

Capella to SysML Viewpoint

Documentation

| SIGNATURES | | | | |
| --- | --- | --- | --- | --- |
|  | Name | Title | Signature | Date |
| Written by | Yann Binot |  |  |  |
| Verified & authorized by | B. VIAUD | System Business Unit Manager |  |  |

|  |  |  |
| --- | --- | --- |
|  | |  |
| REVISIONS | | |
| Ed1 | Creation | |

[1 Introduction 4](#_Toc27391005)

[1.1 Document contents and organization 4](#_Toc27391006)

[2 Installation procedure 5](#_Toc27391007)

[3 User Interfaces 6](#_Toc27391008)

[3.1 Import SysML 6](#_Toc27391009)

[3.1.1 Pre-requisite 6](#_Toc27391010)

[3.1.2 SysML Preferences 6](#_Toc27391011)

[3.1.3 Import SysML wizard 7](#_Toc27391012)

[3.1.4 Iterative import 8](#_Toc27391013)

[3.2 Export SysML 8](#_Toc27391014)

[3.2.1 Export SysML wizard 8](#_Toc27391015)

[3.2.2 Iterative export 10](#_Toc27391016)

[4 Mapping rules SysML / Capella 11](#_Toc27391017)

[4.1 Activities 11](#_Toc27391018)

[4.2 Blocks 13](#_Toc27391019)

[4.2.1 Structure blocks 13](#_Toc27391020)

[4.2.2 Parametric blocks 15](#_Toc27391021)

[4.3 Actors 16](#_Toc27391022)

[4.4 Use Cases 16](#_Toc27391023)

[4.5 Requirements 17](#_Toc27391024)

[4.6 Constraints 17](#_Toc27391025)

[4.7 Value Types 18](#_Toc27391026)

[4.8 State / Mode 18](#_Toc27391027)

# Introduction

## Document contents and organization

The goal of this document is both:

* To describe the installation procedure as well as the user guide for importing / exporting the SysML model into / from Capella,
* To describe the mapping rules between SysML and Capella.

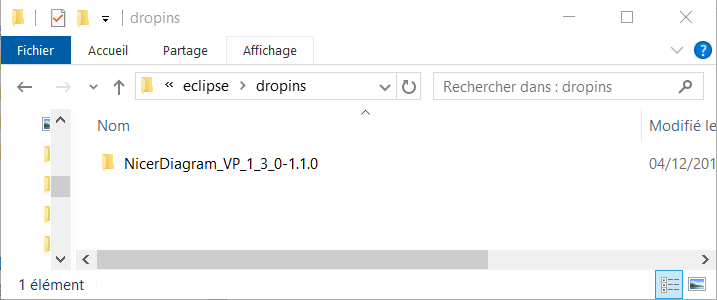
|  |  |
| --- | --- |
|  |  |

# Installation procedure

The Capella2SysML add-on in the dropin format allows to install the extension quickly.

Unzip the extension CapellaSysMLBridge\_1\_2\_1-1.0.7-dropin.zip in the dropins folder of the capella installation repository.

Launch Capella\_1.3.1. The CapellaSysMLBridge extension is installed.



# User Interfaces

## Import SysML

### Pre-requisite

#### UML input file.

Capella user need to have an UML input file containing the SysML model to import.

This input file is generated from Cameo System Modeler by an export of the cameo model in Eclipse UML2 (v5.x) XMI file format.

#### XML configuration file

In case the SysML model structure is different from the supported default structure, the Capella user needs to provide a XML configuration file that specify where to find the data in the SysML model:

The XML configuration file is structured as described below:

*<?xml version="1.0" encoding="UTF-8"?>*

*<configurations>*

*<useCasesPath path="02 Behavior/02 Use Cases" />*

*<activitiesPath path="02 Behavior/02 Functional Architecture" />*

*<partPath path="03 Structure/Parts" />*

*<productPath path="03 Structure/Product" />*

*<parametricPath path="04 Parametric" />*

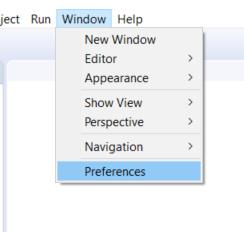
*</configurations>*

* *useCasePath* provides the UseCase package containing actors and use cases
* activitiesPath provides the root activity containing all the CallBihaviourAction referencing Activities.
* *partPath* provides the Part package path containing the Block.
* *productPath* provides the the root Block.

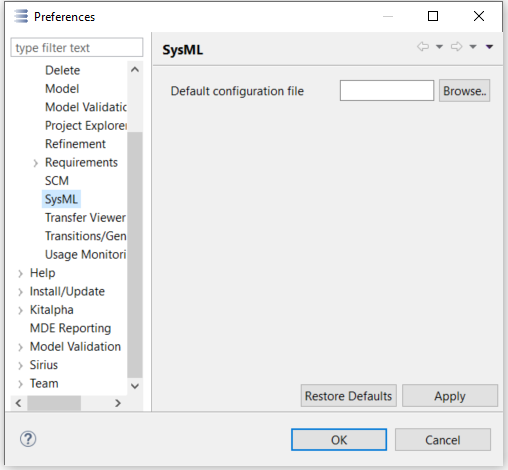
### SysML Preferences

The Capella user can select this xml configuration file in the Capella preferences. This way, the Capella user doesn’t need to set the xml configuration file at each launch.

Click on the windows -> preferences Capella menu.



In the Preferences wizard, select the Capella -> SysML preference.

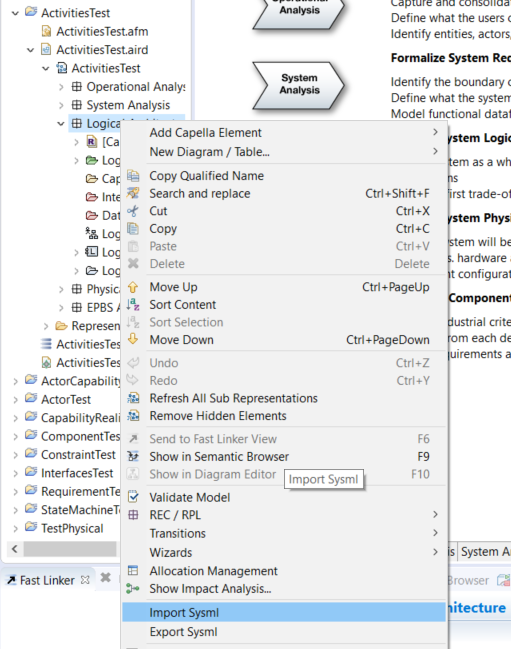


In the Default configuration file field, set a new XML configuration file.

Click Apply and OK button.

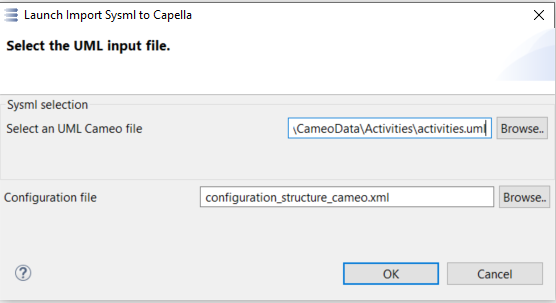
### Import SysML wizard

Right click on the logical architecture layer in the Capella project explorer, then in the contextual menu, select the menu “Import SysML”.

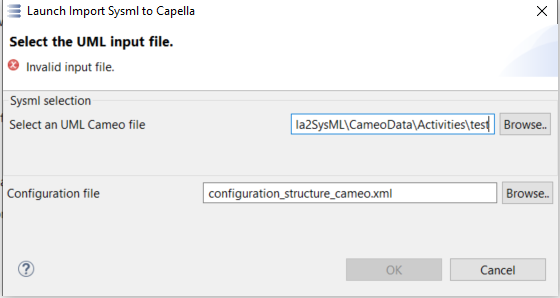


A window requests the user to set two inputs:

* Select an UML Cameo file
* Configuration file



If input file has not an UML extension or doesn’t exist, the OK button is not available and an error message informs that input file is invalid.

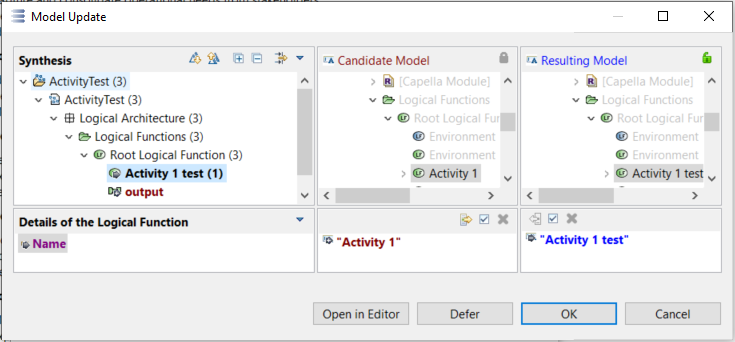


### Iterative import

The user can import several times the same SysML model (with local modifications) in the same Capella model.

In this case, a merge wizard is displayed and shows the differences between the previous import and the new import.

If there are conflicts, the wizard proposes to remove or to keep the new modifications.

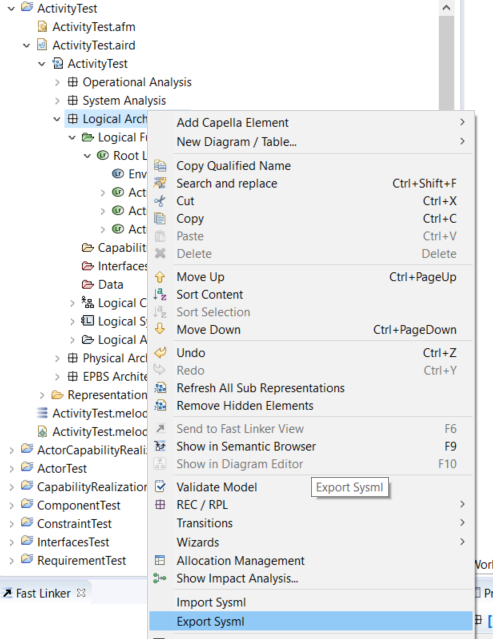


## Export SysML

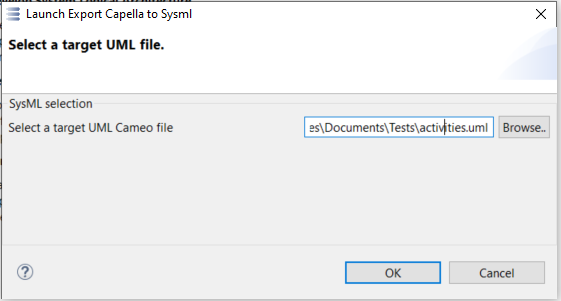
### Export SysML wizard

Right click on the logical architecture layer in the Capella project explorer.

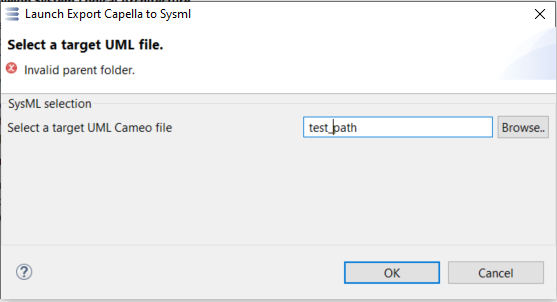
In the contextual menu, select the menu “Export SysML”.



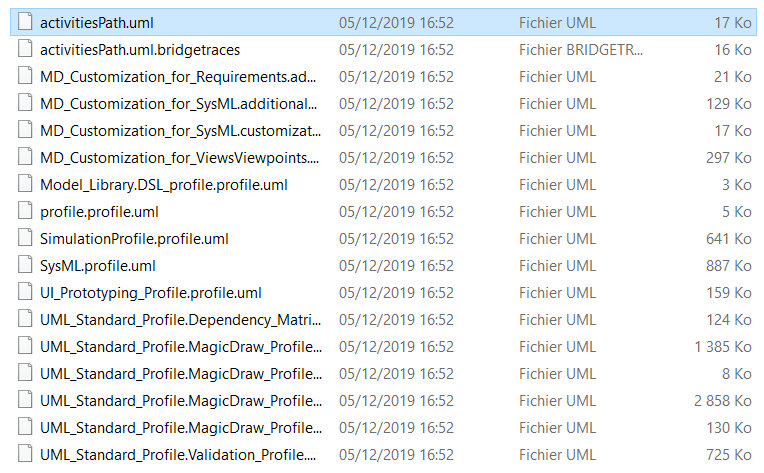
A window requests the user to select a target UML Cameo file:



If the target uml file doesn’t exist, the OK button is not available and an error message informs that there is not input file defined.



All the profiles files used in the exported Cameo UML file are generated in the same directory.

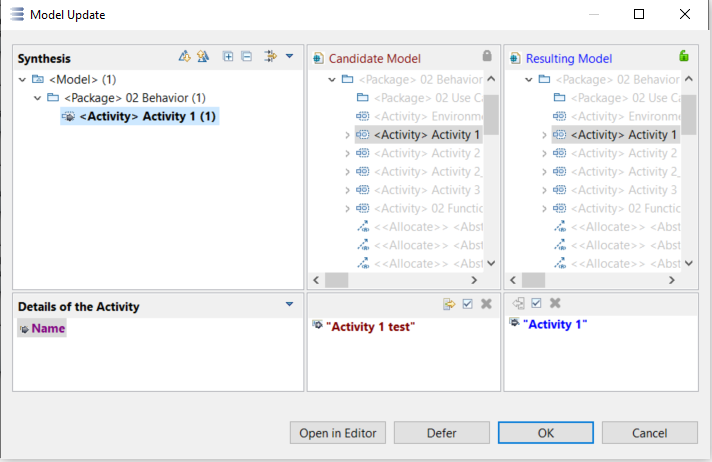


### Iterative export

The user can export several times the same Capella model (with local modifications) in the same SysML model.

In this case, a merge wizard is displayed and shows the differences between the previous export and the new export.

If there are conflicts, the wizard proposes to remove or to keep the new modifications.



# Mapping rules SysML / Capella

## Activities

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Activity  Container : Behavior package | LogicalFunction  Container : Logical Functions package |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| CallBehaviorAction  Container : Activity | Logical Function  Container : Logical Function |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| ActivityNodeParameter, Pin  **Direction** :  - CallBehaviorAction results OutputPin  - CallBehaviorAction arguments InputPin  **Container** : Activity | Function Port  **Direction**:   * FunctionOutPutPort * FunctionInputPort   **Container** : Logical Function |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| ObjectFlow  **Source:** ActivityNodeParameter or Pin  **Target:** ActivityNodeParameter or Pin  **Container** : Activity | FunctionalExchange  **Source:** FunctionPort  **Target**: FunctionPort  **Container** : Logical Function |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| ObjectFlow  **Source:** ActivityNodeParameter or Pin  **Target:** ActivityNodeParameter or Pin  **Container** : Activity | FunctionalExchange  **Source:** FunctionPort  **Target**: FunctionPort  **Container** : Logical Function |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Abstraction (Allocation stereotype)  **Client**: Activity  **Supplier**: Class (Block stereotype)  **Container** : Behavior package | ComponentFunctionalAllocation  **Target**: LogicalFunction  **Source**: LogicalComponent  **Container** : LogicalComponent |

## Blocks

### Structure blocks

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Class (Block stereotype)  **Attribute**: property (part stereotype)  Property (constraint stereotype)  **Container** : Structure package | Logical Component  **Feature**: Part  **ownedExtension**: constraint  **Container** : Logical System (LogicalComponent) |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Port  **Type**: Class (Interface stereotype)  **Direction**:  type.getdirection = in  isConjugated = true direction = out  isConjugated = false direction = in  type.getdireciton = out  isConjugated = true direction = in  isConjugated = false direction = out  **Container** : Class (block stereotype) | ComponentPort  **RequiredInterface:**  Interface  **ProvideInterface:** Interface  **Direction:** in or out  **Container** : LogicalComponent |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Connector (Binding connector stereotype)  **Ends:** Port  **Type**: port.getType= Class (Interface Stereotype)  **Container** : Class (block stereotype) | ComponentPort  **Source:** ComponentPort  **Target:** ComponentPort  **ConvoyedInformation:** ExchangeItem  **Container** : LogicalComponent |

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Connector (Binding connector stereotype)  **Ends:** Port  **Type**: port.getType= Class (Interface Stereotype)  **Container** : Class (block stereotype) | ComponentPort  **Source:** ComponentPort  **Target:** ComponentPort  **ConvoyedInformation:** ExchangeItem  **Container** : LogicalComponent |

### Parametric blocks

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Class (block stereotype)  **Attributes**:   * Property (Value stereotype) * Property (Constraint stereotype)   **Container** : parametric package | Class  **feature**: Property  **ownedExtension**: Constraint  **Container** : Data package |

**Note:** The Cameo ConstraintBlocks are transformed to constraint in Capella. The ConstraintBlock concept doesn’t exist in Capella. In direction Capella to Cameo transformation, the constraint block is lost.Interfaces

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Class (Interface stereotype)  **Attribute**: Property (Flow stereotype) type Class (block)  **Container** : parametric package | Interface references ExchangeItem  **ExchangeItemElement:** Class  **ExchangeItem kind** : Flow  **Container** :  Interface -> Interfaces package  ExchangeItem -> Data package |

## Actors

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Actor  **Container** : UseCase package | LogicalActor and Part  **Container** :  Actor -> LogicalActor package  Part -> LogicalContext |

## Use Cases

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| UseCase  **Includes**: Include  **Associations**: Association  **Ends** : Actor and UseCase  **Container** : UseCase package | CapabilityRealization  **Include**: AbstractCapabilityInclude  **ActorCapabilityRealizations**: ActorCapabilityRealization  **Involved**: LogicalActor  **Container**: CapabilityRealization  **Container** :  Capabilities package |

## Requirements

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Requirement  **Trace:** Abstraction(Trace stereotype)  **Satisfy:** Abstraction (Satisfy stereotype)  **Refine:** Abstraction (Refine stereotype)  **Allocate:** Abstraction (Allocate stereotype)  **Container** : Requirement package | Requirement  **Requirement Allocation**: CapellaRelation  **Source**: Requirement or CapellaElement  **Target**: Requirement or CapellaElement  Capella relation type -> RelationType.Trace  Capella relation type -> RelationType.Satisfy  Capella relation type -> RelationType.Refine  Capella relation type -> RelationType.Allocate  **Container** : CapellaModule |

## Constraints

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| Constraint  **Specification**: OpaqueExpression  **Bodies**: String expression  **Langage**: langage expression  **Container** : Cameo element | Constraint  **Specification**: OpaqueExpression  **Bodies**: String expression  **Langage**: langage expression  **Container** : CapellaElement |

## Value Types

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| DataType (ValueType stereotype)  **Attributes**: Property  **Container** : Parametric package | Class primitive  **Features**: Property  **Container** : Data package |

## State / Mode

|  |  |
| --- | --- |
| **Cameo** | **Capella** |
|  |  |
| StateMachine/ State  **Regions:** Region  **States**: State  **PseudoStates:** PseudoState  **Transitions :** Transition  **Container** : Behavior package | StateMachine/Mode  **Regions:** Region  **States**: Mode  **PseudoStates:** PseudoState  **Transitions :** StateTransition  **Container** : LogicalComponent |