Inclusive Modeling with SysMD

C. Grimm, S. Post, A. Ratzke, C. Zivkovic, A. Khushnood

S. Dalecke, H. Heermann, F. Wawrzik, J. Martin

TU Kaiserslautern, Chair of Cyber-Physical Systems

eniall!

ARROWHEAD TOOLS

AGILA/SysMD has been developed with support from the German BMBF and the European Commission within the projects GENIAL and ArrowheadTools.

Inclusive Modeling with SysMD

SysMD

1. "Inclusive" systems engineering
2. SysMD Notebook & SysMD language
3. Development environment and software details
4. Roadmap for SysMD - *How can I contribute?*

Why “systems engineering”?
Why “inclusiveness”?

SysMD

First known complex project reported by literature [Genesis 11:1–9] is the tower of Bable:

> *“… let’s confuse their language, so that they may not understand one another's speech. … and they left off building the city.”*

Lesson learned: *successful system development* requires

1. **understanding** of people from different disciplines; they clearly use different languages.
2. **motivation** to use and invest in a “common language”.

Inclusive Modeling with SysMD

Modeling and analysis of requirements, specification, knowledge

- Inclusive modeling = we want to allow *everybody* in a development team to
    - document his knowledge and needs,
    - read a specification and requirements documents,
    - maintain documents & models.
- Motivate everybody by additional values beyond "documentation"
    - Consistency checking, from left (requirements, development) to right (operation),
    - AI based recommendations & queries,
    - Links with simulation, operation.

Related work

- Markdown [Aaron Schwatz, John Gruber: http://www.aaronsw.com/weblog/001189]
  - Document software, i.e. GitHub
  - Jupyter Notebook, Matlab Notebook – Describe, Code, Execute approach
- DOORS [IBM]
  - Document, tracking requirements, manage of changes.
- OWL [https://www.w3.org/TR/owl-features/, https://www.w3.org/TR/turtle/]
  - Model knowledge; ~between natural and formal languages
- SysML [OMG]
  - Draw diagrams, comment/documentation model
- SysMLv2 [OMG, https://github.com/Systems-Modeling/SysML-v2-Release]
  - Textual language SysMLv2, interoperability via REST API, Metamodel

Related work

- Markdown [Aaron Schwatz, John Gruber: http://www.aaronsw.com/weblog/001189]
  - Document software, i.e. GitHub
  - Jupyter Notebook, Matlab Notebook – describe, code, execute approach
- DOORS [IBM]
  - Document, tracking requirements, manage of changes.
- OWL [https://www.w3.org/TR/owl-features/, https://www.w3.org/TR/turtle/]
  - Model knowledge; ~between natural and formal languages
- SysML [OMG]
  - Draw diagrams, comment/documentation model
- SysMLv2 [OMG, https://github.com/Systems-Modeling/SysML-v2-Release]
  - Textual language SysMLv2, interoperability via REST API, Metamodel

SysMD

1) **First:** Describe, explain
2) **Then:** Model
3) **Continuously:** Check, update

Inclusive modeling with SysMD

SysMD

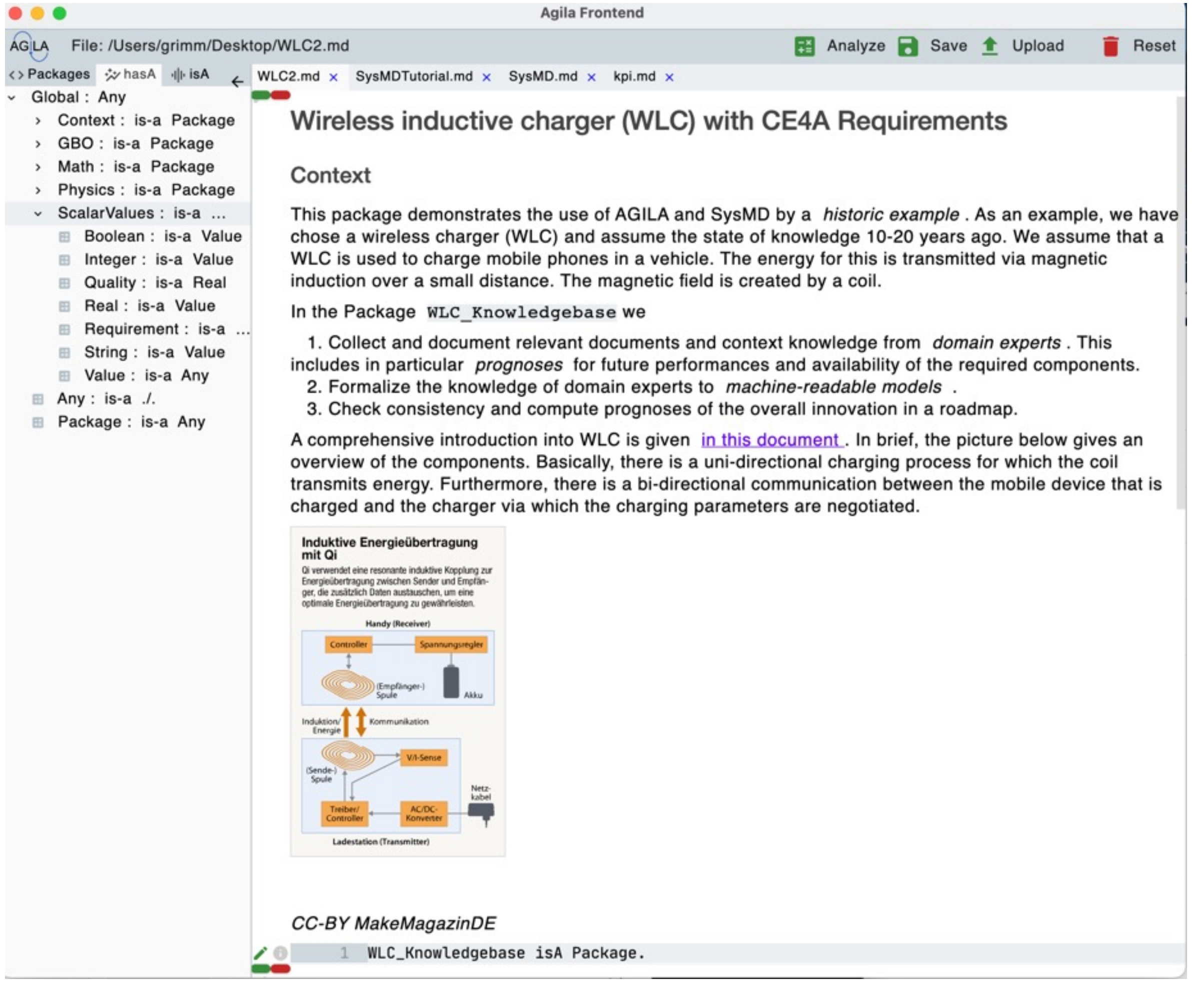
1. Introduction
2. SysMD notebook & SysMD language
3. Development environment and software details
4. Roadmap for SysMD - *How can I contribute?*

SysMD Notebook & Language Overview

SysMD Notebook

Notebook-like tool
Markdown editor
Markdown renderer
Code editor for SysMD/SysMLv2
Compiler
…

Proof-of-Concept implementation, work in progress



Wireless inductive charger (WLC) with CE4A Requirements

Context

This package demonstrates the use of AGILA and SysMD by a *historic example*. As an example, we have chose a wireless charger (WLC) and assume the state of knowledge 10-20 years ago. We assume that a WLC is used to charge mobile phones in a vehicle. The energy for this is transmitted via magnetic induction over a small distance. The magnetic field is created by a coil.

In the Package `WLC_Knowledgebase` we

1. Collect and document relevant documents and context knowledge from *domain experts*. This includes in particular *prognoses* for future performances and availability of the required components.
2. Formalize the knowledge of domain experts to *machine-readable models*.
3. Check consistency and compute prognoses of the overall innovation in a roadmap.

A comprehensive introduction into WLC is given in this document. In brief, the picture below gives an overview of the components. Basically, there is a uni-directional charging process for which the coil transmits energy. Furthermore, there is a bi-directional communication between the mobile device that is charged and the charger via which the charging parameters are negotiated.

Induktive Energieübertragung mit Qi

Qi verwendet eine resonante induktive Kopplung zur Energieübertragung zwischen Sender und Empfänger, die zusätzlich Daten austauschen, um eine optimale Energieübertragung zu gewährleisten.

CC-BY MakeMagazinDE

```
WLC_Knowledgebase isA Package.
```

SysMD Language

modeling & documentation language

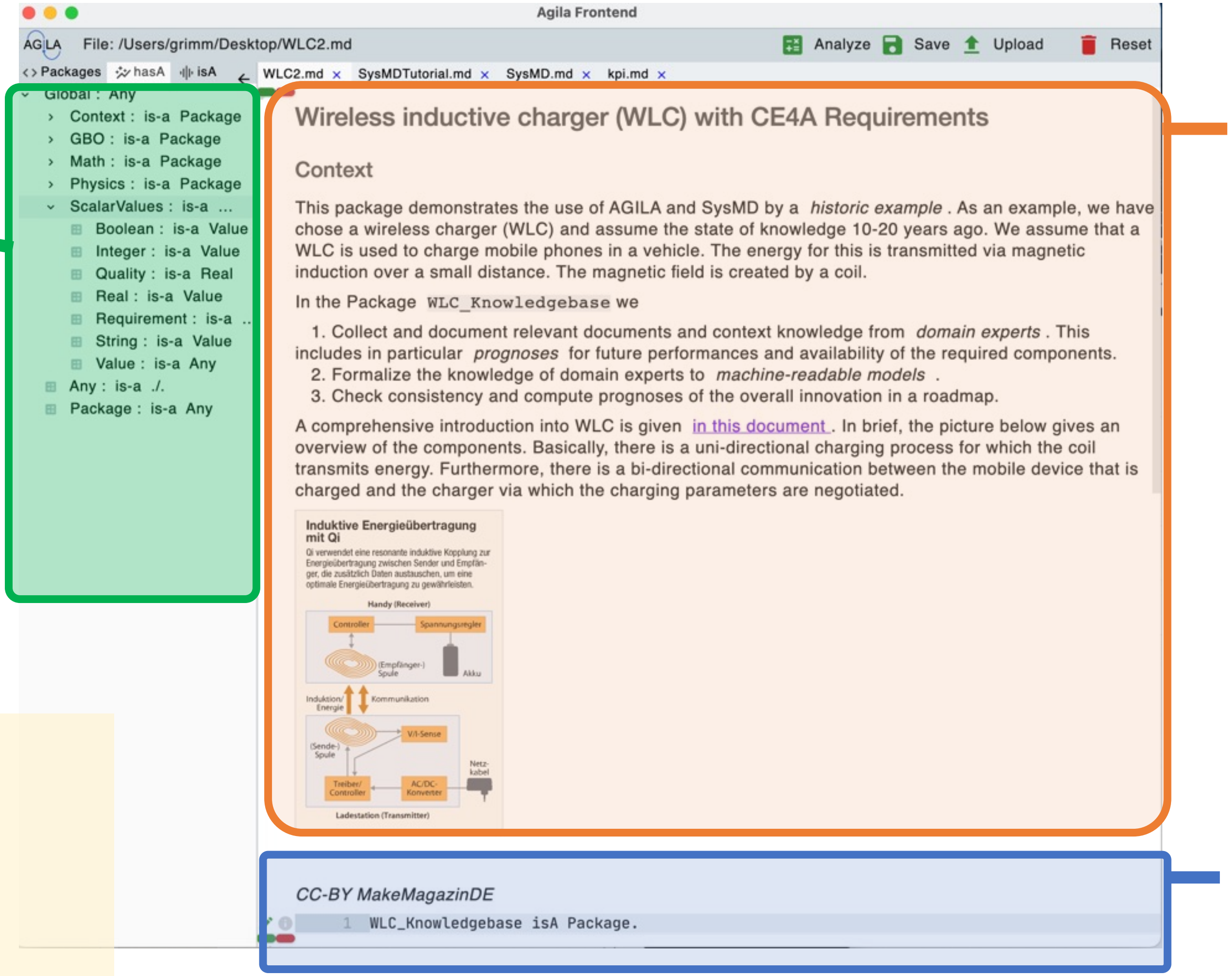
- *Markdown (MD)*
- *Feature models*
- *Requirements*
- *Constraints*
- *…*

(further windows for results of analysis)

SysMD Notebook: UI Overview

Navigation

- Projects
    - Branches, Commits
- Taxonomy
- Decomposition/ownership
- Relationships



Wireless inductive charger (WLC) with CE4A Requirements

Context

This package demonstrates the use of AGILA and SysMD by a *historic example* . As an example, we have chose a wireless charger (WLC) and assume the state of knowledge 10-20 years ago. We assume that a WLC is used to charge mobile phones in a vehicle. The energy for this is transmitted via magnetic induction over a small distance. The magnetic field is created by a coil.

In the Package `WLC_Knowledgebase` we

1. Collect and document relevant documents and context knowledge from *domain experts* . This includes in particular *prognoses* for future performances and availability of the required components.
2. Formalize the knowledge of domain experts to *machine-readable models* .
3. Check consistency and compute prognoses of the overall innovation in a roadmap.

A comprehensive introduction into WLC is given in this document . In brief, the picture below gives an overview of the components. Basically, there is a uni-directional charging process for which the coil transmits energy. Furthermore, there is a bi-directional communication between the mobile device that is charged and the charger via which the charging parameters are negotiated.

Induktive Energieübertragung mit Qi

Qi verwendet eine resonante induktive Kopplung zur Energieübertragung zwischen Sender und Empfänger, die zusätzlich Daten austauschen, um eine optimale Energieübertragung zu gewährleisten.

CC-BY MakeMagazinDE

```
WLC_Knowledgebase isA Package.
```

Documentation

- Markdown-format
- Tables, figures, links, …

Models

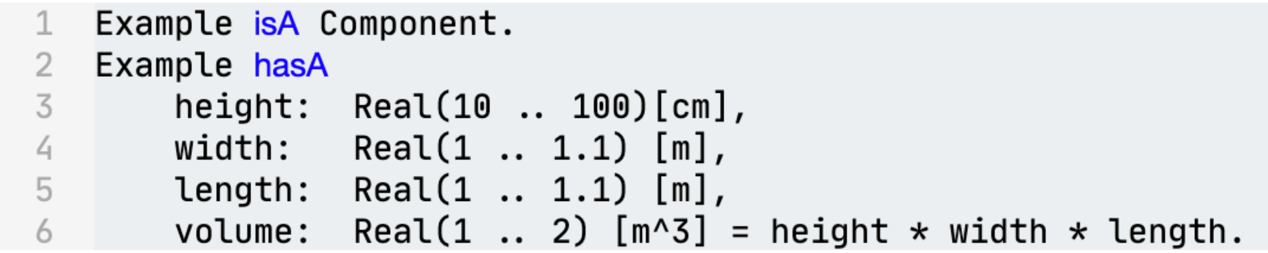
- In SysMD, SysMLv2 textual
- Taxonomy
- Decomposition
- Values, constraints
- Relationships

Not shown are windows for

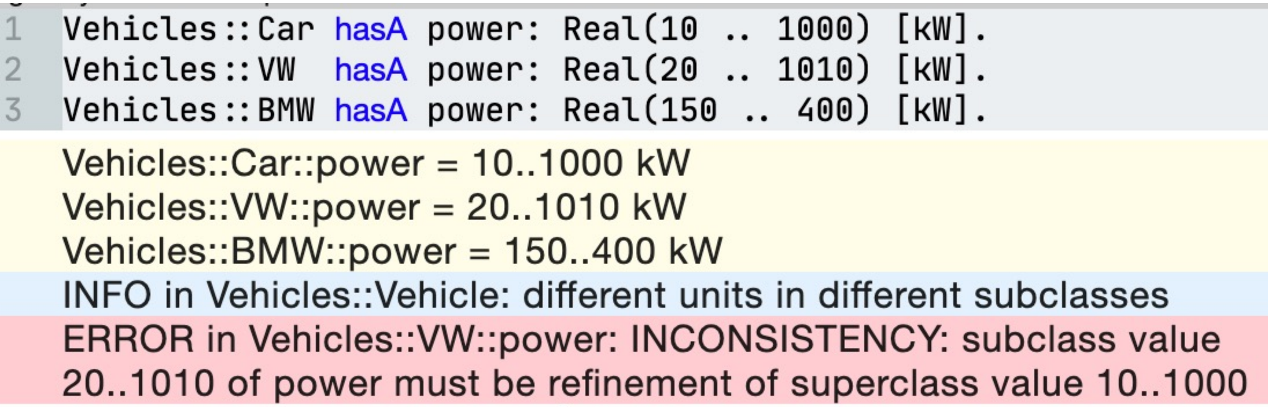
- *Results of analysis*
- *Agenda*
- *Warnings, errors*

SysMD Notebook: Constraint Propagation (Bi-Dir.)

- Direct dependencies given by expressions
  - Bi-directional constraint propagation for Reals, Integers;
    - Check and conversion of Units, Domains (SI, national units, dB, Date/Time)
  - Satisfiability problem for Booleans
- Inheritance
  - Models variants or potential solutions of similar things
  - Consistency check: Liskov principle satisfied?
- Decomposition
  - SUM(…) computes aggregations (transitive)
  - Constraint propagation includes cardinality

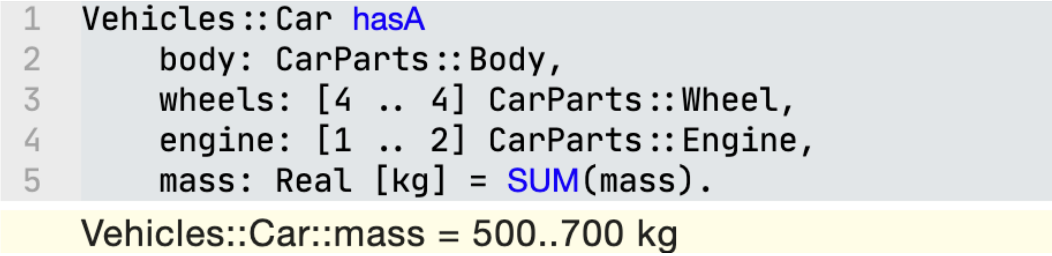


```
Example isA Component.
Example hasA
    height:  Real(10 .. 100)[cm],
    width:   Real(1 .. 1.1) [m],
    length:  Real(1 .. 1.1) [m],
    volume:  Real(1 .. 2) [m^3] = height * width * length.
```



```
Vehicles::Car hasA power: Real(10 .. 1000) [kW].
Vehicles::VW  hasA power: Real(20 .. 1010) [kW].
Vehicles::BMW hasA power: Real(150 .. 400) [kW].
```

```
Vehicles::Car::power = 10..1000 kW
Vehicles::VW::power = 20..1010 kW
Vehicles::BMW::power = 150..400 kW
INFO in Vehicles::Vehicle: different units in different subclasses
ERROR in Vehicles::VW::power: INCONSISTENCY: subclass value
20..1010 of power must be refinement of superclass value 10..1000
```



```
Vehicles::Car hasA
    body: CarParts::Body,
    wheels: [4 .. 4] CarParts::Wheel,
    engine: [1 .. 2] CarParts::Engine,
    mass: Real [kg] = SUM(mass).
```

```
Vehicles::Car::mass = 500..700 kg
```

SysMLv2 vs. SysMD language

SysML v2 (textual)		SysMD
Based on KerML, SysML API		Based on KerML, SysML API (subset; deviations are "Bug").
Syntax close to programming languages.	≠	Closer to **natural language**, "top-down", interactive
Target: modeling and SE experts.	≠	Target: users are **domain experts**.
Documentation added to model.		Model added to **documentation** (text, videos, …).
Expressions for modeling of constraints, spec.	≠	Syntax separates **specification** and **modeling**.

```
Wheel {
  value mass: Real = 70 [kg];
  // model mass with 50 to 100 kg
}

Car :> Vehicle {
  part Wheel [4 .. 8];
  in value mass = … // model constraint, unit, …
```

```
Car isA Vehicle.

Car hasA
  wheel: [4 .. 8] Wheel,
  mass: Mass(100..1000) kg = sumHasA(mass).

Wheel hasA
  mass: all Mass(50 .. 100) kg = ….
```

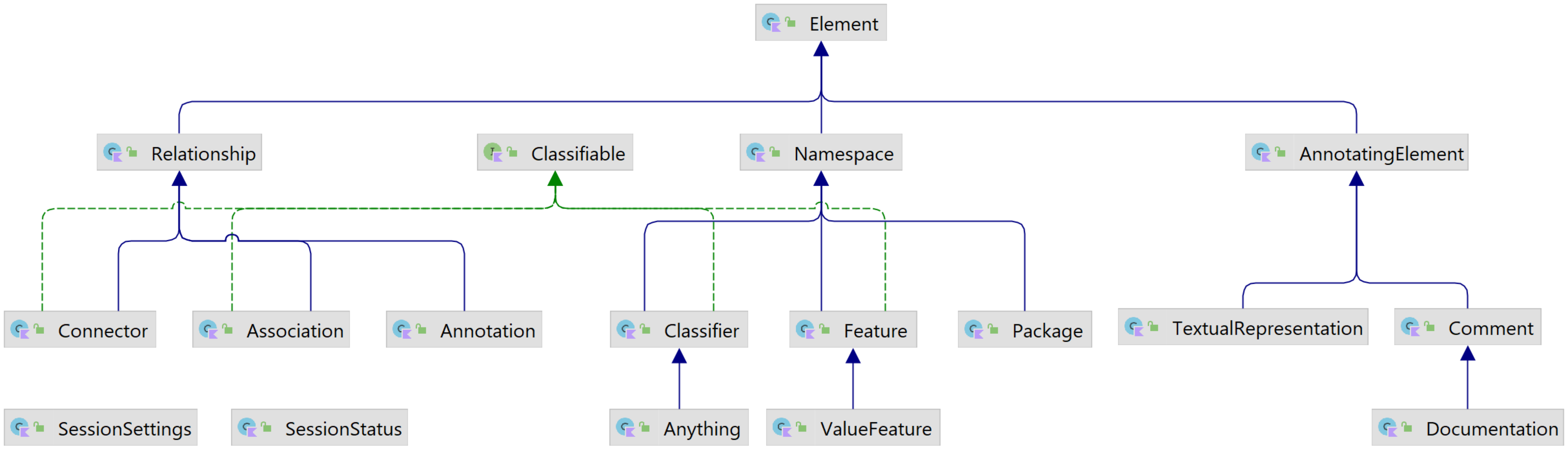
SysMD Syntax Cheatsheat

Subject	Predicate	Object	More objects	End
Name *<Element>*	**defines**	*isA Definition, next line*	(; Definition)*	.
	isA	*Name* *<Classifiable, ClassOfMetamodel>*		
	hasA	*Name*: [**all|one**] [Multiplicity] *Name<Type>* [Constraints] [= Expr.]	(, Object)*	
	imports	*Name<Project, Namespace>*	(, Name)*	
	Name *<Association>*	*Name* *<Element>*	(, Name)*	

Pre-defined classes and projects

- Any(thing) = root of all taxonomies (isA); Global = root of ownership/features (hasA)
- ScalarValues (Classifies Real, Boolean, Integer, … as in SysMLv2)
- ISO26262 Ontology: Element, Function, Component, Part, SoftwareUnit, (…), also relationships:
  - Component *implements* Function, Component *satisfies* Requirement, Processor *executes* Software
- GBO, MissionProfiles, Math, Physics.

KerML metamodel implementation



Note:

1) We are not yet fully compatible … working on it, but quite ok.
2) We strive to consolidate number of classes a bit. (e.g., ValueFeature includes Expression, Multiplicity, FeatureValue, …)
3) We strive to increase performance, reduce complexity – not all relationships represented by instances of Relationship (e.g. ownership, inheritance)

Inclusive Modeling with SysMD

SysMD

1. Introduction
2. SysMD notebook & SysMD language
3. Development environment and software details
4. Roadmap for SysMD - *How can I contribute?*

Development environment

SysMD

- Gradle v7 and/or IntelliJ IDEA
  - Commonmark Markdown parser
  - Apache math (LP solver) and jAADD for CSP/nonlinear/discrete problems
- Kotlin JVM
  - Jetpack Compose Desktop for UI
- Optional for REST API, Backend
  - Spring boot, ArrangoDB as repository
- Junit Jupiter (500-1000+ tests depending on branch)

Development environment

SysMD

- Nothing is better than a live look at the code
  - Build: “gradle run”

- … and, of course, running code & demo ☺

(live … not as video)

Contents

SysMD

1. Introduction
2. SysMD Notebook
3. SysMD Language
4. Roadmap for SysMD - *How can I contribute?*

SysMD wants you!

SysMD home page

- `https://cpsgit.cs.uni-kl.de/open/sysmd`

- Students (projects/theses/ …)
  - Improvements in Markdown rendering
  - Improvements in code editor
  - SysMLv2 textual, KerML interoperatbility
  - SAT/SMT interfaces
  - Tests
  - Knowledge bases, models
  - … any own ideas? …
- Industry
  - EC or nationally funded projects
  - Case studies

Outlook

SysMD

- Currently, still some issues and bugs
  - Some industrial users for evaluation
  - WiP: Runtime-Verification, simulation-data needs integration
  - WiP: More beautiful Web-Frontend (React JS, Hierarchical documents, etc.)

- 1st Release to public (open source) Summer 2022
  - Basically, as shown, but with less bugs & some libraries
  - Open source for most parts

    (Small parts in probabilistic CSP are patent pending; NOT the modeling; is not necessarily needed)

- 2nd Release end 2023/2024: “modular digitalization toolkit”
  - Integrated DevOps interface
  - Generation of interfaces to virtual prototypes