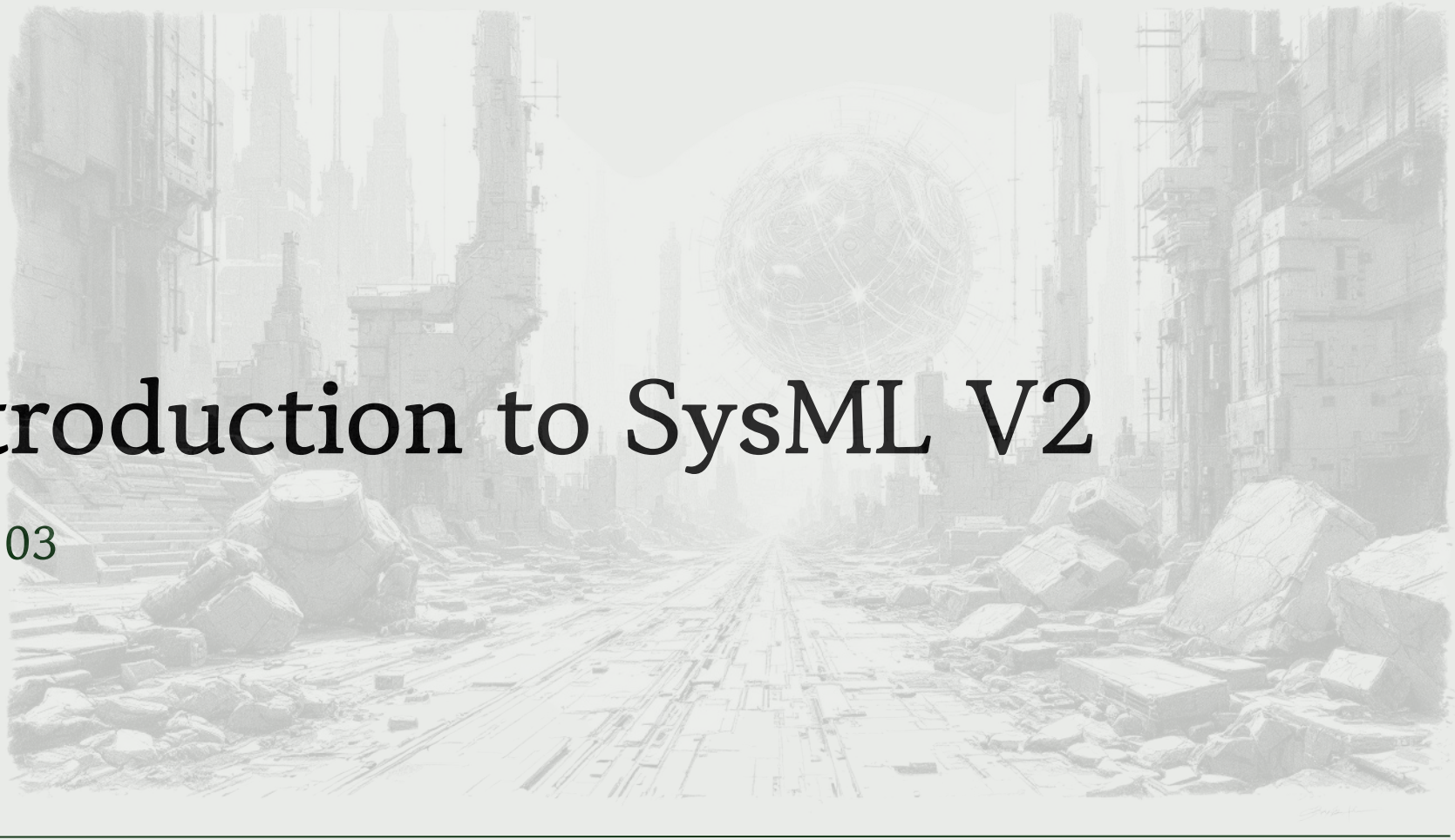


Introduction to SysML V2

Week 03

The SysML V2 Shift

From Visual Semantics to Machine-Readable Models

SysML v2 advances systems modeling by making semantics explicit, precise, and machine-readable — enabling automation, validation, and reuse at scale.

Key Shifts:

- SysML v1 had **semantic meaning**, but it was often tied to **diagrams** and **human interpretation**
- SysML v2 makes semantics **formal and computable**, allowing tools to reason about models directly
- The model is now a **structured data representation**, not just a diagram with meaning attached
- This enables **automated analysis, simulation, and digital thread integration**

SysML V2 - Modeling as a First-Class Citizen

With SysML v2, the model itself becomes the authoritative source, equally accessible through diagrams and text, ensuring clarity and consistency.
Textual and graphical modeling are fully integrated

- **No redundancy**—diagrams and text reflect the same model
- Enables **rigorous analysis, traceability, and reuse**
- Designed for **machine processing** and **system automation**

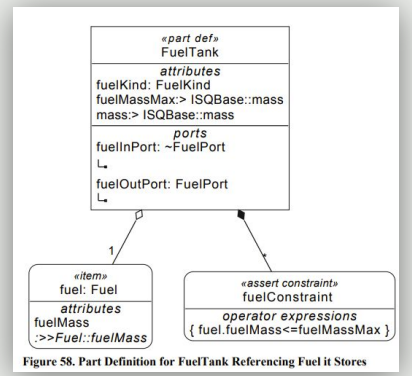
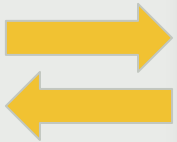


Figure 58. Part Definition for FuelTank Referencing Fuel it Stores



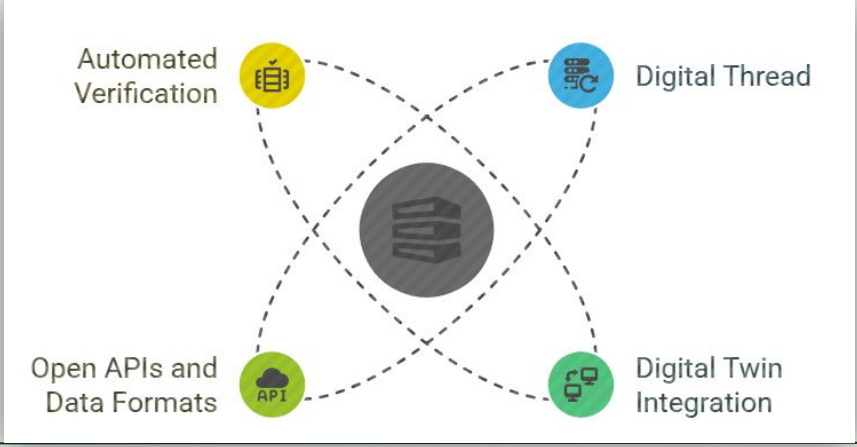
```
part def FuelTank{
    attribute mass :> ISQ::mass;
    attribute fuelKind:FuelKind;
    ref item fuel:Fuel{
        attribute redefines fuelMass;
    }
    attribute fuelMassMax:>ISQ::mass;
    assert constraint fuelConstraint {fuel.fuelMass<=fuelMassMax}
    port fuelOutPort:FuelPort;
    port fuelInPort:~FuelPort;
}
```

Image from OMG Systems Modeling Language™ (SysML®) Version 2.0 Beta 3 (Release 2025-02), Page 638.

Built for the Digital Engineering Era

SysML v2 is designed with the future in mind—one where digital models must integrate seamlessly across lifecycle stages and engineering tools.

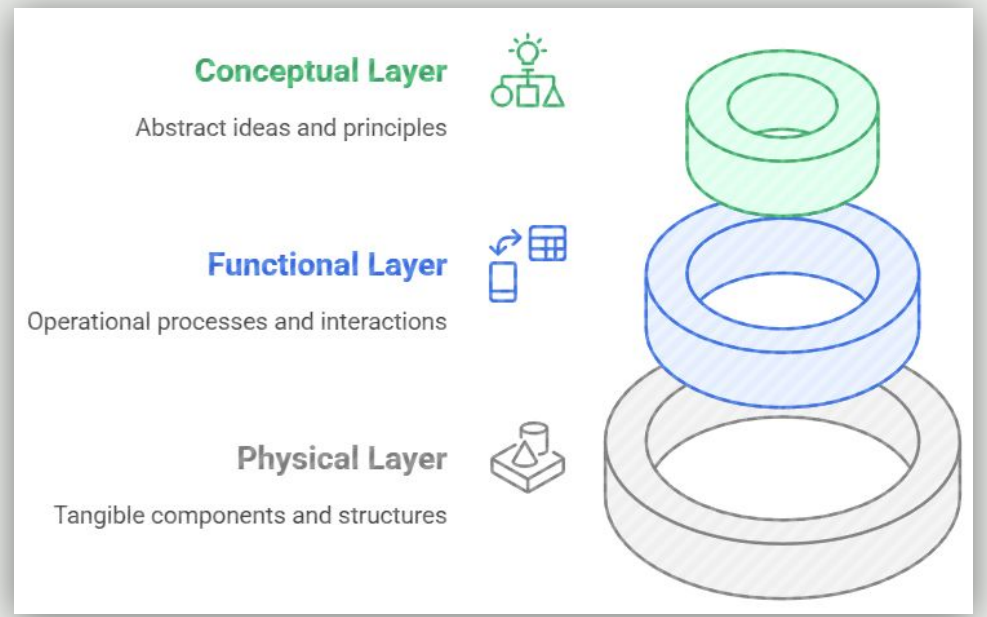
- Enables **Digital Thread** continuity from concept to retirement
- Supports **Digital Twin integration** with live system data
- Open **standard APIs and data formats** for tool interoperability
- Optimized for **automated verification, simulation, and analysis**



Elements

The Building Blocks of a System Model
A SysML model is made up of elements that represent real or abstract parts of a system — from physical hardware to behaviors, constraints, and goals.

- Everything in a model is represented as a **distinct element**
- Elements can be physical (like sensors), functional (like control logic), or conceptual (like requirements)
- Elements are the **core units** used to structure, analyze, and communicate system knowledge

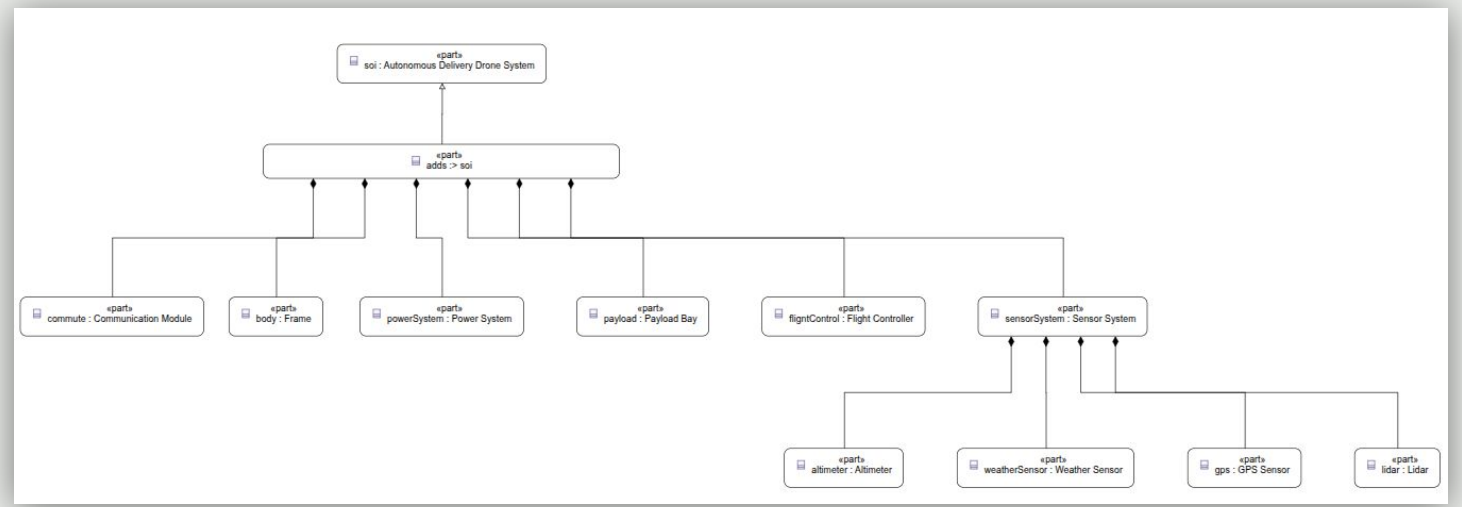


Structure

Modeling What the System *Is*

Structural modeling allows us to define the system's architecture — its parts, interfaces, and how they fit and work together.

- Models the **components** of a system and their **interconnections**
- Essential for understanding **physical layout, interfaces, and system boundaries**
- Forms the **foundation** for analysis, traceability, and simulation

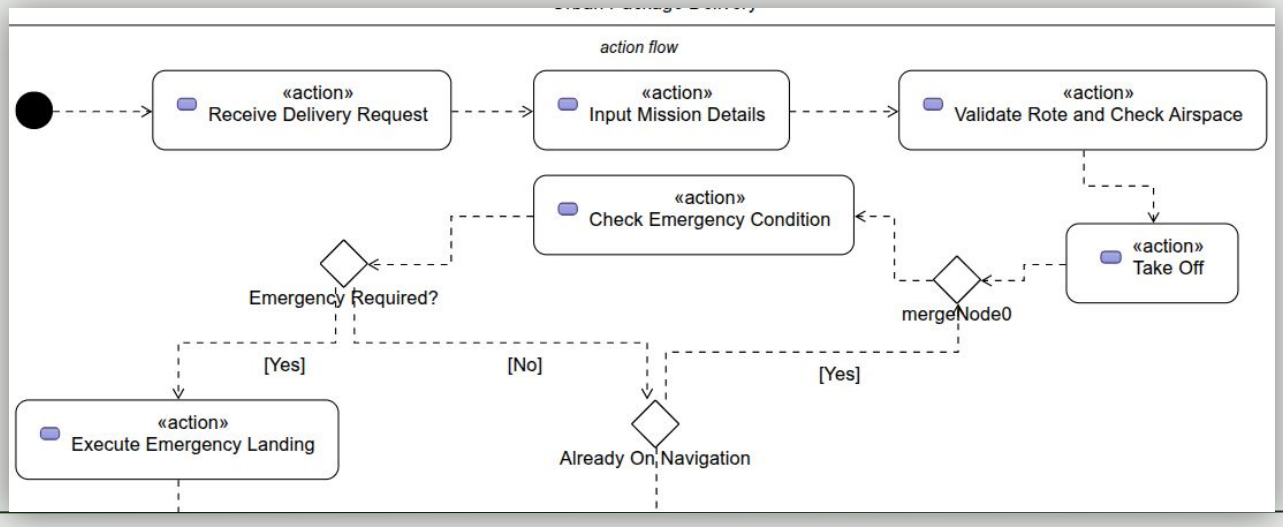


Behavior

Modeling What the System *Does*

While structure defines what a system *is*, behavior modeling describes how it acts, reacts, and evolves over time.

- Captures **operations**, **responses to events**, and **workflows**
- Supports the modeling of **state changes**, **scenarios**, and **missions**
- Crucial for **simulation, control logic, and performance analysis**

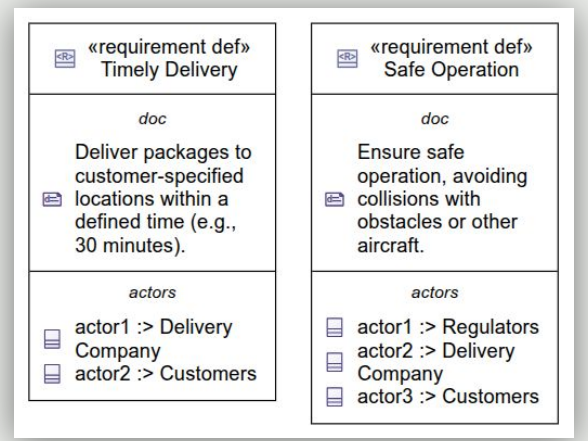


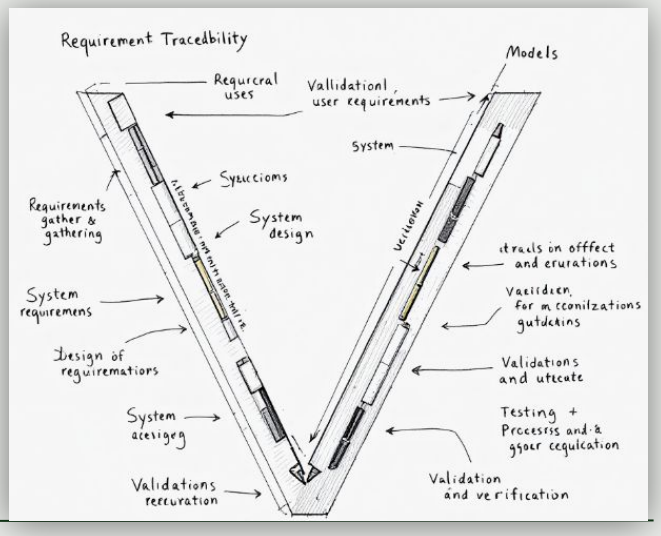
Requirements

Modeling *Why* the System Exists

Requirements define the purpose and constraints of a system — they represent the "why" behind what we build.

- Represent **goals**, **needs**, and **conditions to satisfy**
- Provide a **bridge between stakeholders and engineers**
- Enable **traceability** from concept through implementation and testing



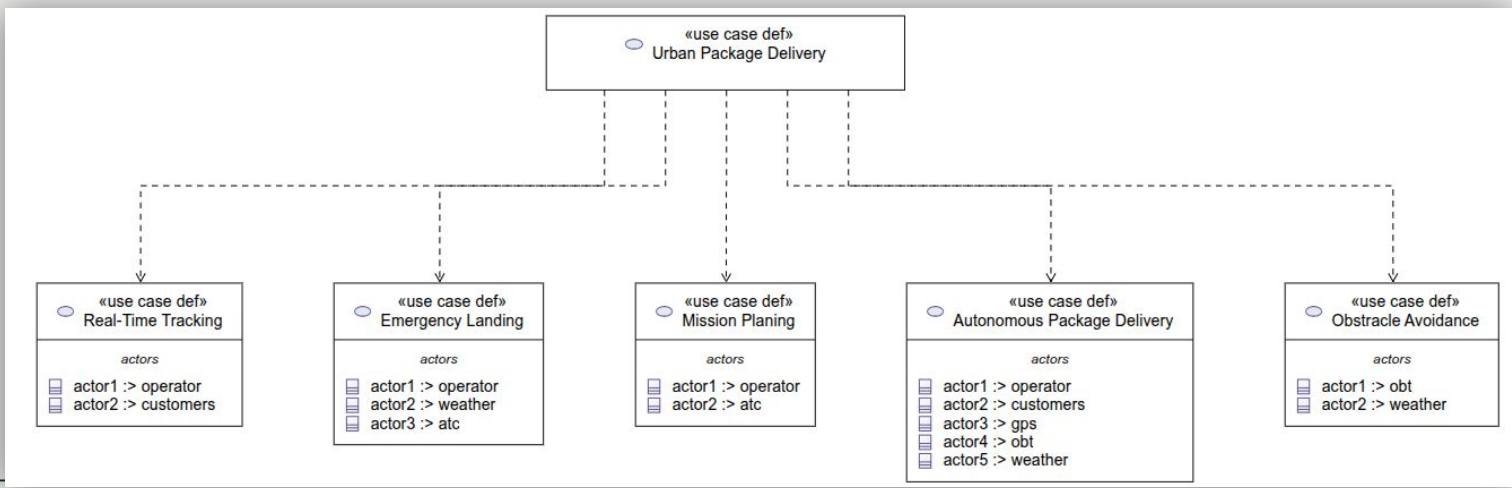


Relationships

Making the Model Cohesive

Relationships connect all parts of the model, enabling consistency, structure, and meaning across system elements.

- Link structure to behavior, behavior to requirements, and more
- Create **traceable, navigable paths** between design decisions
- Support **reuse**, **refinement**, and **impact analysis**



Modeling Domains in SysML v2

In systems engineering, we look at the system from different perspectives — SysML v2 captures these through distinct but connected modeling domains. Each domain offers a specific lens on the system:

- **Structure** — how the system is physically and logically organized
- **Behavior** — how it functions and reacts over time
- **Requirements** — what it must achieve or comply with
- **Constraints / Parametrics** — mathematical and logical limits
- **Verification** — how we know the system meets its purpose

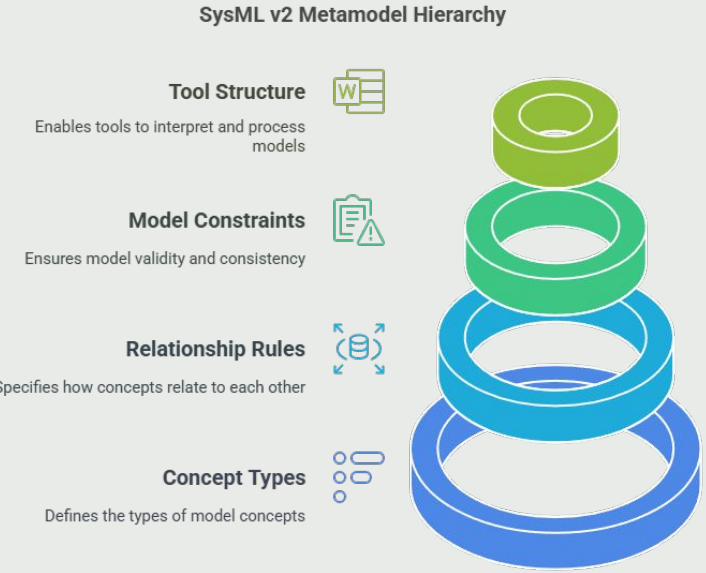
These domains do not exist in isolation — they are **interconnected** and provide a **multi-faceted understanding** of a complex system.

Why a Metamodel Matters

Just like every language needs grammar, every modeling language needs a metamodel. SysML v2's metamodel defines:

- The types of concepts you can model
- The rules for how those concepts relate
- The constraints that ensure your model is valid
- The structure needed for tools to interpret and process your model

Without a metamodel, the model would be just a drawing — not something you can simulate, validate, or automate

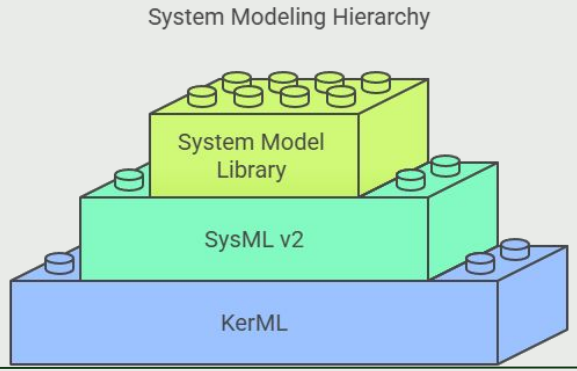


A Layered Foundation

SysML v2 stands on top of a **layered architecture**:

- **KerML**: the abstract modeling language — provides concepts like types, elements, and relationships
- **SysML v2**: specializes KerML for systems engineering
- **System Model Library**: a curated set of reusable engineering model elements like SI units, physical flows, and behavioral patterns

Together, these layers provide a **modular and extensible foundation** for modeling across engineering disciplines.



Why This Structure Powers MBSE

This layered and domain-driven foundation is not just elegant — it's what makes SysML v2 truly work for real-world Model-Based Systems Engineering.

- Supports **cross-domain integration**: structure ↔ behavior ↔ requirements
- Enables **tool interoperability** and **digital thread continuity**
- Makes models **machine-readable**, **queryable**, and **verifiable**
- Facilitates **reuse** and **automation** across the lifecycle

What is an Element Taxonomy?

In SysML v2, the modeling language is organized like a tree — with elements grouped by their nature and purpose. This structure is called the **element taxonomy**, and it helps us:

- Understand what kinds of things we can model
- Know how those things are categorized
- Navigate the modeling space more intuitively

It's like understanding the components of a car: engine, chassis, wheels — all categorized for clarity, even if they work together.

Top-Level Categories of Elements

SysML v2 elements fall into several major categories:

- **Structure Elements** — Represent physical or logical system parts
- **Behavior Elements** — Define actions, interactions, and state changes
- **Requirement Elements** — Capture goals, constraints, and verifications
- **Expression Elements** — Model values, conditions, and formulas
- **Connections and Relationships** — Tie everything together semantically

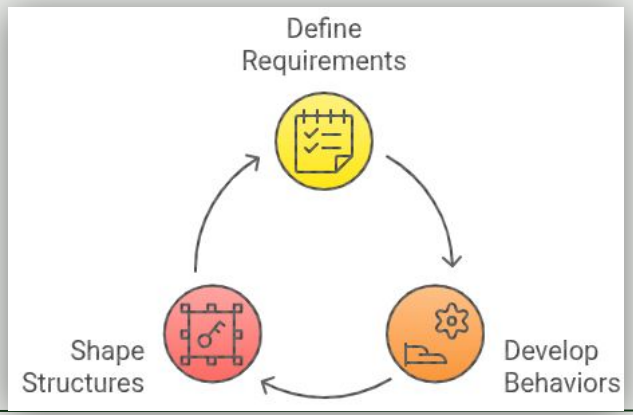
These categories **reflect the modeling domains** we discussed earlier — but here, they're implemented as reusable modeling types.

Elements Are Not Isolated

In real models, elements are connected across categories — forming a unified system model. Example:

- A **requirement** is linked to a **behavior** it constrains
- A **structure** element provides the **context** for an action
- A **formula (expression)** defines how a property changes over time

This **cross-linking** is where the power of SysML v2 emerges — and where automation, simulation, and traceability become possible.



Why the Taxonomy Helps Engineers

- Helps you **choose the right modeling concept** for your intent
- Supports **model validation** by tools
- Makes the model **modular and navigable**
- Enables reuse of common patterns across projects and teams

Think of taxonomy as the **vocabulary shelf** — it helps you speak the modeling language fluently, without confusion.

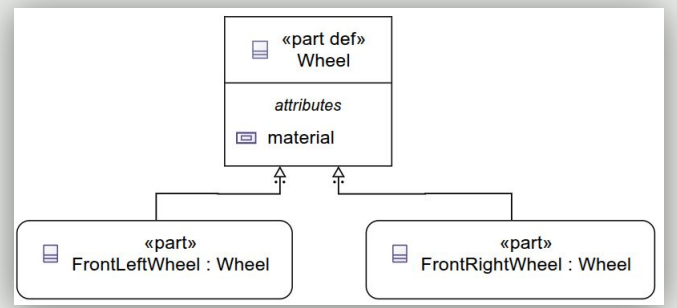
Definition and Usage

SysML v2 separates **“what something is”** from **“how it’s used”** — this principle enables reuse, clarity, and scalability.

- A **definition** describes a general concept, like a *type of part*, *behavior*, or *requirement*
- A **usage** represents a specific application of that definition in a given context
- Every usage must be linked to a definition — either explicitly or implicitly
- This applies across the model: parts, actions, attributes, states, requirements, and more

Example: A *“Wheel”* can be defined once. Then in different vehicle models, you can reuse it as:

- FrontLeftWheel (usage of Wheel)
- RearRightWheel (another usage of Wheel)



What Makes a Good SysML Model

A good SysML model is not just correct — it's purposeful, connected, and usable.

Characteristics of a well-structured model:

- **Cohesive**: All parts of the model align toward a system purpose
- **Consistent**: No contradictions between structure, behavior, or requirements
- **Traceable**: Clear links from needs → design → verification
- **Reusable**: Uses definitions for common elements, avoids duplication
- **Understandable**: Easy for others (and your future self) to navigate
- **Tool-compatible**: Valid and complete for analysis, transformation, or simulation

Tip: Always ask — *"Can this model answer important system questions?"*

SysON Introduction

SysON is a modern, open-source SysML v2 modeling tool — designed for model navigation, creation, and visualization.

In this session, we will:

- Explore the **SysON user interface**
- Learn to navigate models using the **browser and inspector**
- Visualize structural and behavioral diagrams

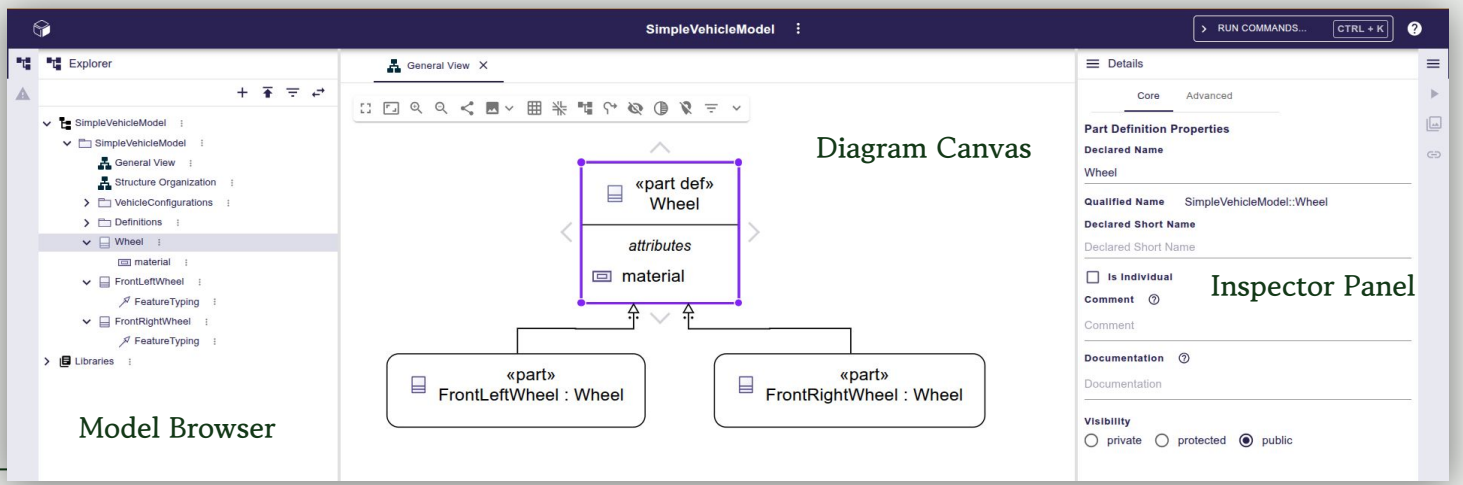
https://mbse-syson.org/

Workspace Orientation

SysON's UI is structured around three core areas: the Browser, the Inspector, and Diagrams.

Key Components:

- **Model Browser**: Explore your system model tree
- **Inspector Panel**: See details of selected elements (type, relations, notes)
- **Diagram Canvas**: View and navigate model structure graphically

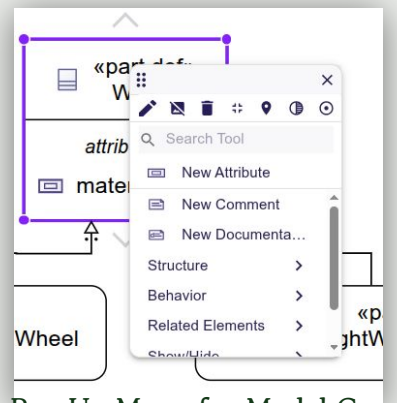


Creating and Inspecting Elements

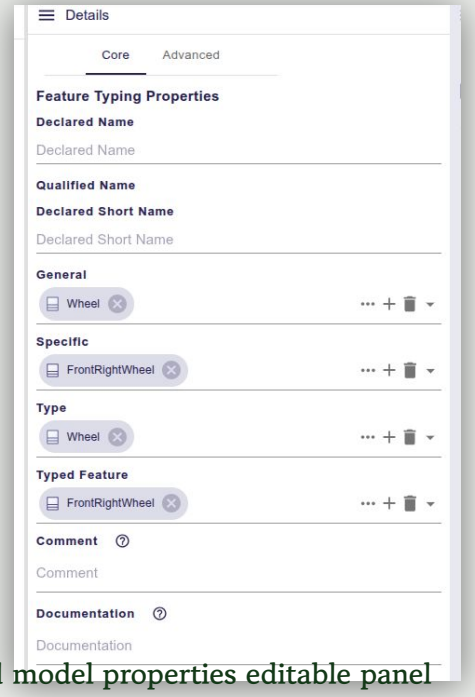
SysON lets you create and explore model elements using guided forms and relationships.

When you select a model element:

- You can **inspect its type and hierarchy**
- You'll see connected elements like **definitions**, **usages**, and **containments**
- You can **add related elements** via GUI tools or command palette



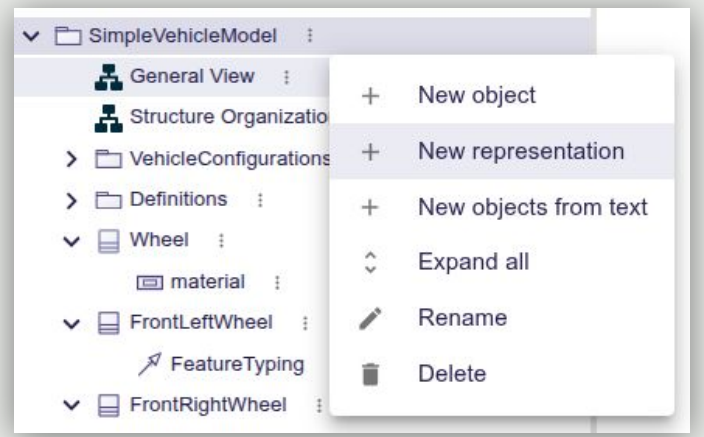
Pop Up Menu for Model Creation



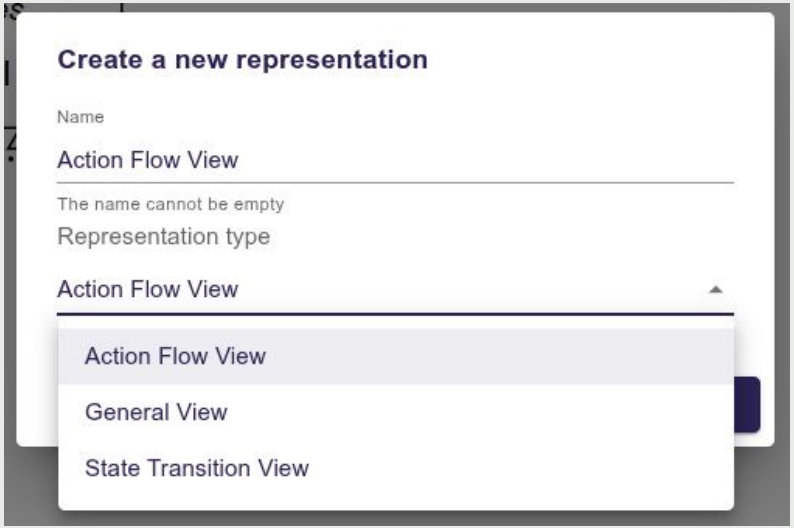
Selected model properties editable panel

Creating Representation Views

In SysON, you can create a new diagram (representation) from the Project Explorer. This opens a blank canvas where you can manually build your model's visual structure.

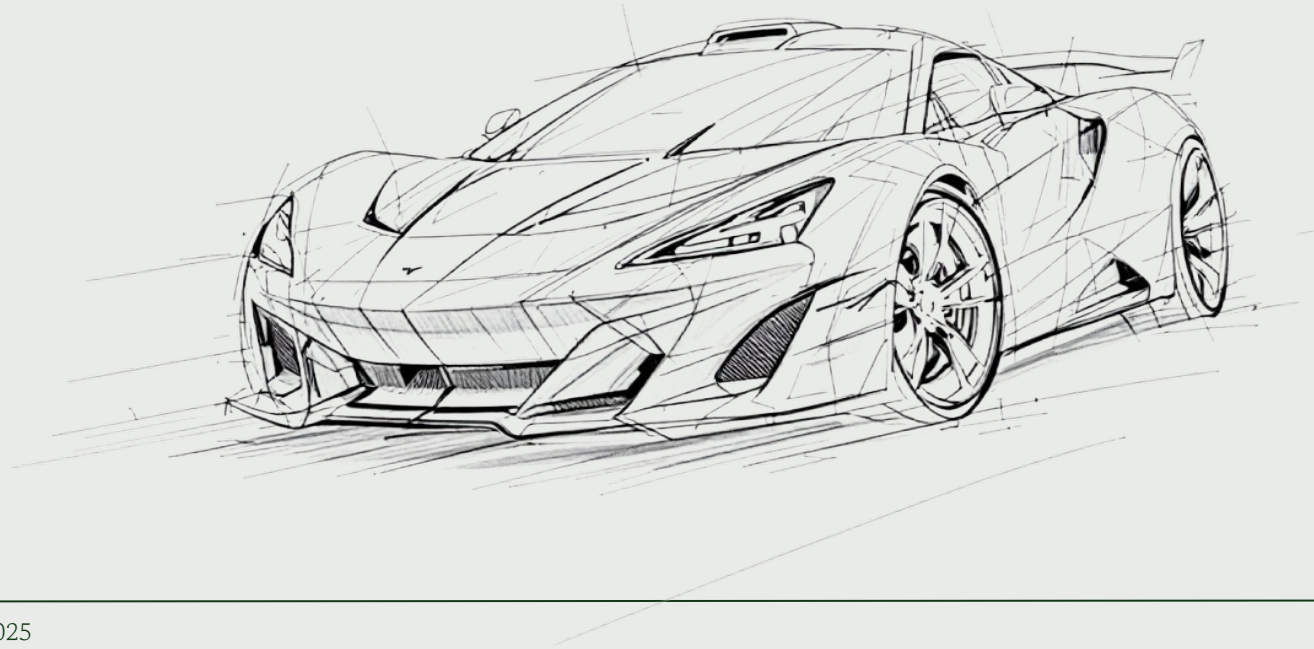


Pop-up menu to create the representation view



Create representation options dialog

Let’s Build It Together



Summary of Week 3

This week, you've built the foundation to think and model like a systems engineer using SysML v2.

Key Takeaways:

- You now understand the **philosophy shift** from SysML v1 to v2
- You've explored the **core modeling concepts**: structure, behavior, requirements, relationships
- You've learned about **modeling domains** and the importance of the **metamodel foundation**
- You now see how SysML v2 elements are **organized and reused** using taxonomy and the definition/usage concept
- You've navigated the SysON tool: browser, inspector, views, and modeling workflow
- You observed a **live demo** building the Simple Vehicle System model

QUESTION!