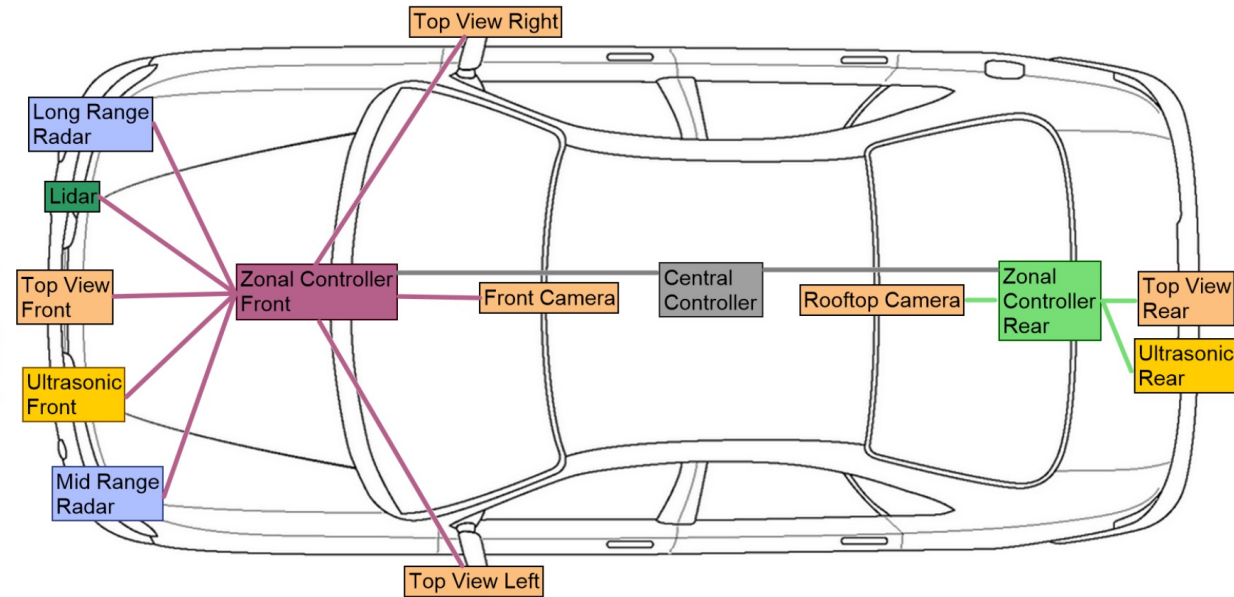
- Estimation of the total Costs of a wire:

$$totalCosts = \sum_{i=1}^N (i.WireType.costsPerMeter * i.wireLength + i.Start.plugCosts + i.End.plugCosts)$$

Case Study: Analysis of Boardnet Architectures



Domain Architecture



Zonal Architecture

Case Study: Definition of the Architecture

- Define cables as connections between two **CarComponents** with a **WireType**
- Define zonal and domain Architecture with all the cables
- Apply the calculations for the estimations of performances
 - For the sum, the *sumOverParts* function of SysMD is used
- Define the Architecture as a summary of all architectures
 - To summarize all values, the SysMD *bySubclasses* function can be used

Case Study: Modelling of variants

```
Language: SysMD   Package: Car
1 DomainArchitecture hasA
2   Value TotalLength: Length [m] = sumOverParts(wireLength),
3   Value TotalWeight: Mass [kg] = sumOverParts(Wiretyp::specificWeight * wireLength),
4   Value TotalCosts: Currency [€] = sumOverParts(Wiretyp::CostsPerMeter * wireLength)
5                                     + sumOverParts(Start::plugCosts + End::plugCosts),
6   Part wireCameraControllerGateway: WireCameraControllerGateway,
7   Part wireUltrasonicControllerGateway: WireUltrasonicControllerGateway,
8   Part wireRadarAndLidarControllerGateway: WireRadarAndLidarControllerGateway,
9   Part wireMidRangeRadarRadarAndLidarController: [0..1] WireMidRangeRadarRadarAndLidarController,
```

- Use [0..1] as the multiplicity of optional components

Conclusion and Outlook

- Advantages of our approach:
 - Uncertain values by using ranges
 - Modeling variants
 - Missing values or open design decisions are possible
 - Analyzing complex architectures
 - Get results in a sufficient time (1 sec for our model)
 - Share results with others across the value chain
- Look at other aspects like safety and reliability
- Support of SysML v2 textual with SysMD extension

Questions