

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



Inclusive Modeling with 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”?

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

“... 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 requirements, development to operation, runtime-verification,
 - 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



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*

The screenshot shows the SysMD Notebook interface with a file named 'WLC2.md' open. The left sidebar displays a package hierarchy under 'Global : Any', including 'Context : is-a Package', 'GBO : is-a Package', 'Math : is-a Package', 'Physics : is-a Package', and 'ScalarValues : is-a ...'. The main content area is titled 'Wireless inductive charger (WLC) with CE4A Requirements' and contains a 'Context' section. The context text describes the use of AGILA and SysMD by a 'historic example' of a wireless charger (WLC) used to charge mobile phones in a vehicle. It lists three steps: 1. Collect and document relevant documents and context knowledge from domain experts, 2. Formalize the knowledge of domain experts to machine-readable models, and 3. Check consistency and compute prognoses of the overall innovation in a roadmap. Below the text is a diagram titled 'Induktive Energieübertragung mit Qi' showing the components of a Qi wireless charging system, including a 'Handy (Receiver)' with a 'Controller', 'Spamungsglied', '(Empfänger-) Spule', and 'Akku', and a 'Ladestation (Transmitter)' with a '(Sender-) Spule', 'V/I-Sense', 'Treiber/Controller', 'AC/DC-Konverter', and 'Netz-kabel'. The diagram illustrates the flow of 'Induktions-Energie' and 'Kommunikation' between the two coils. At the bottom, there is a status bar indicating '1 WLC_Knowledgebase isA Package.'

(further windows for results of analysis, errors, ...)

SysMD Language

modeling & documentation
language

- Markdown (MD)
- Feature models
- Requirements
- Constraints
- ...

SysMD Notebook: UI Overview

Navigation

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

Not shown are windows for

- Results of analysis
- Agenda
- Warnings, errors

The screenshot shows the SysMD Notebook interface. The top bar includes the AGILA logo, the file path, and buttons for Analyze, Save, Upload, and Reset. The left sidebar displays a package hierarchy for 'WLC2.md'. The main content area shows the 'Wireless inductive charger (WLC) with CE4A Requirements' package, including its context, a list of requirements, and a diagram of inductive energy transfer. The bottom status bar shows the license 'CC-BY MakeMagazinDE' and a list of packages.

AGILA File: /Users/grimm/Desktop/WLC2.md Analyze Save Upload Reset

<> Packages hasA isA WLC2.md SysMDTutorial.md SysMD.md kpi.md

Global: Any

- > Context: is-a Package
- > GBO: is-a Package
- > Math: is-a Package
- > Physics: is-a Package
- > ScalarValues: is-a ...
 - Boolean: is-a Value
 - Integer: is-a Value
 - Quality: is-a Real
 - Real: is-a Value
 - Requirement: is-a ..
 - String: is-a Value
 - Value: is-a Any
- Any: is-a ./
- Package: is-a Any

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.

Handy (Receiver)

Controller Spannungsglied Akku

Induktions-Spule (Empfänger) Spule

Induktions-Energie Kommunikation

Sende-Spule V/I-Sensor

Treiber/Controller AC/DC-Konverter

Ladestation (Transmitter)

Netz-kabel

CC-BY MakeMagazinDE

1 WLC_Knowledgebase isA Package.

Documentation

- Markdown-format
- Tables, figures, links, ...

Models

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

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

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

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

Vehicles::Car::power = 10..1000 kW

Vehicles::VW::power = 20..1010 kW

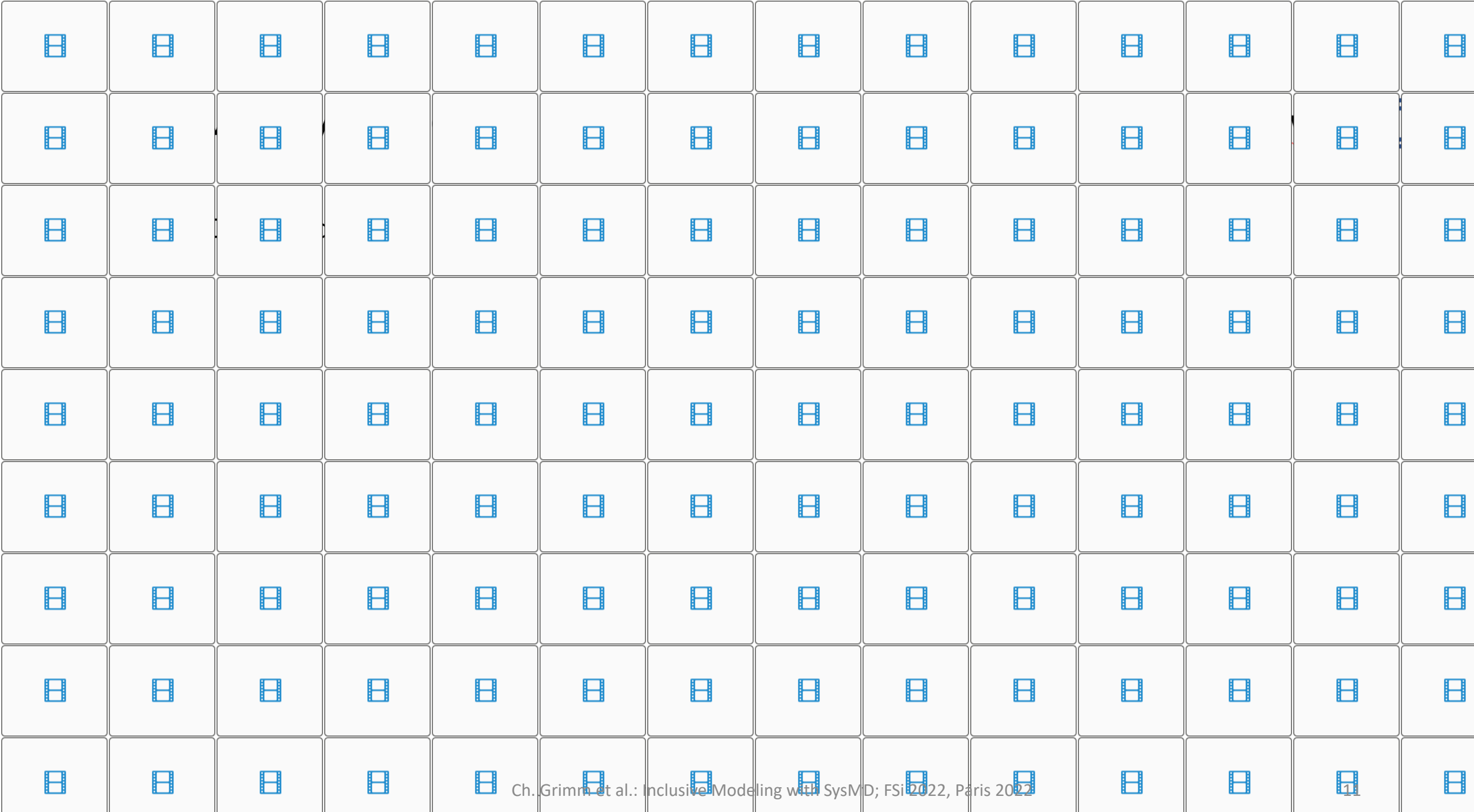
Vehicles::BMW::power = 150..400 kW

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

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

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



SysMLv2 vs. SysMD language



SysML v2 (textual)

- Users: modeling and SE experts.
- Syntax close to programming languages.
- Documentation added to model.
- Expressions for modeling of constraints, spec.
- Based on KerML metamodel, SysML API

```
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, ...
```

SysMD

- ≠ • Users: **domain experts**.
- ≠ • Closer to **natural language**, “top-down”, interactive
- ↻ • Model added to **documentation** (text, videos, ...).
- ≠ • Syntax separates **specification** and **modeling**.
- • Based on KerML metamodel, SysML API (subsets).

```
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 Cheatsheet

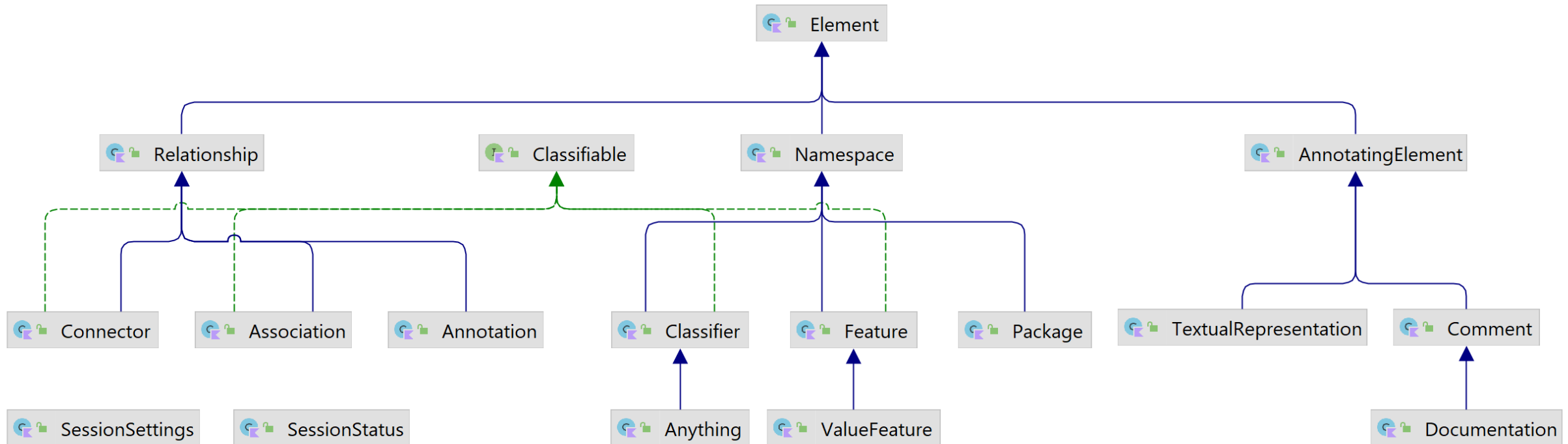


Subject	Predicate	Object	More	End
<i>Name</i> <i><Element></i>	defines	<i>Classification</i>	(; Classification)*	.
	<div>Classification</div> isA	<i>Name</i> <i><Classifiable, ClassOfMetamodel></i>		
	<div>FeatureSpec.</div> hasA	<i>Name</i> : [all one] [Multiplicity] <i>Name</i> <i><Type></i> [Constraints] [= Expr.] <div>→ Instantiation</div>	(, FeatureSpec)*	
	<div>Import</div> imports	<i>Name</i> <i><Project, Namespace></i>	(, Name)*	
	<div>Relationship</div> <i>Name</i> <i><Association></i>	<i>Name</i> <i><Element></i>	(, 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



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



Development environment



- Gradle v7 and/or IntelliJ IDEA, dependencies
 - 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



- Nothing is better than a live look at the code
 - Build: “gradle run”
- ... and, of course, running code & demo 😊

(live ... not as video)

SysMD home page

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

Contents



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



SysMD wants you!



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

Outlook



- **Currently, still a few issues and bugs**
 - Some industrial users do 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