ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Simulations Mathématiques

Author: Sparx Systems
Date: 7/11/2024
Version: 17.0

CRÉÉ AVEC ENTERPRISE ARCHITECT

Table des Matières

Simulations Mathématiques 4
Solveurs 8
Démarrage avec Solveurs 9
Consoles Solveur 11
Solveurs en simulations 14
Solveur d'octave GNU 15
Solveur MATLAB 17
Exemples Solveur Statemachine 20
Exemples Solveur Diagramme d'activités 26
Simulation SysPhS 33
Référencement des bibliothèques Simulation SysPhS 35
Utilisation de la boîte à outils SysPhS 38
Utilisation des Motifs SysPhS 41
Travailler avec les composants SysPhS 43
Définition des valeurs 47
Création de blocs spécifiques à Modelica 49
Créer des blocs spécifiques à Simulink et Simscape 51
Configuration des blocs à la fois comme Modelica et Simulink 55
Simulation 58
Configurer Simulation SysML 60
Affichage du Modèle généré 65
Conseils de débogage SysPhS 67
Analyse de Modèle utilisant Ensemble de Données 70
Exemples Simulation SysPhS 72
Exemple Simulation de circuit d'amplificateur opérationnel 73
Exemple Simulation électronique numérique 82
Exemple d'humidificateur 91
Mise à jour de la configuration SysMLSim 98
SysML Simulation Paramétrique 101
Configurer Simulation SysML 103
Création d'un Modèle Paramétrique 108
Analyse de Modèle utilisant Ensemble de Données 120
Exemples Simulation SysML 122
Exemple Simulation de circuit électrique 123
Exemple Simulation d'oscillateur masse-ressort-amortisseur 131
Régulateur de pression du réservoir d'eau 138
Intégration de Simscape 147
Intégration Simulink 149
Modélisation pour Simulink 150
Intégration de Stateflow 151
Modélisation pour Stateflows 153
Intégration OpenModelica 157
OpenModelica sur Windows 159
OpenModelica sur Linux 161
Simulation SysPhS 164
SysML Simulation Paramétrique 166
Modélisation et Simulation avec OpenModelica Bibliothèque 168

Simulations Mathématiques

Enterprise Architect propose une large gamme d'options pour introduire des outils et des fonctionnalités mathématiques avancés dans vos simulations.

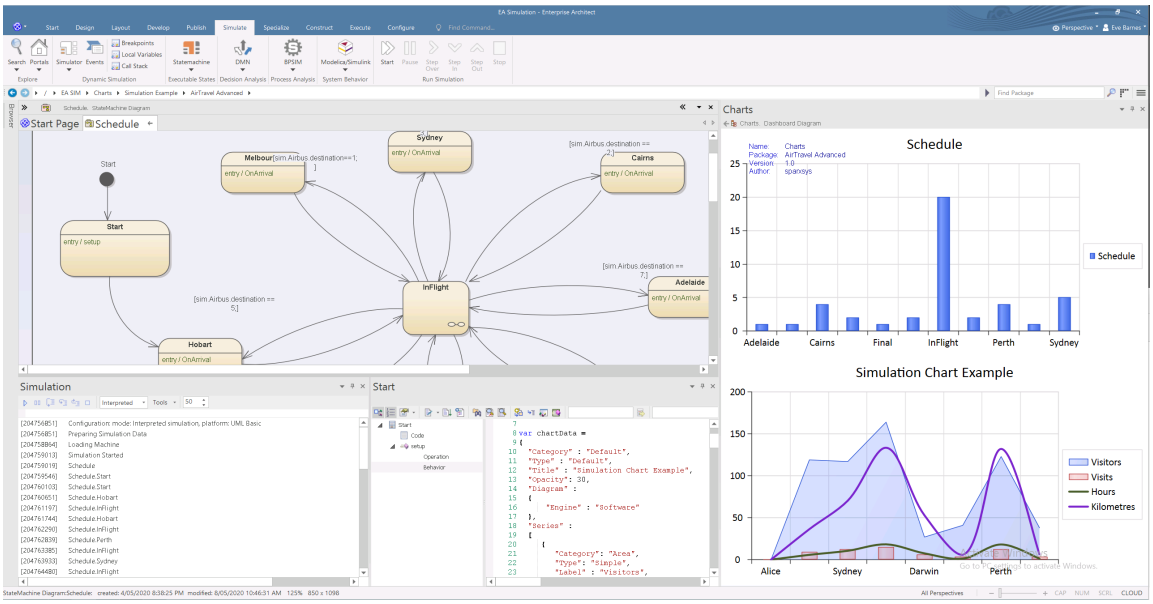
Vous pouvez intégrer la puissance d'outils externes intégrés tels que MATLAB dans vos modèles grâce à l'utilisation des classes Solveur , et pouvez également exporter vos modèles pour les exécuter dans d'autres outils externes tels que MATLAB Simulink, Stateflow et Simscape, ou OpenModelica.

Enterprise Architect inclut une vaste bibliothèque de fonctions mathématiques au sein du moteur JavaScript , offrant les avantages d'une capacité Simulation considérablement étendue.

Enterprise Architect fournit également une large gamme de graphiques dynamiques ; sans avoir recours à des outils externes, vous pouvez configurer ces graphiques pour extraire et tracer des informations à partir de simulations qui ont été directement exécutées dans Enterprise Architect .

Explorez le :

- Classes Solveur dans Enterprise Architect qui appellent MATLAB ou Octave pour intégrer des mathématiques complexes dans vos simulations basées sur des modèles
- Bibliothèque mathématique interne complète basée sur la bibliothèque de fonctions populaire Cephes
- Intégration avec la norme OMG SysPhS, vous permettant de configurer votre modèle pour l'exportation vers des outils courants
- Support de l'exportation de modèles vers MATLAB Simulink, Simscape et Stateflow ; vous pouvez créer votre modèle dans Enterprise Architect et l'exécuter dans MATLAB
- support étendue de Modelica ; vous pouvez créer et configurer votre modèle dans Enterprise Architect et l'exécuter dans Modelica
- Présentation des résultats de votre modélisation et simulation sous forme de graphiques, soit au sein d'un outil de présentation graphique dédié, soit via les facilités de création de graphiques dynamiques d' Enterprise Architect



Intégrations disponibles

Produit	Description
MATLAB	MATLAB est un environnement de calcul numérique et un langage de programmation très répandus et largement utilisés, développés par MathWorks. Il fournit une multitude d'expressions et de formules mathématiques qui peuvent être

	traitées au sein de l'application elle-même ou appelées dans d'autres applications telles qu'Enterprise Enterprise Architect . L'intégration MATLAB d' Enterprise Architect se connecte via l'API MATLAB, ce qui permet à vos simulations Enterprise Architect et à d'autres scripts de s'exécuter en fonction des valeurs des fonctions et expressions MATLAB sélectionnées. Vous pouvez appeler MATLAB via les classes Solveur ou exporter votre modèle vers MATLAB Simulink, Simscape et/ou Stateflow. Note : l'intégration avec MATLAB nécessite la version MATLAB R2018b ou supérieure.
Simulink	Simulink est une application MATLAB de base permettant d'exécuter des simulations SysML de messages dirigés entre blocs. Enterprise Architect peut traduire un modèle SysML au format Simulink, exécuter automatiquement la simulation et tracer les sorties des variables sélectionnées sous forme de graphiques. Vous pouvez également ouvrir le fichier Simulink généré directement dans Simulink, ce qui vous permet de modifier et d'affiner les paramètres de simulation et les fonctionnalités de sortie. Vous pouvez glisser-déposer des blocs de bibliothèque Simulink intégrés courants directement à partir des motifs Simulink Enterprise Architect , ou référencer vos propres blocs personnalisés avec de nouveaux paramètres de stéréotype standard SysPhS. Simulink est une option alternative à OpenModelica pour développer et exécuter des simulations dans Enterprise Architect .
Simscape	Simscape est une extension optionnelle de MATLAB Simulink, qui vous permet de modéliser des systèmes physiques et de demander à MATLAB de simuler et de tracer les sorties demandées, en utilisant la vaste gamme de blocs de bibliothèque de Simscape dans de nombreux domaines physiques différents. Enterprise Architect peut traduire diagrammes Bloc internes SysML en Simscape.
Flux d'état	Stateflow est également une extension optionnelle de MATLAB Simulink, offrant la possibilité de générer diagrammes MATLAB Stateflow à exécuter sous Simulink. Dans Enterprise Architect , cela vous aide à guider vos simulations SysML à l'aide Statemachines modélisées dans Enterprise Architect , qui sont traduites en diagrammes Stateflow.
Modèleica	Modelica est un langage standard ouvert pour modélisation , la simulation, l'optimisation et l'analyse de systèmes dynamiques complexes. Il définit et fournit une structure de fichiers accessible et exploitable par des applications telles que OpenModelica (open source gratuit) et Dymola et Wolfram Modeller (disponibles dans le commerce ; ceux-ci peuvent fonctionner avec Enterprise Architect mais n'ont pas été testés ou intégrés au logiciel Sparx Systems).
OpenModelica	OpenModelica est un environnement libre et open source basé sur le langage standard ouvert Modelica ; OpenModelica permet de lire, d'éditer et de simuler des fichiers Modelica. Enterprise Architect est intégré à OpenModelica et supporte son utilisation sous le standard SysPhS pour définir des constantes et des variables dans la simulation de diagrammes Statemachine et diagrammes Paramétriques . Vous pouvez également afficher les diagrammes Bloc SysML de vos modèles dans Enterprise Architect dans l'éditeur de connexion OMEdit - OpenModelica, qui affiche les alias et notes des blocs. OpenModelica est une option alternative à Simulink pour développer et exécuter des simulations dans Enterprise Architect .
Octave GNU	GNU Octave est une bibliothèque de fonctions mathématiques. À partir du moteur JavaScript d' Enterprise Architect , vous pouvez intégrer un interpréteur Octave

	pour utiliser n'importe laquelle des fonctions Octave disponibles. Octave fournit une alternative aux fonctions MATLAB, avec un accent particulier sur les séquences et les matrices.
JavaScript Math Library	La JavaScript Math Library est une implémentation de la bibliothèque mathématique Cephes intégrée directement dans JavaScript dans Enterprise Architect , pour faciliter l'utilisation de fonctions mathématiques avancées dans une Simulation scriptée (ou dans un graphique dynamique, un Add-In basé sur Modèle ou de nombreux autres scénarios).

Solveurs

Produit	Description
La classe Solveur	La classe Solveur fournit une API commune à une variété d'outils externes ; elle est disponible dans tout moteur JavaScript utilisé par Enterprise Architect et est particulièrement valeur pour appeler des fonctions mathématiques depuis MATLAB ou Octave. Vous pouvez révision les résultats du traitement dans l'outil externe ou les importer dans le moteur JavaScript pour les présenter dans Enterprise Architect .
MATLAB	Le solveur MATLAB est disponible lorsque MATLAB est installé sur votre ordinateur. Le solveur utilise l'API MATLAB pour donner accès à la vaste gamme de fonctions MATLAB disponibles.
Octave	Le solveur Octave est disponible lorsque Octave est installé sur votre ordinateur. Le solveur communique directement avec l'interpréteur Octave pour vous permettre d'accéder aux fonctions et aux données dans un environnement Octave.

Configuration des simulations

Type	Description
Artefacts de configuration	L'artefact de configuration SysMLSim est un artefact spécialement conçu pour spécifier les caractéristiques et les paramètres d'une simulation SysML dans Enterprise Architect . Vous configurez la spécification via la fenêtre Configurer Simulation SysML.
L'extension SysML pour Simulation d'interaction physique et de flux de signaux (SysPhS)	La Norme SysPhS fournit une méthode plus simple, basée sur un modèle, de partage de simulations, de définition de variables, de constantes et de valeurs initiales au sein de chaque élément plutôt que via un fichier de configuration. Cela permet une approche visuelle de la configuration d'une simulation, car les variables, les constantes et les valeurs initiales peuvent être rendues visibles dans diagrammes dans des compartiments supplémentaires sur les blocs SysML.
Définir plusieurs ensembles de données	Plusieurs jeux de données peuvent être définis par rapport aux blocs SysML utilisés dans une configuration Simulation dans un modèle Paramétriques . Cela permet des variations de simulation reproductibles à l'aide du même modèle SysML.

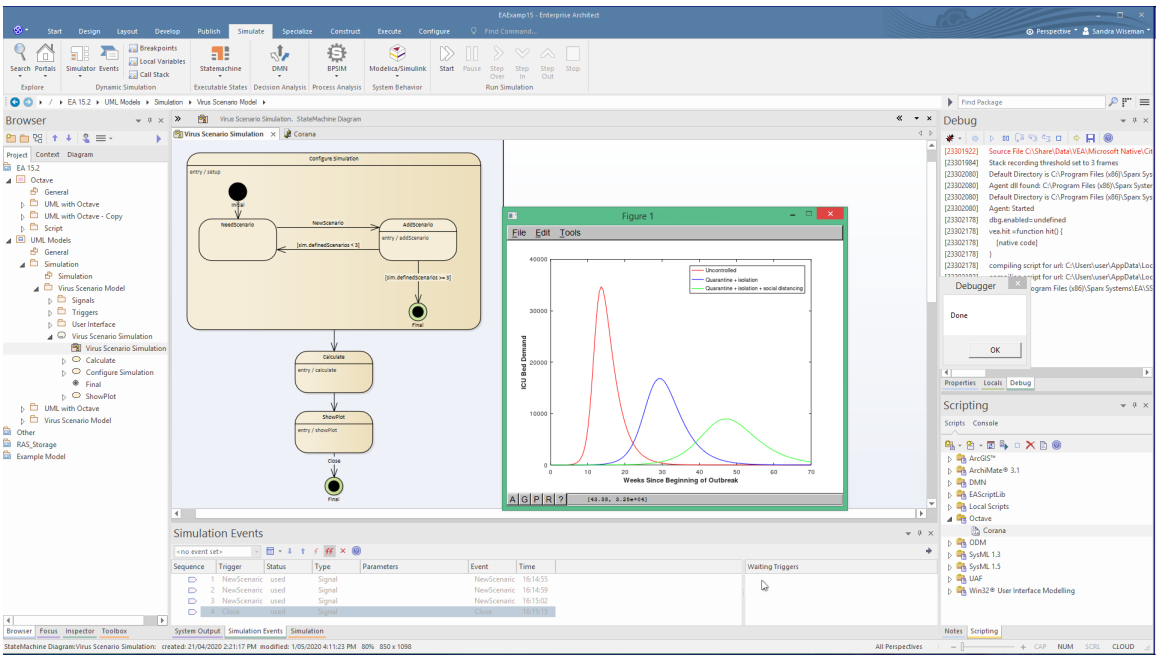
Cas d'utilisation courants

Nom	Description
Consoles Solveur	• Testez rapidement les commandes à utiliser dans un script ou une simulation • Appelez une fonction MATLAB pour voir si elle renvoie ce que vous attendez et s'exécute sans erreur • Coupez et collez un extrait JavaScript qui ne doit être exécuter qu'une seule fois, plutôt que de créer un script puis de le supprimer
Solveurs en simulations	• Appeler une fonction mathématique complexe qui a été définie comme une fonction Octave • Appelez les routines API de MATLAB pour déterminer un flux de décision
Simulation SysML SysPhS	• Modèle un nouveau système ABS automobile dans Enterprise Architect SysML et simulez-le à l'aide de Simulink • Concevez et modélisez un système hydraulique dans Enterprise Architect et simulez le système dans OpenModelica à l'aide d'une bibliothèque existante de composants Modelica
Générer un StateChart et Affiner et Déboguer dans StateFlow	• Créez une Statemachine SysML pour définir rapidement l'action d'un utilisateur allumant et éteignant le système à plusieurs reprises ; générez la simulation et ouvrez-la dans Stateflow pour afficher les paramètres d'état en « temps réel » et ajustez les paramètres de Stateflow
Modèle et Test un StateChart	• Modéliser entièrement un StateChart avant de le simuler dans Stateflow

Solveurs

À partir de la version 15.2 Enterprise Architect , vous pouvez utiliser JavaScript pour appeler des « Solveurs » qui assurent l'intégration avec des outils externes tels que MATLAB et Octave. La classe Solveur fournit une connectivité en temps réel à votre outil mathématique externe, grâce auquel vous appelez des fonctions mathématiques complexes et d'ordre élevé pendant l'exécution de simulations dans Enterprise Architect .

Solveurs vous aident à intégrer la puissance de calcul de MATLAB et d'Octave dans Enterprise Architect dans des simulations, Add-Ins basés sur des modèles et des scripts personnalisés. La classe Solveur est une construction intégrée au moteur JavaScript d' Enterprise Architect .



Démarrage avec Solveurs

La variable de classe Solveur peut être créée une fois par simulation ou, si nécessaire, plusieurs instances de classe Solveur peuvent être créées. Note que le démarrage d'une nouvelle instance de classe Solveur peut parfois prendre plusieurs secondes. Il est généralement recommandé de créer une instance de classe Solveur au début de la simulation et de la réutiliser lorsque cela est nécessaire.

Pour utiliser la classe Solveur , vous devez avoir une connaissance des fonctions disponibles dans votre Bibliothèque mathématique préférée et des paramètres qu'elles utilisent, comme décrit dans la documentation du produit Bibliothèque .

Vous définissez d'abord la ou les Bibliothèque mathématiques que vous souhaitez utiliser dans votre script. Pour MATLAB, vous tapez :

var matlab = new Solver('matlab');

Pour Octave vous tapez :

var octave = new Solver('octave');

Ensuite, partout où vous devez utiliser une fonction mathématique dans votre script, pour MATLAB, vous tapez :

matlab.exec('complexMathsFunction', paramètres);

Pour Octave, vous tapez :

octave.exec('complexMathsFunction', *paramètres*);

Ces deux lignes de script exécutent la fonction dans l'outil approprié et y affichent les résultats. Si vous souhaitez récupérer les résultats dans Enterprise Architect , vous faites précéder la ligne par :

var resultFrom'Nom de l'outil';

C'est-à-dire:

var resultFromMatlab = matlab.exec('complexMathsFunction', paramètre1, paramètre2); ou

var resultFromOctave = octave.exec('complexMathsFunction', paramètre1, paramètre2);

Faisant partie du moteur JavaScript , les classes Solveur sont également immédiatement accessibles aux auteurs de Add-In créant Add-Ins JavaScript basés sur des modèles.

Note : si un nouveau Solveur est créé dans une section de JavaScript appelée plusieurs fois, les performances de simulation seront considérablement diminuées (par exemple, sur un nœud Statemachine entré plusieurs fois).

Consultez les rubriques d'aide *de GNU Octave Solveur* et *MATLAB Solveur* pour plus d'informations sur ces deux Solveurs .

Solveur Constructeur

Constructeur	Description
Solver(string solverName)	Crée un nouveau Solveur connecté à une nouvelle instance de l'application d'assistance spécifiée.

Méthodes Solveur

Méthode	Description
get(string name)	Récupère une valeur nommée à partir de l'environnement du solveur.

set(string name, object value)	Affecte une nouvelle valeur à une variable nommée dans l'environnement du solveur.
exec(string name, string arguments, int returnValues)	Exécute une fonction nommée. Les fonctions réelles dépendent du type de solveur utilisé.

Consoles Solveur

La console Solveur est un éditeur facile d'accès et simple à utiliser. Elle est principalement destinée à la gestion Solveurs MATLAB et Octave, offrant un accès simple et rapide aux fonctions de test prises en charge dans ces applications externes. La console Solveur peut être utilisée pour créer, vérifier et maintenir les appels utilisés dans les simulations, ainsi que dans les scripts écrits en JavaScript .

Accéder

Ruban	Simuler > Console > Solveurs > MATLAB Simuler > Console > Solveurs > Octave

Initialisation Solveur

Le Solveur choisi est automatiquement défini sur l'un de ceux-ci :

MATLAB:

- var matlab = new Solver("matlab");

Octave:

- var octave = new Solver("octave");

Si l'arrière-plan Octave ou MATLAB est accessible, la console prompt les détails de départ et sera prête à accepter n'importe quelle entrée.

Usage

La console accepte les commandes JavaScript sur une seule ligne qui seront exécutées une par une. Saisissez les nouvelles commandes dans la section de saisie de texte inférieure et appuyez sur la touche Entrée pour exécuter la commande.

La commande exécutée sera ajoutée au panneau de sortie principal, avec n'importe quelle sortie de la commande.

Les Solveurs sont ensuite accessibles à l'aide des fonctions de la classe Solveur ; par exemple :

MATLAB:

- matlab.set('<nom_variable>', <valeur>)
- matlab.get('<nom_variable>')
- matlab.exec(<fonction>, <paramètres>)

Octave:

- octave.set('<nom_variable>', <valeur>)
- octave.get('<nom_variable>')
- octave.exec(<fonction>, <paramètres>)

Pour plus de détails, consultez les rubriques d'aide *d'Octave Solveur* et de *MATLAB Solveur* .

Les commandes précédentes qui ont été saisies peuvent être réutilisées en appuyant sur la touche Flèche vers le haut ou Flèche vers le bas.

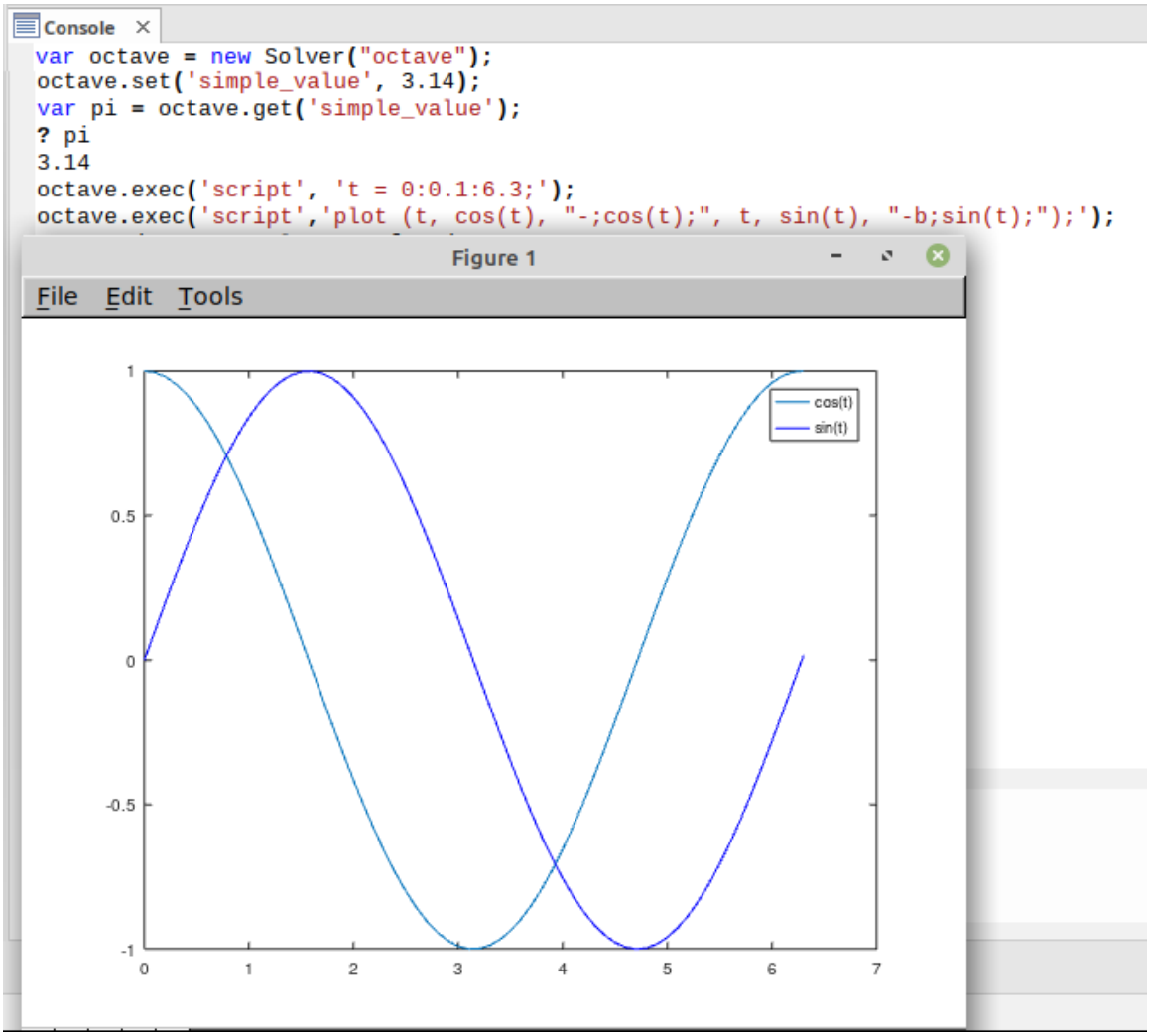
Sortir

Pour afficher la valeur d'une variable ou d'une fonction, vous pouvez utiliser :

- ? <nom_variable>
- ? <fonction>

Exemple

Cette illustration montre quelques exemples d'utilisation des fonctions Set, Get et Exec d'Octave Solveur , ainsi que la sortie utilisant ?, dans la console Octave :



Le tracé d'octave est la sortie générée par les deux instructions ocatave.exec().

Commandes de la console

Les commandes de la console sont précédées du caractère ! et indiquent à la console d'effectuer une action. Les commandes de la console incluent :

- !clear - efface l'affichage de la console
- !save - enregistre l'affichage de la console dans un fichier
- !help - imprime une liste de commandes
- !close - ferme la console
- !include <scriptname> - exécute l'élément de script nommé ; le nom du script est au format GroupName.ScriptName (les espaces sont autorisés dans les noms)
- ? - liste les commandes (identique à !help)
- ? <variable ou fonction> - renvoie la valeur

Solveurs en simulations

Un Solveur au sein d'une simulation vous aide à inclure le comportement des modèles mathématiques, en interrogeant l'état si nécessaire pour ajuster le comportement du modèle UML correspondant.

Création et initialisation d'un Solveur

Il est préférable de créer un Solveur une fois au début d'une simulation. Cela permet de l'utiliser tout au long de la simulation sans subir le retard d'une période de construction répétée. C'est également le bon moment pour charger les modules requis, définir les fonctions et définir un état initial pour le modèle.

Dans une simulation Statemachine , les bons endroits pour effectuer cette initialisation sont dans l'effet pour la transition menant de l' State initial, ou dans l'entrée pour un State qui n'est pas ré-entré pendant la simulation.

Dans une simulation d'activité, vous devrez ajouter le Solveur à l'effet d'une Action .

Mise à jour d'un Modèle Solveur

Au fur et à mesure de la progression de la simulation, vous pouvez utiliser l'un des champs Effet pour mettre à jour les paramètres du modèle mathématique. Cela peut consister à mettre à jour les taux ou à changer d'algorithme. Dans une simulation Statemachine :

- Un effet de transition peut être utilisé lorsque le changement est nécessaire pour un flux particulier
- Une entrée State peut être utilisée pour garantir que le comportement est modifié pendant que la simulation est dans un State , quelle que soit la manière dont elle y parvient ; cela inclut tous States imbriqués
- Une sortie State peut être utilisée pour garantir que le comportement est modifié à chaque fois que la simulation quitte un State , quelle que soit la manière dont elle le quitte.
- Une activité d’ State se comporte de manière similaire à une action d’entrée.

Dans une simulation d’activité, il n’y a que des effets Action .

Interrogation d'un Modèle Solveur

Avec une protection de transition, appelez *solver.get()* . Vous pouvez également appeler *solver.exec()* pour appeler une fonction sans effets secondaires.

Solveur d'octave GNU

GNU Octave est une bibliothèque de fonctions mathématiques, avec un accent particulier sur les séquences et les matrices. Vous pouvez appeler chaque fonction dans un script écrit en JavaScript .

Vous pouvez invoquer des fonctions mathématiques arbitraires depuis Octave au moment exécuter , en utilisant une construction simple appelée classe Solveur , écrite en JavaScript ; une classe Solveur pour Octave peut appeler l'outil Octave externe et lier les fonctions mathématiques avancées directement dans votre simulation en cours d'exécution. Par exemple :

var octave = nouveau Solveur (octave);

var resultFromOctave = octave.exec('complexMathsFunction', paramètre1, paramètre2);

Voir la rubrique d'aide *Solveurs dans Simulations* .

Fonctionnalités incluent :

- Appel de vecteurs, de matrices, de nombres et de chaînes depuis Octave
- Les vecteurs d'octave sont renvoyés sous forme de tableaux unidimensionnels JavaScript (et les tableaux unidimensionnels JavaScript sont renvoyés sous forme de vecteurs d'octave)
- Les matrices d'octave renvoient des tableaux bidimensionnels JavaScript (et les tableaux bidimensionnels JavaScript renvoient des matrices d'octave)
- Vous pouvez récupérer les valeurs des variables d'Octave en utilisant :
  octave.get(<nom>)
- Vous pouvez appeler n'importe quelle fonction Octave avec des valeurs JavaScript en utilisant :
  octave.exec(<nom>, [<arguments>], 0/1)
- Tous les arguments sont passés dans un tableau JavaScript
- Vous pouvez également utiliser le résultat en JavaScript ; passez 1 si vous voulez un résultat, 0 si vous ne voulez pas de résultat
- Vous pouvez exécuter n'importe quelle instruction Octave en utilisant :
  octave.exec("script", <instruction>, 0/1)

Octave est disponible comme alternative à la bibliothèque MATLAB et peut être utilisé dans tous les mêmes contextes que MATLAB.

Installation et configuration

Après avoir installé Octave, Enterprise Architect lira l'emplacement à partir du registre pour fournir une intégration automatique.

Usage

Tâche	Description
Attribution de valeurs	Utilisez la commande octave.set ; par exemple : octave.set("valeur_simple", 3.14); octave.set("exemple_séquence", [0, 1, 2]); octave.set("identité", [[1, 0], [0, 1]]);
Construire	Dans l'éditeur JavaScript , créez un nouveau Solveur connecté à Octave en passant « octave » au constructeur Solveur .

	var octave = new Solver("octave");

Récupération des valeurs	Utilisez la commande octave.get ; par exemple :
	var simple_value = octave.get("simple_value");
	var exemple_séquence = octave.get("exemple_séquence");
	var identity= octave.get("identity");

Fonctions d'appel	Passez le nom de la fonction comme premier argument à Solveur .exec. Transmettez tous les paramètres à cette fonction dans un tableau comme deuxième argument. Lorsque vous souhaitez qu'une valeur soit renvoyée par cette fonction dans JavaScript , transmettez une valeur différente de zéro comme troisième argument. Par exemple : var séquence = octave.exec("linspace", [0, 10, 1001],1);
Exécution des instructions	Transmettez « script » comme nom pour le premier argument de Solveur .exec. Transmettez une instruction Octave entière dans une string comme deuxième argument. octave **.** exec **(** 'script' **,** 't = 0:0.1:6.3;' **);** octave **.** exec **(** 'script' **,** 'plot (t, cos (t), "-; cos (t); " , t, sin (t), "-b; sin (t); ");' **);**

Tracer()	La méthode Trace(statement) vous permet d'imprimer des instructions de trace à n'importe quel moment de votre simulation. Il s'agit d'un excellent moyen de compléter le log Simulation avec des informations de sortie supplémentaires pendant l'exécution.
	La sortie Trace() est écrite dans la fenêtre Simulation .
	Voici un exemple d'utilisation de la méthode Trace() :
	octave.set('valeur_simple', 3.14);
	var pi = octave.get('valeur_simple');
	Trace("PI = " + pi); // affiche cette valeur dans la fenêtre Simulation

Vidéos

Sparx Systems propose une vidéo YouTube sur l'utilisation d'un Solveur pour générer un tracé avec Octave. Voir :

- Plotting with Octave

Solveur MATLAB

MATLAB est un environnement de calcul numérique et un langage de programmation qui comprend une grande bibliothèque de fonctions mathématiques, chacune pouvant être appelée à partir d'un script écrit en JavaScript .

Vous pouvez invoquer des fonctions mathématiques arbitraires à partir de MATLAB au moment exécuter à l'aide d'une construction simple, appelée classe Solveur , écrite en JavaScript . Une classe Solveur pour MATLAB peut appeler l'outil MATLAB externe via son API et lier les fonctions mathématiques directement à votre simulation en cours d'exécution. Par exemple :

var matlab = new Solver("matlab");

var resultFromMatlab = matlab.exec('complexMathsFunction', paramètre1, paramètre2);

Voir la rubrique d'aide *Solveurs dans Simulations* .

Fonctionnalités incluent :

- Récupération de vecteurs, de matrices, de nombres et de chaînes à partir de MATLAB
- Les vecteurs MATLAB sont renvoyés sous forme de tableaux unidimensionnels JavaScript (et les tableaux unidimensionnels JavaScript sont renvoyés sous forme de vecteurs MATLAB)
- Les matrices MATLAB renvoient des tableaux bidimensionnels JavaScript (et les tableaux bidimensionnels JavaScript renvoient des matrices MATLAB)
- Vous pouvez récupérer des valeurs de variables à partir de MATLAB en utilisant : matlab.get(<nom>)
- Vous pouvez appeler n'importe quelle fonction MATLAB avec des valeurs JavaScript en utilisant : matlab.exec(<nom>, [<arguments>])
- Tous les arguments sont passés à l'intérieur d'un object JavaScript
- Vous pouvez également utiliser le résultat en JavaScript
- Vous pouvez exécuter n'importe quelle instruction MATLAB en utilisant : matlab.exec("script")

MATLAB est disponible comme alternative à la bibliothèque GNU Octave et peut être utilisé dans tous les mêmes contextes que GNU Octave.

Note : l'intégration avec MATLAB nécessite la version MATLAB R2018b ou supérieure.

Installation et configuration

Après avoir installé MATLAB, Enterprise Architect lira l'emplacement à partir du registre pour fournir une intégration automatique.

Si MATLAB ne se charge pas automatiquement, définissez le chemin (comme dans la rubrique d'aide *Configurer la fenêtre Simulation SysML*) sur la valeur obtenue en exécutant « matlabroot » dans la console MATLAB.

Usage

Utiliser	Discussion
Construire	Créez un nouveau Solveur connecté à MATLAB en passant *« matlab »* au constructeur Solveur . C'est-à-dire : var matlab = new Solver('matlab');
Attribution de valeurs	Attribuer des valeurs à l'aide de la fonction matlab.set. Par exemple :

	matlab.set('simple_value', 3.14); ou var myString = "ceci est un exemple string "; matlab.set('maChaîne', maChaîne);
Récupération des valeurs	Récupérez les résultats de MATLAB à l'aide de la fonction matlab.get. Par exemple : var simple_value = matlab.get('simple_value'); var maChaîne = matlab.get('maChaîne');

Appel de fonctions	Passez le nom de la fonction comme premier paramètre à Solveur .exec.
	Soit:
	• Pour les fonctions qui prennent un seul paramètre, transmettez simplement la valeur comme deuxième paramètre ; par exemple :
	var résultat = matlab.exec(' ceil ', 7.4);
	ou
	• Si deux paramètres ou plus sont requis, transmettez tous les paramètres sous forme d' Object JavaScript comme deuxième paramètre ; cela peut être fait en ligne, comme :
	var résultat = matlab.exec('moins', {0 : 8, 1 : 4,5});
	Note : l'ordre des paramètres est déterminé par l'ordre alphabétique des noms object
	Une fonction d'aide peut être utilisée ici pour éviter les erreurs :
	// Enveloppez un nombre variable d'arguments dans un object à transmettre à solver.exec
	fonction args() {
	var obj = {};
	pour (var i = 0; i < arguments.length; i++) {
	obj[i] = arguments[i];
	}
	retourner obj;
	}
	var résultat = matlab.exec('minus', args(8, 4.5));

Tracer()	La méthode Trace(statement) vous permet d'imprimer des instructions de trace à n'importe quel moment de votre simulation. Il s'agit d'un excellent moyen de compléter le log Simulation avec des informations de sortie supplémentaires pendant l'exécution.
	La sortie Trace() est écrite dans la fenêtre Simulation . Par exemple :
	matlab.set('simple_value', 3.14);
	var pi = matlab.get('simple_value');
	Trace("Valeur simple = " + pi);

Vidéos

Sparx Systems propose une vidéo YouTube sur l'utilisation de la console MATLAB pour créer un Solveur MATLAB. Voir :

- The MATLAB Console

Deux autres vidéos illustrent l'utilisation du Solveur MATLAB pour réaliser une Simulation Statemachine d'un essai de vaccin et d'un tirage au sort. Voir :

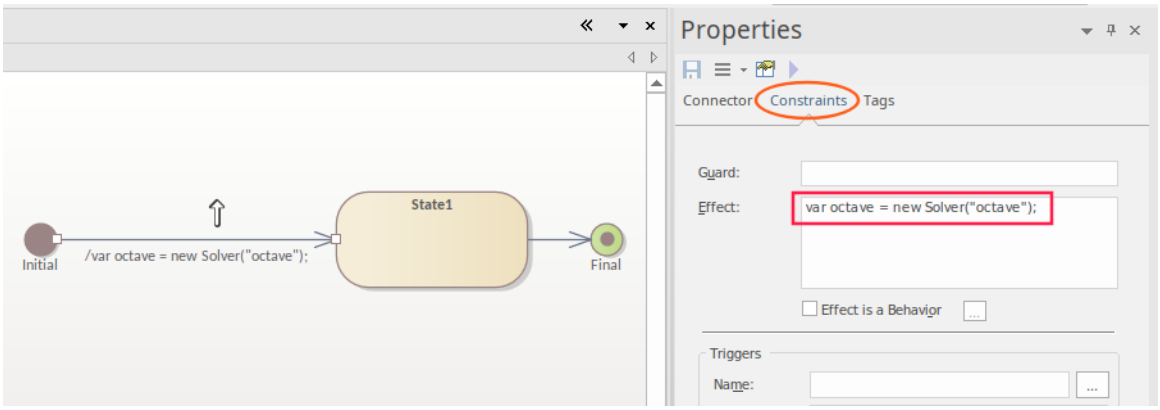
- Matlab Solveur
- Statemachine Simulation - Vaccine Trial
- Statemachine Simulation - Lottery Draw

Exemples Solveur Statemachine

Cette rubrique fournit quelques exemples simples d'utilisation et de mise en œuvre Solveurs dans les simulations Statemachine , notamment l'utilisation de scripts dans les opérations Statemachine ainsi que dans les connecteurs de transition. Les images de cette rubrique montrent des exemples Solveur Octave. Cependant, sauf indication contraire, le même script peut être utilisé pour MATLAB dans le Solveur MATLAB.

Initialisation du Solveur

Le script d'initialisation d'un Solveur dans une Statemachine peut être placé soit dans l'Effet d'une Transition, soit dans l'Opération d'Entrée d'un State . Il est généralement placé dans l'Effet de la première Transition sortant de l'Initial.

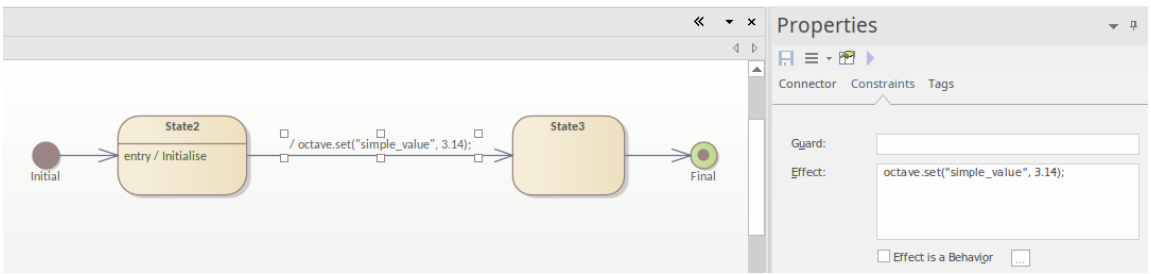


Attribution de valeurs et exécution de commandes

Pour attribuer une valeur , utilisez la fonction octave.set() ou matlab.set().

Pour Statemachines la méthode peut être placée dans l'effet d'une transition ou dans le comportement (entrée/do/sortie) d'une opération.

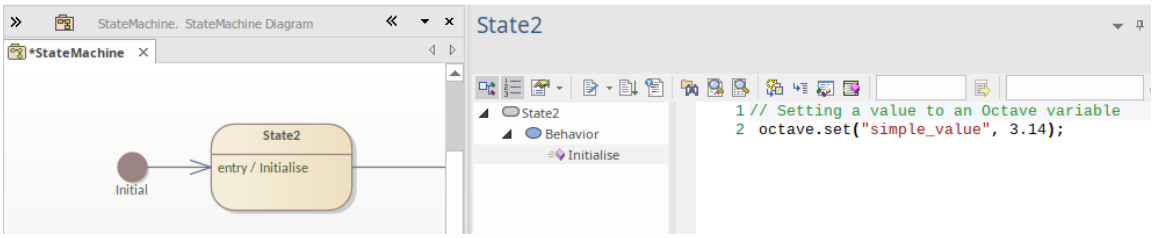
Pour éditer l'effet d'une transition, utilisez la page « Contraintes » de la dialogue « Propriétés ».



Pour placer la méthode dans une opération, ouvrez le script de comportement de l'opération :

1. Créez une nouvelle opération d'entrée dans l' State à partir du diagramme ou de la fenêtre Navigateur , en cliquant avec le bouton droit sur l'élément et en sélectionnant l'option de menu contextuel « Fonctionnalités > Opérations ».
2. Cliquez sur la nouvelle opération d’entrée et appuyez sur Alt+7.

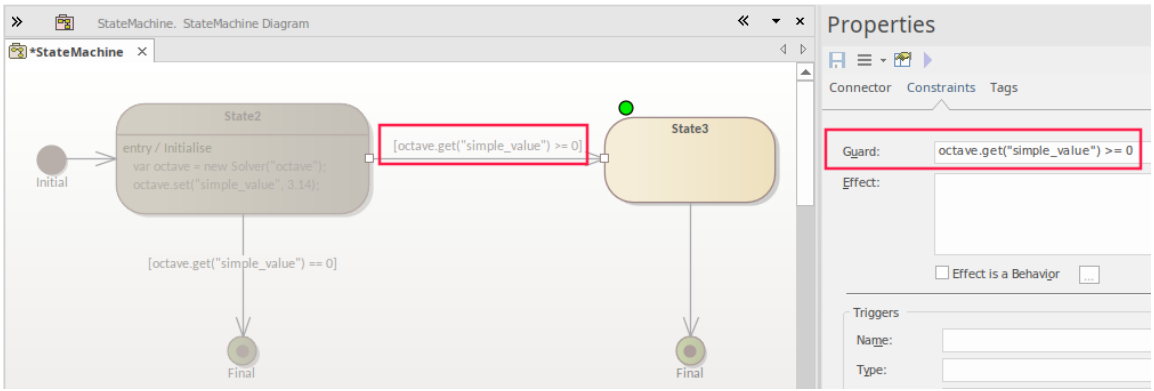
Cela ouvrira le script de comportement dans l'éditeur.



Lors de l'exécution de commandes MATLAB ou Octave à l'aide de la fonction .exec(), les commandes peuvent être placées dans l'effet de la transition ou dans le script de comportement de l'opération.

Branchement conditionnel

Lors de l'utilisation d'un branchement conditionnel dans une Statemachine , la condition peut être placée dans la garde d'une transition et peut contenir un script appelant des fonctions MATLAB ou Octave. Par exemple :



Note :

- La condition peut également être définie à l'aide d'un choix ; le placement de l'énoncé de condition est le même, dans la mesure où il est défini dans la garde de la transition sortante.
- L'exemple montre une simulation au Point d'Arrêt situé sur l'état 3

Obtenir des résultats

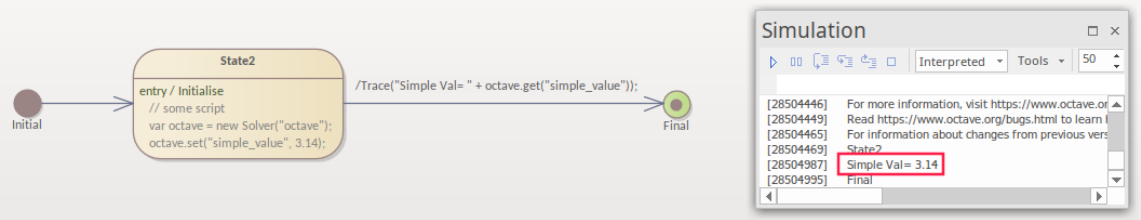
Lorsque vous renvoyez les résultats des appels de fonctions externes, vous pouvez utiliser la fonction get() de Solveur pour renvoyer les résultats dans le script. Il existe trois options principales pour transmettre ensuite les résultats du script à l'utilisateur :

- Tracer()
- Parcelle
- Affichage Win32

Avec les options Trace et Win32, vous devez renvoyer une copie locale dans votre JavaScript , en utilisant la commande .get() de Solveur . Comme exemple d'utilisation de la commande .get(), voir la garde dans l'image précédente.

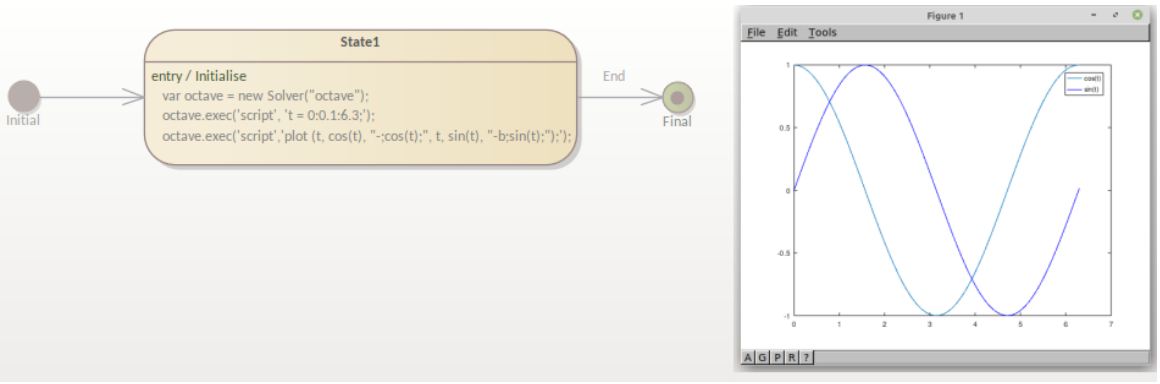
Utilisation de Trace

La commande Trace() est utile lors de l'écriture et du débogage d'une simulation, car elle vous permet de vérifier les résultats de votre script à différentes étapes. Les résultats sont affichés dans la fenêtre Simulation .



Exécution de complots

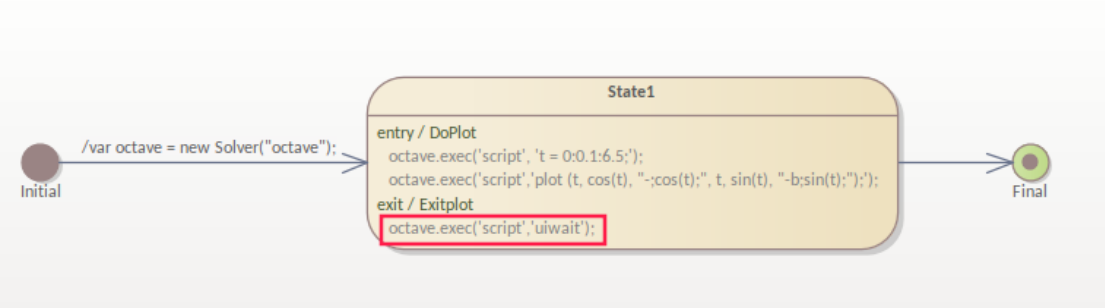
Octave et MATLAB mettent l'accent sur la capacité à générer un tracé, car il s'agit d'une méthode clé pour générer des résultats. Pour générer un tracé, vous utilisez la fonction .exec() de Solveur pour appeler une fonction de génération de tracé.



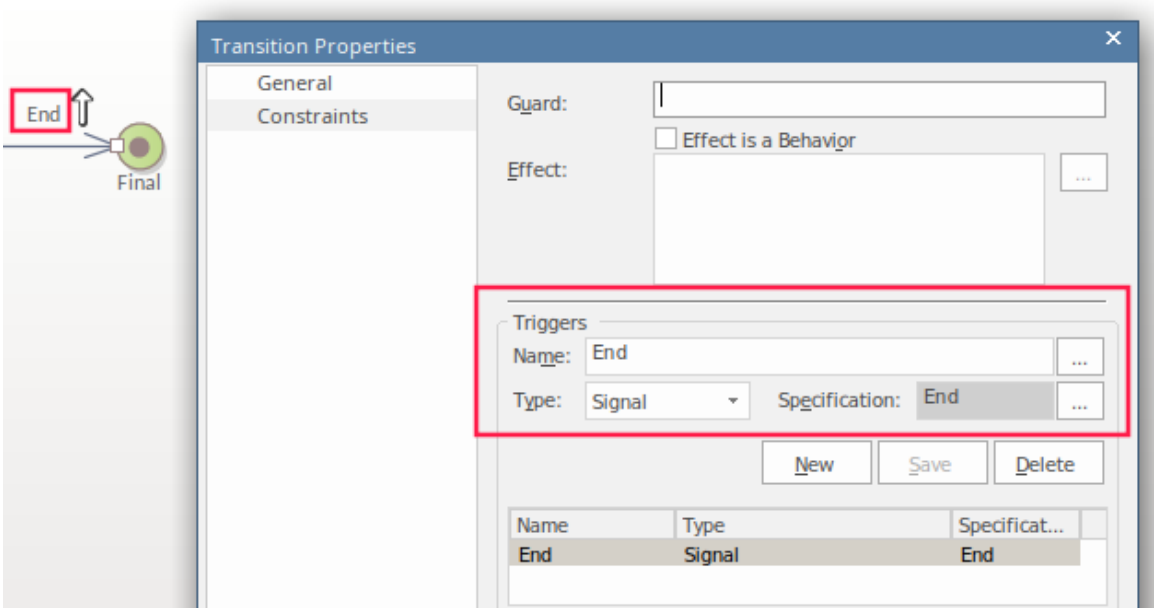
Tenir le terrain

Lors de l'exécution d'un tracé dans une simulation, si la simulation n'est pas mise en pause, le tracé ne sera visible que pendant un bref instant. Ainsi, pour Statemachines , il existe deux options pour mettre en pause le flux pendant que le tracé est visualisé :

1. Utilisation de l'interface utilisateur dans Octave ou MATLAB. Par exemple, cela peut être défini dans l'opération de sortie ou dans un effet de transition. Voici un exemple utilisant Octave :
   octave.exec('script','uiwait');

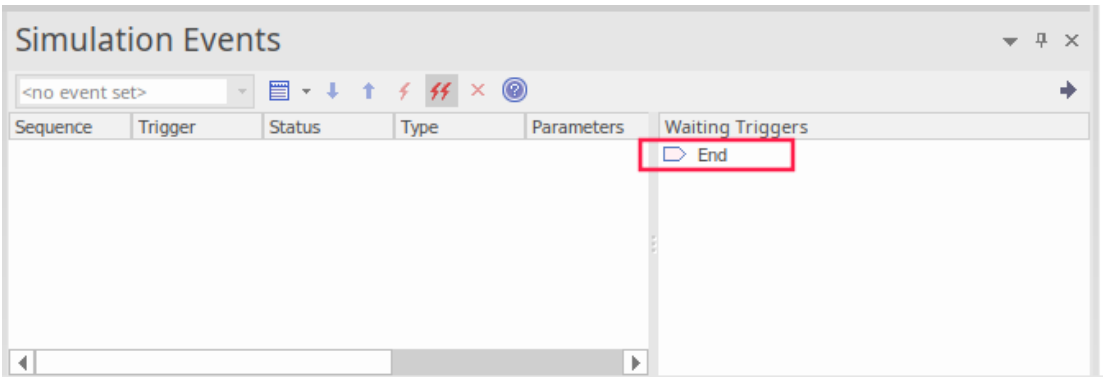


2. Définition d'une séquence Déclencheur /Evénement pour mettre en pause la simulation dans une transition en dehors de l' State où le tracé a été généré. Dans le cas d'exemple, il s'agit de la transition finale.



Pour progresser au-delà de la transition, vous pouvez soit :

- Cliquez sur le Déclencheur affiché dans la fenêtre Simulation Événements ou
- Utilisez un BroadcastSignal() à partir, par exemple, d'un bouton dans l'écran Win32 (ceci est abordé dans la section *Utilisation de l'interface Win32* , ci-après)



Utilisation de l'interface Win32

Lors de l'utilisation de l'interface Win32, les grandes étapes à suivre sont les suivantes :

- Créer une dialogue Win32
- Définir une ligne de script pour ouvrir le dialogue
- Obtenir une valeur d'un champ dans le dialogue
- Transmettez cette valeur au solveur
- Utilisez un bouton pour déclencheur l'intrigue

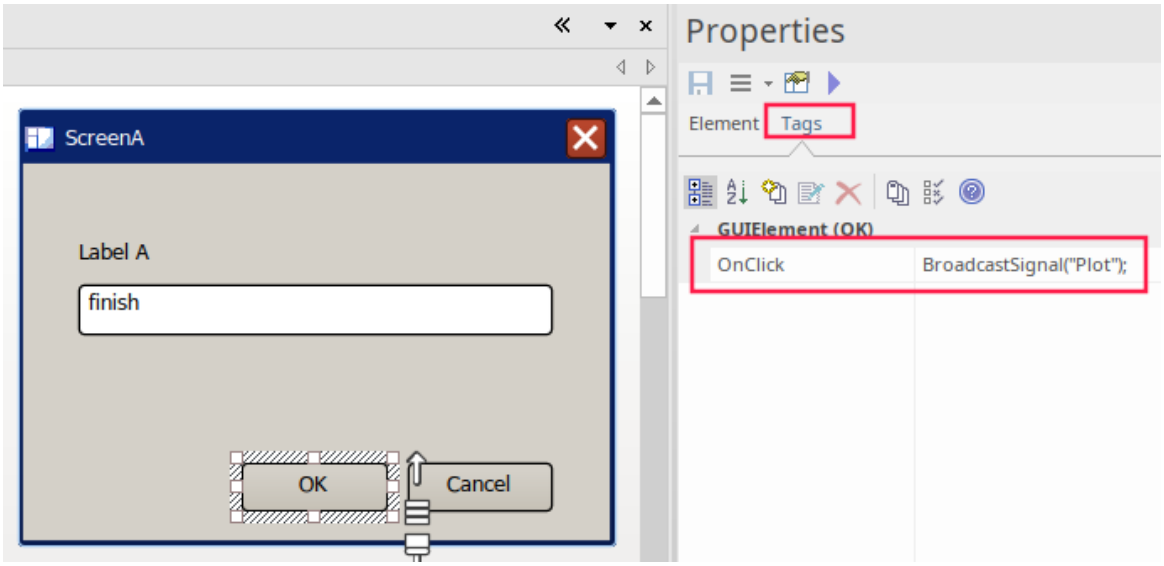
Les étapes pour configurer une interface Win32 sont :

- Créez un « modèle Win32 de démarrage » à l'aide du Constructeur de Modèle - sélectionnez l'ensemble de perspectives « Conception UX » et la perspective « Modèles UI Win32 »
- Changer le nom de la <<Dialogue d'écran Win32>> en « ScreenA »
- Remplacez le nom de « Contrôle d'édition A » par quelque chose de plus significatif, comme « Terminer »
- Cliquez sur State1 et appuyez sur Alt+7 pour ajouter - dans le script de l'opération d'entrée - un appel pour ouvrir le

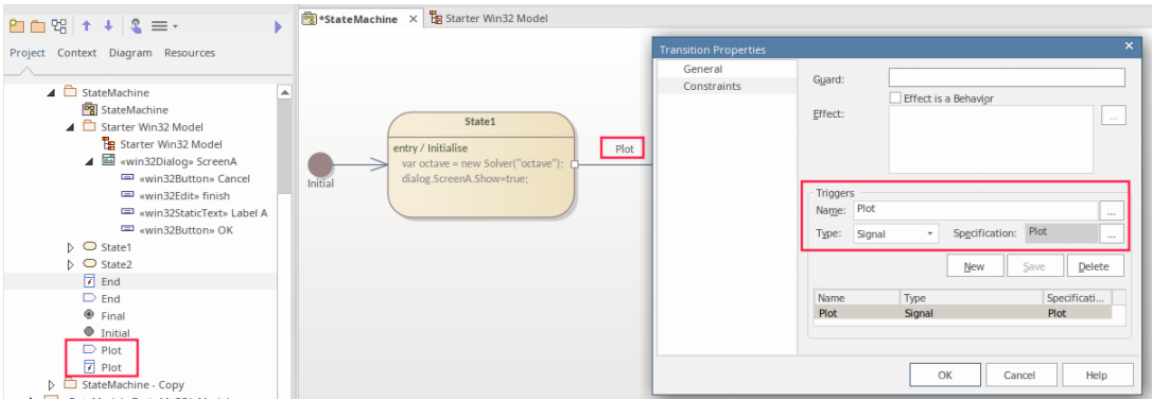
dialogue :
dialogue . ScreenA . Show = vrai ;

Pour obtenir une valeur d'entrée utilisateur valide, l'utilisateur doit cliquer sur le bouton OK , donc à partir du bouton OK , vous devez diffuser un événement pour un Déclencheur pour démarrer le processus. Pour recevoir le Déclencheur vous devez avoir un Signal et définir une Transition à déclencher par cette diffusion vers le Signal.

Consultez les étapes pour créer et définir le Déclencheur et le Signal « Fin », comme indiqué dans la section *Utilisation d'un Déclencheur pour maintenir le tracé* , plus haut. Dans ce cas, nous définissons la même transition, mais pour un nouveau Déclencheur appelé « Plot ». Cela consiste à envoyer un BroadcastSignal('Plot') en utilisant la Valeur Étiquetée OnClick sur un bouton :



La transition sortant de l'état 1 est configurée pour être déclenchée par le BroadcastSignal :

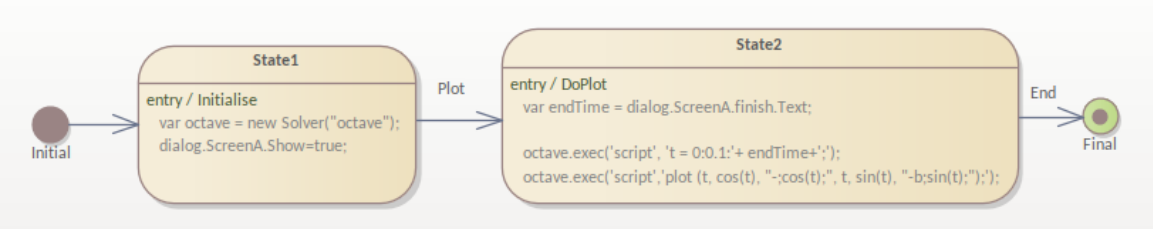


Sur la Transition, vous devez créer un Déclencheur , définir le Trigger-Type sur Signal et créer le Signal. Pour plus de détails, consultez la rubrique d'aide *Simulation d'Interface Utilisateur Win32* .

Le script pour exécuter le plot sera dans l'opération Entry de State2. Pendant la simulation la transition vers State2 ne se produira qu'après avoir cliqué sur le bouton OK Win32 et envoyé le déclencheur Plot. On va donc maintenant :

- Ajoutez une opération d'entrée et ouvrez le comportement à l'aide de Alt+7
- Dans l'opération d'entrée de cet état, nous obtenons la valeur du champ en utilisant :
  var Fin des Temps = dialog.ScreenA.finish.Text;
  Cela définit une variable avec la valeur du dernier paramètre à envoyer à Octave pour le tracé
- Dans l'instruction octave.exec(), nous plaçons la variable pour définir le nombre de secondes à tracer :
  octave.exec('script' , 't = 0:0.1:' + finHeure+ ';');

Cela donne deux States avec les scripts dans les opérations d'entrée :

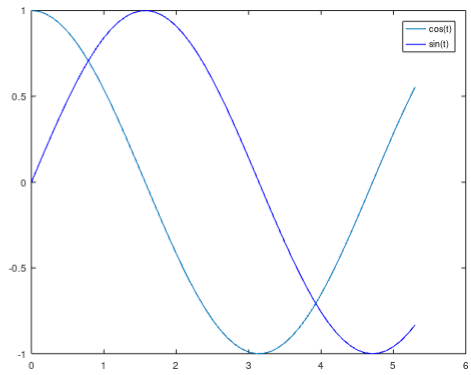


Exécution de la Simulation

Pour exécuter la simulation :

- Sélectionnez le ruban Simuler et cliquez sur ' Exécuter Simulation > Démarrer '

Lorsque vous entrez une valeur et cliquez sur le bouton OK , cela renvoie :

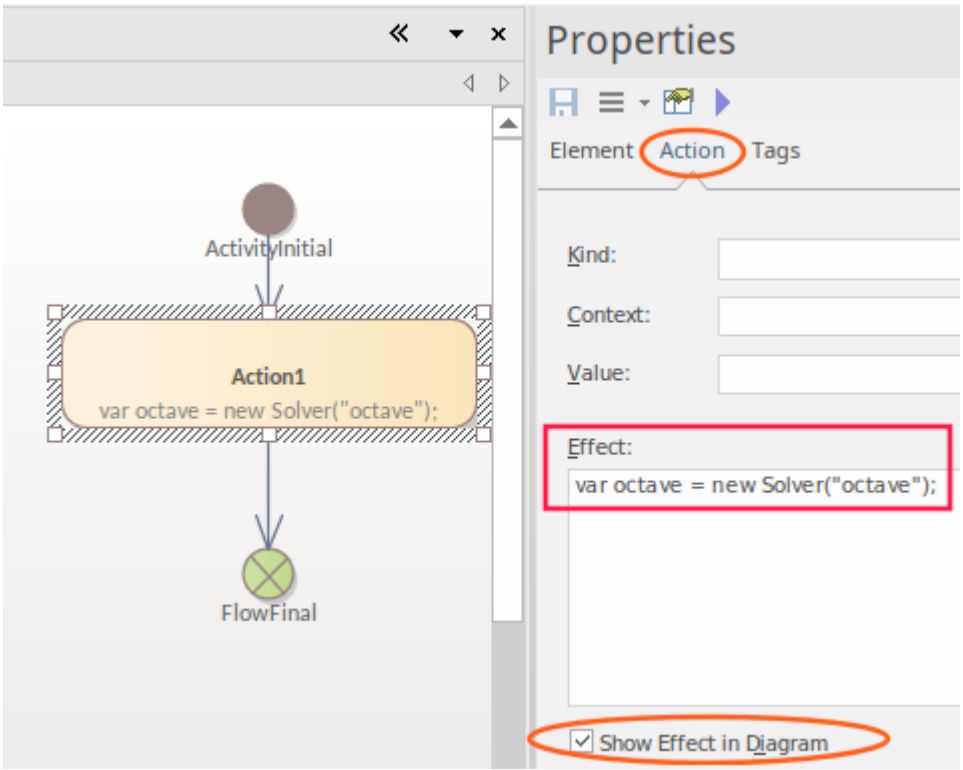


Exemples Solveur Diagramme d'activités

Dans cette rubrique, nous exécuter en revue quelques exemples simples de l'endroit et de la manière d'appliquer Solveurs dans les simulations diagramme d'activité. Cela inclut l'utilisation de scripts dans Action Effects, ainsi que dans ControlFlow Guards. Les images de cette rubrique montrent des exemples d'Octave Solveur ; cependant, le même script peut être utilisé pour MATLAB en remplaçant Octave Solveur par MATLAB Solveur .

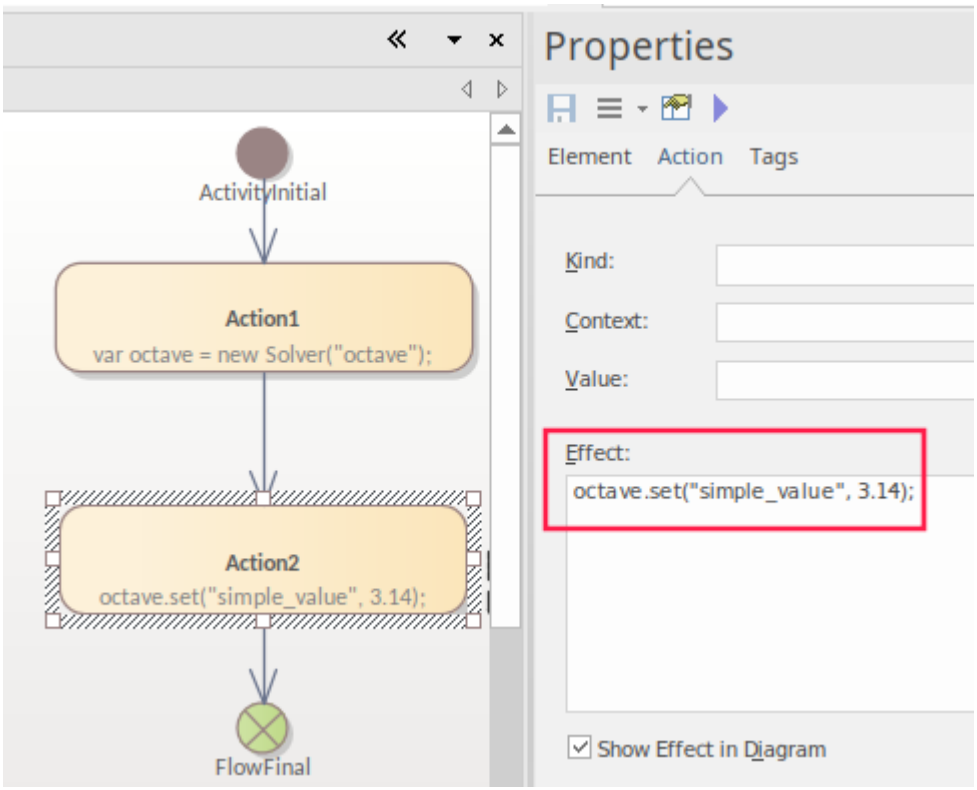
Initialisation du Solveur

Le script pour initialiser un Solveur dans un diagramme d'Activité peut être placé dans l'Effet d'une Action , généralement l'Effet de la première Action sortant de l'ActivitéInitial.



Attribution de valeurs et exécution de commandes

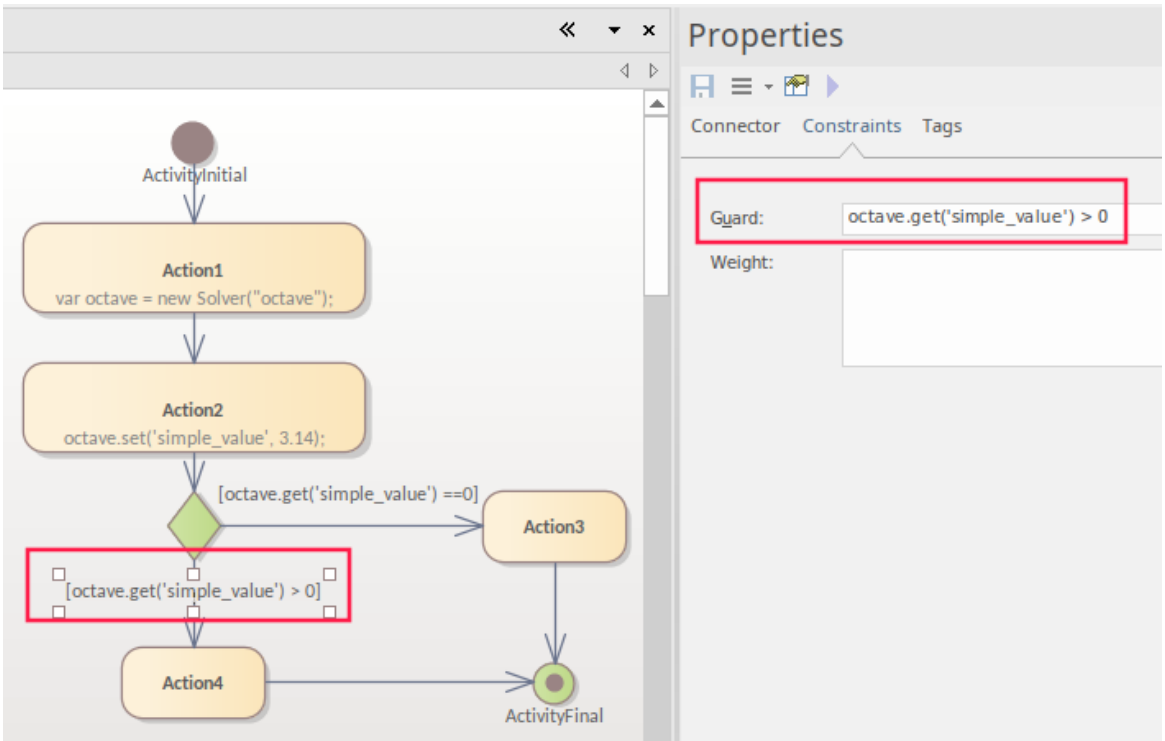
Pour affecter une valeur , utilisez les fonctions octave.set() ou matlab.set(). Pour diagrammes d'activité, la fonction peut être placée dans l'effet d'une Action .



Lors de l'exécution de commandes MATLAB ou Octave à l'aide de la fonction .exec(), les commandes peuvent à nouveau être placées dans l'effet de l' Action .

Branchement conditionnel

Pour la création de branches conditionnelles dans une Statemachine , la condition peut être placée dans la garde d'un ControlFlow et contenir un script qui appelle n'importe quelle fonction MATLAB ou Octave. Par exemple :



Obtenir des résultats

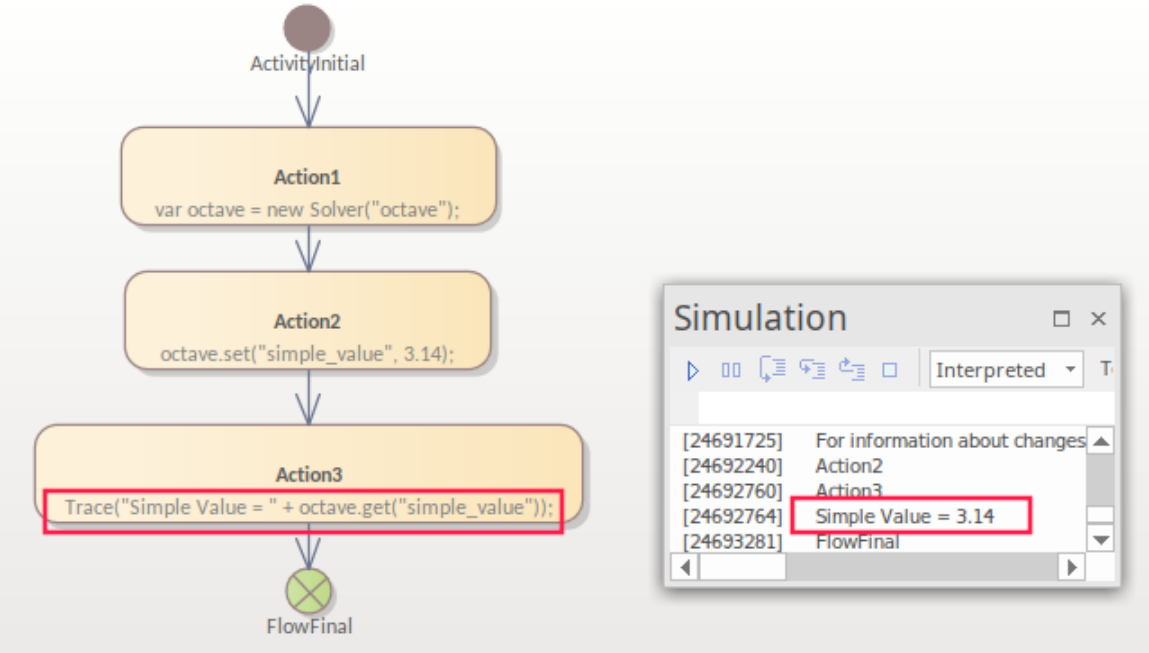
Pour renvoyer les résultats des appels de fonctions externes, utilisez la fonction .get() de Solveur . Il existe trois options principales pour la manière dont les résultats sont ensuite transmis du script à l'utilisateur :

- Tracer()
- Parcelle
- Affichage Win32

Avec les options Trace et Win32, vous devez renvoyer une copie locale dans votre JavaScript à l'aide de la fonction Solveur .get(). L'image précédente inclut un exemple d'utilisation de la fonction .get() dans un Guard.

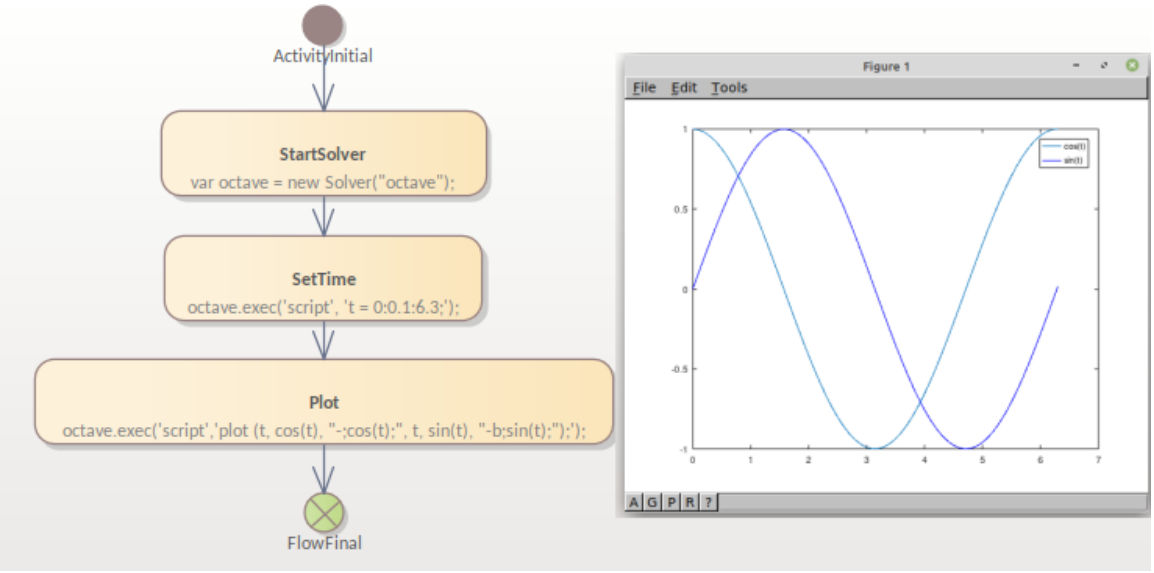
Utilisation de Trace

La commande Trace() est utile lors de l'écriture et du débogage d'une simulation, car elle vous permet de vérifier les résultats de votre script à différentes étapes. Les résultats sont affichés dans la fenêtre Simulation .



Exécution de complots

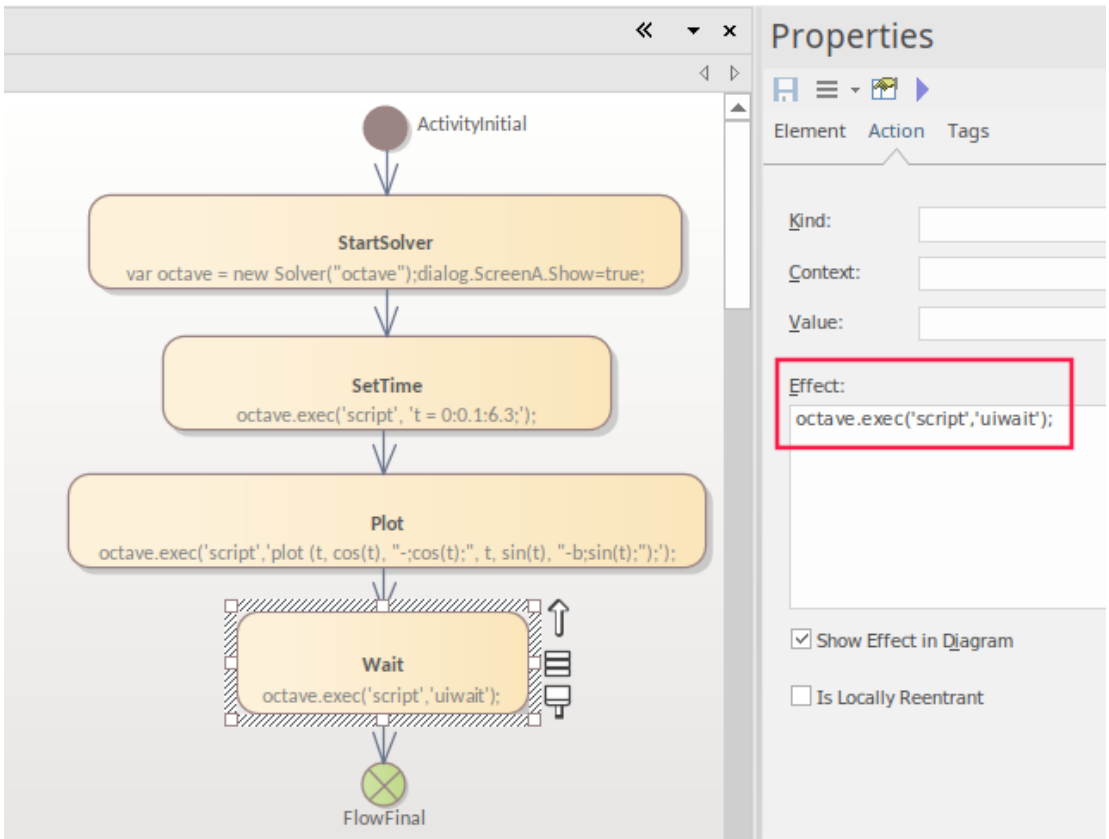
Octave et MATLAB accordent une grande importance à la génération de tracés, car il s'agit d'une méthode clé pour générer des résultats. Pour générer un tracé, vous utilisez la fonction .exec() de Solveur pour appeler une génération de tracé.



Tenir le terrain

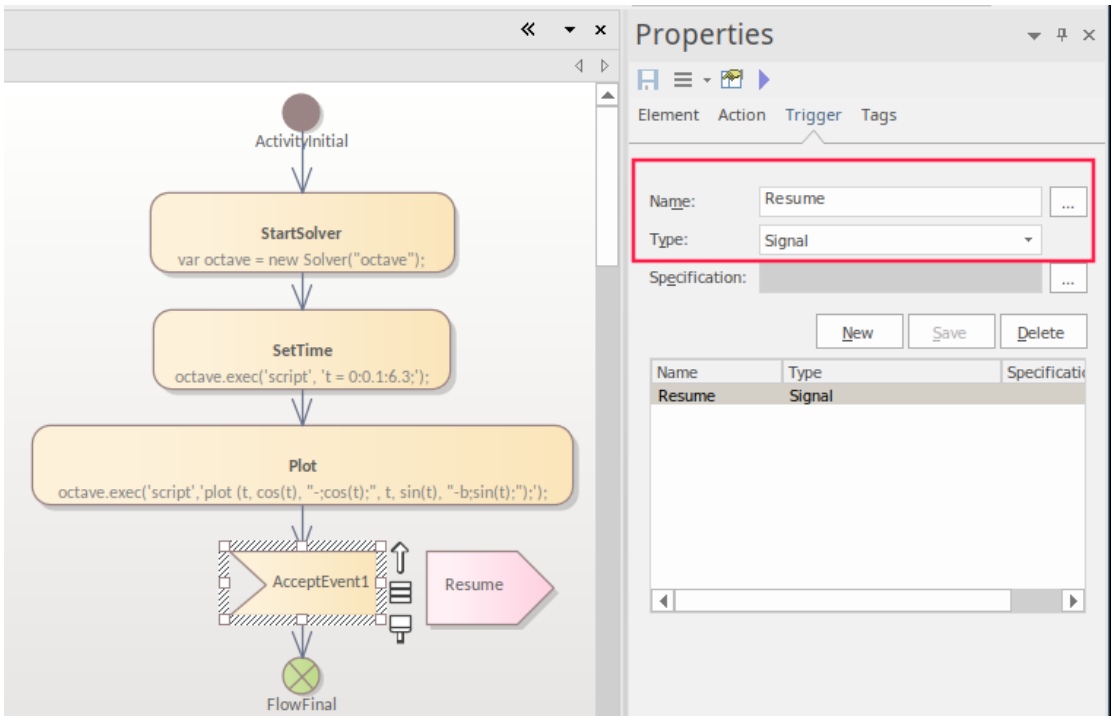
Lors de l'exécution d'un tracé dans une simulation, si la simulation n'est pas mise en pause, le tracé ne sera visible que brièvement. Pour diagrammes d'activité, il existe deux options pour mettre en pause le flux pendant que le tracé est visualisé :

1. En utilisant la fonction uiwait dans Octave ou MATLAB, en la définissant dans l'effet de l' Action . Voici un exemple utilisant Octave :
   octave.exec('script','uiwait');



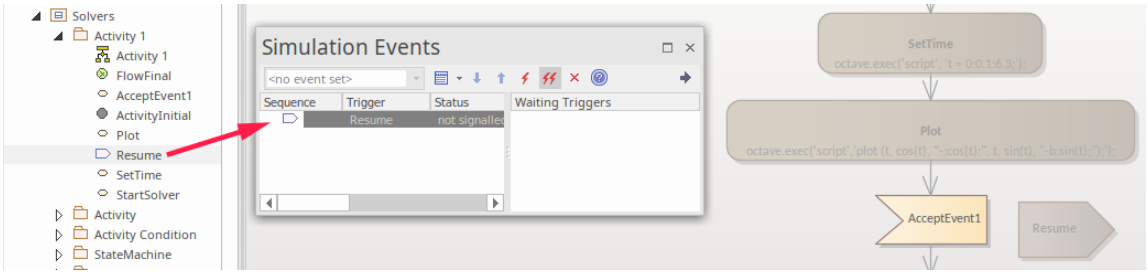
2. Définissez une Action de type AcceptEvent pour mettre en pause la simulation après la génération du tracé, ce qui

nécessite de définir une référence à un Déclencheur d'activité sur l'AcceptEvent. Celui-ci peut être créé et référencé à l'aide du bouton ... , qui affiche la dialogue « Sélectionner Déclencheur ». Cliquez sur le bouton Ajouter un nouveau pour créer le Déclencheur .



Pour progresser au-delà de l'AcceptEvent, vous devez :

- Faites glisser le Déclencheur (Reprise) du Navigateur vers la fenêtre Simulation Événements
- Double-cliquez sur cette transition



Utilisation de l'interface Win32

Lors de l'utilisation de l'interface Win32, les étapes à suivre sont les suivantes :

1. Créer une dialogue Win32.
2. Définissez une ligne de script pour l'ouvrir.
3. Obtenir une valeur d'un champ dans le dialogue .
4. Transmettez cette valeur au Solveur .
5. Utilisez un bouton pour déclencheur l'intrigue.

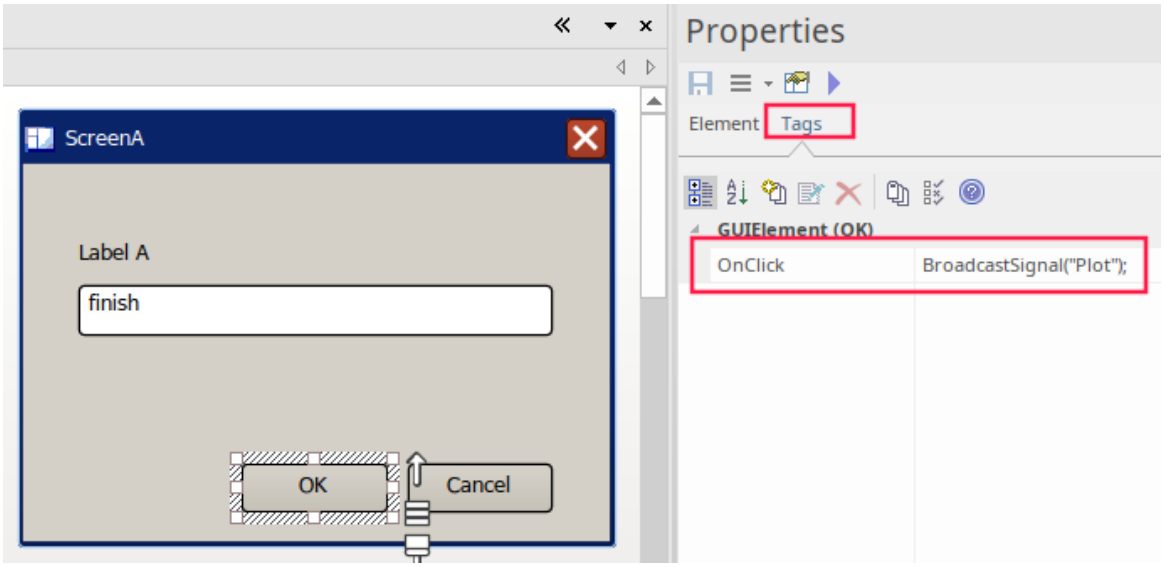
Voici les étapes pour configurer une interface Win32 :

1. Créez un 'Starter Win32 Model' à l'aide de la dialogue ' Constructeur de Modèle ' (Ctrl+Shift+M).
2. Modifiez le nom de la <<Dialogue d'écran Win32>> en « ScreenA ».
3. Modifiez le nom de « Contrôle d'édition A » par quelque chose de plus significatif, comme « Terminer ».

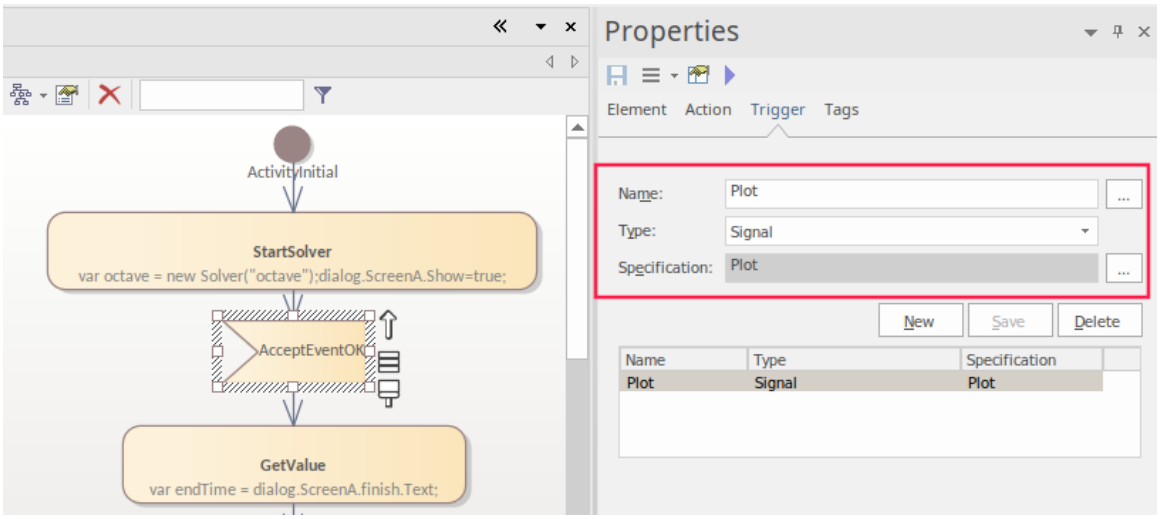
4. Appuyez sur Alt+7 sur State1 pour ajouter un appel pour ouvrir le dialogue , dans le script de l'opération d'entrée : dialogue .ScreenA.Show=true;

Pour obtenir une valeur d'entrée utilisateur valide, l'utilisateur doit cliquer sur le bouton OK , donc à partir du bouton OK , vous devez diffuser un événement pour qu'un Déclencheur démarre le processus. Pour recevoir le Déclencheur vous définissez un Signal et une Transition à déclencher par cette Diffusion.

Voir les étapes pour créer et définir le Déclencheur et le Signal 'End', comme indiqué dans *Holding the Plot* plus haut. Dans ce cas, nous définissons la même chose, mais pour un nouveau Déclencheur appelé 'Plot'. Cette illustration envoie un BroadcastSignal('Plot') en utilisant la Valeur Étiquetée OnClick sur un bouton.



Sur le ControlFlow quittant StartSolver, il y a maintenant une Action « AcceptEvent ». Celle-ci est configurée pour être déclenchée comme indiqué :



Dans l'onglet ' Déclencheur ' de la fenêtre Propriétés , créez un Déclencheur , définissez le type de déclencheur sur 'Signal' et créez un signal pour celui-ci. Pour plus de détails, consultez la rubrique d'aide *Simulation d'Interface Utilisateur Win32* .

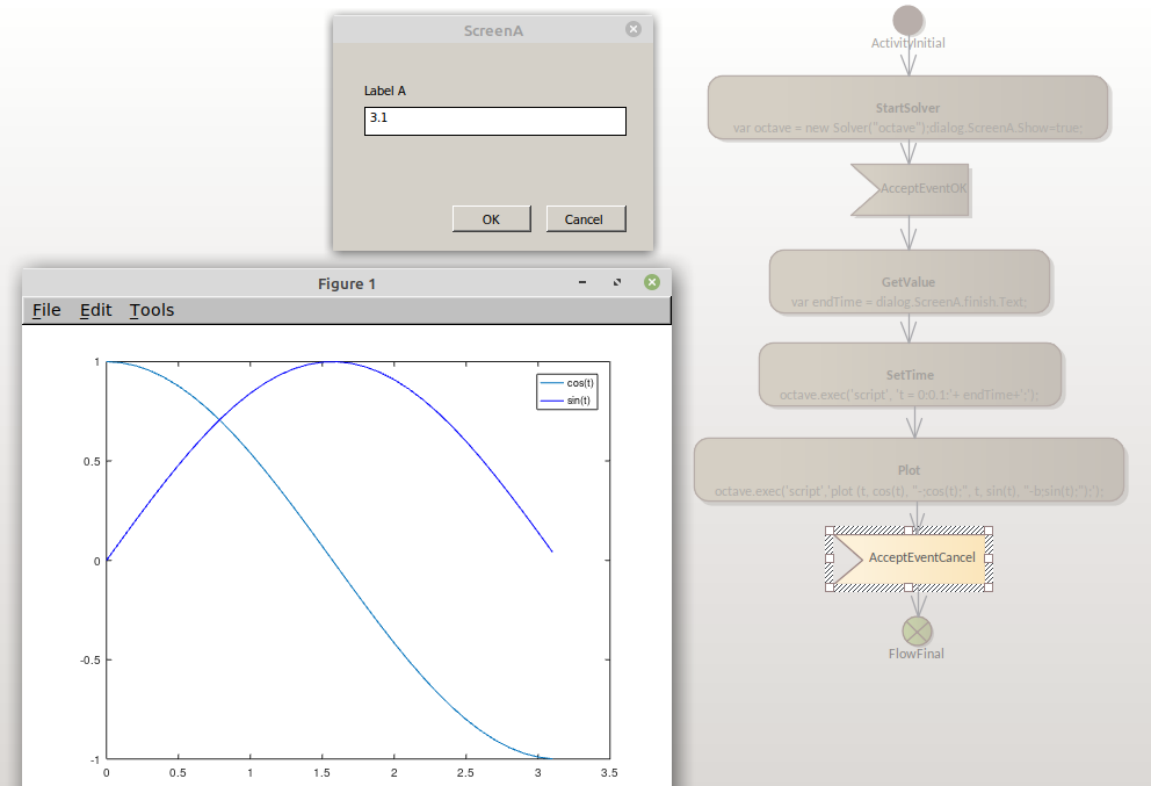
Le bouton Annuler est configuré de la même manière que le bouton OK (avec un signal Broadcast) et les références Déclencheur sont définies sur AcceptEventCancel pour utiliser le Déclencheur /Signal 'annuler'.

Exécution de la Simulation

Pour exécuter la simulation :

- Sélectionnez le ruban Simuler
- Cliquez sur ' Exécuter Simulation > Démarrer '

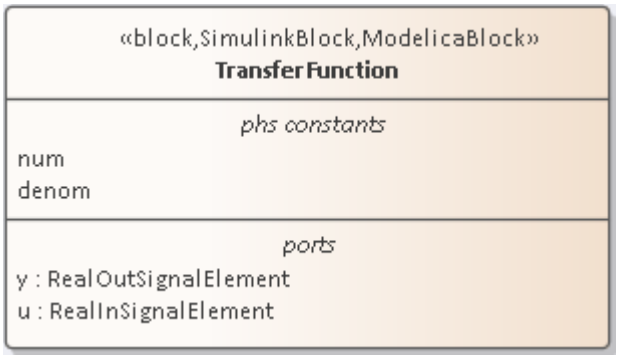
Lorsque vous entrez une valeur et cliquez sur le bouton OK , cela renvoie :



Simulation SysPhS

La spécification *SysML Extension for Physical Interaction and Signal Flow Simulation* (SysPhS) est une spécification de Object Management Group (OMG) qui étend SysML pour fournir une plate-forme modélisation commune permettant de définir des modèles cohérents. Ces modèles peuvent être traduits vers l'une des deux principales plates-formes de simulation, Modelica et Simulink/Simscape de MATLAB.

Les stéréotypes OMG SysPhS vous aident à définir les caractéristiques de la simulation du modèle au sein même du modèle, plutôt que dans une spécification de configuration de simulation. Ils offrent une meilleure visibilité du type d' object ou de propriété dans la fenêtre Navigateur et la fenêtre Propriétés , ainsi que dans le diagramme avec des compartiments d'éléments spécifiques pour les types de propriétés et pour les valeurs initiales.



La norme est représentée dans Enterprise Architect par le profil OMG SysPhS, ainsi que :

- Bibliothèques SysPhS d'éléments pour le flux de signaux et pour l'interaction physique (nécessaires pour effectuer des simulations selon la norme SysPhS)
- Une page dédiée à la boîte à outils
- Une large gamme de motifs d'éléments de composants à partir desquels générer des éléments de simulation courants tels que des composants électroniques, logiques et fluides ; les motifs font référence aux blocs de bibliothèque dans les bibliothèques standard OpenModelica ou Simulink
- Fonctionnalités pour simuler des tracés à l'aide de Modelica ou de Simulink, Simscape et Stateflow de MATLAB.

Fonctionnalités de SysPhS

Fonctionnalité	Description
Référencement des bibliothèques SysPhS	Les principales ressources pour travailler avec SysPhS sont les bibliothèques Simulation SysPhS, qui incluent des ressources réutilisables que vous devez référencer dans votre modèle.
Boîte à outils SysPhS	Les pages SysPhS de la boîte à outils Diagramme contiennent des éléments SysML de base pour OpenModelica et MATLAB Simulink.
Motifs SysPhS	Les Motifs SysPhS fournissent des blocs SysPhS prédéfinis qui font référence à des composants MATLAB et Modelica équivalents. Ces blocs simples peuvent être utilisés comme points de départ lorsque vous travaillez avec des modèles SysPhS.
Composants SysPhs	Les composants SysPhS vous permettent de définir des références aux composants Modelica et Simulink.
Simulation	Vous pouvez définir des modèles IBD ou Paramétriques avec des informations supplémentaires pour piloter une simulation, puis utiliser la configuration de simulation pour générer le modèle dans Modelica, Simulink ou Simscape, afin de

	produire un graphique des résultats.
Exemples de SysPhS	Il existe plusieurs exemples d'utilisation de SysPhS pour la configuration de simulations.
Mise à jour de SysMLSim pour SysPhys	Vous pouvez mettre à jour les anciennes configurations de simulation (antérieures à Enterprise Architect 15.2) pour refléter l'utilisation de la norme SysPhS.

Options

Les options supplémentaires pour les variables et les constantes, telles que isContinuous et isConserved, sont automatiquement définies comme Valeur Étiquetés , évitant ainsi de devoir les définir dans la spécification de configuration. Ces options sont également visibles sur le Bloc lui-même et dans la fenêtre Propriétés ancrée.

Vidéos

- SysPhS Motifs for the Simulation of an Electrical Circuit
- Simulating Digital Electronics using SysPhS and Modelica

Référencement des bibliothèques Simulation SysPhS

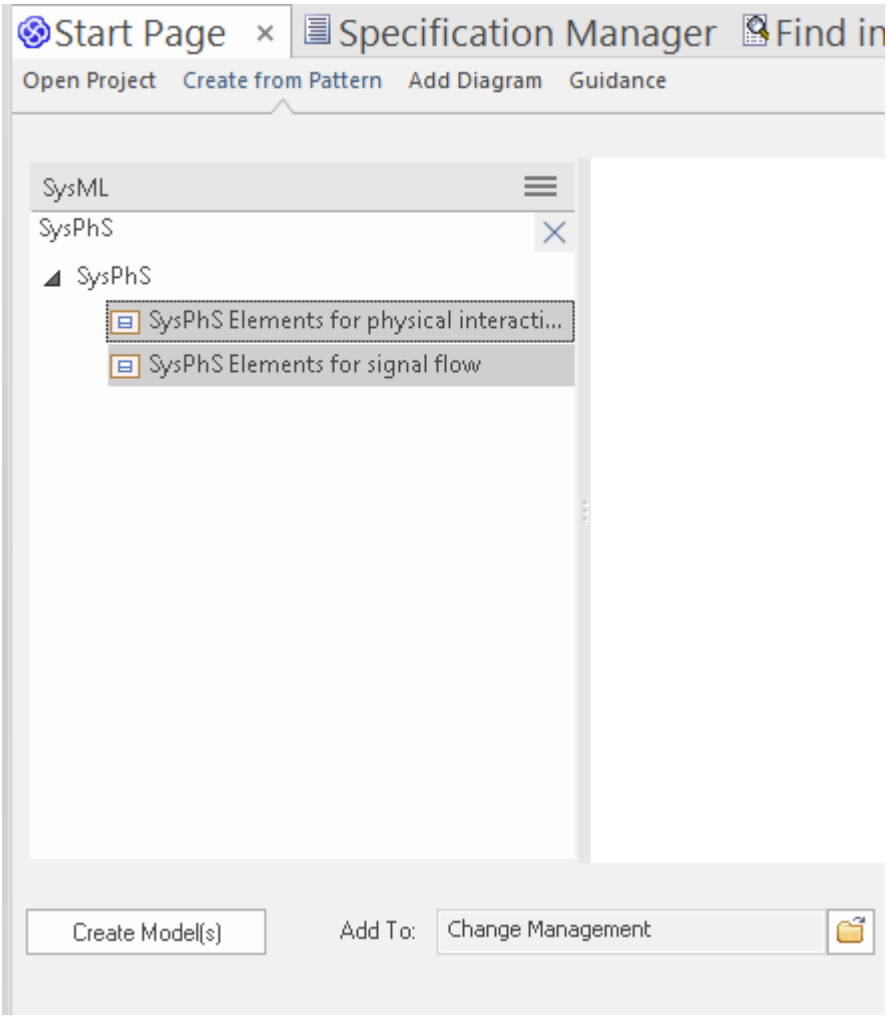
La spécification SysPhS fournit une définition d'une Bibliothèque de composants indépendante de la plate-forme qui se compose de deux groupes de composants : Flux de signaux et Interaction physique. Comme cette bibliothèque de composants est une ressource sous-jacente utilisée pour modélisation , une copie de cette bibliothèque doit être contenue dans le référentiel sur lequel vous travaillez, pour référence dans vos modèles SysPhS.

Afin de réaliser des simulations sous la norme SysPhS, vous devez télécharger deux bibliothèques SysPhS depuis le Constructeur de Modèle :

- Éléments SysPhS pour l'interaction physique et
- Éléments SysPhS pour le flux de signaux
- Ces bibliothèques peuvent être utilisées sur plusieurs modèles SysPhS créés dans le référentiel. Il est recommandé de les télécharger dans un Paquetage unique, après quoi une référence unique à ce Paquetage peut être définie dans n'importe quel modèle SysPhS.

Pour télécharger les bibliothèques :

1. Créez un Paquetage pour contenir les bibliothèques.
2. Sélectionnez le Paquetage dans le Navigateur .
3. Réglez la Perspective ( , coin supérieur droit de l'espace de travail) sur « Ingénierie des Systèmes > SysML ».
4. Appuyez sur Ctrl+Shift+M pour ouvrir la dialogue Constructeur de Modèle .
5. Dans la barre de filtre, sous le nom de la perspective, saisissez « SysPhS ».
6. Cliquez sur « Éléments SysPhS pour l'interaction physique » et Ctrl+clic sur « Éléments SysPhS pour le flux de signaux ».



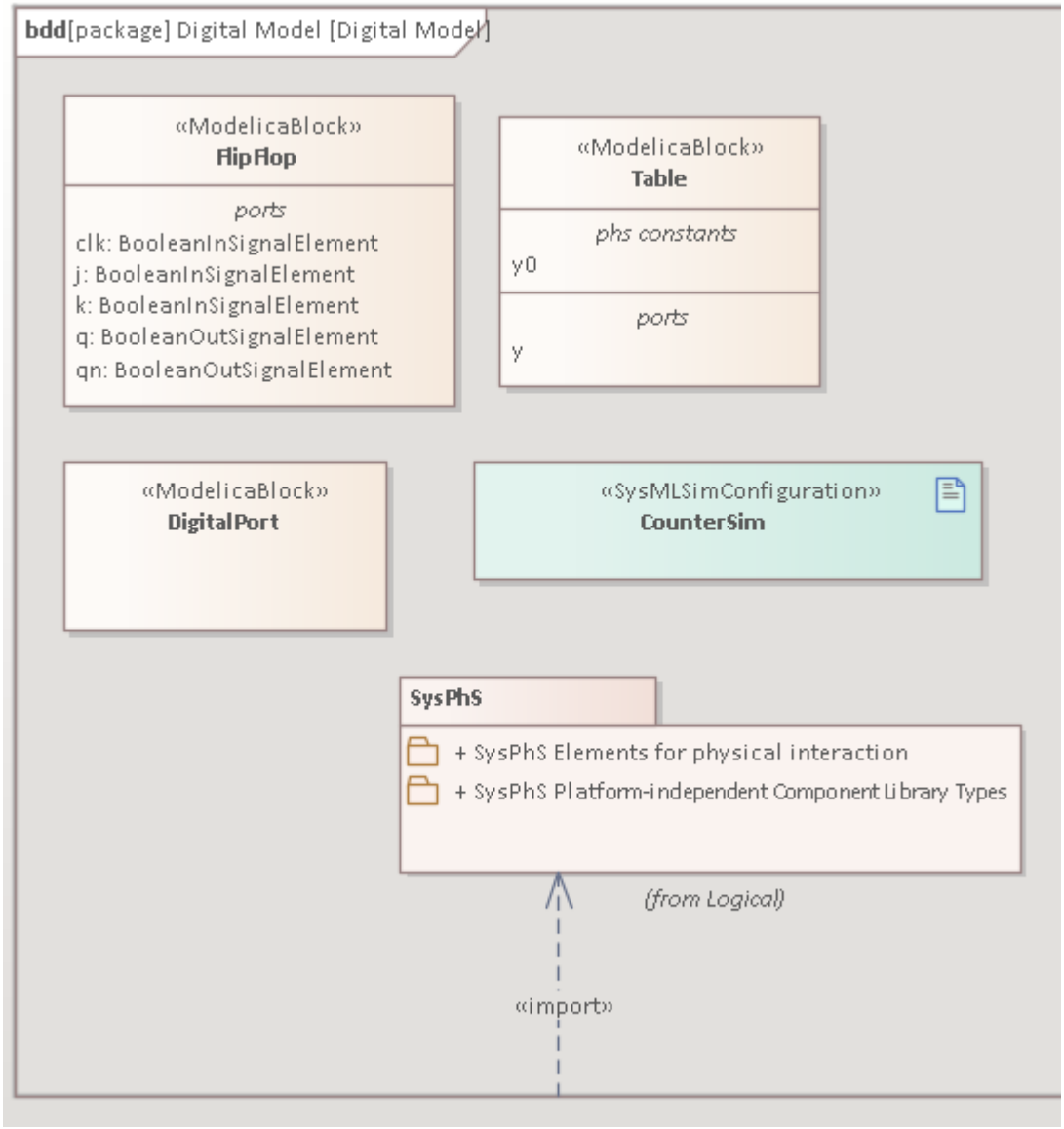
7. Cliquez sur le bouton Créer Modèle .

Les bibliothèques sont chargées dans le Paquetage sélectionné.

Pour chaque simulation SysML utilisant SysPhS, le cadre diagramme Bloc Definition doit avoir une référence à la bibliothèque Paquetage . Pour définir cela, liez la bibliothèque Paquetage au cadre du diagramme SysML avec un connecteur Import :

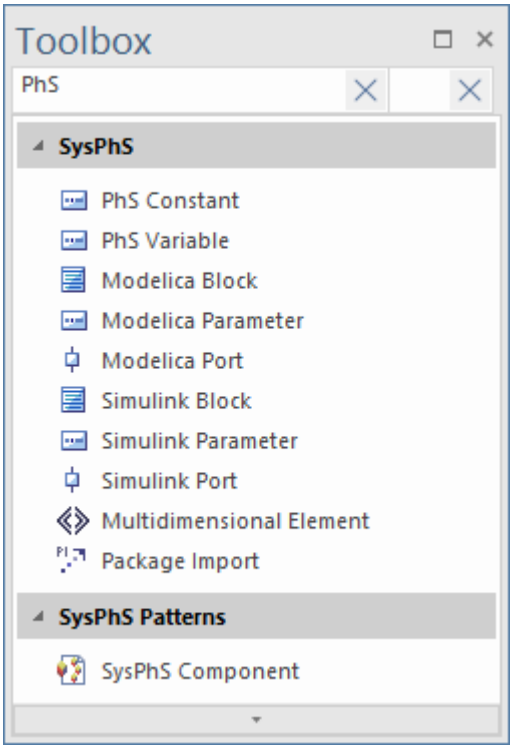
1. Créez un diagramme de définition Bloc pour la simulation.
   Cela aura automatiquement un cadre Bordure autour des éléments.
2. Faites glisser la bibliothèque SysPhS Paquetage sur le diagramme .
   Une prompt s'affiche concernant le type d'opération « ajouter » à effectuer.
3. Sélectionnez l'option «Élément Paquetage ».
   L'élément de bibliothèque Paquetage est ajouté au diagramme dans le cadre Bordure .
4. Cliquez-droit sur le cadre Bordure pour afficher le menu contextuel, et assurez-vous que l'option « Sélectionnable » est cochée.
5. Changez la boîte à outils vers la page « SysPhS » :
   Cliquez sur l'icône de relation ' Paquetage Import' et faites glisser le curseur entre le cadre Bordure et la bibliothèque Paquetage pour créer un connecteur Import.
6. Enregistrez le diagramme (appuyez sur Ctrl+S).

Voici un exemple de référence Paquetage SysPhS utilisant le connecteur <<import>> :


bdd[package] Digital Model [Digital Model]
«ModelicaBlock»
FlipFlop
ports
clk: BooleanInSignalElement
j: BooleanInSignalElement
k: BooleanInSignalElement
q: BooleanOutSignalElement
qn: BooleanOutSignalElement
«ModelicaBlock»
Table
phs constants
y0
ports
y
«ModelicaBlock»
DigitalPort
«SysMLSimConfiguration»
CounterSim
SysPhS
+ SysPhS Elements for physical interaction
+ SysPhS Platform-independent Component Library Types
(from Logical)
«import»

Utilisation de la boîte à outils SysPhS

Le profil SysPhS fournit une page de boîte à outils dédiée, contenant des éléments SysML de base pour OpenModelica et MATLAB Simulink et, sur une page « Motifs SysPhS », l'icône « Composant SysPhS ».



Éléments SysPhS

Icône d'élément	Description
Constante PhS	La constante PhS définit des valeurs qui sont constantes lors d'une simulation exécuter . Pour Modelica, ceux-ci correspondent à des variables de paramètres. Pour Simscape, ceux-ci correspondent à des paramètres (constants).
Variable PhS	La variable PhS définit des valeurs qui sont variables lors d'une simulation exécuter . Pour Modelica, PhSVariables avec : • isContinuous=true correspond aux composants *continus* de Modelica • isContinuous=false correspond à des composants *discrets* Pour Simulink, les PhSVariables correspondent aux *variables* Simscape.
Bloc Modelica	Le type d'élément Bloc Modelica est utilisé pour définir un composant correspondant dans la bibliothèque Modelica. Pour définir un Bloc Modelica correspondant à un composant de la bibliothèque Modelica, il faut que la valeur de la propriété Modelica SysPhS Name soit définie sur le nom complet du composant de Modelica (le chemin de classe).
Paramètre Modelica	Un élément de type Paramètre Modelica permet de définir un paramètre d'un

	composant de Bibliothèque Modelica. Ceux-ci sont définis dans la fenêtre Propriétés > onglet Elément > « ModelicaParameter » (de SysPhS) qui possède deux paramètres : • Nom : Correspond au nom du paramètre Modelica (Nom des paramètres) • Valeur: Le typage de ce paramètre et sa valeur initiale peuvent être définis dans l'onglet Propriétés > Propriété dans les champs : • Type • Initial
Port de Modelica	Un port Modelica correspond à un port défini dans la Bibliothèque Modelica. Le nom du port Modelica correspondant est défini dans : Propriétés > Élément > « *ModelicaPort» (depuis SysPhs)* > Nom.
Bloc Simulink	Le type d'élément Simulink Bloc est utilisé pour définir un composant correspondant dans la bibliothèque Simulink. Pour définir un Bloc Simulink qui correspond à un composant de la bibliothèque Simulink, il faut que la valeur de la propriété Simulink SysPhS Name soit définie sur le nom complet du composant de Simulink (trouvé dans l'explorateur Modèle Simulink sous « Contenu de »).
Paramètre Simulink	Un élément de type Simulink Parameter permet de définir un paramètre d'un composant Simulink Bibliothèque . Ceux-ci sont définis dans la fenêtre Propriétés > onglet Elément > « SimulinkParameter » (de SysPhS), qui possède deux paramètres : • Nom : correspond au nom du paramètre dans la bibliothèque Simulink • Valeur: Le typage de ce paramètre et sa valeur initiale peuvent être définis dans l'onglet Propriétés > Propriété dans les champs : • Type • Initial
Port Simulink	Un port Simulink correspond à un port défini dans la Bibliothèque Simulink. Le nom du port Simulink correspondant est défini dans : Propriétés > Élément > *« SimulinkPort » (depuis SysPhs*) > Nom.
Élément multidimensionnel	Dans modélisation des systèmes, alors qu'un vecteur peut être spécifié à l'aide de la multiplicité, pour les tableaux multidimensionnels, nous avons besoin de plusieurs multiplicités. Le stéréotype MultidimensionalElement permet cela, en prenant en charge efficacement un tableau de multiplicités définissant les dimensions. Ceci est appliqué à un Bloc en faisant glisser l'icône de l'élément multidimensionnel.
Importation de Paquetage	Le connecteur Paquetage Import est nécessaire pour définir une référence du Bloc principal aux bibliothèques de simulation SysPhS. Pour plus de détails, consultez la rubrique d'aide *Référencement des bibliothèques Simulation SysPhS* .

Motifs SysPhS

La spécification OMG SysPhS contient une liste de composants SysPhS communs à Modelica et Simulink. Enterprise Architect fournit cette série de composants sous la forme d'un ensemble de *Motifs SysPhS.* Ces composants de base sont des éléments de départ utiles lors de la création d'un nouveau modèle. Pour plus de détails sur leur utilisation, consultez

la rubrique d'aide *Utilisation Motifs SysPhS* .

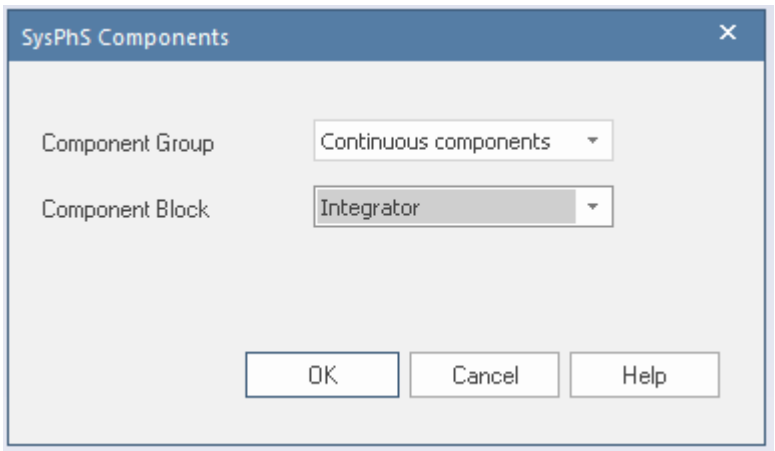
Utilisation des Motifs SysPhS

La spécification OMG SysPhS répertorie une série de composants SysPhS sous la forme d'une *Bibliothèque de composants indépendants de la plate-forme* . Les *Motifs SysPhS* Enterprise Architect fournissent cette série de composants communs à Modelica et à Simulink. Lorsqu'un élément est sélectionné, il crée un Bloc typé à la fois pour Modelica et Simulink. Ensuite, lors de l'exécution d'une simulation définie sur Modelica ou Simulink, ce type d'élément sera créé dans l'outil correspondant.

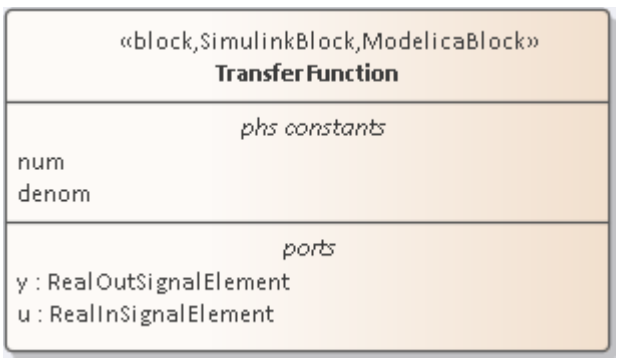
Le regroupement général de ces composants est

- Composants continus
- Composants discrets
- Composants non linéaires
- Composantes Mathématiques
- Sources et puits
- Composants de routage
- Composants logiques
- Composants électriques

Lorsque vous faites glisser l'icône « Composant SysPhS » sur le diagramme , la dialogue « Composants SysPhS » s'affiche.



Dans cette dialogue vous sélectionnez d'abord la catégorie d'élément à créer, puis le type d'élément à créer, tous deux en sélectionnant dans une liste déroulante dans le champ « Groupe de composants » ou « Bloc de composants ». La liste déroulante de chaque champ est renseignée à partir des bibliothèques standard OpenModelica et Simulink. Par exemple :



Notez que l'élément possède des stéréotypes pour Simulink et Modelica, il peut donc être utilisé dans l'un ou l'autre de ces outils. Pour chaque élément, les Propriétés et les ports corrects pour le type d'élément sont automatiquement inclus dans l'élément, en tant qu'éléments dans les compartiments d'élément. Vous pouvez faire glisser les éléments structurels réels de la fenêtre Navigateur vers l'élément si vous préférez leur présence physique sur le diagramme .

Pour une liste de tous les composants pris en charge et plus de détails, voir la section : *11.3.2 Composants à valeur réelle*

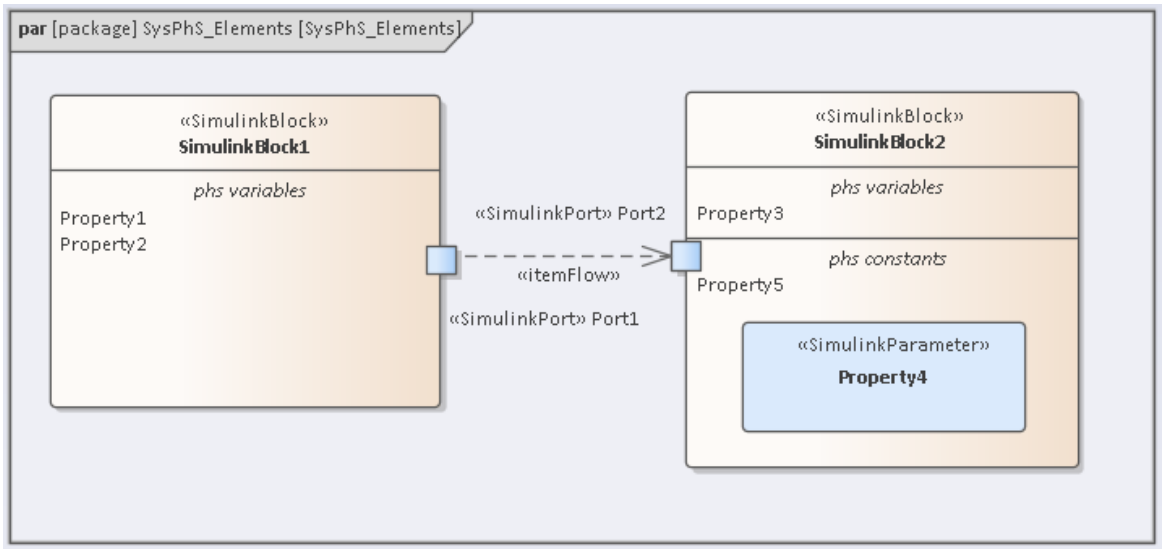
dans le *PDF OMG SysPhS 1.0.*

Travailler avec les composants SysPhS

Si vous travaillez avec un modèle d'exemple conservé dans Modelica ou Simulink et que vous souhaitez faire référence, dans Enterprise Architect , à des composants qui existent déjà dans ce modèle, vous pouvez faire glisser les types d'éléments Bloc , Paramètre ou Port appropriés de la boîte à outils sur un diagramme pour créer les éléments de référence.

Affichage Propriétés et des pièces

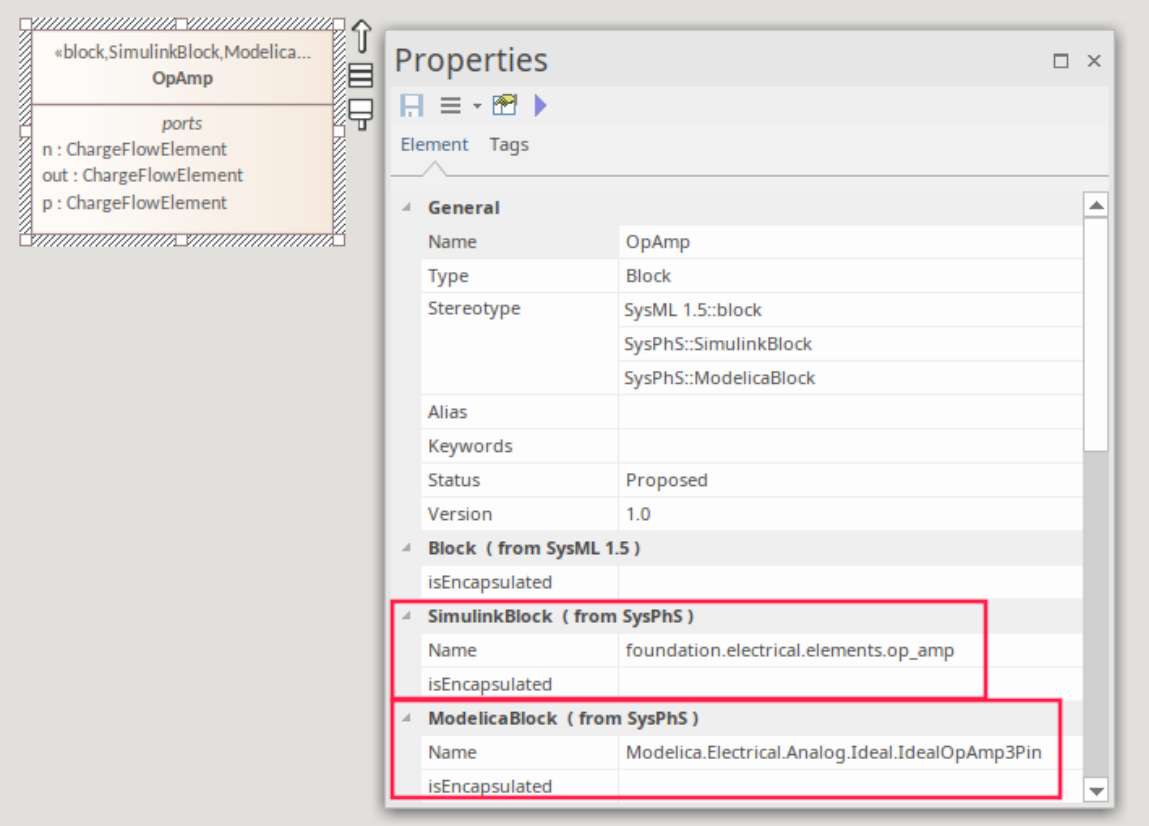
Il est important de note lorsque vous travaillez avec Propriétés et des parties dans un diagramme SysML, leur affichage par défaut est celui d' object Part. Ils peuvent être conservés dans ce rendu ou configurés pour s'afficher sous forme de texte dans des compartiments. Voici un exemple des deux :



Propriétés 1, 2, 3 et 5 ont été glissées sur le diagramme en tant qu'éléments qui ont ensuite été supprimés du diagramme (mais pas de la fenêtre Navigateur). Lors de la suppression, elles ont été remplacées sur le diagramme par des entrées de texte dans leurs compartiments Bloc respectifs. La propriété 4 a été laissée sur le diagramme en tant qu'élément Part ; si elle était supprimée du diagramme , elle deviendrait également une entrée de texte dans un compartiment.

Bloc de type élément

Faire glisser un type de bloc Modelica ou Simulink sur un diagramme crée un Bloc SysPhS non spécifique. Pour définir ce Bloc sur un Bloc Modelica ou Simulink spécifique, définissez le champ « Nom » sous le segment **SimulinkBlock (SysPhS)** ou **ModelicaBlock (SysPhS)** dans la fenêtre Propriétés .

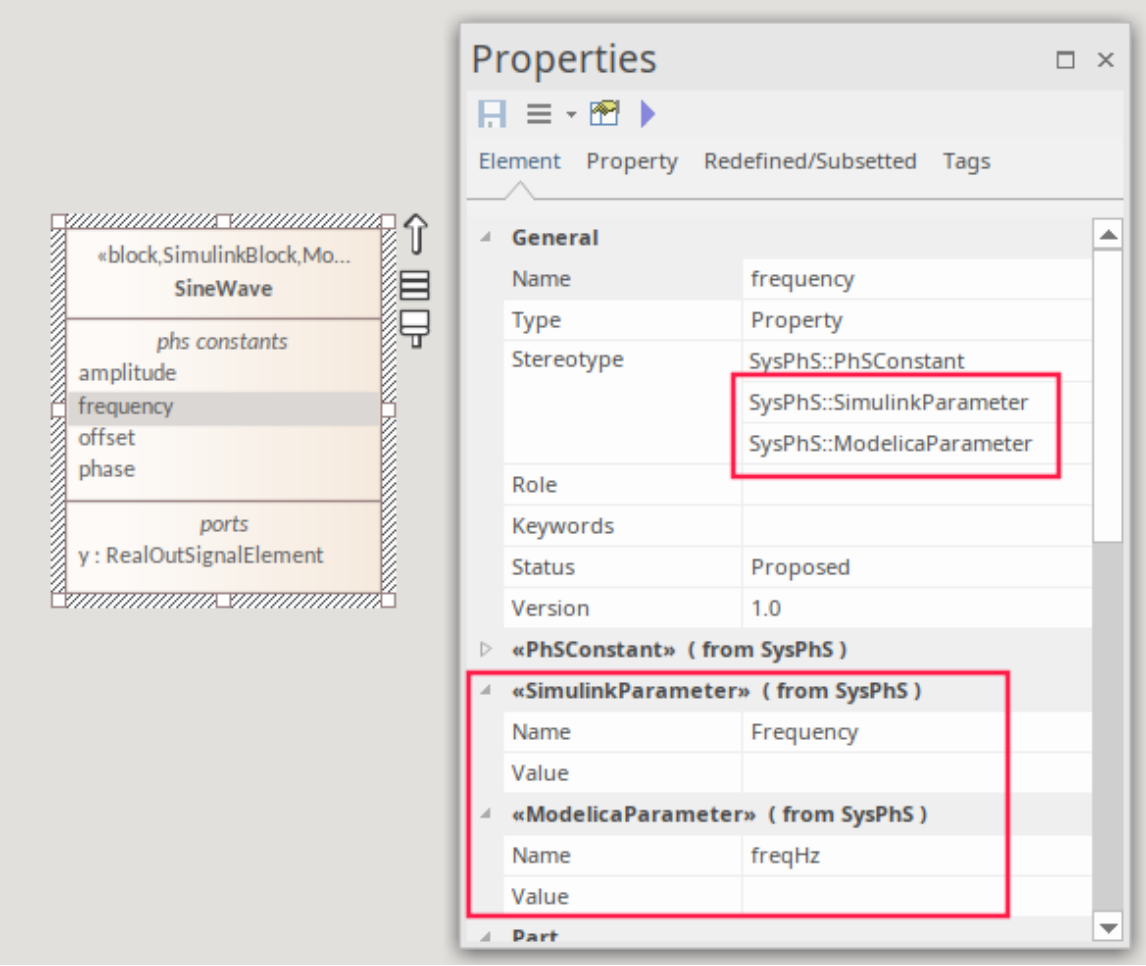


Pour plus de détails sur le référencement des Propriétés Simulink et Modelica dans ces outils externes, consultez les rubriques d'aide *Création de blocs spécifiques à Modelica* et *Création de blocs spécifiques à Simulink* .

Note : comme indiqué dans l'image, les deux stéréotypes SysPhS peuvent tous deux être appliqués si vous souhaitez simuler le modèle dans les deux outils externes. Consultez la rubrique d'aide *Définition des blocs comme étant à la fois Modelica et Simulink* .

Élément de paramètre de type

L'élément de type Parameter crée des éléments Property avec les stéréotypes SimulinkParameter ou ModelicaParameter. Si vous supprimez les éléments du diagramme , ils sont répertoriés dans le compartiment *des constantes phs* de l'élément Bloc parent. Voici un exemple d'un paramètre SysPhS défini à la fois sur Modelica et Simulink, affichant un stéréotype pour chacun ainsi qu'une référence au nom du paramètre du produit respectif dans les champs « Nom ».

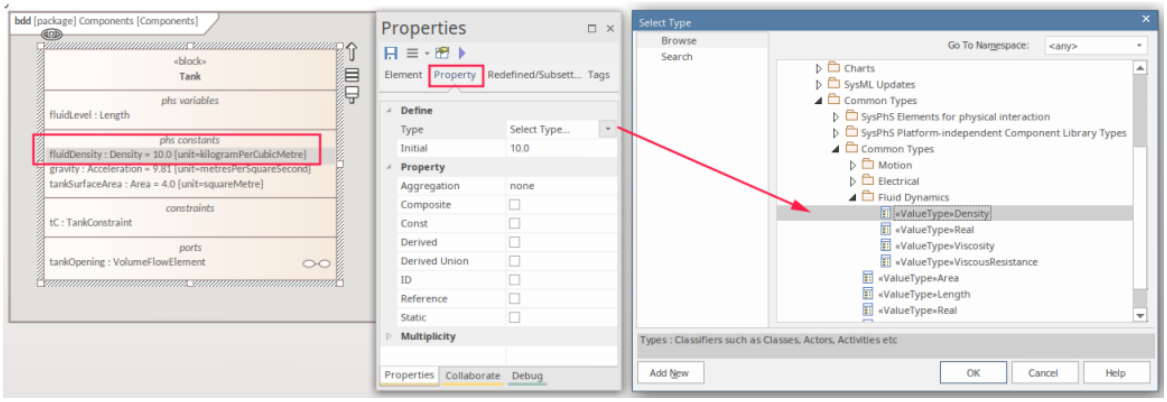


PhsConstante et PhsVariable

Pour définir les propriétés constantes et variables d'un élément Bloc , vous pouvez faire glisser les icônes « Constante PhS » et « Variable PhS » sur un élément du diagramme . Là encore, si vous supprimez les éléments du diagramme ils sont répertoriés dans le compartiment des *constantes PhS* ou *des variables PhS* .

Lors de la définition de la valeur des constantes dans le modèle, les valeurs peuvent être définies dans le Bloc ou dans une Partie dérivée du Bloc . Dans le cas, par exemple, de la gravité en tant que constante absolue, il est préférable de la définir dans le Bloc . Les valeurs de la Partie ou Bloc peuvent être modifiées dans les propriétés Simulation .

L'illustration suivante montre la densité du fluide définie pour l'eau, qui pourrait être remplacée dans la pièce ou la simulation pour définir la densité d'un autre fluide (par exemple, l'huile). Lorsqu'un Bloc est utilisé de manière répétitive avec des valeurs différentes - par exemple une résistance de 3,3 kohms et une autre de 5,6 kohms - alors la valeur initiale est mieux définie dans les pièces spécifiques, dans l'IBD, qui sont dérivées d'un Bloc qui n'a pas valeur initiale.



L'onglet ' Propriétés > Propriété' comporte deux champs :

- Type
- Initial

Le type peut être défini comme un type standard ou, comme dans ce cas, comme SysML *ValueType* référencé dans le modèle.

Note que si une PhsConstant ou une PhsVariable a une valeur initiale définie, elle est affichée sur le diagramme dans un compartiment *valeur initiale* de l'élément.

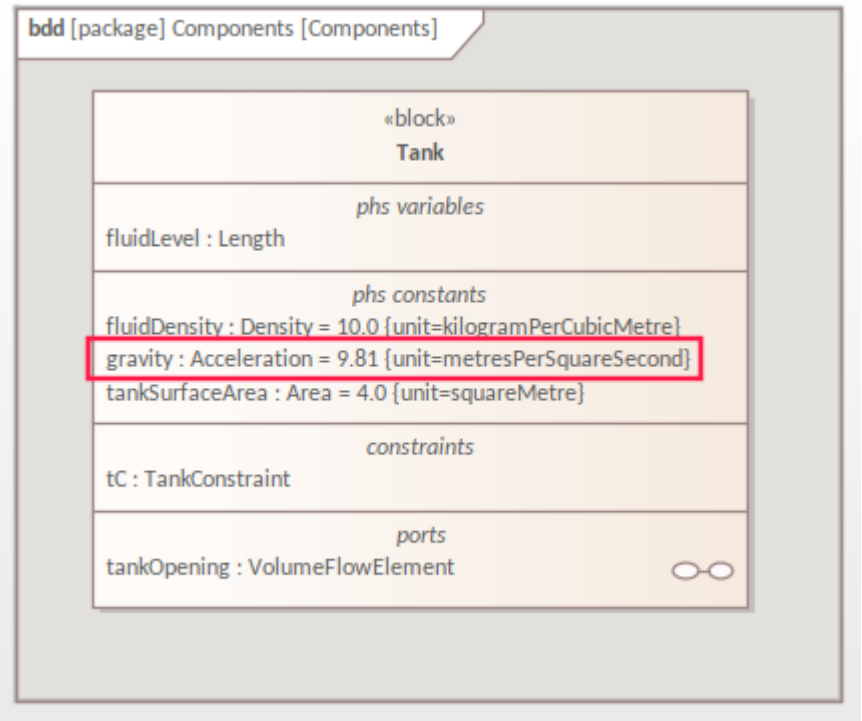
Définition des valeurs

Après avoir défini un Bloc ou une pièce qui fait référence à un composant, que ce soit dans Modelica ou Simulink, vous souhaitez pouvoir définir des valeurs pour les propriétés ou les paramètres de ce composant externe. La décision de placer une valeur dans le Bloc plutôt que dans une pièce dérivée de ce Bloc dépend de la possibilité d'une variation de cette valeur pour chaque pièce.

Bloc Propriétés

Dans l'exemple d'un réservoir d'eau, dans lequel la constante 'Gravité' est définie et fixée pour chaque instance de réservoir, la valeur est mieux placée dans le Bloc source.

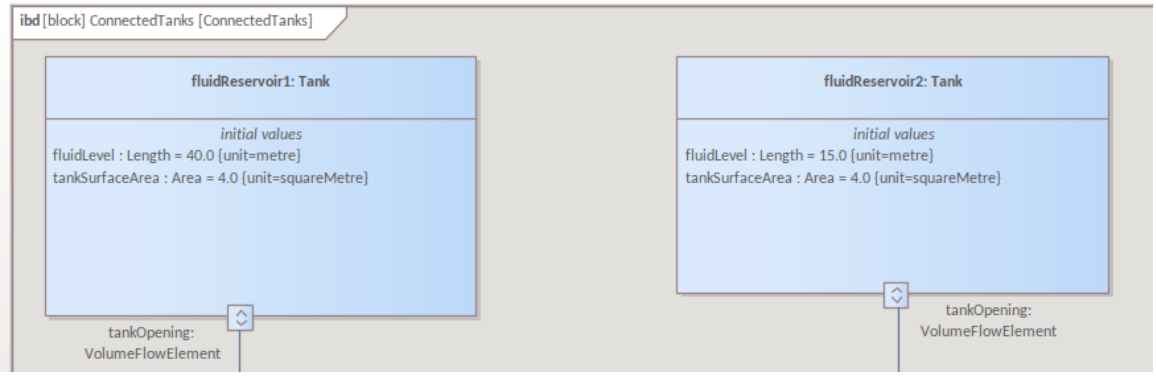
Cette illustration montre également la densité du fluide définie pour l'eau (10 kg/ ^{m3}), qui peut être remplacée soit dans une pièce, soit dans une simulation pour définir la densité d'un autre fluide, tel que l'huile.



Propriétés de la partie

Lorsqu'un Bloc est utilisé de manière répétitive avec des valeurs différentes - par exemple pour représenter une résistance de 3,3 kilohms et une autre de 5,6 kilohms - alors la valeur initiale est mieux définie dans le Bloc comme vide, avec des valeurs individuelles définies dans des parties spécifiques dérivées du Bloc .

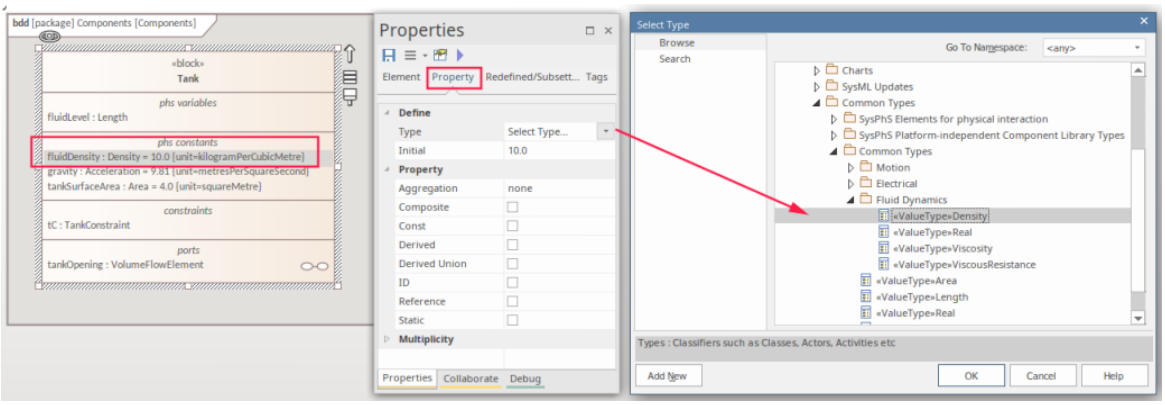
Dans cet exemple, nous avons deux réservoirs avec le même rayon, mais des profondeurs différentes, donc le niveau du fluide (profondeur du réservoir) est différent pour les deux parties du réservoir créées dans cet IBD.



Note que le Type est dérivé du type Bloc FluidLevel.

Définition du Type

Vous pouvez définir Propriétés dans un Bloc pour référencer un Type de valeur spécifique. Cela se définit dans le champ « Type » de l'onglet Propriétés > « Propriété » ; il est préférable de référencer un Type de valeur à l'aide du bouton ... .



Pour plus de détails, consultez la rubrique d'aide *Modélisation de quantité à l'aide de types de valeur* .

Définition des valeurs dans une Simulation

Dans les simulations, il arrive souvent que les variables doivent être définies au moment exécuter plutôt que dans le modèle SysML. C'est là que les jeux de données peuvent être utilisés pour définir un ensemble de valeurs à exécuter dans une série de variantes des simulations. Pour plus de détails, consultez la rubrique d'aide *Modèle d'analyse à l'aide de jeux de données* .

Création de blocs spécifiques à Modelica

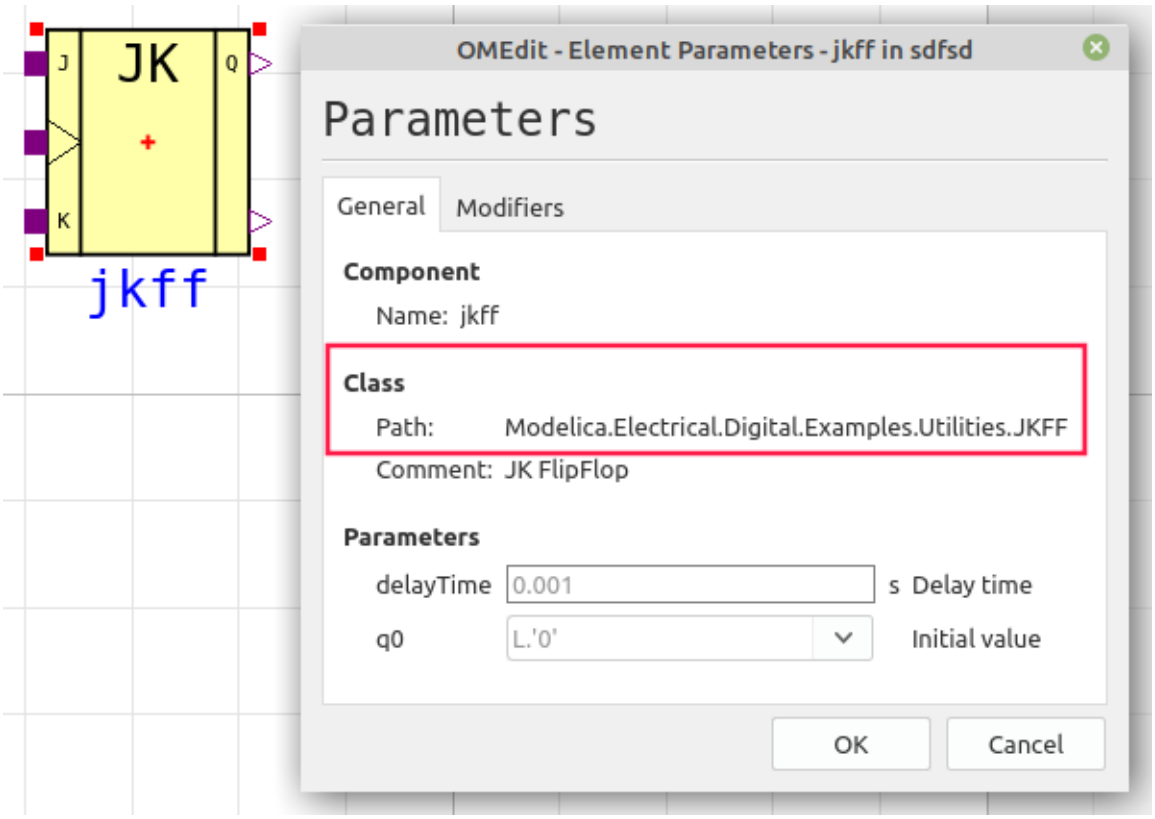
Étant donné la large gamme de différents types de pièces pouvant être utilisés dans Modelica, il y aura des cas où vous devrez modéliser des composants et des pièces Modelica que vous ne pouvez pas dériver des blocs de base fournis dans les motifs de composants SysPhS. Dans de tels cas, vous définissez un composant Enterprise Architect pour référencer le composant Modelica.

Utilisation du chemin de classe Modelica pour un Bloc

Le processus pour référencer un composant Modelica dans un composant Enterprise Architect SysPhS est le suivant :

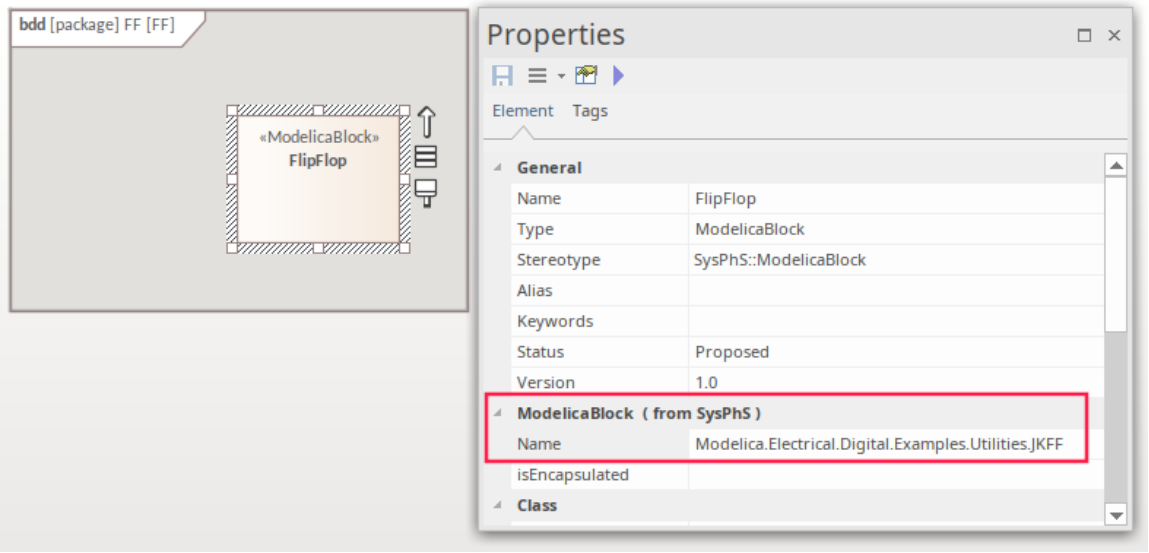
- Dans Modelica, accédez au chemin de classe du composant Modelica
- Copier ce texte
- Dans Enterprise Architect , collez le texte copié dans le champ « Nom » du Bloc Modelica

Cette illustration est un exemple d'un composant Modelica (une bascule JK) et de sa dialogue « Paramètres », montrant le chemin de classe.



- Note : le chemin de classe peut être sélectionné en cliquant et en faisant glisser le curseur sur le chemin, puis en appuyant sur Ctrl+C sur le bloc de texte.

Placez le texte copié dans la fenêtre Propriétés du Bloc SysPhS Modelica, dans le champ « Nom » sous **ModelicaBlock (de SysPhS) :** .



Configuration des ports SysPhS

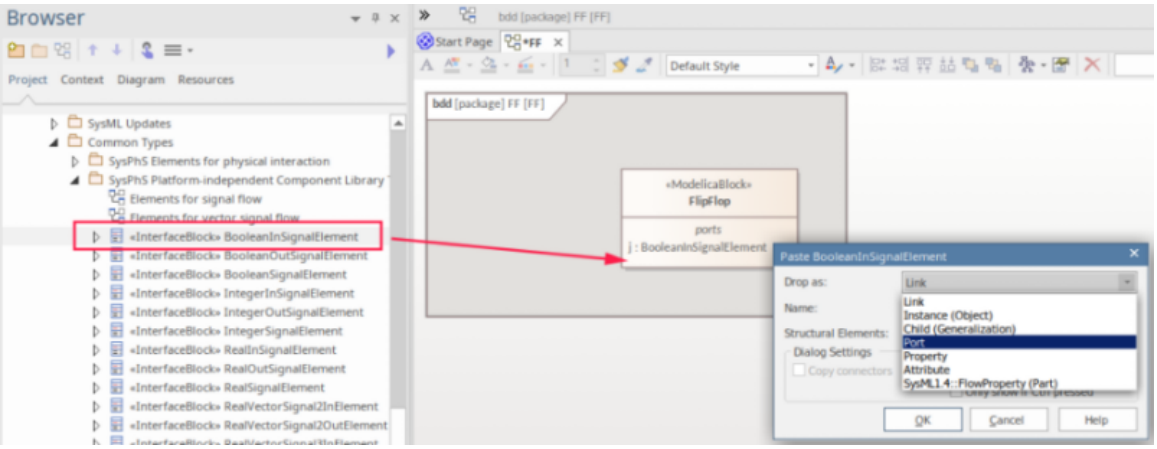
Les ports d'un Bloc peuvent être des ports non typés ou des types de ports prédéfinis.

Les ports non typés sont créés en faisant glisser un port Modelica de la boîte à outils sur un diagramme .

Pour référencer les types de ports prédéfinis SysPhS, tels qu'un *port d'entrée de signal booléen* ou un *port de sortie de signal analogique* :

- Accéder aux types de ports dans la *Bibliothèque de composants indépendants de la plate-forme SysPhS* dans la fenêtre Navigateur
- Faites glisser un type de port de la Bibliothèque vers un Bloc sur un diagramme
- Définissez-le comme port sur le Bloc

Par exemple, cette illustration montre les ports « j » et « k » en cours de création sur la bascule, avec le port « j » défini et le port « k » en cours de définition en tant que port.



Créer des blocs spécifiques à Simulink et Simscape

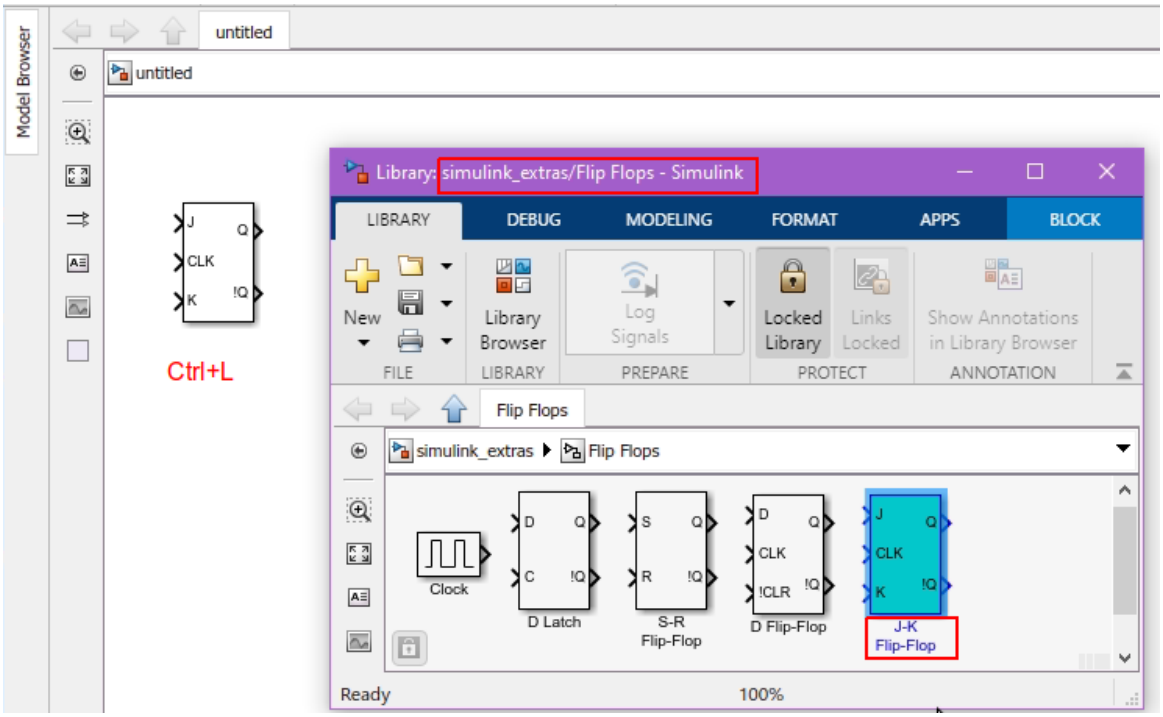
Étant donné la large gamme de différents types de composants pouvant être utilisés dans Simulink, il y aura des cas où vous devrez modéliser des composants Simulink et Simscape que vous ne pourrez pas dériver des blocs de base fournis dans les motifs de composants SysPhS. Dans de tels cas, vous pouvez définir une référence dans un Bloc SysPhS à ce type de composant Simulink. Cela peut être pour un Bloc , un composant ou une interface vers un composant.

Utilisation du chemin de classe Simulink pour un Bloc

Le processus pour référencer un composant Simulink dans un composant Enterprise Architect SysPhS est :

- Dans un diagramme Simulink contenant le composant, cliquez sur le composant et appuyez sur Ctrl+L pour accéder à ce type de composant dans la Bibliothèque
- Dans la fenêtre Bibliothèque , note le chemin dans le titre de la fenêtre et le nom du composant sous l'élément Composant dans le corps de la fenêtre
- Dans Enterprise Architect , saisissez le chemin et le nom du composant dans la fenêtre Propriétés , dans le champ « Nom » sous **SimulinkBlock (à partir de SysPhS) :**

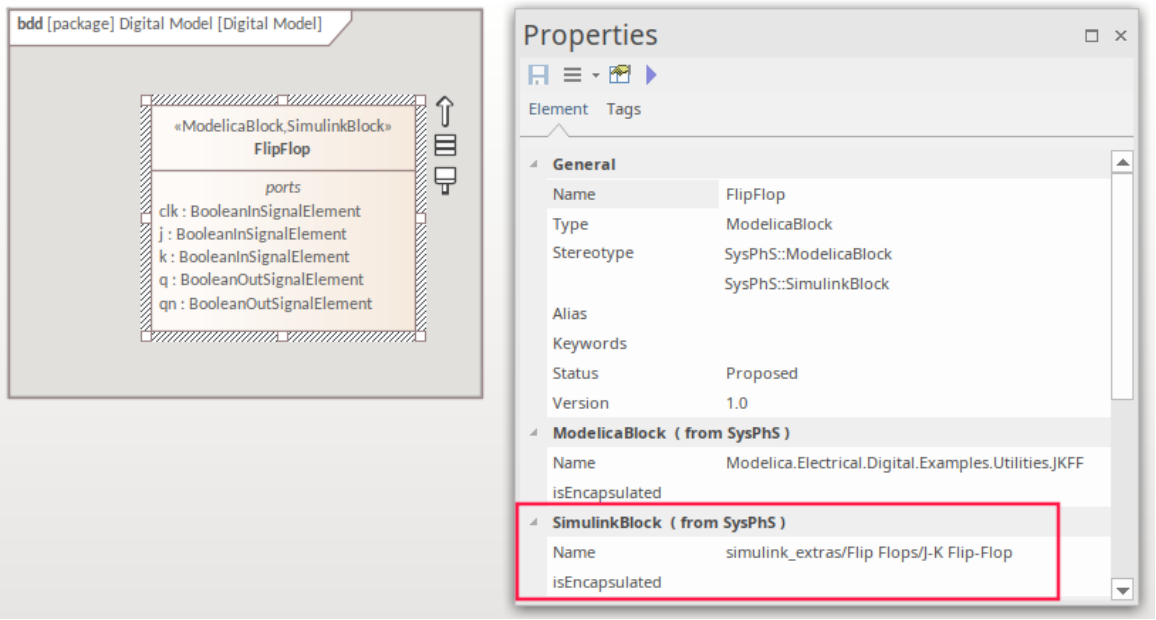
Voici un exemple de bascule Simulink JK et de la dialogue « Paramètres », montrant le chemin de classe.



Dans ce cas, le chemin/nom est :

- simulink_extras/Tongs/Tong JK

Cette illustration montre le texte ajouté à la fenêtre Propriétés du SysPhS SimulinkBlock, dans le champ « Nom ».



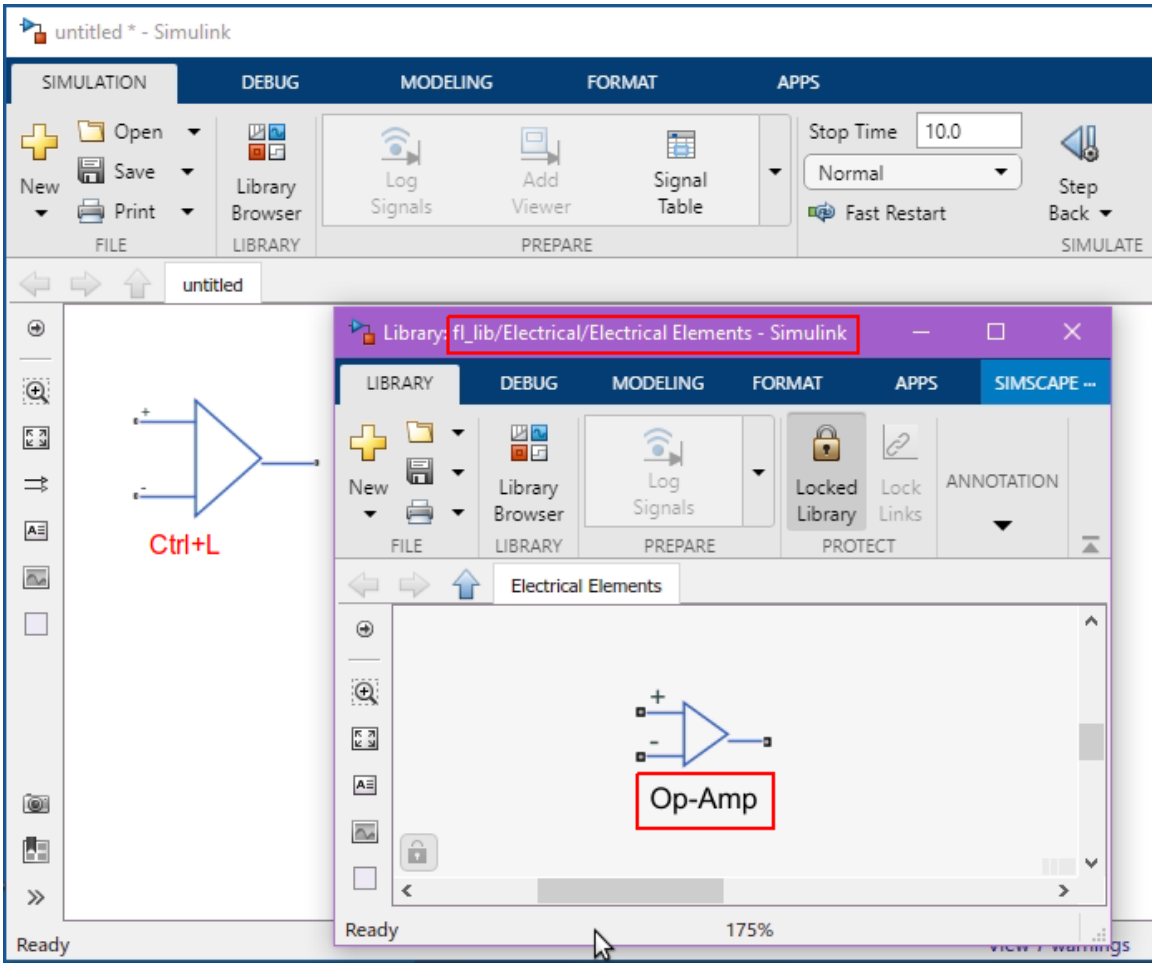
Pour plus de détails sur la définition du chemin de classe Simulink, consultez la rubrique d'aide *« Déclaration des composants membres »* de MathWorks.

Utilisation du chemin de classe Simscape pour un Bloc

Le processus pour référencer un composant Simscape dans un composant Enterprise Architect SysPhS est :

- Dans un diagramme Simulink contenant le composant Simscape, cliquez sur le composant et appuyez sur Ctrl+L pour accéder à ce type de composant dans la Bibliothèque
- Dans la fenêtre Bibliothèque , note le chemin dans le titre de la fenêtre et le nom du composant sous l'élément Composant dans le corps de la fenêtre
- Dans Enterprise Architect , saisissez le chemin et le nom du composant dans la fenêtre Propriétés , dans le champ « Nom » sous **SimulinkBlock (à partir de SysPhS) :**

Voici un exemple de composant Simscape pour un amplificateur opérationnel.



Note que fl_lib est référencé par « Foundation », donc le « Nom » de SysPhS est :

- fondation.éléments.électriques.op_amp

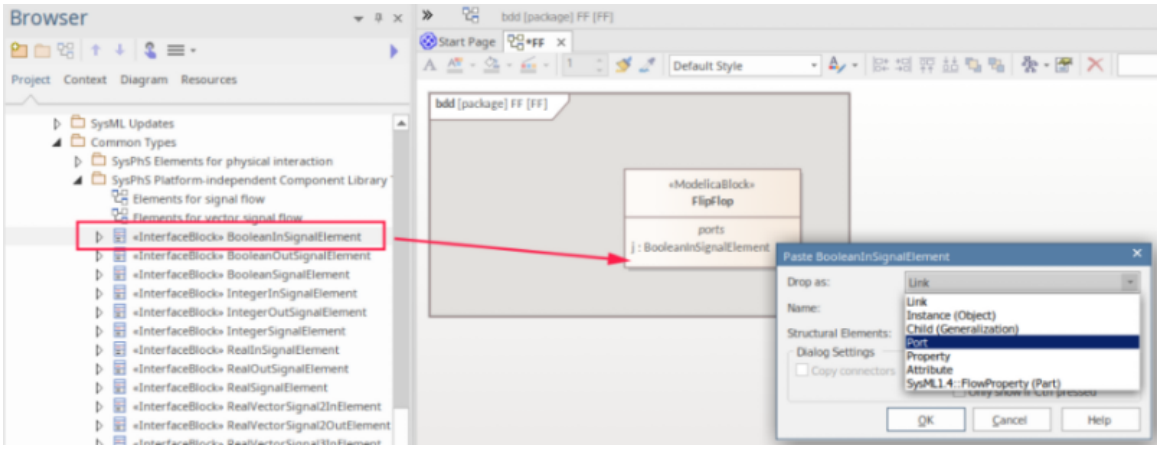
Configuration des ports SysPhS

Les ports d'un Bloc peuvent être des ports non typés ou des types de ports prédéfinis.

Les ports non définis sont créés en faisant glisser un port Simulink de la boîte à outils sur un diagramme .

Pour référencer les types de ports prédéfinis SysPhS, tels qu'un *port d'entrée de signal booléen* ou un *port de sortie de signal analogique* , ouvrez la *Bibliothèque de composants indépendants de la plate-forme SysPhS* dans la fenêtre Navigateur . Un type de port peut être glissé sur le Bloc depuis la Bibliothèque et défini comme port sur le Bloc .

Par exemple, voici un instantané du processus de création des ports j et k sur la bascule, montrant l'ensemble de ports j et le port k en cours de définition en tant que port.



Commande de port Simulink

Les ports Simulink sont définis à l'aide d'un tableau (contrairement à la référence de nom dans Modelica), l'ordre de création des ports est donc essentiel. L'ordre des ports IN est distinct de l'ordre des ports OUT. L'ordre des ports peut être visualisé dans Simulink ; le premier port est affiché en haut du Bloc .

En utilisant l'exemple des ports Flip Flop, l'ordre Simulink des ports IN serait :

j, clk, k, (voir l'élément « JK Flip Flop » dans la première image de ce sujet)

Ces ports doivent donc être créés dans Enterprise Architect dans cet ordre. Note que les ports sont affichés dans le diagramme de définition Bloc par ordre alphabétique et non par ordre de création.

Un scénario courant dans lequel l'ordre n'a pas été appliqué est celui où un port OUT est correctement affiché dans le diagramme SysML, mais est connecté de manière incorrecte à un port IN lors de la simulation dans Simulink. Si cela se produit, assurez-vous que l'ordre de création des ports est correctement appliqué.

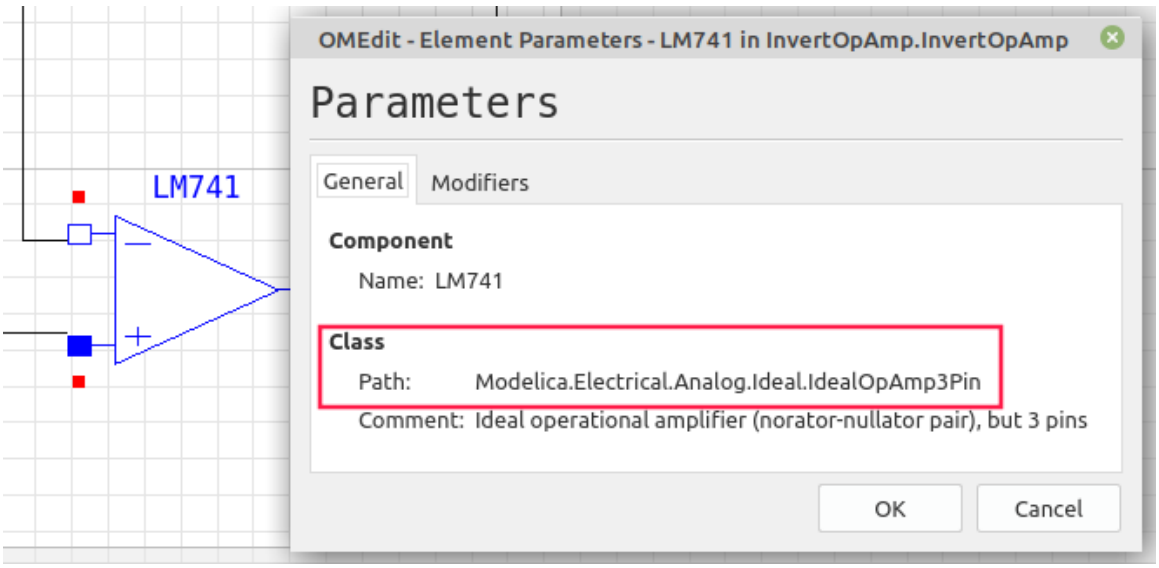
Configuration des blocs à la fois comme Modelica et Simulink

Pour les cas où Modelica et Matlab sont tous deux utilisés, ou pour définir gabarits génériques pour couvrir les deux produits, vous pouvez définir les références au type de composant des deux produits dans un Bloc SysPhS.

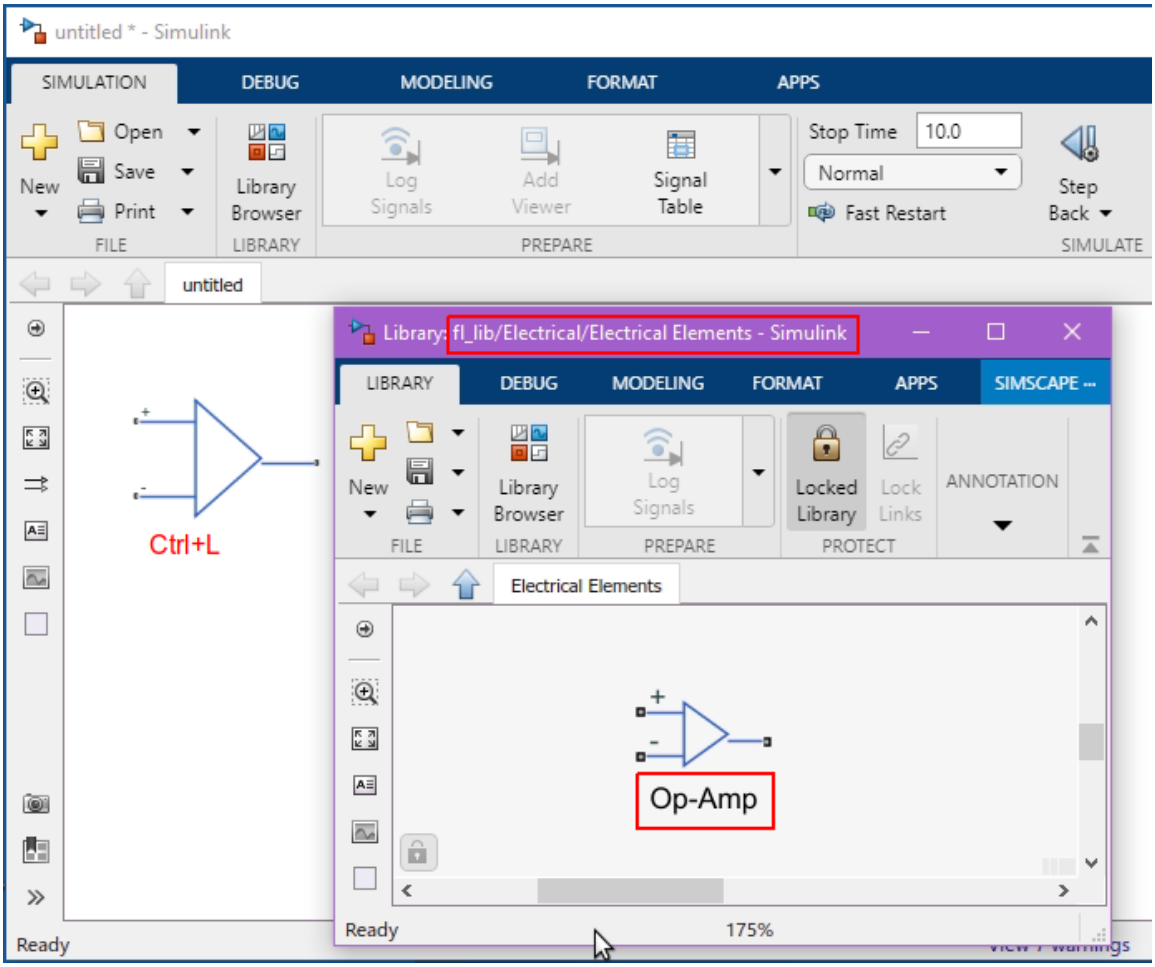
Les blocs prédéfinis fournis avec les Motifs SysPhS incluent déjà les propriétés Modelica et Simulink.

Exemple

Voici un exemple de composant Modelica pour un ampli-op.

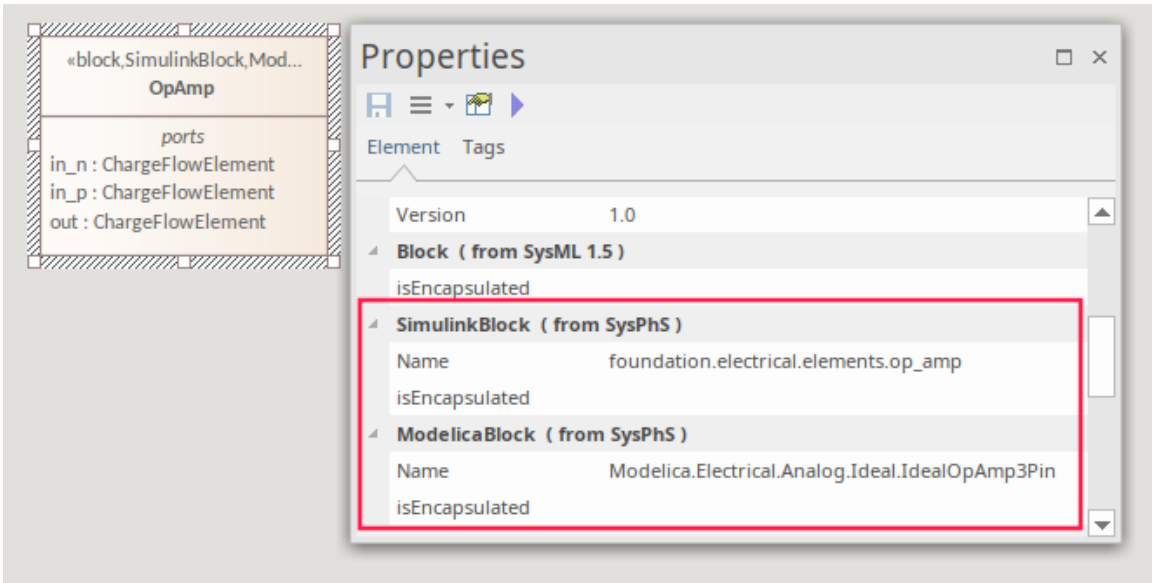


Voici un exemple de composant Simscape pour un amplificateur opérationnel.



Note que comme fl_lib est référencé par « Foundation », le « Nom » de SysPhS est *foundation.electrical.elements.op_amp.*

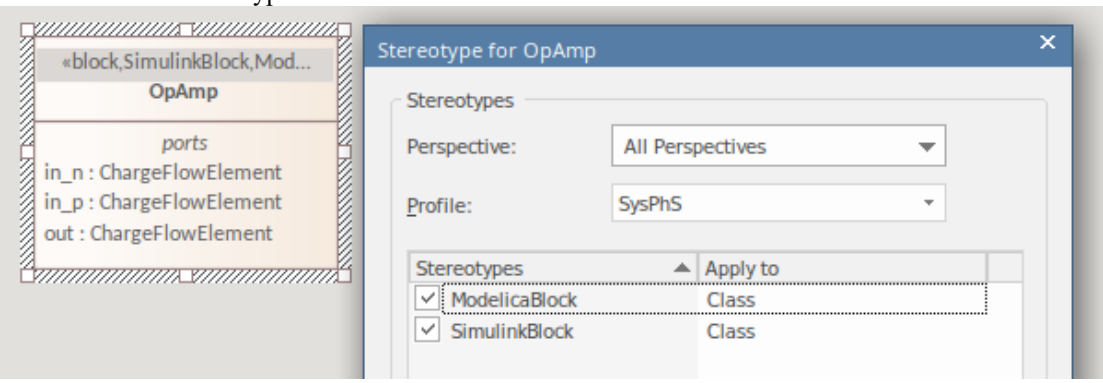
Voici les paramètres « Nom » pour Modelica et Simulink.



Pour définir manuellement les propriétés des deux produits, vous devez définir le stéréotype pour les deux. Pour effectuer cette opération sur un élément SysPhs Bloc nouvellement créé, dans la fenêtre Propriétés de cet élément :

- Sélectionnez « Stéréotype »
- Cliquez sur le bouton [...]

- Sélectionnez SysPhS dans le profil
- Cochez les deux stéréotypes SimulinkBlock et ModelicaBlock



Simulation

Enterprise Architect fournit une intégration avec OpenModelica et Simulink de MATLAB, pour support une évaluation rapide et robuste du comportement d'un modèle SysML dans différentes circonstances.

Les bibliothèques OpenModelica sont des ressources complètes qui fournissent de nombreux types, fonctions et modèles utiles. Lors de la création de modèles SysML dans Enterprise Architect , vous pouvez référencer les ressources disponibles dans ces bibliothèques.

L'intégration MATLAB d' Enterprise Architect se connecte via l'API MATLAB, permettant à vos simulations Enterprise Architect et autres scripts d'agir en fonction de la valeur de toutes les fonctions et expressions MATLAB disponibles. Vous pouvez exporter votre modèle vers MATLAB Simulink, Simscape et/ou Stateflow.

Cette section décrit le processus de définition d'un modèle Diagramme Interne de Bloc ou Paramétriques , l'annotation du modèle avec des informations supplémentaires pour piloter une simulation et l'exécution d'une simulation pour générer un graphique des résultats.

Introduction aux modèles Bloc internes et Paramétriques SysML

Les modèles SysPhS support l'analyse technique des paramètres critiques du système, notamment l'évaluation des indicateurs clés tels que les performances, la fiabilité et d'autres caractéristiques physiques. Ces modèles combinent des modèles d'exigences avec des modèles de conception de système, en capturant des contraintes exécutables basées sur des relations mathématiques complexes. diagrammes Paramétriques sont diagrammes Bloc internes spécialisés qui vous aident, en tant que modélisateur, à combiner des modèles de comportement et de structure avec des modèles d'analyse technique tels que les modèles de performances, de fiabilité et de propriétés de masse.

Pour plus d'informations sur les concepts des modèles SysML Paramétriques , reportez-vous au site officiel OMG SysML et à ses sources liées.

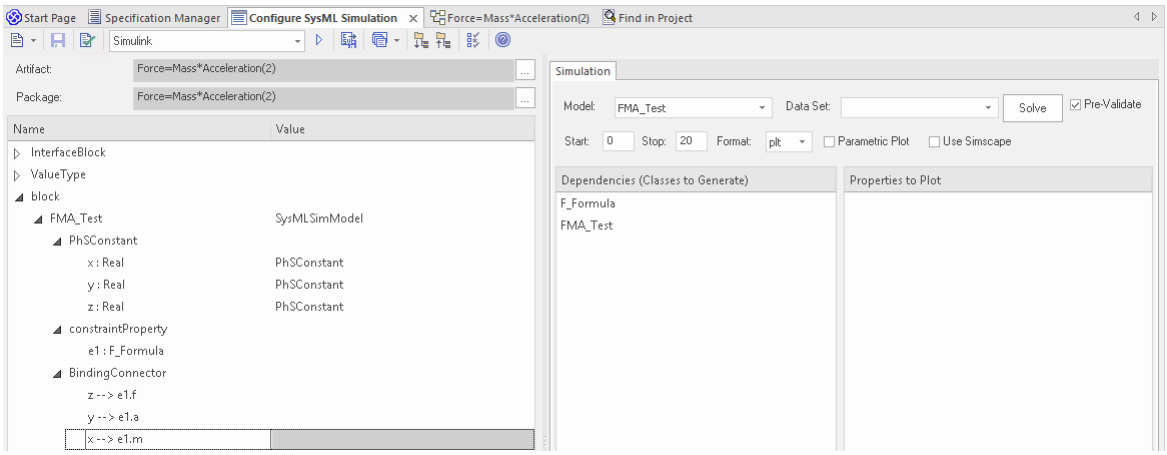
Ces rubriques décrivent les principales options permettant de définir et d'exécuter des simulations Ingénierie des Systèmes à l'aide de SysML et SysPhS.

Sujet	Description
Support de la norme SysPhS	La *norme SysPhS* est une *extension SysML pour Simulation d'interaction physique et de flux de signaux* . Elle définit une méthode standard de traduction entre un modèle SysML et un modèle Modelica ou un modèle Simulink/Simscape, offrant ainsi une méthode plus simple basée sur un modèle pour le partage de simulations.
Artefact de configuration SysMLSim	Enterprise Architect vous aide à étendre l'utilité de vos modèles SysML Paramétriques en les annotant avec des informations supplémentaires qui permettent de simuler le modèle. Le modèle résultant est ensuite généré sous forme de modèle pouvant être résolu (simulé) à l'aide de MATLAB Simulink ou d'OpenModelica. Les propriétés de simulation de votre modèle sont stockées dans un artefact Simulation . Cela préserve votre modèle d'origine et supporte plusieurs simulations configurées par rapport à un seul modèle SysML. L'artefact Simulation se trouve sur la page de la boîte à outils « Artefacts ».
Interface Utilisateur	L'interface utilisateur de la simulation SysML est décrite dans la rubrique *Configurer Simulation SysML* .
Affichage du Modèle généré	Une fois qu'un modèle a été généré sur Modelica ou Simulink, vous pouvez travailler directement avec ce modèle dans l'application externe.
Conseils de débogage SysPhS	Comme de nombreux facteurs peuvent être à l’origine d’une erreur dans un script généré, nous fournissons quelques conseils pour isoler la source d’un problème dans le modèle.

Analyse Modèle à l'aide d'ensembles de données	Lors de l'utilisation d'une simulation, il peut arriver que plusieurs variantes doivent être testées. support cela, plusieurs jeux de données peuvent être définis par rapport aux blocs, ce qui permet des variations de simulation reproductibles à l'aide du même modèle SysML.
Exemples Simulation SysPhS	Pour vous aider à comprendre comment créer et simuler un modèle SysPhS, trois exemples ont été fournis pour illustrer trois domaines différents. Ces trois exemples utilisent les bibliothèques OpenModelica. Ces exemples et ce que vous pouvez en apprendre sont décrits dans la rubrique *Exemples Simulation SysPhS* .

Configurer Simulation SysML

La fenêtre Configurer Simulation SysML est l'interface par laquelle vous pouvez fournir des paramètres d'exécution pour exécuter la simulation d'un modèle SysML. La simulation est basée sur une configuration de simulation définie dans un élément d'artefact SysMLSimConfiguration.

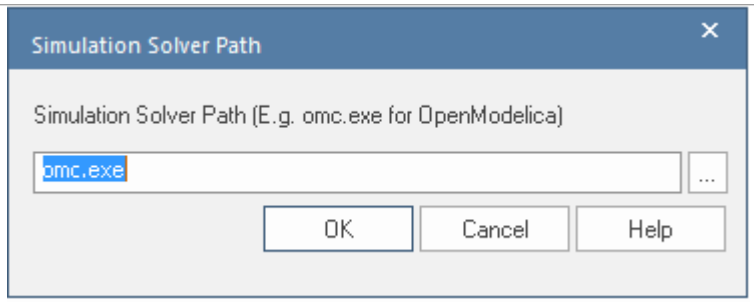


Accéder

Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager
Autre	Double-cliquez sur un artefact avec le stéréotype SysMLSimConfiguration.

Options de la barre d'outils

Option	Description
	Cliquez sur la flèche déroulante et sélectionnez l'une de ces options : • Sélectionner un artefact — Sélectionnez et chargez une configuration existante à partir d'un artefact avec le stéréotype SysMLSimConfiguration (si aucun n'a déjà été sélectionné) • Créer un artefact — Créez une nouvelle configuration SysMLSim ou sélectionnez et chargez un artefact de configuration existant • Sélectionner Paquetage — Sélectionnez un Paquetage pour rechercher les éléments SysML à configurer pour la simulation • Recharger — Recharger le gestionnaire de configuration avec les modifications apportées au Paquetage actuel • Configurer Solveur Simulation — Affichez la dialogue « Chemin Solveur Simulation », dans laquelle vous pouvez saisir ou rechercher le chemin d'accès au Solveur à utiliser. Pour MATLAB/Simulink, le chemin sera automatiquement détecté. Il ne doit donc être modifié qu'en cas de problème avec la détection automatique ou si plusieurs versions de MATLAB sont installées.

	
	Cliquez sur ce bouton pour enregistrer la configuration de l'artefact actuel.
	Cliquez sur cette icône pour valider spécifiquement le modèle par rapport à la configuration SysML maintenant . Les résultats de la validation s'affichent dans l'onglet « Simulation SysML » de la fenêtre Sortie système. Vous pouvez également sélectionner une option pour pré-valider automatiquement le modèle avant l'exécution de chaque simulation. Voir l'option « Pré-valider » dans le tableau *de l'onglet Simulation* .
	Cliquez sur cette icône pour développer chaque élément de la hiérarchie dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour réduire tous les éléments développés dans la hiérarchie du modèle dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour afficher une liste des types object qui peuvent être supprimés dans la simulation. Cliquez sur la case à cocher en regard de chaque object à supprimer ou cliquez sur le bouton Tous pour sélectionner tous les éléments à supprimer. Vous pouvez également utiliser la Barre de Filtre en haut de la colonne « Option » pour afficher uniquement les éléments ayant la lettre ou string de texte spécifiée dans le nom.
	Cliquez sur la flèche déroulante et sélectionnez l'application sous laquelle la simulation est exécuter - comme OpenModelica ou Simulink.
	Cliquez sur ce bouton pour générer, compiler et exécuter la configuration actuelle, et afficher les résultats.
	Après la simulation, le fichier de résultats est généré au format plt, mat ou csv. C'est-à-dire avec le nom de fichier : • ModelName_res.mat (la valeur par défaut pour OpenModelica) • ModelName_res.plt ou • Nom du modèle_res.csv Cliquez sur ce bouton pour spécifier un répertoire dans lequel Enterprise Architect copiera le fichier de résultats.
	Cliquez sur ce bouton pour sélectionner parmi ces options : • Exécuter le dernier code - Exécuter le code le plus récemment généré • Générer du code — Générer le code sans le compiler ni l'exécuter • Ouvrir le répertoire Simulation — Ouvrir le répertoire dans lequel le code OpenModelica ou Simulink sera généré • Modifier Gabarits — Personnalisez le code généré pour OpenModelica ou

	Simulink, à l'aide de l'éditeur de code Gabarit

Artefact Simulation et sélection Modèle

Champ	Action
Artefact	Cliquez sur l'icône  et recherchez et sélectionnez un artefact SysMLSimConfiguration existant ou créez un nouvel artefact.
Paquetage	Si vous avez spécifié un artefact SysMLSimConfiguration existant, ce champ correspond par défaut au Paquetage contenant le modèle SysML associé à cet artefact. Sinon, cliquez sur l'icône  et recherchez et sélectionnez le Paquetage contenant le modèle SysML à configurer pour la simulation. Vous devez spécifier (ou créer) l'Artefact avant de sélectionner le Paquetage .

Objets Paquetage

Ce tableau décrit les types d' object du modèle SysML qui seront répertoriés sous la colonne « Nom » de la fenêtre Configurer Simulation SysML, pour être traités dans la simulation. Chaque type object se développe pour répertorier les objets nommés de ce type et les propriétés de chaque object qui nécessitent une configuration dans la colonne « Valeur ».

De nombreux niveaux de types object , de noms et de propriétés ne nécessitent pas de configuration, de sorte que le champ « Valeur » correspondant n'accepte aucune saisie. Lorsque la saisie est appropriée et acceptée, une flèche déroulante s'affiche à l'extrémité droite du champ ; lorsque vous cliquez sur cette flèche, une courte liste de valeurs possibles s'affiche pour la sélection. Certaines valeurs (telles que « SimVariable » pour une pièce) ajoutent des couches supplémentaires de paramètres et de propriétés, où vous cliquez sur le bouton  pour, à nouveau, sélectionner et définir des valeurs pour les paramètres. Pour les jeux de données, la dialogue de saisie vous permet de saisir ou d'importer des valeurs, telles que des valeurs initiales ou par défaut ; consultez la rubrique d'aide *Analyse de Modèle utilisant Ensemble de Données* .

Type d'élément	Comportement
Type de valeur	Les éléments ValueType sont généralisés à partir d'un type primitif ou sont substitués par SysMLSimReal pour la simulation.
Bloc	Les éléments Bloc mappés aux éléments SysMLSimClass ou SysMLSimModel support la création d'ensembles de données. Si vous avez défini plusieurs ensembles de données dans une SysMLSimClass (qui peuvent être généralisés), vous devez identifier l'un d'entre eux comme étant l'ensemble de données par défaut (à l'aide de l'option de menu contextuel « Définir comme ensemble de données par défaut »). Comme un SysMLSimModel est un élément de niveau supérieur possible pour une simulation et ne sera pas généralisé, si vous avez défini plusieurs ensembles de données, l'ensemble de données à utiliser est choisi pendant la simulation.
Propriétés	La méthode préférée pour spécifier des constantes ou des variables et leurs paramètres consiste à utiliser les stéréotypes SysPhS PhSConstant et PhSVariable sur les Propriétés elles-mêmes. Le stéréotype PhSVariable possède des propriétés

	intégrées pour *isContinuous* , *isConserved* et *changeCycle* . Les Propriétés seront répertoriées sous PhSConstant ou PhSVariable et la valeur ne peut pas être modifiée. Il est également possible de définir les paramètres dans la fenêtre Configurer Simulation SysML. Dans ce cas, ils seront répertoriés sous « Propriétés ». Propriétés d'un Bloc peuvent être configurées comme des SimConstants ou des SimVariables. Pour une SimVariable, vous configurez ces attributs : • isContinuous — détermine si la valeur de la propriété varie en continu (« true », la valeur par défaut) ou discrètement (« false ») • isConserved — détermine si les valeurs de la propriété sont conservées ('true') ou non ('false', la valeur par défaut) ; lors de modélisation d'une interaction physique, les interactions incluent des échanges de substances physiques conservées telles que le courant électrique, la force ou l'écoulement d'un fluide • changeCycle — spécifie l'intervalle de temps auquel une valeur de propriété discrète change ; la valeur par défaut est « 0 » - changeCycle peut être défini sur une valeur autre que 0 uniquement lorsque isContinuous = 'faux' - La valeur de changeCycle doit être positive ou égale à 0
Port	Aucune configuration requise.
Fonction Sim	Les fonctions sont créées sous forme d'opérations dans des blocs ou des blocs de contraintes, stéréotypés comme « SimFunction ». Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML.
Généralisation	Aucune configuration requise.
Connecteur de liaison	Lie une propriété à un paramètre d'une propriété de contrainte. Aucune configuration n'est requise ; cependant, si les propriétés sont différentes, le système propose une option pour les synchroniser.
Connecteur	Connecte deux ports. Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML. Cependant, vous devrez peut-être configurer les propriétés du type de port en déterminant si l'attribut isConserved doit être défini sur « False » (pour les propriétés potentielles, afin que le couplage d'égalité soit établi) ou sur « True » (pour les propriétés de flux/conservées, afin que le couplage somme à zéro soit établi).
Bloc de contrainte	Aucune configuration requise.

Onglet Simulation

Ce tableau décrit les champs de l'onglet « Simulation » de la fenêtre Configurer Simulation SysML.

Champ	Action
Modèle	Cliquez sur la flèche déroulante et sélectionnez le nœud de niveau supérieur (un élément SysMLSimModel) pour la simulation. La liste est renseignée avec les noms des blocs définis comme nœuds de modèle de niveau supérieur.

Ensemble de données	Cliquez sur la flèche déroulante et sélectionnez l'ensemble de données pour le modèle sélectionné.
Pré-valider	Cochez cette case pour valider automatiquement le modèle avant l'exécution de chaque simulation du modèle.
Démarrer	Type le temps d'attente initial avant lequel la simulation est démarrée, en secondes (valeur par défaut est 0).
Arrêt	Type le nombre de secondes pendant lesquelles la simulation s'exécutera.
Format	Cliquez sur la flèche déroulante et sélectionnez « plt », « csv » ou « mat » comme format du fichier de résultat, qui pourrait potentiellement être utilisé par d'autres outils.
Graphique Paramétriques	• Cochez cette case pour tracer la légende A sur l'axe des Y par rapport à la légende B sur l'axe des X. • Décochez la case pour tracer les légendes sur l'axe des Y en fonction du temps sur l'axe des X Note : avec la case à cocher sélectionnée, vous devez sélectionner deux propriétés à tracer.
Utiliser Simscape	(si l'outil mathématique sélectionné est Simulink) Cochez la case si vous souhaitez également traiter la simulation dans Simscape.
Dépendances	Répertorie les types qui doivent être générés pour simuler ce modèle.
Propriétés à construire	Fournit une liste des propriétés des variables impliquées dans la simulation. Cochez la case en regard de chaque propriété à tracer.

Affichage du Modèle généré

Une fois qu'un modèle a été généré vers Modelica ou Simulink, il peut arriver que vous souhaitiez travailler directement avec ce modèle dans l'application externe. Cela inclut tous les cas où le modèle n'est pas généré correctement et où vous devez trouver le problème dans l'application externe.

Exécuter une génération de modèle en utilisant le bouton ou Solve puis visualisez le code généré.

Affichage du code généré

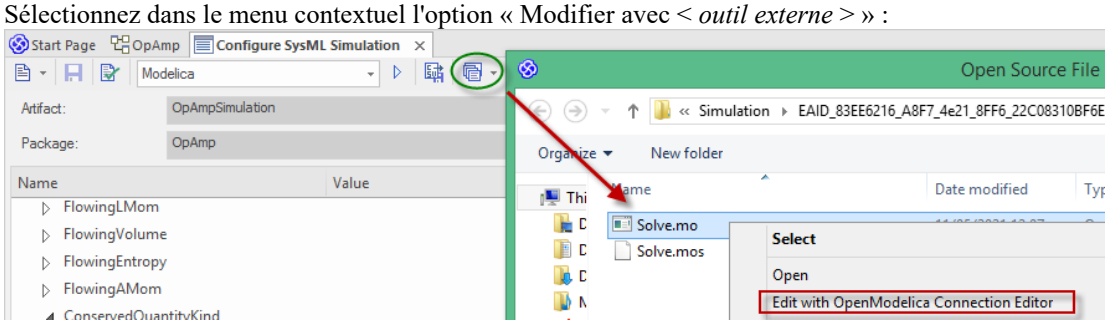
Pour afficher le code Modelica ou Simulink généré dans l'éditeur de code Enterprise Architect :

- Cliquez sur l'icône dans la barre d'outils
- Sélectionnez l'option « Ouvrir le répertoire Simulation »
  Cela ouvre le répertoire dans lequel le code OpenModelica ou Simulink a été généré
- Cliquez sur le fichier pour voir le script

Accéder au Modèle généré

Pour accéder au modèle Modelica ou Simulink en utilisant le code généré :

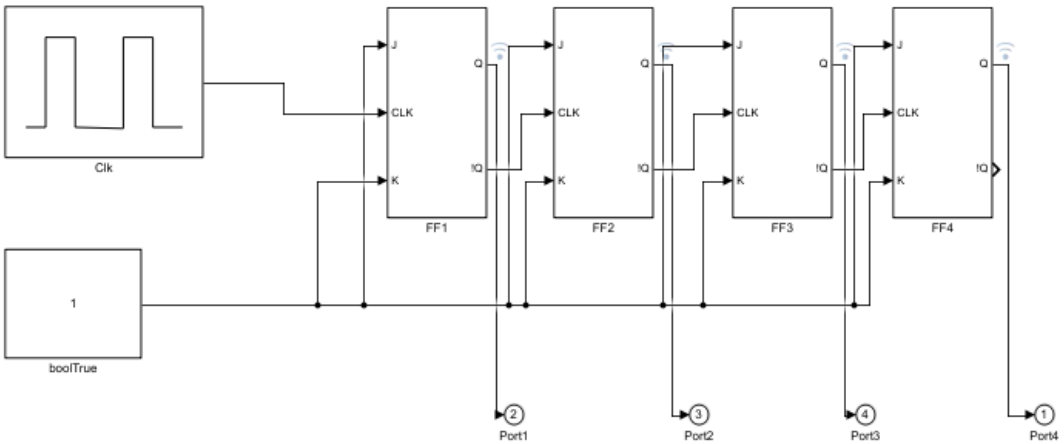
- Cliquez sur l'icône dans la barre d'outils
- Sélectionnez l'option « Ouvrir le répertoire Simulation »
  Cela ouvre le répertoire dans lequel le code OpenModelica ou Simulink a été généré
- Cliquez-droit sur le fichier
- Sélectionnez dans le menu contextuel l'option « Modifier avec < *outil externe* > » :



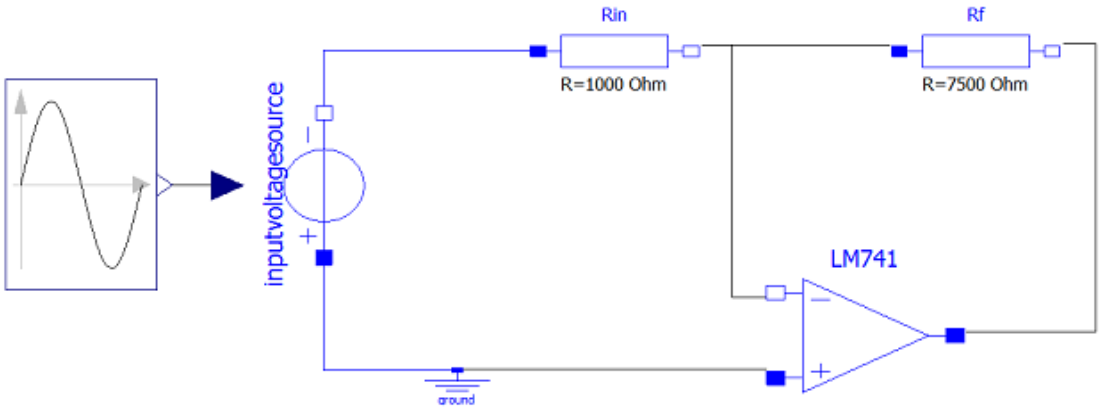
Dans ce cas, un fichier Modelica .mo est affiché. Pour un fichier généré par Simulink, le nom du fichier est Solve.m.

Double-cliquez sur le fichier pour ouvrir l'application externe :

- Pour Simulink, pour visualiser le modèle généré, vous devez ouvrir le fichier .slx nommé d'après le Bloc défini sur SysMLSimModel ; voici une vue de l'exemple Flip-flop, généré et ouvert dans Simulink :



- Pour Modelica, dans le navigateur de bibliothèques, le diagramme est nommé conformément au Bloc SysMLSimModel - double-cliquez dessus pour l'ouvrir ; voici une vue de l'exemple OpAmp, généré et ouvert dans Modelica :



Consultez la rubrique d'aide *Configurer Simulation SysML* pour plus de détails.

Conseils de débogage SysPhS

Comme pour toute génération de code à partir d'un modèle, de nombreux facteurs peuvent être à l'origine d'une erreur dans le script généré utilisé dans l'application externe. D'où la nécessité de disposer d'options permettant de trouver la cause du problème. Cette rubrique fournit quelques conseils pour isoler la source d'un problème dans le modèle.

Avant de vérifier le code généré, un script externe doit être généré à l'aide du bouton  ou Solve.

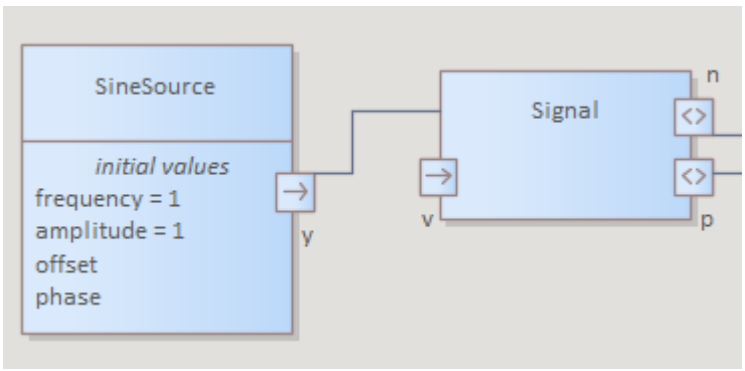
Trouver un problème

Pour trouver d'éventuels problèmes avec le script généré, les principales options à utiliser incluent :

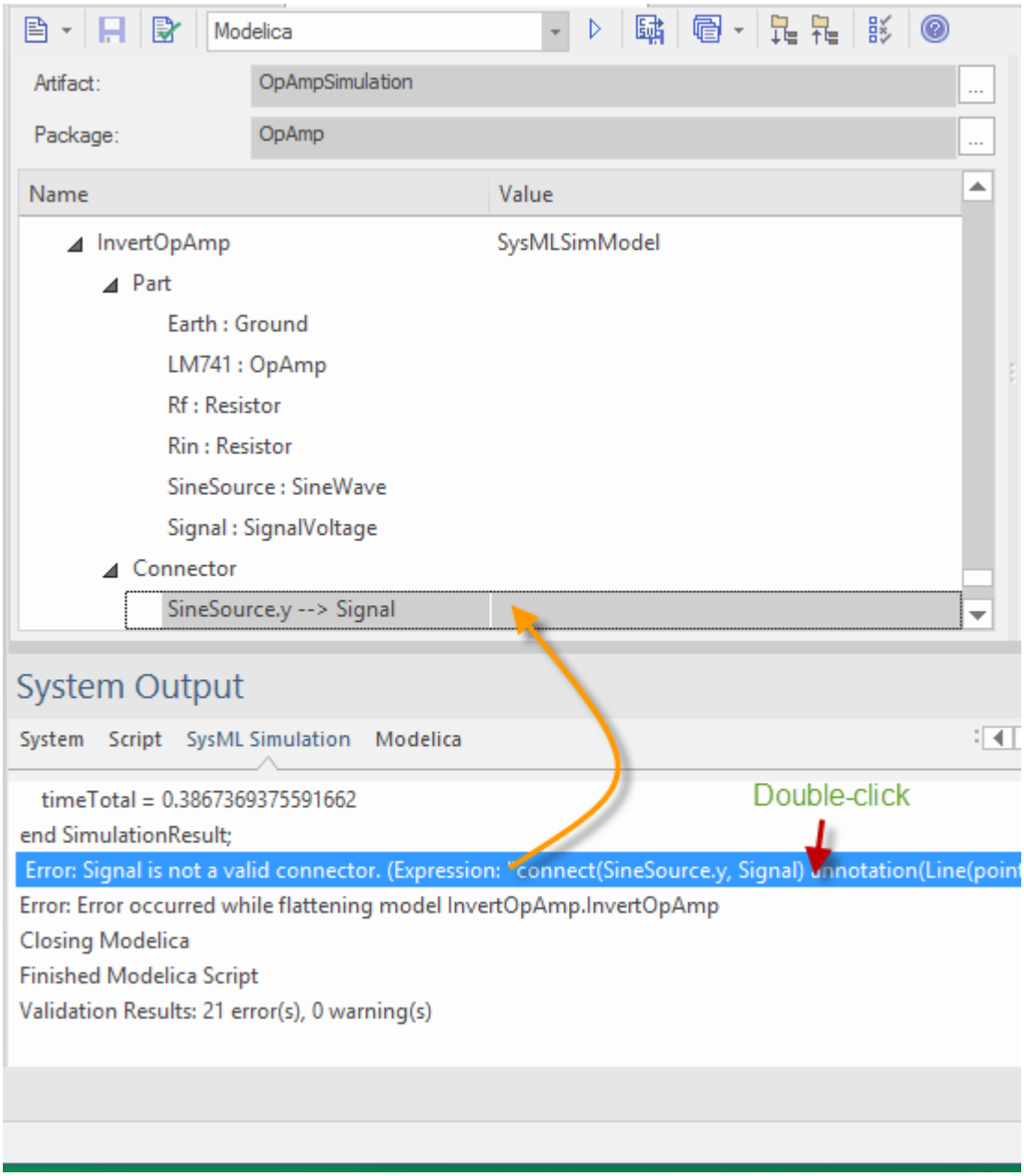
- Pour le processus de génération, une liste des étapes exécuter et des erreurs générées sont enregistrées dans la vue Sortie système (Ctrl+Maj+8)
- Pour toute erreur répertoriée dans la fenêtre Sortie système, si elle est liée à un élément spécifique, un double-clic sur cette erreur localisera l' object associé dans le volet gauche de la fenêtre Configurer Simulation
- Si l'accès au script généré est nécessaire, utilisez l'option « Ouvrir le répertoire Simulation » sous l'icône  dans la barre d'outils ; voir la rubrique d'aide *Affichage du Modèle généré*

Exemple

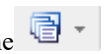
Dans cet exemple, nous avons une pièce dans l'IBD qui est incorrectement liée à une autre pièce. Il s'agit d'un lien direct et non d'un lien via un port. Cela provoque une erreur lors de la génération vers Modelica et Simulink.



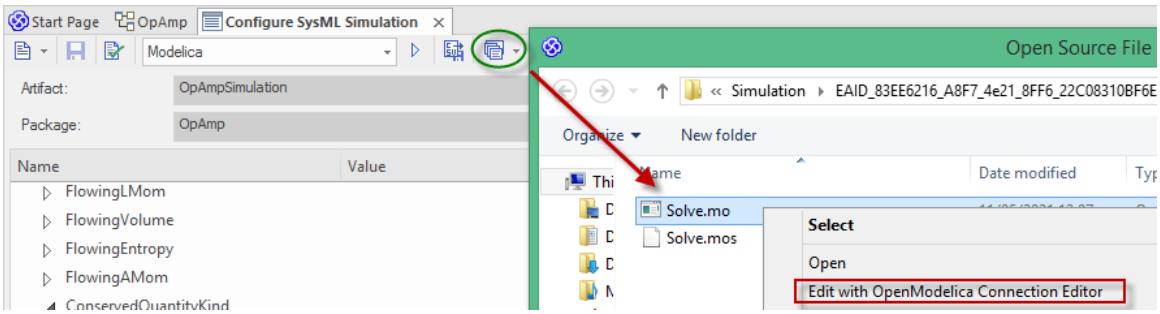
Ce qui est généré dans la sortie système est une erreur. Un double-clic sur cette erreur met en surbrillance l' object associé dans la liste Simulation .



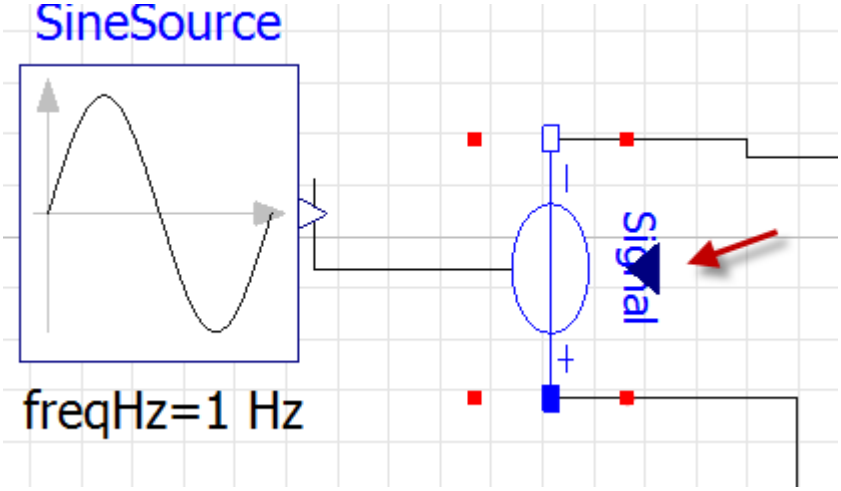
Ceci peut être vérifié plus en détail en ouvrant le script généré dans l'application externe et en visualisant le diagramme généré, en :

- Cliquez sur l'icône  dans la barre d'outils
- Sélection de l'option *Ouvrir le répertoire Simulation*
- Utiliser le menu contextuel pour ouvrir le fichier dans l'application externe

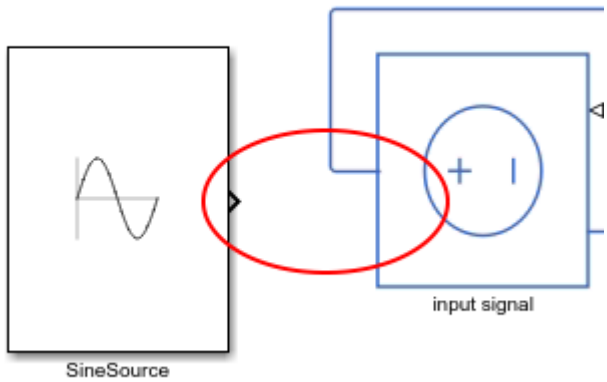
Par exemple, dans ce cas dans Modelica :



Lors de l'ouverture du modèle généré dans Modelica, en utilisant l'icône dans la barre d'outils, voici la vue montrant que le connecteur n'est pas lié au triangle qui est l'import vers l' object Signal :



De même pour Simulink, voici le modèle résultant montrant l'absence de connecteur de la sortie SineOutput :



Analyse de Modèle utilisant Ensemble de Données

Chaque Bloc SysML utilisé dans un modèle Paramétriques peut, dans la configuration Simulation , avoir plusieurs jeux de données définis par rapport à lui. Cela permet des variations de simulation reproductibles en utilisant le même modèle SysML.

Un Bloc peut être typé en tant que SysMLSimModel (un nœud de niveau supérieur qui ne peut pas être généralisé ou faire partie d'une composition) ou en tant que SysMLSimClass (un élément de niveau inférieur qui peut être généralisé ou faire partie d'une composition). Lorsque vous exécutez une simulation sur un élément SysMLSimModel, si vous avez défini plusieurs jeux de données, vous pouvez spécifier le jeu de données à utiliser. Cependant, si une SysMLSimClass dans la simulation comporte plusieurs jeux de données, vous ne pouvez pas sélectionner celui à utiliser pendant la simulation et devez donc identifier un jeu de données comme jeu de données par défaut pour cette classe.

Accéder

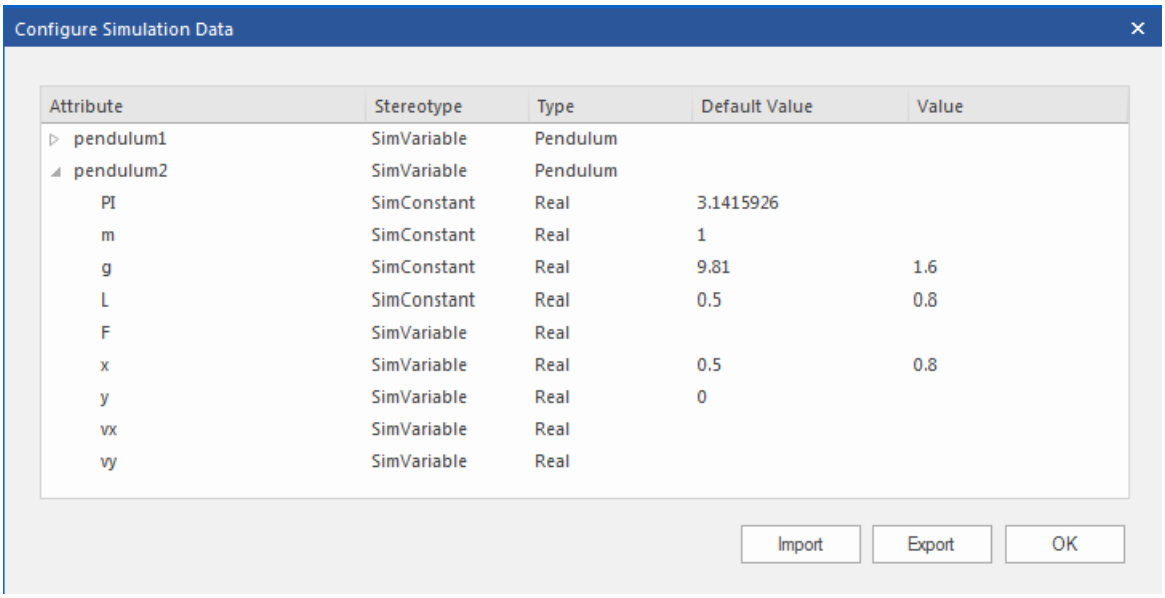
Ruban	Simuler > Comportement du Système > Modelica/Simulink > Gestionnaire de configuration SysMLSim > dans le groupe « bloc » > Colonne Nom > Menu contextuel sur l'élément bloc > Créer un jeu de données Simulation

Gestion des jeux de données

Tâche	Action
Créer	Pour créer un nouveau jeu de données, cliquez-droit sur le nom d'un Bloc et sélectionnez l'option « Créer un jeu de données Simulation ». Le jeu de données est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour configurer le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Double	Pour dupliquer un jeu de données existant comme base pour la création d'un nouveau jeu de données, cliquez-droit sur le nom du jeu de données et sélectionnez l'option « Dupliquer ». Le jeu de données dupliqué est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour modifier le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Supprimer	Pour supprimer un jeu de données qui n'est plus nécessaire, cliquez-droit sur le jeu de données et sélectionnez l'option « Supprimer le jeu de données ».
Définir par défaut	Pour définir le jeu de données par défaut utilisé par une SysMLSimClass lorsqu'elle est utilisée comme type de propriété ou héritée (et lorsqu'il existe plusieurs jeux de données), cliquez-droit sur le jeu de données et sélectionnez l'option « Définir par défaut ». Le nom du jeu de données par défaut est mis en surbrillance en gras. Les propriétés utilisées par un modèle utiliseront cette configuration par défaut, sauf si le modèle les remplace explicitement.

Configurer les données Simulation

Cette dialogue est principalement destinée à l'information. La seule colonne dans laquelle vous pouvez directement ajouter ou modifier des données est la colonne « Valeur ».



Colonne	Description
Attribut	La colonne « Attribut » fournit une arborescence de toutes les propriétés du Bloc en cours d'édition.
Stéréotype	La colonne « Stéréotype » identifie, pour chaque propriété, si elle a été configurée pour être une constante pendant toute la durée de la simulation ou une variable dont la valeur est censée changer au fil du temps.
Type	La colonne « Type » décrit le type utilisé pour la simulation de cette propriété. Il peut s'agir soit d'un type primitif (tel que « Réel »), soit d'une référence à un Bloc contenu dans le modèle. Propriétés référençant des Blocs afficheront les propriétés enfants spécifiées par le Bloc référencé en dessous d'elles.
Valeur par défaut	La colonne « Valeur par défaut » indique la valeur qui sera utilisée dans la simulation si aucune substitution n'est fournie. Cela peut provenir du champ « Valeur initiale » dans le modèle SysML ou du jeu de données par défaut du type parent.
Valeur	La colonne « Valeur » vous permet de remplacer la valeur par défaut pour chaque valeur primitive.
Exporter / Importer	Cliquez sur ces boutons pour modifier les valeurs de l'ensemble de données actuel à l'aide d'une application externe telle qu'une feuille de calcul, puis les réimporter dans la liste.

Exemples Simulation SysPhS

Cette section fournit un exemple concret pour chacune de ces étapes : création d'un modèle SysML pour un domaine, simulation de celui-ci et évaluation des résultats de la simulation. Les exemples appliquent les informations abordées dans les rubriques précédentes.

Exemples

Modèle	Description
Circuit électrique analogique OpAmp	Le premier exemple concerne la simulation d'un circuit électronique simple. L'exemple commence par un diagramme de définition Bloc de blocs définissant les composants du circuit et un Bloc contenant un diagramme de définition Bloc interne qui décrit le circuit. Le modèle est ensuite simulé et les tensions sine aux bornes source et cible de l'amplificateur opérationnel sont évaluées et comparées aux valeurs attendues.
Exemple Simulation électronique numérique	Le deuxième exemple montre un composant Flip Flop standard prédéfini dans Modelica et Simulink, pour créer un simple diviseur de fréquence de signal numérique basé sur Flip Flop. Cela suggère comment utiliser la notation SysPhS pour référencer des composants disponibles dans Modelica et Simulink, mais non définis dans les motifs SysPhS.
Exemple Simulation d'humidificateur	L'exemple décrit l'utilisation d'un diagramme Statemachine SysML pour définir l'état marche/arrêt d'un humidificateur en fonctionnement. Il est généré dans diagramme StateFlow de Modelica et MATLAB pour la simulation.

En savoir plus (vidéos et WebEA)

- SysML Simulation in Simulink
- SysML Tank Example
- MATLAB Stateflow for Statemachines
- SysPhS OpAmp example
- SysPhS Starter
- Simulating Digital Electronics using Modelica
- WebEA : EA Sim examples

Exemple Simulation de circuit d'amplificateur opérationnel

Pour cet exemple, nous parcourons la création d'un modèle SysPhS pour un circuit électronique simple, puis utilisons une simulation pour prédire et cartographier le comportement de ce circuit.

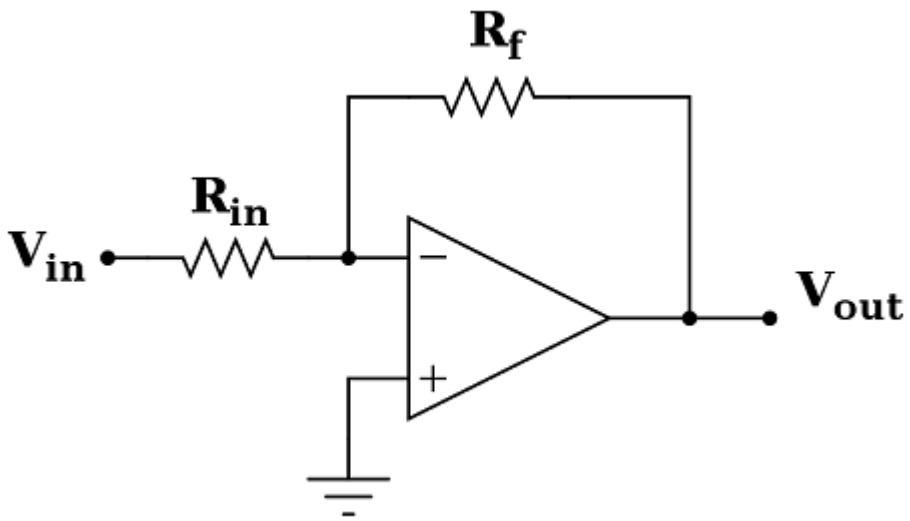
Prérequis

L'exécution de cette simulation nécessite :

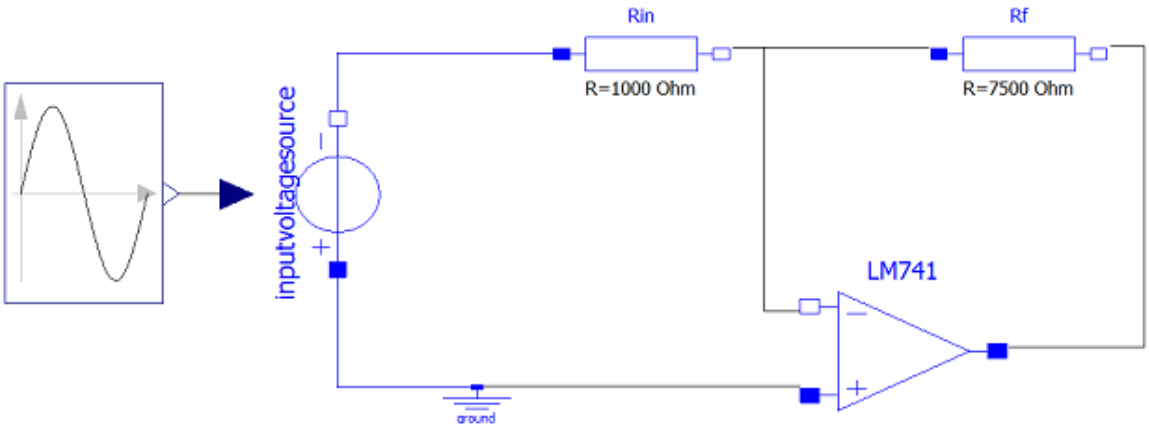
- OpenModelica ou
- Simulink et Simscape de MATLAB

Diagramme de circuit

Le circuit électronique que nous allons modéliser est illustré dans cette figure, en utilisant la notation standard des circuits électroniques.



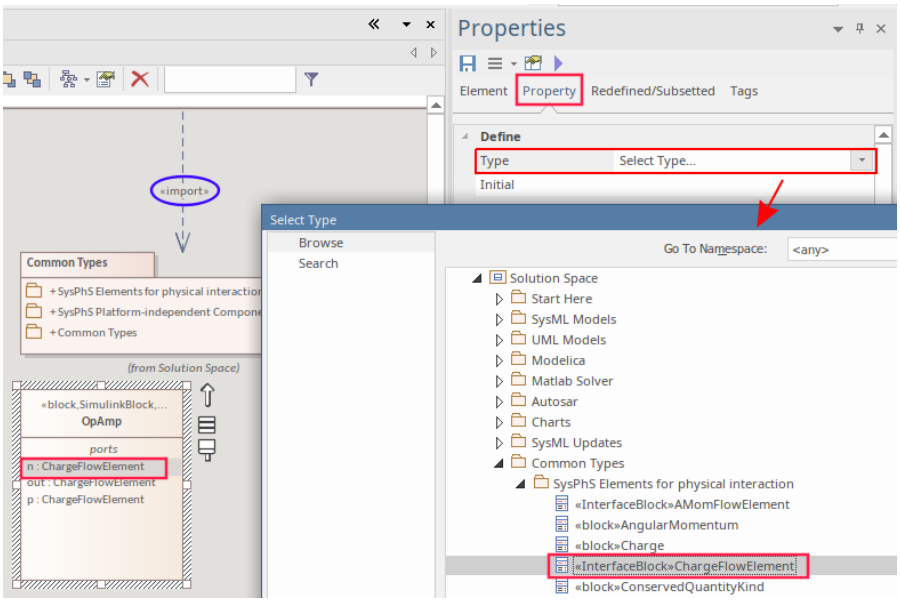
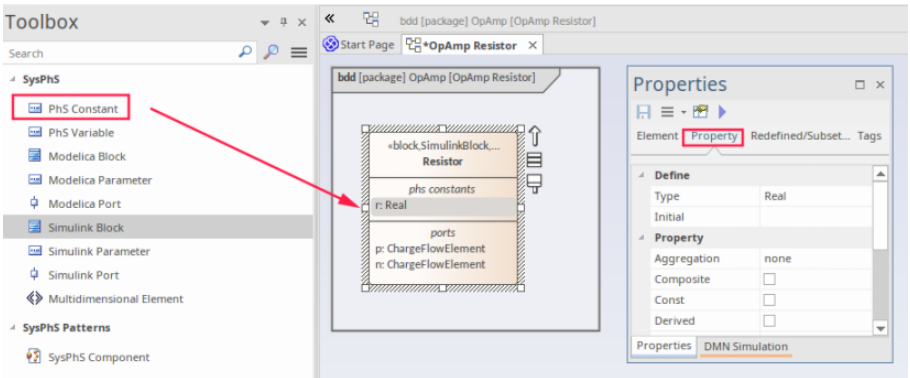
Le circuit de cet exemple comprendra une source de signal sine , une terre, deux résistances et un circuit intégré d'amplificateur OpAmp. La figure suivante montre le diagramme résultant, tel que généré à partir d'un diagramme SysML utilisant SysPhS, dans une application externe - dans ce cas Modelica.

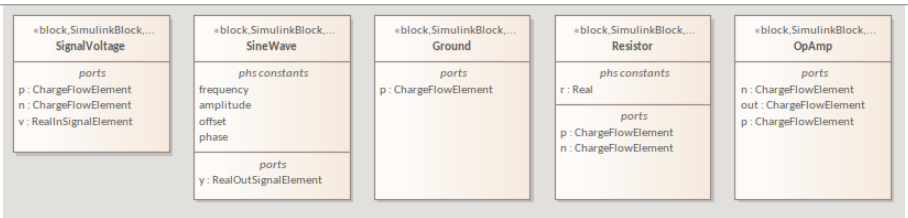


Créer Modèle SysML

Ce tableau montre comment nous pouvons créer un modèle SysML complet pour représenter le circuit, en commençant par les types de niveau le plus bas et en construisant le modèle une étape à la fois.

Composant	Action
Blocs	Dans SysML, en utilisant SysPhS, le circuit et chacun des types de composants sont représentés sous forme de blocs. Dans un diagramme de définition Bloc (BDD), créez un composant de circuit Bloc . Le circuit est composé de cinq parties : un amplificateur opérationnel, une source de signal, un convertisseur de signal en tension, une masse et une résistance. Ces parties sont de types différents, avec des comportements différents. Les blocs clés, y compris l'OpAmp, la masse et la résistance, sont accessibles sous forme de blocs prédéfinis, à l'aide des Motifs de composants SysPhS. Cependant, pour plus de clarté, nous allons exécuter leur création à partir de zéro. Créez un Bloc SysPhS pour chacun des types de composants. Les composants de la définition Bloc interne du circuit (IBD) seront connectés via des ports, qui représentent pins électriques. Ceux-ci sont définis dans le BDD. La résistance possède une broche positive et une broche négative. La terre n'a qu'une seule broche, qui est positive. L'électricité (charge électrique) est transmise via les pins . Les deux ports sont nommés « p » (positif) et « n » (négatif), et ils sont de type ChargeFlowElement. Cette figure montre le BDD, avec des blocs définissant le type de composants utilisés. Cela comprend une onde sine source de signal, un convertisseur de tension de signal, une masse, un type de résistance et un type d'amplificateur opérationnel.  Note : ces blocs sont créés à partir de la boîte à outils SysPhS à l'aide de blocs Modelica ou de blocs Simulink. Vous pouvez saisir chaque Bloc dans les deux outils. Pour plus d'informations, consultez la rubrique d'aide *Définition de blocs comme étant à la fois Modelica et Simulink* .
Ports Phs	Les types de valeur utilisés pour les ports sont prédéfinis dans les bibliothèques Simulation SysPhS. Les deux quantités clés utilisées sont les types de valeur

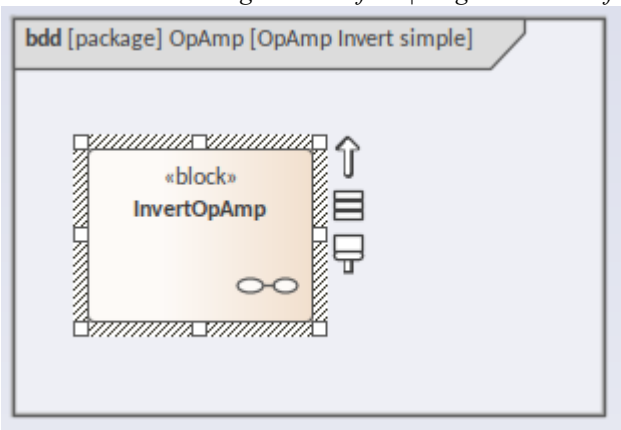
	ChargeFlowElement et RealSignalInElement, Cette figure montre le Bloc OpAmp dans le diagramme de définition Bloc , avec les ports du Bloc OpAmp définis sur le type de valeur ChargeFlowElement, et où celui-ci est référencé dans la fenêtre Navigateur .  Le connecteur d'importation vers le Paquetage des types communs montre comment ceux-ci doivent être définis. Pour plus d'informations sur l'importation Paquetage (encerclée), consultez *Référencement des bibliothèques Simulation SysPhS* Rubrique d'aide.
Constantes Phs	Les blocs Résistance et Onde Sine ont tous deux des propriétés définies dans leurs composants respectifs dans MATLAB et Modelica. Prenons l'exemple de la résistance. Pour ces composants, dans leurs outils respectifs, nous devons définir une valeur de résistance en Ohms. Pour la plupart des circuits, il peut y avoir plusieurs composants de résistance dérivés de ce Bloc de résistance, modélisés comme composants dans les diagrammes IBD ou Paramétriques . Par conséquent, leur valeur (en Ohms) sera définie dans le diagramme IBD, Paramétriques ou éventuellement dans les jeux de données de simulation. Pour définir la propriété définissant la résistance, nous devons créer une variable Phs « r ». La valeur initiale n'est pas définie.  **Note** : les Motifs SysPhS incluent un Bloc de type Résistance déjà construit. Pour définir les valeurs de résistance dans chaque composant individuel (partie), consultez la ligne *Valeurs initiales* de ce tableau , qui affiche les blocs définis avec les ports et les constantes Phs.



Structure interne

Pour la structure interne, nous créons un Bloc avec un diagramme IBD enfant.

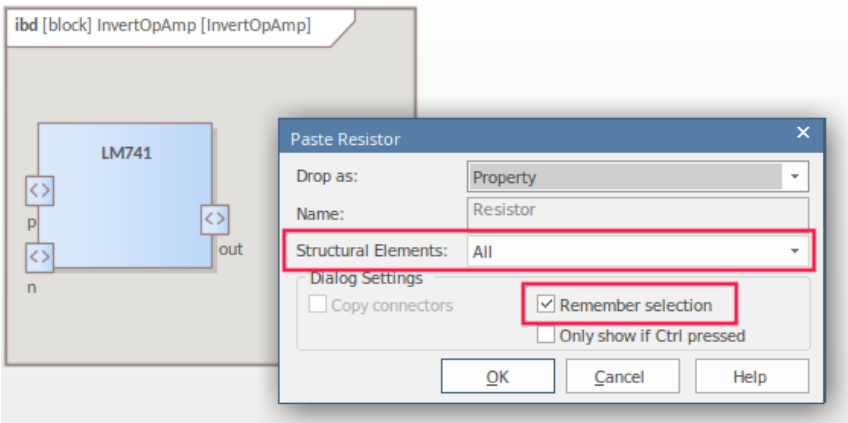
- Créer un Bloc pour un circuit OpAmp inverseur
- Créez ensuite un Diagramme Interne de Bloc (IBD) à l'aide de l'option du menu contextuel :
  Créer un nouveau Diagramme enfant | diagramme de définition Bloc interne



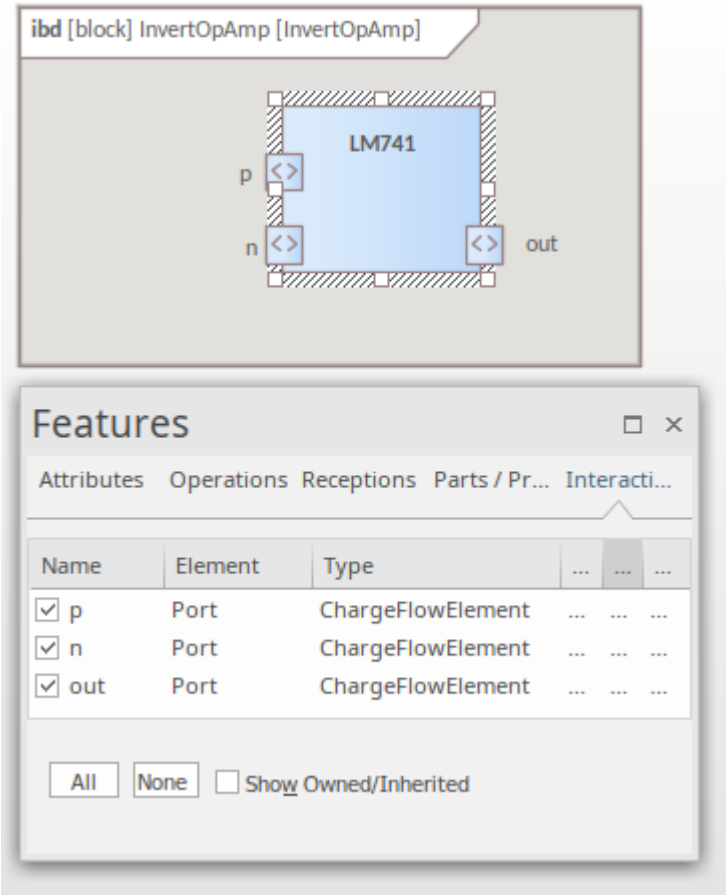
- Double-cliquez pour ouvrir l'IBD
- Définissez le diagramme pour afficher uniquement le nom du port, en utilisant :
  - F5 | Élément | Apparence de l'élément
- Désactivez ces deux options :
  - Montrer des stéréotypes
  - Afficher Type de propriété
  Cela affichera alors uniquement le port et Type

Sur l'IBD, créez des pièces et connectez-les :

- Depuis le Navigateur , faites glisser les Blocs sur l'IBD en tant que Parties (Propriétés)
- Pour visualiser les pièces dans les compartiments, supprimez-les du diagramme
  Note : définissez l'option Coller « Éléments structurels » sur « TOUS »



- Sinon, dans la vue Fonctionnalités , pour l'onglet 'Interaction' cliquez sur le bouton Tous, et pour l'onglet 'Parts / Propriétés ' cliquez sur le bouton Tous.

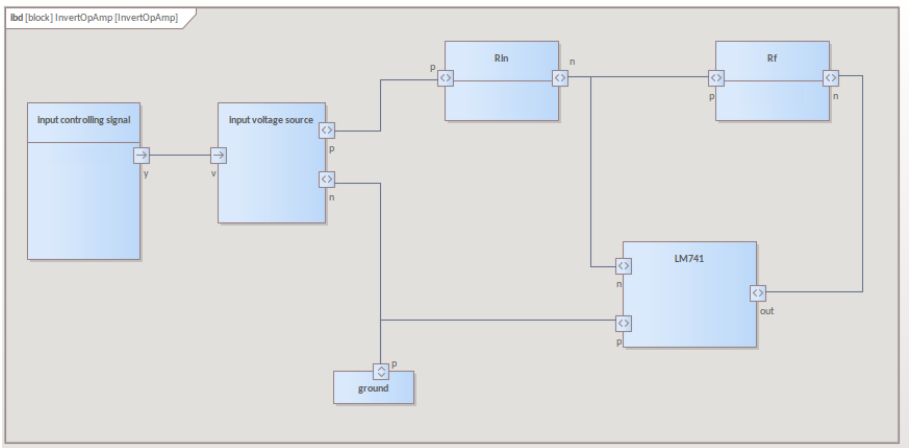


- Ajoutez des propriétés pour le signal source, le convertisseur, 2 résistances, l'amplificateur opérationnel et la terre, saisies par les blocs correspondants

Fixations

Pour modéliser le câblage de ces composants :

- Définissez les connexions entre les ports avec des connecteurs de type Connecteur.

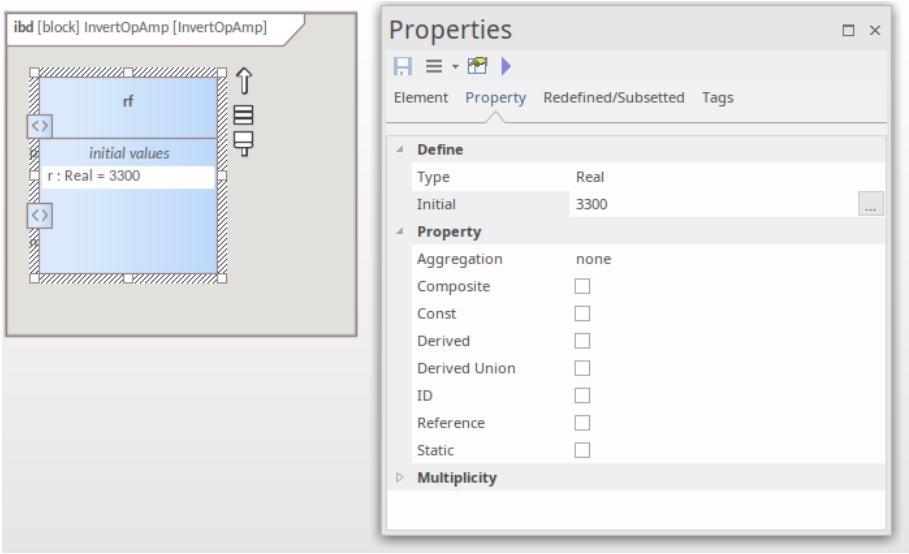


Notez que cela suit la même structure que le diagramme de circuit d'origine, mais les symboles de chaque composant ont été remplacés par des propriétés typées par les blocs que nous avons définis.

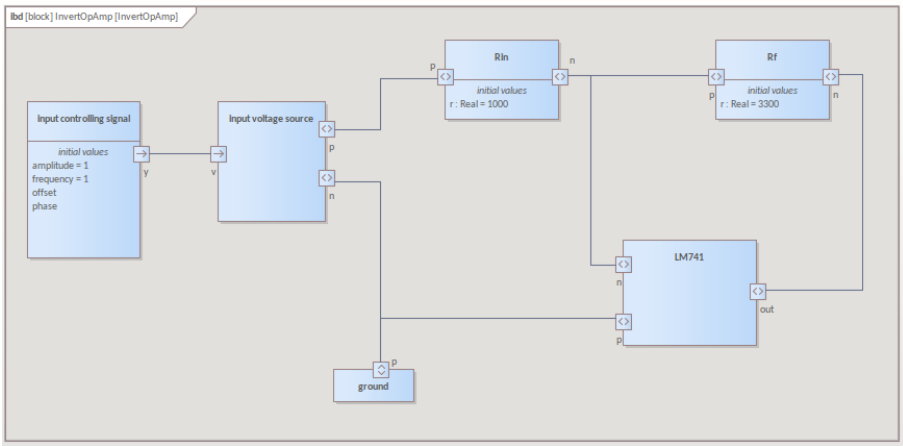
Valeurs initiales

Comme il y a deux résistances, la résistance d'entrée (RIn) et la résistance de rétroaction (Rf) doivent être créées comme deux Propriétés (pièces) différentes dans l'IBD, provenant du Bloc de résistances, car chacune d'elles aura des valeurs différentes. Les valeurs doivent être définies dans la fenêtre Propriétés , onglet «

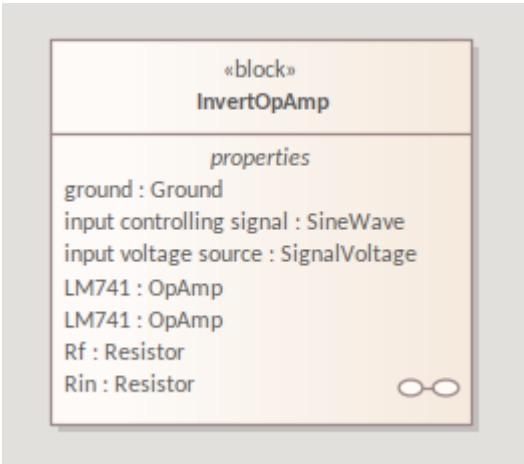
Propriété », champ « Initiale ».



Pour le InputControllingSignal, des valeurs initiales sont requises pour l'amplitude et la fréquence.

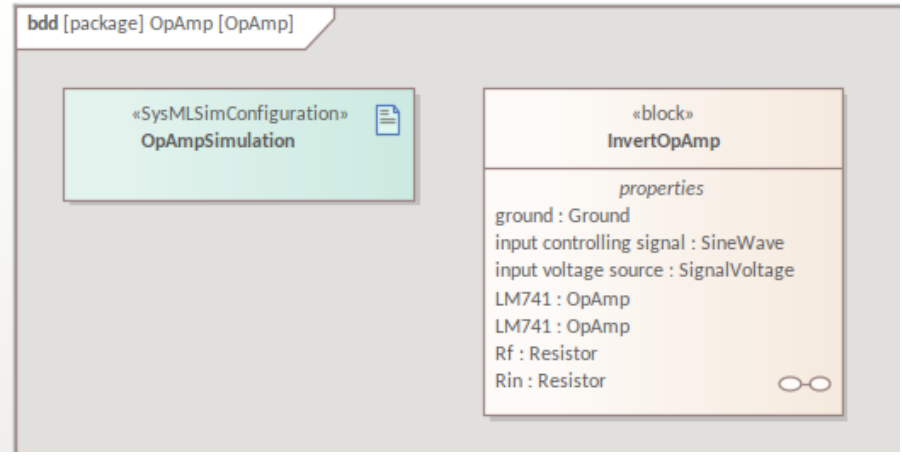
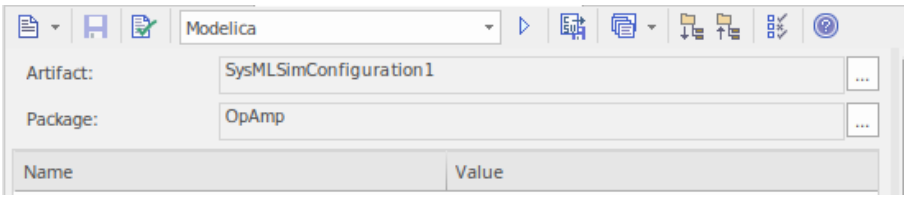


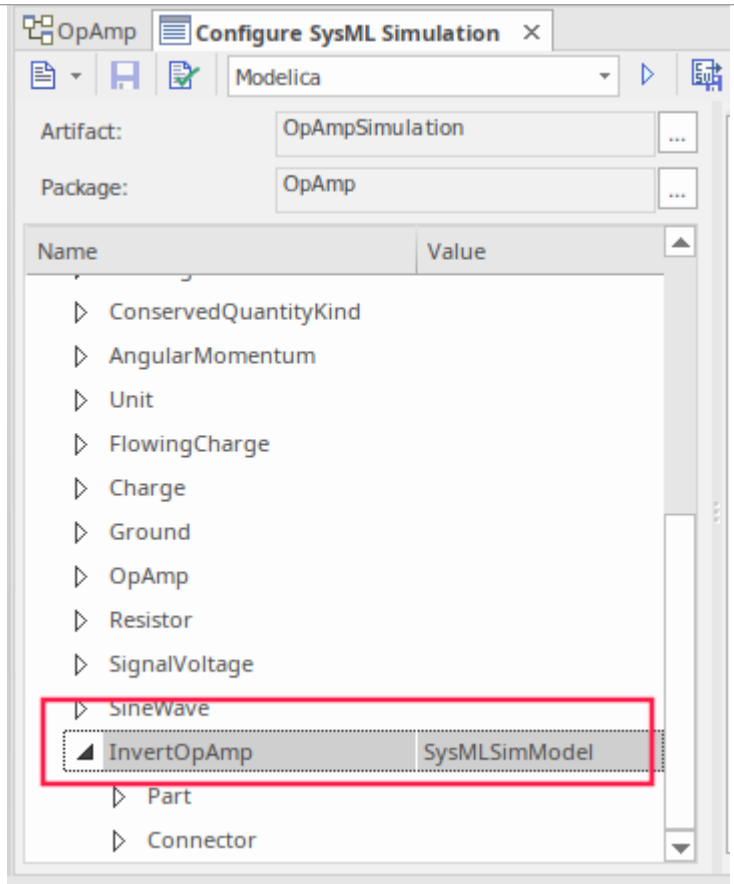
En revenant au BDD, vous devriez maintenant avoir le Bloc InvertOpAmp affiché comme suit :



Configurer le comportement Simulation

Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

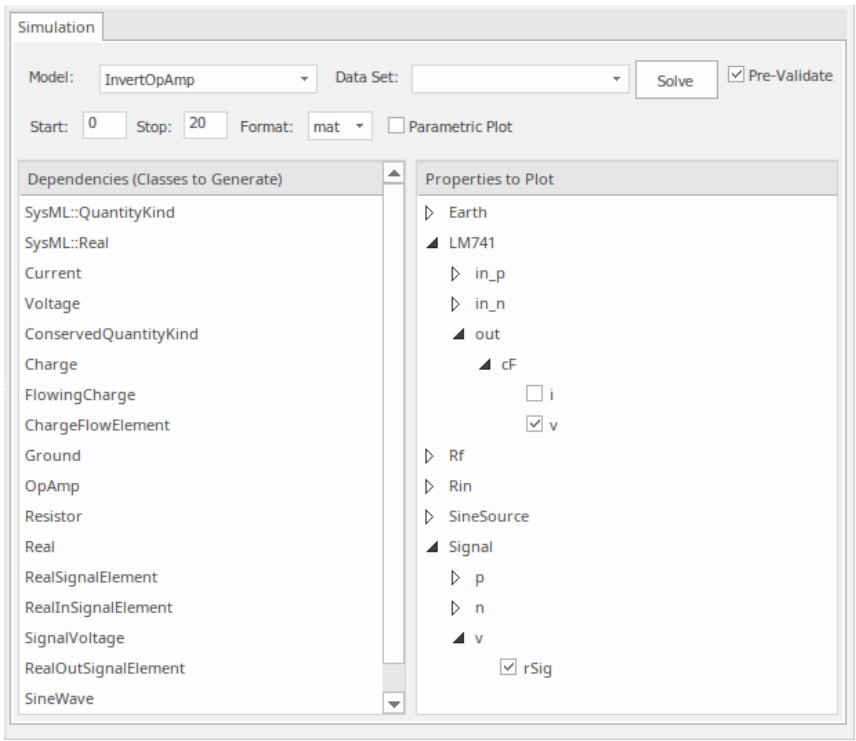
Étape	Action
Créer un artefact SysMLSimConfiguration	• Ouvrez le diagramme de définition Bloc • Cliquez sur l'espace ouvert dans le diagramme • Appuyez sur la barre d'espace • Dans le sous-menu « Artefacts », sélectionnez « Configuration SysMLSim » Cela crée un nouvel artefact de configuration SysMLSim : 
Définir le Paquetage	• Double-cliquez sur l'artefact de configuration SysMLSim Cela ouvre la fenêtre de configuration de SysML • Dans le champ ' Paquetage ' cliquez sur le bouton [...] et sélectionnez le Paquetage contenant le diagramme SysML : 
Configurer Modelica ou Simulink	Dans la liste déroulante supérieure, vous pouvez sélectionner l'outil de simulation à utiliser : • Modèleica • Simulink Pour plus de détails sur ces paramètres, consultez la rubrique d'aide *Configurer Simulation SysML* .
Régler le Bloc pour simuler	• Dans la liste de gauche, sous « Nom », recherchez « InvertOpAmp » • Dans la colonne « Valeur », cliquez sur le menu déroulant et sélectionnez « SysMLSimModel »



Sélectionner Propriétés à tracer

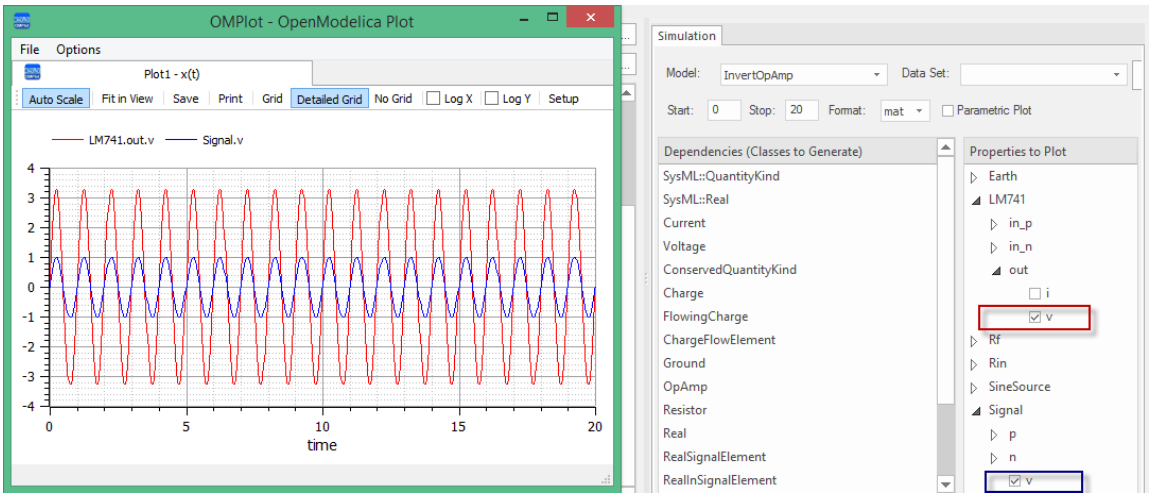
Vous pouvez maintenant sélectionner les Propriétés à tracer :

- Dans le volet de droite, sélectionnez les ports à tracer



Exécuter Simulation

Dans l'onglet « Simulation », cliquez sur le bouton Résoudre. Cet exemple montre un tracé généré dans Modelica :



Dans la légende, vous pouvez voir LM741.out.v et 'Signal.v qui sont tracés, et les faire correspondre avec les deux propriétés sélectionnées sous *Propriétés à tracer* .

Vue le Modèle dans Modelica ou Simulink

Pour visualiser le modèle généré dans les applications externes, Modelica ou Simulink, consultez la rubrique d'aide *Vue le Modèle dans Modelica ou Simulink* , ainsi que des conseils pour déboguer les problèmes dans le code généré.

Exemple Simulation électronique numérique

Pour cet exemple, nous parcourons la création d'un modèle SysPhS pour un circuit électronique numérique simple, puis utilisons une simulation pour prédire et cartographier le comportement de ce circuit.

Cet exemple fonctionne avec des composants qui ne sont pas inclus dans les composants communs SysPhS. Il exécute donc le processus de création de blocs pour correspondre aux composants externes à partir de zéro. Pour un webinaire illustrant cela, consultez le lien dans *En savoir plus* à la fin de la rubrique.

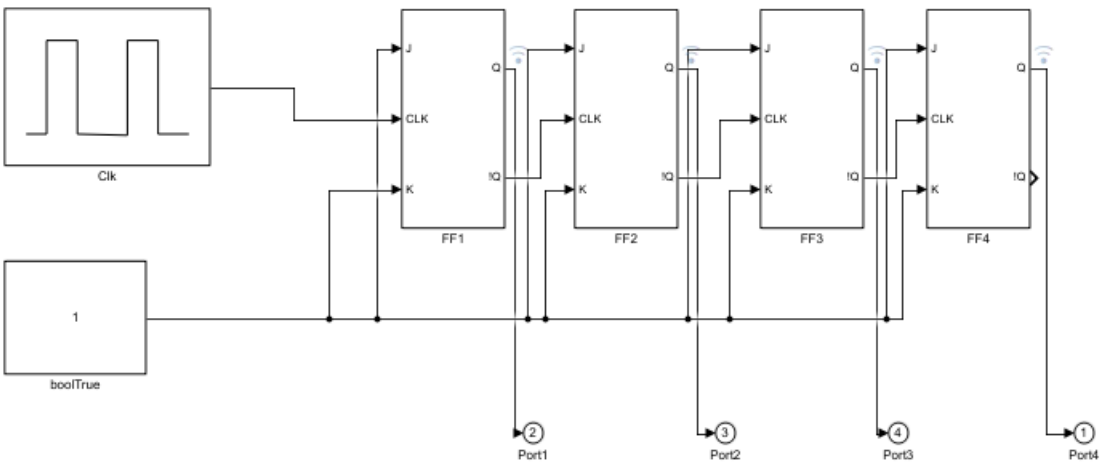
Prérequis

L'exécution de cette simulation nécessite :

- OpenModelica ou
- Simulink de MATLAB

Diagramme de circuit - un diviseur de fréquence numérique

Le circuit électronique numérique que nous allons modéliser est représenté dans cette figure, qui utilise la notation standard des circuits électroniques.

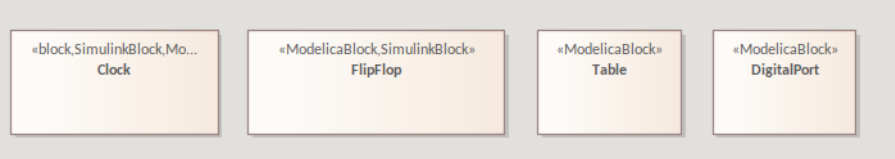
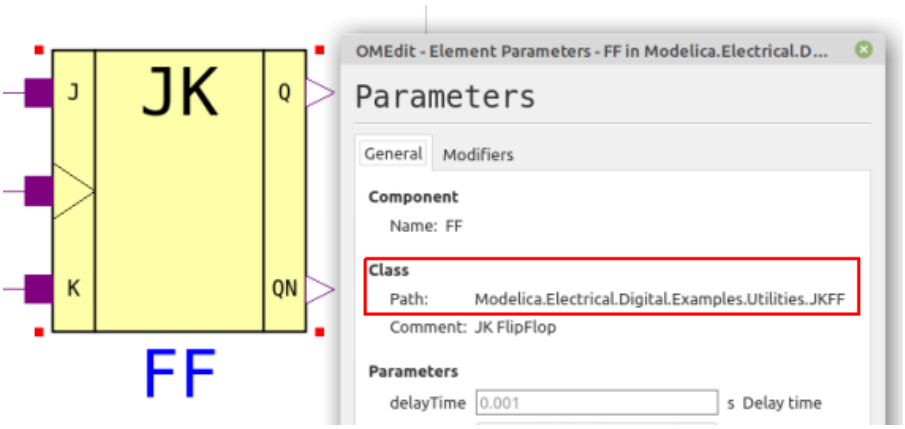
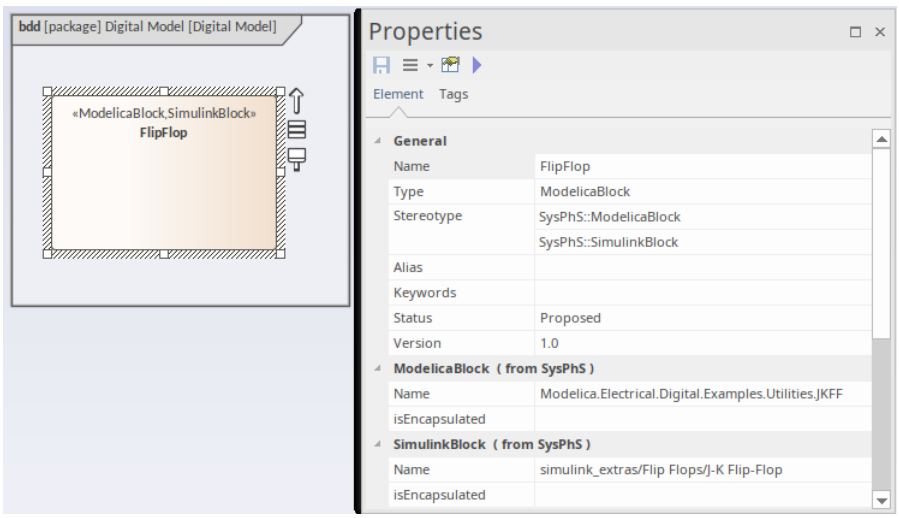


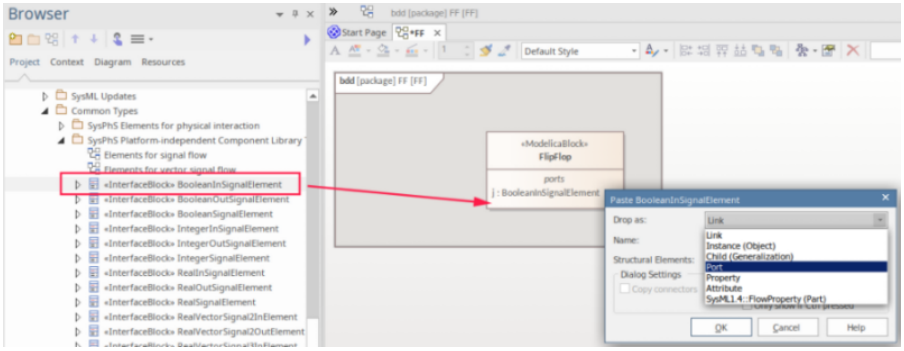
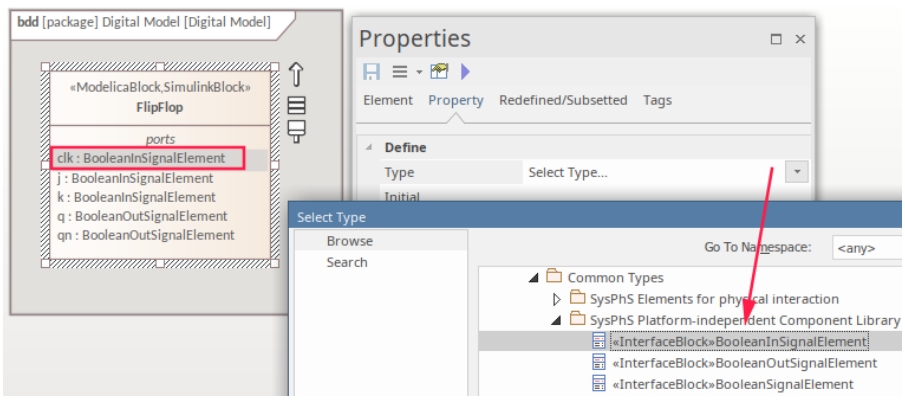
Le circuit de cet exemple comprend une source de signal numérique pulsé, quatre bascules et un état logique booléen vrai pour former un circuit diviseur de fréquence simple.

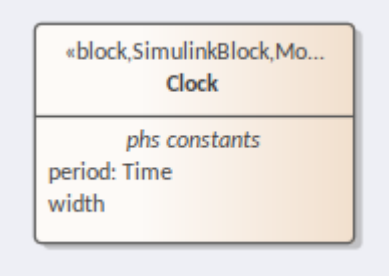
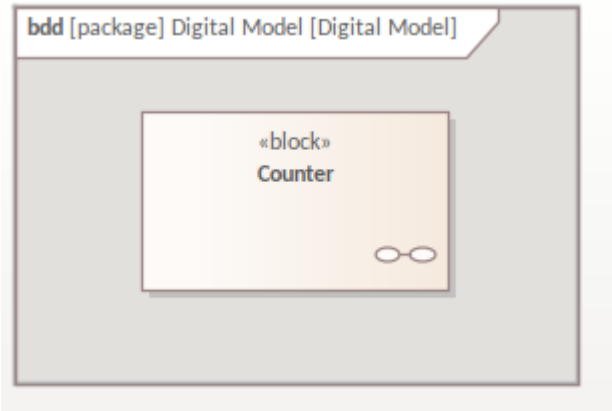
Créer Modèle SysML

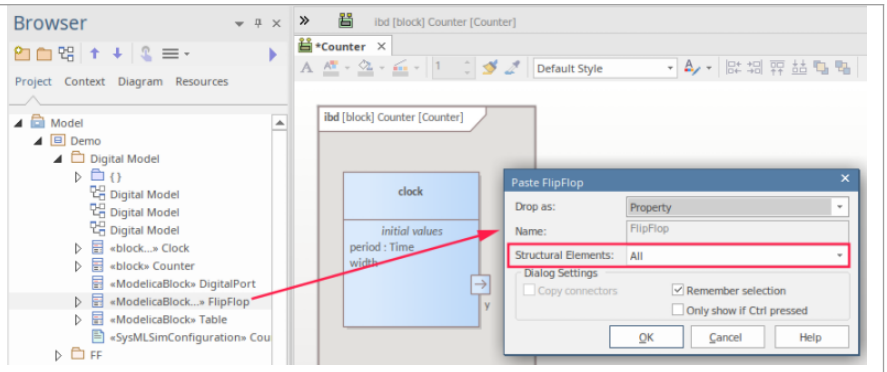
Ce tableau montre comment nous pouvons créer un modèle SysML complet pour représenter le circuit, en commençant par les types de niveau le plus bas et en construisant le modèle une étape à la fois.

Composant	Action
Blocs	Dans SysML, en utilisant SysPhS, le circuit et chacun des types de composants peuvent être représentés à l'aide d'un Bloc . Créez d'abord un diagramme de définition Bloc (BDD) sous un Paquetage appelé ' Modèle numérique'. Dans le BDD, vous allez créer un ensemble de composants pour le circuit, sous

	forme de blocs SysML. Le circuit contient des représentations de quatre types de composants : une source de signal numérique pulsé, une bascule, un port booléen et un état logique booléen vrai. Ces composants sont de types différents, avec des comportements différents. Créez un Bloc SysPhS pour chacun des types de composants. Les composants de la définition Bloc interne (IBD) du circuit seront connectés via des ports, qui représentent pins électriques. Ceux-ci doivent être définis dans le BDD. Cette figure montre le BDD, avec des blocs définissant les types de composants utilisés.  Note que ces blocs sont créés à partir de la boîte à outils SysPhS à l'aide de blocs Modelica ou de blocs Simulink. Vous pouvez saisir ces blocs dans les deux outils. Pour plus d'informations, consultez la rubrique d'aide *Définition des blocs comme Modelica et Simulink* .
Définition du chemin Modelica et Simulink	Afin de définir un Bloc spécifique à Modelica ou Simulink, vous devez accéder au Chemin du Composant dans l'application respective, puis le définir dans les Propriétés du Bloc . Par exemple, nous pouvons trouver le composant Flip-Flop dans Modelica.  Nous copions ensuite cela dans les propriétés du Bloc .  Pour plus de détails, consultez les rubriques d'aide *Création de blocs spécifiques à Modelica* et *Création de blocs spécifiques à Simulink* .

Ports PhS	Pour définir les ports sur les blocs (dans ce cas, le port d'horloge Flip-Flop), faites glisser un port Modelica ou Simulink PhS sur le Bloc . Ce port doit ensuite être saisi sous la forme d'un BooleanInSignal.  **Note :** • Pour Simulink, l'ordre de création des ports est essentiel - voir *Ordre des ports Simulink* dans la rubrique d'aide *Création de blocs spécifiques à Simulink et Simscape* • Ces ports peuvent être définis à la fois sur Modelica et Simulink en ajoutant le stéréotype de l'autre type de port
Types courants	Pour commencer tous les modèles SysPhS, vous devez vous assurer que les types communs SysPhS sont chargés dans le référentiel et référencés dans le nouveau modèle, à l'aide du connecteur Paquetage Import. Pour plus d'informations, consultez *Référencement des bibliothèques Simulation SysPhS* Rubrique d'aide. Les types de valeur utilisés pour les ports sont prédéfinis dans les bibliothèques Simulation SysPhS. Les deux principaux types utilisés sont les types de valeur BooleanInSignal et BooleanOutSignal. Cette figure montre le Bloc Flip-Flop dans le diagramme de définition Bloc , avec le port d'horloge défini sur le type de valeur BooleanInSignal et celui-ci étant référencé dans la fenêtre Navigateur . 
Constantes Phs	Les blocs Clock et Boolean-true ont tous deux des propriétés définies dans leurs composants respectifs dans MATLAB et Modelica. Prenons l'exemple de l'horloge. Pour ce type de composant, dans Simulink comme dans Modelica, nous devons définir une valeur pour la période de chaque impulsion et la largeur de chaque impulsion. Ces Propriétés doivent être définies et typées. La valeur des Propriétés sera définie dans l'IBD, le diagramme Paramétriques ou éventuellement dans les jeux de données de simulation. Pour définir la propriété définissant la période : • Faites glisser une constante Phs de la boîte à outils SysPhS vers l'horloge

	• Supprimer l'élément du diagramme pour l'afficher dans son compartiment • Dans la fenêtre Propriétés , onglet Propriétés , sélectionnez le champ ' Type ' • Cliquez sur [...] et sélectionnez « Heure » comme type  Pour définir les valeurs dans le composant individuel (Partie), consultez la ligne *Valeurs initiales* dans ce tableau .
Structure interne - le circuit	Pour la structure interne, nous créons un Bloc avec un diagramme IBD enfant. • Créer un Bloc pour un circuit Flip-Flop • Sous ce Bloc créez un diagramme définition Bloc interne (IBD) à l'aide de l'option de menu contextuel *Créer un nouveau Diagramme enfant | diagramme de définition Bloc interne*  • Double-cliquez pour ouvrir l'IBD • Définissez le diagramme pour afficher uniquement le nom du port, en utilisant : - F5 | Élément | Apparence de l'élément • Désactivez ensuite ces deux options : - Montrer des stéréotypes - Afficher Type de propriété Cela affichera alors uniquement le port et Type Sur l'IBD, vous créez des pièces et les connectez : • Depuis la fenêtre Navigateur , faites glisser les blocs sur l'IBD en tant que parties (Propriétés) • Pour visualiser ces pièces dans des compartiments, supprimez-les du diagramme Note : définissez l'option Coller pour « Éléments structurels » sur « TOUS »

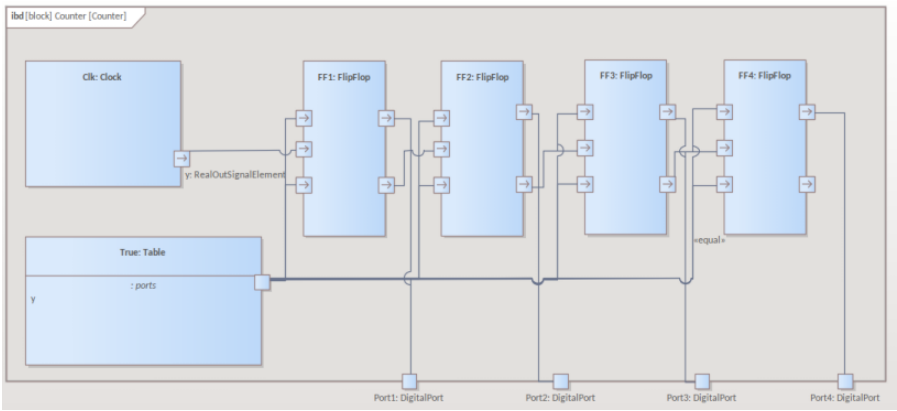


- Ajoutez une horloge, quatre tongs et un tableau
- Ajoutez quatre ports numériques à la bordure du compteur IBD

Fixations

Pour modéliser le câblage de ces composants :

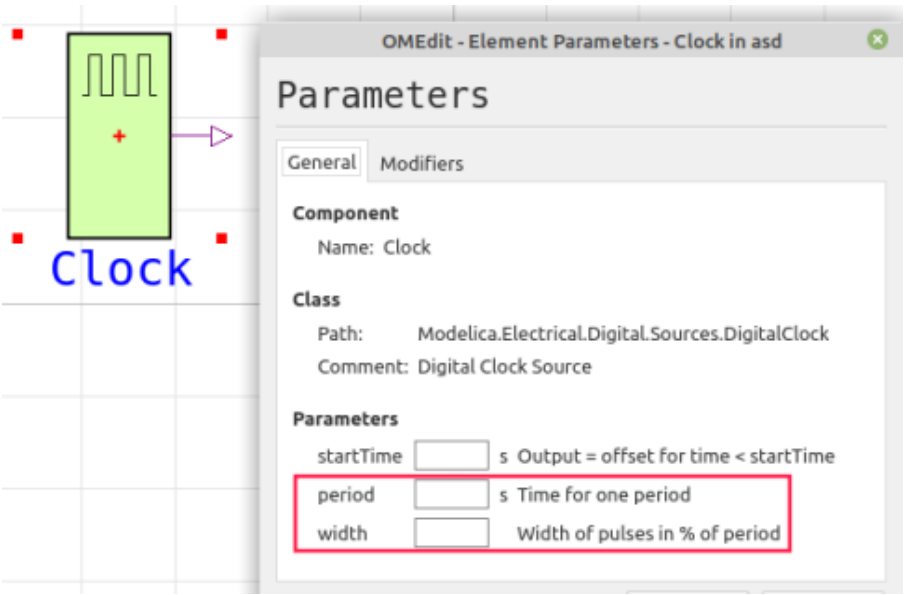
- Créer des connexions entre les ports avec des connecteurs de type « Connecteur »



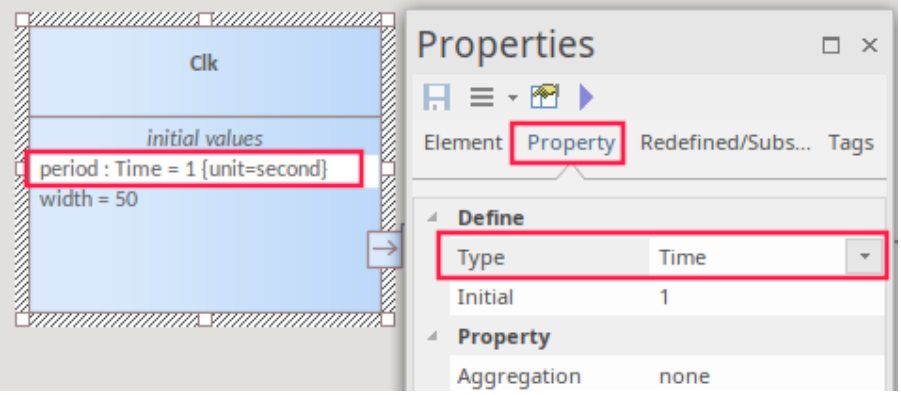
Notez que cela suit la même structure que le diagramme de circuit d'origine, mais les symboles de chaque composant ont été remplacés par des propriétés typées par les blocs que nous avons définis.

Valeurs initiales

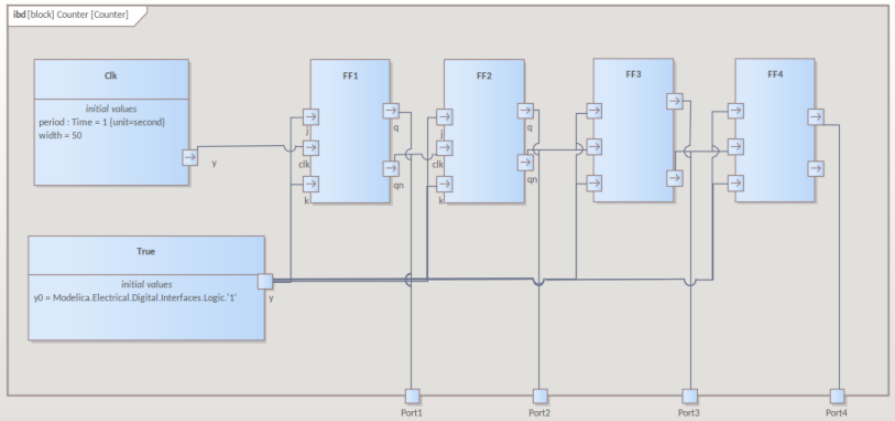
La source d'impulsions numériques est un composant DigitalClock dans Modelica et Simulink. Cela nécessite deux paramètres - « Période » et « Largeur », comme indiqué dans l'éditeur Modelica.



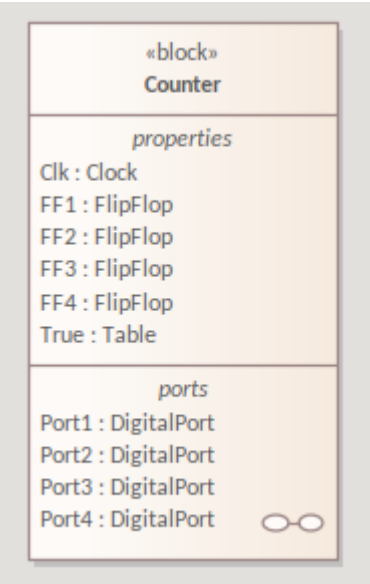
Les valeurs de ces paramètres doivent être définies dans la partie IBD 'Clk', dans la fenêtre Propriétés , onglet 'Propriété', champ 'Initial'



Les ports J et K nécessitent un état logique fixe « True ». Celui-ci est défini à l'aide d'un Tableau défini sur « true » à l'aide d'une valeur initiale, comme indiqué ici pour Modelica.

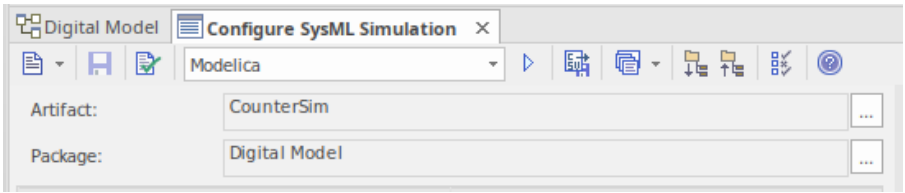


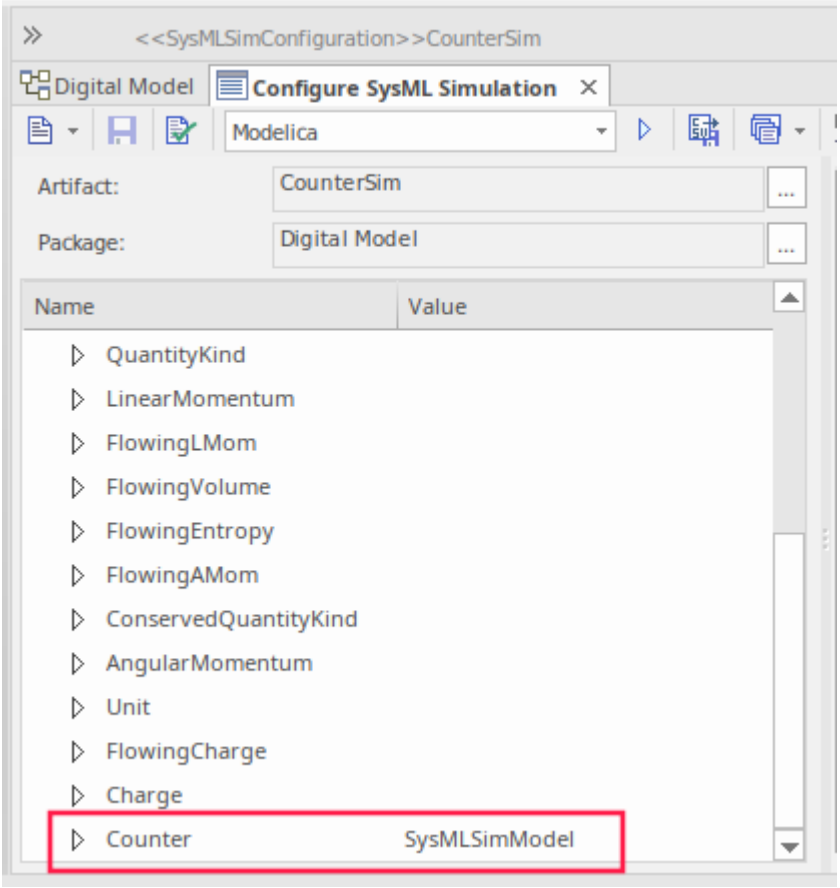
En revenant au BDD, vous devriez maintenant avoir le Counter Bloc affiché comme suit :



Configurer le comportement Simulation

Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

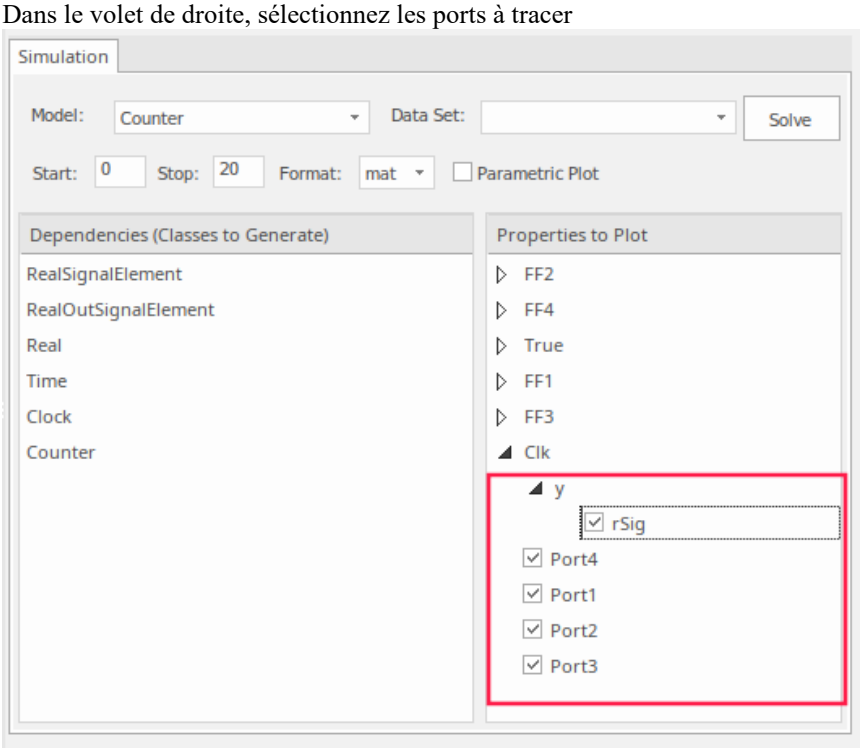
Étape	Action
Créer un artefact SysMLSimConfiguration	• Ouvrez le diagramme de définition Bloc • Cliquez sur l'espace ouvert dans le diagramme • Appuyez sur la barre d'espace • Dans le sous-menu « Artefacts », sélectionnez « Configuration SysMLSim » Cela crée un nouvel artefact de configuration SysMLSim
Définir le Paquetage	• Double-cliquez sur l'artefact de configuration SysMLSim Cela ouvre la fenêtre de configuration de SysML • Dans le champ « Paquetage », cliquez sur le bouton [...] et sélectionnez le Paquetage contenant le diagramme SysML 
Configurer Modelica ou Simulink	Dans la liste déroulante supérieure, sélectionnez l'outil de simulation à utiliser : • Modèleica • Simulink Pour plus de détails sur ces paramètres, consultez la rubrique d'aide *Configurer Simulation SysML* .
Régler le Bloc pour simuler	• Dans la liste de gauche, sous « Bloc », recherchez « InvertOpAmp » • Dans la colonne « Valeur », cliquez sur le menu déroulant et sélectionnez « SysMLSimModel »



Sélectionner Propriétés à tracer

Vous pouvez maintenant sélectionner les Propriétés à tracer :

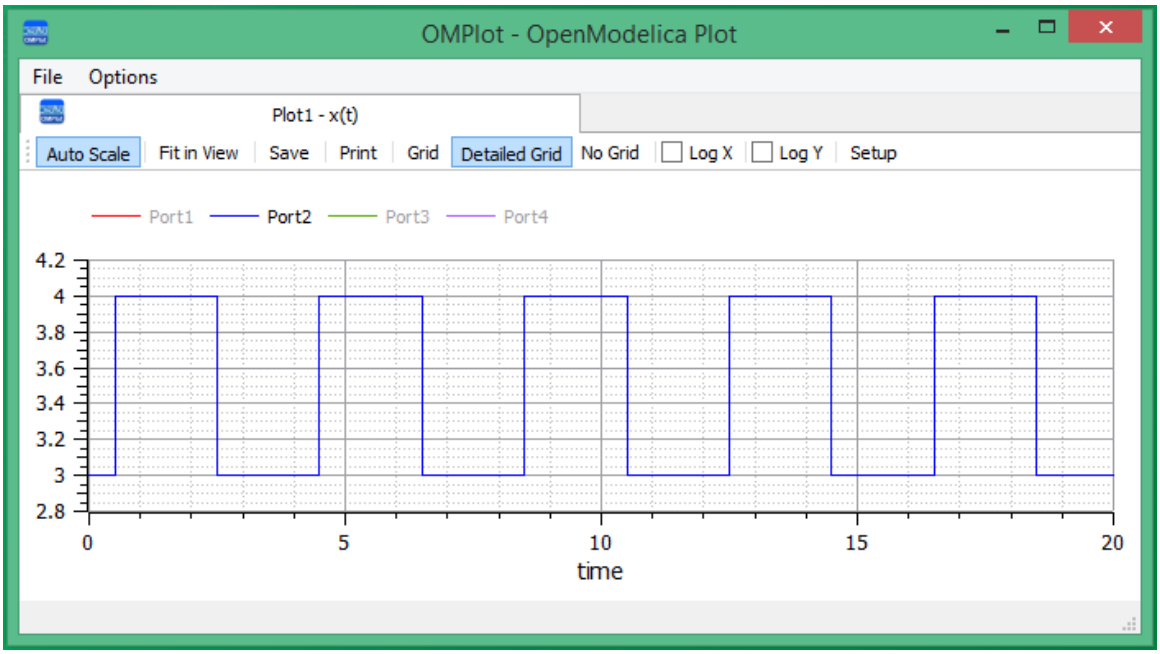
- Dans le volet de droite, sélectionnez les ports à tracer



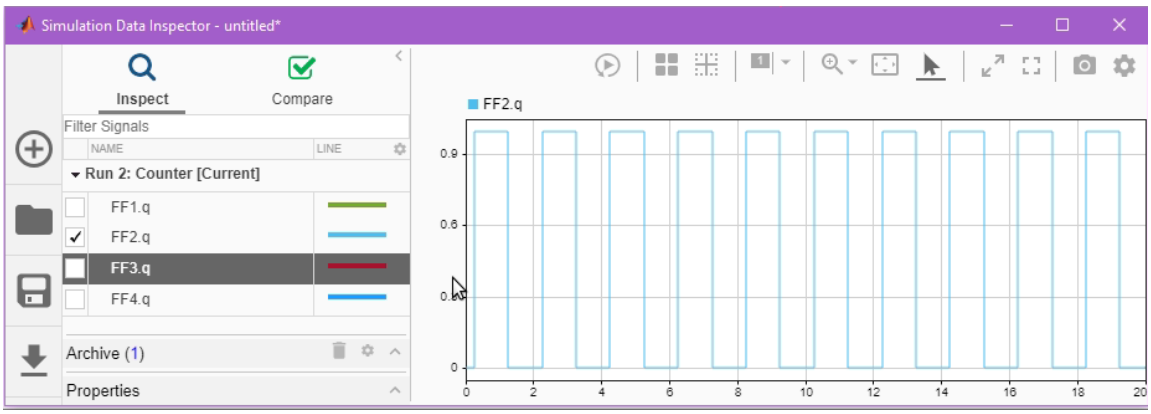
Exécuter Simulation

Sur la page « Simulation », cliquez sur le bouton Résoudre. Cela montre un exemple du tracé généré dans :

Modèleica



Simulink



Dans la légende, vous pouvez voir que le port 2 est sélectionné, tandis que les autres ports ont été désélectionnés pour afficher un *tracé* simple.

Vue le Modèle dans Modelica ou Simulink

Pour visualiser le modèle généré dans les applications externes, Modelica ou Simulink, consultez la rubrique d'aide *Visualisation du Modèle généré* . Consultez également les conseils de débogage des problèmes éventuels dans le code généré, dans la rubrique d'aide *Conseils de débogage SysPhS* .

Exemple d'humidificateur

Pour cet exemple, nous allons parcourir la création d'un modèle SysPhS d'un humidificateur d'ambiance pour illustrer l'utilisation des flux de signaux et Statemachines . Les signaux de cet exemple reflètent des quantités physiques, mais diffèrent de l'interaction physique dans le sens de substances physiques avec des débits et des potentiels.

L'exemple d'humidificateur complet est un ensemble complexe de modèles SysML, mais il contient une section qui nécessite l'utilisation de Stateflow de MATLAB pour la simulation. Nous utiliserons cette section de l'exemple pour travailler sur l'utilisation de la Statemachine SysML et voir comment elle est configurée et exécuter dans MATLAB à l'aide de Stateflow, ainsi que dans Modelica.

Prérequis

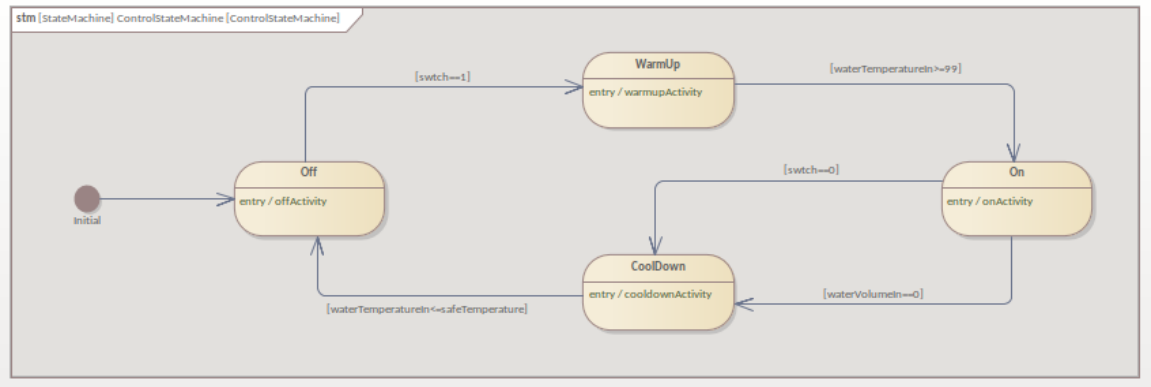
L'exécution de cette simulation nécessite :

- OpenModelica ou
- Simulink et StateFlow de MATLAB

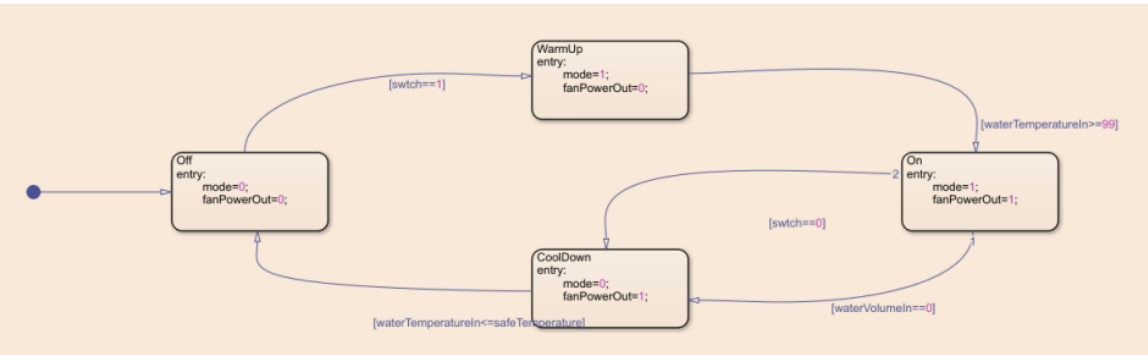
Aperçu

L'exemple de simulation complet est disponible dans le modèle WebEA . Consultez les liens sous *En savoir plus* à la fin de cette rubrique.

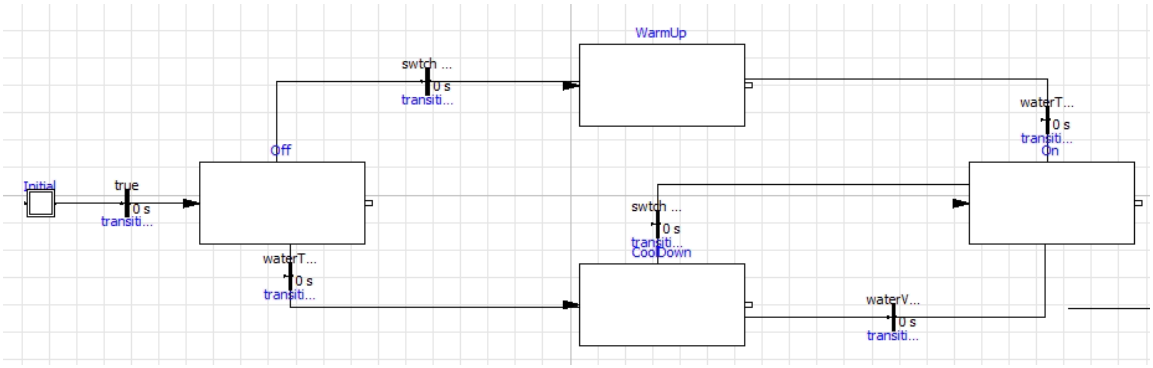
Le diagramme Statemachine SysML est :



Pour MATLAB, le diagramme Stateflow généré se compose de :



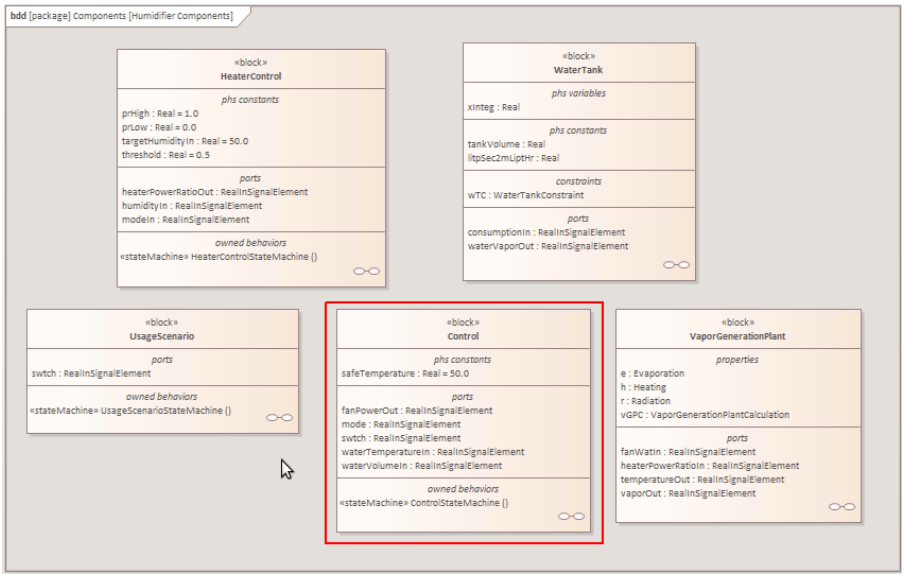
Pour Modelica, le diagramme généré ressemble à ceci :

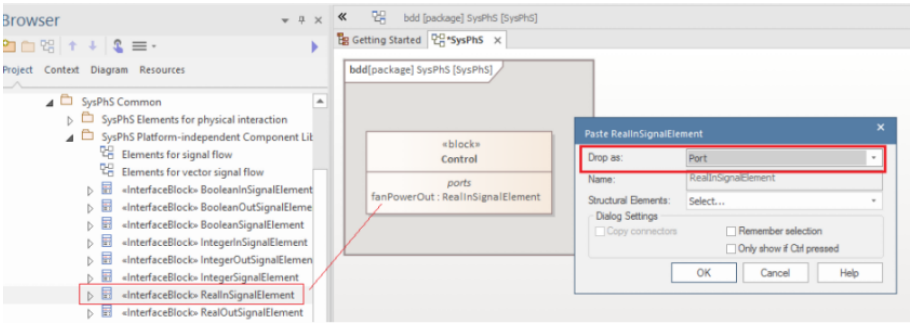
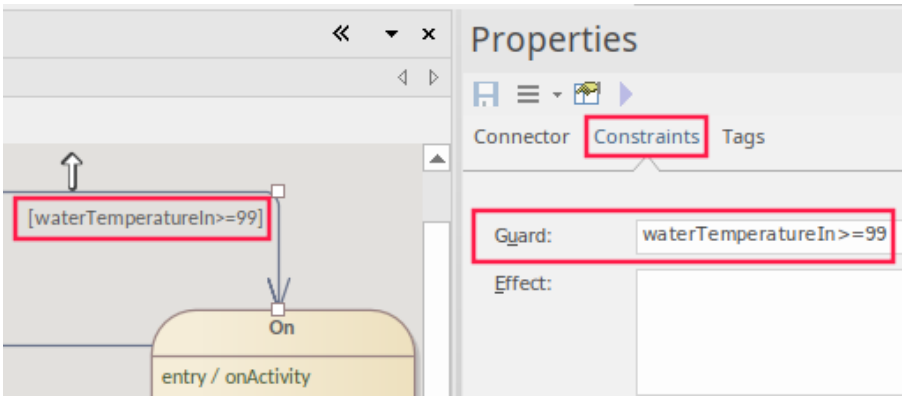


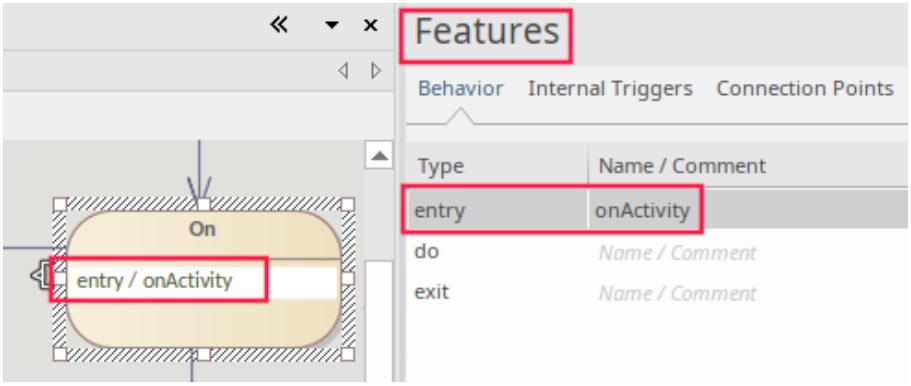
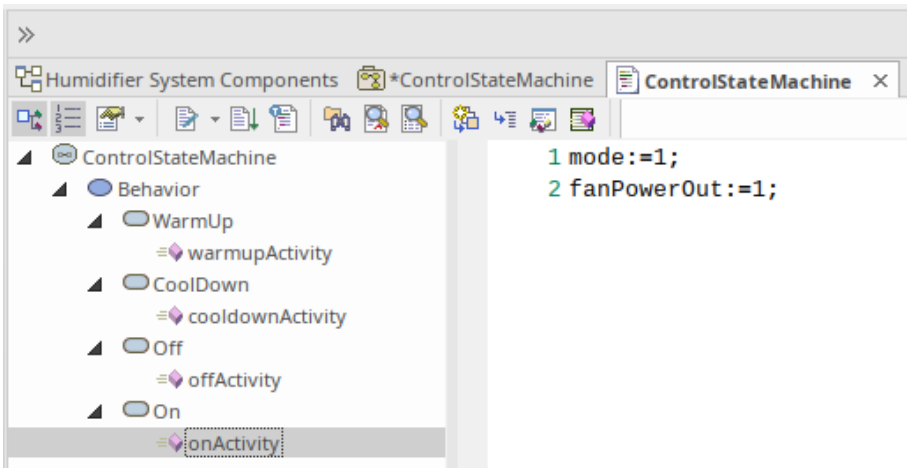
Créer Modèle Statemachine SysML

Ce tableau montre comment nous pouvons construire le modèle SysML Statemachine pour représenter les états de commutation.

Note : les fonctionnalités complètes de SysML Statemachine ne sont que partiellement prises en charge dans Stateflow et Modelica de MATLAB. Par conséquent, pour créer diagrammes State pour la simulation dans ces produits, il est nécessaire que les types de connecteurs et les types d'objets utilisés dans le modèle Statemachine d' Enterprise Architect contiennent uniquement les fonctionnalités correspondantes prises en charge dans Stateflow et Modelica.

Étape	Description
Blocs	Dans SysML, en utilisant SysPhS, les composants principaux de l'humidificateur sont représentés par des blocs. Pour cet exemple, où nous créons une simulation Statemachine , l'accent est mis sur le Bloc « Contrôle ». Le diagramme de définition Bloc (BDD) appelé « Composants de l'humidificateur », qui inclut le Bloc « Contrôle », n'est qu'un des nombreux BDD définissant l'humidificateur.  Le Bloc « Contrôle » contient la Statemachine « ControlStateMachine » sous forme de diagramme composite ; ceci est illustré dans le premier diagramme de la section *Présentation* au début de cette rubrique. Le Bloc « Contrôle » est l'endroit où sont définis les ports et une constante PhS pour le contrôle de l'humidificateur. Les ports sont de type RealInSignal, qui sont des flux de signaux.
Types courants	Pour commencer tous les modèles SysPhS, vous devez vous assurer que les types

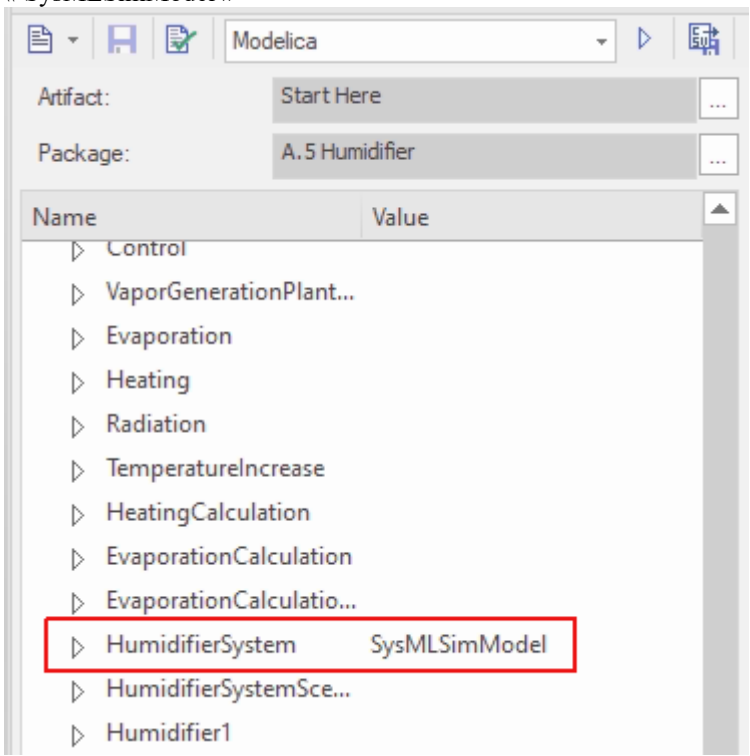
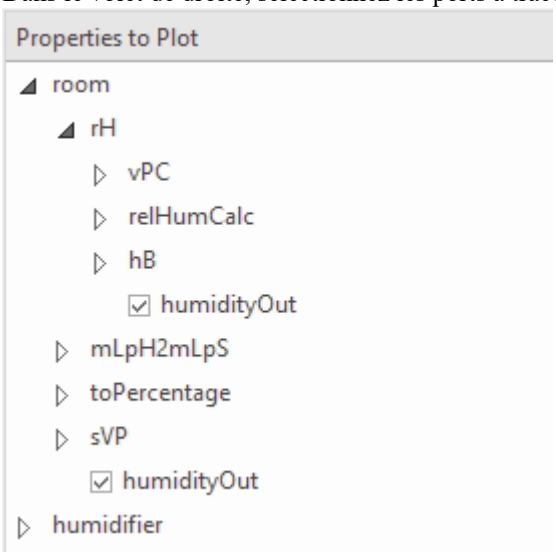
	communs SysPhS sont chargés dans le référentiel et référencés dans le nouveau modèle à l'aide du connecteur Paquetage 'Import'. Pour plus d'informations, consultez *Référencement des bibliothèques SysPhS* Rubrique d'aide.
Ports Phs	Les types de valeur utilisés pour les ports sont prédéfinis dans les bibliothèques Simulation SysPhS. Le type de clé utilisé est le RealInSignal ValueType (RealInSignalElement). Celui-ci peut être glissé sur le Bloc en tant que port, puis renommé. À titre d'exemple de définition d'un port sur le Bloc , dans ce cas le port *fanPowerOut* , vous faites glisser un RealInSignalElement sur le Bloc et le définissez comme port.  **Note :** • Ces ports peuvent être définis à la fois sur Modelica et Simulink en ajoutant le stéréotype de l'autre type de port
Création de la Statemachine	Pour définir une Statemachine en tant que diagramme enfant composite du Bloc : • Sélectionnez « Nouveau Diagramme enfant > Statemachine » dans le menu contextuel du Bloc , puis • Sélectionnez « Nouveau Diagramme enfant > Sélectionner un composite » dans le menu contextuel du Bloc , puis sélectionnez le diagramme qui vient d'être créé.
Créer les States et les transitions	Dans le diagramme Statemachine enfant, vous ajoutez l'élément initial, quatre States et un élément final. Nommez-les ensuite de manière appropriée. Note : en appuyant sur la barre d'espace lorsque le diagramme est sélectionné, vous pouvez sélectionner un object initial dans le menu contextuel, puis utiliser le Quick Linker pour ajouter les objets restants.
Gardes de transition	Les Transitions entre les States contiennent des conditions dans leurs Gardes. Ces conditions sont définies dans l'onglet 'Contraintes' de la fenêtre Propriétés . Les variables utilisées sont la constante Phs et les ports du Bloc « Contrôle ». Voir l'image dans la ligne *Blocs* précédente. 

Comportements	Chaque State contient un script définissant le mode et l'état du ventilateur. Ces scripts sont définis dans l'opération « Entrée ».  Pour modifier le script, sélectionnez la Statemachine racine et appuyez sur Alt+7. 

Configurer le comportement Simulation

Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

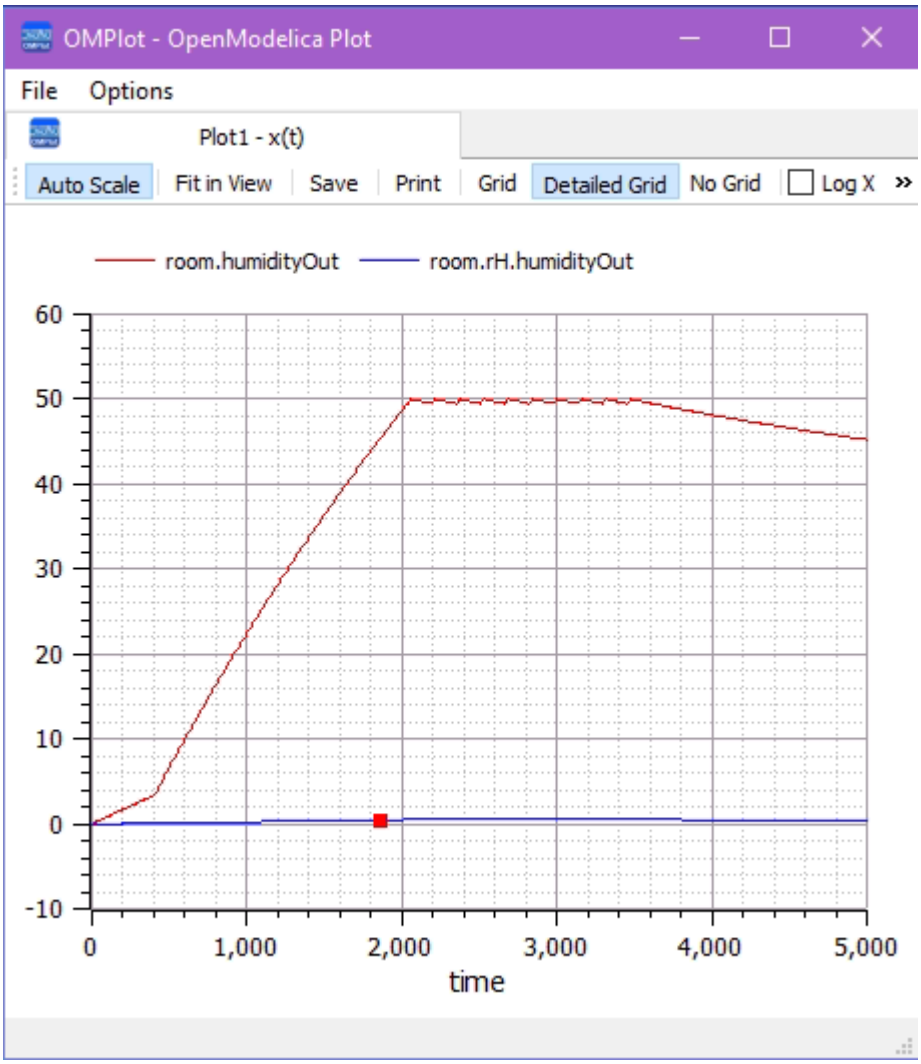
Étape	Action
Créer un artefact SysMLSimConfiguration	• Ouvrez le diagramme de définition Bloc • Cliquez sur l'espace ouvert dans le diagramme • Appuyez sur la barre d'espace • Dans le sous-menu « Artefacts », sélectionnez « Configuration SysMLSim » Cela crée un nouvel artefact de configuration SysMLSim
Définir le Paquetage	• Double-cliquez sur l'artefact de configuration SysMLSim Cela ouvre la fenêtre Configurer la Simulation SysML • Dans le champ « Paquetage », cliquez sur le bouton [...] et sélectionnez le Paquetage contenant le diagramme SysML
Configurer Modelica ou Simulink	Dans la liste déroulante supérieure, sélectionnez l'outil de simulation à utiliser : • Modèleica • Simulink

	Pour plus de détails sur ces paramètres, consultez la rubrique d'aide *Configurer Simulation SysML* .
Régler le Bloc pour simuler	• Dans la liste de gauche, sous « Bloc » dans la colonne « Nom », recherchez « HumidifierSystem » • Dans la colonne « Valeur », cliquez sur la flèche déroulante et sélectionnez « SysMLSimModel » 
Sélectionner Propriétés à tracer	Vous pouvez maintenant sélectionner les propriétés à tracer : • Dans le volet de droite, sélectionnez les ports à tracer 

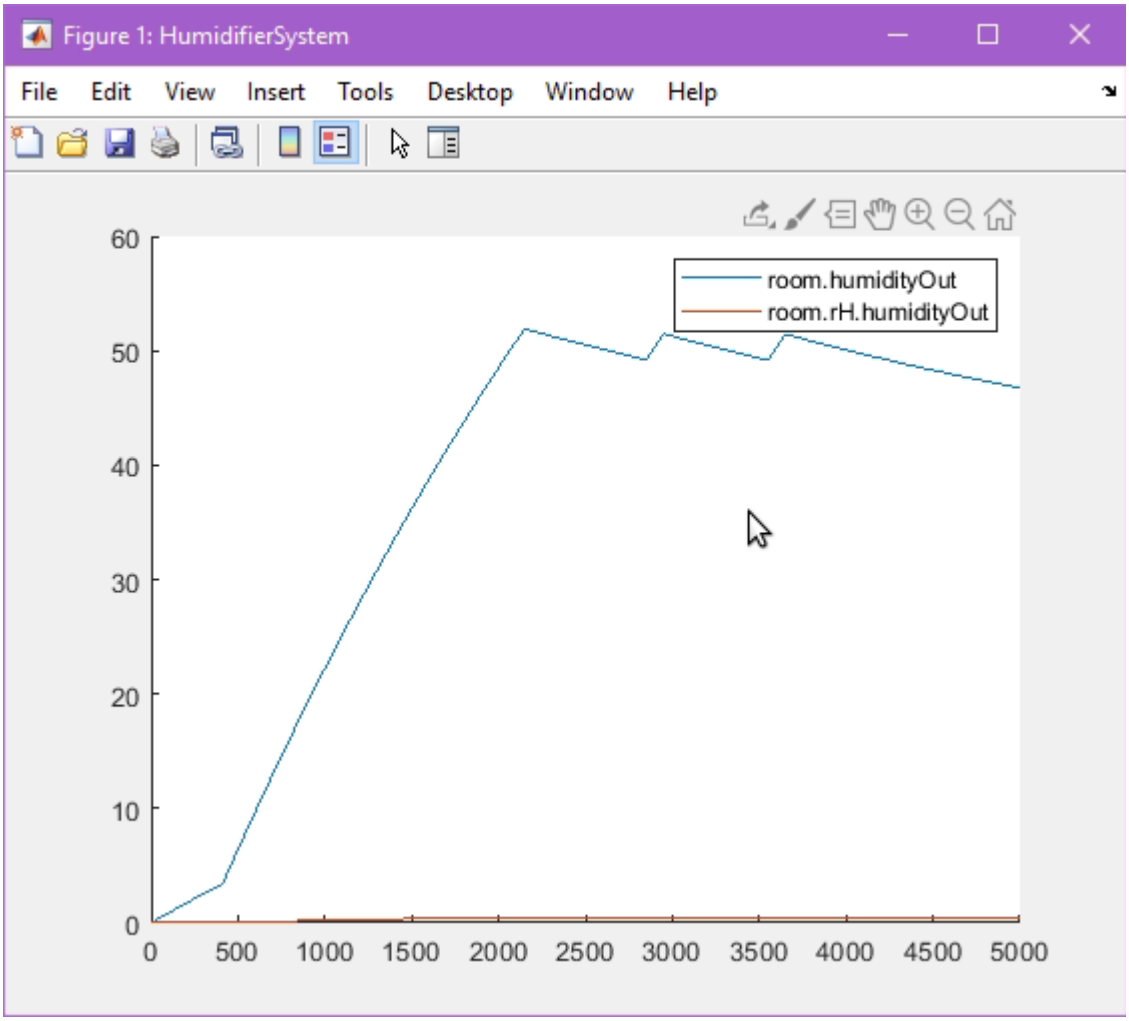
Exécuter Simulation

Dans la page « Simulation », cliquez sur le bouton Résoudre. Cela montre un exemple du tracé généré dans :

Modèleica



Simulink



Note : la sortie dans Simulink est différente en raison de la précision par défaut des modèles Simulink qui n'est pas suffisante pour cet exemple spécifique. Les paramètres Simulink peuvent être modifiés en ouvrant le fichier généré directement dans Simulink.

Vue le Modèle dans Modelica ou Simulink

Pour afficher le modèle généré dans les applications externes, Modelica ou Simulink, consultez la rubrique d'aide *Vue le Modèle dans Modelica ou Simulink* , qui inclut des conseils pour déboguer les problèmes dans le code généré.

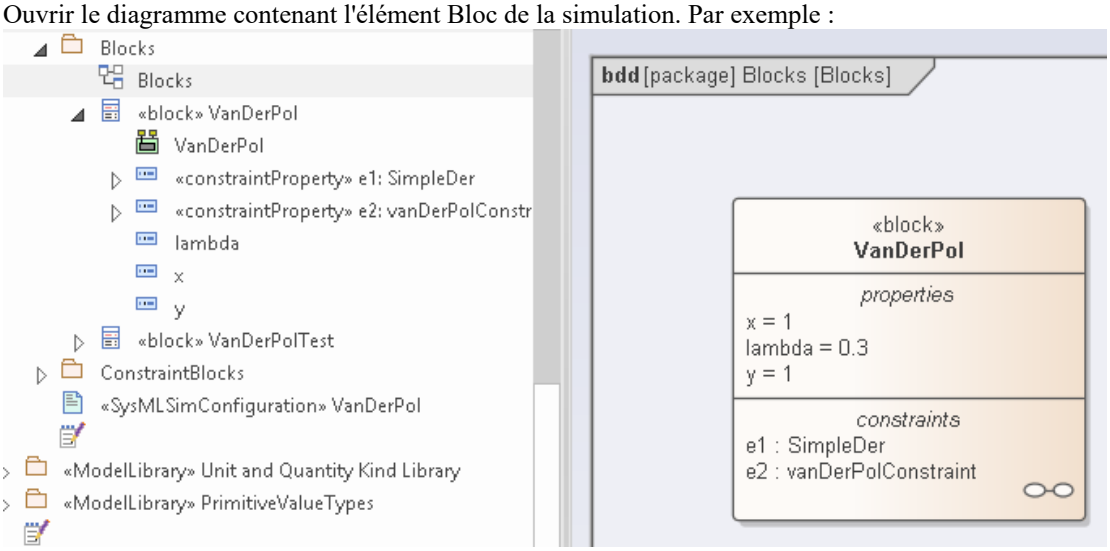
Mise à jour de la configuration SysMLSim

Dans les versions d' Enterprise Architect antérieures à la version 15.2, pour chaque pièce/propriété, le type et les options étaient définis dans la fenêtre Configurer Simulation SysML, ouverte à partir de l'artefact de configuration SysMLSim. Cette approche est toujours valable et disponible ; cependant, en utilisant les dernières fonctionnalités de la norme SysPhS, vous pouvez maintenant définir les paramètres de simulation dans le modèle lui-même, ce qui signifie que vous pouvez générer vers différents outils de simulation (Simulink et Modelica), à partir du même modèle et effectuer des ajustements au niveau élémentaire.

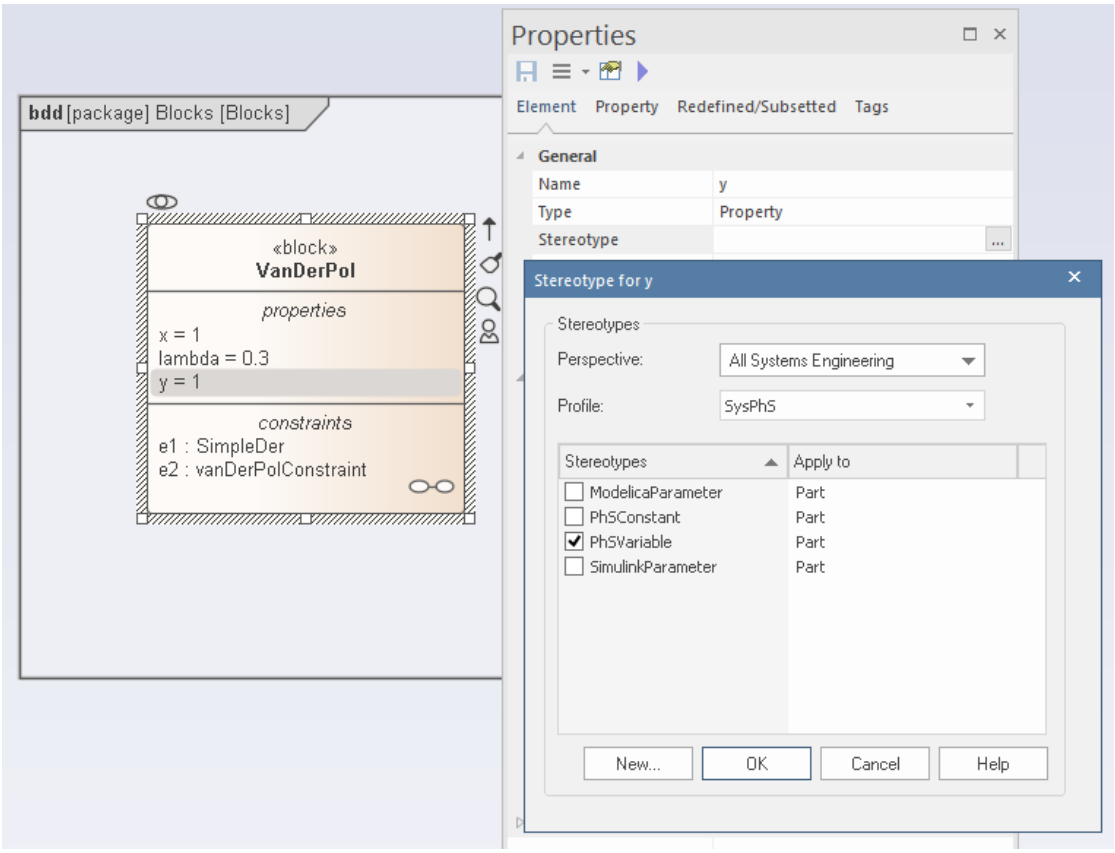
Si vous préférez utiliser l'option SysPhS, vous pouvez mettre à jour les configurations de simulation récentes pour refléter l'utilisation de la norme SysPhS et support la simulation via différents outils.

Pour mettre à jour les configurations existantes :

1. Assurez-vous de disposer d'une référence *d'importation* Paquetage vers les bibliothèques Simulation SysPhS. Pour plus de détails, consultez la rubrique d'aide *Référencement des bibliothèques Simulation SysPhS* .
2. Ouvrir le diagramme contenant l'élément Bloc de la simulation. Par exemple :

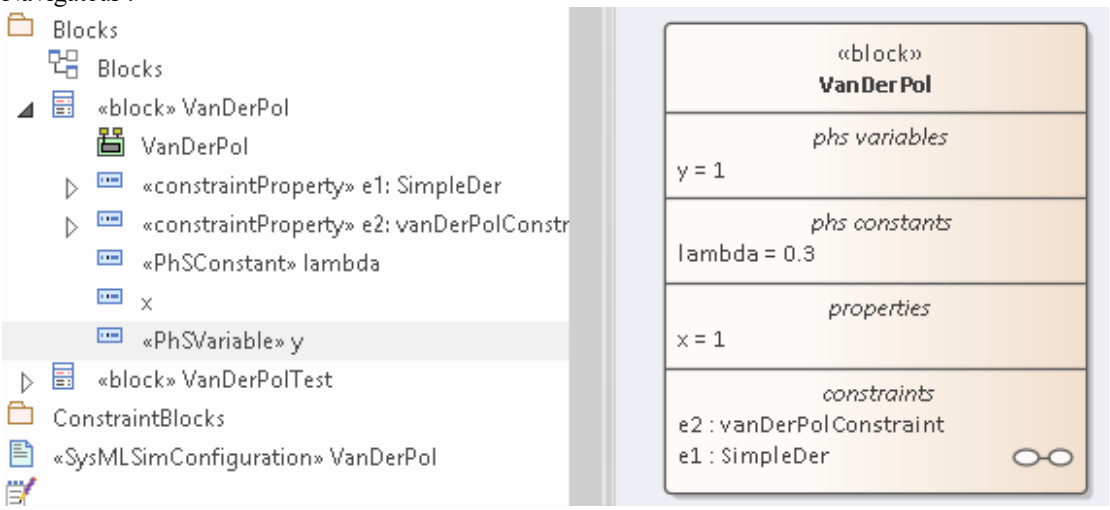


3. Remarquez les éléments Propriété (dans notre illustration, *x* , *y* et *lambda*) dans la fenêtre Navigateur ; ni là, ni dans le diagramme , ni dans la fenêtre Propriétés , il n'est possible de voir de quel type de propriété chacun d'eux est.
4. Dans l'élément du diagramme , cliquez sur une propriété et, dans le champ « Stéréotype » de la fenêtre Propriétés , cliquez sur l'icône ... pour afficher la dialogue « Stéréotype pour <nom de la propriété> ». Assurez-vous que le champ « Perspective » est défini sur « All Ingénierie des Systèmes » et que le champ Profil est défini sur « SysPhS ».

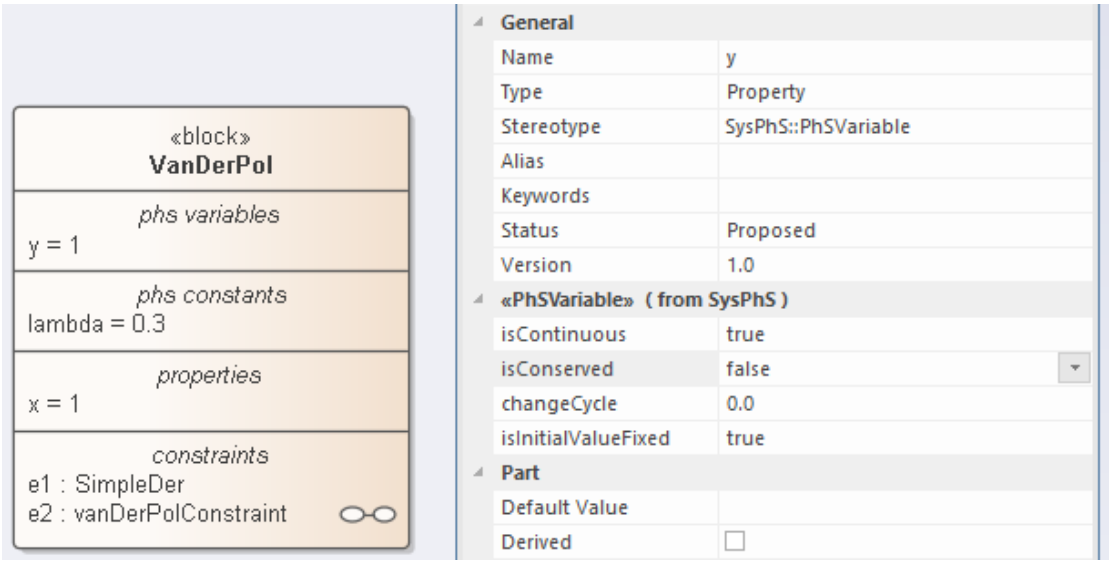


5. Cliquez sur la case à cocher correspondant au type de propriété approprié - PhSVariable ou PhSConstant - et cliquez sur le bouton OK .

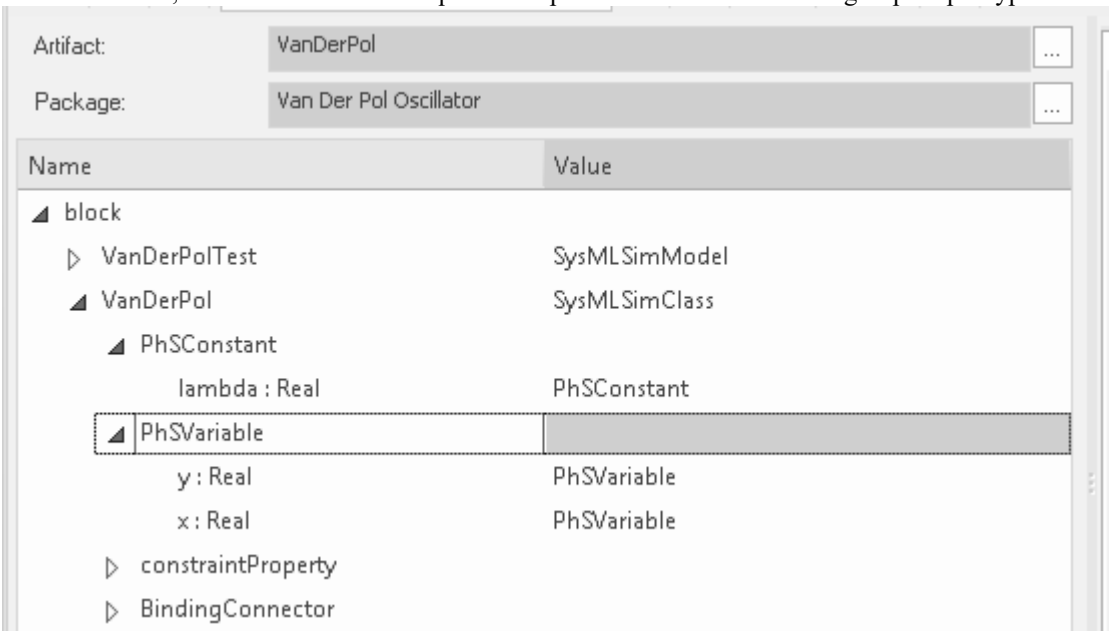
6. Note comment les constantes et les variables sont affectées à leurs propres compartiments d'éléments sur le diagramme , et comment les types de propriétés (en tant que stéréotypes) peuvent également être vus dans la fenêtre Navigateur .



7. Notez également que, dans la fenêtre Propriétés , vous pouvez voir le stéréotype définissant le type de propriété, et les options pour ce type de propriété (comme Valeur Étiquetés) telles que définies dans la norme SysPhS.



8. Enfin, si vous ouvrez la fenêtre Configurer la Simulation SysML pour l'artefact de configuration SysMLSim pour cette simulation, vous verrez à nouveau que les Propriétés sont identifiées et regroupées par type.



SysML Simulation Paramétrique

Enterprise Architect fournit une intégration avec OpenModelica et MATLAB Simulink pour support une évaluation rapide et robuste du comportement d'un modèle SysML dans différentes circonstances.

Les bibliothèques OpenModelica sont des ressources complètes qui fournissent de nombreux types, fonctions et modèles utiles. Lors de la création de modèles SysML dans Enterprise Architect , vous pouvez référencer les ressources disponibles dans ces bibliothèques.

L'intégration MATLAB d' Enterprise Architect se connecte via l'API MATLAB, permettant à vos simulations Enterprise Architect et autres scripts d'agir en fonction de la valeur de toutes les fonctions et expressions MATLAB disponibles. Vous pouvez appeler MATLAB via une classe Solveur ou exporter votre modèle vers MATLAB Simulink, Simscape et/ou Stateflow.

fonctionnalités de SysML Simulation

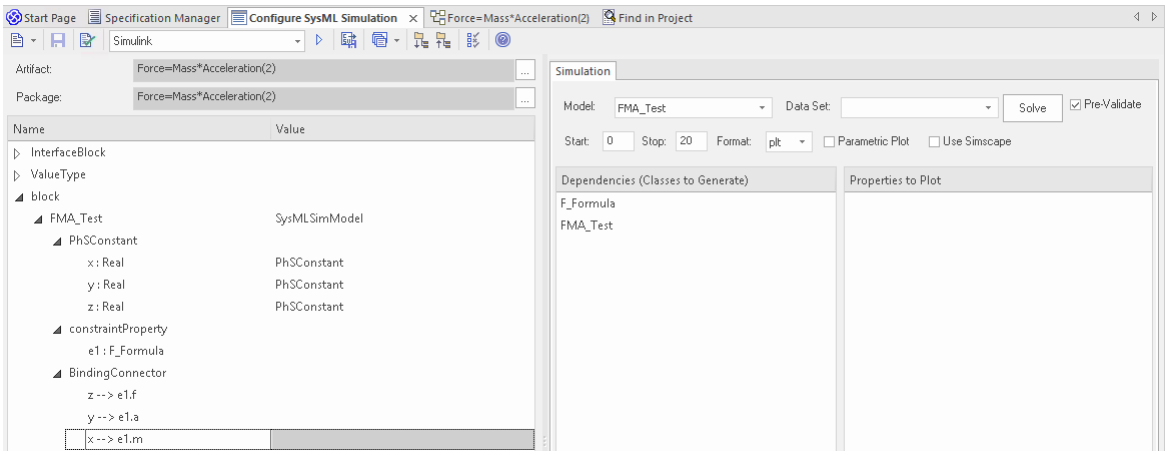
Ces sections décrivent le processus de définition d'un modèle Paramétriques , l'annotation du modèle avec des informations supplémentaires pour piloter une simulation et l'exécution d'une simulation pour générer un graphique des résultats.

Section	Description
Introduction aux modèles Paramétriques SysML	Les modèles SysML Paramétriques support l'analyse technique des paramètres critiques du système, notamment l'évaluation des mesures clés telles que les performances, la fiabilité et d'autres caractéristiques physiques. Ces modèles combinent les modèles Exigences avec les modèles de conception de système, en capturant les contraintes exécutables basées sur des relations mathématiques complexes. diagrammes Paramétriques sont diagrammes Bloc internes spécialisés qui vous aident, en tant que modélisateur, à combiner des modèles de comportement et de structure avec des modèles d'analyse technique tels que les modèles de performances, de fiabilité et de propriétés de masse. Pour plus d'informations sur les concepts des modèles SysML Paramétriques , reportez-vous au site officiel OMG SysML et à ses sources liées.
Création d'un Modèle Paramétrique	Un aperçu du développement d'éléments de modèle SysML pour la simulation, de la configuration de ces éléments dans la fenêtre Configurer Simulation SysML et de l'observation des résultats d'une simulation.
Artefact de configuration SysMLSim	Enterprise Architect vous aide à étendre l'utilité de vos modèles SysML Paramétriques en les annotant avec des informations supplémentaires qui permettent de simuler le modèle. Le modèle résultant est ensuite généré sous forme de modèle pouvant être résolu (simulé) à l'aide de MATLAB Simulink ou d'OpenModelica. Les propriétés de simulation de votre modèle sont stockées dans un artefact Simulation . Cela préserve votre modèle d'origine et supporte plusieurs simulations configurées par rapport à un seul modèle SysML. L'artefact Simulation se trouve sur la page de la boîte à outils « Artefacts ».
Interface Utilisateur	L'interface utilisateur de la simulation SysML est décrite dans la rubrique *Configurer la fenêtre Simulation SysML* .
Analyse Modèle à l'aide d'un ensemble de données	En utilisant la configuration Simulation un Bloc SysML peut avoir plusieurs jeux de données définis par rapport à lui. Cela permet d'exécuter des variations répétables sur une simulation du modèle SysML.
Support de la norme	La *norme SysPhS* est une *extension SysML pour Simulation d'interaction physique*

SysPhS	*et de flux de signaux* . Elle définit une méthode standard de traduction entre un modèle SysML et un modèle Modelica ou un modèle Simulink/Simscape, offrant ainsi une méthode plus simple basée sur un modèle pour le partage de simulations. Consultez la rubrique d'aide *Support de la norme SysPhS* .
Exemples	Pour vous aider à comprendre comment créer et simuler un modèle SysML Paramétriques , trois exemples ont été fournis pour illustrer trois domaines différents. Ces trois exemples utilisent les bibliothèques OpenModelica. Ces exemples et ce que vous pouvez en apprendre sont décrits dans la rubrique *Exemples Simulation SysML* .

Configurer Simulation SysML

La fenêtre Configurer Simulation SysML est l'interface par laquelle vous pouvez fournir des paramètres d'exécution pour exécuter la simulation d'un modèle SysML. La simulation est basée sur une configuration de simulation définie dans un élément d'artefact SysMLSimConfiguration.

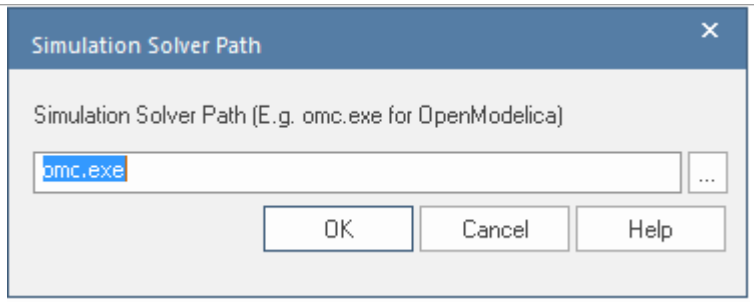


Accéder

Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager
Autre	Double-cliquez sur un artefact avec le stéréotype SysMLSimConfiguration.

Options de la barre d'outils

Option	Description
	Cliquez sur la flèche déroulante et sélectionnez l'une de ces options : • Sélectionner un artefact — Sélectionnez et chargez une configuration existante à partir d'un artefact avec le stéréotype SysMLSimConfiguration (si aucun n'a déjà été sélectionné) • Créer un artefact — Créez une nouvelle configuration SysMLSim ou sélectionnez et chargez un artefact de configuration existant • Sélectionner Paquetage — Sélectionnez un Paquetage pour rechercher les éléments SysML à configurer pour la simulation • Recharger — Recharger le gestionnaire de configuration avec les modifications apportées au Paquetage actuel • Configurer Solveur Simulation — Affichez la dialogue « Chemin Solveur Simulation », dans laquelle vous pouvez saisir ou rechercher le chemin d'accès au Solveur à utiliser. Pour MATLAB/Simulink, le chemin sera automatiquement détecté. Il ne doit donc être modifié qu'en cas de problème avec la détection automatique ou si plusieurs versions de MATLAB sont installées.

	
	Cliquez sur ce bouton pour enregistrer la configuration de l'artefact actuel.
	Cliquez sur cette icône pour valider spécifiquement le modèle par rapport à la configuration SysML maintenant . Les résultats de la validation s'affichent dans l'onglet « Simulation SysML » de la fenêtre Sortie système. Vous pouvez également sélectionner une option pour pré-valider automatiquement le modèle avant l'exécution de chaque simulation. Voir l'option « Pré-valider » dans le tableau *de l'onglet Simulation* .
	Cliquez sur cette icône pour développer chaque élément de la hiérarchie dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour réduire tous les éléments développés dans la hiérarchie du modèle dans la colonne « Nom » de la fenêtre.
	Cliquez sur cette icône pour afficher une liste des types object qui peuvent être supprimés dans la simulation. Cliquez sur la case à cocher en regard de chaque object à supprimer ou cliquez sur le bouton Tous pour sélectionner tous les éléments à supprimer. Vous pouvez également utiliser la Barre de Filtre en haut de la colonne « Option » pour afficher uniquement les éléments ayant la lettre ou string de texte spécifiée dans le nom.
	Cliquez sur la flèche déroulante et sélectionnez l'application sous laquelle la simulation est exécuter - comme OpenModelica ou Simulink.
	Cliquez sur ce bouton pour générer, compiler et exécuter la configuration actuelle, et afficher les résultats.
	Après la simulation, le fichier de résultats est généré au format plt, mat ou csv. C'est-à-dire avec le nom de fichier : • ModelName_res.mat (la valeur par défaut pour OpenModelica) • ModelName_res.plt ou • Nom du modèle_res.csv Cliquez sur ce bouton pour spécifier un répertoire dans lequel Enterprise Architect copiera le fichier de résultats.
	Cliquez sur ce bouton pour sélectionner parmi ces options : • Exécuter le dernier code - Exécuter le code le plus récemment généré • Générer du code — Générer le code sans le compiler ni l'exécuter • Ouvrir le répertoire Simulation — Ouvrir le répertoire dans lequel le code OpenModelica ou Simulink sera généré • Modifier Gabarits — Personnalisez le code généré pour OpenModelica ou

	Simulink, à l'aide de l'éditeur de code Gabarit

Artefact Simulation et sélection Modèle

Champ	Action
Artefact	Cliquez sur l'icône  et recherchez et sélectionnez un artefact SysMLSimConfiguration existant ou créez un nouvel artefact.
Paquetage	Si vous avez spécifié un artefact SysMLSimConfiguration existant, ce champ correspond par défaut au Paquetage contenant le modèle SysML associé à cet artefact. Sinon, cliquez sur l'icône  et recherchez et sélectionnez le Paquetage contenant le modèle SysML à configurer pour la simulation. Vous devez spécifier (ou créer) l'Artefact avant de sélectionner le Paquetage .

Objets Paquetage

Ce tableau décrit les types d' object du modèle SysML qui seront répertoriés sous la colonne « Nom » de la fenêtre Configurer Simulation SysML, pour être traités dans la simulation. Chaque type object se développe pour répertorier les objets nommés de ce type et les propriétés de chaque object qui nécessitent une configuration dans la colonne « Valeur ».

De nombreux niveaux de types object , de noms et de propriétés ne nécessitent pas de configuration, de sorte que le champ « Valeur » correspondant n'accepte aucune saisie. Lorsque la saisie est appropriée et acceptée, une flèche déroulante s'affiche à l'extrémité droite du champ ; lorsque vous cliquez sur cette flèche, une courte liste de valeurs possibles s'affiche pour la sélection. Certaines valeurs (telles que « SimVariable » pour une pièce) ajoutent des couches supplémentaires de paramètres et de propriétés, où vous cliquez sur le bouton  pour, à nouveau, sélectionner et définir des valeurs pour les paramètres. Pour les jeux de données, la dialogue de saisie vous permet de saisir ou d'importer des valeurs, telles que des valeurs initiales ou par défaut ; consultez la rubrique d'aide *Analyse de Modèle utilisant Ensemble de Données* .

Type d'élément	Comportement
Type de valeur	Les éléments ValueType sont généralisés à partir d'un type primitif ou sont substitués par SysMLSimReal pour la simulation.
Bloc	Les éléments Bloc mappés aux éléments SysMLSimClass ou SysMLSimModel support la création d'ensembles de données. Si vous avez défini plusieurs ensembles de données dans une SysMLSimClass (qui peuvent être généralisés), vous devez identifier l'un d'entre eux comme étant l'ensemble de données par défaut (à l'aide de l'option de menu contextuel « Définir comme ensemble de données par défaut »). Comme un SysMLSimModel est un élément de niveau supérieur possible pour une simulation et ne sera pas généralisé, si vous avez défini plusieurs ensembles de données, l'ensemble de données à utiliser est choisi pendant la simulation.
Propriétés	La méthode préférée pour spécifier des constantes ou des variables et leurs paramètres consiste à utiliser les stéréotypes SysPhS PhSConstant et PhSVariable sur les Propriétés elles-mêmes. Le stéréotype PhSVariable possède des propriétés

	intégrées pour *isContinuous* , *isConserved* et *changeCycle* . Les Propriétés seront répertoriées sous PhSConstant ou PhSVariable et la valeur ne peut pas être modifiée. Il est également possible de définir les paramètres dans la fenêtre Configurer Simulation SysML. Dans ce cas, ils seront répertoriés sous « Propriétés ». Propriétés d'un Bloc peuvent être configurées comme des SimConstants ou des SimVariables. Pour une SimVariable, vous configurez ces attributs : • isContinuous — détermine si la valeur de la propriété varie en continu (« true », la valeur par défaut) ou discrètement (« false ») • isConserved — détermine si les valeurs de la propriété sont conservées ('true') ou non ('false', la valeur par défaut) ; lors de modélisation d'une interaction physique, les interactions incluent des échanges de substances physiques conservées telles que le courant électrique, la force ou l'écoulement d'un fluide • changeCycle — spécifie l'intervalle de temps auquel une valeur de propriété discrète change ; la valeur par défaut est « 0 » - changeCycle peut être défini sur une valeur autre que 0 uniquement lorsque isContinuous = 'faux' - La valeur de changeCycle doit être positive ou égale à 0
Port	Aucune configuration requise.
Fonction Sim	Les fonctions sont créées sous forme d'opérations dans des blocs ou des blocs de contraintes, stéréotypés comme « SimFunction ». Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML.
Généralisation	Aucune configuration requise.
Connecteur de liaison	Lie une propriété à un paramètre d'une propriété de contrainte. Aucune configuration n'est requise ; cependant, si les propriétés sont différentes, le système propose une option pour les synchroniser.
Connecteur	Connecte deux ports. Aucune configuration n'est requise dans la fenêtre Configurer Simulation SysML. Cependant, vous devrez peut-être configurer les propriétés du type de port en déterminant si l'attribut isConserved doit être défini sur « False » (pour les propriétés potentielles, afin que le couplage d'égalité soit établi) ou sur « True » (pour les propriétés de flux/conservées, afin que le couplage somme à zéro soit établi).
Bloc de contrainte	Aucune configuration requise.

Onglet Simulation

Ce tableau décrit les champs de l'onglet « Simulation » de la fenêtre Configurer Simulation SysML.

Champ	Action
Modèle	Cliquez sur la flèche déroulante et sélectionnez le nœud de niveau supérieur (un élément SysMLSimModel) pour la simulation. La liste est renseignée avec les noms des blocs définis comme nœuds de modèle de niveau supérieur.

Ensemble de données	Cliquez sur la flèche déroulante et sélectionnez l'ensemble de données pour le modèle sélectionné.
Pré-valider	Cochez cette case pour valider automatiquement le modèle avant l'exécution de chaque simulation du modèle.
Démarrer	Type le temps d'attente initial avant lequel la simulation est démarrée, en secondes (valeur par défaut est 0).
Arrêt	Type le nombre de secondes pendant lesquelles la simulation s'exécutera.
Format	Cliquez sur la flèche déroulante et sélectionnez « plt », « csv » ou « mat » comme format du fichier de résultat, qui pourrait potentiellement être utilisé par d'autres outils.
Graphique Paramétriques	• Cochez cette case pour tracer la légende A sur l'axe des Y par rapport à la légende B sur l'axe des X. • Décochez la case pour tracer les légendes sur l'axe des Y en fonction du temps sur l'axe des X Note : avec la case à cocher sélectionnée, vous devez sélectionner deux propriétés à tracer.
Utiliser Simscape	(si l'outil mathématique sélectionné est Simulink) Cochez la case si vous souhaitez également traiter la simulation dans Simscape.
Dépendances	Répertorie les types qui doivent être générés pour simuler ce modèle.
Propriétés à construire	Fournit une liste des propriétés des variables impliquées dans la simulation. Cochez la case en regard de chaque propriété à tracer.

Création d'un Modèle Paramétrique

Dans cette rubrique, nous abordons la manière dont vous pouvez développer des éléments de modèle SysML pour la simulation (en supposant que vous possédez déjà des connaissances en matière modélisation SysML), configurer ces éléments dans la fenêtre Configurer Simulation SysML et observer les résultats d'une simulation selon certaines des différentes définitions et approches modélisation . Les points sont illustrés par Instantanés de diagrammes et d'écrans issus des exemples Simulation SysML fournis dans ce chapitre.

Lors de la création d'un Modèle Paramétrique , vous pouvez appliquer l'une des trois approches pour définir les équations de contrainte :

- Définition d'équations de contrainte en ligne sur un élément Bloc
- Créer des ConstraintBlocks réutilisables et
- Utilisation des propriétés de contrainte connectées

Vous prendrez également en considération :

- Flux dans les interactions physiques
- Valeurs par défaut et valeurs initiales
- Fonctions Simulation
- Répartition de la valeur et
- Paquetages et importations

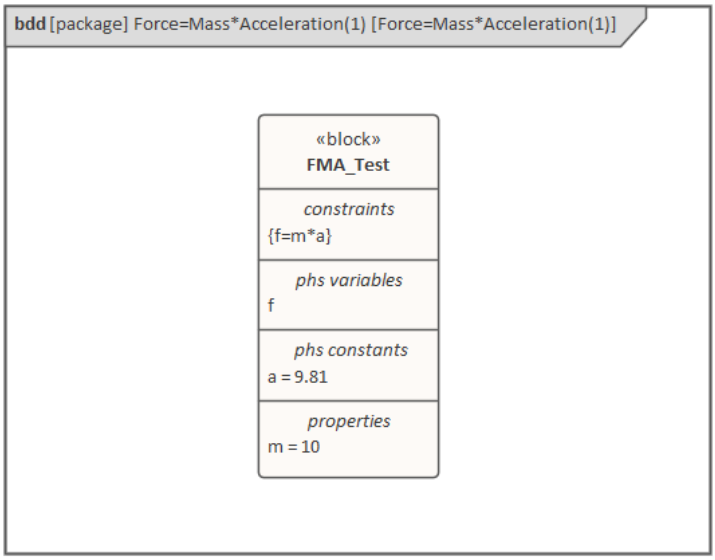
Accéder

Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager

Définition d'équations de contraintes en ligne sur un Bloc

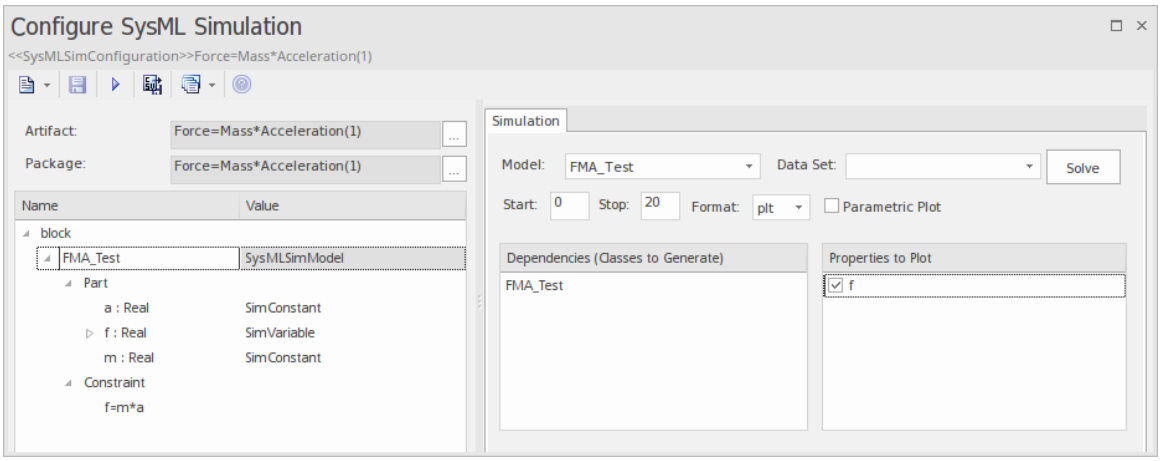
Définir des contraintes directement dans un Bloc est simple et constitue le moyen le plus simple de définir des équations de contraintes.

Dans cette figure, la contrainte 'f = m * a' est définie dans un élément Bloc .



Conseil : Vous pouvez définir plusieurs contraintes dans un même Bloc .

1. Créez un artefact de configuration SysMLSim « Force=Mass*Acceleration(1) » et pointez-le vers le Paquetage « FMA_Test ».
2. Pour « FMA_Test », dans la colonne « Valeur », définissez « SysMLSimModel ».
3. Pour les parties « a », « m » et « f », dans la colonne « Valeur » : définissez « a » et « m » sur « PhSConstant » et (éventuellement) définissez « f » sur « PhSVariable ».
4. Dans l'onglet « Simulation », dans le panneau « Propriétés à tracer », cochez la case en regard de « f ».
5. Cliquez sur le bouton Résoudre pour exécuter la simulation.

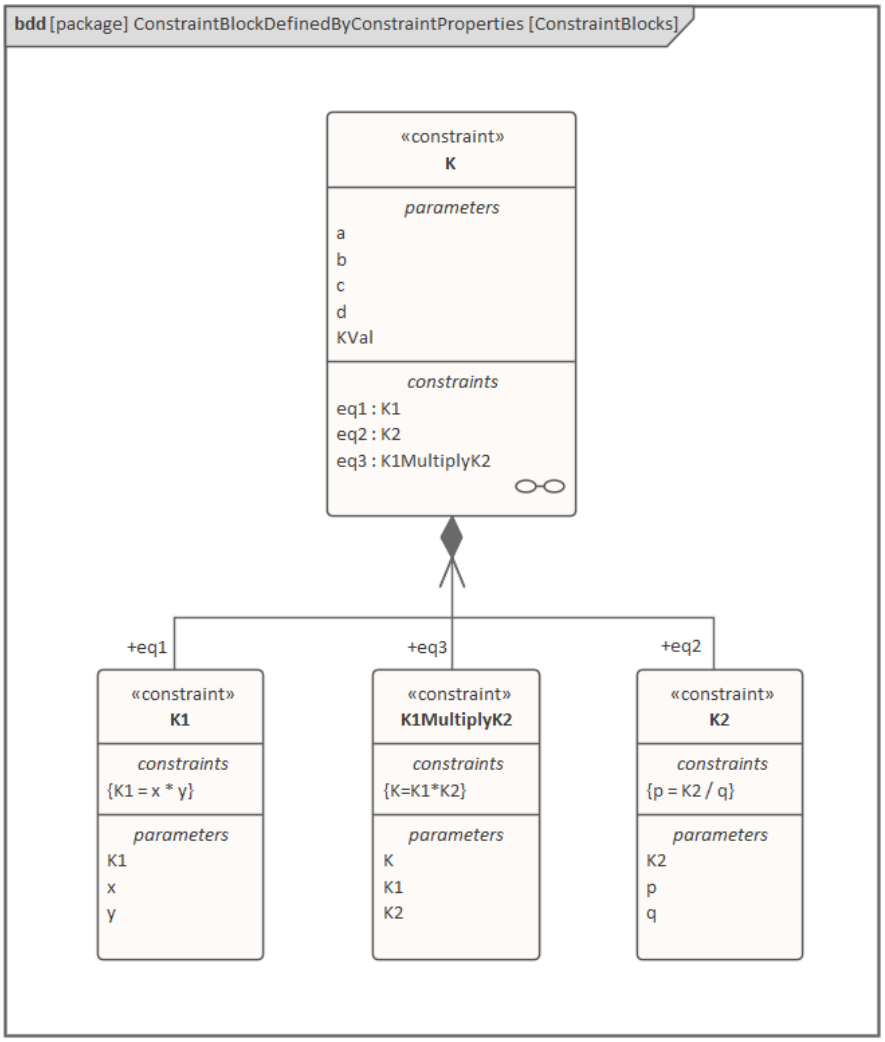


Un graphique doit être tracé avec f = 98,1 (qui vient de 10 * 9,81).

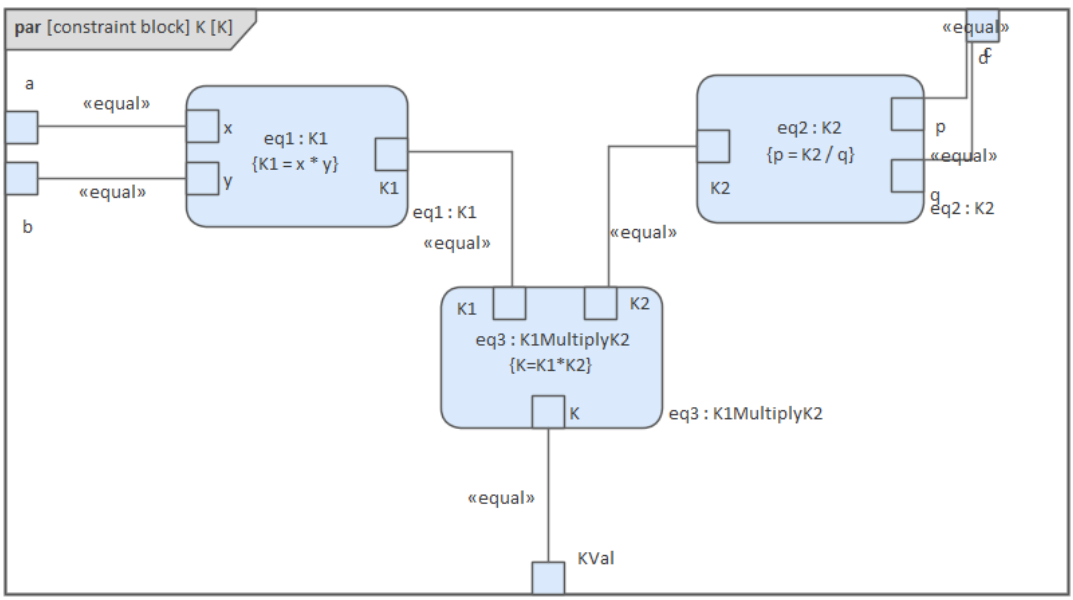
Propriétés de contrainte connectée

Dans SysML, les propriétés de contrainte existant dans ConstraintBlocks peuvent être utilisées pour offrir une plus grande flexibilité dans la définition des contraintes.

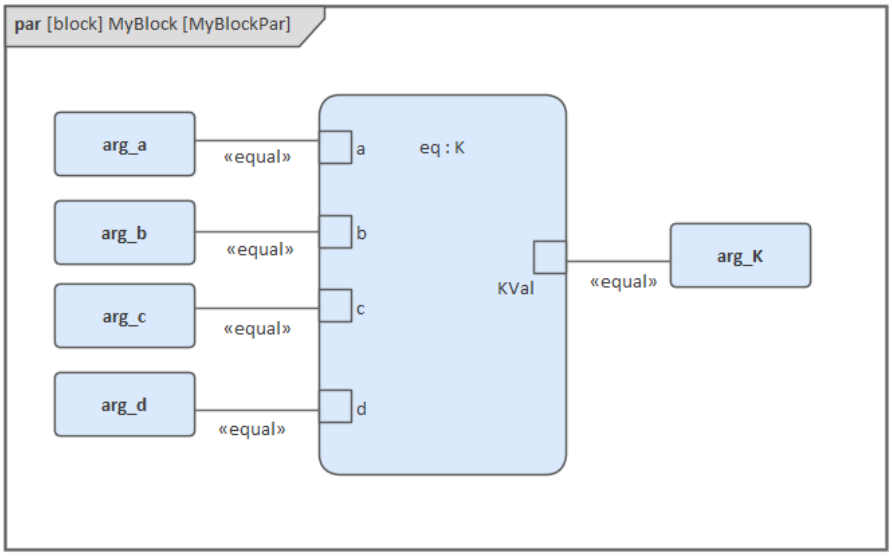
Dans cette figure, ConstraintBlock 'K' définit les paramètres 'a', 'b', 'c', 'd' et 'KVal', ainsi que trois propriétés de contrainte 'eq1', 'eq2' et 'eq3', typées respectivement 'K1', 'K2' et 'K1MultiplyK2'.



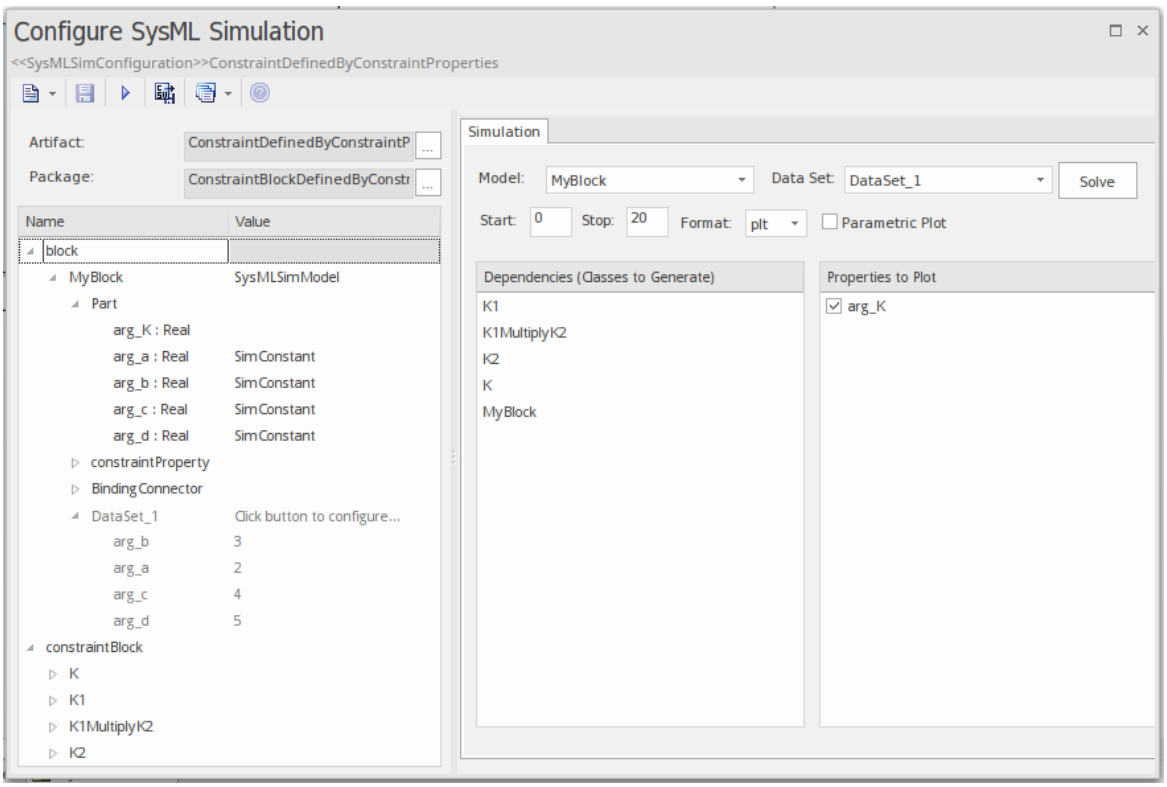
Créez un diagramme Paramétriques dans ConstraintBlock 'K' et connectez les paramètres aux propriétés de contrainte avec des connecteurs de liaison, comme indiqué :



- Créer un modèle MyBlock avec cinq Propriétés (Parties)
- Créez une propriété de contrainte « eq » pour MyBlock et affichez les paramètres
- Lier les propriétés aux paramètres



- Fournir des valeurs (arg_a = 2, arg_b = 3, arg_c = 4, arg_d = 5) dans un ensemble de données
- Dans la dialogue « Configurer Simulation SysML », définissez « Modèle » sur « MyBlock » et « Ensemble de données » sur « DataSet_1 »
- Dans le panneau « Propriétés à tracer », cochez la case en regard de « arg_K »
- Cliquez sur le bouton Résoudre pour exécuter la simulation



Le résultat 120 (calculé comme 2 * 3 * 4 * 5) sera calculé et représenté graphiquement. C'est la même chose que lorsque

nous faisons un développement avec un stylo et du papier : K = K1 * K2 = (x*y) * (p*q), puis lions avec les valeurs (2 * 3) * (4 * 5) ; nous obtenons 120.

Ce qui est intéressant ici, c'est que nous définissons intentionnellement l'équation de K2 comme étant « p = K2 / q » et cet exemple fonctionne toujours.

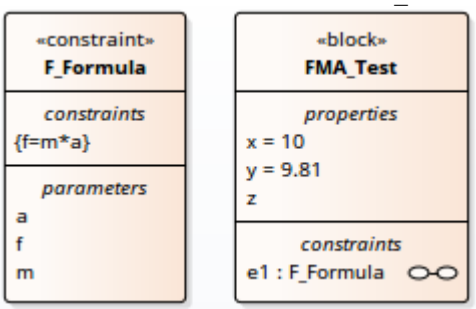
Nous pouvons facilement résoudre K2 comme étant p * q dans cet exemple, mais dans certains exemples complexes, il est extrêmement difficile de résoudre une variable à partir d'une équation ; cependant, Enterprise Architect SysMLSim peut toujours y parvenir.

En résumé, l'exemple vous montre comment définir un ConstraintBlock avec une plus grande flexibilité en construisant les propriétés de contrainte. Bien que nous n'ayons démontré qu'une seule couche dans le ConstraintBlock, ce mécanisme fonctionnera sur des modèles complexes pour un niveau d'utilisation arbitraire.

Créer des blocs de contraintes réutilisables

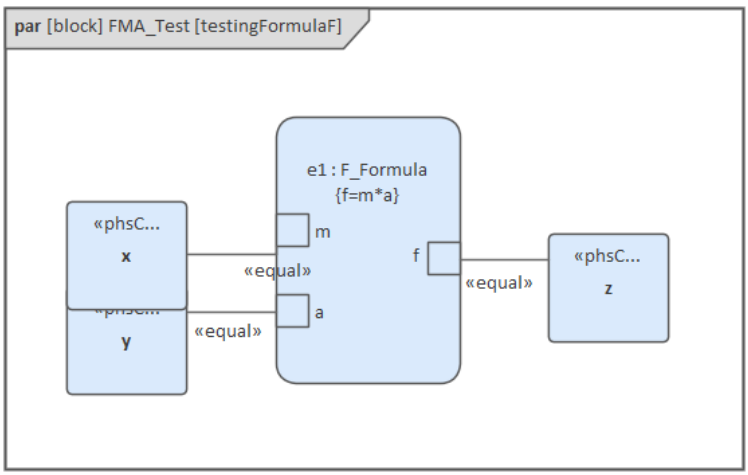
Si une équation est couramment utilisée dans plusieurs blocs, un bloc de contrainte peut être créé pour être utilisé comme propriété de contrainte dans chaque Bloc . Voici les modifications que nous apportons, sur la base de l'exemple précédent :

- Créez un élément ConstraintBlock 'F_Formula' avec trois paramètres 'a', 'm' et 'f', et une contrainte 'f = m * a'



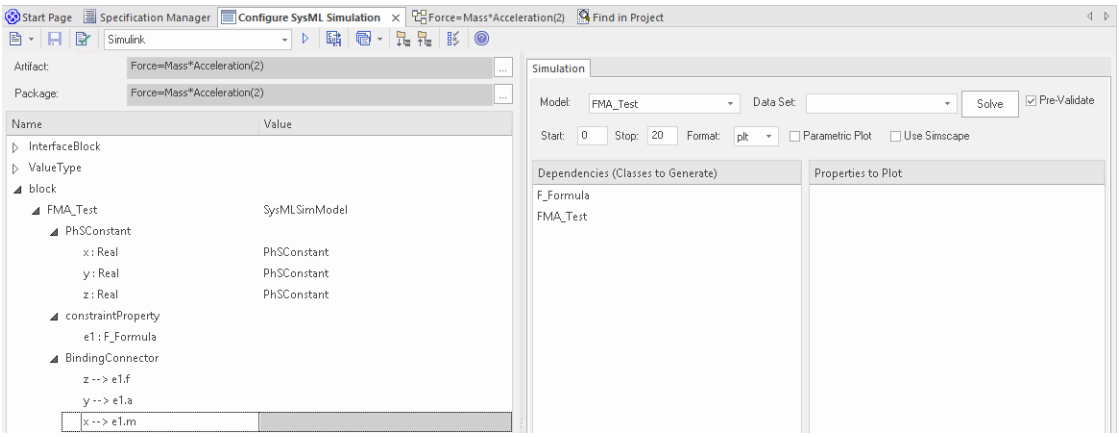
Conseil : Le type primitif 'Real' sera appliqué si les types de propriété sont vides

- Créez un Bloc « FMA_Test » avec trois propriétés « x », « y » et « z », et donnez à « x » et « y » les valeurs par défaut « 10 » et « 9,81 » respectivement
- Créer un diagramme Paramétriques dans 'FMA_Test', montrant les propriétés 'x', 'y' et 'z'
- Créez une ConstraintProperty « e1 » typée « F_Formula » et affichez les paramètres
- Dessinez les connecteurs de liaison entre « x—m », « y—a » et « f—z » comme indiqué :



- Créez un élément d'artefact SysMLSimConfiguration et configurez-le comme indiqué dans le dialogue :
  - Dans la colonne « Valeur », définissez « FMA_Test » sur « SysMLSimModel »
  - Dans la colonne « Valeur », définissez « x » et « y » sur « PhSConstant »
  - Dans le panneau « Propriétés à tracer », cochez la case en regard de « Z »

- Cliquez sur le bouton Résoudre pour exécuter la simulation



Un graphique doit être tracé avec f = 98,1 (qui vient de 10 * 9,81).

Flux dans les interactions physiques

Lors de modélisation de l'interaction physique, les échanges de substances physiques conservées telles que le courant électrique, la force, le couple et le débit doivent être modélisés comme des flux, et les variables de flux doivent être définies sur l'attribut « isConserved ».

Deux types différents de couplage sont établis par les connexions, selon que les propriétés d'écoulement sont potentielles (par défaut) ou d'écoulement (conservées) :

- Couplage d'égalité, pour les propriétés potentielles (également appelées efforts)
- Couplage somme à zéro, pour les propriétés de flux (conservées) ; par exemple, selon la loi de courant de Kirchoff dans le domaine électrique, la conservation de la charge fait que tous les flux de charge en un point sont somme à zéro

Dans le code OpenModelica généré de l'exemple « ElectricalCircuit » :

connecteur ChargePort

flux actuel i ; // le mot-clé flow sera généré si 'isConserved' = true

Tension v;

fin ChargePort;

modèle de circuit

Source source;

Résistance résistance;

Sol sol;

équation

connect(source.p, résistance.n);

connect(terre.p, source.n);

connect(résistance.p, source.n);

fin du circuit;

Chaque équation de connexion est en fait étendue à deux équations (il existe deux propriétés définies dans ChargePort), l'une pour le couplage d'égalité, l'autre pour le couplage somme à zéro :

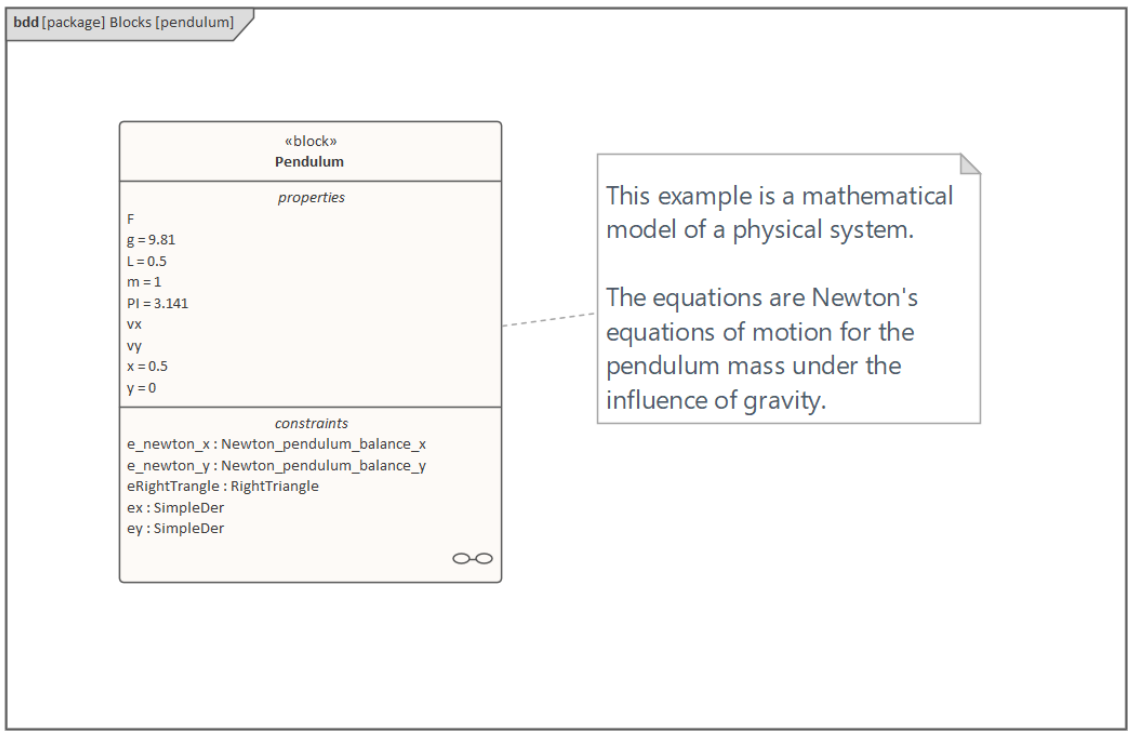
source.pv = résistance.nv;

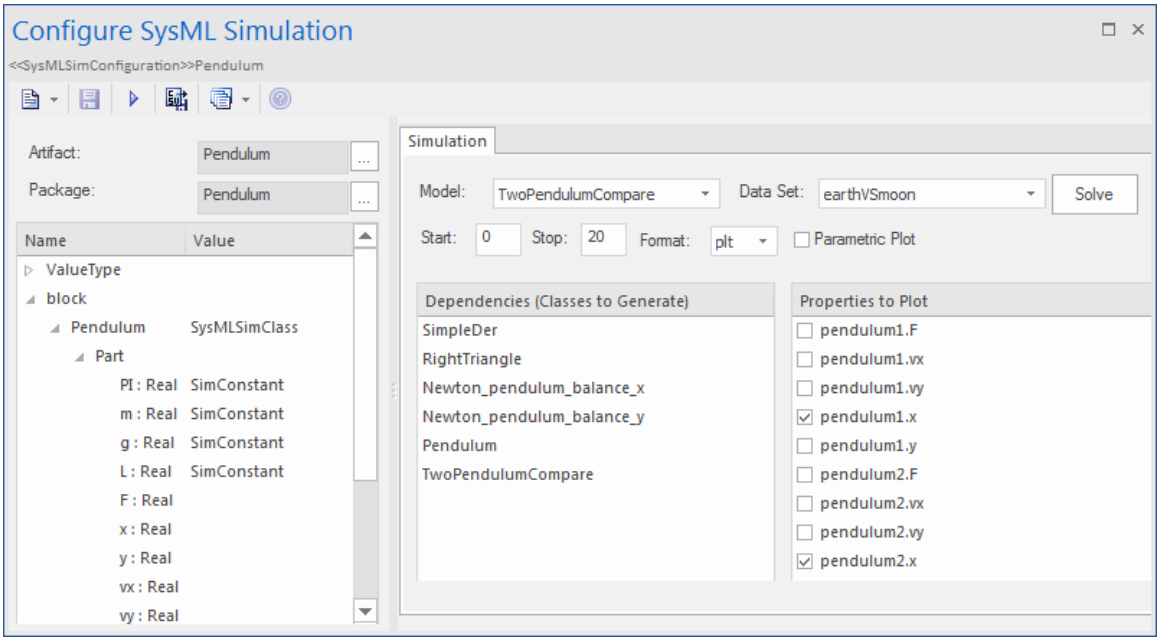
source.pi + résistance.ni = 0;

Valeur par défaut et valeurs initiales

Si des valeurs initiales sont définies dans les éléments de propriété SysML (dialogue « Propriétés » > page « Propriété » > champ « Initiale »), elles peuvent être chargées comme valeur par défaut pour un PhSConstant ou comme valeur initiale pour un PhSVariable.

Dans cet exemple de pendule, nous avons fourni des valeurs initiales pour les propriétés « g », « L », « m », « PI », « x » et « y », comme indiqué sur le côté gauche de la figure. Étant donné que « PI » (la constante mathématique), « m » (la masse du pendule), « g » (le facteur de gravité) et « L » (la longueur du pendule) ne changent pas pendant la simulation, définissez-les comme « PhSConstant ».





Le code Modelica généré ressemble à ceci :

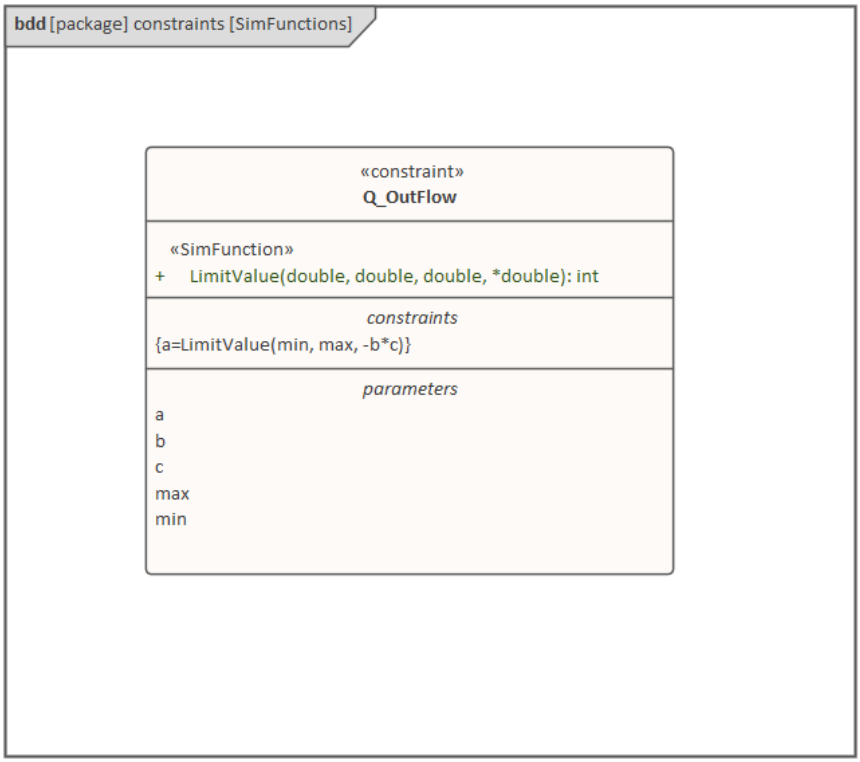
classe Pendule
paramètre PI réel = **3,141 ;**
paramètre Réel m = **1** ;
paramètre réel g = **9,81** ;
paramètre Réel L = **0,5** ;
Réel F;
Réel x **(début=0,5)** ;
Réel y **(début=0)** ;
VX réel;
Vrai vy;
.....
équation
.....
fin du pendule;

- Propriétés 'PI', 'm', 'g' et 'L' sont constantes et sont générées sous forme d'équation de déclaration
- Propriétés 'x' et 'y' sont variables ; leurs valeurs de départ sont respectivement 0,5 et 0, et les valeurs initiales sont générées sous forme de modifications

Fonctions Simulation

Une fonction Simulation est un outil utile pour écrire une logique complexe et est facile à utiliser pour les contraintes. Cette section décrit une fonction de l'exemple TankPI.

Dans le ConstraintBlock 'Q_OutFlow', une fonction 'LimitValue' est définie et utilisée dans la contrainte.



- Sur un Bloc ou un ConstraintBlock, créez une opération ('LimitValue' dans cet exemple) et ouvrez l'onglet 'Opérations' de la fenêtre Fonctionnalités
- Donnez à l'opération le stéréotype « SimFunction »
- Définissez les paramètres et définissez la direction sur « entrée/sortie »

Conseils : Plusieurs paramètres peuvent être définis comme 'out', et l'appelant récupère la valeur au format :

(out1, out2, out3) = nom_fonction(in1, in2, in3, in4, ...); //Forme d'équation

(out1, out2, out3) := function_name(in1, in2, in3, in4, ...); //Formulaire de déclaration

- Définissez le corps de la fonction dans le champ de texte de l'onglet « Code » de la fenêtre Propriétés , comme indiqué :

  pLim :=
  si p > pMax alors
  pMax
  sinon si p < pMin alors
  pMin
  autre
  p;

Lors de la génération de code, Enterprise Architect collecte toutes les opérations stéréotypées comme « SimFunction » définies dans ConstraintBlocks et Blocks, puis génère un code ressemblant à ceci :

fonction LimitValue

entrée Réel pMin;

entrée pMax réel ;

entrée Réel p;

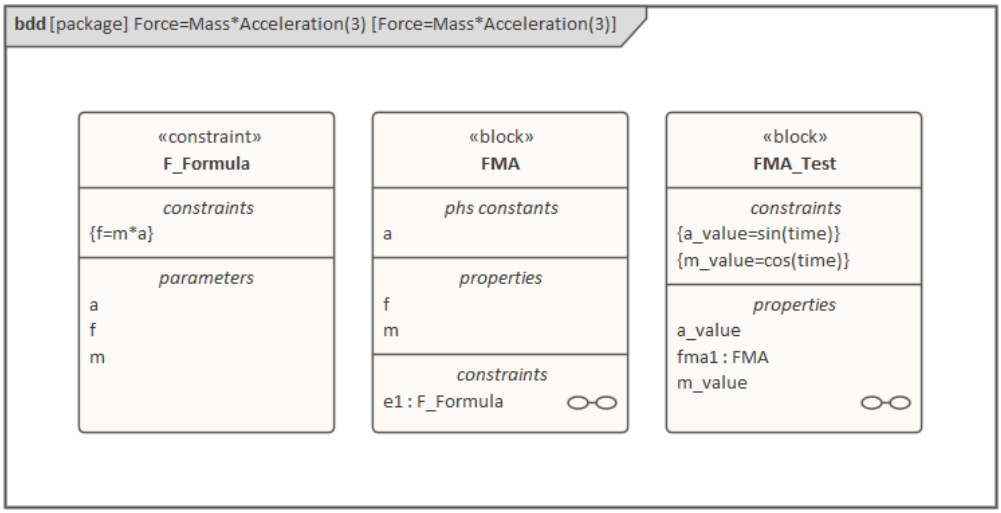
sortie Real pLim;

algorithme

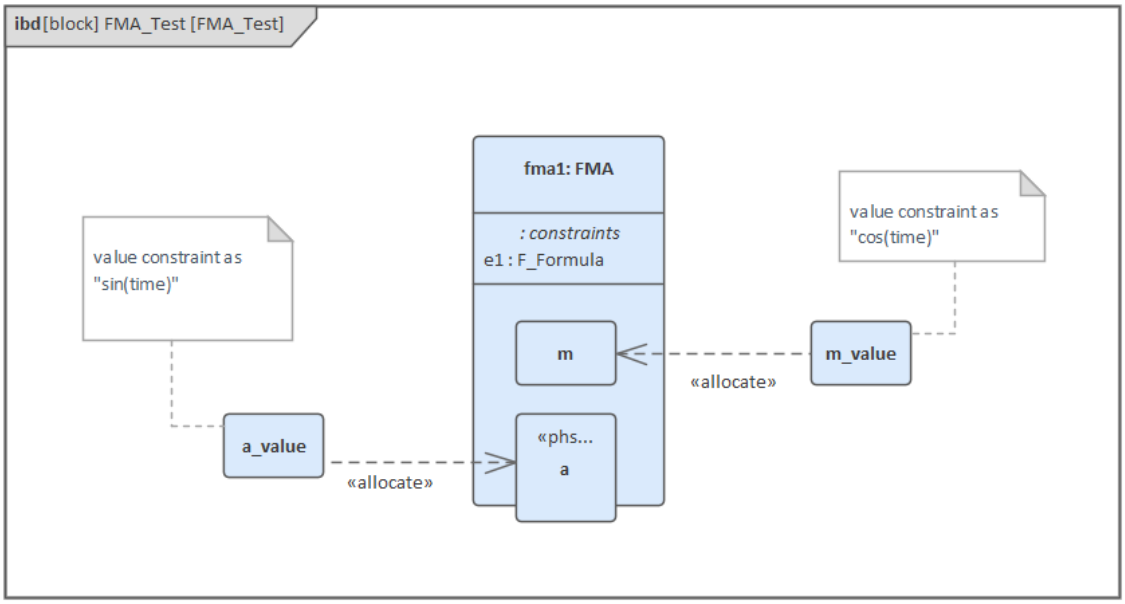
pLim :=
si p > pMax alors
pMax
sinon si p < pMin alors
pMin
autre
p;
fin de LimitValue;

Répartition de la valeur

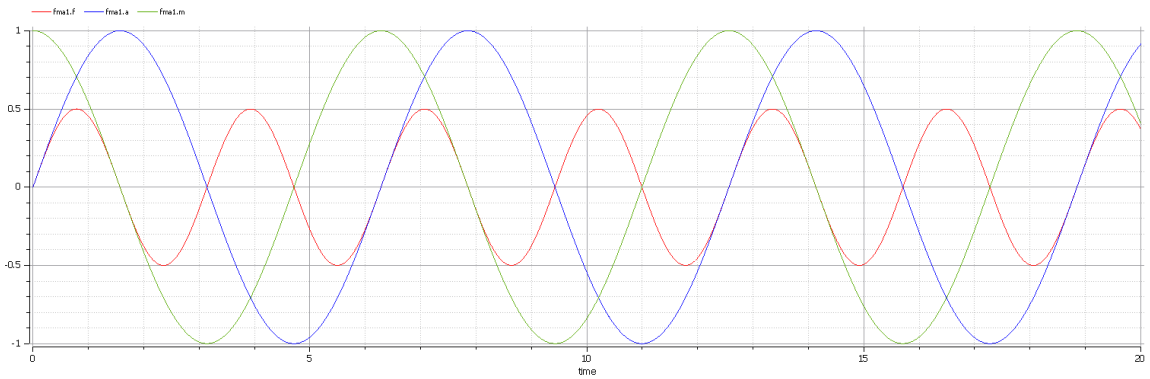
Cette figure montre un modèle simple appelé « Force = Masse * Accélération ».



- Un Bloc « FMA » est modélisé avec les propriétés « a », « f » et « m » et une constraintProperty « e1 », typée sur Bloc de contrainte « F_Formula »
- Le Bloc 'FMA' n'a aucune valeur initiale définie sur ses propriétés, et les propriétés 'a', 'f' et 'm' sont toutes variables, donc leur changement valeur dépend de l'environnement dans lequel elles sont simulées
- Créez un Bloc 'FMA_Test' en tant que SysMLSimModel et ajoutez la propriété 'fma1' pour tester le comportement du Bloc 'FMA'
- Contrainte 'a_value' à être ' sin (time)'
- Contrainte 'm_value' à être ' cos (temps)'
- Dessinez des connecteurs d'allocation pour allouer des valeurs de l'environnement au modèle « FMA »



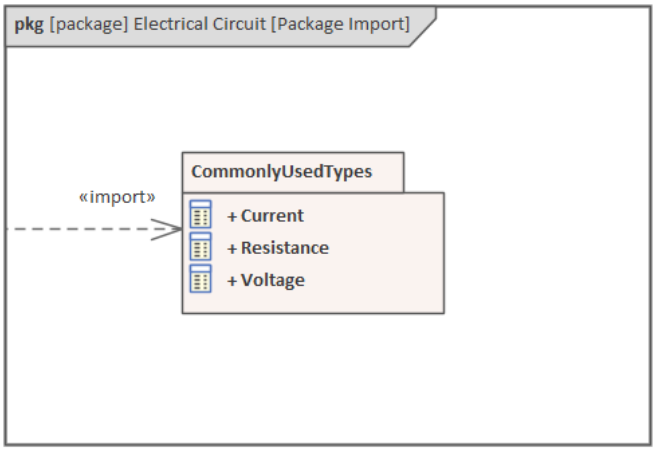
- Cochez les cases « Propriétés à tracer » pour « fma1.a », « fma1.m » et « fma1.f »
- Cliquez sur le bouton Résoudre pour simuler le modèle



Paquetages et importations

L'artefact SysMLSimConfiguration collecte les éléments (tels que les blocs, les blocs de contrainte et les types de valeur) d'un Paquetage . Si la simulation dépend d'éléments qui ne sont pas détenus par ce Paquetage , tels que des bibliothèques réutilisables, Enterprise Architect fournit un connecteur d'importation entre les éléments Paquetage pour répondre à cette exigence.

Dans l'exemple de circuit électrique, l'artefact est configuré sur le Paquetage « ElectricalCircuit », qui contient presque tous les éléments nécessaires à la simulation. Cependant, certaines propriétés sont typées sur des types valeur tels que « Tension », « Courant » et « Résistance », qui sont couramment utilisés dans plusieurs modèles SysML et sont donc placés dans un Paquetage appelé « CommonlyUsedTypes » en dehors des modèles SysML individuels. Si vous importez ce Paquetage à l'aide d'un connecteur d'importation, tous les éléments du Paquetage importé apparaîtront dans le gestionnaire de configuration SysMLSim.


pkg [package] Electrical Circuit [Package Import]
«import»
CommonlyUsedTypes
+ Current
+ Resistance
+ Voltage

Analyse de Modèle utilisant Ensemble de Données

Chaque Bloc SysML utilisé dans un modèle Paramétriques peut, dans la configuration Simulation , avoir plusieurs jeux de données définis par rapport à lui. Cela permet des variations de simulation reproductibles en utilisant le même modèle SysML.

Un Bloc peut être typé en tant que SysMLSimModel (un nœud de niveau supérieur qui ne peut pas être généralisé ou faire partie d'une composition) ou en tant que SysMLSimClass (un élément de niveau inférieur qui peut être généralisé ou faire partie d'une composition). Lorsque vous exécutez une simulation sur un élément SysMLSimModel, si vous avez défini plusieurs jeux de données, vous pouvez spécifier le jeu de données à utiliser. Cependant, si une SysMLSimClass dans la simulation comporte plusieurs jeux de données, vous ne pouvez pas sélectionner celui à utiliser pendant la simulation et devez donc identifier un jeu de données comme jeu de données par défaut pour cette classe.

Accéder

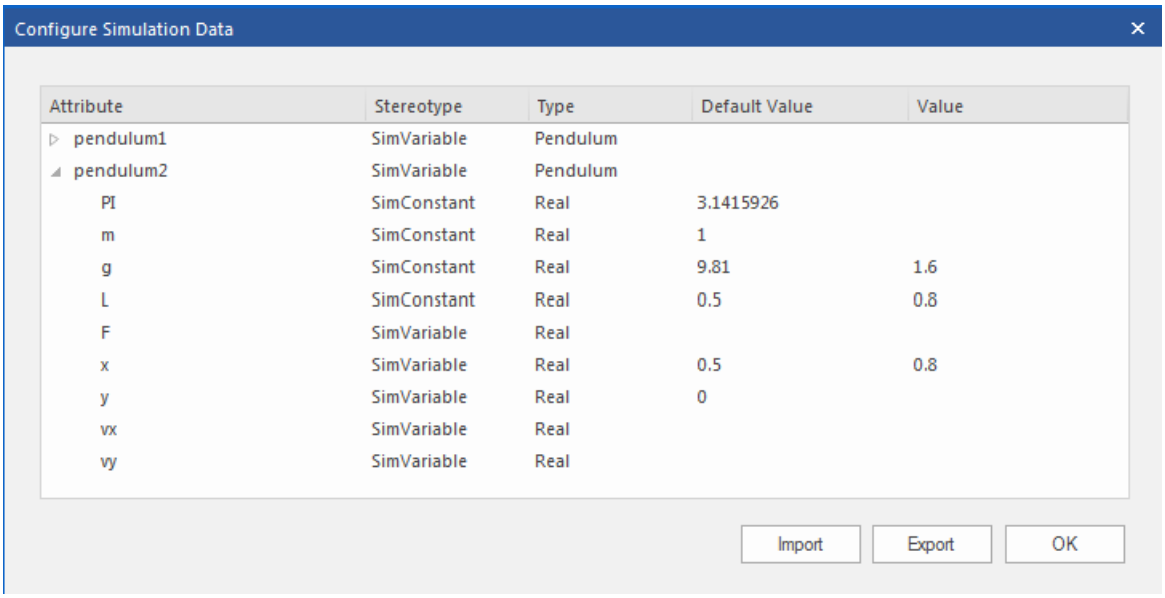
Ruban	Simuler > Comportement du Système > Modelica/Simulink > Gestionnaire de configuration SysMLSim > dans le groupe « bloc » > Colonne Nom > Menu contextuel sur l'élément bloc > Créer un jeu de données Simulation

Gestion des jeux de données

Tâche	Action
Créer	Pour créer un nouveau jeu de données, cliquez-droit sur le nom d'un Bloc et sélectionnez l'option « Créer un jeu de données Simulation ». Le jeu de données est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour configurer le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Double	Pour dupliquer un jeu de données existant comme base pour la création d'un nouveau jeu de données, cliquez-droit sur le nom du jeu de données et sélectionnez l'option « Dupliquer ». Le jeu de données dupliqué est ajouté à la fin de la liste des composants sous le nom Bloc . Cliquez sur le bouton  pour modifier le jeu de données dans la dialogue « Configurer les données Simulation » (voir le tableau *Configurer les données Simulation*).
Supprimer	Pour supprimer un jeu de données qui n'est plus nécessaire, cliquez-droit sur le jeu de données et sélectionnez l'option « Supprimer le jeu de données ».
Définir par défaut	Pour définir le jeu de données par défaut utilisé par une SysMLSimClass lorsqu'elle est utilisée comme type de propriété ou héritée (et lorsqu'il existe plusieurs jeux de données), cliquez-droit sur le jeu de données et sélectionnez l'option « Définir par défaut ». Le nom du jeu de données par défaut est mis en surbrillance en gras. Les propriétés utilisées par un modèle utiliseront cette configuration par défaut, sauf si le modèle les remplace explicitement.

Configurer les données Simulation

Cette dialogue est principalement destinée à l'information. La seule colonne dans laquelle vous pouvez directement ajouter ou modifier des données est la colonne « Valeur ».



Colonne	Description
Attribut	La colonne « Attribut » fournit une arborescence de toutes les propriétés du Bloc en cours d'édition.
Stéréotype	La colonne « Stéréotype » identifie, pour chaque propriété, si elle a été configurée pour être une constante pendant toute la durée de la simulation ou une variable dont la valeur est censée changer au fil du temps.
Type	La colonne « Type » décrit le type utilisé pour la simulation de cette propriété. Il peut s'agir soit d'un type primitif (tel que « Réel »), soit d'une référence à un Bloc contenu dans le modèle. Propriétés référençant des Blocs afficheront les propriétés enfants spécifiées par le Bloc référencé en dessous d'elles.
Valeur par défaut	La colonne « Valeur par défaut » indique la valeur qui sera utilisée dans la simulation si aucune substitution n'est fournie. Cela peut provenir du champ « Valeur initiale » dans le modèle SysML ou du jeu de données par défaut du type parent.
Valeur	La colonne « Valeur » vous permet de remplacer la valeur par défaut pour chaque valeur primitive.
Exporter / Importer	Cliquez sur ces boutons pour modifier les valeurs de l'ensemble de données actuel à l'aide d'une application externe telle qu'une feuille de calcul, puis les réimporter dans la liste.

Exemples Simulation SysML

Cette section fournit un exemple concret pour chacune de ces étapes : création d'un modèle SysML pour un domaine, simulation de celui-ci et évaluation des résultats de la simulation. Les exemples appliquent les informations abordées dans les rubriques précédentes.

Exemples

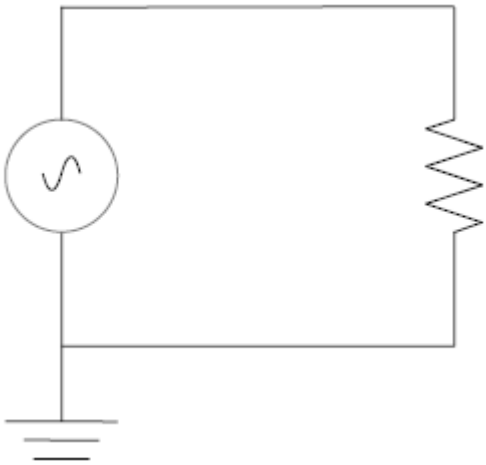
Modèle	Description
Exemple Simulation de circuit électrique	Le premier exemple concerne la simulation du chargement d'un circuit électrique. L'exemple part d'un diagramme de circuit électrique et le convertit en un modèle paramétrique. Le modèle est ensuite simulé et la tension aux bornes source et cible d'une résistance est évaluée et comparée aux valeurs attendues.
Exemple Simulation d'oscillateur masse-ressort-amortisseur	Le deuxième exemple utilise un modèle physique simple pour démontrer le comportement oscillatoire d’un système masse-ressort-amortisseur.
Régulateur de pression du réservoir d'eau	Le dernier exemple montre les niveaux d'eau de deux réservoirs où l'eau est distribuée entre eux. Nous simulons d'abord un système bien équilibré, puis nous simulons un système où l'eau débordera du deuxième réservoir.

Exemple Simulation de circuit électrique

Pour cet exemple, nous parcourons la création d'un modèle SysML Paramétriques pour un circuit électrique simple, puis utilisons une simulation paramétrique pour prédire et cartographier le comportement de ce circuit.

Diagramme de circuit

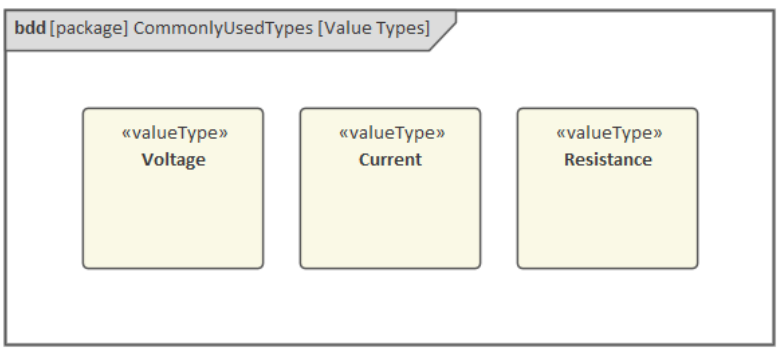
Le circuit électrique que nous allons modéliser, montré ici, utilise une notation de circuit électrique standard.

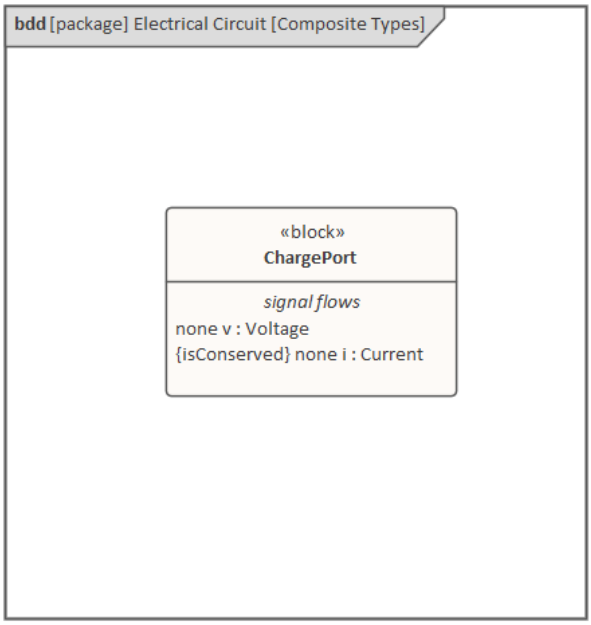


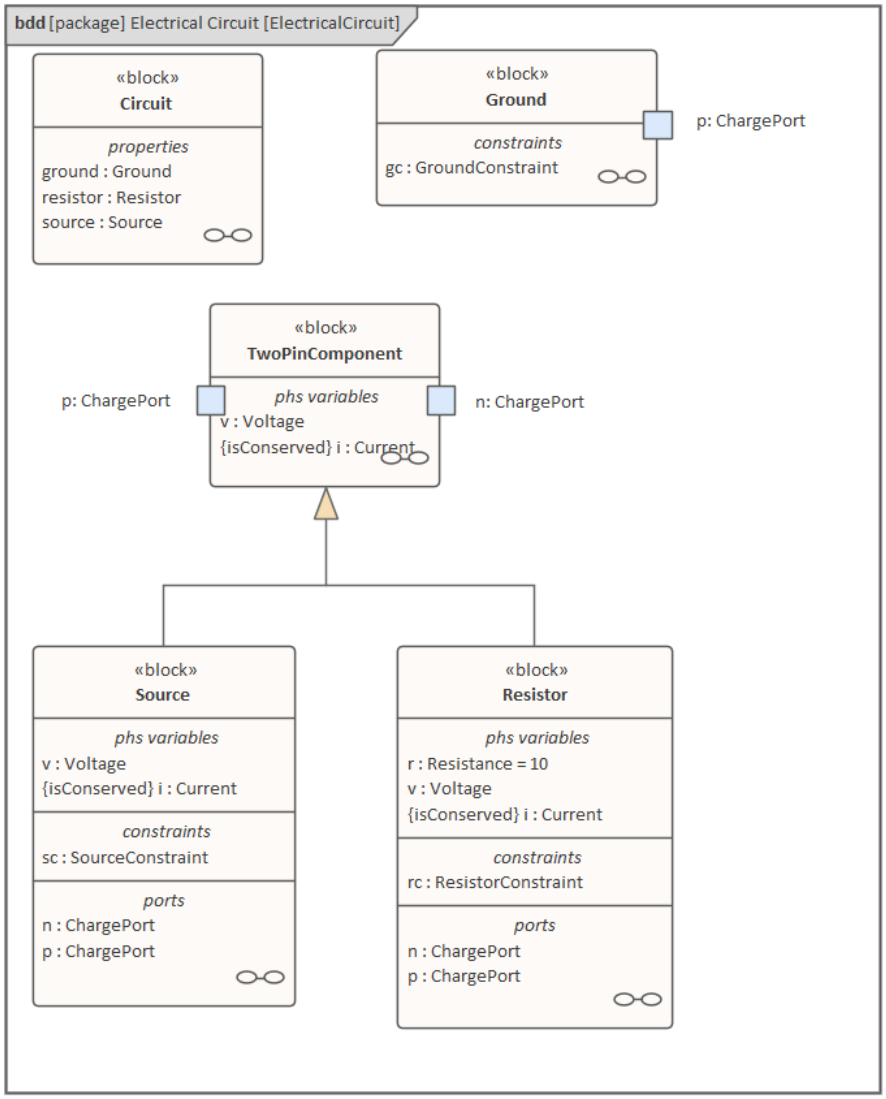
Le circuit comprend une source d'alimentation CA, une terre et une résistance, reliées entre elles par un fil électrique.

Créer Modèle SysML

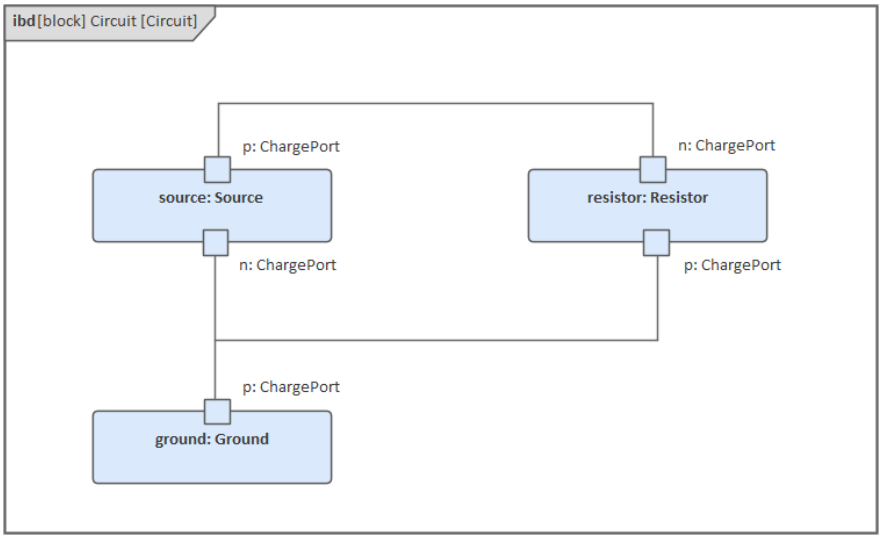
Ce tableau montre comment nous pouvons créer un modèle SysML complet pour représenter le circuit, en commençant par les types de niveau le plus bas et en construisant le modèle une étape à la fois.

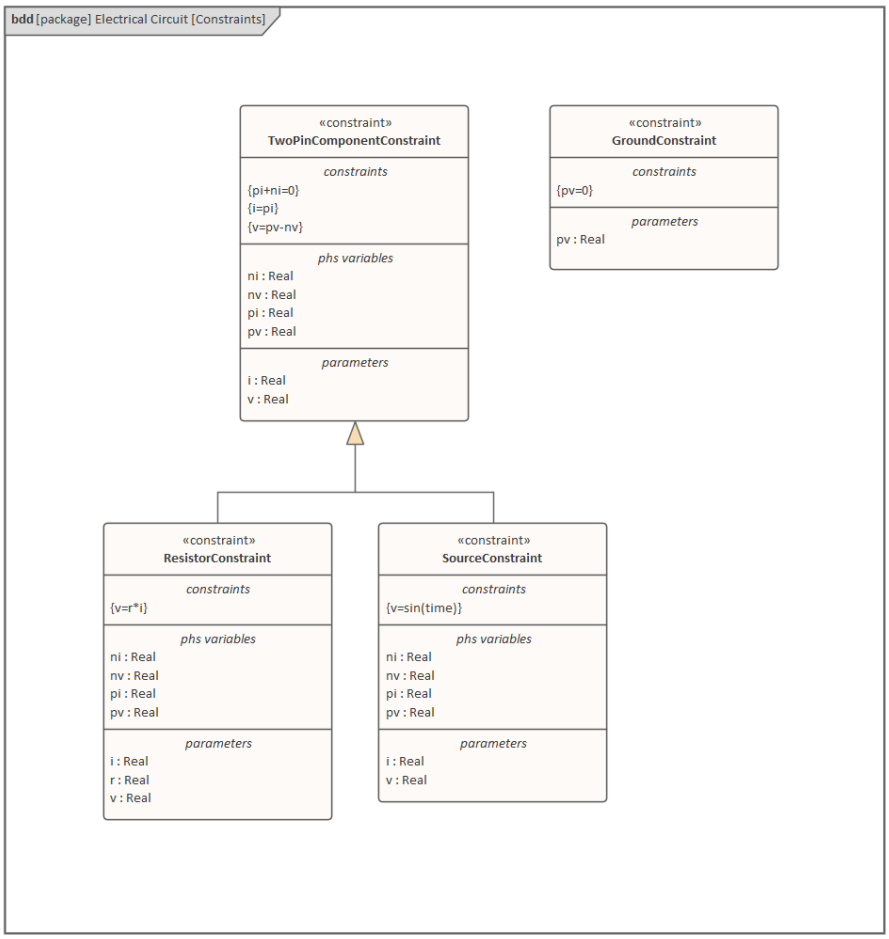
Composant	Action
Types	Définissez les types de valeur pour la tension, le courant et la résistance. Le type d'unité et de quantité n'est pas important pour les besoins de la simulation, mais il serait défini lors de la définition d'un modèle SysML complet. Ces types seront généralisés à partir du type primitif « Réel ». Dans d'autres modèles, vous pouvez choisir de mapper un Type de valeur à un type de simulation correspondant distinct du modèle.  De plus, définissez un type composite (Bloc) appelé ChargePort, qui inclut des propriétés pour le courant et la tension. Ce type nous permet de représenter l'énergie

	électrique au niveau des connecteurs entre les composants. 
Blocs	Dans SysML, le circuit et chacun des composants seront représentés sous forme de blocs. Dans un Interrompre lorsqu'une Variable Change de Valeur (BDD), créez un Circuit Bloc . Le circuit comporte trois parties : une source, une masse et une résistance. Ces parties sont de types différents, avec des comportements différents. Créez un Bloc pour chacun des types de composants. Les trois composants du Bloc de circuit sont connectés via des ports, qui représentent pins électriques. La source et la résistance ont une broche positive et une broche négative. La terre n'a qu'une seule broche, qui est positive. L'électricité (charge électrique) est transmise via les pins . Créez un bloc abstrait « TwoPinComponent » avec deux ports (pins). Les deux ports sont nommés « p » (positif) et « n » (négatif), et ils sont de type ChargePort. Cette figure montre le BDD, avec les blocs Circuit, Ground, TwoPinComponent, Source et Resistance.



Structure interne	Créez un Diagramme Interne de Bloc (IBD) pour Circuit. Ajoutez les propriétés de la Source, de la Résistance et de la Terre, typées par les Blocs correspondants. Connectez les Ports avec des connecteurs. La broche positive de la Source est connectée à la broche négative de la Résistance. La broche positive de la Résistance est connectée à la broche négative de la Source. La Terre est également connectée à la broche négative de la Source.

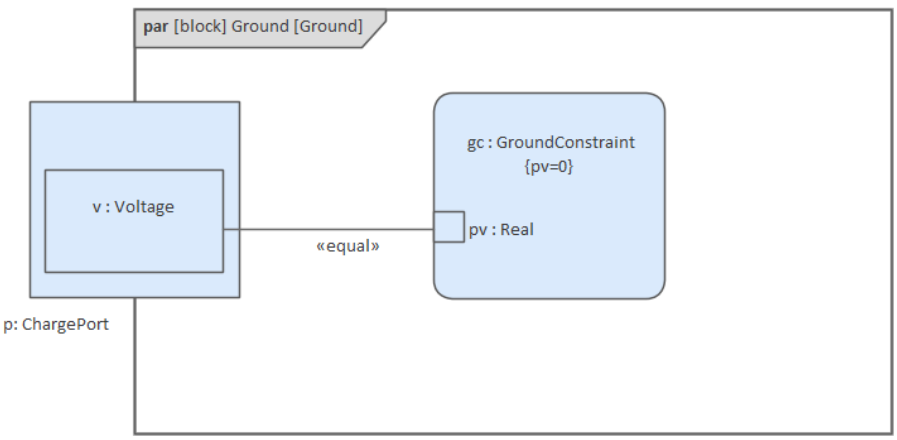
	 Notez que cela suit la même structure que le diagramme de circuit d'origine, mais les symboles de chaque composant ont été remplacés par des propriétés typées par les blocs que nous avons définis.
Contraintes	Les équations définissent des relations mathématiques entre des propriétés numériques. Dans SysML, les équations sont représentées sous forme de contraintes dans des ConstraintBlocks. Les paramètres de ConstraintBlocks correspondent aux PhSVariables et PhSConstants des Blocks ('i', 'v', 'r' dans cet exemple), ainsi qu'aux PhSVariables présents dans le type des Ports ('pv', 'pi', 'nv', 'ni' dans cet exemple). Créez un bloc de contraintes « TwoPinComponentConstraint » pour définir les paramètres et les équations communs aux sources et aux résistances. Les équations doivent indiquer que la tension du composant est égale à la différence entre les tensions aux pins positive et négative. Le courant du composant est égal au courant traversant la broche positive. La somme des courants traversant les deux pins doit être égale à zéro (l'une est la négative de l'autre). La contrainte Ground indique que la tension à la broche Ground est nulle. La contrainte Source définit la tension comme une onde sine avec le temps de simulation du courant comme paramètre. Cette figure montre comment ces contraintes sont rendues dans un BDD.



Fixations

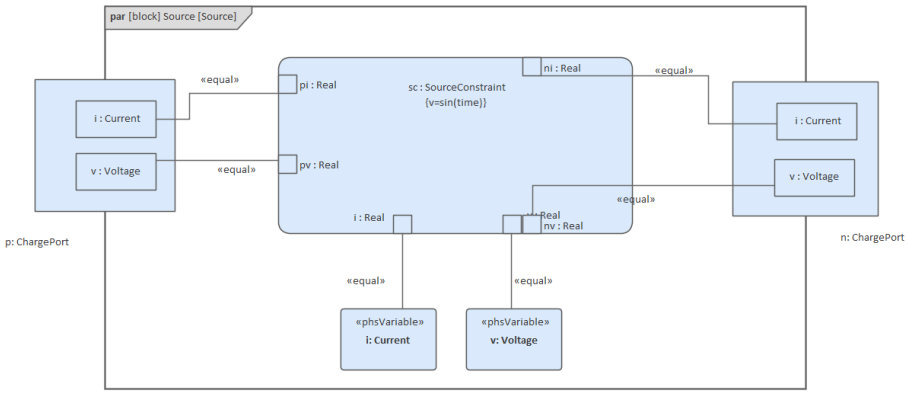
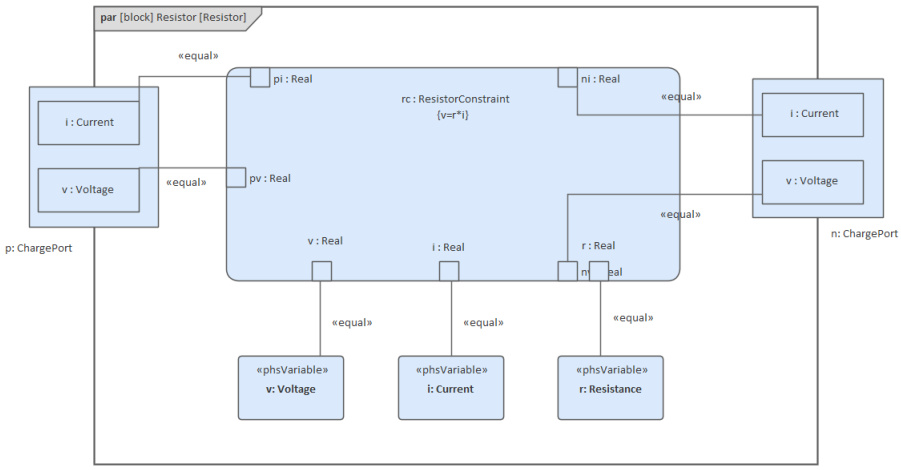
Les valeurs des paramètres de contrainte sont assimilées à des valeurs variables et constantes avec des connecteurs de liaison. Créez des propriétés de contrainte sur chaque Bloc (propriétés typées par ConstraintBlocks) et liez les variables et constantes Bloc aux paramètres de contrainte pour appliquer la contrainte au Bloc . Ces figures montrent les liaisons pour la terre, la source et la résistance respectivement.

Pour la contrainte Ground, liez gc.pv à pv



Pour la contrainte Source, liez :

- sc.pi vers pi
- sc.pv vers pv

	• sc.v à v • sc.i à i • sc.ni à ni et • sc.nv à nv  Pour la contrainte de résistance, liez : • rc.pi vers pi • rc.pv vers pv • rc.v à v • rc.i à i • rc.ni à ni • rc.nv à nv et • rc.r à r 

Configurer le comportement Simulation

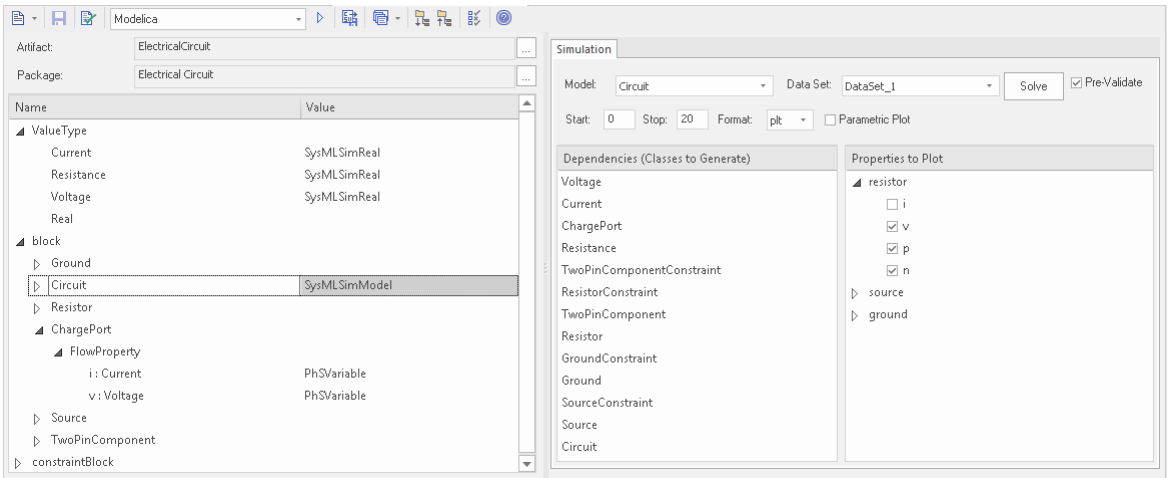
Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

Étape	Action
Artefact de configuration SysMLSim	• Sélectionnez 'Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager'.

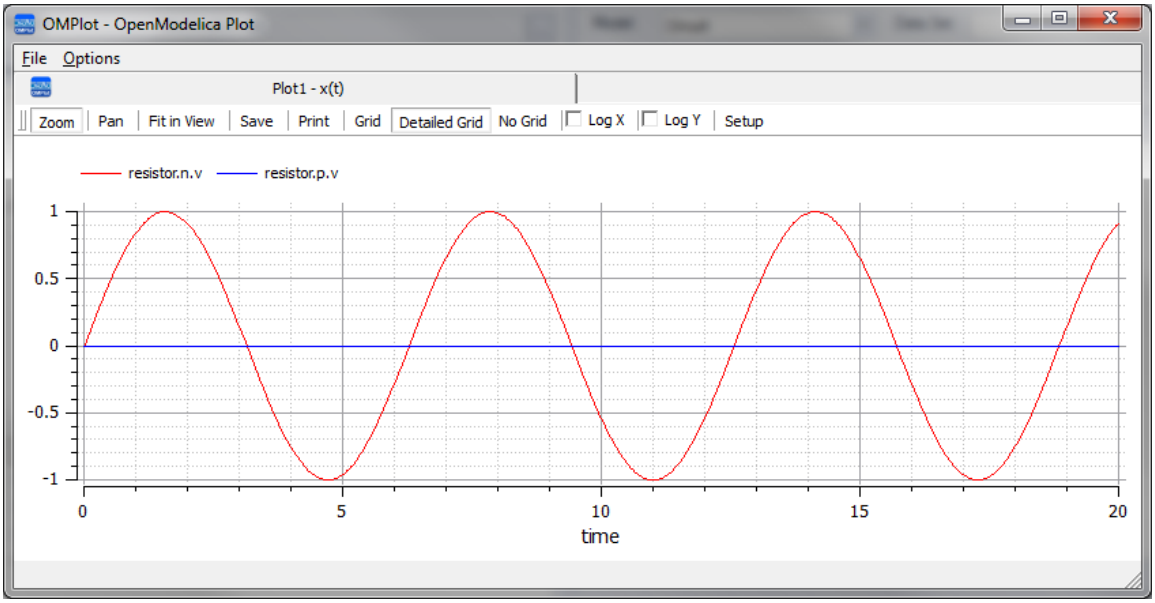
	• Dans la liste déroulante de la première icône de la barre d'outils, sélectionnez « Créer un artefact » et créez l'élément artefact • Sélectionnez le Paquetage qui possède ce modèle SysML
Créer des éléments racines dans Configuration Manager	• Type de valeur • Bloc • Bloc de contrainte
Substitution de type de valeur	Développez ValueType et pour chacun des paramètres Courant, Résistance et Tension, sélectionnez « SysMLSimReal » dans la zone de liste déroulante « Valeur ».
Définir la propriété comme flux	• Développez « block » dans ChargePort | FlowProperty | i : Current et sélectionnez « SimVariable » dans la zone de liste déroulante « Value » • Pour « SysMLSimConfiguration », cliquez sur le bouton  pour ouvrir la dialogue « Configurations des éléments » • Réglez « isConserved » sur « True »
Modèle SysMLSim	C'est le modèle que nous voulons simuler : définissez le Bloc 'Circuit' sur 'SysMLSimModel'.

Exécuter Simulation

Dans la page « Simulation », cochez les cases « résistance.n » et « résistance.p » pour le traçage et cliquez sur le bouton Résoudre.



Les deux légendes « resistor.n » et « resistor.p » sont tracées, comme indiqué.

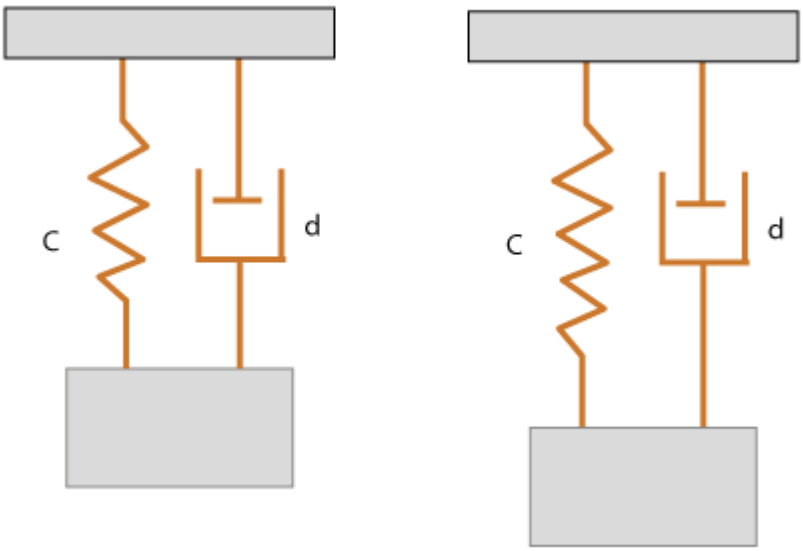

OMPlot - OpenModelica Plot
File Options
Plot1 - x(t)
Zoom Pan Fit in View Save Print Grid Detailed Grid No Grid Log X Log Y Setup
resistor.n.v
resistor.p.v
1
0.5
0
-0.5
-1
0
5
10
15
20
time

Exemple Simulation d'oscillateur masse-ressort-amortisseur

Dans cette section, nous allons parcourir la création d'un modèle paramétrique SysML pour un oscillateur simple composé d'une masse, d'un ressort et d'un amortisseur, puis utiliser une simulation paramétrique pour prédire et tracer le comportement de ce système mécanique. Enfin, nous effectuons une analyse hypothétique en comparant deux oscillateurs fournis avec des valeurs de paramètres différentes via des ensembles de données.

Système en cours de modélisation

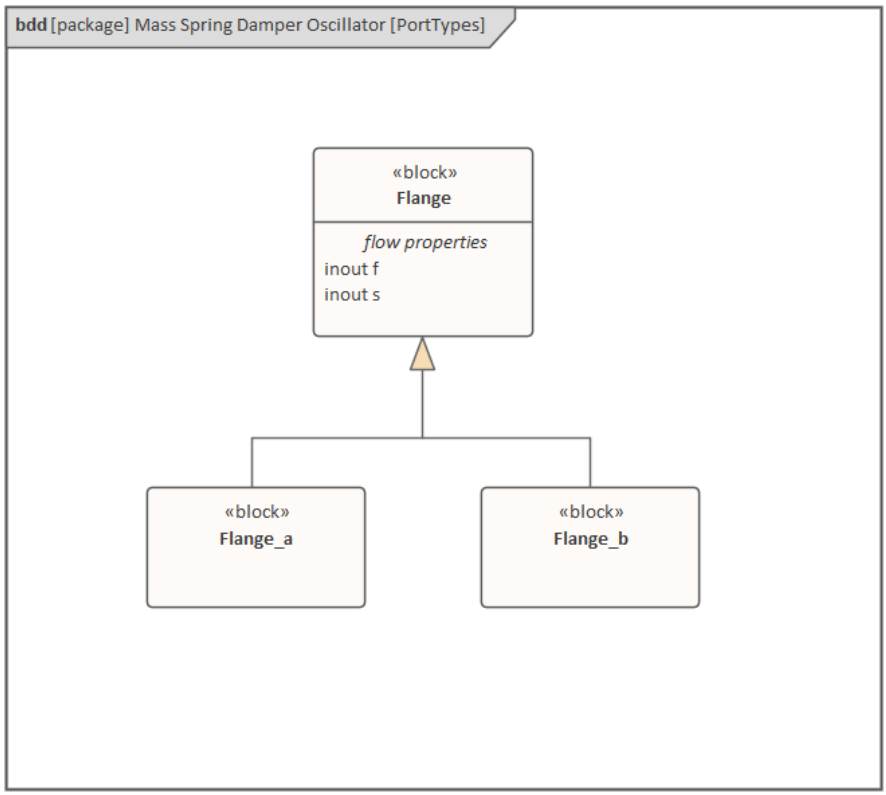
Une masse est suspendue à un ressort et à un amortisseur. Le premier état représenté ici représente le point initial à l'instant = 0, juste au moment où la masse est relâchée. Le deuxième état représente le point final lorsque le corps est au repos et que les forces du ressort sont en équilibre avec la gravité.



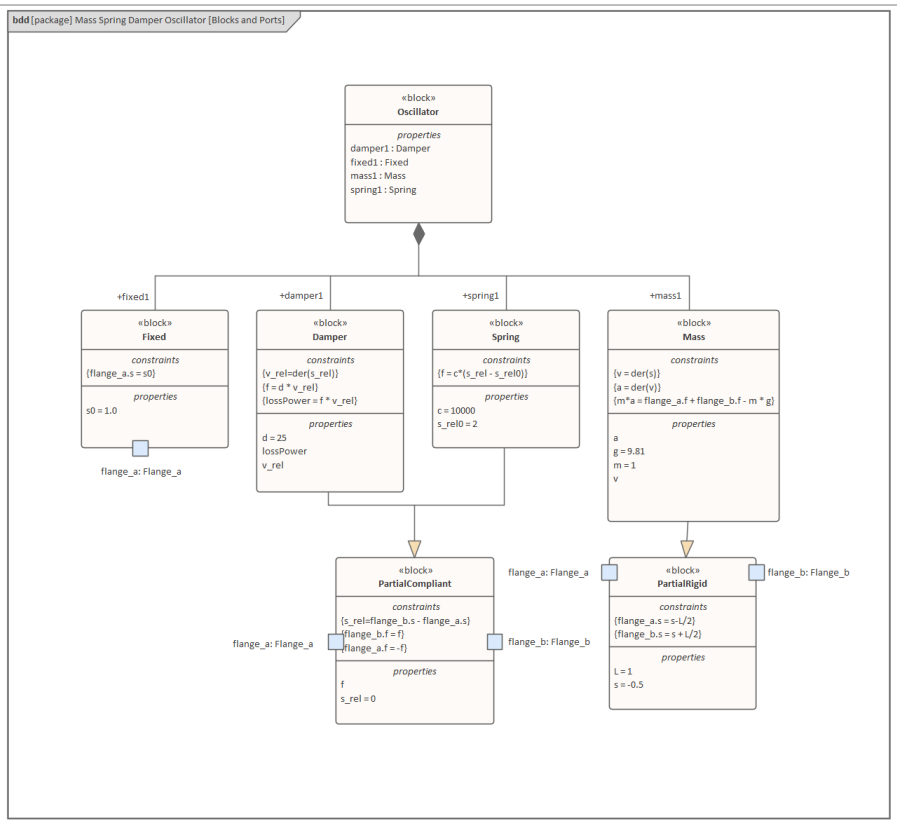
Créer Modèle SysML

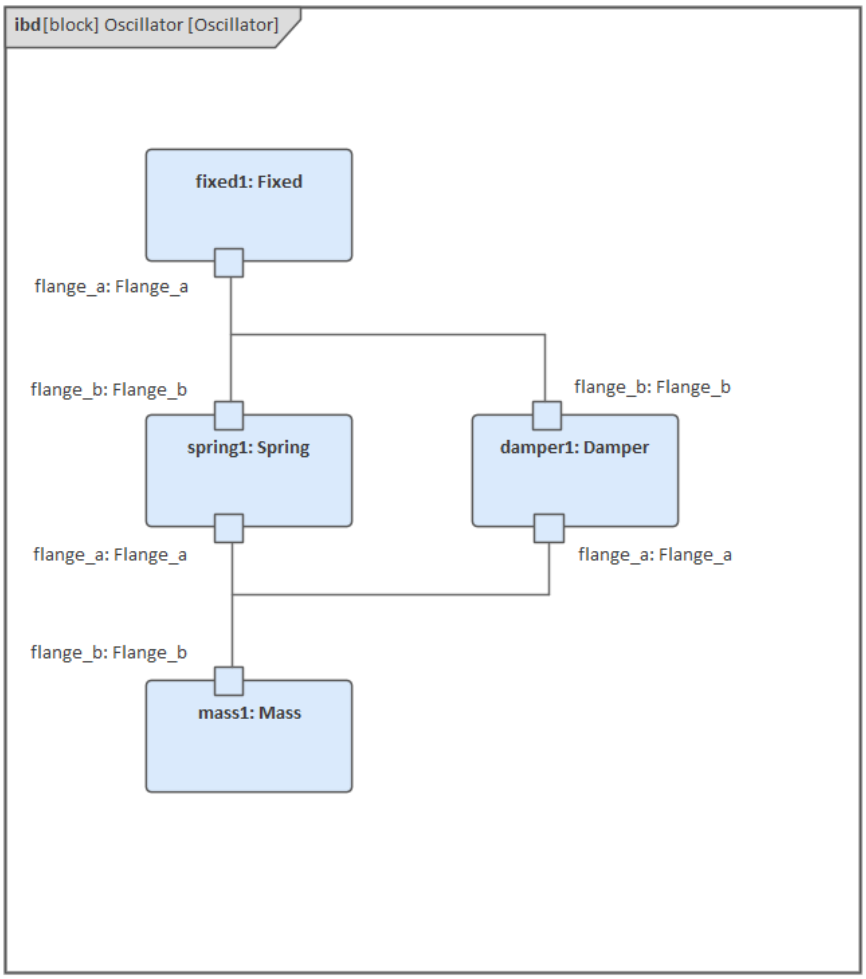
Le modèle MassSpringDamperOscillator dans SysML possède un Bloc principal, l' *Oscillateur* . L'Oscillateur est composé de quatre parties : un *plafond* fixe, un *ressort* , un *amortisseur* et un *corps de masse* . Créez un Bloc pour chacune de ces parties. Les quatre parties du Bloc Oscillateur sont connectées via des Ports, qui représentent des brides mécaniques.

Composants	Description
Types de ports	Les blocs « Flange_a » et « Flange_b » utilisés pour les brides dans le domaine mécanique de transition 1D sont identiques mais ont des rôles légèrement différents, quelque peu analogues aux rôles de PositivePin et NegativePin dans le domaine électrique. Les forces sont transmises à travers les brides. L'attribut *isConserved* de la propriété d'écoulement *Flange.f* doit donc être défini sur True.



Blocs et ports	• Créez les blocs « Ressort », « Amortisseur », « Masse » et « Fixe » pour représenter respectivement le ressort, l'amortisseur, le corps de masse et le plafond • Créez un Bloc « PartialCompliant » avec deux ports (brides), nommés « flange_a » et « flange_b » — ceux-ci sont respectivement de type Flange_a et Flange_b ; les blocs « Spring » et « Damper » généralisent à partir de « PartialCompliant » • Créez un Bloc « PartialRigid » avec deux ports (brides), nommés « flange_a » et « flange_b » — ceux-ci sont respectivement de type Flange_a et Flange_b ; le Bloc « Mass » se généralise à partir de « PartialRigid » • Créez un Bloc « Fixe » avec une seule bride pour le plafond, qui n'a que le port « flange_a » typé sur Flange_a

	
Structure interne	Créez un diagramme Bloc internes (IBD) pour « Oscillateur ». Ajoutez des propriétés pour le plafond fixe, le ressort, l'amortisseur et le corps de masse, typées par les blocs correspondants. Connectez les ports avec des connecteurs. • Connectez « flange_a » de « fixed1 » à « flange_b » de « spring1 » • Connectez « flange_b » de « damper1 » à « flange_b » de « spring1 » • Connectez « flange_a » de « damper1 » à « flange_a » de « spring1 » • Connectez « flange_a » de « spring1 » à « flange_b » de « mass1 »

	
Contraintes	Pour plus de simplicité, nous définissons les contraintes directement dans les éléments Bloc ; vous pouvez éventuellement définir des ConstraintBlocks, utiliser des propriétés de contrainte dans les Blocks et lier leurs paramètres aux propriétés du Bloc .

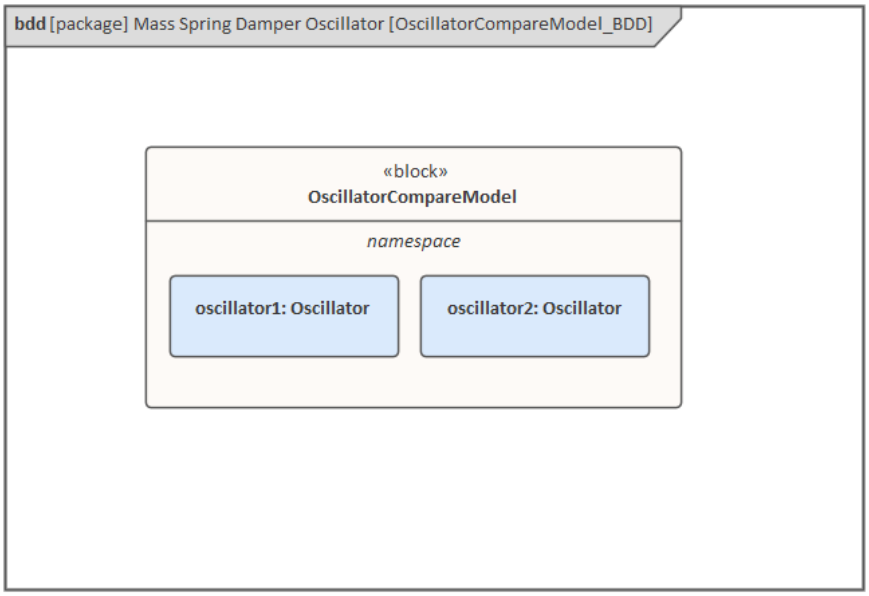
Comparaison de deux plans d'oscillateurs

Après avoir modélisé l'oscillateur, nous souhaitons effectuer une analyse hypothétique. Par exemple :

- Quelle est la différence entre deux oscillateurs avec des amortisseurs différents ?
- Et s'il n'y a pas d'amortisseur ?
- Quelle est la différence entre deux oscillateurs avec des ressorts différents ?
- Quelle est la différence entre deux oscillateurs avec des masses différentes ?

Voici les étapes pour créer un modèle de comparaison :

- Créer un Bloc nommé « OscillatorCompareModel »
- Créez deux Propriétés pour « OscillatorCompareModel », appelées *oscillator1* et *oscillator2,* et saisissez-les avec le Bloc *Oscillator*



Configurer DataSet et Exécuter Simulation

Créez un artefact de configuration SysMLSim et attribuez-le à ce Paquetage . Créez ensuite ces ensembles de données :

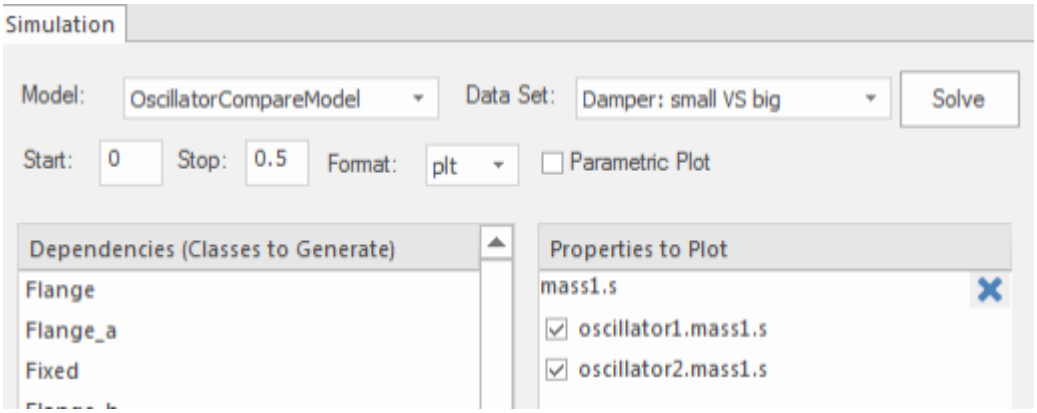
- Amortisseur : petit ou grand
  fournir 'oscillator1.damper1.d' avec la valeur 10 et 'oscillator2.damper1.d' avec la valeur la plus élevée 20
- Amortisseur : non ou oui
  fournir à 'oscillator1.damper1.d la valeur 0 ; ('oscillator2.damper1.d' utilisera la valeur par défaut 25)
- Printemps : petit vs grand
  fournir 'oscillator1.spring1.c' avec la valeur 6000 et 'oscillator2.spring1.c' avec la valeur plus grande 12000
- Masse : légère vs lourde
  fournir 'oscillator1.mass1.m' avec la valeur 0,5 et 'oscillator2.mass1.m' avec la valeur la plus élevée 2

La page configurée ressemble à ceci :

OscillatorCompareModel	SysMLSimModel
Part	
Damper: small VS big	Click button to configure...
oscillator2.damper1.d	20
oscillator1.damper1.d	10
Spring: small VS big	Click button to configure...
oscillator2.spring1.c	12000
oscillator1.spring1.c	6000
Damper: no VS yes	Click button to configure...
oscillator1.damper1.d	0
Mass: light VS Heavy	Click button to configure...
oscillator2.mass1.m	2
oscillator1.mass1.m	0.5

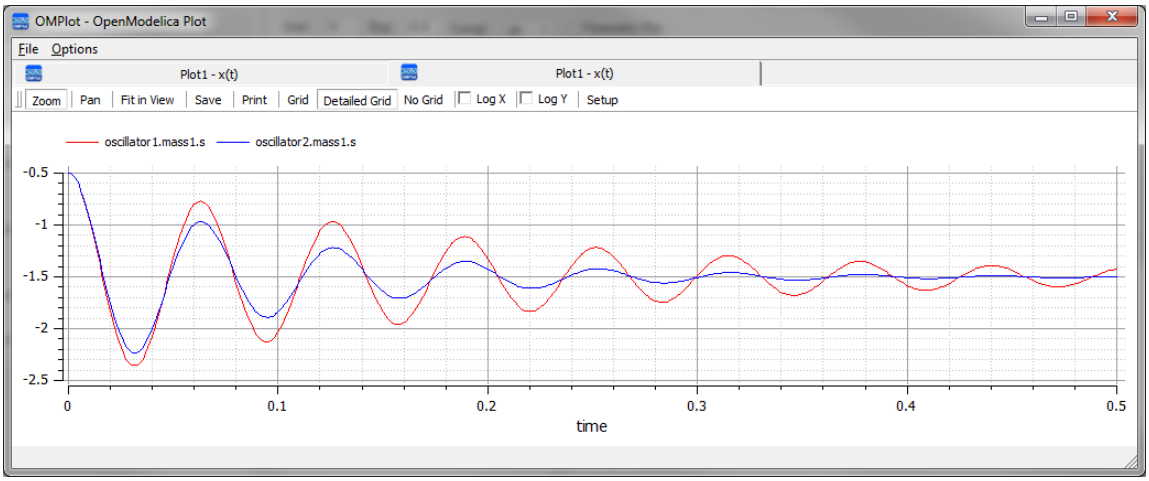
Sur la page « Simulation », sélectionnez « OscillatorCompareModel », tracez « oscillator1.mass1.s » et « oscillator2.mass1.s », puis choisissez l'un des ensembles de données créés et exécuter la simulation.

Conseil : S'il y a trop de propriétés dans la liste des parcelles, vous pouvez basculer la barre de filtre en utilisant le menu contextuel sur l'en-tête de la liste, puis taper 'mass1.s' dans cet exemple.

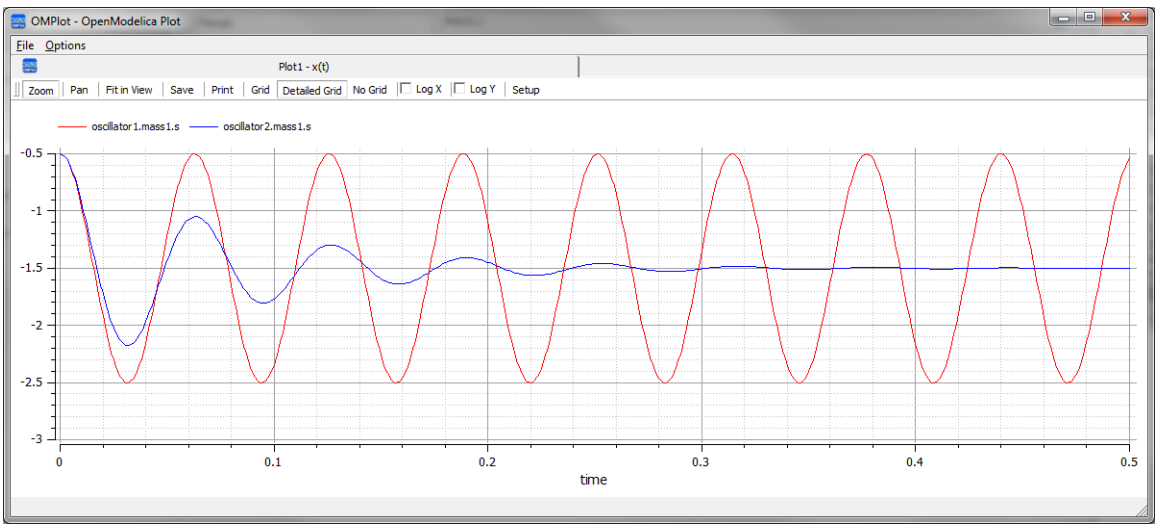


Voici les résultats de la simulation :

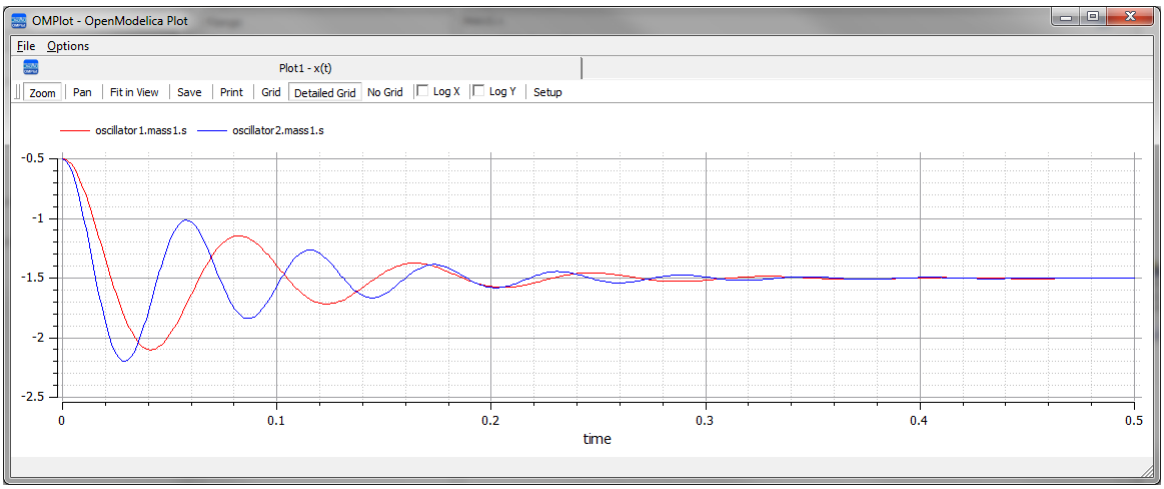
- Amortisseur, petit ou grand : le plus petit amortisseur permet au corps d'osciller davantage



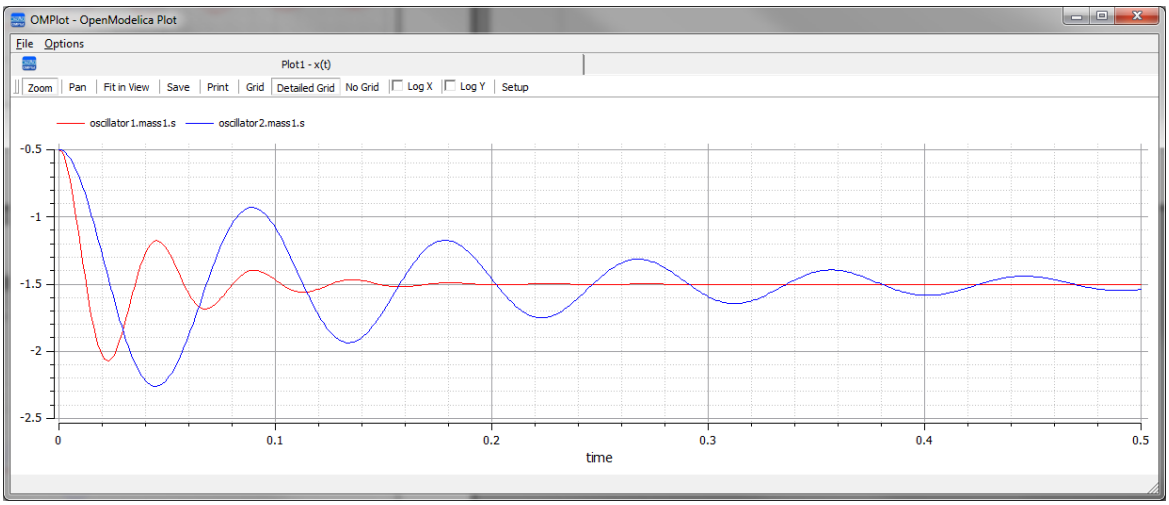
- Amortisseur, non vs oui : l'oscillateur ne s'arrête jamais sans amortisseur



- Ressort, petit ou grand : le ressort avec le plus petit « c » oscillera plus lentement



- Masse, légère vs lourde : l' object avec la plus petite masse oscillera plus vite et régulera plus rapidement



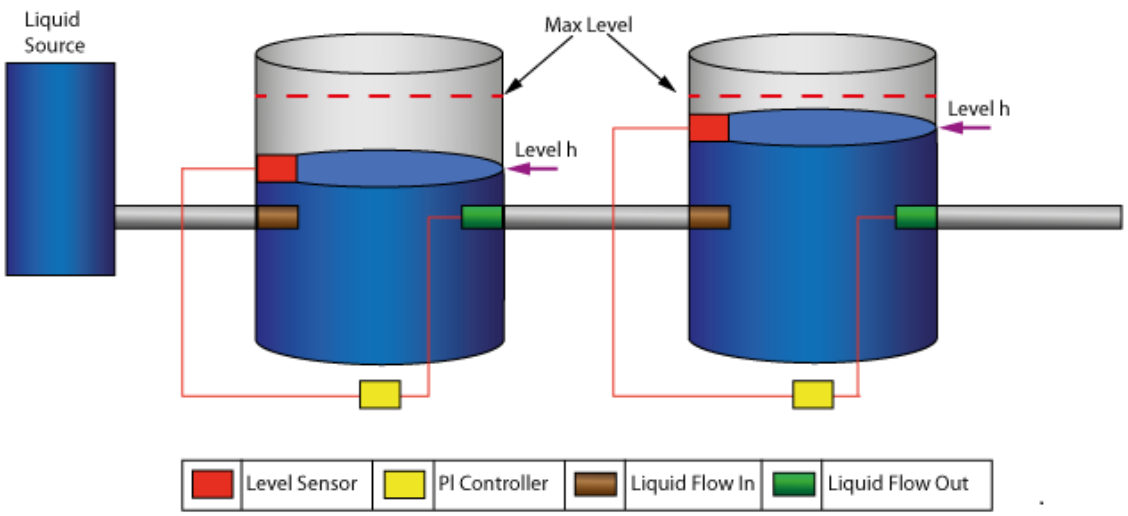
Régulateur de pression du réservoir d'eau

Dans cette section, nous allons parcourir la création d'un modèle SysML Paramétriques pour un régulateur de pression de réservoir d'eau, composé de deux réservoirs connectés, d'une source d'eau et de deux contrôleurs, chacun surveillant le niveau d'eau et contrôlant la vanne pour réguler le système.

Nous expliquerons le modèle SysML, le créerons et configurerons les Configurations SysMLSim. Nous exécuter ensuite la Simulation avec OpenModelica.

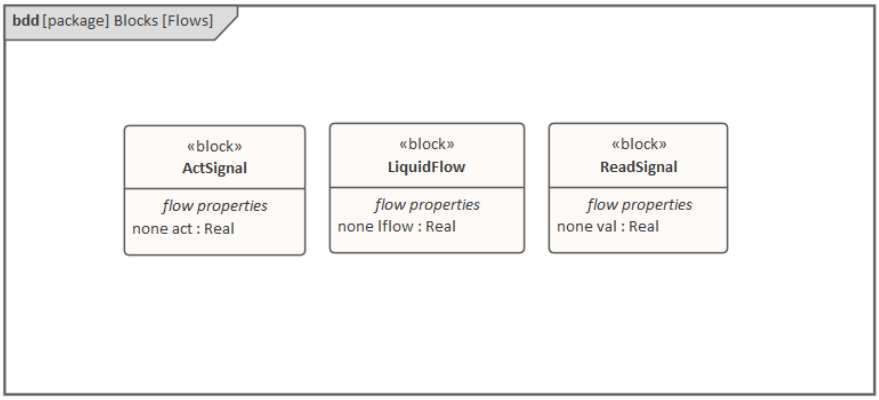
Système en cours de modélisation

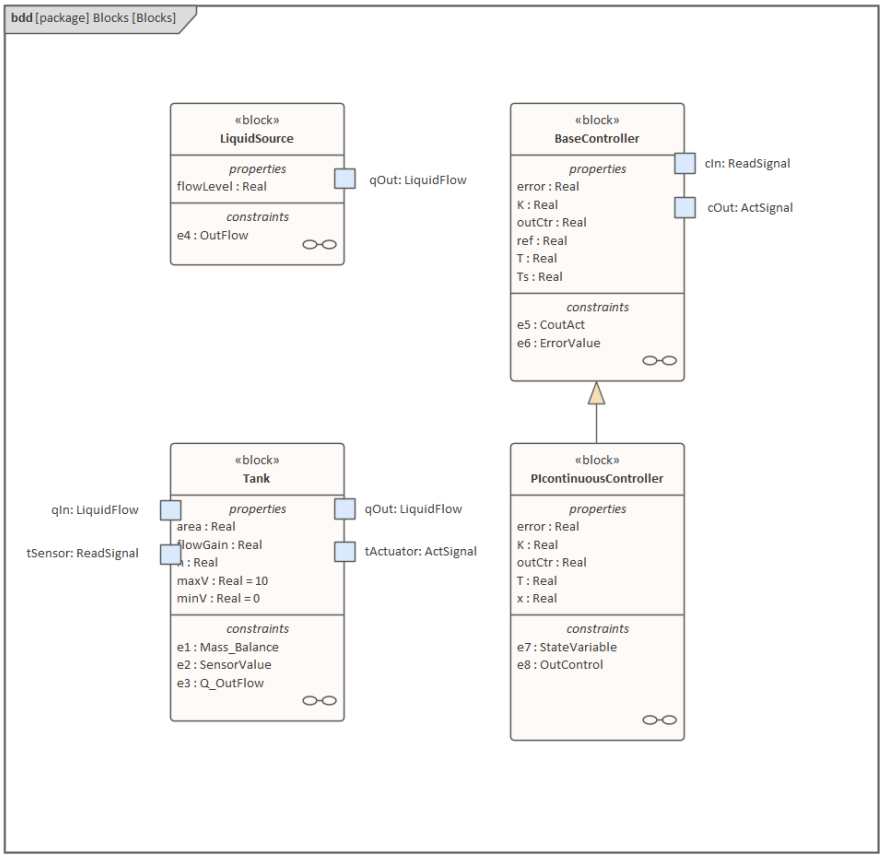
Ce diagramme représente deux réservoirs reliés entre eux et une source d'eau qui remplit le premier réservoir. Chaque réservoir est relié à un contrôleur continu proportionnel-intégral (PI), qui régule le niveau d'eau contenu dans les réservoirs à un niveau de référence. Pendant que la source remplit le premier réservoir d'eau, le contrôleur continu PI régule le débit sortant du réservoir en fonction de son niveau réel. L'eau du premier réservoir s'écoule dans le deuxième réservoir, que le contrôleur continu PI tente également de réguler. Il s'agit d'un problème physique naturel, non spécifique à un domaine.



Créer Modèle SysML

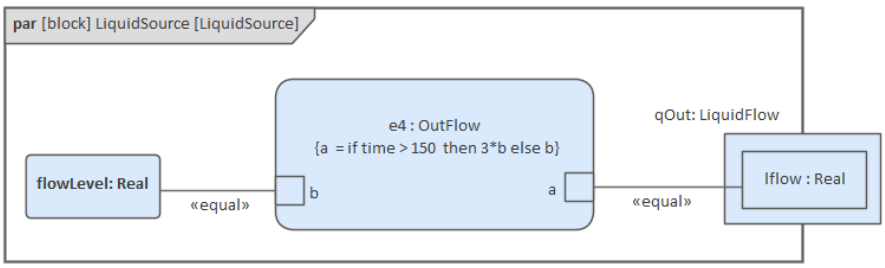
Composant	Discussion
Types de ports	Le char dispose de quatre ports qui sont typés sur ces trois blocs : • ReadSignal : lecture du niveau de liquide ; cette propriété a une unité « m » • ActSignal : Le signal envoyé à l'actionneur pour régler la position de la vanne • LiquidFlow : Le débit de liquide aux entrées ou aux sorties ; il a une propriété « lflow » avec l'unité « m^3/s »

	
Interrompre lorsqu'une Variable Change de Valeur	LiquidSource : l'eau entrant dans le réservoir doit provenir de quelque part, c'est pourquoi nous avons un composant de source liquide dans le système de réservoir, avec la propriété *flowLevel* ayant une unité de « m ^{3} /s ». Un port « qOut » est typé sur « LiquidFlow ». Réservoir : Les réservoirs sont connectés aux contrôleurs et aux sources de liquide via des ports. • Chaque réservoir dispose de quatre ports : - qIn : pour le flux d'entrée - qOut : pour le flux de sortie - tSensor : pour fournir des mesures de niveau de fluide - tActionneur : pour régler la position de la vanne à la sortie de le réservoir • Propriétés : - volume (unité='m ^{3} ') : capacité du réservoir, intervenant dans le *bilan massique* équation - h (unité = 'm') : niveau d'eau, intervenant dans le *bilan de masse* équation ; sa valeur est lue par le capteur - flowGain (unité = 'm ^{3} /s') : le débit de sortie est lié à la vanne position par *flowGain* - minV, maxV : Limites du débit de la vanne de sortie BaseController : ce Bloc peut être le parent ou l'ancêtre d'un contrôleur continu PI et d'un contrôleur discret PI. • Ports: - cIn : Niveau du capteur d'entrée - cOut : Contrôle vers l'actionneur • Propriétés : - Ts (unité = 's') : Période de temps entre les échantillons discrets (non utilisée dans cet exemple) - K : Facteur de gain - T (unité = 's') : Constante de temps du contrôleur - ref : niveau de référence - erreur : différence entre le niveau de référence et le niveau réel niveau d'eau, obtenu à partir du capteur - outCtr : signal de commande vers l'actionneur pour contrôler la vanne position PIcontinuousController : spécialisation à partir de BaseController • Propriétés : - x : la variable d'état du contrôleur



Blocs de contraintes

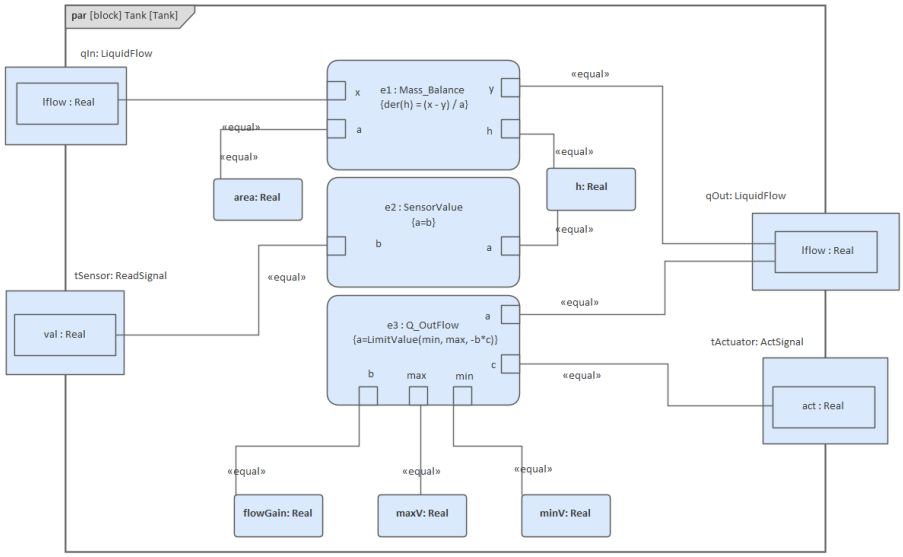
Le débit augmente brusquement à l'instant = 150 jusqu'à un facteur trois du niveau de débit précédent, ce qui crée un problème de contrôle intéressant que le contrôleur du réservoir doit gérer.



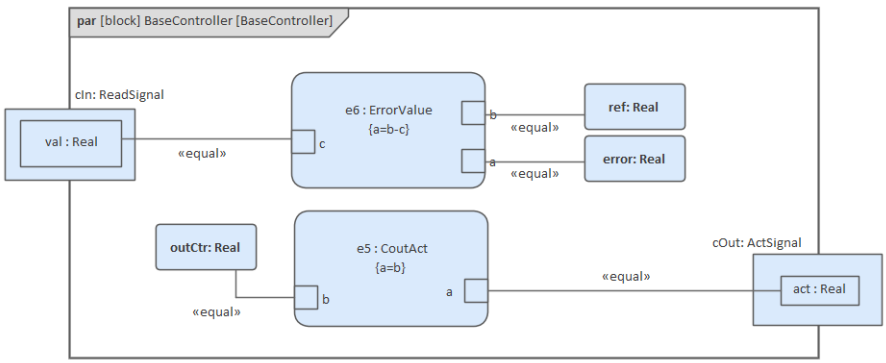
L'équation centrale régulant le comportement du réservoir est l'équation *de bilan massique* .

Le débit de sortie est lié à la position de la vanne par un paramètre « flowGain ».

Le capteur lit simplement le niveau du réservoir.



Les contraintes définies pour « BaseController » et « PIcontinuousController » sont illustrées dans ces figures.



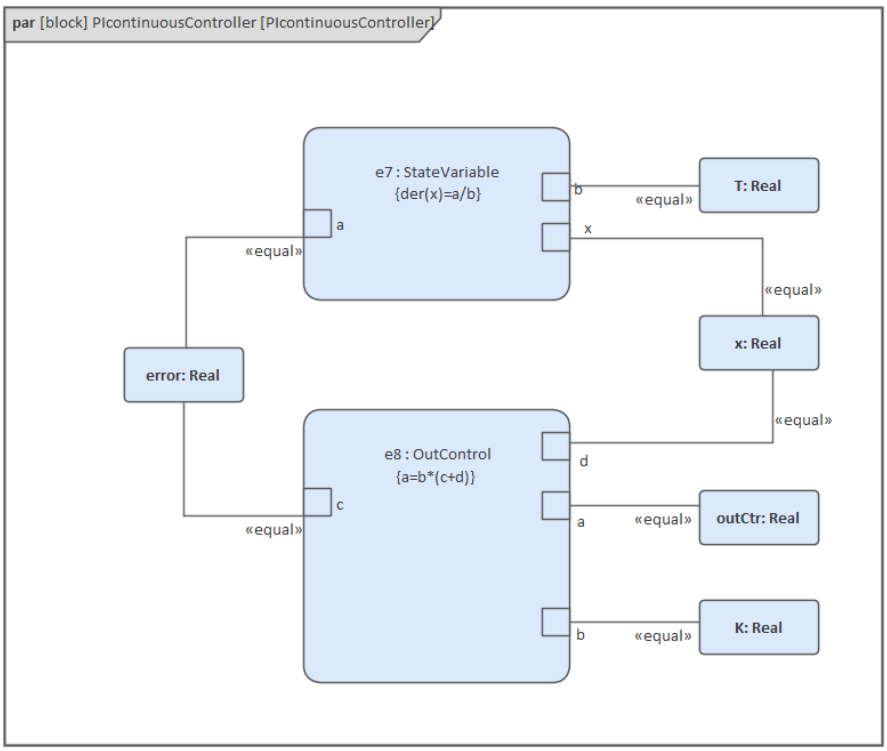
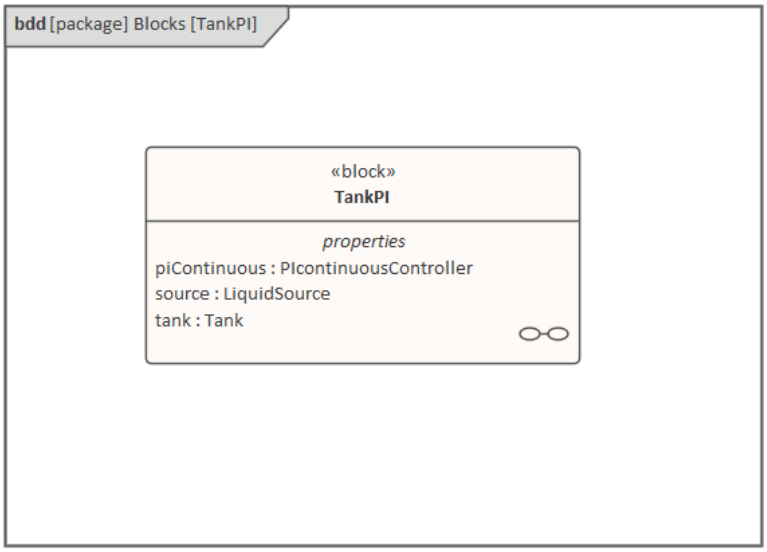
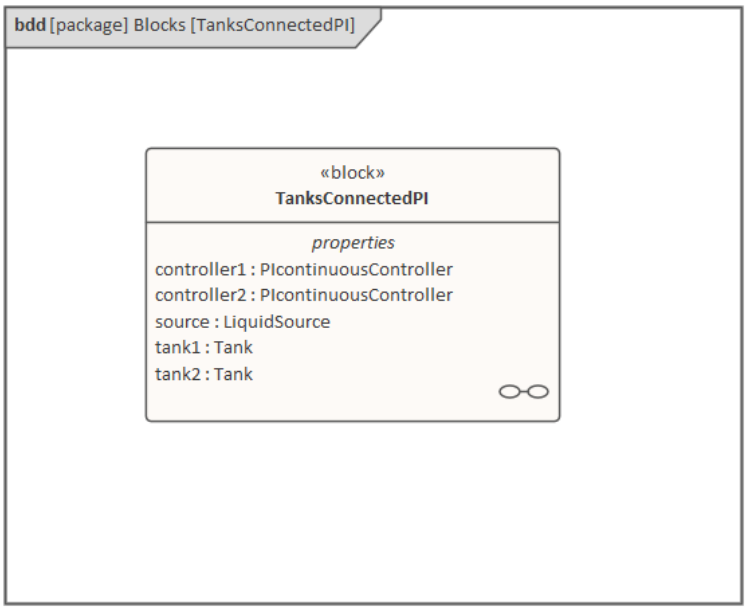


Diagramme Interne de Bloc	Il s'agit du diagramme Bloc interne d'un système avec un seul réservoir.  Il s'agit du diagramme Bloc interne d'un système avec deux réservoirs connectés. 

Exécuter Simulation

Étant donné que *TankPI* et *TanksConnectedPI* sont définis comme « SysMLSimModel », ils seront répertoriés dans la zone de liste déroulante « Modèle » sur la page « Simulation ».

Sélectionnez *TanksConnectedPI* et observez les modifications suivantes dans l'interface graphique :

- Combobox « Ensemble de données » : sera rempli avec tous les ensembles de données définis dans *TanksConnectedPI*
- Liste « Dépendances » : collectera automatiquement tous les blocs, contraintes, fonctions SimFunctions et types de valeurs directement ou indirectement référencés par *TanksConnectedPI* (ces éléments seront générés sous forme de code OpenModelica)
- « Propriétés à tracer » : une longue liste de propriétés de variables « feuille » (c'est-à-dire qui n'ont pas de propriétés)

sera collectée ; vous pouvez en choisir une ou plusieurs à simuler, et les Propriétés seront affichées dans la légende du tracé

Créer un artefact et configurer

Sélectionnez 'Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager'.

Les éléments du Paquetage seront chargés dans le Gestionnaire de Configuration.

Configurez ces blocs et leurs propriétés comme indiqué dans ce tableau .

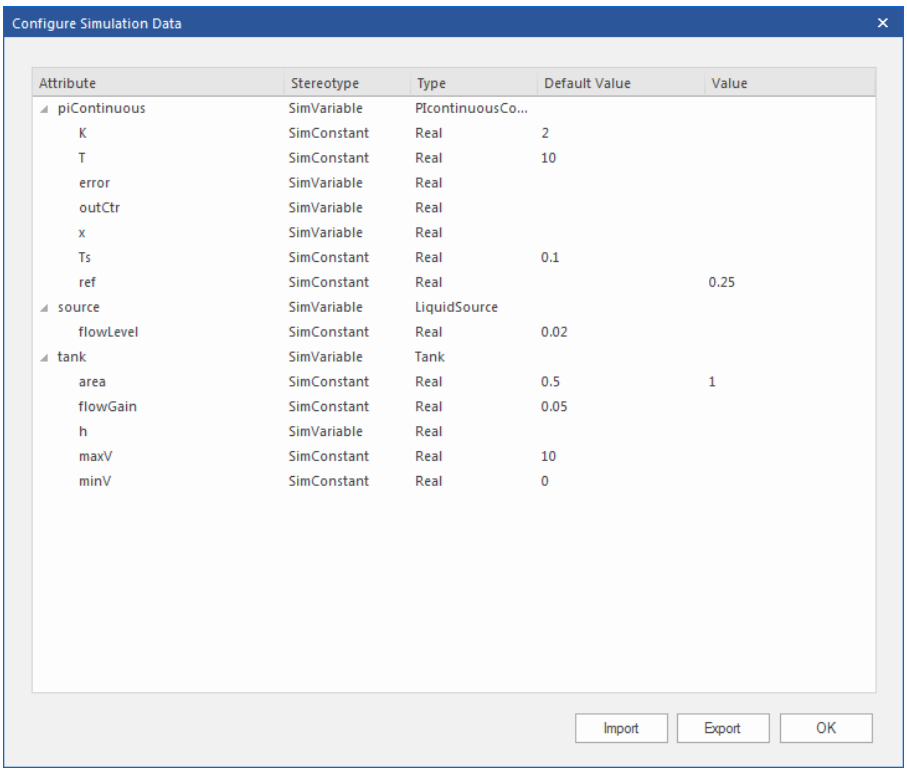
Note : Propriétés non configurées comme 'SimConstant' sont 'SimVariable' par défaut.

Bloc	Propriétés
Source liquide	Configurer comme « SysMLSimClass ». Configuration Propriétés : • flowLevel : défini comme « SimConstant »
Réservoir	Configurer comme « SysMLSimClass ». Configuration Propriétés : • zone : définir comme « SimConstant » • flowGain : défini sur "SimConstant" • maxV : défini comme « SimConstant » • minV : défini comme « SimConstant »
Contrôleur de base	Configurer comme « SysMLSimClass ». Configuration Propriétés : • K : définir comme « SimConstant » • T : définir comme « SimConstant » • Ts : définir comme « SimConstant » • ref : définir comme « SimConstant »
Contrôleur continu PI	Configurer comme « SysMLSimClass ».
RéservoirPI	Configurer comme « SysMLSimModel ».
RéservoirsConnectedPI	Configurer comme « SysMLSimModel ».

Configuration du jeu de données

Cliquez-droit sur chaque élément, sélectionnez l'option « Créer un jeu de données Simulation » et configurez les jeux de données comme indiqué dans ce tableau .

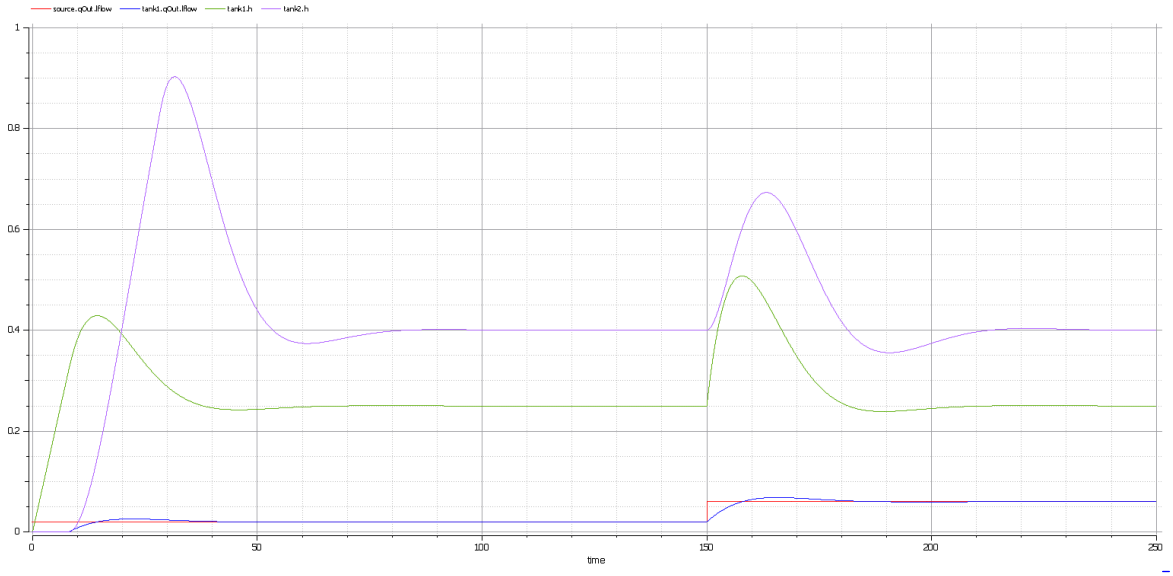
Élément	Ensemble de données
Source liquide	Niveau de débit : 0,02
Réservoir	h.début: 0

	Gain de débit : 0,05 superficie: 0,5 maxV: 10 minV: 0
Contrôleur de base	T: 10 K: 2 Ts: 0,1
Contrôleur continu PI	Aucune configuration nécessaire. Par défaut, le Bloc spécifique utilisera les valeurs configurées à partir du jeu de données par défaut du super Bloc .
RéservoirPI	Ce qui est intéressant ici, c'est que la valeur par défaut peut être chargée dans la dialogue « Configurer les données Simulation ». Par exemple, les valeurs que nous avons configurées comme dataSet par défaut sur chaque élément Bloc ont été chargées comme valeurs par défaut pour les propriétés de TankPI. Cliquez sur l'icône de chaque ligne pour étendre les structures internes de la propriété à une profondeur arbitraire.  Cliquez sur le bouton OK et revenez au gestionnaire de configuration. Les valeurs suivantes sont alors configurées : • tank.area: 1 ceci remplace la valeur par défaut 0,5 définie dans l'ensemble de données du Tank Bloc • piContinuous.ref: 0.25
RéservoirsConnectedPI	• contrôleur1.ref: 0.25 • contrôleur2.ref: 0.4

Simulation et analyse 1

Sélectionnez ces variables et cliquez sur le bouton Résoudre. Ce graphique devrait prompt :

- source.qOut.lflow
- réservoir1.qOut.lflow
- réservoir1.h
- réservoir2.h



Voici les analyses du résultat :

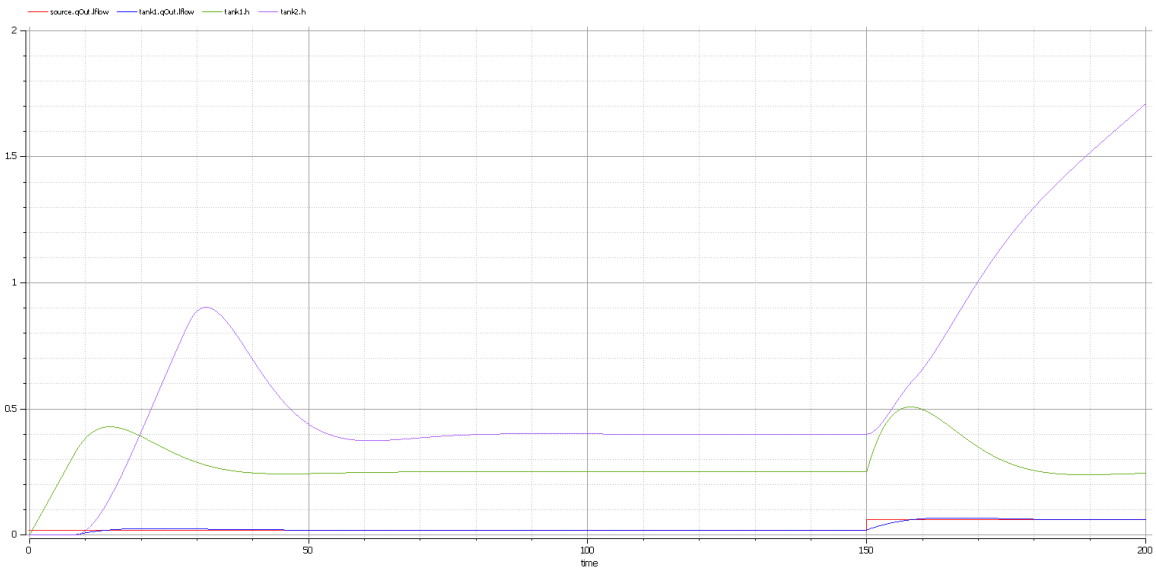
- Le débit du liquide augmente brusquement à l'instant = 150, jusqu'à 0,06 m ^{3} /s, soit un facteur trois du débit précédent (0,02 m ^{3} /s)
- Réservoir 1 régulé à une hauteur de 0,25 et réservoir 2 régulé à une hauteur de 0,4 comme prévu (nous avons défini la valeur du paramètre via l'ensemble de données)
- Les réservoirs 1 et 2 ont été régulés deux fois pendant la simulation ; la première fois avec un débit de 0,02 m ^{3} /s ; la deuxième fois avec un débit de 0,06 m ^{3} /s
- Le réservoir 2 était vide avant qu'il n'y ait un flux provenant du réservoir 1

Simulation et analyse 2

Dans l'exemple, nous avons défini les propriétés du réservoir « minV » et « maxV » sur les valeurs 0 et 10 respectivement. Dans le monde réel, un débit de 10 m ^{3} /s nécessiterait l'installation d'une vanne de très grande taille sur le réservoir.

Que se passerait-il si nous changions la valeur de « maxV » à 0,05 m ^{3} /s ? Sur la base du modèle précédent, nous pourrions effectuer les modifications suivantes :

- Sur le « DataSet_1 » existant de TanksConnectedPI, cliquez-droit et sélectionnez « Dupliquer le DataSet », puis renommez-le en « Tank2WithLimitValveSize »
- Cliquez sur le bouton pour configurer, développez « tank2 » et saisissez « 0,05 » dans la colonne « Valeur » pour la propriété « maxV »
- Sélectionnez « Tank2WithLimitValveSize » sur la page « Simulation » et tracez les propriétés
- Cliquez sur le bouton Résoudre pour exécuter la simulation



Voici les analyses des résultats :

- Notre changement s'applique uniquement au réservoir 2 ; le réservoir 1 peut réguler comme avant sur 0,02 m^3/s et 0,06 m^3/s
- Lorsque le débit de la source est de 0,02 m^3/s, le réservoir 2 peut réguler comme avant
- Cependant, lorsque le débit de la source augmente à 0,06 m^3/s, la vanne est trop petite pour permettre au débit de sortie de correspondre au débit d'entrée ; le seul résultat est que le niveau d'eau du réservoir 2 augmente
- Il appartient alors à l'utilisateur de résoudre ce problème ; par exemple, changer pour une vanne plus grande, réduire le débit de la source ou fabriquer une vanne supplémentaire.

En résumé, cet exemple montre comment ajuster les valeurs des paramètres en dupliquant un DataSet existant.

Intégration de Simscape

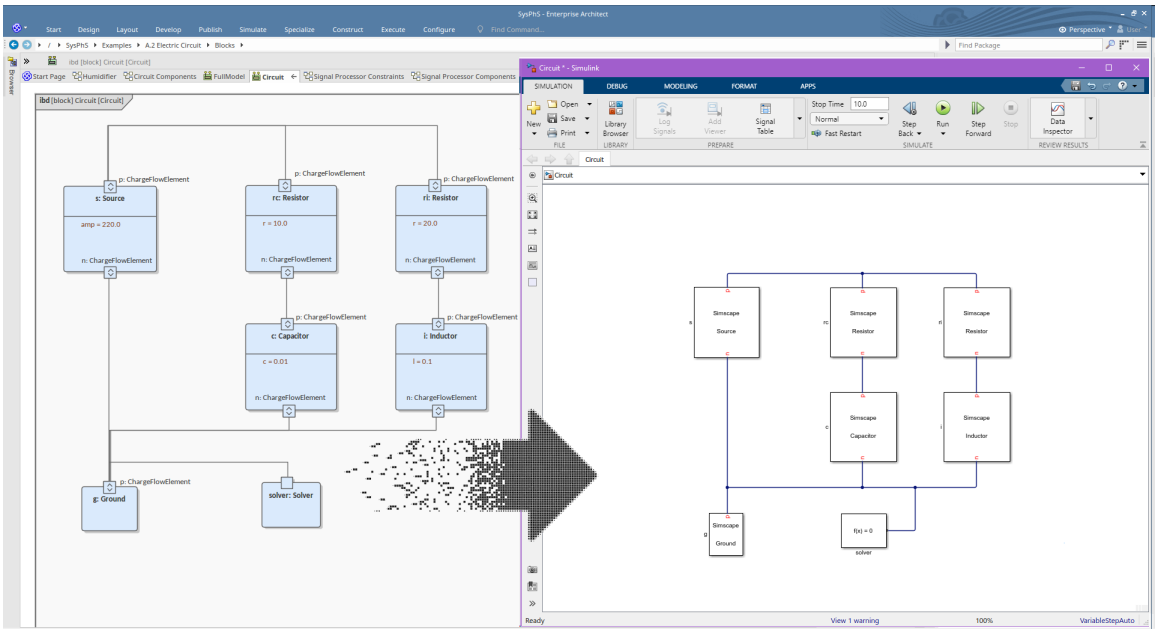
Simscape est utilisé pour modéliser des systèmes physiques avec des flux physiques, en traduisant diagrammes Bloc internes SysML en un modèle Simulink, ouvrant la vaste gamme de blocs de bibliothèque de Simscape dans de nombreux domaines physiques différents.

Enterprise Architect peut traduire diagrammes Bloc internes SysML en Simscape de MATLAB, une extension facultative de Simulink qui permet modélisation des systèmes physiques et demande à MATLAB de simuler et de tracer les sorties demandées. Les blocs représentent des objets physiques et les flux représentent le flux physique d'une substance ou d'une énergie (comme un liquide, un courant, un gaz, un flux, une force/un couple, un flux de chaleur, etc. ; par exemple, l'eau s'écoulant d'un réservoir à l'autre ou le courant traversant une résistance). Vous pouvez utiliser les motifs SysPhS intégrés pour accéder à la vaste gamme de blocs de bibliothèque Simscape prédéfinis ou créer des références à vos propres blocs de bibliothèque personnalisés à l'aide du stéréotype SimulinkBlock.

Voir le lien vers " *Principes de base de Modélisation des réseaux physiques »* pour plus de détails sur l'utilisation de Simulink et Simscape.

Les propriétés physiques des flux doivent être typées par Bloc et inclure une PhSVariable conservée et une PhSVariable non conservée. Utilisez les motifs Constructeur de Modèle pour SysPhS disponibles sous la perspective SysML, qui incluent :

- Éléments SysPhS pour l'interaction physique
- Éléments SysPhS pour le flux de signaux



Exigences

- L'option « Utiliser Simscape » doit être cochée dans la fenêtre Configurer Simulation SysML
- Les ports avec une direction « inout » (ou aucune) seront considérés comme des flux physiques
- Tout Bloc doté d'un port « inout » sera généré en tant que Simscape. S'il possède également des ports « in » ou « out », ceux-ci seront définis sur les « signaux physiques » de Simscape. Ceux-ci peuvent être connectés aux « signaux physiques » de sortie ou aux entrées et sorties Simulink. Le convertisseur Simulink requis pour cela sera automatiquement généré et inséré dans le modèle lors de la génération.

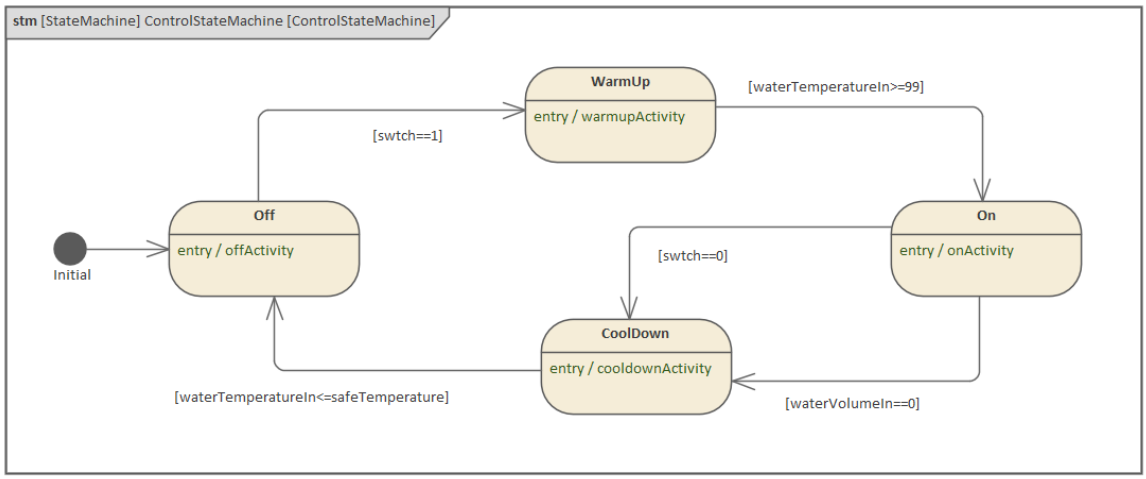
Vidéos

Sparx Systems propose une vidéo YouTube sur la génération d'un tracé Simulation SysML à l'aide de Simscape. Voir :

- SysML Simulation Plotting Using Simscape

Intégration Simulink

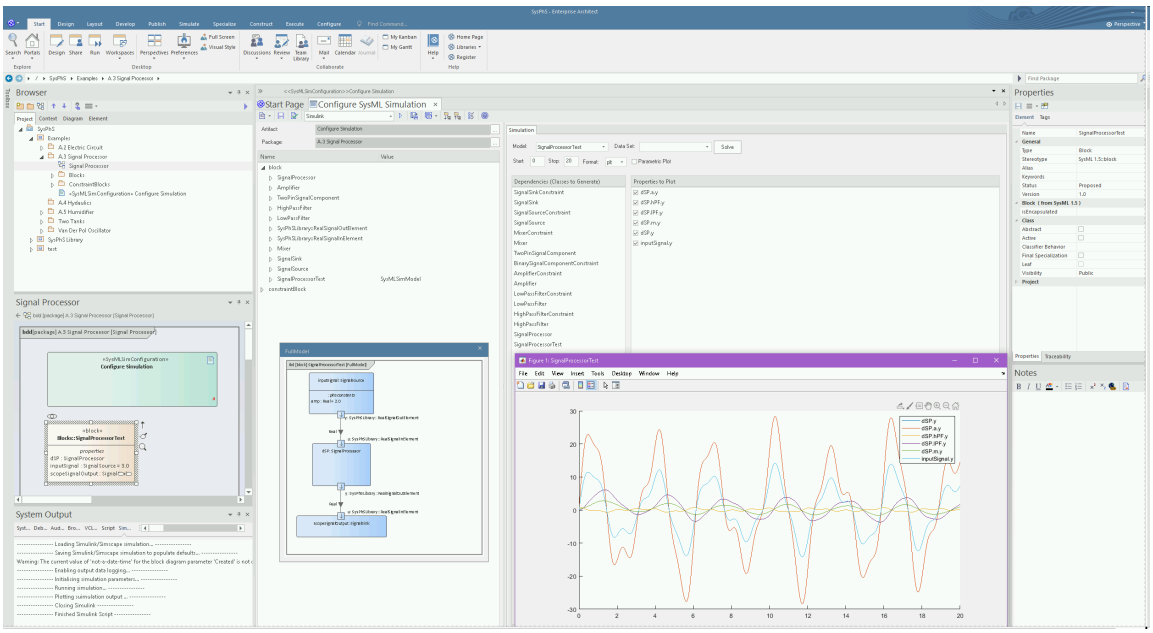
Enterprise Architect traduit un modèle SysML au format Simulink et exécuter automatiquement la simulation, en traçant les sorties des variables sélectionnées. Le fichier Simulink généré peut également être ouvert directement dans Simulink, ce qui permet de modifier et d'affiner les paramètres de simulation et les fonctionnalités de sortie.



Étant donné que Simulink utilise une approche diagramme Bloc où le flux de signal est unidirectionnel, Enterprise Architect supporte l'exportation de modèles vers Simulink pour simuler des messages dirigés entre les blocs, produisant des graphiques des résultats de simulation.

Vous disposez d'un accès par glisser-déposer aux blocs de bibliothèque Simulink intégrés courants directement via motifs Enterprise Architect ou vous pouvez référencer vos propres blocs personnalisés avec des paramètres de stéréotype sous la norme SysPhS.

Simulation avec Simulink est une alternative à l'utilisation d'OpenModelica, également disponible dans Enterprise Architect .



Modélisation pour Simulink

Pour diagrammes modélisation exportables vers Simulink, vous disposez d'un accès par glisser-déposer aux blocs de bibliothèque Simulink intégrés courants directement via motifs Enterprise Architect , ou vous pouvez référencer vos propres blocs personnalisés avec des paramètres de stéréotype sous la norme SysPhS.

Simulink peut être utilisé pour modéliser le flux de signaux, c'est-à-dire qu'un Bloc envoie une information à un autre Bloc . Cela signifie que, pour modélisation Simulink, tous les ports doivent avoir une direction définie comme *entrante* ou *sortante* . Les flux Item entre les ports doivent correspondre aux directions entrante/sortante. (Note que Modelica supporte modélisation des signaux ou des systèmes physiques.)

Exigences

Afin de se conformer aux exigences unidirectionnelles de Simscape :

- Les ports doivent avoir une direction définie comme « entrée » ou « sortie »
- Les ports doivent être connectés en fonction de leur direction ; par exemple, un port « in » ne peut pas être connecté à un autre port « in »
- Les paramètres de contrainte seront considérés comme des variables, sauf s'ils sont spécifiés comme stéréotype « PhSConstant » (ou définis sur SimConstant dans la fenêtre Configurer Simulation SysML)
- Les équations de contrainte doivent avoir un seul terme sur le côté gauche (lhs) du signe égal, et ce terme doit être soit une sortie, soit une dérivée

Vidéos

Sparx Systems propose une vidéo YouTube sur la réalisation d'une Simulation Statemachine SysML dans Simulink. Voir :

- SysML Statemachine Simulation in Simulink

Notes

- Simulink permettra uniquement de tracer les ports de sortie ; il pourrait y avoir une amélioration future pour permettre également de tracer d'autres variables - note qu'un port d'entrée est intrinsèquement le même que le port de sortie qui s'y connecte

Apprendre encore plus

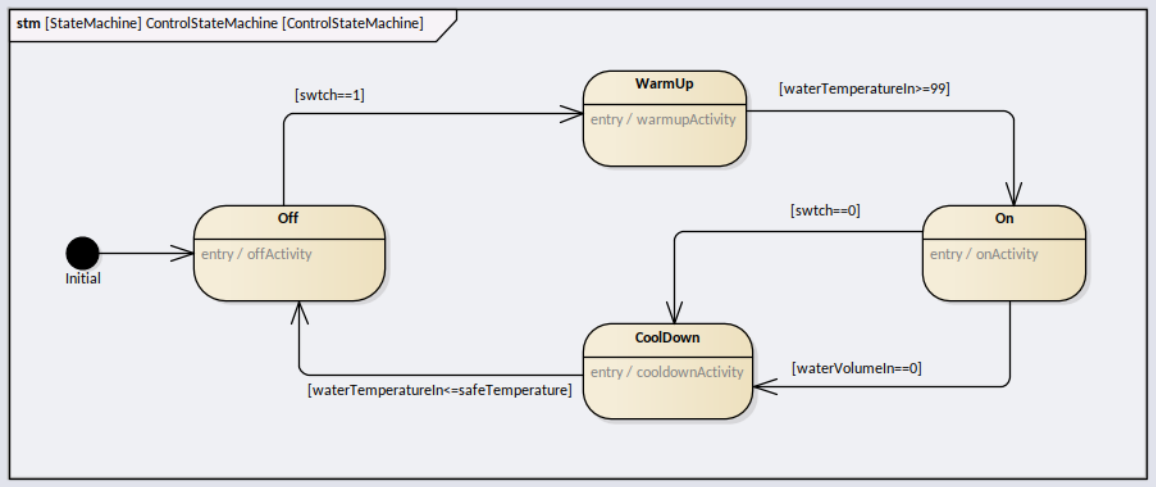
- Creating Simulink and Simscape Specific Blocks

Intégration de Stateflow

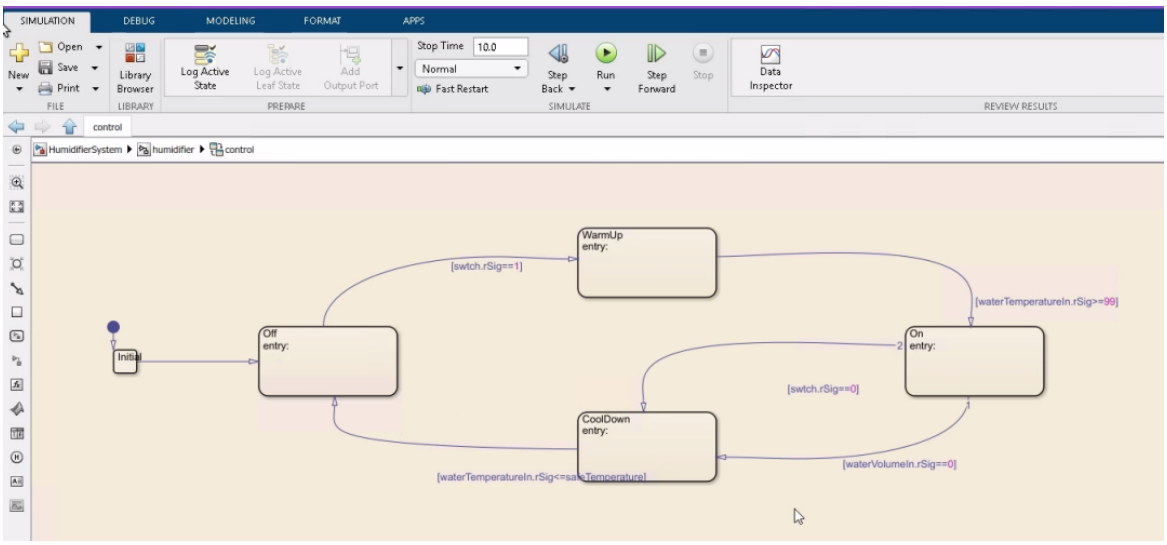
Une fonctionnalité importante lors de l'utilisation de la simulation SysML d' Enterprise Architect est la possibilité de générer diagrammes MATLAB Stateflow à exécuter sous Simulink, vous permettant de guider vos simulations SysML à l'aide Statemachines modélisées dans Enterprise Architect qui sont traduites en diagrammes Stateflow.

Les diagrammes Stateflow utilisent des éléments et des connecteurs qui ont des équivalents proches de la norme OMG Statemachine , tels que States et les Transitions. Enterprise Architect supporte toutes les fonctionnalités de Stateflow ; cependant, Stateflow n'utilise qu'un sous-ensemble des normes OMG Statemachine .

Voici un exemple de diagramme Statemachine dans Enterprise Architect :



Voici un exemple de la manière dont le code généré par Enterprise Architect , à partir du diagramme ci-dessus, est rendu sous forme de diagramme Stateflow :



Types d'éléments pris en charge

Les types d'éléments SysML qui peuvent être traduits en objets StateFlow sont :

- State
- Initial
- Final
- Jonction

- Choix (converti en Stateflow Junction)
- Histoire

Types de connecteurs pris en charge

Le type de connecteur clé SysML Statemachines a une traduction directe vers le type de connecteur Simulink « Transition ».

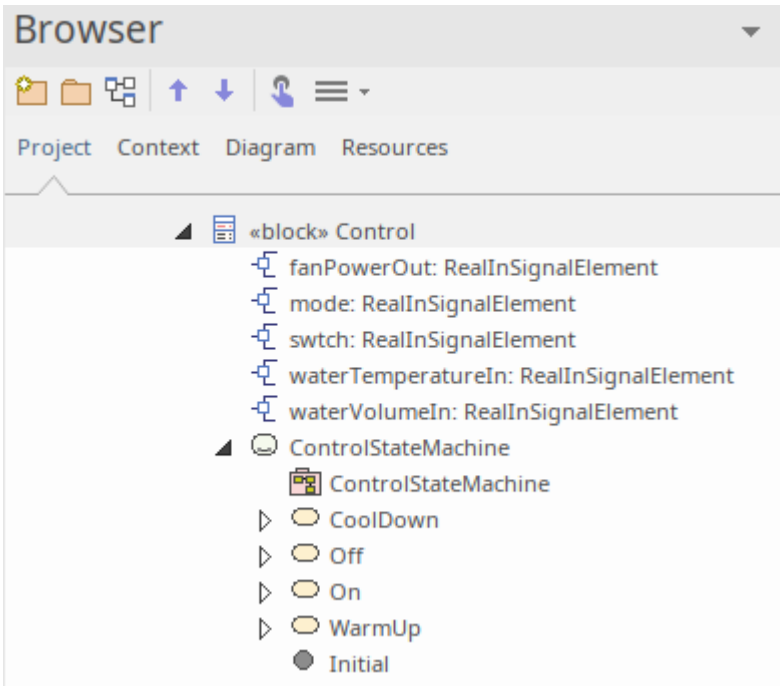
Modélisation pour Stateflows

Vous disposez de plusieurs options lors de la création d'un diagramme Statemachine destiné à être rendu sous forme de diagramme Stateflow. Celles-ci incluent les éléments et les connecteurs pouvant être utilisés, le placement du Statemachine , l'emplacement du code pouvant être défini dans le modèle et le format de code pouvant être utilisé.

Placement Statemachine

Lors de la création d'un modèle à générer sous forme de diagramme Stateflow, le diagramme Statemachine doit être placé sous un Bloc SysML et avoir un élément Statemachine enfant, avec un diagramme Statemachine sous cet élément Statemachine .

Par exemple, dans ce cas, le Bloc « Control » a une Statemachine enfant nommée « ControlStateMachine », qui a un diagramme Statemachine enfant « ControlStateMachine » :



Note que le Bloc ne doit contenir qu'un diagramme Statemachine ; il ne doit pas contenir d'autres diagrammes .

Définition du code

Il existe plusieurs options pour placer un script dans un modèle Statemachine . Les principales considérations à prendre en compte sont les suivantes : quel est le format du script et où peut-il être placé ?

Format de code

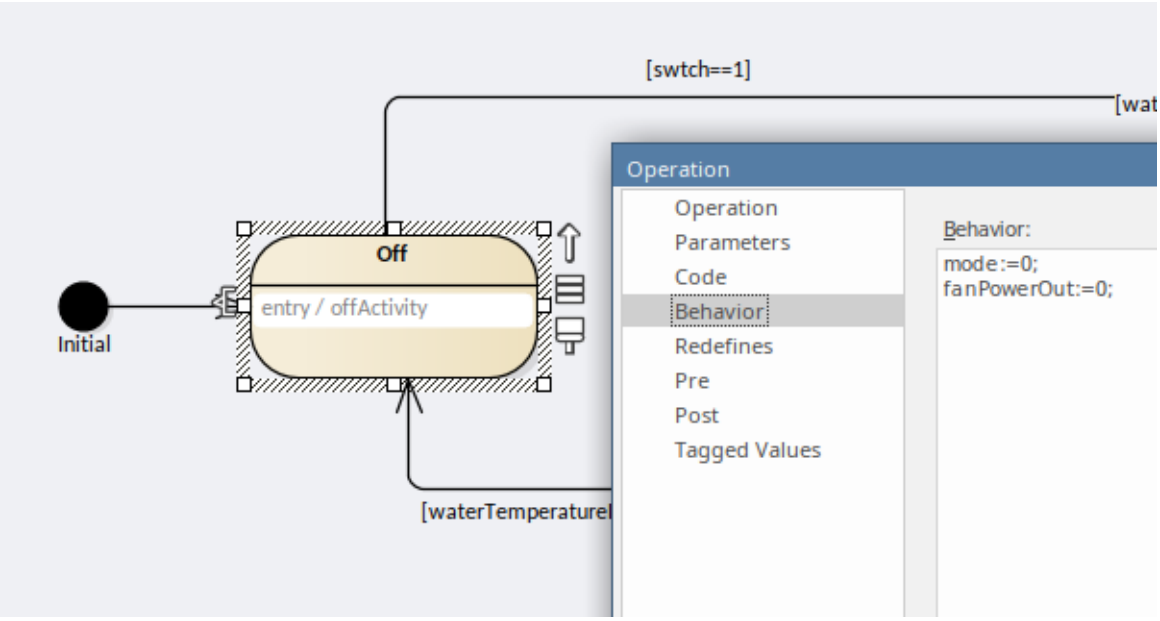
Lorsque vous écrivez un code SysML valide, il sera traduit en code Stateflow valide. La spécification SysPhS définit le langage mathématique Modelica comme « standard », ce Enterprise Architect accepte comme extraits de code dans le diagramme Statemachine , et les traduit en équivalent MATLAB/Stateflow. Pour plus de détails, reportez-vous à la section 10 de la *spécification OMG SysPhS* .

Placement du code

Interne à un State et conformément à la spécification UML , Stateflow supporte les trois opérations standard :

- Entrée
- Sortie
- Faire

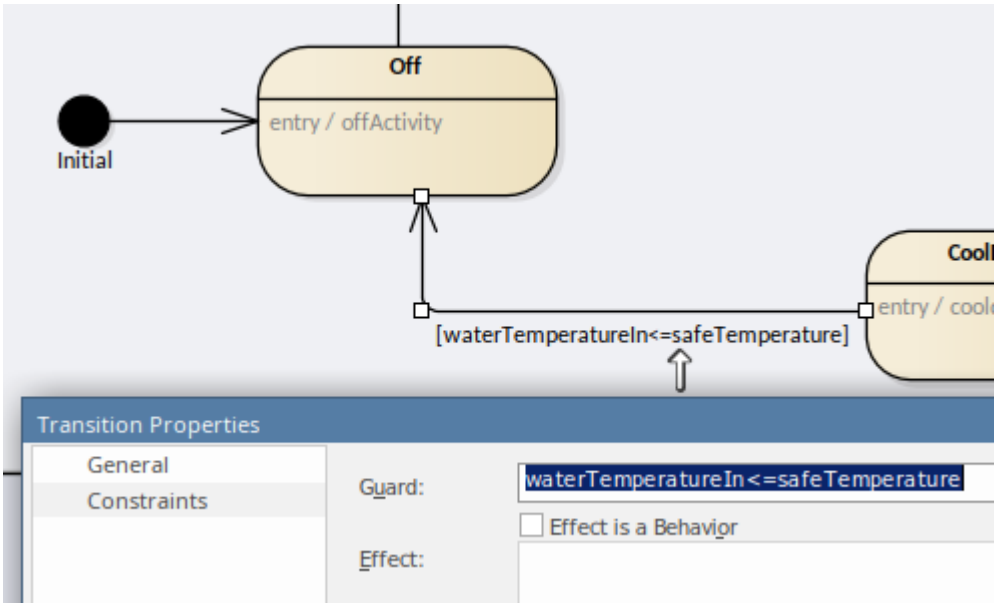
Chacun d'entre eux peut contenir du code, qui est défini dans l'opération de State sous Comportement. Par exemple, voici un extrait de code pour une entrée :



Dans Transitions, il existe trois options clés pour utiliser le code :

- Garde de transition
- Effet de transition
- Déclencheur

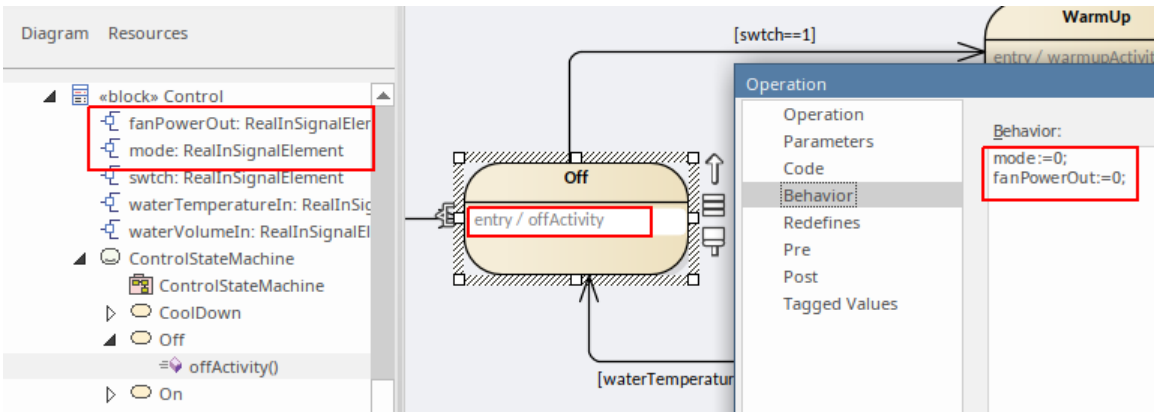
Par exemple, il s'agit d'une déclaration de condition telle que définie dans une garde de transition :



Utilisation Bloc Propriétés

Pour le code, les Propriétés Bloc telles que les constantes et les ports peuvent être référencées dans le script. Dans le cas des ports, les détails saisis dans le Bloc sont dérivés à l'aide du nom du port d'entrée et, de la même manière, des valeurs peuvent être attribuées à une variable du même nom que le port de sortie.

Par exemple:



Notes

Ces fonctionnalités SysML Statemachine ne sont pas disponibles dans Stateflow :

- État de l'histoire profonde
- États d'historique qui ont des transitions sortantes (c'est-à-dire States initiaux) à utiliser sur toutes les entrées par défaut
- Fourche/Jointure
- Synchroniser
- Jonction (uniquement Choix - Jonction sera remplacée par Choix lorsque cela est possible)
- Points d'entrée/sortie

- Références à d'autres Statemachines ; Stateflow peut faire Statemachines enfants mais elles ne peuvent pas être référencées à nouveau
- Une seule Statemachine par Bloc SysML

Intégration OpenModelica

OpenModelica est un environnement libre et open source basé sur le langage modélisation Modelica pour modélisation , la simulation, l'optimisation et l'analyse de systèmes dynamiques complexes. Enterprise Architect est intégré à OpenModelica et supporte son utilisation dans le cadre de la norme SysPhS (*extension SysML pour Simulation d'interaction physique et de flux de signaux*) pour définir des constantes et des variables dans les blocs SysML plutôt que dans la configuration Simulation . Cela fournit une méthode basée sur un modèle plus simple pour définir et partager des simulations.

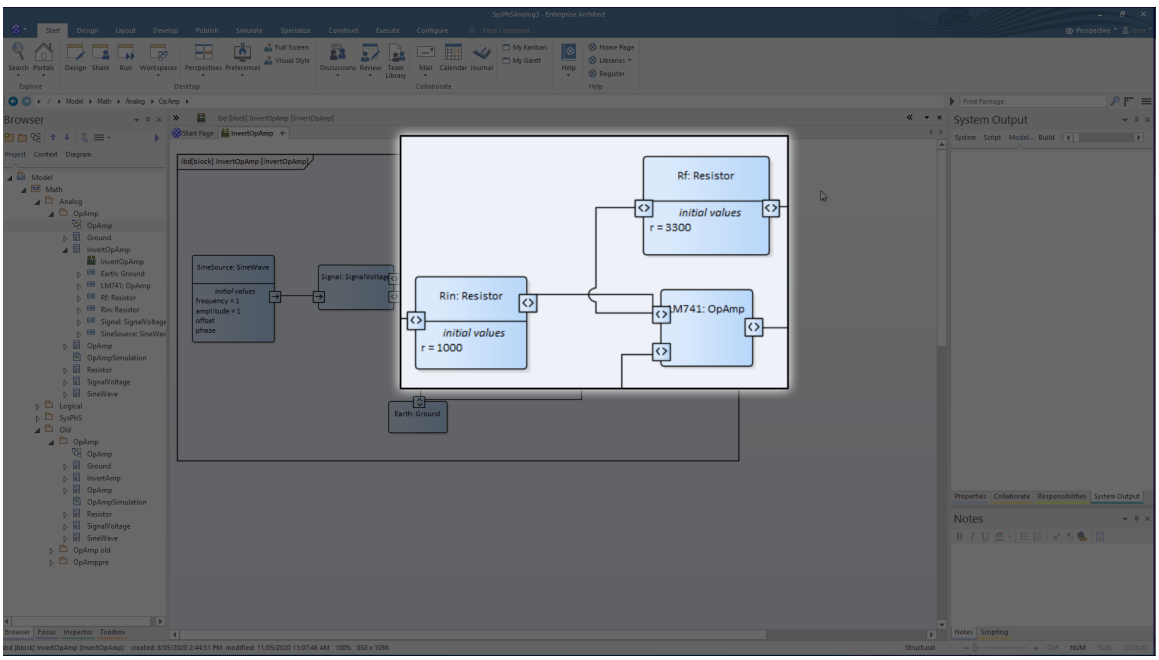
Vous pouvez également afficher les diagrammes Bloc SysML de vos modèles dans Enterprise Architect dans l'éditeur de connexion OpenModelica, OMEdit, qui affiche les alias et notes des blocs.

Vous pouvez créer des blocs à la volée en utilisant les nouveaux motifs SysPhS prêts à être simulés dans OpenModelica, en référençant des blocs de bibliothèque OpenModelica existants ou des blocs personnalisés définis par l'utilisateur. Avec la dernière génération de code OpenModelica, vous pouvez visualiser vos composants SysML dans des clients OpenModelica compatibles tels que OMEdit, ainsi que simuler des tracés.

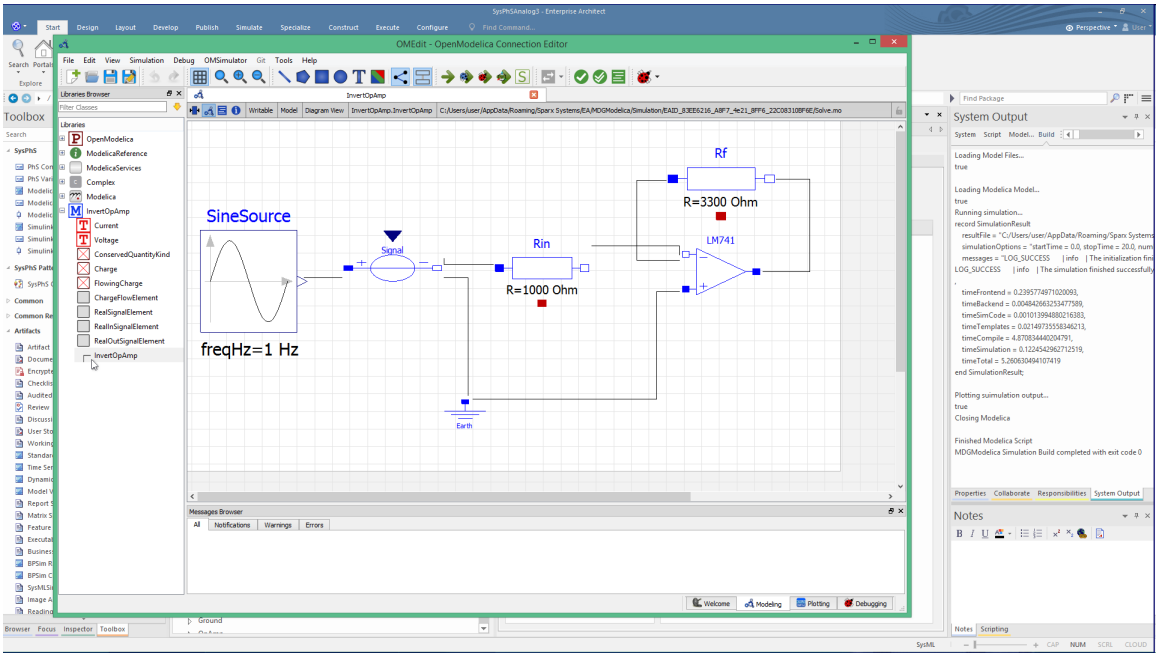
Pour plus de détails sur l'installation d'OpenModelica et la connexion Enterprise Architect à celui-ci, consultez la rubrique d'aide de la plateforme sur laquelle Enterprise Architect est installé.

L'utilisation d'OpenModelica est une alternative à l'utilisation de MATLAB Simulink pour effectuer la simulation de modèles Paramétriques dans Enterprise Architect . Dans les deux cas, vous configurez vos modèles à l'aide de la norme SysPhS, qui définit comment effectuer la traduction entre un modèle SysML et un modèle OpenModelica ou un modèle Simulink/Simscape.

Voici un exemple de composants définis à l'aide de parties SysML spécifiques à SysPhS :



Les composants sont tels qu'illustrés dans ce diagramme OpenModelica généré à partir du modèle SysPhS :



Installation

Plate-forme	Détail
Windows	Si Enterprise Architect est installé sur une plate-forme Windows , consultez la rubrique d'aide *OpenModelica sur Windows* .
Linux	Si Enterprise Architect est installé sur une plate-forme Linux, consultez la rubrique d'aide *OpenModelica sur Linux* .

Application

Utiliser Bibliothèque OpenModelica	Détails sur le référencement des ressources disponibles dans les bibliothèques OpenModelica.
Analyse de Modèle utilisant Ensemble de Données	Grâce à la configuration Simulation , un Bloc peut être configuré pour avoir plusieurs jeux de données définis par rapport à lui. Cela permet des variations de simulation reproductibles à l'aide du même modèle SysML.
Simulation de dépannage	Cette rubrique décrit les problèmes possibles qui peuvent survenir lors de l'utilisation d'OpenModelica (ou de MATLAB Simulink) pour la simulation.

OpenModelica sur Windows

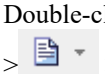
Lors de l'installation d'OpenModelica pour Enterprise Architect fonctionnant sur une plate-forme Windows , vous installez d'abord l'application OpenModelica, puis configurez les paramètres dans Enterprise Architect pour accéder à OpenModelica.

Installer OpenModelica

Étape	Action
1	Téléchargez l'installateur OpenModelica à partir de : https://openmodelica.org/download/download-windows
2	Double-cliquez sur l'installateur OpenModelica et suivez les instructions de « Assistant ». Nous vous recommandons d'accepter le chemin par défaut pour l'installation.
3	Vérifiez que vous pouvez localiser l'exécutable omc.exe. Par exemple : C:\OpenModelica1.9.2\bin\omc.exe

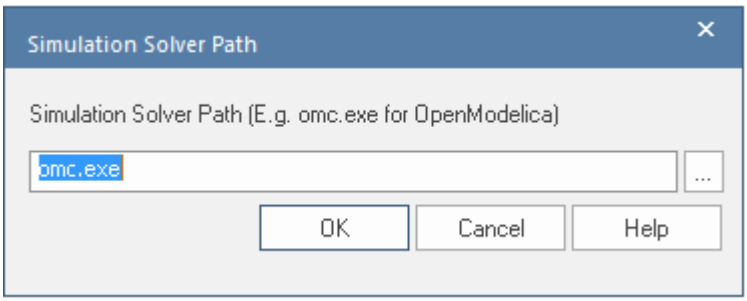
Accéder

Utilisez l'un de ces chemins d'accès pour afficher la dialogue « Chemin Solveur Simulation » afin de configurer le Solveur .

Méthode	Sélectionner
Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager >  > Configure Simulation Solveur
Autre	Double-cliquez sur un artefact avec le stéréotype SysMLSimConfiguration >  > Configure Simulation Solveur

Configurer le Solveur

Pour Windows , la dialogue « Chemin Solveur Simulation » ressemble à ceci :



Type ou recherchez le chemin d'accès au Solveur OpenModelica à utiliser dans Enterprise Architect .

OpenModelica sur Linux

Si Enterprise Architect est installé sur Linux, il est nécessaire de l'utiliser avec OpenModelica installé sur la même plateforme. L'installation d'OpenModelica Linux est documentée publiquement pour Debian et Ubuntu ; cependant, elle peut également être installée sous Linux Mint.

Cette rubrique d'aide fournit guidage sur :

1. Installation d'OpenModelica sur :
   - Linux Debian / Ubuntu
   - Linux Mint
2. Configuration Enterprise Architect pour accéder à OpenModelica.

Linux Debian / Ubuntu

Pour installer OpenModelica sur un système Linux Debian / Ubuntu, reportez-vous à l'URL :

https://openmodelica.org/download/download-linux

Ceci fournit les instructions pour Paquetages Debian / Ubuntu.

Exécuter ces scripts sur un terminal.

Étape	Action
1	Pour ajouter OpenModelica à votre liste de référentiels supplémentaires : pour deb dans deb deb-src ; faire echo "$deb http://build.openmodelica.org/apt `lsb_release -cs` nightly" ; fait | sudo tee /etc/apt/sources.list.d/openmodelica.list
2	Importez la clé GPG utilisée pour signer les versions : wget -q http://build.openmodelica.org/apt/openmodelica.asc -O- | sudo apt-key add -
3	Mettre à jour et installer OpenModelica : sudo apt-get update sudo apt-get installe openmodelica sudo apt-get install omlib-.* # Installe les bibliothèques Modelica facultatives (la plupart n'ont pas été testées avec OpenModelica)
4	Pour vérifier cette installation, assurez-vous que vous pouvez trouver le fichier /usr/bin/omc en exécutant, par exemple, cette commande sur le terminal : • ~ $ /usr/bin/omc --version Votre installation est réussie si la commande renvoie une string ressemblant à ceci : • OpenModelica 1.13.0~dev-1322-g53a43cf

Linux Mint

Pour installer OpenModelica sur Linux Mint, vous effectuez d'abord une installation pour Ubuntu, puis modifiez le nom de code Linux Mint pour qu'il corresponde au nom de code Ubuntu.

Il s'agit d'une liste de mappages du nom de code Linux Mint vers le nom de code Ubuntu (à utiliser dans les étapes ultérieures) :

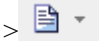
- Linux Mint 17.3 (Rosa) = Ubuntu 14.04 (Trusty) : **rosa** = **trusty**
- Linux Mint 18 (Sarah) = Ubuntu 16.04 (Xenial) : **sarah** = **xenial**
- Linux Mint 18.1 (Serena) = Ubuntu 16.04 (Xenial) : **serena** = **xenial**
- Linux Mint 18.2 (Sonya) = Ubuntu 16.04 (Xenial) : **Sony** = **xenial**
- Linux Mint 18.3 (Sylvia) = Ubuntu 16.04 (Xenial) : **sylvia** = **xenial**
- Linux Mint 19 (Tara) = Ubuntu 18.04 (Bionic) : **tara** = **bionic**

Cliquezhere pour une liste complète de l'historique de Linux Mint et des mappages avec Ubuntu.

Étape	Action
1	Exécuter ce script dans un terminal : pour deb dans deb deb-src ; faire echo "$deb http://build.openmodelica.org/apt `lsb_release -cs` nightly" ; fait | sudo tee /etc/apt/sources.list.d/openmodelica.list
2	Pour modifier l'URL du référentiel dans Linux Mint : • Sur l'écran principal de Linux Mint, sélectionnez : 'Menu | Barre de recherche | Sources de logiciels (saisissez le mot de passe) | Référentiels supplémentaires | Sélectionnez « Openmodelica » | Modifier l'URL' • Remplacez le nom de Linux Mint (par exemple, *rosa)* par le nom d'Ubuntu correspondant (par exemple, *trusty*) comme dans la liste en haut de ce tableau ; c'est-à-dire : deb http://build.openmodelica.org/apt **rosa** nocturne deb http://build.openmodelica.org/apt **confiance** nocturne • Cliquez sur le bouton OK
3	• Sélectionnez « Openmodelica (Sources) » | Modifier l'URL • Modifiez le nom de Linux Mint selon la liste en haut du tableau Par exemple, remplacez le nom Linux Mint *rosa* par le nom Ubuntu correspondant *trusty* • Cliquez sur le bouton OK
4	Pour mettre à jour et installer OpenModelica, exécuter ces scripts dans un terminal : sudo apt-get update sudo apt-get installe openmodelica sudo apt-get install omlib-.* # Installe les bibliothèques Modelica facultatives (la plupart n'ont pas été testées avec OpenModelica)

Accéder

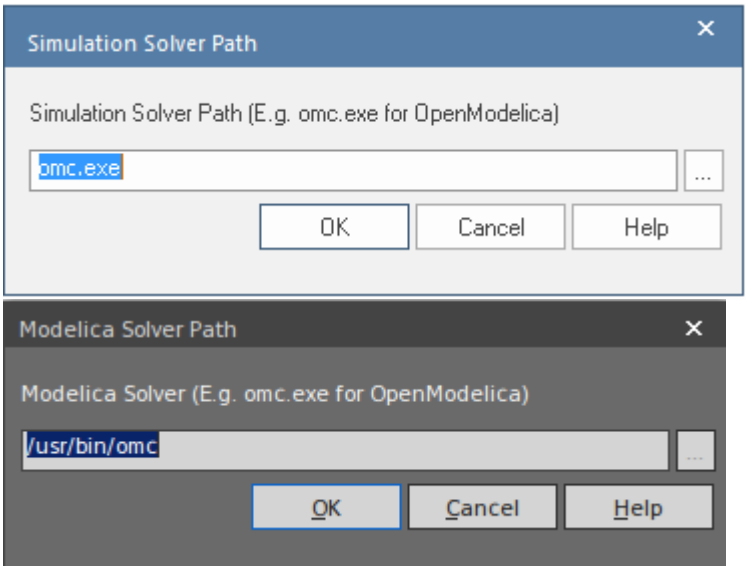
Utilisez l'un de ces chemins d'accès pour afficher la dialogue « Chemin Solveur Simulation » afin de configurer le solveur.

Méthode	Sélectionner
Ruban	Simulate > Comportement du Système > Modelica/Simulink > SysMLSim Configuration Manager >  > Configure Simulation Solveur
	Double-cliquez sur un artefact avec le stéréotype SysMLSimConfiguration

Autre	>  > Configure Simulation Solveur

Configurer le Solveur

La dialogue « Chemin Solveur Simulation » ressemble à ceci :

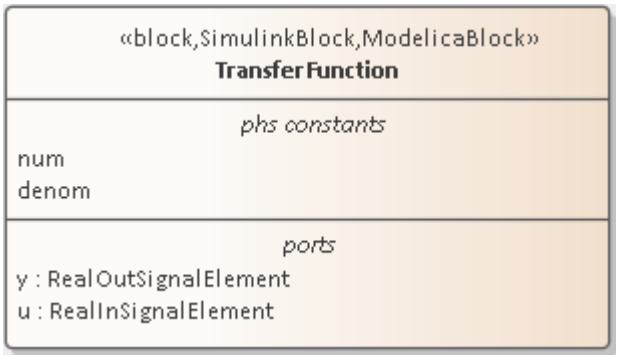


Type ou recherchez le chemin d'accès au solveur à utiliser.

Simulation SysPhS

La spécification *SysML Extension for Physical Interaction and Signal Flow Simulation* (SysPhS) est une spécification de Object Management Group (OMG) qui étend SysML pour fournir une plate-forme modélisation commune permettant de définir des modèles cohérents. Ces modèles peuvent être traduits vers l'une des deux principales plates-formes de simulation, Modelica et Simulink/Simscape de MATLAB.

Les stéréotypes OMG SysPhS vous aident à définir les caractéristiques de la simulation du modèle au sein même du modèle, plutôt que dans une spécification de configuration de simulation. Ils offrent une meilleure visibilité du type d' object ou de propriété dans la fenêtre Navigateur et la fenêtre Propriétés , ainsi que dans le diagramme avec des compartiments d'éléments spécifiques pour les types de propriétés et pour les valeurs initiales.



La norme est représentée dans Enterprise Architect par le profil OMG SysPhS, ainsi que :

- Bibliothèques SysPhS d'éléments pour le flux de signaux et pour l'interaction physique (nécessaires pour effectuer des simulations selon la norme SysPhS)
- Une page dédiée à la boîte à outils
- Une large gamme de motifs d'éléments de composants à partir desquels générer des éléments de simulation courants tels que des composants électroniques, logiques et fluides ; les motifs font référence aux blocs de bibliothèque dans les bibliothèques standard OpenModelica ou Simulink
- Fonctionnalités pour simuler des tracés à l'aide de Modelica ou de Simulink, Simscape et Stateflow de MATLAB.

Fonctionnalités de SysPhS

Fonctionnalité	Description
Référencement des bibliothèques SysPhS	Les principales ressources pour travailler avec SysPhS sont les bibliothèques Simulation SysPhS, qui incluent des ressources réutilisables que vous devez référencer dans votre modèle.
Boîte à outils SysPhS	Les pages SysPhS de la boîte à outils Diagramme contiennent des éléments SysML de base pour OpenModelica et MATLAB Simulink.
Motifs SysPhS	Les Motifs SysPhS fournissent des blocs SysPhS prédéfinis qui font référence à des composants MATLAB et Modelica équivalents. Ces blocs simples peuvent être utilisés comme points de départ lorsque vous travaillez avec des modèles SysPhS.
Composants SysPhs	Les composants SysPhS vous permettent de définir des références aux composants Modelica et Simulink.
Simulation	Vous pouvez définir des modèles IBD ou Paramétriques avec des informations supplémentaires pour piloter une simulation, puis utiliser la configuration de simulation pour générer le modèle dans Modelica, Simulink ou Simscape, afin de

	produire un graphique des résultats.
Exemples de SysPhS	Il existe plusieurs exemples d'utilisation de SysPhS pour la configuration de simulations.
Mise à jour de SysMLSim pour SysPhys	Vous pouvez mettre à jour les anciennes configurations de simulation (antérieures à Enterprise Architect 15.2) pour refléter l'utilisation de la norme SysPhS.

Options

Les options supplémentaires pour les variables et les constantes, telles que isContinuous et isConserved, sont automatiquement définies comme Valeur Étiquetés , évitant ainsi de devoir les définir dans la spécification de configuration. Ces options sont également visibles sur le Bloc lui-même et dans la fenêtre Propriétés ancrée.

Vidéos

- SysPhS Motifs for the Simulation of an Electrical Circuit
- Simulating Digital Electronics using SysPhS and Modelica

SysML Simulation Paramétrique

Enterprise Architect fournit une intégration avec OpenModelica et MATLAB Simulink pour support une évaluation rapide et robuste du comportement d'un modèle SysML dans différentes circonstances.

Les bibliothèques OpenModelica sont des ressources complètes qui fournissent de nombreux types, fonctions et modèles utiles. Lors de la création de modèles SysML dans Enterprise Architect , vous pouvez référencer les ressources disponibles dans ces bibliothèques.

L'intégration MATLAB d' Enterprise Architect se connecte via l'API MATLAB, permettant à vos simulations Enterprise Architect et autres scripts d'agir en fonction de la valeur de toutes les fonctions et expressions MATLAB disponibles. Vous pouvez appeler MATLAB via une classe Solveur ou exporter votre modèle vers MATLAB Simulink, Simscape et/ou Stateflow.

fonctionnalités de SysML Simulation

Ces sections décrivent le processus de définition d'un modèle Paramétriques , l'annotation du modèle avec des informations supplémentaires pour piloter une simulation et l'exécution d'une simulation pour générer un graphique des résultats.

Section	Description
Introduction aux modèles Paramétriques SysML	Les modèles SysML Paramétriques support l'analyse technique des paramètres critiques du système, notamment l'évaluation des mesures clés telles que les performances, la fiabilité et d'autres caractéristiques physiques. Ces modèles combinent les modèles Exigences avec les modèles de conception de système, en capturant les contraintes exécutables basées sur des relations mathématiques complexes. diagrammes Paramétriques sont diagrammes Bloc internes spécialisés qui vous aident, en tant que modélisateur, à combiner des modèles de comportement et de structure avec des modèles d'analyse technique tels que les modèles de performances, de fiabilité et de propriétés de masse. Pour plus d'informations sur les concepts des modèles SysML Paramétriques , reportez-vous au site officiel OMG SysML et à ses sources liées.
Création d'un Modèle Paramétrique	Un aperçu du développement d'éléments de modèle SysML pour la simulation, de la configuration de ces éléments dans la fenêtre Configurer Simulation SysML et de l'observation des résultats d'une simulation.
Artefact de configuration SysMLSim	Enterprise Architect vous aide à étendre l'utilité de vos modèles SysML Paramétriques en les annotant avec des informations supplémentaires qui permettent de simuler le modèle. Le modèle résultant est ensuite généré sous forme de modèle pouvant être résolu (simulé) à l'aide de MATLAB Simulink ou d'OpenModelica. Les propriétés de simulation de votre modèle sont stockées dans un artefact Simulation . Cela préserve votre modèle d'origine et supporte plusieurs simulations configurées par rapport à un seul modèle SysML. L'artefact Simulation se trouve sur la page de la boîte à outils « Artefacts ».
Interface Utilisateur	L'interface utilisateur de la simulation SysML est décrite dans la rubrique *Configurer la fenêtre Simulation SysML* .
Analyse Modèle à l'aide d'un ensemble de données	En utilisant la configuration Simulation un Bloc SysML peut avoir plusieurs jeux de données définis par rapport à lui. Cela permet d'exécuter des variations répétables sur une simulation du modèle SysML.
Support de la norme	La *norme SysPhS* est une *extension SysML pour Simulation d'interaction physique*

SysPhS	*et de flux de signaux* . Elle définit une méthode standard de traduction entre un modèle SysML et un modèle Modelica ou un modèle Simulink/Simscape, offrant ainsi une méthode plus simple basée sur un modèle pour le partage de simulations. Consultez la rubrique d'aide *Support de la norme SysPhS* .
Exemples	Pour vous aider à comprendre comment créer et simuler un modèle SysML Paramétriques , trois exemples ont été fournis pour illustrer trois domaines différents. Ces trois exemples utilisent les bibliothèques OpenModelica. Ces exemples et ce que vous pouvez en apprendre sont décrits dans la rubrique *Exemples Simulation SysML* .

Modélisation et Simulation avec OpenModelica Bibliothèque

Cette fonctionnalité est disponible à partir de la version 14.1 Enterprise Architect .

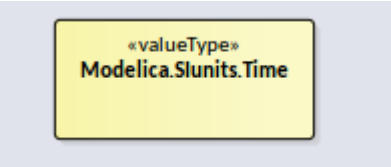
Les bibliothèques OpenModelica sont des ressources complètes qui fournissent de nombreux types, fonctions et modèles utiles. Lors de la création de modèles SysML dans Enterprise Architect , vous pouvez référencer les ressources disponibles dans les bibliothèques OpenModelica.

Référencement d'un Type défini dans Bibliothèque OpenModelica

Pour configurer une simulation pour référencer une Bibliothèque OpenModelica, créez d'abord un élément ValueType pointant vers la bibliothèque OpenModelica et enregistrez-le dans la configuration Simulation .

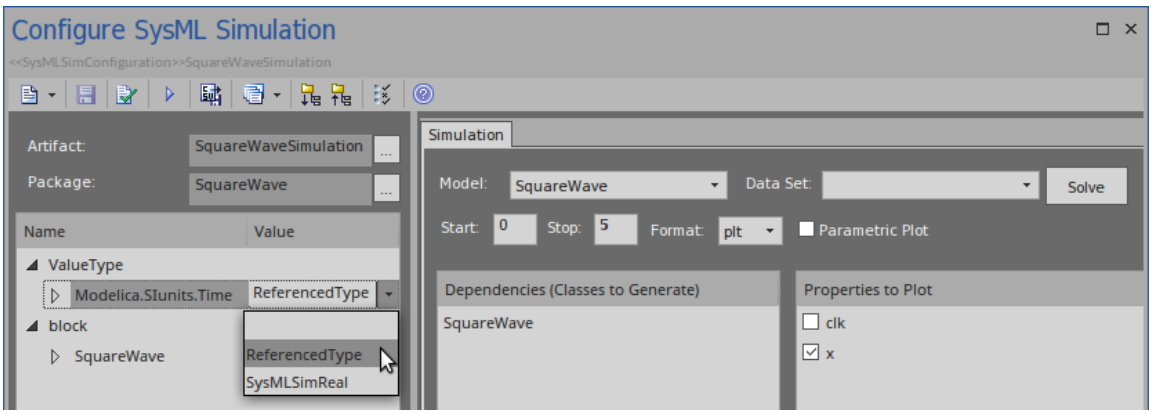
Créer un élément pour un Type OpenModelica référencé

- Créez un élément ValueType avec le nom complet du chemin de la Bibliothèque OpenModelica



Configurer l'élément ValueType comme « ReferencedType » :

- Double-cliquez sur l'élément SysMLSimConfiguration pour ouvrir l'onglet « Configurer la configuration SysML »
- Accéder à l'élément ValueType
- Dans le champ déroulant, définissez la valeur sur « ReferencedType »



Comme l'élément ValueType est configuré comme « ReferencedType », l'élément ne s'affichera pas dans la liste « Dépendances » et ne sera pas généré en tant que nouvelle définition de classe dans le fichier OpenModelica.

Définissez le type d'une propriété sur l'élément ValueType

Dans Enterprise Architect , une propriété SysML peut être définie comme un type primitif ou un élément tel qu'un Bloc ou un ValueType.

Option 1 :

- Sélectionnez la propriété (partie ou port)
- Appuyez sur Ctrl+2 pour ouvrir la fenêtre Propriétés
- Passez à l'onglet « Propriété » et choisissez « Sélectionner Type … »
- Accédez à l'élément ValueType que vous avez créé

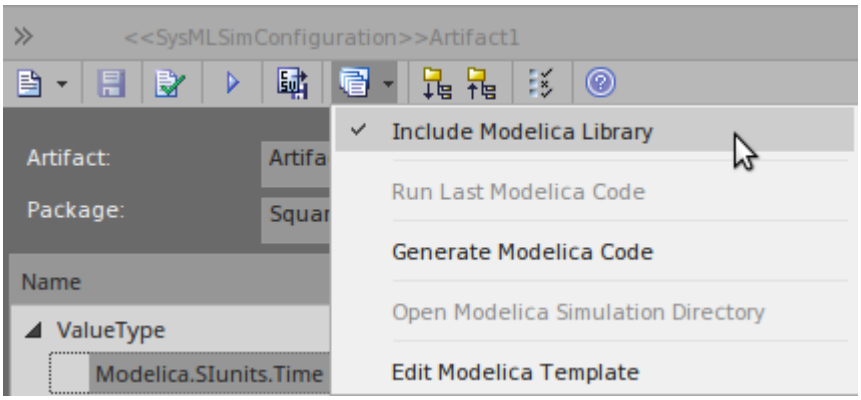
Option 2 :

- Sélectionnez la propriété (partie ou port)
- Appuyez sur Ctrl+L sur la propriété
- Accédez à l'élément ValueType que vous avez créé

Inclure une Bibliothèque OpenModelica dans une Simulation

Lorsque vous utilisez des types référencés à partir d'une bibliothèque OpenModelica dans un modèle, vous devez charger le modèle OpenModelica dans l'environnement pour que la simulation fonctionne.

- Développez l'option de menu et sélectionnez « Inclure Bibliothèque Modelica »



- Si cette option est cochée, cette fonction sera générée dans « Solve.mos » par défaut :
  chargerModel(Modelica);

Cliquez here pour une description détaillée de la fonction de script loadModel().

Personnaliser le script OpenModelica Gabarit

Vous pouvez modifier le gabarit du script OpenModelica pour ajouter des bibliothèques supplémentaires requises par le modèle et la simulation. Sélectionnez l'option du ruban :

Développer > Code source > Options > Modifier le code Gabarits

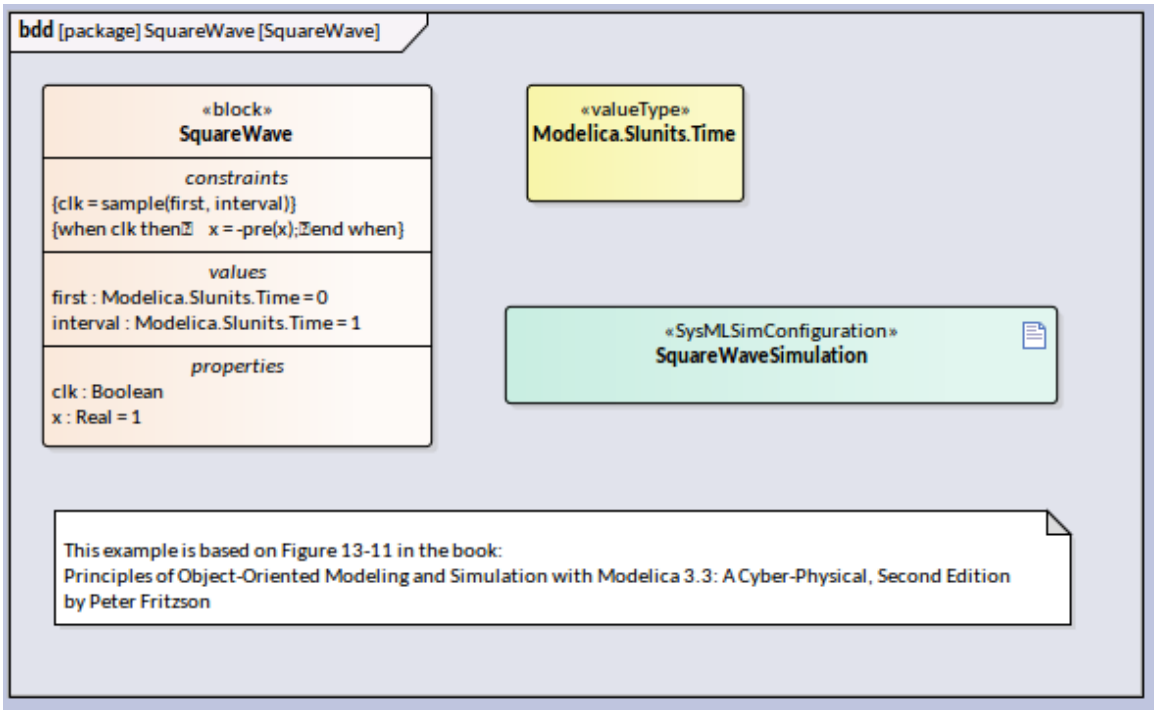
Dans le champ « Langue », sélectionnez « Modelica » et dans la liste « Scripts », sélectionnez « Script SysMLSim ».

```
10 %if sysmlSim_IncludeLibrary == "T"%
11 msg := "---------------- Loading Modelica Model... ----------------";
12 loadModel(Modelica);
13 %endIf%
```

Lorsque vous ajoutez des bibliothèques supplémentaires après « loadModel(Modelica) », les ressources des bibliothèques peuvent être référencées par votre modèle.

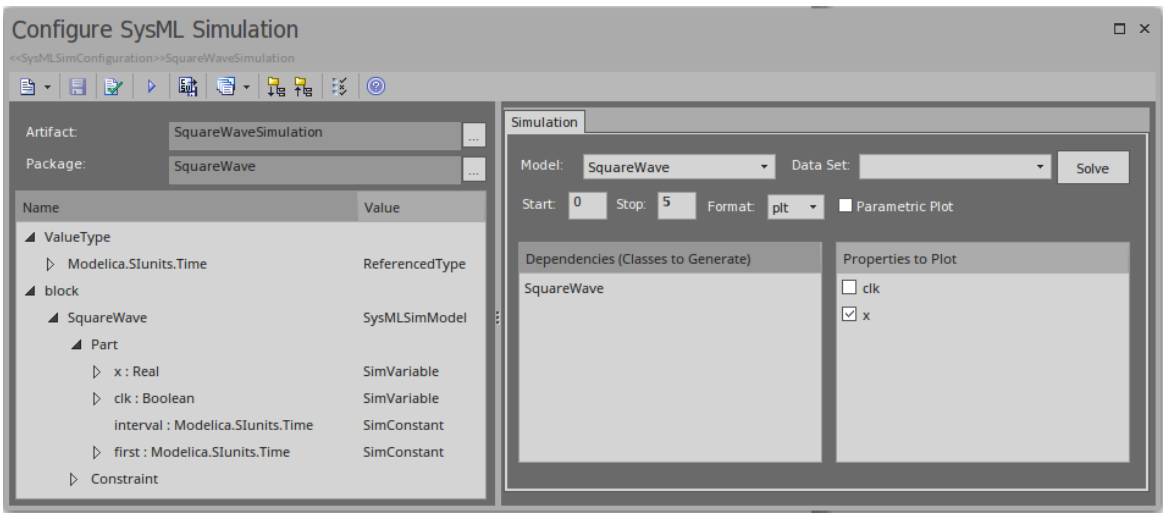
Exemple de SquareWave

Cet exemple est basé sur la figure 13-11 dans : *Principles of Object-Oriented Modélisation and Simulation with Modelica 3.3: A Cyber-Physical* , deuxième édition, par Peter Fritzson.

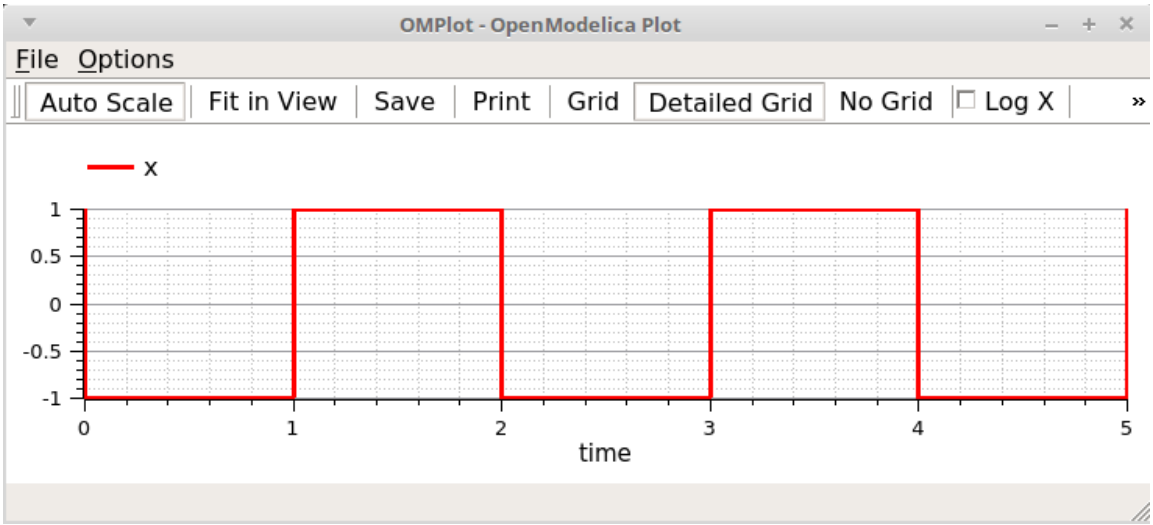


Dans cet exemple :

- Nous créons un ValueType *Modelica.SIunits.Time* , qui est utilisé pour les propriétés *first* et *interval* du Bloc *SquareWave*
- ValueType *Modelica.SIunits.Time* est configuré comme « ReferencedType » dans la fenêtre Simulation SysML
- Sélectionnez l'élément de menu « Inclure Bibliothèque Modelica »

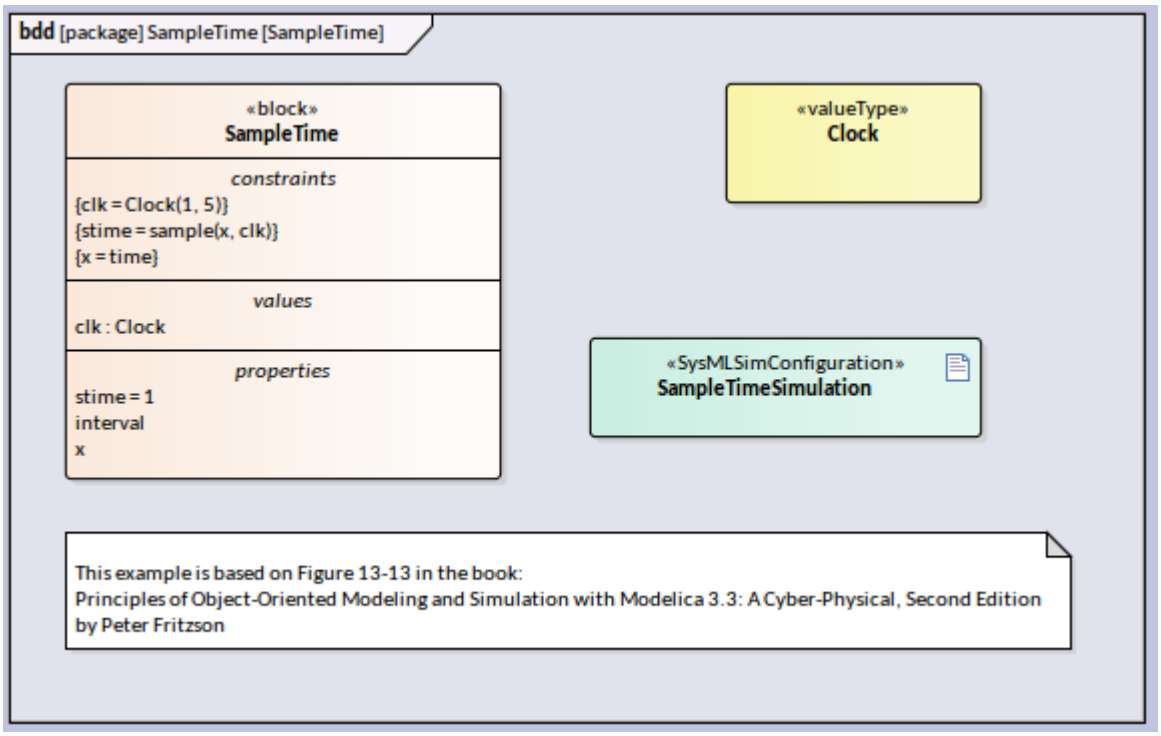


Exécuter la simulation ; la variable *x* est tracée comme ceci :



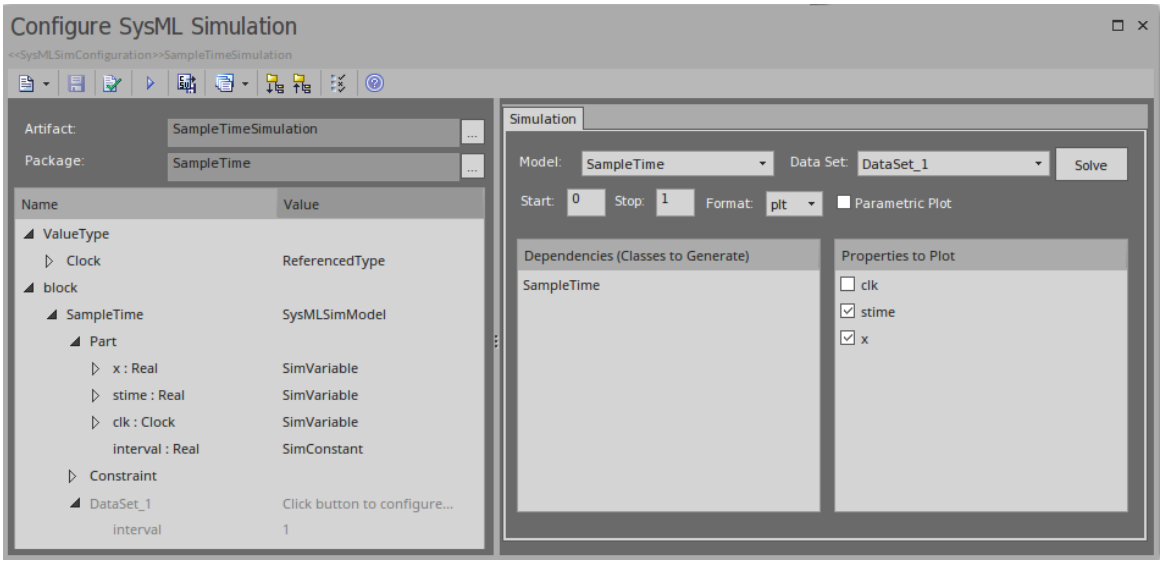
Exemple de SampleTime

Cet exemple est basé sur la figure 13-13 dans : *Principles of Object-Oriented Modélisation and Simulation with Modelica 3.3: A Cyber-Physical* , deuxième édition, par Peter Fritzson.

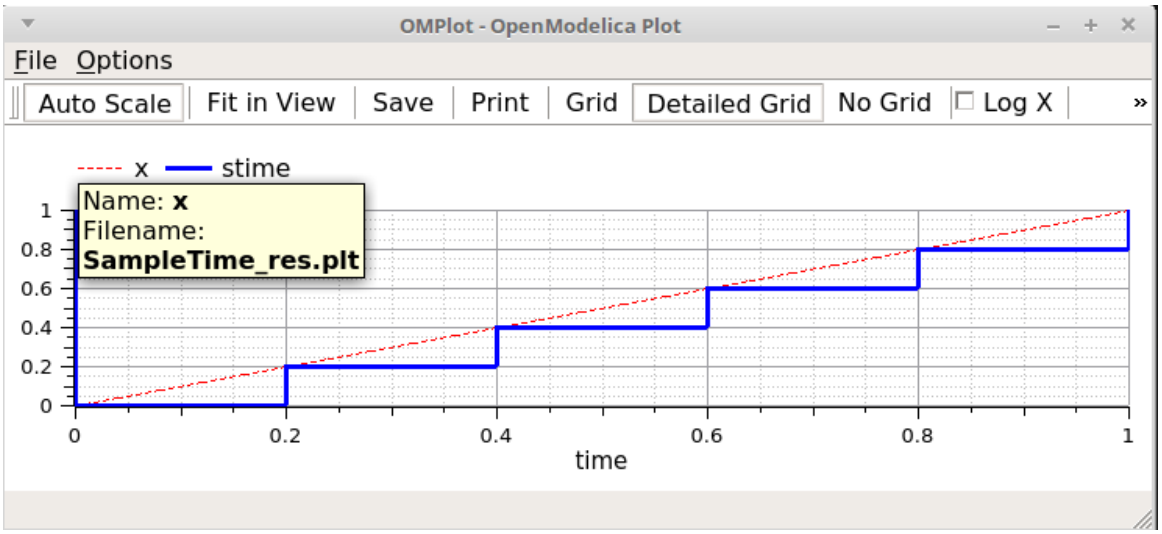


Dans cet exemple :

- Nous avons créé une *horloge* ValueType, qui est utilisée pour la propriété *clk* du Bloc *SampleTime*
- *L'horloge* ValueType est configurée comme « ReferencedType » dans la fenêtre Simulation SysML
- L'élément de menu « Inclure Bibliothèque Modelica » n'est pas sélectionné



Exécuter la simulation ; le tracé de la variable *x* et *stime* ressemble à ceci :



Dépannage de Simulation OpenModelica

Bien que cette rubrique décrive les problèmes possibles qui peuvent survenir lors de l'utilisation d'OpenModelica, de nombreux points s'appliquent également à la réalisation d'une simulation à l'aide de MATLAB Simulink.

Problèmes Simulation courants

Ce tableau décrit certains problèmes courants qui peuvent empêcher la simulation d'un modèle lors de l'utilisation d'OpenModelica. Vérifiez la sortie dans l'onglet « Build » de la fenêtre Sortie système. Les messages sont extraits du compilateur OpenModelica (omc.exe), qui vous renvoie normalement aux lignes du code source d'OpenModelica. Cela vous aidera à localiser la plupart des erreurs.

Problème
Le nombre d'équations est inférieur au nombre de variables. Vous avez peut-être oublié de définir certaines propriétés sur « PhSConstant », ce qui signifie que la valeur ne change pas pendant la simulation. Vous devrez peut-être fournir les valeurs de la propriété « PhSConstant » avant le démarrage de la simulation. (Définissez les valeurs via un ensemble de données Simulation .)
Les blocs qui saisissent des données dans les ports peuvent ne pas contenir de propriétés conservées. Par exemple, un Bloc « ChargePort » contient deux parties : « v : Voltage » et « i : Current ». La propriété « i : Current » doit être définie comme PhSVariable avec l'attribut « isConserved » défini sur « True ».
Les PhSConstants peuvent ne pas avoir de valeurs par défaut : elles doivent leur être fournies.
Une variable PhS peut ne pas avoir de valeur initiale pour commencer — une doit être fournie.
Les propriétés peuvent être typées par des éléments (blocs ou Type de valeur) externes au Paquetage configuré ; utilisez un connecteur d'importation Paquetage pour résoudre ce problème.

Filtres de configuration Simulation SysML

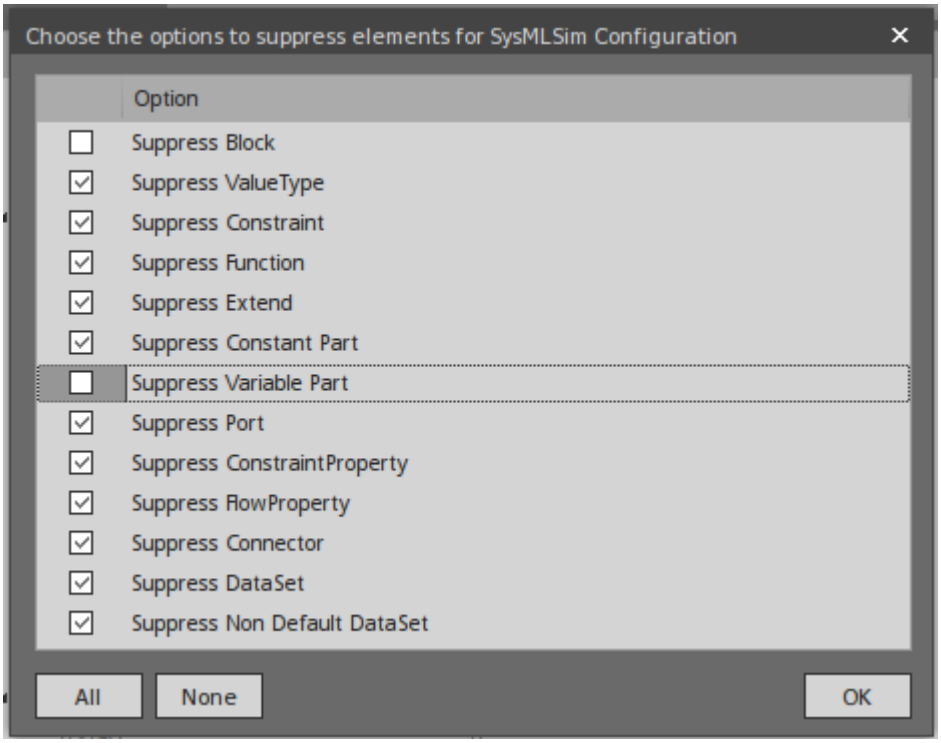
La dialogue « Configuration Simulation SysML » affiche tous les éléments du Paquetage par défaut, notamment les types de valeur, les blocs, les blocs de contrainte, les parties et les ports, Propriétés de contrainte, les connecteurs, les contraintes et Ensembles de données. Pour un modèle de taille moyenne, la liste complète peut être assez longue et il peut être difficile de trouver une erreur modélisation potentielle.

Dans l'exemple TwoTanks, si nous effaçons la propriété Tank.area 'PhSConstant' puis effectuons une validation, nous trouverons cette erreur :

Erreur : Trop peu d'équations, système sous-déterminé. Le modèle comporte 11 équation(s) et 13 variable(s).

Cette erreur indique que nous avons peut-être oublié de définir certaines propriétés pour « PhSConstant ».

Ce que nous pouvons faire maintenant est de cliquer sur le deuxième bouton à droite de la barre d'outils (Filtre pour la configuration) et d'ouvrir le dialogue illustrée ici. Cliquez sur le bouton Tout, puis décochez les cases « Supprimer Bloc » et « Supprimer la partie variable » et cliquez sur le bouton OK .



Nous aurons maintenant une liste de variables beaucoup plus courte, à partir de laquelle nous pourrons constater que « area » ne change pas pendant Simulation . Nous définissons ensuite cela comme une « PhSConstant » et fournissons une valeur initiale pour résoudre le problème.

Exemples de validation Modèle

Message	Discussion
Variable non définie dans la contrainte	Dans l'exemple TwoTanks, lorsque nous accédons à « constraintBlock.Outcontrol.Constraint », supposons que nous trouvions une erreur de frappe : nous avons tapé « v » au lieu de « b » dans la contrainte. Donc, au lieu de : *a=b*(c+d)* Nous avons tapé : *a=v*(c+d)* Cliquez sur le bouton Valider de la barre d'outils. Ces messages d'erreur apparaîtront dans l'onglet « Modelica » : *Validation du modèle...* *Erreur : variable v introuvable dans la portée OutControl. (Expression : « a=v*(c+d); ")* *Erreur : une erreur s'est produite lors de l'aplatissement du modèle TanksConnectedPI* *Nombre d'erreurs et d'avertissements détectés : 2* Double-cliquez sur la ligne d'erreur ; la liste de configuration s'affiche avec la contrainte en surbrillance. Remplacez « v » par « b » et cliquez à nouveau sur le bouton Valider. Aucune erreur n'est détectée et le problème est résolu.

	Conseils : L'utilisation de la vue Configuration Simulation SysML est un moyen rapide de modifier les contraintes d'un Bloc ou Bloc de contraintes. Vous pouvez : • Modifier une contrainte en place • Supprimer à l'aide du menu contextuel d'une contrainte • Ajouter une nouvelle contrainte en utilisant le menu contextuel d'un Bloc ou Bloc de Contrainte
Noms de variables en double	Dans l'exemple TwoTanks, accédez à *constraintProperty* . Supposons que nous ayons donné le même nom à deux propriétés : • Cliquez-droit sur *e1* , sélectionnez l'option ' Rechercher dans Projet Navigateur ', et changez le nom en *e2 ;* recharger la dialogue 'SysML Simulation Configuration' Cliquez sur le bouton Valider de la barre d'outils ; ces messages d'erreur apparaissent dans l'onglet « Modelica » : *Validation du modèle...* *Erreur : Les éléments en double (en raison d'éléments hérités) ne sont pas identiques : (Expression : "SensorValue e2 ; ")* *Erreur : une erreur s'est produite lors de l'aplatissement du modèle TanksConnectedPI* *Nombre d'erreurs et d'avertissements détectés : 2* Double-cliquez sur la ligne d'erreur ; la liste de configuration s'affiche avec les propriétés de contrainte mises en surbrillance. Modifiez le nom de l'un d'eux de *e2* à *e1* et cliquez à nouveau sur le bouton Valider ; aucune erreur n'est trouvée et le problème est résolu.
Propriétés définies dans ConstraintBlocks non utilisées	Dans l'exemple TwoTanks, dans la fenêtre Navigateur , nous naviguons jusqu'à l'élément 'Exemple Modèle . Ingénierie des Systèmes .ModelicaExamples.TwoTanks.constraints.OutFlow'. Supposons que nous ajoutions une propriété « *c* » et potentiellement une nouvelle contrainte, mais que nous oublions de synchroniser les instances - les ConstraintProperties. Cela provoquera une erreur *système Too few equations, under-determined* si nous exécuter pas la validation. Rechargez le Paquetage dans la dialogue « Configuration Simulation SysML » et cliquez sur le bouton Valider dans la barre d'outils. Ces messages d'erreur apparaîtront dans l'onglet « Modelica » : *Validation du modèle...* *Erreur : la propriété ConstraintProperty « e4 » ne contient pas les paramètres définis dans le bloc de contrainte de saisie « OutFlow ». (Manquant : c)* *Erreur : Trop peu d'équations, système sous-déterminé. Le modèle comporte 11 équation(s) et 12 variable(s).* *Nombre d'erreurs et d'avertissements détectés : 2* Double-cliquez sur la ligne d'erreur ; la liste de configuration s'affiche avec la propriété ConstraintProperty en surbrillance. La propriété ConstraintProperty est typée sur *outFlow* et le nouveau paramètre ' *c* ' est manquant. Cliquez-droit sur la propriété ConstraintProperty dans la liste de configuration, sélectionnez l'option 'Rechercher dans tous Diagrammes ' et cliquez-droit sur la propriété 'Contrainte' sur le diagramme ; sélectionnez ' Fonctionnalités | Parts / Propriétés ' et cochez la case 'Afficher les propriétés possédées / héritées', puis cliquez sur 'c'. Rechargez le modèle dans la dialogue « Configuration Simulation SysML » et cliquez sur le bouton Valider. Ces messages d'erreur apparaîtront dans l'onglet « Modelica » :

	Validation du modèle... *Erreur : ConstraintProperty 'e4' n'a aucun connecteur de liaison entrant ou sortant pour le paramètre 'c'.* *Erreur : Trop peu d'équations, système sous-déterminé. Le modèle comporte 11 équation(s) et 12 variable(s).* *Nombre d'erreurs et d'avertissements détectés : 2* Afin de résoudre ce problème, nous pouvons faire l'une des deux choses suivantes en fonction de la logique réelle : 1. Si la propriété « c » est nécessaire dans le ConstraintBlock et qu'une contrainte est définie à l'aide de « c », nous devons alors ajouter une propriété dans le contexte de ConstraintProperty et la lier au paramètre « c ». 2. Si la propriété « c » n'est pas requise, nous pouvons cliquer sur cette propriété dans le Bloc de contraintes et appuyer sur les touches Ctrl+D. (Les propriétés de contrainte correspondantes auront « c » automatiquement supprimé.)