

Digital Engineering

Introducing next level Requirements Engineering using SysML v2



Peter Schedl
Program Manager
IBM Engineering Lifecycle Management
peter.schedl@de.ibm.com



Agenda

1. Models & SysML
2. Why a new Version
3. SysML v2
4. SysML v2 as base for next generation RE / MBSE
5. The Future of MBSE with SysML v2



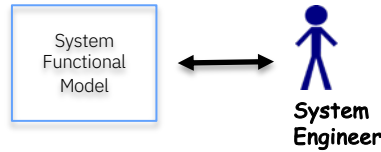
Modeling Approaches in Requirements Engineering

- Enrich Textual Requirements w Diagrams (Context Diagram, Architecture Overview, State Diagram)
- Model parts drive Requirements chapters i.e. Static Architecture (Structure & Interfaces)
- Model and Requirements evolve side by side: End-2-End Methods like Harmony
- Model only: Requirements completely derived from the Model

Model Based Systems Engineering (MBSE)

A visual approach to understanding requirements and realizing them into a robust system design

“The future of Systems Engineering is predominantly Model-Based”
(INCOSE)
...System Models enable fully traceable, self documenting and verifiable systems.
Important for the Digital Thread!



Why Modeling ?

...to find and communicate a robust system architecture on the right level of abstraction

MBSE is a standards based Systems Engineering practice that incorporates:

- Modeling **language** – SysML
- Modeling **method**
- Modeling **tool**
 - + Requirements management tool

What is SysML?

SysML ([OMG System Modeling Language](#)) is the official standard notation for MBSE:

SysML is a general-purpose modeling language for modeling systems that is intended to facilitate a model-based systems engineering (MBSE) approach to engineer systems. It provides the capability to create and visualize models that represent many different aspects of a system. This includes representing the requirements, structure, and behavior of the system, and the specification of analysis cases and verification cases used to analyze and verify the system.

Latest v1 Version:

Title:	OMG Systems Modeling Language
Acronym:	SysML®
Version:	1.7
	This version was superseded by a newer inventory. The latest version can be found here: SysML
Document Status:	formal ⓘ
Publication Date:	June 2024
Categories:	Modeling Systems Engineering Platform
IPR Mode ⓘ	RF-Limited ⓘ

There has been an industry-wide call for a more comprehensive MBSE standard

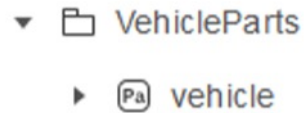
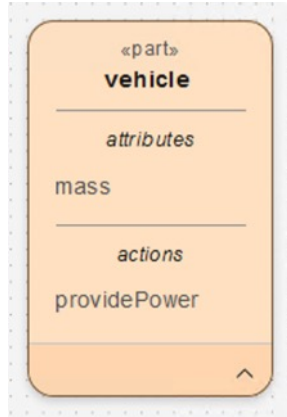
- Remove dependency on UML = more SE oriented
- More formal definition = more consistent, exchangeable models
- How to model best practice guides
- Simple to use tooling

Requires **NEW**:

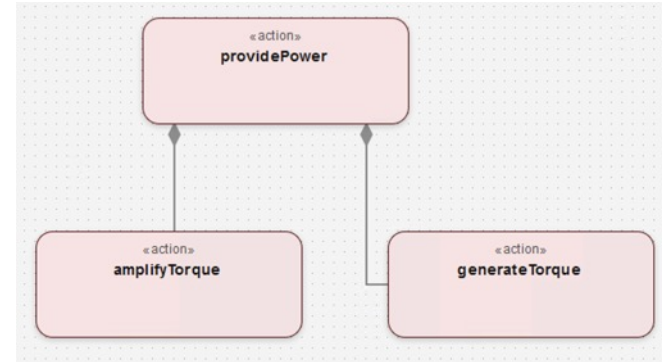
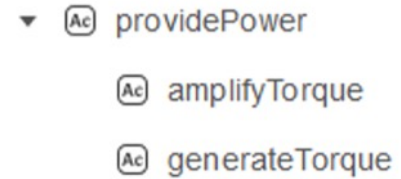
- Modeling **language** – SysML v2
- Modeling **method**
- Modeling **tool**
 - + Requirements management tool

SysML v2 Examples of textual and graphical Notation

```
package 'VehicleParts' {  
  part vehicle {  
    attribute mass;  
    perform providePower;  
  }  
}
```



```
action providePower {  
  action generateTorque;  
  action amplifyTorque;  
}
```



SysML v2 Language Concepts

Requirements

Structure

- decomposition
- interconnection
- classification

Behaviour

- function-based
- state-based
- sequence-based
- use cases

Views

Analysis

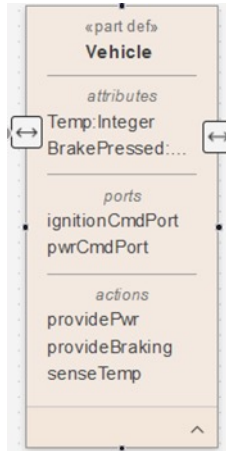
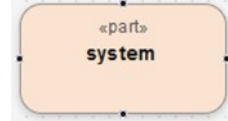
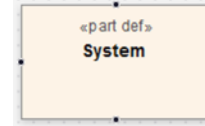
- analysis cases
- expression language

Verification

- verification cases

SysML v2 Concepts - Structure

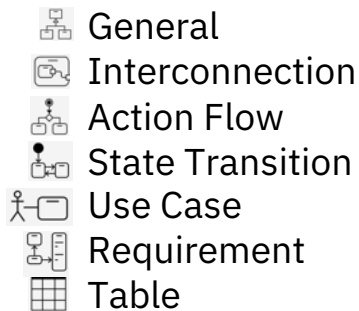
- Each element has **definition** (square corners) (<<def>>) and **usage** (rounded corners)
- <<part>> definition: Can contain features such as attributes, ports, actions, and states
- <<attribute>> definition: Data type, Can include quantity kinds such as Torque with units
Can include complex data types such as vectors
- <<port>> definition: Connection point on parts, directed features with in, out and in/out
Can be conjugated (~)
- <<interface>>, <<connection>> (incl. Flow), <<allocation>> definition



SysML v2 Concepts - Views

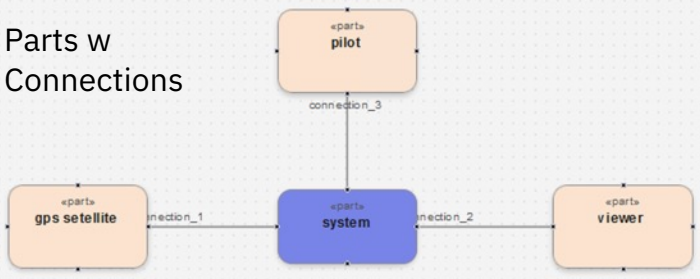
Standard Views:

- General (was Pkg, BDD)
- Interconnection (was IBD)
- Action Flow, State Transition
- Case, Sequence
- Geometry
- Grid, Browser



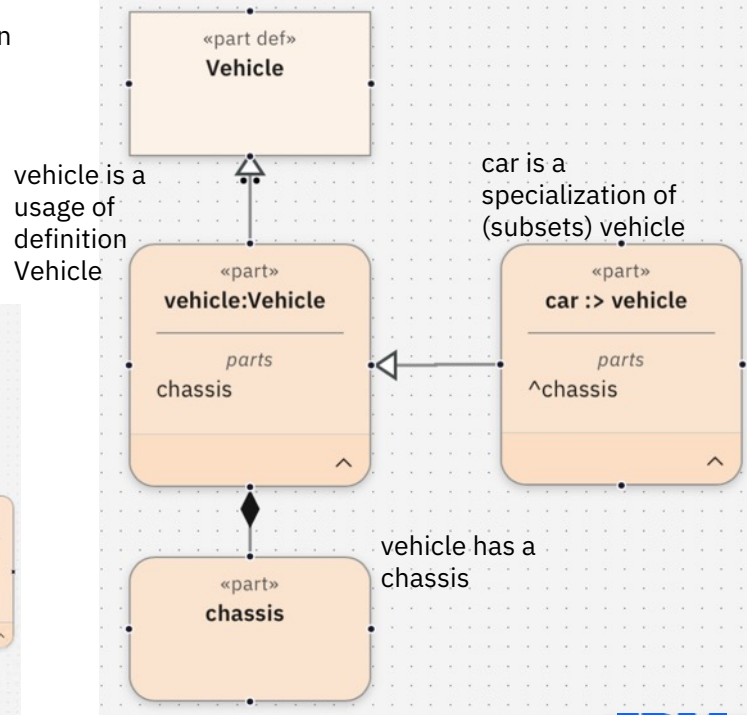
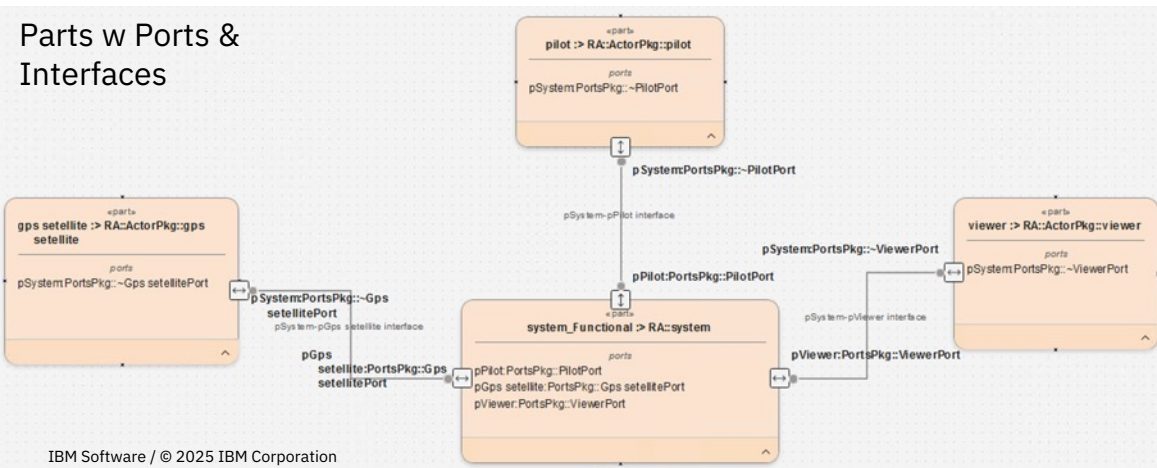
SysML v2 Diagram Examples

Parts w Connections



Parts may be used w/o definition

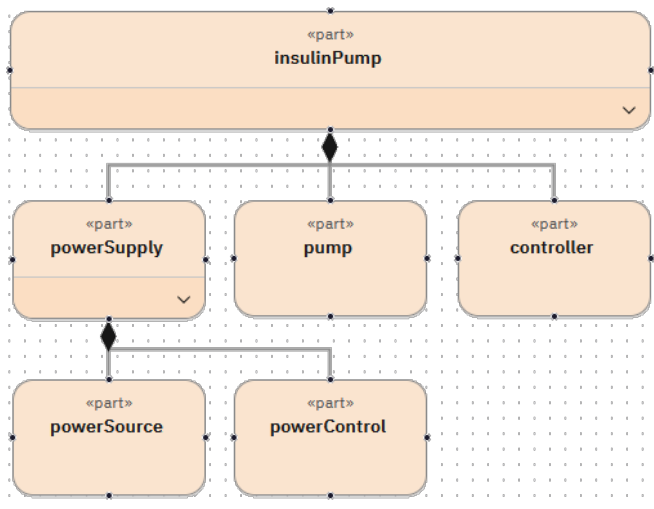
Parts w Ports & Interfaces



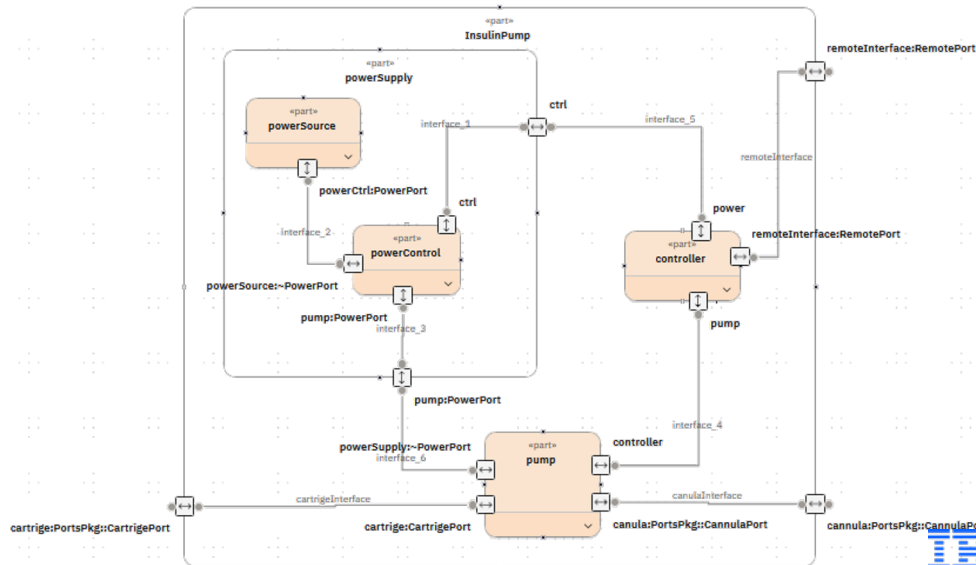
SysML v2 Examples - System Structure

A **system** is modeled as a composite **part**, and its part usages may themselves have further composite structure

Visualized as a Tree View



Visualized as a hierarchical Interconnection View

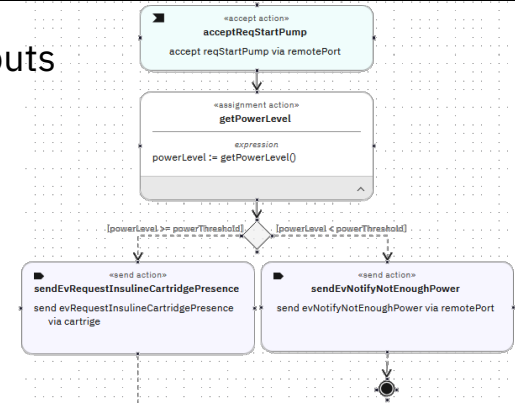


SysML v2 Concepts - Behavior

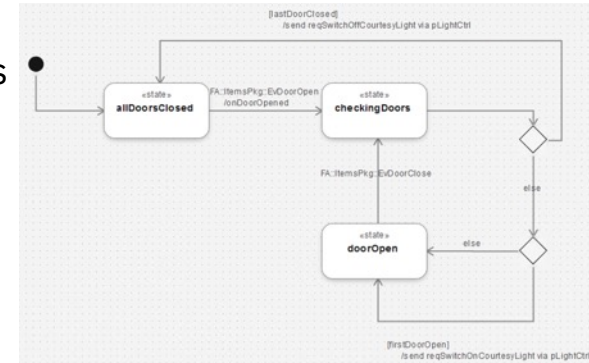
- **<<action>>** definition: Kind of behaviour that transforms inputs to outputs
Can have Fork & Join, Decision & Merge nodes

Action types: <<assign>>, <<loop>>, <<accept>>, <<send>>

Actions generate effects on items and parts



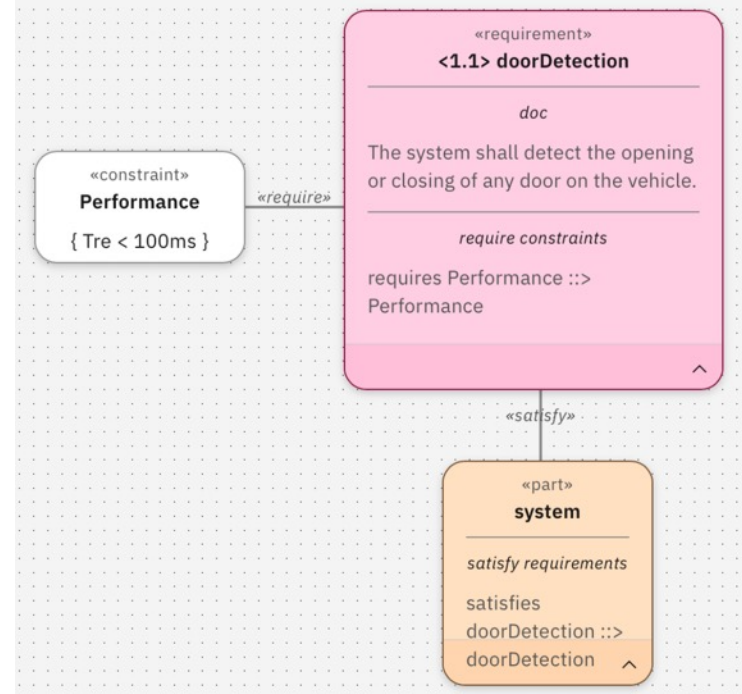
- **<<state>>** definition: Kind of behaviour that responds to events
Can be <<parallel>>
Enables entry, exit, and do actions



SysML v2 Concepts - Requirement

- **<<requirement>>** definition: Has subject it applies to
Can have assume and require constraints which can be bound to item attributes with **<<satisfies>>**
Alternative is a stakeholder **<<concern>>**.

	Req ID	Req Name	Req Text
1	1.1	doorDetection	The system shall detect the opening or closing of any door on the vehicle.
2	1.2	doorStatusTracking	The system shall keep track of how many doors are currently open.
3	1.3	courtesyLightOn	The system shall switch on the courtesy light when door is opened.
4	1.4	courtesyLightOff	The system shall switch off the courtesy light when all doors are closed.
5	1.5	courtesyLightDim	The system shall dim the courtesy light for a period of 30 seconds before switching it off.



SysML v2 vs. v1

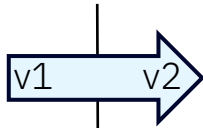
SysML v2	SysML v1
part / part def	part property / block
attribute / attribute def	value property / value type
port / port def	proxy port / interface block
action / action def	action / activity
state / state def	state / state machine
constraint / constraint def	constraint property / constraint block
requirement / requirement def	requirement
connection / connection def	connector / association block
view / view def	view

Improving Systems Engineering with SysML v2

Model-based systems engineering



Derived from
Software Engineering
Simplified for
Systems Engineers



**Next Generation
Systems Modeling Language**

Defined by
Systems Engineers
for
Systems Engineers

New Paradigm for Modeling!

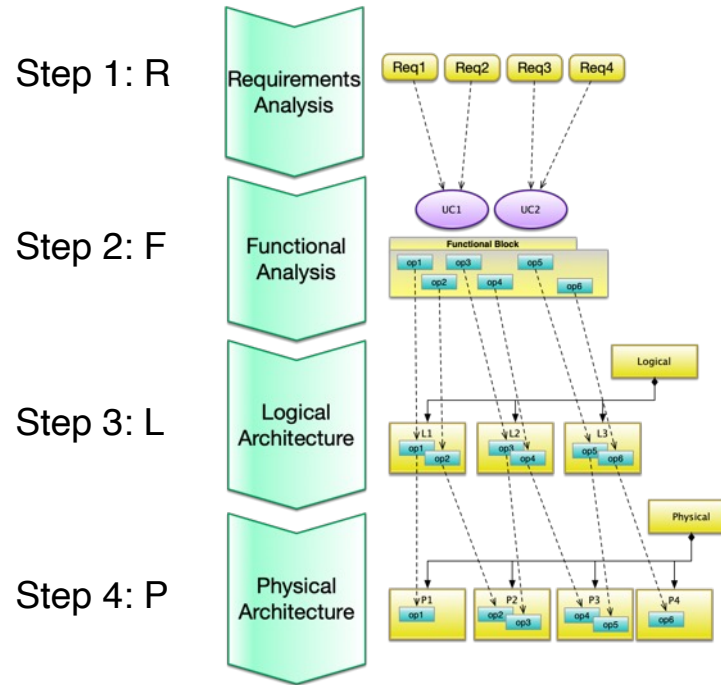
- Formal but **easy** to learn and use
- Standard API's allow process automation

SysML v2 the
Language for
MBSE

Modeling Approaches in Requirements Engineering

- Enrich Textual Requirements (Context Diagram, Architecture Overview, State Diagram)
- Model parts drive Requirements chapters i.e. Static Architecture (Structure & Interfaces)
- Model and Requirements evolve side by side: End-2-End Methods like Harmony
- Model only: Requirements completely derived from the Model

The Harmony Method: A MBSE Best Practice



MBSE practice providing a structured and automated workflow

- Evolved over 20 years working with Automotive and A&D
- Follows RFLP paradigm
- Guides the engineer on what views artifact needs to be developed and automates creation of views from other views

The Harmony Method Benefits

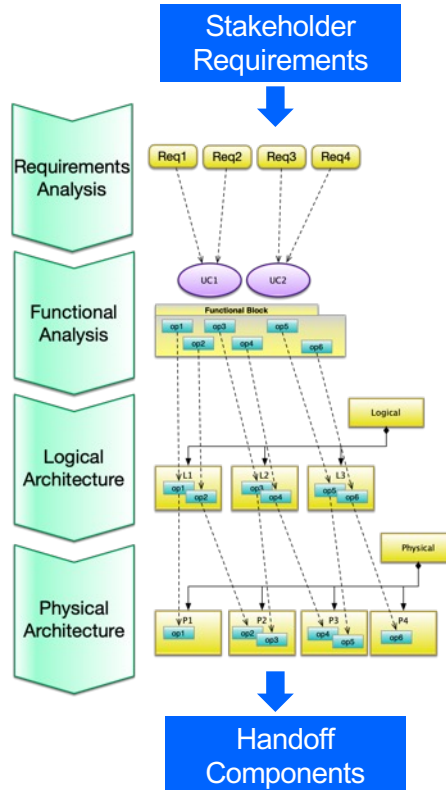
Guides the User how to derive System Requirements from Stakeholder Requirements

Connects Requirements & Model via Links

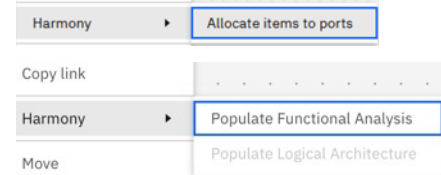
Describes how to define the System Context & Functional Flows

Automates allocation of Operations into the System Architecture

Exports logical and/or physical Components for follow on Domains

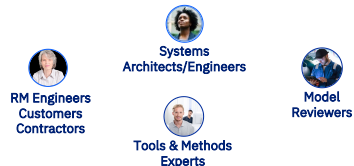


Harmony – Proven MBSE method



Automation requires Tool support

Why a (new) Tool?



Realtime, Web based, Model Based Collaborations



SysML v2 Language

- + OSLC
- + REST APIs
- + Cross-domain Digital Threads

Easy to Deploy (Web- & Cloud-native)

Easy to Use

Easy Deliverables (Handoff)

Supporting collaboration

Model-based integrations

Tool extensibility

IBM
Rhapsody
Systems
Engineering



Easy to Use

The screenshot displays the IBM Rhapsody Systems Engineering interface. The main workspace shows a 'system context view' diagram with a central 'system' component connected to four sub-components: 'door[2..5]', 'driver', 'light', and 'heater'. Each connection is labeled with a system name and 'connection'. The interface includes a 'Model Browser' on the left, a 'Diagrams Area' at the bottom, and a 'Properties Area' on the right. Callouts highlight the 'Drawing Tools' and 'Edit Tools' palettes.

Model Browser

- 1_RequirementsAnalysisPkg
 - ActorPkg
 - RequirementsPkg
 - UseCasePkg
 - system context view
 - system
 - door_system connection
 - driver_system connection
 - heater_system connection
 - light_system connection
- 2_FunctionalAnalysisPkg
- 3_LogicalArchitecturePkg
- 4_PhysicalArchitecturePkg

Diagrams Area

Properties Area

system context view properties

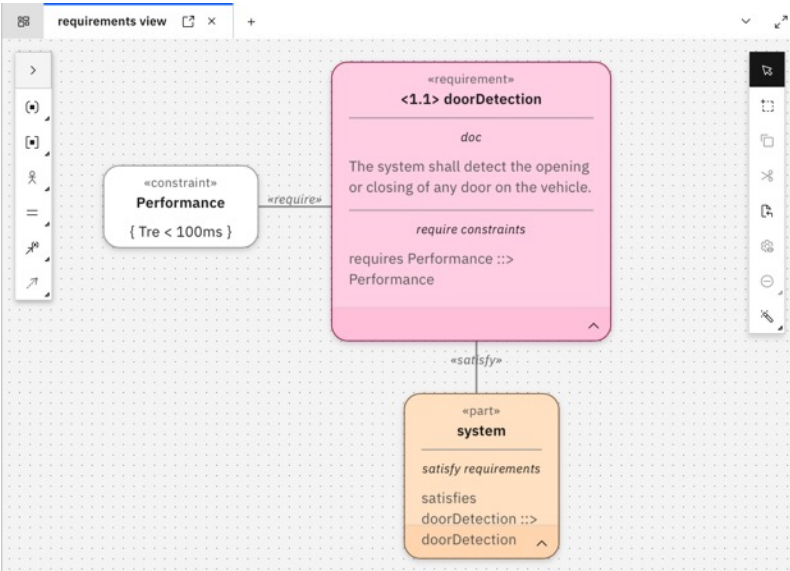
- Direction: In
- Feature value
- Redefined features: None
- Subsetted features: None
- Description
- Other
- Owned relationships
- References
- Links

Drawing Tools

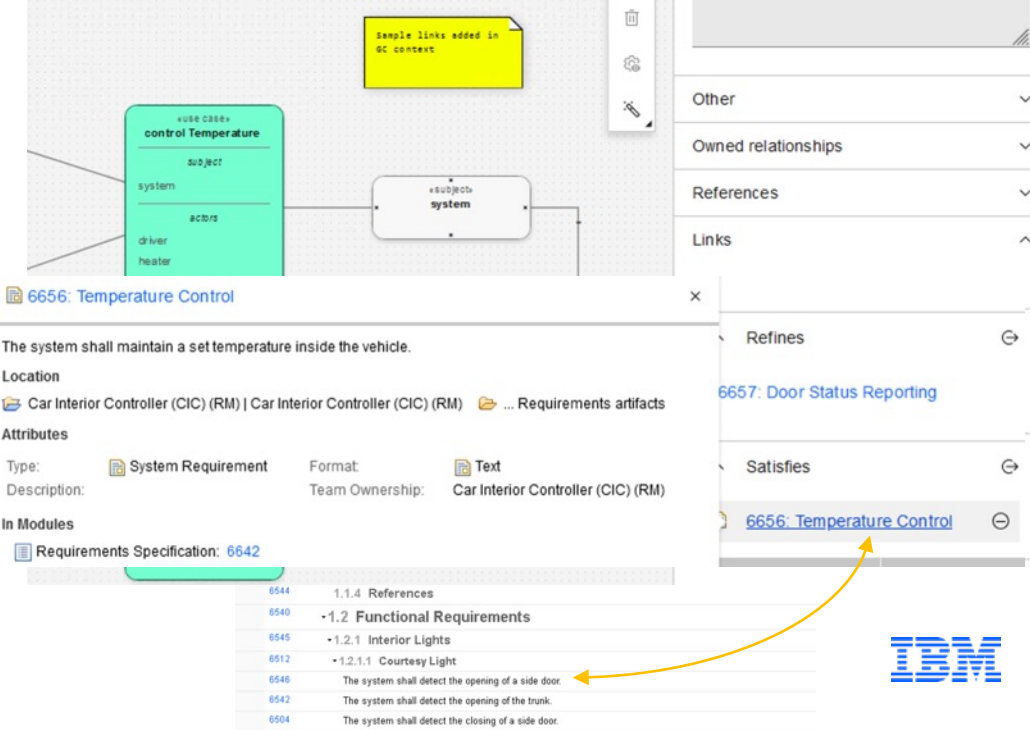
Edit Tools

Models & Requirements

Local Requirements



OSLC Connected Requirements - DOORS Next Example

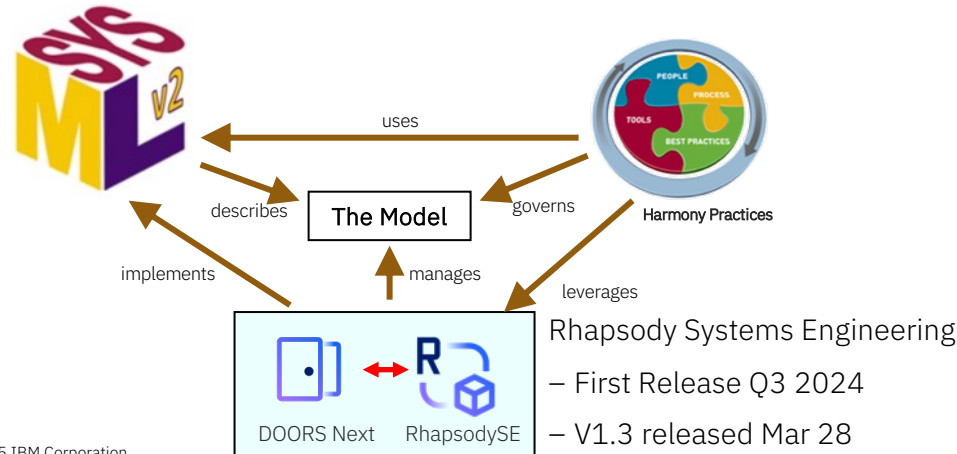


SysML v2 drives MBSE to the next Level

Combination of **new**

- Modeling language – **SysML v2**
- Modeling method – **Harmony v2**
- Modeling tool – **Rhapsody Systems Engineering**
- Requirements tool – DOORS Next Generation

...enables
MBSE for Everyone



Thank You



The banner features a blue background with a circuit-like pattern. On the left is a circular profile picture of Peter Schedl, a man with glasses. To the right is a glowing blue sphere with a network of white lines. Below the profile picture is the email address peter.schedl@de.ibm.com.

Peter Schedl ✓
Program Manager Industry Solutions at IBM
Greater Munich Metropolitan Area · [Contact info](#)
500+ connections
www.linkedin.com/in/peter-schedl-3a9aab216

IBM

Further information:

[IBM Engineering Lifecycle Management Automotive Compliance](#)

[IBM Engineering Lifecycle Management Overview](#)

[IBM Rhapsody Systems Engineering](#)

© Copyright IBM Corporation 2025. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and trademark information](#).

