



Sparx Enterprise Architect

SysML v2 Preview

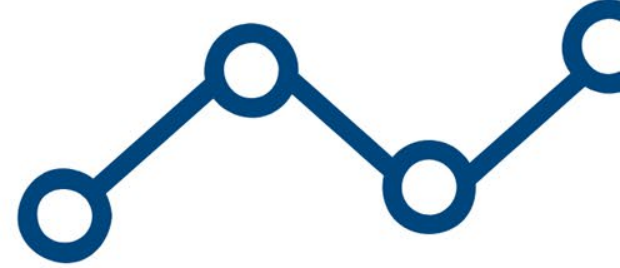
SysML v2 Roadmap



“Align information technology and systems engineering capabilities with business strategy using industry best practices and industry leading tooling to deliver world -class results.”

- ✓ Graphical notation
- ✓ Import models (system libraries, kpar, XMI)
- ✓ Export models
- ✓ Model migration
- ✓ API Services
- ✓ Metadata extensions
- ✓ Behavioral model simulation
- ✓ Mathematical model simulation

Sparx Enterprise Architect SysML v2 Examples



- Model Organization
- Part Definition
- Part Usage
- Individuals and Snapshots
- Dynamic Views
- Custom Views

Model Organization

The screenshot displays the Enterprise Architect interface with a SysML model diagram titled "Figure 55. Model Organization for SimpleModel. Custom Diagram". The diagram shows a hierarchical structure of packages:

- SimpleVehicleModel** (Root Package)
 - VehicleConfigurations** (Package)
 - VehicleConfiguration_a** (Package)
 - VehicleConfiguration_b** (Package)
 - PartsTree** (Package)
 - ActionTree** (Package)
 - Requirements** (Package)
 - VehicleIndividuals** (Package)
 - VehicleAnalysis** (Package)
 - VehicleVerification** (Package)
 - Views_Viewpoints** (Package)
 - ISQ** (Package, shown with a dashed border)

Relationships are indicated by dashed arrows labeled "import**".

The interface includes a top menu bar (Start, Design, Layout, Develop, Simulate, Execute, Construct, Specialize, Publish, Settings, Find Command), a ribbon with various tool icons, a left-hand Browser pane showing a project tree, a right-hand Properties pane with tabs for General, Diagram, Compartments, Objects ZOrder, and Documentation, and a bottom status bar with file information.

Part Definition

The screenshot displays the Enterprise Architect interface with a SysML Part Definition for a Vehicle. The main diagram area shows the following structure:

```
part def
  Vehicle
  attributes
  mass -> mass
  dryMass -> mass
  cargoMass -> mass
  position -> length
  velocity -> speed
  acceleration -> acceleration
  electricalPower -> power
  Tmax -> thermodynamicTemperature
  maintenanceTime -> DateTime
  brakePedalDepressed -> Boolean
  exhibit states
  vehicleStates
  perform actions
  providePower
  provideBraking
  controlDirection
  performSelfTest
  applyParkingBrake
  senseTemperature
  ports
  ignitionCmdPort : IgnitionCmdPort
  pwrCmdPort : PwrCmdPort
  vehicleToRoadPort : VehicleToRoadPort
  statusPort : StatusPort
```

The left sidebar shows a project browser with a tree view including 'Standard Libraries', 'Imported Models', 'Example Model', 'Testing', 'Brad', 'Brendan', 'Eye', 'Figure 60. Part Usage for vehicle_b', 'MultipleEnds', 'Package1', 'SysML 2 Overview', 'Dynamic Models', 'User Interface', 'Icons', 'Properties Window', 'Properties Window', 'model view Live Examp', 'Image Properties-Attr', 'Image Properties-Calcu', 'Custom SysML Views', 'SysML 2 Overview', 'Miles', 'Neil', 'Nethya', 'Demonstrations', 'SysML 2 Overview', 'Graphical Notation', 'Graphical Notation', 'Dynamic Models', and 'User Interface'.

The right sidebar shows the 'Properties' panel with 'General' and 'Project' sections, and a 'Documentation' panel with text formatting options.

At the bottom, the status bar indicates: 'Custom Diagram:Figure 56. Part Definition for Vehicle: created: 15/01/2024 2:36:20 pm modified: 15/01/2024 3:26:43 pm 125% 826 x 1169'.

Part Usage

The screenshot displays the Enterprise Architect interface with a SysML diagram titled "Figure 60. Part Usage for vehicle_b". The diagram is a SysML Part Definition Diagram (PDD) for a component named "vehicle_b".

Diagram Content:

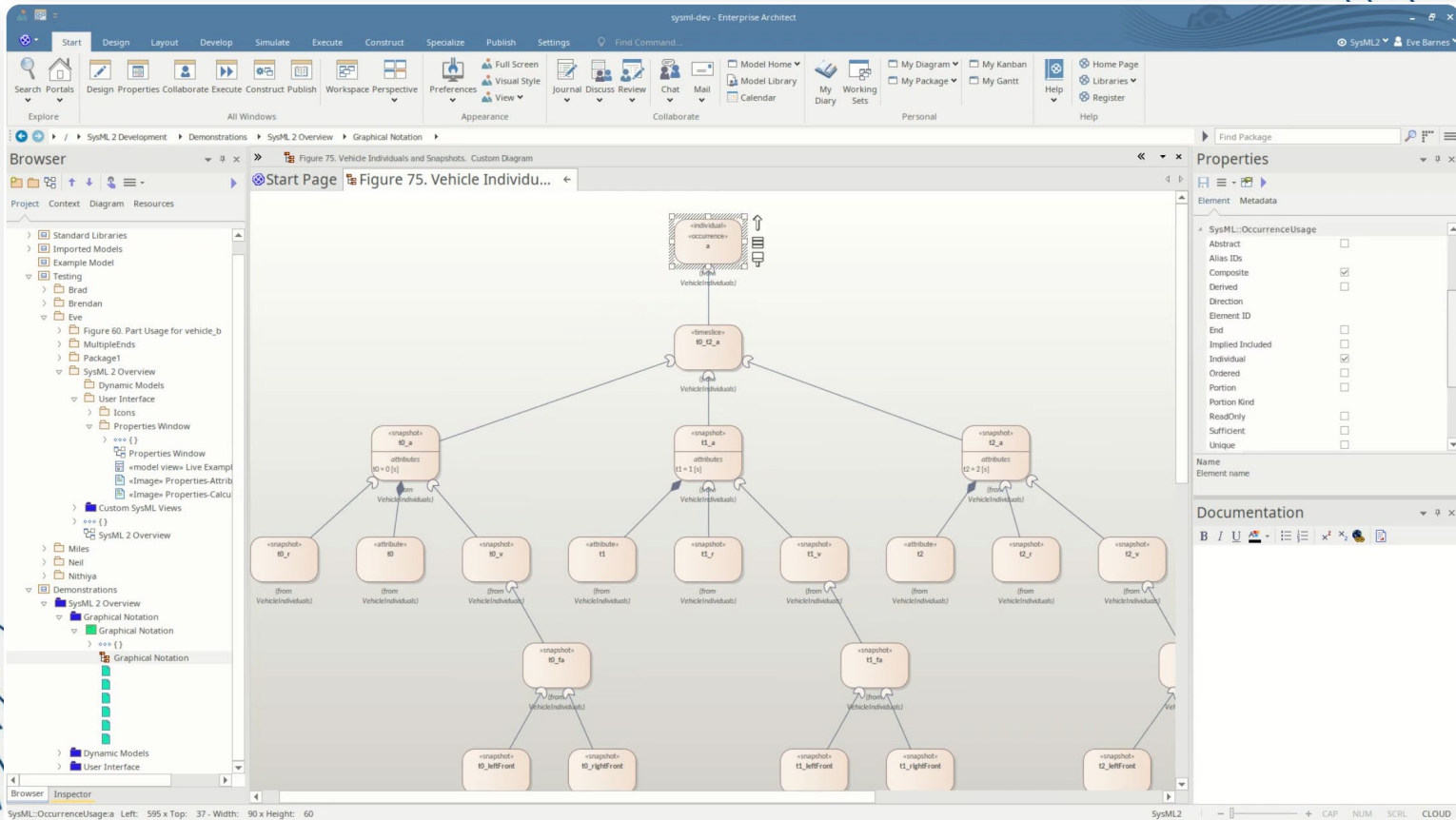
- Attributes:**
 - `#mop mass >> Vehicle::mass = dryMass+cargoMass+fuelTank.fuel.fuelMass`
 - `dryMass >> Vehicle::dryMass = sum(part::Masses)`
 - `cargoMass >> Vehicle::cargoMass default = 0 [kg]`
 - `part::Masses = fuelTank.mass.frontAxeAssembly.mass.rearAxeAssembly.mass.engine.mass.transmission.mass.driveshaft.mass`
 - `avgFuelEconomy >> distancePerVolume`
- Parts:**
 - `FuelTank : FuelTank`
 - `frontAxeAssembly : AxeAssembly`
 - `rearAxeAssembly : AxeAssembly`
 - `starterMotor : StarterMotor`
 - `engine : Engine`
 - `transmission : Transmission`
 - `driveshaft : Driveshaft`
 - `vehicleSoftware : VehicleSoftware`
 - `speedSensor : SpeedSensor`
 - `bodyAssy : BodyAssy`
 - `interior`
- Exhibit States:**
 - `vehicleStates >> Vehicle::vehicleStates`
- Perform Actions:**
 - `providePower >> Vehicle::providePower`
 - `performSelfTest >> Vehicle::performSelfTest`
 - `applyParkingBrake >> Vehicle::applyParkingBrake`
 - `senseTemperature >> Vehicle::senseTemperature`
- Ports:**
 - `fuelCmdPort : FuelCmdPort >> Vehicle::pwCmdPort`
 - `setSpeedPort : SetSpeedPort`
 - `vehicleToRoadPort >> Vehicle::vehicleToRoadPort`

Properties Panel:

- General:** Name: vehicle_b, Type: SysML::PartUsage, Stereotype: SysML::PartUsage, Alias: SysML::PartUsage, Keywords: Status: Proposed, Version: 1.0.
- Documentation:** Includes formatting options (Bold, Italic, Underline, Text Color, Background Color, Bulleted List, Numbered List, Indentation, Font Size, Copy, Paste, Undo, Redo).

Browser Panel: Shows a tree view of the project structure, including "SysML 2 Overview", "Dynamic Models", "User Interface", and "Graphical Notation".

Individuals and Snapshots



Dynamic Views – Specification

The screenshot displays the Sparx Systems Enterprise Architect interface. The main workspace shows a SysML diagram titled "Dynamic SysML 2 Views". The diagram includes several elements and relationships:

- Class: SysML Metaclasses** (stereotype: <code><view></code>): Contains a note: "SysML Metaclasses exposes the Namespace containing the SysML 2 reflexive model. The NamespaceExpose is indicated by the <code><expose></code> keyword with a trailing <code>'</code>. Additionally, it is typed by a ViewDefinition with a filter, which means it will only show elements matching that filter." Below this is another note: "Any elements matching the filter within the specified Namespaces that are added or removed will update the diagram automatically."
- Class: Systems** (stereotype: <code><system></code>): Represented as a package.
- Class: <code><view def: MetaclassHierarchyView</code>** (stereotype: <code><view def: MetaclassHierarchyView</code>): Contains a note: "(from MetamodelReference)".
- Class: <code><operator expression></code>** (stereotype: <code><operator expression></code>): Contains a note: "(from MetamodelReference)".
- Class: <code><rep></code>** (stereotype: <code><rep></code>): Contains a note: "@Metaclass or @MetadataDefinition or @Subclassification".

Relationships shown in the diagram:

- A dashed association line from **SysML Metaclasses** to **Systems** with the stereotype <code><expose></code>.
- A solid association line from **SysML Metaclasses** to **<code><view def: MetaclassHierarchyView</code>** with the stereotype <code><expose></code>.
- A solid association line from **<code><view def: MetaclassHierarchyView</code>** to **<code><operator expression></code>** with the stereotype <code><element filter membership></code>.
- A solid association line from **<code><operator expression></code>** to **<code><rep></code>** with the stereotype <code><rep></code>.

The interface also shows a left-hand browser, a top menu bar, and a right-hand properties and documentation pane. The documentation pane contains the following text:

Documentation

SysML Metaclasses exposes the Namespace containing the SysML 2 reflexive model. The NamespaceExpose is indicated by the <code><expose></code> keyword with a trailing <code>'</code>. Additionally, it is typed by a ViewDefinition with a filter, which means it will only show elements matching that filter.

Dynamic Views – Generation

The screenshot shows the Sparx Systems Enterprise Architect interface. The main workspace displays a SysML Metaclasses diagram with the following elements:

- StateSubactionMembership** (metadata def): attributes: kind: Systems:StateSubactionKind -> Base:dataValues [1..1] (from Systems)
- TransitionFeatureMembership** (metadata def): attributes: kind: Systems:TransitionFeatureKind -> Base:dataValues [1..1] (from Systems)
- EventOccurrenceUsage** (metadata def): attributes: isReference: ScalarValues:Boolean -> Usage:isReference [1..1] (from Systems)
- ActionUsage** (metadata def): (from Systems)
- PerformActionUsage** (metadata def): (from System)
- StateUsage** (metadata def): attributes: isParallel: ScalarValues:Boolean -> Base:dataValues [1..1] (from Systems)
- CalculationUsage** (metadata def): (from Systems)
- LoopActionUsage** (abstract metadata def): (in System)
- ExhibitStateUsage** (metadata def): (from System)
- CaseUsage** (metadata def): (from Systems)
- WhileLoopActionUsage** (metadata def): (from System)
- ForLoopActionUsage** (metadata def): (from System)

The Properties panel on the right shows the following details for SysML:ViewUsage:

- General: Name: SysML MetaClasses, Type: SysML:ViewUsage, Stereotype: SysML:ViewUsage, Alias: SysML:ViewUsage, Keywords: SysML:ViewUsage, Status: Proposed, Version: 1.0
- SysML:ViewUsage: Abstract: , Alias IDs: , Composite: , Derived: , Direction: , Element ID: , End: , Implied Included: , Individual: , Ordered: , Portion: , Portion Kind: , Read Only: , Sufficient: , Unique: , Variation:
- Project:

Custom View – Specification

This is how the base hierarchy view is defined. It's a ViewDefinition element with an ElementFilterMembership to an expression, which defines which elements should be shown.

This custom views created in this demonstration will expose elements from

New Diagram
 Package: Example ViewDefinition
 Diagram: KerML and SysML abstract Metaclasses
 Select From: SysML2
 Diagram Types: Standard Libraries::sysml-StandardViewDefinitions, Imported Models::sysml-examples-SimpleVehicleModel-Views_Viewpoi..., Imported Models::sysml-training-Views Example, Imported Models::sysml-validation-11b-Safety and Security Feaure Vie..., Standard Libraries::sparx-SparxViewDefinitions, Imported Models::sparx-examples-MetamodelReference, Demonstrations::SysML 2 Overview::Dynamic Models::Example ViewDef..., Abstract Metaclasses, ViewUsage
 This view specification shows only metaclasses that are abstract. It extends MetaClassHierarchyView to get the type constraints, then adds the constraint that they must be abstract.



“Align systems engineering and information technology capabilities with business strategy using industry best practices and industry leading tooling to deliver world -class results.”

Thanks for your time and attention!

Chris Armstrong, President/Chief Architect
chris.armstrong@sparxsystems.us

**Providing a common
platform for engineering
collaboration**