


Predefined Relation Maps

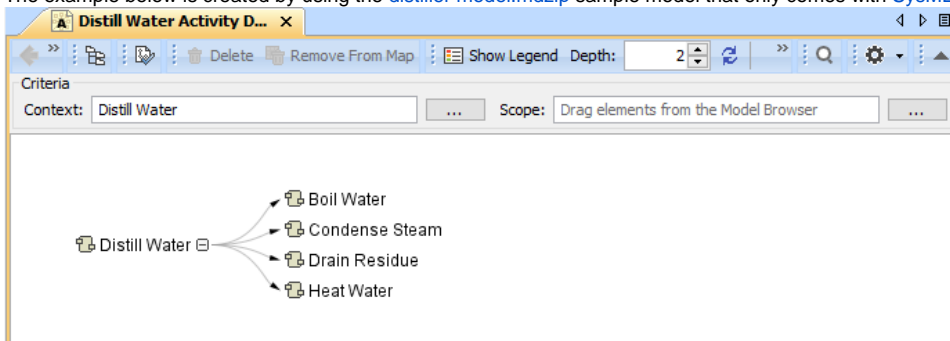
You can use all predefined relation maps to represent the traceability of system requirements and design elements. The main purpose of relation maps is to review and analyze relations among the elements and create new elements directly in the relation map. It is a special kind of diagram that automatically updates and renders an element's dependency tree according to predefined dependency criteria. You can create five kinds of predefined relation maps:

- [Activity Decomposition Map](#).
- [Structure Decomposition Map](#).
- [Instance Map](#).
- [Requirement Containment Map](#).
- [Requirement Derivation Map](#).

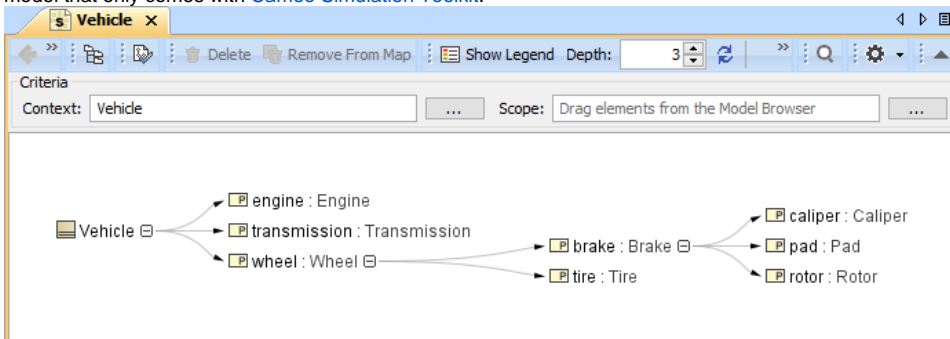
 You can create predefined relation maps only if you have the [SysML Plugin](#) installed. [How to install SysML Plugin >>](#)

The different purposes for each relation map are illustrated below:

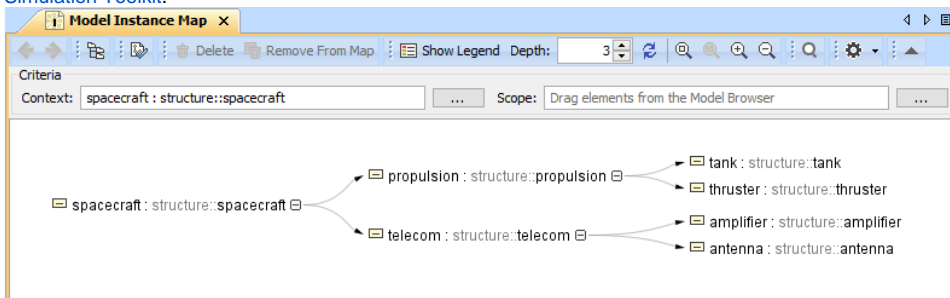
- **Activity Decomposition Map** displays an Activity decomposition of the selected context. You can review, analyze, and decompose the Activities. The example below is created by using the [distiller model.mdzip](#) sample model that only comes with [SysML Plugin](#).



- **Structure Decomposition Map** displays the decomposition of the [Part Properties](#) related through the [Composition relationship](#) of the specified context. You can review, analyze, and decompose the Part Properties. The example below is created by using the [VehicleStructure.mdzip](#) sample model that only comes with [Cameo Simulation Toolkit](#).



- **Instance Map** displays a hierarchy of the [Instance Specifications](#) of the selected context. You can review and analyze the hierarchy of Instance Specifications. The example below is created by using the [SpacecraftMassRollup_HTMLTable.mdzip](#) sample model that only comes with [Cameo Simulation Toolkit](#).



- **Requirement Containment Map** displays the decomposition of the [Requirements](#) related through the [Containment relationship](#) of the specified context. You can review, analyze, and decompose the Requirements. The example below is created by using the [hybrid sport utility vehicle.mdzip](#)

Criteria

Context: ... Scope: ...

```

graph LR
    HSUV[HSUV Specification] --> R4[R 4 Capacity]
    HSUV --> R1[R 1 Eco-Friendliness]
    HSUV --> R3[R 3 Ergonomics]
    HSUV --> R2[R 2 Performance]
    HSUV --> R5[R 5 Qualification]
    R4 --> R41[R 4.1 CargoCapacity]
    R4 --> R42[R 4.2 FuelCapacity]
    R4 --> R43[R 4.3 PassengerCapacity]
    R1 --> R121[R 1.2.1 Emissions]
    R3 --> R24[R 2.4 Acceleration]
    R2 --> R21[R 2.1 Braking]
    R2 --> R22[R 2.2 FuelEconomy]
    R2 --> R23[R 2.3 OffRoadCapability]
    R5 --> R51[R 5.1 SafetyTest]
  
```

-
- The screenshot shows the Hsuv Specification Requirements tool interface. The left pane displays the 'Criteria' tree with 'Performance' selected. The main area shows a hierarchical diagram of requirements. The root node is 'Performance' (R 2), which is an owned member of the 'Criteria' set. It branches into four sub-requirements: 'Acceleration' (R 2.4), 'Braking' (R 2.1), 'FuelEconomy' (R 2.2), and 'OffRoadCapability' (R 2.3). 'Acceleration' and 'OffRoadCapability' are derived from 'Performance' and both point to 'Power' (R d.4). 'Braking' is an owned member of 'Performance' and points to 'RegenerativeBraking' (R d.1). 'FuelEconomy' is an owned member of 'Performance' and points to 'PowerSourceManagement' (R d.3), 'Range' (R d.2), and 'RegenerativeBraking' (R d.1). 'Power' (R d.4) is a derived requirement that points to 'PowerSourceManagement' (R d.3). A legend indicates that blue lines represent 'Derived' relationships and green lines represent 'Owned Member' relationships.

[Lean more about Relation Map Diagram >>](#)

- Creating relation map
- Specifying criteria, layout, and depth
- Changing the context
- Cutting element names
- Showing single node per element
- Displaying element numbering
- Specifying colors and legend
- Manipulations in the relation map

\$body