ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Guide de Simulation et Comportement

Author: Sparx Systems & Stephen Maguire
Date: 7/11/2024
Version: 17.0

CRÉÉ AVEC ENTERPRISE ARCHITECT

Table des Matières

Introduction

Enterprise Architect is a platform that provides a unique set of developer tools integrated into a sophisticated development environment. In addition to the standard facilities available to work with static code, there is a wide range of facilities that support the simulation and automatic code generation from the models.

In an era dominated by digital disruption and the need for organizations to be agile and responsive to dynamic business changes, these facilities provide an effective insurance policy that will ensure an organization becomes a true digital enterprise.

Enterprise Architect holds a unique position and, regardless of whether you use Enterprise Architect's standard code engineering features or another Integrated Development Environment such as Eclipse or Visual Studio, these facilities provide productivity gains not matched in other tools.

Model Simulation brings your behavioral models to life with instant, real-time behavioral model execution. Coupled with tools to manage triggers, events, guards, effects, breakpoints and simulation variables, plus the ability to visually track execution at run-time, the Simulator is a valuable means of 'watching the wheels turn' and verifying the correctness of your behavioral models. With Simulation you can explore and test the dynamic behavior of models. In the Corporate, Unified and Ultimate Editions, you can also use JavaScript as a run-time execution language for evaluating guards, effects and other script-able pieces of behavior.

With extensive support for triggers, trigger sets, nested states, concurrency, dynamic effects and other advanced simulation capabilities, Enterprise Architect provides a remarkable environment in which to build interactive and working models that help explore, test and visually trace complex business, software and system behavior. With JavaScript enabled, it is also possible to create embedded COM objects that will do the work of evaluating guards and executing effects - allowing the simulation to be tied into a much larger set of dependent processes. For example, a COM object evaluating a guard condition on a State Transition might query a locally running process, read and use a set of test data, or even connect to an SOA web service to obtain some current information.

As Enterprise Architect uses a dynamic, script driven Simulation mechanism there is no need to generate code or compile your model before running a simulation. It is even possible to update simulation variables in real time using the Simulation console window. This is useful for testing alternative branches and conditions 'on the fly', either at a set Simulation break point or when the Simulation reaches a point of stability (for example, when the Simulation is 'blocked').

In the Professional Edition of Enterprise Architect, you can manually walk through simulations - although no JavaScript will execute - so all choices are manual decisions. This is useful for testing the flow of a behavioral model and highlighting possible choices and processing paths.

Simulations Dynamiques

Simulation de Modèle donne vie à vos modèles comportementaux grâce à une exécution instantanée et en temps réel. Associé à des outils de gestion déclencheurs , des événements, des gardes, des effets, des points d'arrêt et des variables Simulation , ainsi qu'à la possibilité de suivre visuellement l'exécution au moment de l'exécution, le simulateur est un moyen polyvalent de « regarder les roues tourner » et de vérifier l'exactitude de vos modèles comportementaux. Avec Simulation vous pouvez explorer et tester le comportement dynamique des modèles. Dans les éditions Corporate , Unified et Ultimate , vous pouvez également utiliser JavaScript comme langage d'exécution au moment de l'exécution pour évaluer les gardes, les effets et d'autres éléments de comportement scriptables.

support étendue des déclencheurs , des ensembles déclencheur , des états imbriqués, de la concurrence, des effets dynamiques et d'autres fonctionnalités Simulation avancées fournit un environnement remarquable dans lequel créer des modèles interactifs et fonctionnels qui aident à explorer, tester et tracer visuellement le comportement complexe des entreprises, des logiciels et des systèmes. Avec JavaScript activé, il est également possible de créer des objets COM intégrés qui effectueront le travail d'évaluation des gardes et d'exécution des effets, ce qui permet à la Simulation d'être liée à un ensemble beaucoup plus vaste de processus dépendants. Par exemple, un object COM évaluant une condition de garde sur une transition State peut interroger un processus exécuté localement, lire et utiliser un ensemble de données de test, ou même se connecter à un service Web SOA pour obtenir des informations actuelles.

Comme Enterprise Architect utilise un mécanisme Simulation dynamique piloté par script qui analyse et utilise directement les constructions UML , il n'est pas nécessaire de générer du code intermédiaire ou de compiler des « exécutables » de simulation avant d'exécuter une Simulation . Il en résulte un environnement Simulation très rapide et dynamique dans lequel des modifications peuvent être apportées et testées rapidement. Il est même possible de mettre à jour les variables Simulation en temps réel à l'aide de la fenêtre Console Simulation . Cela est utile pour tester des branches et des conditions alternatives « à la volée », soit à un point d'arrêt Simulation défini, soit lorsque la Simulation atteint un point de stabilité (par exemple, lorsque la Simulation est « bloquée »).

Dans l'édition Professional d' Enterprise Architect , vous pouvez parcourir manuellement les simulations (bien qu'aucun JavaScript ne soit exécuté), de sorte que tous les choix sont des décisions manuelles. Cela est utile pour tester le flux d'un modèle comportemental et mettre en évidence les choix et les chemins de traitement possibles. Dans les éditions Corporate , Unified et Ultimate il est possible de :

- Exécutez dynamiquement vos modèles comportementaux
- Évaluer les protections et les effets écrits en JavaScript standard
- Définir et déclencher déclencheurs dans les simulations en cours d'exécution
- Définir et utiliser des ensembles de déclencheurs pour simuler différentes séquences d'événements
- Ensembles déclencheur à déclenchement automatique pour simuler des historiques d'événements complexes sans intervention de l'utilisateur
- Mettre à jour les variables Simulation « à la volée » pour modifier le déroulement des simulations
- Créez et appelez des objets COM pendant une Simulation pour étendre la portée et les possibilités d'entrée/sortie de Simulation
- Inspecter les variables Simulation au moment de l' exécuter
- Définir un « prologue » de script pour définir des variables, des constantes et des fonctions avant l'exécution
- Utilisez plusieurs Scripts d'Analyseur avec différents « prologues » pour exécuter la Simulation dans un large éventail de conditions

Dans les éditions Unified et Ultimate , il est également possible de simuler des modèles BPMN.

En utilisant le simulateur Modèle , vous pouvez simuler l'exécution de modèles conceptuels contenant un comportement. Lorsque vous démarrez une Simulation , le modèle actuel Paquetage est analysé et un processus Simulation dynamique est déclenché pour exécuter le modèle.

Pour démarrer avec Simulation , les seules étapes requises sont :

- Construire un diagramme comportemental (State ou Activité pour exécution manuelle ou dynamique, Séquence pour interaction manuelle uniquement)
- Facultatif : charger la disposition « Espace de travail Simulation » - un moyen rapide d'afficher toutes les fenêtres Simulation fréquemment utilisées

- Cliquez sur le bouton Jouer au simulateur

Si le diagramme contient des éléments externes (ceux qui ne sont pas dans le même Paquetage que le diagramme), vous devrez créer un connecteur d'importation du Paquetage du diagramme vers le Paquetage contenant les éléments externes. Pour cela, faites glisser les deux Paquetages de la fenêtre Navigateur sur un diagramme , puis utilisez la flèche Quick Linker pour créer le connecteur entre eux.

Présentation Simulation

Aspect
Présentation du simulateur Modèle
Utilisation de la fenêtre Simulation et Windows associées, et exécution d'une Simulation
Configurer une Simulation et activer un script Simulation
Configurer et utiliser Points d'Arrêt Simulation
Simuler l'utilisation des objets
L'utilisation de différents types d' Action dans Simulation
Effectuer Simulation dynamique avec JavaScript
L'utilisation des Gardes et Effets dans les simulations
L'utilisation des Déclencheurs dans les simulations
Comportements d'Appels et variables
Simulation des retours d'activité
Simulation du comportement d'une activité structurée
Simulation de Processus multithread
Simulation de sous-processus dans Diagrammes séparés
Réalisation de simulations BPMN
Simuler le comportement Dialogue Win32

Plateformes et éditions disponibles

Plateforme/Édition	Détails
Modèles et plateformes pris	Le Modèle Simulator prend actuellement supporte l'exécution des modèles d'activité, d'interaction et Statemachine UML et Processus Métier BPMN sur les

en charge	plateformes Simulation : • UML de base • BPMN
Support des éditions	Simulation de Modèle est disponible à différents niveaux dans la gamme des éditions d' Enterprise Architect : • Professional - Simulation manuelle uniquement • Corporate et supérieur - Ajoute une évaluation JavaScript dynamique ; JavaScript est actuellement activé pour Statemachines et les graphiques d'activité ; il n'est pas activé pour diagrammes d'interaction • Unified et Ultimate - Ajoute Simulation BPMN

À quoi ça ressemble

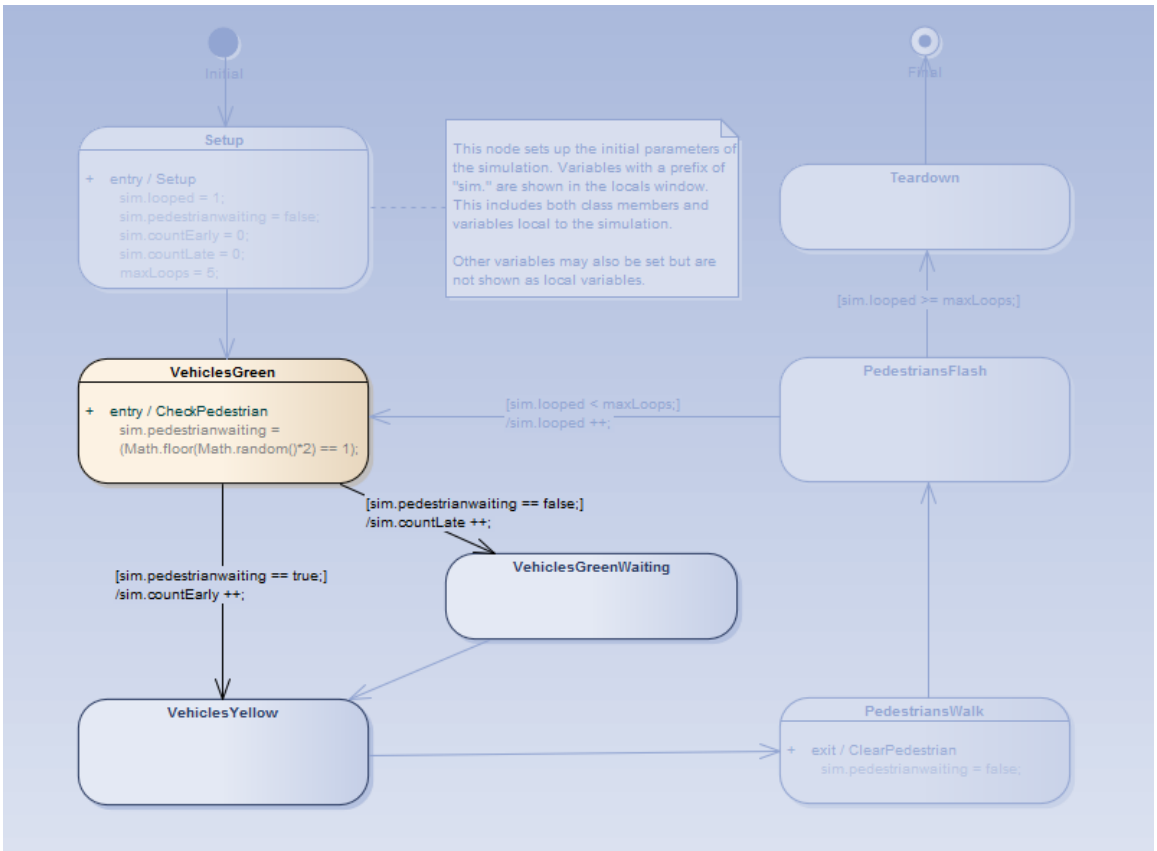
Enterprise Architect dispose d'une méthode spéciale pour afficher les informations du modèle pendant Simulation . Cela permet de concentrer l'attention sur les nœuds en cours d'exécution ou actifs.

Pendant une Simulation , Enterprise Architect suit et met en surbrillance de manière dynamique les nœuds actifs de votre modèle. Si un nœud d'un autre diagramme est activé, ce diagramme est automatiquement chargé et le nœud actuel est mis en surbrillance. Il est possible de modifier le diagramme pendant l'exécution de la Simulation . Toutefois, les modifications apportées ne sont pas reconnues tant que la Simulation en cours n'est pas terminée et qu'une nouvelle n'est pas démarrée.

Mise en évidence du ou des nœuds Actif pendant Simulation

Dans cet exemple, le nœud actuellement actif (VehiclesGreen) est mis en surbrillance dans les couleurs normales Enterprise Architect et toutes les transitions possibles hors du nœud actuel sont rendues à pleine puissance.

Les éléments qui sont des cibles possibles des transitions sortantes du nœud actif actuel sont rendus dans un style semi-estompé afin qu'ils soient lisibles et clairement différents des autres éléments du diagramme . Tous les autres éléments sont rendus dans un style entièrement estompé pour montrer qu'ils ne sont pas des cibles de l'étape Simulation suivante. Au fur et à mesure que la Simulation progresse (en particulier si elle exécuter automatiquement), cette mise en évidence permet de concentrer l'attention sur l'élément actuel et son contexte visuel.



Fenêtres de Simulation

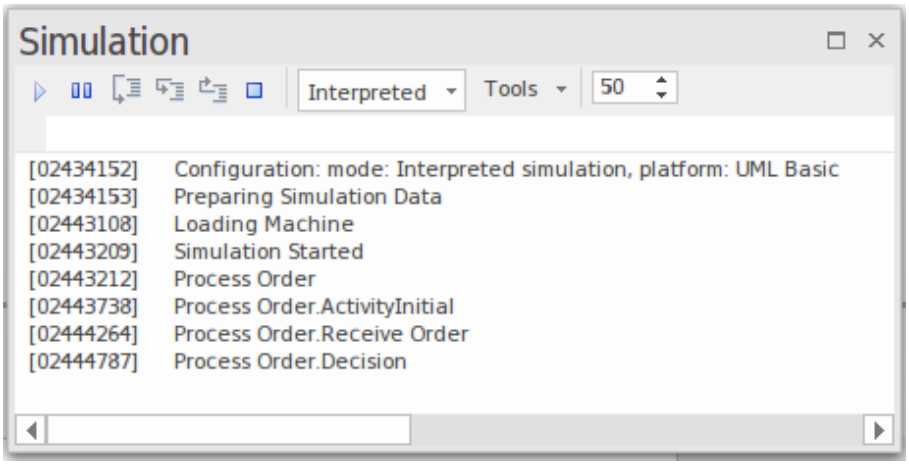
Lors de l'exécution d'une Simulation dans Enterprise Architect il est possible de définir des points d'arrêt, de déclencher déclencheurs , d'examiner des variables, d'enregistrer une trace d'exécution, de définir la vitesse Simulation , d'afficher la Pile d'Appel et de tracer visuellement les nœuds actifs au fur et à mesure du déroulement de la Simulation .

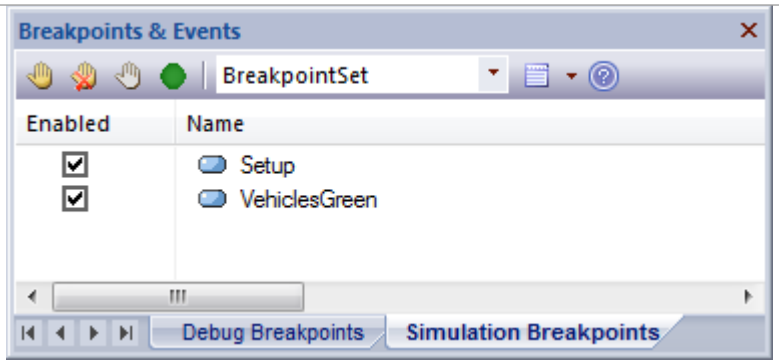
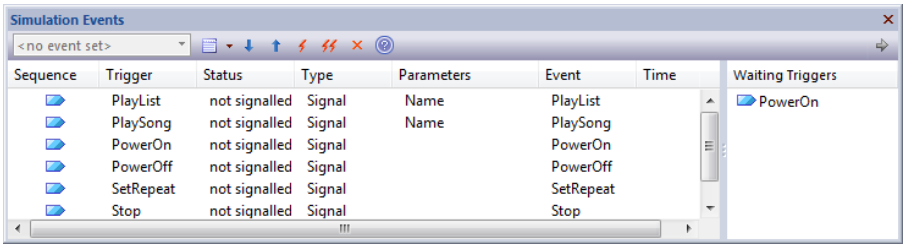
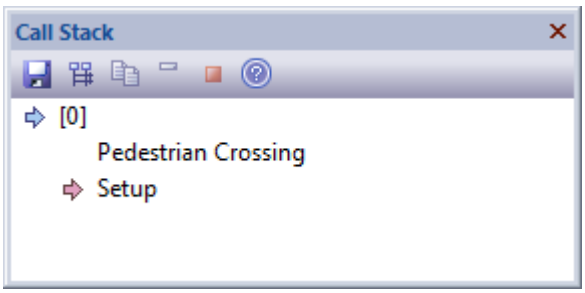
Lorsqu'une Simulation s'exécute, certains aspects tels que la sortie et l'entrée de la console se trouvent dans la fenêtre Simulation elle-même, tandis que d'autres aspects tels que les variables locales et Pile d'Appel utilisent les fenêtres standard Analyseur d'Exécution . La rubrique fournit un aperçu des principales fenêtres utilisées pendant Simulation .

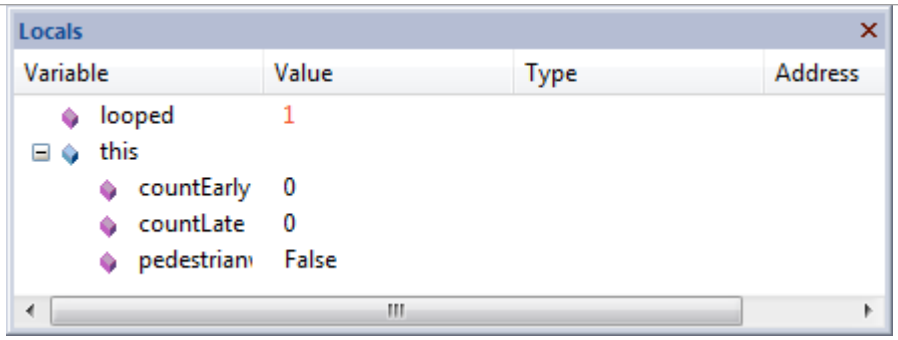
Accéder

Ruban	Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation

Windows

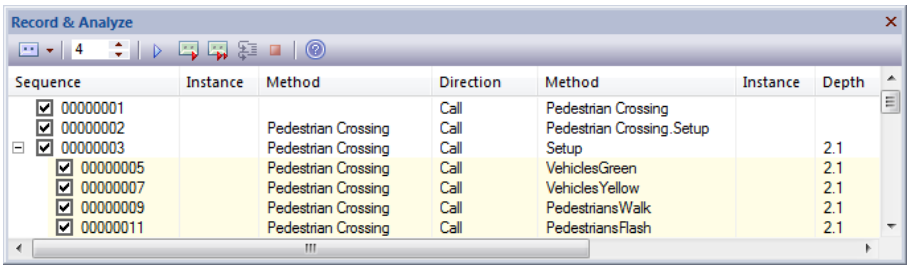
Fenêtre	But
Exécution et console	La fenêtre Simulation fournit l'interface principale pour démarrer, arrêter et parcourir votre Simulation . Pendant l'exécution, elle affiche les résultats relatifs à l'étape en cours d'exécution et d'autres informations importantes. Consultez la rubrique *Exécuter Simulation de Modèle* pour plus d'informations sur les commandes de la barre d'outils. Note la zone de saisie de texte juste en dessous de la barre d'outils. Il s'agit de la zone de saisie de la console. Vous pouvez y saisir des commandes JavaScript simples telles que : *this.count = 4;* pour modifier dynamiquement une variable de simulation nommée « count » à 4. De cette façon, vous pouvez influencer dynamiquement la simulation au moment de l'exécution. 
Vitrine Points d'Arrêt & Événements	Le processus Simulation utilise également l'onglet « Points d'Arrêt Simulation » de la fenêtre Points d'Arrêt et Marqueurs ('Simuler > Simulation Dynamique > Points d'Arrêt '). Ici, vous définissez des points d'arrêt d'exécution sur des éléments et des messages spécifiques dans une Simulation . Voir la rubrique *Points d'Arrêt Simulation* pour plus de détails.

	
Fenêtre Simulation Événements	La fenêtre Simulation Événements ('Simulate > Simulation Dynamique > Événements ') fournit des outils pour gérer et exécuter déclencheurs . Déclencheurs sont utilisés pour contrôler l'exécution des transitions Statemachine . 
Fenêtre de Pile d'Appel	Pendant la Simulation la fenêtre Pile d'Appel ('Simuler > Simulation Dynamique > Pile d'Appel ') affiche des informations sur les threads et le contexte d'exécution actuel de la Simulation . Le simulateur supporte les simulations multithread et inclura une entrée de thread pour chaque thread d'exécution actif et en pause. Pour chaque thread, la fenêtre Pile d'Appel affichera le contexte de départ ou d'entrée (tel qu'un élément Statemachine) ainsi que l'élément actif actuel dans ce thread. Si l'élément actif actuel est le point d'entrée d'un état composite Activity ou SubMachine, la pile inclura également l'élément actif actuel dans ce sous-contexte (et tous les autres sous-états composites actifs imbriqués). 
Fenêtre de variable locale Simulation	Le simulateur utilise la fenêtre Variables locales standard ('Simuler > Simulation Dynamique > Variables locales') pour afficher toutes les variables de simulation actuelles lorsque la simulation est en cours pas à pas ou interrompue à un point d'arrêt. Note qu'il est possible de mettre à jour dynamiquement ces variables à l'aide de la console du simulateur.



Enregistrement

Pendant l'exécution de votre simulation, un enregistrement de toutes les activités est conservé et affiché dans la fenêtre Enregistrer et analyser ('Exécuter > Outils > Enregistreur > Ouvrir Enregistreur '). Cela fonctionne de manière similaire à l'enregistrement normal des appels dans l' Analyseur d'Exécution Visuelle .



Configurer Script de Simulation

Vous pouvez utiliser Scripts Simulation pour contrôler précisément le démarrage d'une Simulation . En général, vous n'avez pas besoin de configurer de script Simulation sauf si :

- Vous souhaitez exécuter une Simulation interprétée qui nécessite que les variables soient initialisées avant le début de la Simulation ; cela est utile pour configurer des variables globales et définir des fonctions
- (Dans l'édition Corporate et supérieure) Vous ne souhaitez pas appliquer le comportement par défaut d'interprétation des gardes (c'est-à-dire que vous préférez utiliser une exécution manuelle), ou
- Vous souhaitez disposer de plusieurs façons d'exécuter le même diagramme

Pour la plupart diagrammes , il est possible d'initialiser un script pour une Simulation simplement en définissant des variables dans le premier élément ou connecteur après l'élément Démarrer . Pour les diagrammes State , il s'agit du connecteur Transit sortant de l'élément initial, et pour les modèles d'activité, il s'agit du premier élément Action .

Vous pouvez également utiliser Scripts Simulation pour initialiser les paramètres avant le démarrage d'une Simulation . Cela est utile pour configurer différents ensembles de valeurs initiales à l'aide de plusieurs Scripts d'Analyseur , afin de pouvoir exécuter votre Simulation dans une gamme de conditions prédéfinies.

Pour configurer un script Simulation , sélectionnez d'abord le Paquetage dans la fenêtre Navigateur , Paquetage Navigateur , Liste Diagramme ou Recherche Modèle . Vous pouvez ensuite utiliser la fenêtre Analyseur d'Exécution pour ajouter un nouveau script pour ce Paquetage sélectionné. Vous utiliserez la page ' Simulation ' de la dialogue ' Analyseur d'Exécution ' pour configurer les propriétés correspondantes.

Accéder

Affichez la fenêtre Analyseur d'Exécution en utilisant l'une des méthodes décrites ici.

Dans la fenêtre Analyseur d'Exécution , soit :

- Localisez et double-cliquez sur le script requis et sélectionnez la page « Simulation » ou
- Cliquez sur  dans la barre d'outils de la fenêtre et sélectionnez la page « Simulation »

Ruban	Develop > Source Code > Analyseur d'Exécution > Edit Scripts d'Analyseur Exécuter > Outils > Analyseur
Menu Contexte	Fenêtre Navigateur | Cliquez-droit sur Paquetage | Analyseur d'Exécution
Raccourcis Clavier	Maj+F12

Configurer un script Simulation

Option	Action
Plate-forme	Pour la simulation d'activité, d'interaction ou Statemachine UML , cliquez sur la flèche déroulante et sélectionnez « UML Basic ». Pour diagrammes BPMN, cliquez sur la flèche déroulante et sélectionnez « BPMN ».
Point d'entrée	Cliquez sur le bouton  et sélectionnez :

	• Point d'entrée pour la Simulation , et • Activité, interaction ou Statemachine à simuler Si vous ne spécifiez pas de point d'entrée, le simulateur tente de parcourir l'intégralité Paquetage .
Évaluer Gardes et Effets à l'aide JavaScript	(Dans les éditions Corporate et supérieures) Laissez la case à cocher décochée pour effectuer une Simulation manuelle, dans laquelle vous sélectionnez l' State suivant vers lequel effectuer la transition et le point où une décision doit être prise. Cochez la case pour exécuter le code du comportement de l'effet dans la Simulation . La Simulation exécute le code JavaScript à ces endroits : • Entrée/sortie/opérations d' State • Garde/effet de transition • Conditions de boucle d'activité BPMN et expressions de condition de Flux séquence À l'exception de la garde, tous ces éléments doivent être une ou plusieurs instructions JavaScript valides, y compris le point-virgule. La garde doit être une expression booléenne valide, également terminée par un point-virgule. Les variables membres de « sim » ou « this » sont répertoriées dans la fenêtre Variables locales lorsqu'un point d'arrêt Simulation est atteint. sim.compte = 0;
Saisir	Lorsque JavaScript est activé, vous pouvez saisir des commandes de script dans ce champ qui s'exécuteront avant l' exécuter de la Simulation .
Script de post-traitement	À l'aide d'un script de post Simulation , vous pouvez exécuter JavaScript une fois la Simulation terminée. Type le nom qualifié d'un script à partir du contrôle de script du modèle. Par exemple, si vous avez un script nommé « MyScript » dans le groupe de scripts « MyGroup », saisissez la valeur « MyGroup.MyScript ».
OK	Cliquez sur ce bouton pour enregistrer vos modifications.

Notes

- Habituellement, tous les éléments et relations Simulation résident dans le Paquetage configuré pour Simulation ; cependant, vous pouvez simuler diagrammes qui incluent des éléments de différents Paquetages , en créant des connecteurs d'importation Paquetage du Paquetage configuré vers chaque Paquetage « externe » (alternativement, pour un modèle BPSim, créez un connecteur de dépendance du Paquetage configuré vers chaque **élément** externe)

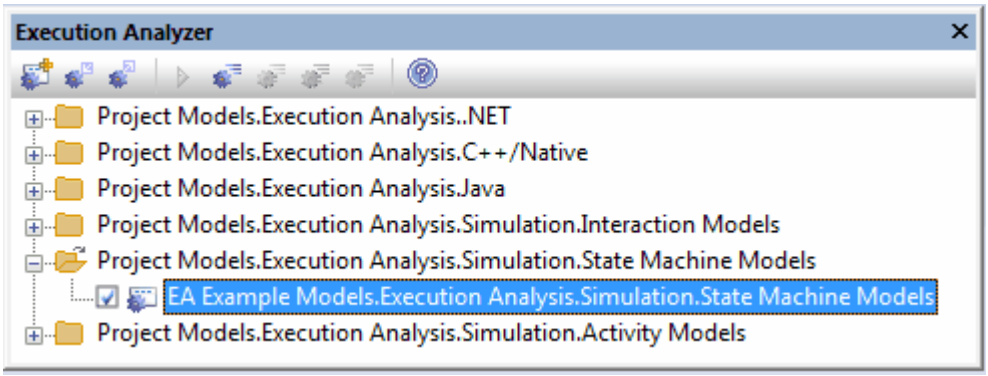
Activer Script de Simulation

Un script d'exécution est configuré pour un Paquetage de modèle définissant les paramètres Simulation . La raison la plus courante pour activer un script d'exécution est lorsque plusieurs Scripts Simulation sont configurés pour un Paquetage et que vous souhaitez exécuter un en particulier.

Accéder

Ruban	Exécuter > Outils > Analyseur Développer > Code Source > Analyseur d'Exécution
Fenêtre de l'analyseur	Cochez la case Analyzer Script pour la rendre active
Raccourcis Clavier	Maj+F12

Activer un script Simulation pour l'exécution

Étape	Action
1	Dans la fenêtre Analyseur d'Exécution , sélectionnez le script d'exécution requis. Il devient alors le script par défaut de votre modèle ouvert, de sorte qu'un clic sur le bouton Simulation Exécuter invoquera automatiquement ce script Simulation . 
2	Cliquez sur la case à cocher à gauche du script pour l'activer.
3	Sélectionnez l'option de ruban « Simuler > Simulateur > Ouvrir la fenêtre Simulation » pour exécuter la simulation.

Exécuter Simulation de Modèle

Une Simulation exécute le modèle étape par étape, ce qui vous permet de valider la logique de votre modèle comportemental. L'étape d'exécution en cours est automatiquement mise en surbrillance dans le diagramme du modèle pour faciliter la compréhension des différents processus et changements d'état au fur et à mesure qu'ils se produisent pendant la Simulation .

Il existe plusieurs manières de démarrer une Simulation de modèle :

- Lorsque le diagramme actif peut être simulé, le bouton Exécuter de la fenêtre principale Simulation traitera le diagramme actuel, soit en exécutant un script existant, soit en définissant un nouveau script temporaire
- Lorsque le diagramme actif ne peut pas être simulé, le bouton Exécuter de la fenêtre principale Simulation exécuter la Simulation pour le script Analyseur d'Exécution actif
- En cliquant avec le bouton droit sur un script Simulation dans la fenêtre Analyseur d'Exécution et en sélectionnant l'option ' Démarrer Simulation '
- En cliquant avec le bouton droit sur un diagramme approprié et en sélectionnant l'une des options « Exécuter Simulation »

Des repères visuels sont présents pendant l'exécution. Lorsque la Simulation est en cours d'exécution, Enterprise Architect met en évidence chaque nœud actif pour chaque étape exécutée. De plus, toutes les transitions sortantes et les flux de contrôle sont mis en évidence, indiquant les chemins possibles vers l'avant. Les éléments à la fin des chemins possibles vers l'avant sont atténués de moitié et tous les autres éléments restants sont « grisés » à 90 %. Cela permet une exécution très dynamique et facile à suivre qui recentre continuellement l'attention sur le contexte d'exécution.

Accéder

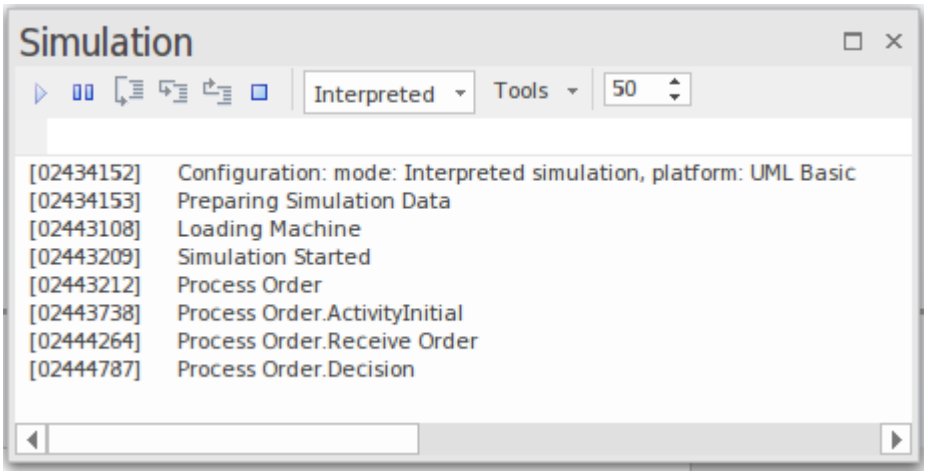
Ruban	Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation Simuler > Exécuter Simulation > Démarrer

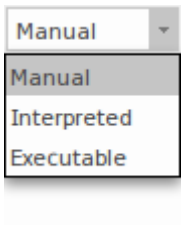
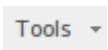
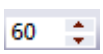
Détails spécifiques à l'édition

Dans l'édition Professional , si une branche est rencontrée lors de l'exécution, le simulateur vous prompts à choisir le chemin approprié à suivre dans votre exécution.

Dans les éditions Corporate , Unified et Ultimate , dans lesquelles JavaScript est activé, la Simulation évalue automatiquement toutes les protections et tous les effets et exécute la Simulation de manière dynamique sans intervention de l'utilisateur. Si la Simulation est bloquée en raison de l'absence de chemins possibles vers l'avant évalués à True (ou de plusieurs chemins évalués à True), vous pouvez modifier les variables Simulation à la volée à l'aide de l'entrée de la console de la fenêtre d'exécution Simulation .

Exécuter une Simulation à l'aide de la barre d'outils



Icône	Action
	Démarrer le Simulateur pour le diagramme courant ou, si le diagramme courant n'est pas simulable, exécuter la Simulation en utilisant le script Simulation activé.
	Mettre la Simulation en pause.
	Lorsque la Simulation est en pause, effectuez des pas en avant, des pas en avant et des pas en arrière pour contrôler l'exécution du simulateur à l'étape requise dans la Simulation du modèle.
	Arrêtez la Simulation .
	Cliquez sur la flèche déroulante et sélectionnez le type de Simulation à exécuter : • « Interprété » - Exécuter l'exécution dynamique d'une Simulation (éditions Corporate , Unified et Ultimate) • « Manuel » : parcourez une Simulation manuellement (la seule option disponible dans l'édition Professional) • « Exécutable » - Sélectionnez lors de l'exécution de la Simulation sur un Statemachine Exécutable
	Cliquez sur la flèche déroulante et sélectionnez dans un menu d'options pour effectuer des opérations spécifiques sur le script Simulation et la sortie, telles que Build, Exécuter , Générer et Vue Points d'Arrêt .
	Faire varier le taux d'exécution de la Simulation , entre 0% et 100% ; à : • 100 %, la Simulation s'exécute à la vitesse la plus rapide possible • 0% le simulateur interrompt l'exécution à chaque instruction

Notes

- L'outil Simulation ne devient actif que lorsqu'un script d'exécution Simulation valide est activé
- Vous pouvez définir un script Simulation comme valeur par défaut actuelle en cochant sa case dans la fenêtre Analyseur d'Exécution .

Simulation Points d'Arrêt

L'onglet ' Simulation Points d'Arrêt ' de la fenêtre Points d'Arrêt & Événements vous permet d'interrompre et d'inspecter le processus Simulation .

Lors de l'exécution dynamique d'une Simulation (dans les éditions Corporate , Unified et Ultimate), le processus se déroule automatiquement. Si vous souhaitez arrêter l'exécution à un moment donné pour examiner des variables, inspecter des piles d'appels ou interagir avec le simulateur, vous pouvez définir un point d'arrêt sur un élément de modèle de la même manière que vous le feriez avec une ligne de code source. Lorsque le simulateur atteint le point d'arrêt, l'exécution est interrompue et le contrôle est renvoyé à Enterprise Architect .

Accéder

Ruban	Simuler > Simulation Dynamique > Points d'Arrêt > Simulation Points d'Arrêt

Points d'Arrêt

La Simulation exécute le modèle étape par étape, vous permettant de valider la logique de votre modèle de comportement ; la Simulation s'arrête lorsqu'elle atteint un élément défini comme point d'arrêt.

Les éléments UML qui peuvent être définis comme points d'arrêt incluent les actions, les activités, States et la plupart des autres nœuds comportementaux tels que Décision , Initial ou Final.

Les relations UML qui peuvent être définies comme points d’arrêt incluent les messages d’interaction.

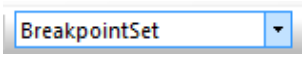
Les points d'arrêt sont stockés sous forme Ensembles Point d'Arrêt pour un projet Enterprise Architect donné.

Les éléments inclus dans une Simulation et qui ont des points d'arrêt sont signalés par un cercle vert près du coin supérieur gauche de l'élément, pendant que la Simulation est en cours. Si la Simulation n'est pas en cours d'exécution, les cercles verts ne s'affichent pas.

Lorsque JavaScript est activé, toutes les variables Simulation seront affichées dans la fenêtre Variables locales et il est possible de modifier ces variables Simulation à l'aide du champ de saisie de la console de la fenêtre Simulation (sous la barre d'outils).

Boutons de la barre d'outils

Item	Description
	Active tous les points d'arrêt définis dans l'ensemble de Point d'Arrêt actuel pour la session Simulation .
	Supprime tous les points d'arrêt définis dans l'ensemble de Point d'Arrêt actuel pour la session Simulation .
	Désactive tous les points d'arrêt défini dans l'ensemble Point d'Arrêt actuel pour la session Simulation .
	Ajoute un point d'arrêt pour l'élément sélectionné ou le message Séquence à l'ensemble Point d'Arrêt actuel.

	Modifie l'ensemble de Point d'Arrêt sélectionné à utiliser dans la session Simulation .
	Exécute les commandes Set Point d'Arrêt : • **Nouvel ensemble** : Créer un nouvel ensemble Point d'Arrêt • **Enregistrer sous l'ensemble** : enregistre l'ensemble Point d'Arrêt actuel sous un nouveau nom • **Supprimer Sélectionnée Set** : Supprime le Set Point d'Arrêt actuel • **Supprimer tous Ensembles** : Supprime tous Ensembles Point d'Arrêt enregistrés pour le diagramme

Objets et Instances dans Simulation

Lorsqu'une activité, un système ou un processus mécanique donné s'exécute, les activités et les actions qu'il contient peuvent générer des objets d'un type spécifique et effectuer des opérations sur ces objets, voire les consommer ou les détruire. Vous pouvez simuler la création, l'utilisation et la consommation de ces objets à l'aide d'un modèle Simulation qui représente les objets et les actions avec des éléments de modèle tels que des classes, des objets d'instance, des attributs, des opérations et des ports (ActionPins et ObjectNodes). Le modèle peut également créer, agir sur et détruire plusieurs objets différents à différentes étapes dans le cadre du même processus. La représentation des données ou des objets du modèle dans Simulation permet à Simulation de refléter plus précisément le processus réel.

Concepts Object

Terme	Description
Type de Sim	Le type d'élément Simulation , tel que Classe, Énumération ou Interface. Il peut s'agir de classificateurs d'objets dans une Simulation .
SimObject	Un object qui est une instance de (est classé par) un élément SimType.
Attribut	Une propriété d'un élément SimType ou d'un nœud spécifié tel qu'un ActivityNode.
Opération	Un comportement d'un élément SimType ou d'un nœud spécifié tel qu'un ActivityNode.
Port	Un port d'une classe ou Object , un ActionPin d'une Action ou un ObjectNode d'une activité. Les ports de classificateurs sont un type, tandis qu'un port d'un object est une réalisation du type.
Paramètre/ Paramètre d'activité	Paramètres des opérations ; Les paramètres d'activité sont, plus précisément, les paramètres des ActivityNodes.
Fente	Une réalisation d'un attribut dans un object . Un Slot possède une valeur de temps exécuter qui peut être initialisée par la valeur d'état exécuter du Slot. Si ces valeurs n'existent pas, le système utilise les valeurs initiales des attributs.
Environnement d'exécution	Tous les objets existent dans l'environnement d'exécution JavaScript , vous pouvez donc utiliser JavaScript pour créer ou modifier des objets de simulation et des variables de simulation.
Variables d'affichage	Tous les objets de simulation, les variables Simulation ou les événements sont identifiés dans la fenêtre Variables locales lorsqu'ils sont en vigueur. Dans certains cas, pour afficher les variables, vous devrez peut-être ajouter des points d'arrêt au modèle afin de suspendre le traitement pendant que la variable existe. Comme tous les objets et variables sont affichés, les variables globales qui existent en dehors de la simulation mais qui sont importantes pour elle (comme les éléments parent Class et Activity dans lesquels un processus est défini) sont également automatiquement représentées comme variables object par défaut. Il en va de même pour la sortie anticipée de l'activité, en tant que variable de retour.

Créer Objets dans une Simulation

Dans un modèle Simulation , vous pouvez créer des classes et soit créer des instances de celles-ci (objets globaux) pour représenter les objets qui existent dans le processus, soit définir des actions pour générer un ou plusieurs objets à tout moment pendant le processus.

Vous disposez de trois options pour créer des objets dans un modèle Simulation :

- Créer manuellement l' Object
- Créer dynamiquement un Object via un élément Action CreateObject
- Utilisez la fonction JavaScript sim.CreateObject (« nom ») comme « Effet » d'un élément Action , pour créer à nouveau un Object de manière dynamique

Après avoir créé un Object de manière dynamique, vous pouvez également instancier tous les objets internes de cet Object , comme une activité sur une classe, et agir sur les propriétés de cet object interne.

Créer un Object manuellement

Créez simplement un élément Object sur un diagramme dans le modèle, soit en :

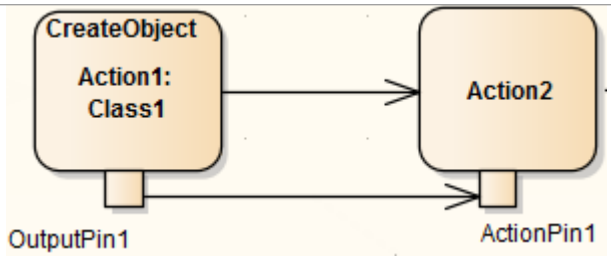
- Faire glisser un élément Object depuis les pages « Object » de la boîte à outils Diagramme et définir son classificateur, ou
- Faire glisser un élément de classificateur depuis la fenêtre Navigateur et le coller dans le diagramme en tant qu'instance

Dans le modèle Simulation , vous pouvez ensuite configurer les propriétés Object elles-mêmes (comme la définition des états d'exécution pour réinitialiser la valeur initiale d'un attribut) ou les comportements des actions pour agir sur l' Object (comme le transmettre le long d'un flux de processus) et observer ce qui arrive à l' Object dans une Simulation .

Créer un Object via une Action CreateObject

Si votre processus génère des objets lors de l'exécution, vous pouvez simuler cela à l'aide d'une Action CreateObject.

Étape	Action
1	Sur votre diagramme d'activité, faites glisser une icône « Action » depuis la boîte à outils Diagramme et sélectionnez l'option de menu contextuel « Autre | CreateObject » pour la définir comme un élément Action CreateObject.
2	Définissez le classificateur de l' Action CreateObject sur la classe dont l' Object sera une instance. Cette option est définie dans la fenêtre Propriétés > CreateObjectAction > Classificateur, à l'aide du bouton [...].
3	Créez une Action Pin sur l' Action CreateObject, de type sortie.
4	Créez ou sélectionnez l’ Action suivante dans la séquence de traitement et ajoutez une broche Action de type entrée. Connectez les deux actions avec un connecteur de flux de contrôle et les deux Pins Action avec un connecteur de flux Object .

	
5	Effectuez une Simulation sur le diagramme . Lorsque l' Action CreateObject est exécutée, elle crée un Object ayant les propriétés du classificateur et le stocke dans sa broche de sortie. L' Object lui-même est transmis via la connexion Object Flow à la broche d'entrée de Action 2, où ses propriétés peuvent être répertoriées dans la fenêtre Locals pour la Simulation .

Créer Object à l'aide de JavaScript

Vous pouvez également créer des objets Simulation de manière dynamique à l'aide d'une commande JavaScript dans le champ « Effet » de l'élément Action . La commande est :

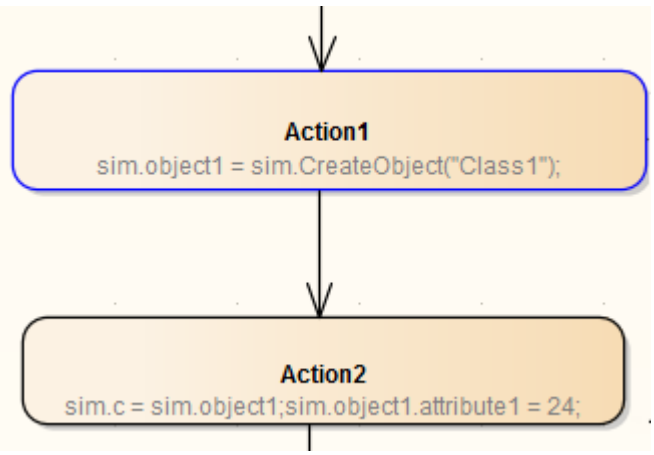
sim.newObject = sim.CreateObject("NomDeClasse");

ou

sim.newObject = new SimObject("ClassName"); (JavaScript naturel)

C'est-à-dire : « Simuler la création d'un Object basé sur la classe <nom> ». La classe classificatrice existerait dans le même Paquetage que l' Action .

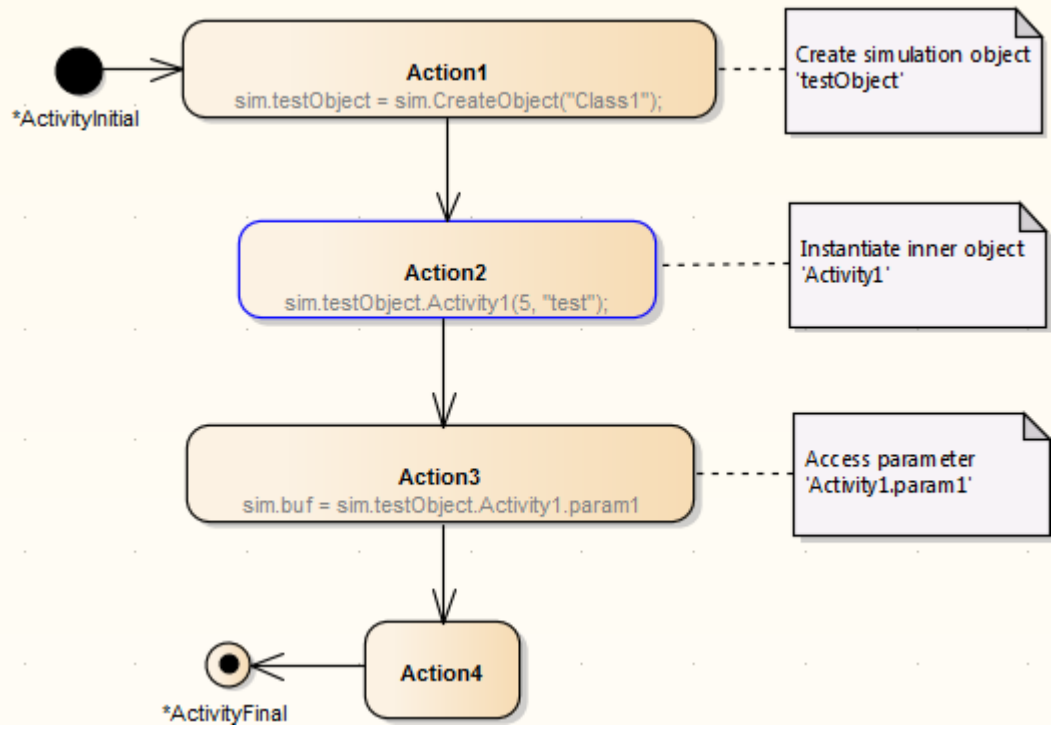
En ce qui concerne l'élément Action CreateObject, l' Object est créé pendant la Simulation et peut être transmis et traité par des éléments « en aval ». Dans cet exemple, l' Object créé est identifié comme sim.object1 et dans Action 2, il est accessible et l'un de ses attributs reçoit une valeur différente (également par JavaScript en tant qu'effet de l' Action).



Instancier des objets internes

Comme décrit précédemment, vous pouvez créer un Object à l'aide JavaScript ou d'une Action CreateObject. De même, vous pouvez instancier des objets internes à l'aide JavaScript ou d'une Action CallBehavior.

Dans cet exemple, à l'aide JavaScript , la Simulation crée d'abord un object de test basé sur la classe 1. La classe 1 comporte un élément Activity et diagramme , avec un paramètre Activity 1 défini sur l' integer 5 et un paramètre Activity 2 défini sur la string 'test'. La valeur du paramètre Activity 1 est capturée sous la forme d'une valeur tampon 'buf'.


*ActivityInitial
Action1
sim.testObject = sim.CreateObject("Class1");
Create simulation object 'testObject'
Action2
sim.testObject.Activity1(5, "test");
Instantiate inner object 'Activity1'
Action3
sim.buf = sim.testObject.Activity1.param1
Access parameter 'Activity1.param1'
Action4
*ActivityFinal

Détruire Objets dans une Simulation

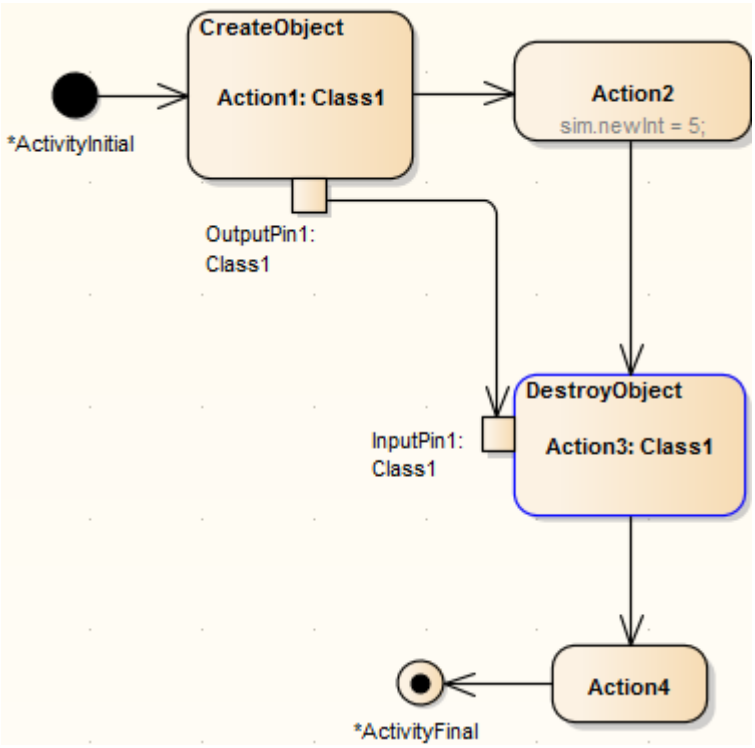
Après avoir créé ou généré des objets dans votre modèle Simulation , vous pouvez définir des actions pour détruire ces objets à tout moment du processus. Tous les objets Simulation sont détruits automatiquement une fois la Simulation terminée.

Vous avez deux options pour détruire les objets de votre modèle Simulation :

- Détruire dynamiquement les objets via un élément Action DestroyObject
- Détruire dynamiquement les objets à l'aide JavaScript dans un élément Action

Le résultat de la suppression peut être observé dans le changement des variables locales, dans la fenêtre Local.

Détruire un Object via une Action DestroyObject

Étape	Action
1	Sur votre diagramme d'activité, faites glisser une icône « Action » depuis la boîte à outils Diagramme et sélectionnez l'option de menu contextuel « Autre | DestroyObject » pour la définir comme un élément Action DestroyObject.
2	Définissez le classificateur de l' Action DestroyObject sur la classe dont l' Object est une instance. (Avancé | Classificateur d'ensembles). Créez une Action Pin sur l' Action DestroyObject, de type *input* .
3	Connectez la broche Action d'entrée à un connecteur de flux Object de la dernière Action ayant fonctionné sur l' Object . Dans cet exemple, la dernière Action ayant fonctionné sur l' Object est l' Action qui l'a créé. 
4	Effectuez une Simulation sur le diagramme . Le processus transmet le nom ou valeur Object à la broche Action d'entrée en tant que paramètre. Lorsque l' Action DestroyObject est exécutée, elle supprime l' Object portant ce nom ou valeur du modèle.

	Dans l'exemple, l'instance de Class1 est spécifiquement détruite avant le traitement d'Action4, mais les résultats d'Action2 ne sont pas affectés.

Détruire un Object à l'aide JavaScript

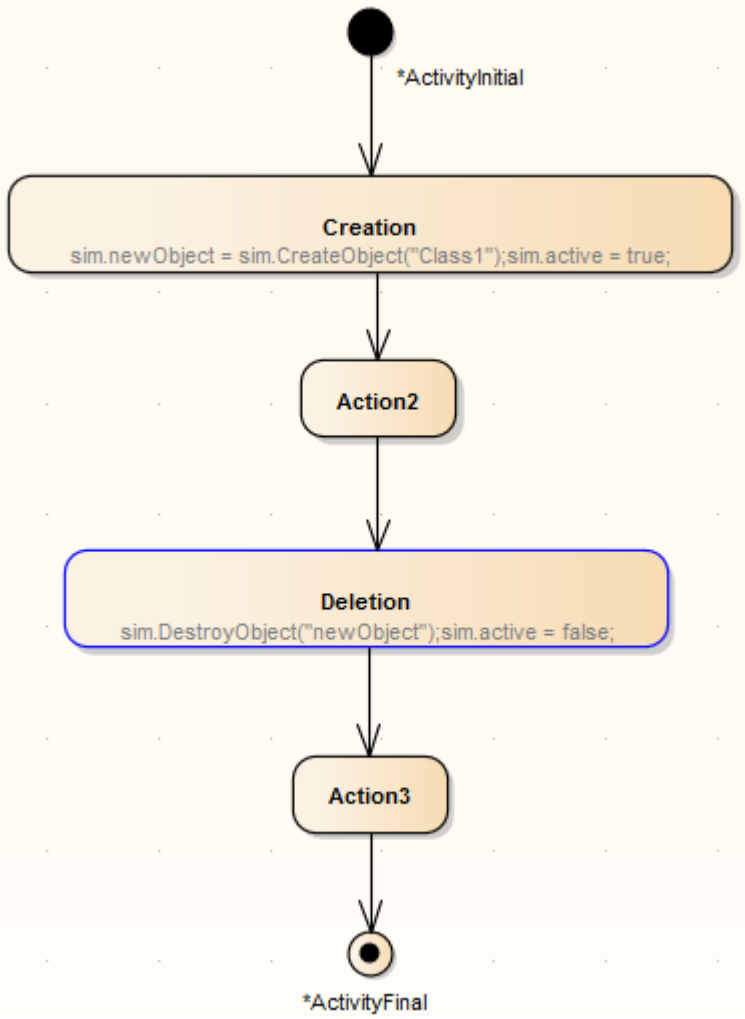
Dans la dialogue « Propriétés » de l'élément Action , dans le champ « Effet » de la page « Effet », saisissez soit :

sim.DestroyObject ("nom d'objet")

ou

supprimer sim.objectFullName

Par exemple:



Notes

- Dans les deux cas, vous pouvez également détruire un object global (un objet créé en dehors du flux de processus) en identifiant l' Object auprès de l' Action effectuant la destruction ; dans le cas de l' Action DestroyObject, en transmettant le nom Object d'un port sur l' Object à la broche d'entrée sur l' Action via un connecteur de flux Object

Simulation dynamique avec JavaScript

Les éditions Corporate , Unified et Ultimate d' Enterprise Architect permettent d'utiliser JavaScript pour évaluer les protections, les effets et d'autres aspects du comportement dans le contexte Simulation . Cela permet une exécution entièrement automatisée et intelligente de votre modèle State ou d'activité, avec un contrôle précis des points d'arrêt, de la vitesse d'exécution et des variables Simulation .

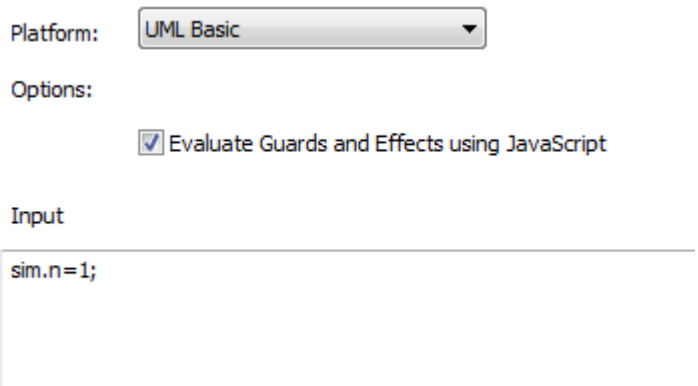
Vous pouvez écrire JavaScript qui utilise n'importe quelle variable. Pour vous permettre d'afficher les valeurs de ces variables via l'interface utilisateur, deux objets intégrés sont définis - **sim** et **this** - dont les membres peuvent être affichés dans la fenêtre Variables locales (également appelée fenêtre Variables locales). Voici quelques exemples de variables pouvant être affichées :

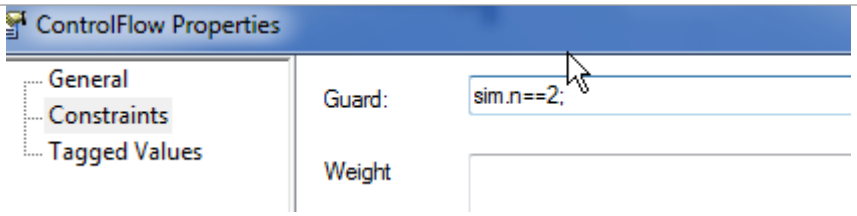
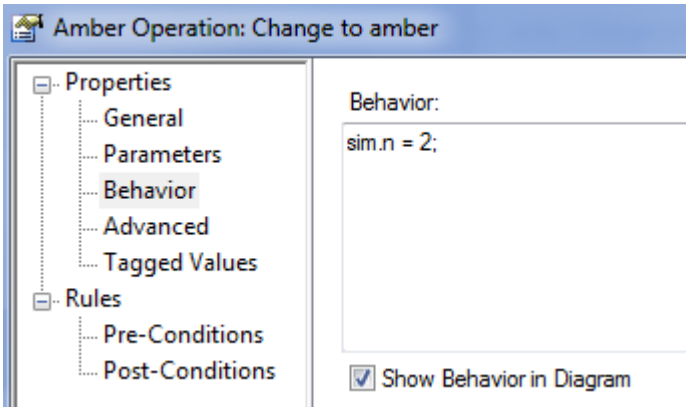
- sim.logger
- sim.Nom.du.client
- ceci.compte
- ce.montant.du.compte

La convention recommandée consiste à ajouter à l' object **sim** toutes les variables globales ou de contrôle non déclarées dans la classe propriétaire. En revanche, il serait normal d'ajouter les attributs du classificateur propriétaire à l' object **this** .

Vous trouverez ici quelques exemples de l'endroit et de la manière dont vous pouvez définir le comportement Simulation à l'aide JavaScript . D'autres exemples sont fournis dans le modèle EAExample.eap fourni avec Enterprise Architect .

Utilisation de JavaScript

Paramètre	Action
Entrée Script d'Analyseur	Si vous entrez du code JavaScript dans le champ « Entrée » de la fenêtre Analyseur d'Exécution , ce code sera injecté dans la Simulation et exécuté avant le démarrage de la Simulation . Ceci est utile pour établir des variables COM, des compteurs globaux, des fonctions et d'autres initialisations. 
Protections de flux de transition et de contrôle	Il s'agit du cheval de bataille de la fonctionnalité Simulation . Comme Enterprise Architect évalue les chemins possibles vers l'avant à chaque nœud d'une Simulation , il teste les gardes sur les transitions sortantes et les flux de contrôle et n'avance que s'il existe un seul véritable chemin à suivre. Dans le cas contraire, la Simulation est considérée comme « bloquée » et une intervention manuelle est requise. Vous devez utiliser l'opérateur « == » pour tester l'égalité.

	
Comportement d'Éléments	Le comportement d'entrée et de sortie peut être défini pour States . Ce code s'exécutera au moment opportun et est utile pour mettre à jour les variables Simulation et effectuer d'autres affectations.  Vous pouvez également simuler le comportement des classes via leurs instances Object et leurs activités dans votre modèle.

Utilisation de COM	Une fonctionnalité très importante de l'implémentation de JavaScript dans le simulateur d' Enterprise Architect est qu'il supporte la création d'objets COM. Cela permet de connecter la Simulation en cours d'exécution à presque tous les autres processus locaux ou distants et d'influencer la Simulation en fonction de données externes ou de modifier potentiellement les données ou le comportement dans le monde extérieur en fonction de l'état actuel Simulation (par exemple, mettre à jour un modèle mécanique ou Simulation logicielle externe à Enterprise Architect). La syntaxe de création d'objets COM est présentée ici :
	this.name="Impair";
	var logger = new COMObject("MySim.Logger");
	logger.Afficher();
	logger.Log(" Simulation démarrée");

Utilisation Solveurs	Anywhere dans Enterprise Architect qui contient du code JavaScript , comme dans Simulation Dynamique , vous pouvez maintenant utiliser une construction JavaScript appelée « Solveur » (la classe Solveur) pour intégrer des outils externes et utiliser directement les fonctionnalités de chaque outil pour exécuter de manière simple et intuitive des fonctions mathématiques et graphiques complexes. Les appels vous aident à échanger facilement des variables entre le moteur JavaScript intégré et chaque environnement. Deux bibliothèques mathématiques prises en charge sont MATLAB et Octave. 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. Faisant partie du moteur JavaScript , ces classes Solveur sont également immédiatement accessibles aux auteurs de Add-In créant Add-Ins JavaScript basés sur des modèles. Consultez les rubriques d'aide *Octave Solveur* , *MATLAB Solveur* et *Solveurs* pour plus de détails.

Actions signalées	Il est possible de déclencher un événement signalé (déclencheur) directement en utilisant JavaScript . La commande BroadcastSignal() permet de déclencher un déclencheur nommé qui pourrait influencer l'état actuel d'une Simulation . Par exemple, ce fragment déclenche le signal (déclencheur) nommé "CancelPressed". BroadcastSignal("AnnulationAppuyée"); Note qu'un déclencheur nommé CancelPressed doit exister dans l'environnement Simulation actuel et doit avoir ce nom de manière unique. Vous pouvez également appeler le signal à l'aide de son GUID. Par exemple : BroadcastSignal(" {996EAF52-6843-41f7-8966-BCAA0ABEC41F} ");
EST_DANS()	L'opérateur IS_IN(state) renvoie True si la Simulation actuelle a un état actif dans un thread correspondant au nom transmis. Par exemple, pour contrôler l'exécution de manière conditionnelle, il est possible d'écrire un code tel que celui-ci : si (IS_IN("En attente d'entrée")) BroadcastSignal("AnnulationAppuyée") Note que le nom doit être unique dans tous les contextes.
Tracer()	La méthode Trace(statement) vous permet d'imprimer des instructions 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 Simulation JavaScript tronquera les chaînes qui dépassent la longueur maximale définie du message Trace.

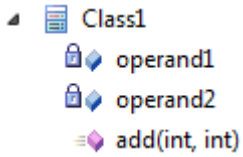
Comportements d'Appels

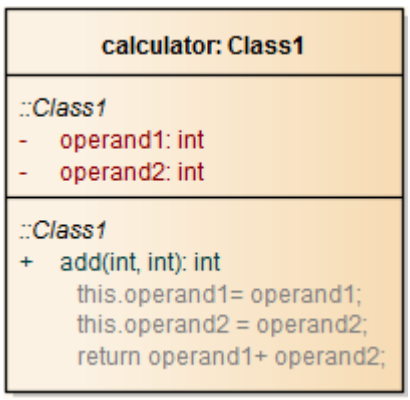
Au cours de la simulation d'un processus, vous pouvez exécuter les comportements définis dans une opération d'une classe (via son Object Simulation) ou d'une activité dans le modèle. Dans chaque cas, vous utilisez JavaScript pour appeler le comportement.

Invoquer le comportement d'une classe

Une classe de votre modèle définit un comportement que vous souhaitez simuler. Ce comportement est défini dans la page Comportement d'une opération de la classe.

Par exemple, la classe est destinée à additionner deux entiers, via l'opération **add(int , int)** . Les entiers dans ce cas sont des paramètres de l'opération, définis par les attributs de la classe, *operand1* et *operand2* .



Étape	Action
1	Dans la fenêtre Propriétés de l'opération, sélectionnez l'onglet « Comportement » et modifiez le champ « Comportement » pour appliquer les objets Simulation JavaScript (*this* ou *sim*) à la définition du comportement. Dans l'exemple : this.opérande1=opérande1; ceci.opérande2=opérande2; retourner opérande1+opérande2
2	Faites glisser la classe sur votre diagramme d'activité Simulation et collez-la en tant qu'instance. Dans l'exemple, l' Object est appelé « calculatrice ». Pour plus de clarté, l'élément affiché ici est configuré pour afficher les attributs et opérations hérités, ainsi que le code de comportement, sur le diagramme . 
3	Sur le diagramme Simulation , pour l'élément Action approprié, ouvrez la dialogue « Propriétés » et sur la page « Effet », saisissez le JavaScript pour capturer et simuler le comportement de l' Object . Dans l'exemple, le JavaScript définit une valeur qui sera fournie en simulant le comportement de l'opération à partir de l' Object , telle qu'elle est effectuée sur deux entiers fournis. C'est-à-dire : sim.result=sim.calculator.add(7,9)

4	Exécuter la Simulation et observez sa progression dans la fenêtre Locals. Au final, le comportement de la classe se reflète dans le résultat de la Simulation . Dans l'exemple : résultat = 16.

Invoquer le comportement d'une activité

Un élément d'activité peut avoir un comportement défini par une opération dans cet élément. À titre d'exemple simple, une activité peut avoir une opération appelée Obtenir le résultat, avec le comportement renvoyé « ON ».

Vous pouvez simuler ce comportement dans le diagramme enfant de l'activité (c'est-à-dire interne à l'activité), avec une instruction JavaScript dans le champ « Effet » de l'élément Action approprié. Dans l'exemple, cela pourrait être :

sim.result=ceci.GetResult();

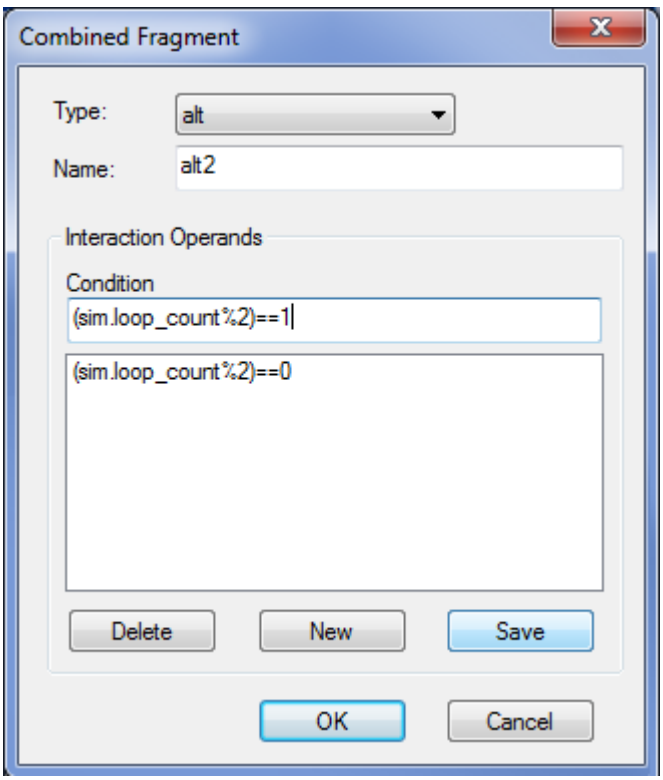
L'instruction appelle l'opération GetResult de l'activité parente et attribue le résultat du comportement de cette opération à sim.result. Vous pouvez observer la progression de la Simulation et le résultat de la simulation du comportement dans la fenêtre Variables locales, où (dans cet exemple) la valeur de résultat « ON » s'affichera finalement.

Condition d'Opérande d'Interaction et Comportement du Message

Lorsque vous simulez le comportement d'un diagramme Séquence , vous pouvez utiliser une condition pour l'opérande d'interaction CombinedFragment, afin de contrôler le flux au cours de la Simulation .

Un message dans diagramme Séquence peut être lié à une opération, de sorte que le comportement de l'opération peut également être utilisé au cours de la Simulation .

Conditions d'opérande d'interaction

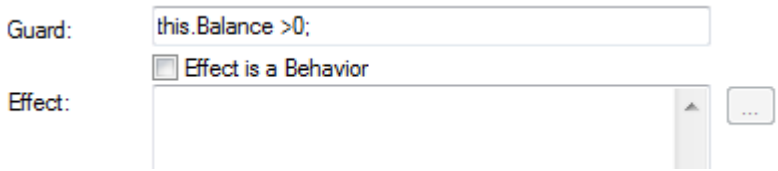
Champ/Colonne	Description
Condition d'opérande	Les conditions d'opérande d'interaction sont des instructions conditionnelles qui sont évaluées chaque fois que le simulateur doit déterminer le chemin à suivre. Les conditions d'opérande ont généralement les caractéristiques suivantes : • Défini dans la dialogue « Fragment combiné » • Écrit en JavaScript • Peut faire référence aux variables définies pendant Simulation
Ajout de conditions d'opérande	Pour ajouter une condition d'opérande : 1. Double-cliquez sur l'élément CombinedFragment pour ouvrir la dialogue « Fragment combiné ». 2. Cliquez sur le bouton Nouveau. 3. Dans le champ « Condition », saisissez le JavaScript pour la condition. 4. Cliquez sur le bouton Enregistrer. 

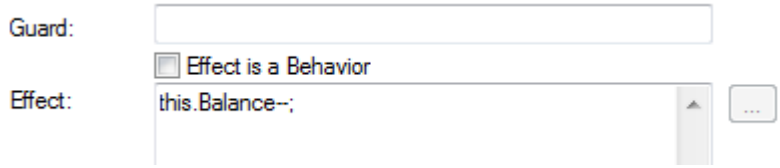
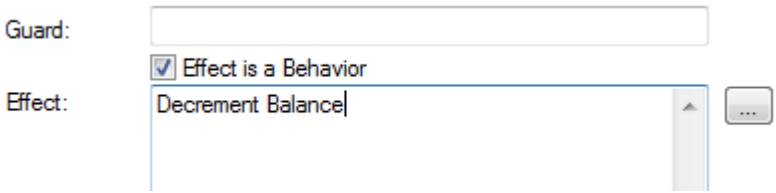
Sémantique d'évaluation	Pendant l'exécution, le simulateur évalue toute condition d'opérande dans le CombinedFragment ; le type CombinedFragment et le résultat de l'évaluation peuvent déterminer le chemin sur lequel la Simulation continue.

Gardes et Effets

Gardes et Effets sont utilisés pour contrôler le déroulement de la Simulation et pour exécuter des actions ou des effets supplémentaires au cours d'une Simulation .

Gardes et Effets

Concept	Détail
Gardes	Les gardes sont des instructions conditionnelles qui sont évaluées chaque fois que le simulateur doit déterminer le chemin à suivre. Les gardes ont généralement les caractéristiques suivantes : • Défini sur les transitions et les flux de contrôle pour régir le déroulement d'une Simulation • Écrit en JavaScript • Peut faire référence aux variables définies pendant Simulation
Ajout de gardes	Les gardes sont définies sur la transition ou le flux de contrôle dans la dialogue « Propriétés » du connecteur sélectionné. Une garde est généralement un morceau de JavaScript qui sera évalué comme étant vrai ou faux. Par exemple, voici une instruction conditionnelle qui fait référence à une variable actuelle (Balance) supérieure à zéro. Note l'utilisation du préfixe « *this* » pour indiquer que la variable est un membre du contexte de classe actuel. 
Sémantique d'évaluation	Lors de l'exécution, le simulateur examinera tous les chemins possibles et évaluera les éventuelles conditions de protection. Cette évaluation pourrait établir que : • Un seul chemin valide vers l'avant est évalué à True ; le simulateur suivra ce chemin • Il existe deux chemins valides ; le simulateur se bloquera en attendant une saisie manuelle via la fenêtre de la console pour résoudre le blocage • Aucun chemin valide n'existe ; la même réponse que lorsque deux chemins sont trouvés : le simulateur attend que l'utilisateur modifie le contexte d'exécution à l'aide de la fenêtre de la console • Aucun chemin n'est évalué à True mais une valeur par défaut (chemin non protégé) existe ; le simulateur prendra le chemin non protégé
Effets	Les effets sont des comportements définis qui sont exécutés à des moments précis : • À l'entrée dans un État • À la sortie d'un État • Lors de la transition d'un état à un autre (effet de transition) Les effets peuvent être soit une section de code JavaScript , soit un appel à un autre élément de comportement dans la simulation actuelle.
Effets JavaScript	Un effet JavaScript pourrait ressembler à cet exemple, dans lequel la variable

	Balance est décrémentée : 
Effets du comportement d'appel	Dans cet exemple, l'effet est un effet de comportement d'appel. Dans ce cas, il appelle dans un modèle l'activité nommée Decrement Balance qui est définie ailleurs. La simulation entrera alors dans ce diagramme /comportement et continuera à s'exécuter jusqu'à revenir au point auquel l'effet a été invoqué. 
Ordre d'exécution des effets	Dans les simulations complexes qui pourraient impliquer une transition d'états profondément imbriqués vers d'autres états profondément imbriqués dans un contexte parent différent, il est important de prendre en compte ces règles concernant l'ordre d'exécution : • Toutes les actions de sortie (effets) rencontrées en quittant un contexte imbriqué sont exécutées dans l'ordre du plus profondément imbriqué au moins profondément imbriqué • Toutes les actions (effets) définies sur les transitions sont exécutées ensuite • Enfin, tous les effets d'entrée sont exécutés du contexte le moins profondément imbriqué au plus profondément imbriqué La règle de base est donc la suivante : toutes les actions de sortie, suivies de toutes les actions de transition, et enfin toutes les actions d'entrée.
Note sur les variables JavaScript	Les variables JavaScript auxquelles il faut accéder et auxquelles il faut faire référence pendant l'exécution Simulation appartiennent à : • sim (par exemple, sim.pedestrianwaiting) - généralement utilisé pour les variables Simulation globales, ou • ceci (par exemple, this.CustomerNumber) - généralement utilisé pour faire référence aux attributs de classe propriétaires Il est important de faire savoir au moteur JavaScript que vous faites référence à une variable Simulation et non à une simple variable locale utilisée, par exemple, lors de calculs de base. Vous pouvez créer des variables Simulation de portée et de profondeur arbitraires. Par exemple, this.customer.name est un nom qualifié légitime.

Déclencheurs

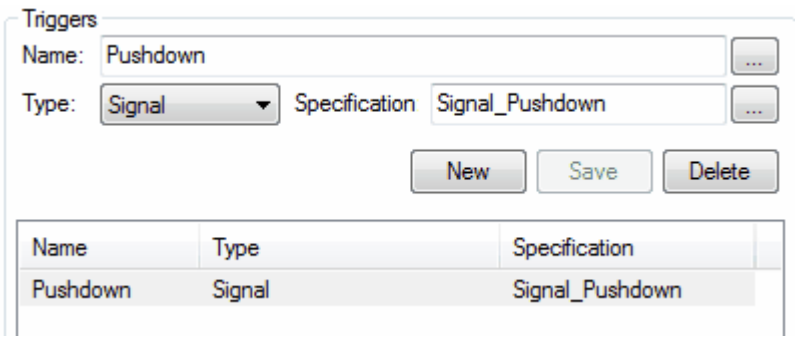
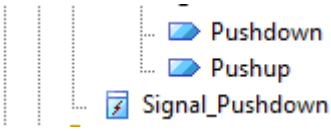
Déclencheurs représentent des signaux et des événements qui peuvent activer des transitions quittant l'état actuel. Un déclencheur peut représenter un signal ou un événement du monde réel tel que :

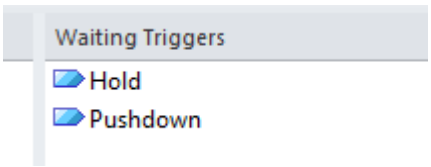
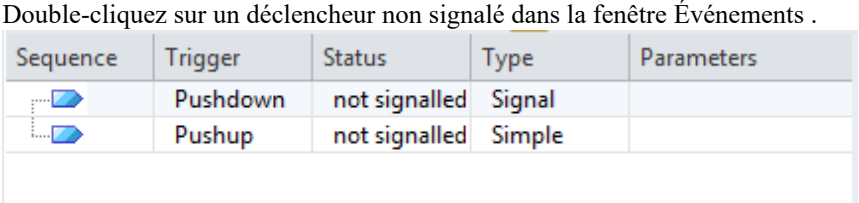
- Un bouton appuyé
- Un message en cours de réception
- Une pédale enfoncée
- Un interrupteur est actionné
- Un état dans une région concurrente en cours d'entrée ou de sortie

Pour qu'une déclencheur ait un effet

- Des transitions doivent être définies qui se déclencheront lorsque la simulation recevra le signal ou l'événement
- L' State Simulation actuel ou son ou ses parents doivent avoir une transition sortante qui accepte ce déclencheur
- La transition activée doit être non protégée ou avoir une protection qui sera évaluée à True

Gérer Déclencheurs

Action	Détail
Création Déclencheurs	Déclencheurs sont créés soit comme une instance d'un élément Signal, soit comme un événement anonyme. Déclencheurs sont connectés aux transitions dans la dialogue « Propriétés de transition », comme illustré ici. Dans cet exemple, un Déclencheur nommé « Pushdown » a été défini sur la base du signal « Signal_Pushdown ». • L'omission des détails Type et Spécification donne un simple Déclencheur anonyme. • Si des paramètres sont nécessaires, ceux-ci sont définis sur le signal et doivent être fournis au moment où l'événement se déclenche  Un déclencheur apparaîtra dans l'onglet « Projet » de la fenêtre Navigateur , comme illustré ici : 
Utilisation Déclencheurs	Déclencheurs sont déployés en les connectant à des transitions, comme dans

	l'exemple précédent, et sont utilisés pendant la simulation en les « tirant » dans la simulation en cours d'exécution selon les besoins. Lors de l'utilisation déclencheurs ces points doivent être pris en compte : • Une transition « déclenchée » ne peut pas avoir lieu tant que son déclencheur effectif n'est pas signalé ou déclenché • Lorsqu'un déclencheur est reçu, il active toutes les transitions en attente en cours dépendant de ce déclencheur (c'est-à-dire que le déclencheur est diffusé) • Déclencheurs sont évalués sur toutes les transitions pour tous les parents d'un état enfant actuel ; cela permet à un état parent de quitter tous les états enfants si nécessaire • Une fois utilisé dans une simulation, un déclencheur est consommé et doit être réactivé si nécessaire. • Ensembles de déclencheurs peuvent être enregistrés et déclenchés manuellement ou automatiquement pour faciliter la simulation automatisée de modèles sous différents modèles d'événements
Tir Déclencheurs	Le déclenchement déclencheurs signifie signaler ou activer un déclencheur dans la simulation en cours. Cela peut activer zéro, une ou plusieurs transitions en attente en fonction de l'état et de la concurrence de la Simulation en cours. Le déclenchement déclencheurs peut être réalisé de plusieurs manières. La plus efficace est la liste « Déclencheurs en Attente ». Au cours de Simulation du modèle, si le simulateur atteint une impasse en raison de déclencheurs requis non disponibles (déclenchés), la liste de tous déclencheurs candidats possibles est affichée dans la liste « Déclencheurs en Attente » de la fenêtre Événements Simulation .  Un double-clic sur un déclencheur dans cette liste le déclenchera dans la Simulation . D'autres façons de déclencher un déclencheur incluent : 1. Double-cliquez sur un déclencheur non signalé dans la fenêtre Événements .  Vous pouvez également utiliser le menu contextuel de ces événements pour signaler un événement non signalé ou pour signaler à nouveau un événement qui a déjà été déclenché précédemment. 2. Utilisez le menu contextuel de la Transition requise pour tirer et sélectionnez l'option de menu 'Signal Déclencheur en Simulation '.

Comportement d'Action par Type

Vous pouvez faire varier le comportement initié par un élément Action en définissant (ou même en redéfinissant) son type. Dans Simulation , vous pouvez appliquer et observer un certain nombre de comportements différents à l'aide des Actions des types et des groupes décrits dans ce tableau .

Types Action

Type	Description
Actions Object	Les actions Object agissent sur un object d'une manière spécifique, comme la création, la destruction ou la lecture de l' object . Elles comprennent : • Créer un objet • Détruire l'objet et • Lire soi-même
Actions variables	Les actions variables ont une variable d'association sous la forme de la variable Valeur Étiquetée avec la valeur du nom d'un object en cours d'exécution. Elles fournissent la variable non seulement en tant object mais également en tant que propriété (comme un attribut ou un port) d'un object . Elles incluent : • Lire la variable • Écrire une variable • Variable claire • Ajouter une valeur variable • Supprimer la variable
Actions de fonctionnalité structurelle	Les actions StructuralFeature agissent sur une fonctionnalité structurelle, à savoir un attribut d'une activité ou du classificateur d'un object . Elles comprennent : • LireStructuralFeature • Écrire une fonctionnalité structurelle • Effacer la structure • Ajouter une valeur de fonctionnalité structurelle • SupprimerStructuralFeatureValue
Actions d'invocation et d'acceptation d'événements	Les actions d'invocation et d'acceptation d'événement définissent les Déclencheurs et les signaux d'un événement. Elles comprennent : • Envoyer un signal • Signal de diffusion • Accepter l'événement • Envoyer un objet • Comportement d'appel • Opération d'appel • Accepter l'appel
Divers Actions	L' Action ValueSpecification évalue une valeur ; elle doit avoir une valeur d'entrée et un code d'évaluation comme comportement ou effet.

Simulation d'Activité Structurée

L'une des structures les plus complexes d'un modèle comportemental est une activité structurée, qui modélise une série d'actions soit dans une structure imbriquée, soit dans un processus d'évaluation et d'exécution. Les types d'évaluation d'une activité structurée sont le nœud conditionnel et le nœud de boucle, que vous pouvez tous deux simuler assez facilement.

Noeud conditionnel

Un nœud conditionnel se compose essentiellement d'une ou plusieurs paires de partitions Test /Corps, chaque paire étant appelée Clause. La partition Test est composée d'éléments diagramme d'activité qui testent une condition et, si cette condition est remplie, d'autres éléments de diagramme d'activité dans la partition Corps sont exécutés pour produire un résultat.

S'il existe une clause, le nœud conditionnel génère soit le résultat de la partition Body, soit aucun résultat. S'il existe plusieurs clauses, le contrôle passe d'un Test au suivant jusqu'à ce qu'une condition soit remplie et qu'une partition Body soit exécutée pour produire un résultat, ou que tous les tests échouent.

Simulation prend actuellement supporte l'utilisation de la case à cocher « Est assuré » dans l'onglet « Condition » de la fenêtre Propriétés . Les deux autres paramètres de case à cocher sont ignorés. Si la case à cocher « Est assuré » est :

- Sélectionné, au moins un Test doit être satisfait, donc son corps est exécuté et un résultat est généré ; si aucun Test n'est satisfait et aucun résultat n'est généré, le nœud conditionnel est bloqué et le traitement ne peut pas continuer au-delà
- Non sélectionné, un Test peut être satisfait et un résultat peut être généré, mais si aucun Test n'est satisfait et aucun résultat n'est généré, le traitement peut toujours se poursuivre au-delà du nœud de condition

Vous pouvez simuler une gamme de chemins et de résultats possibles en saisissant des instructions JavaScript *sim.* qui définissent ou conduisent à des résultats Test et des résultats de corps spécifiques, dans les champs « Effet » des éléments Action au sein de chaque partition de chaque clause. Ces instructions *sim.* doivent identifier le chemin complet du nœud conditionnel, de la clause et de la broche de sortie en cours de définition. Par exemple, dans un test visant à déterminer si une personne est considérée comme une personne âgée :

si (sim.Person.age >=65)

sim.AgeCondition.Clause1.Decider1=true;

autre

sim.AgeCondition.Clause1.Decider1=false;

Le nœud de condition s'appelle *AgeCondition* , le test est dans *Clause1* et le OutputPin pour ce test est *Decider1* .

Noeud de boucle

Un nœud d'activité structuré en boucle représente généralement les équivalents modélisation des instructions de boucle While, Repeat et For. Chaque nœud de boucle comporte trois partitions :

- Configuration - qui initie les variables à utiliser dans la condition de sortie de la boucle ; elle est exécutée une fois à l'entrée de la boucle
- Test - qui définit la condition de sortie de la boucle
- Corps - qui est exécuté à plusieurs reprises jusqu'à ce que le Test produise une valeur fausse

Vous définissez les nœuds de boucle en faisant glisser les éléments diagramme d'activité depuis les pages de la boîte à outils vers les partitions Configuration, Test et Corps. La partition Corps peut contenir des structures d'éléments assez complexes, définissant ce que le nœud de boucle produit réellement au cours du processus.

Le nœud de boucle possède un certain nombre de Pins Action :

- Variable de boucle (entrée) - la valeur initiale à traiter via la boucle

- Variable de boucle (sortie) - la variable variable sur laquelle le Test est effectué
- Décideur - une broche de sortie dans la partition Test , dont la valeur est examinée après chaque exécution du Test pour déterminer s'il faut exécuter le corps de la boucle
- Sortie du corps - la valeur de sortie du traitement dans la partition Corps, qui met à jour la broche de sortie de la variable de boucle pour la prochaine itération de la boucle, et
- Résultat - la valeur de l'exécution finale de la partition Test (qui est la valeur renvoyée depuis la dernière exécution de la partition Body)

Vous pouvez simuler les effets de différentes actions et sorties via la boucle, en saisissant des instructions JavaScript *sim.* qui définissent ou conduisent à des résultats Test et des résultats de corps spécifiques, dans les champs « Effet » des éléments Action au sein de chaque partition. Ces instructions *sim.* doivent identifier le chemin du nœud de boucle et de la broche de sortie en cours de définition. Par exemple, dans une Action dans la partition Test :

```
sim.LoopNode1.decider = (sim.LoopNode1.loopVariable>0);
```

Simulation de Valeur de Retour d'Activité

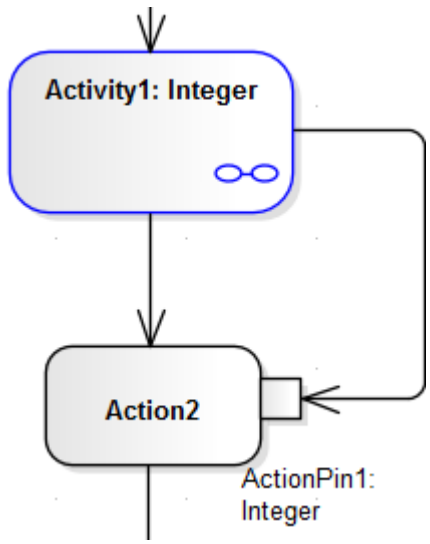
Une activité est susceptible de produire une valeur de retour en tant que sortie du processus qu'elle représente. Vous pouvez simuler la manière dont cette valeur de retour est transmise à l'étape suivante du processus, selon trois scénarios :

- L'activité produit simplement une valeur de retour, qui est transmise directement à l' Action suivante
- L'activité comporte un ou plusieurs paramètres d'activité - représentés sur un diagramme par des nœuds d'activité - qui acceptent les valeurs d'entrée ou contiennent les valeurs de sortie des actions enfants de l'activité, et le paramètre d'activité de sortie collecte et transmet la valeur de retour
- L'activité est instanciée par une Action CallBehavior qui reproduit le comportement de l'activité et transmet la valeur de retour.

Valeur de retour de l'activité Échec

(Cette méthode est unique à la simulation Enterprise Architect , imitant l'effet d'un paramètre d'activité sans qu'il n'en existe un.)

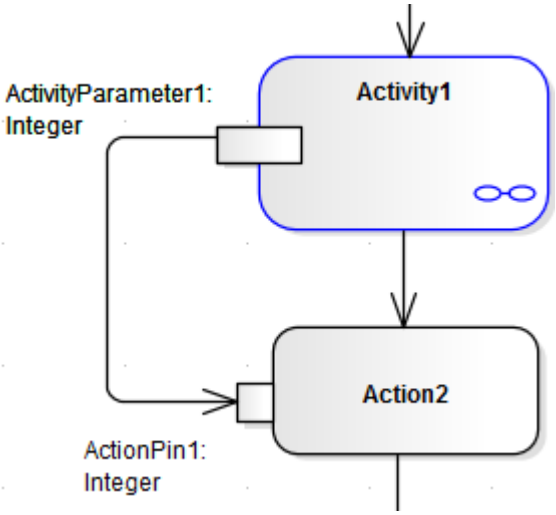
L'activité a une valeur de retour, qui est transférée de l'élément Activité à une broche Action sur l' Action suivante du processus via un connecteur de flux Object .



Vous pouvez simuler cela en définissant une instruction JavaScript simple pour définir la valeur de retour dans l'élément enfant de l'activité (comme this.return=12;) et, en exécutant la simulation, voir la valeur transférée à l'épingle Action dans la fenêtre Locals.

Paramètre d'activité Émission

Si l'activité possède un paramètre d'activité, sa valeur passe au nœud d'activité correspondant, puis, via un connecteur de flux Object , à l'ActionPin d'entrée de l' Action suivante du processus, comme indiqué :

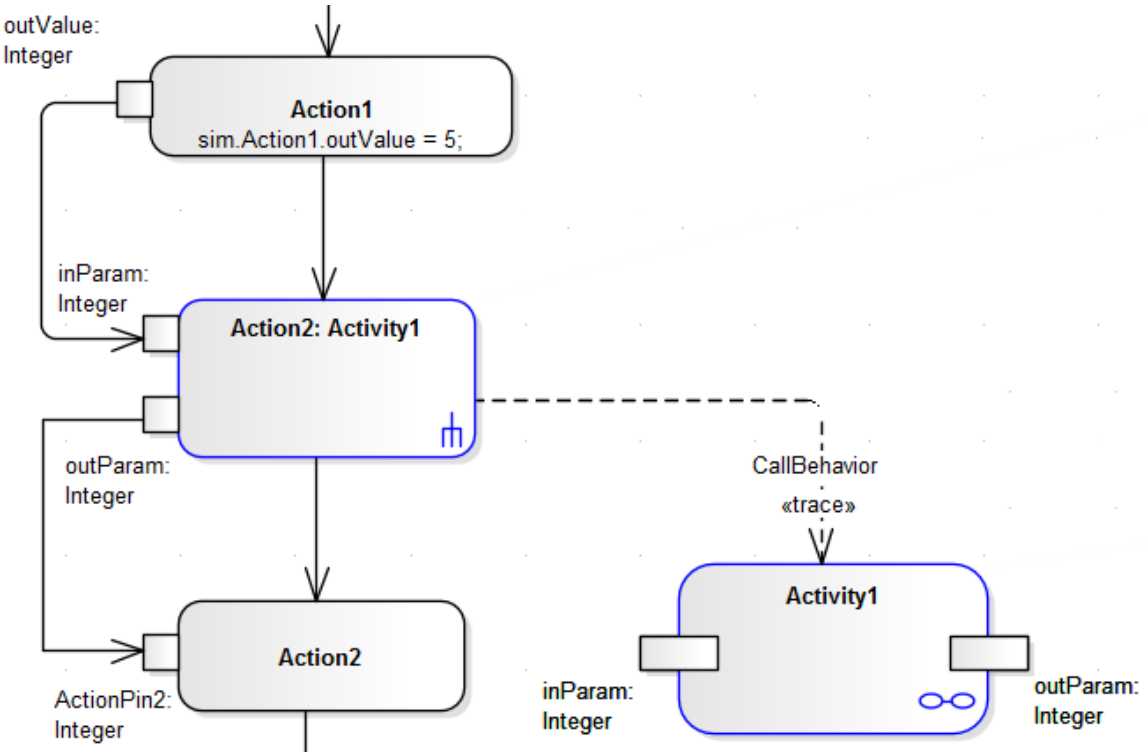


Dans la fenêtre Variables locales, vous pouvez soit observer la transmission valeur par défaut du paramètre à l'ActionPin, soit utiliser JavaScript dans les actions enfants de l'activité pour simuler une mise à jour de la valeur au sein de l'activité. Par exemple :

ceci.ActivityParameter1=20;

Action CallBehavior

Une activité peut être utilisée plusieurs fois dans un processus, auquel cas vous souhaiterez peut-être générer une instance distincte de l'activité à chaque fois. Vous pouvez le faire à l'aide d'une Action CallBehavior pour créer un object de l'activité et exécuter son comportement. Les paramètres d'activité d'entrée et de sortie sont liés aux Pins Action d'entrée et de sortie correspondantes (arguments) sur l' Action CallBehavior.



Lorsque vous simulez la partie du processus contenant l'activité, vous fournissez une valeur d'entrée (comme dans Action 1) qui passe dans la broche Action d'entrée sur l' Action CallBehavior, ce qui crée un Object de l'activité. Le comportement CallBehavior exécute le comportement de l'activité, en utilisant la broche Action d'entrée pour agir comme paramètre d'activité d'entrée et la broche Action de sortie pour recevoir le retour comme paramètre d'activité de

sortie. La valeur de retour de l'activité est ensuite transmise à une broche Action sur l' Action suivante, à l'aide d'un connecteur de flux Object . Vous pouvez fournir des instructions JavaScript dans les actions de l'activité pour agir sur la valeur d'entrée et générer une valeur de retour, telle que :

sim.buf=this.inParam; et

ceci.outParam=sim.buf + 11 :

Fenêtre Simulation Événements

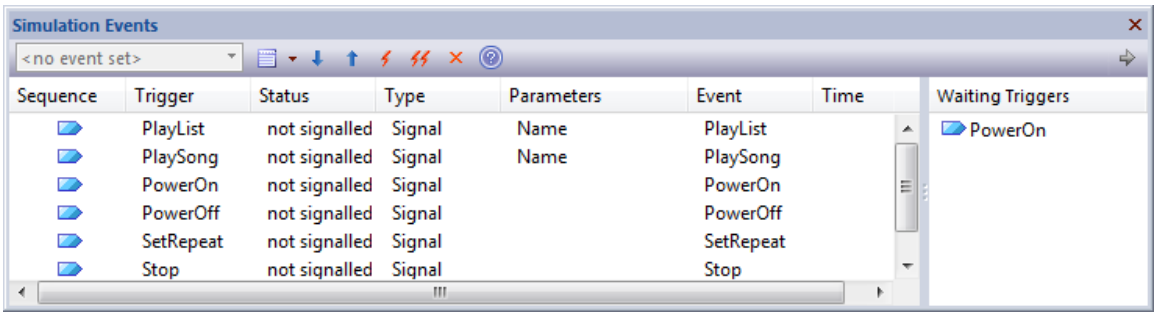
La fenêtre Événements Simulation permet de gérer déclencheurs et les ensembles d'événements d'une simulation. Ses principales fonctions sont les suivantes :

- Ajouter, supprimer et reséquencer un ensemble de déclencheurs pour une simulation
- Afficher une liste des événements déclenchés, perdus et en attente pour la simulation en cours d'exécution
- Fournir des options pour déclencher n'importe quel déclencheur arbitraire dans la simulation actuelle
- Fournir une liste pratique de « Déclencheurs en Attente » des déclencheurs que la simulation attend
- Enregistrez les ensembles déclencheur pour une utilisation ultérieure dans les simulations manuelles et automatisées
- Accepter déclencheurs glissés depuis la fenêtre Navigateur vers la liste actuelle
- Saisir les paramètres déclencheur pour un déclencheur en attente avant le déclenchement

Au fur et à mesure que déclencheurs sont consommés dans la simulation, leur statut et leur position sont enregistrés dans le corps principal de la fenêtre Événements Simulation .

Vous pouvez enregistrer le log des déclencheurs déclenchés sous forme d'ensemble déclencheur ou d'ensemble d'événements à réappliquer dans une autre Simulation exécuter , que vous pouvez exécuter manuellement ou automatiquement. Consultez la rubrique *Ensemble Déclencheur et Auto-Tir* pour plus d'informations sur la création et l'utilisation des ensembles Déclencheur .

Cette image illustre la fenêtre Simulation Événements pendant l'exécution.



Accéder

Ruban	Simuler > Simulation Dynamique > Événements

Détails de la colonne

Champ/Colonne	Action
Séquence	Pendant et après la simulation, indique la position dans la séquence à laquelle un déclencheur a été déclenché ou est censé être déclenché. Note que si un déclencheur est déclenché hors séquence, il sera déplacé vers le bas de la section des événements signalés.
Déclencheur	Le nom du déclencheur - identifie le Déclencheur utilisé pour initier l'événement.
Statut	Indique l'état du Déclencheur . Les valeurs peuvent être :

	• utilisé - le déclencheur a été déclenché et le traitement a été transmis • perdu - la déclencheur a été déclenchée dans la liste, mais elle n'a eu aucun effet • signalé - un déclencheur a été déclenché et consommé par une ou plusieurs transitions • non signalé - le déclencheur n'a pas encore été actionné
Type	Indique le type de déclencheur . Actuellement, seuls supporte : • Signal • (pas de type) un déclencheur anonyme
Paramètres	Pour un Signal Déclencheur , affiche initialement les paramètres requis pour le déclenchement par la spécification Signal. Par exemple, un signal « Login » peut inclure des paramètres de nom d'utilisateur et de mot de passe - et chaque invocation déclenchée peut utiliser des paramètres différents. A chaque fois que la simulation déclenche le déclencheur , le système vous prompt des valeurs. Vous pouvez également modifier les valeurs directement dans la liste lorsque le déclencheur est réglé sur non signalé. Les paramètres sont très utiles pour tester la logique conditionnelle dans votre simulation et pour simuler une variété d'entrées et de données provenant de l'extérieur de la simulation.
Événement	Pour un : • Signal Déclencheur , identifie la spécification Signal • Pour Déclencheurs anonymes n'a aucune valeur
Temps	L'heure de simulation à laquelle le déclencheur a été signalé. Note qu'il s'agit d'une heure absolue (du monde réel) et non d'une heure relative d'événement de simulation.
Déclencheurs en Attente	Répertorie les Déclencheurs disponibles pour la sélection à partir des états actuels, y compris ceux où plusieurs déclencheur sont possibles lors d'une même transition. Double-cliquez sur un déclencheur pour l'ajouter et le signaler comme déclencheur suivant dans la séquence d'événements en cours. Vous pouvez afficher et masquer ce panneau en cliquant sur la flèche grise juste au-dessus du panneau.

Items de la barre d'outils

Option	Action
	Utilisez cette liste déroulante pour sélectionner et travailler avec des ensembles déclencheur précédemment définis. Avant d' exécuter une simulation, sélectionnez un ensemble déclencheur préalablement défini à utiliser pour la prochaine simulation. Vous pouvez choisir de ne pas utiliser d'ensemble déclencheur en sélectionnant l'option <aucun ensemble d'événements>.
	Cliquez pour créer et supprimer des ensembles déclencheur :

	• Enregistrer l'ensemble - Enregistrer la liste déclencheur actuelle en tant que nouvel ensemble déclencheur ; le système vous prompts un nom pour le nouvel ensemble • Enregistrer l'ensemble sous - Créer une copie de l'ensemble actuel sous un nouveau nom d'ensemble • Supprimer Sélectionnée Set - Supprimer le set déclencheur actuel • Supprimer tous Ensembles pour Diagramme - Supprimez tous les ensembles déclencheur enregistrés pour le diagramme actuel
	Déplacez le déclencheur sélectionné d'une ligne vers le bas dans la séquence de tir des déclencheurs . Cette option n'est pas disponible s'il n'y a pas déclencheurs signalés sous la ligne sélectionnée.
	Déplacez l'entrée déclencheur sélectionnée d'une ligne vers le haut dans la séquence de déclenchement des déclencheurs . Cette option n'est pas disponible s'il n'y a pas déclencheurs signalés au dessus de la ligne sélectionnée.
	Cliquez pour déclencher le déclencheur sélectionné. Vous pouvez également déclencher le déclencheur en double-cliquant dessus.
	Cliquez pour activer et désactiver le tir automatique. Le déclenchement automatique déclenchera les déclencheurs non signalés de votre ensemble déclencheur de manière séquentielle. Si votre ensemble correspond à un chemin d'exécution valide, la simulation exécuter automatiquement. déclencheurs hors séquence ou inutilisés seront « perdus ». Un point d'arrêt interrompt le tir automatique et vous devrez cliquer sur le déclencheur suivant pour reprendre le tir automatique de la simulation.
	Supprimez le(s) déclencheur (s) sélectionné(s) de la liste.

Options Menu Contexte

Option	Action
Signal sélectionné	Signalez, ou déclenchez, le déclencheur sélectionné non signalé.
Supprimer la sélection	Supprimer un déclencheur non signalé de la séquence.
Re-signaler la sélection	Appuyer à nouveau sur un déclencheur utilisé ou signalé.
Régler tout sur Non signalé	Réglez tous déclencheurs utilisés ou signalés sur non signalés.
Effacer la liste Déclencheur	Effacer tous déclencheurs de la fenêtre, quel que soit leur statut.

Déclencheurs en Attente

Lorsqu'une simulation atteint un point où tout changement d'état (pour n'importe quel thread) nécessite qu'un Déclencheur se déclenche, la simulation est effectivement mise en pause et le contrôle revient au système. La simulation attend maintenant effectivement une forme d'événement (un signal du monde réel) pour se déclencher. La liste Déclencheurs en Attente est utile pour aider à déterminer quel Déclencheur doit être signalé manuellement.

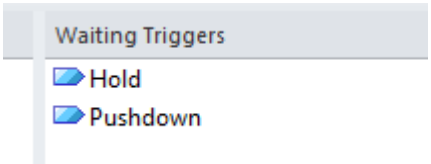
Accéder

Ruban	Simuler > Simulation Dynamique > Événements Le volet de droite répertorie Déclencheurs disponibles.

La liste Déclencheurs en Attente sur la fenêtre Simulation Événements est la suivante :

- Rempli à chaque cycle Simulation avec tous Déclencheurs qui auraient un effet immédiat s'ils étaient signalés
- Rempli d'un ensemble discret (les doublons éventuels ne sont pas affichés car un Déclencheur est effectivement diffusé à toutes les transitions à la fois)
- Activé en double-cliquant sur le Déclencheur d'intérêt
- Inclut tous déclencheurs possibles - y compris ceux qui activent les transitions sur les parents des états actuellement imbriqués

Cet exemple montre que la simulation actuelle a atteint un point où deux Déclencheurs possibles peuvent influencer le flux d'exécution.



En raison de la nature des Déclencheurs et de leurs effets, la liste peut se référer à chacune de ces situations d'exemple de manière tout aussi valable :

- Un seul état a deux transitions sortantes qui attendent respectivement Hold et Pushdown ; le déclenchement de l'une d'entre elles activera la transition associée dans la simulation
- Un même état a deux ou plusieurs déclencheurs possibles pour la même transition, comme une caméra de sécurité allumée par un détecteur de mouvement, un détecteur de son ou un détecteur de chaleur.
- Deux (ou plusieurs) threads (régions simultanées) ont chacun un état en attente de Hold ou Pushdown ; le déclenchement de l'un de ces déclencheurs entraînera l'attente du ou des threads sur ce déclencheur pour continuer tandis que les autres threads resteront bloqués
- Un état enfant attend l'un des déclencheurs tandis qu'un état parent attend l'autre ; le déclenchement d'un déclencheur entraînera le déclenchement de la transition associée et l'enfant ou le parent procédera en conséquence.
- Toute combinaison de ceux-ci

Déclencheurs de Re-Signal

Il est possible de re-signaler un Déclencheur comme raccourci pour faire glisser des instances Déclencheur supplémentaires pour la signalisation.

Accéder

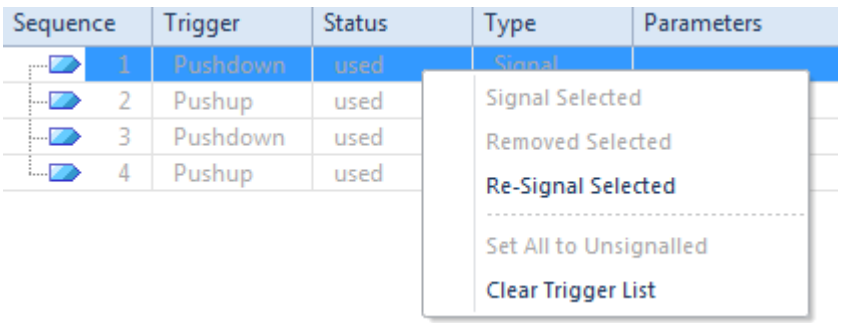
Affichez la fenêtre Simulation Événements , puis cliquez-droit sur un Déclencheur dans cette fenêtre et sélectionnez l'option 'Re-Signal Selected'.

Ruban	Simuler > Simulation Dynamique > Événements > cliquez-droit sur déclencheur existant > Re-Signal sélectionné

Liste Déclencheur

La fenêtre Simulation Événements contient une liste de Déclencheurs qui ont déjà été déclenchés. En cliquant avec le bouton droit de la souris sur un Déclencheur que vous souhaitez signaler à nouveau, vous pouvez utiliser le menu contextuel pour provoquer le déclenchement à nouveau.

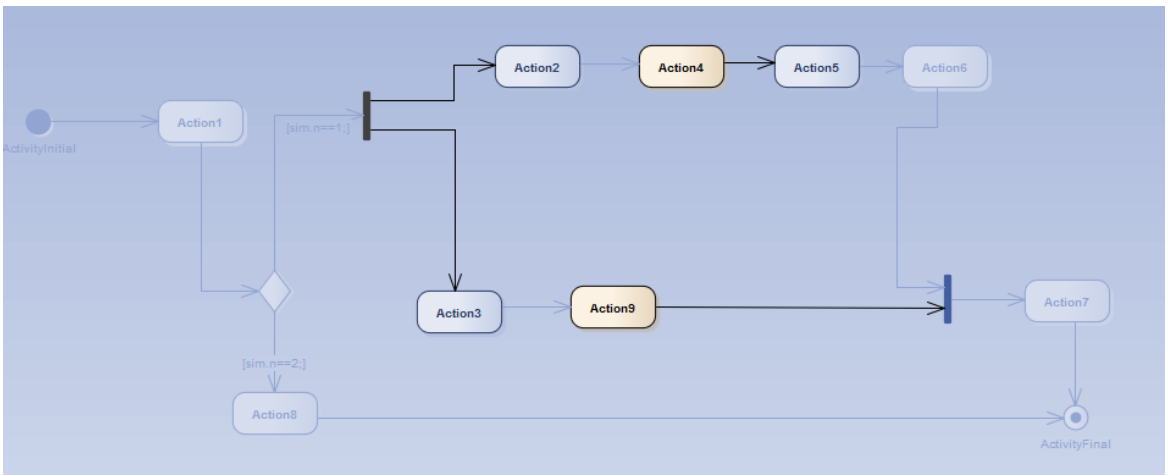
Cette image montre la re-signalisation en action. Lorsqu'un signal est re-signalé, une nouvelle copie est créée et placée à la fin de la liste déclencheurs signalés, où elle est automatiquement déclenchée à nouveau.



Multi-filetage - Fourches et Jointures

Le simulateur Modèle offre la possibilité de gérer des simulations multithreads à l'aide de nœuds Fourche et Joindre .

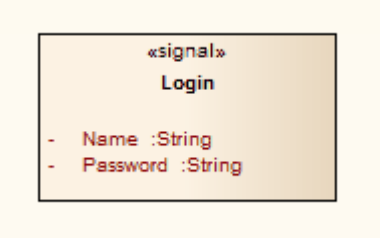
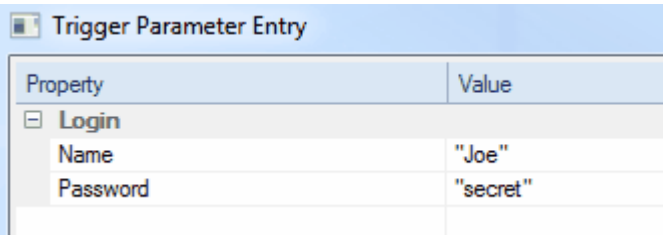
- Dans l'exemple, le point d'exécution actuel s'est divisé en deux threads, chacun avec son propre nœud actif
- Au fur et à mesure que cet exemple progresse, la branche inférieure attendra au niveau du nœud Join jusqu'à ce que la branche supérieure ait terminé toutes ses actions
- Une fois que les deux threads fusionnent à nouveau en un seul, la Simulation se poursuivra comme un seul thread jusqu'à la fin.
- Lors de l'exécution automatique, chaque thread sera considéré comme exécutant une seule étape au cours d'un « cycle » de simulation - bien que lors d'une exécution pas à pas unique ou à un point d'arrêt, le comportement consiste à alterner l'exécution pas à pas entre les threads lorsque chaque thread reçoit du temps de traitement
- Note que la fenêtre Pile d'Appel affichera deux threads actifs et un thread « en pause » dans l'exemple ; une fois les threads fusionnés, il y aura un retour à l'exécution à thread unique
- note également que les variables locales sont partagées (globales) entre tous les threads ; si vous souhaitez simuler des variables privées sur un thread, vous devez créer de nouvelles variables Simulation au début de chaque thread, en préchargeant ces variables avec les données globales existantes.

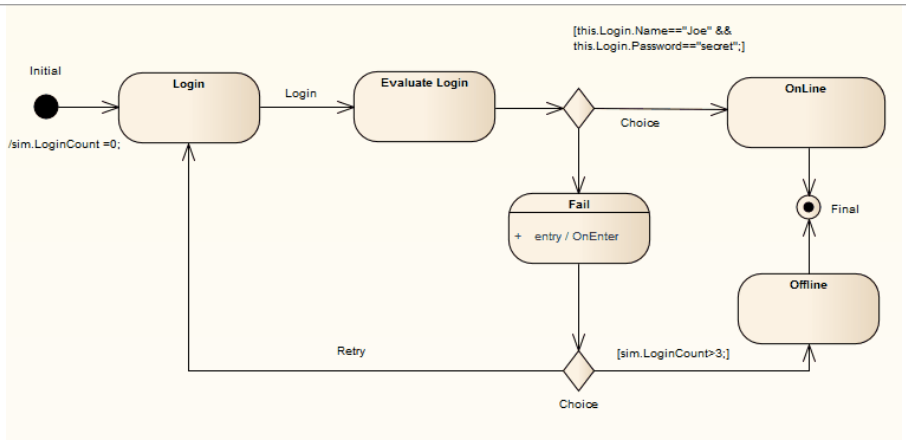


Paramètres Déclencheur

Les paramètres Déclencheur sont des arguments passés dans la simulation avec un déclencheur lorsqu'il est déclenché. Ils permettent de spécifier un comportement complexe et de prendre des décisions en fonction des variables et des données passées dans une simulation au moment de l'exécution par un déclencheur déclenché (événement).

Paramètres

Paramètre	Détail
Introduction	Pour utiliser les paramètres déclencheur vous devez : • Créez d'abord un élément Signal avec les attributs appropriés qui deviendront vos paramètres au moment de l' exécuter • Sur une transition appropriée dans votre diagramme , créez un déclencheur basé sur le signal créé précédemment • Au moment de l'exécution, il sera demandé de saisir les paramètres appropriés - ils sont ensuite transmis avec le déclencheur
Signaux	Un élément Signal est un gabarit ou une spécification à partir duquel déclencheurs réels peuvent être construits. Cet exemple comporte deux arguments, un nom et un mot de passe. Ceux-ci seront renseignés au moment de l'exécution, soit manuellement, soit dans le cadre d'un ensemble déclencheur prédéfinis. 
Paramètres Déclencheur	L' prompt des paramètres Déclencheur demande des valeurs appropriées pour chaque paramètre. Note que vous devez placer les chaînes entre guillemets, sinon l'interpréteur pensera que vous faites référence à d'autres variables. 
Exemple Diagramme	Il s'agit d'un exemple diagramme qui utilise des paramètres déclencheur . À l'état Évaluer la connexion, la simulation examine les variables transmises en tant que paramètres déclencheur et prend la décision d'accepter les informations d'identification ou de les refuser.


[this.Login.Name=="Joe" &&
this.Login.Password=="secret";]
Initial
Login
Login
Evaluate Login
Choice
OnLine
/sim.LoginCount =0;
Fail
+ entry / OnEnter
Final
Offline
Retry
[sim.LoginCount>3;]
Choice

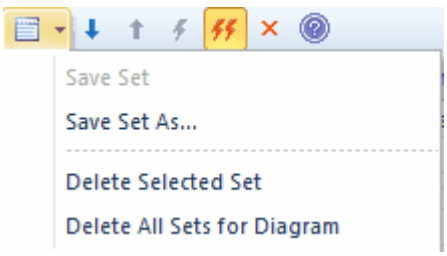
Ensemble Déclencheur et Auto-Tir

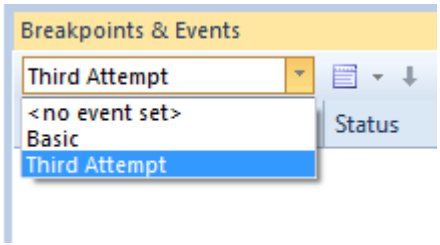
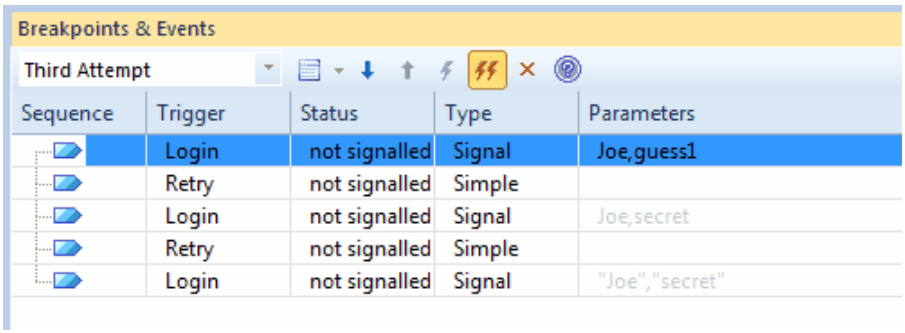
Ensembles Déclencheur sont un moyen efficace d'automatiser et de rationaliser l'exécution, les tests et la validation des modèles de simulation. En réutilisant des ensembles de déclencheurs (avec ou sans paramètres), il est possible de parcourir rapidement et efficacement de nombreux scénarios de simulation, soit manuellement, soit automatiquement à l'aide de l'outil « auto-firing ».

Accéder

Ruban	Simuler > Simulation Dynamique > Événements

À propos des Ensembles de Déclencheurs

Aspect	Détails
Ensembles Déclencheur	• Stocké avec un diagramme associé • Constitué d'une liste de Déclencheurs dans une séquence définie • Peut inclure des paramètres Déclencheur si nécessaire • Peut être utilisé manuellement en double-cliquant sur Déclencheurs pour tirer selon les besoins • Peut être utilisé dans le cadre du comportement « tir automatique » pour automatiser l'exécution • Géré depuis la fenêtre Simulation Événements
Gestion Ensembles	Les ensembles Déclencheur peuvent être créés en faisant glisser manuellement déclencheurs dans la liste déclencheurs actifs, puis en utilisant le menu déroulant « Gérer Ensembles Déclencheur » pour enregistrer un nouvel ensemble. Il est également possible de sauvegarder un ensemble de déclencheurs créés au cours d'une même simulation en tant que nouvel ensemble. Cela est pratique pour créer plusieurs chemins de test au cours d'une simulation, en sauvegardant les déclencheurs déclenchés manuellement pour chaque cas de test.  Vous pouvez également supprimer un ensemble et supprimer tous les ensembles du diagramme actuel. Il est également possible de charger un ensemble, de modifier les paramètres et/ou l'ordre de déclenchement et de sauvegarder l'ensemble sous un nouveau nom. Il s'agit d'une méthode pratique pour créer rapidement une suite de scripts de test de simulation.
Utilisation Ensembles	Pour utiliser un ensemble déclencheur sélectionnez-le d'abord par son nom dans la

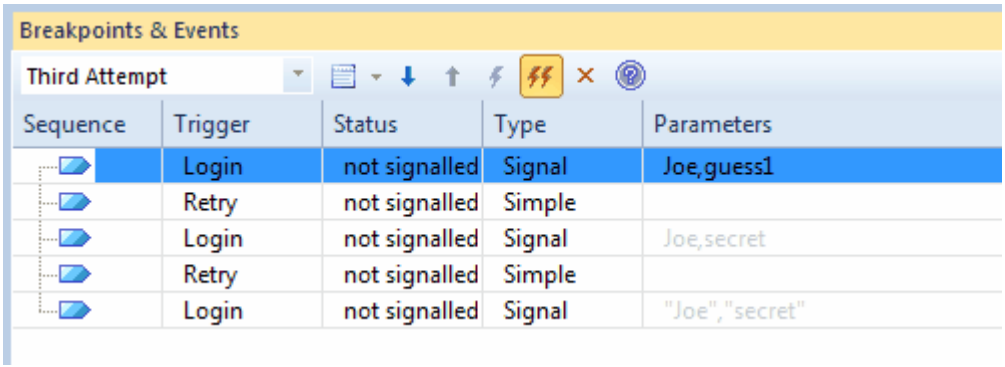
	liste déroulante des ensembles déclencheur , comme dans cette image d'exemple. Une fois sélectionné, il charge la fenêtre Liste Déclencheur avec l'ensemble déclencheur défini. Note que l'élément spécial *<no event set>* signifie qu'aucun ensemble n'est actuellement sélectionné. Au début de chaque simulation, si un ensemble est sélectionné, il sera chargé à nouveau pour le prochain exécuter . Si *<no event set>* est sélectionné, la liste déclencheur sera effacée.  Une fois que vous avez sélectionné un ensemble déclencheur et que la liste des déclencheurs est chargée, vous avez deux options : • Actionnez les déclencheurs manuellement selon les besoins • Utilisez la fonctionnalité de tir automatique pour automatiser entièrement la simulation 
Tir automatique	Le déclenchement automatique est un moyen pratique de rationaliser vos simulations. Une fois que vous avez chargé un ensemble déclencheur , si vous sélectionnez le bouton de déclenchement automatique  , Enterprise Architect sélectionnera automatiquement déclencheurs en attente lorsqu'il atteindra une impasse dans la simulation. En pratique, cela signifie que les ensembles déclencheur correspondant exactement à un chemin dans la simulation exécuter automatiquement sans votre intervention. Comme vous pouvez enregistrer n'importe quel nombre d'ensembles de déclencheur avec différents chemins et paramètres déclencheur , vous pouvez tester et travailler efficacement et rapidement avec de nombreux scénarios différents.
Règles de tir automatique	Lorsqu'une simulation s'exécute avec le déclenchement automatique activé, Enterprise Architect attend jusqu'à ce qu'un point soit atteint où la simulation est « bloquée » ou stable, en attendant qu'un ou plusieurs déclencheurs fassent avancer la simulation. À ce moment-là, le premier déclencheur non déclenché de la liste sera sélectionné et déclenché dans la simulation. Le résultat dépend de la pertinence et peut-être des paramètres du déclencheur . • Si le déclencheur correspond à un déclencheur « en attente », il est immédiatement consommé et la simulation avance • Si le déclencheur ne correspond à aucun déclencheur « en attente » ou à une transition parent possible, alors le déclencheur est « perdu » et la simulation reste dans l'état actuel ; cela correspond à un scénario tel qu'un utilisateur appuyant plusieurs fois de suite sur un bouton « marche » - il n'y a aucun effet autre que celui occasionné par la première pression

Utiliser Ensembles Déclencheur pour simuler une Séquence d'événements

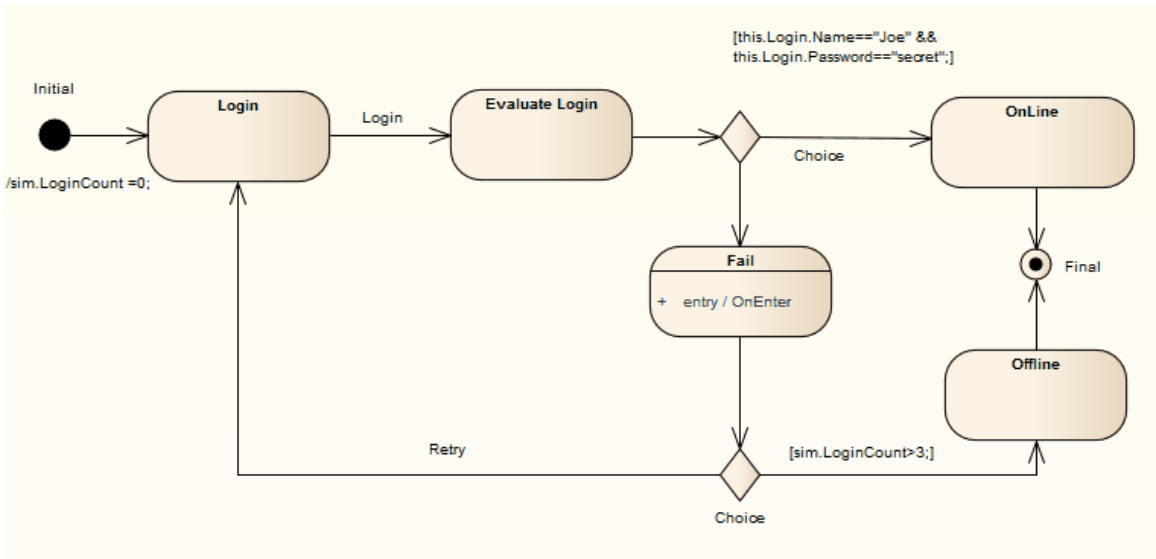
À titre d'exemple simple de l'utilité des ensembles déclencheur , considérons cet exemple d'ensemble déclencheur et diagramme associé.

Dans cet exemple, en utilisant un nom d'utilisateur et un mot de passe, nous simulons un processus de connexion simple de type « trois coups et vous êtes dehors ». Le chemin de réussite attend le nom « Joe » et le mot de passe « secret ». (Note : il est très important que les paramètres référençant des chaînes soient placés entre guillemets, sinon l'interpréteur pense que le nom fait référence à une autre variable dans la simulation.)

- Pass 1 essaie *Joe* et *guess1* - qui échoue
- Le passage 2 essaie *Joe* et *secret* , mais comme ils font référence à des variables et non à des chaînes, cela échoue également
- Le passage 3 montre la manière correcte de référencer les paramètres déclencheur et la simulation réussira



Voici un diagramme simple simulant un processus de connexion nécessitant une paire nom d'utilisateur et mot de passe.



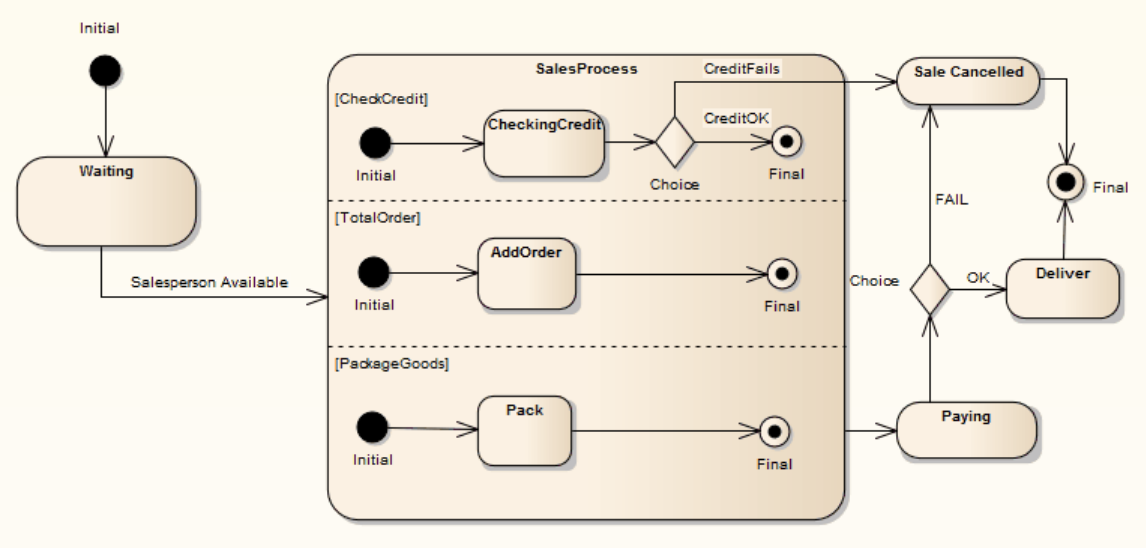
Multi-filetage - Régions State concurrentes

Les régions concurrentes au sein d'un State représentent les changements d'état et le traitement qui se produisent en parallèle au sein d'un State parent global. Cela est particulièrement utile lorsqu'une région déclenche des événements ou modifie des variables de simulation dont dépend une autre région. Par exemple, une région peut contenir un minuteur simulé qui déclenche des événements à des intervalles définis qui invoquent des changements d'état dans les States d'autres régions.

Les régions simultanées sont essentiellement les mêmes que les fourches et Jointures avec une logique et des règles de traitement similaires.

Dans l'exemple :

- Lorsque la transition vers SalesProcess est effectuée, chaque région est activée simultanément
- Le crédit est vérifié, la commande totalisée et les marchandises requises emballées
- Cependant, en cas d'échec de la vérification de crédit, cela déclencheurs la transition vers l'état Vente annulée ; note que lorsque cela se produit, l'ensemble de l'état parent et toutes les régions détenues sont immédiatement quittés, quel que soit leur état de traitement
- Si la vérification de crédit réussit, la région passe à l'état final et une fois que les autres régions ont toutes atteint leur propre état final, l'état parent peut alors être quitté

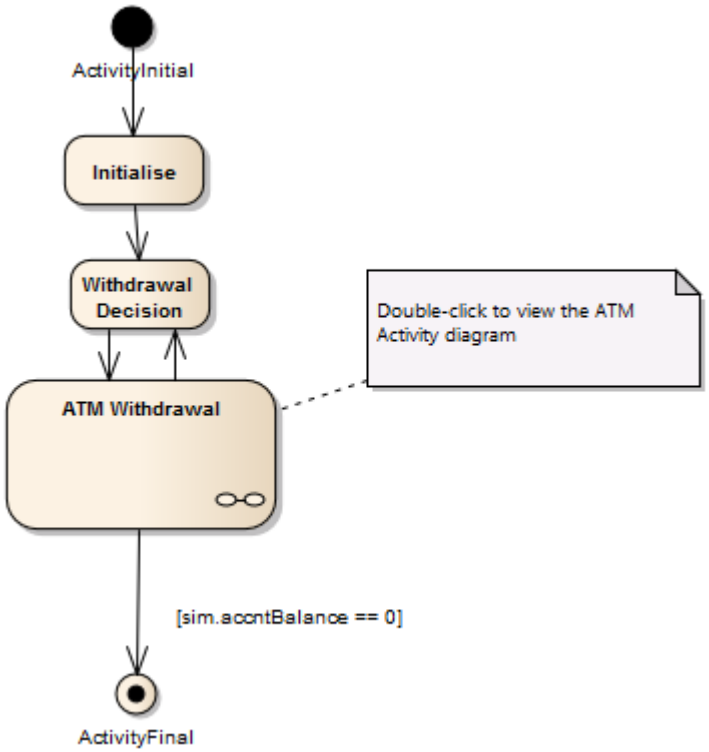


Utilisant Diagrammes composites

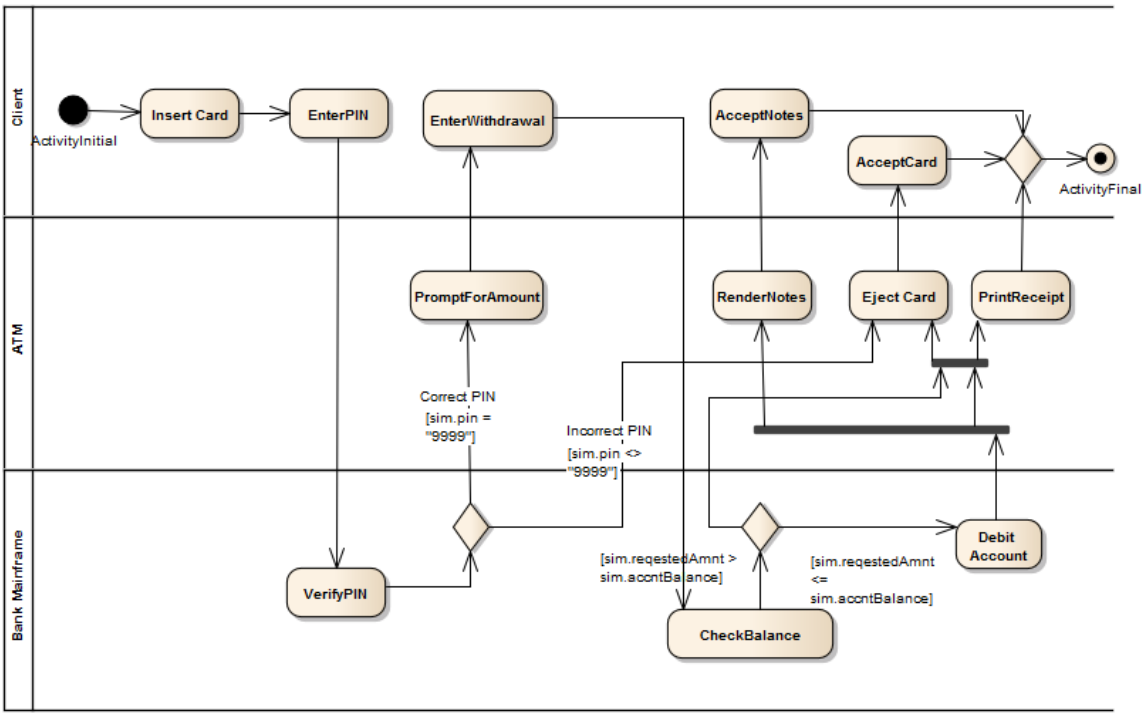
Si vous souhaitez simuler un traitement qui inclut une branche représentée sur un diagramme différent (par exemple, pour réduire la complexité sur le diagramme principal ou pour masquer les zones de traitement qui ne sont traitées qu'en cas d'exception), vous pouvez utiliser un élément Composite pour représenter et accéder à la branche sur son diagramme Composite enfant. Lorsque vous exécuter la simulation et qu'elle atteint l'élément Composite, elle ouvre le diagramme enfant et le traite avant de revenir (le cas échéant) au chemin de traitement principal. Il s'agit d'une excellente méthode pour suivre le chemin de traitement dans un processus complexe, en représentant des sections du processus avec des éléments d'activité composite qui développent le traitement réel dans leurs diagrammes enfants respectifs. Vous pouvez avoir plusieurs éléments Composite représentant différentes étapes ou branches du processus.

Un aspect à surveiller (et qui serait révélé par un échec dans la simulation) est d'avoir plusieurs threads qui traitent simultanément sur diagrammes distincts. La simulation ne peut pas passer à un nouveau diagramme si elle suit également un autre thread sur le diagramme actuel.

Ce diagramme donne un aperçu du processus de retrait d'espèces à un distributeur automatique de billets :



L'activité Retrait au DAB est un élément composite. Si vous double-cliquez dessus, vous ouvrez et affichez le diagramme enfant, qui est une analyse plus détaillée du processus de retrait. De même, une simulation ouvrira et traitera le diagramme enfant.


Client
ATM
Bank Mainframe
ActivityInitial
Insert Card
EnterPIN
EnterWithdrawal
AcceptNotes
AcceptCard
ActivityFinal
PromptForAmount
RenderNotes
Eject Card
PrintReceipt
Correct PIN
[sim.pin =
"9999"]
Incorrect PIN
[sim.pin <>
"9999"]
VerifyPIN
[sim.reqestedAmnt >
sim.acontBalance]
[sim.reqestedAmnt
<=
sim.acontBalance]
Debit
Account
CheckBalance

Simulation d'Interface Utilisateur Win32

Enterprise Architect supporte la simulation de boîtes de dialogue et d'écrans créés avec le profil Win32 ® Interface Utilisateur , pour intégrer la conception de l'interface utilisateur au comportement défini du système. Les boîtes de dialogue peuvent être référencées et invoquées par programmation à l'aide de commandes JavaScript dans un modèle comportemental tel qu'une Statemachine , offrant ainsi une exécution entièrement personnalisable et entièrement interactive de votre modèle comportemental.

Les boutons de commande peuvent être utilisés pour diffuser des signaux, en déclenchant un déclencheur lorsque le bouton est cliqué. Les arguments de signal peuvent être renseignés à partir des champs de saisie le dialogue , par exemple pour capturer et envoyer un nom d'utilisateur et un mot de passe pour évaluation.

Les boîtes de dialogue conçues à l'aide du profil Win32 Interface Utilisateur (et existant dans la même branche Paquetage que le modèle comportemental en cours d'exécution) seront créées en tant que nouvelles fenêtres en arrière-plan au début de la simulation. Diverses propriétés pouvant affecter l'apparence et le comportement de chaque dialogue et contrôle peuvent être personnalisées au moment de la conception via les Valeur Étiquetés fournies par le profil Win32 Interface Utilisateur .

Pour interagir avec un dialogue via JavaScript pendant la simulation, le mot clé de niveau simulation ' dialogue ' est utilisé, suivi d'un point et du nom le dialogue . Propriétés et méthodes sont alors accessibles ; par exemple, pour afficher le dialogue , ou pour définir la valeur de texte d'un 'Edit Control' :

```
dialog.Login.Show=true;

dialog.Login.Username.Text="admin";
```

Exemples

Pour visualiser un exemple de Simulation d'Interface Utilisateur Win32 , ouvrez le modèle EAExample et localisez le diagramme :

Exemple Modèle > Simulation de Modèle > Statemachine Models > Connexion client > Client > Connexion client

Propriétés communes

Ces propriétés et méthodes communes sont disponibles sur la plupart des types de contrôle UI Win32 pris en charge.

Propriété/Méthode	Description
Activer	Booléen Active ou désactive l'interaction de l'utilisateur.
Déplacer vers (x, y, largeur, hauteur)	Déplacez la fenêtre aux coordonnées spécifiées et définissez la hauteur et la largeur de la fenêtre.
Montrer	Booléen Afficher ou masquer le dialogue . Lorsque cette propriété est définie sur False, le dialogue est déplacé hors de l'écran.
Texte	String Définissez le titre de le dialogue ou de la fenêtre.

Fonctions JavaScript

Fonction	Description
BroadcastSignal (string Signal)	Envoie un signal à la file d'attente des événements de simulation. Paramètres: • Signal : String – le nom du signal à diffuser
UIBroadcastSignal (string Signal, tableau Paramètres)	Envoie un signal avec des paramètres supplémentaires à la file d'attente des événements de simulation. Paramètres: • Signal : String – le nom du signal à diffuser • Paramètres : Tableau – paramètres supplémentaires à fournir pour ce signal Exemple: UIBroadcastSignal("Connexion",{Nom : dialogue .Login.Username.Text, Mot de passe : dialogue .Login.Password.Text});
ShowInterface (string InterfaceName, booléen Afficher)	Obsolète . Voir la propriété **Show** sur le contrôle « Dialogue ». Par exemple : dialogue .HelloWorld.Show = vrai;
InterfaceOperation (string InterfaceName, string ControlName, string OperationName,[string arg1],[string arg2])	Obsolète . Les opérations peuvent être référencées directement à partir du contrôle. Par exemple : dialogue .HelloWorld.ListControl.InsertItem(" Test ", 2);
GetInterfaceValue (string InterfaceName, string ControlName, string OperationName,[string arg1],[string arg2])	Obsolète . Propriétés peuvent être référencées directement à partir du contrôle. Par exemple : dialogue .HelloWorld.EditControl.Text;

Notes

- Les commandes doivent être dans un dialogue ; toutes les commandes en dehors d'un dialogue ne seront pas interprétées
- Les boîtes de dialogue et les contrôles doivent être sur une interface Win32 diagramme Interface Utilisateur
- Les contrôles UI simples et les contrôles UI de base peuvent également être utilisés dans une simulation, mais leurs fonctionnalités sont limitées par rapport aux contrôles UI Win32
- Les noms Dialogue et les noms de contrôle doivent être uniques ; si plusieurs contrôles du même nom existent, la simulation ne pourra pas les différencier
- Les espaces dans les noms dialogue et les noms de contrôle sont traités comme des traits de soulignement
- Les noms Dialogue et les noms de contrôle sont sensibles à la casse.

Contrôles UI Win32 pris en charge

Ce tableau identifie tous les contrôles UI Win32 disponibles dans Enterprise Architect pour la conception et la simulation de l'interface utilisateur.

Accéder

Ruban	Design > Diagramme > Toolbox :  > Spécifiez ' Interface Utilisateur - Win32' dans la dialogue ' Trouvez Item de Boîte à Outils '
Raccourcis Clavier	Ctrl+Shift+3 :  > Spécifiez ' Interface Utilisateur - Win32' dans la dialogue ' Trouvez Item de Boîte à Outils '

Contrôles UI Win32

Contrôle	Description
Bouton	Les contrôles de bouton sont un moyen courant de permettre l'interaction de l'utilisateur pendant l'exécution ; par exemple, un bouton OK dans un écran de connexion. Un bouton peut répondre à un événement de clic, défini en ajoutant une Valeur Étiquetée « OnClick ». En réponse à un événement de clic, un bouton peut être utilisé pour, par exemple, envoyer un signal, provoquant le déclenchement d'un déclencheur pendant l'exécution. Propriétés de conception personnalisables : • Client Edge • Bouton par défaut • Désactivé • Plat • Alignement horizontal • Cadre modal • Multiligne • Aligner le texte à droite • Ordre de lecture de droite à gauche • Bordure statique • Tabstop • Transparent • Alignement vertical • Visible Valeur Étiquetés : • OnClick – spécifie une commande JavaScript à exécuter en réponse à un événement de clic sur ce bouton

	Propriétés : • Activer • Montrer • Texte Opérations : • Déplacer vers
Case à cocher	Propriétés de conception personnalisables : • Auto • Client Edge • Désactivé • Plat • Alignement horizontal • Texte de gauche • Cadre modal • Multiligne • Aligner le texte à droite • Ordre de lecture de droite à gauche • Bordure statique • Tabstop • Alignement vertical • Visible Valeur Étiquetés : • OnCheck – spécifie une commande JavaScript à exécuter en réponse à un changement de la valeur de cette case à cocher Propriétés : • Checker – valeur integer [0|1] • Activer • Montrer • Texte
Zone de liste déroulante	Propriétés de conception personnalisables : • Auto • Client Edge • Données – string de valeurs délimitée par des points-virgules pour remplir la zone de liste déroulante au moment de l'exécution ; par exemple, « oui ; non ; peut-être » • Désactivé • A des cordes • Minuscule • Cadre modal • Aligner le texte à droite • Ordre de lecture de droite à gauche • Trier • Bordure statique • Tabstop

	• Type • Majuscule • Défilement vertical • Visible Opérations • AddString (string) • SupprimerTout() • DeleteItem (numéro) – supprimer l'élément à l'index spécifié • DeleteString (string) – supprime tous les éléments correspondant string • Obtenir le nombre () • GetString (nombre) • InsertItem (nombre, string) • InsertString (nombre, string) • SetString (nombre, string) Propriétés : • Activer • Sélection – index de l'élément actuellement sélectionné • Montrer
Dialogue	Propriétés de conception personnalisables : • Alignement absolu • Fenêtre d'application • Bordure - Redimensionnement ou cadre Dialogue uniquement • Centre • Client Edge • Centre de la souris • Clip Frères et sœurs • Désactivé • Barre de défilement horizontale • Barre de défilement gauche • Édition locale • Maximiser la boîte • Réduire la boîte • Pas d'activation • Fenêtre superposée • Fenêtre de palette • Aligner le texte à droite • Ordre de lecture de droite à gauche • Définir le premier plan • Menu système • Système modal • Barre de titre • Fenêtre d'outils • Le plus haut • Transparent

	• Barre de défilement verticale • Visible • Bordure de fenêtre Propriétés : • Activer • Montrer • Texte Opérations : • Déplacer vers
Contrôle d'édition / Contrôle d'édition enrichi	Propriétés de conception personnalisables • Aligner le texte • Défilement automatique HScroll • Défilement automatique • Frontière • Client Edge • Désactivé • Minuscules (contrôle d'édition uniquement) • Cadre modal • Multiligne • Nombre • Mot de passe • Lecture seule • Aligner le texte à droite • Ordre de lecture de droite à gauche • Bordure statique • Tabstop • Transparent • Majuscules (contrôle d'édition uniquement) • Visible • Je veux revenir Propriétés : • Activer • Montrer • Texte
Boîte de groupe	Propriétés de conception personnalisables : • Client Edge • Désactivé • Plat • Alignement horizontal • Cadre modal • Aligner le texte à droite • Bordure statique • Tabstop

	• Visible Propriétés : • Activer • Montrer • Texte
Liste déroulante	Propriétés de conception personnalisables : • Frontière • Client Edge • Désactiver le défilement nul • Désactivé • Barre de défilement gauche • Cadre modal • Aligner le texte à droite • Sélection • Trier • Bordure statique • Tabstop • Défilement vertical • Visible Opérations : • AddString (string) • SupprimerTout() • DeleteItem (numéro) – supprimer l'élément à l'index spécifié • DeleteString (string) – supprime tous les éléments correspondant string • Obtenir le nombre () • GetString (nombre) • InsertItem (nombre, string) • InsertString (nombre, string) • SetString (nombre, string) Propriétés : • Activer • Sélection – index de l'élément actuellement sélectionné • Montrer
Contrôle de liste	Propriétés de conception personnalisables : • Alignement • Toujours afficher la sélection • Frontière • Client Edge • Désactivé • Modifier les étiquettes • Barre de défilement gauche • Cadre modal • Pas d'en-tête de colonne

	• Pas de défilement • Sélection unique • Trier • Bordure statique • Tabstop • Vue • Visible Valeur Étiquetés : • Colonnes – string permettant d'initialiser les noms et les tailles des colonnes pour ce contrôle de liste, séparées par des points-virgules : par exemple, « Colonne1 ; 100 ; Colonne2 ; 150 ; " Opérations : • AddString (string) • SupprimerTout() • DeleteItem (numéro) – supprimer l'élément à l'index spécifié • DeleteString (string) – supprime tous les éléments correspondant à la string • Obtenir le nombre () • GetString (nombre, nombre) • InsertItem (nombre, string) • InsertString (nombre, string) • SetString (nombre, nombre, string) Propriétés : • Activer • Sélection – index de l'élément actuellement sélectionné • Montrer
Contrôle de l'image	L'image initiale du Picture Control peut être définie à l'aide de la Valeur Étiquetée 'Image'. Définissez la valeur sur un nom de fichier accessible par la simulation. L'image peut être modifiée au moment de l'exécution à l'aide de la méthode ChangeImageFile en JavaScript . Cela prend un seul paramètre string du nom de fichier à charger. Définissez la propriété « Type d'image » sur le type de fichier approprié (Bitmap, Métafichier amélioré ou Icône). Ce paramètre ne peut pas être modifié lors de l'exécution. Propriétés de conception personnalisables : • Frontière • Image centrale • Client Edge • Couleur (couleur du cadre) • Désactivé • Type d'image • Cadre modal • Image en taille réelle • Bordure statique • Tabstop • Vue • Visible

	Opérations : • ChangeImageFile (string) - nom de fichier Propriétés : • Montrer
Contrôle de progression	Propriétés de conception personnalisables : • Frontière • Client Edge • Désactivé • Chapiteau • Cadre modal • Lisse • Bordure statique • Tabstop • Verticale • Visible Valeur Étiquetés : • Plage – string spécifiant les valeurs minimales et maximales pour ce contrôle, séparées par un point-virgule : par exemple, « 1 ; 100 » Propriétés : • Activer • Pos • Gamme • Montrer • Étape
Bouton radio	Propriétés de conception personnalisables : • Auto • Client Edge • Désactivé • Plat • Groupe • Alignement horizontal • Texte de gauche • Cadre modal • Multiligne • Bordure statique • Tabstop • Alignement vertical • Visible Valeur Étiquetés : • OnChangeSelection – spécifie une commande JavaScript à exécuter en réponse à un changement de sélection de ce bouton radio Propriétés : • Checker – valeur integer [0|1] • Activer

	• Sélection – valeur integer • Montrer
Contrôle du curseur	Propriétés de conception personnalisables : • Coche automatique • Frontière • Client Edge • Désactivé • Activer la plage de sélection • Cadre modal • Orientation • Indiquer • Bordure statique • Tabstop • Coches • Transparent • Fond transparent • Info-bulles • Visible Valeur Étiquetés : • Plage – string spécifiant les valeurs minimales et maximales pour ce contrôle, séparées par un point-virgule : par exemple, « 1 ; 100 » Propriétés : • Activer • Taille de la page • Pos • Gamme • Montrer
Contrôle de rotation	Propriétés de conception personnalisables : • Alignement • Touches fléchées • Auto Buddy • Client Edge • Désactivé • Cadre modal • Pas de milliers • Orientation • Définir Integer Buddy • Bordure statique • Tabstop • Visible • Envelopper Valeur Étiquetés : • Plage – string spécifiant les valeurs minimales et maximales pour ce contrôle, séparées par un point-virgule : par exemple, « 1 ; 100 »

	Propriétés : • Activer • Pos • Gamme • Montrer
Texte statique / Étiquette	Propriétés de conception personnalisables : • Aligner le texte • Frontière • Client Edge • Désactivé • Fin des points de suspension • Cadre modal • Ellipse de chemin • Pas d'emballage • Notifier • Ellipse de chemin • Aligner le texte à droite • Simple • Bordure statique • Creux • Tabstop • Visible • Ellipse des mots Propriétés : • Activer • Montrer • Texte
Contrôle des onglets	Propriétés de conception personnalisables : • Frontière • Bas • Boutons • Client Edge • Désactivé • Boutons plats • Focus • Piste chaude • Cadre Modèle • Multiligne • Aligner le texte à droite • Bordure statique • Style • Tabstop • Info-bulles

	• Visible Valeur Étiquetés : • Onglets – string spécifiant les noms de chaque onglet pour ce contrôle, séparés par un point-virgule : par exemple, « Onglet 1 ; Onglet 2 ; Onglet 3 ; " Propriétés : • Activer • Montrer
Contrôle des arbres	Propriétés de conception personnalisables : • Toujours afficher la sélection • Frontière • Cases à cocher • Client Edge • Désactiver le glisser-déposer • Désactivé • Modifier les étiquettes • Sélection de ligne complète • A des boutons • A des lignes • Défilement horizontal • Barre de défilement gauche • Lignes à la racine • Cadre modal • Aligner le texte à droite • Ordre de lecture de droite à gauche • Rouleau • Simple Développer • Bordure statique • Tabstop • Info-bulles • Sélection de piste • Visible Opérations : • Delete() - supprime le TreeItem spécifié • InsertItem (string) - chemin en pointillé du nouvel élément d'arborescence à insérer ; tous les éléments parents de ce chemin en pointillé qui n'existent pas encore seront créés automatiquement • InsertString (string) - Voir *InsertItem* • TreeItem (string) - chemin en pointillés de l'élément d'arborescence auquel accéder ; utilisez la propriété *Text* pour définir le texte de cet élément d'arborescence ou utilisez l'opération *Delete* pour supprimer cet élément de l'arborescence Propriétés : • Activer • Sélection – string contenant le chemin en pointillés de l'élément d'arbre sélectionné • Montrer

	• Texte – obtenir ou définir le texte d'un TreeItem spécifié
	Exemples :
	dialogue .MyDialog.MyTreeControl.InsertItem("Racine.Parent.Enfant");
	dialogue .MyDialog.MyTreeControl.TreeItem("Root.Parent.Child").Text = "Modifié";
	dialogue .MyDialog.MyTreeControl.Selection = "Racine.Parent";
	dialogue .MyDialog.MyTreeControl.TreeItem("Root.Parent.Modified").Delete();

Win32 Control Valeur Étiquetés

Diverses propriétés pouvant affecter l'apparence et le comportement de chaque dialogue et contrôle Win32 peuvent être personnalisées au moment de la conception via Valeur Étiquetés fournies par le profil Interface Utilisateur Win32.

Valeur Étiquetés

Certains types de contrôles support l'ajout de Valeur Étiquetés spéciales qui modifient leur comportement.

Les contrôles tels que les boutons, les cases à cocher et les boutons radio peuvent réagir aux événements de l'interface graphique et exécuter une commande JavaScript . Pour permettre à un contrôle de répondre à un événement, créez une nouvelle Valeur Étiquetée avec un nom approprié, par exemple « OnClick », puis saisissez la commande JavaScript dans la valeur .

Les contrôles à onglets peuvent utiliser une Valeur Étiquetée « Onglets » pour définir les onglets qui apparaîtront dans ce contrôle lorsqu'il est simulé.

Les contrôles de curseur, les contrôles de rotation et les contrôles de progression peuvent utiliser une Valeur Étiquetée « Plage » pour définir les valeurs minimales et maximales par défaut acceptées par le contrôle pendant la simulation.

Étiquette	Description
Colonnes	**S'applique à :** Contrôle de liste **Utilisation :** initialise les noms et largeurs de colonnes pour un contrôle de liste. Chaque nom et largeur de colonne sont séparés par un point-virgule ; par exemple, « Colonne1 ; 100 ; Colonne2 ; 150 ; " .
Sur clic	**S'applique à :** Bouton **Utilisation :** identifie la commande JavaScript à exécuter en réponse à un événement de clic sur un contrôle Button.
Survérification	**S'applique à :** case à cocher **Utilisation :** identifie la commande JavaScript à exécuter en réponse à un changement de valeur d'un contrôle Check Box.
Sélection sur le changement	**S'applique à :** bouton radio **Utilisation :** identifie la commande JavaScript à exécuter en réponse à un changement de valeur d'un contrôle Radio Button.
Gamme	**S'applique à :** Contrôle du curseur, Contrôle de rotation, Contrôle de la progression **Utilisation :** spécifie les valeurs minimales et maximales par défaut du contrôle, séparées par un point-virgule : par exemple, « 1 ; 100 ».
Onglets	**S'applique à** : Contrôle des onglets **Utilisation :** spécifie le nom de chaque onglet à créer pour le contrôle d'onglets, séparé par un point-virgule : par exemple, « Onglet 1 ; Onglet 2 ; Onglet 3 ; " .

Simulation BPMN

La simulation BPMN est une méthode de visualisation et de validation du comportement de vos diagrammes BPMN Processus Métier . Grâce aux indications visuelles de toutes les activités en cours d'exécution et des activités possibles qui peuvent être exécutées ensuite, vous pourrez facilement identifier et résoudre les problèmes potentiels du processus que vous avez modélisé.

La simulation de modèles BPMN est similaire à la simulation de modèles Comportementale UML standard, sauf que BPMN :

- Utilise des types d'éléments différents (tels que Passerelle au lieu de Décision) et
- Fonctionne sur des scripts placés, généralement, dans le champ ' Valeur Étiquetée ' approprié associé aux connecteurs et éléments, au lieu des champs ' Propriétés ' (et, si vous préférez, plutôt que dans la dialogue ' Analyseur d'Exécution Build Scripts ') ; le script est écrit en JavaScript

Travailler avec Simulation BPMN

Activité	Détail
Créer un Modèle de Simulation BPMN	Lorsque vous créez un modèle BPMN adapté à la simulation, vous prenez en compte la manière dont vous représentez le point de départ, le flux et les conditions à tester.
Comparer les activités UML aux Processus BPMN	La simulation des modèles BPMN Processus Métier présente un certain nombre de différences avec la simulation des diagrammes d'activité UML .

Notes

- La simulation BPMN est disponible dans les éditions Unified et Ultimate d' Enterprise Architect

Créer un Modèle de Simulation BPMN

Dans le cadre du processus de développement d'un modèle de simulation, déterminez laquelle des trois options de réalisation de la simulation vous préférez appliquer :

- Exécutez un script de simulation pour initialiser les variables du diagramme - sélectionnez « BPMN » comme plate-forme, exécutez la simulation comme « En tant que script » et sélectionnez le script ; vous définirez ensuite les conditions et les décisions comme des déclarations JavaScript dans les Valeur Étiquetés des éléments et des connecteurs sur le diagramme , soit avant de démarrer la simulation, soit pendant la simulation
- N'utilisez pas de script, mais initialisez les variables dans la première activité et, encore une fois, modifiez les conditions et les décisions dans les Valeur Étiquetés des éléments et des connecteurs, puis exécutez la simulation comme « Interprétée » ; vous pouvez ensuite réinitialiser les variables pendant la simulation, ainsi que les conditions
- Exécutez la simulation en mode « Manuel » et gérez le flux et les conditions manuellement à chaque étape

Créer un diagramme BPMN adapté à la simulation

Étape	Action
1	Créez un diagramme Processus Métier ou BPEL à partir de la technologie BPMN 2.0. Si vous créez un diagramme BPEL, Enterprise Architect affiche des boîtes de dialogue spécialisées pour simplifier la création de modèles conformes.
2	Nous vous recommandons de créer un événement Démarrer pour indiquer clairement où démarre votre simulation. Vous avez plusieurs choix pour le Type d'événement ; le choix n'influence pas la simulation de votre modèle. Si aucun Événements Démarrer n'est défini, la simulation démarrera à partir d'une activité qui n'a pas de flux Séquence entrants.
3	Ajoutez toutes les activités impliquées dans le processus modélisé. Vous avez plusieurs choix pour le Type de tâche ; le choix n'influence pas la simulation de votre modèle. Le comportement des activités peut être décomposé davantage en spécifiant un Type d'activité de sous-processus et en sélectionnant Embedded ou CallActivity. Les boucles standard sont également prises en charge.
4	Ajoutez des flux Séquence entre vos activités. Dans la dialogue « Propriétés BPEL », vous pouvez saisir la condition qui doit être satisfaite (True) avant que la Flux séquence ne soit suivie. Vous pouvez également définir le type de condition sur « Par défaut » pour garantir que ce flux sera pris si toutes les autres branches échouent à la condition spécifiée. Si vous ne travaillez pas avec un diagramme BPEL, vous utilisez les valeurs conditionExpression et conditionType Valeur Étiquetés .
5	Ajoutez Événements de fin pour toutes les conditions qui provoqueront la fin du processus ou du chemin d'exécution actif. Vous avez plusieurs choix pour le Type d'événement ; parmi ceux-ci, seul le type Terminate influencera l'exécution. Dans les simulations avec plusieurs nœuds actifs, cela provoque la fin de l'ensemble du processus au lieu de seulement du thread qui atteint ce nœud.

Notes

- Pour inclure des activités qui se trouvent dans Paquetages externes au Paquetage simulé, dessinez un :
  - Connecteur Paquetage Import depuis le Paquetage contenant le diagramme
  étant simulé pour chaque Paquetage externe, ou
  - Connecteur de dépendance du Paquetage contenant le diagramme

étant simulé pour chaque activité dans les Paquetages externes

Initialiser Variables et Conditions

Pour un modèle de simulation BPMN, vous pouvez initialiser vos variables dans un script Analyseur d'Exécution . Vous pouvez également initialiser ces variables dans les Valeur Étiquetés du premier élément Activité du processus, ce qui vous donne une plus grande flexibilité pour ajouter et modifier des variables au fur et à mesure de la simulation. De même, vous pouvez définir les conditions et les valeurs à appliquer aux différents points de décision (Gateways) du processus, dans les Valeur Étiquetés des connecteurs Flux séquence .

Si vous souhaitez intégrer une interface utilisateur dans votre processus de simulation, en utilisant Win32, vous utilisez à nouveau Valeur Étiquetés pour identifier le dialogue ou prompt à afficher, dans l'élément Activité juste avant le point auquel la valeur ou la décision est traitée.

Pour la simulation de diagrammes UML , les variables à l'intérieur de l' object « sim » et object « this » sont affichées dans la fenêtre Variables locales.

Accéder

Affichez l'onglet 'Tags' de la fenêtre Propriétés en utilisant l'une des méthodes décrites ici.

Ruban	Explorer > Portails > Windows > Propriétés > Propriétés > Étiquettes
Raccourcis Clavier	Ctrl+2 > onglet 'Tags' de la fenêtre Propriétés

Initialiser les variables

1. Sur le diagramme , cliquez sur le premier élément Activité du processus.
2. Dans l'onglet 'Tags' de la fenêtre Propriétés , cliquez sur la flèche déroulante du champ « valeur » de taskType et sélectionnez « Script ».
3. Dans le champ « valeur » du script, saisissez le code JavaScript approprié, tel que :

   sim.loan=true; sim.status="undefined';

Définir les conditions

1. Sur le diagramme , cliquez sur un connecteur Flux séquence issu d'un élément Passerelle .
2. Dans l'onglet 'Tags' de la fenêtre Propriétés , cliquez sur la flèche déroulante du champ « Valeur » de conditionType et sélectionnez « Expression ».
3. Dans le champ « Valeur » de conditionExpression (<mémo>*), cliquez sur le bouton  pour afficher la fenêtre Note Valeur Étiquetée . Type le code JavaScript approprié, par exemple :
   sim.status=="Maintenir"
4. Cliquez sur le bouton OK . Le texte de l'instruction s'affiche comme étiquette du connecteur.

Incorporer Interface Utilisateur Win32

1. Sur le diagramme , cliquez sur l’élément Activité qui représente l’endroit où la décision est prise.

2. Dans l'onglet 'Tags' de la fenêtre Propriétés , cliquez sur la flèche déroulante du champ « taskType valeur » et sélectionnez « Script ».
3. Dans le champ « script valeur », saisissez le code JavaScript approprié, tel que :
   dialogue .Screen1.Show=Vrai;
   (Cette instruction affiche le dialogue Screen1. Vous pouvez masquer temporairement le dialogue en changeant « Afficher » sur False.)

Comparaison des activités UML et Processus BPMN

L'exécution et la simulation de modèles BPMN présentent un certain nombre de différences par rapport à l'exécution et à la simulation de diagrammes d'activité UML . La mise en correspondance de concepts similaires et les différences entre les deux méthodes d'expression du comportement d'un système sont présentées ici.

Comparaison des activités UML et Processus BPMN

Activité UML	BPMN Processus Métier
Le point de départ est défini par un nœud initial. Aucune méthode permettant de spécifier la raison pour laquelle l'activité a été démarrée n'est disponible.	Le point de départ est défini par un événement Démarrer . Cela implique une cause spécifique pour le démarrage de l'activité, bien qu'elle puisse être non spécifiée.
L'unité de comportement de base d'une activité est l'élément Action . UML fournit de nombreuses formes différentes d'actions, bien que la simulation n'utilise qu'un petit sous-ensemble de celles-ci.	L'unité de comportement de base d'une activité est l'élément Activité. Plusieurs types de tâches différents sont disponibles. Ceux-ci décrivent généralement différentes méthodes d'exécution (par exemple, manuelle) par opposition à ce qui se passe.
Un flux de contrôle est utilisé pour connecter les éléments d'un diagramme d'activité. Une fonctionnalité distinctive est qu'un seul flux de contrôle peut être suivi à partir de n'importe quel nœud, à l'exception d'un nœud de fourche explicite. Pour restreindre le flux sur un flux de contrôle, ajoutez une garde.	Une Flux séquence permet de relier les éléments sur un diagramme Processus Métier . Ceux-ci diffèrent des diagrammes d'activité UML dans la mesure où tous les flux de séquence valides sont pris par défaut. Pour restreindre le flux sur une Flux séquence définissez la conditionType Valeur Étiquetée sur 'Expression' et créez le script dans la conditionExpression Valeur Étiquetée .
Un nœud Décision est utilisé pour modéliser explicitement une décision en cours de prise. Un nœud Merge, qui utilise la même syntaxe, est utilisé lorsque les flux potentiels sont combinés en un seul.	Un nœud Passerelle défini sur « Exclusif » est utilisé lorsqu'un seul chemin doit être sélectionné. Il est également utilisé pour combiner à nouveau les flux potentiels. Une direction peut être spécifiée comme « Convergente » ou « Divergente » pour sélectionner explicitement entre les deux modes.
Un nœud Fork est utilisé pour exécuter simultanément plusieurs nœuds, tandis qu'un nœud	Un nœud Passerelle défini sur « Parallèle » est utilisé pour modéliser explicitement l'exécution simultanée de plusieurs nœuds. Il est également utilisé pour attendre que tous les flux entrants soient disponibles et repartir avec un seul flux. Une direction peut être spécifiée comme « Convergente » ou « Divergente » pour sélectionner

Join, utilisant la même syntaxe, est utilisé pour attendre que tous les flux entrants soient disponibles et partir avec un seul flux.	explicitement entre les deux modes.
Il n'est pas possible d'exécuter simultanément uniquement certaines sorties d'un nœud pour les activités UML . Si vous en avez besoin, ajoutez ultérieurement des flux de contrôle avec les protections appropriées.	Un nœud Passerelle défini sur Inclusive est utilisé pour modéliser explicitement la situation dans laquelle tous les flux sortants avec une condition vraie sont exécutés simultanément.
Une Action de comportement d'appel est utilisée lorsque le comportement doit être davantage décomposé en faisant référence à une activité externe.	Les éléments d’activité sont définis comme un sous-processus CallActivity lorsque le comportement doit être davantage décomposé en faisant référence à une activité externe.
Activité Action Appel Comportement Action .	Les éléments d’activité sont définis comme un sous-processus intégré lorsque le comportement doit être décomposé davantage sans faire référence à une activité externe.

BPSim Simulations Métier

La spécification ouverte BPSim fournit un ensemble complet de documents sur la manière de configurer et d'attribuer des ressources aux activités ou aux tâches, de déclencher des événements, de prendre des décisions et d'autres capacités concrètes. Une fois configuré selon la spécification BPSim, un modèle de processus métier (construit dans BPMN) peut être transmis à un moteur de simulation BPSim approprié et exécuter selon le processus défini dans le modèle BPMN, à l'aide des données de configuration jointes dans les informations BPSim.

La spécification BPSim est très détaillée et offre au modélisateur et au stratège commercial intéressé une flexibilité sans précédent dans l'attribution d'informations opérationnelles à un modèle, puis dans l'évaluation de la qualité de la solution en fonction des informations reçues du moteur Simulation . Cette section décrit en détail les différents écrans et options disponibles lors de la configuration d'un modèle pour l'exécution de BPSim.

Sparx Systems fournit un simulateur compatible BPSim - le **Moteur d'Exécution BPSim.** Ce Add-In s'intègre aux modèles BPSim et BPMN définis dans Enterprise Architect , offrant la possibilité d' exécuter et de stocker les résultats de plusieurs simulations et d'effectuer des comparaisons pratiques sur l'ensemble des résultats de chaque configuration.

Le **Moteur d'Exécution BPSim** est un prérequis pour accéder et utiliser les facilités de configuration de BPSim. Le Moteur d'Exécution est intégré aux éditions Unified et Ultimate d' Enterprise Architect ; pour une utilisation dans l'édition Corporate , il peut être acheté et installé sous licence séparée.

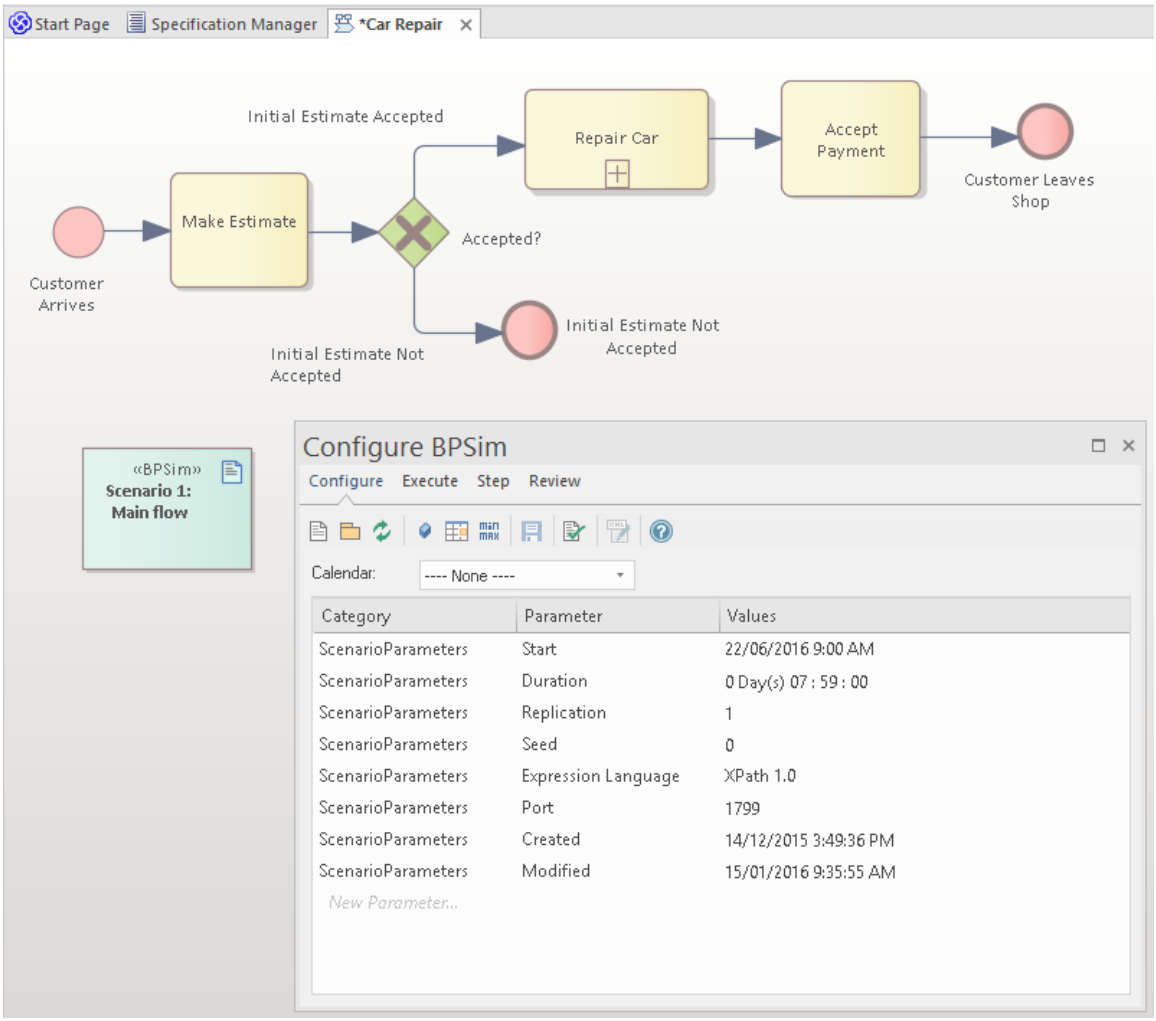
Une fois que vous avez défini une configuration BPSim, le processus d'exécution de la simulation exporte le modèle BPMN avec ses données BPSim sous une forme standard. Cela garantit que les modifications apportées au modèle sont toujours intégrées à la simulation. De même, le processus d'exportation du modèle capture le modèle BPMN avec ses données BPSim sous une forme qui peut être importée dans un autre modèle et utilisée par le Moteur d'Exécution BPSim Sparx Systems ou par tout autre moteur BPSim conforme aux normes.

Installer BPSim

Bien que BPSim soit intégré aux éditions Unified et Ultimate d' Enterprise Architect , il est distinct de l'édition Corporate et - après l'achat - doit être installé sur votre système.

Pour les trois éditions, vous devez vous assurer que les bonnes versions de Java Runtime Environment (JRE) et de Java Development Kit (JDK) sont également installées sur votre système.

Modèle BPMN avec Simulation BPMN



La fenêtre Configurer BPSim vous permet de définir plusieurs catégories de paramètres Simulation , chaque catégorie se concentrant sur un aspect de la configuration Simulation . Par exemple, vous pouvez définir :

- ScenarioParameters, qui définissent comment la Simulation elle-même doit se dérouler
- Paramètres de contrôle, qui examinent la manière dont l'activité se déroule dans le processus métier, modérés par la probabilité d'une séquence d'événements et les priorités de certains événements
- Paramètres temporels, qui examinent comment la durée d'une ou plusieurs phases du traitement d'une activité influence le processus métier
- Paramètres des ressources, qui examinent l'implication des types et des rôles des travailleurs et d'autres ressources, leur nombre requis, leurs coûts et leur disponibilité

Vous pouvez également gérer plusieurs versions d'une configuration (en tant qu'artefacts BPSim distincts) et comparer facilement les différences entre les versions pour voir comment chaque configuration modifiera le flux de la Simulation proposée ou l'exécution du processus. Vous pouvez, par exemple, établir une configuration de base, puis créer plusieurs configurations « et si ? » qui font varier un ou plusieurs paramètres. Une fois que vous avez exécuter ces configurations via un moteur Simulation , vous pouvez examiner chaque résultat et décider des mérites relatifs de chaque configuration. Un principe utile à appliquer ici est l'héritage simple de données communes et inchangées dans une configuration par une autre configuration qui ne contient que les données modifiées ; vous pouvez donc exécuter la simulation sur un ensemble actuel de variables, qui s'appuie en même temps sur la configuration de données standard.

Les utilisateurs peuvent combiner les facilités BPSim et Charting pour faire varier, simuler et comparer rapidement les aspects d'un modèle Processus Métier , et montrer les différences entre les simulations dans l'un des nombreux formats de graphiques.

Si vous travaillez sur plusieurs projets, vous pouvez exporter et importer les configurations BPSim entre eux. La configuration intègre automatiquement le modèle BPMN 2.0 sur lequel elle est basée.

L'outil de configuration Enterprise Architect Processus Métier Simulation est basé sur le Framework BPSim développé

par la Workflow Management Coalition (WfMC).

Notes

- Si vous cliquez sur un élément ou un connecteur de processus métier dans un diagramme ou dans la fenêtre Navigateur , il est mis en surbrillance et sélectionné dans la fenêtre Configurer BPSim
- Le Processus Métier que vous simulez peut contenir des éléments de plusieurs Paquetage ; pour inclure les éléments externes dans la simulation, vous devez créer un diagramme Paquetage contenant le Paquetage 'parent' et soit les Paquetages 'externes' contenant les éléments externes, soit les éléments externes eux-mêmes ; créez un :
  - Connecteur d'importation de Paquetage du Paquetage parent vers chaque Paquetage externe, ou
  - Connecteur de dépendance du Paquetage parent à chaque élément externe

Installation de BPSim

Bien que BPSim soit intégré aux éditions Unified et Ultimate d' Enterprise Architect , il doit être installé sur votre système. Pour ces éditions, vous devez vous assurer que les bonnes versions de Java Runtime Environment (JRE) et Java Development Kit (JDK) sont également installées sur votre système.

Installer JDE et JDK

Pour utiliser le Moteur d'Exécution Sparx Systems BPSim, vous devez avoir sur votre système Java Runtime Environment (JRE) version 1.7 ou supérieure et, si votre configuration BPSim contient des paramètres de propriété, vous devez également avoir Java Development Kit (JDK) version 1.7 ou supérieure.

Vous n'avez pas besoin d'effectuer d'autres configurations pour le moteur, sauf si vous disposez de plusieurs versions de JRE/JDK sur votre système et que vous souhaitez spécifier la version que le moteur d'exécution doit utiliser. Dans ce cas, appliquez ces variables d'environnement comme indiqué :

1. Cliquez sur l'icône Windows « Démarrer » et sélectionnez l'option « Ordinateur ».
2. Dans le menu bannière, sélectionnez l'option « Propriétés système ».
3. Dans le panneau latéral, sélectionnez l’option « Paramètres système avancés ».
4. Dans l'onglet « Avancé » de la dialogue « Propriétés système », cliquez sur le bouton Variables d'environnement.
5. Dans la dialogue « Variables d’environnement », dans le panneau « Variables système », cliquez sur le bouton Nouveau.
6. Dans la dialogue « Nouvelle variable système », complétez les champs avec les valeurs affichées :

   Pour JRE : Nom de la variable : MDG_BPSIM_JRE_HOME
   valeur de la variable : C:\Program Files\Java\jre7

   Pour JDK : Nom de la variable : MDG_BPSIM_JDK_HOME
   valeur de la variable : C:\Program Files\Java\jdk1.7.0_51
7. Cliquez sur le bouton OK .
8. Vous devez redémarrer votre machine pour que les nouvelles variables prennent effet.

Configuration de BPSim

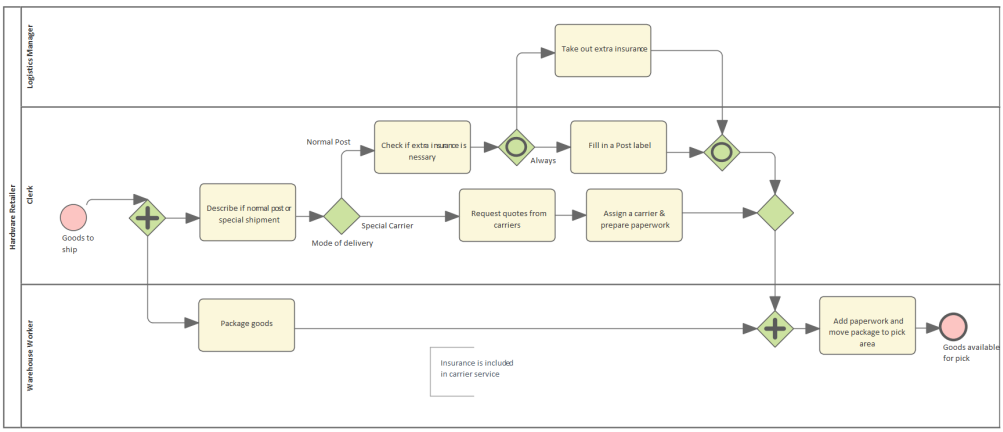
Une configuration Processus Métier Simulation (BPSim) est représentée et contenue dans un élément Artefact Processus Métier Simulation , que vous pouvez créer sur un diagramme dans n'importe quel Paquetage du même projet que le modèle BPMN avec lequel vous travaillez.

Créer un Modèle Processus Métier

Chaque configuration BPSim est créée spécifiquement pour et à partir d'un Processus Métier existant, défini dans BPMN. Vous devrez donc créer ou importer le modèle BPMN sur lequel la configuration doit être basée, avant d'utiliser l'artefact Simulation Processus Métier .

Cet exemple diagramme peut être trouvé et exploité dans le modèle EAExample, dans :

Analyse et Modélisation Métier > Exemples BPMN 2.0 > Diagrammes de processus > Processus d'expédition d'un détaillant en quincaillerie



Créer un artefact Simulation Processus Métier

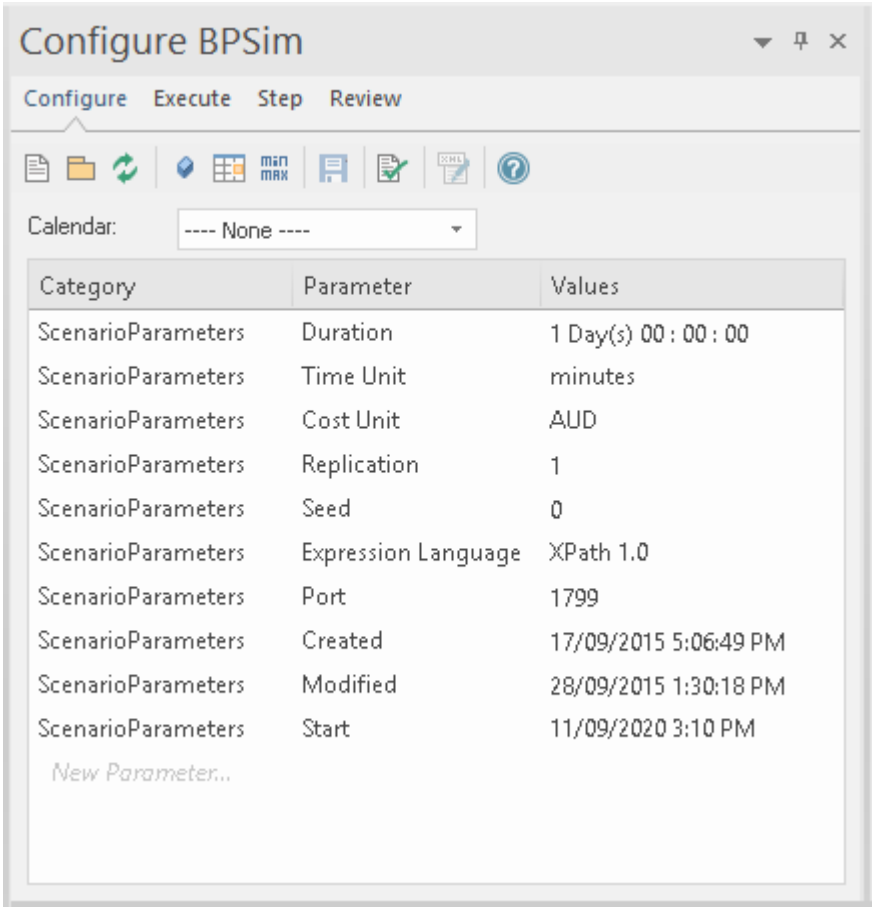
Ouvrez un diagramme dans lequel créer l'Artefact, et affichez la Boîte à outils Diagramme (appuyez sur Ctrl+Maj+3). Développez la page commune ' Simulation ' et faites glisser l'icône ' Processus Métier Simulation ' sur le diagramme .

Lorsque vous configurez l'artefact, envisagez de créer un artefact pour définir la configuration de base et d'autres artefacts pour définir des variations ou des ajouts dans certains aspects de la simulation. Vous utiliserez des connecteurs de généralisation entre les artefacts « de base » et « de variation » afin que les variations héritent des données que vous avez définies dans l'artefact « de base ». De cette façon, vous n'avez pas besoin de redéfinir sans cesse la configuration complète dans chaque artefact que vous créez.

Double-cliquez sur l'élément et donnez-lui un nom approprié, tel que (pour l'exemple) « Configuration BPSim de base ».

Présentation de la fenêtre Configurer BPSim

Cliquez-droit sur l'élément Artefact (soit dans le diagramme , soit dans la fenêtre Navigateur) et sélectionnez l'option 'Configurer BPSim'. La fenêtre Configurer BPSim s'affiche pour l'Artefact.



Cette fenêtre contient quatre onglets : Configurer, Exécuter, Étape et Révision .

- Configurer : configurer les paramètres BPSim pour chaque élément BPMN ; définir les paramètres de propriété, les calendriers et les paramètres de scénario
- Exécuter : exécuter le modèle BPMN avec une configuration BPSim
- Étape : passer au-dessus/entrer pour fournir un aperçu du processus d'exécution, y compris l'état du jeton, les valeurs des propriétés et les allocations de ressources par heure/étape
- Révision : révision / comparaison des artefacts de configuration, génération de rapports de résultats de simulation standard ou personnalisés

BPSim - Page de configuration

L'Artefact BPSim sera configuré sur un Paquetage . Tous les éléments BPMN sous ce Paquetage ou ses sous-Packages seront chargés. Par défaut, le Paquetage contenant cet Artefact sera configuré lors du chargement dans cette fenêtre.

Cette fenêtre est contextuelle. Lorsqu'un élément est sélectionné sur un diagramme ou dans la fenêtre Navigateur , la liste affiche les configurations actuelles de l'élément ; de plus, les listes déroulantes de valeurs n'affichent que les paramètres disponibles pour l'élément.

Lorsque l'artefact BPSim est l'élément de contexte, la liste affichera les ScenarioParameters.

Accéder

Ruban	Simuler > Analyse de Processus > BPSIM > Ouvrir BPSIM Manager > Configurer

Options de la barre d'outils

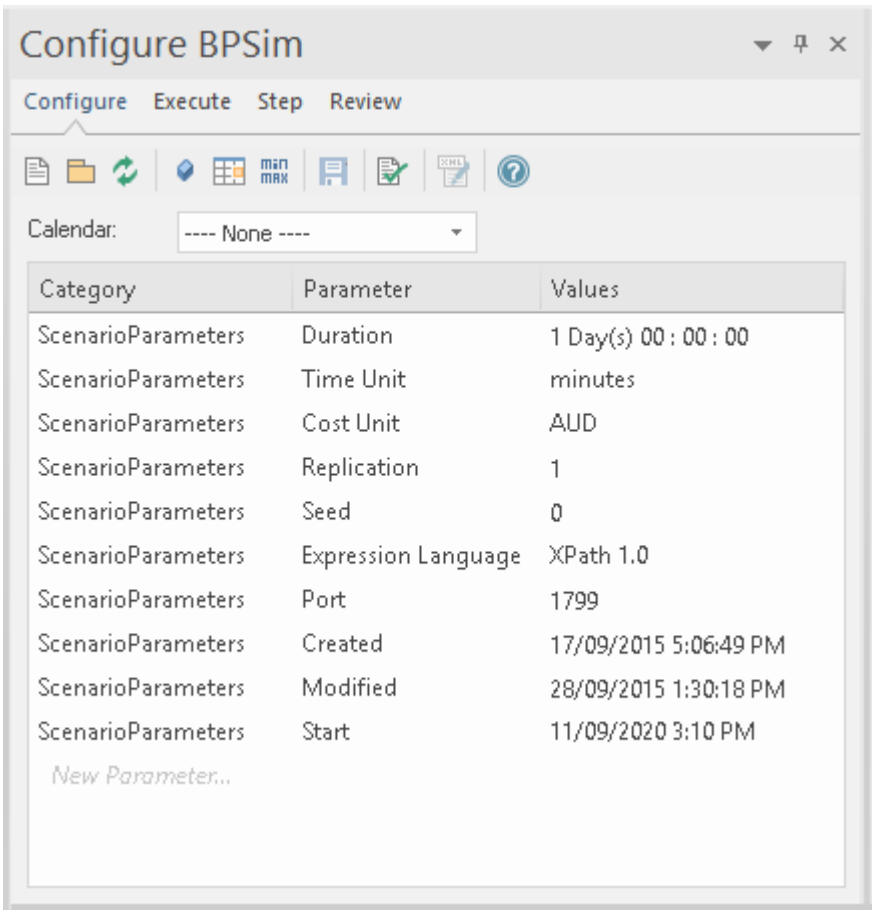
Option	Description
	Cliquez sur ce bouton pour sélectionner ou créer un élément BPSimConfiguration.
	Cliquez sur ce bouton pour définir un Paquetage pour l'Artefact BPSim. Tous les éléments BPMN sous ce Paquetage ou ses sous-Packages seront inclus.
	Cliquez sur ce bouton pour recharger les éléments BPMN à partir des Paquetages configurés. Par exemple, lorsque certains éléments BPMN sont modifiés, exécuter cette commande pour recharger le Paquetage afin que les modifications soient prises en compte pour BPSim Simulation .
	Cliquez sur ce bouton pour définir Propriétés , qui peuvent être utilisées comme paramètres de propriété sur les éléments BPMN.
	Cliquez sur ce bouton pour définir les calendriers, qui peuvent être utilisés pour configurer les paramètres des éléments.
	Cliquez sur ce bouton pour afficher ou masquer la colonne « Demande de résultat ». La configuration de la demande de résultat est requise pour une simulation personnalisée. Le rapport d'exécution ne contiendra que les résultats demandés.
	Cliquez sur ce bouton pour enregistrer les informations de la fenêtre Configurer BPSim dans un élément Artefact BPSim.
	Cliquez sur ce bouton pour valider le modèle BPMN et les configurations BPSim. Des messages d'erreur ou d'avertissement peuvent s'afficher dans la fenêtre Sortie système s'ils sont générés.
	Cliquez sur ce bouton pour exporter le modèle BPMN avec la configuration BPSim. Ce fichier BPMN exporté est conforme aux spécifications BPMN et BPSim et peut être utilisé par des moteurs d'exécution BPSim tiers.

Paramètres du scénario

Un scénario est composé d'un ensemble de paramètres d'éléments. Le scénario lui-même définit les paramètres utilisés par tous les éléments en tant que paramètres globaux. Tous les paramètres ne s'afficheront pas pour un élément, mais vous pouvez les ajouter à la liste en procédant comme suit :

1. Cliquer sur le *nouveau paramètre* texte, en cliquant sur la flèche déroulante et en sélectionnant « ScenarioParameter ».
2. Cliquez sur la flèche déroulante dans le champ « Paramètre » puis sélectionnez le type de paramètre dans la liste.

Note qu'une fois que vous avez ajouté tous les paramètres possibles pour un scénario, la fenêtre Configurer BPSim ne vous permet pas de tenter d'en ajouter davantage.



Nom	Description
Démarrer	La date et l'heure à laquelle le processus commence à prendre effet. Vous pouvez modifier ceci en remplaçant les valeurs ou, pour la date, en la sélectionnant dans un calendrier déroulant.
Durée	La durée du processus. Le paramètre « Durée » est une valeur obligatoire. Elle doit être suffisamment longue pour permettre une simulation complète ; par exemple, si un processus (et donc sa simulation) prend trois heures pour se terminer, le paramètre « Durée » doit être défini sur une valeur supérieure à trois heures. Vous pouvez modifier ceci en remplaçant le segment approprié au format « jours

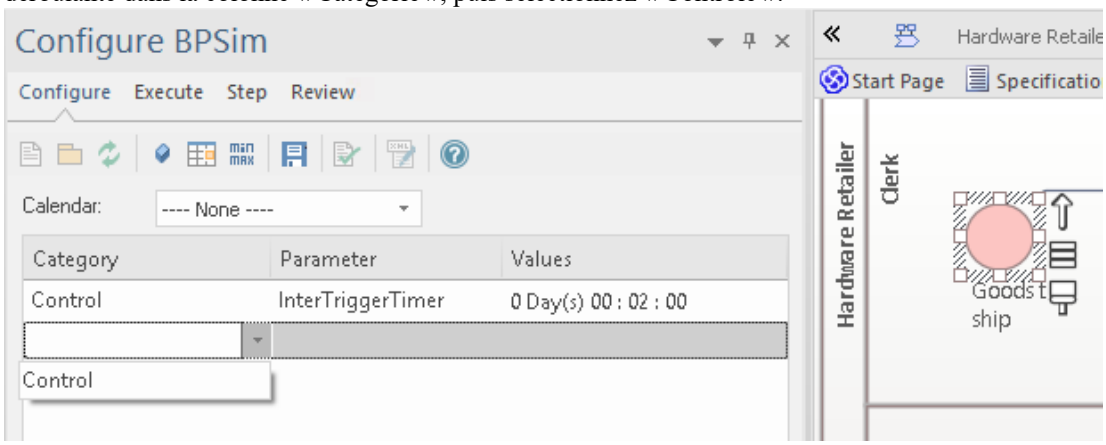
	heures : minutes : secondes ».
Unité de temps	Unité de base dans laquelle les périodes de temps sont exprimées dans ce scénario. Toutes les valeurs numériques et flottantes représentant le temps doivent être considérées comme étant exprimées dans cette unité, sauf si elles sont remplacées localement. Vous pouvez modifier cela en cliquant sur la flèche déroulante et en sélectionnant l'unité.
Unité de coût	Unité monétaire de tous les coûts enregistrés dans le processus. Toutes les valeurs numériques et flottantes représentant un coût doivent être considérées comme étant exprimées dans ce code de devise, sauf si elles sont remplacées localement. Vous pouvez modifier ceci en cliquant sur la flèche déroulante et en sélectionnant l'abréviation de l'unité.
Réplication	Nombre de réplications du scénario à exécuter. La valeur par défaut est 1. Vous pouvez modifier ceci en saisissant simplement une valeur .
Graine	Une graine aléatoire à utiliser pour initialiser un générateur de nombres pseudo-aléatoires. Vous pouvez modifier ceci en saisissant simplement une valeur .
Langage d'expression	XPath 1.0 et Java - XPath 1.0 est le langage par défaut. Si Java est spécifié comme langage d'expression, JDK Home doit être défini. Vous pouvez modifier cela en cliquant sur la flèche déroulante et en sélectionnant la langue.
Module DMN	Lorsque des tâches de règles métier sont utilisées dans le Modèle BPMN, vous pouvez implémenter ces tâches en tant que Modèle DMN. Vous pouvez d'abord créer un Modèle DMN et générer un serveur DMN en Java, puis cliquer sur le bouton ... pour spécifier le fichier de serveur DMN généré.
Accueil JRE	Le Moteur d'Exécution Enterprise Architect BPSim fonctionne dans un environnement Java, il faut donc spécifier un répertoire JRE Home. Cliquez sur le bouton ... pour choisir un répertoire ; par exemple, C:\Program Files\Java\jre7. Vous pouvez modifier ceci en cliquant à nouveau sur le bouton ... pour parcourir le répertoire.
Accueil JDK	Lorsque le langage d'expression est Java, le Moteur d'Exécution Enterprise Architect BPSim va générer du code Java et compiler avec javac comme extension fournisseur. Il faut donc spécifier un répertoire JDK Home. Utilisez le bouton ... pour choisir un répertoire (par exemple, C:\Program Files\Java\jdk1.7.0_80). Vous pouvez modifier ceci en cliquant à nouveau sur le bouton ... pour parcourir le répertoire.
Port	Numéro de port utilisé par Enterprise Architect pour communiquer avec le Moteur d'Exécution BPSim. Le numéro de port par défaut est 1799.
Créé	Champ en lecture seule. L'horodatage de la création de l'artefact BPSim.
	Champ en lecture seule. L'horodatage de la dernière modification de l'artefact

Modifié	BPSim.

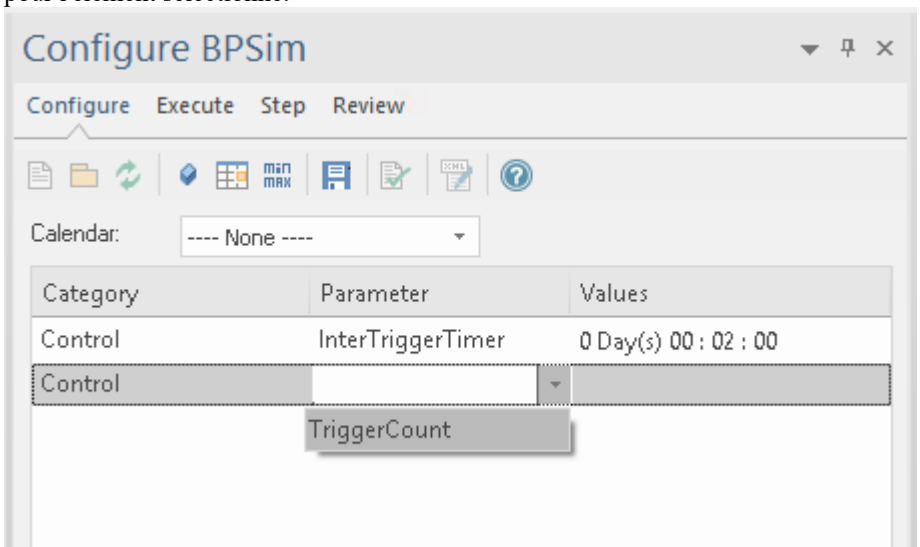
Paramètres de contrôle

Pour commencer à définir les paramètres de contrôle pour l'élément approprié (tel qu'un événement ou Passerelle) :

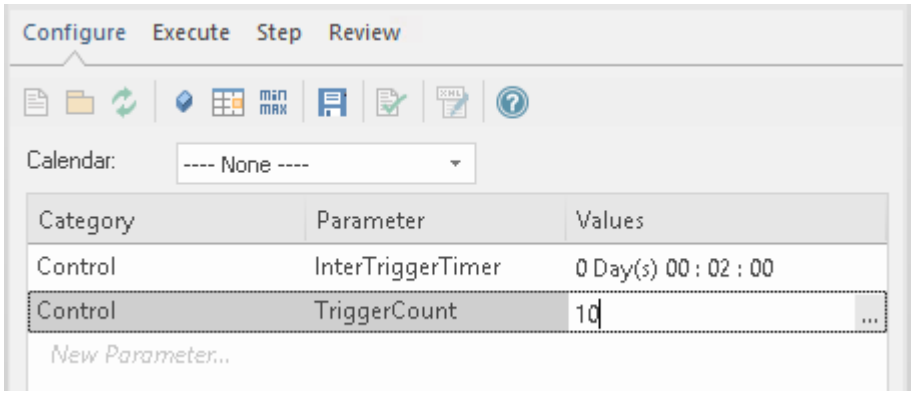
1. Sélectionnez l'élément sur le diagramme , puis cliquez sur le bouton *Nouveau paramètre* texte et sur la flèche déroulante dans la colonne « Catégorie », puis sélectionnez « Contrôle ».



2. Cliquez sur la flèche déroulante dans le champ « Paramètre », qui affichera les paramètres non attribués disponibles pour l'élément sélectionné.



3. Sélectionnez le paramètre approprié, puis cliquez sur le champ « Valeurs » ; vous pouvez soit saisir la valeur du paramètre dans le champ, soit utiliser le bouton [...] pour ouvrir la dialogue « Valeur du paramètre ».

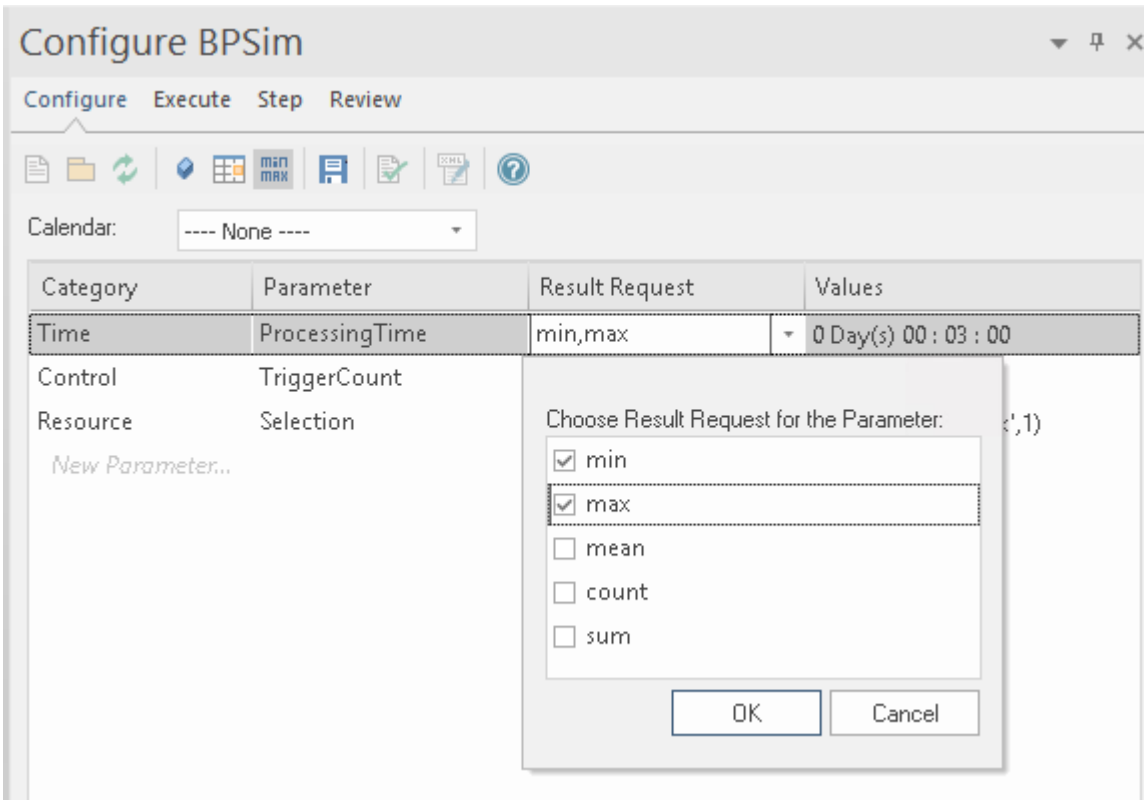


Note que la page vous permet de fournir uniquement les paramètres appropriés pour l'élément. Une fois que vous avez spécifié ces paramètres, les champs ne permettent plus de saisie ou de sélection.

Paramètres temporels

Pour commencer à définir les paramètres de temps pour l'élément approprié (comme une tâche BPMN) :

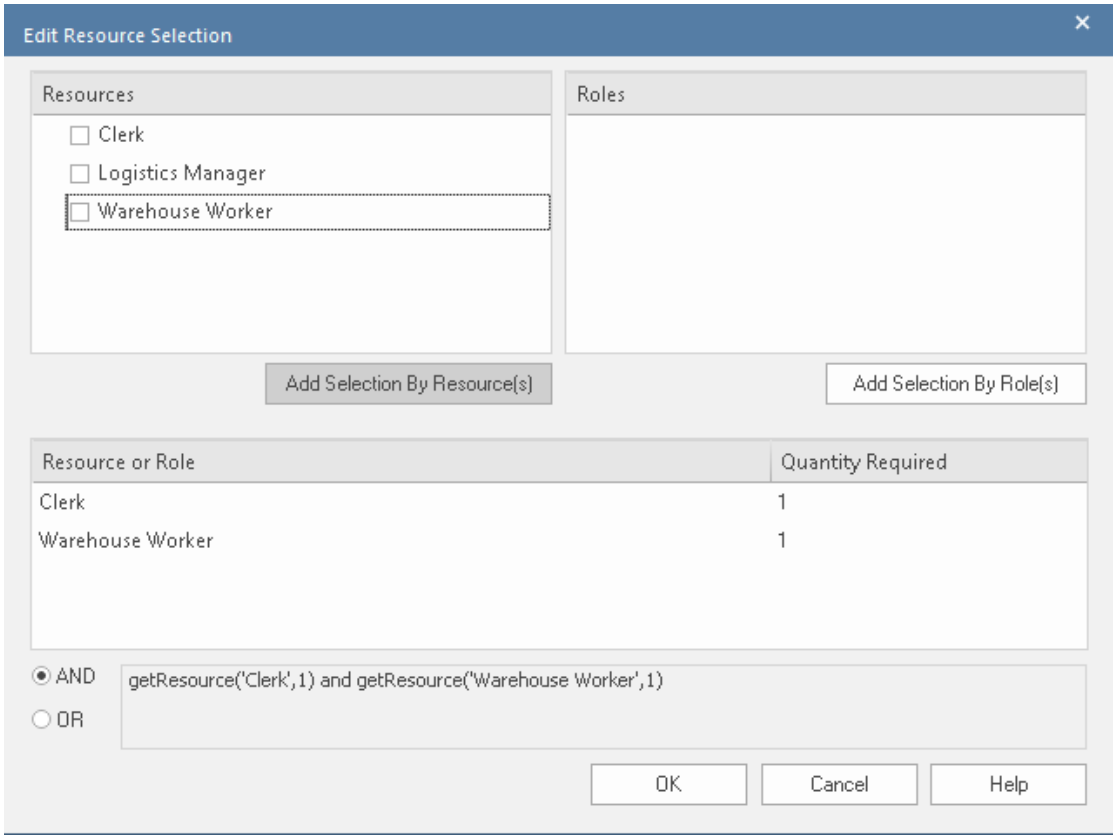
1. Sélectionnez l'élément sur le diagramme .
2. Cliquez sur le texte *Nouveau paramètre* et sur la flèche déroulante, puis sélectionnez « Heure » dans la liste.
3. Après avoir sélectionné « Heure », cliquez sur la flèche déroulante dans le champ « Paramètre » et sélectionnez l'un des paramètres disponibles pour l'élément.
4. Dans le champ « Valeurs », saisissez la valeur ou cliquez sur le bouton  pour ouvrir la dialogue « Valeur du paramètre ».
5. Vous pouvez basculer la colonne « Demande de résultat » en cliquant sur le bouton min/max de la barre d'outils pour personnaliser la sortie de simulation en exigeant certains résultats



Paramètres des ressources

Pour commencer à définir les paramètres de ressources pour l'élément approprié (comme une tâche BPMN) :

1. Sélectionnez l'élément sur le diagramme .
2. Cliquez sur le texte *Nouveau paramètre* et sur la flèche déroulante, puis sélectionnez « Ressource » dans la liste.
3. Dans le champ « Paramètre », cliquez sur la flèche déroulante et cliquez sur « Sélection » dans la liste.
4. Dans le champ « Valeurs », cliquez sur le bouton ... pour ouvrir la dialogue « Modifier la sélection de ressources ».

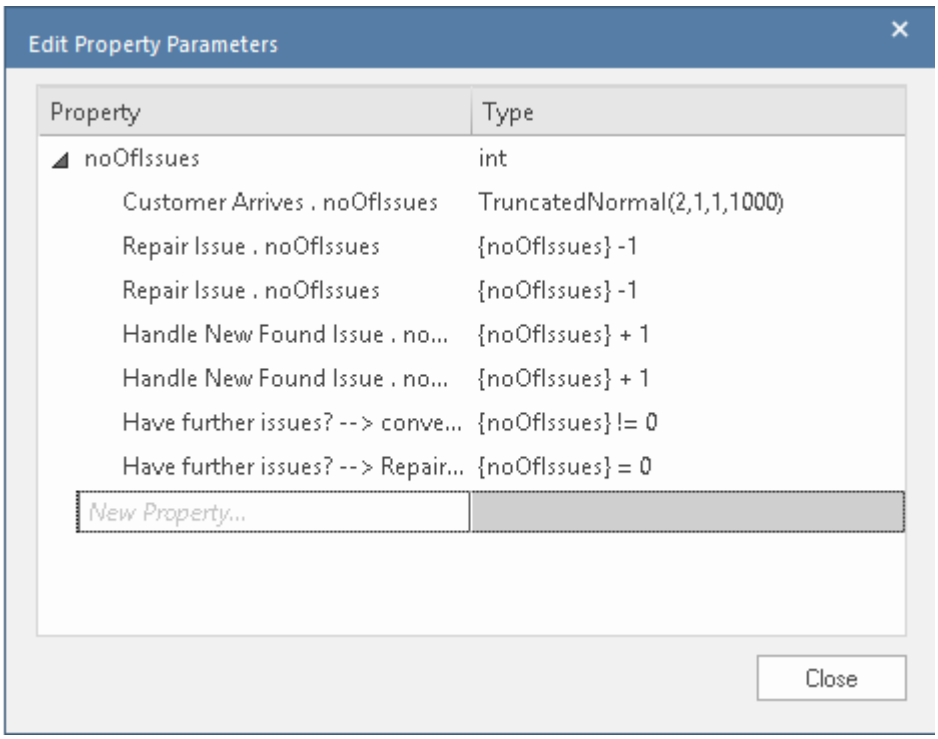


- Le panneau supérieur gauche répertorie les éléments de ressources définis ; cliquez sur les ressources à affecter et sur le bouton Ajouter une sélection par ressource(s) pour déplacer la sélection vers le panneau « Ressource ou rôle »
- Le panneau supérieur droit répertorie les rôles définis (le cas échéant) pour les éléments de ressource ; cliquez sur les rôles requis et sur le bouton Ajouter une sélection par rôle(s) pour déplacer la sélection vers le panneau « Ressource ou rôle »
- La colonne « Quantité requise » est définie par défaut sur 1 pour chaque ressource/rôle ; si une quantité plus importante est requise, remplacez cette valeur par le nombre approprié
- Cliquez sur le bouton radio approprié pour définir la relation logique sur AND ou OR pour la sélection
- L'expression finale pour la sélection des ressources est composée et affichée dans le champ de texte
- Cliquez sur le bouton OK pour revenir à la fenêtre Configurer BPSim, où l'expression est affichée dans le champ « Valeurs »

Paramètres de la propriété

Pour commencer à définir les paramètres de propriété, cliquez sur le bouton dans la barre d'outils. La dialogue

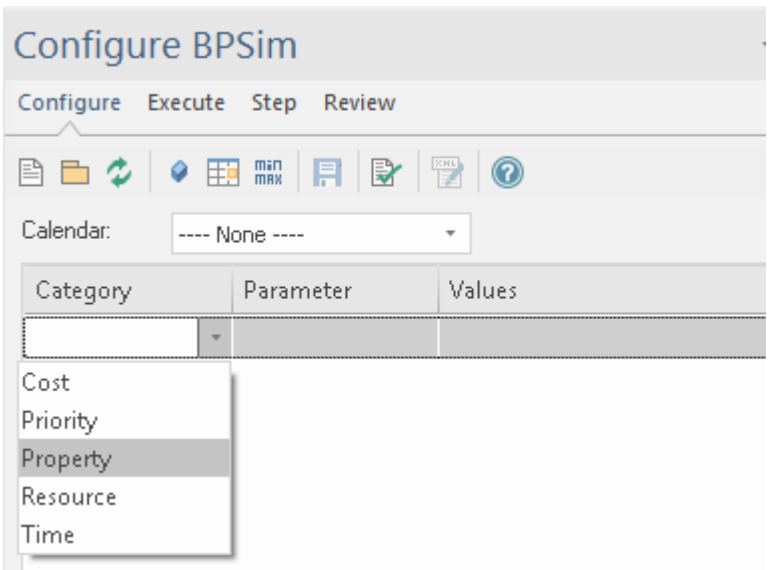
« Modifier les paramètres de propriété » s'affiche.



Les propriétés définies et leurs références sont répertoriées.

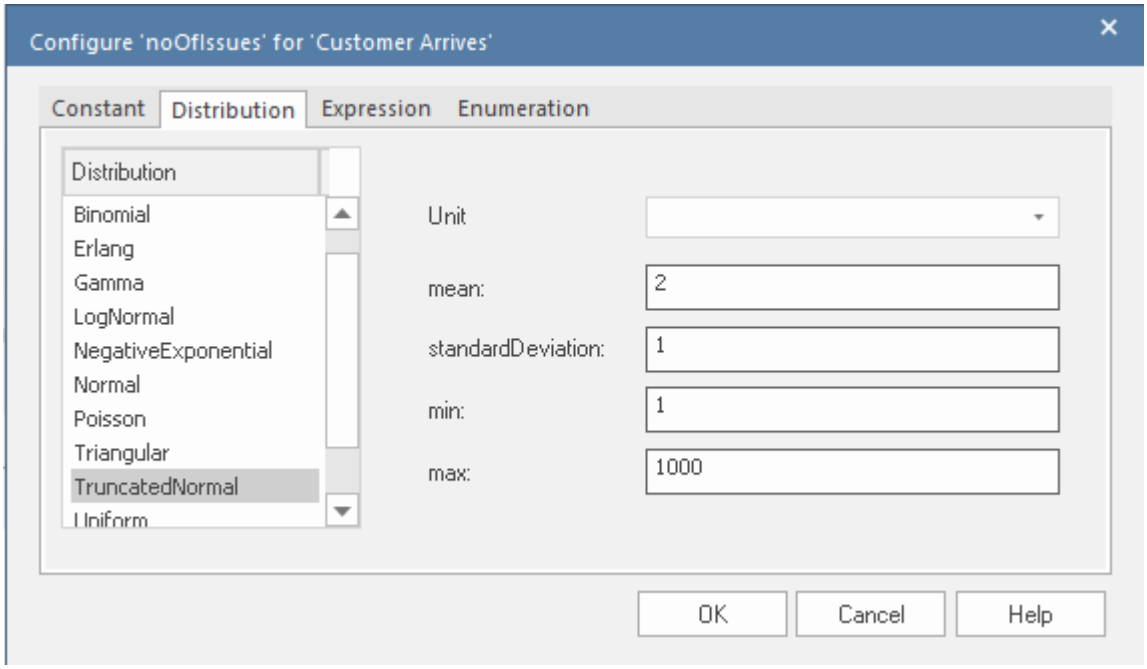
Vous pouvez ajouter une nouvelle propriété, supprimer une propriété sélectionnée (en utilisant l'option du menu contextuel), écraser le nom d'une propriété ou sélectionner un type différent pour une propriété.

Après avoir vérifié les propriétés définies, vous pouvez définir les paramètres de propriété sur les éléments BPMN.



Choisissez « Propriété » comme catégorie, puis cliquez sur la flèche déroulante dans le champ « Paramètre » et sélectionnez une propriété.

Cliquez sur le bouton ... dans la colonne « Valeurs » pour afficher la dialogue valeur du paramètre (nommée à partir de la propriété et de l'élément parent).



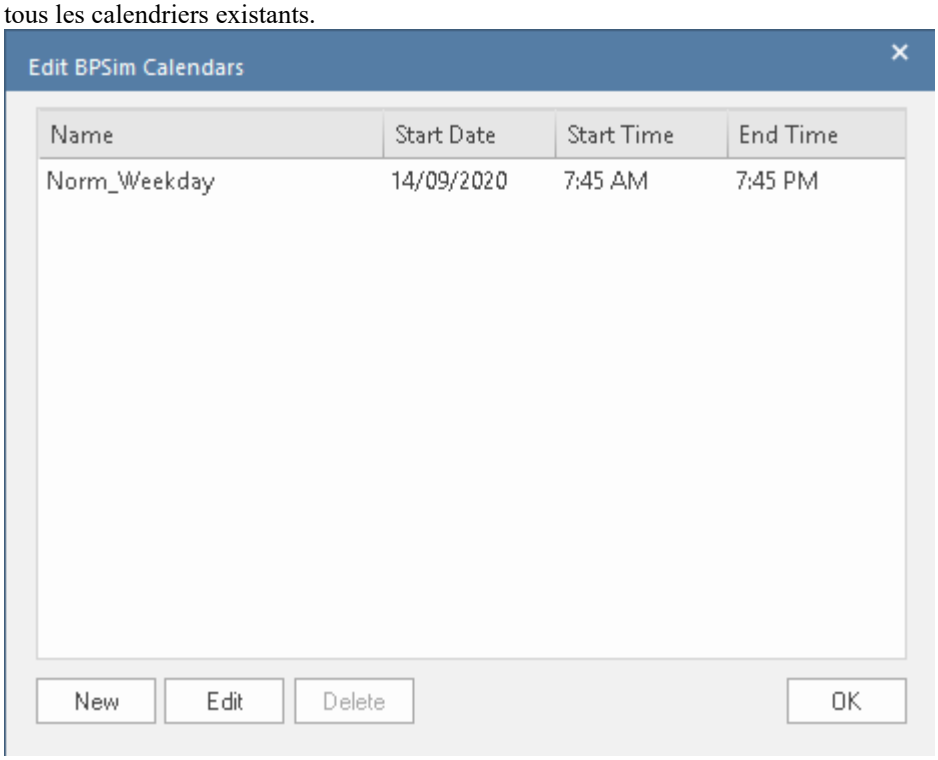
Accédez à l'onglet approprié pour sélectionner et définir le type de valeur et valeur réelle, puis cliquez sur le bouton OK . La valeur s'affiche dans le champ « Valeur ».

Calendriers

Les calendriers vous aident à définir un nombre quelconque de périodes de temps spéciales qui peuvent influencer le processus, telles que les jours ouvrables, les quarts de travail, les jours fériés ou les événements périodiques (par exemple, l'inventaire, l'inventaire ou l'audit).

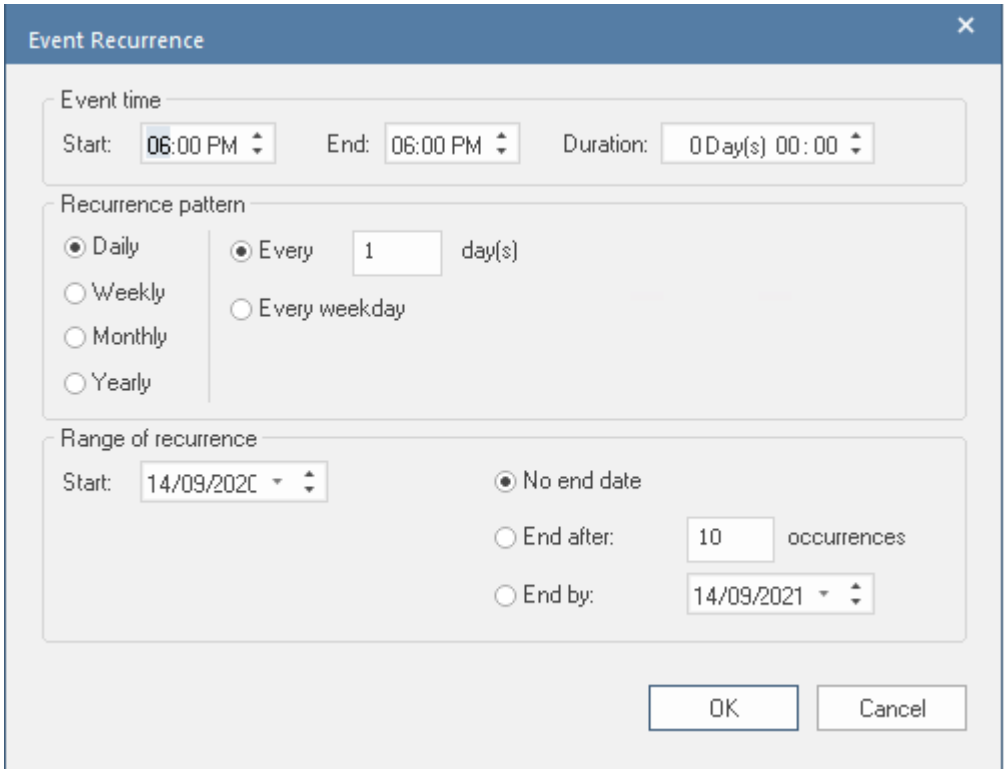
Pour commencer à définir les calendriers :

1. Cliquez sur le bouton dans la barre d'outils ; la dialogue « Modifier les calendriers BPSim » s'affiche, indiquant tous les calendriers existants.



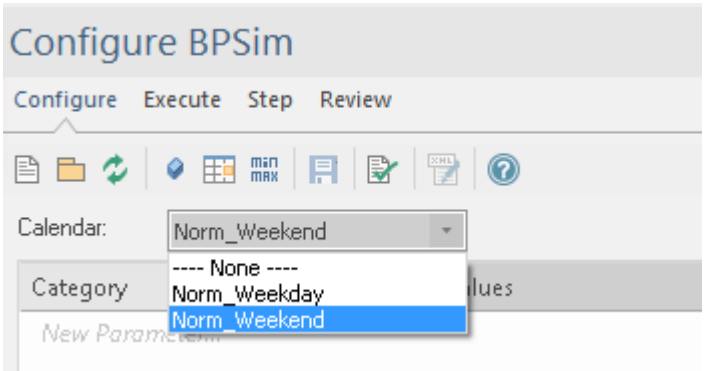
Vous pouvez ajouter un nouveau calendrier, ou modifier ou supprimer un calendrier sélectionné.

2. Pour ajouter une nouvelle période de calendrier, cliquez sur le bouton Nouveau pour afficher la dialogue « Récurrence de l'événement ».



3. Dans le panneau « Heure de l'événement », les champs « Démarrer » et « Fin » sont tous deux définis par défaut sur l'heure actuelle. Le champ « Démarrer » est l'ancre ; une modification du champ « Fin » ou du champ « Durée » met automatiquement à jour l'autre champ, en référence au champ « Démarrer ». Cliquez sur les segments d'heure et de minute de chaque champ (et, pour le champ « Durée », sur le segment « Jour(s) ») séparément, et utilisez les flèches de rotation pour définir l'heure de début et l'heure de fin ou la durée de la période.
4. Dans le panneau « motif de récurrence », sélectionnez le bouton radio correspondant à l'intervalle auquel la période calendaire se répète. Chaque option affiche un ensemble de champs approprié à droite du panneau pour affiner cet intervalle à chaque jour/semaine/mois ou tous les deux/trois/quatre jours/semaines/mois, un jour particulier de la semaine, un jour ou une date du mois, ou un jour ou une date de l'année. Sélectionnez les cases à cocher ou les valeurs dans les listes déroulantes selon le cas.
5. Dans le panneau « Plage de récurrence », sélectionnez la date à laquelle la période calendaire prend effet et sélectionnez le bouton radio approprié pour définir le moment où la période cesse de s'appliquer : jamais, après un nombre défini d'occurrences ou à une date spécifique. Vous pouvez sélectionner une date de fin soit à partir d'un calendrier déroulant, soit à l'aide des flèches de rotation sur chaque segment de la date.
6. Cliquez sur le bouton OK pour définir la période du calendrier.

Lorsque vous définissez des périodes de calendrier, elles sont répertoriées par ordre de date et/ou d'heure de début, la plus ancienne en premier.



Avec des calendriers définis, vous pouvez configurer des paramètres sur un calendrier sélectionné.

Validation

Après avoir configuré les paramètres BPSim pour certains éléments BPMN, cliquez sur le bouton  pour exécuter une validation de la simulation. Toutes les erreurs/avertissements BPMN ou BPSim seront affichés dans la fenêtre Sortie système. Corrigez les problèmes en fonction des messages.

Après avoir effectué cette opération, passez à la rubrique d'aide suivante : *Page d'exécution de BPSim* .

BPSim – Page d’Exécuter

Après avoir défini la configuration, vous pouvez choisir d'effectuer une simulation standard ou une simulation personnalisée. L'exécution générera un rapport de résultats et une liste d'enregistrements utilisés pour rejouer (parcourir) la simulation.

Accéder

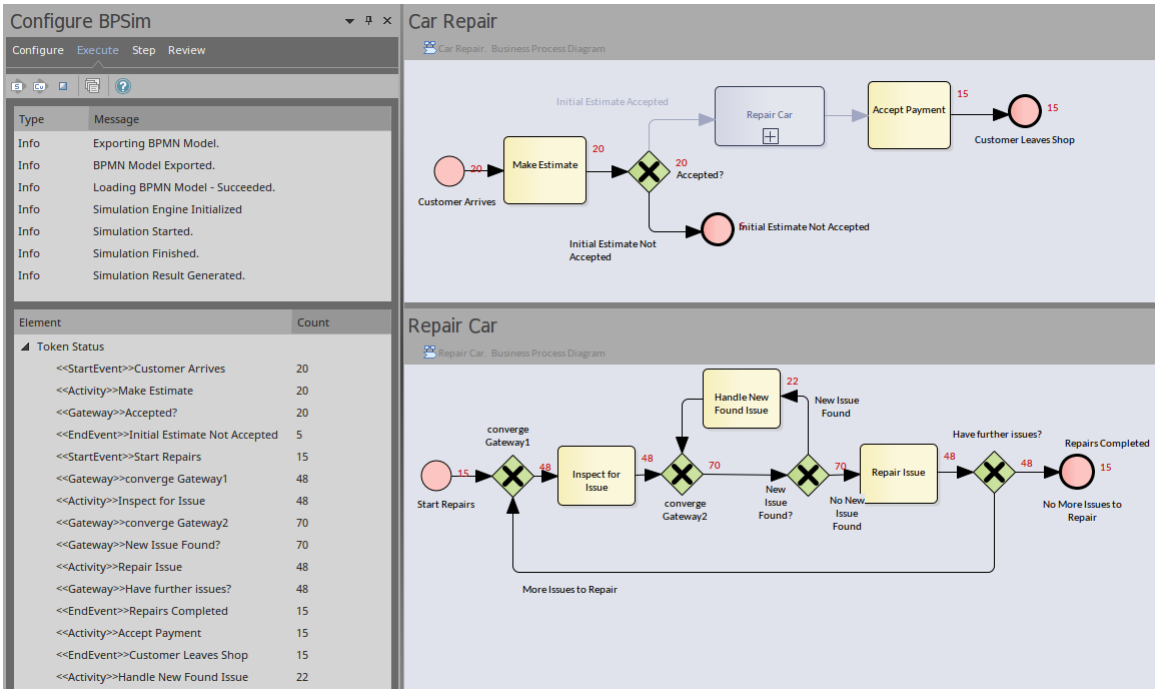
Ruban	Simuler > Analyse de Processus > BPSIM > Ouvrir BPSIM Manager > Page Exécuter

Options de la barre d'outils

Option	Description
	Cliquez sur ce bouton pour exécuter le modèle BPMN avec la configuration BPSim et générer un rapport standard.
	Cliquez sur ce bouton pour exécuter le modèle BPMN avec la configuration BPSim et générer un rapport personnalisé basé sur les paramètres « Demande de résultat » définis sur la page « Configurer ».
	Cliquez sur ce bouton pour arrêter l'exécution et quitter la simulation.
	Cliquez sur ce bouton pour ouvrir le rapport généré dans la page ' Révision '.

Exécution

Lorsque vous cliquez sur le bouton Exécuter Simulation ou sur le bouton Exécuter Simulation personnalisée, le modèle BPMN avec la configuration BPSim sera exporté et chargé dans le moteur d'exécution.



Pendant la simulation :

- La liste d'état du jeton clignotera avec les valeurs d'exécution
- Le diagramme clignotera avec le nombre de jetons d'exécution

Cependant, la simulation peut s'exécuter trop rapidement pour que cela soit visible. Vous pouvez voir ces changements si vous utilisez la page « Étape » pour exécuter la simulation étape par étape.

Dans cet exemple, les éléments BPMN sous le processus « Réparation de voiture » et le sous-processus « Réparation de voiture » sont déclenchés lorsque de nouveaux clients arrivent à intervalles réguliers.

BPSim - Page d'Étape

Une fois l'exécution réussie, le système génère un rapport d'exécution qui vous indique l'état du processus en général, comme (pour l'exemple de réparation automobile) le temps moyen d'une tâche, le temps d'attente total des clients et le nombre de problèmes réparés.

De plus, vous pouvez inspecter le processus sous différents angles. Par exemple :

- D'après l'horodatage, quel était l'état de ce processus à 9h30 ?
- À partir du jeton, qu'a fait le 3ème client dans la boutique ?
- Depuis la propriété - comment le nombre de problèmes diminue et augmente pour la 2ème voiture ?
- À partir de plusieurs threads, puis-je voir les clients entrer et simuler automatiquement sur le diagramme ?
- D'après les ressources : quand un agent support est-il occupé ou inactif ? Pourquoi un client attend-il 40 minutes ?

Tous ces types de questions peuvent trouver une réponse sur la page « Étape ».

Accéder

Ruban	Simuler > Analyse de Processus > BPSIM > Ouvrir BPSIM Manager > Page Étape

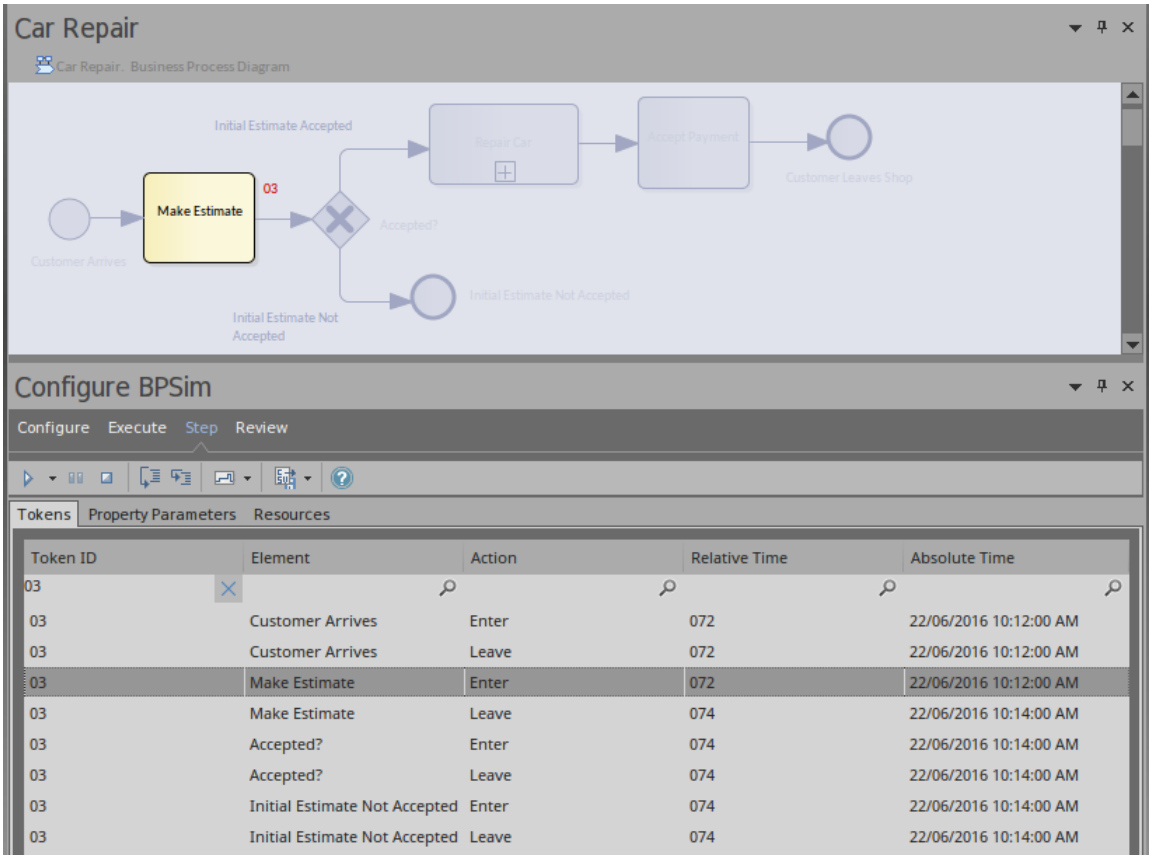
Options de la barre d'outils

Option	Description
	Cliquez sur ce bouton pour simuler automatiquement le processus en fonction du résultat de l'exécution. Cliquez sur la flèche déroulante et sur l'option de menu « Définir la vitesse de relecture » et ajustez la vitesse de simulation en tant que multiple de la vitesse normale. Par exemple, si vous saisissez « 60 », la simulation sera 60 fois plus rapide que l'activité réelle ; 1 minute dans la vie réelle sera simulée en 1 seconde.
	Cliquez sur ce bouton pour mettre en pause la simulation de relecture automatique.
	Cliquez sur ce bouton pour arrêter la simulation.
	Cliquez sur ce bouton pour passer à l'horodatage suivant. Chaque « Pas à pas » peut contenir plusieurs « étapes ».
	Cliquez sur ce bouton pour jouer une seule étape. Cela représente un seul mouvement d'un jeton dans le processus.
	Cliquez sur ce bouton pour générer un Diagramme de temps pour la simulation. Vous pouvez choisir dans le menu, soit « Générer une seule chronologie pour chaque jeton », soit « Générer plusieurs chronologies pour chaque jeton ». Voir *Générer Diagramme de temps* plus loin dans cette rubrique.
	Cliquez sur ce bouton pour exporter les enregistrements filtrés de cette page d'étape

	vers un fichier CSV. Vous pouvez choisir à partir de quel onglet (« Jetons », « Paramètres de propriété » ou « Ressources ») les données sont exportées.

Onglet Jetons

Après avoir exécuté l'exécution, cette page sera remplie avec des informations sur les jetons pendant la simulation ; la séquence des entrées est dans l'ordre du moment de déclenchement.

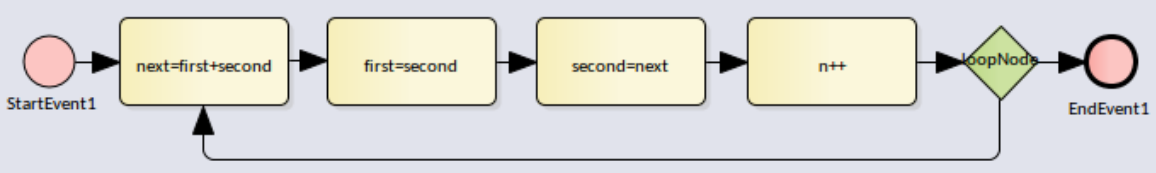


- En utilisant la Barre de Filtre dans la bande d'en-tête (cliquez-droit sur l'en-tête de colonne et sélectionnez « Basculer Barre de Filtre »), vous pouvez filtrer les résultats affichés ; par exemple, en tapant 03 dans la colonne « ID de jeton », seuls les enregistrements du jeton 03 seront affichés.
- Si vous cliquez une fois sur le bouton Entrer, un enregistrement de la liste sera lu
- Si vous double-cliquez sur un enregistrement, la simulation « passera à » cet enregistrement depuis le début
- Si des paramètres de temps sont définis sur les éléments, cliquer sur le bouton Passer au suivant exécuter le dernier enregistrement du prochain événement temporel
- Lorsqu'un enregistrement de la liste est joué, la simulation instantané s'affiche sur le diagramme

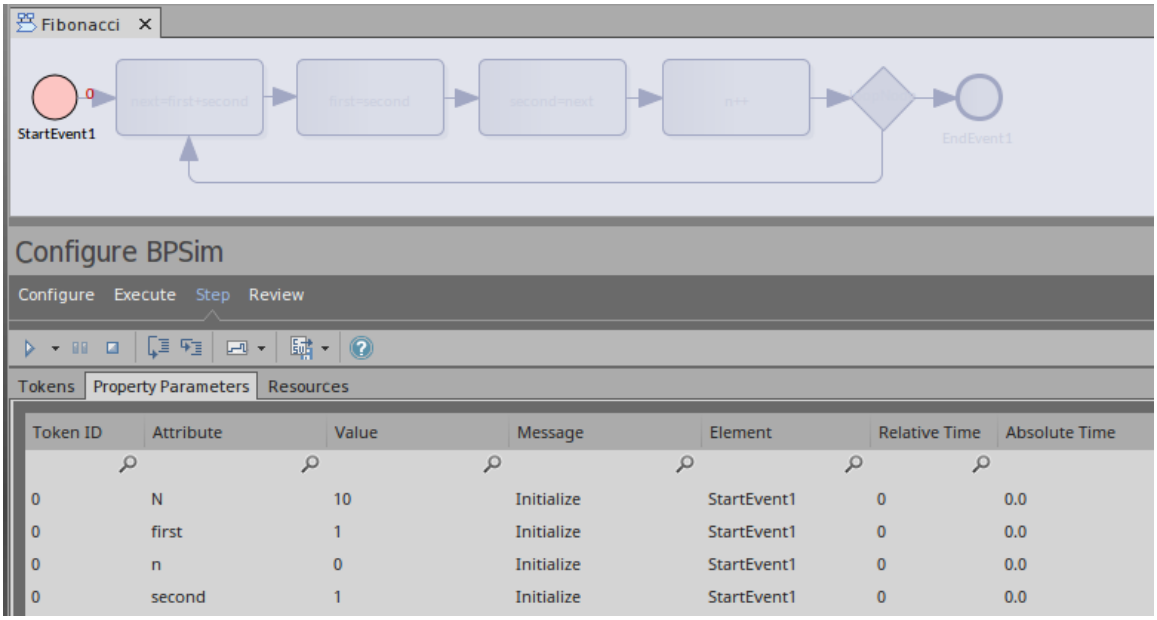
Onglet Paramètres de propriété

Pendant que les enregistrements de l'onglet « Jetons » sont lus, l'onglet « Paramètres de propriété » affiche la valeur d'exécution des propriétés à l'horodatage.

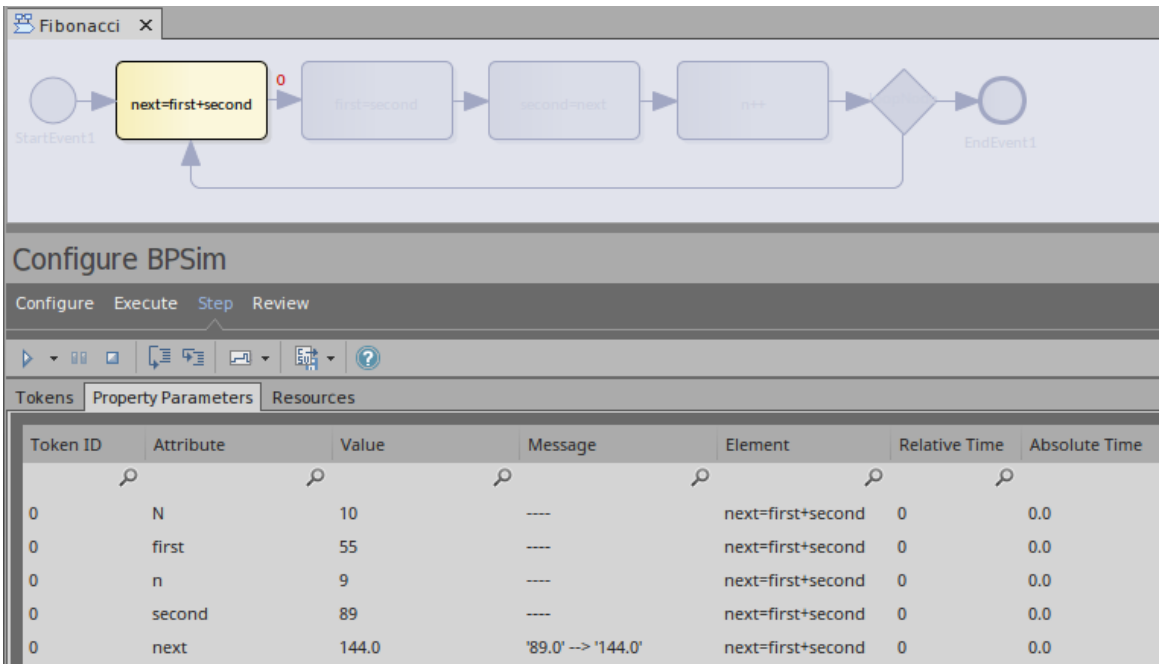
Par exemple, un processus BPMN pour calculer les nombres de Fibonacci pourrait être modélisé de cette manière :



Après avoir défini les paramètres de propriété, configuré les paramètres BPSim pour chaque élément et exécuté le modèle, nous sommes prêts pour la simulation des étapes :



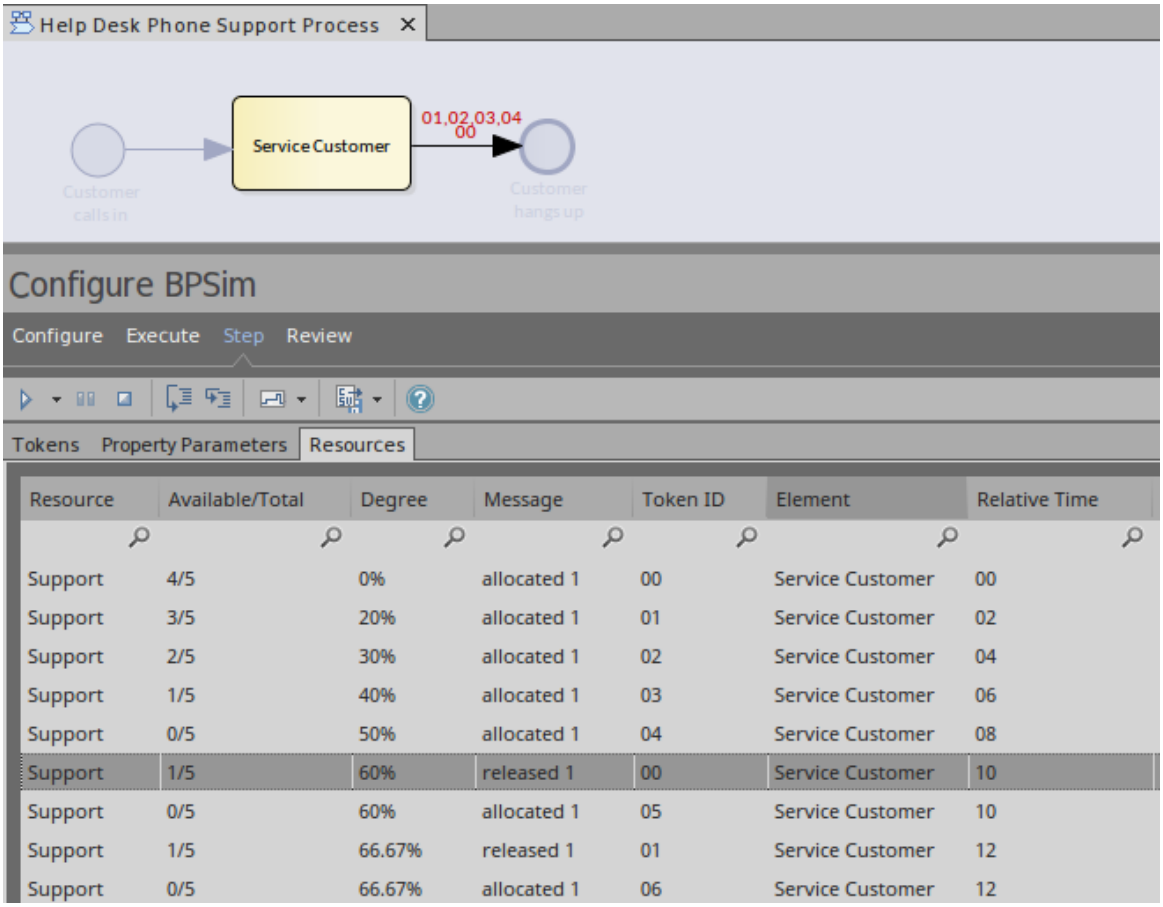
La colonne « Message » indique que les propriétés « N », « first », « n » et « second » sont initialisées.



Si vous continuez à cliquer sur le bouton Entrer, les valeurs des propriétés de la liste changeront. L'illustration montre qu'en entrant dans la tâche 'next = first + second', la valeur de la propriété 'next' passe de 89 à 144.

Onglet Ressources

Pendant que les enregistrements de l'onglet « Jetons » sont lus, l'onglet « Ressources » affiche la ressource d'exécution disponible, la quantité de cette ressource disponible et les événements d'allocation ou de libération à l'horodatage.

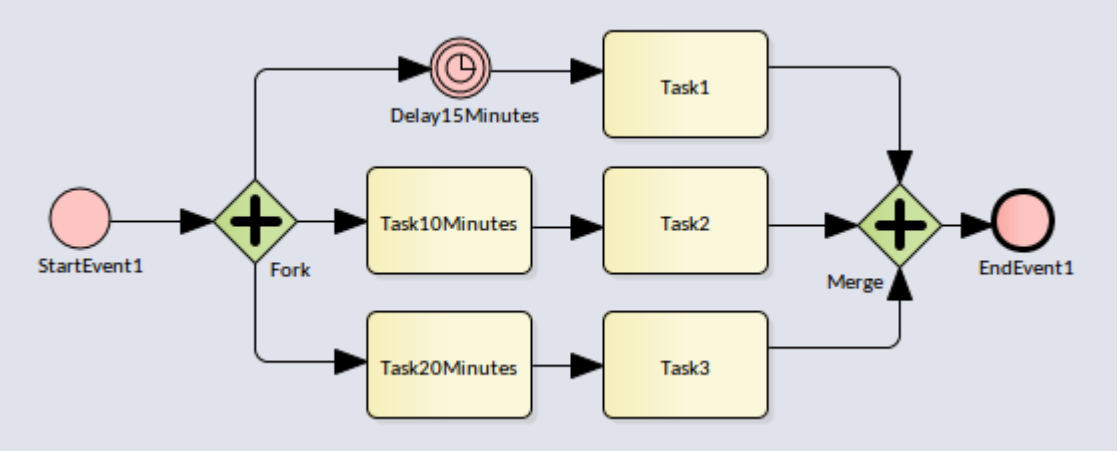


Générer Diagramme de temps

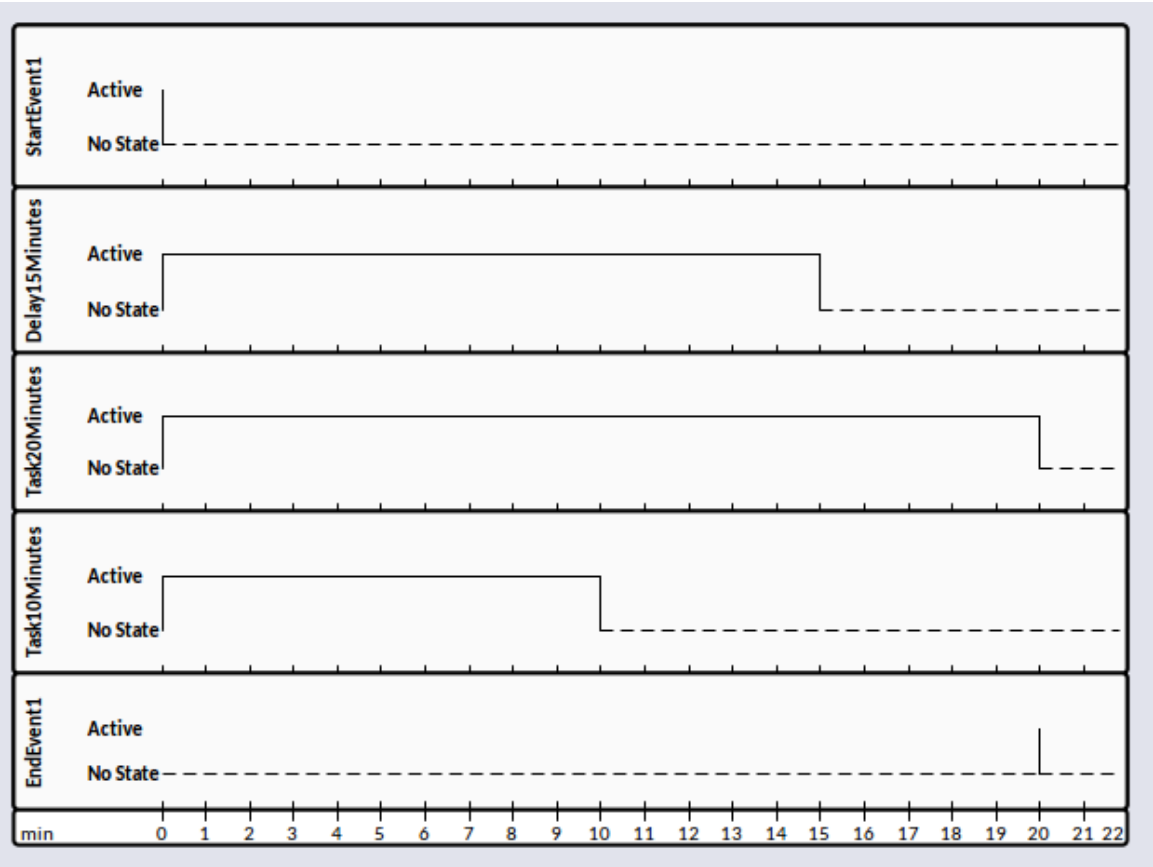
Lorsque les paramètres de temps sont configurés sur les éléments BPMN, Enterprise Architect peut générer un diagramme de temps pour le processus de simulation.

- Générer une chronologie unique pour chaque jeton - utilisez cette option pour un processus « à thread unique » ; c'est-à-dire, aucun sous-processus Passerelle parallèle ou d'événement
- Générer plusieurs chronologies pour chaque jeton - utilisez cette option dans les cas où l'option « Générer une seule chronologie pour chaque jeton » ne s'applique pas

Par exemple:



Exécutez ce modèle et cliquez sur « Générer plusieurs chronologies pour chaque jeton », le diagramme de temps généré ressemble à ceci :



BPSim - Page de Révision

Cette page révision contient trois onglets :

- Résumé de la configuration
- Rapport de résultats standard
- Rapport de résultats personnalisé

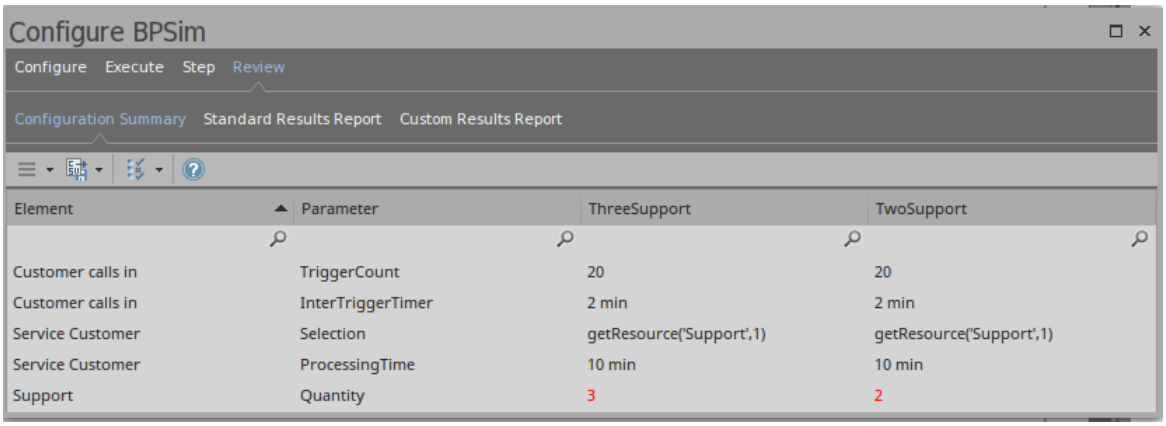
Ces onglets fonctionnent de manière similaire : ajoutez un artefact à révision ou plusieurs artefacts à comparer. Cela vous permet de réaliser facilement des analyses de simulation.

Accéder

Ruban	Simuler > Analyse de Processus > BPSIM > Ouvrir BPSIM Manager > Révision

Analyse de What-If

Dans l'exemple Support du service d'assistance technique, nous pouvons comparer deux artefacts et leurs résultats correspondants.



Dans cette illustration, nous avons cliqué sur l'icône dans la barre d'outils et sélectionné l'option « Afficher uniquement les éléments différents » pour voir quelles différences les valeurs de paramètres modifiées ont provoquées.

Element	Parameter	ThreeSupport- Result	TwoSupport- Result
Help Desk Phone Support Process	Maximum Time	34	64
Help Desk Phone Support Process	Standard Deviation Time	7.72	17.23
Help Desk Phone Support Process	Average Time	21.4	37
Service Customer	Average Number Of Tokens Waiting For Resource	3.17	5.29
Service Customer	Average Time In Task	21.4	37
Service Customer	Average Time Waiting For Resource	11.4	27
Service Customer	Maximum Number Of Tokens Waiting For Resource	8	12
Service Customer	Maximum Time In Task	34	64
Service Customer	Maximum Time Waiting For Resource	24	54
Service Customer	Total Time In Task	428	740
Service Customer	Total Time Waiting For Resource	228	540
Support	Degree Of Utilisation	92.59%	98.04%
Support	Number Started Immediately	3	2
Support	Total Time Available	216	204
Support	Total Time Idle	16	4
Support	Average Number Available	0.22	0.04

Nous constatons que lorsque le nombre de personnel support diminue de 3 à 2, le temps moyen d'attente pour les ressources augmente de 11,4 minutes à 27 minutes.

Utilisant le Dialogue de Valeur de Paramètre

La dialogue « Valeur de paramètre » vous aide à définir des valeurs pour une large gamme de paramètres dans toute la configuration BPSim. Elle supporte la définition de valeurs fixes simples jusqu'aux distributions et expressions qui génèrent une valeur dérivée. Tous les types de valeur ou de dérivation ne conviennent pas à tous les types de paramètres.

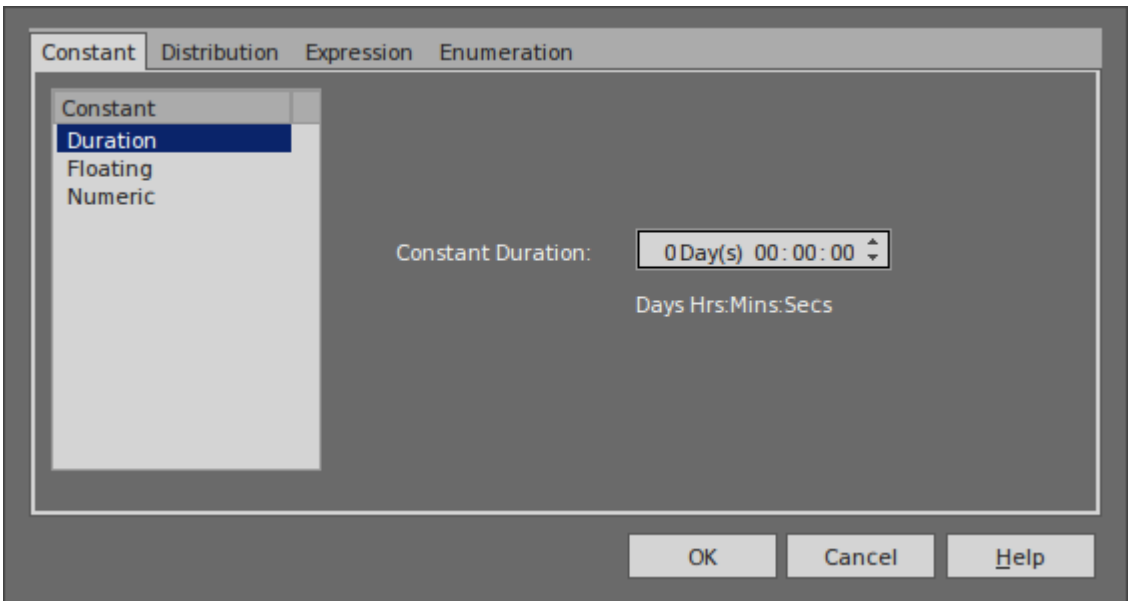
Le nom de dialogue est tiré du nom object et du nom du paramètre en cours de définition ; par exemple, Configurer « Traitement » pour « Activité1 ».

Accéder

Avec un artefact BPSim chargé dans la fenêtre Configurer BPSim, sélectionnez un élément BPMN dans le diagramme ou dans la fenêtre Navigateur , puis cliquez sur  dans le champ « Valeurs ». (Si le paramètre n'est pas déjà créé, choisissez Catégorie et Paramètre dans la liste pour en créer un nouveau.)

Onglet constant

Utilisez cet onglet pour définir une valeur spécifique pour le paramètre - un nombre, string de texte ou une heure, par exemple.



Dans le panneau « Constante », sélectionnez le type de constante :

- Flottant
- Numérique
- String
- DateHeure
- Booléen, ou
- Durée

Les champs appropriés s'affichent à droite du panneau ; saisissez la valeur et, si nécessaire, l'unité dans laquelle la valeur est exprimée (par exemple, une unité de temps ou de devise). Pour certains types de paramètres, une liste déroulante est disponible dans laquelle vous pouvez sélectionner une valeur .

Onglet Distribution

Dans cet onglet, vous pouvez appliquer une méthode d'échantillonnage statistique pour obtenir la valeur du paramètre ; pour chaque type de distribution disponible, les champs appropriés s'affichent pour vous permettre de saisir les paramètres de la distribution. Toutes les distributions nécessitent que vous identifiiez l'unité d'expression.

Les paramètres de distribution ne sont pas nécessaires pour le processus métier que vous développez, mais (si vous dérivez des valeurs d'une distribution) sont requis pour la simulation.

Vous pouvez choisir parmi ces types de distribution :

- **Bêta** - une distribution de probabilité continue fournissant des valeurs « réelles » dans une plage courte, généralement de 0 à 1
- **Weibull** - une distribution de probabilité continue fournissant des valeurs « réelles », couramment utilisée pour l'analyse de la durée de vie object
- **Gamma** - une distribution de probabilité continue fournissant des valeurs « réelles », utile pour modélisation de variables aléatoires distribuées de manière exponentielle
- **Binomiale** - une distribution « integer », fournissant des valeurs basées sur le nombre d'essais et la probabilité d'un certain résultat
- **Erlang** - fournit des valeurs « réelles » basées sur la valeur *K* d'Erlang et la moyenne de la distribution
- **Normal** - fournit des valeurs « réelles » basées sur la moyenne et l'écart type de la distribution
- **LogNormal** - une distribution de probabilité continue de variables aléatoires « réelles » dont le logarithme est normalement distribué
- **Poisson** - une distribution de probabilité discrète (« integer ») qui exprime la probabilité qu'un nombre donné d'événements se produisent indépendamment dans un intervalle de temps ou d'espace fixe (volume, distance ou surface)
- **NegativeExponential** - fournit des valeurs « réelles » basées sur la moyenne de la distribution
- **Triangulaire** - fournit des valeurs « réelles » basées sur le mode de distribution et les valeurs minimales et maximales d'une plage
- **TruncatedNormal** - fournit des valeurs « réelles » basées sur la moyenne et l'écart type des points compris entre les valeurs minimales et maximales d'une plage
- **Uniforme** - fournit des valeurs « réelles » entre les valeurs minimales et maximales dans une plage

Onglet Expression

Dans cet onglet, vous saisissez une expression XPATH 1.0 pour combiner des valeurs explicites, des opérateurs et des fonctions à traiter lors de l'exécution afin de fournir une valeur . Chaque paramètre de propriété d'une expression doit être placé entre accolades - {xxx}.

Exemple 1 : Pour représenter c = a + b + 10, nous attribuons cette expression à une propriété « c » :

{a} + {b} + 10

où « a » et « b » sont des propriétés définies dans le modèle BPSim.

Exemple 2 : Afin de représenter $c = t - p * (a - b)^2$, nous attribuons cette expression à une propriété 'c' :

{t} - {p} * Math. pow ({a} - {b}, 2.0)

Note : lors de la simulation d'un modèle avec cette expression, veuillez sélectionner « Java » comme langage afin d'utiliser la fonction intégrée Java Math. pow ().

Onglet Énumération

Dans l'onglet « Énumération », vous pouvez définir une énumération pour fournir une collection de valeurs constantes. Vous pouvez obtenir ces valeurs à partir de données historiques réelles ou à partir de l'analyse et de la simulation d'un modèle. Chaque fois que le paramètre est évalué, la valeur d'énumération suivante est renvoyée.

Lorsque vous définissez chaque valeur de numération, cliquez sur le bouton Enregistrer pour l'ajouter à la liste des valeurs possibles, puis cliquez sur le bouton Nouveau pour effacer les champs de données et être prêt à saisir une autre valeur . Pour certains types de valeur d'énumération, il peut vous être demandé de définir l'unité dans laquelle la valeur est exprimée. Les types d'énumération que vous pouvez définir incluent :

- String
- Flottant
- Numérique
- Durée
- DateHeure
- Booléen

Utilisation du Moteur d'Exécution BPSim

Le Moteur d'Exécution BPSim est un Add-In intégré aux éditions Unified et Ultimate d' Enterprise Architect , pour exécuter les simulations que vous avez définies à l' facilité du Processus Métier Simulation (BPSim). Le Moteur est un prérequis pour accéder et utiliser les facilités de BPSim.

Accéder

Ouvrez la fenêtre Configurer BPSim à l'aide de l'une des méthodes de ce tableau , cliquez sur le bouton  et recherchez un artefact Simulation Processus Métier .

Ruban	Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager (ou Simuler > Analyse de Processus > BPSim > Rechercher les artefacts de configuration BPSim)
Menu Contexte	Cliquez-droit sur un élément d'artefact Processus Métier Simulation | Configurer BPSim
Autre	Cliquez-droit sur un élément d'artefact Processus Métier Simulation | Simulez BPMN avec BPSim

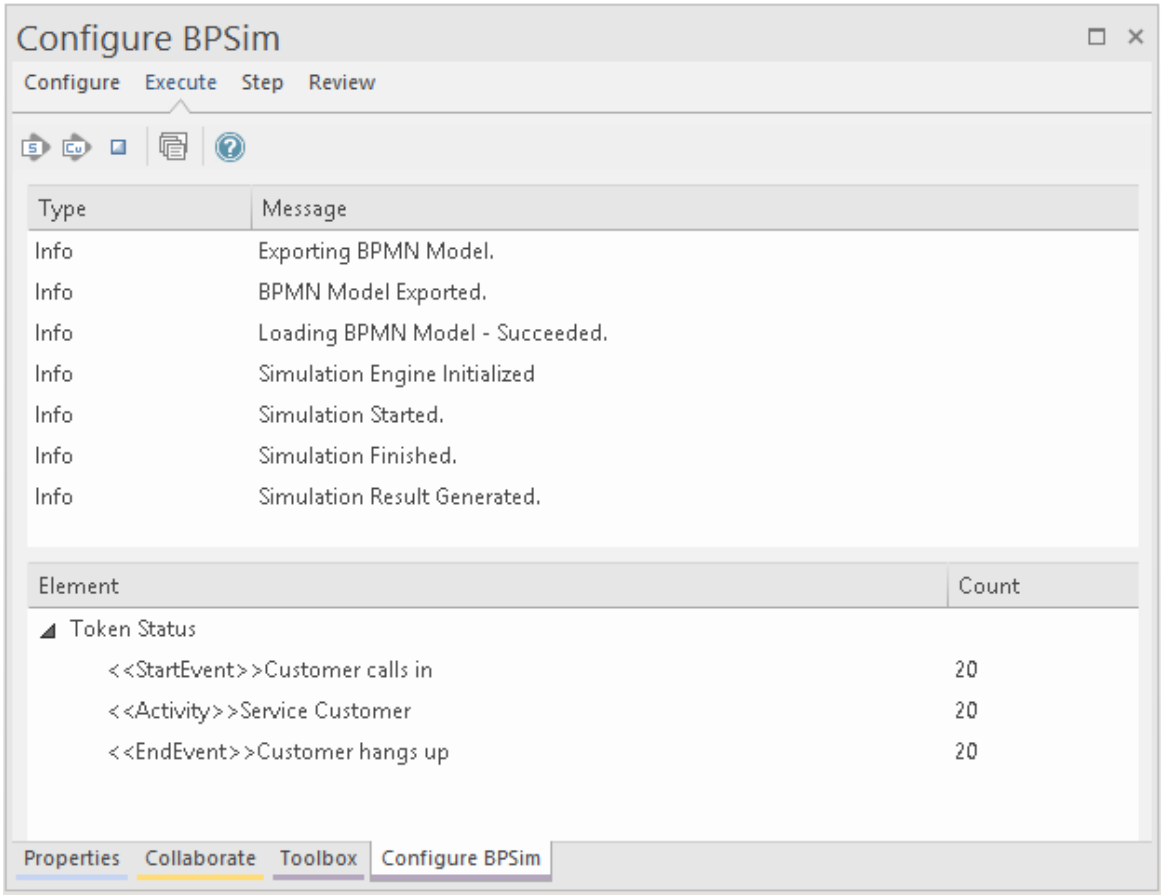
Exécuter et contrôler une Simulation

Cliquez sur l'onglet « Exécuter » et sur :

-  pour démarrer une Simulation standard ou
-  pour démarrer une Simulation personnalisée

Ces options déclencheur le même traitement, sauf que tandis qu'une Simulation standard génère un rapport sur tous les paramètres intégrés définis dans la simulation, une Simulation personnalisée extrait les résultats uniquement pour les paramètres que vous avez spécifiquement marqués à l'aide des colonnes « Demande de résultat » dans la configuration.

La simulation s'exécute, affichant les messages de traitement dans la partie supérieure de le dialogue , ainsi que les éléments et paramètres traités avec les valeurs d'exécution utilisées à partir de la configuration.



Pendant que la simulation est en cours, vous pouvez cliquer sur l'icône pour annuler la simulation.

Les résultats de la simulation sont écrits dans un élément Artefact ajouté au Paquetage parent Processus Métier . Une simulation Standard écrit dans un Artefact stéréotypé <<BPSimReport>>, tandis qu'une simulation Personnalisée écrit dans un Artefact stéréotypé <<BPSimCustomReport>>.

Suivre les valeurs des propriétés

Outre les paramètres intégrés, vous pouvez définir vos propres paramètres de propriété (attributs) spécifiques au processus dans la configuration. Une fois la simulation terminée et si vous avez défini des paramètres de propriété, le bouton Attributes est activé. Lorsque vous cliquez sur ce bouton, la dialogue « BPSim PropertyParameter Values » s'affiche, grâce à laquelle vous pouvez suivre la manière dont les valeurs d'exécution de tous les paramètres de propriété augmentent ou changent au cours du processus métier.

Révision d'une Simulation

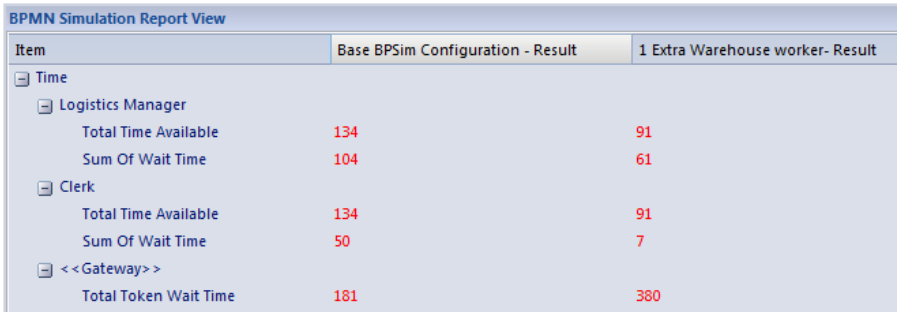
Une fois le traitement de la simulation terminé, cliquez sur le bouton Ouvrir le résultat. L'onglet « BPMN Simulation Rapport Vue » s'ouvre dans la zone de travail principale, affichant les résultats des paramètres intégrés dans la simulation actuelle (mais pas pour les paramètres de propriété définis par l'utilisateur). Si vous avez déjà exécuter une simulation d'une autre configuration basée sur le même processus métier, celle-ci s'affiche également dans le rapport sous la forme d'une colonne supplémentaire. Sinon, vous pouvez cliquer sur l'élément Artefact du rapport et le faire glisser sur l'onglet du rapport, pour comparer les valeurs d'exécution des paramètres intégrés sous deux (ou plusieurs) configurations.

Pour faciliter la visualisation des données du rapport, vous pouvez faire glisser l'onglet « BPMN Simulation Rapport Vue » hors de la vue principale afin qu'il devienne une fenêtre flottante et agrandir la fenêtre à une taille appropriée.

Cliquez sur les cases d'extension en regard des paramètres que vous souhaitez vérifier. Vous pouvez également exposer et filtrer les informations en utilisant les options du menu contextuel cliquez-droit .

Vous pouvez représenter des différences spécifiques entre les résultats de simulations distinctes sous forme de graphiques. Les artefacts de résultat de simulation (*nom* <<BPSimReport>> - éléments de résultat) doivent exister avant de pouvoir configurer les artefacts de graphique. Il existe un gabarit d'artefact de graphique pour les simulations standard et un autre pour les simulations personnalisées.

Options Rapport Simulation BPMN

Option	Description
Réduire tout	Sélectionnez cette option pour réduire la hiérarchie des paramètres aux seuls noms des onglets parents.
Développer tout	Sélectionnez cette option pour étendre la hiérarchie des paramètres jusqu'au type valeur le plus bas.
Afficher uniquement les éléments différents	(Lorsque vous avez deux simulations ou plus affichées.) Sélectionnez cette option pour limiter l'affichage aux paramètres dont les valeurs diffèrent entre les simulations. Cliquez à nouveau sur l'option pour la désélectionner.
Mettre en évidence différents Items	(Lorsque vous avez deux simulations ou plus affichées et que certaines de leurs valeurs de paramètres sont différentes.) Affiche les valeurs de paramètres différentes en rouge. Cette option est désactivée si vous sélectionnez l'option « Afficher uniquement Items différents ».  BPMN Simulation Report View Item | Base BPSim Configuration - Result | 1 Extra Warehouse worker- Result Time Logistics Manager Total Time Available 134 91 Sum Of Wait Time 104 61 Clerk Total Time Available 134 91 Sum Of Wait Time 50 7 <<Gateway>> Total Token Wait Time 181 380
Afficher uniquement Items non vides	Sélectionnez cette option pour filtrer l'affichage afin d'afficher uniquement les paramètres qui ont une valeur spécifique autre que 0.
Supprimer Modèle	(Lorsque vous avez sélectionné un résultat spécifique, qui identifie la simulation dans le rapport.) Sélectionnez cette option pour supprimer la colonne de simulation du rapport.

BPSim Moteur d'Exécution - Langage Simulation

Le Moteur d'Exécution BPSim supporte la simulation sur XPath 1.0 ou Java, où le langage approprié est défini comme langage d'expression dans la configuration de simulation. Il prend également supporte l'utilisation de données d'instance de processus dans les paramètres de propriété BPSim, où la valeur réelle n'est déterminée que pendant l'exécution.

Opérateurs XPath 1.0

Ces opérateurs peuvent être utilisés dans les paramètres d'expression BPSim.

Opérateur	Description
|	L'opérateur de l'Union, utilisé pour l'acquisition de ressources. Exemple : getResource('w1',1) | getResource('w2',1)
+	Ajout. Exemple : 4 + 6
-	Soustraction. Exemple : 6 - 4
*	Multiplication. Exemple : 6 * 4
div	Division. Exemple : 8 div 4
=	Égalité. Exemple : 4 = 4 (vrai)
!=	Pas égal. Exemple : 5 != 3
<	Moins que. Exemple : 6 < 9
<=	Inférieur ou égal à. Exemple : x <= 6
>	Plus grand que. Exemple : 9 > 6
>=	Supérieur ou égal à. Exemple : n >= 7
ou	Alternative. Exemple : n = 6 ou n <= 6

et	Combinaison. Exemple : n = 5 et m < 8
mod	Division du module. Exemple : 5 mod 2
obtenir une propriété	Obtenez une valeur de propriété. Exemple : getProperty (« montant »)
obtenir des ressources	Obtenez une affectation de ressources. Exemple : getResource ('w1',1)

Note

La langue d'expression peut être définie dans la fenêtre Configurer BPSim, dans l'onglet « Configurer » ; les deux options « XPath 1.0 » et « Java » sont disponibles comme valeurs du paramètre « Expression ».

Si vous sélectionnez « Java », vous devez définir la propriété « JDK Home » sur un répertoire JDK valide.

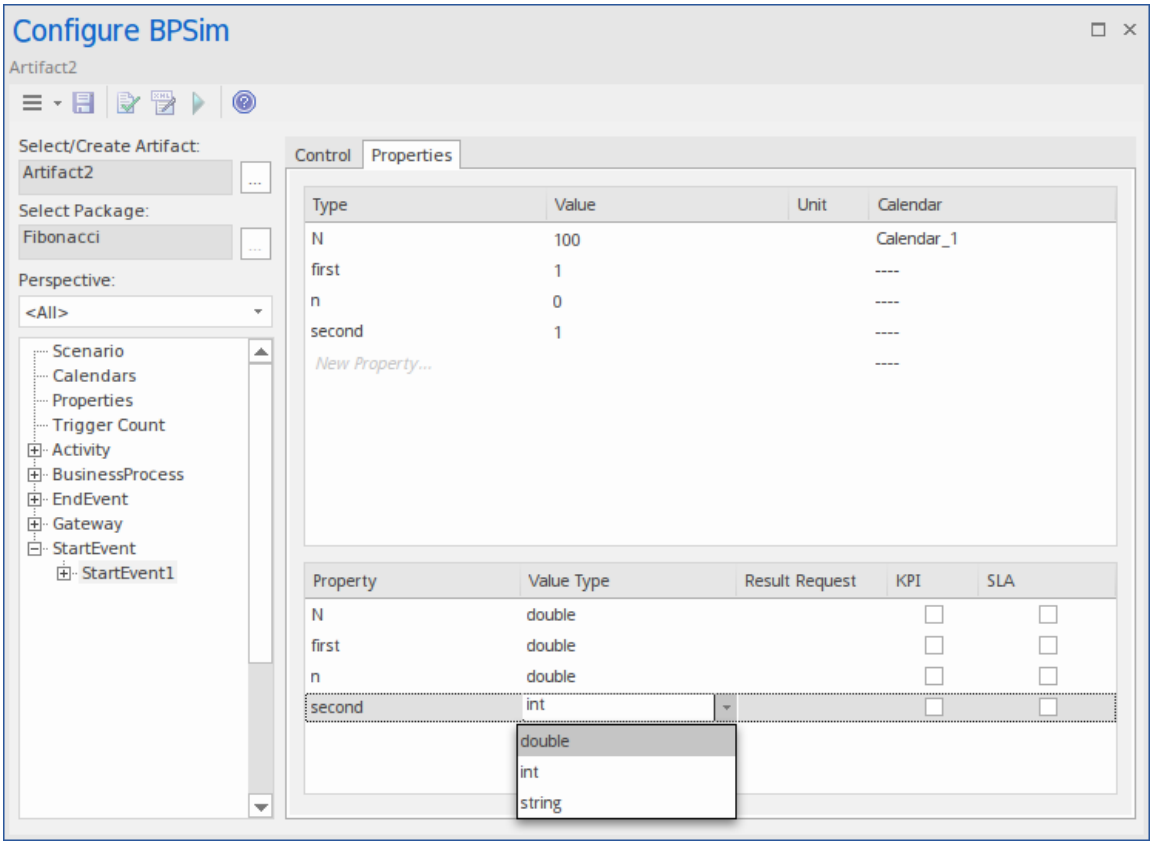
Conseil : Vous pouvez utiliser {PropertyParameterName} comme forme courte de getProperty('PropertyParameterName'), ce qui est utile lors de l'écriture de la valeur des expressions ; par exemple :

{n} < {N} au lieu de getProperty('n') < getProperty('N')

La forme courte de l'opérateur getProperty peut être utilisée à la fois dans XPath 1.0 et Java.

Paramètres de propriété BPSim

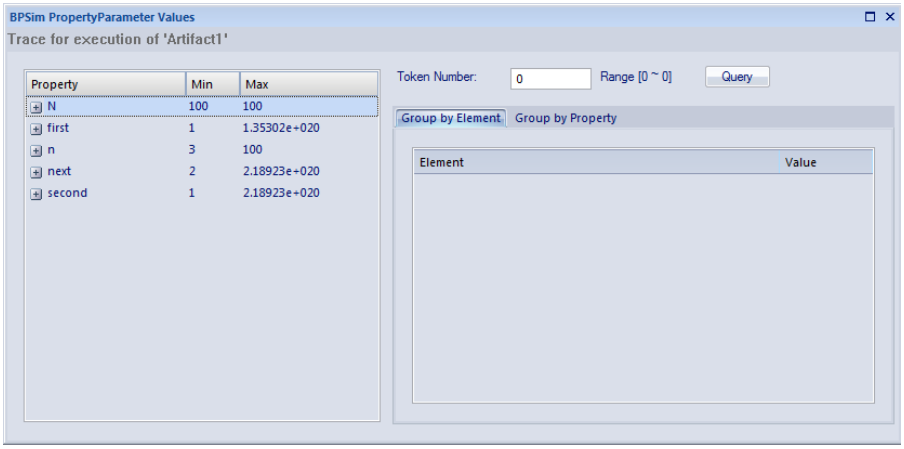
À partir de la version 13.0 Enterprise Architect , les paramètres de propriété BPSim peuvent contenir des données d'instance de processus auxquelles aucune valeur n'est attribuée jusqu'au moment exécuter . Vous pouvez définir le type de paramètre de propriété sur la page « Propriétés » de la fenêtre Configurer BPSim ; les types pris en charge sont « int », « double » et « string ».


Configure BPSim
Artifact2
Select/Create Artifact:
Artifact2
Select Package:
Fibonacci
Perspective:
<All>
Scenario
Calendars
Properties
Trigger Count
Activity
BusinessProcess
EndEvent
Gateway
StartEvent
StartEvent1
Control
Properties
Type
Value
Unit
Calendar
N
100
Calendar_1
first
1
n
0
second
1
New Property...
Property
Value Type
Result Request
KPI
SLA
N
double
first
double
n
double
second
int
double
int
string

Suivi des valeurs Paramètres Propriété

Le simulateur Processus Métier (BPSim) vous aide à modéliser et tester les détails de fonctionnement d'un processus métier, tels que l'affectation des ressources aux activités et aux tâches, l'intervention des événements et l'impact des points de décision et des décisions prises à ces points. Vous ajoutez ces paramètres de propriété spécifiques au processus, ou attributs, à la configuration BPSim et, lorsque vous exécuter des simulations sur le modèle en fonction de la configuration, le moteur BPSim vous aide à capturer les valeurs d'exécution des paramètres de propriété pour chaque itération de la simulation et à filtrer les résultats pour examiner des chemins ou des points de décision spécifiques. Cela vous donne un aperçu incroyablement détaillé de ce qui pourrait réellement se produire dans votre processus métier dans une condition ou une combinaison de conditions spécifique, soit pour générer un résultat, soit pour montrer quel chemin de traitement produit ce résultat.

Accéder

Menu Contexte	Cliquez-droit sur une configuration Processus Métier Simulation définie Artefact | Simuler BPMN avec BPSim... : Exécuter (sélectionner le type de simulation) : Attributes (Le bouton Attributes n'est pas disponible si la configuration de simulation ne contient aucun paramètre de propriété) 

Champs dialogue Valeurs des PropertyParameter BPSim

Option	Action
Propriété	Ce tableau répertorie les propriétés définies pour le processus et affiche les valeurs minimales et maximales possibles pour chaque propriété pour l'ensemble du processus. Si vous cliquez sur la zone d'extension d'une propriété, le tableau affiche les valeurs minimales et maximales de la propriété à chaque activité ou événement (élément) pendant le processus.
Numéro de jeton	Type le numéro du « jeton » à examiner ; ce numéro doit être compris dans la plage indiquée à droite du champ. Un « jeton » est un déclencheur indépendant, tel qu'un client ou une commande entrant dans une entreprise et déclenchant le processus d'entreprise sous révision . Il peut y avoir un nombre quelconque de clients ou de

	commandes, chacun n'ayant aucune relation avec un autre client ou commande, et chacun suivant potentiellement un chemin différent dans le processus d'entreprise. Lorsqu'il n'existe qu'une seule instance possible d'un événement déclencheur possible, comme l'activation d'un interrupteur marche/arrêt, le jeton est considéré comme étant 0.
Query	Cliquez sur ce bouton pour lancer une requête sur la simulation du jeton, pour remplir les onglets « Grouper par élément » et « Grouper par propriété ».
Grouper par élément	Affiche les résultats du point de vue de la façon dont valeur d'un paramètre de propriété change dans un élément sélectionné. L'onglet affiche une liste des éléments du processus et, pour chaque élément, la valeur de chaque propriété appliquée dans l'élément à chaque itération de la simulation. En utilisant l'option « Basculer Barre de Filtre » sur la barre d'en-tête, vous pouvez affiner l'affichage pour afficher uniquement une propriété particulière et voir à quelle fréquence elle est utilisée par l'élément et avec quelles valeurs.
Regrouper par propriété	Affiche les résultats du point de vue de la façon dont la valeur de chaque propriété change au cours de l'ensemble du processus. L'onglet affiche une liste des propriétés appliquées au cours du processus et, pour chaque propriété, la valeur de chaque activité (élément) à chaque itération du processus.

Exemples

Dans le Modèle EAExample, vous pouvez étudier deux exemples de génération d'informations sur les paramètres de propriété à partir d'une simulation d'un modèle de processus métier BPMN. Ceux-ci montreront comment définir les paramètres de propriété dans la configuration, en fonction du modèle. Vous pouvez d'abord simplement exécuter une simulation sur chaque exemple et examiner le résultat comme décrit ici. Vous pouvez ensuite examiner les processus métier et les configurations eux-mêmes, et modifier ou ajouter des paramètres de propriété fournis.

Les exemples sont décrits dans la rubrique *Tracking Property ParameterValues - Exemples* . En bref, ils sont les suivants :

- 'Fibonacci' - un processus métier récursif très simple qui calcule une série de nombres de Fibonacci à travers dix itérations ; vous pouvez voir comment les paramètres de propriété augmentent à chaque itération à travers les éléments du processus (dans Exemple Modèle > Simulation de Modèle > Modèles BPSim > Fibonacci)
- « Réparation automobile » - un processus plus complexe et réaliste qui représente ce qui pourrait se produire lorsqu'une série de clients individuels « sans rendez-vous » apportent des véhicules dans un atelier de réparation automobile pour estimation et réparation (dans Exemple de Modèle > Simulation de Modèle > Modèles BPSim > Processus de réparation automobile)

Il existe également un petit exemple du comportement des paramètres de temps (dans Exemple Modèle > Simulation de Modèle > Modèles BPSim > Paramètre de temps).

Notes

- Si une configuration BPSim contient des demandes de résultats et qu'une simulation personnalisée est effectuée sur celle-ci, le « Rapport Simulation BPMN » affiche uniquement les paramètres intégrés demandés dans la configuration ; en revanche, la dialogue « Valeurs PropertyParameter BPSim » répertorie tous les paramètres de propriété, quels que soient les paramètres de demande de résultats ou le type de simulation

Valeurs Paramètres Propriété de suivi - Exemples

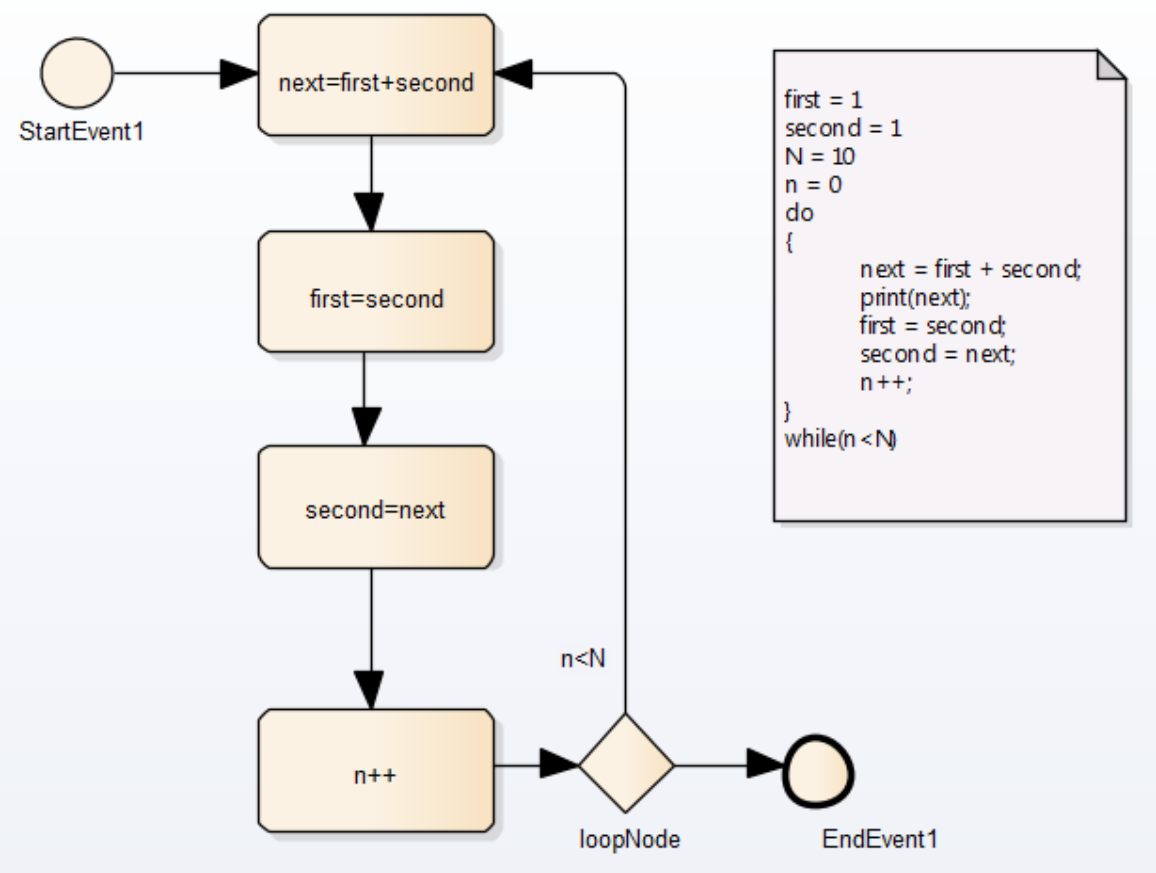
Pour vous aider à comprendre la facilité de génération d'informations sur les paramètres de propriété à partir d'une simulation d'un modèle de processus métier BPMN, Sparx Systems fournit deux exemples que vous pouvez explorer dans le modèle EAExample. Il s'agit des éléments suivants :

- Processus de Fibonacci - un exemple très simple pour vous aider à vous familiariser avec les facilités de suivi des paramètres
- Processus de réparation automobile - un exemple plus complexe que vous pouvez manipuler pour voir comment un processus réel peut être étudié

À la fin de cette rubrique se trouve une section qui explique brièvement comment vous pouvez travailler avec un processus basé sur des entiers contenant des paramètres initialisés par des distributions « réelles », et une section décrivant l'exemple du comportement des paramètres de temps.

L'exemple de Fibonacci

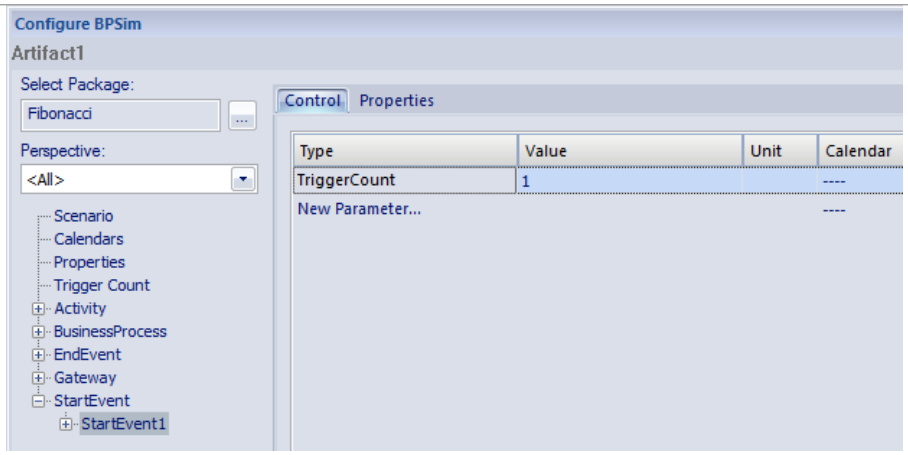
Il s'agit d'un processus métier récursif très simple qui calcule une série de nombres de Fibonacci au cours de dix itérations ; vous pouvez voir comment les paramètres de propriété augmentent à chaque itération à travers les éléments du processus. Ouvrez Example Modèle > Simulation de Modèle > BPSim Models > Fibonacci.



Le pseudo-code du processus est affiché dans l'élément Notes du diagramme . L'instruction 'print(next)' produira la série de nombres 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.

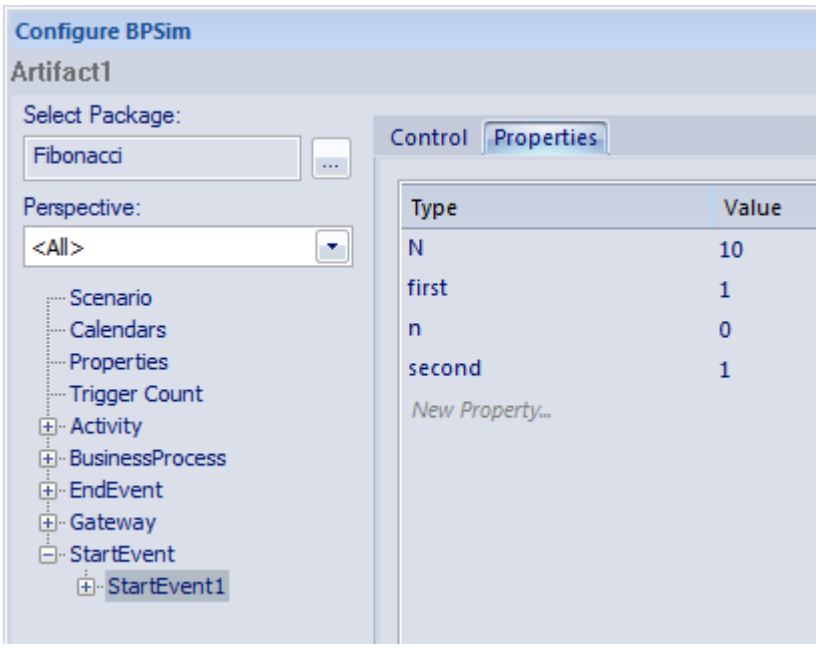
La configuration BPSim pour ce processus est configurée comme décrit ici.

Étape	Action
1	



Dans l'onglet « Contrôle » de l'élément StartEvent, définissez « TriggerCount » sur « 1 », puis dans l'onglet « Propriétés », créez et initialisez les propriétés :

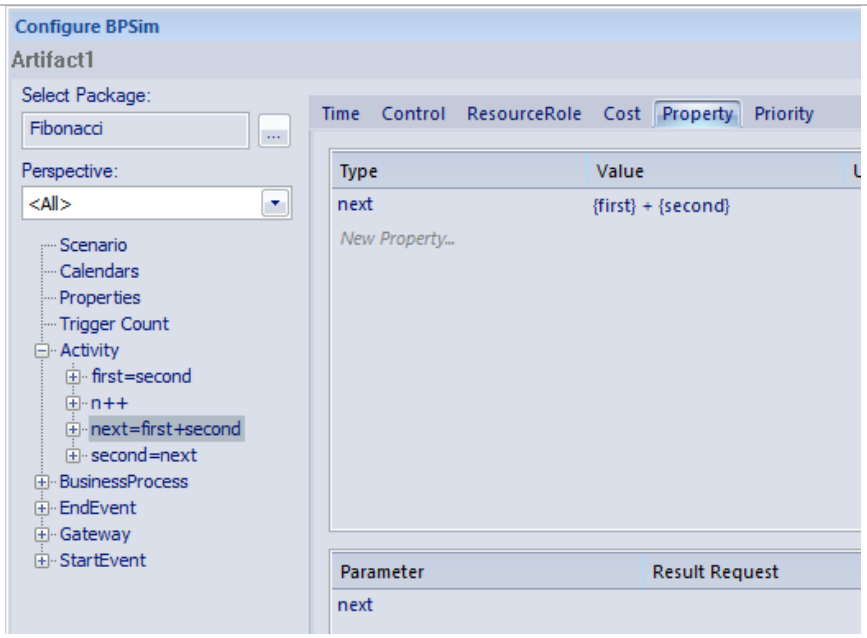
- « N » comme « 10 »
- « premier » comme « 1 »
- « second » comme « 1 »
- 'n' comme '0'



2

Définissez maintenant les propriétés de chacune des activités du processus, dans l'onglet « Propriétés ». Note que les valeurs de ces propriétés sont dérivées d' **expressions** , dont les composants doivent être placés entre accolades - {xxx}. Pour l'activité :

- next=first+second - définissez la propriété 'next' et définissez la valeur comme l'expression {first} + {second}



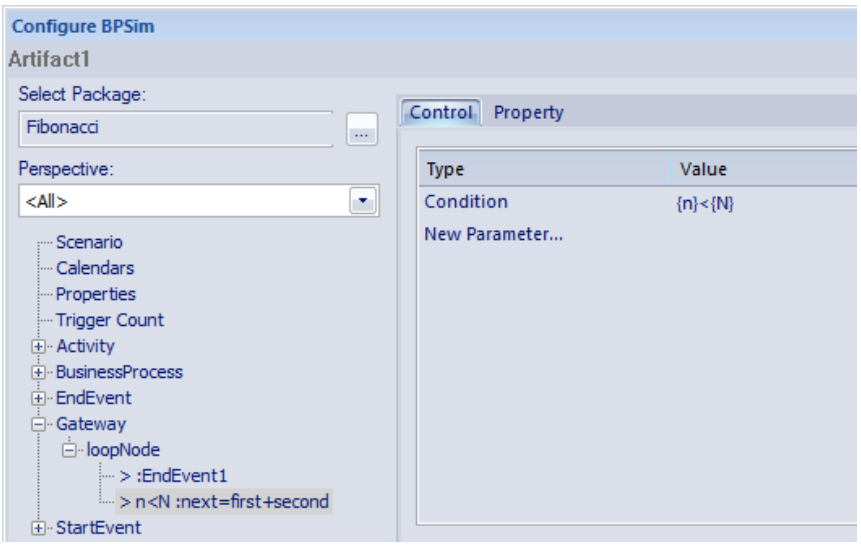
- first=second - définissez la propriété « first » et définissez la valeur comme l'expression {second}
- second=next - définissez la propriété « second » et définissez la valeur comme l'expression {next}
- n++ - définir la propriété 'n' et définir la valeur comme l'expression {n} + 1

3

Définissez les paramètres de la propriété « Condition » pour les deux connecteurs Sequenceflow issus de l'élément Passerelle « loopNode », dans l'onglet « Contrôle ».

Développez l'élément Passerelle | loopNode et pour le lien vers :

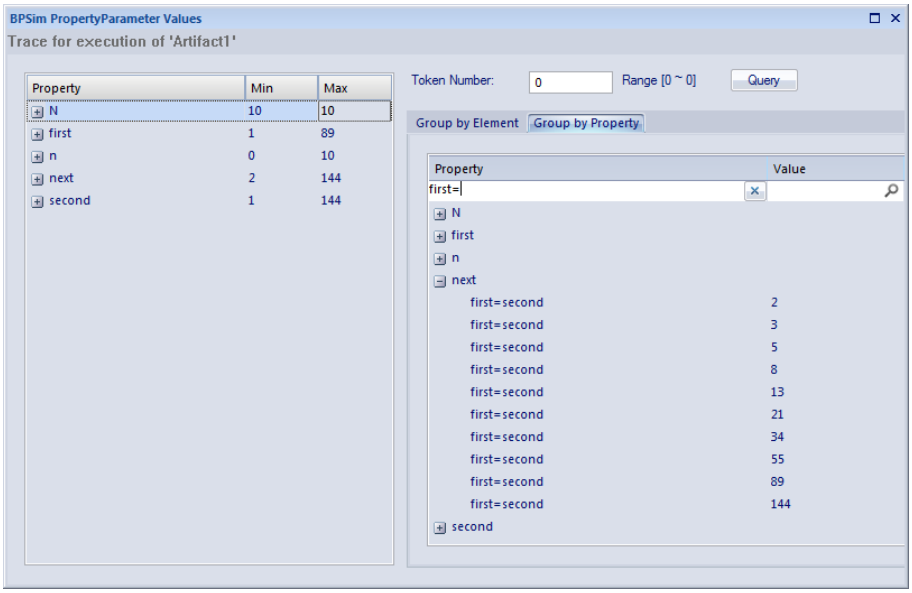
- next=first+second - définissez le paramètre Control sur « Condition » et définissez la valeur comme Expression {n}<{N}



- EndEvent1 - définissez le paramètre Control sur « Condition » et définissez la valeur comme Expression {n}=>{N}

4

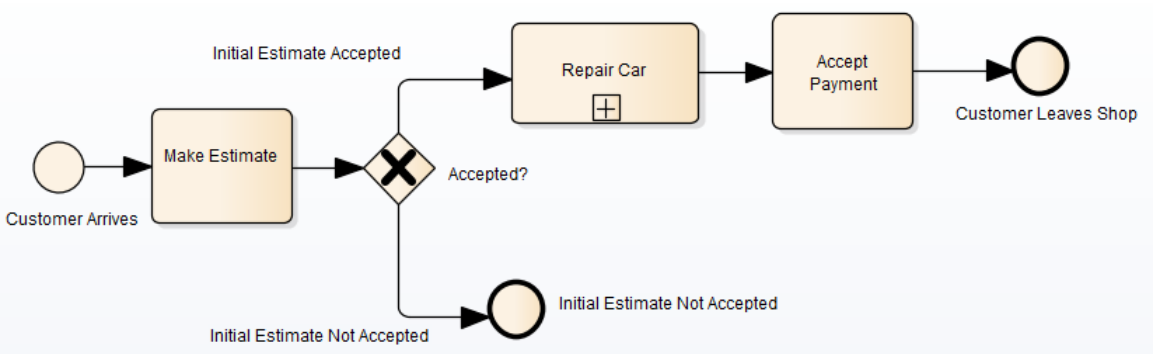
Une fois la configuration terminée, cliquez sur le bouton Exécuter de la fenêtre Configurer BPSim et sur la dialogue 'Contrôleur Simulation BPSim' en sélectionnant une simulation Standard.

	Une fois la simulation terminée, cliquez sur le bouton Attributes . Dans la dialogue « Valeurs PropertyParameter BPSim », définissez le champ « Numéro de jeton » sur « 0 » et cliquez sur le bouton Query .
5	Examinez maintenant les valeurs de la propriété « next » lors de la saisie de la première = seconde activité à chaque itération de la simulation. Cliquez sur l'onglet « Grouper par propriété » et développez l'élément « next ». La liste des valeurs est longue, cliquez-droit donc sur les en-têtes de colonne et sélectionnez l'option « Activer/désactiver Barre de Filtre ». Sous l'en-tête de colonne « Propriété », saisissez « first= ». Cela filtre la liste pour afficher uniquement les valeurs des paramètres de propriété lors de la saisie de l'activité first=second. 

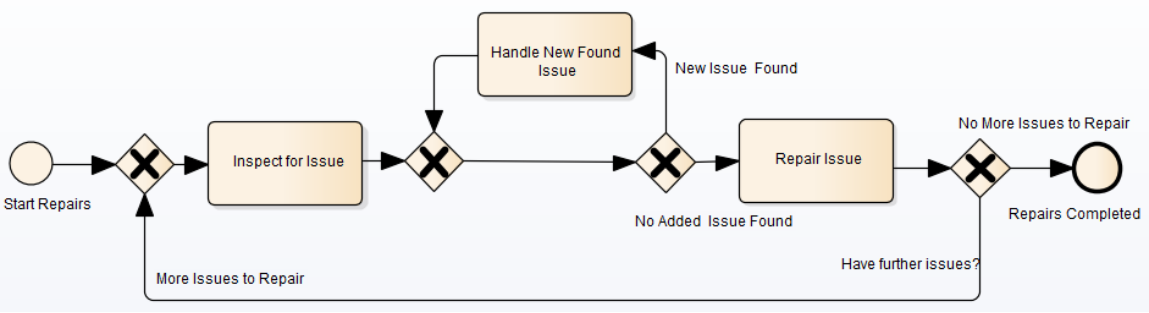
L'exemple de la réparation automobile

Cet exemple plus complexe est basé sur un modèle réaliste d'un processus de réparation automobile, où un certain nombre de clients individuels demandent un devis de réparation et procèdent à la réparation ou refusent de continuer ; vous pouvez voir comment les paramètres de propriété varient à mesure que différentes décisions sont prises au cours du processus. Ouvrez Example Modèle > Simulation de Modèle > BPSim Models > Car Repair Process.

Le processus global est représenté par ce diagramme :



L'activité *Réparation de voitures* est un élément composite qui contient ce diagramme sous-processus :



Étape	Action
1	Dans la fenêtre Navigateur , développez le Paquetage enfant « BPSim » sous le Paquetage « Processus de réparation automobile » et double-cliquez sur l'artefact « Scénario 1 : Flux principal ». La fenêtre Configurer BPSim s'affiche. Dans l'onglet « Scénario », regardez le champ « Durée » ; celui-ci a été défini sur 2 jours et 12 heures (soit 60 heures).
2	Dans la hiérarchie des processus à gauche de la fenêtre, développez la catégorie ' Démarrer Event' et cliquez sur 'Client arrive'. Sélectionnez l'onglet 'Contrôle' et regardez le paramètre 'InterTriggerTimer' qui a la valeur 24 minutes ; c'est-à-dire qu'un client arrive toutes les 24 minutes (donc sur la durée de 60 heures, 150 clients passent par l'atelier de réparation). Chaque client entre dans l'atelier de réparation avec un ou plusieurs problèmes à évaluer et à réparer. Le nombre de problèmes que chaque client présente peut être généré aléatoirement en utilisant l'une des distributions prises en charge par BPSim. Comme le nombre de problèmes est compté en unités discrètes (plutôt que mesuré sur une échelle continue), nous utiliserons une distribution « integer ». Si vous sélectionnez l'onglet « Propriétés » pour l'événement Démarrer « Arrivée du client », vous verrez que la valeur de la propriété « noOfIssues » est initialisée à partir d'une distribution de Poisson avec une moyenne de 3.
3	Développez maintenant la Passerelle de décision « Acceptée ? » et ses connecteurs dans la hiérarchie des processus. « Estimation initiale acceptée » a un paramètre de contrôle « Probabilité » défini sur 0,67. Le connecteur alternatif, « Estimation initiale non acceptée » a un paramètre de contrôle similaire « Probabilité » défini sur 0,33. Autrement dit, nous nous attendons à ce qu'en moyenne, un problème sur trois soit retiré (ou non traité) par le client.
4	Plus loin dans le processus, lorsqu'un problème est évalué sur le véhicule, il est possible qu'un autre problème soit découvert. Dans la liste des éléments Passerelle , le dernier « élément sans nom » a deux chemins : « Nouveau problème détecté » et « Aucun problème ajouté détecté ». Cliquez sur chacun d'eux et regardez l'onglet « Contrôle » ; le paramètre « Probabilité » pour « Nouveau problème détecté » est défini sur 0,25 et, pour « Aucun problème ajouté détecté », sur 0,75. Ainsi, en moyenne, pour quatre problèmes signalés et évalués, un nouveau problème est découvert. Le chemin « Nouveau problème détecté » amène le processus vers l'activité « Gérer les nouveaux problèmes détectés », qui ajoute 1 au nombre de problèmes à traiter pour le client actuel. Développez le groupe d'activités et cliquez sur l'élément « Gérer les nouveaux problèmes détectés » et sur l'onglet « Propriété ». Vous verrez que la propriété « noOfIssues » ici a la valeur Expression {noOfIssues} **+ 1** .
5	Lorsqu'un problème avec le véhicule est résolu, l'activité « Problème de réparation » déduit 1 du nombre de problèmes à réparer pour le client actuel. Cliquez sur l'élément « Problème de réparation » dans le groupe Activité et sur l'onglet « Propriété ». Vous verrez que la propriété « noOfIssues » ici a la valeur Expression {noOfIssues} **- 1** .

6	La valeur de l'activité « Réparer le problème » est testée à la Passerelle « Vous avez d'autres problèmes ? ». Cliquez sur le connecteur « Autres problèmes à réparer » et sur l'onglet « Contrôle » ; le paramètre Condition pour suivre ce chemin est défini sur la valeur Expression {noOfIssues} **> 0** ; le flux passe à la Passerelle avant l'activité « Inspecter pour détecter un problème ». De même, si vous cliquez sur le connecteur « Plus de problèmes à réparer » et sur l'onglet « Contrôle », le paramètre Condition pour suivre ce chemin est défini sur la valeur Expression {noOfIssues} **=< 0** et le flux passe à l'événement de fin « Réparations terminées ». Maintenant que vous avez examiné le flux de processus et les paramètres de configuration, vous pouvez exécuter une simulation et révision les résultats.
7	Dans la fenêtre Configurer BPSim, cliquez sur le bouton Exécuter , puis dans la dialogue 'Contrôleur Simulation BPSim', cliquez à nouveau sur le bouton Exécuter , en sélectionnant la simulation 'Standard' (cependant, le type de simulation ne fait aucune différence pour les paramètres de propriété réviser). Dans la dialogue « Contrôleur Simulation BPSim », vous pouvez révision le statut du jeton (et constater qu'un client supplémentaire parvient à entrer dans la boutique à la toute dernière minute), mais il est difficile de voir exactement comment ces données récapitulatives ont été générées. Cliquez sur le bouton Attributes pour obtenir les informations détaillées sur les valeurs des paramètres de propriété dans la dialogue « Valeurs PropertyParameter BPSim ».
8	Sur le côté gauche de le dialogue se trouve un résumé des valeurs minimales et maximales du paramètre de propriété (attribut) pour chaque élément du processus. Par exemple, pour l'élément « Arrivée du client », le paramètre « noOfIssues » a un minimum de 0 et un maximum de 8, comme généré par la distribution de Poisson (3). Dans le champ « Numéro de jeton », saisissez un nombre (N) compris entre 0 et 150 à sélectionner pour le Nième client entré dans l'atelier de réparation. Cliquez sur le bouton Query pour obtenir les valeurs des paramètres de propriété utilisés dans le processus pour ce client. Révision les résultats sur chacun des deux onglets : • Dans l'onglet « Grouper par élément », voyez comment la valeur de l'attribut change dans chaque élément ; par exemple, pour le client 24, le paramètre « noOfIssues » est initialisé avec une valeur de 4 par la distribution aléatoire, et l'activité « Inspecter les problèmes » est appelée six fois avec la valeur du paramètre ajustée à 3 pour trois de ces appels avant de passer à 1, et l'activité « Gérer les nouveaux problèmes détectés » est appelée deux fois avec la valeur du paramètre à 3 à chaque fois • Dans l'onglet « Grouper par propriété », voyez comment la valeur du paramètre change à mesure que le processus parcourt les activités jusqu'à leur achèvement, en commençant par 4, en étant ajusté entre 3 et 4 un certain nombre de fois, puis en décrémentant jusqu'à 0 à la fin
9	Continuez à explorer les résultats selon vos besoins, en sélectionnant différents clients (Tokens). Vous pouvez également revenir à la configuration BPSim et modifier les initialisations des paramètres et en ajouter de nouvelles, ou modifier les points de décision, pour expérimenter le processus.

Répondre aux nombres réels dans la simulation d'un processus basé sur des nombres entiers

Dans certains cas, vous devrez peut-être générer des valeurs de paramètres de propriété à l'aide d'une distribution qui renvoie des nombres « réels » lorsque les activités du processus fonctionnent avec des entiers, ou lorsque vous souhaitez voir quel impact le forçage des valeurs integer a sur le processus.

Un mécanisme à appliquer dans de tels cas consiste à définir des conditions pour éviter les nombres absolus. Ainsi, par exemple, vous pouvez avoir un compteur qui décrémente de 1, qui est initialisé à un nombre « réel ». Si vous définissez une condition sur « valeur==0 » (égal à 0) ou « valeur !=0 » (différent de 0), les deux conditions peuvent ne jamais être vraies ou toujours être vraies, respectivement, provoquant une boucle infinie. Pour éviter cela, dans les conditions, vous utiliseriez des opérateurs tels que :

' valeur > 0 '

' valeur < 0'

' valeur >= 0'

' valeur <= 0'

Un autre mécanisme consiste à éditer le gabarit de code utilisé par le moteur BPSim, pour intercepter et remplacer les nombres réels fournis à des paramètres spécifiques par des entiers, comme indiqué :

1. Sélectionnez l'option de ruban « Développer > Code source > Options > Modifier le code Gabarits ».
2. Dans l'éditeur de code Gabarit , dans le champ « Langue », cliquez sur la flèche déroulante et sélectionnez « MDGBPSimExecutionEngineExtension ».
3. Dans la liste des gabarits (Java), cliquez sur 'MDGBPSimExecutionEngineExtension Compute Value'. Le contenu gabarit s'affiche dans le panneau ' Gabarit '.
4. Trouvez cette ligne :

   double %bpsimPropertyParameterName% = (double) distribution.next();

   Changez-le en :

   % **si** bpsimPropertyParameterName == "noOfIssues" ou bpsimPropertyParameterName == "noOfVisitors"%
   double %bpsimPropertyParameterName% = (int) distribution.next();
   //double %bpsimPropertyParameterName% = ceil (distribution.next());
   //double %bpsimPropertyParameterName% = Math. floor (distribution. next());
   //double %bpsimPropertyParameterName% = round (distribution.next());
   % **autre** %
   double %bpsimPropertyParameterName% = (double) distribution.next();
   % **finSi** %
5. Remplacez les noms des paramètres de propriété par vos propres paramètres de propriété.
6. Cliquez sur le bouton Enregistrer, fermez l'éditeur de code Gabarit et rechargez le projet.

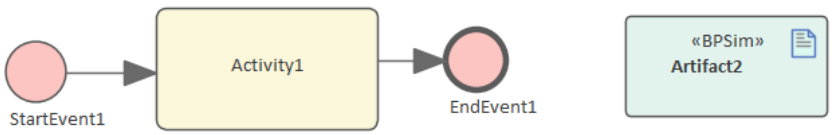
Comme présenté, pour chaque paramètre spécifié, le gabarit de code remplacera simplement tout nombre « réel » initialisé par la distribution par un integer . Si vous préférez, vous pouvez utiliser l'une des lignes commentées à la place :

- ceil () prendra le nombre « réel » et le convertira en integer le plus élevé suivant
- Math. floor () prendra le nombre « réel » et le convertira en integer le plus bas suivant
- Math. round () prendra le nombre « réel » et l' round vers le haut ou vers le bas selon qu'il est supérieur ou inférieur à n.5

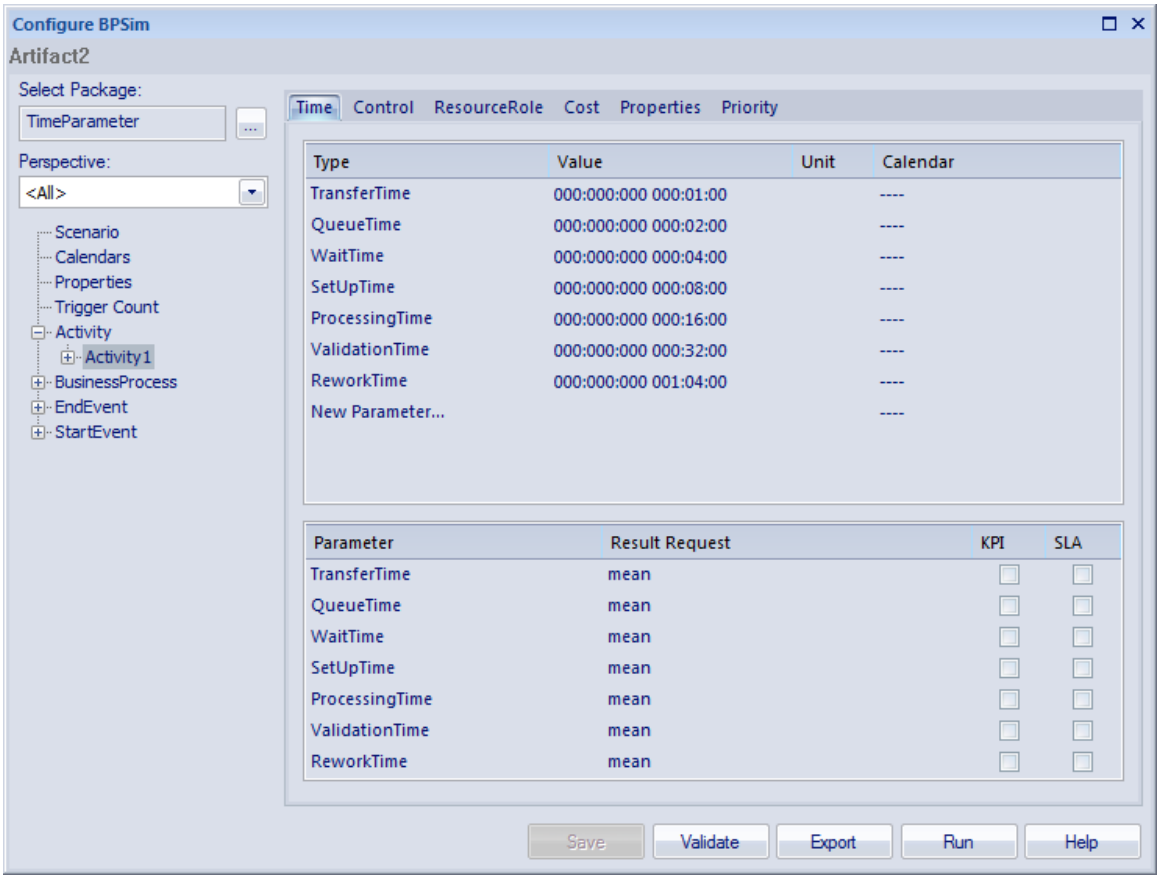
Comportement des paramètres temporels

Dans la configuration BPSim, vous pouvez définir un certain nombre de paramètres de temps pour une activité, tels que le temps d'attente et le temps d'attente. Vous pouvez également définir une demande de résultat sur chacun d'eux, pour une simulation personnalisée. Cependant, le moteur de simulation BPSim combine ces paramètres en une seule quantité « Temps de traitement ».

Considérons le modèle simple TimeParameter dans le Modèle d'exemple (Modèle d'exemple > Simulation de Modèle > Modèles BPSim > Paramètre de temps), représenté par ce diagramme :

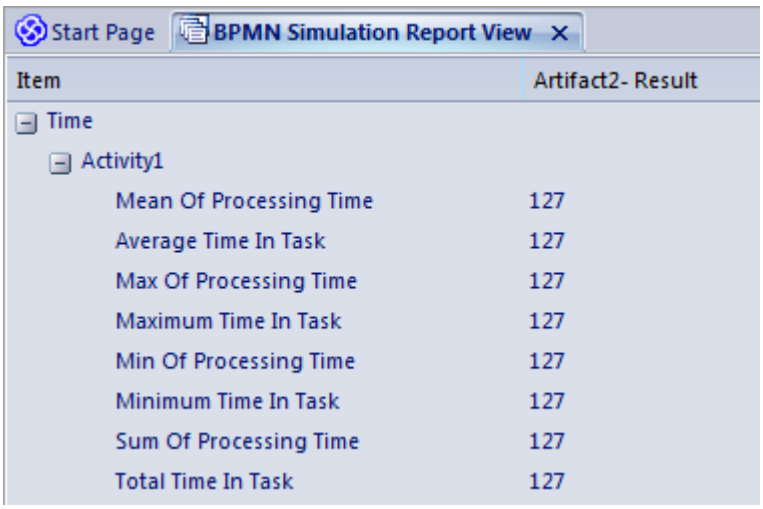


Si vous double-cliquez sur l'élément Artifact2, la fenêtre Configurer BPSim s'affiche. Cliquez sur l'élément Activity1 dans le diagramme pour développer le groupe Activity, pour sélectionner Activity1 dans la hiérarchie à gauche de le dialogue et pour afficher le premier onglet, 'Time', pour l'élément dans la configuration, comme indiqué.



Note que dans le panneau supérieur, il y a sept paramètres « Heure » fournis par le système, auxquels ont été attribuées des valeurs initiales de - 1, 2, 4, 8, 16, 32 et 64 minutes (les 64 minutes correspondent à 1 heure et 4 minutes). Note également que dans le panneau inférieur, chacun d'entre eux possède une demande de résultat pour la valeur moyenne de durée d'exécution du paramètre.

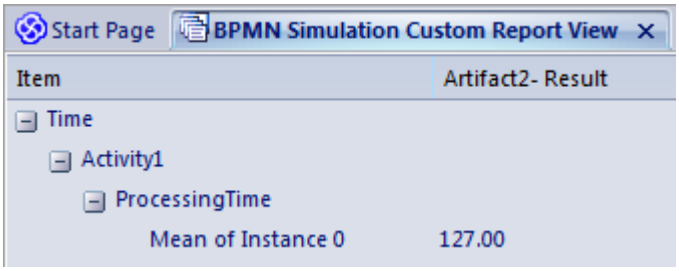
Cliquez sur le bouton Exécuter , puis dans la dialogue « Contrôleur Simulation BPSim », cliquez sur le bouton Exécuter et sélectionnez « Simulation standard ». La simulation est configurée pour parcourir le processus une fois. Lorsque la simulation est terminée, cliquez sur le bouton Ouvrir le résultat, puis dans la boîte de dialogue « Rapport Vue Simulation BPMN », cliquez-droit et sélectionnez l'option « Afficher uniquement Items non vides ». Cela vous donne, pour l'élément Activity1 sur lequel les paramètres ont été définis, ces résultats :



Tous ces résultats dérivés correspondent à 127 minutes, soit la somme des valeurs initiales des sept paramètres « Heure » d'origine. Les paramètres individuels ne sont pas traités séparément.

Si vous revenez à la dialogue « Contrôleur Simulation BPSim » et cliquez sur le bouton Exécuter , en sélectionnant cette fois « Simulation personnalisée », le bouton Ouvrir le résultat affiche le « Rapport Vue personnalisé Simulation BPMN

». Dans la configuration, les demandes de résultats concernaient les valeurs moyennes des sept paramètres. Dans le Rapport Vue de la simulation, vous ne voyez que la moyenne du paramètre agrégé unique, ProcessingTime, à 127 minutes.



Comparer Configurations BPSim

Lorsque vous développez une configuration BPSim, vous pouvez définir une large gamme de paramètres à définir avant d'exécuter des simulations et d'observer les effets de ces paramètres et des modifications apportées aux paramètres sélectionnés. Pour faciliter l'utilisation et la gestion de plusieurs scénarios hypothétiques, il est recommandé de créer des copies de la configuration d'origine (en tant qu'éléments d'artefact) et d'effectuer les modifications de paramètres dans les copies.

Une facilité utile pour créer des variantes d'une configuration consiste à appliquer l'héritage, selon laquelle les données et les paramètres que vous n'avez PAS l'intention de modifier sont conservés dans une configuration, et seuls les paramètres que vous modifiez sont conservés dans une autre. La configuration « variable » utilise (hérite) des données communes contenues dans la configuration de base, vous n'avez donc pas besoin de recréer ces données communes dans la configuration « variable ».

Vous pouvez ensuite exécuter des simulations sur les configurations modifiées et sur la « ligne de base » d'origine et comparer les rapports de simulation pour voir quelles différences se sont produites dans les variables d'exécution, puis exécuter et afficher les comparaisons des configurations pour voir quelles modifications dans les paramètres ont donné lieu à ces différences d'exécution.

En exécutant des simulations sous les configurations d'origine et de copie, en comparant les résultats et les modifications qui ont provoqué les résultats, et en modifiant le modèle en conséquence, vous pouvez atteindre un très haut degré de contrôle dans la rationalisation du processus métier que vous développez.

Accéder

Menu Contexte	Dans un diagramme ou dans la fenêtre Navigateur | Cliquez-droit Processus Métier Simulation Artifact | Afficher la configuration BPSim > onglet Révision > onglet Résumé de la configuration

Onglet Résumé de la configuration

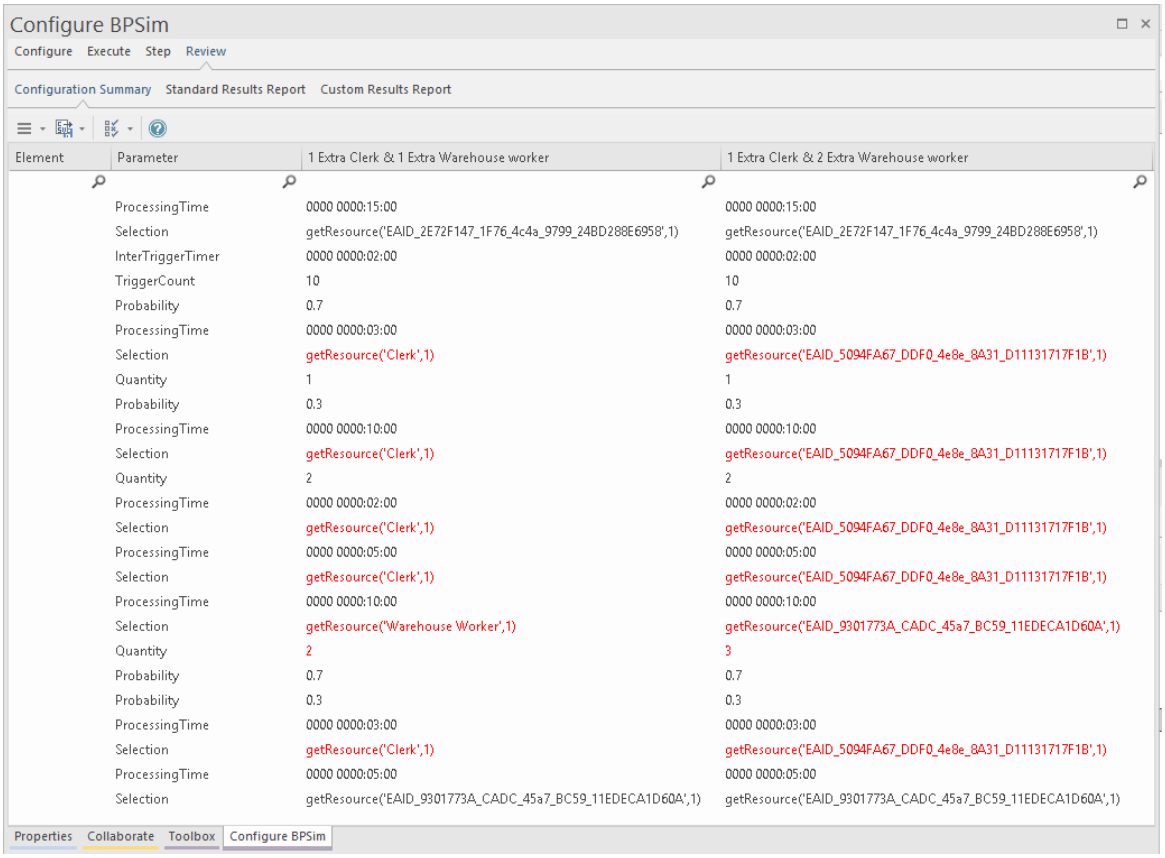
Cette vue affiche initialement les paramètres de la configuration sélectionnée qui ont des valeurs et, dans la colonne sous le nom de la configuration, les valeurs définies pour ces paramètres. Si la fenêtre est ancrée, vous pouvez révision les résultats plus facilement en faisant glisser l'onglet hors de l'espace de travail pour le transformer en affichage flottant et en étendant l'affichage à la taille plein écran.

Pour comparer deux (ou plusieurs) configurations, cliquez sur un autre élément d'artefact de configuration BPSim dans la fenêtre Navigateur ou dans diagramme et :

- Faites-le glisser sur la vue du rapport ou
- Sélectionnez l'option « Afficher la configuration BPSim »

Vous pouvez également cliquer sur l'icône  dans l'onglet « Résumé de la configuration », cliquer sur l'option « Ajouter une configuration BPSim » et rechercher et sélectionner l'élément Artefact dans la dialogue « Sélectionner l'élément ».

La hiérarchie des paramètres contient maintenant tous les paramètres supplémentaires de cette configuration, et ses valeurs de paramètres s'affichent dans une colonne à gauche des valeurs de configuration d'origine, avec le nom de l'artefact dans l'en-tête de colonne.



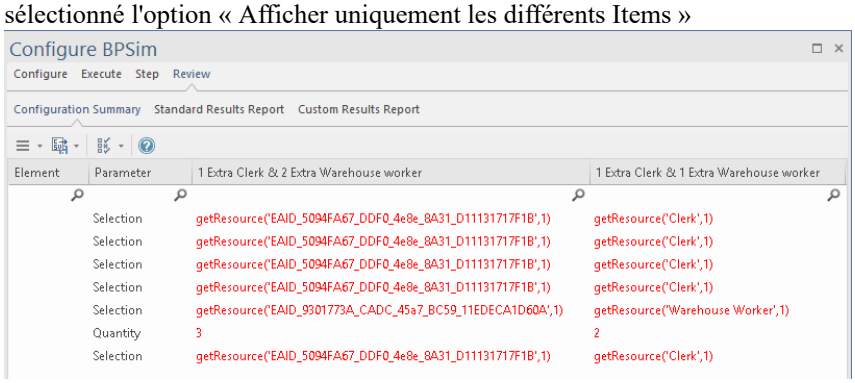
Vous pouvez révision et manipuler les informations du rapport à l'aide des options disponibles dans la barre d'outils et le menu contextuel cliquez-droit .

Options de résumé de configuration Simulation BPMN

Option	Description
	Sélectionnez cette option de la barre d'outils pour afficher un court menu d'options : • Ajouter une configuration BPSim - affiche la dialogue « Sélectionner un élément », à travers laquelle vous pouvez rechercher et sélectionner un autre artefact BPSim et afficher ses paramètres à côté de ceux déjà présents sur la page • Recharger - actualiser l'affichage et rétablir les modifications des colonnes aux largeurs et positions d'origine • Effacer la liste - efface les artefacts sélectionnés et leurs paramètres de l'affichage • Supprimer Rapport <nom> - efface l'artefact nommé et ses paramètres de la page ; l'une de ces options est fournie pour chaque artefact actuellement affiché sur la page
	Cliquez sur cette icône de la barre d'outils pour exporter les enregistrements de cette page vers un fichier CSV ou XML. Le navigateur « Enregistrer sous » s'affiche, à partir duquel vous pouvez sélectionner l'emplacement dans lequel enregistrer le fichier.
	Affiche deux options pour filtrer les informations affichées sur la page ; les options

sont activées lorsque vous avez deux ou plusieurs configurations affichées :

- Afficher uniquement Items différents - sélectionnez cette option pour restreindre l'affichage aux seuls paramètres dont les valeurs diffèrent entre les configurations ; cliquez à nouveau sur l'option pour la désélectionner
- Mettre en surbrillance les différents Items - (lorsque deux ou plusieurs configurations ont des valeurs de paramètres différentes) affiche les différentes valeurs de paramètres en rouge ; cette option est désactivée si vous avez déjà sélectionné l'option « Afficher uniquement les différents Items »



Graphiques BPSim

La page « Graphiques » de la boîte à outils Diagramme propose deux icônes spécifiquement destinées à générer des graphiques qui reflètent les résultats sélectionnés des simulations BPSim. Il s'agit de :

- Graphique de résultats BPSim - pour générer un graphique qui reflète les résultats sélectionnés à partir d'une série de simulations BPSim standard
- Graphique de résultats personnalisé BPSim - pour générer un graphique qui reflète les résultats d'une série de simulations BPSim personnalisées

Comme pour les autres artefacts graphiques, les deux types de graphiques BPSim peuvent être rapidement configurés pour afficher les résultats de la simulation dans des variantes d'un graphique chronologique, d'un graphique à barres bidimensionnel ou d'un graphique à barres tridimensionnel.

Prérequis

Pour renseigner les graphiques créés à partir des artefacts Simulation Processus Métier , vous sélectionnez les artefacts de résultat créés lors de la simulation de chaque configuration que vous souhaitez afficher. Par conséquent, les simulations initiales doivent être effectuées en premier, puis les artefacts Rapport générés.

Accéder

Affichez la page « Graphiques » de la boîte à outils Diagramme en utilisant l’une des méthodes décrites dans ce tableau .

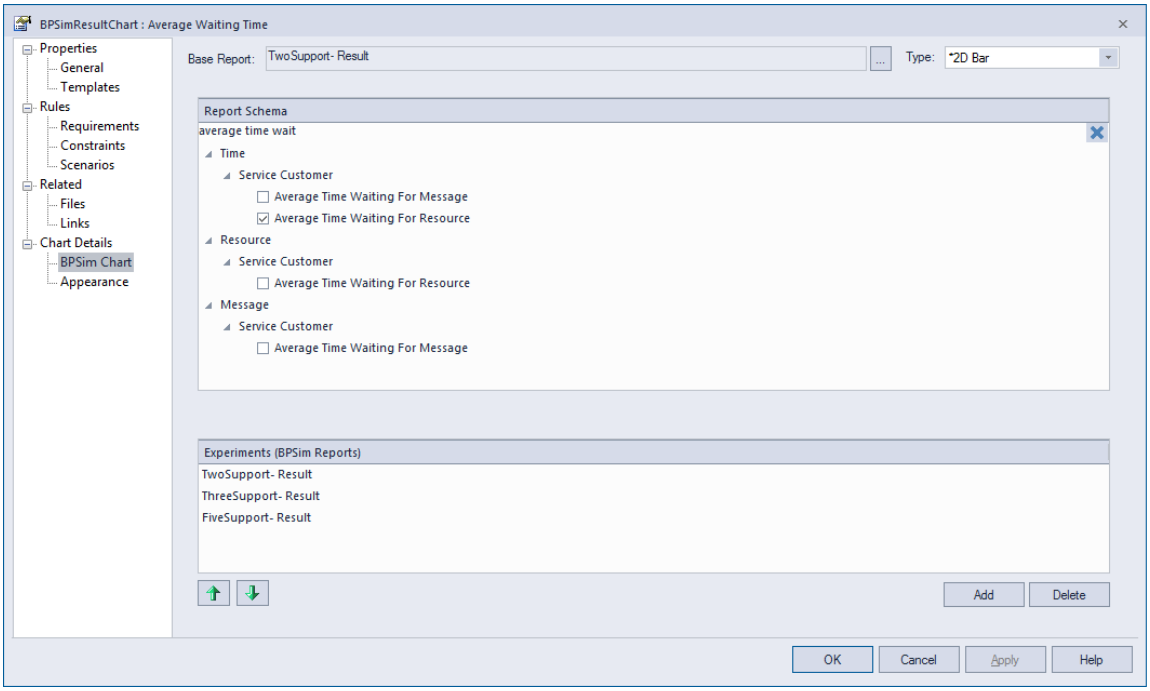
Ensuite, faites glisser l'icône d'artefact de graphique BPSim <type> sur le diagramme : un nouvel élément de graphique est créé.

Double-cliquez sur le nouvel élément Graphique pour ouvrir la dialogue « Propriétés », affichant la page « Graphique BPSim ».

Ruban	Conception > Diagramme > Boîte à outils > Graphiques
Raccourcis Clavier	Ctrl+Maj+3 |Graphiques
Autre	Vous pouvez afficher ou masquer la boîte à outils Diagramme en cliquant sur les icônes ou à l'extrémité gauche de la barre de légende en haut de la Vue Diagramme .

Sélectionner les résultats à afficher dans le graphique

Remplissez les champs de la page « Graphique BPSim » de la dialogue « Propriétés » du graphique.



Option	Action
Rapport de base	Cliquez sur le bouton ... pour afficher la dialogue « Sélectionner << BPSimReport>> Artefact » et sélectionnez l'artefact de rapport pour les résultats de simulation avec lesquels les autres résultats de simulation seront comparés. Cet artefact de résultat est également ajouté au panneau « Expériences » comme premier dans la liste des résultats de rapport à comparer.
Type	Cliquez sur la flèche déroulante et sélectionnez le type de format du graphique dans lequel afficher les résultats : Barre 2D, Barre 3D ou Ligne. Après avoir spécifié les paramètres du rapport à comparer, vous pouvez sélectionner la page « Apparences » de le dialogue et définir l'apparence du graphique à barres ou du graphique chronologique.
Schéma Rapport	Développez la hiérarchie si nécessaire et cochez la case en regard de chaque propriété à afficher sur le graphique. Chaque propriété sera représentée par une ligne ou un groupe de barres distinct sur le graphique. En général, vous sélectionnez des objets similaires (tels que des ressources différentes) et la même propriété unique pour chaque object (tel que le degré d'utilisation de la ressource). Vous disposez d'une multitude de propriétés à examiner et à comparer, mais plus de deux sur le même graphique rendent le graphique difficile à lire.
Expériences (Rapports BPSim)	Ce panneau répertorie les résultats du rapport de simulation (sous forme d'artefacts de résultat BPSim) que vous avez sélectionnés pour comparaison à l'aide du graphique, dans l'ordre dans lequel leur paramètre sélectionné sera affiché sur le graphique. En général, le Rapport de base reste en premier dans la liste et le résultat de son paramètre est affiché à gauche du graphique. Si vous souhaitez modifier l'ordre, cliquez sur le nom de l'artefact de résultat et cliquez sur le bouton fléché vert Haut/Bas approprié. Pour ajouter d'autres noms d'artefacts de résultat à la liste, cliquez sur le bouton Ajouter et recherchez et sélectionnez l'artefact dans la dialogue « Sélectionner << BPSimReport>> Artefact ».

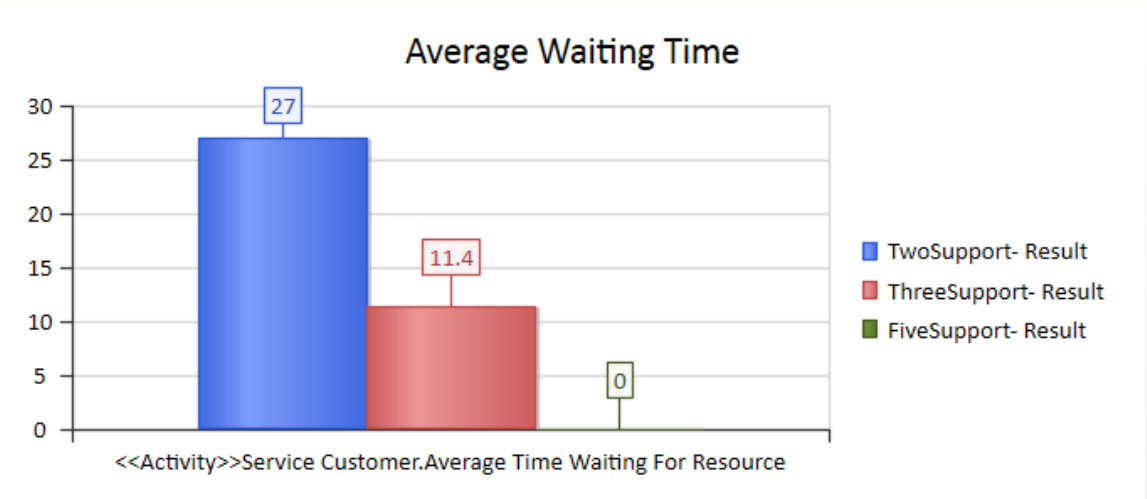
Un exemple de graphique BPSim

Dans l'exemple Support téléphonique du service d'assistance, nous avons créé trois artefacts BPSim pour effectuer une analyse hypothétique de « De combien de ressources support avons-nous besoin pour répondre aux questions des clients par téléphone de manière économique ? »

Nous avons commencé avec deux ressources support , puis nous avons essayé trois et cinq ressources. Après simulation, nous avons un Rapport BPSim basé sur différentes configurations : TwoSupport-Result, ThreeSupport-Result et FiveSupport-Result.

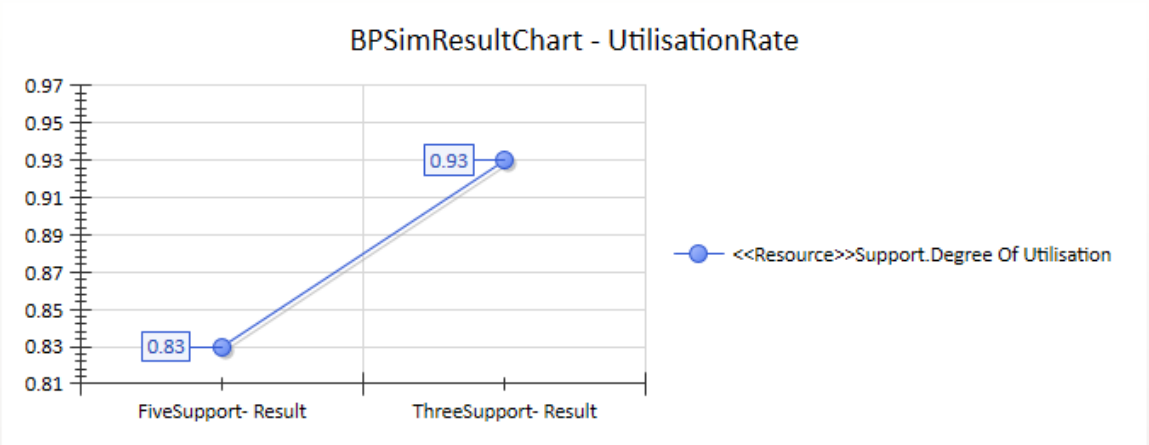
Voici les étapes à suivre pour créer un graphique permettant de comparer le temps moyen d'attente du client pour support :

1. Créez un graphique de résultats BPSim sur le diagramme et nommez- *le Temps d'attente moyen.*
2. Double-cliquez sur le graphique pour ouvrir la fenêtre Propriétés , puis ouvrez l'onglet « Graphique BPSim ».
3. Cliquez sur le bouton ... pour sélectionner un Rapport de base, à partir duquel nous définissons le schéma (légendes) à utiliser dans le graphique. Sélectionnez « TwoSupport-Result ».
4. Choisissez ce schéma :
   - Heure | Service Client | Temps moyen d'attente pour une ressource
5. Cliquez sur le bouton Ajouter pour ajouter deux autres Rapports BPSim : « ThreeSupport-Result » et « FiveSupport-Result ».
6. Cliquez sur le bouton OK et ajustez la taille de l'élément Graphique. Ce graphique nous donne des informations directes.



Voici les étapes à suivre pour créer un graphique permettant de comparer le degré d'utilisation du Support :

1. Créez un graphique de résultats BPSim sur le diagramme et nommez- *le Taux d'utilisation.*
2. Double-cliquez sur le graphique pour ouvrir la fenêtre Propriétés , puis ouvrez l'onglet « Graphique BPSim ».
3. Cliquez sur le bouton ... et sélectionnez un Rapport de base à partir duquel définir le schéma (légendes) à utiliser dans le graphique. Sélectionnez « TwoSupport-Result ».
4. Choisissez ce schéma :
   - Ressource | Support | Degré d'utilisation
5. Cliquez sur le bouton Ajouter pour ajouter deux autres Rapports BPSim : « ThreeSupport-Result » et « FiveSupport-Result ».
6. Cliquez sur le bouton OK et ajustez la taille de l'élément Graphique. Ce graphique fournit des informations spécifiques.


BPSimResultChart - UtilisationRate
0.97
0.95
0.93
0.91
0.89
0.87
0.85
0.83
0.81
0.93
0.83
FiveSupport- Result
ThreeSupport- Result
<<Resource>>Support.Degree Of Utilisation

Exemples de BPSim

Cette section fournit plusieurs exemples de modélisation BPMN, de configuration BPSim et d'analyse des résultats de simulation.

Ces exemples sont tous accessibles à partir du modèle EAExample.

Pour exécuter les simulations d'exemple, vous devez avoir installé le Moteur d'Exécution BPSim et Java Runtime Environment (JRE) 1.7 ou supérieur. Pour travailler avec les paramètres de propriété, vous devez également avoir installé le kit de développement Java (JDK) version 1.7 ou supérieure.

Collaboration pour la commande de repas, version 1

Dans cet exemple, nous créons un modèle très simple pour simuler la communication entre un client et un restaurant pour une commande de repas.

Pour le processus du client :

1. Un client envoie un message au restaurant pour commander un repas.
2. Le client attendra la livraison.
   Si la livraison n'est pas effectuée dans les 60 minutes, ils appelleront le restaurant, puis continueront à attendre.
3. Lors de la livraison, le client dînera.

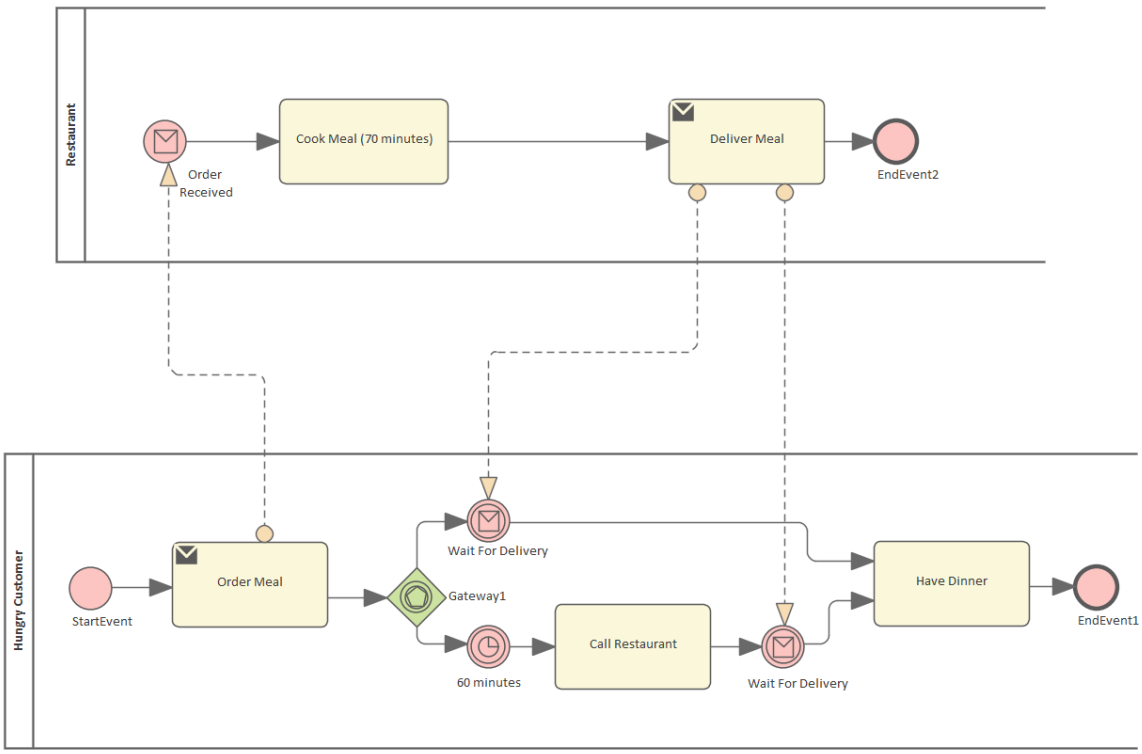
Pour le processus du restaurant :

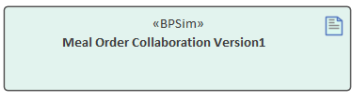
1. Le processus démarre dès la réception d'une commande de repas du client.
2. Le temps de cuisson peut être défini par l'utilisateur. Cela permet d'expérimenter avec différentes durées d'événements, par exemple 30 minutes, 70 minutes
3. Le restaurant livre le repas et termine le processus.

Créer Modèle BPMN

Pour configurer un modèle BPMN pouvant être utilisé pour cette simulation BPSim, vous devez :

- Créer un Modèle de collaboration avec 2 pools
- Dans chaque pool, créez un élément pour chaque processus
- Connectez les éléments avec les flux de messages pour la communication du processus
- Inclure un artefact BPSim pour définir les détails de la simulation.

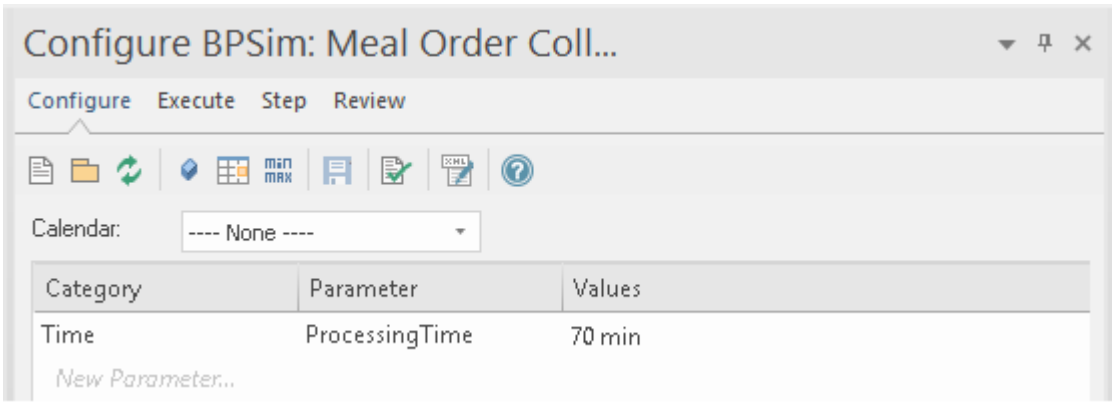




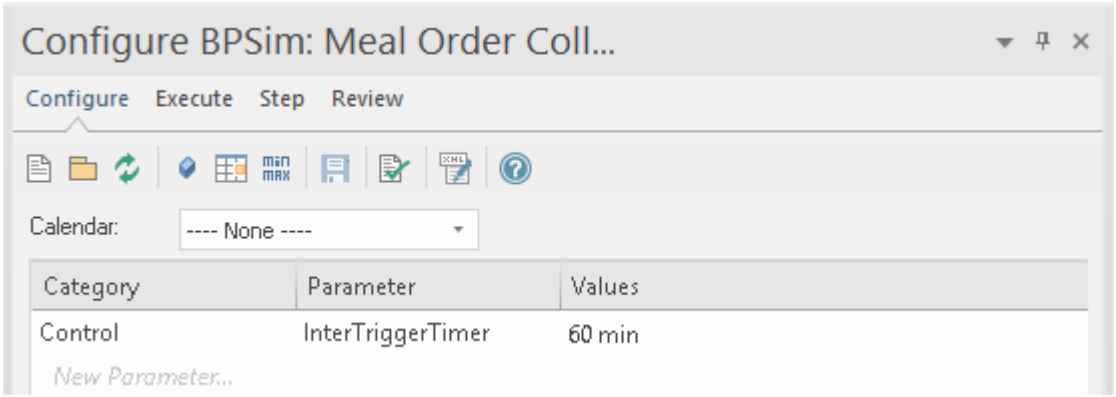
Configurer BPSim

Dans cet exemple, nous configurons les paramètres BPSim suivants :

- Définissez le nombre de déclencheurs de StartEvent dans Hungry Customer sur 1
- Réglez le temps de traitement du repas cuit sur 70 minutes



- Réglez l'InterTriggerTimer de l'événement intermédiaire sur 60 minutes

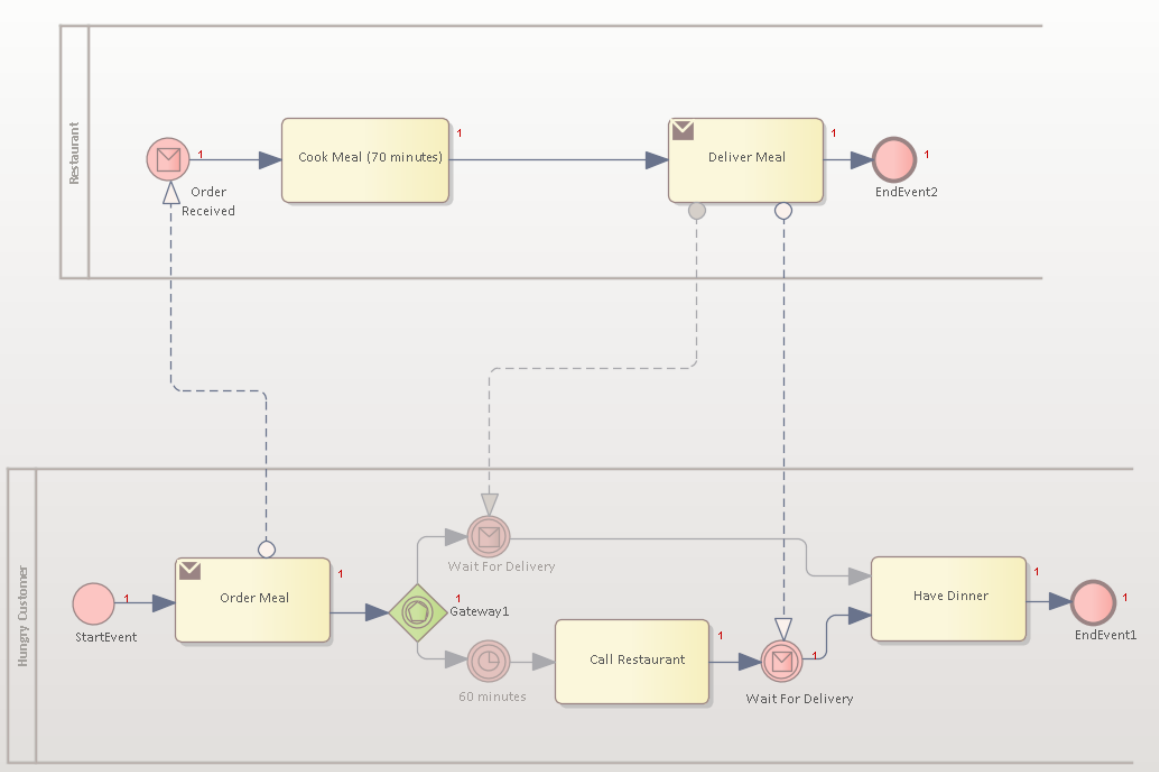


- Paramètres par défaut sur d'autres paramètres BPSim, voici une liste de configurations, vous pouvez consulter via Révision > Résumé de configuration

Simulation

Assurez-vous que la fenêtre Config BPSim est ouverte (Simuler > Analyse de Processus > BPSIM > Ouvrir BPSIM Manager).

Accédez à l'onglet Exécuter et exécuter la Simulation standard :



L'événement exclusif Passerelle a été déclenché par l'événement Minuterie de 60 minutes lorsque la tâche Cuisiner un repas a duré 70 minutes.

Si nous modifions le paramètre BPSim pour la tâche : *Cuisiner un repas* : Temps de traitement de 70 minutes à 30 minutes, la Passerelle d'événement exclusive sera déclenchée par l'événement de message *Attendre la livraison* et la tâche *Appeler le restaurant* ne sera pas du tout activée.

Collaboration pour la commande de repas, version 2

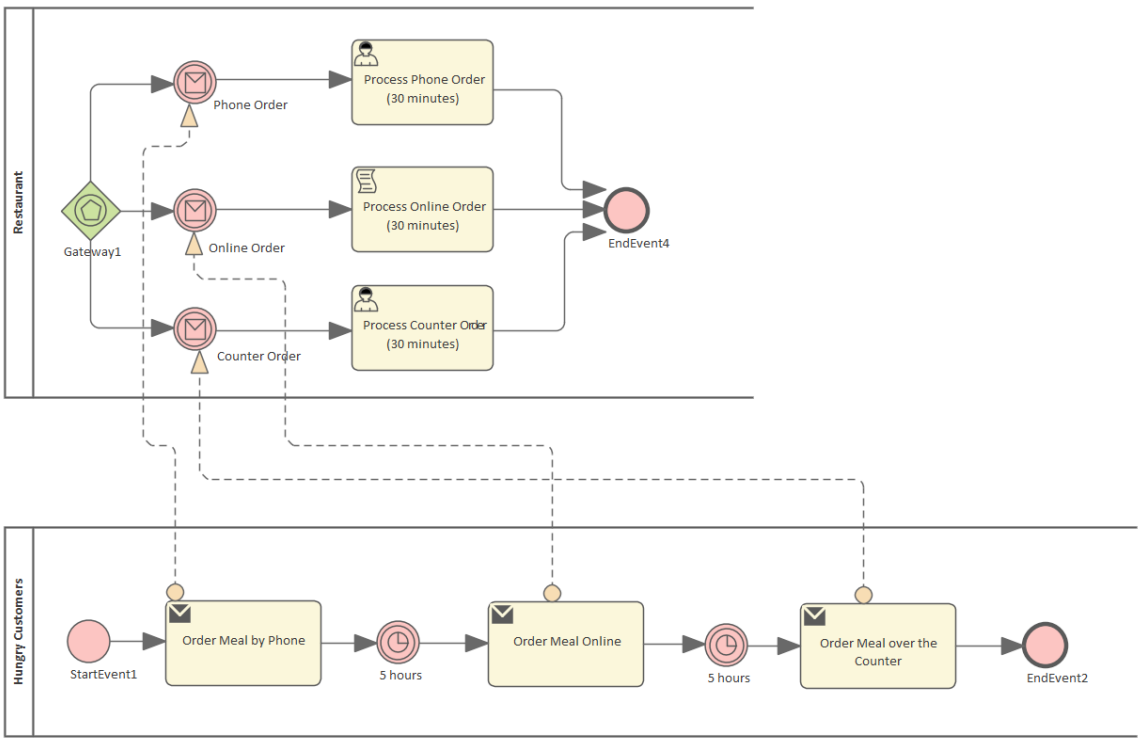
Dans cet exemple, nous créons un modèle pour simuler la communication entre un client et un restaurant pour une commande de repas. Le client passe trois commandes différentes : par téléphone, en ligne et au comptoir.

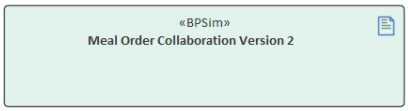
Créer Modèle BPMN

Pour configurer un modèle BPMN pouvant être utilisé pour cette simulation BPSim, vous devez :

- Créer un Modèle de collaboration avec 2 pools
- Dans chaque pool, créez un élément pour chaque processus
- Créer des flux de messages décrivant la communication du processus
- Inclure un artefact BPSim pour définir les détails de la simulation.

note : l'attribut *instantiate* de la passerelle Event-Exclusive a été défini sur **true** , ce qui signifie que le processus a été démarré par ce *Démarrer Event Passerelle* .

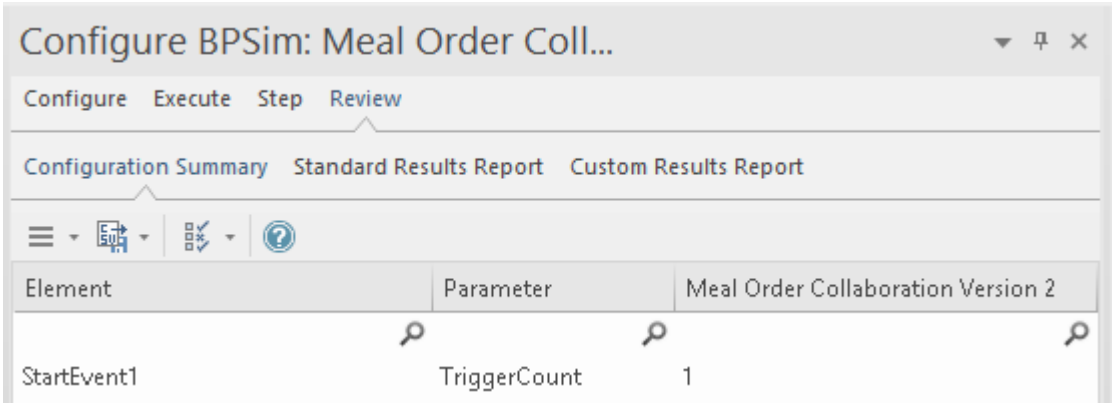




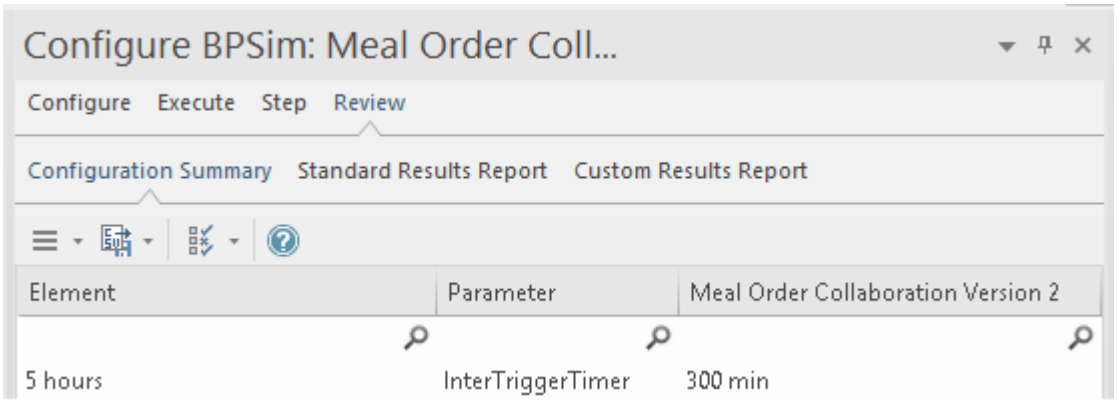
Configurer BPSim

Dans cet exemple, nous configurons les paramètres BPSim suivants :

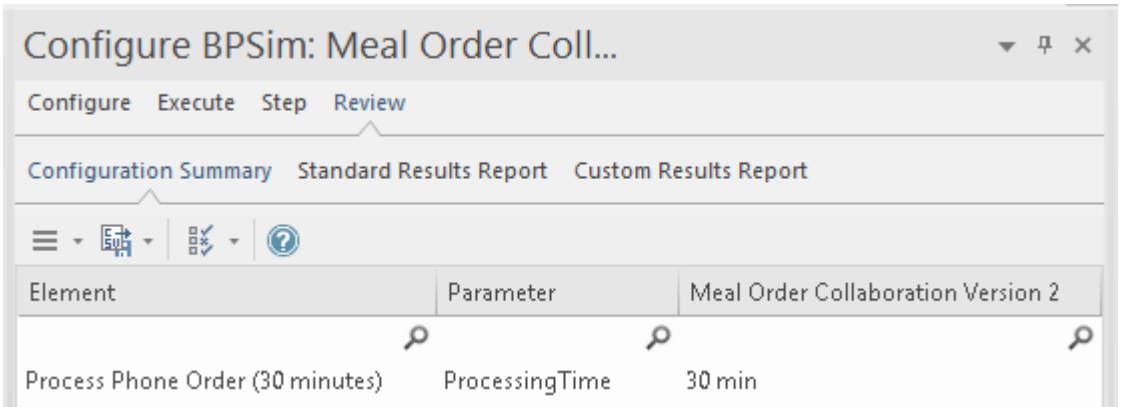
- Pour l'événement Démarrer , définissez le paramètre TriggerCount sur valeur 1



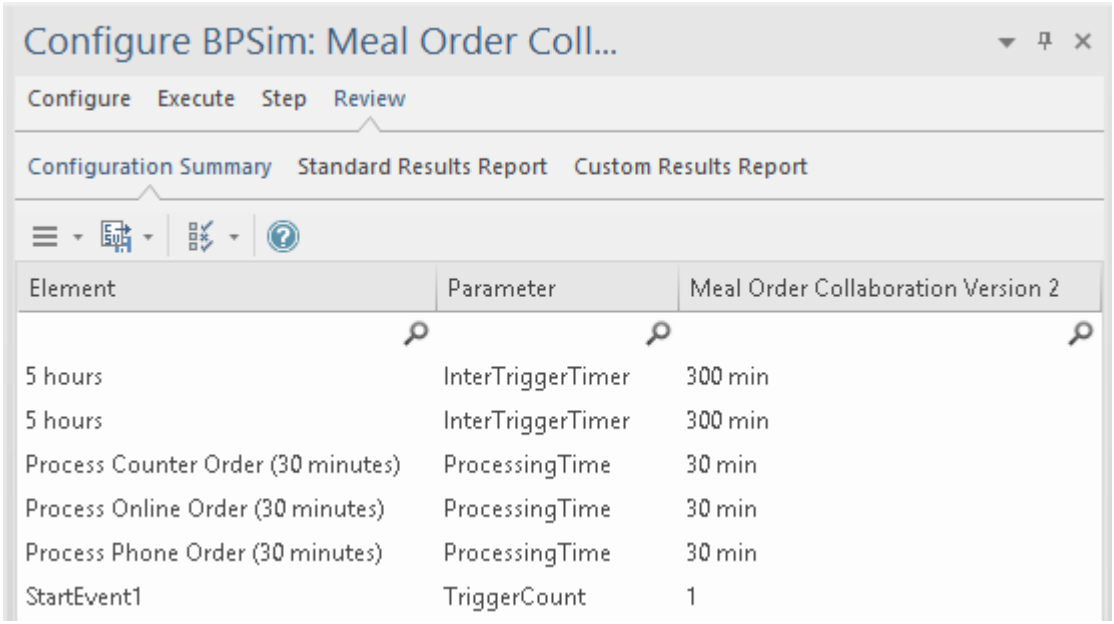
- Pour deux des 5 heures de l'événement intermédiaire, définissez le paramètre InterTriggerTimer sur valeur 300 min



- Pour trois des tâches dans Restaurant, définissez le paramètre ProcessingTime sur valeur 30 min

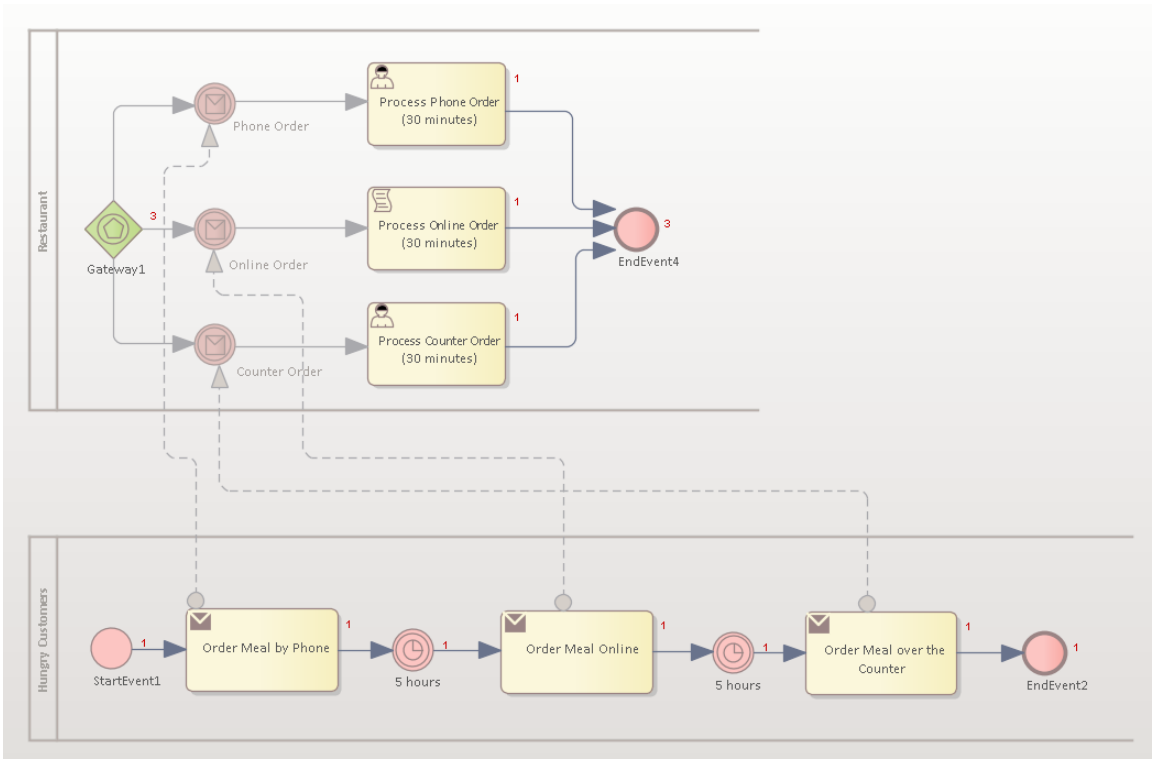


- Paramètres par défaut sur d'autres paramètres BPSim, voici une liste de configurations, vous pouvez consulter via Révision > Résumé de configuration



Simulation

Accédez à l'onglet Exécuter et exécuter la Simulation standard :



Le processus *Restaurant* a été instancié 3 fois par la passerelle Event-Exlusive via 3 messages différents.

Simulation Support téléphonique du service d'assistance

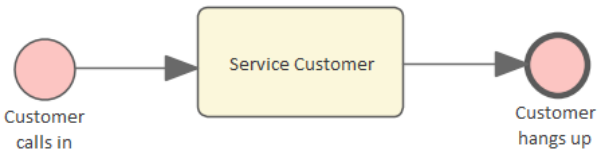
Dans cet exemple, nous créons un modèle très simple pour simuler un processus support téléphonique Help Desk.

Nous mettons en place un scénario dans lequel les ressources sont limitées et les demandes doivent être mises dans une file d'attente pour une ressource. Nous essayons ensuite de trouver un point d'équilibre entre le temps d'attente d'un client et le nombre de ressources, à l'aide d'une analyse de simulation.

Tout d'abord, nous modélisons ce processus étape par étape, en commençant par un simple paramétrage pouvant être calculé avec un stylo et du papier, puis en le vérifiant avec BPSim. Après cela, nous effectuons une analyse de simulation qui pourrait aider le manager à prendre une décision.

Créer Modèle BPMN

Le modèle lui-même est très simple, composé d'un événement Démarrer , d'une tâche et d'un événement de fin.



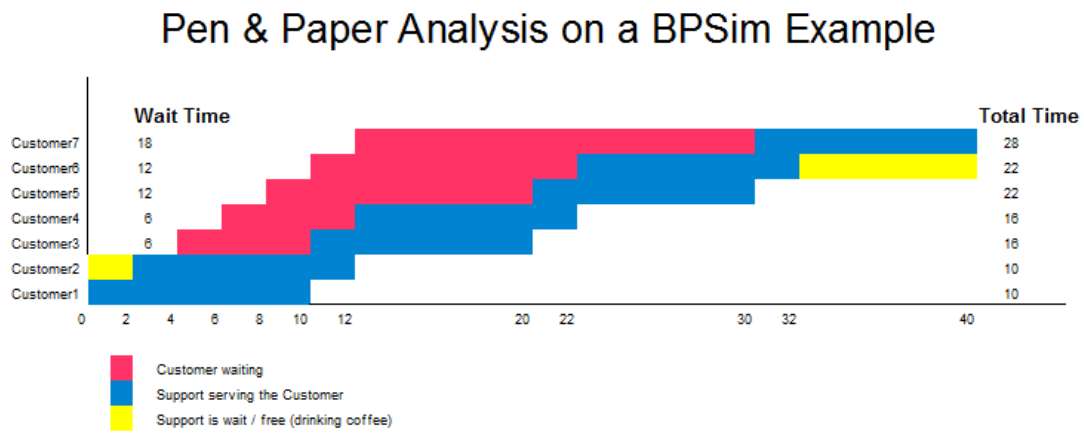
- Créez un événement Démarrer appelé *Appels clients*
- Ajouter une Flux séquence à une tâche abstraite cible appelée *Service Client*
- Ajoutez une Flux séquence à un événement de fin cible appelé *Le client raccroche*

Créez une ressource BPMN2.0 nommée *Support ;* cet élément sera utilisé dans la configuration BPSim.

Analyse du stylo et du papier

Nous utiliserons un stylo et du papier pour analyser ce cas :

- 7 clients appellent à 2 minutes d'intervalle
- 2 ressources support sont disponibles
- Chaque service prendra 10 minutes



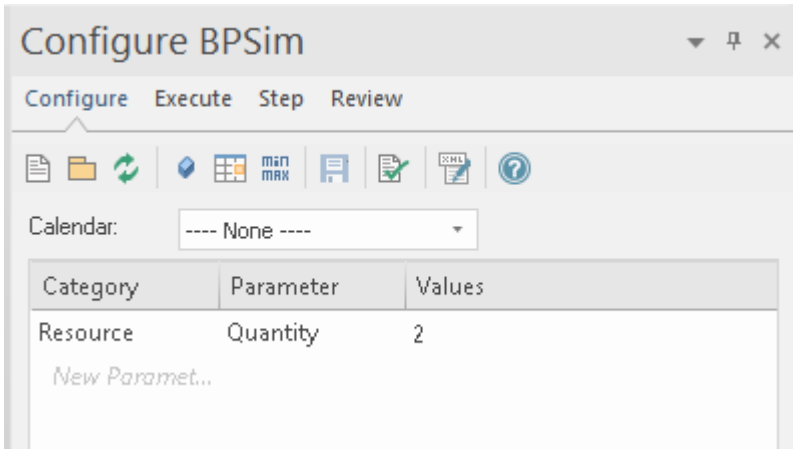
Now we can perform some calculations using a pen and paper based approach:

Total Time Waiting For Resource:	(18+12+12+6+6+0+0) = 54	number of
Total Time In Task:	(28+22+22+16+16+10+10) = 124	
Average Time in Task:	124/7 = 17.71	
Average Time Waiting For Resource:	54/7 = 7.71	
Maximum Time In Task:	28	
Maximum Time Waiting For Resource:	18	
Sum of Support's wait time:	2+8 = 10	number of
Sum of Support's time for Task:	7*10 = 70	number of
Degree of Utilisation:	70/(10+70) * 100% = 87.5%	

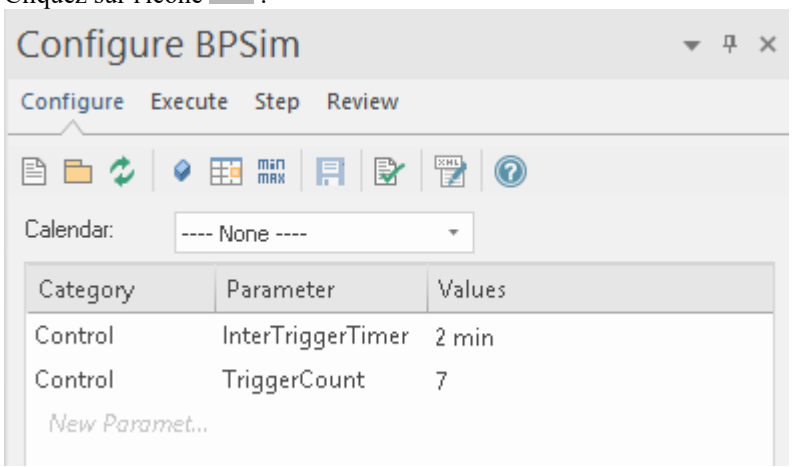
Vu ce résultat, il s'agit déjà d'un calcul très compliqué pour un modèle aussi simple lorsque des contraintes de ressources sont appliquées. Lorsque le processus s'étend et que davantage de contraintes sont appliquées, l'analyser avec un stylo et du papier deviendra rapidement impossible. Nous démontrerons comment BPSim peut vous aider.

Configuration de BPSim

1. Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager').
2. Cliquez sur l'icône et sur le bouton Ajouter nouveau, puis créez un artefact Simulation Processus Métier nommé *Pen & Paper Analysis 7 Customers.*
3. Cliquez sur l'icône et recherchez et sélectionnez le Paquetage contenant le modèle BPMN 2.0 correspondant.
4. Ouvrez le diagramme du modèle et cliquez sur l'élément Ressource appelé « Support ».
5. Dans la colonne « Catégorie » de la fenêtre, cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Ressource », puis cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Quantité », et dans le champ « Valeurs », saisissez « 2 ».
6. Cliquez sur l'icône .

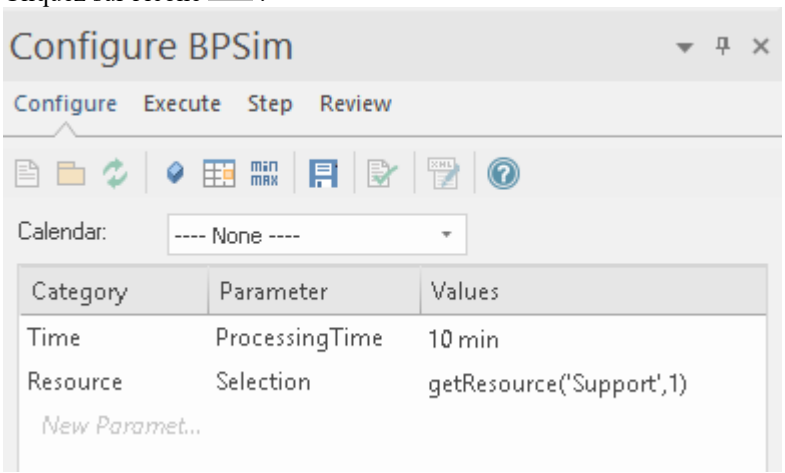


1. Dans le diagramme , cliquez sur l'élément Démarrer Event 'Appel client'.
2. Dans la colonne « Catégorie » de la fenêtre, cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Contrôle ».
3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « InterTriggerTimer ».
4. Dans le champ « Valeur », cliquez sur le bouton ... , sélectionnez l'onglet « Constante » et « Numérique », tapez « 2 » dans le champ « Constante numérique » et sélectionnez « minutes » dans le champ « Unité de temps », puis cliquez sur le bouton OK .
5. Répétez les étapes 2 et 3 en sélectionnant « TriggerCount » dans le champ « Paramètre » et dans le champ « Valeur », saisissez « 7 ».
6. Cliquez sur l'icône .



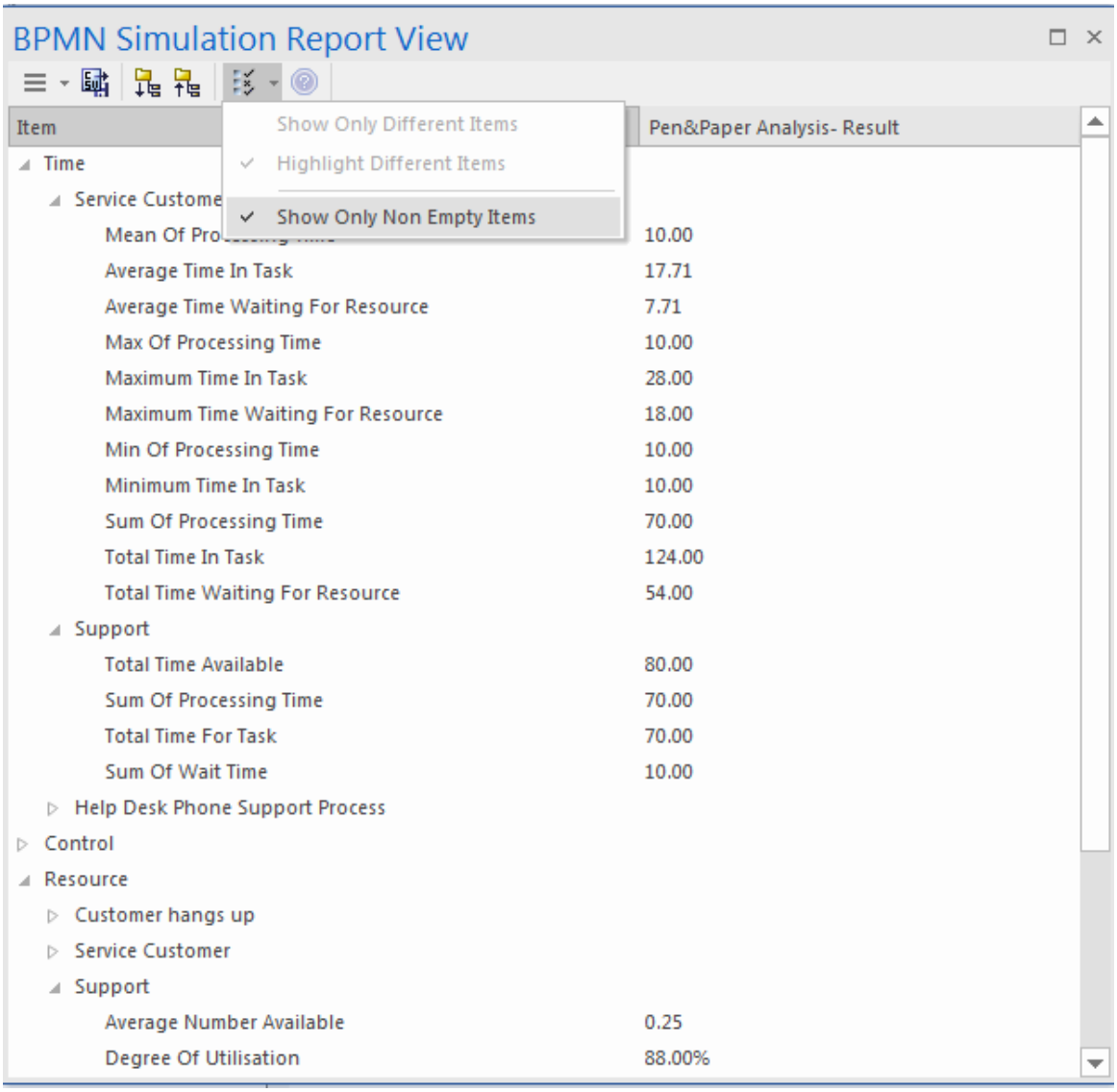
1. Dans le diagramme , cliquez sur l'élément d'activité « Service Client ».
2. Dans la colonne « Catégorie » de la fenêtre, cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Heure ».
3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Temps de traitement ».
4. Dans le champ « Valeur », cliquez sur le bouton ... , sélectionnez l'onglet « Constante » et « Numérique », saisissez « 10 » dans le champ « Constante numérique » et sélectionnez « minutes » dans le champ « Unité de temps », puis cliquez sur le bouton OK .
5. Dans la colonne « Catégorie » de la fenêtre, cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Ressource ».
6. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Sélection ».
7. Dans le champ « Valeur », cliquez sur le bouton ... , sélectionnez « Support » (le nom de l'élément Ressource que vous avez créé dans le modèle) et cliquez sur le bouton Ajouter une sélection par ressource(s) pour déplacer « Support » dans la colonne « Ressource ou rôle ».

8. Dans la colonne « Quantité requise », saisissez 1.
9. Cliquez sur le bouton OK . Dans le champ « Valeurs », l'expression générée automatiquement *bpsim:getResource(' Support ',1)* s'affiche.
10. Cliquez sur l'icône .



Exécuter la Simulation

1. Dans la fenêtre Configurer BPSim, cliquez sur l'onglet « Exécuter » et sur l'icône dans la barre d'outils.
2. Une fois la simulation terminée, cliquez sur l'icône pour ouvrir l'onglet « Résumé de la configuration » de l'onglet « Résultats »



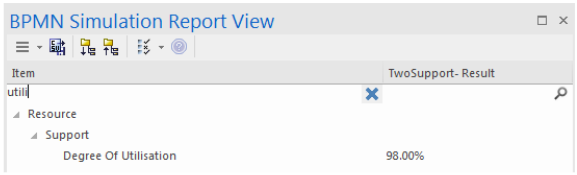
Les résultats correspondent à l'analyse papier-crayon.

Simulation - 2 ressources Support pour 20 clients

Vous pouvez créer un nouvel artefact Simulation Processus Métier en copiant une configuration BPSim existante. Copiez l'élément *clients Pen & Paper Analysis 7* et appuyez sur Ctrl+Shift+V pour le coller, en donnant au nouvel élément le nom *TwoSupport* .

1. Double-cliquez sur *TwoSupport* pour ouvrir la fenêtre Configurer BPSim ; vous pouvez voir que toutes les configurations sont conservées de la source copiée
2. Dans le diagramme , cliquez sur l'élément 'Client appelle dans' Démarrer l'événement.
3. Dans la colonne 'Catégorie' de la fenêtre, cliquez sur le champ 'Valeur' pour 'Contrôle' - ' Déclencheur Count' et changez la valeur en '20'.
4. Cliquez sur l'icône  .

Exécuter la simulation et analyser les résultats



D'après le rapport, vous pouvez voir que :

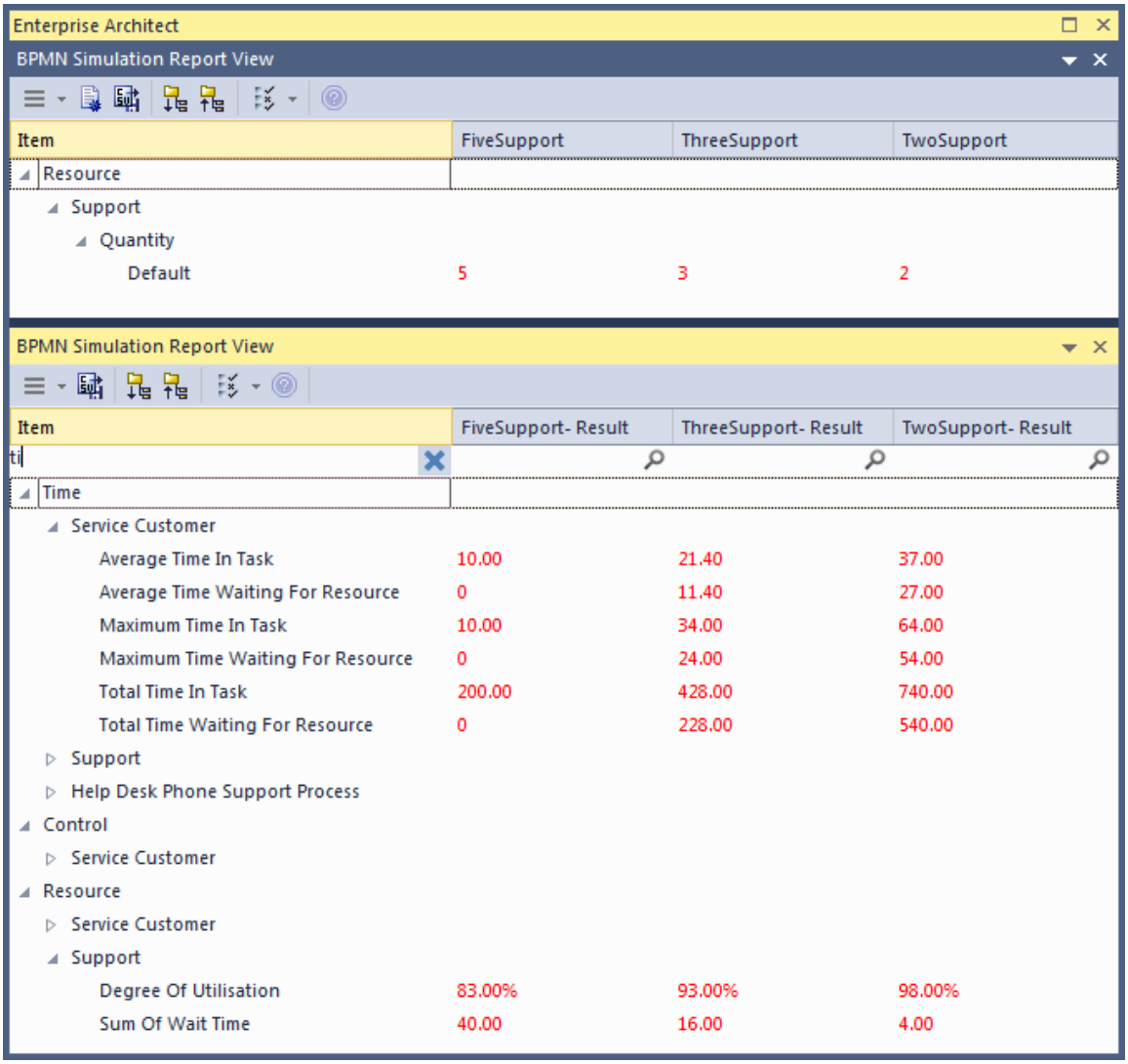
- Le « Temps d'attente moyen pour une ressource » est de 27 minutes et le « Temps d'attente maximum pour une ressource » est de 54 minutes.
- Les deux ressources Support : sont-elles occupées ? S'ils ne l'étaient pas, nous devrons peut-être modifier le processus pour utiliser tout leur temps et réduire le temps d'attente du client ; cependant, le « degré d'utilisation » est de 98 %, ce qui indique que les ressources n'ont eu pratiquement aucun temps d'inactivité.

« Et si » j'avais plus de personnel ? Comparez 2 ressources Support avec 3 et 5 ressources Support

1. Copiez *TwoSupport* et appuyez sur Ctrl+Maj+V pour coller, en donnant au nouvel élément le nom *ThreeSupport.*
2. Double-cliquez sur *ThreeSupport* pour ouvrir la dialogue « Configurer BPSim ».
3. Dans le diagramme , cliquez sur l'élément Ressource « Support ».
4. Dans la fenêtre Configurer BPSim, dans le champ « Valeurs » pour « Ressource » - « Quantité », saisissez « 3 ».
1. Copiez à nouveau *TwoSupport* et appuyez sur Ctrl+Maj+V pour coller, en donnant à ce nouvel élément le nom *FiveSupport.*
2. Double-cliquez sur *FiveSupport* pour ouvrir la dialogue « Configurer BPSim ».
3. Dans le diagramme , cliquez sur l'élément Ressource « Support ».
4. Dans la fenêtre Configurer BPSim, dans le champ « Valeurs » pour « Ressource » - « Quantité », saisissez « 5 ».

Exécuter les simulations et faire une comparaison ; dans la fenêtre Navigateur :

1. Ctrl+clic sur *TwoSupport* , *ThreeSupport* et *FiveSupport* , puis cliquez-droit et sélectionnez l'option 'Afficher la configuration BPSim'.
2. Ctrl+clic sur *TwoSupport-Result* , *ThreeSupport-Result* et *FiveSupport-Result* , puis cliquez-droit et sélectionnez l'option 'Afficher Rapport BPSim '.



Conseils :

- Cliquez sur le bouton  et sur l'option « Afficher uniquement Items différents » pour les deux vues
- Vous pouvez ancrer les vues ensemble, afin qu'elles fournissent des comparaisons directes : CE sont les différences de résultats causées par CES différences de configuration
- Basculez la barre de filtre pour filtrer les éléments qui vous intéressent

Analyse

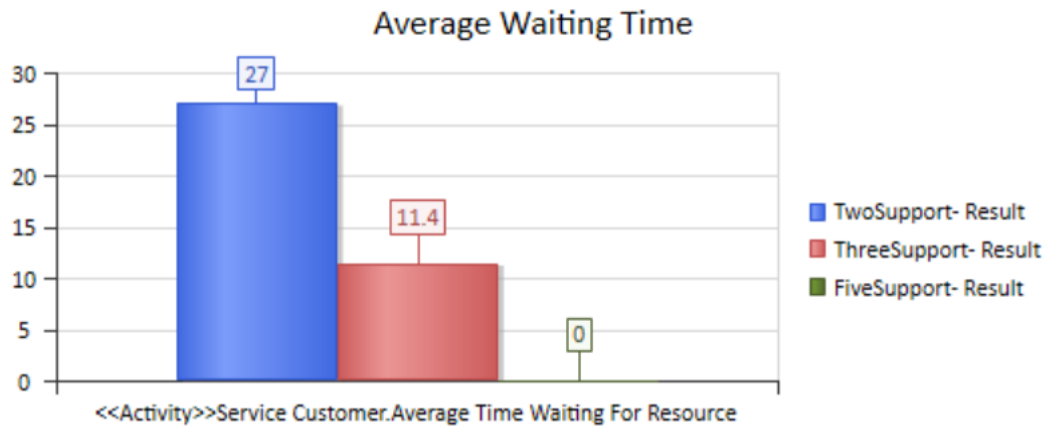
Les vues de comparaison ancrées montrent les différences de configuration et les différences de résultats correspondantes.

- Le temps d'attente du client est passé de 27 minutes (2 ressources Support) à 11,4 minutes (3 ressources Support) et encore plus bas à 0 minute (5 ressources Support)
- Le « degré d'utilisation » est passé de 98 % (2 ressources Support) à 93 % (3 ressources Support) et encore plus bas à 83 % (5 ressources Support)

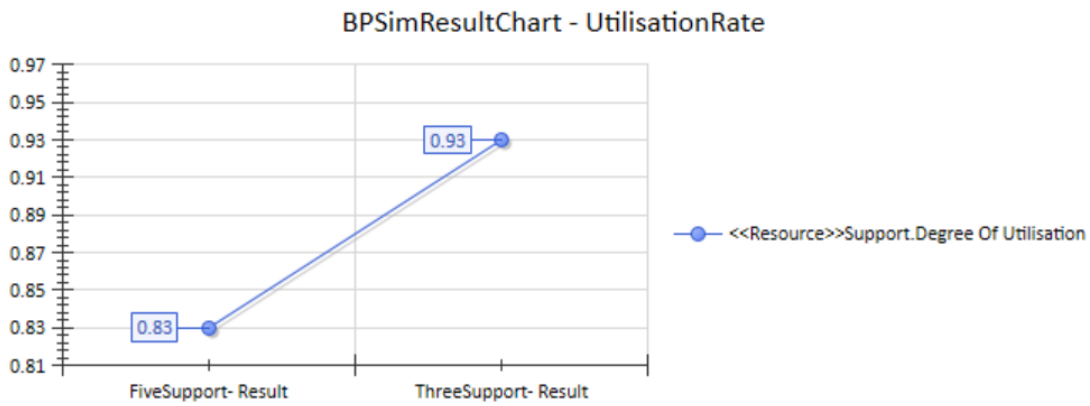
Les clients seront probablement satisfaits avec 5 ressources Support ; cependant, le coût peut être hors budget. Ainsi, 3 ou éventuellement 4 ressources Support peuvent constituer un point d'équilibre dans ce cas. Essayez de copier l'un des artefacts Simulation Processus Métier et de configurer et exécuter une simulation pour 4 ressources Support .

Afficher le résultat avec un graphique

1. Faites glisser une icône « Graphique de résultats BPSim » de la boîte à outils sur le diagramme et créez un artefact de graphique de résultats BPSim ; appelez- *le Temps d'attente moyen.*
2. Cliquez-droit sur l'Artefact et sélectionnez l'option ' Propriétés ' pour afficher la dialogue de l'élément ' Propriétés ' ; cliquez sur la page 'BPSim Chart'.
3. Cliquez sur le bouton ... et sélectionnez un Rapport de base à partir duquel définir le schéma (légendes) à utiliser dans le graphique ; sélectionnez *TwoSupport-Result.*
4. Choisissez le schéma « Heure » | « Service Client » | « Temps moyen d'attente de la ressource ».
5. Cliquez sur le bouton Ajouter pour ajouter deux autres Rapports BPSim : *ThreeSupport-Result* et *FiveSupport-Result*
6. Cliquez sur le bouton OK et ajustez la taille de l'élément Graphique ; ce graphique nous a donné des informations très simples



1. Créez un autre artefact de graphique de résultats BPSim sur le diagramme , appelé *taux d'utilisation.*
2. Double-cliquez sur l'Artefact pour afficher la dialogue « Propriétés » de l'élément et cliquez sur l'onglet « Graphique BPSim ».
3. Cliquez sur le bouton ... et sélectionnez un Rapport de base à partir duquel définir le schéma (légendes) à utiliser dans le graphique ; sélectionnez *TwoSupport-Result.*
4. Choisissez le schéma « Ressource » | « Support » | « Degré d'utilisation ».
5. Cliquez sur le bouton Ajouter pour ajouter deux autres Rapports BPSim : *ThreeSupport-Result* et *FiveSupport-Result.*
6. Cliquez sur le bouton OK et ajustez la taille de l'élément Graphique.


BPSimResultChart - UtilisationRate
0.97
0.95
0.93
0.91
0.89
0.87
0.85
0.83
0.81
0.93
0.83
FiveSupport- Result
ThreeSupport- Result
<<Resource>>Support.Degree Of Utilisation

Simulation Support téléphonique basée sur un calendrier

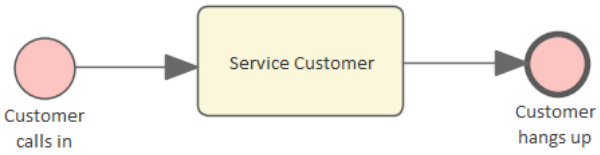
Dans cet exemple, nous créons un modèle très simple pour simuler un processus support téléphonique du Help Desk, basé sur les paramètres du calendrier. Nous supposons que :

- Les clients appellent à des intervalles différents en semaine et le week-end
- Les délais de traitement diffèrent entre les jours de semaine et les week-ends
- Il existe différents nombres de ressources support en semaine et le week-end

Nous modélisons ce processus étape par étape, puis créons des calendriers et configurons la simulation Processus Métier , qui est suffisamment simple pour être calculée à la main. Après cela, nous exécuter la simulation pour comparer ce résultat avec l'analyse papier.

Créer Modèle BPMN

Le modèle lui-même est très simple, composé d'un événement Démarrer , d'une Tâche et d'un événement Fin.

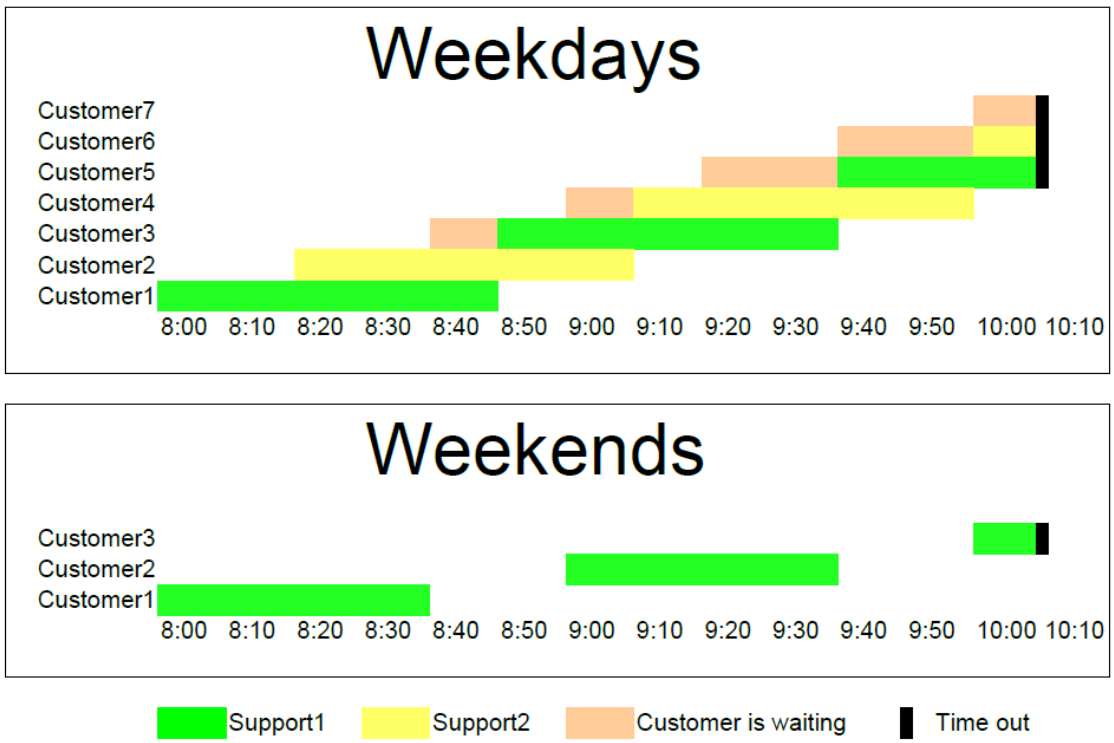


1. Créer un événement Démarrer *Le client appelle.*
2. Ajoutez une Flux séquence à la tâche abstraite cible Activity *Service Customer.*
3. Ajoutez une Flux séquence à l'événement final cible . *Le client raccroche.*
4. Créez une ressource BPMN2.0 nommée *Support* .
5. Créez un BPMN2.0::ResourceRole dans *Service Customer* , donnez-lui le nom *support* et définissez l' étiquette *resourceRef* sur le nom de l'élément Resource *Support* .

Analyse papier-crayon

Nous pouvons utiliser un stylo et du papier pour analyser ce cas :

- La durée de la simulation est de 2 heures et 10 minutes, de 8h00 à 10h10
- Un client appelle toutes les 20 minutes en semaine
- Un client appelle toutes les 60 minutes le week-end
- Il faut 50 minutes pour servir chaque client en semaine
- Il faut 40 minutes pour servir chaque client le week-end
- Il y a 2 ressources support en semaine
- Il y a 1 ressource support le week-end



En regardant ce résultat, lorsque des contraintes de ressources sont appliquées, le calcul est assez compliqué pour un modèle aussi simple.

En semaine

- 7 clients ont appelé à des intervalles de 20 minutes sur une durée de 2 heures et 10 minutes
- 4 appels clients ont été terminés normalement
- 2 appels clients ont été interrompus en raison d'un dépassement de délai
- 1 appel client n'a pas été répondu
- Support1 a fonctionné en continu pendant 130 minutes, Support2 a fonctionné en continu pendant 110 minutes

Le week-end

- 3 clients ont appelé à des intervalles de 60 minutes sur une durée de 2 heures et 10 minutes
- 2 appels clients ont été terminés normalement
- 1 appel client a été interrompu en raison d'un dépassement de délai
- Support1 a travaillé 90 minutes, par blocs de 40 minutes avec un intervalle de 20 minutes entre les appels

Nous allons maintenant voir comment BPSim peut vous aider.

Configuration de BPSim

Dans cette section, nous créons d'abord les Calendriers, puis nous paramétrons les Durée et Démarrer .

Pour les paramètres d'élément, vous pouvez spécifier un ou plusieurs calendriers pour un paramètre donné. Cependant, **si un calendrier est défini pour un paramètre valeur , une valeur par défaut (sans calendrier spécifié) doit exister** , sinon la simulation ne fonctionnera pas.

Cliquer sur le bouton  dans la barre d'outils de la fenêtre Configurer BPSim vérifiera automatiquement cette contrainte pour vous.

Tâche	Action
Créer un artefact BPSim et définir Paquetage	1. Ouvrez la fenêtre Configurer BPSim ('Simulate > Analyse de Processus > BPSim > Open BPSim Manager'). 2. Créez un artefact Simulation Processus Métier nommé *Simulation de processus Support basé sur le calendrier* . 3. Sélectionnez le Paquetage contenant le modèle BPMN 2.0 correspondant. 4. Ouvrez le diagramme contenant le modèle à simuler.
Calendriers	1. Dans l'onglet « Configurer » de la fenêtre Configurer BPSim, cliquez sur l'icône  dans la barre d'outils. La dialogue « Modifier les calendriers BPSim » s'affiche. 2. Cliquez sur le bouton Nouveau pour afficher la dialogue « Récurrence de l'événement » et remplissez les champs comme décrit ici, pour créer un calendrier. (Vous allez créer deux calendriers.) 3. Dans le panneau « Heure de l'événement », définissez « Démarrer » sur 08h00 et « Fin » sur 17h00. 4. Dans le panneau « Motif de récurrence », sélectionnez « Hebdomadaire » et cochez les cases allant de « lundi » à « vendredi ». 5. Dans le panneau « Plage de récurrence », définissez « Démarrer » sur « 02/11/2020 » et sélectionnez « Pas de date de fin ». 6. Cliquez sur le bouton OK . Vous êtes invité à saisir un nom de calendrier ; remplacez *Calendar_1* par « Weekdays » et cliquez sur le bouton OK . 7. Cliquez à nouveau sur le bouton Nouveau et répétez les étapes 3 à 6 avec ces valeurs : - ' Démarrer ' - 08:00 - 'Fin' - 17h00 - 'Hebdomadaire' - 'Samedi' et 'Dimanche' - ' Démarrer ' à '07/11/2020' et 'Pas de date de fin' - Remplacez *Calendar_2* par « Week-end » 8. Cliquez sur le bouton OK .
Durée	Sur le diagramme , cliquez sur l'artefact BPSim *Simulation de processus Support basé sur le calendrier* et, dans l'onglet « Configurer » de la fenêtre Configurer BPSim, avec le champ « Calendrier » défini sur « ----Aucun---- », créez ou modifiez ce paramètre de scénario : • Durée - avec une valeur constante de 0000 002:10:00, ce qui signifie 0 jour, 2 heures et 10 minutes
Arrivée des clients	Sur le diagramme , cliquez sur les *appels clients dans* StartEvent et, dans l'onglet « Configurer » de la fenêtre Configurer BPSim, créez ou modifiez ce paramètre de contrôle : • InterTriggerTimer - Valeur : 0 00:00:00, avec le champ « Calendrier » défini sur « ----Aucun---- » (cette valeur par défaut est nécessaire) • InterTriggerTimer - Valeur : 0 00:20:00, avec le champ « Calendrier » défini sur « Jours de la semaine » • InterTriggerTimer - Valeur : 0 01:00:00, avec le champ « Calendrier » défini sur « Week-ends »
Délais de traitement	Sur le diagramme , cliquez sur l'activité *Service Client* et, dans l'onglet « Configurer » de la fenêtre Configurer BPSim, créez ou modifiez ce paramètre Temps :

	• ProcessingTime - Valeur : 0 00:00:00, avec le champ « Calendrier » défini sur « ----Aucun---- » (cette valeur par défaut est nécessaire) • ProcessingTime - Valeur : 0 00:50:00, avec le champ « Calendrier » défini sur « Jours de la semaine » • ProcessingTime - Valeur : 0 00:40:00, avec le champ « Calendrier » défini sur « Week-ends »
Ressources	Sur le diagramme , cliquez sur la ressource *Support* et, dans l'onglet « Configurer » de la fenêtre Configurer BPSim, créez ou modifiez ce paramètre de ressource • Quantité - Valeur : 0 ; Calendrier, avec le champ « Calendrier » défini sur « ----Aucun---- » (cette valeur par défaut est nécessaire) • Quantité - Valeur : 2 ; Calendrier, avec le champ « Calendrier » défini sur « Jours de la semaine » • Quantité - Valeur : 1 ; Calendrier, avec le champ « Calendrier » défini sur « Week-ends »
Sélection (allocation) des ressources	Sur le diagramme , cliquez sur l'activité *Service Client* et, dans l'onglet « Configurer » de la fenêtre Configurer BPSim, avec le champ Calendrier défini sur « ----Aucun---- », vérifiez que le champ « Valeurs » pour le paramètre de ressource « Sélection » est défini sur : **bpsim::getResource(' Support ',1)** comme expression Cette expression est chargée par défaut à partir de votre modèle BPMN. Vous pouvez effectuer certaines configurations avancées pour la sélection des ressources pour une tâche.

Exécuter Simulation

Jours de la semaine

1. Cliquez sur le champ « Calendrier » et sélectionnez « Jour de la semaine ».
2. Cliquez sur l'onglet « Exécuter » et sur l'icône de la barre d'outils  .

Un fichier nommé *Simulation de processus Support basée sur le calendrier - Résultat* est généré. Ce fichier de rapport contient le résultat d'une simulation de jour de semaine, qui est affiché dans l'onglet « Révision » de la fenêtre Configurer BPSim, dans l'onglet « Rapport de résultats standard ».

Les week-ends

1. Cliquez sur le champ « Calendrier » et sélectionnez « Week-end ».
2. Cliquez sur l'onglet « Exécuter » et sur l'icône de la barre d'outils  .

Le fichier *de résultats de Simulation du processus Support basé sur le calendrier* est mis à jour pour afficher le résultat d'une simulation de week-end et affiché dans l'onglet « Révision » de la fenêtre Configurer BPSim, dans l'onglet « Rapport de résultats standard ».

Dans chaque cas, vérifiez la correspondance entre le fichier de résultats et notre analyse avec un stylo et du papier.

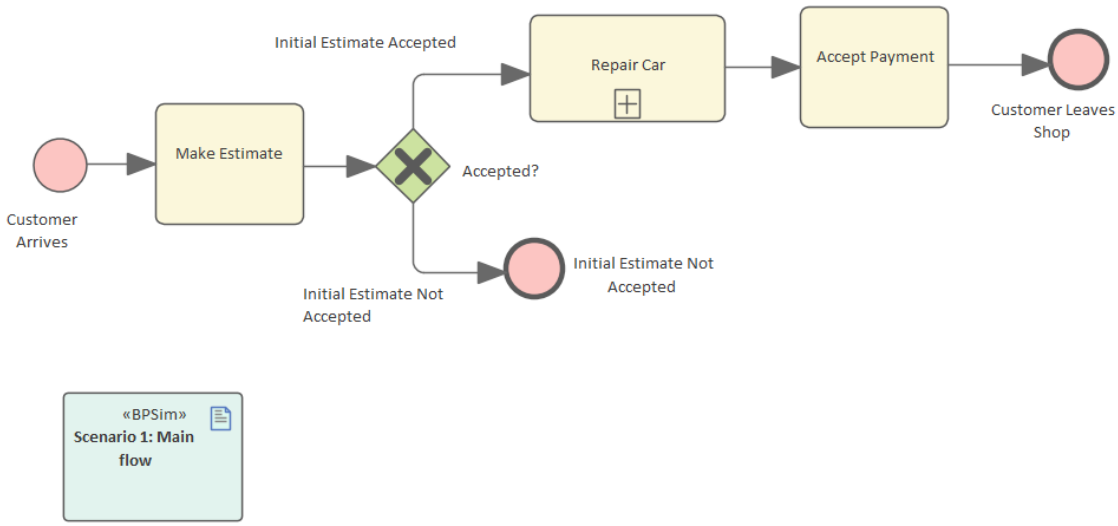
Processus de réparation automobile

Cet exemple simule le flux de processus d'un atelier de réparation automobile. La configuration de BPSim :

- Utilise un paramètre de propriété initialisé par la distribution pour générer un nombre aléatoire de problèmes pour chaque client
- Applique la probabilité pour simuler :
  - Acceptation ou non du devis initial
  - Si de nouveaux problèmes seront détectés lors de la réparation
- Incrémente ou décrémente la valeur du paramètre de propriété dans chaque tâche
- Utilise la valeur du paramètre de propriété sur les conditions des séquences sortant des passerelles
- Simule les arrivées des clients pour un début et une durée donnés

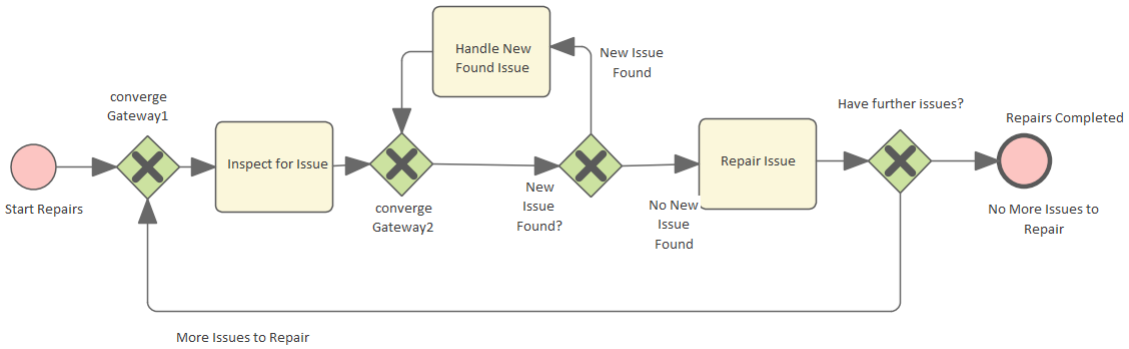
Créer Modèle BPMN

Créer le processus principal



1. Créez un événement Démarrer *Le client arrive.*
2. Ajoutez une Flux séquence à une tâche abstraite cible Activité *Créer une estimation.*
3. Ajouter une Flux séquence à une cible Passerelle Exclusive *Acceptée ?.*
4. Ajoutez des flux Séquence à :
   - Une cible Événement final *Estimation initiale non acceptée*
   - Un sous-processus cible *Réparation de voiture*
5. À partir de *Réparation de voiture* , ajoutez une Flux séquence à une tâche abstraite cible Activité *Accepter le paiement.*
6. Ajoutez une Flux séquence à un événement final cible *Le client quitte la boutique.*

Créer le sous-processus Réparation de voiture



1. Créer un événement Démarrer *Démarrer les réparations.*
2. Ajoutez une Flux séquence à une Passerelle exclusive cible *convergente Gateway1.*
3. Ajoutez une Flux séquence à une tâche abstraite. Activité *Inspecter pour détecter un problème.*
4. Ajoutez une Flux séquence à une Passerelle exclusive Passerelle *converge2.*
5. Ajoutez une Flux séquence à un *nouveau numéro* exclusif Passerelle . Vous l'avez trouvé ?
6. Ajoutez des flux Séquence à :
   - Une tâche abstraite cible Activité *Gérer un nouveau problème détecté* , puis ajouter un Flux séquence retour pour *converger vers Gateway2*
   - Une tâche abstraite cible *Réparer le problème* , puis ajouter une Flux séquence à un cible Passerelle exclusive *Vous avez d'autres problèmes ?*
7. Depuis la Passerelle *Vous avez d'autres problèmes ?* Ajoutez Séquence Flows à :
   - L'événement final cible *Réparations terminées*
   - *converger Gateway1*

Configurer BPSim

Tâche	Description
Artefact et Paquetage	1. Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager'). 2. Créez un artefact Simulation Processus Métier nommé *Scénario 1 : Flux principal.* 3. Sélectionnez le Paquetage contenant le modèle BPMN 2.0 correspondant.
Démarrer et Durée	Nous allons simuler les processus dans un garage dont les horaires d'ouverture sont de 9h00 à 17h00, soit une période de 8 heures. Nous supposons également qu'un client entrant après 16h50 ne sera pas servi ce jour-là. Par conséquent, l'heure Démarrer de la simulation est 9h00 et la durée est de 7 heures et 50 minutes. Sur le diagramme « Réparation automobile », cliquez sur l'artefact Simulation Processus Métier nommé *Scénario 1 : Flux principal* et, dans la fenêtre Configurer BPSim, mettez à jour ces paramètres de scénario : • Démarrer - remplacez le champ « Valeurs » par n'importe quelle date (au format jj/mm/aaaa) et remplacez la section horaire par « 9 h 00 » • Durée - cliquez sur le bouton ... dans le champ « Valeurs » et définissez-le sur une durée constante de « 0 07:50:00 »
Le client arrive	Nous allons simuler un client arrivant toutes les 24 minutes.

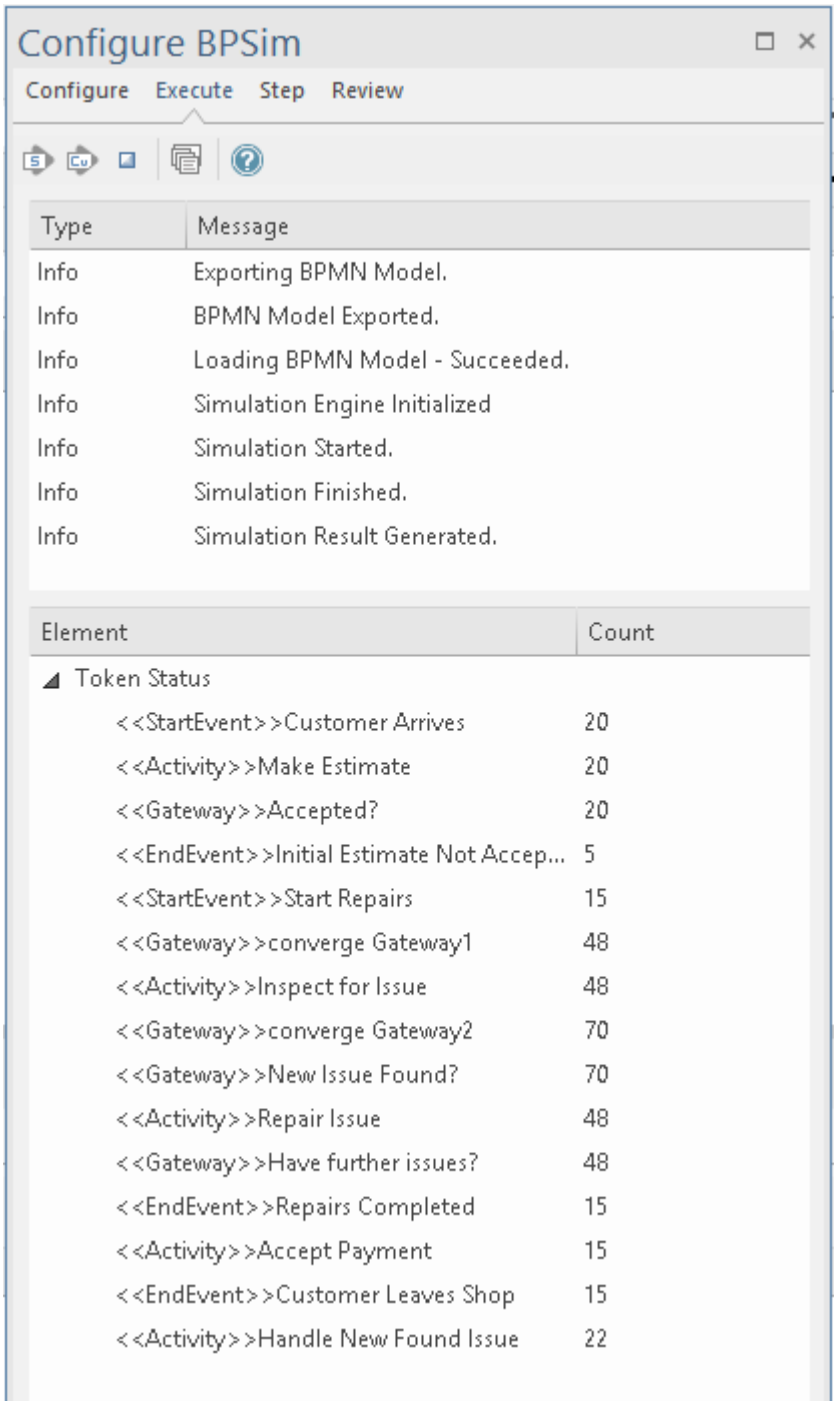
	Le premier client arrive à 9h00 et le dernier arrive à 16h36 (le client arrivant à 17h00 ne sera pas servi aujourd'hui car cela est limité par le paramètre « Durée »). Avec un stylo et du papier, nous pouvons calculer qu'il y a 20 clients servis (9h00 à 16h36 = 456 minutes ; le nombre de clients est 456/24 + 1 = 19 + 1 = 20). Nous vérifierons cela avec le résultat de la simulation plus tard. Sur le diagramme 'Réparation automobile', cliquez sur l'élément Démarrer Event *Arrivée du client* , et dans la fenêtre Configurer BPSim : 1. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Contrôle ». 2. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « InterTriggerTimer ». 3. Dans le champ « Valeurs », cliquez sur le bouton  et définissez une valeur numérique constante de « 24 minutes ». Cliquez sur le bouton OK et sur l'icône Enregistrer de la barre d'outils.
Paramètres de la propriété	Nous supposons que le véhicule de chaque client peut initialement présenter un nombre différent de problèmes. Cela pourrait être reflété à l'aide d'un générateur de nombres aléatoires. BPSim fournit un certain nombre de distributions adaptées à vos besoins. Dans cet exemple, nous utilisons une distribution normale tronquée pour initialiser la propriété *noOfIssues.* Les tâches *Réparer le problème* et *Gérer le nouveau problème détecté* décrémenteront et incrémenteront respectivement la valeur de *noOfIssues* . 1. Sur le diagramme 'Réparation auto', cliquez sur l'événement Démarrer *Arrivée client* . 2. Dans l'onglet « Configurer » de la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de propriété appelé *noOfIssues.* 3. Dans le champ « Valeurs », cliquez sur le bouton  ; la dialogue « Configurer « noOfIssues » pour « CustomerArrives » s'affiche. 4. Cliquez sur l'onglet « Distribution » et sélectionnez « TruncatedNormal » ; dans les champs : - 'Moyenne', tapez '2' - « Écart type », tapez « 1 » - 'Min', tapez '1' - 'Max', tapez '1000' **Note importante :** les distributions telles que « TruncatedNormal » renvoient une valeur à virgule flottante, mais la propriété est utilisée comme un integer . La définition du type de la propriété est importante, en particulier dans les expressions de condition lors des tests d'égalité. Par exemple, l'expression de condition *getProperty('noOfIssues') = 0* ne sera presque jamais satisfaite, car *noOfIssues* a été initialisé par une distribution à virgule flottante. **Conseil : Comment personnaliser le type d'un bien** Après avoir créé la propriété et défini une valeur , cliquez sur l'icône  dans la barre d'outils, puis cliquez sur l'icône  pour afficher la dialogue « Modifier les paramètres de la propriété ». Dans le champ « Type » de la propriété, cliquez sur la flèche déroulante et sélectionnez la valeur « int » au lieu de la valeur par défaut « double ». 1. Sur le diagramme « Réparer la voiture », cliquez sur le *problème de réparation de l'activité.* 2. Dans l'onglet « Configurer » de la fenêtre Configurer BPSim, cliquez sur la

	flèche déroulante *Nouveau paramètre* et créez un paramètre de propriété appelé *noOfIssues.* 3. Dans le champ « Valeurs », cliquez sur le bouton ... . La dialogue « Configurer « noOfIssues » pour « Réparer le problème » s'affiche. 4. Cliquez sur l'onglet 'Expression' et, dans le champ 'Expression', tapez *{noOfIssues} -1* ; cliquez sur le bouton OK . 5. Sur le diagramme « Réparer la voiture », cliquez sur l'activité *Gérer les nouveaux problèmes détectés* . 6. Dans l'onglet « Configurer » de la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de propriété appelé *noOfIssues.* 7. Dans le champ « Valeurs », cliquez sur le bouton ... . La dialogue « Configurer « noOfIssues » pour « Gérer les nouveaux problèmes détectés » s'affiche. 8. Cliquez sur l'onglet 'Expression' et, dans le champ 'Expression', tapez *{noOfIssues} +1* ; cliquez sur le bouton OK .
Probabilité sur les flux Séquence	Nous estimons qu'un client sur trois n'acceptera pas le devis initial de réparation et que les deux autres l'accepteront. Nous estimons également que pour une réparation sur quatre, de nouveaux problèmes seront détectés et que pour les trois réparations restantes, aucun nouveau problème ne sera détecté. Sur le diagramme 'Réparation automobile', référez-vous à l'élément Passerelle *Accepté ?* . Clique sur le: • Flux séquence *estimation initiale acceptée* et dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Probabilité » ; dans le champ 'Valeurs', tapez '0.67' • Flux séquence *d'estimation initiale non acceptée* , et dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Probabilité » ; dans le champ 'Valeurs', tapez '0.33' Sur le diagramme 'Réparation automobile', référez-vous à l'élément Passerelle *Nouveau problème trouvé ?* . Clique sur le: • *Plus de problèmes pour réparer* Flux séquence et dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Probabilité » ; dans le champ 'Valeurs', tapez '0,75' • *Plus de problèmes à réparer dans* Flux séquence et dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Probabilité » ; dans le champ 'Valeurs', tapez '0,25'
Condition sur les flux Séquence	Nous utilisons une expression pour renvoyer une valeur booléenne comme condition d'une Flux séquence , qui joue un rôle clé dans la logique du flux. Sur le diagramme « Réparer la voiture », reportez-vous à la *section Avez-vous d'autres problèmes ?* Élément Passerelle . Clique sur le: • *Plus de problèmes à réparer dans* Flux séquence et dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Condition » ; dans le champ 'Valeurs' cliquez sur le bouton ... , cliquez sur l'onglet 'Expression' et tapez **{noOfIssues} != 0** dans le champ 'Expression' • *Plus de problèmes pour réparer* Flux séquence et dans la fenêtre Configurer

	BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « Condition » ; dans le champ 'Valeurs' cliquez sur le bouton  , cliquez sur l'onglet 'Expression' et tapez **{noOfIssues} = 0** dans le champ 'Expression' Note : Toutes les transitions sortantes d'une Passerelle doivent inclure 100% de la logique ; par exemple, vous ne devez pas saisir **{noOfIssues} > 10** et **{noOfIssues} < 5** comme expressions de condition, car les valeurs comprises dans la plage **[5, 10]** ne seront gérées par aucun flux Séquence sortant.

Exécuter Simulation

1. Dans la fenêtre Configurer BPSim, cliquez sur l'onglet « Exécuter » et sur l'icône  dans la barre d'outils.
2. Une fois la simulation terminée, l'onglet Exécuter fournit des résultats similaires à ceux-ci :



Analyse des jetons

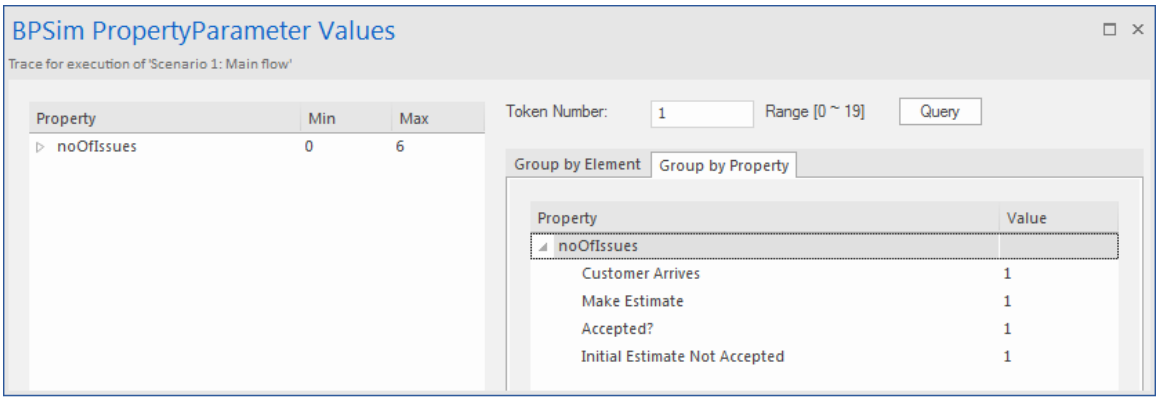
- 20 clients sont arrivés, ce qui correspond au nombre que nous avons calculé manuellement (voir *Arrivée du client* dans le tableau *Configurer BPSim*)
- 8 clients sur 20 n'ont pas accepté le devis initial, tandis que 12 sur 20 l'ont accepté et ont fait réparer leur voiture ; ces chiffres correspondent approximativement aux probabilités 1/3 et 2/3
- 64 jetons ont passé la Passerelle *New Issue Found ?* , dont 19 ont eu de nouvelles émissions et 45 non ; ces chiffres correspondent approximativement aux probabilités 1/4 et 3/4

Analyse sur les clients individuels

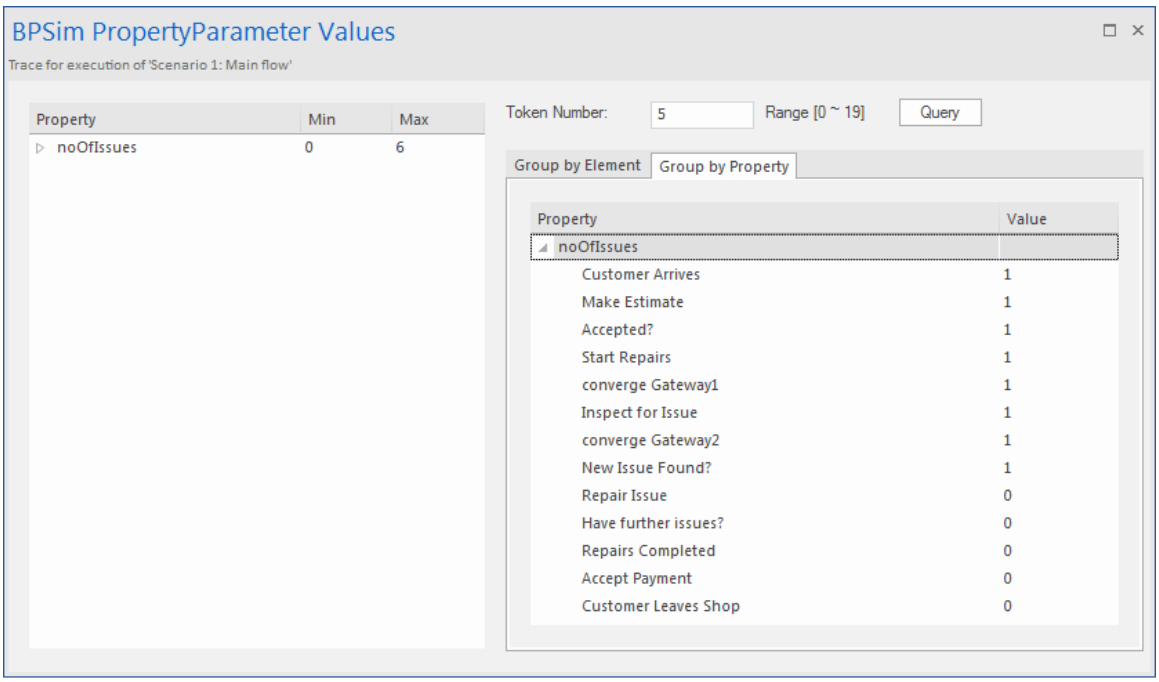
Cliquez sur le bouton  de la barre d'outils pour ouvrir la dialogue « Valeurs PropertyParameter BPSim ». Comme il y a 20 clients (jetons), vous pouvez saisir une valeur entre 0 et 19 dans le champ « Numéro de jeton » et cliquer sur le

bouton Query pour effectuer une analyse :

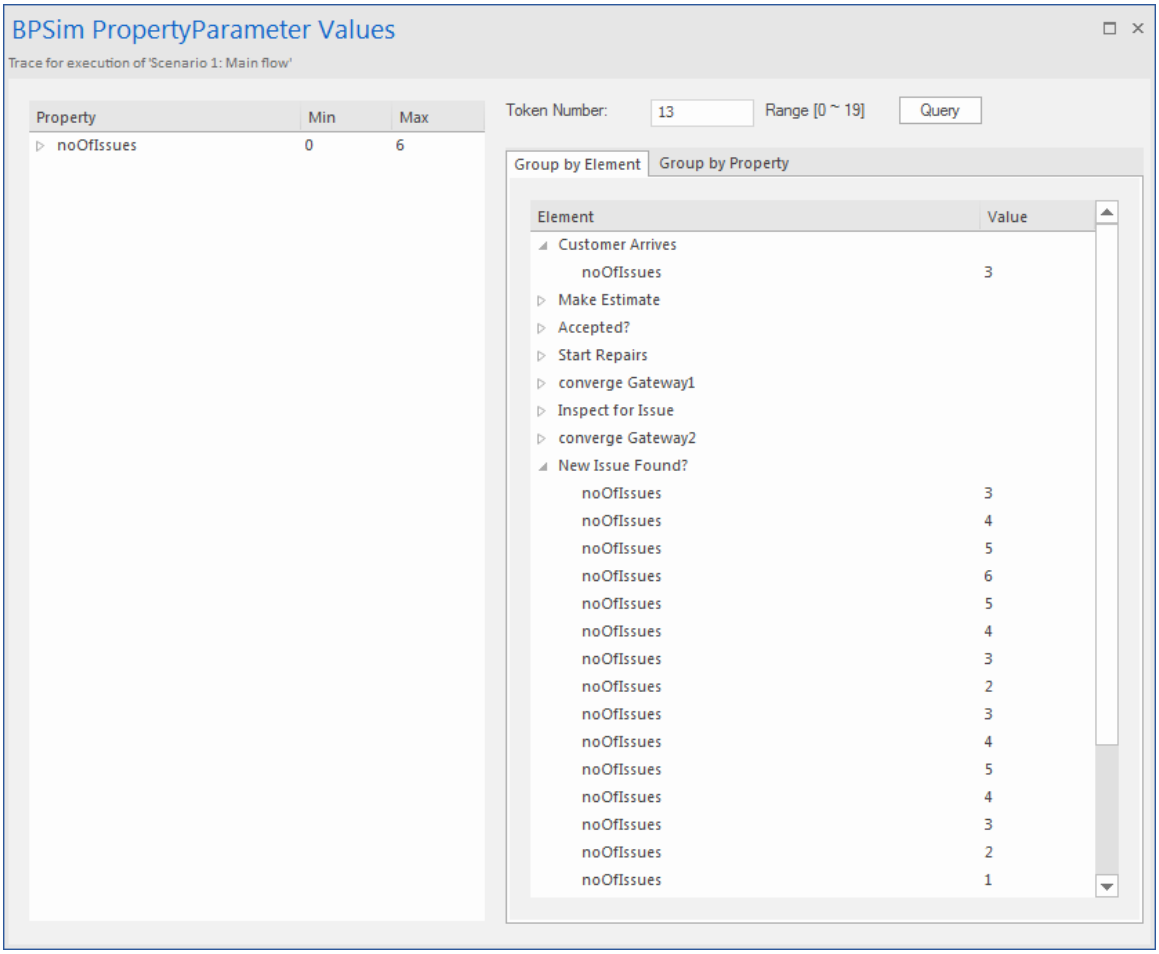
- Ce client n'a pas accepté l'estimation initiale, comme indiqué dans l'onglet « Grouper par propriété » :



- La voiture de ce client n'avait qu'un seul problème, qui a été résolu :



- La voiture de ce client avait trois problèmes connus et trois autres problèmes ont été découverts lors de la réparation, donc au total six problèmes ont été résolus (il s'agit peut-être d'une très vieille voiture) ; en passant à l'onglet « Grouper par élément » :


BPSim PropertyParameter Values
Trace for execution of 'Scenario 1: Main flow'
Property
Min
Max
noOfIssues
0
6
Token Number:
13
Range [0 ~ 19]
Query
Group by Element
Group by Property
Element
Value
Customer Arrives
noOfIssues
3
Make Estimate
Accepted?
Start Repairs
converge Gateway1
Inspect for Issue
converge Gateway2
New Issue Found?
noOfIssues
3
noOfIssues
4
noOfIssues
5
noOfIssues
6
noOfIssues
5
noOfIssues
4
noOfIssues
3
noOfIssues
2
noOfIssues
3
noOfIssues
4
noOfIssues
5
noOfIssues
4
noOfIssues
3
noOfIssues
2
noOfIssues
1

Exemples d'événements BPMN2.0

Un événement est quelque chose qui se produit au cours d'un processus. Événements affectent le déroulement du processus, ont généralement une cause ou un impact et nécessitent ou permettent généralement une réaction. Par exemple, le début d'une activité, la fin d'une activité, le changement d'état d'un document ou l'arrivée d'un message peuvent tous être considérés comme Événements .

Événements permettent de décrire des Processus « pilotés par événements ». Dans ces Processus , on distingue trois principaux types d'événements :

- Démarrer Événements , qui indiquent où un processus va démarrer
- Événements de fin, qui indiquent où le chemin d'un processus se terminera
- Événements intermédiaires, qui indiquent où quelque chose se produit entre le début et la fin d'un processus

Au sein de ces trois types, Événements peuvent être de deux sous-types :

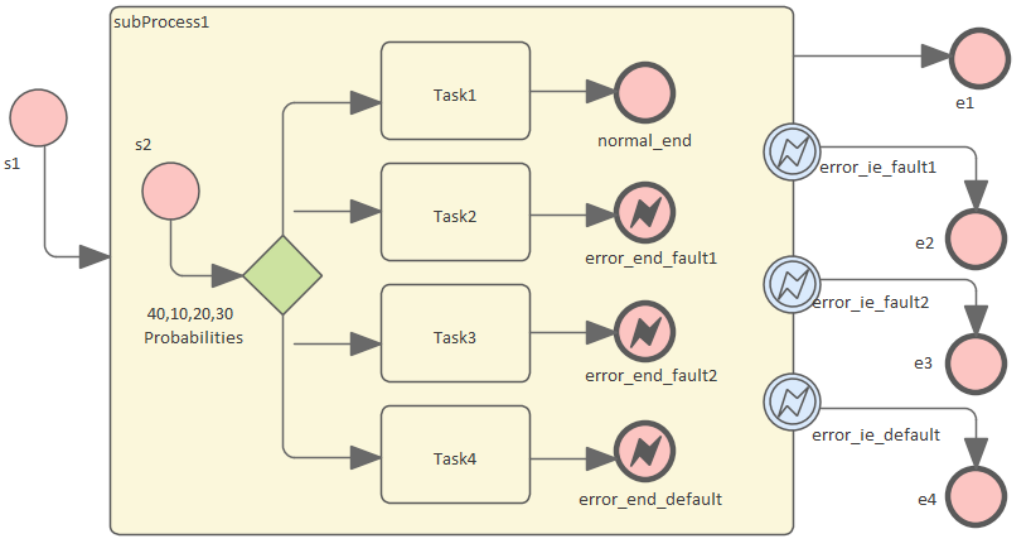
- Événements qui déclenchent un déclencheur - tous Événements Démarrer et certains Événements intermédiaires rattrapent Événements
- Événements qui génèrent un résultat - tous Événements de fin et certains Événements intermédiaires génèrent Événements qui pourraient éventuellement être détectés par un autre événement

Dans cette section, nous fournissons des exemples illustrant de nombreux événements BPMN 2.0 couramment utilisés. Dans chaque exemple, nous fournissons des instructions modélisation BPMN et de configuration BPSim étape par étape, ainsi qu'une analyse approfondie du résultat de la simulation. Tous les exemples sont disponibles dans le modèle EAExample.

Événement d'erreur

Lorsqu'un événement d'erreur intermédiaire se connecte à la bordure d'une activité, il devient partie intégrante d'un flux d'exception. L'événement est déclenché lorsqu'un jeton provoque la génération d'un nom d'erreur dans le flux normal, jusqu'à un événement de fin d'erreur.

Créer Modèle BPMN



Créer le processus principal

- Créer un événement Démarrer *s1*
- Ajoutez une Flux séquence à un élément d'activité cible *subProcess1 ;* agrandissez l'activité et cliquez-droit , en sélectionnant l'option « Est développé », puis ouvrez la dialogue « Propriétés » et définissez « Type » sur « sous-processus »
- Ajouter une Flux séquence à un élément d'événement de fin cible *e1* (' Type ' défini sur ' Aucun ')
- Créez trois Événements intermédiaires Bordure , en faisant glisser les éléments depuis la boîte à outils et en les déposant sur *le sous-processus 1* ; dans les menus instantanés, sélectionnez « Monté sur bord » et « Erreur » :
  - *error_ie_fault1* ; ajouter une Flux séquence à un élément EndEvent cible *e2* (' Type ' défini sur 'Aucun')
  - *error_ie_fault2* ; ajouter une Flux séquence à un EndEvent cible élément *e3* (' Type ' défini sur 'Aucun')
  - *error_ie_default* ; ajouter une Flux séquence à un élément EndEvent cible *e4* (' Type ' défini sur 'Aucun')

Créer le sous-processus

Dans le cadre de l'activité *subProcess1* :

- Créez un Démarrer Event *s2* , 'Standalone' et définissez ' Type ' sur 'Aucun'
- Créez une Flux séquence vers un élément Passerelle cible défini sur « Exclusif » et portant le nom « *40,10,20,30 Probabilités »*
- Créez des flux Séquence vers quatre éléments d'activité cibles de Type « abstractTask » appelés :
  - *Tâche 1* , et ajouter une Flux séquence à un événement final cible appelé *normal_end* , « Type » défini sur « Aucun »
  - *Tâche 2* , et ajouter une Flux séquence à un événement final cible appelé *error_end_fault1, «* Type » défini sur « Erreur »
  - *Tâche 3* , et ajouter une Flux séquence à un événement final cible appelé *error_end_fault2,* « Type » défini sur « Erreur »

- *Tâche 4* , et ajouter une Flux séquence à un événement final cible appelé *error_end_default,* « Type » défini sur 'Erreur'

Créer des éléments BPMN2.0::Error

Créez les éléments d'erreur *Fault1* et *Fault2* , qui seront utilisés comme code d'erreur par Événements .

- Double-cliquez sur l'élément *error_end_fault1* et, dans la boîte de dialogue ' Propriétés ' onglet 'BPMN2.0', localisez l' étiquette 'errorRef'
- Dans le champ « Valeur », cliquez sur le bouton ... et accédez au Paquetage contenant ce modèle
- Cliquez sur le bouton Ajouter nouveau et, dans le champ « Nom », saisissez le nom *Fault1* , puis cliquez sur le bouton Enregistrer
- Cliquez à nouveau sur le bouton Ajouter nouveau et, dans le champ « Nom », saisissez le nom *Fault2* , puis cliquez sur le bouton Enregistrer
- Cliquez sur le bouton OK , puis à nouveau sur le bouton OK suivant

Configurer Événements pour les codes d'erreur

- Double-cliquez sur l'élément *error_end_fault1* et, dans la boîte de dialogue ' Propriétés ' onglet 'BPMN2.0', localisez l' étiquette 'errorRef'
- Dans le champ « Valeur », cliquez sur le bouton ... et accédez au Paquetage contenant ce modèle
- Cliquez sur *Fault1* , puis sur le bouton OK , et encore sur le bouton OK .

Faites de même pour ces éléments :

- *error_end_fault2* , en cliquant sur *Fault2*
- *error_ie_fault1* , en cliquant sur *Fault1*
- *error_ie_fault2* , en cliquant sur *Fault2*

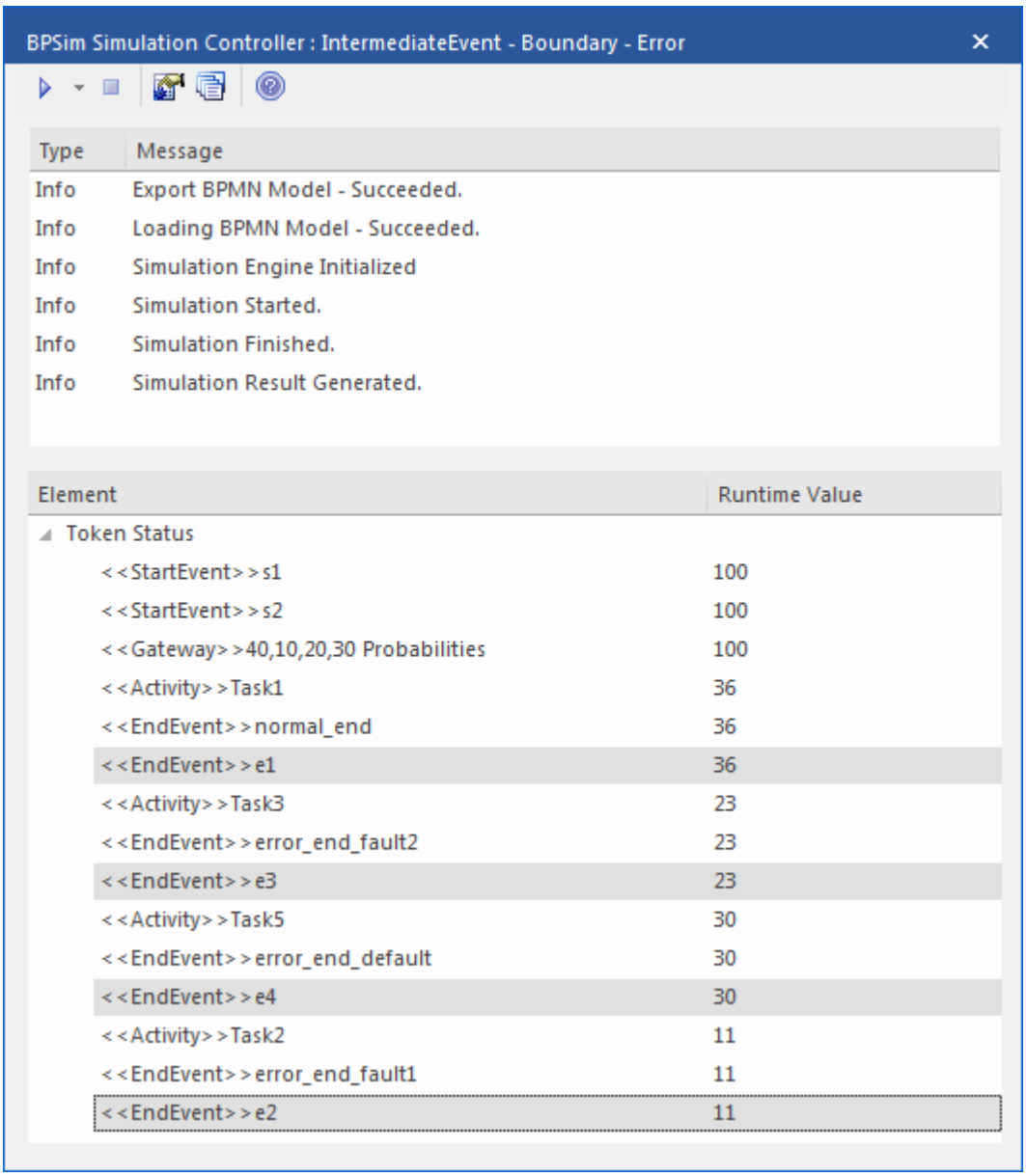
Configurer BPSim

Object	Action
Artefact et Paquetage	• Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager') • Créez un artefact nommé « IntermediateEvent - Bordure - Error » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton ... et sélectionnez son Paquetage parent et cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer et sur le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la fenêtre Configurer BPSim.
s1	• Dans l'arborescence à gauche de la fenêtre Configurer BPSim, développez « StartEvent » et cliquez sur « s1 » • Dans l'onglet « Contrôle », dans le champ « Nouveau paramètre... », cliquez sur la flèche déroulante et sélectionnez « TriggerCount » • Dans le champ « Valeur », saisissez « 100 »
Probabilité	Dans l'arborescence à gauche de la fenêtre Configurer BPSim, développez « Passerelle | Probabilités 40,10,20,30 ». *Conseils : Vous pouvez également faire flotter la fenêtre Configurer BPSim, puis cliquer sur l'élément ou les connecteurs du diagramme BPMN ; l'élément dans la fenêtre Configurer BPSim sera automatiquement sélectionné.*

	Pour chacun des éléments *de la tâche* , dans l'onglet « Contrôle », cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Probabilité », puis saisissez la valeur correspondante dans le champ « Valeur » : • Pour la tâche 1, tapez « 0,4 » • Pour la tâche 2, tapez « 0,1 » • Pour la tâche 3, tapez « 0,2 » • Pour la tâche 4, tapez « 0,3 »

Exécuter Simulation

- Dans la barre d'outils de la dialogue « Configurer BPSim », cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur BPSim »
- Cliquez sur le bouton Exécuter et sélectionnez « Simulation standard »
- Les résultats de la simulation ressemblent à ceci :



Analyse:

L'ensemble Probabilité sur les flux Séquence sortant de *40,10,20,30 est respectivement de* 0,4, 0,1, 0,2 et 0,3.

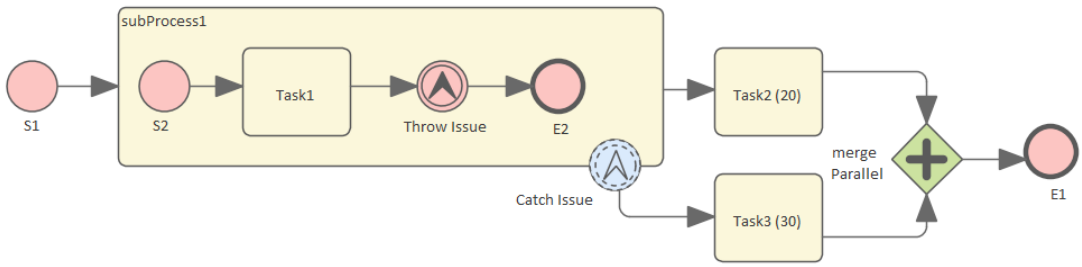
- 36 passes sur 100 se sont terminées à *normal_end* , qui a abouti à *e1*
- 11 passes sur 100 se sont terminées à *error_end_fault1* , ce qui a déclenché *error_ie_fault1* par ErrorRef *Fault1,* et l'exception s'est déroulée vers *e2*
- 23 passes sur 100 se sont terminées à *error_end_fault2* , ce qui a déclenché *error_ie_fault2* par ErrorRef *Fault2,* et l'exception s'est déroulée vers *e3*
- 30 passes sur 100 se sont terminées à *error_end_default* , ce qui a déclenché *error_ie_default* car ils n'ont pas défini ErrorRef et l'exception s'est déroulée jusqu'à *e4*

Les nombres 36, 11, 23 et 30 totalisent 100, ce qui a été défini comme TriggerCount dans *s1,* ils correspondent donc à la probabilité de 100 %

Événement d'escalade

Dans BPMN, l'escalade est l'équivalent non-interruptible de l'erreur, avec un comportement de réception/d'interception similaire. Cependant, contrairement à l'erreur, les sorties de flux normal et de flux d'exception de l'activité sont des chemins parallèles et non alternatifs.

Créer Modèle BPMN



Créer le processus principal

- Créer un événement Démarrer *S1*
- Ajoutez une Flux séquence à une activité cible *subProcess1 ;* agrandissez l'activité et cliquez-droit , en sélectionnant l'option « Est développé », puis ouvrez la dialogue « Propriétés » et définissez « Type » sur « sous-processus »
- Ajoutez une Flux séquence à un élément d'activité cible abstractTask *Task2 (20)* (ouvrez la dialogue « Propriétés » et définissez le champ « Type » sur « abstractTask »)
- Ajouter une Flux séquence à une cible parallèle Élément Passerelle *fusion Parallèle* (ouvrir la dialogue « Propriétés » et définir le champ « Type » sur « parallèle »)
- Ajouter une Flux séquence à un événement de fin cible *E1*
- Sur *le sous-processus1* , ajoutez une bordure non-interrupting Escalation Event *Catch Issue* (faites glisser l'icône « Événement intermédiaire » sur *le sous-processus1* , et dans les menus instantanés, sélectionnez « Monté sur bordure » et « Escalade » ; double-cliquez sur l'élément pour afficher la dialogue « Propriétés » et ajoutez le nom, puis dans le champ « Type » sélectionnez « Bordure non-interrupting > Escalation »)
- Ajouter une Flux séquence à un élément d'activité cible abstractTask *Task3 (30)* (ouvrez la dialogue « Propriétés » et définissez le champ « Type » sur « abstractTask »)
- Ajouter une Flux séquence à la *fusion d'éléments cibles en parallèle*

Créer le sous-processus

- Dans (ou sous) *subProcess1* , créez un Démarrer Event *S2*
- Ajoutez une Flux séquence à un élément d'activité cible abstractTask *Task1* (ouvrez la dialogue « Propriétés » et définissez le champ « Type » sur « abstractTask »)
- Ajouter une Flux séquence à une cible Lancer Escalade Événement intermédiaire *Lancer Problème* (ouvrez la dialogue « Propriétés » et dans le champ « Type », sélectionnez « Lancer > Escalade »)
- Ajouter une Flux séquence à un événement de fin cible *E2*

Créer des éléments BPMN2.0::Escalation

Depuis la boîte à outils Diagramme , développez la page « Types BPMN 2.0 », faites glisser l'icône « Escalation » sur le diagramme et donnez à l'élément le nom *Escalation1* ; celui-ci sera utilisé comme code d'escalade par les Événements .

Configurer Événements pour les codes d'escalade :

- Double-cliquez sur *Throw Issue* et dans le champ « Valeur » de l' étiquette escalationRef, cliquez sur l'icône  et recherchez et sélectionnez *Escalation1*
- Double-cliquez sur *Catch Issue* et, encore une fois, dans le champ « Valeur » de l' étiquette escalationRef, cliquez sur l'icône  et recherchez et sélectionnez *Escalation1*

(Les sorties de flux d'exception de l'activité sont parallèles.)

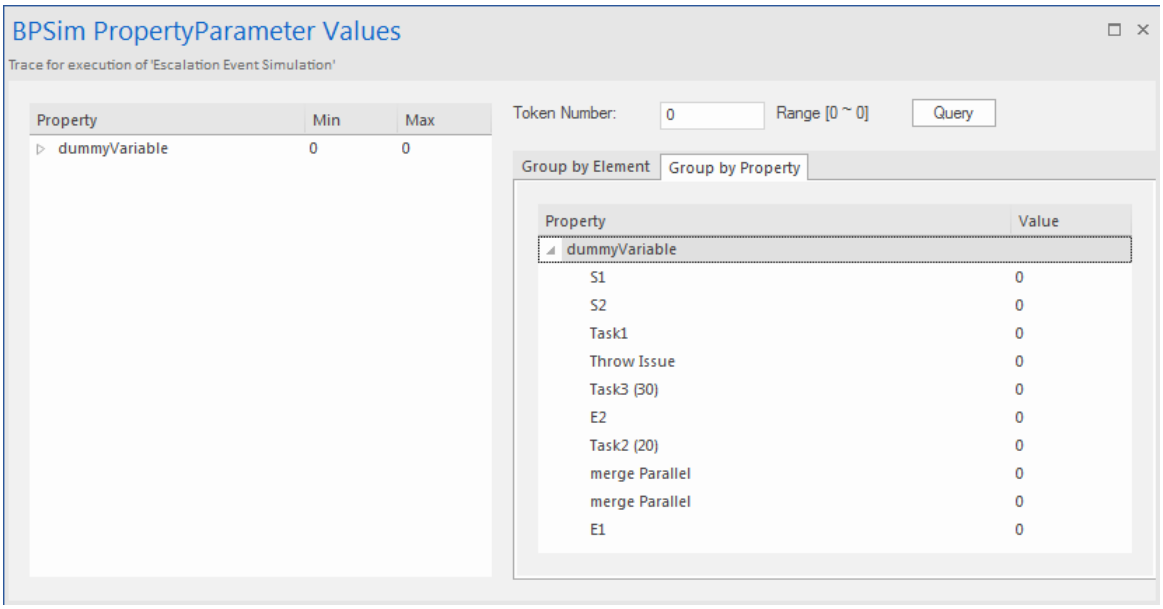
Configurer BPSim

Tâche	Action
Artefact et Paquetage	• Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager') • Créez un artefact nommé « Simulation d'événement d'escalade » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton  et sélectionnez son Paquetage parent et cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer et sur le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la fenêtre Configurer BPSim.
Déclencheur Count of Démarrer Event	• Dans l'arborescence à gauche de la fenêtre Configurer BPSim, développez « StartEvent » et cliquez sur *S1* • Dans l'onglet « Contrôle », dans le champ « Nouveau paramètre... », cliquez sur la flèche déroulante et sélectionnez « TriggerCount » • Dans le champ « Valeur », saisissez « 1 »
Temps de traitement	• Dans l'arborescence de gauche, développez « Activité » et cliquez sur *Tâche2 (20)* ; dans le champ « Valeur » pour « Temps de traitement », saisissez « 20 » et dans le champ « Unité », saisissez « s » (pour 20 secondes) • Cliquez sur *Task3 (30) ;* de la même manière, définissez « ProcessingTime » sur 30 secondes
dummyVariable pour Trace	Afin d'afficher la trace exacte d'un jeton donné, vous devez définir une variable factice sur *S1* . • Dans la hiérarchie de gauche, cliquez sur *S1* , puis dans l'onglet « Propriétés », remplacez le texte *de la nouvelle propriété* par le nom d'une variable (par exemple « dummyVariable ») • Dans le champ « Valeur », cliquez sur le bouton  et, dans la dialogue « <<StartEvent>>S1 : <nom de la variable> », cliquez sur « Numérique » et saisissez une valeur « Constante numérique » de « 0 » ; cliquez sur le bouton OK

Exécuter Simulation

- Dans la barre d'outils de la dialogue « Configurer BPSim », cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur Simulation BPSim »
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »

- Après la simulation, cliquez sur le bouton  dans la barre d'outils pour afficher la dialogue « Valeurs PropertyParameter BPSim »
- Cliquez sur le bouton Query et sur l'onglet « Grouper par propriété », puis développez « dummyVariable » (ou le nom que vous avez attribué à la variable)



Analyse:

Contrairement à *Error* , les sorties de flux normal et de flux d'exception du *sous-processus 1* ne sont pas des chemins alternatifs mais parallèles. Cette fonctionnalité peut être facilement découverte à partir de la trace :

- *E2* et *Task2 (20)* sont toujours parcourus après le démarrage *de Task3 (30)*
- *E1* a été atteint après que *mergeParallel* ait été parcouru deux fois

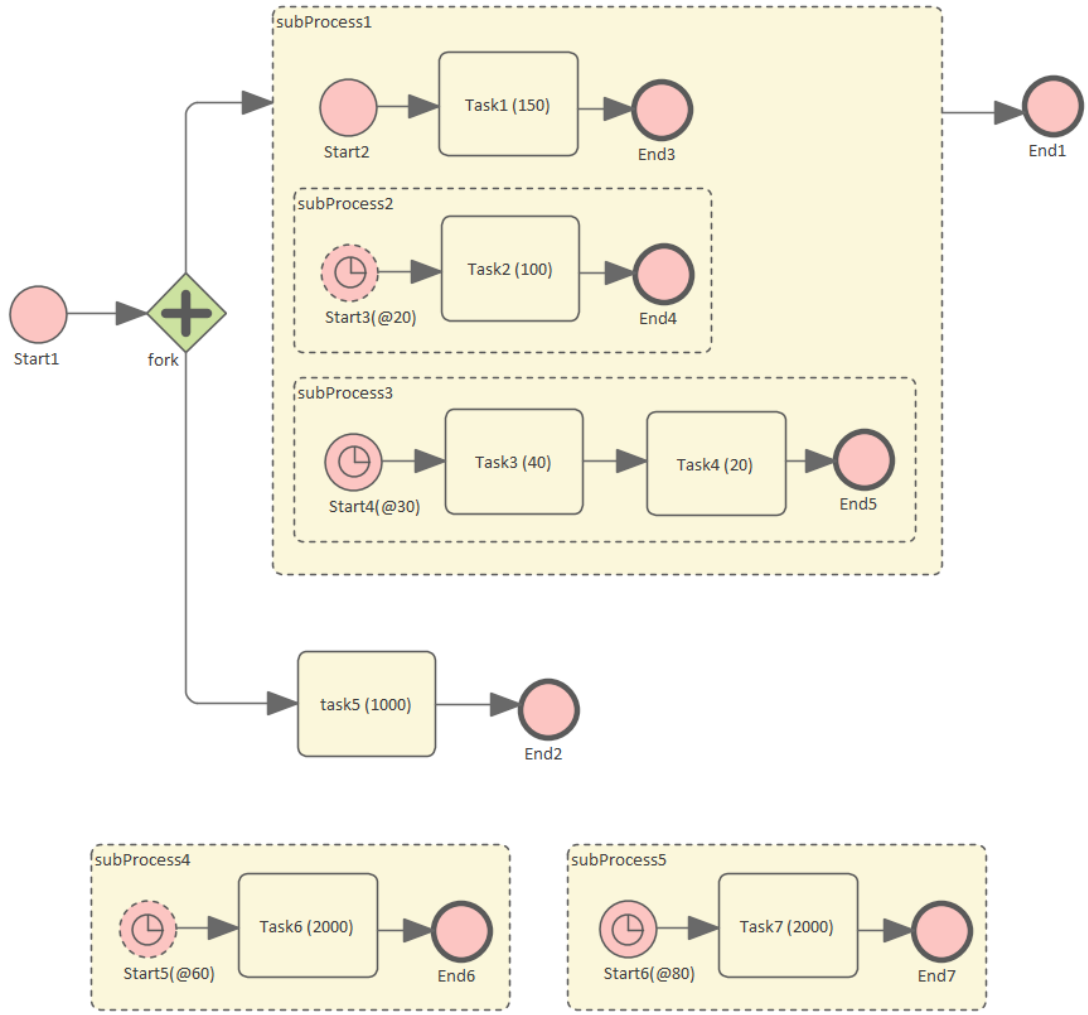
Sous-processus d'événement

Les sous-processus d'événement permettent à votre système de gérer un événement dans le contexte d'un sous-processus ou d'un processus donné. Un sous-processus d'événement commence toujours par un événement Démarrer ; il n'est pas instancié par le flux de contrôle normal, mais uniquement lorsque l'événement Démarrer associé est déclenché. Les sous-processus d'événement sont autonomes et NE DOIVENT PAS être connectés au reste des flux Séquence dans les sous-processus.

- Si l'attribut isInterrupting de son événement Démarrer est défini, un sous-processus d'événement annule l'exécution du sous-processus englobant
- Si l'attribut isInterrupting n'est pas défini, l'exécution du sous-processus englobant se poursuit en parallèle avec le sous-processus d'événement

Dans cet exemple, nous démontrons comment les sous-processus d'événement interrompant et non interrompant affectent la ligne de vie du sous-processus et du processus englobants.

Créer Modèle BPMN



Modèle le processus principal

- Créer un StartEvent *Start1*
- Ajouter une Flux séquence à une *fourchette* d'élément Parallel Passerelle cible
- Ajoutez une Flux séquence à

- un sous-processus *subProcess1* , et à partir de celui-ci ajouter un flux de *séquence* à un élément d'événement de fin cible
Fin1
- une tâche abstraite *Task5* , et à partir de là ajouter une Flux séquence à un élément d'événement de fin cible *End2*

Conseils sur la façon de modéliser un sous-processus événementiel

- Faites glisser une activité de la boîte à outils « BPMN2.0 - Processus Métier » sur le diagramme
- Double-cliquez sur l'activité pour afficher la dialogue « Propriétés » et, dans le champ « Type », sélectionnez « subProcess » ; définissez « triggeredByEvent » sur « true » et cliquez sur le bouton OK
- Cliquez-droit sur l'élément et sélectionnez l'option « Est développé » ; cela affichera le nom de l'élément dans le coin supérieur gauche

Modèle les sous-processus événementiels pour le processus principal

- Créer un sous-processus d'événement *subProcess4*
- Créez un Timer Démarrer Event *Start5(@60)* , puis double-cliquez dessus pour afficher la dialogue ' Propriétés ' et, dans le champ « Type », sélectionnez « Sous-processus d'événement non interrompu > Minuterie » ; cliquez sur le bouton OK
- Ajouter une Flux séquence à une tâche abstraite cible Activité *Tâche 6 (2000)*
- Ajouter une Flux séquence à un élément d'événement de fin cible *End6*
- Créer un sous-processus d'événement *subProcess5*
- Créez un Timer Démarrer Event *Start6(@80)* , puis double-cliquez dessus pour afficher la dialogue ' Propriétés ' et, dans le champ « Type », sélectionnez « Interruption du sous-processus d'événement > Minuterie » ; cliquez sur le bouton OK
- Ajouter une Flux séquence à une tâche abstraite cible Activité *Tâche 7 (2000)*
- Ajouter une Flux séquence à un élément d'événement de fin cible *End7*

Modèle le sous-processus subProcess1 et les sous-processus événementiels inclus

- Créer un StartEvent *Start2*
- Ajouter un flux de *séquence* à une tâche abstraite cible Activité *Tâche1(150)*
- Ajouter un Flux séquence vers une cible Fin de l'événement *Fin3*
- Créer un sous-processus d'événement *subProcess2*
- Créez un Timer Démarrer Event *Start3(@20)* , puis double-cliquez dessus pour afficher la dialogue ' Propriétés ' et, dans le champ « Type », sélectionnez « Sous-processus d'événement non interrompu > Minuterie »
- Ajouter une Flux séquence à une tâche abstraite cible Activité *Tâche2(100)*
- Ajouter une Flux séquence à un élément d'événement de fin cible *End4*
- Créer un sous-processus d'événement *subProcess3*
- Créez un Timer Démarrer Event *Start4(@30)* , puis double-cliquez dessus pour afficher la dialogue ' Propriétés ' et, dans le champ « Type », sélectionnez « Interruption du sous-processus d'événement > Minuterie »
- Ajouter une Flux séquence à une tâche abstraite cible Activité *Tâche 3(40)*
- Ajouter une Flux séquence à une tâche abstraite cible Activité *Tâche 4(20)*
- Ajouter une Flux séquence à un élément d'événement de fin cible *End5*

Configurer BPSim

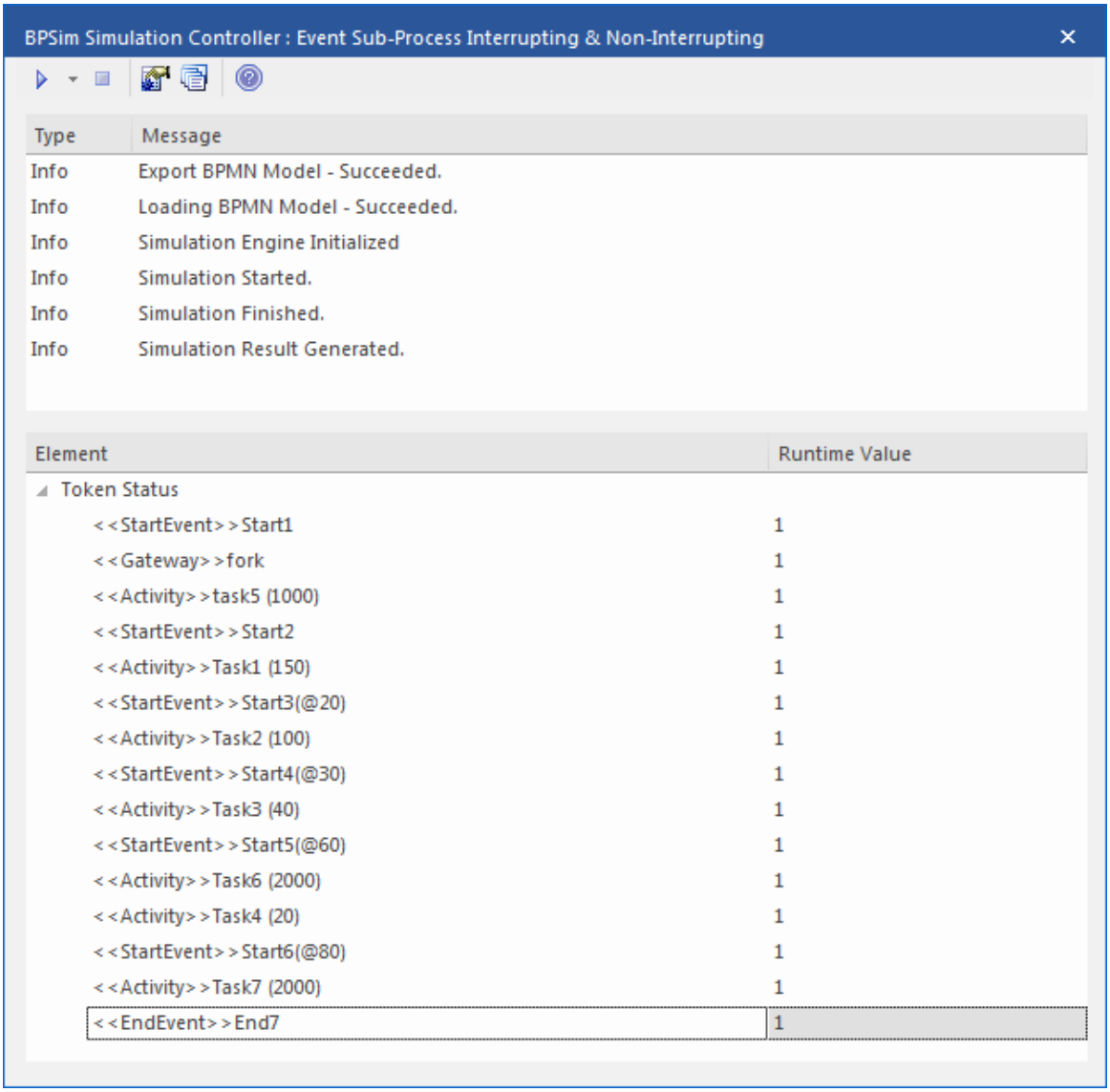
À l'aide de ce tableau , nous créons l'artefact dans le Paquetage de configuration et configurons les valeurs des paramètres de chaque élément.

Tâche	Action

Créer un artefact	• Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager') • Créez un artefact nommé « Sous-processus d'événement interrompant et non interrompant » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton  et sélectionnez son Paquetage parent et cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer et sur le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la fenêtre Configurer BPSim.
InterTriggerTimer pour Démarrer Événements dans le sous-processus d'événement	Dans l'arborescence à gauche de la dialogue « Configurer BPSim », développez « StartEvent ». Pour chacun des éléments répertoriés ici, dans l'onglet « Contrôle », cliquez sur la flèche déroulante dans le champ « Nouveau paramètre... » et sélectionnez le paramètre « InterTriggerTimer ». Cliquez sur le bouton  dans le champ « Valeur » pour ouvrir la dialogue « Paramètre » et sélectionnez « Constante > Numérique », puis saisissez la valeur et sélectionnez « secondes ». • Start3(@20) : 20 secondes • Start4(@30) : 30 secondes • Start5(@60) : 60 secondes • Start6(@80) : 80 secondes
Temps de traitement des tâches	Dans l'arborescence à gauche de la fenêtre Configurer BPSim, développez « Activité ». Pour chacun des éléments répertoriés ici, dans l'onglet « Heure », cliquez sur la flèche déroulante dans le champ « Nouveau paramètre... » et sélectionnez le paramètre « ProcessingTime ». Cliquez sur le bouton  dans le champ « Valeur » pour ouvrir la dialogue « Paramètre » et sélectionnez « Constante > Numérique », puis saisissez la valeur et sélectionnez « secondes ». • Tâche 1 (150) : 150 secondes • Tâche 2 (100) : 100 secondes • Tâche 3 (40) : 40 secondes • Tâche 4 (20) : 20 secondes • Tâche 5 (1000) : 1000 secondes • Tâche 6 (2000) : 2000 secondes • Tâche 7 (2000) : 2000 secondes

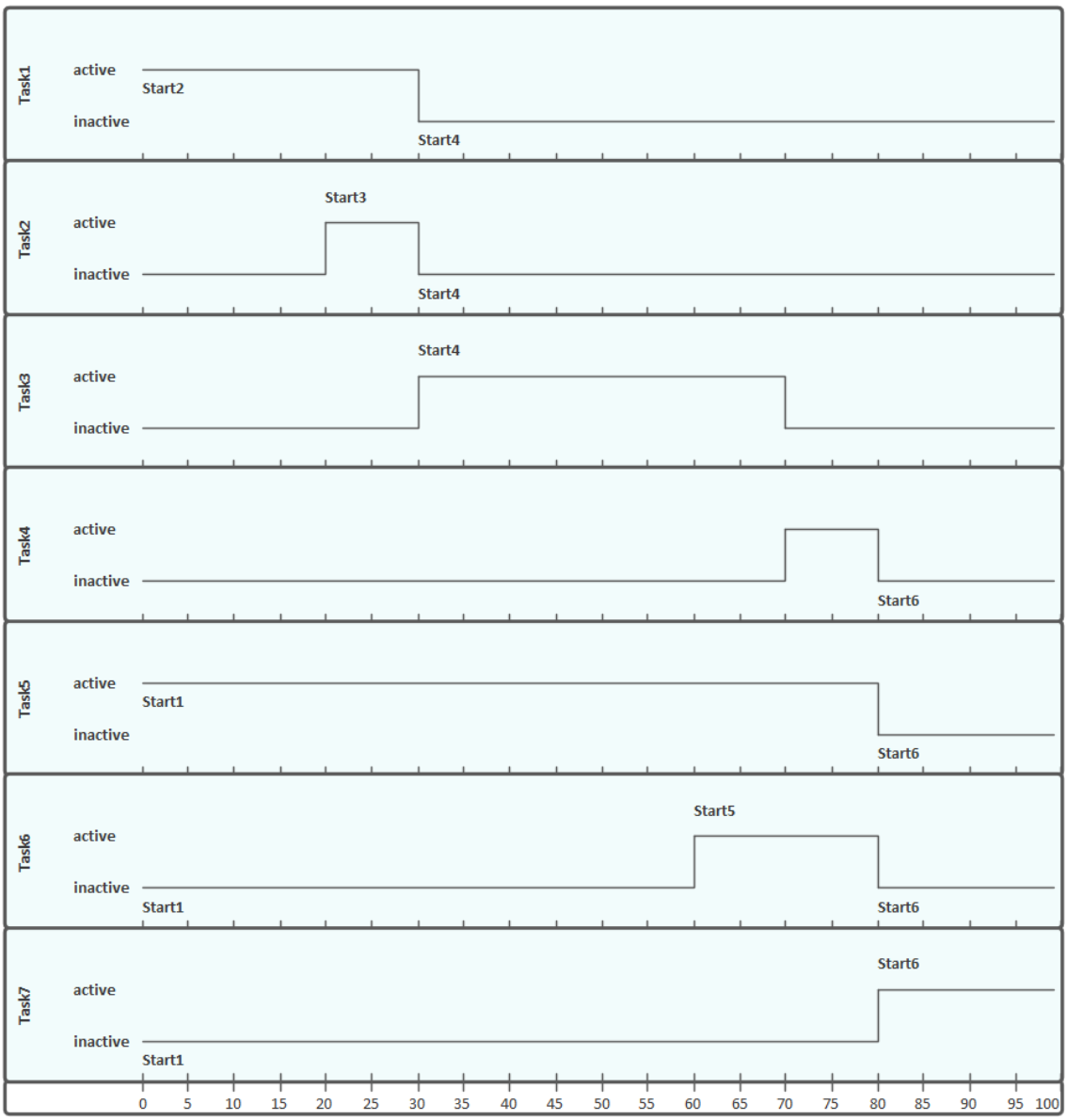
Exécuter Simulation

- Dans la barre d'outils de la dialogue « Configurer BPSim », cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur Simulation BPSim »
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »



Analyse

En lisant les résultats, il n'est peut-être pas tout à fait évident de voir ce qui s'est passé ; cependant, si nous traçons la ligne de vie de chaque tâche dans un diagramme de temps, cela devient plus clair.



- L'événement *Start3 (@20)* n'est pas interrompu, il n'a pas arrêté *la tâche 1* à 20 secondes
- L'événement *Start4 (@30)* est en cours d'interruption, il a arrêté *Task1* et *Task2* au bout de 30 secondes ; il n'a pas affecté *Task5* car le niveau du processus englobant *Task5* (processus principal) est supérieur à celui du sous-processus englobant *Start4* (*subProcess1*)
- L'événement *Start5 (@60)* n'est pas interrompu, il a démarré *la tâche 6* à 60 secondes sans affecter *la tâche 3* ou *la tâche 5*
- L'événement *Start6 (@80)* est interrompu. Il a démarré *la tâche 7* à 80 secondes et a interrompu les tâches en cours d'exécution (*Task4* , *Task5* , *Task6*) qui se trouvaient au même niveau ou à un niveau inférieur de son processus englobant.
- Seul *End7* est atteint comme prévu

Générateur de nombres de Fibonacci avec événement de lien

Un Link Event est un mécanisme permettant de connecter deux sections d'un Process. Les Link Événements peuvent être utilisés :

- Pour créer des scénarios en boucle, en tant qu'objets génériques « Go To » au niveau du processus
- Pour éviter les longues lignes Flux séquence , Événements de lien appariés peuvent être utilisés comme connecteurs « hors page » pour imprimer un processus sur plusieurs pages

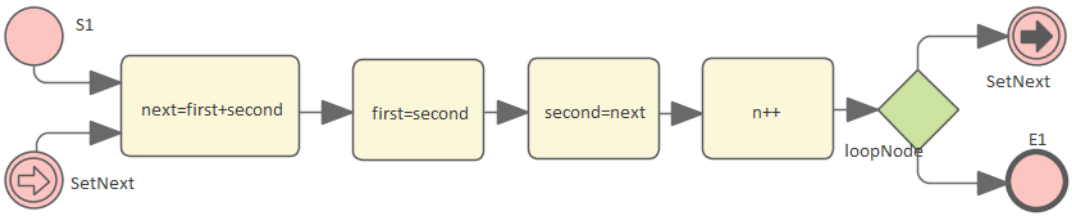
L'utilisation des Link Événements est limitée à un seul niveau de Processus (c'est-à-dire qu'ils ne peuvent pas lier un Processus parent à un sous-Processus).

Il peut y avoir plusieurs Événements de lien source, mais il ne peut y avoir qu'un seul événement de lien cible.

- Le marqueur d'événement de lien cible n'est pas rempli, pour « attraper » le lien source
- Le marqueur d'événement de lien source est rempli pour « lancer » vers le lien cible

Lorsque le Moteur d'Exécution EABPSim exécute la simulation, les Événements de Lien source-cible sont appariés par l'élément NOM, ils ne peuvent donc pas être vides.

Créer Modèle BPMN



- Créer un StartEvent *S1*
- Ajoutez une Flux séquence à un élément d'activité abstractTask cible *next=first+second* (ouvrez la dialogue « Propriétés » et définissez le champ « Type » sur « abstractTask »)
- Ajouter une Flux séquence à une cible abstractTask Élément d'activité *premier=deuxième*
- Ajouter une Flux séquence à un élément d'activité abstractTask cible *second=next*
- Ajouter une Flux séquence à un élément d'activité abstractTask cible *n++*
- Ajoutez une Flux séquence à un élément Passerelle exclusif *loopNode* cible (dans le menu instantané, sélectionnez « Exclusif »)
- Ajoutez une Flux séquence à chacun de ces éléments cibles :
  - Un élément d'événement intermédiaire de lien de lancement *SetNext* (ouvrez la dialogue « Propriétés » et définissez le
  (Type du champ « Lancer > Lien ») et
  - Un élément d'événement de fin *E1*
- Créez un élément d'événement intermédiaire de lien de capture *SetNext* (ouvrez la dialogue « Propriétés » et définissez le champ « Type » sur « Capture > Lien »)
- Ajouter une Flux séquence à l'élément cible *next=first+second*

Configurer BPSim

Nous utiliserons les paramètres de propriété pour définir comment le flux de séquence forme une boucle au cours de laquelle un nombre de Fibonacci sera généré. Le mécanisme de boucle est implémenté via la paire d' Événements Link.

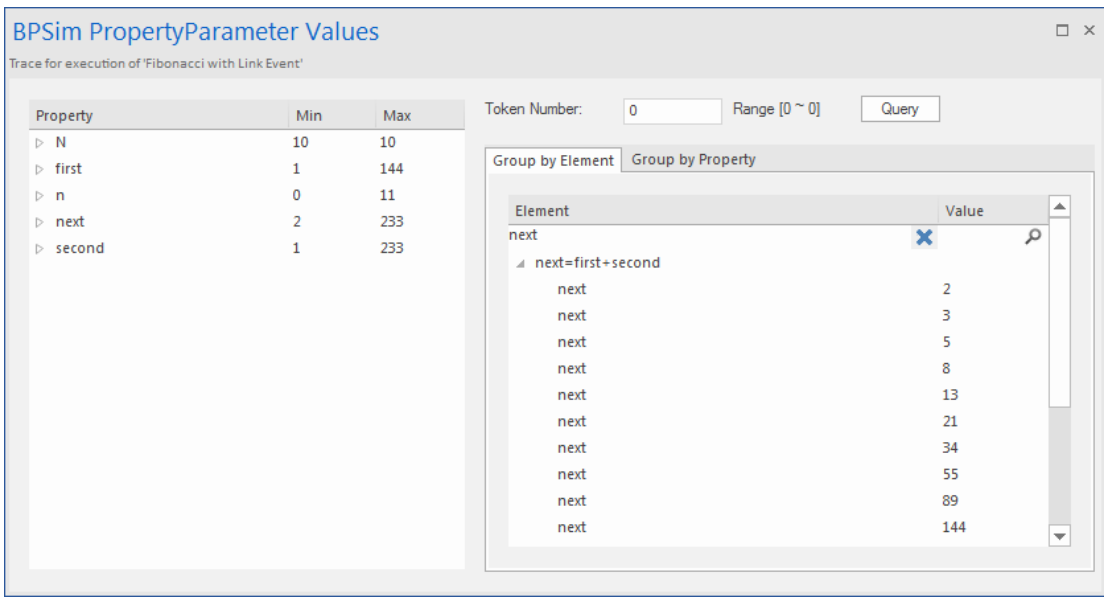
Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager')

Tâche	Action
Élément: S1	Dans la liste des types d'éléments sur la gauche, développez le groupe Démarrer Event et cliquez sur *S1* . Cliquez sur l'onglet « Contrôle » et sur la flèche déroulante « Nouveau paramètre » ; sélectionnez « TriggerCount ». Dans le champ « Valeur », saisissez « 1 ». **Cliquez sur l'onglet ' Propriétés '** Remplacez le texte *de la nouvelle propriété* pour créer ces propriétés : • N - et tapez « 10 » dans le champ « Valeur » comme nombre total de nombres de Fibonacci à générer • premier - et tapez « 1 » dans le champ « Valeur » • deuxièmement - et tapez « 1 » dans le champ « Valeur » • n - et tapez « 0 » dans le champ « Valeur » comme n- *ième* nouveau nombre de Fibonacci
Élément : suivant=premier+second	Dans la liste des types d'éléments, développez le groupe Activité et cliquez sur *next=first+second* . Cliquez sur l'onglet « Propriétés » et remplacez le texte *de la nouvelle propriété* par « suivant ». Dans le champ « Valeur », cliquez sur le bouton  , cliquez sur l'onglet « Expression » et saisissez l'expression « {premier}+{second} ». Cliquez sur le bouton OK .
Élément : premier=second	Dans la liste des types d'éléments, dans le groupe Activité, cliquez sur *premier=second* . Cliquez sur l'onglet « Propriétés » et remplacez le texte *de la nouvelle propriété* par « first ». Dans le champ « Valeur », cliquez sur le bouton  , cliquez sur l'onglet « Expression » et saisissez l'expression « {seconde} ». Cliquez sur le bouton OK .
Élément : second=next	Dans la liste des types d'éléments, dans le groupe Activité, cliquez sur *second=next* . Cliquez sur l'onglet « Propriétés » et remplacez le texte *de la nouvelle propriété* par « second ». Dans le champ « Valeur », cliquez sur le bouton  , cliquez sur l'onglet « Expression » et tapez l'expression « {next} ». Cliquez sur le bouton OK .
Élément : n++	Dans la liste des types d'éléments, dans le groupe Activité, cliquez sur *n++* . Cliquez sur l'onglet « Propriétés » et remplacez le texte de *la nouvelle propriété* par « n ». Dans le champ « Valeur », cliquez sur le bouton  , cliquez sur l'onglet « Expression » et saisissez l'expression « {n}+1 ». Cliquez sur le bouton OK .

Conditions de Passerelle	Dans la liste des types d'éléments, développez le groupe Passerelle et l'élément LoopNode et cliquez sur *SetNext* . Cliquez sur l'onglet « Contrôle » et sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition ». Dans le champ « Valeur », cliquez sur le bouton ... , cliquez sur l'onglet « Expression » et saisissez l'expression « {n} <={N} ». Cliquez sur le bouton OK . Cliquez maintenant sur *E1* . Cliquez sur l'onglet « Contrôle » et sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition ». Dans le champ « Valeur », cliquez sur le bouton ... , cliquez sur l'onglet « Expression » et saisissez l'expression « {n} > {N} ». Cliquez sur le bouton OK .

Exécuter Simulation

- Dans la dialogue « Configurer BPSim », dans la barre d'outils, cliquez sur l'icône « Exécuter » ; la dialogue « Contrôleur Simulation BPSim » s'affiche
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »
- Une fois la simulation terminée, cliquez sur l'icône dans la barre d'outils ; la dialogue « BPSim PropertyParameter Values » s'affiche.
- Cliquez sur le bouton Query et sur l'onglet « Grouper par élément », puis développez « next=first+second » ; toutes les valeurs instantané des attributs sont répertoriées
- Appliquez un filtre 'suivant' (cliquez-droit sur l'en-tête de la liste, sélectionnez 'Toggle Barre de Filtre ' et tapez 'suivant' sous l'en-tête 'Elément') ; les résultats ressembleront à cette image :



Dix autres nombres de Fibonacci sont générés :

2,3,5,8,13,21,34,55,89,144

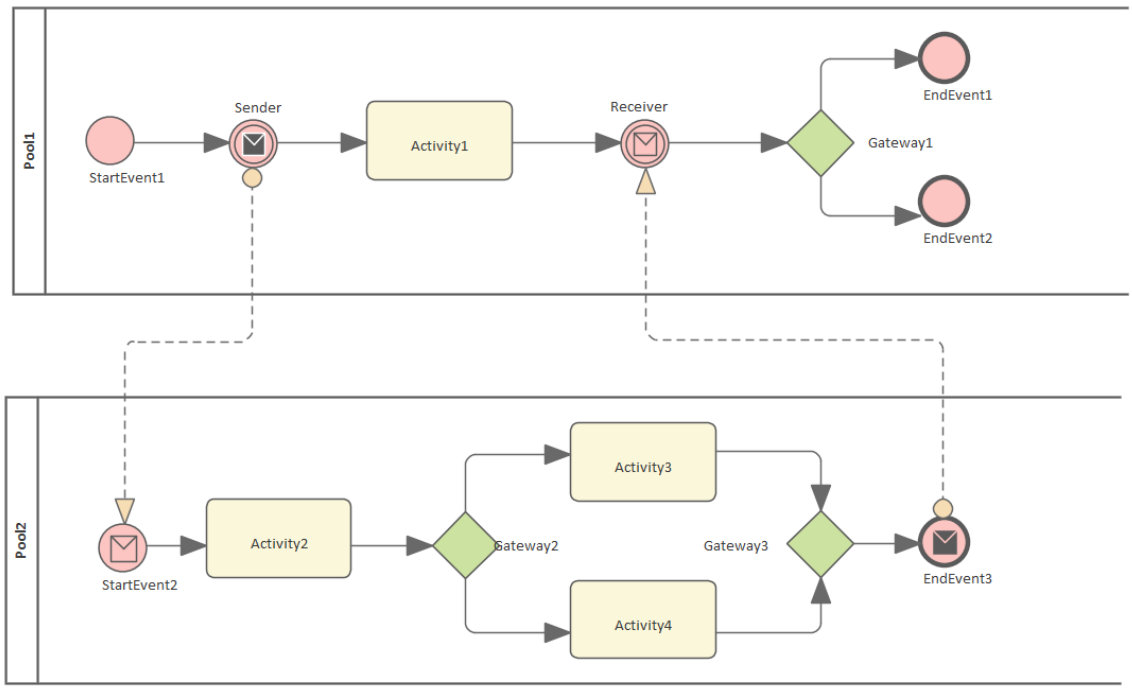
Message d'événement

Lorsqu'il est utilisé dans un flux de séquence normal, l'événement de message peut être utilisé pour envoyer ou recevoir un message.

- Lors de l'envoi d'un message à un participant, les valeurs de tous les paramètres de propriété sont copiées ; une fois le message envoyé, le jeton continue le long du flux de séquence
- Lors de la réception d'un message, l'événement est déclenché lorsqu'un message est reçu.

Cet exemple illustre les fonctionnalités de Message Event. Nous allons d'abord créer le modèle BPMN, puis configurer BPSim et exécuter la simulation.

Créer Modèle BPMN



Séquence

Piscine 1

- Le jeton démarre à partir de StartEvent1
- À la réception du jeton, l'expéditeur (un événement de lancement de message intermédiaire) crée un message et copie les valeurs de propriété actuelles dans le message
- L'expéditeur envoie le *message* au participant « À » (Pool2, StartEvent2)
- L'expéditeur transmet le *jeton* le long de son flux de séquence, jusqu'au récepteur
- Le jeton attend chez le récepteur qu'un message arrive

Piscine 2

- StartEvent2 reçoit un message et démarre un jeton
- StartEvent2 copie les valeurs du message et les définit dans le jeton
- StartEvent2 transmet le jeton le long de son flux de séquence jusqu'à EndEvent3
- EndEvent3 crée un message et copie les valeurs de propriété actuelles dans le message
- EndEvent3 envoie le message au participant « À » (Pool1, Receiver)

Pool1 suite

- Le récepteur en attente est réveillé et les valeurs des propriétés sont mises à jour à partir du message entrant

Créer Diagramme

- Créer un diagramme de collaboration BPMN 2.0
- Sélectionnez l'option « Créer ce diagramme dans un nouveau CollaborationModel »
- Créez *Pool1* et *Pool2* en faisant glisser l'icône « Pool » de la boîte à outils sur le diagramme

Dans le pool 1

- Créer un événement Démarrer de type « Aucun », nommé *StartEvent1*
- Ajoutez une Flux séquence à l'événement intermédiaire cible de type « Message de lancement », appelé *expéditeur*
- Ajoutez une Flux séquence à l'activité cible de type « abstrait », appelée *Activité1*
- Ajoutez des flux Séquence à la cible :
  - Événement de fin de type « Aucun », appelé *EndEvent1*
  - Événement de fin de type « Aucun », appelé *EndEvent2*

Dans le pool 2

- Créer un événement Démarrer de type « Message », appelé *StartEvent2*
- Ajoutez une Flux séquence à l'activité cible de type « abstrait », appelée *Activité2*
- Ajoutez une Flux séquence à la Passerelle cible de type « Exclusive », appelée *Gateway2*
- Ajoutez des flux Séquence à la cible :
  - Activité de type 'abstrait', appelée *Activity3*
  - Activité de type « abstrait », appelée *Activity4*
- Ajoutez des flux Séquence de *l'activité 3* et *de l'activité 4* à la Passerelle cible de type « Exclusif », appelée *Gateway3*
- Ajoutez une Flux séquence à l'événement de fin de type « Message », appelé *EndEvent3*

Flux de messages

- Ajouter un flux de messages de *l'expéditeur* à *StartEvent2*
- Ajouter un flux de messages de *EndEvent3* à *Receiver*

Configurer BPSim

Afin de montrer la capacité du flux de messages à transporter des valeurs, nous créons une Paramètres Propriété 'M1' et modifions sa valeur dans chaque activité. Nous utilisons ensuite la valeur de 'M1' comme partie de l'expression de la condition de la Flux séquence .

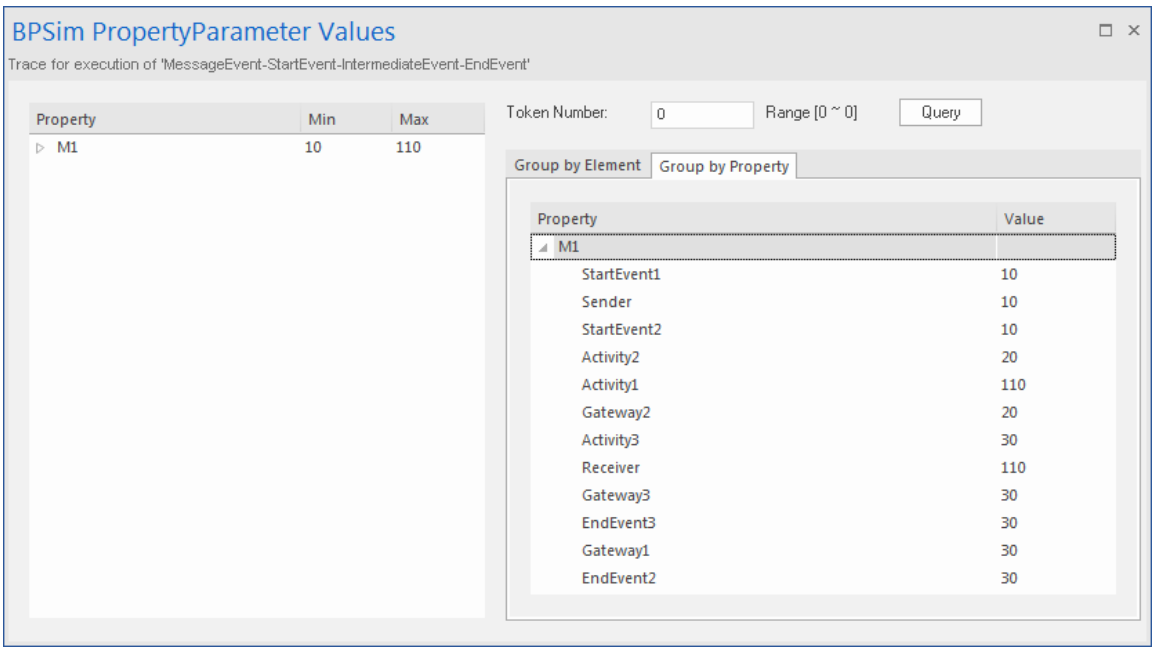
Tâche	Description
Créer un artefact et Paquetage	• Ouvrez la fenêtre Configurer BPSim ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager') • Cliquez sur le bouton  dans le champ « Sélectionner/Créer un artefact » et créez un artefact appelé « MessageEvent-StartEvent-IntermediateEvent-EndEvent » • Dans le champ « Sélectionner Paquetage », sélectionnez le Paquetage contenant le modèle Tous les éléments BPMN du modèle sont chargés dans la fenêtre Configurer BPSim.

Valeurs des propriétés	Nous allons donner à la Paramètres Propriété 'M1' une valeur initiale de 10 à *StartEvent1* . Ensuite, nous modifions la valeur au fur et à mesure que le jeton circule dans les processus et que la valeur est copiée entre les participants. Dans la liste des éléments à gauche le dialogue : • Développez le groupe « StartEvent », cliquez sur *StartEvent1* et sur l'onglet « Propriétés », et remplacez *Nouvelle propriété* par « M1 » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Constante » et « Numérique », et saisissez « 10 » dans le champ « Constante numérique » • Développez le groupe « Activité », cliquez sur *Activité1* et sur l'onglet « Propriétés », et remplacez *Nouvelle propriété* par « M1 » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », et saisissez « {M1} + 100 » dans le champ « Expression » • Cliquez sur *Activity2* et sur l'onglet « Propriétés », et remplacez *Nouvelle propriété* par « M1 » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », et saisissez « {M1} + 10 » dans le champ « Expression » • Cliquez sur *Activity3* et sur l'onglet « Propriétés », puis remplacez *Nouvelle propriété* par « M1 » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », puis saisissez « {M1} + 10 » dans le champ « Expression » • Cliquez sur *Activity4* et sur l'onglet « Propriétés », et remplacez *Nouvelle propriété* par « M1 » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », et saisissez « {M1} + 1 » dans le champ « Expression » *Conseil : Le format de* { PropertyName } *est une forme courte pratique de* getProperty(" PropertyName ") .
Paramètres de contrôle	Nous n'avons besoin que d'un seul jeton dans cette simulation pour évaluer le comportement du modèle. • Dans le groupe « StartEvent » développé, cliquez sur *StartEvent1* et sur l'onglet « Contrôle » ; cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Déclencheur Count », puis saisissez une « Valeur » de « 1 » Configurez maintenant les conditions des flux Séquence sortants des passerelles. Dans la liste des éléments à gauche de le dialogue , développez le groupe « Passerelle » : • Développez *Gateway1* , cliquez sur *EndEvent1* et sur l'onglet « Contrôle », puis cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », puis saisissez « {M1} >= 50 » dans le champ « Expression » • Cliquez sur *EndEvent2* et sur l'onglet « Contrôle », puis cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », puis saisissez « {M1} < 50 » dans le champ « Expression » • Développez *Gateway2* , cliquez sur *Activity3* et sur l'onglet « Contrôle », puis cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition » ; dans le champ « Valeur », cliquez sur le bouton ... et sélectionnez « Expression », puis saisissez « {M1} >= 15 » dans le champ « Expression »

	• Cliquez sur *Activity4* et sur l'onglet « Contrôle », puis cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Condition » ; dans le champ « Valeur », cliquez sur le bouton  et sélectionnez « Expression », puis saisissez « {M1} < 15 » dans le champ « Expression »

Exécuter Simulation

- Dans la barre d'outils de la boîte dialogue « Configurer BPSim », cliquez sur le bouton Exécuter ; la boîte dialogue « Contrôleur Simulation BPSim » s'affiche
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard » ; la simulation démarre
- Lorsque la simulation est terminée, cliquez sur le bouton  ; la dialogue « Paramètres Propriété Values » s'affiche, traçant les valeurs des propriétés pendant la simulation
- Dans le champ « Numéro de jeton », saisissez « 0 », puis cliquez sur le bouton Query et sur l'onglet « Grouper par propriété »



Analyse

Comme le « ProcessingTime » de *l'Activité1* a été défini comme une valeur de distribution, il s'avère que :

- valeur « M1 » de [Process1] après *Pool1.StartEvent1* est « 10 », comme prévu
- *[Process2] valeur 'M1' de *Pool2.StartEvent2* est '10' ; cette valeur est issue d'un message envoyé par *Pool1.Sender*

Il existe en fait deux « M1 » : *Process1.M1* et *Process2.M1*

- [Process2] *Pool2.Activity2* a augmenté *Process2.M1* de 10 ; *[Process2.M1 == 20]*
- [Process1] *Pool1.Activity1* a augmenté *Process1.M1* de 100 ; [*Process1.M1 == 110]*
- [Process2] Les expressions de condition sont évaluées ; comme « 20 > 15 », le jeton sera transmis à *l'activité 3 [Process2.M1 == 20]*
- [Process2] *Pool2.Activity3* a augmenté *Process2.M1* de 10 ; *[Process2.M1 == 30]*
- [Process1] *Pool1.Receiver* est atteint et en attente *[Process1.M1 == 110]*
- [Process2] *Pool2.Gateway3* sert de nœud de fusion et continue jusqu'à *EndEvent3 [Process2.M1 == 30]*

- *[Process1] *Pool1.Receiver est* réveillé par un message (transportant *M1 == 30*) et valeur de *Process1.M1* passe de 110 à 30
- [Process1] Les expressions de condition sont évaluées ; comme « 30 < 50 », le jeton sera transmis à *EndEvent2 [Process1.M1 == 30]*

Notes

- Les lignes marquées d'un astérisque (*) sont les effets des flux de messages
- L'ordre au sein d'un processus est défini ; cependant, l'ordre entre deux processus n'est pas toujours prévisible
- L'événement Throwing Message lance un autre processus ; le message Catching sert de synchronisation de thread

Signal Événements

Un événement Signal fournit la facilité de coupler de manière souple les « lanceurs » et les « capteurs » par une intégration de type publication-abonnement. Un « lanceur » diffusera un signal plutôt que de l'adresser à un processus particulier ; tout processus à l'écoute de cet événement particulier pourrait déclencheur une nouvelle instance à l'aide d'un événement Signal Démarrer .

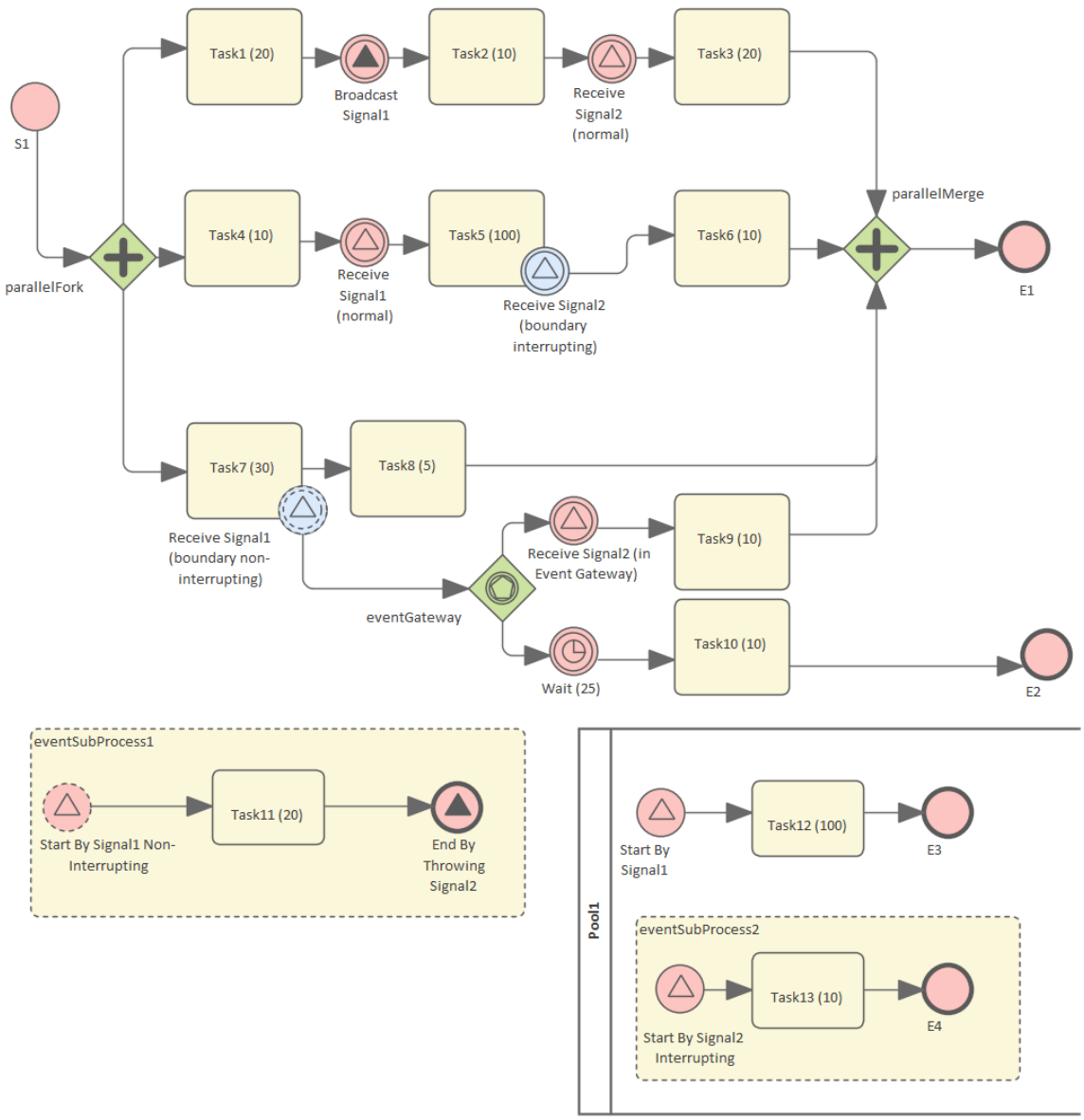
Un signal peut être lancé soit à partir d'un événement intermédiaire de lancement, soit à partir d'un événement de fin de lancement, et peut être attrapé dans un événement Démarrer ou un événement intermédiaire de capture (y compris un événement de signal bordure).

Dans cet exemple, nous démontrons ces Événements de signal et leur impact sur les lignes de vie des tâches, via les paramètres BPSim.

- Démarrer Signal Événement :
  - *Démarrer par Signal1* dans le processus de niveau supérieur (Pool1)
  - *Démarrer Par Signal2* Interruption dans le sous-processus événementiel *eventSubProcess2*
    - *Démarrer par Signal1* Non interrompu dans le sous-processus événementiel *eventSubprocess1*
- Lancer un événement de signal intermédiaire :
  - *Signal de diffusion1*
- Capture d'un événement de signal intermédiaire :
  - *Réception du signal 1 (normal)*
    - *Réception du signal 2 (normal)*
    - *Réception du signal 2 (bordure d'interruption)*
    - *Réception du signal 1 (bordure non interrompue)*
    - *Recevoir Signal2 (dans Event Passerelle)*
- Signal de fin d'événement :
  - *Terminez en lançant le signal 2*

Créer Modèle BPMN

Afin de démontrer la capacité de communiquer entre les processus via un événement de signal, nous créons un modèle de collaboration avec un pool principal et un processus dans un autre pool (*Pool1*).



Créer la collaboration et le processus principal

Créez un nouveau diagramme de collaboration BPMN2.0 appelé *CollaborationForTestingSignalEvents* , (choisissez l'option 'Créer ce diagramme dans un nouveau Modèle de collaboration'). Cliquez-droit sur le nom diagramme dans la fenêtre Navigateur et sélectionnez l'option 'Encapsuler le processus'.

Un Pool *PoolMain* et un processus *BusinessProcess_PoolMain* sont créés, et ces étiquettes sont définies avec les valeurs automatiques :

- *CollaborationForTestingSignalEvents.mainPool* est défini sur *PoolMain*
- *PoolMain.processRef* est défini sur *BusinessProcess_PoolMain*

Créer les éléments du processus principal

Créer un Démarrer Event *S1* et ajouter une Flux séquence à une Fork Parallel Passerelle *parallelFork*

Ajoutez des flux Séquence à :

- Une tâche abstraite *Tâche 1 (20)* puis ajoutez cette chaîne de flux Séquence :
  - Vers un événement de signal intermédiaire de lancement *Signal de diffusion1*
  - Ensuite, vers une tâche abstraite *Tâche 2 (10)*
  - Ensuite, vers un événement de signal intermédiaire de capture , *recevoir le signal 2 (normal)*

- Ensuite, une tâche abstraite *Tâche 3 (20)*
- Ensuite vers une Passerelle Parallèle Merge *parallelMerge*
- Puis vers un événement final *E1*

- Une tâche abstraite *Tâche 4 (10)* puis ajoutez cette chaîne de flux Séquence :
  - Vers un événement de signal intermédiaire de capture *Recevoir Signal1 (normal)*
  - Ensuite à une Tâche Abstraite *Tâche 5 (100),* sur laquelle vous créez une Bordure Interrupting Catching Événement de signal intermédiaire *Réception du signal 2 (bordure interrompante)*
  - Ensuite, une tâche abstraite *Tâche 6 (10)*
  - Ensuite, à la précédente Merge Parallel Passerelle *parallelMerge*
- Une tâche abstraite *Tâche 7 (30)* , puis ajoutez cette chaîne de flux Séquence :
  - À une tâche abstraite *Tâche 8 (5)*
  - Ensuite, à la précédente Merge Parallel Passerelle *parallelMerge*

Dans *la tâche 7 (30)* , créez un événement de réception de signal intermédiaire de capture Bordure non interrompue *Signal1 (bordure non interrompue)* . Ajoutez une Flux séquence à une Passerelle d'événements *eventGateway* , puis ajoutez à cela des flux Séquence à :

- Un événement de signal intermédiaire de capture *reçoit le signal 2 (dans l'événement Passerelle),* puis cette chaîne de flux Séquence :
  - À une tâche abstraite *Tâche 9 (10)*
    - Ensuite, à la précédente Merge Parallel Passerelle *parallelMerge*
- Un événement de temporisation intermédiaire de capture *Wait (25)* , puis cette chaîne de flux Séquence :
  - À une tâche abstraite *Tâche 10 (10)*
  - Puis vers un événement final *E2*

Créer un sous-processus d'événement (déclenché par un événement Démarrer Signal non interrompant) dans le processus principal

- Créez un événement d'activité *SubProcess1* et, dans sa dialogue « Propriétés », définissez le champ « Type » sur *subProcess* et modifiez l'attribut « triggeredByEvent » sur *true*
- Dans *eventSubProcess1* créez un événement Démarrer *Démarrer By Signal1 Non Interrupting* et, dans sa dialogue ' Propriétés ', définissez le champ ' Type ' sur *Event Sub-Process Non-Interrupting > Signal*
- Ajouter une Flux séquence à une cible Tâche abstraite Tâche *11 (20)*
- Ajoutez une Flux séquence à un événement de fin cible *End By Throwing Signal2* et, dans la dialogue « Propriétés » de l'élément, définissez le champ « Type » sur *Signal*

Créer un autre processus

- Depuis la boîte à outils, faites glisser et déposez l'icône « Pool » sur le diagramme et nommez l'élément *Pool1*
- Cliquez-droit sur *Pool1* dans la fenêtre Navigateur et sélectionnez l'option 'Encapsuler le processus' ; un processus *BusinessProcess_Pool1* est créé et l' étiquette 'Pool1.processRef' est définie sur *BusinessProcess_Pool1*

Créer le processus principal pour *Pool1*

- Créer un événement Signal Démarrer *Démarrer par Signal1*
- Ajouter une Flux séquence à une cible Tâche abstraite Tâche *12 (100)*
- Ajouter une Flux séquence à un événement de fin cible *E3*

Créer un sous-processus d'événement pour interrompre *Pool1*

- Créez un événement d'activité *SubProcess1* et, dans la dialogue « Propriétés », définissez le champ « Type » sur *subProcess* ; modifiez l'attribut « triggeredByEvent » sur *true*
- Dans *eventSubProcess2,* créez un événement Démarrer *Démarrer By Signal2 Interrupting* et, dans la dialogue « Propriétés », définissez le champ « Type » sur *Event Sub-Process Interrupting > Signal*
- Ajouter une Flux séquence à une cible Tâche abstraite Tâche *13 (10)*
- Ajouter une Flux séquence à un événement de fin cible *E4*

Créez les éléments de signal BPMN2.0 et configurez-les pour Événements de signal

Dans la boîte à outils BPMN 2.0, développez la page « BPMN 2.0 - Types » et faites glisser l'icône « Signal » sur le

diagramme ; nommez l'élément *Signal1* . Faites glisser à nouveau l'icône sur le diagramme pour créer *Signal2* . Il s'agit d'éléments racines (qui peuvent être utilisés par tous les processus) et ils seront donc créés directement sous le modèle Paquetage .

Double-cliquez sur chacun des éléments de l'événement Signal et, dans le champ « Valeur » de l' étiquette « signalRef », cliquez sur le bouton ... et accédez à l'élément Signal approprié.

Conseils : Alternativement, vous pouvez faire glisser l'élément Signal depuis la fenêtre Navigateur et le déposer sur les éléments Événement dans le diagramme ; un menu contextuel s'affiche, à partir duquel vous sélectionnez l'option 'set signalRef'.

- Régler signalRef sur « Signal1 » sur :
  - *Signal de diffusion1*
    - *Démarrer par Signal1* dans le processus de niveau supérieur (*Pool1*)
  - *Démarrer par Signal1 sans interruption* dans le sous-processus Event *eventSubprocess1*
    - *Réception du signal 1 (normal)*
    - *Réception du signal 1 (bordure non interrompue)*
- Régler signalRef sur « Signal2 » sur :
  - *Démarrer par Signal2 Interrompre* dans le sous-processus Event *eventSubProcess2*
    - *Réception du signal 2 (normal)*
    - *Recevoir le signal 2 (bordure d'interruption)*
    - *Recevoir Signal2 (dans Event Passerelle)*

Configurer BPSim

Dans cette section, nous créons l'artefact de configuration, spécifions le modèle Paquetage et configurons les valeurs des paramètres de chaque élément.

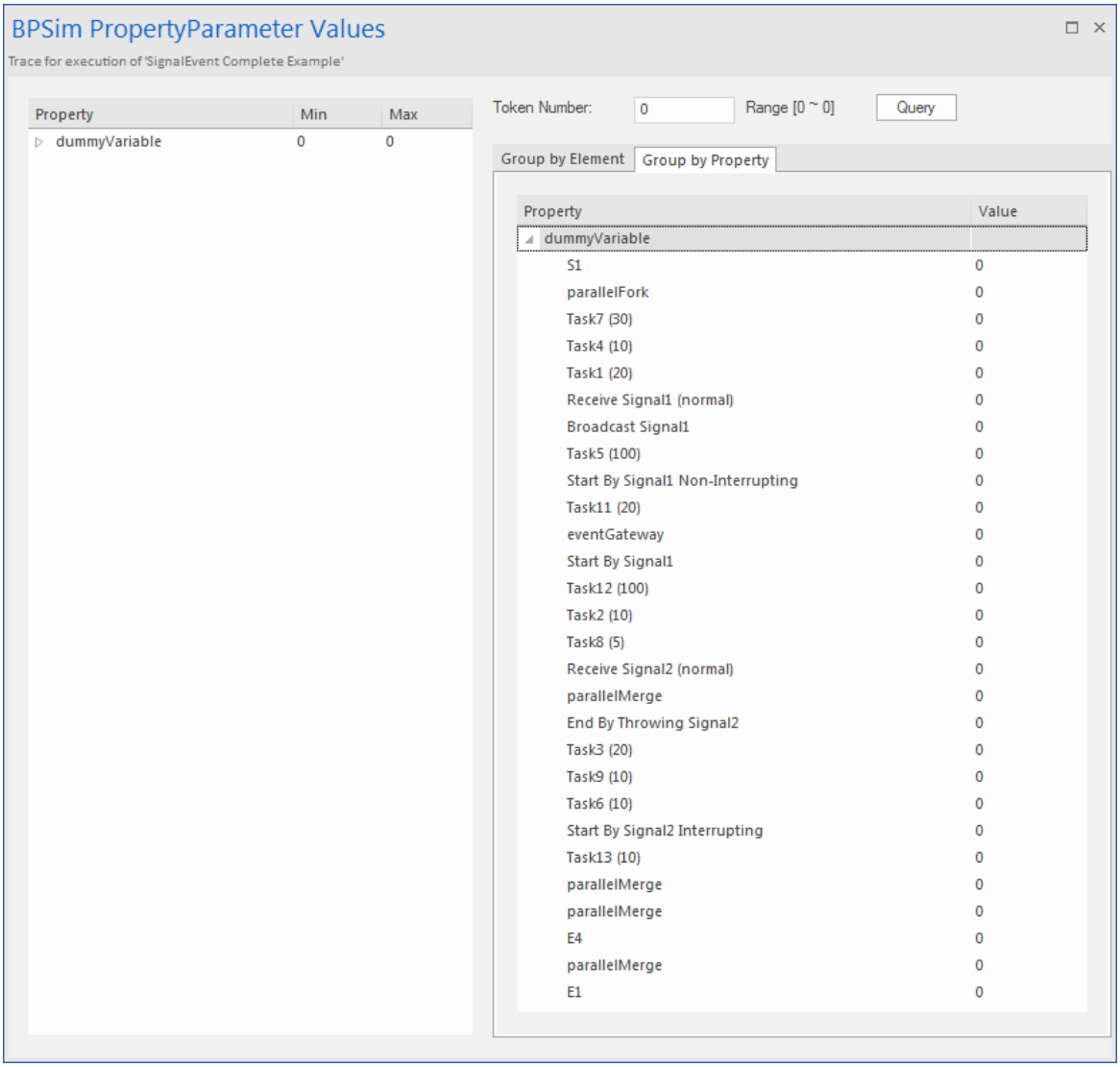
La configuration est assez simple car aucun des Événements Signal ne nécessite de configuration BPSim. Il suffit de paramétrer le temps de traitement des tâches pour pouvoir observer comment les processus, les threads et les tâches sont démarrés et interrompus.

Tâche	Description
Configurer la configuration	• Ouvrez la fenêtre Configurer BPSim ('Simulate > Analyse de Processus > BPSim > Open BPSim Manager') • Créez un artefact nommé « Exemple complet de SignalEvent » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton ... et sélectionnez son Paquetage parent et cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer. et le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la fenêtre Configurer BPSim.
Événements sans signal	• Dans la liste des éléments à gauche de le dialogue , développez le groupe 'StartEvent', puis cliquez sur *S1* et sur l'onglet 'Contrôle' ; cliquez sur la flèche déroulante 'Nouveau paramètre' et sélectionnez ' Déclencheur Count', puis saisissez '1' dans le champ 'Valeur' • Développez le groupe « IntermediateEvent », puis cliquez sur *Wait (25)* et sur l'onglet « Control » ; cliquez sur la flèche déroulante « New Parameter » et sélectionnez « InterTriggerTimer », puis cliquez sur le bouton ... dans le champ « Value » ; sélectionnez « Constant » et « Numeric », et saisissez « 25 » dans le champ « Constant Numeric » et « seconds » dans le champ « TimeUnit »
Variable fictive pour le processus	Le contrôleur de simulation affiche une liste indiquant le nombre de jetons d'exécution pour chaque élément. Par exemple, 4 jetons ont dépassé l'élément

	Passerelle *parallelMerge* dans une simulation. Cela est très utile pour certaines statistiques et analyses. Cependant, cela n'indique pas QUAND *parallelMerge* a été traversé pendant la simulation. Afin d'obtenir la trace exacte d'un seul jeton, nous utilisons l'utilitaire de trace de propriété, qui s'appuie sur des paramètres de propriété. Nous créons donc un paramètre factice. Dans la dialogue « Configuration BPSim », développez le groupe « BusinessProcess ». • Cliquez sur *BusinessProcess_Main* et sur l'onglet « Propriétés », et remplacez *New Property* par *dummyVariable ;* dans le champ « Value », cliquez sur le bouton  et sur « Constant » et « Numeric », et dans le champ « Constant Numeric », saisissez « 0 » • Cliquez sur *BusinessProcess_Pool1* et effectuez exactement les mêmes actions que pour *BusinessProcess_Main*
Temps de traitement des tâches	Développez le groupe « Activité » et pour chaque élément de tâche répertorié ici : sélectionnez l'onglet « Heure », cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Temps de traitement », puis cliquez sur le bouton  dans la colonne « Valeur », sélectionnez « Constante » et « Numérique », saisissez la valeur comme indiqué dans le champ « Constante numérique » et sélectionnez « secondes » dans le champ « Unité de temps ». • Tâche 1 (20) : 20 secondes • Tâche 2 (10) : 10 secondes • Tâche 3 (20) : 20 secondes • Tâche 4 (10) : 10 secondes • Tâche 5 (100) : 100 secondes • Tâche 6 (10) : 10 secondes • Tâche 7 (30) : 30 secondes • Tâche 8 (5) : 5 secondes • Tâche 9 (10) : 10 secondes • Tâche 10 (10) : 10 secondes • Tâche 11 (20) : 20 secondes • Tâche 12 (100) : 100 secondes • Tâche 13 (10) : 10 secondes

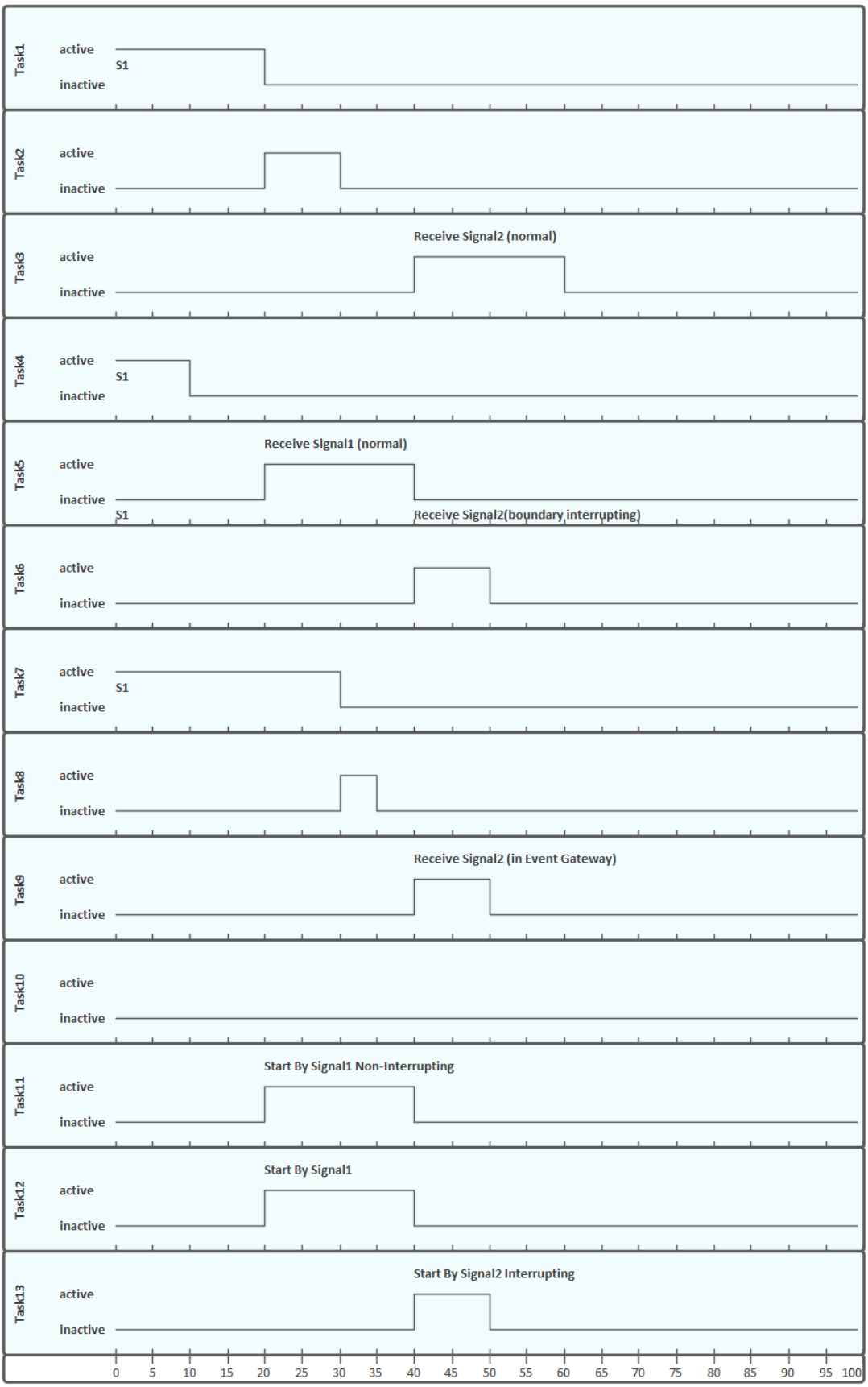
Exécuter Simulation

- Dans la barre d'outils de la dialogue « Configurer BPSim », cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur Simulation BPSim »
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »
- Après la simulation, cliquez sur le bouton  dans la barre d'outils pour afficher la dialogue « Valeurs PropertyParameter BPSim »
- Cliquez sur le bouton Query et sur l'onglet « Grouper par propriété », puis développez « dummyVariable »



Analyse

À partir des résultats directs de la simulation, il n'est peut-être pas évident de savoir ce qui s'est passé ; cependant, si nous traçons la ligne de vie de chaque tâche, cela devient assez clair.



- *Tâche 1, Tâche 4* et *Tâche 7* démarrées en parallèle

- *La tâche 2* a commencé immédiatement après la fin de *la tâche 1* (sans s'arrêter à l'événement de lancer)
- À 20 secondes, *Signal1* a été diffusé par le *Signal1 de diffusion* d'événement intermédiaire de lancer et :
  - *Le signal de réception 1 (normal)* a été activé et *la tâche 5* a démarré
  - *Démarrer By Signal1 Non-Interrupting* a été activé et *Task11* dans *eventSubProcess1* a démarré
  - *Démarrer By Signal1* a été activé et *Task12* dans *Pool1 a* démarré
- À 40 secondes, *Signal2* a été diffusé par l'événement de fin *End By Throwing Signal2* et :
  - *Le signal de réception 2 (normal)* a été activé et *la tâche 3* a démarré
  - *La tâche 5* a été interrompue et *la tâche 6* a démarré
  - *Le signal de réception 2 (dans Event Passerelle)* a été activé et *la tâche 9* a démarré
  - *Démarrer By Signal2 L'interruption* a été activée, et :
    > Le processus principal de *Pool1* a été interrompu et *Task12* s'est arrêté

  > *La tâche 13* dans *eventSubProcess2* a démarré
- L' *événement EventSubProcess2* dans *BusinessProcess_Pool1* s'est terminé lorsque *E4* a été atteint à 50 secondes
- Le *BusinessProcess_MainPool* s'est terminé lorsque *E1* a été atteint à 60 secondes
- L'événement de temporisation intermédiaire *Wait (25)* n'a pas été activé car l'événement de signal dans la Passerelle a été activé en premier ; par conséquent, *la tâche 10* n'a jamais été démarrée

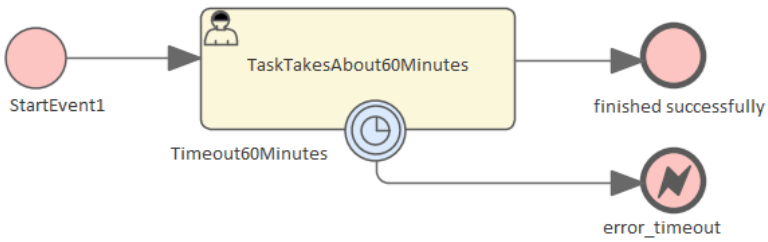
Note : Le temps d'exécution réel de chaque tâche peut être observé à partir de l'élément BPSimReport généré, en :

1. Double-cliquez sur l'élément <<BPSimReport>>.
2. Extension du groupe « Heure ».
3. Extension de l'élément de tâche.
4. Vérification du « Temps total dans la tâche ».

Par exemple, pour l'élément *Task5 (100)* , bien que nous ayons défini son processingTime à 100 secondes, le **temps total dans la tâche** était de 20 secondes, ce qui a été interrompu par *Receive Signal2 (bordure d'interruption)* à 20 secondes.

Événement de minuterie - Bordure

Créer Modèle BPMN



- Créer un événement Démarrer *StartEvent1*
- Ajouter une Flux séquence à une tâche utilisateur cibleTâchePrend *environ 60 minutes*
- L'ajout d'une Flux séquence à un événement de fin cible *s'est terminé avec succès*
- Créez un événement intermédiaire en faisant glisser l'icône depuis la boîte à outils et en la déposant sur *TaskTakesAbout60Minutes ;* sélectionnez « Edge-Mounted » et « Timer » dans les menus automatiques et appelez l'élément *Timeout60Minutes*
- Ajouter une Flux séquence à un événement de fin cible (erreur) *error_timeout*

Configurer BPSim

Dans cette section, nous créons l'artefact de configuration, identifions le Paquetage parent et définissons les valeurs des paramètres de chaque élément.

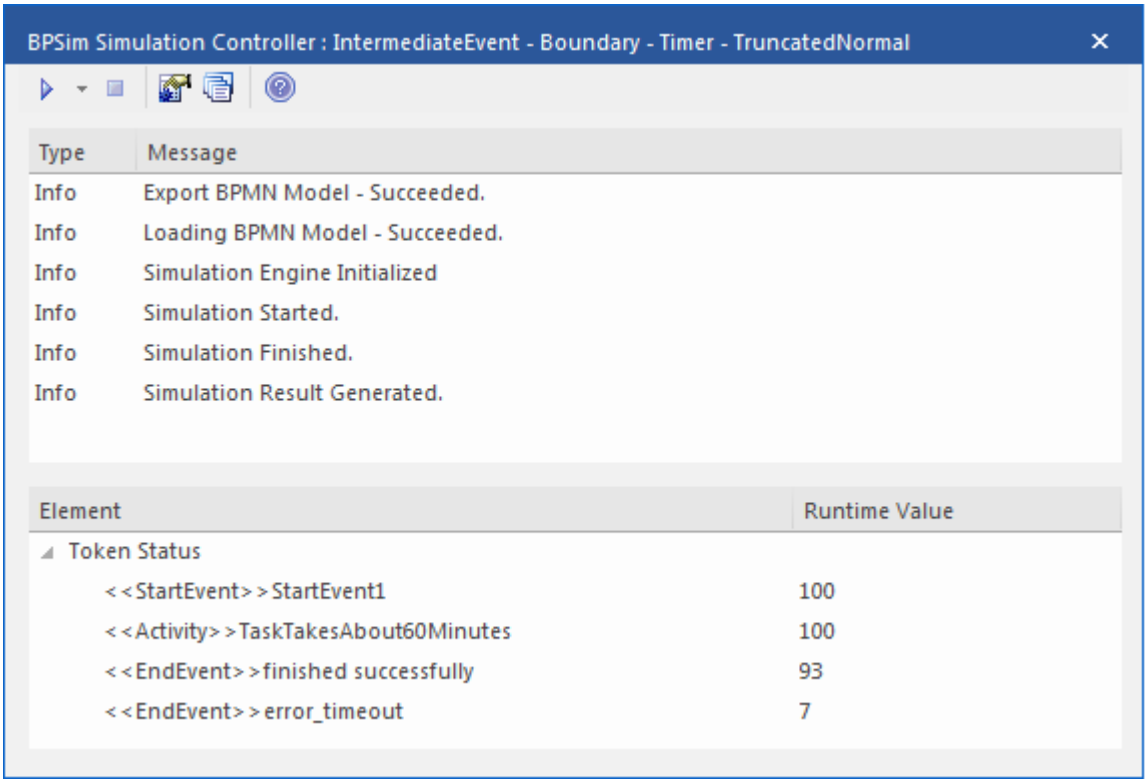
Objets	Action
Créer un artefact et Paquetage	• Ouvrez la dialogue « Configurer BPSim » ('Simuler > Analyse de Processus > BPSim > Ouvrir le gestionnaire BPSim') • Créez un artefact nommé « IntermediateEvent - Bordure - Timer - TruncatedNormal » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton ... et sélectionnez son parent Paquetage , cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer et sur le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la dialogue « Configurer BPSim ».
StartEvent1	Dans la liste des éléments à gauche de le dialogue , développez le groupe « StartEvent » et cliquez sur *StartEvent1.* • Cliquez sur l'onglet « Contrôle » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « TriggerCount » • Dans le champ « Valeur », saisissez « 100 »
La tâche dure environ 60 minutes	Dans la liste des éléments sur la gauche de le dialogue , développez le groupe « Activité » et cliquez sur *TaskTakesAbout60Minutes.* • Cliquez sur l'onglet « Heure »

	• Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « ProcessingTime » • Dans le champ « Valeur », cliquez sur le bouton  et sélectionnez « Distribution » et « TruncatedNormal ». • Dans le champ « Moyenne », saisissez « 50 » • Dans le champ « Écart type », saisissez « 10 » • Dans le champ « Min », saisissez « 0 » • Dans le champ « Max », saisissez « 1 000 » • Cliquez sur le bouton OK
Délai d'attente60 minutes	Dans la liste des éléments à gauche de le dialogue , développez le groupe « IntermediateEvent » et cliquez sur *Timeout60Minutes.* • Cliquez sur l'onglet « Contrôle » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « InterTriggerTimer » • Définissez la valeur sur « 000:000:000 001:00:00 » (soit 1 heure)

Exécuter Simulation

- Dans la barre d'outils de la fenêtre Configurer BPSim, cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur Simulation BPSim »
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »
- Après la simulation, cliquez sur le bouton  dans la barre d'outils pour afficher la dialogue « Valeurs PropertyParameter BPSim »
- Cliquez sur le bouton Query et sur l'onglet « Grouper par propriété », puis développez « dummyVariable »

En simulation, on obtient ce résultat :



Analyse

Étant donné que le ProcessingTime de *TaskTakesAbout60Minutes* a été défini comme une valeur de distribution, il s'avère que :

- 93 sur 100 ont terminé en 1 heure, donc le flux normal pour *terminer avec succès* prend effet
- 7 sur 100 terminés en plus d'une heure, donc le flux d'exception vers *error_timeout* prend effet

Autres Configurations

Dans le dossier d'exemple, il existe deux autres artefacts Simulation Processus Métier qui définissent le ProcessingTime comme une valeur constante de 50 minutes et 80 minutes, les autres paramètres restent les mêmes.

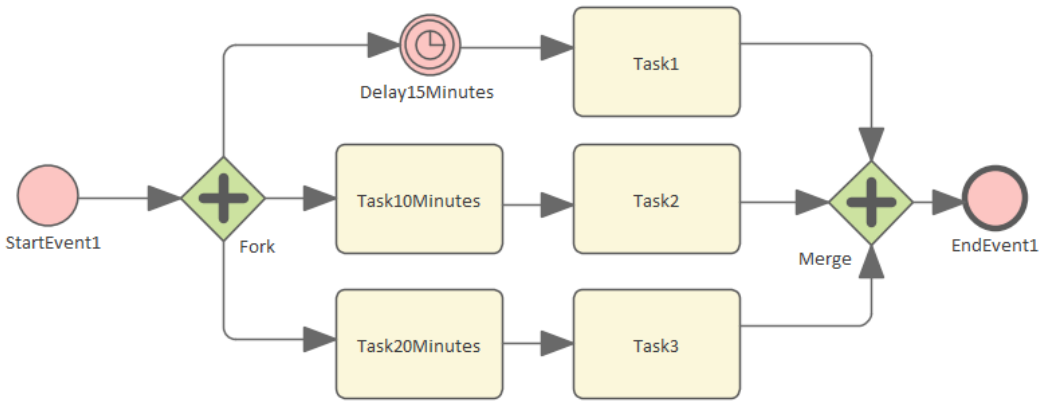
Exécuter une simulation sur ces deux Artefacts :

- Le temps de traitement configuré sur 50 minutes se termine toujours normalement
- Le temps de traitement configuré sur 80 minutes se termine toujours par un flux d'exception

Événement de minuterie - Événement intermédiaire autonome

Lorsqu'un événement intermédiaire de minuterie est utilisé dans le flux de séquence normal en tant qu'élément autonome, il agit comme un mécanisme de retard.

Créer Modèle BPMN



- Créez un événement Démarrer appelé *StartEvent1*
- Ajoutez une Flux séquence à une Passerelle parallèle cible appelée *Fork*
- Ajoutez des flux Séquence à :
  - Un événement intermédiaire de temporisation autonome appelé *Delay15Minutes* , et à partir de là une Flux séquence
  à une activité appelée *Tâche1*
    - Une activité appelée *Task10Minutes* , et à partir de là une Flux séquence vers une activité appelée *Task2*
    - Une activité appelée *Task20Minutes* , et à partir de là une Flux séquence vers une activité appelée *Task3*
- À partir de *Task1* , *Task2* et *Task3,* créez des flux Séquence vers une Passerelle parallèle de fusion appelée *Merge*
- Ajoutez une Flux séquence à un événement final cible appelé *EndEvent1*

Configurer BPSim

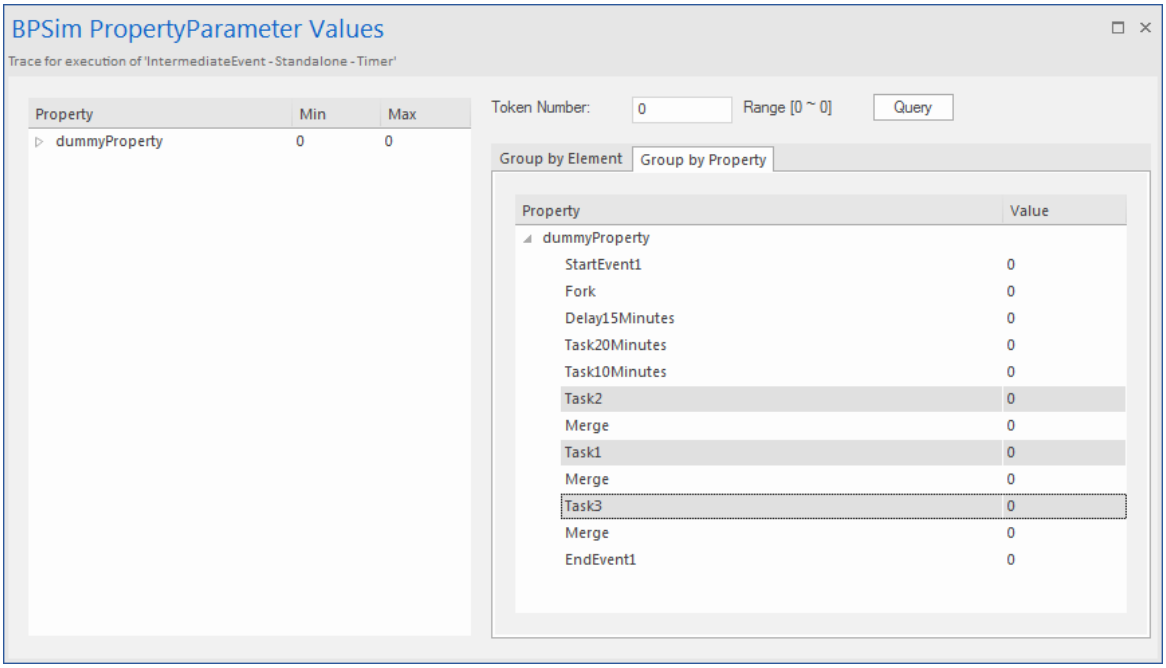
Dans cette section, nous créons l'artefact de configuration, spécifions le modèle Paquetage et configurons les valeurs des paramètres de chaque élément.

Object	Action
Créer un artefact et Paquetage	• Ouvrez la dialogue « Configurer BPSim » ('Simuler > Analyse de Processus > BPSim > Ouvrir le gestionnaire BPSim') • Créez un artefact nommé « IntermediateEvent - Standalone - Timer » (dans le champ « Sélectionner/Créer un artefact », cliquez sur le bouton ... et sélectionnez son Paquetage parent et cliquez sur le bouton Ajouter un nouveau, puis saisissez le nom de l'élément et cliquez sur le bouton Enregistrer et sur le bouton OK) Ensuite, tous les éléments BPMN seront chargés dans la dialogue « Configurer

	BPSim ».
StartEvent1	• Dans la liste des éléments à gauche de le dialogue , développez le groupe « StartEvent », puis cliquez sur *StartEvent1* et sur l'onglet « Contrôle » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « Déclencheur Count », puis saisissez « 1 » dans le champ « Valeur » • Cliquez sur l'onglet ' Propriétés ' • Remplacez *la nouvelle propriété* par *dummyProperty ;* dans le champ « Valeur », cliquez sur le bouton  et sur « Constante » et « Numérique », et dans le champ « Constante numérique », saisissez « 0 » Avec cette propriété, la dialogue « Trace de propriété » pourra afficher la séquence des flux d'éléments pendant la simulation.
Retard15Minutes	• Dans la liste des éléments à gauche de le dialogue , développez le groupe « IntermediateEvent », puis cliquez sur *Delay15Minutes* et sur l'onglet « Control » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « InterTriggerTimer », puis définissez le champ « Valeur » sur 15 minutes (« 000:000:000 000:15:00 »)
Tâche10 minutes	• Dans la liste des éléments à gauche de le dialogue , développez le groupe « Activité », puis cliquez sur *Tâche10Minutes* et sur l'onglet « Heure » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « ProcessingTime », puis définissez le champ « Valeur » sur 10 minutes (« 000:000:000 000:10:00 »)
Tâche20 minutes	• Dans la liste des éléments à gauche de le dialogue , développez le groupe « Activité », puis cliquez sur *Tâche20Minutes* et sur l'onglet « Heure » • Cliquez sur la flèche déroulante « Nouveau paramètre » et sélectionnez « ProcessingTime », puis définissez le champ « Valeur » sur 20 minutes (« 000:000:000 000:20:00 »)

Exécuter Simulation

- Dans la barre d'outils de la dialogue « Configurer BPSim », cliquez sur l'icône « Exécuter » pour ouvrir la dialogue « Contrôleur Simulation BPSim »
- Cliquez sur la flèche déroulante de l'icône « Exécuter » et sélectionnez « Simulation standard »
- Après la simulation, cliquez sur le bouton  dans la barre d'outils pour afficher la dialogue « Valeurs PropertyParameter BPSim »
- Cliquez sur le bouton Query et sur l'onglet « Grouper par propriété »



Analyse

La Passerelle parallèle *Fork* activera simultanément les flux Séquence sortants (l'ordre n'est pas défini et n'a pas d'importance). Cependant, nous nous attendons à ce que l'ordre des tâches soit exactement :

- *Tâche 2*
- *Tâche 1*
- *Tâche 3*

Cet ordre est déterminé par les paramètres BPSim définis sur deux des activités (ProcessingTime) et l'événement intermédiaire du minuteur (InterTriggerTimer). La séquence affichée dans la dialogue « Valeurs PropertyParameter BPSim » confirme que *Task2* précède *Task1* , qui précède *Task3* .

Simulation du processus de peinture des murs (Activité d'Appel)

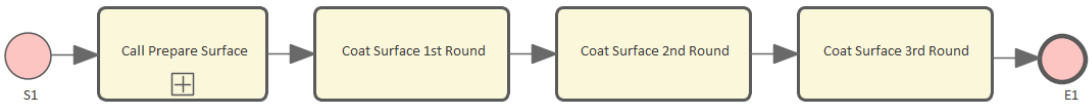
Il s'agit d'un exemple simple pour simuler le processus de peinture d'un mur. Nous définissons le processus principal comme la préparation de la surface, puis sa peinture trois fois. La préparation de la surface est ensuite divisée en tâches telles que le ponçage et le nettoyage.

Nous supposons que l'application de chacune des trois couches de peinture est le même processus, sauf que le temps passé aléatoirement sur chaque couche peut être différent.

Créer Modèle BPMN

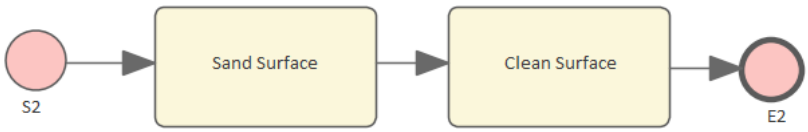
Cette simulation fonctionne sur deux processus.

Le processus principal - Processus de peinture murale



1. Créez un événement Démarrer appelé *S1*.
2. Ajoutez une Flux séquence à un callProcessActivity cible appelé *Call Prepare Surface.*
3. Ajoutez une Flux séquence à un appel cible GlobalTaskActivity appelé *Coat Surface 1st Round.*
4. Ajoutez une Flux séquence à un appel cible GlobalTaskActivity appelé *Coat Surface 2nd Round.*
5. Ajoutez une Flux séquence à un appel cible GlobalTaskActivity appelé *Coat Surface 3rd Round.*
6. Ajoutez une Flux séquence à un événement de fin cible appelé *E1*.

Le processus réutilisé - Processus de préparation de surface



1. Créez un événement Démarrer appelé *S2.*
2. Ajoutez une Flux séquence à une tâche abstraite cible appelée *Surface de sable.*
3. Ajoutez une Flux séquence à une tâche abstraite cible appelée *Nettoyer la surface.*
4. Ajoutez une Flux séquence à un événement final cible appelé *E2.*

Définir une tâche globale et réutiliser le processus pour appeler des activités

1. Créez une activité de tâche globale appelée *Coat Surface.*
2. Double-cliquez sur chacune des *surfaces de revêtement 1er tour, 2e tour* et *3e tour* , puis définissez l' étiquette « calledActivityRef » sur *Surface de revêtement.*
   Conseil : Vous pouvez également faire glisser la tâche globale 'Coat Surface' depuis la fenêtre Navigateur et la déposer sur l'élément Call Activity, en cliquant sur l'option 'Set calledActivityRef' dans le menu contextuel.
3. Double-cliquez sur *Appeler Préparer la surface* et définissez l' étiquette « calledActivityRef » sur *Préparer le processus de surface.*
   Conseil : Vous pouvez également faire glisser le processus 'Préparer le processus de surface' depuis la fenêtre Navigateur et le déposer sur l'élément Appeler l'activité, en cliquant sur l'option 'Définir calledActivityRef' dans le menu contextuel.

Configurer BPSim

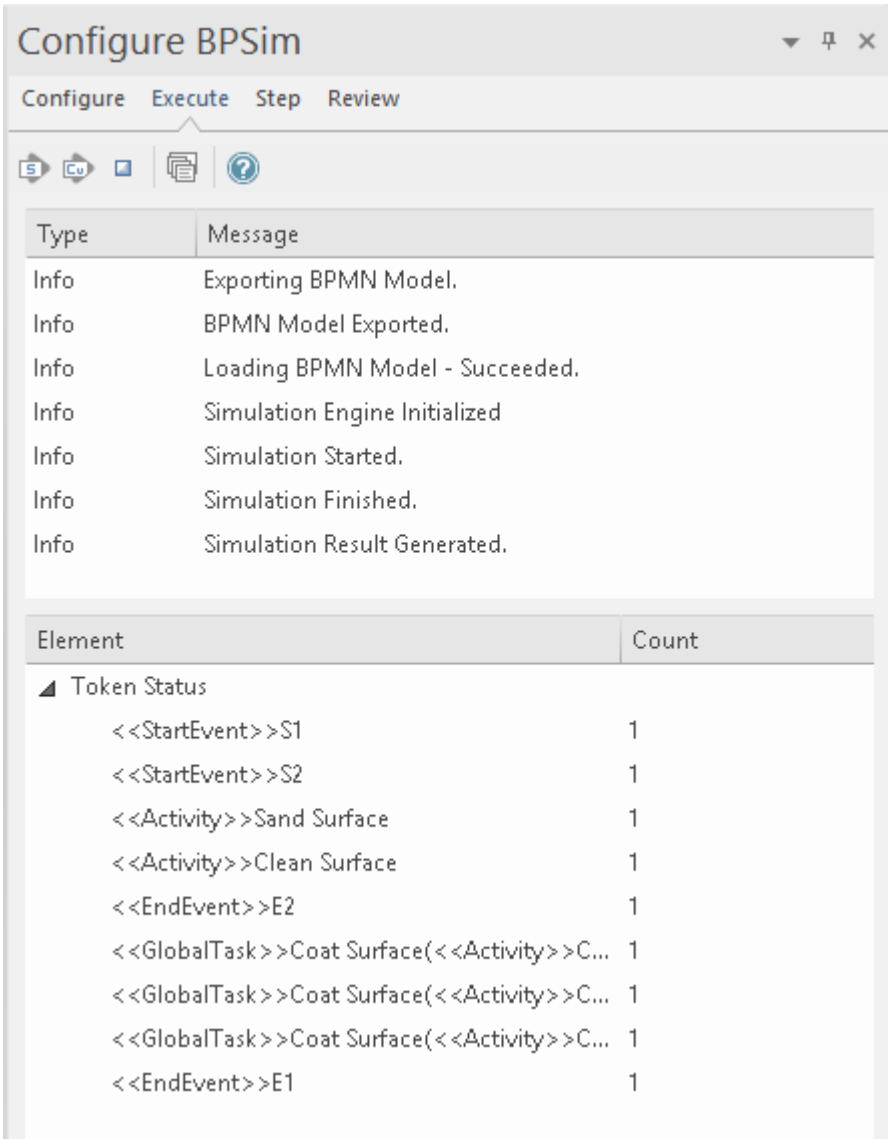
1. Ouvrez la dialogue « Configurer BPSim » ('Simuler > Analyse de Processus > BPSim > Ouvrir BPSim Manager').
2. Cliquez sur l'icône et créez un artefact Processus Métier Simulation nommé *Paint Wall Simulation* .
3. Cliquez sur l'icône et sélectionnez le Paquetage contenant le modèle BPMN 2.0 correspondant.

Object	Activité
Temps de mise à l'échelle fixe	1. Ouvrez le diagramme « Préparer le processus de surface ». 2. Cliquez sur la *surface de sable* d'activité et, dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de temps appelé « Temps de traitement ». 3. Dans le champ « Valeurs », modifiez le paramètre sur 0 00:30:00 (soit 30 minutes). 4. Cliquez sur la *surface de nettoyage* de l'activité et, dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de temps appelé « Temps de traitement ». 5. Dans le champ « Valeurs », modifiez le paramètre sur 0 00:10:00 (soit 10 minutes). 6. Cliquez sur l'icône .
Durée de revêtement aléatoire	1. Dans la fenêtre Navigateur , cliquez sur la *surface de revêtement de l'activité de tâche globale.* 2. Dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de temps appelé « Temps de traitement ». 3. Dans le champ « Valeurs », cliquez sur le bouton , puis dans la dialogue des paramètres, cliquez sur l'onglet « Distribution » et sur « Poisson ». 4. Dans le champ « Moyenne », saisissez « 10 », puis cliquez sur le bouton OK . 5. Cliquez sur l'icône . Avec ce paramètre, la valeur moyenne des nombres aléatoires générés par la distribution de Poisson est de 10. Si vous préférez, vous pouvez choisir d'autres types de distribution.
Nombre de déclencheurs sur S1	Sur le diagramme 'Paint Wall Process' cliquez sur l'événement Démarrer *S1*. 1. Dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de contrôle appelé « TriggerCount ». 2. Dans le champ « Valeurs », saisissez « 1 ». 3. Cliquez sur l'icône .

Exécuter Simulation

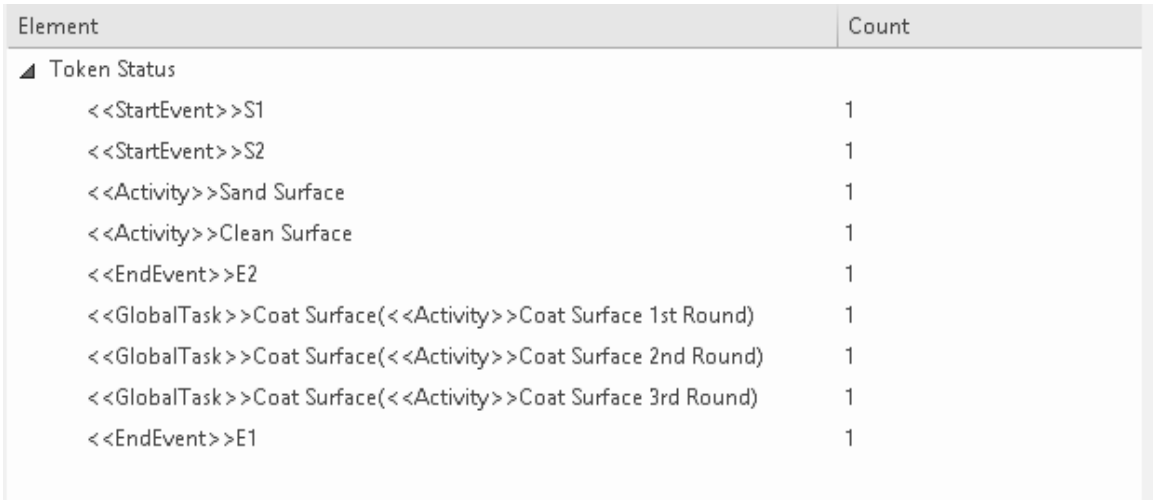
1. Dans l'onglet « Exécuter » de la fenêtre Configurer BPSim, cliquez sur l'icône .

Une fois la simulation terminée, elle fournit un résultat similaire à celui-ci :



Analyse de flux

Pour le seul token démarré sur *S1* , nous pouvons voir depuis l'onglet « Exécuter » de la fenêtre Configurer BPSim comment le flux se développe :

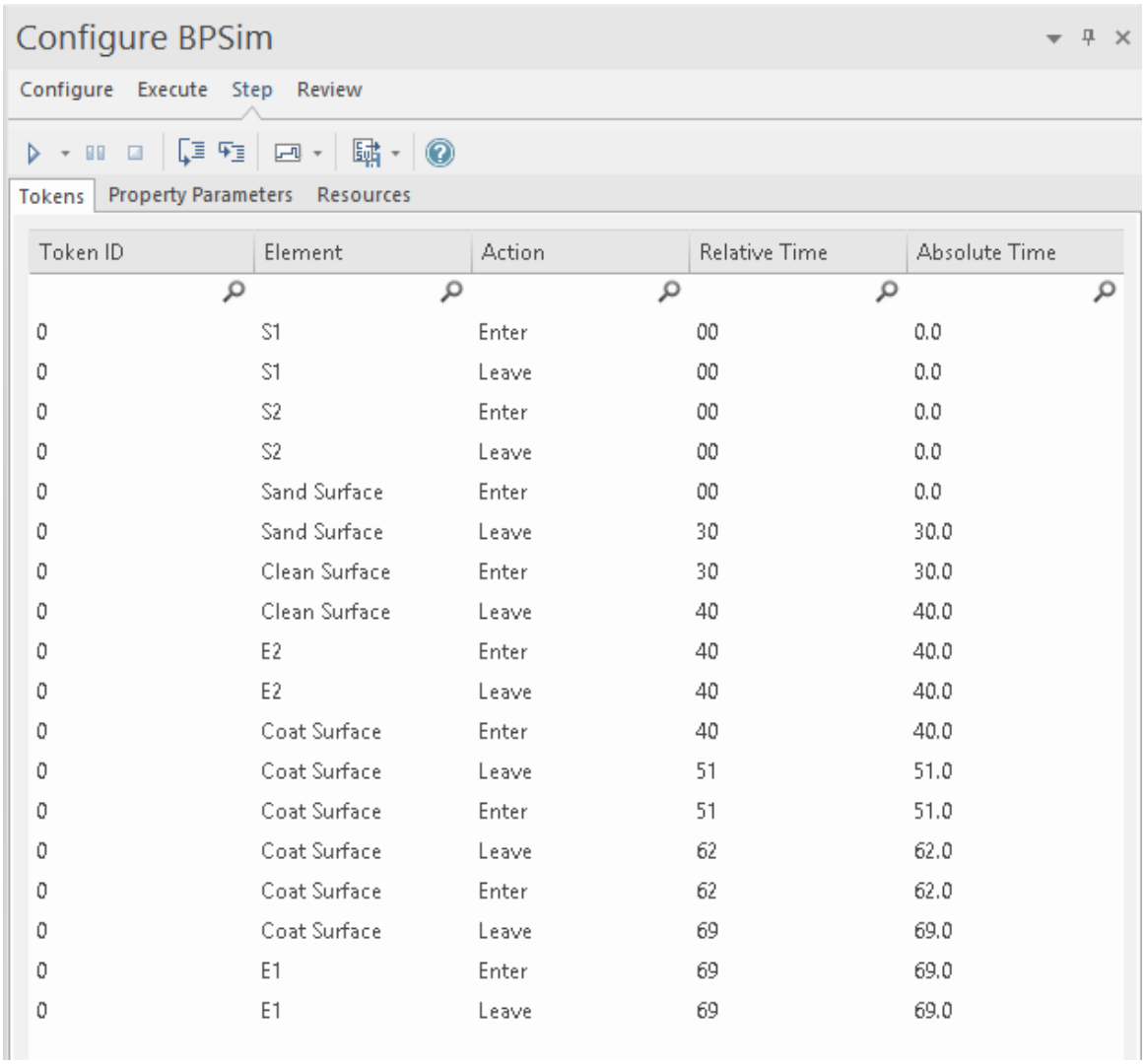


- En atteignant le callProcessActivity, le processus appelé est activé ; nous avons donc *S2 ~ E2*

- Lorsqu'un callGlobalTaskActivity est atteint, la tâche globale appelée est activée - la notation se lit comme suit : *nom de la tâche globale (nom de l'activité appelée)* ; la *surface de revêtement* globale a été appelée trois fois :
  - *Surface du manteau (Surface du manteau 1er tour)*
  - *Surface du manteau (Surface du manteau 2e tour)*
  - *Surface du manteau (Surface du manteau 3ème tour)*

Analyse du temps

Cliquez sur l'onglet « Étapes » de la fenêtre Configurer BPSim, puis sur l'onglet Jetons, qui ressemble à cette illustration :



Token ID	Element	Action	Relative Time	Absolute Time
0	S1	Enter	00	0.0
0	S1	Leave	00	0.0
0	S2	Enter	00	0.0
0	S2	Leave	00	0.0
0	Sand Surface	Enter	00	0.0
0	Sand Surface	Leave	30	30.0
0	Clean Surface	Enter	30	30.0
0	Clean Surface	Leave	40	40.0
0	E2	Enter	40	40.0
0	E2	Leave	40	40.0
0	Coat Surface	Enter	40	40.0
0	Coat Surface	Leave	51	51.0
0	Coat Surface	Enter	51	51.0
0	Coat Surface	Leave	62	62.0
0	Coat Surface	Enter	62	62.0
0	Coat Surface	Leave	69	69.0
0	E1	Enter	69	69.0
0	E1	Leave	69	69.0

Vous pouvez vérifier le timing dans la liste telle quelle, mais pour faciliter le processus, saisissez « Quitter » dans le champ de la barre de filtre de la colonne « Action » pour afficher uniquement les enregistrements contenant cette string de texte dans cette colonne.

Tokens | Property Parameters | Resources

Token ID	Element	Action	Relative Time	Absolute Time
		Leave		
0	S1	Leave	00	0.0
0	S2	Leave	00	0.0
0	Sand Surface	Leave	30	30.0
0	Clean Surface	Leave	40	40.0
0	E2	Leave	40	40.0
0	Coat Surface	Leave	51	51.0
0	Coat Surface	Leave	62	62.0
0	Coat Surface	Leave	69	69.0
0	E1	Leave	69	69.0

Le rapport s'affiche comme indiqué et nous pouvons effectuer cette analyse :

- L'activité d'appel *Préparation de la surface* a duré 40 minutes, composée de *ponçage de la surface* (30 minutes) et *de nettoyage de la surface* (10 minutes), comme défini
- *Coat Surface (1er tour)* a pris 11 minutes ; *Coat Surface (2e tour)* a pris 11 minutes ; *Coat Surface (3e tour)* a pris 7 minutes - les chiffres 11, 11, 7 sont générés aléatoirement par la distribution de Poisson(10) ; ce qui est important ici est que chaque instance d'appel de la tâche globale a ses propres valeurs
- *Coat Surface* a un temps total collecté à partir de toutes les instances : 11 + 11 + 7 = 29
- Le temps de traitement total pour le *processus de peinture du mur* est de 69 minutes, composé des quatre activités d'appel : 40 + 11 + 11 + 7 = 69

Simulation personnalisée

Nous pouvons configurer une « Demande de résultat » sur les éléments BPMN pour personnaliser le rapport de simulation afin de ne rapporter que les paramètres qui nous intéressent.

Configurer la demande de résultat

1. Sur le diagramme « Processus de peinture des murs », cliquez sur l'activité *Surface de revêtement 1er tour* .
2. Dans la fenêtre Configurer BPSim, cliquez sur la flèche déroulante *Nouveau paramètre* et créez un paramètre de temps appelé « ProcessingTime ».
3. Cliquez sur l'icône de la barre d'outils 

. La colonne « Demande de résultat » s'affiche à droite de la colonne « Paramètre » ; cliquez sur la flèche déroulante et cochez la case « somme ». Cliquez sur le bouton OK .
4. Dans le champ « Valeurs », saisissez « 1 ».
5. Cliquez sur l'icône  .
6. Répétez les étapes 1 à 5 pour les activités *Appeler Préparer la surface* , *Enduire la surface 2e tour* , *Enduire la surface 3e tour*

- Développez le groupe « Processus Métier » et répétez ces étapes pour *le processus Peinture murale*

Exécuter Simulation

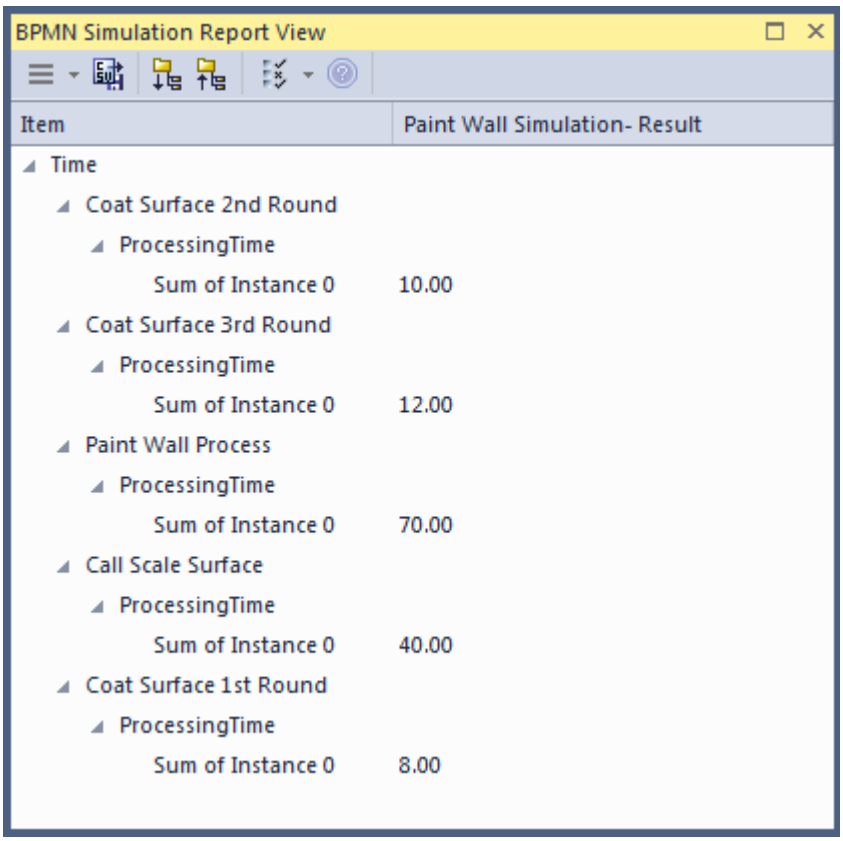
- Dans la barre d'outils de la boîte dialogue « Configurer BPSim », cliquez sur le bouton Exécuter ; la boîte dialogue « Contrôleur Simulation BPSim » s'affiche
- Cliquez sur la flèche déroulante du bouton Exécuter et sélectionnez, dans ce cas, « Simulation personnalisée »

Analyse de flux

L'analyse de flux est exactement la même que pour une Simulation standard.

Analyse du temps

Dans la barre d'outils dialogue « BPSim Simulation Controller », cliquez sur le bouton  ; la « Vue Rapport Simulation BPMN » s'affiche.



L'analyse temporelle est la même que pour une Simulation standard ; cependant, le rapport ne contient que la « somme » des résultats que nous avons demandés.

Note : Actuellement, dans l'analyse temporelle, nous ne pouvons pas demander de temps de traitement ni sur le processus appelé lui-même ni sur les activités contenues dans le processus appelé. Si vous avez cette exigence, utilisez la Simulation standard.

Paramètres de coût de BPSim

BPSim 1.0 permet de définir des paramètres de coûts et de recevoir des statistiques de coûts issues d'expériences de simulation de processus. BPSim fournit un cadre permettant de déterminer les coûts **variables** en fonction de deux paramètres, tous deux liés au niveau d'activité effectué dans le processus simulé. Ces paramètres sont :

- Coût d'achèvement (« coût fixe » dans la spécification BPSim) - Le coût encouru chaque fois qu'une opération est terminée ; ce coût peut être lié à des éléments de tâche, de processus, de sous-processus, d'activité d'appel ou de ressource
- Coût temporel (« coût unitaire » dans la spécification BPSim) - Le coût encouru chaque fois qu'une tâche, un processus, un sous-processus, une activité d'appel ou une ressource est occupé pendant une période donnée.

Les paramètres de coût sont pris en charge sur les activités, les ressources et les processus. Sur :

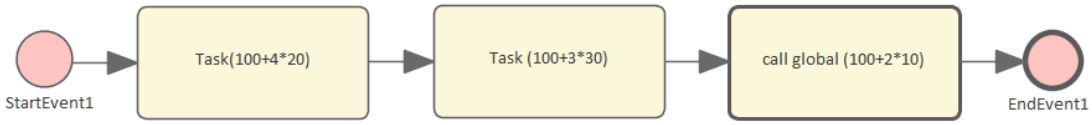
- Une activité, un coût d'achèvement et un coût temporel (coût unitaire * temps) sont tous deux engagés chaque fois qu'une activité se termine.
- Une ressource, un coût d'achèvement et un coût de temps sont tous deux engagés chaque fois que chaque ressource impliquée termine une activité.
- Un processus, un coût d'achèvement et un coût temporel sont tous deux engagés chaque fois qu'un processus se termine.

Les coûts connus sans besoin de simulation (par exemple, les coûts globaux de l'emploi de la main-d'œuvre) ne sont pas pris en charge par BPSim.

La configuration et la simulation des paramètres de coûts sont démontrées par deux exemples :

- Set Cost Parameters on Activity
- Set Cost Parameters on Resource

Définir les paramètres de coût de l'activité



Créer le Modèle BPMN (Activités)

1. Dans la fenêtre Navigateur , créez un *StartEvent1* , un *GlobalTask1* , deux AbstractTasks et un *EndEvent1*.
2. Ctrl+glissez les éléments de la fenêtre Navigateur sur un diagramme , en collant *GlobalTask1* comme une (Activité d'Appel) appelée *call global (100+2*10).*
3. Donnez des noms aux éléments et connectez-les avec des flux Séquence ; les deux AbstractTasks doivent être appelées :
   - *Tâche (100+3*30)* et
   - *Tâche (100+4*20).*

Configuration BPSim

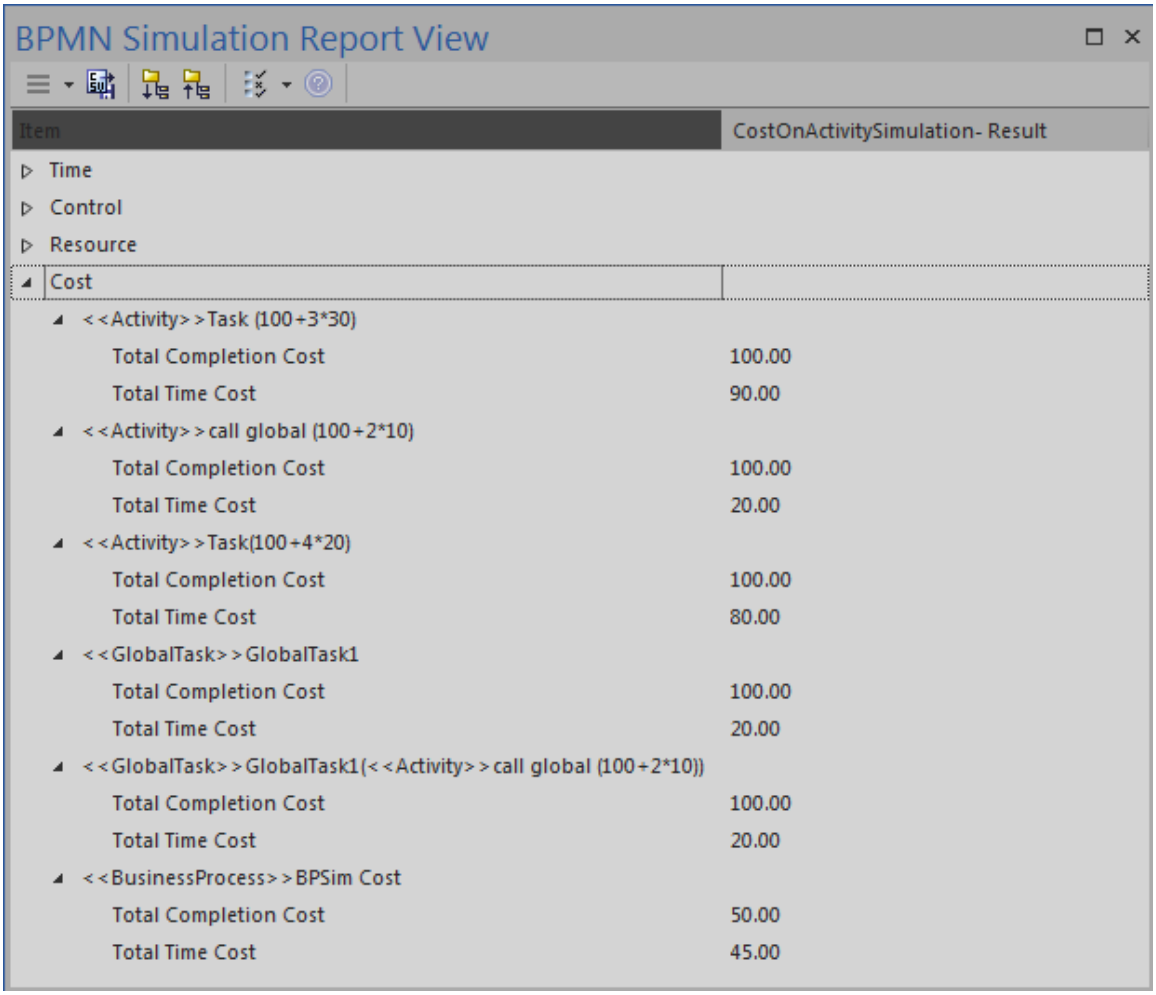
Créez un Artefact de configuration Processus Métier Simulation dans le diagramme , cliquez-droit dessus et sélectionnez l'option 'Configurer BPSim'. Définissez la configuration pour qu'elle soit liée au Paquetage contenant les éléments du modèle BPMN et configurez ces paramètres BPSim comme indiqué.

Paramètre	Paramètres
Paramètres du scénario	1. Cliquez sur l'artefact de configuration BPSim et, pour le paramètre de scénario « Unité de temps », cliquez sur la flèche déroulante « Valeur » et sélectionnez « heures ». 2. Dans le champ « Valeur » du paramètre « Durée », définissez la valeur sur « 0001 00:00:00 » (1 jour). Cette unité de temps est utilisée pour calculer le coût du temps (coût du temps = coût unitaire * temps), assurez-vous donc que le coût unitaire est basé sur la bonne unité de temps.
Paramètres de contrôle	1. Sur le diagramme , cliquez sur *StartEvent1*. 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Contrôle ». 3. Dans le champ « Paramètre », cliquez sur la flèche déroulante et sélectionnez « TriggerCount ». 4. Dans le champ « Valeur », saisissez « 1 ».
Paramètres de temps	1. Sur le diagramme cliquez sur *Tâche (100+4*20)* . 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Heure ». 3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Temps de traitement ». 4. Dans le champ « Valeur », définissez la valeur sur « 000:000:000 004:00:00 » (4 heures). 5. Cliquez sur *Tâche (100+3*30)* sur le diagramme et répétez les étapes 2, 3 et 4, en définissant le champ « Valeur » sur « 000:000:000 003:00:00 » (3 heures).

	6. Cliquez sur *GlobalTask1* sur le diagramme et répétez les étapes 2, 3 et 4, en définissant le champ « Valeur » sur « 000:000:000 002:00:00 » (2 heures).
Paramètres de coût	1. Sur le diagramme , cliquez sur *Tâche (100+4*20)* . 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Coût ». 3. Dans le champ « Paramètre », cliquez sur la flèche déroulante et sélectionnez successivement : - 'FixedCost', puis dans le champ 'Valeur' cliquez sur le bouton  , sélectionnez l'onglet « Constante » et « Flottant », et dans le champ « Constante flottante » tapez « 100 » ; cliquez sur le bouton OK - 'UnitCost' - faites de même, en définissant le champ 'Constant Floating' sur '20'. 4. Sur le diagramme , cliquez sur *Tâche (100+3*30)* et répétez les étapes 2 et 3, en définissant : - 'FixedCost' à '100 - 'UnitCost' à '30'. 5. Sur le diagramme , cliquez sur *GlobalTask1* et répétez les étapes 2 et 3 en définissant : - 'FixedCost' à '100 - 'UnitCost' à '10'. 6. Sur le diagramme , cliquez sur *Coût BPSim* et répétez les étapes 2 et 3 en définissant : - « Coût fixe » à « 50 » - 'UnitCost' à '5'.

Simulation

1. Dans la dialogue « Configurer BPSim », cliquez sur l'onglet « Exécuter ».
2. Cliquez sur le bouton '  '.
3. Une fois la simulation terminée, cliquez sur l'onglet « Révision », puis sur l'onglet « Rapport de résultats standard ».
4. Filtrez le rapport en cliquant sur le bouton  et en sélectionnant l'option « Afficher uniquement Items non vides ».

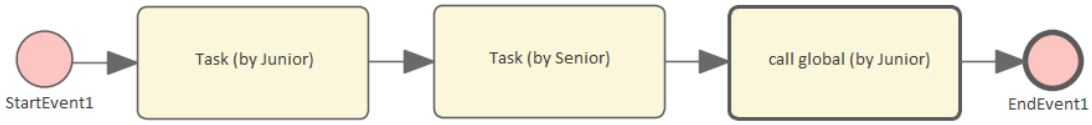


Analyse

Activité	Analyse
Tâche (100+4*20)	• Le coût total d'achèvement est de 100, ce qui correspond au paramètre FixedCost (100) dans BPSim • Le coût total du temps est de 80, calculé comme Temps de traitement (4 heures) * Coût unitaire (20/heure)
Tâche (100+3*30)	• Le coût total d'achèvement est de 100, ce qui correspond au paramètre FixedCost (100) dans BPSim • Le coût total du temps est de 90, calculé comme Temps de traitement (3 heures) * Coût unitaire (30/heure)
appel global (100+2*10)	• Le coût total d'achèvement est de 100, ce qui correspond au coût fixe (100) du paramètre *GlobalTask1* dans BPSim • Le coût total du temps est de 20, calculé comme le temps de traitement (2 heures) * le coût unitaire (10/heure) sur *GlobalTask1*
Processus de coûts BPSim	• Le coût total d'achèvement est de 50, ce qui correspond au paramètre FixedCost (50) dans BPSim

	• Le coût total du temps est de 45, calculé comme le temps de traitement total de toutes les tâches (4 + 3 + 2 = 9 heures) * coût unitaire (5/heure)

Définir les paramètres de coût sur la ressource



Créer le Modèle BPMN (Ressources)

1. Dans la fenêtre Navigateur , créez un *StartEvent1* , un *GlobalTask1* , deux abstractTasks appelées *Task (par Junior)* et *Task (par Senior)* et un *EndEvent1*.
2. Ctrl+glissez les éléments de la fenêtre Navigateur sur un diagramme , en collant *GlobalTask1* comme une (Activité d'Appel) nommée *call global (par Junior).*
3. Connectez les éléments avec des flux Séquence .
4. Créez deux éléments de ressources BPMN2.0 : *développeur junior* et *développeur senior.*

Configuration de BPSim

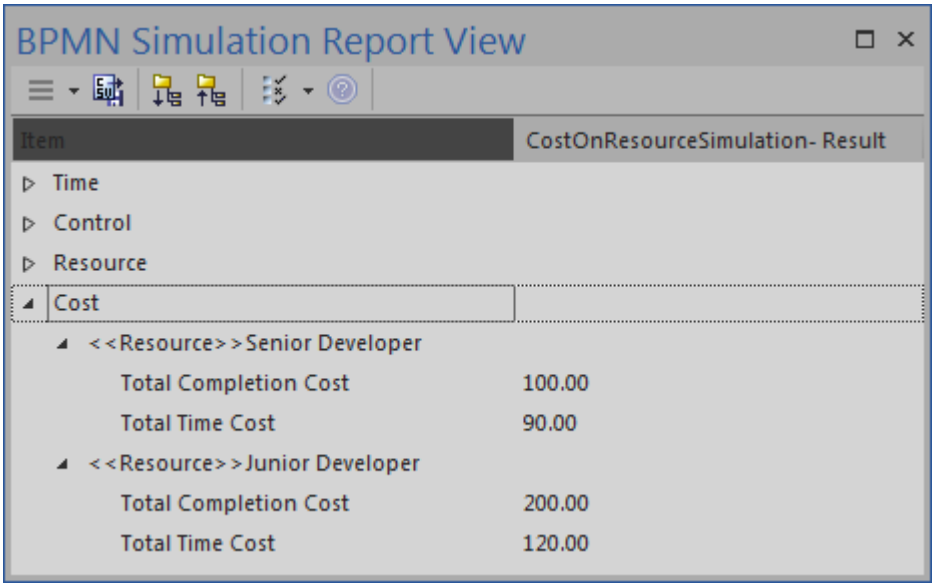
Créez un artefact de configuration Processus Métier Simulation dans le diagramme , cliquez-droit dessus et sélectionnez l'option 'Configurer BPSim', puis définissez la configuration à lier au Paquetage contenant les éléments du modèle BPMN et configurez ces paramètres BPSim comme indiqué.

Paramètre	Paramètre
Paramètres du scénario	1. Cliquez sur l'artefact de configuration BPSim et, pour le paramètre de scénario « Unité de temps », cliquez sur la flèche déroulante « Valeur » et sélectionnez « heures ». 2. Dans le champ « Valeur » du paramètre « Durée », définissez la valeur sur « 0001 00:00:00 » (1 jour). Cette unité de temps est utilisée pour calculer le coût du temps (coût du temps = coût unitaire * temps), assurez-vous donc que le coût unitaire est basé sur la bonne unité de temps.
Paramètres de contrôle	1. Sur le diagramme , cliquez sur *StartEvent1*. 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Contrôle ». 3. Dans le champ « Paramètre », cliquez sur la flèche déroulante et sélectionnez « TriggerCount ». 4. Dans le champ « Valeur », saisissez « 1 ».
Paramètres de temps	1. Sur le diagramme cliquez sur *Tâche (par Junior)* . 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Heure ». 3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Temps de traitement ». 4. Dans le champ « Valeur », définissez la valeur sur « 000:000:000 004:00:00 » (4 heures). 5. Cliquez sur *Tâche (par Senior)* sur le diagramme et répétez les étapes 2, 3 et 4, en définissant le champ « Valeur » sur « 000:000:000 003:00:00 » (3 heures).

	6. Cliquez sur *GlobalTask1* sur le diagramme et répétez les étapes 2, 3 et 4, en définissant le champ « Valeur » sur « 000:000:000 002:00:00 » (2 heures).
Paramètres des ressources	1. Sur le diagramme , cliquez sur la ressource *Développeur Junior* . 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Ressource ». 3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez « Sélection ». 4. Dans le champ « Valeurs », cliquez sur le bouton  pour ouvrir la dialogue « Modifier la sélection de ressources ». 5. Cliquez sur « Développeur junior » et sur le bouton Ajouter une sélection par ressource(s) pour déplacer la sélection vers le panneau « Ressource ou rôle ». 6. La colonne « Quantité requise » est définie par défaut sur « 1 » ; remplacez cette valeur par « 10 ». 7. Cliquez sur le bouton radio AND pour définir la relation logique ; l'expression finale pour la sélection des ressources est composée et affichée dans le champ de texte. 8. Cliquez sur le bouton OK pour revenir à la fenêtre Configurer BPSim, où l'expression est affichée dans le champ « Valeurs ». 9. Cliquez sur la ressource *Développeur Senior* et répétez les étapes 2 à 8, en tapant « 5 » dans le champ « Quantité requise ».
Paramètres de coût	1. Sur le diagramme cliquez sur *Développeur Junior* . 2. Cliquez sur la flèche déroulante *Nouveau paramètre* et sélectionnez « Ressource ». 3. Cliquez sur la flèche déroulante « Paramètre » et sélectionnez tour à tour : - 'FixedCost', puis dans le champ 'Valeur' cliquez sur le bouton  , sélectionnez l'onglet « Constante » et « Flottant », puis dans l'onglet « Constante flottante » saisissez le champ « 100 » et dans le champ « CurrencyUnit », saisissez « AUD » ; cliquez sur le bouton OK - 'UnitCost' - faites de même, en définissant le champ 'Constant Floating' sur '20'. 4. Sur le diagramme cliquez sur *Senior Developer* et répétez les étapes 2 et 3 en définissant : - « Coût fixe » à « 100 » - 'UnitCost' à '30'.

Simulation

1. Dans la dialogue « Configurer BPSim », cliquez sur l'onglet « Exécuter ».
2. Cliquez sur le bouton '  '.
3. Une fois la simulation terminée, cliquez sur l'onglet « Révision », puis sur l'onglet « Rapport de résultats standard ».
4. Filtrez le rapport en cliquant sur le bouton  et en sélectionnant l'option « Afficher uniquement Items non vides ».



Analyse

Ressource	Résultats
Développeur junior	• Le coût total d'achèvement est de « 200 », calculé comme coût fixe (100) * nombre d'activités impliquées (2) • Le coût total du temps est de « 120 », calculé comme le temps de traitement (4 + 2 = 6 heures) * le coût unitaire (20 / heure)
Développeur Senior	• Le coût total d'achèvement est de « 100 », calculé comme coût fixe (100) * nombre d'activités impliquées (1) • Le coût total du temps est de « 90 », calculé comme Temps de traitement (3 heures) * Coût unitaire (30 / heure)

Exporter une configuration BPSim

Lorsque vous avez défini une configuration BPSim dans un modèle, vous pouvez l'exporter vers un fichier XMI pour l'importer dans d'autres projets. Le modèle BPMN 2.0 sur lequel la configuration est basée est également exporté avec la configuration. Le modèle est lié à la configuration BPSim appropriée lorsque vous importez le fichier XMI dans un autre projet.

Accéder

Menu Contexte	Sur un diagramme ou dans la fenêtre Navigateur , cliquez-droit sur Processus Métier Simulation Artefact | Exporter la configuration BPSim
Autre	Barre d'outils de la fenêtre Configurer BPSim |  l'icône Exporter

Publier Modèle Paquetage

Le processus d'exportation d'une configuration BPSim et de son modèle utilise la dialogue « Publier Modèle Paquetage » pour publier un modèle dans un fichier XMI.

Option	Description
Paquetage	La valeur par défaut est le nom du Paquetage contenant l'artefact Simulation Processus Métier .
Nom de fichier	Type ou recherchez (cliquez sur l'icône ) le chemin d'accès au fichier et le nom du fichier XML dans lequel exporter le modèle.
Type XML	Sélectionnez « BPMN 2.0 XML ».
Exporter	Cliquez sur ce bouton pour exporter la configuration et le modèle BPMN 2.0. L'exportation est terminée lorsqu'un message de confirmation s'affiche dans le champ « Progression ».
Format de sortie XML	La valeur par défaut est sélectionnée ; laissez sélectionnée.
Vue XML	Si vous souhaitez examiner le XML exporté, cliquez sur ce bouton.

Notes

- Pour importer le modèle de XMI dans un nouveau projet, sélectionnez le Paquetage cible dans le nouveau projet et sélectionnez l'option de ruban « Publier > Échange de Modèles > Importer > Fichier natif » ou « Importer > Fichier XMI »

Decision Model and Notation (DMN)

Créer et simuler des modèles détaillés de Décisions d'entreprise

Les entreprises sont confrontées à des environnements opérationnels de plus en plus difficiles, avec une concurrence féroce et souvent imprévisible de la part des acteurs du marché existants et nouveaux, des changements dans les réglementations gouvernementales et industrielles et des bouleversements dans le tissu social de leur clientèle. Les décisions qu'une entreprise prend dans ce contexte sont essentielles à sa réussite et à sa capacité à se frayer un chemin sûr dans ces eaux inexplorées. En utilisant fonctionnalités Decision Model and Notation (DMN) d'Enterprise Architects, vous pouvez non seulement modéliser les décisions prises par votre entreprise, mais également exécuter des simulations à partir de ces modèles pour prédire les résultats en fonction d'exemples de données. La puissance du langage est que les professionnels peuvent facilement comprendre et travailler avec diagrammes Décision Exigences simples mais expressifs qui détaillent les décisions, les entrées des décisions et les sorties attendues. Les règles peuvent être documentées de plusieurs manières, notamment par tableaux de décision faciles à définir. Une fois terminés, ces diagrammes accompagnés d'exemples de données d'entrée peuvent être simulés pour montrer les résultats des décisions.

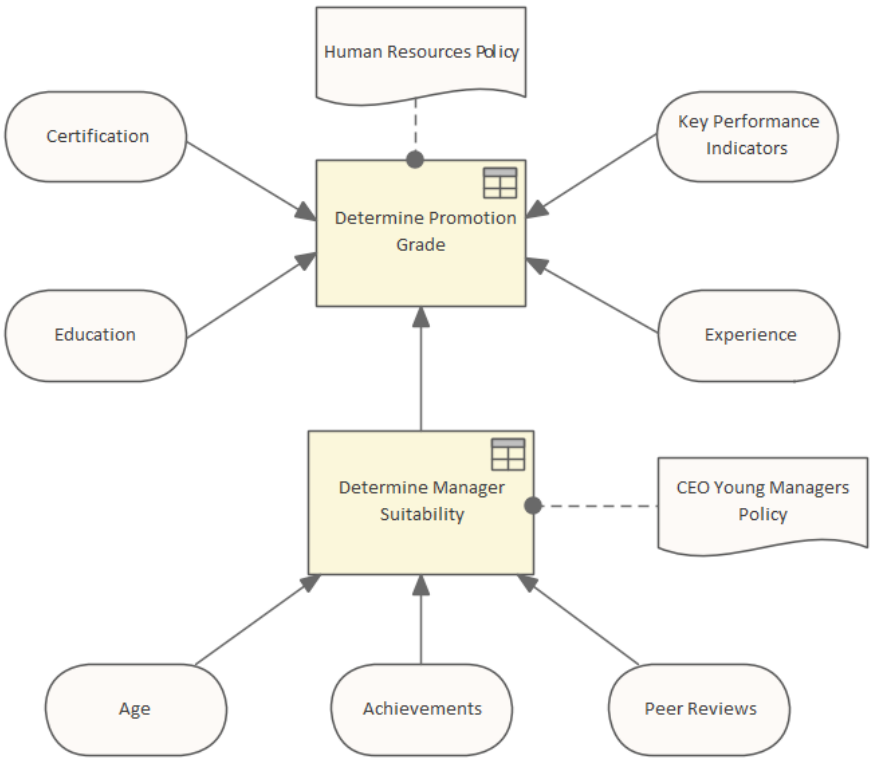


diagramme Décision Exigences montrant une Décision avec un Modèle de Connaissance Métier et un certain nombre d'entrées dont une autre Décision .

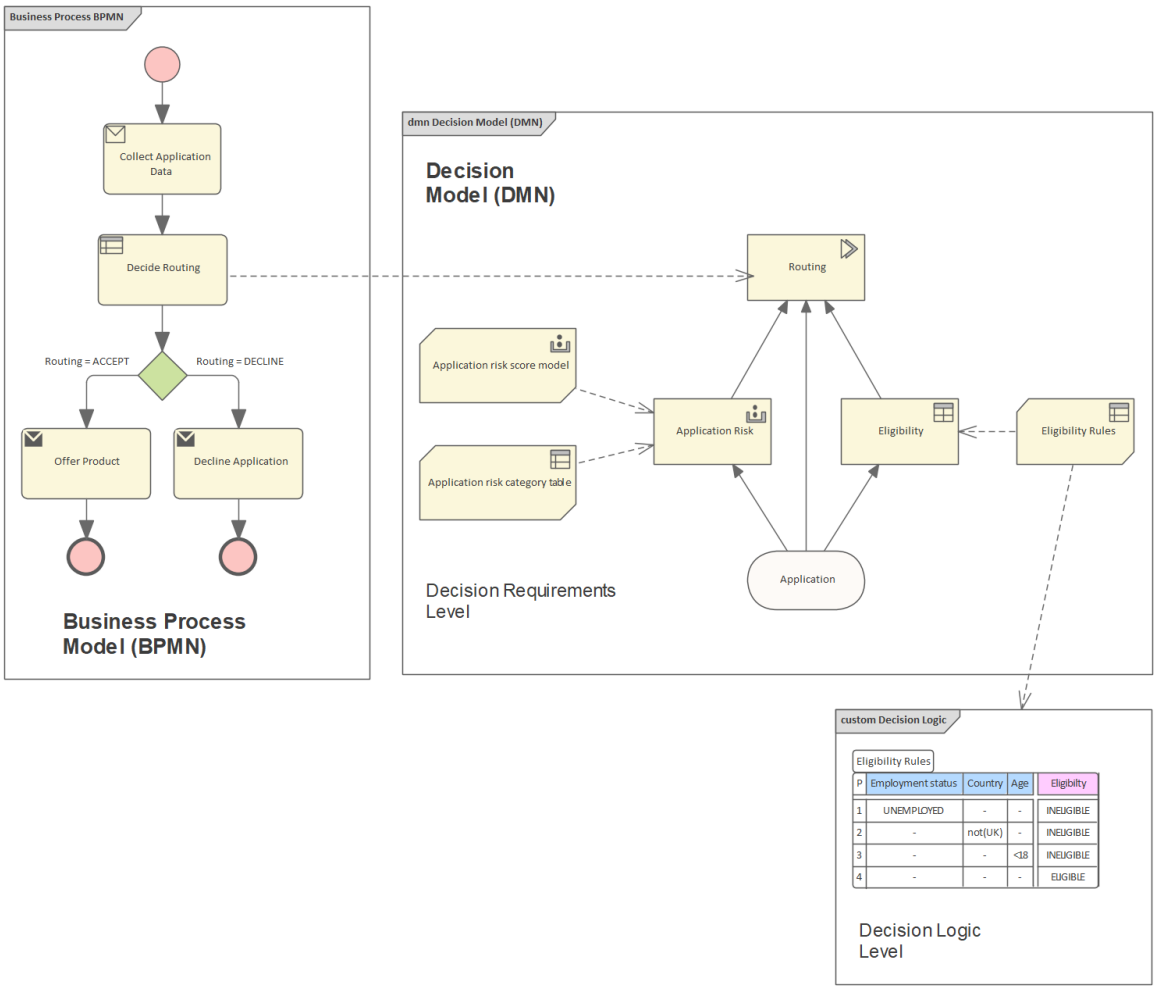
Une fois ces modèles définis, simulés et testés par l'entreprise, les technologues et les ingénieurs peuvent ensuite affiner ces modèles et générer automatiquement des artefacts logiciels, y compris du code de programmation directement à partir des modèles, réduisant ainsi la possibilité d'erreurs d'interprétation et réduisant le temps de mise en œuvre.

Qu'est-ce que DMN ?

DMN est destiné à fournir un pont entre les modèles de processus métier et les modèles de logique de décision :

- Les modèles de processus Métier définiront les tâches au sein des processus métier où la prise de décision doit avoir lieu
- Diagrammes Décision Exigences définiront les décisions à prendre dans ces tâches, leurs interrelations et leurs exigences en matière de logique de décision.

- La logique Décision définira les décisions requises de manière suffisamment détaillée pour permettre la validation et/ou l'automatisation



Pris ensemble, diagrammes et la logique de décision Décision Exigences vous permettent de construire un Modèle Décision complet qui complète un modèle de processus métier en spécifiant - en détail - la prise de décision effectuée dans les tâches du processus.

DMN fournit des constructions couvrant à la fois les exigences de décision et modélisation de la logique de décision.

- Pour modélisation des exigences de décision, on définit le concept de Graphe Décision Exigences (DRG) comprenant un ensemble d'éléments et leurs règles de connexion, et une notation correspondante : le Diagramme Décision Exigences (DRD).
- Pour modélisation de la logique de décision, il fournit un langage appelé FEEL pour définir et assembler Tableaux Décision , des calculs, une logique if/then/else, des structures de données simples et une logique définie en externe à partir de Java et PMML dans des expressions exécutables avec une sémantique formellement définie.

Avantages de l'utilisation de DMN dans Enterprise Architect

Modélisation des processus décisionnels à l'aide de DMN vous permet d'enregistrer, de spécifier et d'analyser des processus décisionnels complexes en tant que système de décisions, de règles métier, d'ensembles de données et de sources de connaissances interdépendants. Ce faisant, vous pouvez décomposer un processus décisionnel très complexe en un réseau de décisions de soutien et de données d'entrée. Cela facilite la compréhension du processus global, supporte la refactorisation des processus et simplifie la tâche de validation du processus, en vous permettant de valider facilement les étapes individuelles qui composent le processus global.

Lorsque vous créez un Modèle Décision dans Enterprise Architect à l'aide de DMN, vous pouvez exécuter des simulations du modèle pour vérifier l'exactitude du modèle. Après avoir vérifié votre modèle, vous pouvez générer un

module DMN en Java, JavaScript , C++ ou C# . Le module DMN généré peut être utilisé avec le Moteur d'Exécution BPSim Enterprise Architect , Statemachine Exécutable ou dans un système logiciel distinct que vous implémentez.

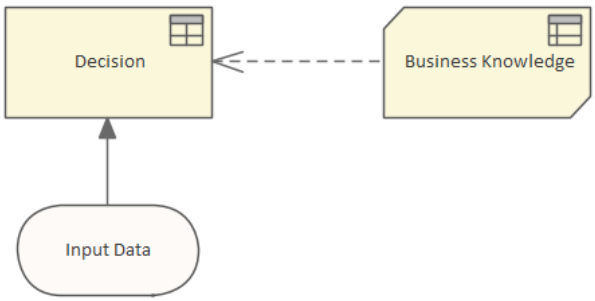
Enterprise Architect fournit également un module Test facilité , qui est un préprocessus pour l'intégration de DMN avec BPMN. L'objectif est de produire des éléments BPMN2.0::DataObject, puis de les utiliser pour vérifier qu'une décision cible spécifiée est évaluée correctement avec le module DMN. Vous configurez ensuite BPSim en chargeant des DataObjects et en attribuant des décisions du module DMN aux Propriétés BPSim.

Cette fonctionnalité est disponible dans les éditions Unified et Ultimate d' Enterprise Architect , à partir de la version 15.0.

Graphiques Exigences Décision

Le modèle de décision DMN se compose d'un graphe Exigences Décision (DRG) représenté par un ou plusieurs Diagrammes Exigences Décision (DRD). Les éléments modélisés sont les décisions, les domaines de connaissances métier, les sources de connaissances métier, les données d'entrée et les services de décision.

Un DRG est un graphe composé d'éléments reliés par des exigences, et est autonome dans le sens où toutes les exigences modélisées pour toute Décision dans le DRG (ses sources immédiates d'information, de connaissances et d'autorité) sont présentes dans le même DRG. Il est important de distinguer cette définition complète du DRG d'un DRD présentant une vue particulière de celui-ci, qui peut être un affichage partiel ou filtré.



Démarrage

Decision Model and Notation (DMN) est une norme publiée et gérée par l' Object Management Group (OMG).

Des parties de ce sujet ont été utilisées textuellement ou sont librement adaptées de la Spécification DMN, qui est disponible sur la page Web DMN de l'OMG (https://www.omg.org/spec/DMN). Une description complète du DMN et de ses capacités est disponible sur le site Web de l'OMG.

L'objectif de DMN est de fournir les constructions nécessaires à la modélisation des décisions, afin que la prise de décision organisationnelle puisse être facilement représentée sous forme diagrammes , définie avec précision par analystes métier et (éventuellement) automatisée. Il est également destiné à faciliter le partage et l'échange de modèles Décision entre les organisations.

Sélection de la perspective

Enterprise Architect divise les nombreuses fonctionnalités de l'outil en Perspectives , ce qui vous permet de vous concentrer sur une tâche spécifique et de travailler avec les outils dont vous avez besoin sans être distrait par d'autres fonctionnalités . Pour travailler avec les fonctionnalités Decision Model and Notation vous devez d'abord sélectionner cette perspective :

 <nom de la perspective> > Exigences > Décision Modélisation

La définition de la Perspective garantit que les diagrammes Decision Model and Notation , leurs boîtes à outils et autres fonctionnalités de la Perspective seront disponibles par défaut.

Exemple Diagramme

Un exemple diagramme fournit une introduction visuelle au sujet et vous permet de voir certains des éléments et connecteurs importants qui sont créés pour spécifier ou décrire la manière dont les décisions sont modélisées. Le diagramme Décision Exigences présentera des éléments tels que Tableaux Décision , les Sources de Connaissances, les Dates de Saisie et bien plus encore. Une grande partie de la puissance d' Enterprise Architect repose sur la capacité à simuler ou à « exécuter » les modèles de décision et à prédire les résultats en fonction de différents ensembles de données. Cette fonctionnalité sera décrite dans les rubriques suivantes, mais commence par la création d'un diagramme Décision Exigences .

Modélisation avec DMN

Cette rubrique vous présente les éléments les plus importants dont vous avez besoin pour créer des modèles Décision . Cela comprend la création d'un diagramme Exigences Décision qui décrit la manière dont les décisions sont liées et les entrées que chaque décision possède potentiellement, y compris d'autres décisions. Vous découvrirez les éléments les plus importants, notamment : les décisions, les modèles de connaissances Métier , les définitions des Items de données d'entrée, l'ensemble de données et les services Décision .

Module de génération et Test de code

Cette rubrique vous présente les principaux concepts du langage, notamment sa structure, architecture et les éléments et connecteurs utilisés pour créer Decision Model and Notation (DMN) . La compréhension de l'intention et de la structure du langage aidera les analystes à créer des modèles de décision significatifs et productifs.

Intégrer dans BPSim pour Simulation

Enterprise Architect permet d' exécuter (simuler) des modèles Décision qui vous permettent de visualiser les résultats des décisions. En plus de cette facilité de base, vous pouvez également intégrer les modèles Décision avec BPSim qui est un moteur de simulation pour la simulation de diagrammes BPMN. Dans cette rubrique, vous découvrirez différentes manières d'intégrer DMN avec BPSIm.

Intégrer dans l'élément de classe UML

Dans cette rubrique, vous apprendrez le processus d'intégration d'un Modèle DMN avec un élément de classe UML . Le module DMN peut être intégré à un élément de classe UML , de sorte que le code généré à partir de cet élément de classe peut réutiliser le module DMN et être bien structuré

Importation de DMN XML

Cette rubrique décrit comment importer un fichier XML DMN à partir d'un autre référentiel Enterprise Architect ou d'un autre outil compatible DMN. L'une des promesses des normes ouvertes est la possibilité de partager des modèles entre différents outils. Enterprise Architect devient souvent l'outil de choix pour modélisation en raison de l'étendue des fonctionnalités et des normes qu'il supporte . Cela permet de relier le modèle Décision à la stratégie, Exigences , Processus Métier , aux éléments d'implémentation logicielle et plus encore.

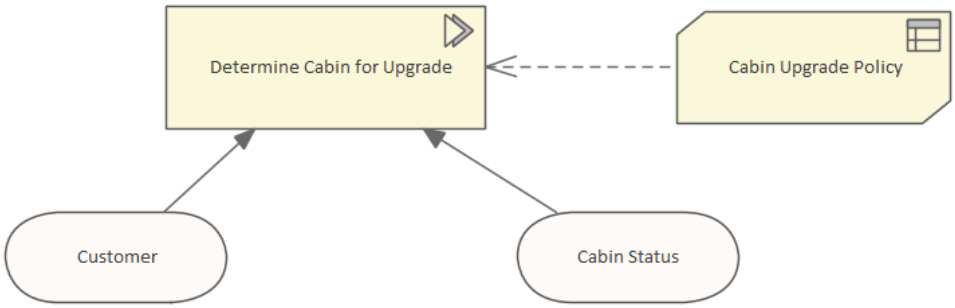
Plus d'informations

Cette section fournit des liens utiles vers d'autres sujets et ressources que vous pourriez trouver utiles lorsque vous travaillez avec les fonctionnalités de l'outil Decision Model and Notation .

Exemple Diagramme

Imaginez que vous êtes un agent de réservation d'une compagnie aérienne nationale très fréquentée. Il est essentiel de faire décoller l'avion à l'heure, car les retards peuvent entraîner des frais appliqués par les contrôleurs de l'aéroport, l'obligation de voler à une altitude inférieure augmentant le coût du carburant et d'autres pénalités.

Un message du superviseur apparaît sur votre écran indiquant que la cabine économique est surbookée ; vous devrez surclasser certains passagers en Métier ou en première classe. Mais quels passagers choisir et dans quelle cabine les surclasser ? Une décision doit être prise, mais quels facteurs doivent être pris en compte ? Cela peut être consigné dans un Modèle Décision à l'aide d'un diagramme Décision Exigences .



Cela est utile, mais l'agent d'enregistrement, très occupé, devra toujours peser tous les facteurs et prendre une décision impartiale. Faut-il donner la priorité à un passager mécontent par rapport à un voyageur fréquent de niveau Gold, ou le fait qu'un passager particulier soit en correspondance avec un vol international doit-il avoir la priorité ? Ces « règles » peuvent toutes être enregistrées dans un Tableau de Décision , indiquant clairement quels passagers doivent bénéficier d'un surclassement et dans quelle cabine : Métier ou Première Classe. Cela facilitera grandement la prise de décision et les règles pourront être formulées, acceptées et vérifiées pour leur cohérence au siège social. Dans cet exemple, nous avons gardé les choses simples et utilisé deux facteurs : tout d'abord le nombre de vols effectués par le passager au cours du dernier mois et ensuite le niveau de surréservation de la cabine.

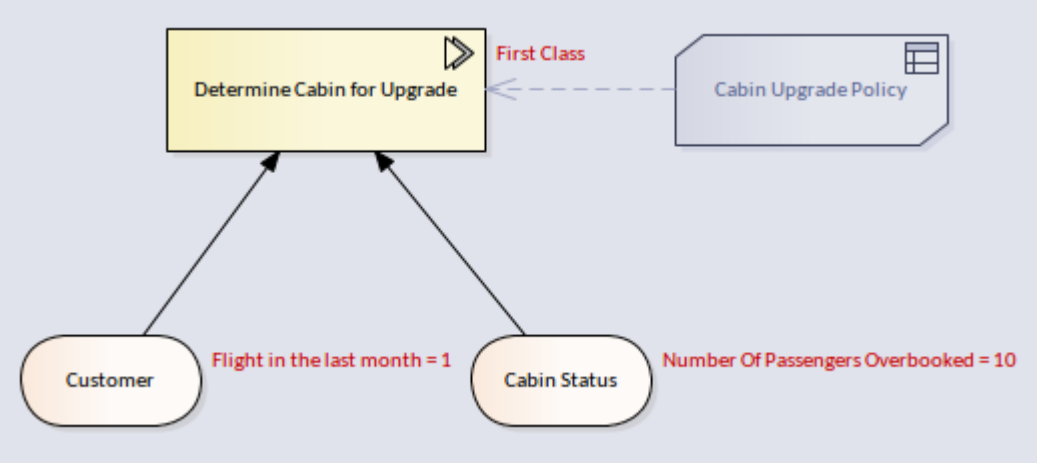
Cabin Upgrade Policy | Input Parameter Values for Simulation

	(Flights in the last month, Number of Pax Overbooked)			
U	Flights in the last month	Number of Pax Overbooked	Upgrade Cabin	Annotation
			Business Class, First Class	
1	<=1	<=2	*Business Class*	
2	<=1	(2..8]	*Business Class*	
3	<=1	>8	*First Class*	*Start Filling First Class when heavily overbooked*
4	(1..5]	<=2	*Business Class*	
5	(1..5]	(2..8]	*Business Class*	
6	(1..5]	>8	*First Class*	
7	>5	<=2	*Business Class*	
8	>5	(2..8]	*Business Class*	
9	>5	>8	*First Class*	*Reward Frequent Flyers*

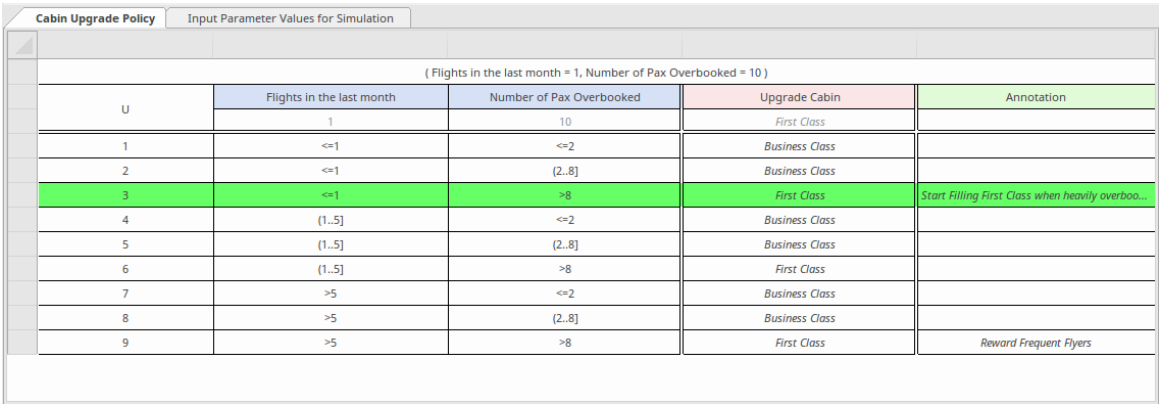
Le tableau est divisé en colonnes et en lignes. Il existe trois types de colonnes : les entrées nécessaires à la prise de décision, les sorties résultant de l'application des règles et les annotations.

C'est encore une fois très utile, mais cela nécessite toujours que l'agent d'enregistrement occupé soit en mesure de trouver toutes les informations nécessaires pour trouver la bonne ligne dans le Tableau de Décision . Même si toutes ces informations étaient disponibles, une mauvaise décision pourrait toujours résulter d'une erreur humaine dans la sélection de la mauvaise ligne dans le tableau .

Heureusement, les modèles Décision peuvent être automatisés et générés en code de programmation pouvant être exécuté par une application. Ainsi, notre agent d'enregistrement n'aurait rien à faire ni à prendre de décision ; pendant qu'il enregistrerait les passagers, si un passager particulier avait droit à un surclassement, cela serait visible sur l'écran de l'ordinateur. Dans le diagramme suivant, le modèle a été simulé afin que le personnel commercial et technique puisse s'accorder sur le fait que le modèle a été défini correctement. N'importe quel nombre d'ensembles de données définis par l'utilisateur peut être utilisé pour tester le modèle avant de générer le code de programmation qui sera exécuter dans le système d'enregistrement et affichera le résultat à l'utilisateur final.



Lors du développement des modèles, un utilisateur technique ou professionnel peut parcourir la simulation et le système lui montrera quelle ligne du Tableau de Décision a été déclenchée pour déterminer le résultat. Ceci est très utile dans les modèles constitués de plusieurs décisions.



Cabin Upgrade Policy | Input Parameter Values for Simulation

(Flights in the last month = 1, Number of Pax Overbooked = 10)

U	Flights in the last month	Number of Pax Overbooked	Upgrade Cabin	Annotation
	1	10	First Class	
1	<=1	<=2	Business Class	
2	<=1	(2..8]	Business Class	
3	<=1	>8	First Class	Start Filling First Class when heavily overboo...
4	(1..5]	<=2	Business Class	
5	(1..5]	(2..8]	Business Class	
6	(1..5]	>8	First Class	
7	>5	<=2	Business Class	
8	>5	(2..8]	Business Class	
9	>5	>8	First Class	Reward Frequent Flyers

Il est fréquent que les règles qui régissent la décision de surclassement changent. Par exemple, le service marketing peut décider de récompenser les passagers qui voyagent sur des vols long-courriers. Le diagramme Décision Exigences peut être modifié pour inclure la nouvelle entrée, le Tableau de Décision modifié et le code de programmation régénéré. Une fois les modifications transmises aux systèmes de l'aéroport, les passagers concernés seront automatiquement surclassés. L'agent d'enregistrement peut toujours consulter les Tableaux Décision lors d'une séance de formation et d'information pour comprendre les règles.

Créer un Modèle Décision

Dans le modèle que nous avons décrit dans *Un exemple de Modélisation Décision* , nous avons montré comment une décision peut être modélisée à l'aide d'un Tableau de Décision , dans lequel un résultat de décision est déterminé en trouvant une ligne dans le tableau où les valeurs d'entrée du tableau correspondent aux valeurs d'entrée considérées, donnant un résultat de sortie particulier.

Nous allons maintenant voir comment un tel modèle peut être créé dans Enterprise Architect , en parcourant le processus de création du modèle de décision pour l'exemple de mise à niveau de cabine de compagnie aérienne.

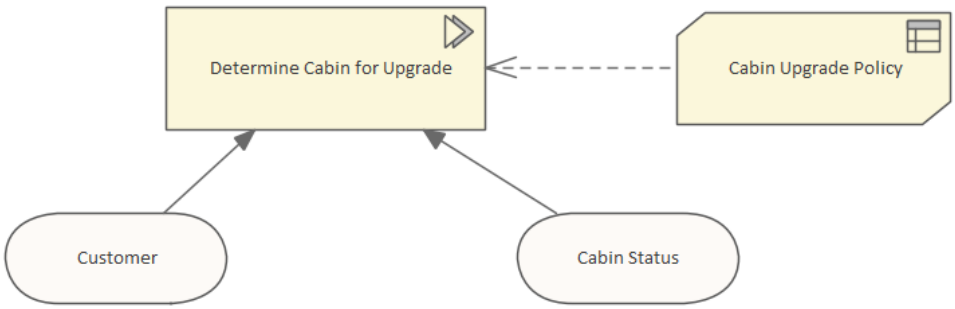
Plusieurs éléments de modèle sont impliqués dans cet exemple, tels que des éléments de données d'entrée, des définitions Item qui sont utilisées pour décrire les données d'entrée (définissant les types de données), un élément Décision et également un élément Métier Knowledge Modèle qui contient la définition du Tableau de Décision .

Créer un Diagramme Exigences Décision

Ces étapes vous guideront dans la création d'un Diagramme Exigences Décision (DRD) simple. Dans cet exemple, nous allons créer le modèle à partir de zéro, plutôt que d'utiliser un motif du Constructeur de Modèle .

Étape	Description
1	Sélectionnez la perspective ' Exigences | Décision Modélisation '. (dialogue Constructeur de Modèle s'affiche, mais nous ne l'utiliserons pas pour cet exemple.)
2	Créez un nouveau diagramme DMN. Nommez-le « Surclassement de cabine de compagnie aérienne ».
3	À l'aide de la boîte à outils diagramme , placez un élément Décision sur le diagramme . Choisissez « Invocation » comme type : nous utiliserons cet élément pour « invoquer » une décision à partir d'un élément Métier Knowledge Modèle . Nommez l'élément « Déterminer la cabine à mettre à niveau ».
4	Placez un élément InputData sur le diagramme . Nommez cet élément « Client ».
5	Placez un autre élément InputData sur le diagramme . Nommez cet élément « Statut de la cabine ».
6	Placer un élément Métier Knowledge Modèle sur le diagramme . Choisissez le type ' Tableau de Décision '. Nommez cet élément « Politique de mise à niveau de la cabine ».
7	Dessinez un connecteur « Besoin d'informations » **à partir de** la décision « Déterminer la cabine pour la mise à niveau » **vers** les données d'entrée « Client ».
8	Dessinez un connecteur « Besoin d'informations » **à partir de** la décision « Déterminer la cabine pour la mise à niveau » **vers** les données d'entrée « Statut de la cabine ».
9	Dessinez un connecteur « Exigences en matière de connaissances » **à partir de** la décision « Déterminer la cabine pour la mise à niveau » **vers** la « Politique de mise à niveau de la cabine » du BKM.

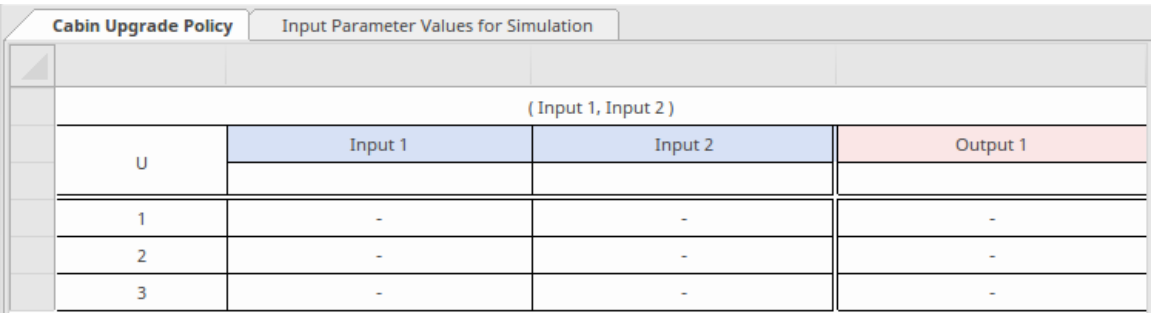
À ce stade, nous devrions avoir un DRD simple, qui ressemble à ceci :



Nous pouvons maintenant préciser les détails pour chacun des éléments composant ce modèle.

Définir le Tableau de Décision

En double-cliquant sur l'élément Métier Knowledge Modèle 'Politique de surclassement de cabine', la fenêtre 'Expression DMN' s'affiche, montrant un Tableau de Décision vide. C'est ici que nous allons définir les règles de notre politique de surclassement de cabine.



Par défaut, les nouveaux Tableaux Décision sont créés avec deux colonnes d'entrée et une colonne de sortie, une ligne d'en-tête et trois lignes de règles vides.

La colonne la plus à gauche du tableau affiche la politique Hit et numérote également les règles. Par défaut, la politique Hit est « U » pour « Unique ». Il s'agit de la politique que nous utiliserons pour notre exemple, vous n'avez donc pas besoin de modifier l'en-tête de cette colonne.

Pour plus d'informations sur les politiques Hit , reportez-vous à la rubrique d'aide *Politique Hit Tableau de Décision* .

Nommer et définir les types d'entrées et de sorties du Tableau de Décision

Étape	Description
1	Dans la barre d'outils de la fenêtre « Expression DMN », cliquez sur le bouton « Modifier les paramètres »,  . La dialogue « Modifier les paramètres » s'affiche.
2	Remplacez le nom du paramètre « Entrée 1 » par « Nombre de passagers surréservés ». Si nécessaire, cliquez sur la flèche déroulante « Type » et définissez le type de ce paramètre sur « nombre ».

3	Remplacez le nom du paramètre « Entrée 2 » par « Nombre de vols au cours du dernier mois par passage ». Définissez également le type de ce paramètre sur « numéro ». Fermez la dialogue « Modifier les paramètres ».
4	Modifiez l'expression d'entrée qui sera évaluée pour la colonne 1. Sélectionnez la cellule d'en-tête (contenant le texte « Entrée 1 ») puis cliquez à nouveau ou appuyez sur F2 pour entrer en mode « Modifier ». Sélectionnez tout le texte de la cellule, puis appuyez sur la barre d'espace. La liste des paramètres d'entrée s'affiche. Cliquez sur « Nombre de personnes surbookées », puis appuyez sur « Entrée ». L' *expression* de la colonne 1 est définie sur « Nombre de personnes surbookées ». **Note** : les expressions d'entrée évaluées pour chaque colonne utilisent généralement simplement le paramètre d'entrée correspondant ; cependant, vous *pouvez* utiliser une expression complexe.
5	Cliquez-droit sur l'expression de la colonne 1 et vérifiez que son type de données est défini sur « nombre ».
6	Modifiez l'expression d'entrée qui sera évaluée pour la colonne 2. Sélectionnez tout le texte, puis appuyez sur la barre d'espace. La liste des paramètres d'entrée s'affiche. Choisissez « Nombre de vols au cours du dernier mois pour le pass », puis appuyez sur « Entrée ». L' *expression* de la colonne 2 est définie sur « Nombre de vols au cours du dernier mois pour le pass ».
7	Cliquez-droit sur l'expression de la colonne 2 et définissez son type de données sur « nombre ».
8	Modifiez le nom de la sortie tableau de décision. Remplacez « Sortie 1 » par « Mettre à niveau la cabine », puis appuyez sur « Entrée ».
9	Définissez le type de données de la sortie de décision. Cliquez-droit sur l'en-tête de la colonne de sortie et choisissez ' string '.
10	Définissez les valeurs autorisées pour la sortie de décision. Dans la cellule située directement sous l'en-tête de la colonne de sortie (mais au-dessus de la ligne 1), définissez les valeurs autorisées pour la sortie. Saisissez « Classe Métier , Première Classe ». **Note** : il n'est pas nécessaire de mettre des guillemets autour des valeurs, car le type de données a été spécifié comme « string ».

Définir les règles du Tableau de Décision

Saisissez des valeurs dans les cellules tableau pour correspondre à cette image.

Cabin Upgrade Policy | Input Parameter Values for Simulation

(Flights in the last month, Number of Pax Overbooked)

U	Flights in the last month	Number of Pax Overbooked	Upgrade Cabin	Annotation
			Business Class, First Class	
1	<=1	<=2	*Business Class*	
2	<=1	(2..8]	*Business Class*	
3	<=1	>8	*First Class*	*Start Filling First Class when heavily overbooked*
4	(1..5]	<=2	*Business Class*	
5	(1..5]	(2..8]	*Business Class*	
6	(1..5]	>8	*First Class*	
7	>5	<=2	*Business Class*	
8	>5	(2..8]	*Business Class*	
9	>5	>8	*First Class*	*Reward Frequent Flyers*

Cliquez sur une cellule pour la sélectionner, puis cliquez à nouveau pour la modifier.

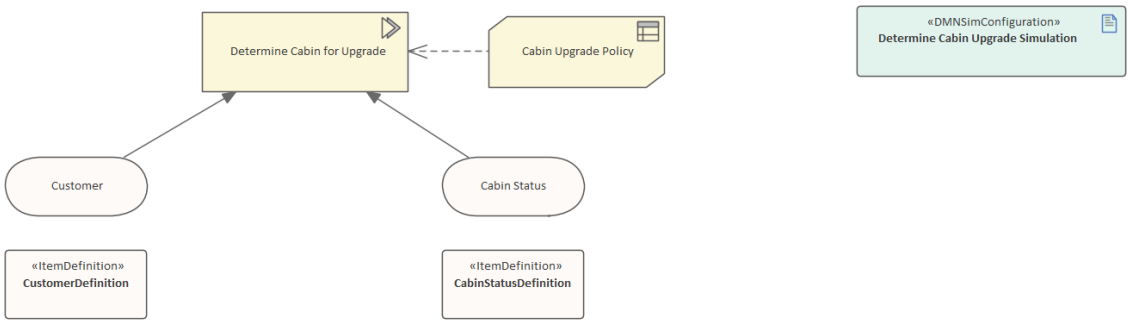
Vous pouvez copier et coller des règles existantes en sélectionnant les lignes à copier (Maj+clic ajoute à la sélection), cliquez-droit et choisissez « Copier », puis cliquez-droit et choisissez « Ajouter ».

Une fois que vous avez terminé de modifier les règles, cliquez sur le bouton Enregistrer .

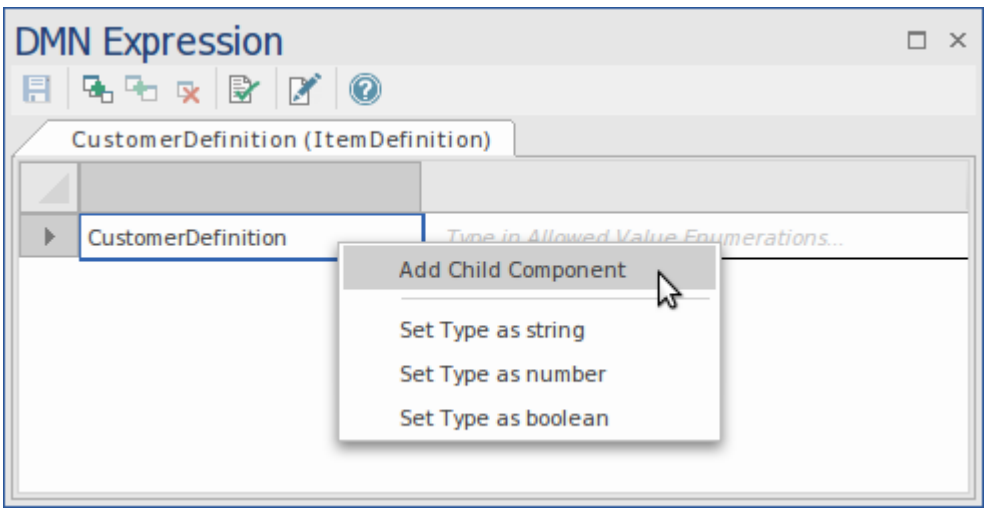
Enfin, cliquez sur le bouton Valider , pour vérifier les erreurs dans le tableau des règles.

Créer des éléments de définition d'élément

Ajoutez deux éléments ItemDefinition au diagramme , un pour chacun des éléments InputData. Nommez un élément « CustomerDefinition » et l'autre « CabinStatusDefinition ».



Double-cliquez sur l'élément ItemDefinition nommé « CustomerDefinition » pour modifier la définition. La fenêtre Expression DMN s'affiche.



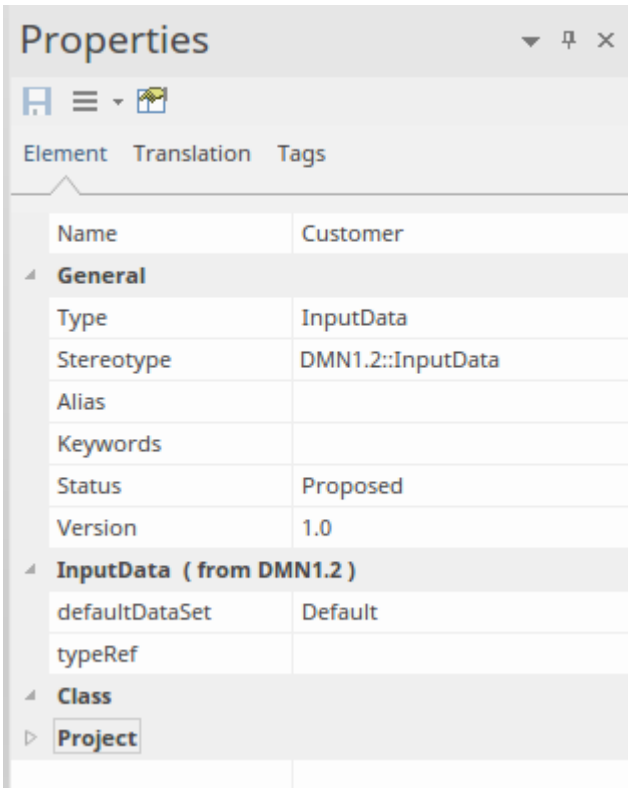
Cliquez-droit sur la cellule 'CustomerDefinition' et choisissez 'Ajouter un composant enfant'. Remplacez le nom du composant enfant par 'Nombre de vols le mois dernier' et remplacez son type de données par 'nombre'. Cliquez sur le

bouton 'Enregistrer' pour enregistrer les modifications et fermez la fenêtre.

De la même manière, double-cliquez sur l'élément ItemDefinition nommé « CabinStatusDefinition », ajoutez un composant enfant nommé « Num of Pax Overbooked » et définissez son type de données sur « number ». Enregistrez les modifications et fermez la fenêtre.

Spécifiez le Type de données pour chaque élément InputData

Sélectionnez l'élément InputData 'Customer'. Dans la fenêtre Propriétés , sélectionnez la propriété 'typeRef' et cliquez sur le bouton ... .



Sélectionnez l'élément « Définition client » comme type. Cliquez sur « OK ».

De même, spécifiez « Définition du statut de la cabine » comme type pour « Statut de la cabine ».

Spécifier les entrées de l'élément Décision

Double-cliquez sur l'élément de décision « Déterminer la cabine pour la mise à niveau »

Dans la fenêtre Expression DMN, recherchez la ligne tableau contenant le texte « Nombre de passagers surréservés » dans la première colonne. Cliquez dans la cellule de la deuxième colonne de cette ligne et appuyez sur la barre d'espace. Une liste de valeurs d'entrée possibles s'affiche. Choisissez « Statut de la cabine . Nombre de passagers surréservés » et appuyez sur « Entrée ». La sélection est écrite dans la cellule.

Répétez ce processus pour la deuxième ligne tableau « Nombre de vols au cours du mois dernier », en choisissant « Client ». Nombre de vols au cours du mois dernier ».

Cliquez sur le bouton Enregistrer.

Cliquez sur le bouton Valider.

Définir Ensembles de données

L'« exactitude » de votre modèle de décision peut être testée en exécutant des simulations à l'aide d'une gamme d'ensembles de données représentatifs pour vérifier que le modèle produit le résultat correct dans toutes les situations.

Vous pouvez créer de nombreux Ensembles de données portant différents noms, en utilisant une plage de valeurs de données. Vous pouvez définir l'un des ensembles de données comme *valeur par défaut* .

Nous allons maintenant créer un ensemble de données pour chacun de nos éléments InputData.

Étape	Description
1	Double-cliquez sur l'élément InputData « Client ». La fenêtre Expression DMN s'affiche.
2	Dans la fenêtre Expression DMN, cliquez sur le bouton « Modifier l'ensemble de données »  . La fenêtre « Modifier l'ensemble de données » s'affiche.
3	Cliquez sur le bouton  . Un nouvel ensemble de données est créé.
4	Remplacez le nom de l'ensemble de données si vous le souhaitez. Laissez le Type « nombre ». Entrez une valeur de 3, par exemple. Cliquez sur l'icône Enregistrer et sur le bouton OK .
5	Répétez l'opération pour l'entrée « Statut de la cabine ». Entrez une valeur de 4, par exemple.

Ajouter un artefact DMNSimConfiguration

Localisez l'artefact « Configuration Simulation » DMN dans la boîte à outils Diagramme . Déposez-en également un sur le diagramme .

Double-cliquez dessus pour ouvrir la fenêtre Simulation DMN dans l'onglet « Simuler ».

Depuis la fenêtre Simulation DMN, vous pouvez exécuter des simulations du Décision Modèle terminé. Vous pouvez également effectuer des validations, générer du code et générer des modules de test.

Étape	Description
1	Localisez le champ d'édition dans la barre d'outils de cette fenêtre.
2	Cliquez sur la flèche déroulante dans ce champ. Une liste s'affiche, indiquant tous les services Décision et les éléments Décision du Paquetage associé à l'artefact de configuration DMNSim. Dans ce cas, « Déterminer la cabine à mettre à niveau » est le seul élément de la liste.
3	Cliquez sur « Déterminer la cabine pour la mise à niveau ».
4	Le corps de la fenêtre affiche maintenant les éléments InputData et les résultats de décision disponibles comme entrées pour la décision sélectionnée.

	Cliquez sur le bouton Enregistrer.
5	Utilisez la colonne 'Valeur' pour sélectionner l'un des DataSets prédéfinis pour les InputValues, puis vous pouvez cliquer sur le bouton ' Exécuter '  dans la barre d'outils inférieure pour exécuter une simulation, en utilisant les ensembles de données sélectionnés.

Diagrammes Exigences Décision

Les éléments modélisés dans les graphes Décision Exigences (DRG) et diagrammes Décision Exigences (DRD) sont Décision , Métier Knowledge Modèle , Input Data, Knowledge Source et Décision Service. Les dépendances entre ces éléments expriment trois types d'exigences : Information, Connaissance et Autorité.

Composantes des Diagrammes Exigences Décision

Ce tableau résume la notation de toutes les composantes d'un diagramme Décision Exigences .

Composant	Description
Décision	Un élément Décision désigne l'acte de déterminer une sortie à partir d'un certain nombre d'entrées (données d'entrée ou Décision), en utilisant une logique de décision exprimée sous forme d'expressions littérales, Tableaux Décision , d'invocations ou de contexte encadré.
Modèle de connaissances Métier	Un Modèle de connaissances Métier désigne un module réutilisable de logique de décision représenté par une fonction, qui comprend zéro, un ou plusieurs paramètres.
Décision Service (élargi)	Un service Décision peut contenir un ensemble de décisions réutilisables qui sont invoquées en interne - par exemple, par un autre Modèle Décision ou Métier - ou en externe - par exemple, par un processus BPMN. Une bonne pratique consiste à utiliser un diagramme pour décrire un seul service Décision étendu.
Décision Service (réduit)	Si un élément du service Décision sert d'élément invocable, connecté avec des exigences de connaissances à d'autres éléments avec une logique d'invocation, nous pouvons masquer les détails du service Décision pour nous concentrer sur les hiérarchies de décision dans leur ensemble.
Données d'entrée	Un élément de données d'entrée désigne les informations utilisées comme entrée pour une ou plusieurs décisions.
Définition Item	Une définition Item est utilisée pour définir le type et la structure des éléments de données utilisés dans le modèle de décision. Elle est principalement référencée par les éléments de données d'entrée comme base pour le type et la structure des données qui devraient être saisies. Elle peut également être référencée pour définir la structure d'une sortie. La définition Item contient Ensembles de données qui fournissent des ensembles de valeurs utiles lors de la réalisation de simulations variées.
Source de connaissances	Un élément Knowledge Source désigne une autorité pour un Métier Knowledge Modèle ou Décision .
Exigences en matière d'information	Une exigence d'information désigne des données d'entrée ou une sortie Décision utilisées comme entrée dans une Décision .
Exigences en matière de connaissances	Une Exigence de Connaissances désigne l’invocation d’un Modèle de Connaissances Métier ou d’un Service Décision .
Exigence d'autorité	Une exigence d’autorité dénote la dépendance d’un élément DRG envers un autre

	élément DRG qui agit comme source de guidage ou de connaissances.

Éditeur d'expressions de Décision

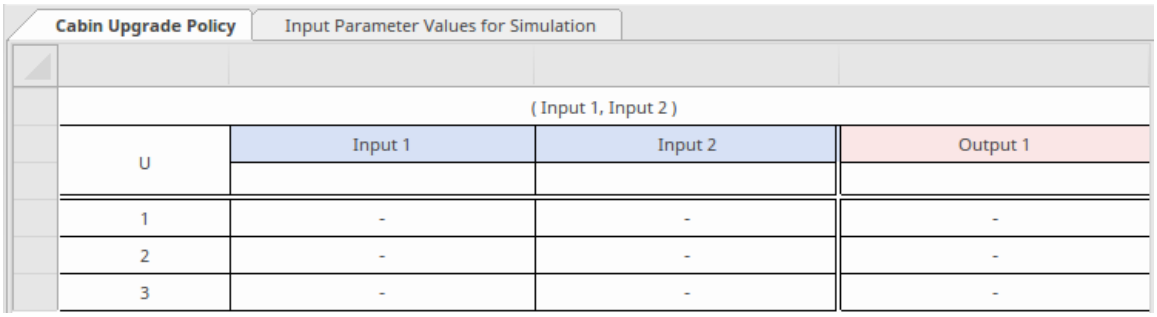
L'éditeur d'expressions DMN est la fenêtre dans laquelle vous définirez, révision et mettrez à jour les détails de la plupart des différents types d'éléments DMN de votre modèle. Il est principalement utilisé pour éditer les expressions de valeur des éléments Décision et des éléments BusinessKnowledgeModel (BKM).

Une version différente de l'éditeur d'expression DMN est affichée pour chacun des quatre types d'expression valeur utilisés par les éléments Décision et les éléments BKM. Pour les éléments BKM, un deuxième onglet de fenêtre est également présenté, pour définir les paramètres d'entrée et de sortie utilisés lors de l'appel du BKM.

Deux versions supplémentaires de l'éditeur d'expressions DMN existent également pour support l'édition des éléments ItemDefinition et InputData.

La barre d'outils affichée et la disposition du contenu de la fenêtre dépendent du type d'élément DMN actuellement sélectionné et, le cas échéant, du type d'expression de valeur défini.

Cette image montre la version de l'éditeur d'expressions DMN utilisée pour définir un Tableau de Décision . Dans ce cas, l'élément sous-jacent est un BusinessKnowledgeModel, et donc la logique de décision est « invoquée » par d'autres éléments, avec une entrée et une sortie transmises via des paramètres.



Des explications détaillées des fonctionnalités de l'éditeur d'expressions DMN pour chaque élément et type d'expression sont fournies dans les rubriques d'aide enfants de cette rubrique.

Accéder

Diagramme	Double-cliquez sur un élément DMN sur un diagramme . La fenêtre de l'éditeur d'expression DMN correspondant à l'élément et à son type d'expression s'affiche.

Expressions de valeur

Ce tableau résume quatre types distincts d'expression valeur avec des références aux rubriques d'aide détaillant chacune d'elles.

Type et Icône	Description
Tableau de Décision	Un Tableau de Décision est une représentation tabulaire d'un ensemble d'expressions d'entrée et de sortie liées, organisées en règles indiquant quelle entrée de sortie s'applique à un ensemble spécifique d'entrées d'entrée.
Expression littérale	Une expression littérale spécifie la logique de décision sous la forme d'une expression textuelle qui décrit comment une valeur de sortie est dérivée de ses valeurs d'entrée. Pour support la simulation et l'exécution, l'expression littérale peut

	utiliser des fonctions JavaScript .
Contexte encadré	Un contexte encadré est une collection d'entrées de contexte, composée de paires (nom, valeur), chacune avec un résultat valeur . Les entrées de contexte fournissent un moyen de décomposer une expression complexe en une série d'expressions simples, fournissant des résultats intermédiaires qui peuvent être utilisés dans les entrées de contexte suivantes.
Invocation	Une invocation fait appel à un autre élément de modèle (un BusinessKnowledgeModel ou un Décision Service) pour fournir un résultat de décision. L'invocation définit des paramètres qui sont transmis à l'élément « invoqué », fournissant un contexte pour l'évaluation de sa logique de décision. Le résultat de la décision est ensuite renvoyé à l'élément « appelant ».

Éléments ItemDefinition et InputData

Élément	Description
Définition de l'élément	Les éléments ItemDefinition sont utilisés pour définir des structures de données et, éventuellement, pour restreindre la plage de valeurs autorisées des données. Les ItemDefinitions peuvent aller d'un type simple à un type structuré complexe. Les ItemDefinitions sont utilisées pour spécifier le type des éléments InputData ainsi que les paramètres d'entrée.
Données d'entrée	Les éléments InputData sont utilisés pour fournir des entrées aux éléments Décision . Le type de données d'un élément InputData est défini à l'aide d'un élément ItemDefinition. Ensembles de données peuvent également être définis dans le cadre d'un ItemDefinition et un élément InputData peut alors spécifier un ensemble de données à utiliser lors de l'exécution d'une simulation.

Tableau de Décision

Un Tableau de Décision est une représentation tabulaire d'un ensemble d'expressions d'entrée et de sortie liées, organisées en règles indiquant quelle entrée de sortie s'applique à un ensemble spécifique d'entrées d'entrée.

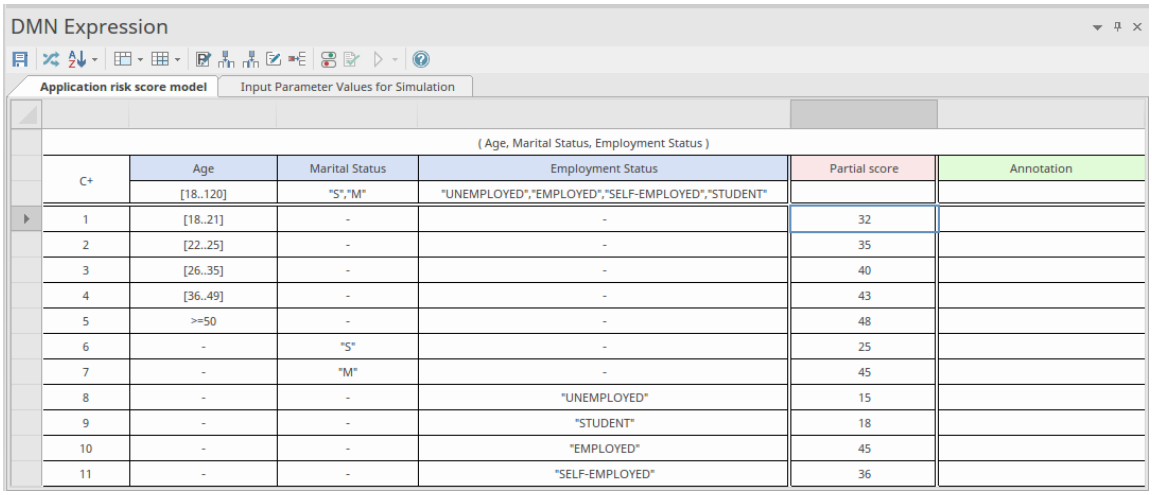
Tableaux Décision sont pris en charge par les types d'éléments *Décision* et *Métier Knowledge Modèle* . Ils sont indiqués par l'icône dans le coin supérieur droit de l'élément sur un diagramme .

Accéder

Diagramme	Sur un diagramme , double-cliquez sur un élément Décision ou sur un élément BusinessKnowledgeModel. La fenêtre Expression DMN s'affiche, affichant les détails de l'élément sélectionné.

Aperçu

Cette image montre la fenêtre Expression DMN telle qu'elle apparaît pour un Tableau de Décision .



Un Tableau de Décision se compose de :

- La politique Hit Tableau (C+, U, A, P, etc.) qui spécifie comment les règles sont appliquées
- Une liste de règles (1, 2, 3, 4, etc.), où chaque ligne de règle contient des entrées d'entrée spécifiques et des entrées de sortie correspondantes
- Une liste de clauses d'entrée (sous les titres bleus), définies comme des expressions qui impliquent généralement une ou plusieurs valeurs d'entrée
- Une liste de clauses de sortie (sous le titre rose), définissant la sortie correspondant à un ensemble spécifique d'entrées
- Annotations facultatives pour chaque règle (sous le titre vert) que vous pouvez ajouter via la barre d'outils de la fenêtre

Une clause d'entrée se compose d'une expression et d'une liste facultative de valeurs autorisées (la ligne située juste en dessous des en-têtes de colonne). Très souvent, l'expression est simplement une valeur d'entrée non modifiée ; cependant, il peut également s'agir d'une expression impliquant plusieurs valeur d'entrée ou peut-être d'une instruction conditionnelle telle que « Score de risque d'application > 100 ». Les valeurs autorisées s'appliquent au *résultat de*

l'expression plutôt qu'aux valeurs d'entrée utilisées.

Chaque clause de sortie se compose d'un identifiant (un nom) et d'une liste facultative de valeurs autorisées pour cette clause.

Le Tableau de Décision doit contenir toutes les entrées - et uniquement ces entrées - nécessaires pour déterminer une sortie.

Pour déterminer quelles règles sont appliquées, les expressions définies dans les clauses d'entrée sont évaluées pour les entrées données et les *résultats de l'expression* sont ensuite utilisés pour trouver des règles avec des entrées d'entrée correspondantes.

Lorsque la fenêtre Expression DMN n'est pas suffisamment large ou profonde pour afficher toutes les colonnes et lignes, vous pouvez utiliser les barres de défilement pour accéder au contenu masqué ou faire glisser les bordures vers l'extérieur pour augmenter la largeur de chaque colonne. Les largeurs des colonnes « Entrée » et « Sortie » sont initialement les mêmes, mais vous pouvez ajuster la largeur de chaque colonne indépendamment des autres, en faisant glisser la bordure de la colonne soit dans le tableau , soit dans la barre grise située juste en dessous des noms des onglets.

Barre d'outils pour l'éditeur Tableau de Décision

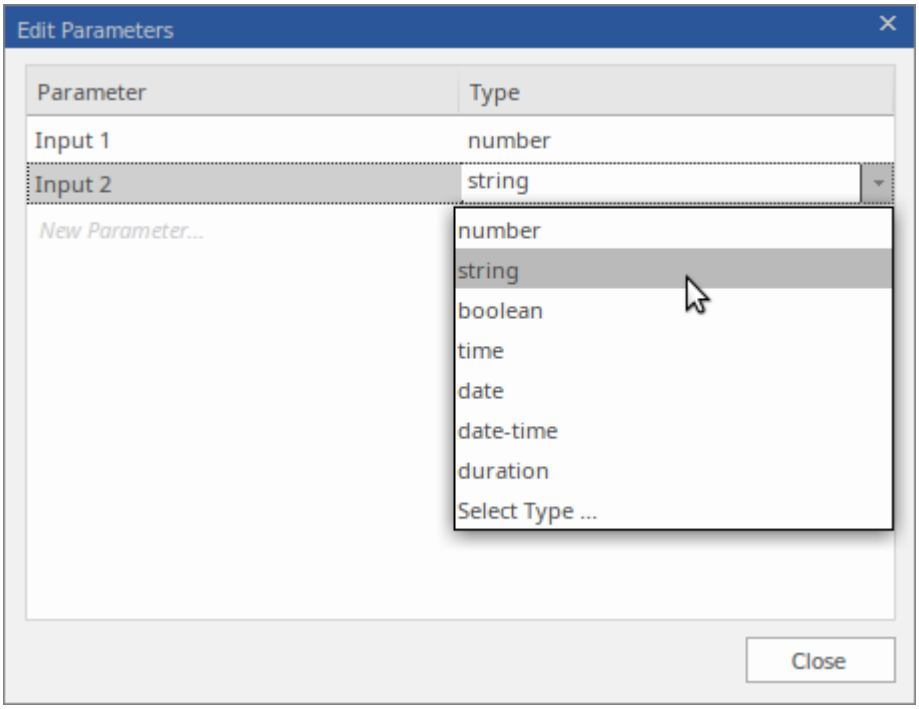
Lorsqu'un Tableau de Décision est sélectionné, les fonctionnalités disponibles dans la fenêtre Expression DMN sont accessibles via la barre d'outils en haut de la fenêtre, comme indiqué :

Pour plus de détails, reportez-vous à la rubrique d'aide de la *barre d'outils de l'éditeur Tableau de Décision* .

Paramètres

Dans le cas des éléments Métier Knowledge Modèle (BKM), des paramètres sont utilisés pour transmettre les valeurs d'entrée fournies par l'élément appelant. La logique de décision du BKM est évaluée à l'aide des paramètres d'entrée et le résultat est renvoyé à l'élément appelant. Par défaut, un élément BKM est créé avec deux paramètres d'entrée, « Entrée 1 » et « Entrée 2 ».

Cliquez sur l'icône  dans la barre d'outils de la fenêtre Expression DMN pour afficher la dialogue « Modifier les paramètres ».



Ici, vous pouvez modifier les noms des paramètres, réorganiser la séquence, définir leurs types de données, créer des paramètres supplémentaires ou supprimer ceux existants.

Politique Hit

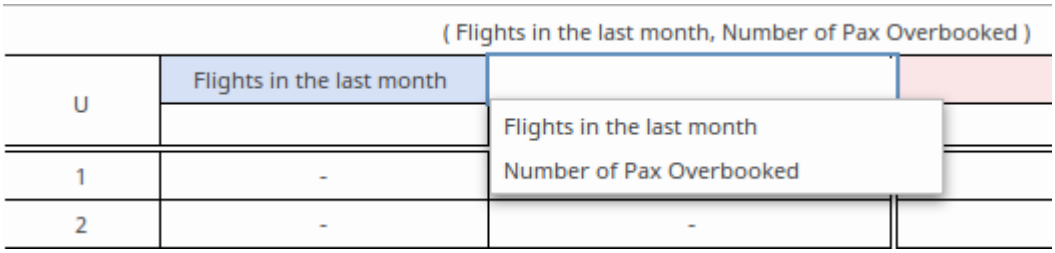
Cliquez-droit sur l'indicateur ' Hit Policy Indicator', puis choisissez la Hit Policy souhaitée dans le menu contextuel. Les différentes Hit Policy Tableau sont décrites en détail dans la rubrique d'aide *Hit Policy Tableau de Décision* .

Clauses d'entrée

Une clause d'entrée d'un Tableau de Décision est définie comme une expression. Très souvent, l'expression est simplement une valeur d'entrée non modifiée ; cependant, il peut également s'agir d'une expression impliquant plusieurs valeur d'entrée ou être définie comme une instruction conditionnelle, telle que « Score de risque d'application > 100 ». Les valeurs autorisées s'appliquent au *résultat de l'expression* plutôt qu'aux valeurs d'entrée utilisées et, par conséquent, le type des valeurs doit correspondre au type du résultat de l'expression.

Tableaux Décision sont créés avec deux clauses d'entrée par défaut, « Entrée 1 » et « Entrée 2 ». Le type de données pour ces deux clauses est « nombre ». Dans la fenêtre Expression DMN, les clauses d'entrée sont affichées sous forme d'en-têtes de colonne sur le Tableau de Décision . Pour modifier une clause d'entrée, cliquez sur l'en-tête de colonne pour sélectionner la cellule, puis cliquez à nouveau ou appuyez sur F2 pour modifier.

La saisie semi-automatique est prise en charge lors de la modification des clauses d'entrée. Cela signifie que pour les éléments Décision , toutes les entrées connectées à l'élément Décision sont disponibles pour sélection dans une liste. De même, pour les éléments Métier Knowledge Modèle , les paramètres d'invocation sont disponibles pour sélection dans une liste. Pour plus d'informations, consultez la rubrique d'aide *sur la saisie semi-automatique des expressions DMN* .



Pour ajouter des colonnes supplémentaires d'entrées au Tableau de Décision , cliquez sur l'icône dans la barre d'outils de la fenêtre Expression DMN.

Pour supprimer les colonnes d'entrée du tableau , cliquez-droit dans la colonne d'entrée indésirable, puis sélectionnez l'option « Supprimer la colonne d'entrée » dans le menu contextuel.

L'ordre des colonnes d'entrée dans le tableau peut être réorganisé en faisant glisser et en déposant les colonnes vers de nouvelles positions. (Faites glisser la cellule non étiquetée tout en haut de la colonne tableau vers la position requise.)

Valeurs autorisées

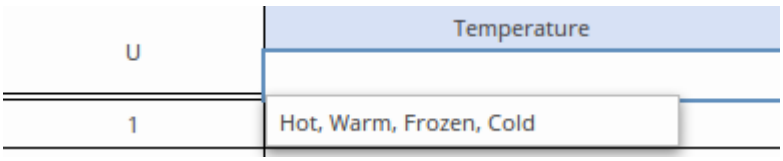
Lors de la définition d'une colonne « Entrée » ou « Sortie », la deuxième ligne de la colonne définit les valeurs autorisées. Il s'agit d'une cellule facultative dans la colonne, mais utile pour clarifier les entrées dans les lignes situées en dessous. Lors de l'exécution d'une validation, chacune des cellules situées sous la cellule Valeurs autorisées est vérifiée pour s'assurer qu'elle est conforme à l'expression de cette cellule.

Les expressions utilisées dans cette cellule dépendent de la manière dont la colonne « Entrée » ou « Sortie » est typée. Par exemple :

- Numéro - [18 ..35]
- String - « Élevé », « Faible », « Moyen »
- Booléen - vrai, faux
- Indéfini - '-'

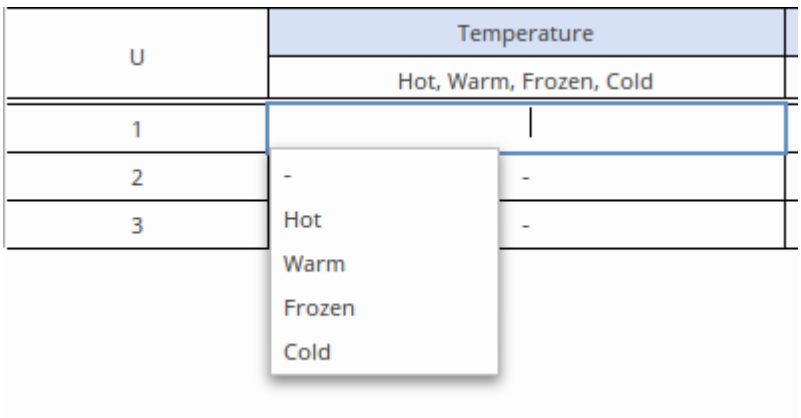
Valeurs autorisées pour le remplissage rapide

L'expression d'entrée/sortie à laquelle cela fait référence peut être une valeur simple ou une expression FEEL complexe ; cependant, si elle est directement liée au champ « Valeurs autorisées » d'un ItemDefinition, appuyer sur la barre d' espace activera une option de remplissage rapide pour définir les « Valeurs autorisées » telles que définies dans l'ItemDefinition (généralement référencées via un élément InputData).



Remplissage rapide des lignes

Une fois le champ « Valeurs autorisées » défini, en plus de restreindre les valeurs pouvant être utilisées lors de la définition des règles dans le tableau , le champ « Valeurs autorisées » fournit également à l'utilisateur une option de remplissage rapide. Cette option est invoquée, dans une cellule de règle, en appuyant sur la barre d'espace et en sélectionnant l'élément requis :



Pour plus de détails, consultez la rubrique d'aide *Complétion automatique des expressions DMN* .

Clauses de sortie

Une clause de sortie se compose d'un nom, d'un type de données et d'une liste facultative de valeurs autorisées. Pour modifier une clause de sortie, cliquez sur la cellule d'en-tête de colonne pour sélectionner la cellule, puis cliquez à nouveau ou appuyez sur F2 pour modifier.

Pour ajouter des colonnes supplémentaires d'entrées de sortie au Tableau de Décision , cliquez sur l'icône dans la barre d'outils de la fenêtre de l'éditeur d'expressions.

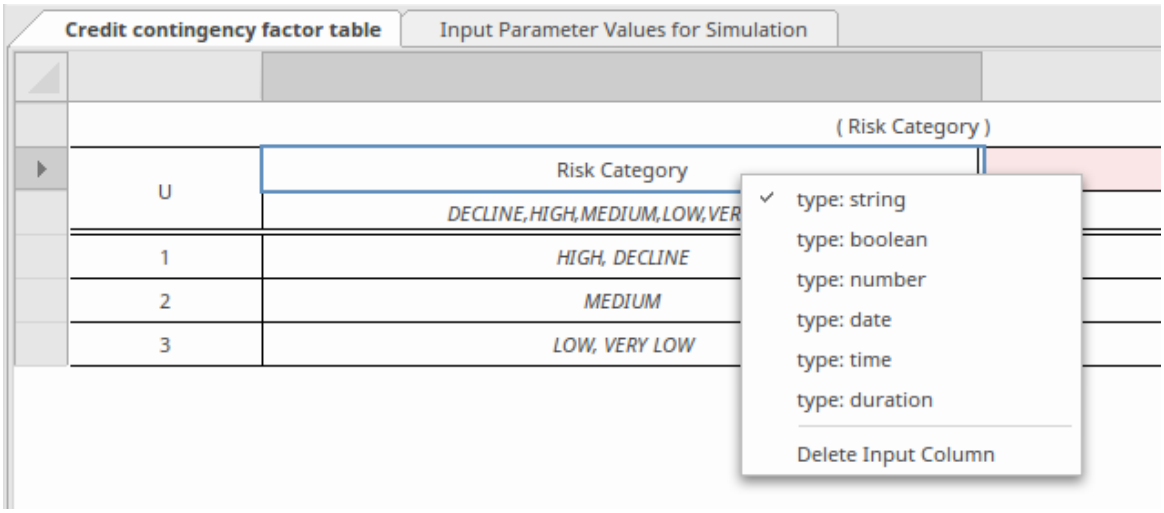
Pour supprimer les colonnes de sortie du tableau , cliquez-droit dans la colonne de sortie indésirable, puis sélectionnez l'option « Supprimer la colonne de sortie » dans le menu contextuel.

L'ordre des colonnes du tableau peut être réorganisé en faisant glisser les colonnes vers de nouvelles positions. (Faites glisser la cellule sans étiquette tout en haut de la colonne tableau vers la position souhaitée.)

Type de données pour les clauses d'entrée/sortie

Pour que la simulation fonctionne, il est essentiel de définir le type de données pour toutes les clauses d'entrée et de sortie. Les validations de plage, d'écart et de chevauchement sont prises en charge pour les clauses de type « nombre », mais la validation ne peut pas être effectuée si le type n'a pas été spécifié. La génération de code pour les langages typés tels que C++, C# et Java nécessite que les types de données soient spécifiés. Lorsque le type de données est spécifié comme « string », il n'est pas nécessaire d'entourer chaque littéral string de guillemets. Les valeurs String sont affichées en italique si le type a été déclaré.

Pour définir le type de données, cliquez-droit sur l'en-tête de la colonne Entrée ou Sortie et sélectionnez le type requis dans la liste.



Définir les règles Tableau de Décision

Les règles Tableau de Décision sont définies en spécifiant des entrées d'entrée et des entrées de sortie correspondantes dans les cellules d'une ligne tableau . Pour les types de données « nombre », les entrées d'entrée peuvent être spécifiées sous forme de valeur unique ou de plage de nombres, telle que « <10 », « >100 » ou « [2..8) ». (Lors de la définition de plages de nombres, l'utilisation de round indique que le nombre limite n'est PAS inclus ; l'utilisation de crochets indique que le nombre limite est inclus.) Les entrées de sortie doivent spécifier une seule valeur par cellule.

Des règles supplémentaires peuvent être ajoutées à la liste des règles en cliquant sur l'icône dans la barre d'outils. Les règles indésirables peuvent être supprimées du tableau en cliquant avec le bouton droit de la souris sur la règle et en sélectionnant l'option « Supprimer la ligne de règle » dans le menu contextuel.

Les règles existantes peuvent être copiées et collées dans le tableau en sélectionnant d'abord les règles (utilisez « Ctrl+Clic » pour ajouter/supprimer de la sélection), puis en utilisant les options de menu « Copier les règles dans le presse-papiers » et « Coller les règles depuis le presse-papiers » pour effectuer le copier-coller. Les règles copiées peuvent ensuite être modifiées en sélectionnant et en modifiant les entrées de cellule individuelles.

Si le champ « Valeurs autorisées » est défini pour une string ou une expression booléenne, la barre d'espace peut être utilisée pour afficher une liste de valeurs parmi lesquelles sélectionner, comme indiqué dans la section *Valeurs autorisées - Remplissage rapide des lignes* précédente.

Les règles peuvent également être triées dans le tableau , soit par :

- Cliquez sur l'icône dans la barre d'outils, puis choisissez « Trier par entrée » ou « Trier par sortie », ou
- Cliquez avec le bouton droit sur les règles individuelles dans le tableau et sélectionnez l'option « Déplacer la règle vers le haut » ou « Déplacer la règle vers le bas » dans le menu contextuel

Pour déterminer les lignes tableau sélectionnées pour la sortie, les *expressions* définies par les clauses d'entrée sont évaluées pour les entrées données et les *résultats* des expressions sont ensuite comparés aux entrées d'entrée des lignes tableau . Lorsque les résultats de l'expression correspondent aux entrées d'entrée d'une ligne tableau , cette ligne est sélectionnée pour la sortie.

La « politique Hit » du Tableau de Décision détermine comment les lignes correspondantes du tableau sont ensuite utilisées pour produire sa sortie.

Formats des règles

Vous pouvez choisir - à l'aide d'une icône de la barre d'outils - d'afficher le Tableau de Décision dans l'un des trois formats, comme indiqué ici.

Format règle sous forme de ligne, où la règle est développée le long des lignes avec les entrées, les sorties et les

annotations définies dans les colonnes :

(Existing Customer, Application Risk Score)				
U	Existing Customer	Application Risk Score	Pre-Bureau Risk Category	Annotations
	true,false		HIGH, MEDIUM, LOW, VERY LOW, DECLINE	
1	true	<80	DECLINE	Use standard letter DEC0004
2	true	[80..90)	HIGH	
3	true	[90..110]	MEDIUM	
4	true	>110	LOW	
5	false	<100	HIGH	
6	false	[100..120)	MEDIUM	
7	false	[120..130]	LOW	
8	false	>130	VERY LOW	Refer to Loans Officer

Format règle sous forme de colonne, où les règles sont développées en colonnes avec les entrées, les sorties et les annotations définies le long des lignes :

(Existing Customer, Application Risk Score)									
Existing Customer	true,false	true	true	true	true	false	false	false	false
Application Risk Score		<80	[80..90)	[90..110]	>110	<100	[100..120)	[120..130]	>130
Pre-Bureau Risk Category	HIGH, MEDIUM, LOW, VERY LOW, DECLINE	DECLINE	HIGH	MEDIUM	LOW	HIGH	MEDIUM	LOW	VERY LOW
Annotations		Use standard letter DEC0004							Refer to Loans Officer
U		1	2	3	4	5	6	7	8

Format de règle sous forme de tableau croisé, où les règles sont formées à partir d'entrées définies comme un ensemble de lignes AND une combinaison de colonnes, avec des sorties définies dans les cellules qui se croisent. (Note que ce format masque les champs « Annotation ») :

(Existing Customer, Application Risk Score)								
Pre-Bureau Risk Category		Application Risk Score						
		<80	[80..90)	[90..110]	>110	<100	[100..120)	[120..130]	>130
Existing Customer	false					HIGH	MEDIUM	LOW	VERY LOW
	true	DECLINE	HIGH	MEDIUM	LOW				

A la fin d'une simulation, dans un Tableau de Décision croisé, les entrées et les sorties associées sont mises en évidence. Par exemple, dans cette simulation, le traitement a abouti à la sortie d'une remise de 0,10 pour un client Métier dont la taille de la commande était inférieure ou égale à 10 et la livraison n'était pas applicable.

Paramètres du tableau croisé

Dans le format Rule-as-Crosstab, comme les entrées forment à la fois des lignes et des colonnes et que les sorties se trouvent aux intersections, les étapes de définition des valeurs sont légèrement différentes de celles des deux autres formats.

1. Pour ajouter un autre type d'entrée, cliquez-droit sur l'en-tête de la colonne d'entrée et sélectionnez l'option « Ajouter une entrée ». Vous êtes invité à saisir le nom de l'entrée ; l'entrée est ajoutée sous la forme d'un ensemble de champs sous les champs de la colonne actuelle.
   Pour supprimer un type d'entrée des colonnes, cliquez-droit sur son ensemble de champs et sélectionnez l'option 'Supprimer l'entrée'. Le nom de l'entrée et son ensemble de champs sont supprimés des en-têtes de colonnes.
2. Pour ajouter un autre type de sortie, cliquez-droit sur le bloc Sortie en haut à gauche de la fenêtre et sélectionnez l'option 'Ajouter une sortie'. Vous êtes invité à saisir le nom de la sortie ; le nom est ajouté au bloc Sortie et une nouvelle ligne est ajoutée à chaque cellule du corps de la fenêtre.
   Pour supprimer un type de sortie, cliquez-droit sur le nom du type dans le bloc Sortie en haut à gauche de la fenêtre

et sélectionnez l'option 'Supprimer la sortie'. Le nom de la sortie et ses champs dans la grille sont supprimés.

3. Vous pouvez basculer entre les types d'entrée pour en sélectionner un pour les en-têtes de ligne. Cliquez-droit sur la ligne et cliquez sur l'option « Sélectionner l'entrée comme en-tête de ligne ». Cela affiche une liste des types d'entrée ; cliquez sur le type à utiliser comme en-tête de ligne ; les autres types sont combinés dans les colonnes.
4. Pour ajouter une ligne ou une colonne de saisie valeur aux entrées, cliquez-droit sur une ligne ou une colonne en cours et sélectionnez l'option « Ajouter une ligne de saisie de valeur » ou « Ajouter une colonne de saisie de valeur », selon le cas. Une prompt s'affiche pour indiquer le nom de l'entrée de valeur ; lorsque vous saisissez ce nom, la ligne ou la colonne appropriée est ajoutée au Tableau de Décision .
   Pour supprimer une ligne ou une colonne de saisie valeur , cliquez-droit dessus et sélectionnez l'option « Supprimer la ligne de saisie de valeur » ou « Supprimer la colonne de saisie de valeur ». La ligne ou la colonne sélectionnée est supprimée du tableau .
5. Dans les colonnes Entrée, chaque ligne correspond à un type d'entrée. Si vous souhaitez déplacer la ligne d'un type d'entrée au-dessus ou en dessous d'un autre, cliquez-droit dessus et sélectionnez l'option « Déplacer l'entrée vers le haut » ou « Déplacer l'entrée vers le bas ». Le menu contextuel ne fournit que les options qui peuvent être actionnées, donc comme il n'est pas possible de déplacer, par exemple, la dernière ligne vers le bas, l'option « Déplacer l'entrée vers le bas » n'est pas répertoriée.

Barre d'outils pour l'éditeur Tableau de Décision

Ce tableau fournit des descriptions des fonctionnalités accessibles dans la fenêtre Expression DMN lorsqu'un Tableau de Décision est sélectionné.

Options de la barre d'outils

Icône	Description
	Enregistrez les modifications apportées à l'élément Décision ou BusinessKnowledgeModel actuellement sélectionné.
	Basculez la vue du Tableau de Décision entre les modes Règle en tant que ligne, Règle en tant que colonne et Règle en tant que tableau croisé. Vous pouvez également cliquer sur la flèche déroulante et sélectionner le format souhaité.
	Cliquez sur « Trier par entrée » pour trier les règles par colonnes d'entrée ; cliquez sur « Trier par sortie » pour trier les règles par colonnes de sortie. Les colonnes peuvent être glissées et déposées pour organiser l'ordre de tri.
	Fusionner les cellules de règles adjacentes, où le contenu des entrées d'entrée est le même. Vous pouvez modifier le contenu des cellules fusionnées. Pendant la simulation, les éléments fusionnés sont mis en surbrillance.
	Diviser les cellules d'entrée qui ont été précédemment fusionnées.
	Affichez la fenêtre « Modifier les paramètres », dans laquelle vous pouvez spécifier les noms et les types de données des paramètres transmis lors de l'appel de la logique de décision d'un élément BusinessKnowledgeModel.
	Ajouter une colonne d'entrée au Tableau de Décision .
	Ajoutez une colonne de sortie au Tableau de Décision .
	Ajoutez une colonne d'annotation (avec une cellule d'en-tête verte) au tableau , dans laquelle vous pouvez enregistrer de courtes notes ou commentaires sur la règle. (Voir les illustrations de la ligne Règle en tant que ligne/Règle en tant que colonne ci-dessus.) Vous pouvez ajouter plusieurs colonnes d'annotation si nécessaire, en saisissant un titre de colonne approprié dans chaque cellule d'en-tête. Pour supprimer une colonne d'annotation, cliquez-droit dessus et sélectionnez l'option « Supprimer la colonne d'annotation ».
	Joindre une règle au Tableau de Décision .
	Afficher ou masquer les champs de valeurs autorisées pour les colonnes « Entrée » et « Sortie ». Les valeurs autorisées définies pour une entrée ou une sortie seront utilisées pour la validation et l'édition de saisie semi-automatique.
	Effectuer la validation du Tableau de Décision . Enterprise Architect effectuera une

	série de validations pour vous aider à détecter d'éventuelles erreurs dans le Tableau de Décision .
	Ce bouton est activé lorsqu'un Tableau de Décision est défini pour un élément BusinessKnowledgeModel. Sélectionnez l'onglet « Valeurs des paramètres d'entrée pour Simulation », complétez les champs, puis cliquez sur ce bouton. Le résultat du test sera présenté sur le Tableau de Décision , avec les valeurs d'exécution des entrées et des sorties affichées et les règles valides mises en évidence. Vous pouvez utiliser cette fonctionnalité pour tester un élément BusinessKnowledgeModel, sans spécifier son contexte. Plusieurs options de menu sont disponibles pour ce bouton de la barre d'outils. Pour plus d'informations, consultez la rubrique d'aide *Simuler Modèle DMN* .

Tableau de Décision Hit Policy

La politique Hit spécifie le résultat du Tableau de Décision en cas de chevauchement des règles. Le caractère unique dans une cellule particulière Tableau de Décision indique le type tableau et reflète sans ambiguïté la logique de décision.

Politiques Hit uniques :

- **Unique** : aucun chevauchement n'est possible et toutes les règles sont disjointes ; une seule règle peut être mise en correspondance (c'est la valeur par défaut)
- **A** ny : il peut y avoir un chevauchement, mais toutes les règles de correspondance affichent des entrées de sortie égales pour chaque sortie, de sorte que n'importe quelle correspondance peut être utilisée
- **Priorité** : plusieurs règles peuvent correspondre, avec différentes entrées de sortie ; cette politique renvoie la règle correspondante avec la priorité de sortie la plus élevée
- **Premièrement** : plusieurs règles (qui se chevauchent) peuvent correspondre, avec différentes entrées de sortie ; le premier résultat par ordre de règle est renvoyé.

Politiques Hit multiples :

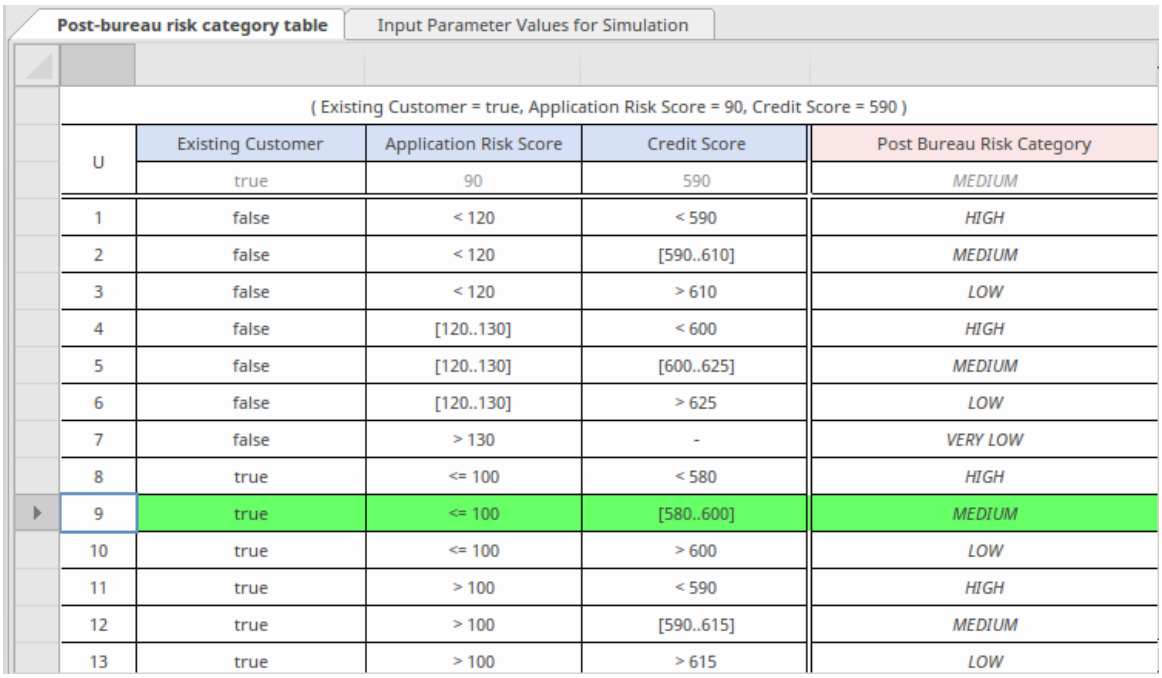
- **Ordre** de sortie : renvoie tous les résultats dans l'ordre de priorité de sortie décroissant
- Ordre des **règles** : renvoie tous les résultats dans l'ordre des règles
- **C** ollect : renvoie tous les résultats dans un ordre arbitraire ; un opérateur ('+', '<', '>', '#') peut être ajouté pour appliquer une fonction simple aux sorties

Les opérateurs de collecte sont :

- + (somme) : le résultat du Tableau de Décision est la somme de toutes les sorties distinctes
- < (min) : le résultat du Tableau de Décision est la plus petite valeur de toutes les sorties
- > (max) : le résultat du Tableau de Décision est la plus grande valeur de toutes les sorties
- # (count) : le résultat du Tableau de Décision est le nombre de sorties distinctes

Exemple de politique Hit unique

La politique Hit « Unique » est le type le plus populaire pour un Tableau de Décision et toutes les règles sont disjointes.



Post-bureau risk category table | Input Parameter Values for Simulation

(Existing Customer = true, Application Risk Score = 90, Credit Score = 590)

U	Existing Customer	Application Risk Score	Credit Score	Post Bureau Risk Category
	true	90	590	*MEDIUM*
1	false	< 120	< 590	*HIGH*
2	false	< 120	[590..610]	*MEDIUM*
3	false	< 120	> 610	*LOW*
4	false	[120..130]	< 600	*HIGH*
5	false	[120..130]	[600..625]	*MEDIUM*
6	false	[120..130]	> 625	*LOW*
7	false	> 130	-	*VERY LOW*
8	true	<= 100	< 580	*HIGH*
9	true	<= 100	[580..600]	*MEDIUM*
10	true	<= 100	> 600	*LOW*
11	true	> 100	< 590	*HIGH*
12	true	> 100	[590..615]	*MEDIUM*
13	true	> 100	> 615	*LOW*

Exemple de politique Hit priorité

Dans un tableau avec la stratégie Hit « Priorité », plusieurs règles peuvent correspondre avec différentes entrées de sortie. Cette stratégie renvoie la règle correspondante avec la priorité de sortie la plus élevée.

Eligibility rules | Input Parameter Values for Simulation

(Pre-Bureau Risk Category, Pre-Bureau Affordability, Age)

P	Pre-Bureau Risk Category	Pre-Bureau Affordability	Age	Eligibility
				INELIGIBLE,ELIGIBLE
1	*DECLINE*	-	-	*INELIGIBLE*
2	-	false	-	*INELIGIBLE*
3	-	-	< 18	*INELIGIBLE*
4	-	-	-	*ELIGIBLE*

Note : la liste des valeurs autorisées permet de définir la priorité de sortie. Ici, les valeurs autorisées sont répertoriées comme INÉLIGIBLE et ÉLIGIBLE ; INÉLIGIBLE est défini comme ayant une priorité plus élevée que ÉLIGIBLE.

Un résultat de simulation possible pourrait ressembler à ceci :

Eligibility rules | Input Parameter Values for Simulation

(Pre-Bureau Risk Category = "HIGH", Pre-Bureau Affordability = false, Age = 25)

P	Pre-Bureau Risk Category	Pre-Bureau Affordability	Age	Eligibility
	HIGH	false	25	*INELIGIBLE*
1	*DECLINE*	-	-	*INELIGIBLE*
2	-	false	-	*INELIGIBLE*
3	-	-	< 18	*INELIGIBLE*
4	-	-	-	*ELIGIBLE*

Les règles de correspondance sont mises en évidence, mais la sortie de la règle 2 est choisie car INELIGIBLE a une priorité plus élevée que ELIGIBLE.

Exemple de politique Hit des sommes

Pour un Tableau de Décision avec la politique Hit « Collect-Sum » (C+), le résultat du Tableau de Décision est la somme de toutes les sorties distinctes.

Application risk score model | Input Parameter Values for Simulation

(Age = 40, Marital Status = "M", Employment Status = "EMPLOYED")

C+	Age	Marital Status	Employment Status	Partial score
	40	"M"	"EMPLOYED"	133
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	"S"	-	25
7	-	"M"	-	45
8	-	-	"UNEMPLOYED"	15
9	-	-	"STUDENT"	18
10	-	-	"EMPLOYED"	45
11	-	-	"SELF-EMPLOYED"	36

Dans cet exemple, le score partiel de sortie est calculé comme suit : 43 + 45 + 45 = 133

Validation Tableau de Décision

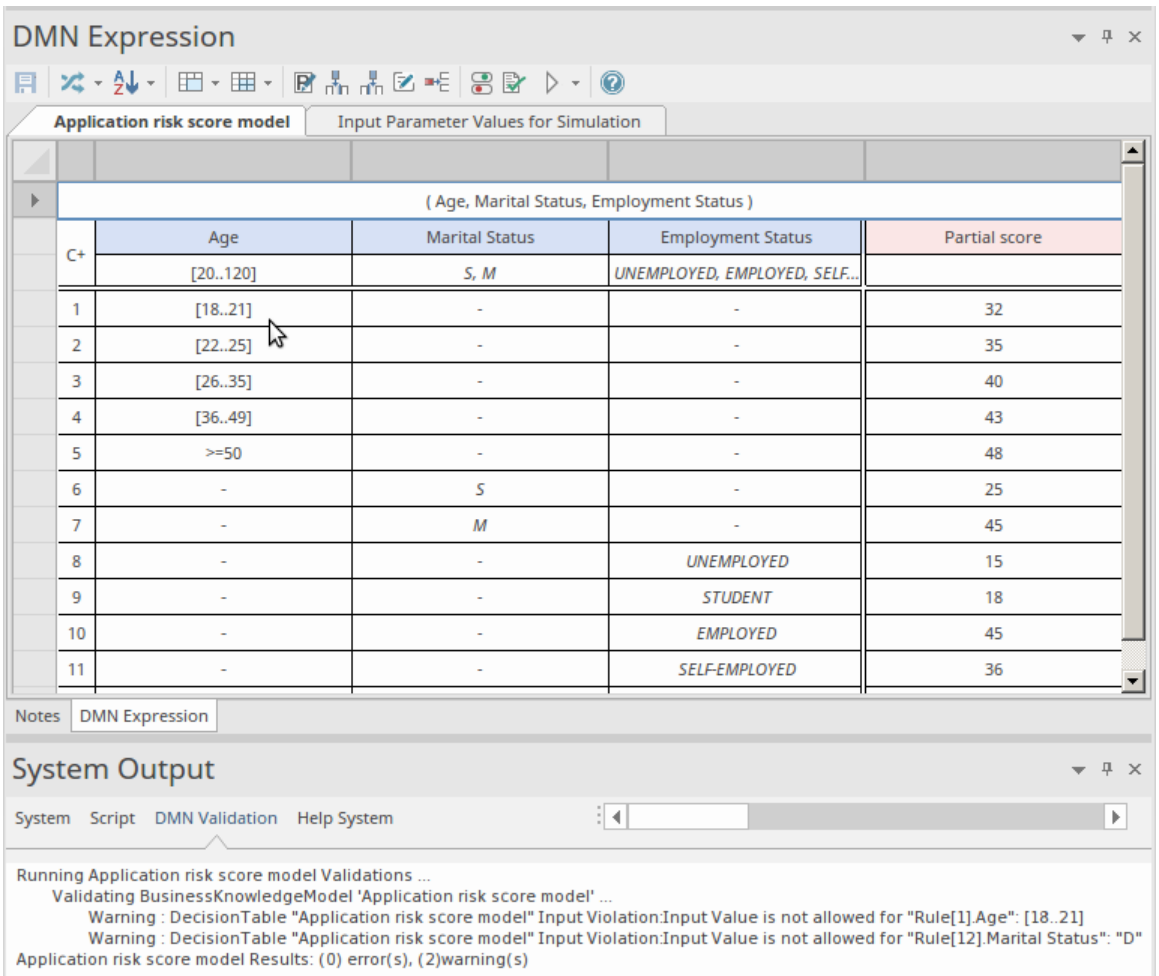
Un Tableau de Décision est l'une des expressions DMN les plus courantes et les plus utiles utilisées pour exprimer la logique de décision. Cependant, modélisation d'un Tableau de Décision peut également être compliquée, en particulier si plusieurs clauses d'entrée sont utilisées en combinaison pour de nombreuses règles Tableau de Décision . Enterprise Architect fournit la facilité de validation Tableaux Décision , comme expliqué dans cette rubrique.

Accéder

Fenêtre d'expression DMN	Simuler > Analyse Décision > DMN > Expression DMN : bouton Valider
Fenêtre Simulation DMN	Simuler > Décision Analysis > DMN > Ouvrir Simulation DMN > Configurer : bouton Valider

Détection des entrées hors de portée

Il est recommandé de définir des « valeurs autorisées » pour les clauses d'entrée et les clauses de sortie d'un Tableau de Décision . La liste des « valeurs autorisées » est utilisée pour effectuer une vérification de plage des valeurs d'entrée et de sortie pour les règles tableau .



Dans cet exemple :

- La clause d'entrée « Âge » définit une plage de [20..120] ; cependant, l'entrée d'entrée pour la règle 1 spécifie une plage de [18..21] ; cela se situe en dehors de la plage de valeurs autorisées, donc la règle 1 est signalée comme non valide
- La clause « État matrimonial » définit ses valeurs autorisées comme une énumération de « S, M » ; la règle 12 spécifie une valeur de « D », cette règle est donc également signalée comme non valide

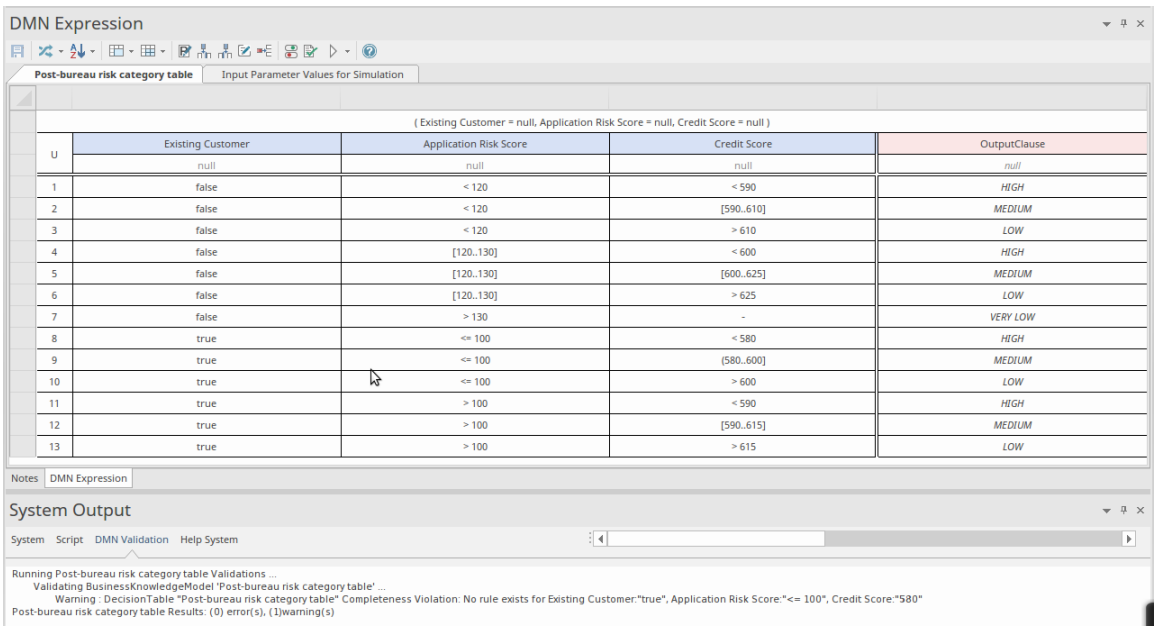
Ces problèmes peuvent être corrigés, soit en mettant à jour les « valeurs autorisées », soit en modifiant les entrées d'entrée pour les règles non valides, en fonction des règles métier réelles.

Détection de l'exhaustivité - signaler les lacunes dans les règles

Les lacunes dans les règles d'un Tableau de Décision signifient que, compte tenu d'une combinaison de valeurs d'entrée, aucune règle ne correspond. Cela indique qu'une logique ou une règle peut être manquante (à moins qu'une sortie par défaut ne soit définie).

Lorsque le Tableau de Décision contient de nombreuses règles spécifiant des plages de nombres, il devient difficile de détecter visuellement les lacunes et il devient assez long de composer et exécuter des cas de test exhaustifs.

Par exemple:



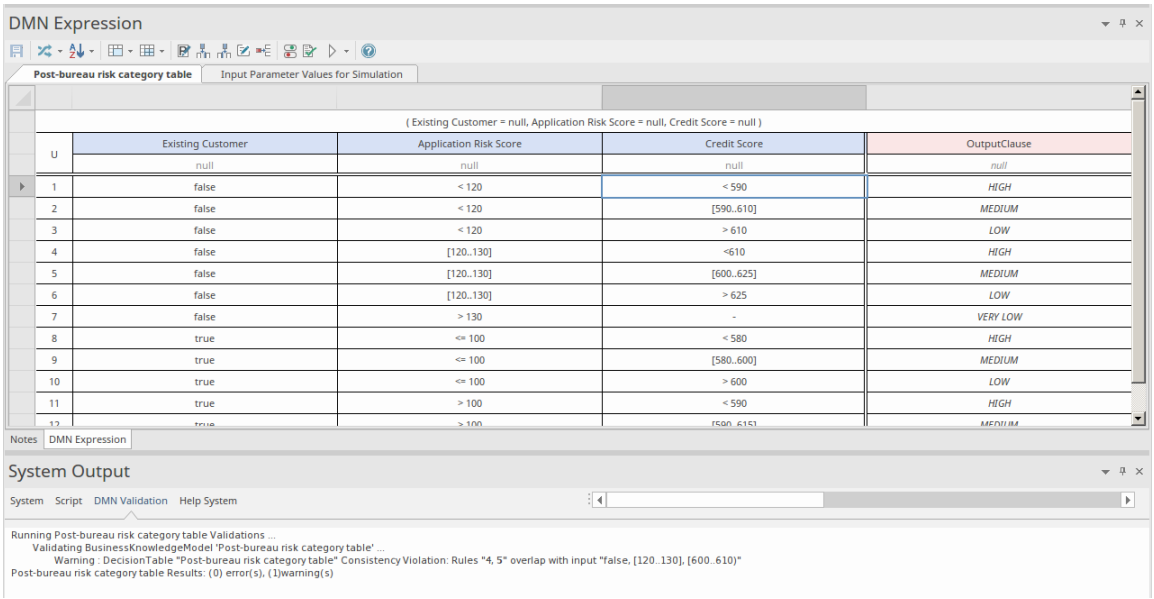
La validation signale une lacune dans les règles. Une inspection plus approfondie révèle une erreur dans la règle 9. L'entrée d'entrée **(** 580..600], devrait être **[** 580..600].

Détection de chevauchement de règles pour la politique Hit unique

Lorsque les règles se chevauchent, pour une combinaison donnée de valeurs d'entrée, plusieurs règles sont mises en correspondance. Il s'agit d'une violation si Tableau de Décision spécifie sa politique Hit comme « Unique ».

Lorsque le Tableau de Décision contient de nombreuses règles spécifiant des plages de nombres, il devient difficile de détecter visuellement les lacunes et il devient assez long de composer et exécuter des cas de test exhaustifs.

Par exemple:



La validation signale un chevauchement dans les règles, impliquant les règles 4 et 5. Une inspection plus approfondie révèle que le chevauchement existe dans la troisième entrée « Score de crédit », où « <610 » chevauche « [600..625] ». Vous pouvez corriger ce problème soit en remplaçant la règle 4 par « <600 » ou en remplaçant la règle 5 par « [610..625] », pour refléter les règles commerciales réelles.

Expression littérale

Une expression littérale est la forme la plus simple d'expression DMN ; elle est généralement définie comme une instruction d'une seule ligne ou un bloc conditionnel if-else. L'expression littérale est un type d'expression valeur utilisé à la fois dans les éléments Décision et dans les éléments Métier Knowledge Modèle (BKM). Au fur et à mesure que l'expression devient plus complexe, vous pouvez préférer un contexte encadré ou, afin d'améliorer la lisibilité, vous pouvez encapsuler une partie de la logique sous forme de fonction dans la Bibliothèque DMN.

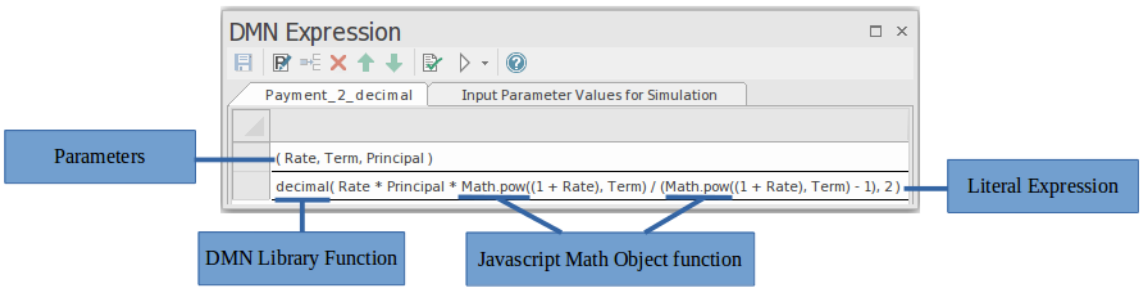
L'icône =X dans le coin supérieur droit de l'élément Décision ou BKM indique qu'il est implémenté en tant *qu'expression littérale* .

Accéder

Diagramme	Sur un diagramme , double-cliquez sur un élément Décision ou sur un élément BusinessKnowledgeModel. La fenêtre de l'éditeur d'expression DMN s'affiche et affiche les détails de l'élément sélectionné.

Aperçu

Cette image montre la fenêtre de l'éditeur d'expression DMN, telle qu'elle apparaît pour une expression littérale.



L'expression littérale est une représentation textuelle de la logique de décision. Elle décrit comment une valeur de sortie est dérivée de ses valeurs d'entrée, à l'aide d'opérations mathématiques et logiques.

La fenêtre de l'éditeur d'expression présente l'expression littérale sous forme de tableau , avec deux lignes clés :

- Paramètres : définit les paramètres d'entrée utilisés dans l'expression
- Expression littérale : où la formule de l'expression est définie - cela définit la sortie de la Décision

Afin de support la simulation et l'exécution, l'expression littérale peut utiliser des fonctions globales JavaScript ou des fonctions object JavaScript . Les utilisateurs peuvent également créer des fonctions DMN Bibliothèque à utiliser dans les expressions.

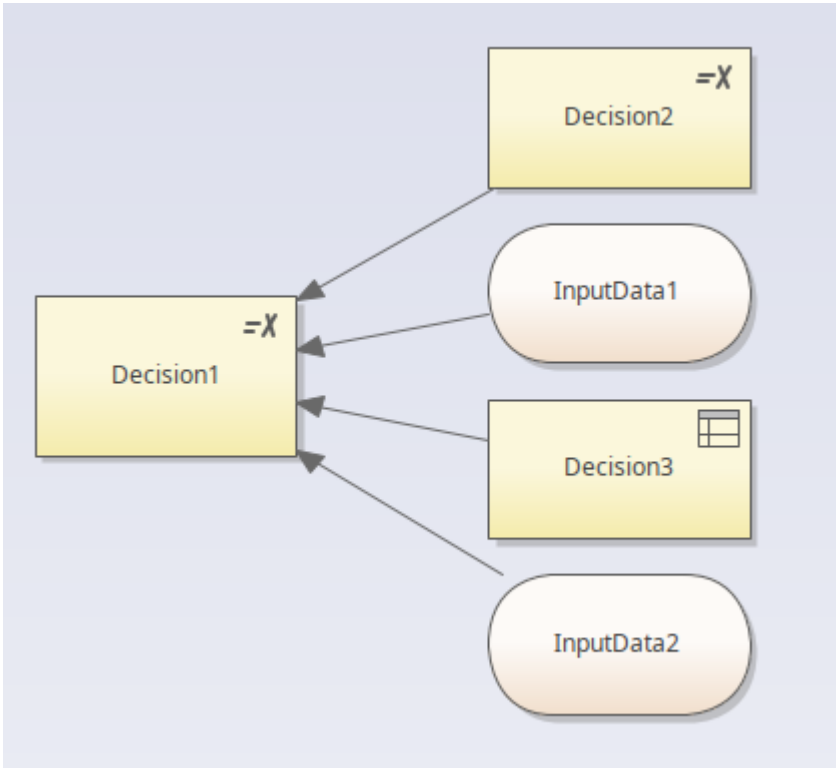
Barre d'outils pour l'éditeur d'expressions littérales

Lorsqu'une expression littérale est sélectionnée, la disposition des fonctionnalités accessibles dans la fenêtre Expression DMN est :

Pour plus de détails, reportez-vous à la rubrique d'aide *Barre d'outils pour l'éditeur d'expressions littérales* .

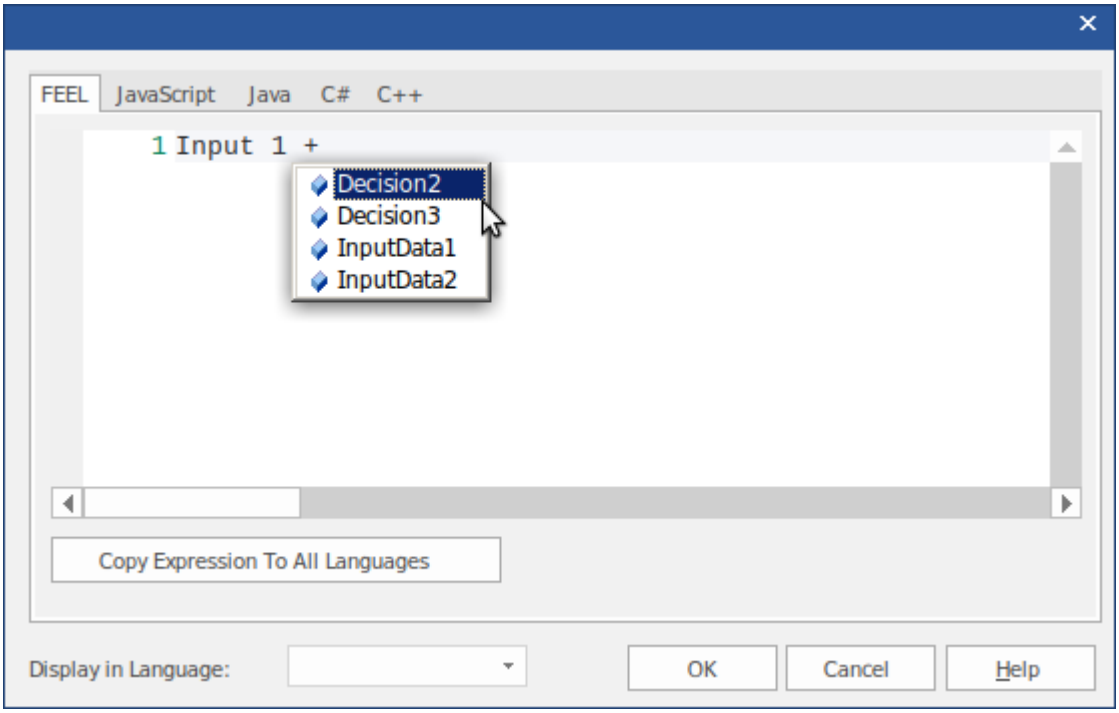
support de l'éditeur d'expressions et d'Intelli-sense

Conformément à la spécification du langage FEEL (Friendly Enough Expression Language), les noms de paramètres peuvent contenir des espaces, ce qui facilite la lecture de l'expression. Enterprise Architect fournit également support Intelli-sense pour l'édition des expressions, ce qui permet de réduire la saisie et les erreurs.



Étant donné une hiérarchie de décision telle que celle illustrée, lors de l'édition de l'expression pour « Décision1 », les entrées de « Décision1 » - à savoir « Décision2 », « Décision3 », « InputData1 » et « InputData2 » - seront disponibles via Intelli-sense dans l'éditeur.

En cliquant avec le bouton droit de la souris sur la ligne « Expression » de la fenêtre Expression DMN, puis en choisissant l'option de menu « Modifier les expressions... », la dialogue de l'éditeur de code d'expression s'affiche. Appuyez sur Ctrl+Espace pour afficher le menu Intelli-sense :



- Pour les éléments « Décision », toutes les entrées de la décision seront affichées
- Pour les éléments Métier Knowledge Modèle (BKM), tous les paramètres d'entrée seront affichés

Le Modèle DMN peut être généré sous forme de code source en JavaScript , Java, C# ou C++. Étant donné que certains langages peuvent avoir une syntaxe différente pour certaines expressions, Enterprise Architect fournit des pages de remplacement de langage pour chaque langage. Si aucun code de remplacement n'est spécifié pour un langage, l'expression définie pour le langage FEEL sera utilisée.

Dans le code généré, l'espace à l'intérieur d'un nom de variable sera remplacé par un trait de soulignement.

Barre d'outils pour l'éditeur d'expressions littérales

Lorsqu'une expression littérale est sélectionnée, la fenêtre Expression DMN affiche une barre d'outils spécifique à ce type d'expression.

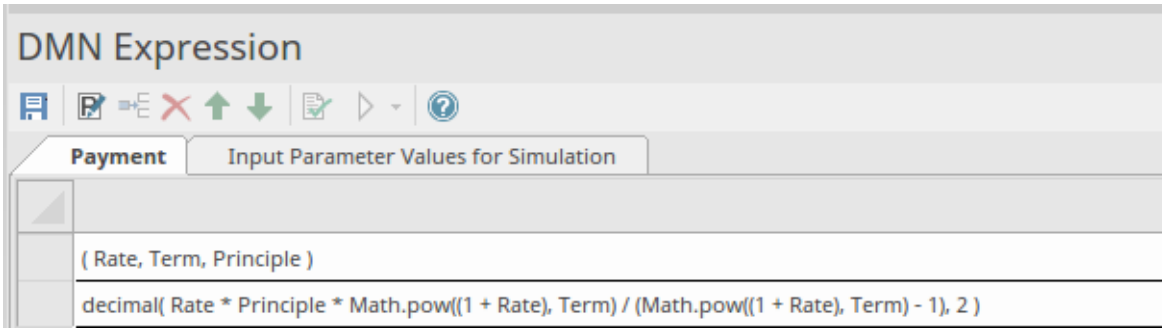
Options de la barre d'outils

Ce tableau fournit des descriptions des fonctionnalités accessibles depuis la barre d'outils de la fenêtre Expression DMN lorsqu'une expression littérale est sélectionnée.

Options	Description
	Cliquez sur ce bouton pour enregistrer la configuration dans le Décision ou le BusinessKnowledgeModel actuel.
	Cliquez sur ce bouton pour modifier les paramètres du Métier Knowledge Modèle .
	Cette option est désactivée pour les expressions littérales.
	Cette option est désactivée pour les expressions littérales.
	Cette option est désactivée pour les expressions littérales.
	Cette option est désactivée pour les expressions littérales.
	Cliquez sur ce bouton pour valider l'expression littérale. Enterprise Architect effectuera une série de validations pour vous aider à localiser les éventuelles erreurs dans l'expression.
	Ce bouton est activé lorsque l'expression littérale est définie pour un élément BusinessKnowledgeModel.

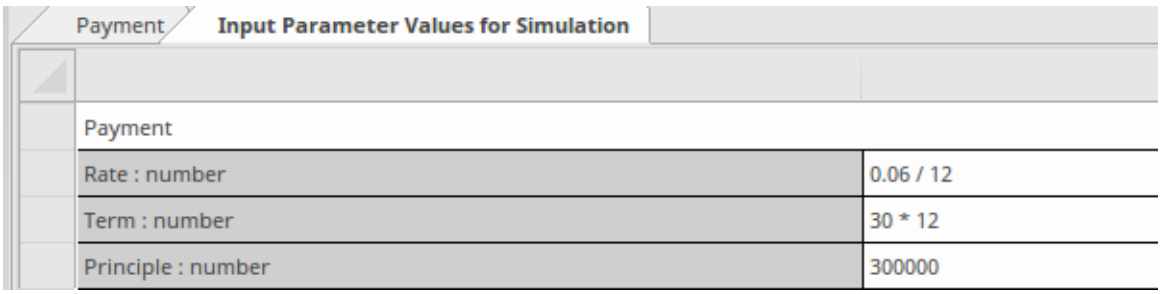
Exemple - Remboursement d'un prêt

Ce *paiement* Métier Knowledge Modèle (BKM) est implémenté sous la forme d'une expression littérale.

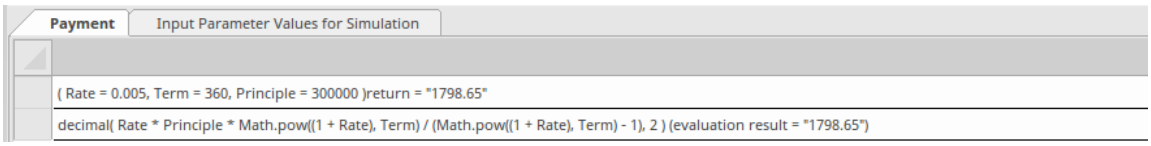


- Le BKM définit trois paramètres : le taux, la durée et le principe

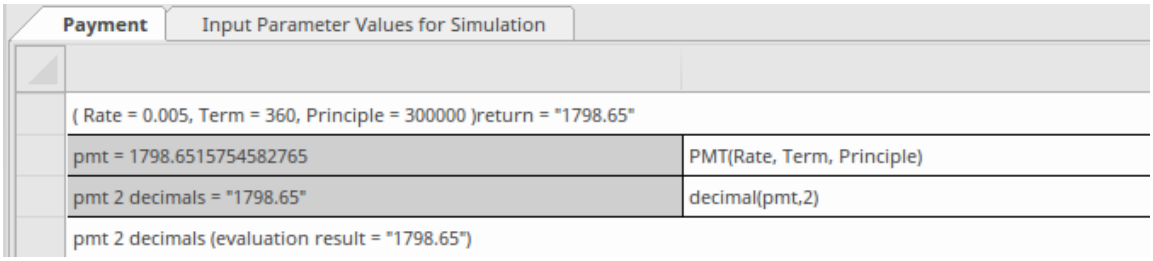
Définissez les valeurs des paramètres d'entrée et évaluez le modèle :



- La valeur du paramètre d'exécution sera affichée ; par exemple, Taux = 00,005
- Le résultat du BKM sera évalué par l'expression littérale et la valeur est affichée sur la ligne de déclaration ; par exemple, return = 1798.65

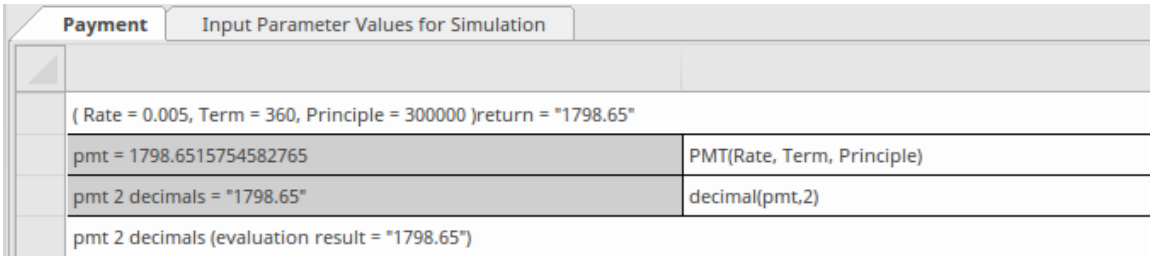


Bien que la formule puisse être écrite en une seule ligne, elle est assez compliquée. Nous pouvons refactoriser ce modèle avec une fonction intégrée et un contexte encadré pour améliorer la lisibilité :



- Le contexte encadré définit deux entrées appariées variable-expression ; ces variables servent de « variables locales », qui peuvent être utilisées dans des expressions ultérieures
- valeur de retour : l'expression peut utiliser la valeur de « variables locales »
- Toutes les expressions dans un contexte encadré peuvent utiliser des fonctions intégrées définies dans la *Bibliothèque* personnalisable Gabarit — DMN ; par exemple, les fonctions PMT(...) et decimal(...) sont utilisées dans cet exemple

Le résultat de la simulation est exactement le même que celui d'une expression littérale :


Payment
Input Parameter Values for Simulation
(Rate = 0.005, Term = 360, Principle = 300000)return = "1798.65"
pmt = 1798.6515754582765
PMT(Rate, Term, Principle)
pmt 2 decimals = "1798.65"
decimal(pmt,2)
pmt 2 decimals (evaluation result = "1798.65")

Contexte encadré

Un contexte encadré est une collection d'entrées de contexte, présentées sous la forme d'un tableau , suivies d'une expression de résultat final.

Ces entrées de contexte se composent d'une variable associée à une expression valeur et peuvent être considérées comme des résultats intermédiaires. Cela permet de décomposer des expressions complexes en une série d'expressions simples, le résultat final étant évalué sous une forme beaucoup plus simple.

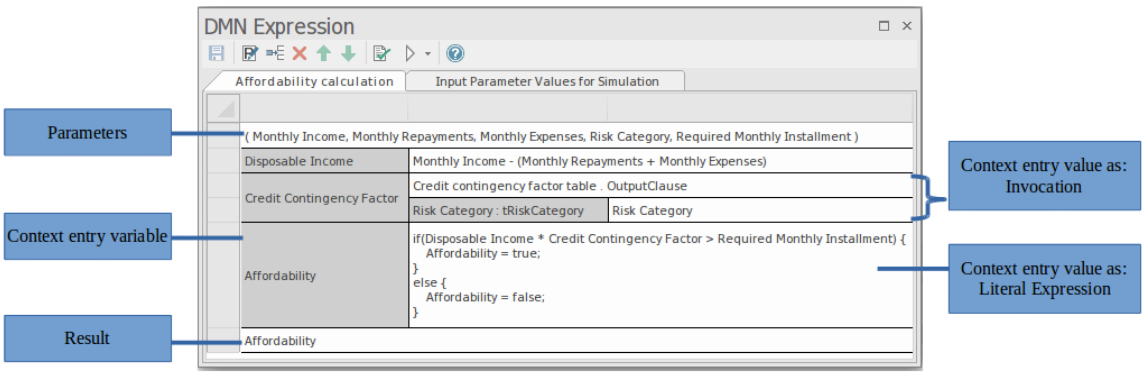
Le type Boxed Context est pris en charge dans les types d'éléments *Décision* et *Métier Knowledge Modèle* . Il est indiqué par l'icône .

Accéder

Diagramme	Sur un diagramme , double-cliquez sur un élément Décision ou un élément BKM. La fenêtre de l'éditeur d'expression DMN s'affiche, affichant les détails de l'élément sélectionné.

Aperçu

Cette image montre la fenêtre de l'éditeur d'expression DMN telle qu'elle apparaît pour une expression de contexte encadré.



Un contexte encadré est une collection d'entrées de contexte, présentées sous la forme d'un tableau , suivies d'une expression de résultat final. Chaque entrée de contexte se compose d'une variable et d'une expression valeur . La variable peut être considérée comme un résultat intermédiaire et peut être utilisée dans l'expression valeur de toute entrée de contexte ultérieure. L'expression valeur d'une entrée de contexte peut être soit une expression littérale, soit une invocation, et peut utiliser toutes les entrées disponibles telles que les paramètres (pour un élément BKM), les données d'entrée ou les résultats de décision, ainsi que toutes les variables de contexte précédemment définies.

Le résultat final d'une expression de contexte encadrée est déterminé en parcourant chaque entrée de contexte à tour de rôle, en évaluant l'expression valeur et en affectant son résultat à la variable, puis en évaluant enfin l'expression de résultat. L'expression de résultat peut également utiliser n'importe quelle variable d'entrée ou locale, mais doit être évaluée pour fournir un résultat.

Barre d'outils pour l'éditeur de contexte en boîte

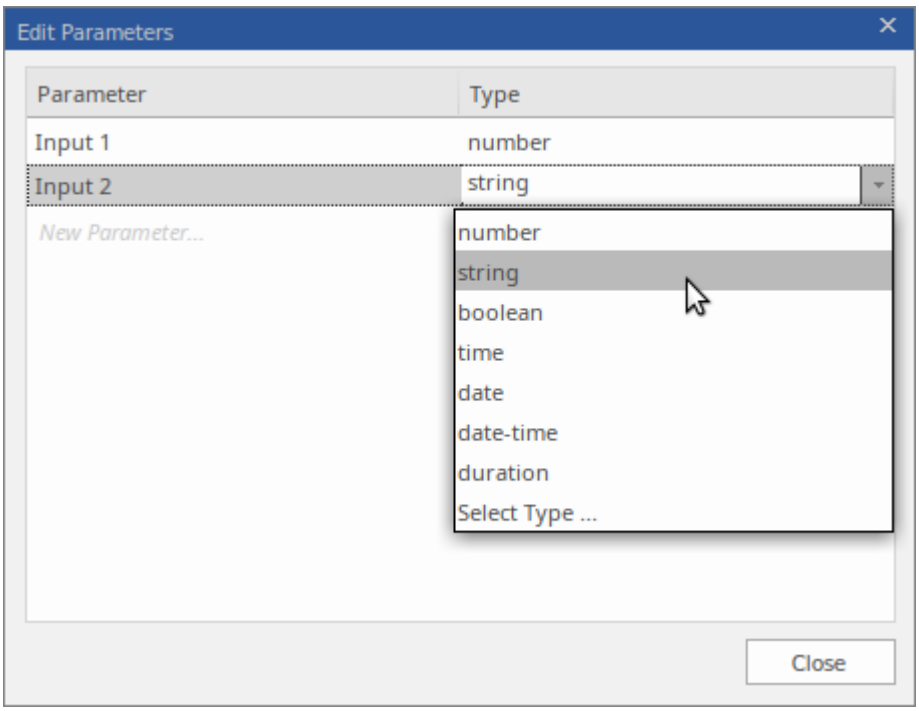
Lorsqu'une expression de contexte encadré est sélectionnée, la disposition des fonctionnalités accessibles dans la fenêtre Expression DMN est :

Pour plus de détails, reportez-vous à la rubrique d'aide « *Barre d'outils pour l'éditeur de contexte encadré* ».

Spécification des paramètres

Dans le cas des éléments BusinessKnowledgeModel, les paramètres sont utilisés pour transmettre les valeurs d'entrée fournies par l'élément appelant. La logique de décision du BKM est évaluée à l'aide des paramètres d'entrée et le résultat est renvoyé à l'élément appelant. Par défaut, un élément BKM est créé avec deux paramètres d'entrée, « Entrée 1 » et « Entrée 2 ».

Cliquez sur l'icône dans la barre d'outils de la fenêtre Expression DMN pour afficher la fenêtre « Modifier les paramètres ».



Ici, vous pouvez modifier les noms des paramètres, définir leurs types de données, créer des paramètres supplémentaires ou supprimer ceux existants.

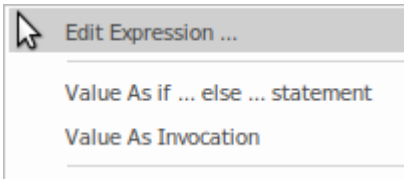
Spécification des entrées de contexte

Chaque entrée de contexte se compose d'une paire variable-expression.

Le nom de la variable peut être n'importe quel texte de votre choix et peut même contenir des espaces. Pour modifier le nom de la variable, cliquez sur la cellule pour la sélectionner, puis cliquez à nouveau ou appuyez sur F2 pour entrer en mode édition. Pour quitter le mode édition, cliquez ailleurs ou appuyez sur la touche Entrée.

En général, il n'est pas nécessaire de spécifier un type de données pour l'expression ou les variables : le type sera déduit de la valeur . Cependant, si vous avez l'intention de générer du code pour des langages compilés tels que Java, C++ ou C# , vous devrez spécifier le type de toutes les variables d'entrée de contexte.

L'expression valeur d'une entrée de contexte peut être une expression littérale ou une invocation et peut utiliser toutes les entrées disponibles, telles que des paramètres (pour un élément Métier Knowledge Modèle), des données d'entrée ou des résultats de décision, ainsi que toutes les variables de contexte précédemment définies. Un clic droit sur la cellule d'expression affiche un menu contextuel qui fournit des options pour afficher un éditeur de code d'expression ou pour définir l'expression valeur comme une instruction If-Else ou une invocation.



Vous pouvez également modifier l’expression valeur en saisissant du texte directement dans la cellule d’expression.

Pour plus d'informations sur la manière de spécifier des expressions littérales ou des invocations, veuillez consulter les rubriques d'aide couvrant ces sujets.

Barre d'outils pour l'éditeur de contexte en boîte

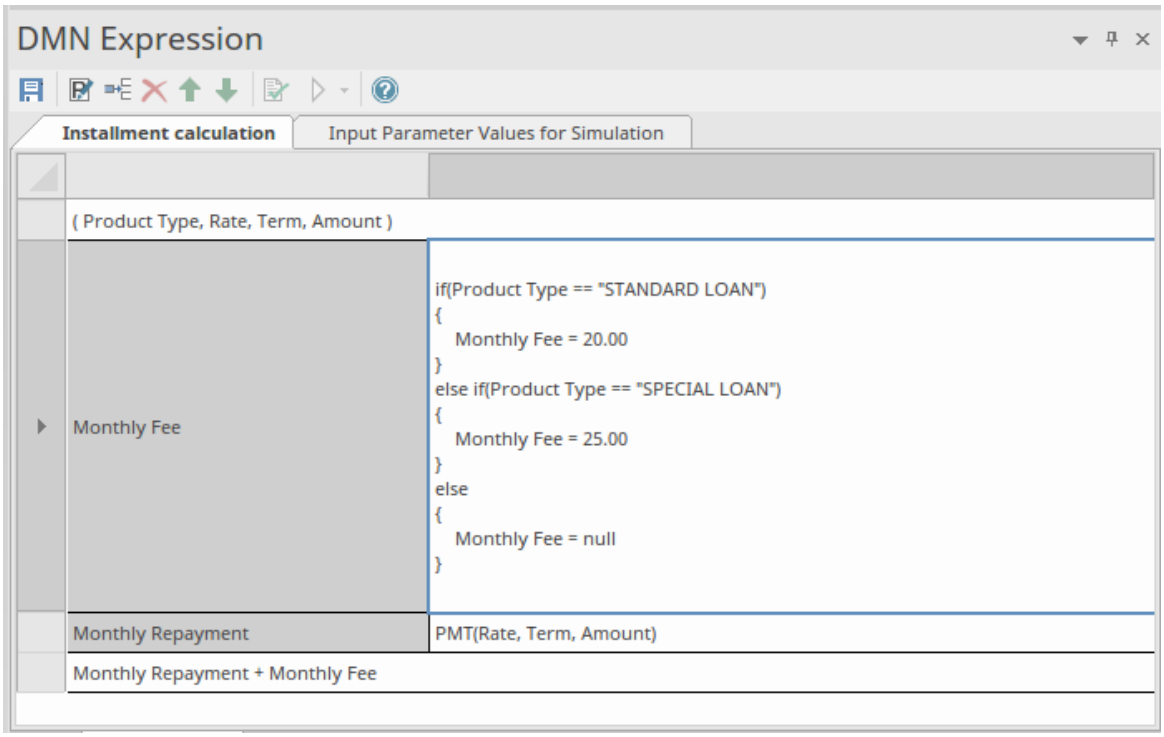
Ce tableau fournit des descriptions des fonctionnalités accessibles dans la fenêtre Expression DMN lorsqu'un contexte encadré est sélectionné.

Options de la barre d'outils

Cette barre d'outils est destinée au contexte encadré.

Options	Description
	Enregistrez les modifications apportées à l'élément Décision ou BusinessKnowledgeModel actuellement sélectionné.
	Affichez la fenêtre « Modifier les paramètres », dans laquelle vous pouvez spécifier le nom et le type de données de chaque paramètre transmis lors de l'appel de la logique de décision d'un élément BusinessKnowledgeModel.
	Créez une nouvelle entrée de contexte et ajoutez-la à la liste des entrées de contexte.
	Supprimer l'entrée de contexte actuellement sélectionnée.
	Déplacer l'entrée de contexte actuellement sélectionnée d'une position vers le haut dans la liste.
	Déplacez l'entrée de contexte actuellement sélectionnée d'une position vers le bas dans la liste.
	Effectuez la validation du BoxedContext. Enterprise Architect effectuera une série de validations pour vous aider à découvrir d'éventuelles erreurs dans la définition du BoxedContext.
	Ce bouton est activé lorsqu'un Tableau de Décision est défini pour un élément Métier Knowledge Modèle . Sélectionnez l'onglet « Valeurs des paramètres d'entrée pour Simulation », complétez les champs, puis cliquez sur ce bouton. Le résultat du test sera présenté sur le Tableau de Décision , avec les valeurs d'exécution des entrées et des sorties affichées et les règles valides mises en évidence. Vous pouvez utiliser cette fonctionnalité pour tester un élément BusinessKnowledgeModel, sans spécifier son contexte. Plusieurs options de menu sont disponibles pour ce bouton de la barre d'outils. Pour plus d'informations, consultez la rubrique d'aide *Simuler Modèle DMN* .

Exemple - Calcul des mensualités d'un prêt



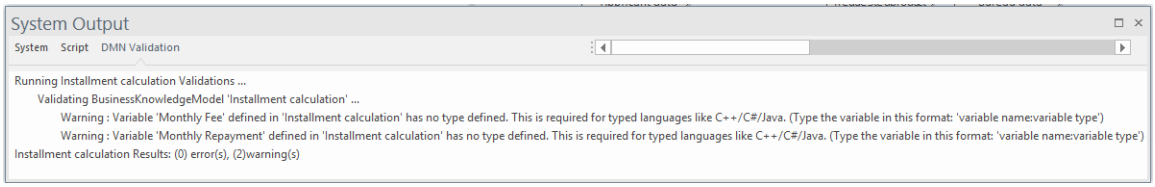
Le *calcul des versements* Métier Knowledge Modèle (BKM) est implémenté sous forme de contexte encadré.

- Le BKM définit quatre paramètres : Type de produit, le taux, la durée et le montant
- Le contexte encadré définit deux entrées de paires variable-expression ; ces variables servent de « variables locales » qui peuvent être utilisées dans des expressions ultérieures
- valeur de retour : L'expression peut utiliser la valeur des « variables locales »
- Toutes les expressions dans un contexte encadré peuvent utiliser des fonctions intégrées, qui sont définies dans la bibliothèque personnalisable Gabarit — *Bibliothèque* ; les fonctions PMT(...) et decimal(...) sont utilisées dans cet exemple

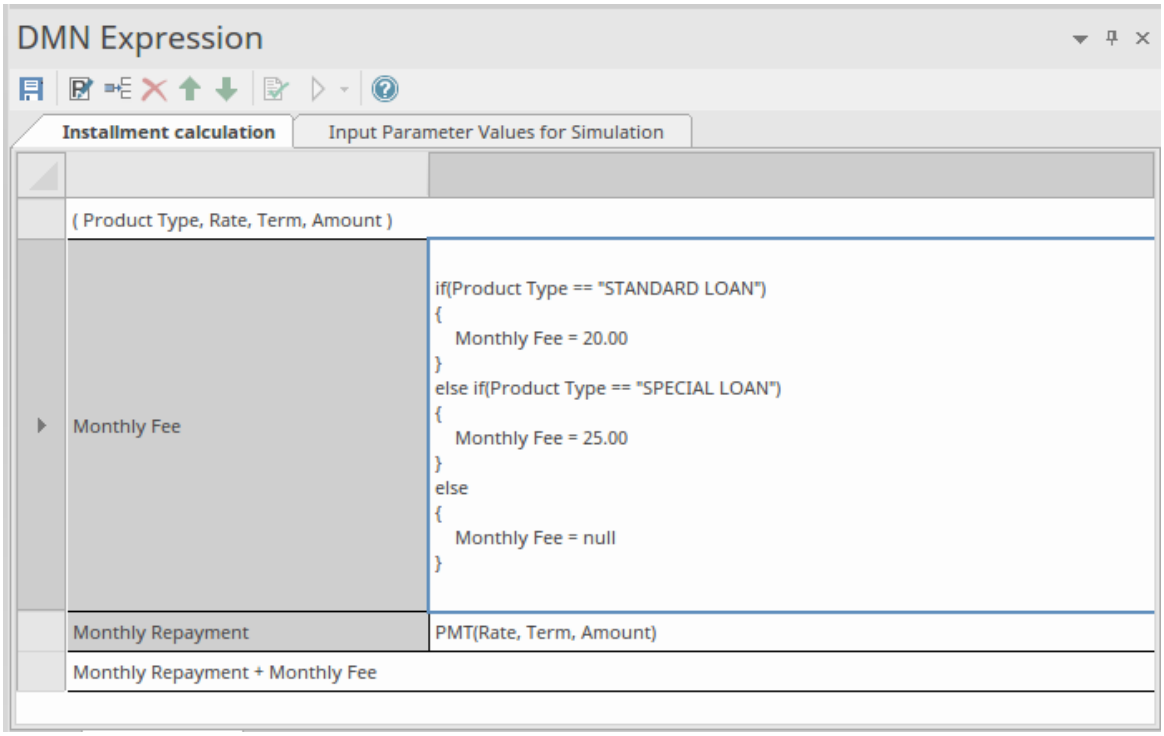
Spécifier Type de variable d'entrée de contexte

En général, l'expression et les variables n'ont pas besoin de spécifier un type, qui est déduit de la valeur fournie. Cette fonctionnalité est prise en charge de manière générique par JavaScript , qui est utilisé pour Simulation DMN d' Enterprise Architect .

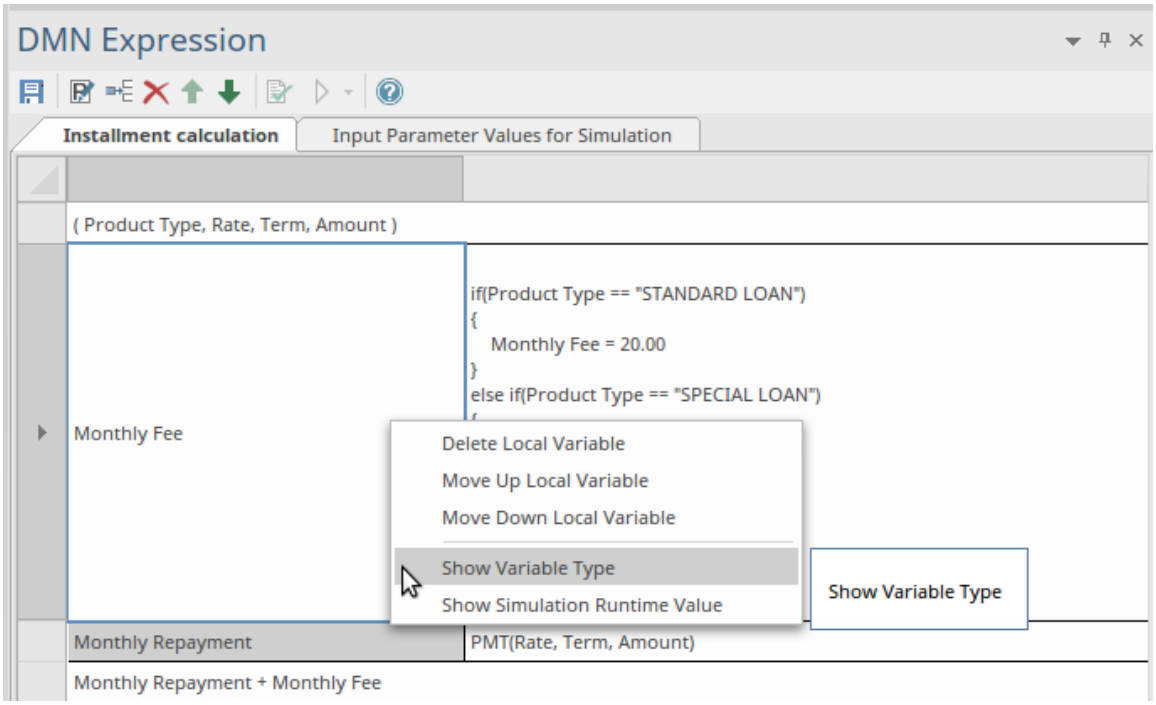
Cependant, si vous souhaitez générer du code à partir d'un modèle DMN vers des langages compilés tels que Java, C++ ou C# , vous devrez spécifier le type de chaque variable d'entrée de contexte. Sinon, si vous validez le modèle, vous verrez des avertissements tels que :



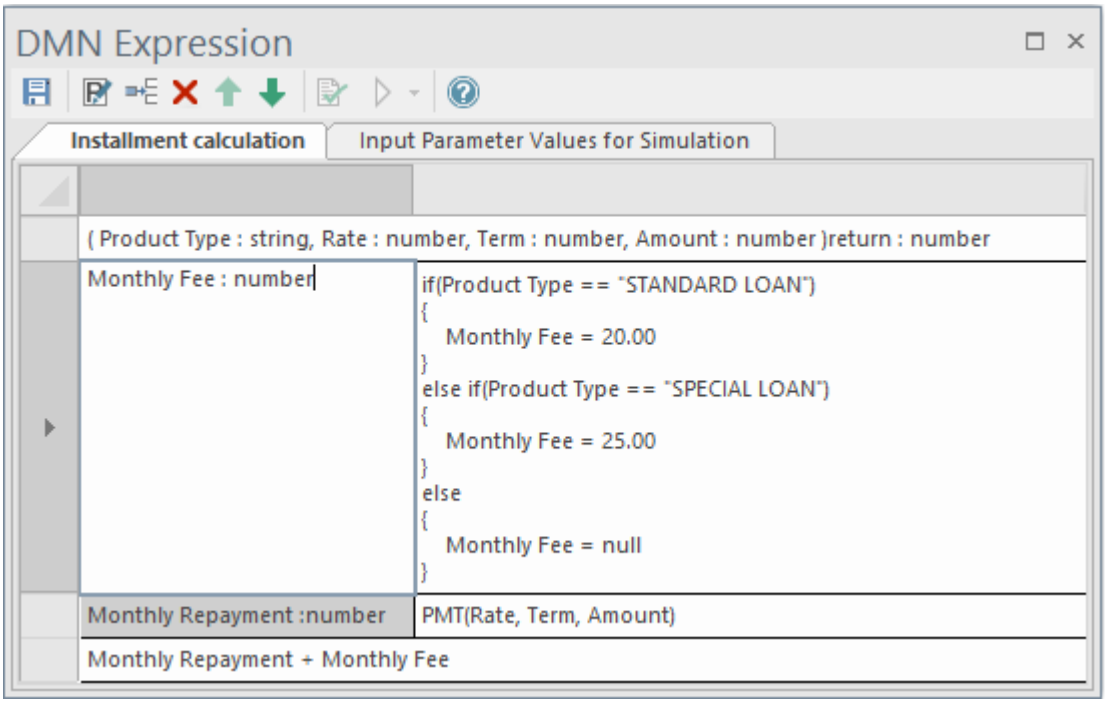
Cliquez-droit sur la variable d'entrée de contexte (frais mensuels, remboursement mensuel) dans ce modèle.



Sélectionnez l'option « Afficher Type variable ».



Tapez maintenant le type de variable, en l'ajoutant au nom de la variable et en le séparant par deux points, comme indiqué ici.

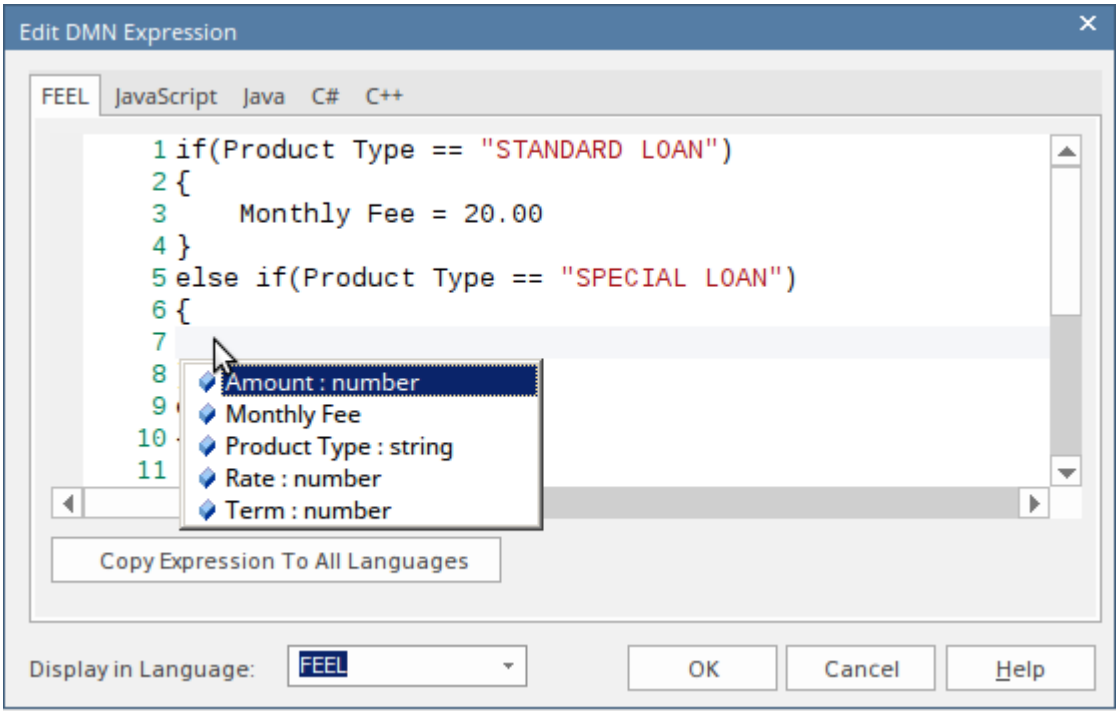


Cliquez ensuite sur le bouton Enregistrer de la barre d'outils pour enregistrer l'expression, puis cliquez sur le bouton pour valider à nouveau le modèle.

Éditeur d'expressions et Support d'Intelli-sense

Le paramètre et le nom de la variable d'entrée de contexte peuvent contenir des espaces, conformément à la spécification du langage FEEL. Cette fonctionnalité facilite la lecture de l'expression. Afin de vous aider à modifier les expressions avec moins de saisie et en faisant moins d'erreurs, Enterprise Architect fournit support Intelli-sense pour la modification des expressions :

Pour éditer une expression, cliquez-droit sur l'expression (dans le champ de droite) et sélectionnez l'option de menu « Modifier les expressions ». La dialogue « Expression » s'affiche. Cliquez sur la ligne requise et appuyez sur Ctrl+Espace pour afficher le menu Intelli-sense :



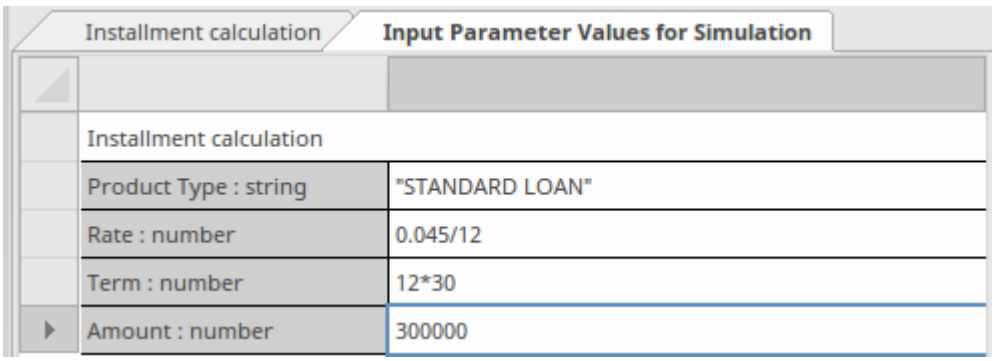
- Toutes les variables d'entrée de contexte antérieures à la variable actuelle seront incluses (les variables d'entrée de contexte postérieures à la variable actuelle sont exclues)
- Pour un Métier Knowledge Modèle (BKM), tous les paramètres seront inclus
- Pour une Décision , toutes les décisions requises seront incluses

Le modèle DMN peut être généré sous forme de code source pour JavaScript , Java, C# et C++. Étant donné que certains langages peuvent avoir une syntaxe différente pour certaines expressions, Enterprise Architect fournit des pages de remplacement de langage pour chaque langage. Si aucun code de remplacement n'est spécifié pour un langage, l'expression définie pour le langage FEEL sera utilisée.

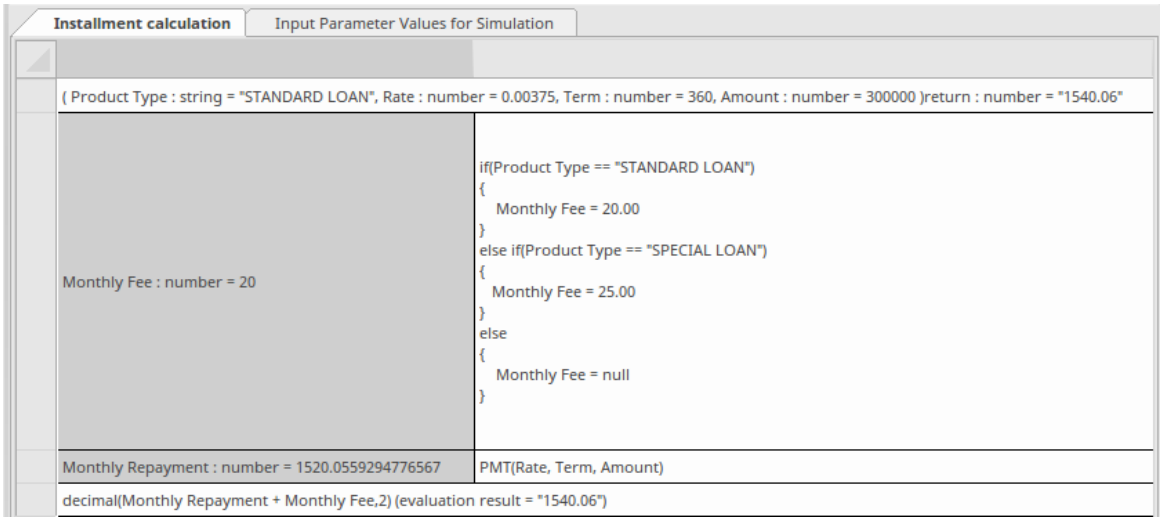
Dans le code généré, l'espace à l'intérieur d'un nom de variable sera remplacé par un trait de soulignement.

Simulation du Modèle de Connaissance Métier

Sélectionnez l'onglet « Valeurs des paramètres d'entrée pour Simulation » et remplissez chaque champ.



Cliquez sur le bouton Enregistrer puis sur le bouton Simulation dans la barre d'outils ; le résultat du test sera présenté dans l'expression de contexte encadrée.



- La valeur du paramètre d'exécution sera affichée ; par exemple, « Taux = 0,00375 »
- La valeur d'exécution de la variable « Entrée de contexte » sera affichée ; par exemple, « Remboursement mensuel = 1 520,06 »
- Le résultat du Métier Knowledge Modèle (BKM) sera évalué par la dernière entrée et les valeurs affichées sur la ligne de déclaration ; par exemple, « retour = 1540,06 »

Vous pouvez utiliser cette fonctionnalité pour tester unitairement un BKM sans connaître le contexte afin qu'il puisse être invoqué ultérieurement par un Décision ou un autre BKM.

Liste encadrée

Une liste encadrée DMN est un élément Décision qui contient une liste d'expressions encadrées. Ces éléments sont disposés sous forme de liste verticale dans la fenêtre Expression DMN.

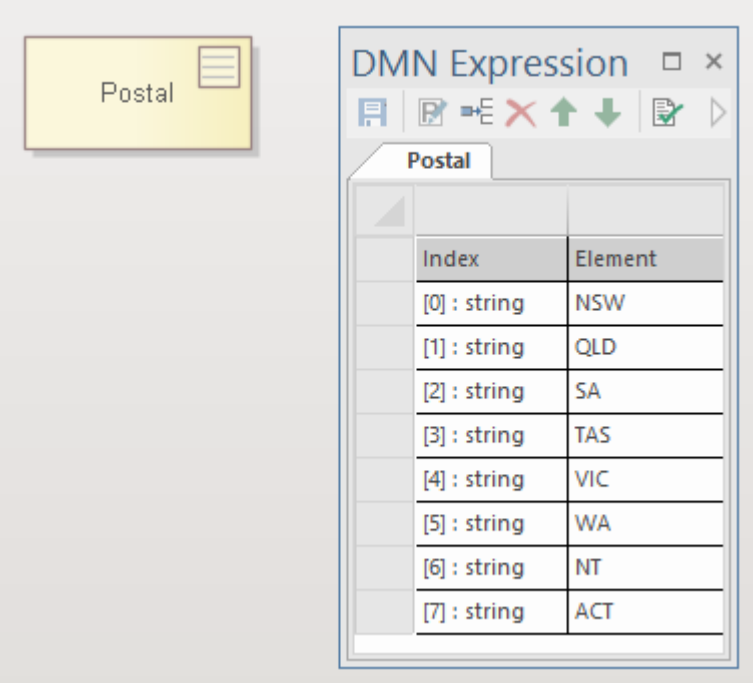
Une liste encadrée est souvent utilisée en conjonction avec une expression *de boucle for* contenue dans un élément Décision associé. L'expression *de boucle for* est utilisée pour parcourir chaque ligne de la liste encadrée, en liant le champ Élément de la liste à la variable correspondante et en évaluant l'expression dans la portée. La sortie de la *boucle for* est une liste contenant l'évaluation de l'expression pour chaque itération individuelle.

Accéder

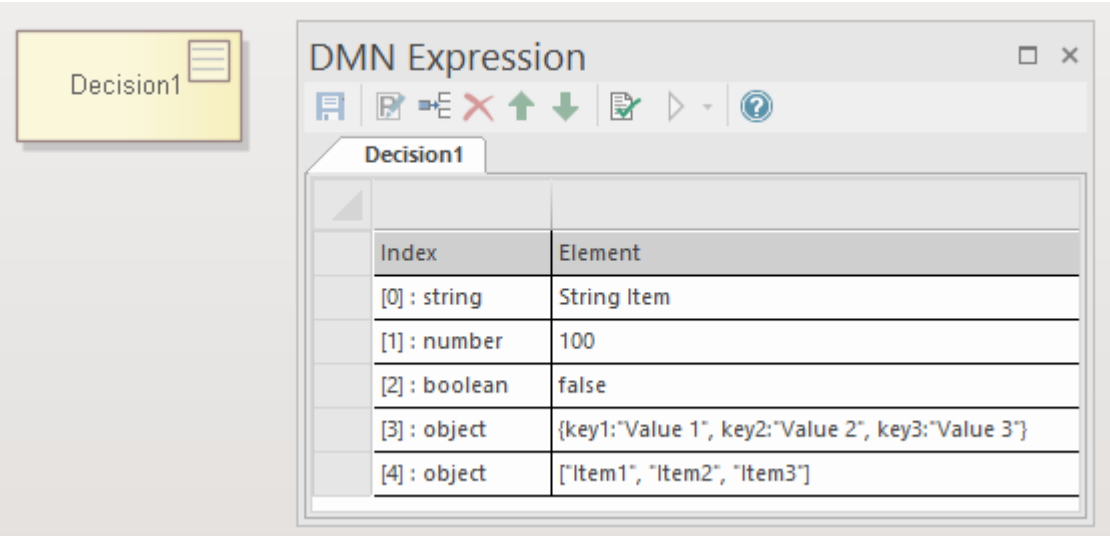
Boîte à outils Diagramme	Faites glisser un élément Décision ou un élément BKM de la boîte à outils sur un diagramme DMN et sélectionnez « Liste » dans le menu d'expression contextuel. Double-cliquez sur l'élément DMN ; la fenêtre Expression DMN s'affiche, montrant les détails de l'élément sélectionné.
Propriétés	Cliquez-droit sur un élément DMN Décision ou BKM du diagramme , et sélectionnez l'option ' Propriétés | Option de menu Propriétés . Sur la page Général, sélectionnez l'onglet 'Tags' , puis dans le champ valeur « expressionType », cliquez sur la flèche déroulante et sélectionnez « Liste ». Cliquez sur le bouton OK . Double-cliquez sur l'élément DMN ; la fenêtre Expression DMN s'affiche, affichant les détails de l'élément sélectionné.

Aperçu

Il est courant que les listes encadrées soient utilisées comme énumérations, où tous les éléments de la liste sont du même type.



Il est également courant que les listes encadrées soient utilisées comme une collection de données, où chaque élément peut avoir un type différent.



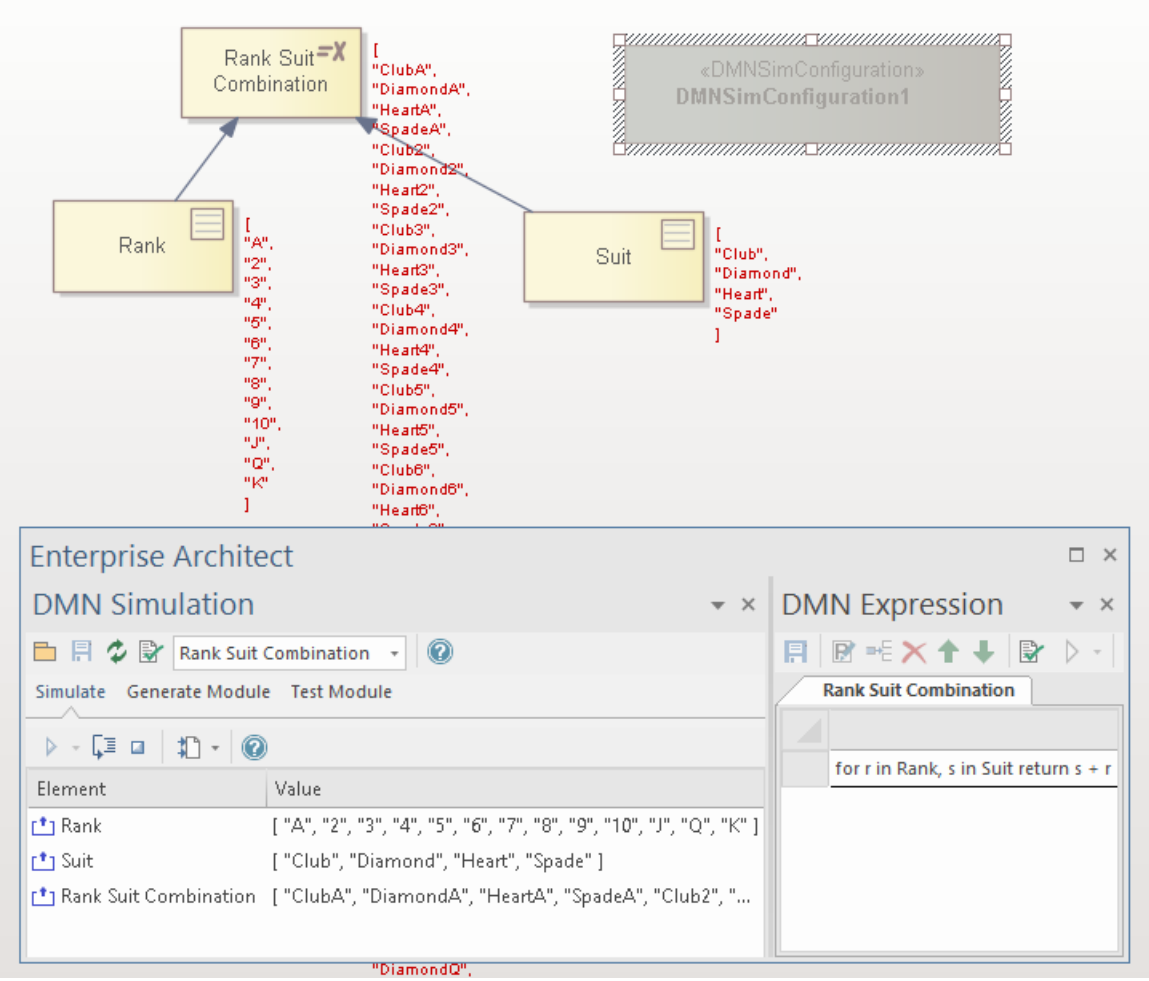
Édition de listes encadrées

La fenêtre Expression DMN dispose d'une barre d'outils proposant les options « Ajouter un nouvel Item de liste », « Supprimer Item de liste existant » et « Déplacer Item vers le haut ou vers le bas ».

Cliquez-droit sur un élément de la liste pour afficher les options du menu contextuel permettant de définir le type de l'élément de la liste : string , nombre, booléen ou object .

Exemple - Classement et couleur au poker

Dans cet exemple, nous avons trois décisions : Rang, Couleur et Combinaison de Couleurs



- Le rang Décision est représenté par une liste encadrée avec 13 Items de « A » à « K »
- Décision Suit est représenté par une liste encadrée avec 4 Items : « Club », « Diamond », « Heart » et « Spade »
- La combinaison Décision Rank Suit est représentée par une expression littérale avec une boucle *for* : *pour r dans Rank, s dans Suit return s + r*

Lorsque plusieurs contextes d'itération sont définis dans la même expression *de* boucle for, l'itération résultante est un produit croisé des éléments des contextes d'itération. L'ordre d'itération va du contexte d'itération interne au contexte d'itération externe.

Dans cet exemple, le produit vectoriel de Rank (13 éléments) et Suit (4 éléments) est une liste de 13 * 4 = 52 éléments.

Relation

Un élément de relation Décision DMN fournit une méthode abrégée pratique pour définir une liste de valeurs liées dans un diagramme DMN. Une relation Décision est comme un tableau de relations avec des colonnes et des lignes. L'en-tête de la grille affiche le nom de chaque colonne. Chaque ligne affiche l'ensemble des valeurs des colonnes correspondantes.

Accéder

Boîte à outils	Pour créer une relation Décision : • Assurez-vous que la Perspective est définie sur : Exigences > Décision Modélisation • Depuis la page Composants DMN de la boîte à outils Diagramme , faites glisser un élément Décision ou un élément BKM sur un diagramme DMN • Sélectionnez « Relation » dans le menu contextuel des expressions • Double-cliquez sur l'élément DMN pour afficher la fenêtre Expression DMN.

Vous pouvez également modifier un élément DMN Décision ou BKM existant en un type Décision-Relation. Pour cela :

- Cliquez-droit sur un élément DMN Décision ou BKM du diagramme , et sélectionnez l'option ' Propriétés | Option de menu Propriétés
- Sur la page Général, sélectionnez l'onglet 'Tags' , puis dans le champ valeur « expressionType », cliquez sur la flèche déroulante et sélectionnez « Relation » ; cliquez sur le bouton OK

Aperçu

Le type de décision *de relation* DMN est une liste verticale contenant des lignes de valeurs. L'une des clés de l'utilisation de la relation Décision est le moyen d'itérer sur les lignes de valeurs à l'aide d'une *boucle For* . La boucle For peut être définie, par exemple, dans une expression littérale Décision associée comme une formule pour traiter les lignes de l'élément de relation de décision.

Modification des relations DMN

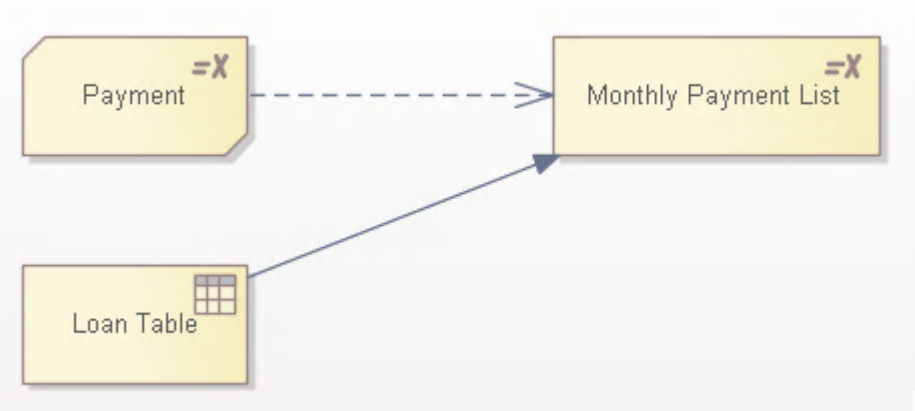
La fenêtre Expression DMN dispose d'une barre d'outils fournissant des options permettant d'ajouter une nouvelle ligne, de supprimer une ligne existante et de déplacer la ligne sélectionnée vers le haut ou vers le bas.

Vous pouvez également :

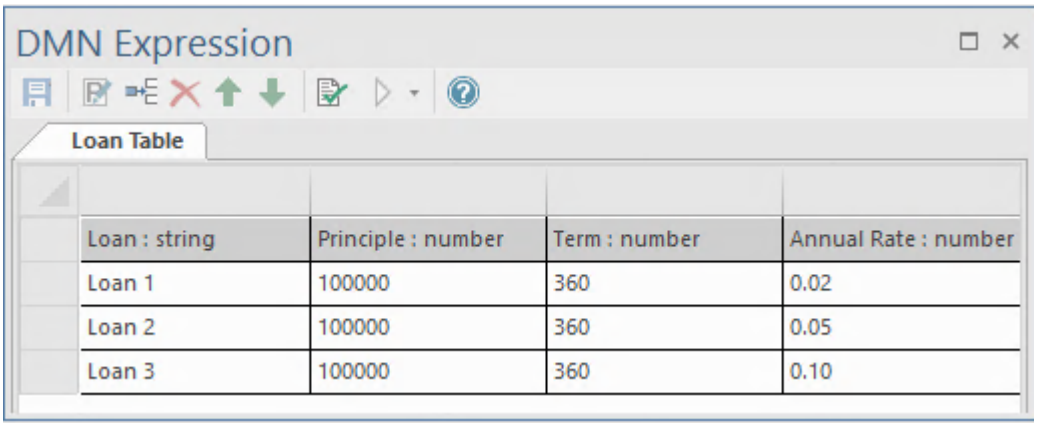
- Faites glisser l'en-tête de la grille pour repositionner les colonnes.
- Cliquez-droit sur les cellules d'en-tête pour afficher les options du menu contextuel permettant de définir le Type de la colonne sur : « string », « nombre », « booléen » ou « object ».

Exemple - Tableau de prêt

Dans cet exemple, nous avons deux décisions - « Tableau de prêt » et « Liste des paiements mensuels » - et un Modèle de connaissances Métier - « Paiement ».

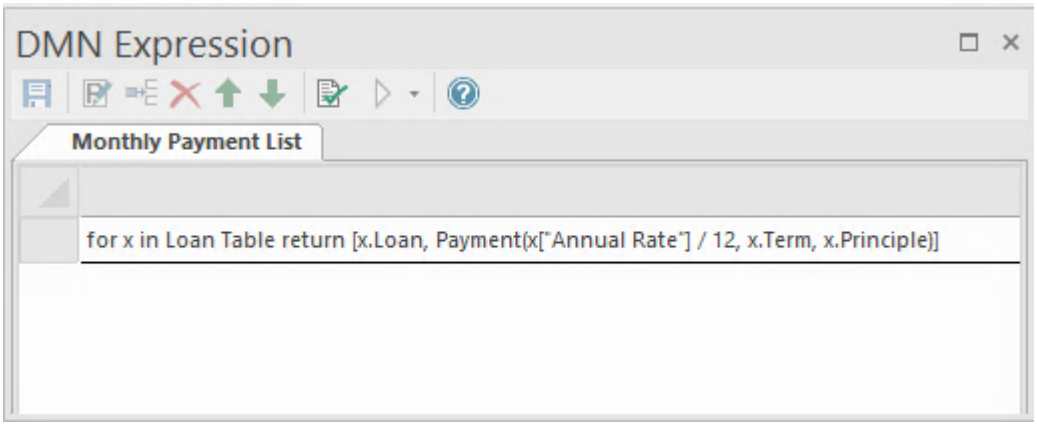


« Tableau de prêt » est une Décision implémentée sous forme de relation avec quatre colonnes : « Prêt », « Principe », « Terme » et « Taux annuel ».



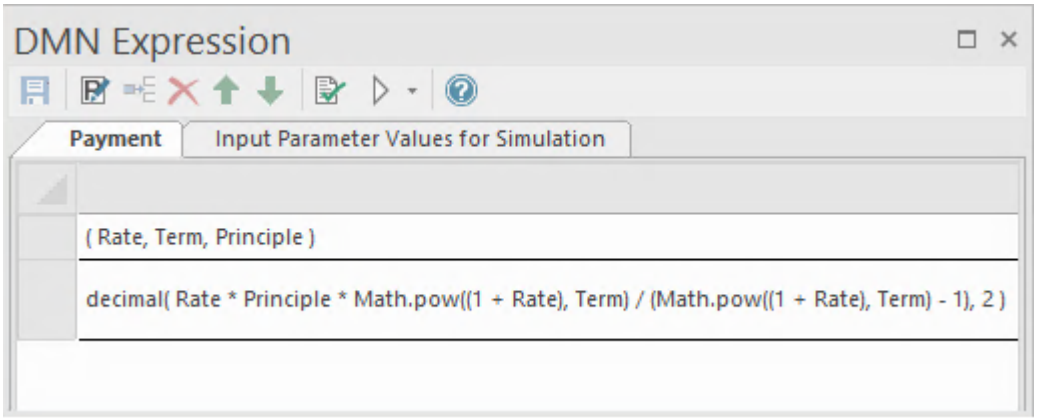
Loan : string	Principle : number	Term : number	Annual Rate : number
Loan 1	100000	360	0.02
Loan 2	100000	360	0.05
Loan 3	100000	360	0.10

« Liste de paiements mensuels » est une Décision implémentée sous forme d'expression littérale avec une boucle *for* :

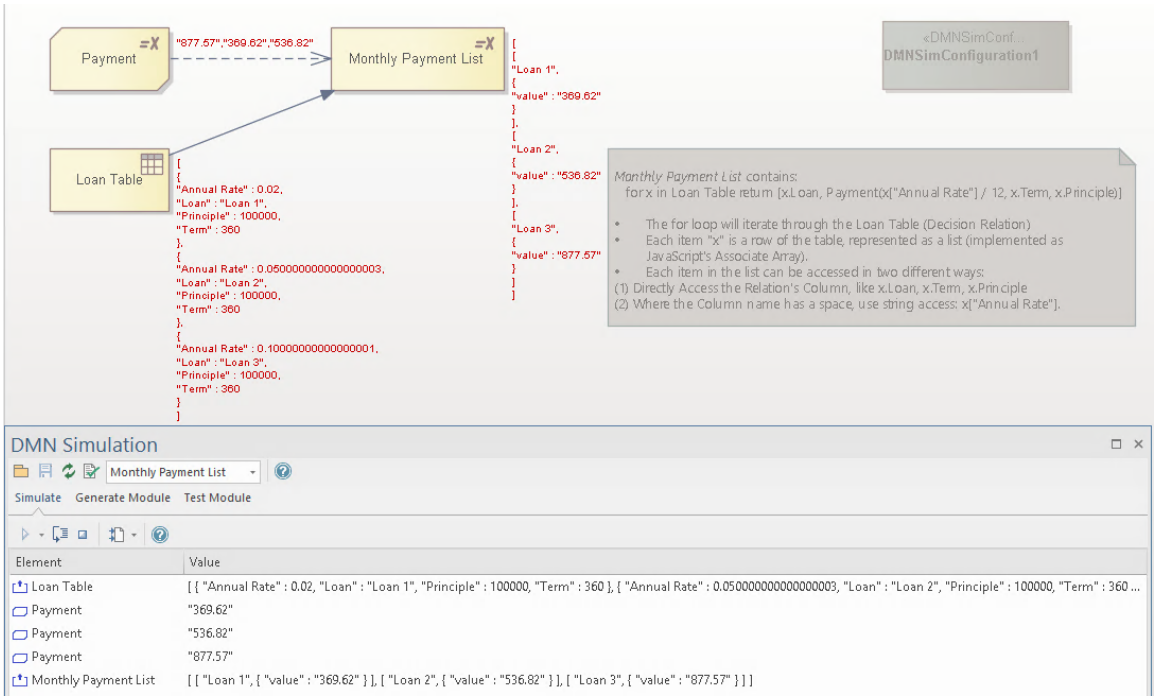


La boucle *for* parcourra « Tableau des prêts » :

- Chaque élément « x » est une ligne du tableau , représentée sous forme de liste
- Avec chaque élément de ligne « x », invoquez Métier Knowledge Modèle « Paiement » avec les éléments de la liste de lignes
- Chaque élément de la liste est accessible de deux manières différentes :
  (1) Accédez directement à la colonne Relation, comme x.Prêt, x.Terme, x.Principe
  (2) Lorsque le nom de la colonne comporte un espace, utilisez l'accès string : x["Taux annuel"]



Lors de la simulation, les valeurs d'exécution s'affichent dans la fenêtre Simulation et sur le diagramme à côté de l'élément ; vous pouvez cliquer sur une étape pour voir le processus de simulation.



Invocation

Une invocation est un conteneur pour les liaisons de paramètres qui fournissent le contexte pour l'évaluation du corps d'un Métier Knowledge Modèle . Il existe deux cas d'utilisation courants pour une invocation :

- Lier une donnée d'entrée au Métier Knowledge Modèle
- Lier des paramètres ou des variables d'entrée de contexte au Métier Knowledge Modèle

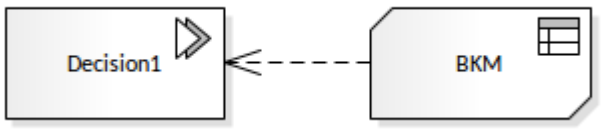
Un exemple de chacun est fourni dans les sous-rubriques de cette rubrique d'aide.

Accéder

Diagramme	Double-cliquez sur l'élément Décision ou l'élément BKM approprié. La fenêtre Expression DMN s'affiche, affichant les détails de l'élément sélectionné.

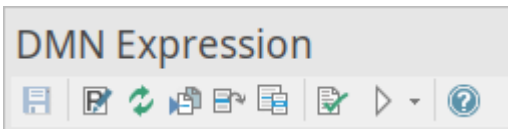
Aperçu

Une invocation est un type d'expression valeur applicable à la fois aux éléments Décision et aux éléments Métier Knowledge Modèle . Il s'agit d'une représentation sous forme de tableau de la manière dont la logique de décision définie dans un élément invocable (un Métier Knowledge Modèle ou un Décision Service) est invoquée par une Décision ou par un autre Métier Knowledge Modèle .



Barre d'outils pour l'éditeur d'invocation

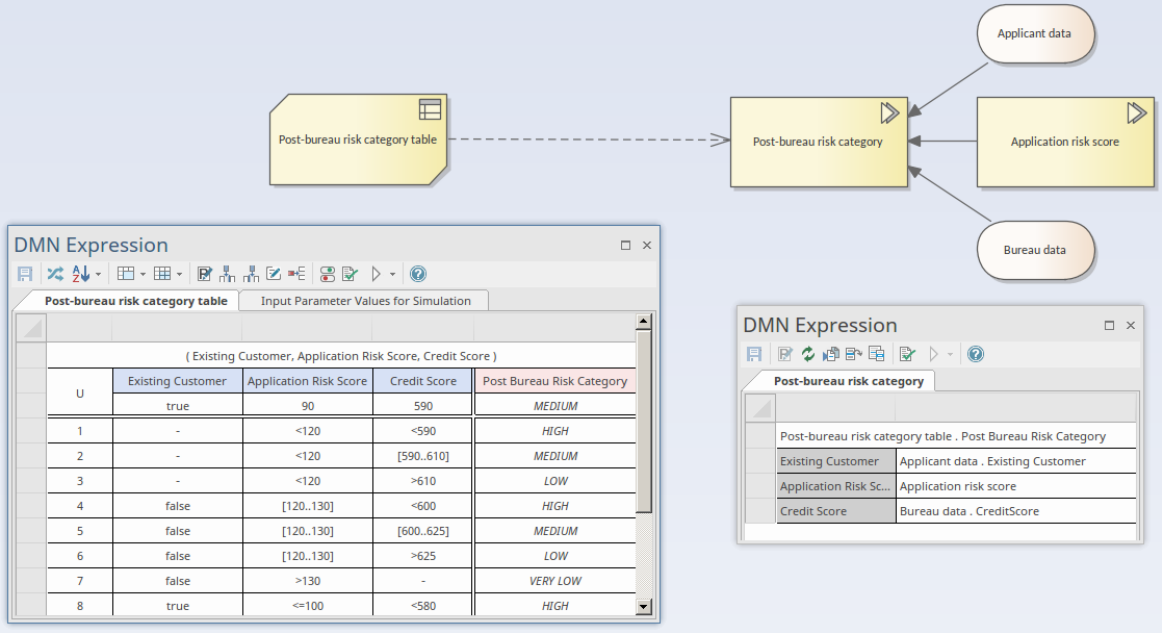
Lorsqu'une invocation est sélectionnée, un certain nombre de facilités permettant de travailler dessus sont accessibles depuis la barre d'outils de la fenêtre Expression DMN :



Pour plus de détails, reportez-vous à la rubrique d'aide « *Barre d'outils pour l'éditeur d'invocation* ».

Fixations

Les liaisons de paramètres d'une invocation fournissent le contexte pour l'évaluation du corps de l'élément invocable.

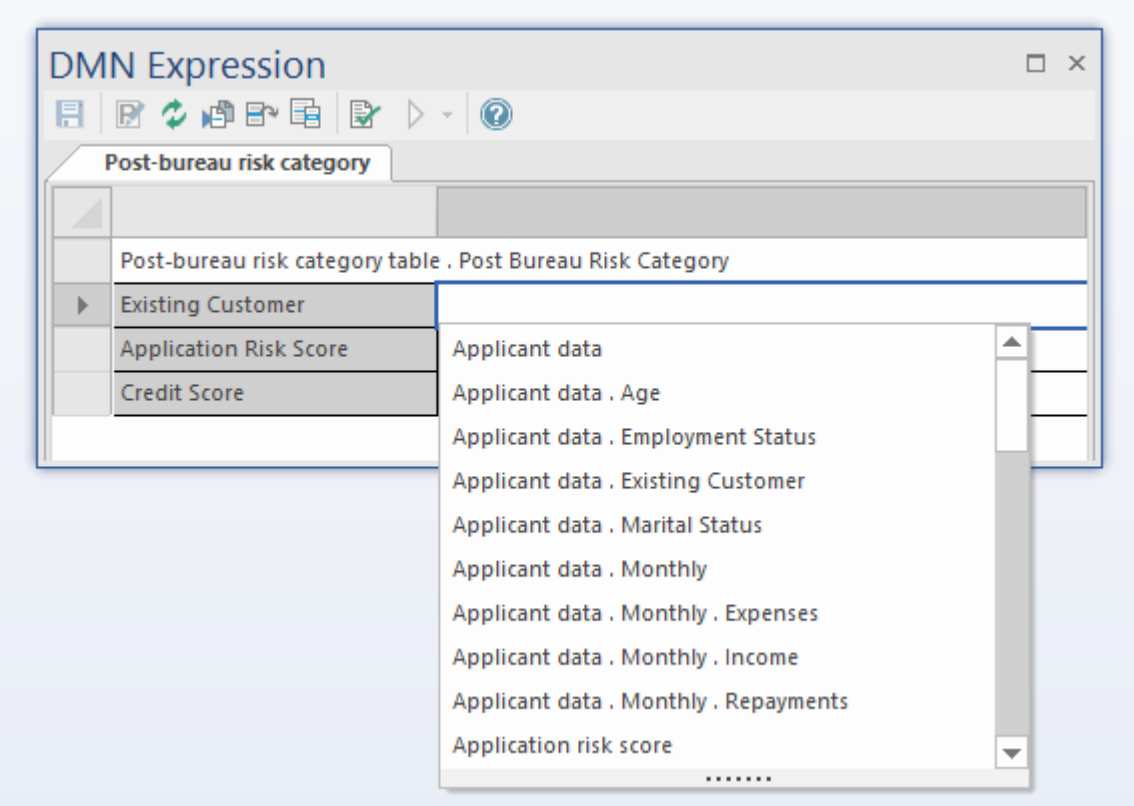


Dans cet exemple :

- La Décision « Catégorie de risque post-bureau » est représentée comme une Invocation se connectant au Métier Knowledge Modèle « tableau de catégorie de risque post-bureau », implémenté comme un Tableau de Décision
- La catégorie de risque « Post-bureau » Décision est la cible de trois connecteurs d'exigences d'information provenant de deux éléments de données d'entrée et d'un élément Décision
- La liste de liaison lie les valeurs d'entrée aux paramètres du Métier Knowledge Modèle
- L'invocation spécifie également la « clause de sortie » demandée ; dans le cas où un Tableau de Décision comporte plusieurs clauses de sortie définies, l'invocation doit explicitement demander une clause de sortie comme résultat de l'expression

Entrées

Les entrées provenant d'autres éléments Decisions et InputData peuvent être définies en appuyant sur la barre d'espace dans le champ :



Sortir

Comme une invocation ne peut invoquer qu'un seul Métier Knowledge Modèle , la sortie est définie par la sortie Métier Knowledge Modèle .

Barre d'outils pour l'éditeur d'invocation

Lorsqu'une expression d'invocation est sélectionnée, la barre d'outils de la fenêtre Expression DMN fournit des options spécifiques à ce type d'expression.

Options de la barre d'outils

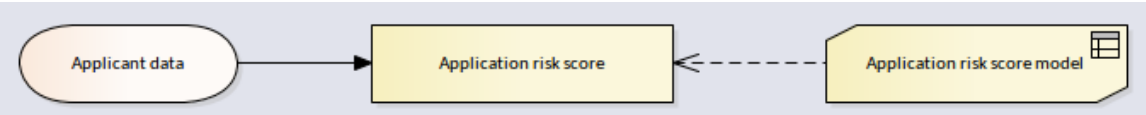
Ce tableau fournit des descriptions des fonctionnalités accessibles dans la fenêtre Expression DMN lorsqu'une invocation est sélectionnée.

Options	Description
	Cliquez sur ce bouton pour enregistrer la configuration dans le Décision ou le BusinessKnowledgeModel actuel.
	Cliquez sur ce bouton pour modifier les paramètres du Métier Knowledge Modèle .
	Applicable aux expressions valeur d'Invocation, pour les éléments Décision et les éléments Métier Knowledge Modèle (BKM). Cliquez sur ce bouton pour effectuer une synchronisation avec le BKM appelé. Par exemple, si le BKM change de nom, de paramètres, de sorties ou de types, cliquez sur ce bouton pour synchroniser ces modifications.
	Applicable aux expressions valeur d'Invocation, pour les éléments Décision et les éléments Métier Knowledge Modèle (BKM). Cliquez sur ce bouton pour définir ou modifier un BKM comme une invocation.
	Applicable aux expressions valeur d'Invocation, pour les éléments Décision et les éléments Métier Knowledge Modèle (BKM). Cliquez sur ce bouton pour ouvrir le BKM invoqué dans la fenêtre Expression DMN.
	Applicable aux expressions valeur d'Invocation, pour les éléments Décision et les éléments Métier Knowledge Modèle (BKM). Lorsqu'un BKM est implémenté en tant que Tableau de Décision , il peut définir plusieurs clauses de sortie ; l'invocation sur ce BKM peut devoir spécifier quelle sortie est demandée. Cliquez sur ce bouton pour lister toutes les sorties disponibles dans un menu contextuel ; la sortie actuellement configurée est cochée.
	Effectuez la validation de l'invocation. Enterprise Architect effectue une série de validations pour vous aider à localiser les erreurs dans la définition de l'invocation.
	Ce bouton est activé lorsque l'invocation est définie pour un Métier Knowledge Modèle . Sélectionnez l'onglet « Valeurs des paramètres d'entrée pour Simulation », complétez les champs et cliquez sur ce bouton. Le résultat du test sera présenté sur le Tableau de Décision , avec les valeurs d'exécution des entrées et des sorties affichées et les règles valides mises en évidence. Vous pouvez utiliser cette fonctionnalité pour tester unitairement un Métier Knowledge Modèle sans connaître le contexte et invoqué ultérieurement par une

	Décision ou un autre Métier Knowledge Modèle . Des options de menu sont disponibles pour ce bouton de la barre d'outils. Pour plus d'informations, consultez la rubrique d'aide *Simulate DMN Modèle* .

Exemple 1 - Lier les données d'entrée au Modèle de connaissances Métier

Un exemple complet peut être créé avec un Modèle Motif (dans le ruban, sélectionnez 'Simulate > Décision Analysis > DMN > Appliquer Perspective > DMN Décision > Décision With BKM : Create Modèle (s)').



Dans cet exemple, *les données d'entrée du demandeur* sont saisies dans *la définition des données du demandeur,* qui comporte trois composants.

Applicant data : Applicant data Definition

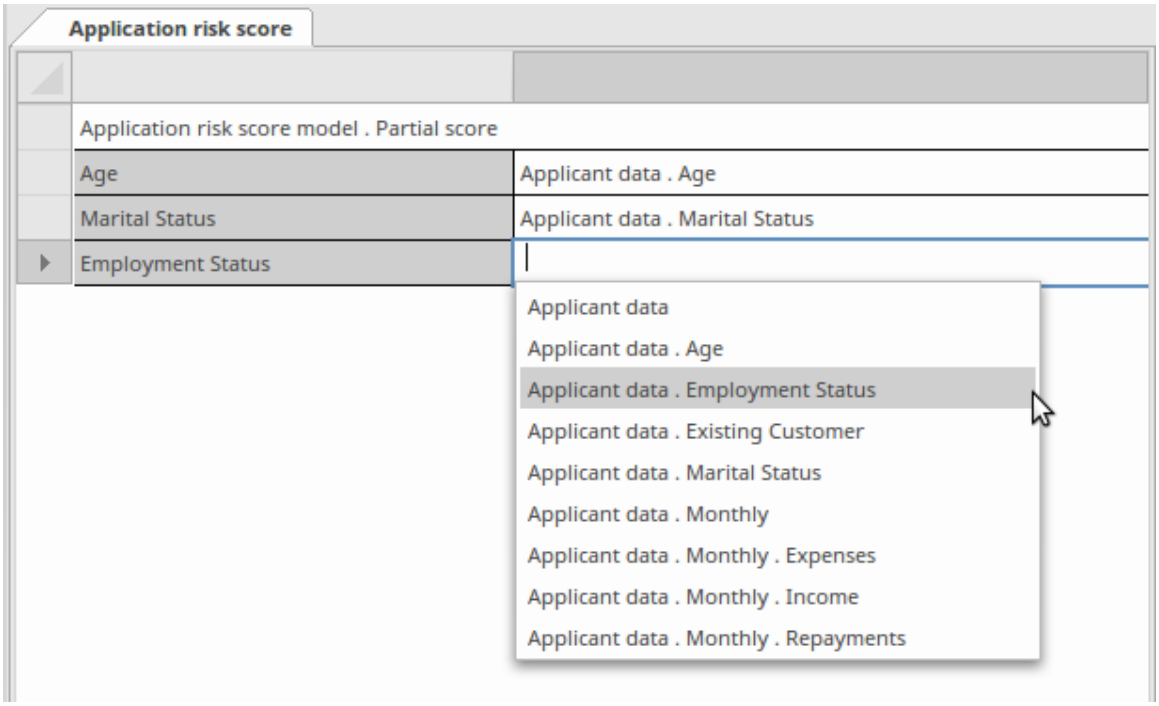
Applicant data Definition	Age : number	40
	Employment Status : string	"EMPLOYED"
	Marital Status : string	"M"

Le *modèle de score de risque d'application* Métier Knowledge Modèle est implémenté sous la forme d'un Tableau de Décision avec trois entrées et une sortie.

Application risk score model | Input Parameter Values for Simulation

	(Age, Marital Status, Employment Status)			
C+	Age	Marital Status	Employment Status	Partial score
	[18..120]	*S,M*	*UNEMPLOYED,STUDENT,EMPLOYE...*	
1	[18..21]	-	-	32
2	[22..25]	-	-	35
3	[26..35]	-	-	40
4	[36..49]	-	-	43
5	>=50	-	-	48
6	-	*S*	-	25
7	-	*M*	-	45
8	-	-	*UNEMPLOYED*	15
9	-	-	*STUDENT*	18
10	-	-	*EMPLOYED*	45
11	-	-	*SELF-EMPLOYED*	36

Le *score de risque de l'application* Décision est implémenté en tant qu'invocation pour lier les composants « feuille » des données d'entrée aux paramètres du BKM.

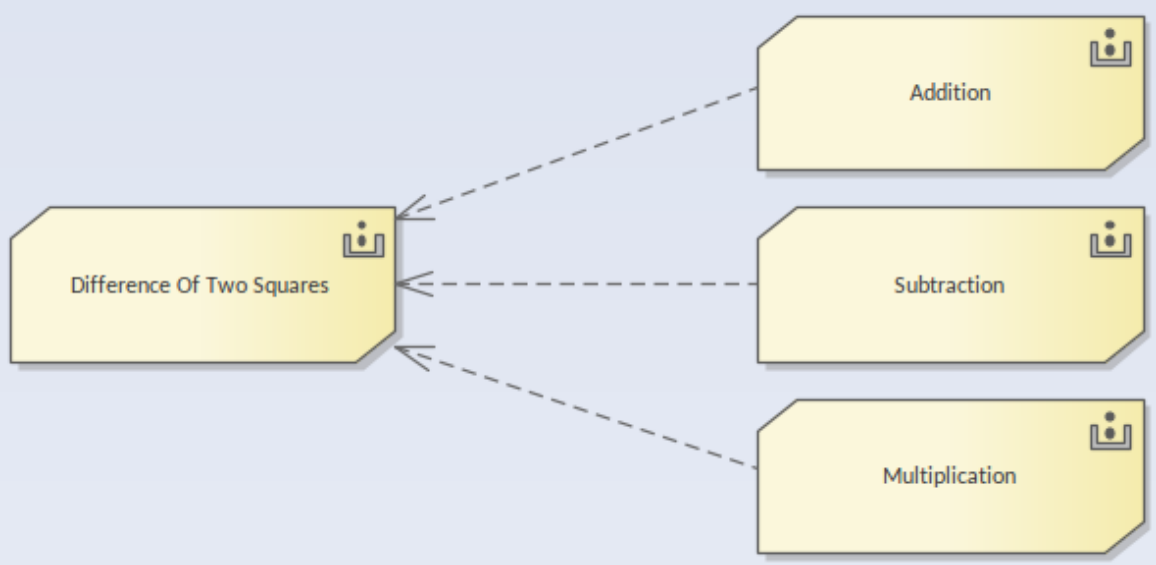


Afin de faciliter la liaison, la saisie semi-automatique est prise en charge pour l'expression de liaison.

Les instructions complètes modélisation et de simulation sont disponibles dans la documentation de Motif .

Exemple 2 - Lier des variables d'entrée de contexte à Métier Knowledge Modèle

Un exemple complet peut être créé avec un Modèle Motif (dans le ruban, sélectionnez 'Simulate > Décision Analysis > DMN > Appliquer Perspective > DMN Métier Knowledge Modèle Exemples > Métier Knowledge Modèle Invocation : Create Modèle (s)).



Dans cet exemple, le Métier Knowledge Modèle (BKM) *Difference Of Two Squares* est implémenté en tant que contexte encadré :

- La *somme variable de ab* est implémentée comme une invocation en liant les paramètres *a* et *b* à *l'addition* BKM
- La *différence variable de ab* est implémentée comme une invocation en liant les paramètres *a* et *b* à *la soustraction* BKM
- La *différence variable des carrés* est implémentée comme une invocation en liant les variables locales *somme de ab* et *différence de ab* à *la multiplication* BKM

Difference Of Two Squares | Input Parameter Values for Simulation

(a, b)		
sum of ab	Addition	
	addend 1	a
	addend 2	b
difference of ab	Subtraction	
	minuend	a
	subtrahend	b
difference of squares	Multiplication	
	factor 1	sum of ab
	factor 2	difference of ab
difference of squares		

Afin de faciliter la liaison, la saisie semi-automatique est prise en charge pour l'expression de liaison.

Les instructions complètes modélisation et de simulation sont disponibles dans la documentation de Motif .

Modifier Dialogue d'expression DMN

La dialogue « Modifier l'expression DMN » permet de définir des expressions dans les types d'éléments Boxed Content, Invocation et Literal Expression. Elle fournit support Intelli-sense pour la construction d'expressions basées sur la grammaire FEEL, ainsi que les langages de code qui peuvent être utilisés pour la génération de code du modèle.

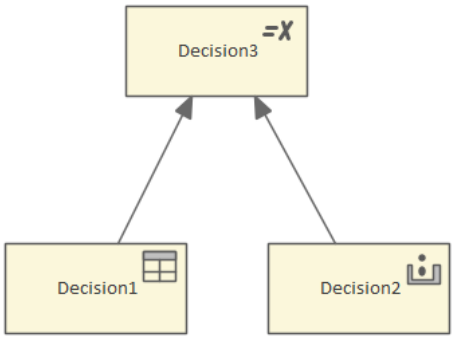
Éditeur d'expression DMN et support d'Intelli-sense

Pour vous aider à modifier des expressions avec moins de saisie et moins d'erreurs, Enterprise Architect fournit support Intelli-sense pour la modification des expressions.

Note que les noms des paramètres et des variables d'entrée de contexte peuvent contenir des espaces, conformément à la spécification du langage FEEL. Cette fonctionnalité est destinée à rendre chaque expression facile à lire.

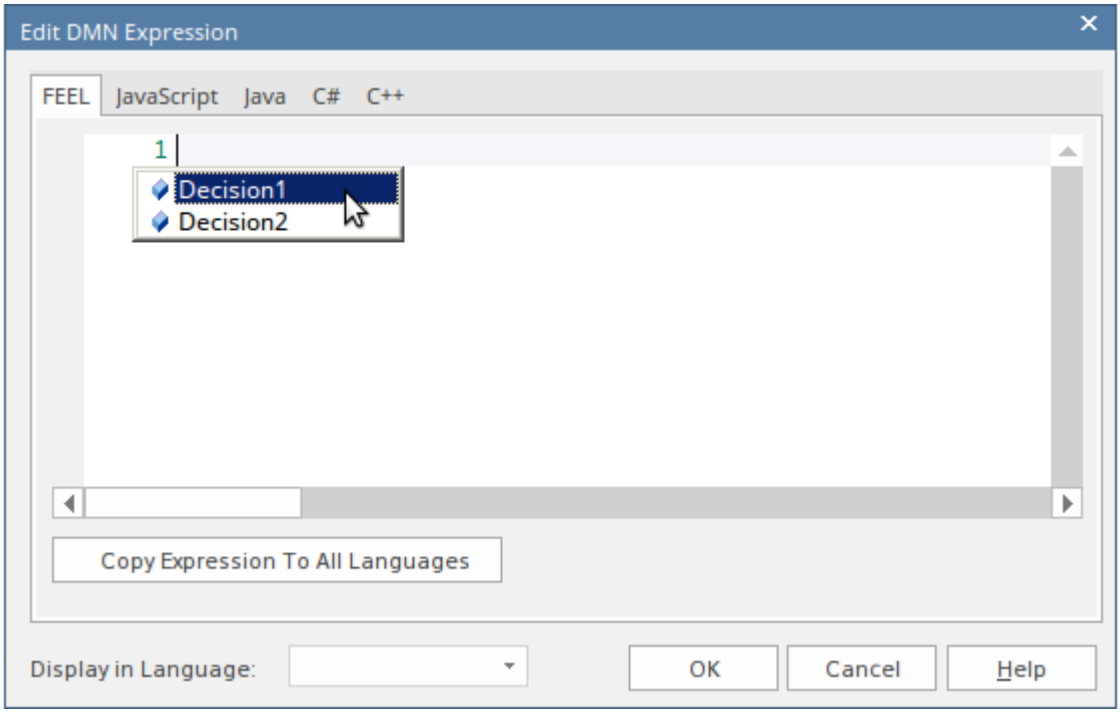
Exemples

Compte tenu de cette hiérarchie de décision, l'expression dans « Décision3 » est capable d'utiliser les sorties des deux décisions référencées.



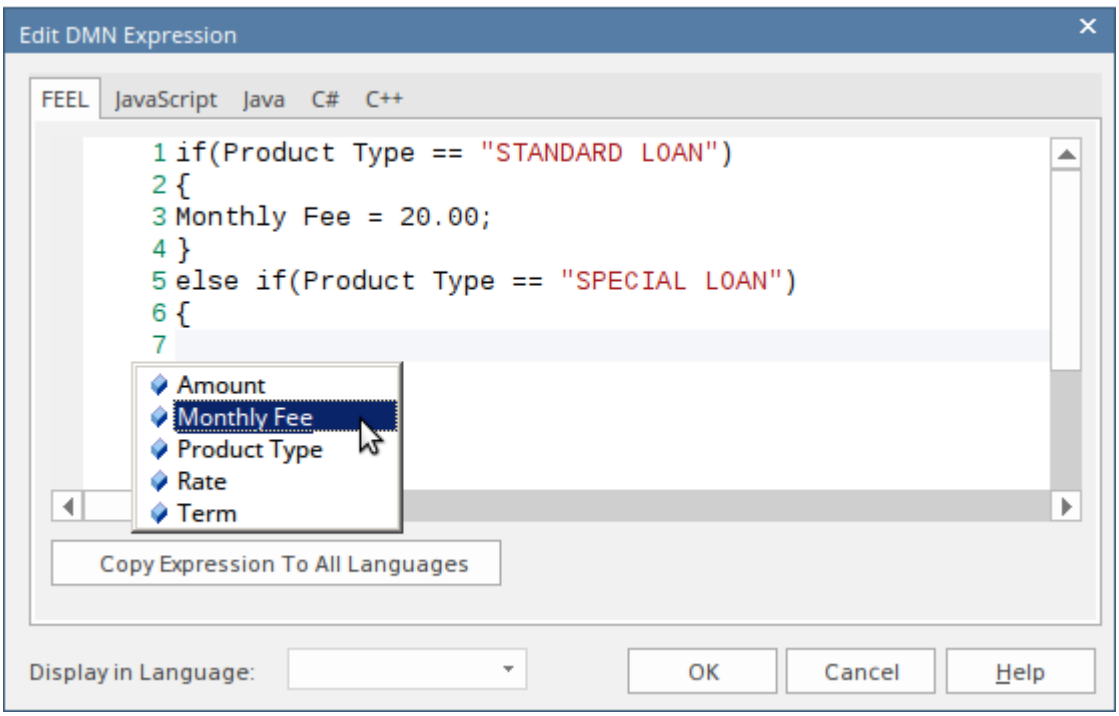
Pour ouvrir la dialogue « Modifier l'expression DMN » :

1. Double-cliquez sur l'élément Décision dans le diagramme pour afficher la fenêtre Expression DMN.
2. Cliquez-droit sur la ligne d'expression et sélectionnez l'option de menu 'Modifier l'expression'. La dialogue 'Modifier l'expression DMN' s'affiche.
3. Cliquez sur une ligne et appuyez sur Ctrl+Barre d'espace pour afficher le menu Intelli-sense :



- Pour une expression BusinessKnowledgeModel, tous les paramètres seront inclus
- Pour l'expression Décision toutes les décisions requises seront incluses
- Toutes les variables d'entrée de contexte antérieures à la variable actuelle seront incluses (les variables d'entrée de contexte postérieures à la variable actuelle sont exclues)

Dans cet exemple, en modifiant une expression de contexte encadré BKM, les paramètres d'entrée sont affichés dans le menu Intelli-sense :



Sélection de la langue

Le Modèle DMN peut être généré sous forme de code source en JavaScript , Java, C# ou C++. Comme la syntaxe diffère selon les langages, Enterprise Architect fournit des pages de remplacement de langage pour chaque langage. Si aucun code de remplacement n'est spécifié pour un langage, l'expression définie pour le langage FEEL sera utilisée.

Note : dans le code généré, l’espace à l’intérieur d’un nom de variable sera remplacé par un trait de soulignement.

Validation de l'expression DMN

DMN définit de nombreuses expressions, telles que FunctionDefinition, DecisionTable, Boxed Context, Invocation et Literal Expression. Les paramètres, les arguments et la logique de ces expressions sont implémentés en grande partie par « texte ».

Pour rendre modélisation plus simple et plus fiable, Enterprise Architect fournit deux fonctionnalités : l'auto-complétion et la validation.

- Validation : identifie les erreurs modélisation causées par des fautes de frappe, une logique incomplète, des incohérences, etc.
- Complétion automatique : vous pouvez sélectionner une string de texte dans une liste d'énumérations plutôt que de saisir le texte.

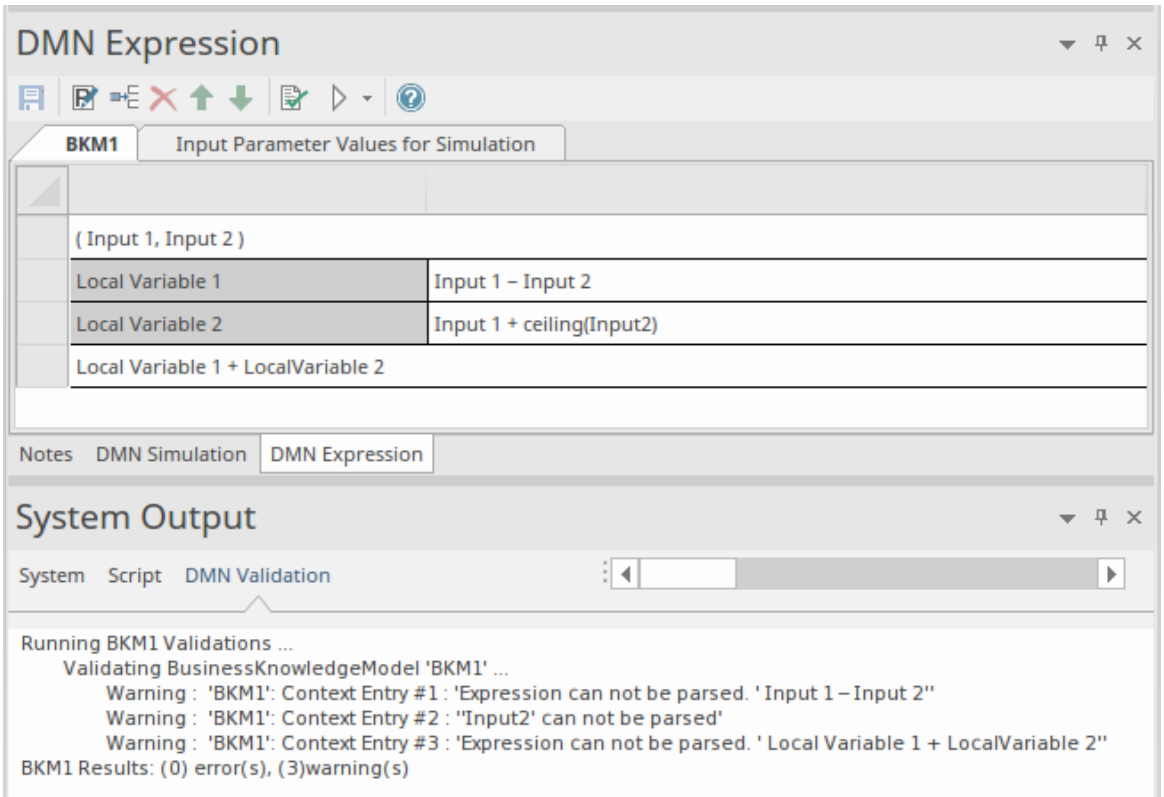
Dans cette rubrique, nous vous montrerons comment valider une expression DMN.

Accéder

Fenêtre d'expression DMN	Simuler > Analyse Décision > DMN > Expression DMN : bouton Valider
Fenêtre Simulation DMN	Simuler > Décision Analysis > DMN > Ouvrir Simulation DMN > Simuler : icône Valider

Validations courantes

Validation du nom de la variable



Dans cet exemple, le Modèle de connaissances Métier BKM1 du contexte encadré définit deux paramètres, « Entrée 1 » et « Entrée 2 », et deux variables locales, « Variable locale 1 » et « Variable locale 2 ». L'expression a été validée et les résultats sont exportés vers l'onglet « Validation DMN » de la fenêtre Sortie système.

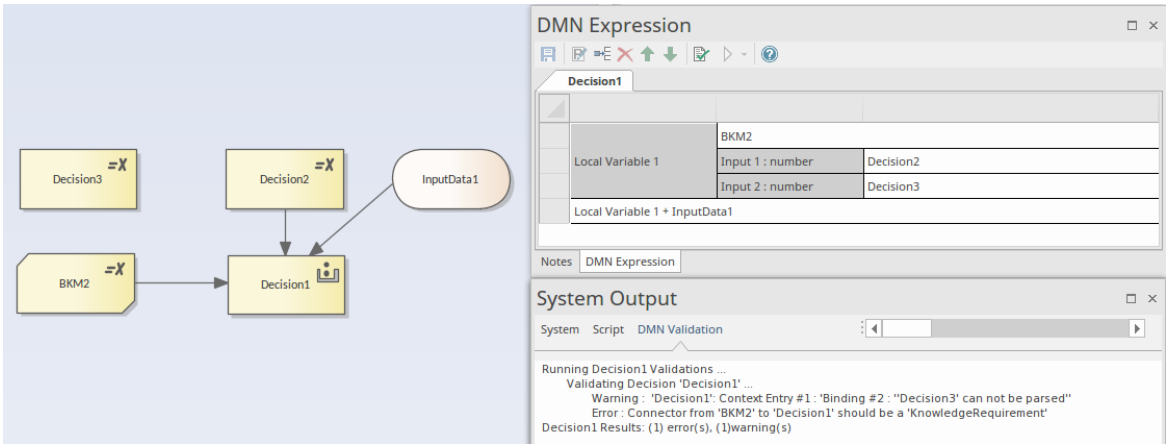
- L'entrée de contexte n°1 a échoué en raison d'une erreur typographique ; il devrait s'agir de l'opérateur « - », mais l'utilisateur a tapé ou copié « – »
- L'entrée de contexte n°2 a échoué car il n'y a pas d'espace entre « Input » et le numéro 2 ; note que la fonction « ceiling() » est définie dans la Bibliothèque DMN afin qu'elle puisse être analysée avec succès
- L'entrée de contexte n°3 a échoué car il n'y a pas d'espace entre « Local » et « Variable »

Il est difficile d'identifier visuellement ce type d'erreur. L'exécution d'une validation peut aider à identifier les erreurs et vous pouvez ensuite facilement effectuer une correction.

Validation des dépendances

Une décision peut nécessiter d'autres décisions, des données d'entrée et des modèles de connaissances métier ; ces relations sont identifiées par les connecteurs InformationRequirement et KnowledgeRequirement.

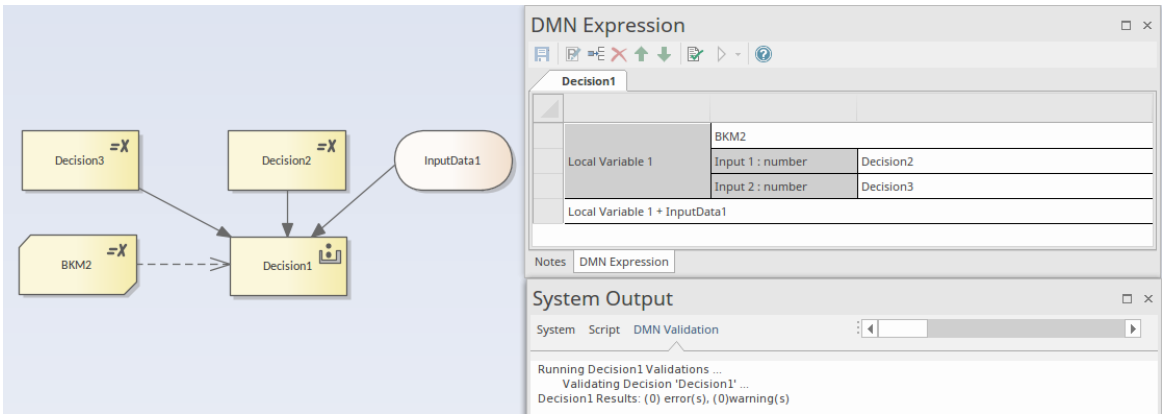
Lorsque le graphique devient complexe, il est fort possible que certains connecteurs soient manquants ou que le mauvais type de connecteur soit utilisé.



Dans cet exemple, cliquez sur le bouton Valider, Enterprise Architect montrera que :

- « Decision3 » est utilisé par « Decision1 » en se liant à un paramètre du BKM2 appelé ; cependant, il n'est pas défini - un connecteur InformationRequirement est manquant
- L'invocation définie dans « Decision1 » n'est pas valide ; le type de connecteur de « BKM2 » à « Decision1 » doit être un KnowledgeRequirement

Après avoir résolu ces problèmes, exécuter à nouveau la validation :



Complétion automatique des expressions DMN

DMN définit de nombreuses expressions, telles que FunctionDefinition, DecisionTable, Boxed Context, Invocation et Literal Expression. Les paramètres, les arguments et la logique de ces expressions sont implémentés en grande partie par du texte.

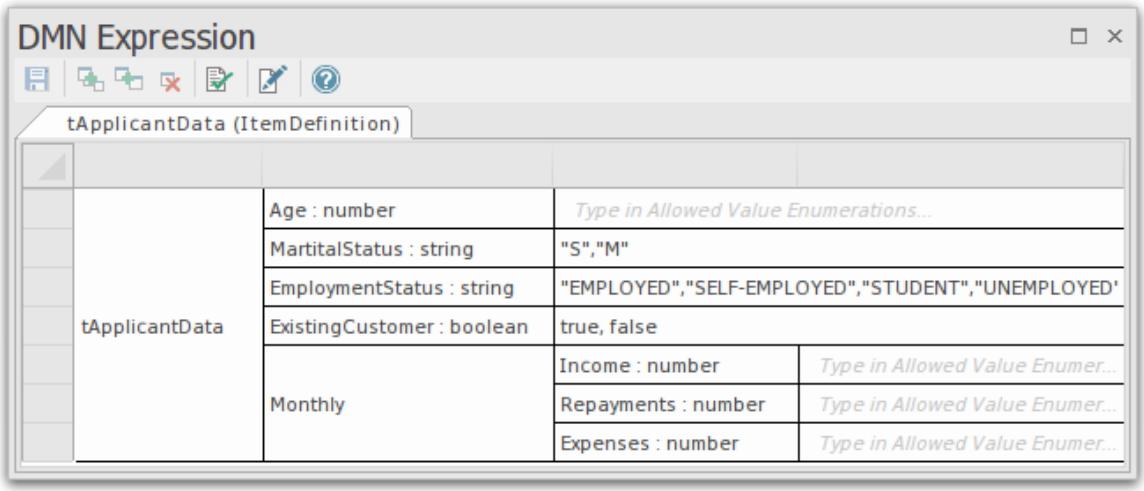
Pour rendre modélisation simple et fiable, Enterprise Architect fournit un facilité saisie semi-automatique, aidant à fournir :

- Valeurs autorisées de ItemDefinition
- Entrées/Sorties d'un Tableau de Décision
- Informations requises

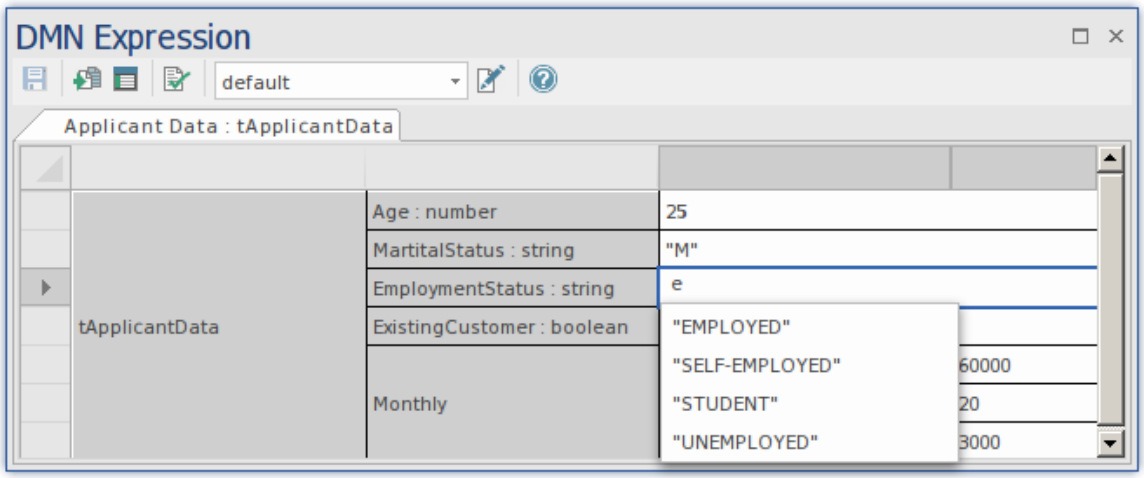
Valeurs autorisées de ItemDefinition

L'idée est de définir les énumérations valeur autorisées dans ItemDefinition, puis de composer une liste pour la sélection chaque fois que ces valeurs sont demandées.

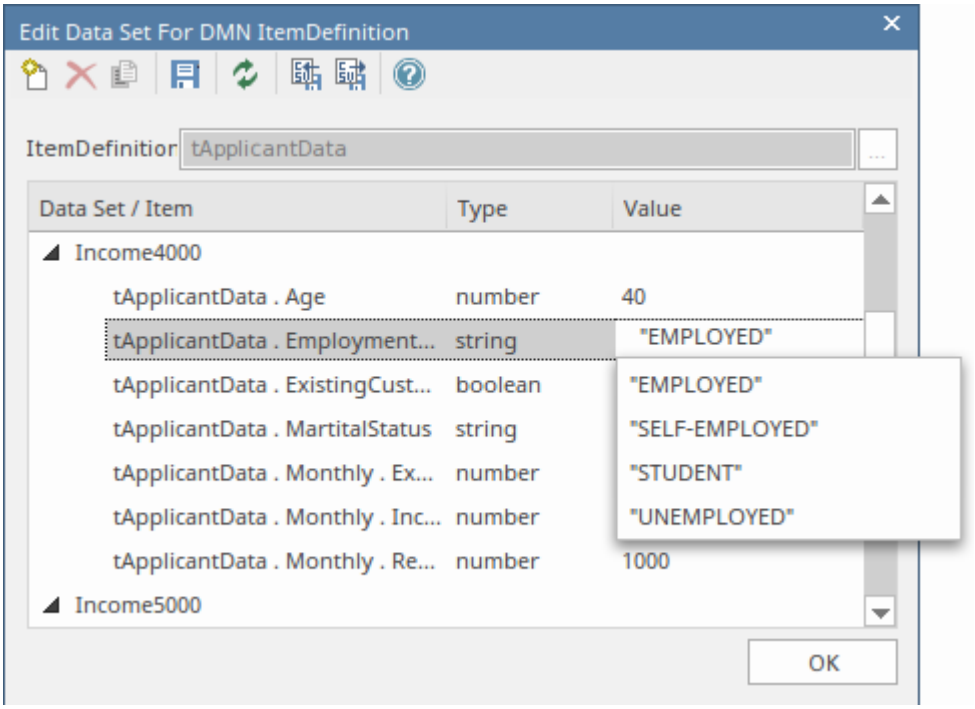
Dans cet exemple, ItemDefinition 'Données du candidat. Statut d'emploi' définit une énumération de valeurs autorisées.



Lors de la modification des valeurs des InputData saisies dans cet ItemDefinition, appuyez sur la barre d'espace du clavier pour afficher une liste de valeurs parmi lesquelles sélectionner.

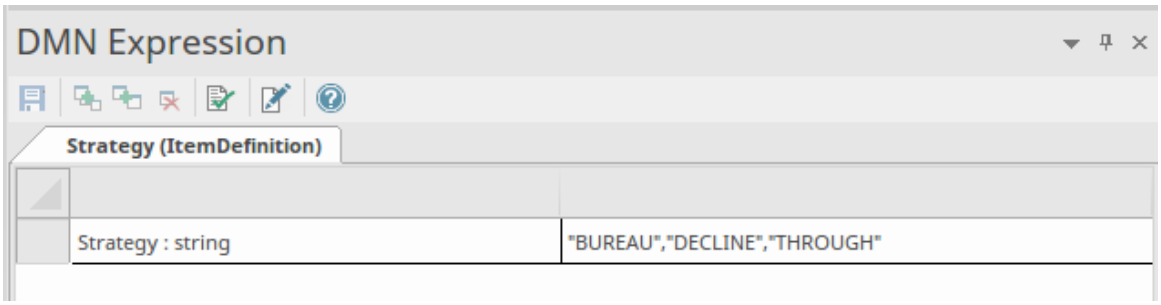


Nous pourrions également définir plusieurs ensembles de données pour InputData, car la fonctionnalité de saisie semi-automatique est disponible dans cette dialogue .

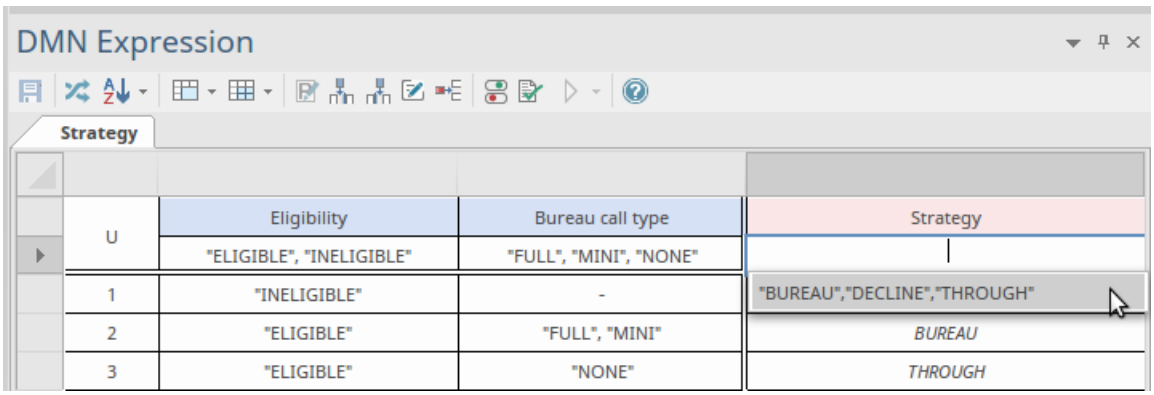


Entrées/Sorties d'un Tableau de Décision

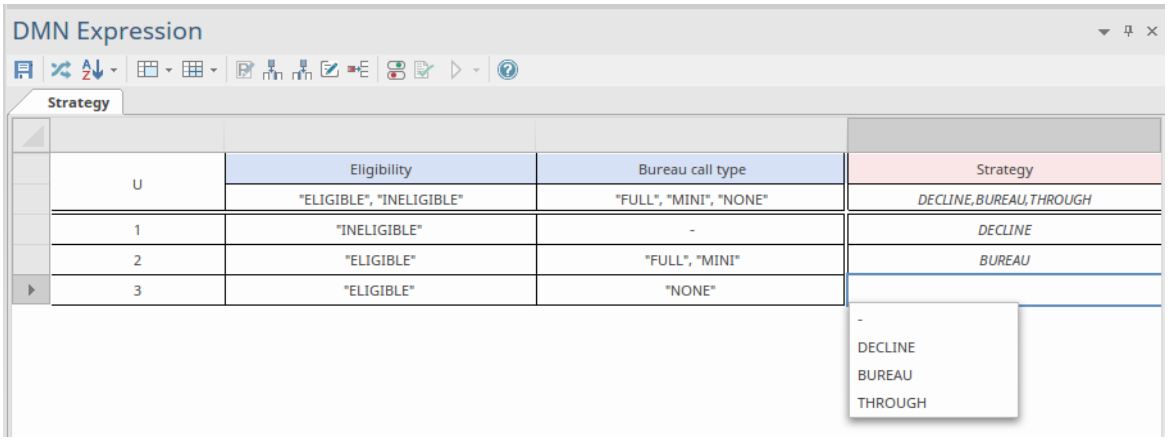
Prenons l'exemple de la définition de l'élément « Stratégie » :



Nous pouvons rapidement remplir le champ « Valeurs autorisées » pour un Tableau de Décision par sélection :



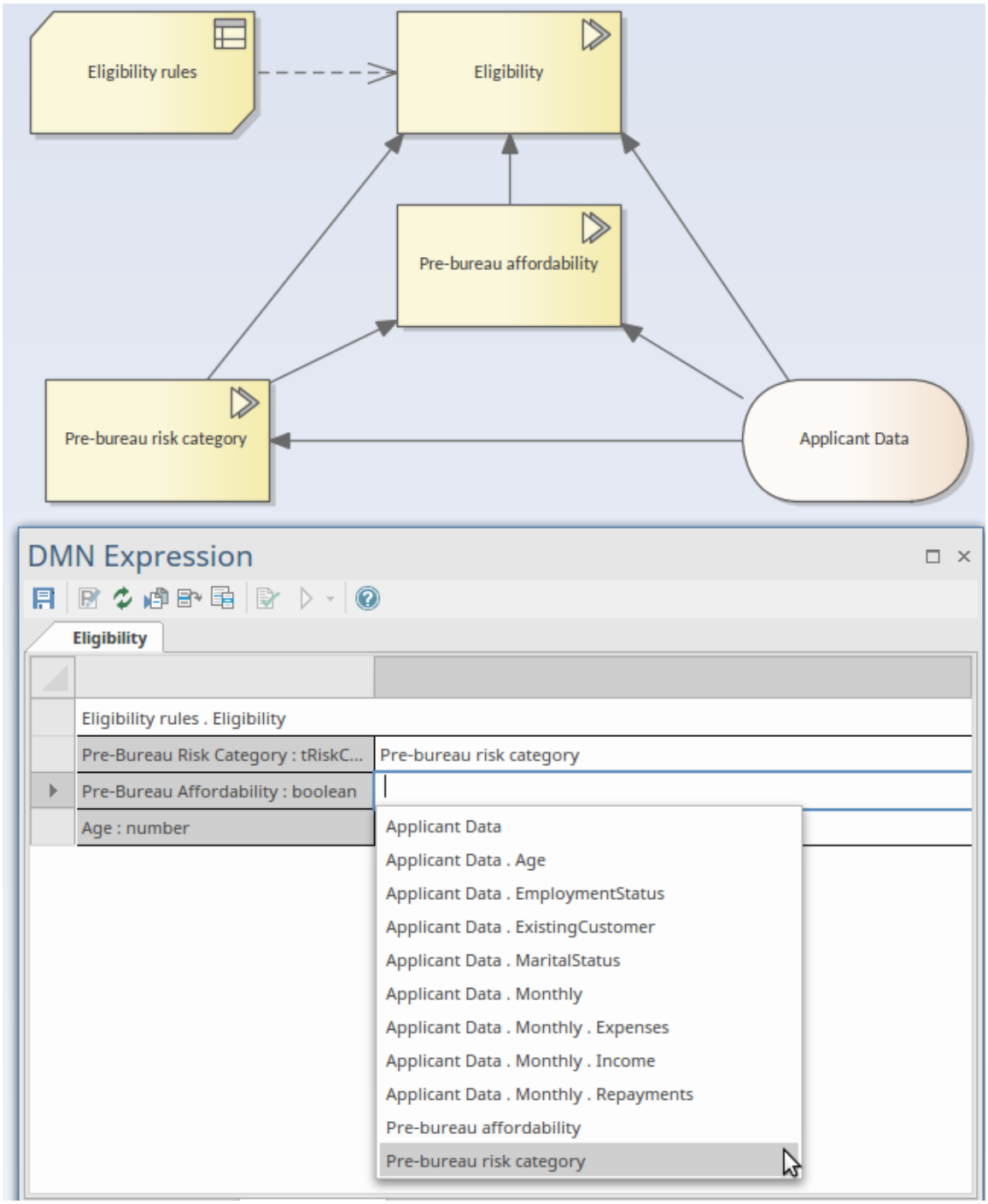
Nous pouvons ensuite rapidement remplir les règles Tableau de Décision par sélection :



Note : la valeur par défaut « - » indique « Indéfini ».

Exigences en matière d'information

Dans une hiérarchie de décision, une décision peut accéder aux décisions requises et aux données d'entrée ; ces éléments requis forment une liste de variables qui peuvent être utilisées par la décision.



Dans cet exemple, Décision « Éligibilité » nécessite deux décisions - « Catégorie de risque avant l'ouverture du bureau » et « Capacité financière avant l'ouverture du bureau » - et un élément de données d'entrée « Données du demandeur ».

Lors de la définition des valeurs de liaison pour les « Règles d'éligibilité » du BusinessKnowledgeModel invoqué, une liste de saisie semi-automatique vous prompt à effectuer une sélection. Cette liste contient des noms de sous-décisions, c'est-à-dire des composants de feuille des données d'entrée. Grâce à cette fonctionnalité , vous pouvez facilement configurer une invocation.

Modélisation avec DMN

Cette rubrique vous présente les éléments les plus importants dont vous avez besoin pour créer des modèles Décision . Comme indiqué dans les rubriques précédentes, un modèle de décision comporte deux parties fondamentales, à savoir le diagramme Exigences Décision et la logique de décision. La création d'un diagramme de décision est simple et la partie la plus onéreuse de l'exercice consistera probablement à comprendre la manière dont une organisation prend des décisions et les éléments qui entrent en jeu dans ces décisions. Les diagrammes contiennent généralement des décisions enchaînées qui décrivent le fait qu'une décision peut fournir des éléments à une autre décision, et ainsi de suite.

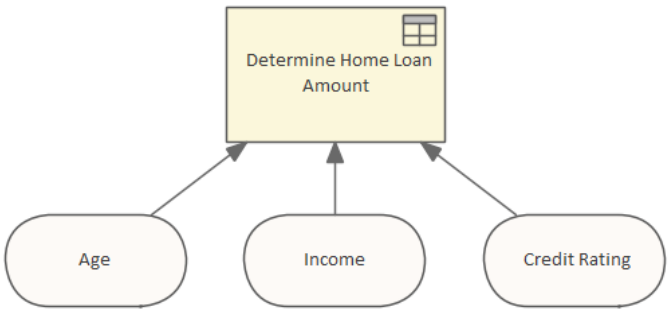


diagramme Exigences Décision simple montrant trois entrées dans une Décision à l'aide d'un Tableau de Décision .

La logique de la décision est décrite à l'aide de plusieurs outils, mais la forme la plus couramment utilisée et la plus accessible est le Tableau de Décision . Le Tableau de Décision contient des lignes et des colonnes comme une feuille de calcul et couvre toutes les combinaisons possibles d'entrées pour produire un certain nombre de sorties. Par exemple, si un demandeur a plus de 21 ans et moins de 65 ans et gagne 60 000 $ par an avec une bonne cote de crédit, la banque lui prêtera 300 000 $ pour un prêt immobilier.

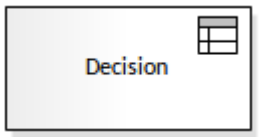
Determine Home Loan Amount

U	Age	Income	Credit Rating	Loan Amount
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-

Un Tableau de Décision montrant trois entrées et une sortie, des lignes seraient ajoutées pour définir les règles.

Décision

Un élément Décision est utilisé pour évaluer une sortie en fonction d'une ou plusieurs entrées. La logique qui détermine la sortie est soit définie dans cet élément Décision , soit elle invoque la logique de décision contenue dans un Métier Knowledge Modèle qui est connecté à l' Décision .



Entrées

Une Décision peut avoir n'importe quel nombre d'entrées, y compris la possibilité de définir les valeurs d'entrée dans l'élément. L'entrée la plus courante consiste à utiliser un élément de données d'entrée.

Sortir

Une Décision peut avoir une ou zéro sortie. La sortie peut être un ensemble de données complexe.

Expressions de valeur

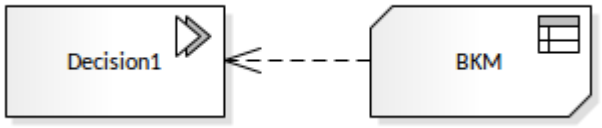
La sortie d'un élément Décision est déterminée à l'aide d'une expression de valeur. L'expression de valeur contient la logique de décision de l'élément et peut prendre l'une des quatre formes suivantes : Tableau de Décision , Expression littérale, Invocation ou Contexte encadré. Les expressions de valeur sont définies et modifiées à l'aide de l'éditeur d'expressions DMN, qui affiche l'un des quatre formats en fonction du type d'expression utilisé.

Lorsqu'il est affiché sur un diagramme , l'élément Décision affiche une icône dans le coin supérieur droit qui indique le type d'expression valeur qu'il utilise.

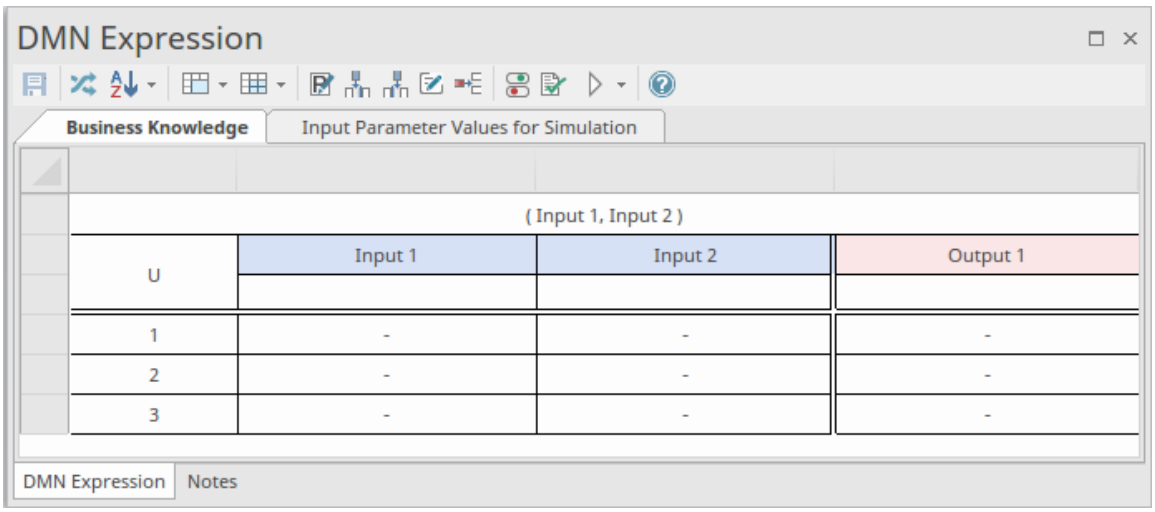
Type	Description
	Un Tableau de Décision est une représentation tabulaire d'un ensemble d'expressions d'entrée et de sortie liées, organisées en règles indiquant quelle entrée de sortie s'applique à un ensemble spécifique d'entrées d'entrée.
	Une expression littérale est la forme la plus simple d'expression DMN. Elle est généralement définie comme une instruction d'une seule ligne ou un bloc conditionnel if-else.
	Une invocation Décision nécessite qu'un élément Métier Knowledge Modèle soit référencé à l'aide d'un connecteur Knowledge Requirement. L'élément Décision contient simplement les paramètres qui fournissent le contexte pour l'évaluation du Métier Knowledge Modèle (BKM). Une partie ou la totalité du résultat renvoyé par le BKM peut être définie pour être transmise en tant que sortie de Décision .
	Un contexte encadré est une collection d'entrées de contexte. Chaque entrée de contexte se compose d'une variable et d'une expression. Le contexte possède également une valeur de résultat.

Modèle de connaissances Métier

Un élément Métier Knowledge Modèle (BKM) représente un élément de logique de décision réutilisable. En général, il est connecté à un élément Décision qui invoque le BKM et transmet un ensemble d'entrées. Le BKM, à l'aide de sa logique interne, évalue une sortie qui est renvoyée à l' Décision .



À moins qu'un BKM ne fonctionne sur des valeurs fixes, il nécessite généralement la définition d'un ensemble de paramètres d'entrée, ainsi que la définition d'une sortie. Les paramètres et la logique de décision sont définis à l'aide de la fenêtre Expression DMN.



Entrées et sorties

Lorsqu'il est utilisé dans un modèle de décision, un BKM doit être connecté via un KnowledgeRequirement à un Décision ou à un autre BKM, par l'intermédiaire duquel il reçoit ses entrées. Les paramètres d'entrée sont définis à l'aide de l'icône . Ceux-ci peuvent être définis comme un type simple ou un type complexe défini à l'aide d'une ItemDefinition. La dénomination des paramètres d'entrée influence la dénomination dans l'expression de valeur.

Sortir

Une sortie BKM se fait via un KnowledgeRequirement qui doit être une entrée d'une Décision ou d'un autre BKM. La sortie est définie à l'aide de :

- L'icône pour une expression littérale
- Colonne(s) de sortie dans le tableau Expression DMN pour un Tableau de Décision , un Contenu encadré et une Invocation.

Une sortie peut être un type simple ou un type complexe défini à l'aide d'un ItemDefinition.

Expressions de valeur

Pour définir un moyen d'évaluation d'un résultat, basé sur la logique de décision, un élément Métier Knowledge Modèle (BKM) contient une expression de valeur. Celle-ci est définie et modifiée à l'aide de la fenêtre Expression DMN, qui comporte quatre formats, le format étant déterminé par le type d'expression de valeur que vous souhaitez utiliser.

L'élément BKM peut être défini avec ces structures pour l'expression de valeur. Chacune est affichée dans le modèle avec une icône.

Type	Description
	Un Tableau de Décision est une représentation tabulaire d'un ensemble d'expressions d'entrée et de sortie liées, organisées en règles indiquant quelle entrée de sortie s'applique à un ensemble spécifique d'entrées d'entrée.
	Une expression littérale est la forme la plus simple d'une expression DMN. Elle est généralement définie comme une instruction d'une seule ligne ou un bloc conditionnel if-else.
	Une invocation Décision nécessite qu'un élément de modèle de connaissances Métier soit référencé à l'aide d'un connecteur d'exigence de connaissances. Il contient simplement les paramètres qui fournissent le contexte pour l'évaluation d'un modèle de connaissances métier.
	Un contexte encadré est une collection d'entrées de contexte. Chaque entrée de contexte se compose d'une variable et d'une expression. Le contexte possède également une valeur de résultat.

Validation et Tester

Pour garantir qu'un élément BKM est capable de produire une sortie correcte, il peut être validé à l'aide de l'icône Validation  . Un BKM peut également être testé en tant qu'unité pour garantir son fonctionnement à l'aide du bouton  Simulation . Pour plus de détails, consultez la rubrique d'aide *Valeurs des paramètres d'entrée pour Simulation* .

Paramètres BKM

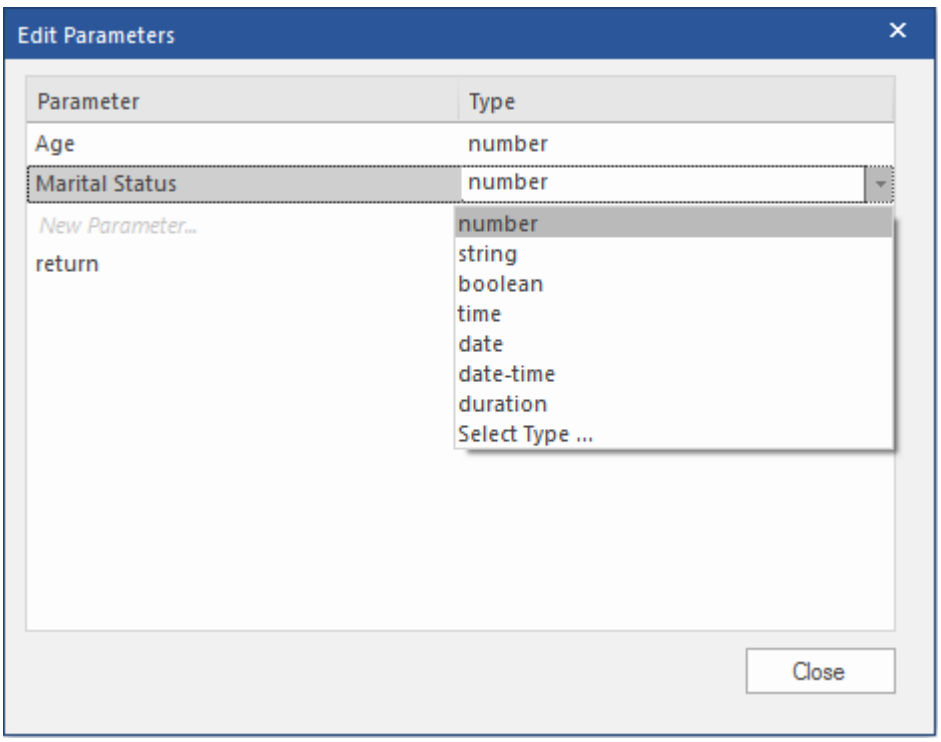
Un Métier Knowledge Modèle (BKM) est implémenté sous la forme d'une définition de fonction, avec des paramètres et une expression DMN comme corps (comme Tableau de Décision , Boxed Context ou Literal Expressions).

Comme un BKM est destiné à fonctionner de manière autonome et à être appelé par Decisions ou d'autres BKM, il est nécessaire de définir tous les paramètres d'entrée. De plus, pour les expressions littérales, vous devez définir le paramètre de sortie.

Lors de la définition de paramètres d'entrée, vous pouvez les définir avec des valeurs par défaut à des fins de test. Après avoir créé un BKM, pour vérifier qu'il fonctionne correctement, vous pouvez exécuter une simulation basée sur ces valeurs par défaut.

Paramètres d'un Modèle de connaissances Métier

Pour ouvrir la dialogue « Modifier les paramètres », dans la fenêtre Expression DMN, cliquez sur le bouton Modifier les paramètres  :



Note : ceci est un exemple d’expression littérale qui inclut un type *de retour* .

Modifier les paramètres

Vous pouvez effectuer ces actions sur les paramètres :

Action	Description
	Ajoutez un nouveau paramètre en le saisissant dans la ligne « Nouveau paramètre... ».
	Modifiez le nom du paramètre existant en l'éditant sur place dans la cellule.

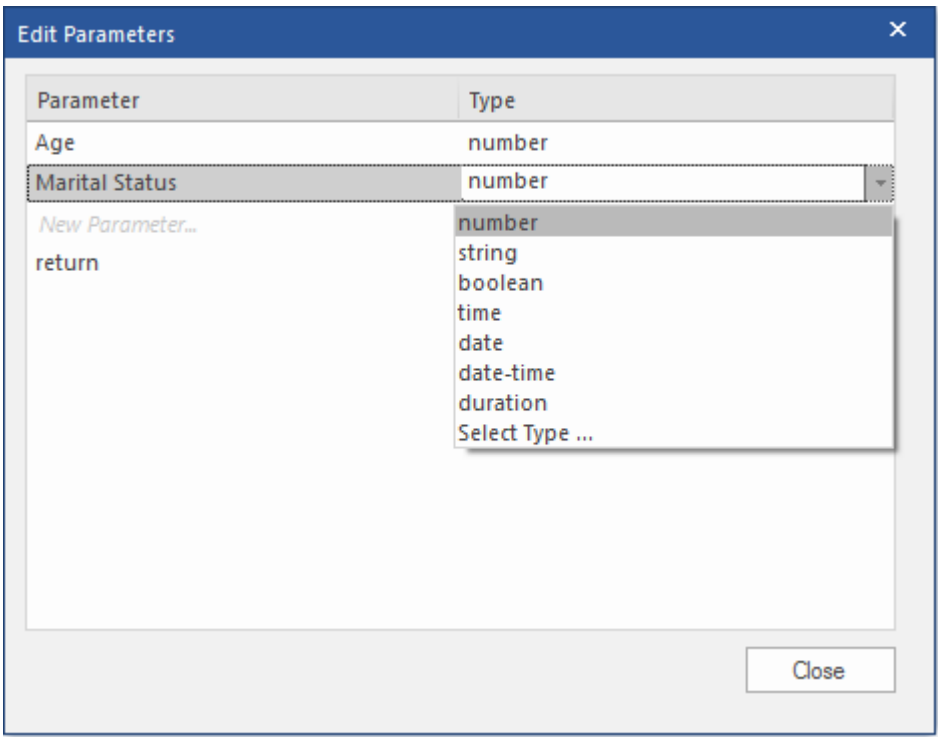
	Supprimer un paramètre existant à l'aide du menu contextuel.
	Cliquez sur le Type pour activer une liste déroulante. Sélectionnez un type pour le paramètre dans la liste déroulante. **Définir un Type de définition Item** Lors du changement du type de paramètre, il existe une option permettant de sélectionner un type prédéfini à partir d'une définition d'élément. L'option pour cela est « Sélectionner Type ... ». Lorsque cette option est sélectionnée, une dialogue permettant de sélectionner une définition d'élément s'ouvre.

Valeurs des paramètres d'entrée pour Simulation

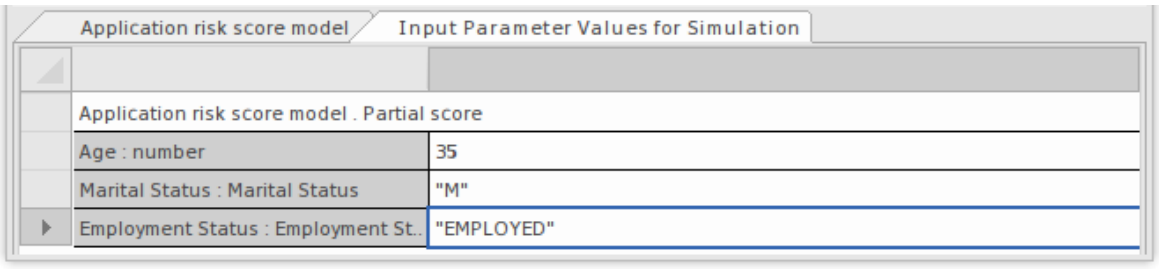
Comme un Métier Knowledge Modèle est autonome, il est possible d'effectuer un « Test unitaire » de simulation en fournissant un ensemble de valeurs par défaut comme entrée pour ses paramètres. Ces valeurs peuvent être définies dans l'onglet *Valeurs des paramètres d'entrée pour Simulation* de la fenêtre Expression DMN.

Paramètres d'un Métier Knowledge Modèle (BKM)

Les paramètres d'un BKM sont accessibles depuis la fenêtre Expression DMN, à l'aide du bouton Modifier les paramètres dans la barre d'outils :



Un ensemble de valeurs par défaut pour ces paramètres, qui peuvent être utilisés dans une simulation du BKM, est défini dans l'onglet « Valeurs des paramètres d'entrée pour Simulation » dans la fenêtre Expression DMN :



Avec ces paramètres définis, le BKM peut être testé à l'aide du bouton Simulation .

Exemples Simulation

Voici deux exemples d'utilisation des *valeurs des paramètres d'entrée pour Simulation* .

Type	Description

Tableau de Décision	Un exemple de simulation d'un élément Tableau de Décision BKM basé sur les valeurs définies dans l'onglet *Valeurs des paramètres d'entrée pour Simulation* .
Expression littérale	Un exemple de simulation d’un élément d’expression littérale BKM basé sur des valeurs définies dans l’onglet *Valeurs des paramètres d’entrée pour Simulation* .

Exemple Simulation Tableau de Décision

L'exemple Métier Knowledge Modèle (BKM) décrit dans cette section est disponible auprès du Constructeur de Modèle (Ctrl+Shift+M). Sélectionnez un Paquetage hôte dans votre modèle, invoquez le Constructeur de Modèle et - dans le menu déroulant Perspectives - sélectionnez ' Exigences | Décision Modélisation '.

Pour accéder à l'exemple utilisé dans cette section :

- Créer un motif pour « DMN Décision | Un exemple complet »
- Naviguez dans la fenêtre Navigateur jusqu'à « Un exemple complet | Modèles de connaissances Métier »

Il est également disponible dans le modèle d'exemple Enterprise Architect (EAExample) :

- Naviguez dans la fenêtre Navigateur jusqu'à 'Analyse et Modélisation Métier > Exemples DMN > Un exemple complet > Métier Knowledge Models'

Double-cliquez sur l'élément « Règles d'éligibilité » pour ouvrir le BKM dans la fenêtre Expression DMN

Lorsqu'un Tableau de Décision est créé pour un Métier Knowledge Modèle , nous pouvons tester ce BKM en liant certaines valeurs :

Eligibility rules | Input Parameter Values for Simulation

(Pre-Bureau Affordability, Pre-Bureau Risk Category, Age)

P	Pre-Bureau Risk Category	Pre-Bureau Affordability	Age	Eligibility
	VERY LOW, LOW, MEDIUM			*INELIGIBLE, ELIGIBLE*
1	*DECLINE*	-	-	*INELIGIBLE*
2	-	false	-	*INELIGIBLE*
3	-	-	<18	*INELIGIBLE*
4	-	-	-	*ELIGIBLE*

Nous pouvons fournir des valeurs de test telles que celles-ci :

Eligibility rules | **Input Parameter Values for Simulation**

Eligibility rules . Eligibility	
Pre-Bureau Affordability	true
Pre-Bureau Risk Category	"VERY LOW"
Age	16

Cliquez sur le bouton Simulation  de la barre d'outils pour obtenir ce résultat :

Eligibility rules | Input Parameter Values for Simulation

(Pre-Bureau Affordability = true, Pre-Bureau Risk Category = "VERY LOW", Age = 16)

P	Pre-Bureau Risk ...	Pre-Bureau Affor...	Age	Eligibility
	VERY LOW	true	16	*INELIGIBLE*
1	*DECLINE*	-	-	*INELIGIBLE*
2	-	false	-	*INELIGIBLE*
3	-	-	<18	*INELIGIBLE*
4	-	-	-	*ELIGIBLE*

- La valeur du paramètre d'exécution remplacera les « Valeurs autorisées » en mode simulation
- Les règles valides sont mises en évidence
- Étant donné que la politique Hit de ce Tableau de Décision est P (Priorité), le résultat final est déterminé par l'ordre des valeurs de sortie ; étant donné que « INÉLIGIBLE » et « ÉLIGIBLE » sont les valeurs de sortie et que « INÉLIGIBLE » vient avant « ÉLIGIBLE », la règle n°3 donnera le résultat final et ce candidat est « INÉLIGIBLE ».

Exemple Simulation d'expression littérale

Le Métier Knowledge Modèle (BKM) décrit dans cette section est disponible auprès du Constructeur de Modèle (Ctrl+Shift+M). Sélectionnez un Paquetage hôte dans votre modèle, invoquez le Constructeur de Modèle et - dans le menu déroulant Perspectives - sélectionnez ' Exigences | Décision Modélisation '.

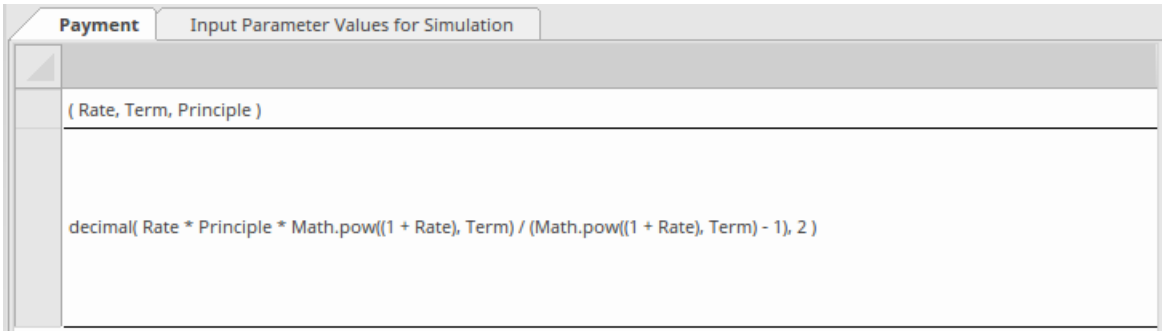
Pour accéder à l'exemple utilisé dans cette section :

- Créer un motif pour 'DMN Métier Knowledge Modèle > Métier Knowledge Modèle Literal Expression'
- Naviguez dans la fenêtre Navigateur jusqu'à « Métier Knowledge Modèle Literal Expression > Payment »

Il est également disponible dans le modèle d'exemple Enterprise Architect (EAExample) :

- Naviguez dans la fenêtre Navigateur jusqu'à ' Simulation de Modèle > DMN Models > Métier Knowledge Modèle > Métier Knowledge Modèle Literal Expression'

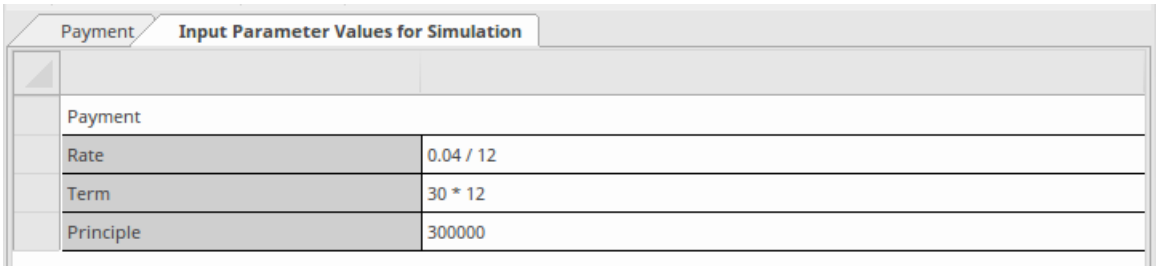
Double-cliquez sur l'élément « Paiement » pour ouvrir le BKM dans la fenêtre Expression DMN.



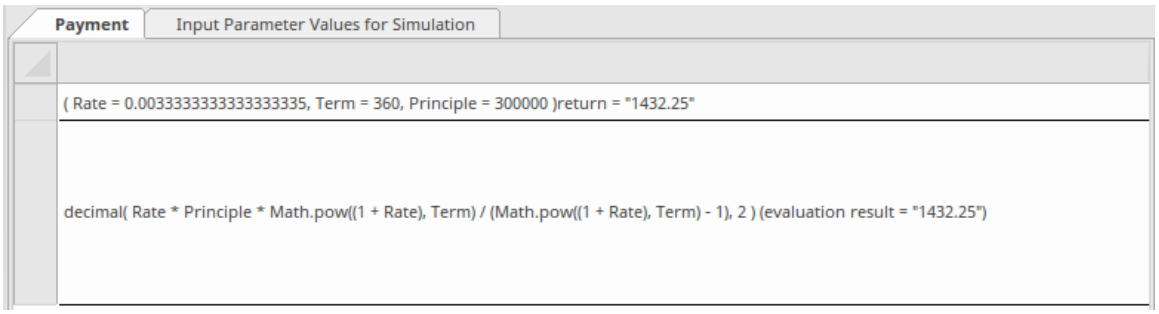
Similairement à un Tableau de Décision , le Modèle de Connaissance Métier implémenté sous forme d'Expression Encadrée peut également être testé.

Prenons l'exemple de l'élément « Paiement ». Ce BKM calculera le remboursement mensuel en fonction du taux d'intérêt, du nombre de termes et du montant principal.

Nous pourrions fournir des valeurs de test telles que celles-ci :



Cliquez sur le bouton  Simulation de la barre d'outils ; ce résultat est obtenu :



Les paramètres d'exécution et les valeurs de retour seront affichés avec un signe égal '=' suivi de la valeur d'exécution. Cette valeur est également affichée sous forme d'étiquette par rapport à l'élément sur son diagramme parent.

Dans cet exemple, étant donné un taux annuel de 4 % sur 30 ans et un capital de 300 000 $, le remboursement mensuel

est de 1 432,25 $.

Note : la Bibliothèque DMN possède déjà une fonction PMT définie ; cet exemple montre principalement comment fonctionne l'expression littérale et comment la tester avec un ensemble d'arguments.

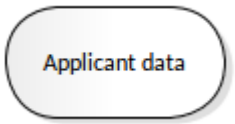
Données d'entrée

Un élément InputData est utilisé pour saisir dans Decisions un ensemble de valeurs provenant de l'extérieur du modèle. Cet ensemble de valeurs est utilisé pour évaluer Decisions. Il dérive son type et un ensemble de valeurs d'un ItemDefinition.

Aperçu

Les éléments InputData sont créés en faisant glisser une icône 

de la boîte à outils sur un diagramme DMN.



Le nom de l'élément InputData doit être unique et ne pas dupliquer le nom d'un autre Décision , InputData, Métier Knowledge Modèle , Décision Service ou Import dans le modèle de décision.

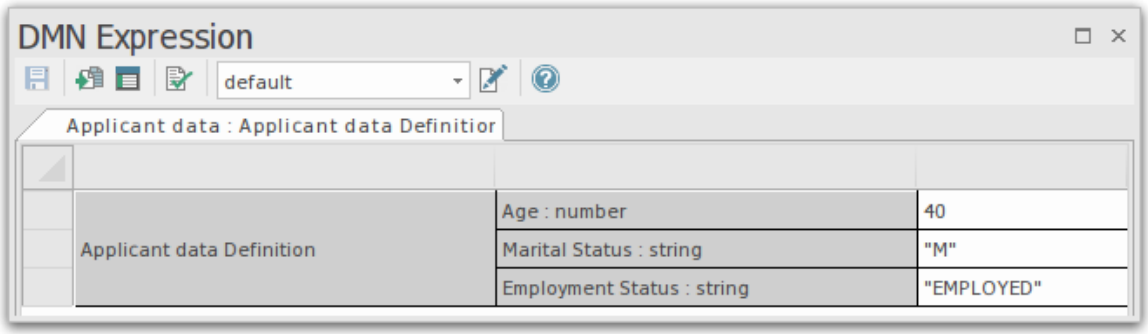
Référencer une définition d'élément

La structure des données, ainsi que les ensembles de valeurs pour un élément InputData, sont définis dans un élément ItemDefinition. Un élément InputData DMN doit être référencé (typé) par un ItemDefinition de l'une des manières suivantes :

- En cliquant sur l'icône  dans la fenêtre Expression DMN de l'élément InputData ou
- Sélectionnez l'élément InputData et appuyez sur Ctrl+L pour sélectionner ItemDefinition dans le dialogue

Propriétés InputData

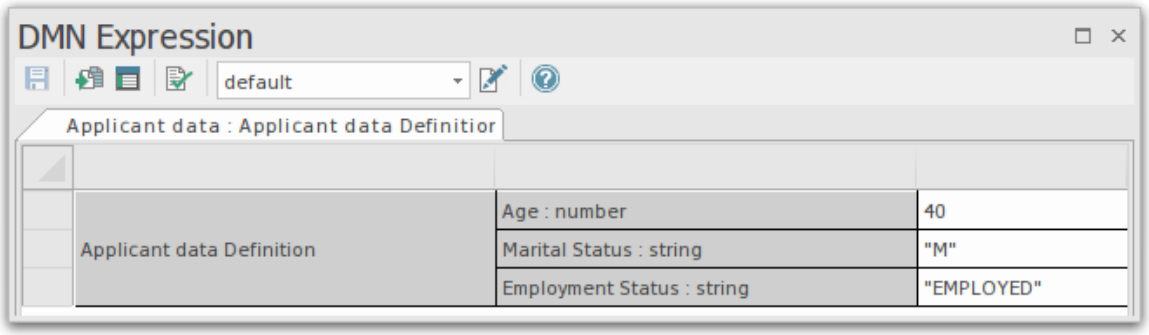
Les propriétés d'un élément InputData sont accessibles via la fenêtre Expression DMN. Double-cliquez sur l'élément InputData pour ouvrir cette fenêtre.



La fenêtre Expression DMN fournit une vue de la structure des données ainsi qu'un accès aux Ensembles de données qui peuvent être utilisés dans les simulations.

Expression DMN de données d'entrée

La fenêtre Expression DMN fournit une vue de la structure de données d'un InputData, des options pour modifier la valeur des Items et un accès aux Ensembles de données qui peuvent être utilisés dans les simulations.



Accéder

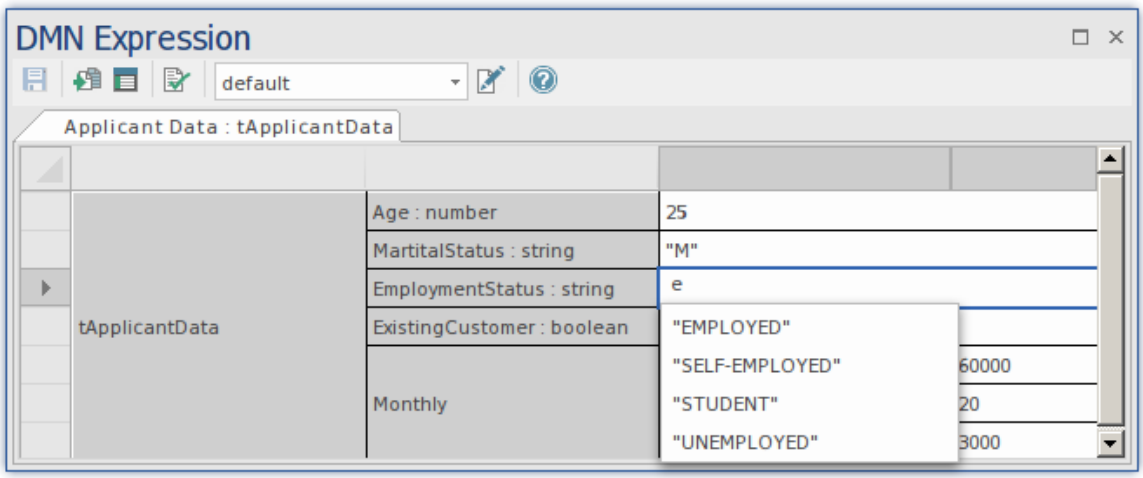
Ruban	Simuler > Analyse Décision > DMN > Expression DMN, puis sélectionner/créer une InputData
Autre	Double-cliquez sur un élément DMN InputData

Options de la barre d'outils

Option	Description
	Enregistre la configuration dans l'élément InputData actuel.
	Ensembles le type d'InputData en sélectionnant une référence à un ItemDefinition.
	Ouvre l'élément ItemDefinition référencé par ce InputData comme définition de type.
	Exécute une validation des données d'entrée. Enterprise Architect effectuera une série de validations pour vous aider à identifier les erreurs dans les données d'entrée.
	Option permettant de sélectionner un ensemble de données tel que défini dans ItemDefinition qui fait référence à cette InputData.
	Ouvre le dialogue permettant d'éditer les ensembles de données pour ces données d'entrée. Chaque InputData peut définir plusieurs ensembles de données. Grâce à cette fonctionnalité , la Simulation DMN peut tester rapidement les résultats d'une décision en choisissant différents ensembles de données.

Complétion automatique

Si InputData possède un champ avec une « Valeur autorisée » définie, le champ peut être renseigné en sélectionnant le champ, en appuyant sur la barre d'espace, puis en sélectionnant une option dans la liste déroulante.



Ensembles de données

Ensembles de données sont définis dans la définition d'élément référencée par l'élément InputData. À l'aide de la liste déroulante de la barre d'outils, vous pouvez sélectionner un ensemble de données dans la définition d'élément. Une fois qu'un ensemble est sélectionné, vous pouvez modifier les valeurs des éléments. Vous pouvez également ajouter de nouveaux Ensembles de données en ouvrant la fenêtre Modifier l'ensemble de données à l'aide de l'icône  .

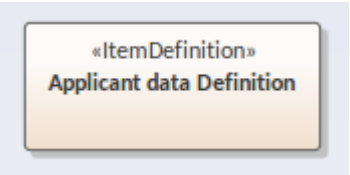
Définition de l'élément

La définition de la structure des éléments de données utilisés dans le modèle est fondamentale pour la création de modèles Décision . Une définition d'élément est utilisée pour définir la structure des données d'entrée et, éventuellement, pour restreindre la plage de valeurs autorisées des données. Les définitions d'élément peuvent aller d'un type unique simple à un type structuré complexe.

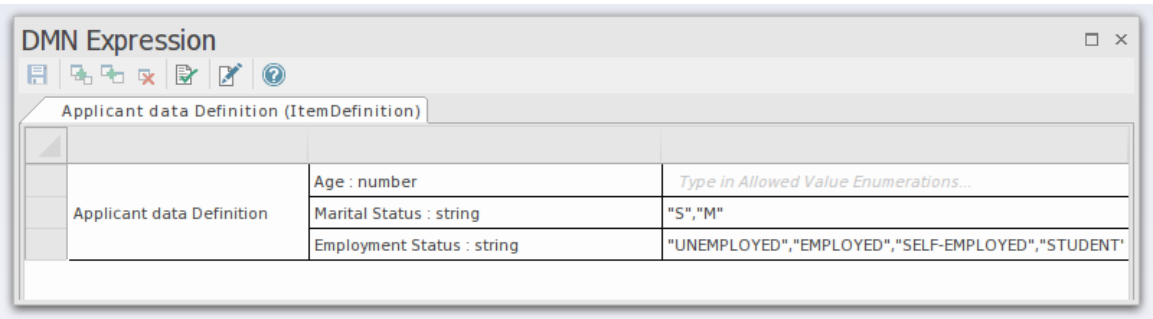
Aperçu

Les éléments ItemDefinition sont créés en faisant glisser une icône 

de la page Boîte à outils DMN sur un diagramme DMN.



Les propriétés principales d'un élément ItemDefinition sont accessibles via la fenêtre Expression DMN.



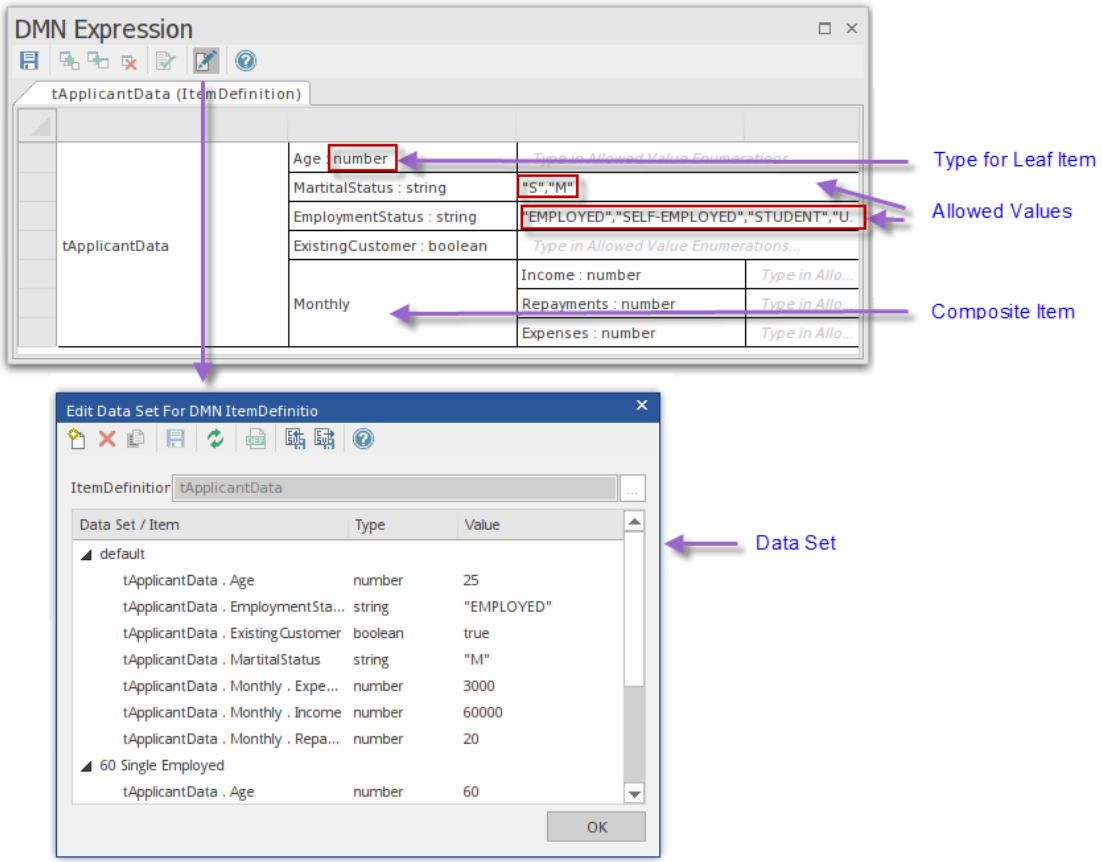
Accéder

Pour ouvrir la fenêtre Expression DMN pour un élément ItemDefinition :

Ruban	Simuler > Décision Analysis > DMN > Expression DMN, puis sélectionnez ou créez une ItemDefinition
Autre	Double-cliquez sur une définition d'élément DMN

Expression et ensemble de données DMN

Cette image est une vue d'ensemble de la fenêtre Expression DMN, montrant un élément de données complexe et la disposition des champs clés utilisés dans la définition des données. Elle comprend une vue d'un ensemble de données défini à l'aide de cette définition d'élément. Un ensemble de données est une « instance » de données conforme à une définition d'élément, qui contient un ensemble de valeurs à utiliser dans la simulation DMN.



Les ItemDefinitions étant des éléments fondamentaux du modèle, il est recommandé de les valider avant de les utiliser dans le modèle. Cela permettra de garantir que tous les problèmes seront résolus dès le début du processus de création d'un modèle complexe.

Pour plus de détails sur la configuration d'ItemDefinitions, consultez ces rubriques d'aide :

- *Définition Item DMN, ensemble de données et données d'entrée*
- *Types de composants*
- *Valeur autorisée de la définition d'élément*
- *Complétion automatique des expressions DMN*
- *Validation de l'expression DMN*

Barre d'outils de définition Item

Ce tableau fournit des descriptions des fonctionnalités accessibles dans la fenêtre Expression DMN lorsqu'un élément ItemDefinition est sélectionné.

Options de la barre d'outils

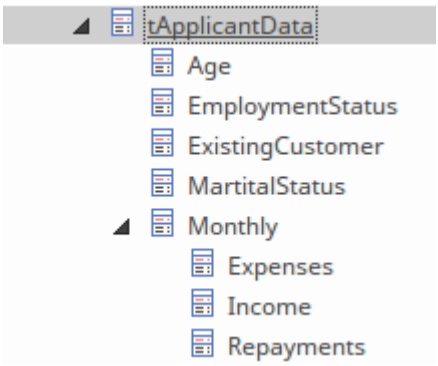
Option	Description
	Enregistre la configuration de l'ItemDefinition actuel.
	Crée un nouveau composant de données en tant qu'enfant du composant sélectionné.
	Crée un nouveau composant de données en tant que frère du composant sélectionné.
	Supprime le composant de données sélectionné.
	Valide l'ItemDefinition ; Enterprise Architect effectuera une série de validations pour vous aider à identifier les erreurs dans l'ItemDefinition.
	Ouvre la dialogue « Modifier l'ensemble de données », dans laquelle vous pouvez créer et modifier des « instances » de l'ItemDefinition à utiliser par les éléments InputData.

Définitions d'éléments et Ensembles de données

Une définition d'élément décrit les types et les structures des éléments de données utilisés dans un modèle Décision . Elle sert de définition de type de données pour les éléments InputData, les éléments Décision et les paramètres Métier Knowledge Modèle . Une définition d'élément peut également définir des ensembles de données qui fournissent des ensembles de valeurs à utiliser dans les simulations DMN. Le basculement entre différents ensembles de données permet d'effectuer des analyses de type « what-if » à l'aide du modèle Décision .

Structure de définition d'élément

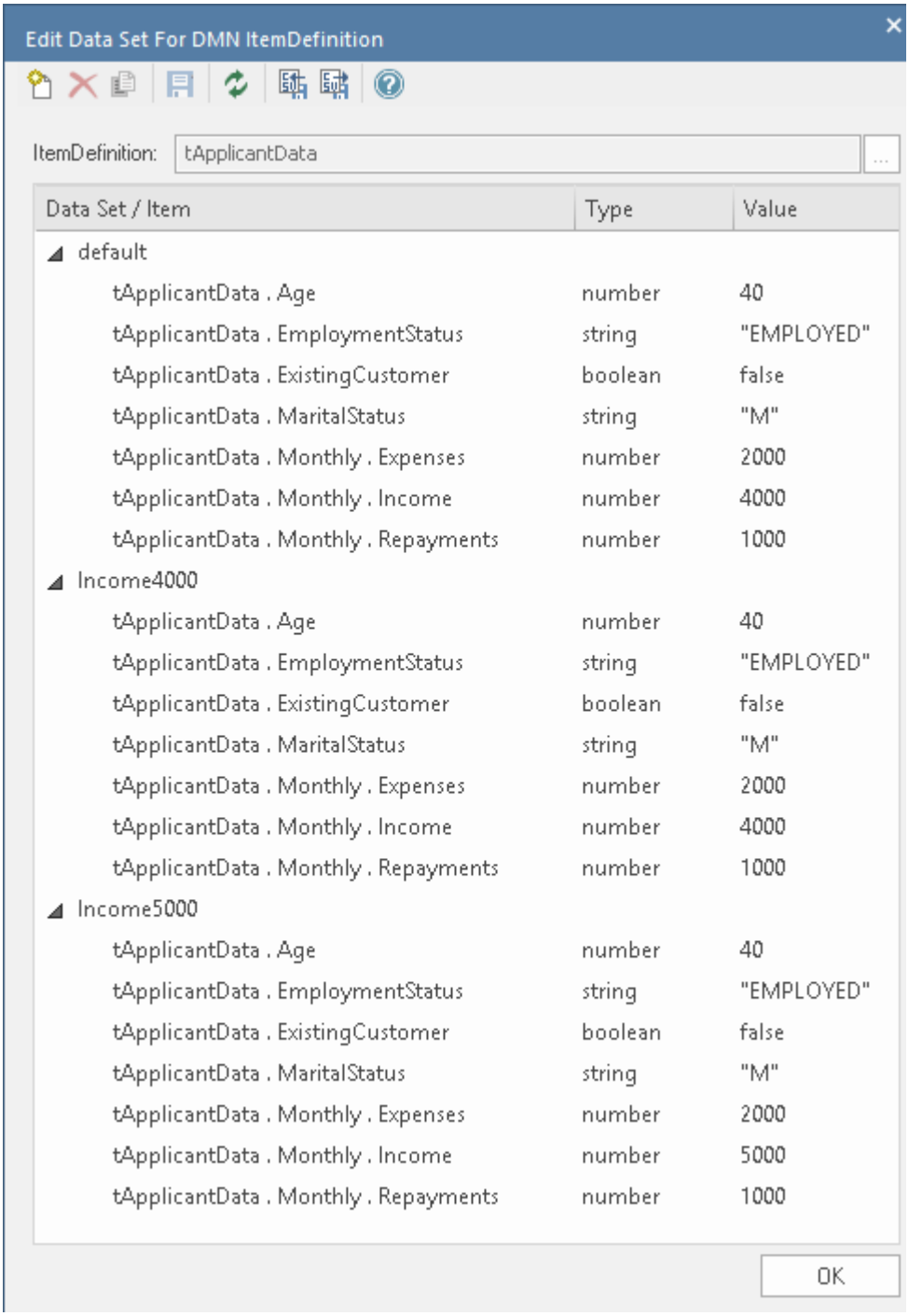
Une définition d'élément complexe est constituée d'éléments imbriqués. Par exemple, *tApplicantData* est structuré comme suit :



L'exemple de définition d'élément tApplicantData est un type composite de cinq éléments enfants. « Mensuel » est composé de trois enfants (Dépenses, Revenus et Remboursements). Les composants Leaf (non composites) auront un type primitif tel qu'un nombre, string ou un booléen.

Ensemble de données

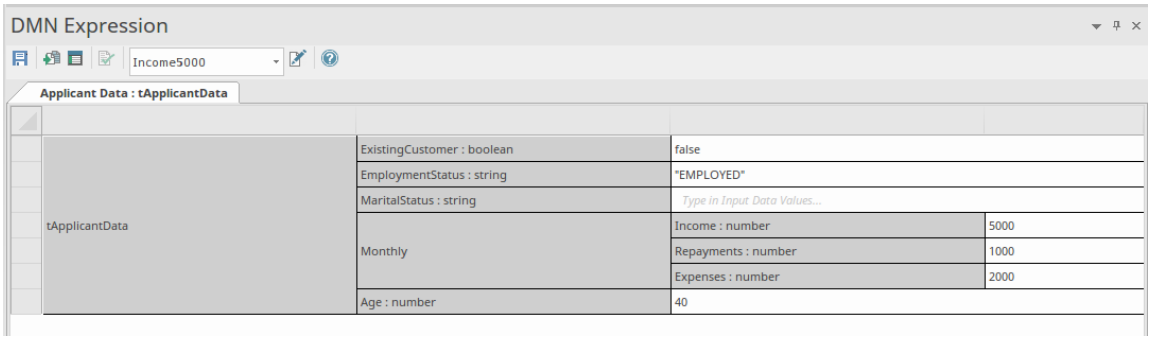
L'ensemble de données de l'ItemDefinition peut être visualisé et modifié à l'aide de l'icône  de la barre d'outils. Avec la dialogue « Modifier l'ensemble de données », vous pouvez ajouter, supprimer et dupliquer les ensembles de données. L'importation et l'exportation d'ensembles de données au format CSV sont également support .



Comme le montre l'exemple, l'ItemDefinition pour *tApplicantData* définit trois ensembles de données :

- défaut
- Revenu4000
- Revenu 5000

Chaque ensemble de données peut être visualisé dans un élément InputData qui est typé selon l'ItemDefinition. Par exemple, l'élément InputData « Données du demandeur » est typé selon l'ItemDefinition « tApplicantData ». La fenêtre Expression DMN pour « Données du demandeur », illustrée ici, affiche les valeurs de données en fonction de l'ensemble de données sélectionné dans la liste déroulante de la barre d'outils de la fenêtre (*Income5000* dans ce cas).



Définition d'une référence à une définition d'élément

Un élément DMN InputData est défini pour être référencé (typé) par un ItemDefinition en utilisant :

- L'icône dans la fenêtre Expression DMN de l'élément InputData ou
- Sélectionnez l'élément InputData et appuyez sur Ctrl+L pour sélectionner ItemDefinition dans le dialogue

Il existe d'autres cas d'utilisation d'ItemDefinitions ; par exemple, lors de la définition du type d'un paramètre d'entrée dans un BKM ou d'un paramètre de sortie dans un Tableau de Décision .

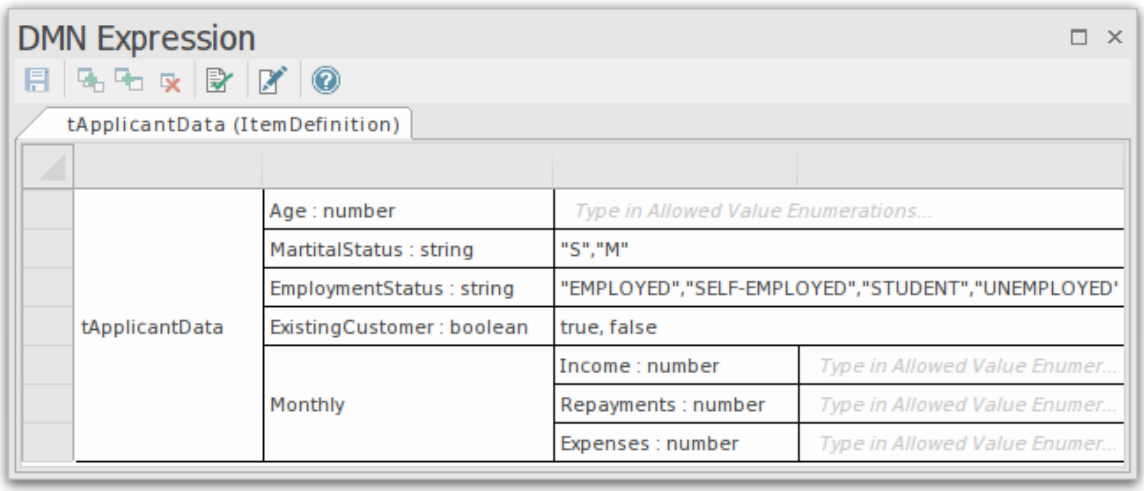
Types de composants

Un élément ItemDefinition peut être défini comme un arbre de composants constitué uniquement d' un des éléments suivants :

- Un type intégré ou
- Une composition d'éléments ItemDefinition

Dans cette arborescence de composants, si un composant est une « feuille » qui n'a pas de composants enfants, il doit être défini comme un type intégré. Si un ItemDefinition a des composants enfants, ce sont ces composants enfants/feuilles qui sont définis comme un type intégré.

Par exemple, *les données du candidat* et les *données mensuelles* sont des compositions, tandis que *l'âge* et *les dépenses* sont des feuilles définies sur un type intégré :



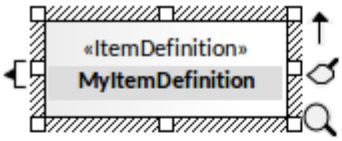
Le langage FEEL possède ces types intégrés :

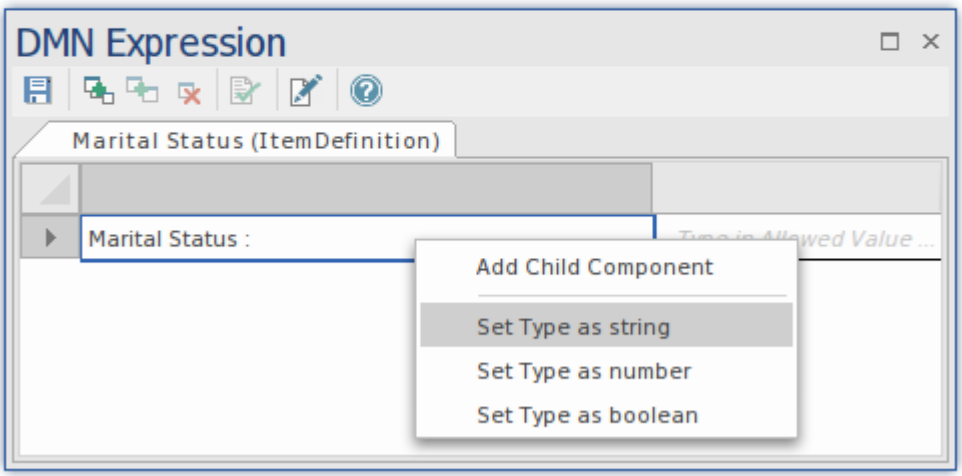
- **nombre**
- **string**
- **booléen**
- jours et durée
- durée des années et des mois
- temps
- date et heure

Note : « nombre », « string » et « booléen » sont pris en charge par Enterprise Architect pour la simulation.

Pour définir un type pour un ItemDefinition « feuille », vous pouvez utiliser l'une des trois méthodes suivantes :

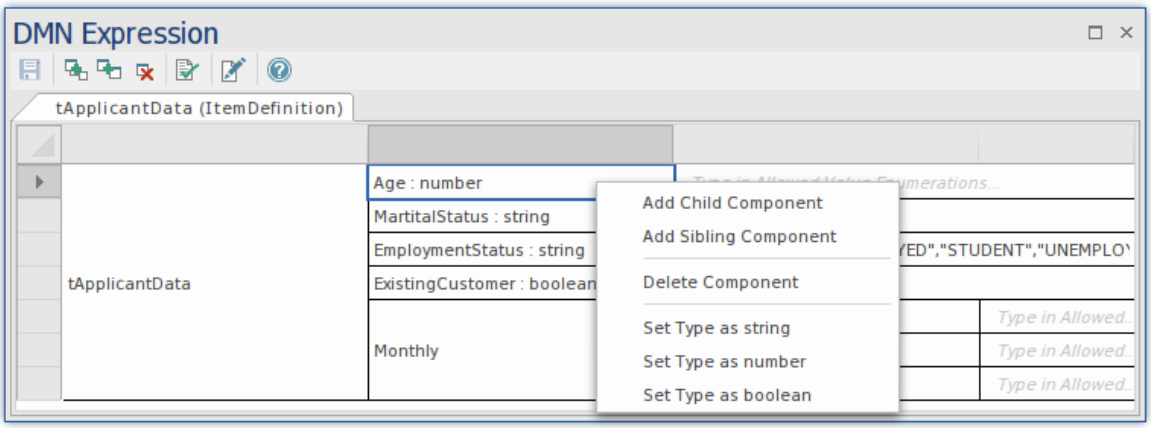
- Sélectionnez l'option de menu contextuel appropriée dans la fenêtre Expression DMN (recommandé)





- Type ': string ', ': boolean' ou ': number' après le nom dans la cellule de la fenêtre Expression DMN
- Type ' string ', 'boolean' ou 'number' comme valeur de l' étiquette ' Type ' dans la fenêtre Propriétés pour l'ItemDefinition

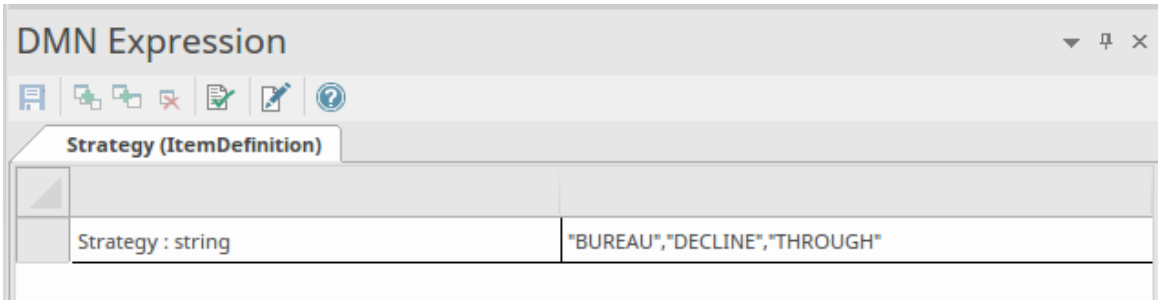
Pour les ItemDefinitions composites, le menu contextuel propose également des options permettant de créer un composant enfant ou frère, ou de supprimer l'élément sélectionné :



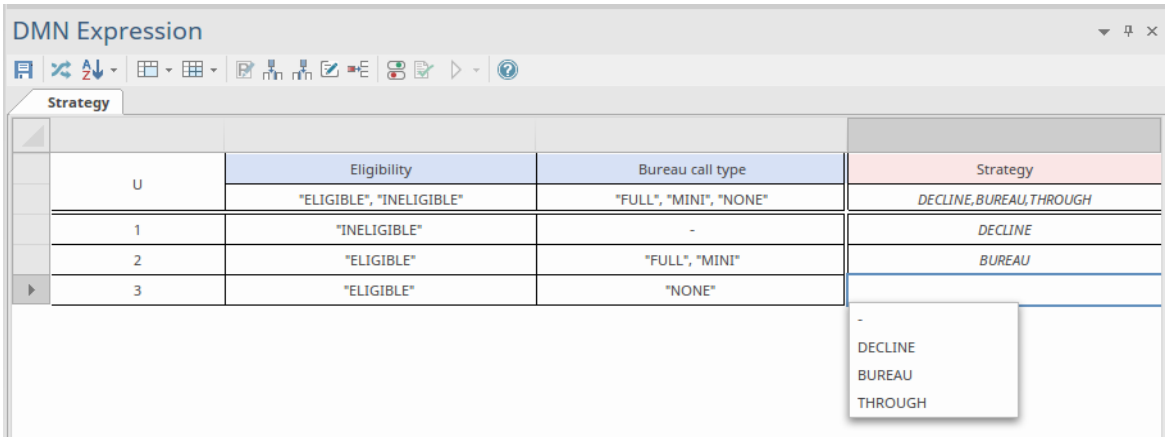
Énumérations de valeurs autorisées

Lors de la définition des entrées de données pour une Décision , il est courant de vouloir restreindre l'ensemble des valeurs autorisées pour une entrée. Par exemple, vous souhaiterez peut-être restreindre les valeurs autorisées pour l'état matrimonial à seulement deux options, « Célibataire » et « Marié ».

Vous pouvez spécifier les valeurs autorisées pour n'importe quel composant feuille d'une stratégie ItemDefinition. Au départ, le champ de données d'un composant feuille contient le texte *Type dans les énumérations de valeurs autorisées* . Il vous suffit de saisir les valeurs autorisées sur ce texte. Par exemple, la *stratégie* ItemDefinition comporte trois valeurs autorisées : BUREAU, DECLINE et THROUGH.



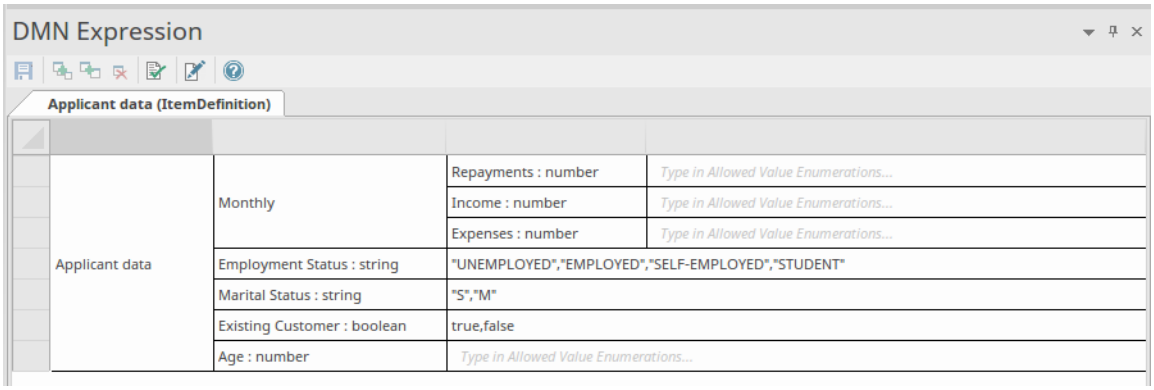
Les énumérations de valeurs autorisées sont également utilisées pour support la saisie semi-automatique. Lors de la spécification de valeurs pour un élément InputData ou un paramètre d'entrée qui référence un ItemDefinition dans lequel des valeurs autorisées ont été définies, l'utilisateur peut simplement appuyer sur la barre d'espace et choisir une valeur dans la liste.



Vous pouvez également effectuer une saisie semi-automatique en tapant la première lettre de l'option que vous souhaitez saisir.

Les paramètres d'entrée et les clauses de sortie de Décision Tableaux support également en charge la spécification des valeurs autorisées. Cela limite les valeurs qui peuvent être utilisées lors de la définition des règles dans le tableau , mais permet également à l'utilisateur de remplir rapidement les règles en appuyant sur la barre d'espace puis en sélectionnant l'élément requis.

Une définition d'élément plus complexe peut inclure un certain nombre d'énumérations de valeurs autorisées ; par exemple :


DMN Expression
Applicant data (ItemDefinition)
Applicant data
Monthly
Repayments : number
Income : number
Expenses : number
Type in Allowed Value Enumerations...
Employment Status : string
"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"
Marital Status : string
"S","M"
Existing Customer : boolean
true,false
Age : number
Type in Allowed Value Enumerations...

Ensembles de données

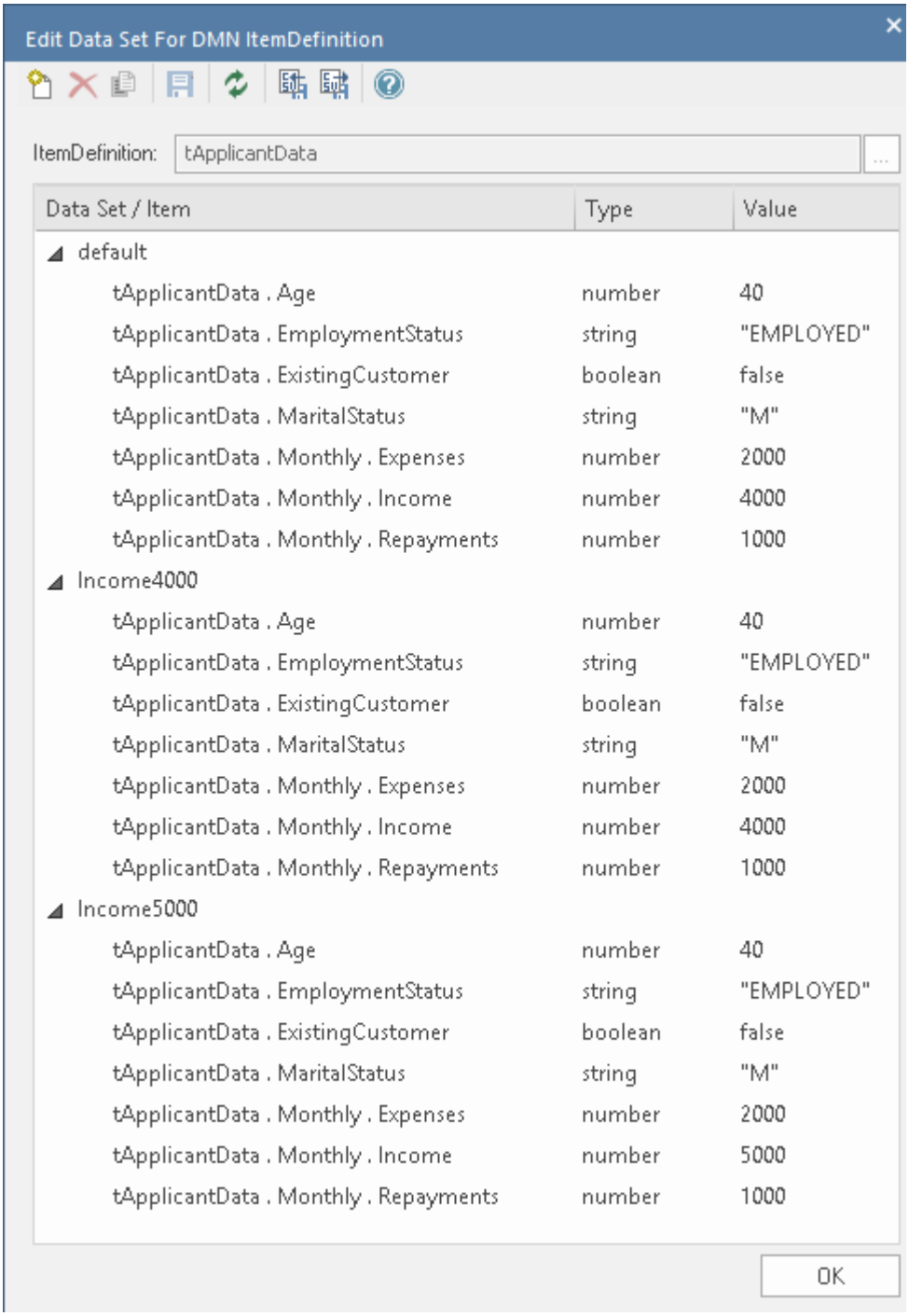
Chaque élément InputData typé par un ItemDefinition possède un ensemble de composants, et plusieurs ensembles de données peuvent être définis pour fournir différents ensembles de valeurs pour ces composants. Avec cette fonctionnalité , un utilisateur effectuant une Simulation DMN peut tester rapidement le résultat d'une décision en choisissant différents ensembles de données. Les ensembles de données sont associés à et basés sur l'ItemDefinition, mais vous pouvez également travailler dessus via l'élément InputData.

Vous pouvez ajouter ou mettre à jour des ensembles de données à l'aide de la dialogue « Modifier l'ensemble de données », que vous appelez à partir de la fenêtre Expression DMN pour l'élément ItemDefinition ou l'élément InputData. Au départ, la dialogue « Modifier l'ensemble de données » affiche un seul ensemble de composants sans valeur, sous le nom d'ensemble « default ». Vous pouvez soit laisser cet ensemble sans valeur, soit fournir des valeurs ; dans les deux cas, vous pouvez l'utiliser comme gabarit à dupliquer pour de nouveaux ensembles de données. Vous ne pouvez pas supprimer l'ensemble de données « default ».

Lorsque vous accédez à un élément InputData dans la fenêtre Expression DMN, les valeurs de l'ensemble de données « par défaut » sont affichées par rapport aux composants de l'élément. Vous pouvez ensuite cliquer sur la flèche déroulante dans la barre d'outils et sélectionner n'importe quel autre ensemble de données dans la liste. Note que si vous laissez l'ensemble de données « par défaut » intact, vous pouvez créer un ensemble de données « par défaut » en double et lui attribuer des valeurs, et cet ensemble « par défaut » fournira les valeurs lorsque vous accéderez initialement à l'élément InputData.

Vous pouvez dupliquer et supprimer tout autre ensemble de données que vous créez, exporter les ensembles de données vers un fichier CSV et les importer à partir d'un fichier CSV.

Note que si vous créez un ensemble de données et n'entrez pas de valeurs, il est supprimé lorsque vous fermez le dialogue .



Accéder

Ruban	Simuler > Analyse Décision > DMN > Expression DMN > cliquez sur l'élément InputData : icône
Autre	Dans un diagramme , double-cliquez sur l'élément DMN InputData : icône .

Options de la barre d'outils

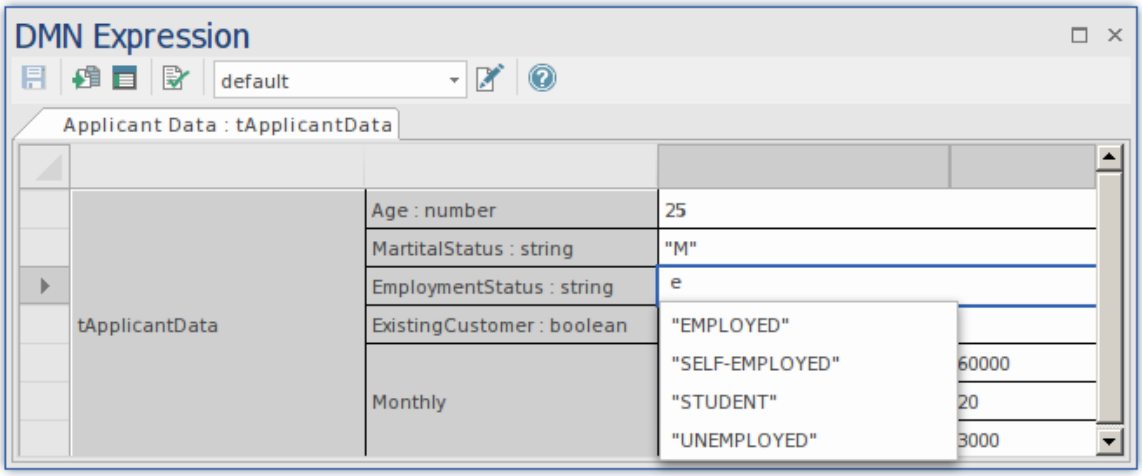
Option	Description
	Cliquez sur ce bouton pour créer un nouvel ensemble de données.
	Cliquez sur ce bouton pour supprimer l'ensemble de données sélectionné.
	Cliquez sur ce bouton pour dupliquer l'ensemble de données sélectionné.
	Cliquez sur ce bouton pour enregistrer les ensembles de données dans InputData.
	Cliquez sur ce bouton pour recharger les ensembles de données pour InputData.
	Cliquez sur ce bouton pour importer des ensembles de données à partir d'un fichier CSV.
	Cliquez sur ce bouton pour exporter les ensembles de données vers un fichier CSV.

Échanger Ensembles de données à l'aide de DataObjects

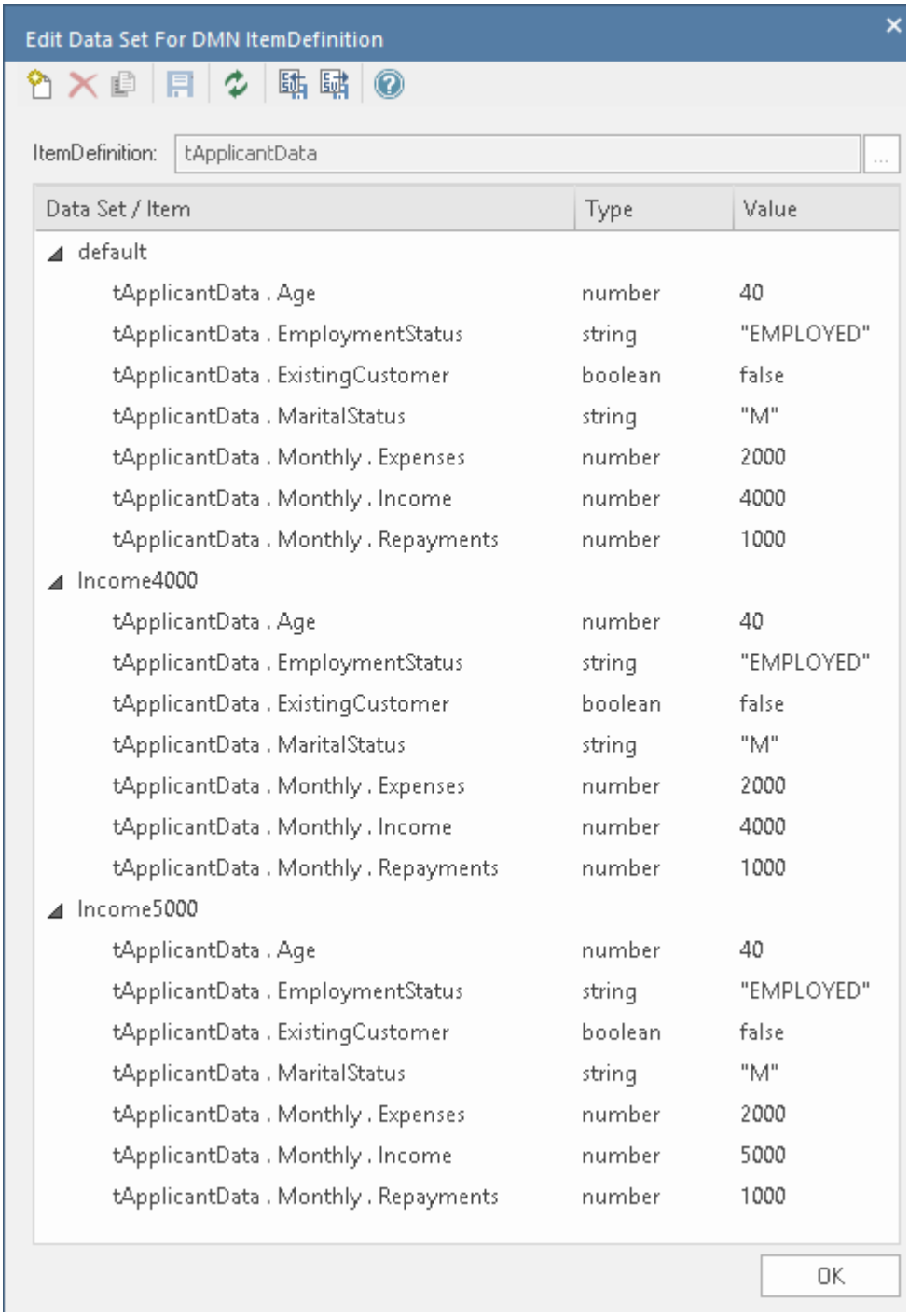
Lorsque vous testez du code généré à partir d'un modèle DMN ou lorsque vous simulez des modèles Business Process Model and Notation (BPMN) qui appellent des modèles DMN, vous avez besoin d'un moyen d'échanger des ensembles de données. Par exemple, dans un appel BPMN d'un modèle DMN, un DataObject BPMN est utilisé pour stocker l'ensemble de variables qui seront transmises au modèle DMN qu'il appelle. Ce DataObject doit être rempli avec des données correspondant à la structure de données de l'objet InputData DMN, prêtes à être transmises à cet object InputData. Ce même DataObject BPMN est utilisé lors du test du code généré à partir d'un modèle DMN.

Cette rubrique décrit le processus de création d'objets de données BPMN à partir Ensembles de données DMN.

Un ensemble de données est stocké dans un élément DMN InputData et est accessible à l'aide de l'icône  dans la fenêtre Expression DMN.



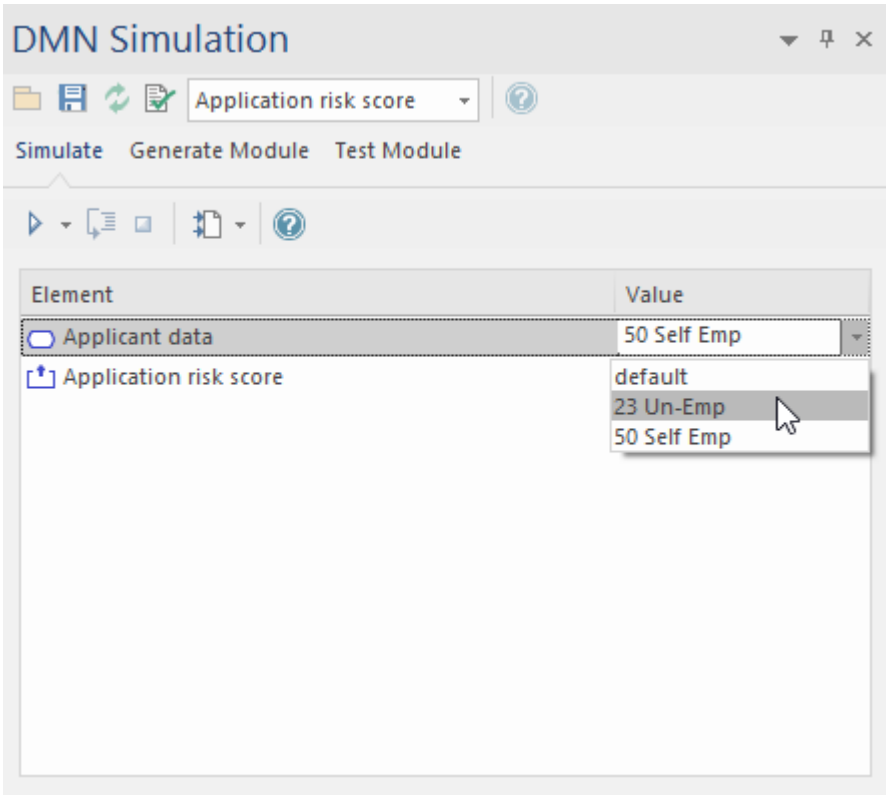
Cela ouvre la dialogue « Modifier l'ensemble de données » d'InputData, qui peut contenir plusieurs ensembles de valeurs :



Il existe deux options pour transférer l'ensemble de données vers un DataObject :

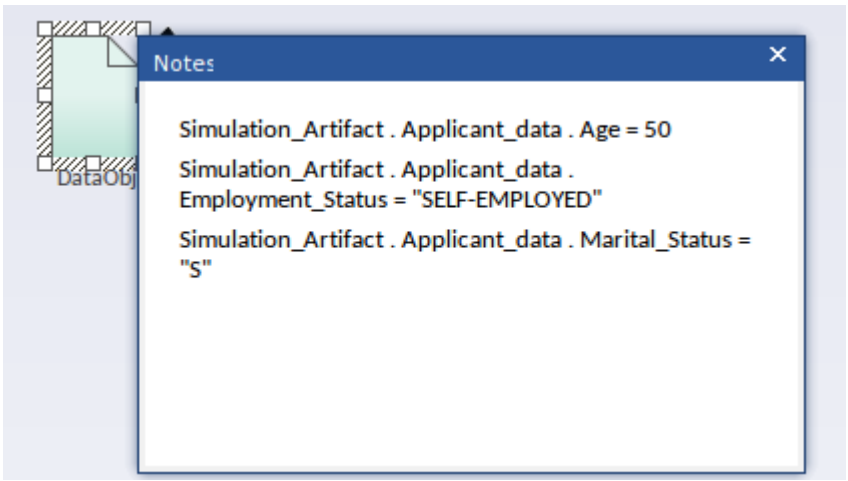
1. Direct

- Créez un DataObject BPMN sous un Paquetage dans la fenêtre Navigateur .
- Ouvrir la fenêtre Simulation DMN



- Sélectionnez un ensemble de données dans la liste déroulante « Valeur »
- Cliquez sur l'icône dans la fenêtre Simulation DMN ; cela ouvre la dialogue « Sélectionner un élément »
- Sélectionnez l'élément BPMN DataObject
- Cliquez sur le bouton OK

L'ensemble de données est maintenant visible dans les Notes de l'objet de données.

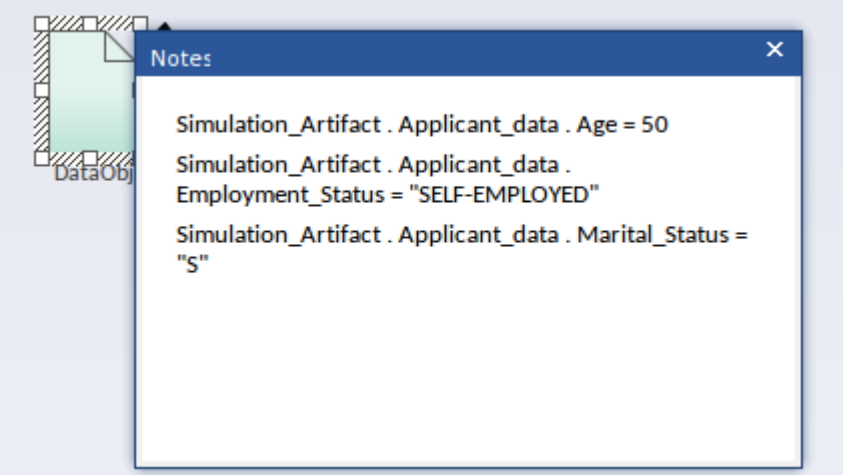


2. Manuel

Pour échanger manuellement cet ensemble de données :

- Ouvrir la fenêtre Expression DMN pour l'élément InputData
- Cliquez sur l'icône Modifier le jeu de données ; cela ouvre la dialogue « Modifier le jeu de données »
- Utilisez l'icône d'exportation CSV pour exporter ces détails vers un fichier

Le texte du fichier CSV peut être ajouté sous forme de texte dans les Notes d'un élément BPMN DataObject.


DataObj
Notes
Simulation_Artifact . Applicant_data . Age = 50
Simulation_Artifact . Applicant_data . Employment_Status = "SELF-EMPLOYED"
Simulation_Artifact . Applicant_data . Marital_Status = "S"

Décision Service

Des parties de ce sujet ont été utilisées textuellement ou sont librement adaptées de la Spécification DMN, qui est disponible à l'adresse suivante : https://www.omg.org/spec/DMN . Ce site contient une description complète du DMN et de ses capacités.

Un service Décision expose une ou plusieurs décisions d'un modèle Décision en tant qu'élément réutilisable, qui peut être invoqué en interne par une autre décision dans le modèle Décision , ou en externe par une tâche dans un modèle de processus BPMN.

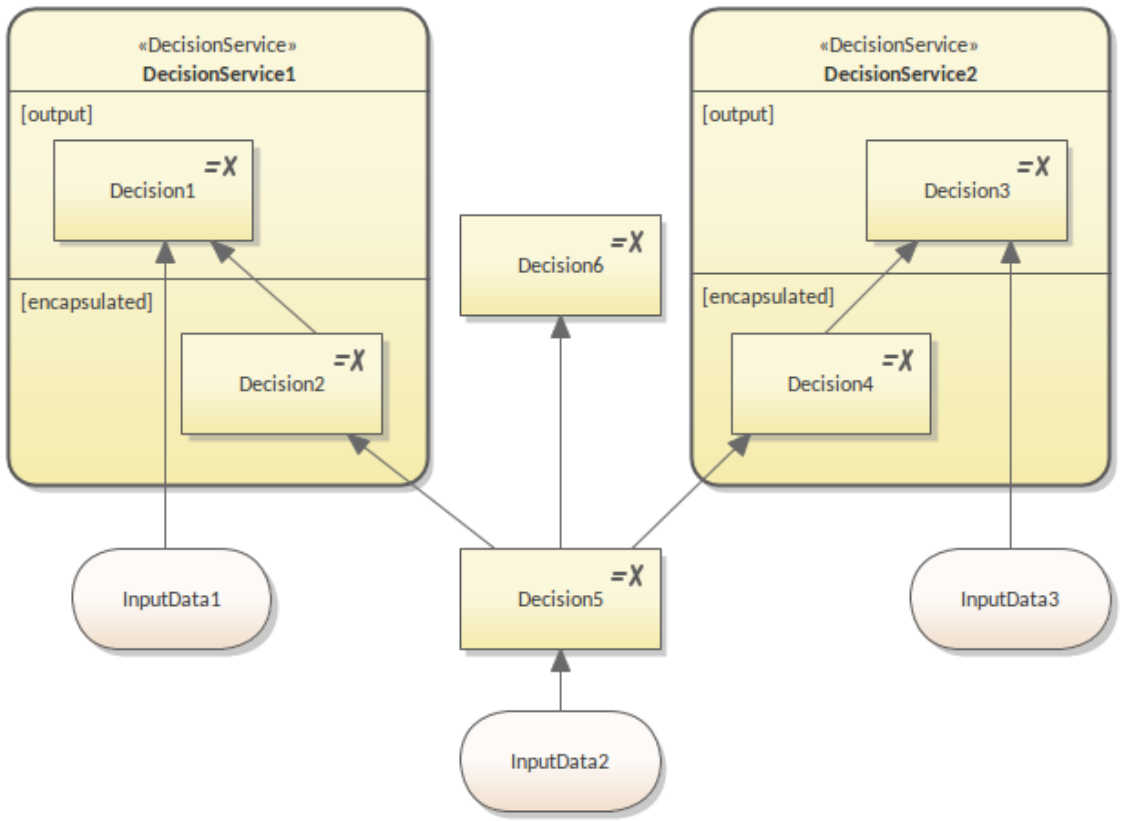
Lorsque le service Décision est appelé avec les données d'entrée et les décisions d'entrée nécessaires, il renvoie les sorties des décisions exposées.

L'interface d'un service Décision

L'interface avec Décision Service est composée de :

- Données d'entrée - instances de toutes les données d'entrée requises par les décisions encapsulées
- Décisions d'entrée - instances des résultats de toutes les décisions d'entrée
- Décisions de sortie - les résultats de l'évaluation (au moins) de toutes les décisions de sortie, en utilisant les décisions d'entrée et les données d'entrée fournies

Lorsque le service Décision est appelé avec les données d'entrée et les décisions d'entrée nécessaires, il renvoie les sorties des décisions exposées.



Cette figure montre un modèle Décision qui comprend six décisions et trois éléments de données d'entrée.

Pour DecisionService1, le :

- La décision de sortie est {Decision1}
- La décision d'entrée est {Decision5}, et

- Les données d'entrée sont {InputData1}

Comme Decision1 requiert Decision2, qui n'est pas fournie au service en entrée, le service doit également encapsuler Decision2 ; par conséquent, les décisions encapsulées sont {Decision1, Decision2}.

Il ressort clairement de la figure que Decision6, Decision3, Decision4 et InputData3 ne sont pas requises par les décisions de DecisionService1. Qu'en est-il de InputData2 ? Bien qu'elles soient requises par Decision5, qui est elle-même requise par DecisionService1, InputData2 n'est en réalité pas requise par DecisionService1. En effet, Decision5 est définie comme la Décision d'entrée. Du point de vue d'un service Décision , nous ignorons toutes les décisions ou données d'entrée requises par une Décision d'entrée.

Pour DecisionService2, le :

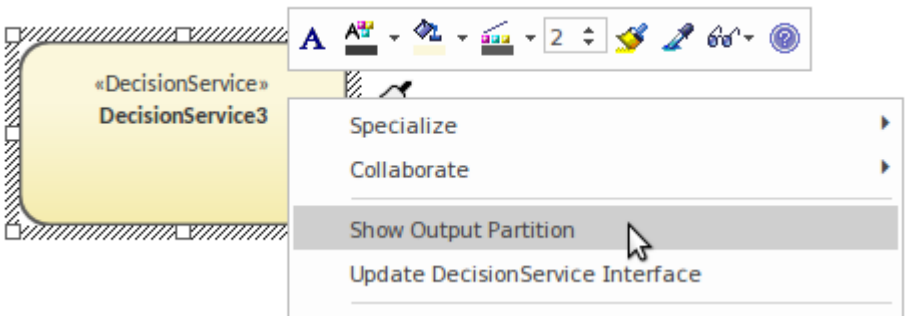
- La décision de sortie est {Decision3}
- La décision d'entrée est {Decision5}, et
- Les données d'entrée sont {InputData3}

Comme Decision3 nécessite Decision4, qui n'est pas fournie au service en entrée, le service doit également encapsuler Decision4 ; par conséquent, les décisions encapsulées sont {Decision3, Decision4}.

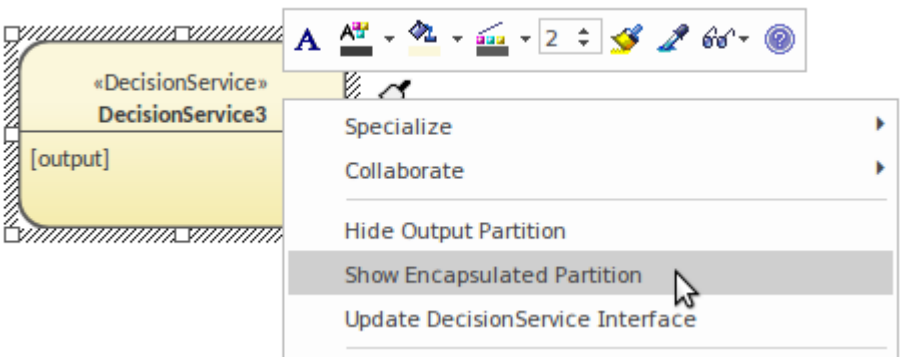
Il est recommandé de créer un diagramme distinct pour chaque service Décision . De cette façon, le diagramme ne contiendra que les éléments d'interface et les décisions encapsulées pour le service Décision ; les éléments qui ne sont pas pertinents n'apparaîtront pas sur le diagramme .

Modélisation d'un service Décision

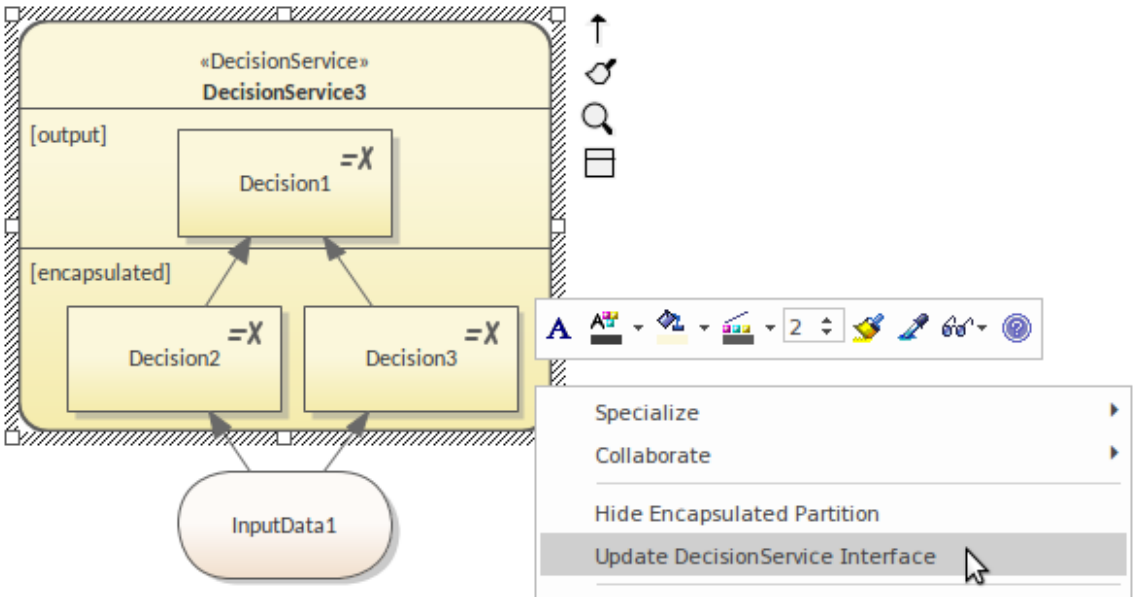
Nous pouvons créer un élément Décision Service à partir des pages DMN de la boîte à outils Diagramme et basculer les partitions [sortie] et [encapsulées] à partir du menu contextuel.



Vous ne pouvez afficher qu'une partition [encapsulée] lorsqu'une partition [de sortie] est affichée.



Une fois les décisions et les données d'entrée placées dans la ou les partitions correctes, vous devez exécuter la commande « Mettre à jour l'interface DecisionService » depuis le menu contextuel pour mettre à jour le modèle.



Important : pour que la simulation DMN fonctionne correctement, veuillez mettre à jour l'interface Décision Service chaque fois que vous :

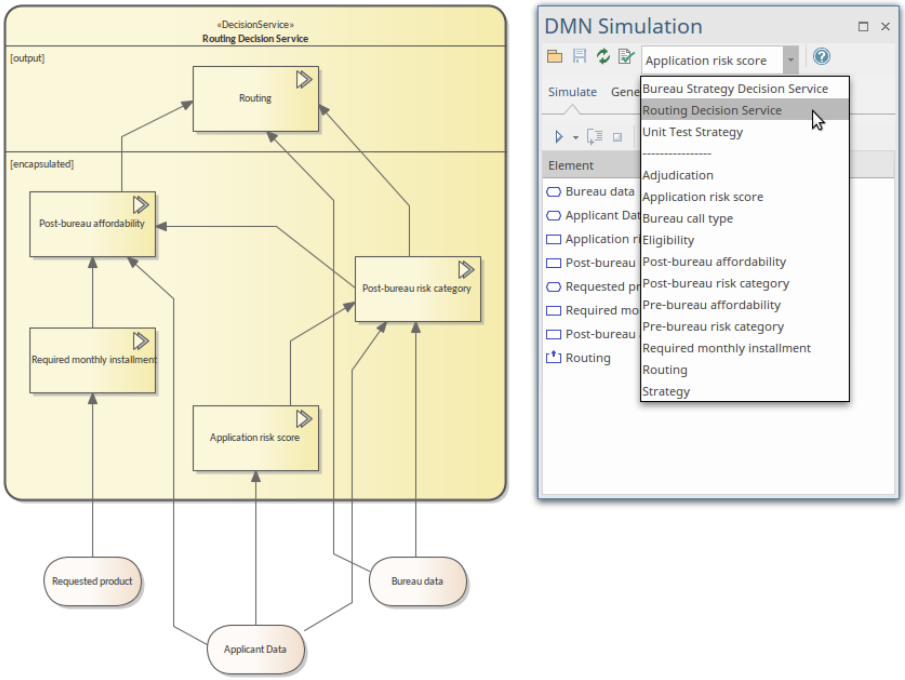
- Afficher/masquer la ou les partitions du service de décision
- Ajouter une décision au service de décision
- Supprimer une décision du service de décision
- Déplacer une décision entre les partitions
- Ajouter/supprimer des entrées du service Décision : données d'entrée ou décisions d'entrée

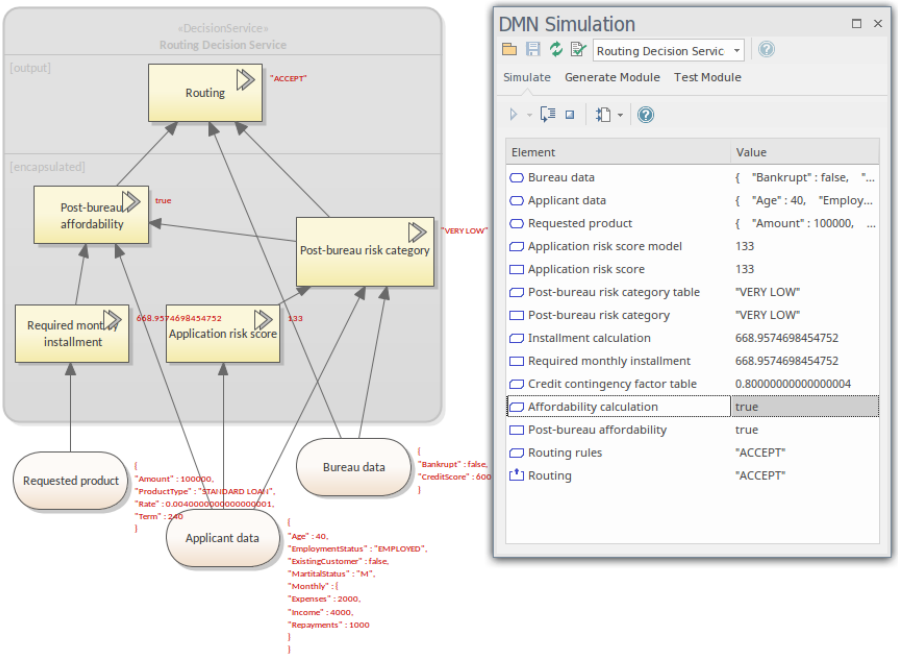
Simuler un service Décision

Il est possible d'effectuer une simulation de modèle sur un Décision Service.

Simulation de service Décision

Pour effectuer une simulation de modèle sur Décision Service, procédez comme suit :

Étape	Description
1	Faites glisser un élément d'artefact de configuration Simulation sur un diagramme à partir de la page « Composants DMN » de la boîte à outils et double-cliquez dessus pour l'ouvrir dans la fenêtre Simulation DMN.  Par défaut, tous les éléments Décision Service et chaque décision individuelle sont répertoriés pour sélection dans le champ déroulant de le dialogue .
2	Sélectionnez un élément Décision Service sur lequel exécuter la simulation. Dans l'exemple, nous avons choisi « Routing Décision Service », donc trois éléments de données d'entrée et cinq décisions encapsulées (dont une décision de sortie) sont chargés dans la liste de simulation. Important : cette liste est établie à partir des données internes du Décision Service ; veillez à exécuter la commande « Mettre à jour l'interface DecisionService » du menu contextuel à chaque modification du diagramme du modèle Décision Service. Rechargez le Décision Modèle en cliquant sur l'icône « Actualiser » (troisième à partir de la gauche) dans la barre d'outils de la fenêtre Simulation DMN.
3	Les données d'entrée et les décisions sont exécutées dans l'ordre correct. Par exemple, « Score de risque d'application » sera exécuté avant « Catégorie de risque post-bureau », « Abordabilité du bureau de poste » et « Acheminement ». Pour chaque élément de données d'entrée, cliquez sur la flèche déroulante dans le champ « Valeur » et sélectionnez les Ensembles de données pour fournir les valeurs des données d'entrée. Validez les données d'entrée et les décisions et apportez les corrections nécessaires

	à l'aide de la fenêtre Expression DMN. Dans la fenêtre Simulation DMN, cliquez sur l'icône Enregistrer et sur le bouton  de la barre d'outils.
4	Le résultat de l'exécution est affiché à la fois dans la liste et sur le diagramme . Vous pouvez également cliquer sur l'icône « Pas à pas » dans la barre d'outils pour déboguer le modèle DMN.  Une bonne pratique consiste à garder la fenêtre Expression DMN ouverte pendant le débogage. L'état de l'expression au moment exécuter (tel que Tableau de Décision , Contexte encadré, Expression littérale ou Invocation) affichera les détails de la logique encapsulée par la Décision ou Métier Knowledge Modèle invoqué.

Module de génération et Test de code

Une fois qu'un modèle Décision est créé et simulé, vous pouvez générer un module DMN en Java, JavaScript , C++ ou C# . Ce module DMN peut être utilisé avec le Moteur d'Exécution Enterprise Architect BPSim, Statemachine Exécutable ou votre propre projet.

Enterprise Architect fournit également une page « Module Test », qui est un préprocessus pour l'intégration de DMN avec BPMN. Le concept consiste à fournir un ou plusieurs éléments BPMN2.0::DataObject, puis à tester si une Décision cible spécifiée peut être évaluée correctement ou non.

Si une erreur ou une exception se produit, vous pouvez créer un script d'analyse pour déboguer le code du module DMN et du client Test .

Après ce processus « Module Test », Enterprise Architect garantit que les éléments BPMN2.0::DataObject fonctionneront bien avec le module DMN.

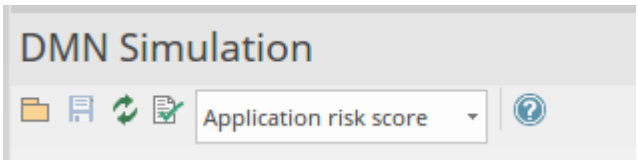
Vous configurez ensuite BPSim en chargeant des DataObjects et en affectant des Décisions du module DMN aux Propriétés BPSim, qui seront ensuite utilisées comme conditions sur les Flux Séquence sortant d'une Passerelle .

Accéder

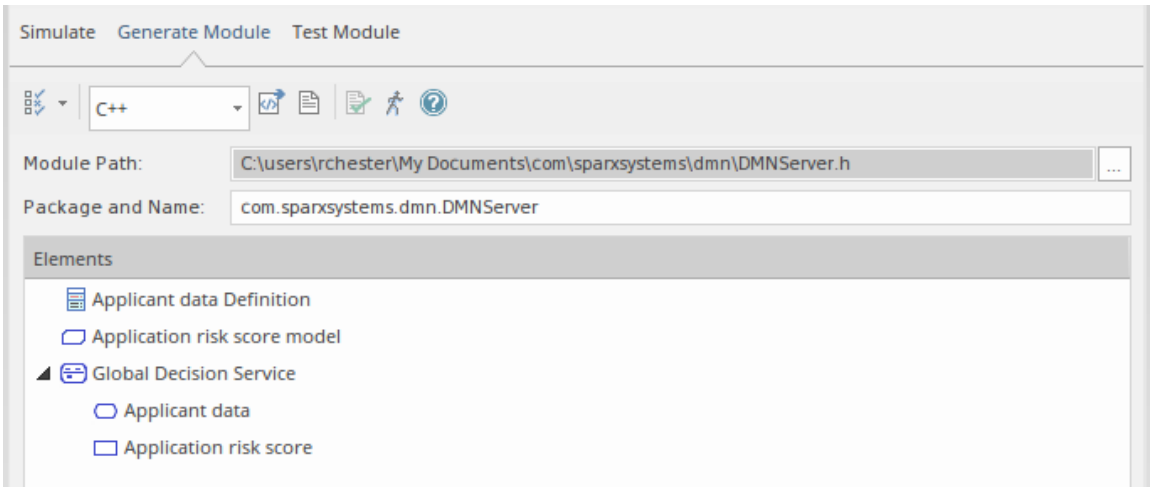
Ruban	Simuler > Décision Analysis > DMN > Open DMN Simulation > Générer Module

Module DMN : Génération de code

Dans la fenêtre Simulation DMN, sélectionnez la structure DMN à partir de laquelle vous souhaitez générer le module, dans le champ de saisie de données de la barre d'outils.



Cliquez sur l'onglet « Générer un module », puis appuyez sur Ctrl+clic sur les noms des éléments DMN que vous souhaitez générer sur le serveur.



Dans le champ de saisie de données de la barre d'outils de l'onglet, sélectionnez la langue dans laquelle générer, puis dans le champ « Chemin du module », cliquez sur l'icône ... et accédez à l'emplacement du chemin dans lequel générer

le module (note : pour Java, le chemin doit correspondre à la structure Paquetage).

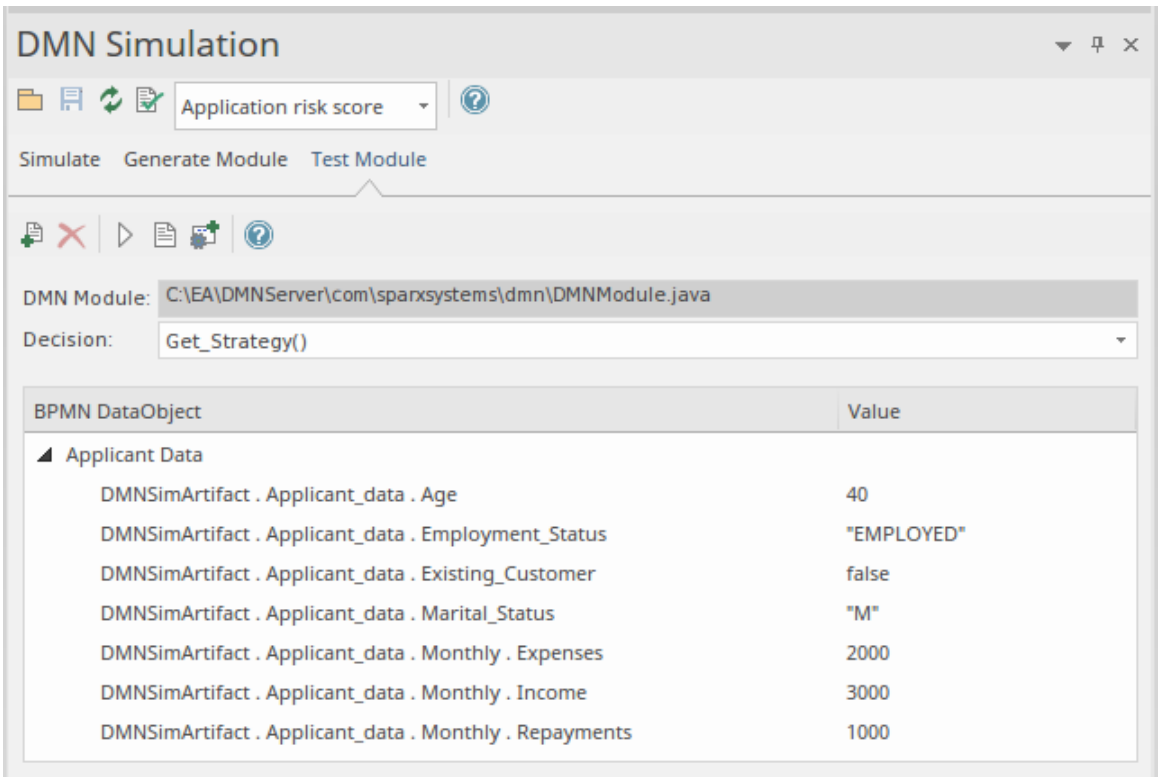
Cliquez sur le bouton Générer ( ).

Une fois la génération terminée, cliquez sur le bouton pour ouvrir l'onglet « Module Test » du module généré.

Serveur DMN : module Test

Lorsque vous utilisez le bouton pour sélectionner l'onglet ' Module Test ', le champ ' Module DMN ' sera automatiquement rempli avec le chemin du serveur DMN généré du module que vous avez généré le plus récemment dans l'onglet ' Générer un module '. Si nécessaire, dans le champ ' Décision ', cliquez sur la flèche déroulante et sélectionnez la Décision requise.

Cliquez sur le bouton Ajouter un objet de données BPMN ( ) dans la barre d'outils et sélectionnez un ou plusieurs (Ctrl+clic) objets de données BPMN2.0 à ajouter à la liste dans le panneau principal.



Cliquez maintenant sur le bouton Exécuter de la barre d'outils. Dans la fenêtre Sortie système, ce message indique que le serveur DMN et l'objet de données BPMN2.0 peuvent fonctionner correctement ensemble pour évaluer la décision sélectionnée :

Exécution du client Test pour le serveur DMN...

dmnServer.Application_risk_score : 133,0

Résultat : 133.0

La course s'est terminée avec succès.

S'il y a des erreurs, créez un script Analyzer en cliquant sur le bouton de la barre d'outils et utilisez le script pour résoudre le problème.

Important : cette étape « Module Test » est recommandée avant d'intégrer DMNServer.java au Moteur d'Exécution Enterprise Architect BPSim. Consultez la rubrique d'aide *Intégrer un module DMN dans BPSim for Simulation* .

Génération de code et connexion à BPMN

- Générer le serveur DMN en Java, JavaScript , C++ ou C#
- Exécuter / Déboguer les tests de la version Java du serveur DMN
- Connecter le serveur DMN à l' Enterprise Architect BPSim Moteur d'Exécution

Erreurs courantes et solutions

- Types de variables : comme les modèles DMN utilisent le langage FEEL (Simulate with JavaScript), la saisie de variables n'est pas obligatoire ; cependant, lors de la génération de code dans des langages compilés, vous devez saisir une variable - il existe des options de menu contextuel et des valeurs étiquette pour définir le type d'une variable
- Étant donné qu'une expression DMN autorise les espaces, afin de clarifier les données d'entrée composites, il doit y avoir un espace avant et après le « . » dans l'expression ; par exemple, « Données du candidat. Âge » est valide, tandis que « Données du candidat. Âge » n'est pas valide
  Note que lorsque vous utilisez la fonctionnalité de saisie semi-automatique, ce problème ne se produira pas
- L'exécution de la validation vous aidera à localiser la plupart des problèmes modélisation ; effectuez-le avant la simulation et la génération de code

Notes

- La compilation avec Java nécessite un accès complet en lecture-écriture au répertoire cible ; la compilation échouera si le chemin du module est défini uniquement sur « C : » ou « C:\Program Files (x86) »

Intégrer dans BPSim pour Simulation

La force de DMN est sa capacité à décrire les exigences métier à travers le diagramme des exigences Décision et à encapsuler la logique complexe dans des expressions polyvalentes telles que le Tableau de Décision et le Contexte encadré.

De même, la force de BPMN réside dans sa capacité à décrire des processus métier avec une Flux séquence de tâches et d'événements, ou à décrire des collaborations de processus avec des flux de messages.

Le diagramme Décision Exigences forme un pont entre les modèles Processus Métier et les modèles logiques de décision :

- Les modèles Processus Métier définissent les tâches au sein des processus métier, où une prise de décision est requise
- diagrammes Décision Exigences définissent les décisions à prendre dans ces tâches, leurs interrelations et leurs exigences en matière de logique de décision
- La logique Décision définit les décisions requises de manière suffisamment détaillée pour permettre la validation et/ou l'automatisation

DMN fournit un modèle Décision complet qui complète un modèle Processus Métier en spécifiant en détail les prises de décision réalisées dans les tâches du processus.

Les deux exemples présentés dans cette rubrique sont accessibles à partir de :

- EAExample Modèle | Simulation de Modèle | Modèles BPSim
- Point de vue | Modélisation Métier | BPSim | Études de cas BPSim

Les expressions BPSim utilisent un modèle DMN de deux manières :

- Le service Décision de DMN - démontré par le processus de demande de prêt
- Modèle de connaissances commerciales de DMN - démontré par le calcul des coûts de livraison

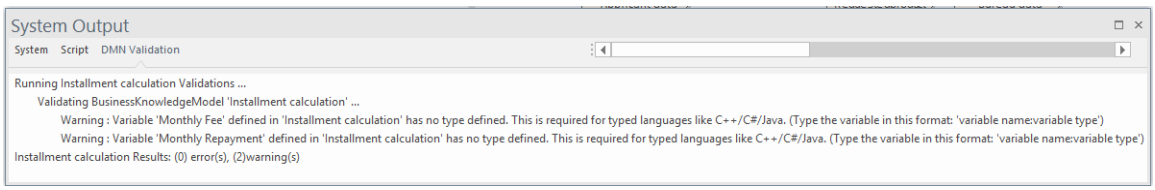
Le processus d'intégration d'un modèle DMN avec un modèle BPSim comprend :

- Validation Modèle DMN, Simulation , génération de code et Tester sur le module généré
- Configurer une dépendance d'utilisation de l'artefact BPSim vers l'artefact DMN
- Générer ou mettre à jour le DataObject BPMN à partir du DataSet DMN
- Créez des paramètres de propriété dans BPSim à utiliser sur les tâches et les flux Séquence sortant des passerelles
- Lier l'interface DMN aux paramètres de propriété BPSim

Validation Modèle DMN pour les langages compilés tels que Java

Lorsque vous créez un modèle DMN et le simulez dans Enterprise Architect , le code pilotant la simulation est JavaScript ; cela signifie que les variables n'ont pas besoin d'être explicitement typées (le type de variable est déduit de la valeur qui lui est attribuée).

Cependant, pour les langages tels que C++, C# et Java, le compilateur signalera une erreur indiquant qu'une variable n'a pas de type.



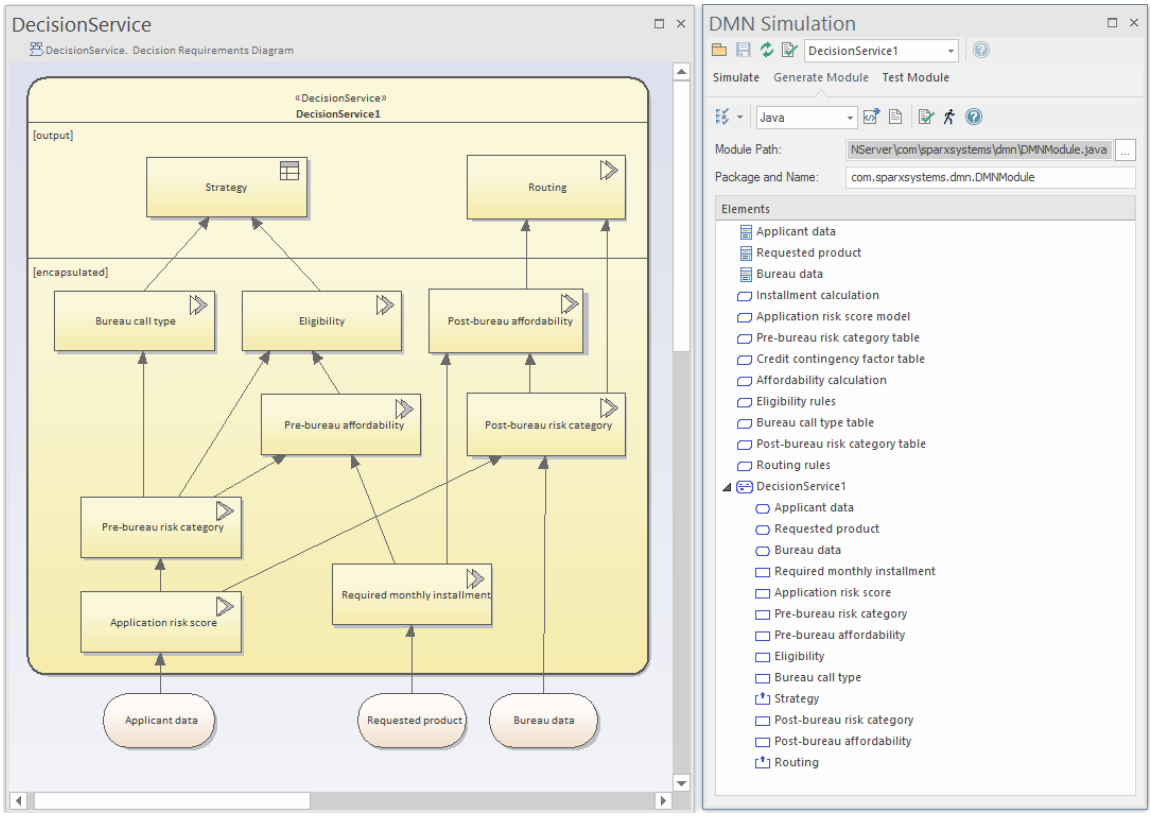
Pour générer ces langages, vous devez exécuter une validation sur le modèle et utiliser les résultats pour trouver les variables qui nécessitent un type défini. Par exemple :

- Paramètre Métier Knowledge Modèle - sélectionnez l'élément BKM à afficher dans la fenêtre Expression DMN, cliquez sur le deuxième bouton pour ouvrir la dialogue « Paramètre », spécifiez un type pour le paramètre

- Type Décision - sélectionnez l'élément Décision , ouvrez la fenêtre Propriétés , pour la propriété 'variableType' sélectionnez dans le champ 'Valeur'
- Clauses Tableau de Décision - sur la clause d'entrée/sortie Tableau de Décision , cliquez-droit pour afficher le menu contextuel et choisir le type
- Variables de contexte encadrées - reportez-vous à la rubrique d'aide sur le Boxed Context

Génération de code DMN en Java

Après avoir utilisé la validation pour résoudre les problèmes de type de variable, nous pouvons passer à la page « Générer un module » dans la fenêtre Simulation DMN.



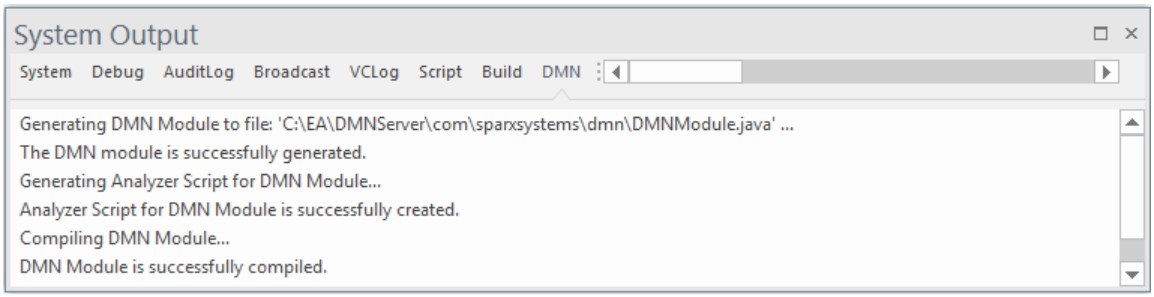
- Sélectionnez *DecisionService1* dans le champ de saisie de données de la barre d'outils supérieure ; tous les éléments impliqués dans *DecisionService1* seront maintenant inclus dans la liste
- La Définition Item et Modèle de Connaissance Métier sont des éléments globaux
- Les données d'entrée et les décisions sont encapsulées dans l'élément DecisionService
- Les langages supportés sont C++, C# , Java et JavaScript ; note que pour JavaScript le fichier .js généré est le même que le script de simulation (onglet 'Simuler' | menu déroulant du bouton Exécuter | Générer un nouveau script (fenêtre Scriptant)) sauf que les codes liés à la simulation sont omis
- Pour Java, la valeur « Module Path » doit correspondre à la structure Paquetage ; dans cet exemple, le DMNModule.java doit être généré dans un répertoire pour former un chemin de fichier qui se termine par « \com\sparxsystems\dmn\DMNModule.java » - vous devez créer manuellement les structures de répertoire pour maintenant

Cliquez sur le bouton Générer du code ( ) dans la barre d'outils. Cet exemple utilisera Java ; cependant, C++ et C# sont identiques. Ces actions sont effectuées :

- Le fichier .java est généré dans le chemin spécifié
- Un script d'analyse (script de construction) pour cet artefact est créé

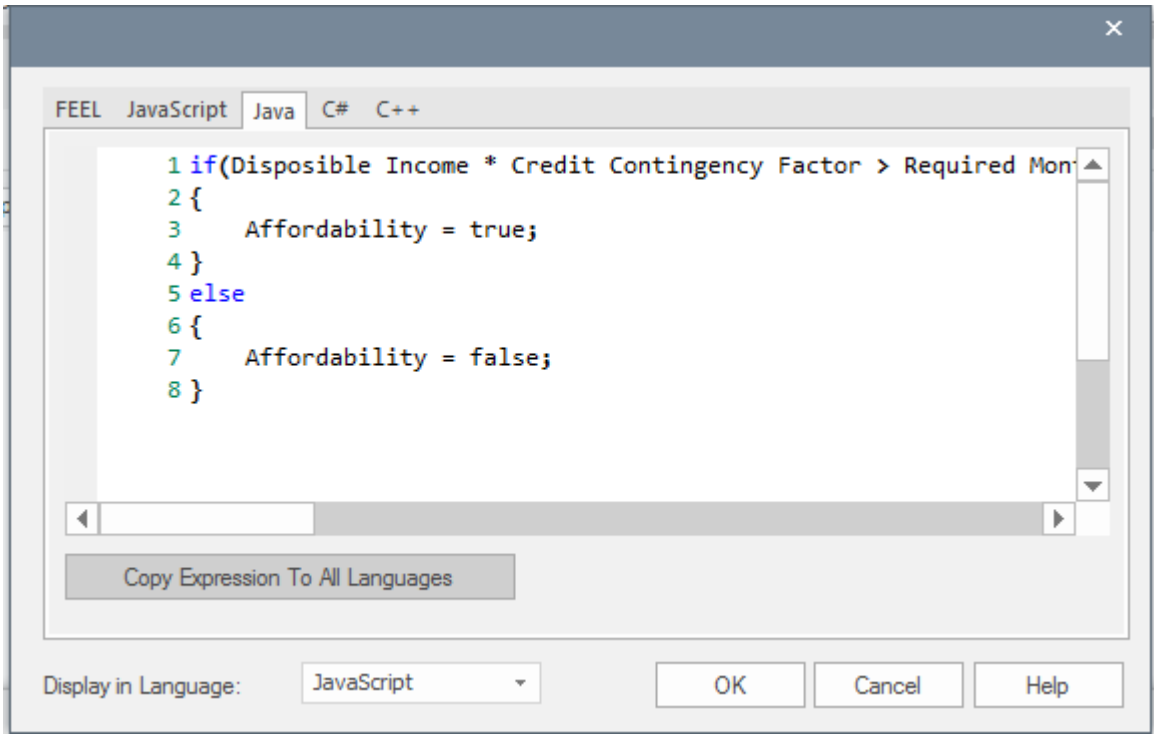
- Le script de construction de ce script d'analyse est exécuté
- Les messages de progression sont signalés dans la fenêtre de sortie du système

Si le modèle est valide, ce processus renverra le message :



S'il y a des erreurs de compilation, vous pouvez ouvrir le fichier .java généré en cliquant sur le bouton  à côté du bouton  dans la barre d'outils, résoudre manuellement le problème et compiler avec le script généré jusqu'à ce que vous réussissiez.

Une raison courante d'un échec de compilation est que les langages peuvent avoir des grammaires différentes pour une expression. Vous devrez peut-être fournir une valeur pour un langage afin de remplacer la valeur par défaut (cliquez-droit sur une expression littérale DMN | Modifier l'expression).

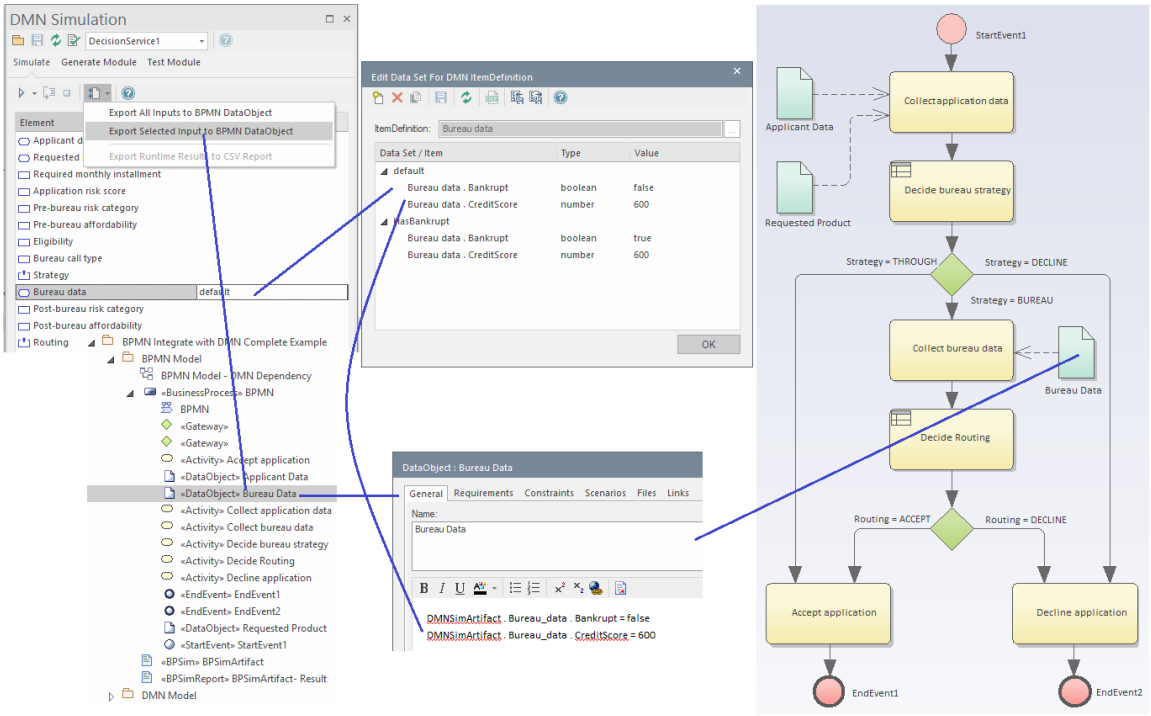


Tester les modules DMN avant utilisation externe

Après avoir généré le modèle en code Java et l'avoir compilé avec succès, nous souhaitons maintenant :

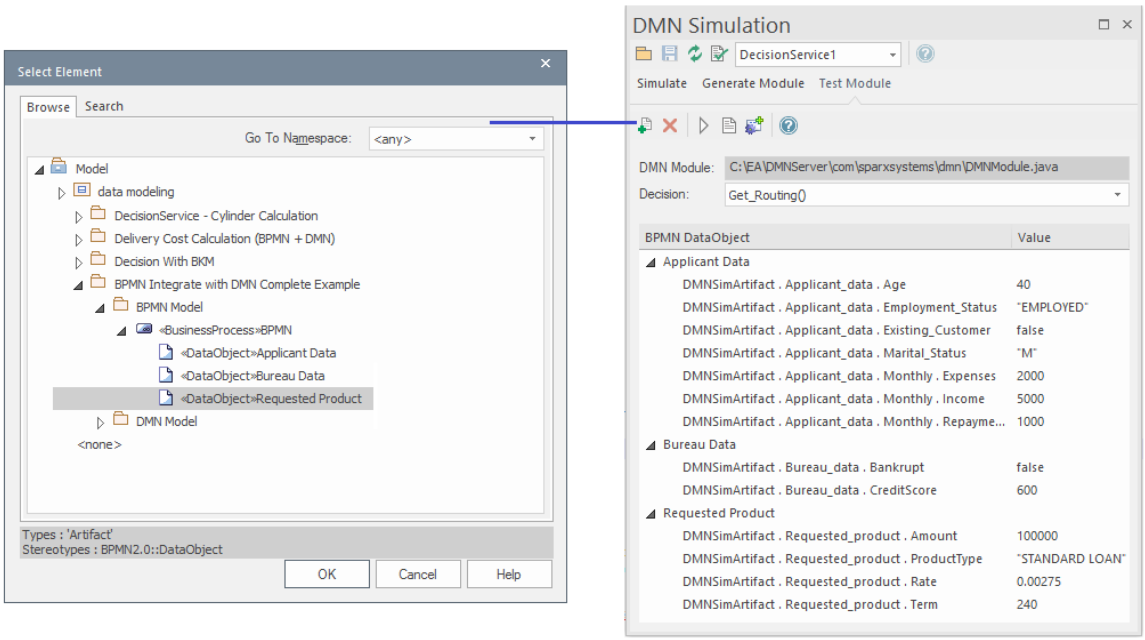
- Test l'exactitude de ce module
- Fournissez-lui des données d'entrée
- Obtenir les valeurs Décision de sortie

Générer un DataObject BPMN

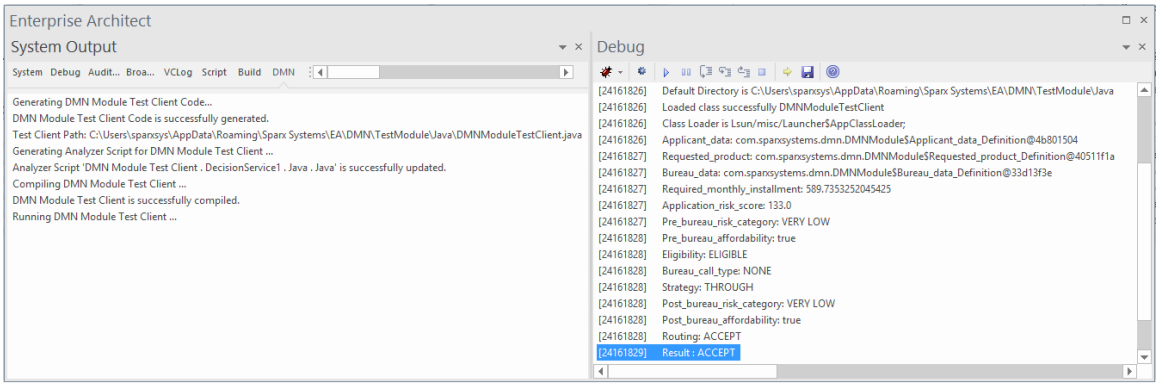


Les données transportées par l'ensemble de données sélectionné seront générées dans le champ « Notes » de l'objet de données BPMN.

- Cliquez sur le bouton  (2ème à droite sur la barre d'outils de l'onglet ' Générer un module') pour ouvrir l'onglet 'Module Test '



- Cliquez sur l'  dans la barre d'outils pour sélectionner les éléments d'entrée de l'objet de données BPMN
- Sélectionnez les sorties disponibles dans la liste déroulante « Décision », telles que Get_Routing(), et cliquez sur le bouton Exécuter de la barre d'outils



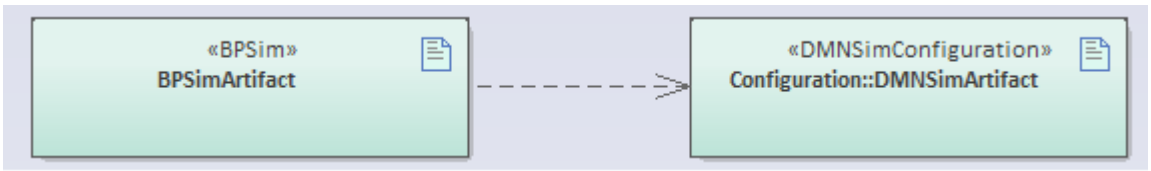
Le résultat de l'exécution sera affiché dans la fenêtre Déboguer . Vous pouvez également ouvrir le fichier du module de test, définir un point d'arrêt sur la ligne et déboguer dans le module DMN pour effectuer un débogage au niveau de la ligne.

Nous vous recommandons fortement de tester votre module DMN avec cette fenêtre pour garantir que le module DMN est fonctionnel avec les entrées fournies (à partir des DataObjects BPMN) et qu'il calculera avec succès le résultat de la sortie.

Note : le chemin du module DMN est enregistré dans la propriété « Filepath » de l'artefact DMNSimConfiguration.

Il est maintenant temps d'intégrer le module DMN avec le modèle BPSim.

La première étape consiste à configurer la dépendance d'utilisation entre l'artefact BPSim et l'artefact Simulation DMN.



Note : un artefact BPSim peut utiliser plusieurs modules DMN si nécessaire. Pour cela, il suffit de placer tous les artefacts DMN sur ce diagramme et de dessiner un connecteur de dépendance entre l'artefact BPSim et chaque artefact Simulation DMN.

Ces rubriques d'aide fournissent deux exemples d'utilisation de ces méthodes. Voir :

- *Exemple : Intégrer le service Décision DMN dans Object de données BPSim et Paramètres Propriété*
- *Exemple : Intégrer DMN Métier Knowledge Modèle dans BPSim Paramètres Propriété*

Exemple : Intégrer le service Décision DMN dans BPSim Data Object et Paramètres Propriété

Un exemple d'intégration d'un service DMN Décision dans le modèle BPSim est fourni dans le Constructeur de Modèle pour BPSim.

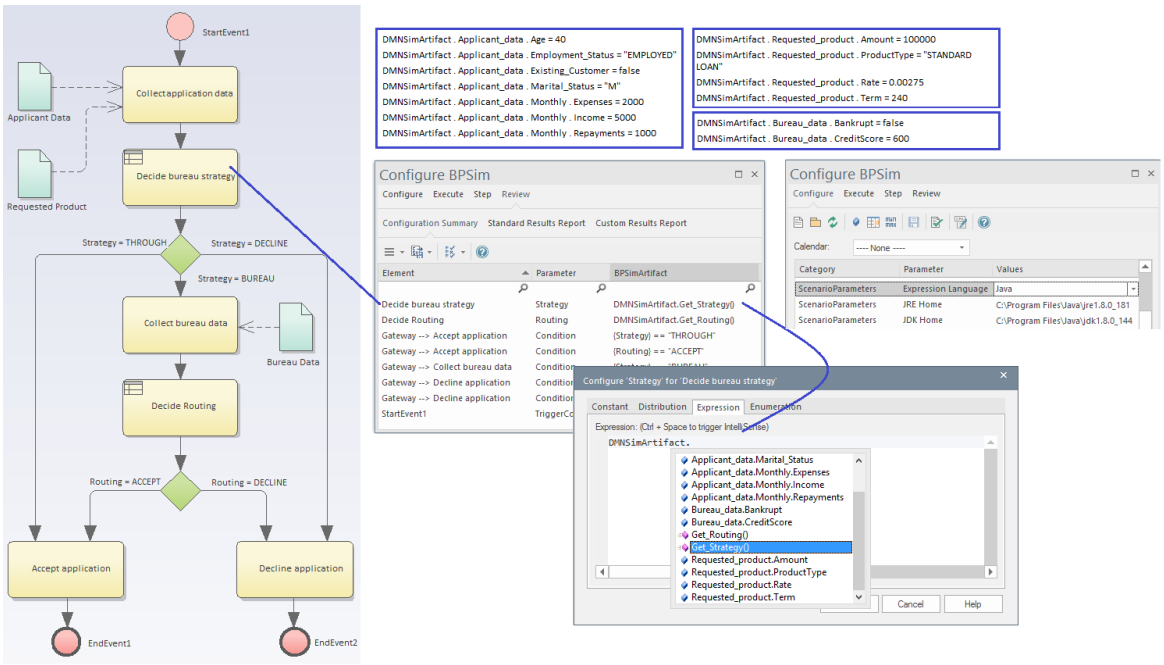
Pour y accéder :

- Poser la *perspective* de *Modélisation Métier > BPSim.* Dialogue Constructeur de Modèle s'affiche.
- Dans le groupe Études de cas BPSim, sélectionnez *Exemple complet d'intégration BPMN avec DMN*
- Cliquez sur le bouton Créer Modèle .

Cela créera des modèles BPMN et DMN configurés pour simuler un appel à un modèle DMN à partir du modèle BPMN.

Note : Pour intégrer le module DMN, le langage d'expression doit utiliser Java et le JRE et le JDK doivent être correctement configurés (la version minimale de Java est 1.7). Voir *Installer le Moteur d'Exécution BPSim* dans la rubrique d'aide BPSim Business Simulations .

Dans ce diagramme BPMN, il y a trois DataObjects (aqua) connectés à des activités BPMN. Ces éléments DataObject transportent des données d'entrée, générées à partir de la fenêtre Simulation DMN.



- Lorsque la simulation est en cours d'exécution, elle charge automatiquement tous les objets de données se connectant à la tâche lorsque le jeton de simulation passe par
- La deuxième tâche Règles Métier « Décider de la stratégie du bureau » est configurée pour définir la propriété « Stratégie » sur la valeur « DMNSimArtifact.Get_Strategy() » ; vous n'avez pas besoin de saisir cela - appuyez sur Ctrl+Espace pour vous aider à modifier l'expression

Une fois ces paramètres définis, cliquez sur l'onglet « Exécuter » et simulez le modèle. Vous pouvez ensuite consulter le rapport ou accéder à la page « Étape » pour effectuer le débogage étape par étape du modèle BPSim.

Exemple : Intégrer DMN Métier Knowledge Modèle dans BPSim Paramètres Propriété

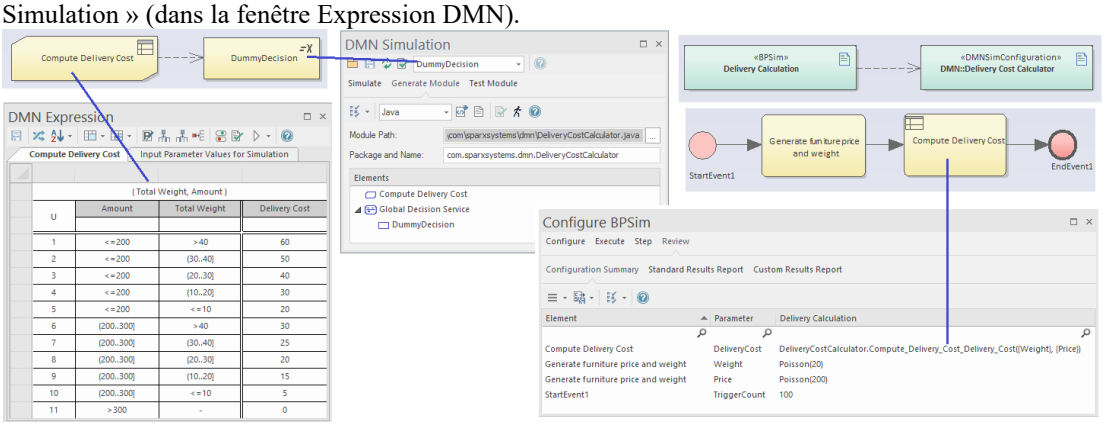
Dans certains cas, vous souhaiterez peut-être simplement concevoir un Tableau de Décision à utiliser dans un modèle BPMN. Dans ce cas, il n'est pas nécessaire de passer par les processus de création d'un Service Décision , Décision , de Données d'Entrée ou même d'une Définition Item , car un Modèle de Connaissances Métier (BKM) peut être directement interfacé.

Un exemple d'intégration d'un DMN BKM dans le modèle BPSim est fourni dans le Constructeur de Modèle pour BPSim.

Pour y accéder :

- Poser la *perspective* de *Modélisation Métier* > *BPSim.* La dialogue Constructeur de Modèle s'affiche.
- Dans le groupe Études de cas BPSim, sélectionnez *BPMN Integrate with DMN - Delivery Cost Calculation*
- Cliquez sur le bouton Créer Modèle

1. Créez un Modèle de connaissances Métier simple sous forme de Tableau de Décision (vous pouvez également créer d'autres expressions telles que le Contexte encadré ou les Expressions littérales) avec des paramètres, puis modélisez la logique (Clause d'entrée, Clause de sortie, règles) et testez-la (onglet « Valeurs des paramètres d'entrée pour Simulation » (dans la fenêtre Expression DMN).



2. Connectez le BKM à une Décision avec un connecteur d'exigence de connaissances. Cette Décision sert de nom de groupe pour un certain nombre de fonctions BKM ; vous pouvez simplement saisir un nombre tel que « 10 » dans l'expression. Par exemple, si vous souhaitez générer du code Java avec seulement cinq BKM (en considérant que votre modèle peut en avoir plus de cent), vous pouvez connecter ces cinq BKM à une Décision et sélectionner cette Décision dans la fenêtre Simulation DMN, puis les cinq BKM seront inclus automatiquement.
3. Générer du code Java et (en supposant que tout soit correct) la compilation réussira.
4. Dans la configuration BPSim, nous utilisons simplement Intelli-sense pour construire l'expression de la tâche « Calculer le coût de livraison ».

Dans cet exemple, la tâche « Générer le prix et le poids des meubles » générera des valeurs aléatoires pour les propriétés « Poids » et « Prix », puis la tâche « Calculer le coût de livraison » transmettra la valeur au Métier Knowledge Modèle et le résultat sera reporté sur la propriété « DeliveryCost ».

Vous pouvez maintenant exécuter la simulation et parcourir le processus de débogage pour observer, par exemple, les changements valeur d'attribut.

Intégrer dans l'élément de classe UML

Une fois qu'un Modèle Décision est créé et simulé, vous pouvez générer un module DMN en Java, JavaScript , C++ ou C# et le tester.

Le module DMN peut être intégré à un élément de classe UML , de sorte que le code généré à partir de cet élément de classe puisse réutiliser le module DMN et être bien structuré. Puisqu'un élément de classe peut définir une Statemachine , après intégration avec le module DMN, la simulation Statemachine Exécutable pourra génériquement utiliser la puissance du module DMN.

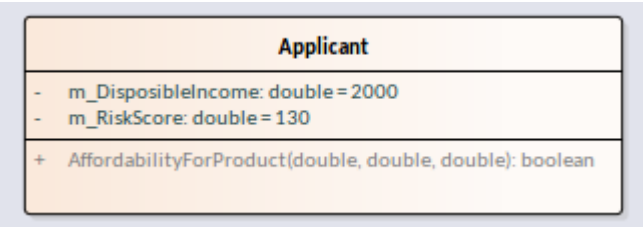
Dans cette rubrique, nous expliquerons le processus d'intégration d'un Modèle DMN avec un élément de classe UML , en tenant compte des éléments suivants :

- Exigence de l'élément de classe
- Modèles DMN
- Liaison DMN à la classe et à Intelli-sense
- Génération de code sur l'élément Class

Exigences relatives à l'élément de classe

Supposons que nous ayons une classe *Demandeur* avec une opération *AffordabilityForProduct* qui évalue si le demandeur peut se permettre un produit de prêt.

Un modèle simplifié ressemble à ceci :



Le *demandeur* de classe contient deux attributs, qui sont en fait calculés à partir de données plus basiques telles que le revenu mensuel du demandeur, ses dépenses, ses remboursements existants, son âge et son statut professionnel.

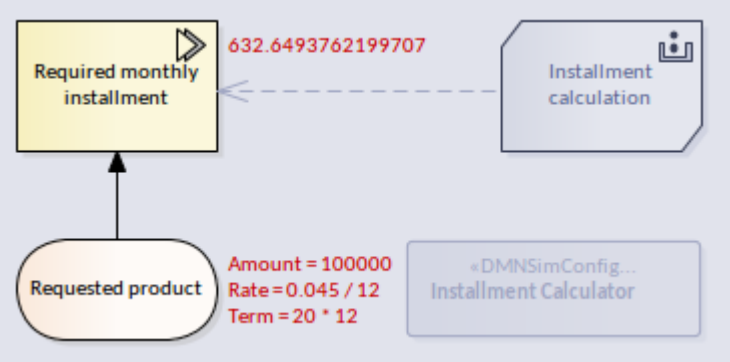
Dans cet exemple, cependant, nous simplifions le modèle en sautant ces étapes et en fournissant directement le revenu disponible et le score de risque.

Modèles DMN

Dans cet exemple, nous avons deux modèles DMN disjoints pour montrer qu'une classe UML peut intégrer plusieurs modèles DMN.

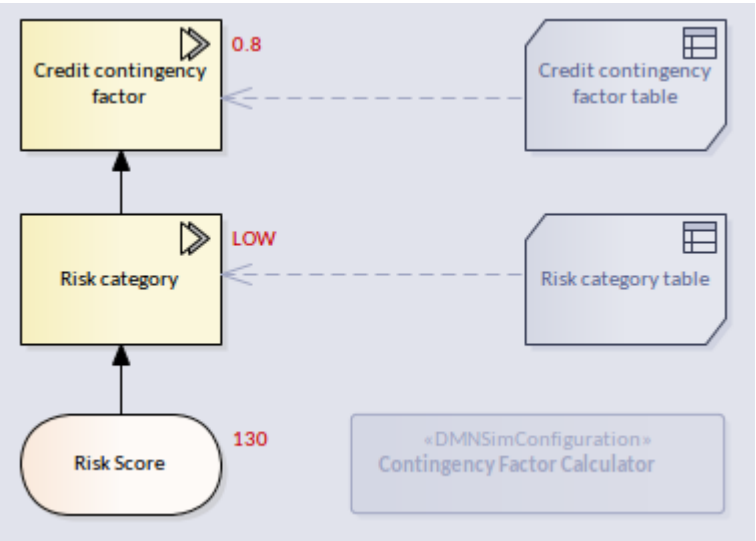
Calculateur de versements

Ce modèle DMN calcule le remboursement mensuel en fonction du montant, du taux et des modalités. Il est composé d'un InputData, d'un Décision et d'un Métier Knowledge Modèle .



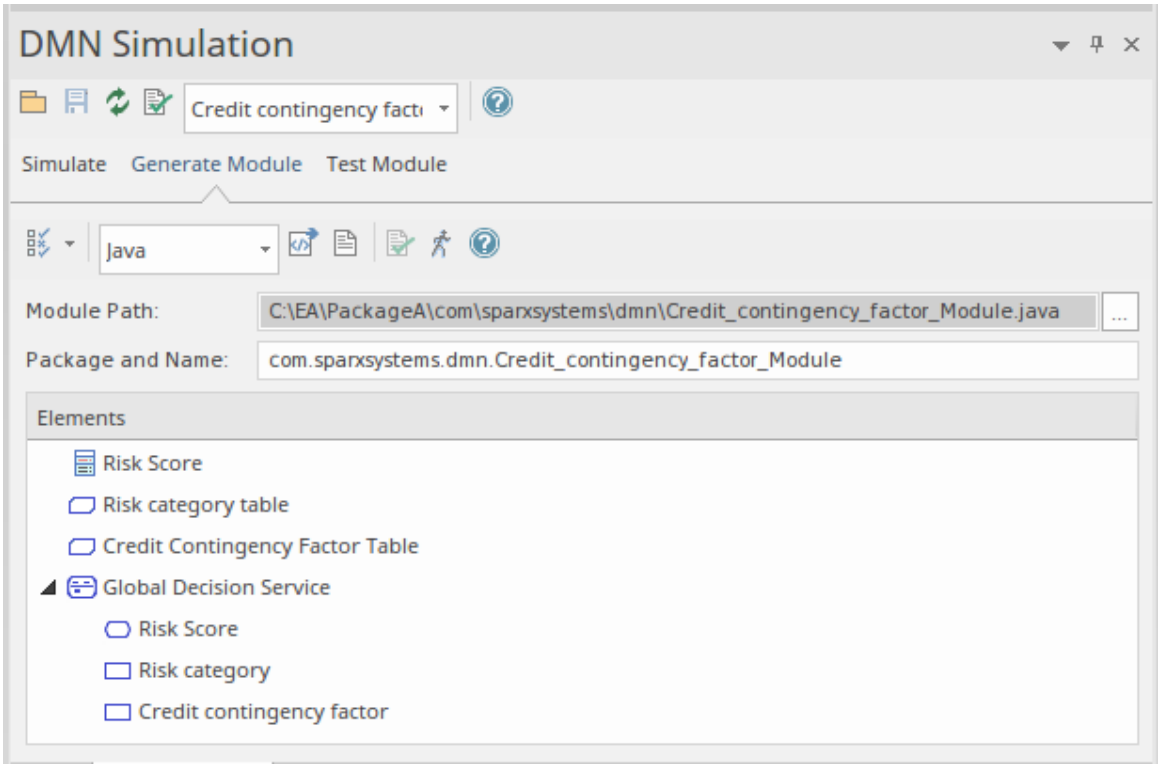
Calculateur de facteur de contingence de crédit

Ce modèle DMN calcule le facteur de contingence de crédit en fonction du score de risque du demandeur. Il est composé d'un InputData, de deux Décisions et de deux Métier Knowledge Models.



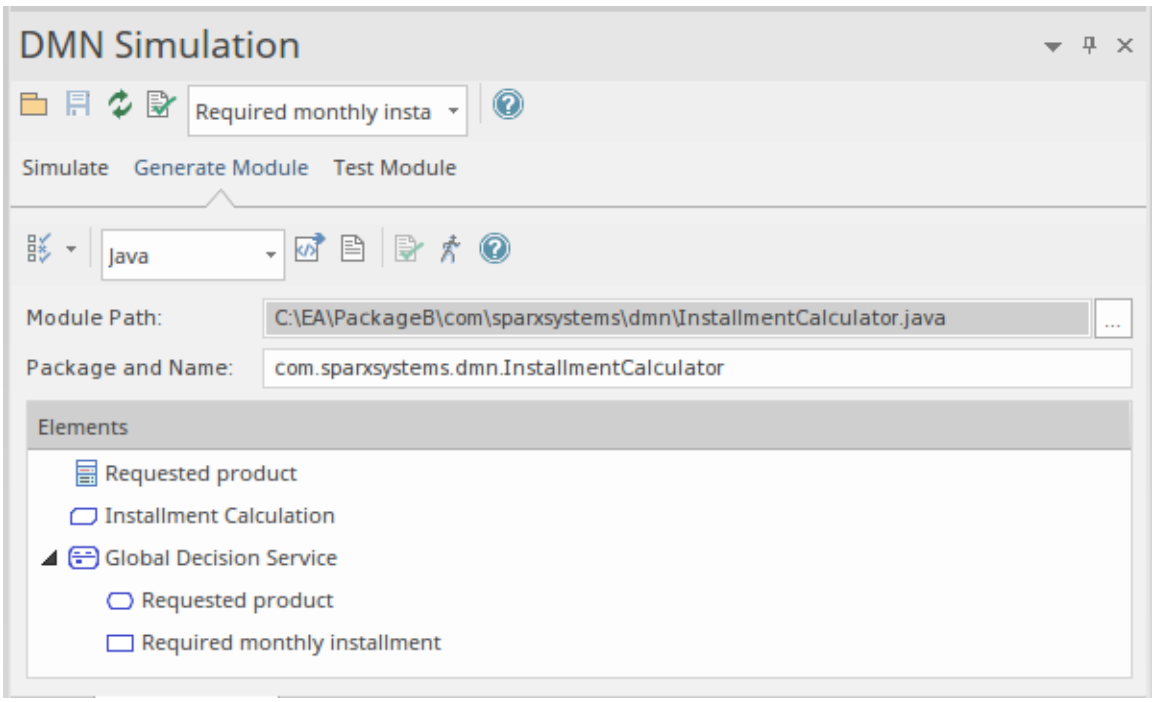
Note : dans cet exemple, nous nous concentrons sur la manière d'intégrer des modules DMN dans un élément de classe ; les détails des éléments DMN ne sont pas décrits ici.

Générer du code pour les deux modèles DMN

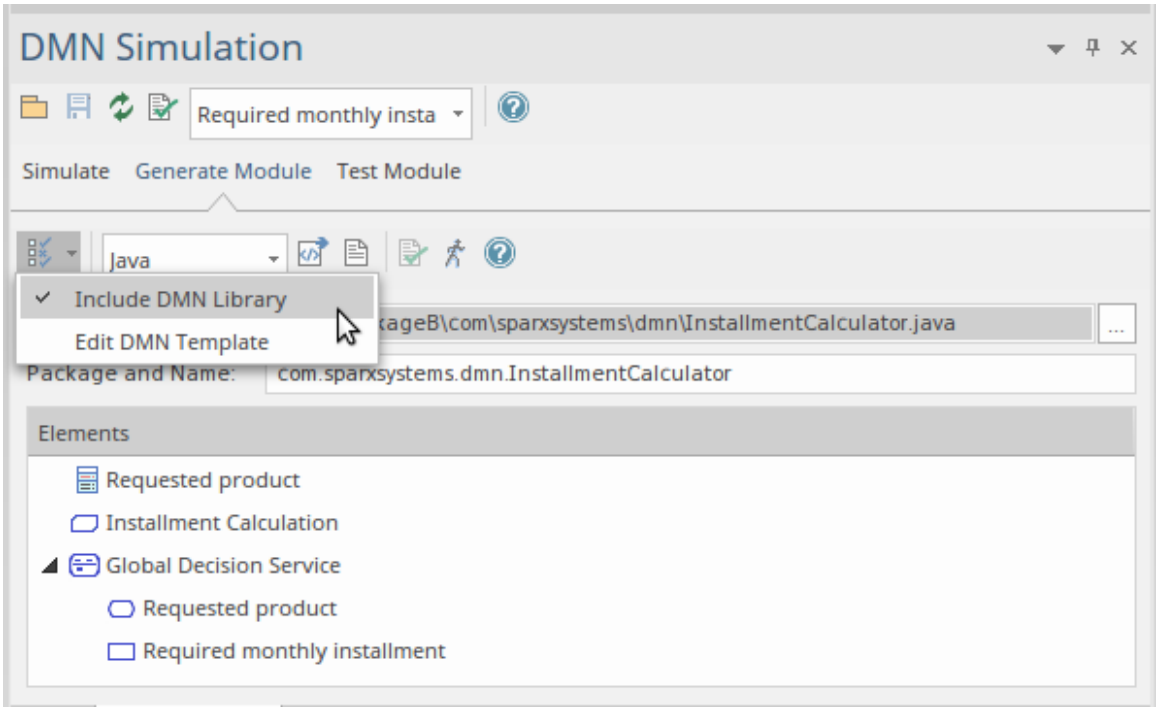


Cliquez sur l'icône Générer un code et vérifiez que vous pouvez voir cette string dans la fenêtre Sortie système, onglet « DMN » :

Le module DMN est compilé avec succès.



Note : Étant donné que ce modèle utilise une fonction PMT intégrée, la Bibliothèque DMN doit être incluse :

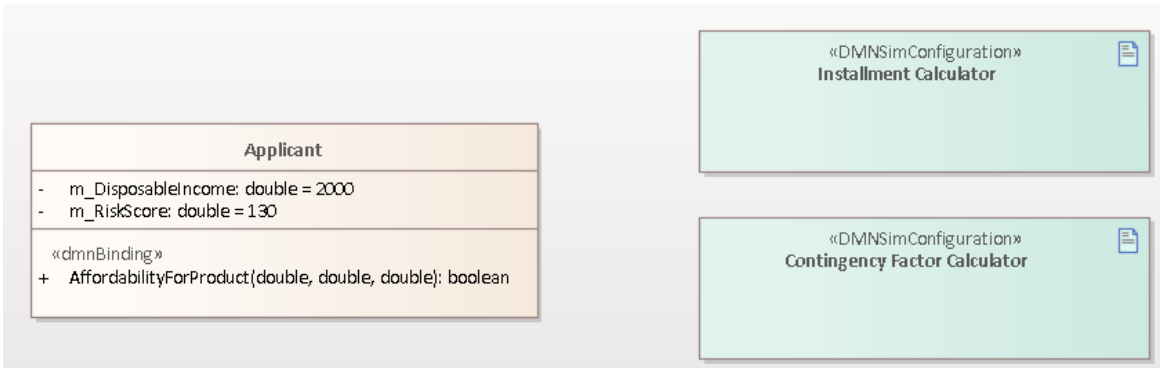


Cliquez sur l'icône Générer un code et vérifiez que vous pouvez voir cette string dans la fenêtre Sortie système, page « DMN » :

Le module DMN est compilé avec succès.

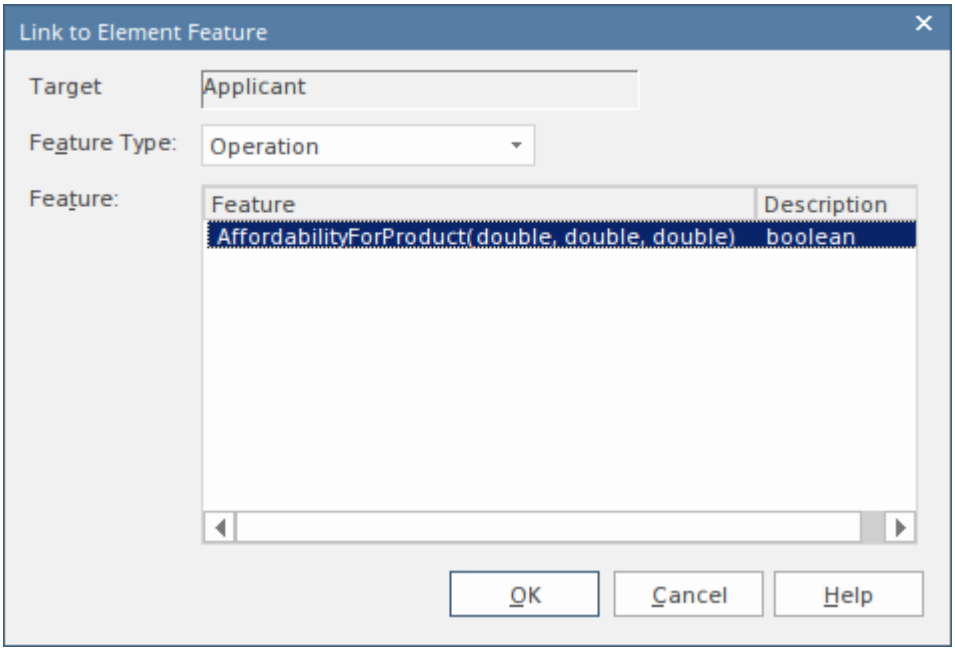
Liaison DMN à la classe et à Intelli-sense

Placez les deux artefacts DMNSimConfiguration sur le diagramme de classe.



Utilisez le Quick Linker pour créer un connecteur de dépendance du *demandeur* de classe vers chacun des artefacts DMN.

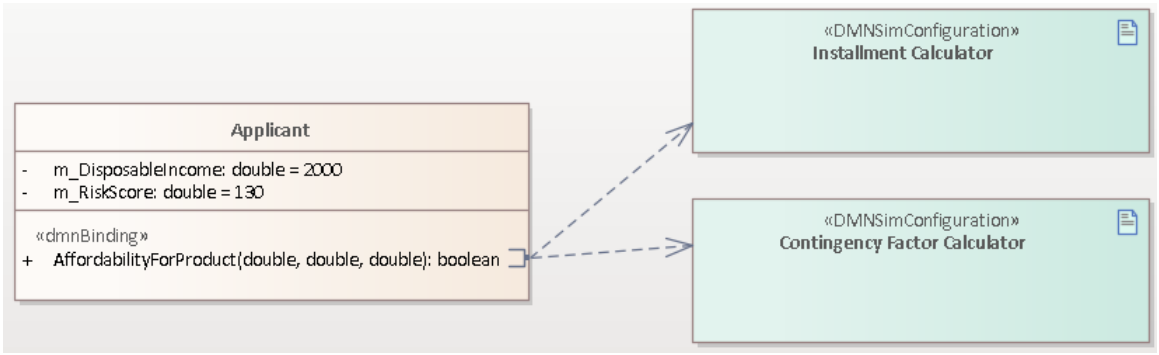
Lors de la création du connecteur, une dialogue vous prompt de choisir l'opération à lier au module DMN.



Lorsque le module DMN est lié à l'opération :

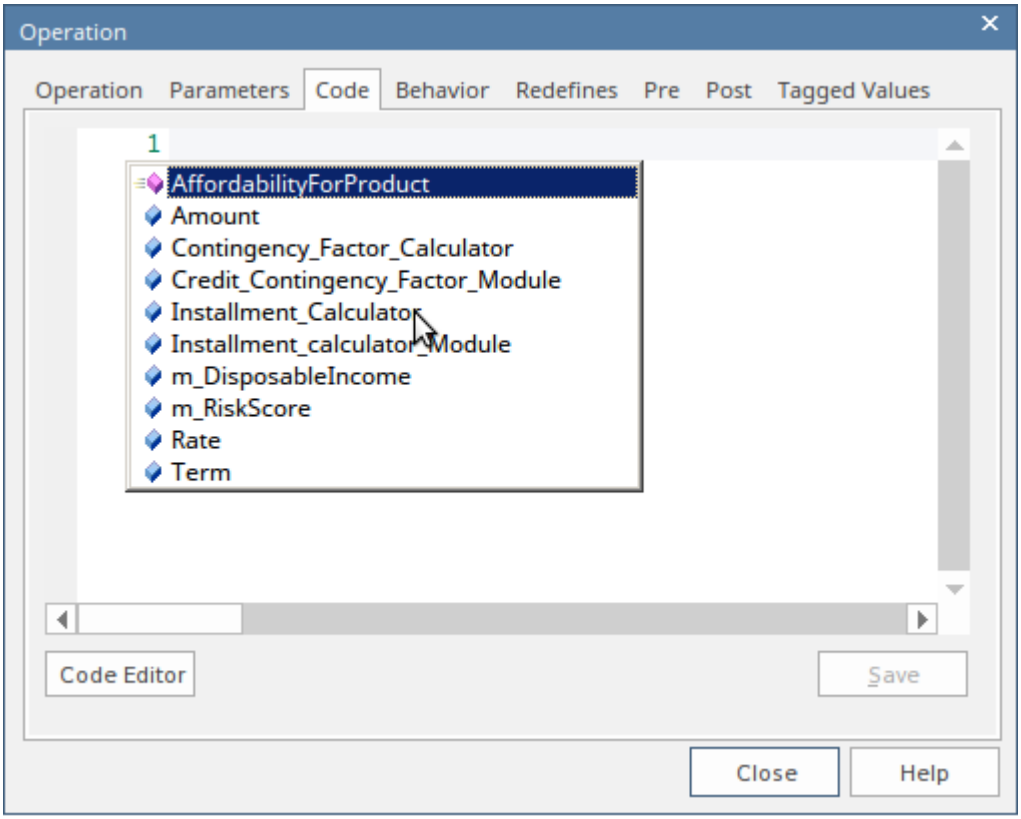
- L'opération prend un stéréotype <<dmnBinding>>
- Le connecteur de dépendance est lié à l'opération

Plusieurs artefacts DMN peuvent être liés à la même opération.



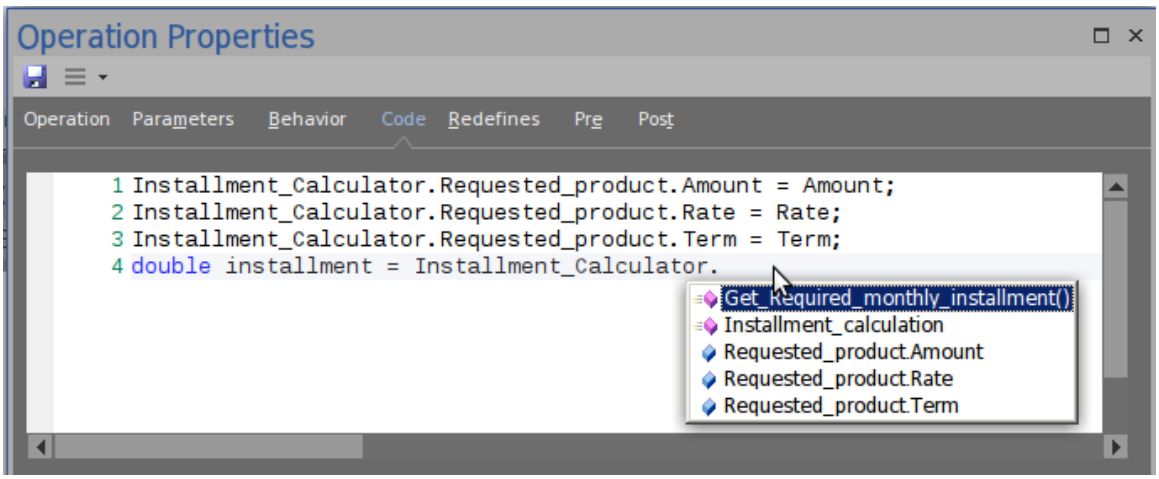
Après les liaisons DMN, Intelli-sense pour l'éditeur de code de l'opération prendra support les modules DMN. Pour déclencheur l'Intelli-sense, utilisez ces combinaisons de touches :

- Ctrl+Espace - dans la plupart des cas
- Ctrl+Maj+Espace - lorsque Ctrl+Espace ne fonctionne pas après une parenthèse '(' ; par exemple, les arguments d'une fonction, ou à l'intérieur des parenthèses d'une condition 'Si'



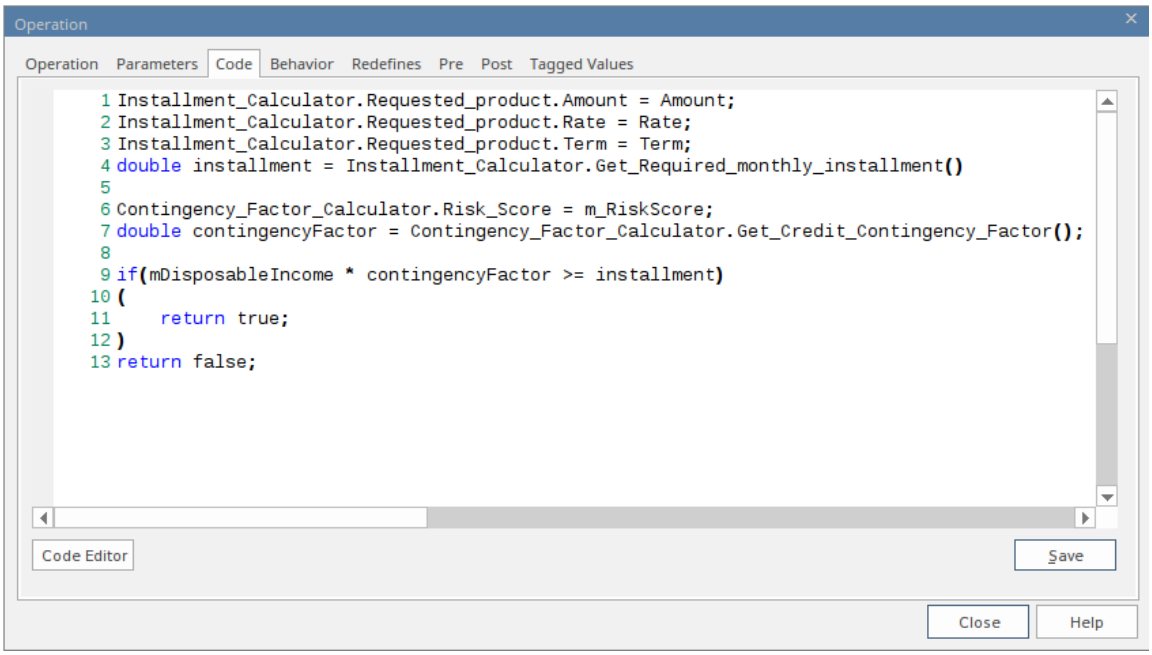
- Les attributs de classe seront répertoriés - m_RiskScore, m_DisposableIncome
- Les paramètres de fonctionnement seront répertoriés - Montant, Taux, Durée
- Les opérations seront répertoriées - AffordabilityForProduct
- Tous les modules DMN liés seront répertoriés - Contingency_Factor_Calculator, Installment_Calculator

Il est assez facile de composer le code avec support Intelli-sense. En accédant au module DMN, toutes les données d'entrée, les décisions et les modèles de connaissances Métier seront répertoriés pour la sélection.



Cette illustration montre que nous sélectionnons « Get_Required_monthly_installment() » dans Installment_Calculator.

Il s'agit de l'implémentation finale de l'opération.



Génération de code pour la classe (avec intégration DMN)

« Générer un code sur la classe candidate » produit ce code :

```
 8 public class Applicant {
 9
10     private double m_DisposibleIncome = 2000;
11     private double m_RiskScore = 130;
12
13     PackageA.ContingencyFactorCalculator Contingency_Factor_Calculator = new PackageA.ContingencyFactorCalculator();
14     PackageB.InstallmentCalculator Installment_Calculator = new PackageB.InstallmentCalculator();
15
16     public boolean AffordabilityForProduct(double Amount, double Rate, double Term){
17         //WARNING: Code in this function will be overwritten when generate from EA because this operation has a flush type of stereotype
18         Installment_Calculator.Requested_product.Amount = Amount;
19         Installment_Calculator.Requested_product.Rate = Rate;
20         Installment_Calculator.Requested_product.Term = Term;
21         double installment = Installment_Calculator.Get_Required_monthly_installment();
22
23         Contingency_Factor_Calculator.Risk_Score = m_RiskScore;
24         double contingencyFactor = Contingency_Factor_Calculator.Get_Credit_contingency_factor();
25
26         if(m_DisposibleIncome * contingencyFactor >= installment) {
27             return true;
28         }
29         return false;
30     }
31 }//end Applicant
```

- Les modules DMN sont générés en tant qu'attributs de la classe
- Le code de l'opération dmnBinding est mis à jour

Note : que l'option de génération soit « Écraser » ou « Synchroniser », le code de l'opération sera mis à jour s'il possède le stéréotype « dmnBinding ».

Importation de DMN XML

Enterprise Architect supporte l'importation d'un fichier XML DMN 1.1 ou 1.2 dans un projet, avec à la fois la sémantique du modèle et les informations d'échange de diagrammes.

Accéder

Dans la fenêtre Navigateur , sélectionnez le Paquetage dans lequel importer le fichier XML. Utilisez ensuite l'une des méthodes décrites ici pour ouvrir la dialogue « Importer Paquetage depuis DMN 1.1 XML ».

Ruban	Publier > Échange de Modèles > Importer > DMN 1.1
Raccourcis Clavier	Ctrl+Alt+I : Autres formats XML > DMN 1.1

Importer DMN 1.1 XML

Pas, pas	Action , Action
1	Dans le champ « Nom de fichier », saisissez le chemin et le nom du fichier source ou cliquez sur l'icône  pour localiser et sélectionner le fichier.
2	Cliquez sur le bouton Importer pour importer le fichier dans le Paquetage .

Importer l'exemple depuis OMG

1. Téléchargez le fichier zip sur this link et extrayez-le dans votre gestionnaire de fichiers.
2. Recherchez le dossier *examples/Chapitre 11/* .
3. Cliquez sur le fichier *Chapitre 11 Exemple.dmn* et importez-le en tant que fichier au format DMN 1.1.

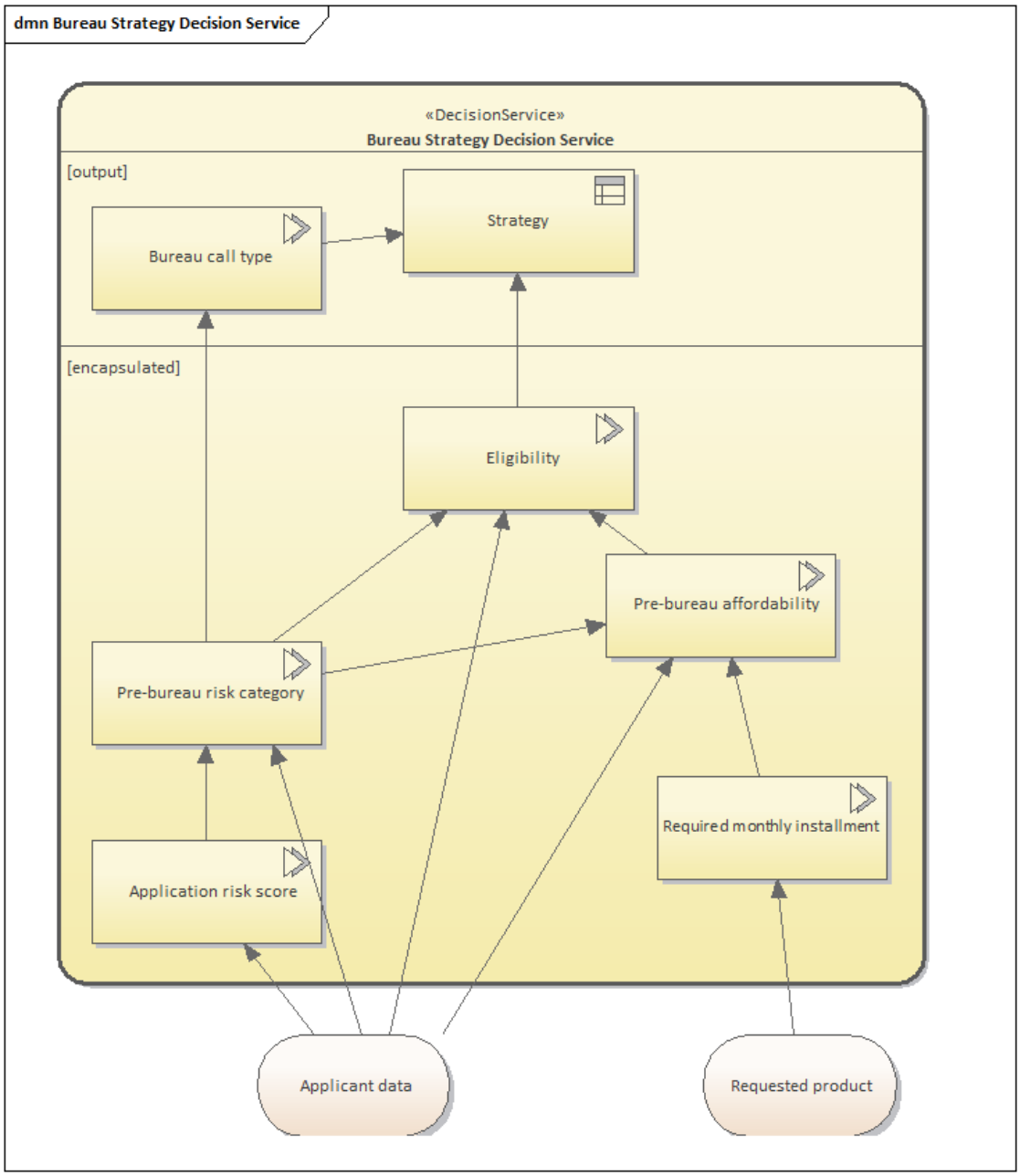
Ces diagrammes sont importés pour montrer différentes perspectives du modèle :

- DRD de toutes les prises de décision automatisées
- DRD pour le point de décision de la demande Révision
- DRD pour le point de décision Decide Routing
- DRD pour le point de décision stratégique du Decide Bureau

Ces diagrammes sont importés pour définir les Décision Services :

- Service Décision stratégique du Bureau
- Service Décision de routage

Le diagramme « Bureau Strategy Décision Service » est présenté ici. Il comporte deux éléments de données d'entrée (données du demandeur, produit demandé), deux Décisions de sortie (type d'appel du Bureau, stratégie) et cinq décisions encapsulées. Note que les modèles de connaissances Métier invoqués ne sont pas représentés sur le diagramme .



Afin de générer du code de production à partir du modèle, vous devrez peut-être exécuter une validation et une simulation pour vous assurer que le modèle importé possède les expressions correctes.

1. Créez un artefact de configuration de simulation DMN sur l'un des diagrammes répertoriés et double-cliquez dessus pour l'ouvrir dans la fenêtre Simulation DMN.
2. Les services Décision Services et Décisions sont répertoriés dans le champ déroulant de la cible. Une fois que vous avez spécifié une cible, tous les éléments requis sont répertoriés dans la fenêtre.
3. Cliquez sur le bouton Valider (4ème sur la barre d'outils). Si des messages d'erreur ou d'avertissement s'affichent, nous vous suggérons de corriger les problèmes comme indiqué dans les descriptions des erreurs ou des avertissements, avant d'effectuer la simulation.
4. Fournissez des valeurs appropriées pour les entrées et exécuter la simulation ou déboguez le modèle étape par étape.

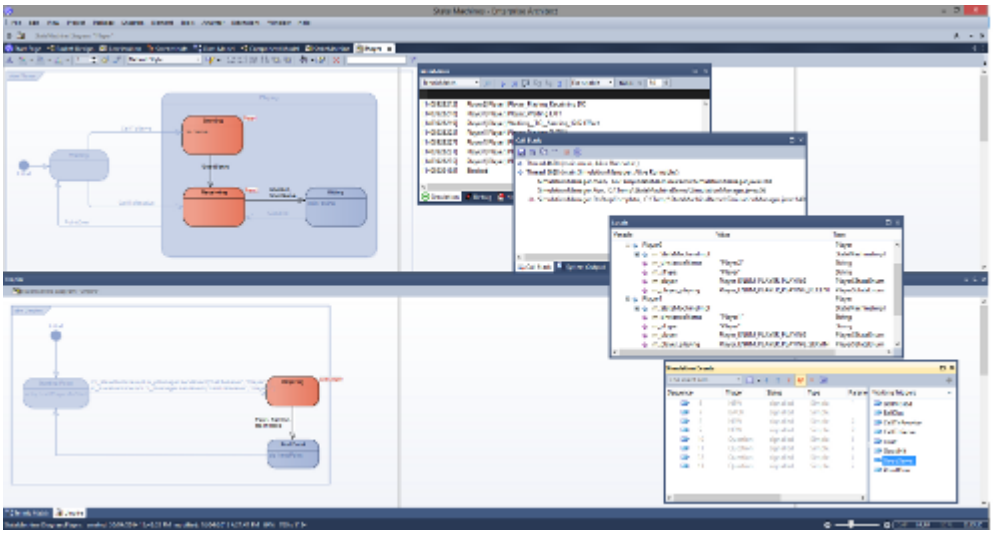
Note : L'exemple 'Bureau Strategy Décision Service' est également disponible dans l'EAExample Modèle . Dans le diagramme ' Démarrage ', sélectionnez ' Métier Modélisation > Exemples DMN > Bureau Strategy Décision Service'.

Plus d'informations

Dans Enterprise Architect , la fonctionnalité Decision Model and Notation (DMN) remplit une fonction essentielle : fournir les éléments essentiels pour modélisation efficace des décisions. Cette fonctionnalité permet de représenter visuellement la prise de décision organisationnelle dans diagrammes au sein de l'outil, permettant analystes métier de définir avec précision les modèles de décision. Enterprise Architect supporte l'automatisation facultative de ces modèles de décision, simplifiant ainsi les processus de prise de décision.

Les fonctionnalités DMN d' Enterprise Architect facilitent le partage et l'échange transparents de modèles Décision entre les organisations. Cette interopérabilité garantit que les modèles de décision peuvent être facilement communiqués et utilisés en collaboration, favorisant ainsi une prise de décision efficace entre les différentes équipes et parties prenantes.

Statemachines Exécutables



Statemachines Exécutables permettent de générer, d'exécuter et de simuler rapidement des modèles d'état complexes. Contrairement à la simulation dynamique de diagrammes State à l'aide du moteur Simulation d' Enterprise Architect , Statemachines Exécutables fournissent une implémentation complète spécifique au langage qui peut former le « moteur » comportemental de plusieurs produits logiciels sur plusieurs plates-formes. La visualisation de l'exécution est basée sur une intégration transparente avec la capacité Simulation . L'évolution du modèle présente maintenant moins de défis de codage. La génération, la compilation et l'exécution du code sont prises en charge par Enterprise Architect . Pour ceux qui ont des exigences particulières, chaque langage est fourni avec un ensemble de gabarits de code. Vous pouvez personnaliser Gabarits pour adapter le code généré de la manière qui vous convient.

Ces rubriques vous présentent les bases de modélisation Statemachines Exécutables et vous aident à comprendre comment les générer et les simuler.

La création et l'utilisation d' Statemachines Exécutables , ainsi que la génération de code à partir de ceux-ci, sont prises en charge par les éditions Unified et Ultimate d' Enterprise Architect .

Présentation de la construction et de l'exécution Statemachines

Créer et utiliser Statemachines Exécutables est assez simple, mais nécessite un peu de planification et quelques connaissances sur la manière de lier les différents composants pour créer un modèle d'exécution efficace. Heureusement, vous n'avez pas besoin de passer des heures à obtenir le bon modèle et à corriger les erreurs de compilation avant de pouvoir commencer à visualiser votre conception.

Après avoir esquissé les grandes lignes de votre modèle, vous pouvez générer le code pour le piloter, le compiler, l'exécuter et le visualiser en quelques minutes. Ces points résument ce qui est nécessaire pour commencer à exécuter et à simuler Statemachines .

Facilité	Description
Construire des modèles de classes et State	La première tâche consiste à créer les modèles de classe et State UML standard qui décrivent les entités et le comportement à construire. Chaque classe qui vous intéresse dans votre modèle doit avoir sa propre Statemachine qui décrit les différents états et transitions qui régissent son comportement global.
Créer un artefact Statemachine Exécutable	Une fois que vous avez modélisé vos classes et vos modèles State , il est temps de concevoir l'artefact Statemachine Exécutable . Celui-ci décrira les classes et les objets impliqués, ainsi que leurs propriétés et relations initiales. C'est le script de liaison qui lie plusieurs objets entre eux et détermine comment ceux-ci communiqueront au moment de l'exécution. Note qu'il est possible d'avoir deux ou plusieurs objets dans un artefact Statemachine Exécutable en tant qu'instances d'une

	seule classe. Ceux-ci auront leur propre état et comportement au moment de l'exécution et pourront interagir si nécessaire.
Générer du code et compiler	Que vous utilisiez JavaScript , C++, Java ou C# , les capacités d'ingénierie d' Enterprise Architect vous offrent un outil efficace, vous permettant de régénérer l'exécutable à tout moment, et sans perte du code personnalisé que vous auriez pu créer. Il s'agit là d'un avantage majeur sur la durée de vie d'un projet. Il convient également de noter que l'ensemble de la base de code générée est indépendante et portable. Le code n'est en aucun cas couplé à une infrastructure utilisée par le moteur de simulation.
Exécuter Statemachines	Alors, comment pouvons-nous voir comment ces Statemachines se comportent ? Une méthode consiste à construire la base de code pour chaque plate-forme, à l'intégrer dans un ou plusieurs systèmes, en examinant les comportements, « in situ », dans peut-être plusieurs scénarios de déploiement. Ou nous pouvons l'exécuter avec Enterprise Architect . Qu'il s'agisse de Java, JavaScript , C, C++ ou C# , Enterprise Architect se chargera de créer le runtime, l'hébergement de votre modèle, l'exécution de ses comportements et le rendu de toutes Statemachines .
Visualiser Statemachines	La visualisation Statemachine Exécutable s'intègre aux outils Simulation d' Enterprise Architect . Observez les transitions d'état au fur et à mesure qu'elles se produisent sur votre diagramme et pour quel(s) object (s). Identifiez facilement les objets partageant le même état. Il est important de noter que ces comportements restent cohérents sur plusieurs plates-formes. Vous pouvez également contrôler la vitesse à laquelle les machines fonctionnent pour mieux comprendre la chronologie des événements.
Déboguer Statemachines	Lorsque les états doivent changer mais ne le font pas, lorsqu'une transition ne doit pas être activée mais l'est, lorsque le comportement est - en bref - indésirable et n'est pas immédiatement apparent à partir du modèle, nous pouvons nous tourner vers le débogage. L' Analyseur d'Exécution Visuelle d' Enterprise Architect est fourni avec des débogueurs pour tous les langages pris en charge par la génération de code ExecutableStateMachine. Le débogage offre de nombreux avantages, dont l'un peut être de vérifier/corroborer le code attaché aux comportements dans une Statemachine pour s'assurer qu'il est réellement reflété dans le processus d'exécution.

Modélisation Statemachines Exécutables

La plupart du travail requis pour modéliser un Statemachine Exécutable est modélisation standard basée sur UML des classes et des modèles State , bien qu'il existe quelques conventions qui doivent être respectées pour garantir une base de code bien formée. La seule construction nouvelle est l'utilisation d'un élément Artifact stéréotypé pour former la configuration d'une instance ou d'un scénario Statemachine Exécutable . L'Artifact est utilisé pour spécifier des détails tels que :

- Le langage de code (JavaScript , C# , Java, C++ y compris C)
- Les classes et Statemachines impliquées dans le scénario
- Les spécifications d'instance, y compris l'état d'exécution ; note que cela peut inclure plusieurs instances de la même Statemachine , par exemple lorsqu'une classe « Joueur » est utilisée deux fois dans une simulation de match de tennis

Outils et objets Modélisation de base pour Statemachines Exécutables

Ce sont les principaux éléments modélisation utilisés lors de la construction Statemachines Exécutables .

Type d'élément	Description
Classes et Diagrammes de classes	Les classes définissent les types object pertinents pour la ou Statemachine modélisées. Par exemple, dans un scénario de match de tennis simple, vous pouvez définir une classe pour chaque joueur, match, Hit et arbitre. Chacun aura sa ou ses propres Statemachine et sera représenté au moment de l'exécution par des instances object pour chaque entité impliquée. Consultez le *Guide Modélisation UML* pour plus d'informations sur les classes et diagrammes de classes.
Statemachines	Pour chaque classe que vous définissez et qui aura un comportement dynamique dans un scénario, vous définirez généralement une ou plusieurs Statemachines UML . Chaque Statemachine déterminera le comportement légal basé sur l'état approprié pour un aspect de la classe propriétaire. Par exemple, il est possible d'avoir une Statemachine qui représente l'état émotionnel d'un joueur, une autre qui suit sa forme physique et ses niveaux d'énergie actuels, et une autre qui représente son état de victoire ou de défaite. Toutes ces Statemachines seront initialisées et démarrées lorsque le scénario Statemachine commencera à s'exécuter.
Artefact Statemachine Exécutable	Cet artefact stéréotypé est l'élément central utilisé pour spécifier les participants, la configuration et les conditions de démarrage d'un Statemachine Exécutable . Du point de vue du scénario, il est utilisé pour déterminer quelles instances (de classes) sont impliquées, quels événements elles peuvent Déclencheur et s'envoyer les unes aux autres, et dans quelles conditions de démarrage elles fonctionnent. Du point de vue de la configuration, l'Artifact est utilisé pour établir le lien vers un script d'analyse qui déterminera le répertoire de sortie, le langage de code, le script de compilation et autres. Un clic droit sur l'Artifact vous permettra de générer, construire, compiler et visualiser l'exécution en temps réel de vos Statemachines .

Constructions Statemachine prises en charge

Ce tableau détaille les constructions Statemachine prises en charge et toutes les limitations ou contraintes générales pertinentes pour chaque type.

Construction	Description

Statemachines	• Statemachine simple : la Statemachine a une région • Statemachine orthogonale : la Statemachine contient plusieurs régions Sémantique d'activation de la région de niveau supérieur (appartenant à Statemachine) : **Activation par défaut :** lorsque la Statemachine commence à s'exécuter. **Point d'entrée Entrée :** Transitions du point d'entrée vers les sommets des régions contenues. • *Note 1 : Dans chaque région de la Statemachine possédant le point d'entrée, il existe au plus une seule transition du point d'entrée à un sommet dans cette région* • *Note 2 : cette Statemachine peut être référencée par un State de sous-machine - les références de point de connexion doivent être définies dans l' State de sous-machine comme sources/cibles de transitions ; la référence de point de connexion représente une utilisation d'un point d'entrée/sortie défini dans la Statemachine et référencé par l' State de sous-machine* **Statemachines multiples :** L'ordre d'affichage dans la fenêtre Navigateur détermine l'ordre d'exécution. • Lorsqu'un State de sous-machine est impliqué, il peut y avoir plusieurs Statemachines sous la classe • Utilisez les flèches Monter ou Descendre dans la barre d'outils de la fenêtre Navigateur pour ajuster l'ordre des Statemachines ; celle du haut sera définie comme Statemachine principale **Non pris en charge** • Statemachine de protocole • Redéfinition Statemachine
States	• State simple : n'a pas de sommets ou de transitions internes • State composite : contient exactement une région • State orthogonal : contient plusieurs régions • State de sous-machine : fait référence à une Statemachine entière
Entrée State composite	• Entrée par défaut • Entrée explicite • Entrée d'histoire superficielle • Entrée d'histoire profonde • Point d'entrée Entrée
Sous-États	• Sous-états et sous-états imbriqués La sémantique d'entrée et de sortie, où la transition comprend plusieurs niveaux d'états imbriqués, obéira à l'exécution correcte des comportements imbriqués (tels que OnEntry et OnExit).
Support aux transitions	• Transition externe • Transition locale • Transition interne (dessinez une transition personnelle et changez le type de transition en interne) • Achèvement Transition et Achèvement Événements • Gardes de transition • Transitions composées

	• Priorités de tir et algorithme de sélection Pour plus de détails, reportez-vous à la *Spécification UML UML* .
Déclencheur et Événements	Un Statemachine Exécutable supporte la gestion des événements pour les signaux uniquement. Pour utiliser les types d'événements d'appel, de synchronisation ou de modification, vous devez définir un mécanisme externe pour générer des signaux en fonction de ces événements.

Signal	Attributes peuvent être définis dans Signaux ; la valeur des attributs peut être utilisée comme arguments d'événement dans Transition Gardes et Effets .
	Par exemple, voici le code défini dans l'effet d'une transition en C++ :
	si(signal->signalEnum == ENUM_SIGNAL2)
	{
	int xVal = ((Signal2*)signal)->maVal;
	}
	Signal2 est généré sous la forme de ce code :
	classe Signal2 : public Signal{
	publique:
	Signal2(){};
	Signal2(std::vector<String>& lstArguments);
	int maVal;
	};
	Note : des détails supplémentaires peuvent être trouvés en générant un Statemachine Exécutable et en se référant au fichier « EventProxy » généré.

Initial	Un pseudo-état initial représente un point de départ pour une région. Il est la source d'au plus une transition ; il ne peut y avoir qu'un seul sommet initial dans une région.
Régions	**Activation par défaut et activation explicite :** Les transitions se terminent sur l' State contenant : • Si un pseudo-état initial est défini dans la région : **activation par défaut** • Si aucun pseudo-état initial n'est défini, la région restera inactive et l' State qui la contient sera traité comme un State simple • Si la transition se termine sur l'un des sommets contenus dans la région : **activation explicite** , entraînant l'activation par défaut de toutes ses régions orthogonales, à moins que ces régions ne soient également saisies explicitement (plusieurs régions orthogonales peuvent être saisies explicitement en parallèle via des transitions provenant du même pseudo-état de fourche) Par exemple, s'il existe trois régions définies pour un State orthogonal et que *les régions A* et *B* ont un pseudo-état initial, alors *la région C* est explicitement activée. L'activation par défaut s'applique aux *régions A* et *B* ; l' State contenant aura trois régions actives.
Choix	Les contraintes de garde sur toutes les transitions sortantes sont évaluées de manière dynamique lorsque la traversée de transition composée atteint ce pseudo-état.
Jonction	Branche conditionnelle statique : les contraintes de garde sont évaluées avant

	l'exécution de toute transition composée.
Fourcher / Rejoindre	Non threadé, chaque région active se déplace d'une étape en alternance, en fonction d'un mécanisme de pool d'événements d'achèvement.
Nœuds de point d'entrée/point de sortie	Non threadé pour State orthogonal ou Statemachine orthogonale ; chaque région active se déplace d'une étape en alternance, en fonction d'un mécanisme de pool d'événements d'achèvement.
Nœuds d'histoire	• DeepHistory : représente la configuration State active la plus récente de son State propriétaire • ShallowHistory : représente le sous-état actif le plus récent de l' State qui le contient, mais pas les sous-états de ce sous-état
Événements différés	Dessinez une transition automatique et modifiez le *type* de transition en interne. Type « defer(); » dans le champ « Effet » pour la transition.
Références des points de connexion	Une référence de point de connexion représente une utilisation (dans le cadre d'un State de sous-machine) d'un point d'entrée/sortie défini dans la Statemachine référencée par l' State de sous-machine. Les références de point de connexion d'un State de sous-machine peuvent être utilisées comme sources et cibles de transitions. Elles représentent des entrées ou des sorties de la Statemachine référencée par l' State de sous-machine.
Comportements State	Les comportements State « entrée », « doActivity » et « sortie » sont définis comme des opérations sur un State . Par défaut, vous saisissez le code qui sera utilisé pour chaque comportement dans le champ Panneau 'Code' de la fenêtre Propriétés pour l'opération Comportement. Note que vous pouvez modifier cela pour saisir le code dans le panneau 'Comportement', en personnalisant le gabarit de génération. Le comportement « doActivity » généré sera exécuter jusqu'à son terme avant de continuer. Le code n'est pas simultané avec d'autres comportements d'entrée ; le comportement « doActivity » est implémenté comme comportement « exécuter dans la séquence après l'entrée ».

Références à des comportements dans d'autres contextes/classes

Si l' State de la sous-machine fait référence à des éléments de comportement en dehors du contexte ou de la classe actuels, vous devez ajouter un connecteur <<import>> de la classe de contexte actuelle à la classe de contexte du conteneur. Par exemple :

State de sous-machine S1 de la classe 1 fait référence à Statemachine ST2 de la classe 2

Par conséquent, nous ajoutons un connecteur <<import>> de la classe 1 à la classe 2 afin que la génération de code Statemachine Exécutable génère correctement le code pour State sous-machine S1. (Sur la classe 1, cliquez sur la flèche Quick Linker et faites-la glisser vers la classe 2, puis sélectionnez « Importer » dans le menu des types de connecteurs.)

Réutilisation d'artefacts Statemachine Exécutable

Vous pouvez créer plusieurs modèles ou versions d'un composant à l'aide d'un seul artefact exécutable. Un artefact représentant une résistance, par exemple, peut être réutilisé pour créer à la fois une résistance à feuille et une résistance à fil enroulé. Cela est susceptible d'être le cas pour des objets similaires qui, bien que représentés par le même classificateur, présentent généralement des états exécuter différents. Une propriété nommée 'resistorType' prenant la

valeur 'wire' plutôt que 'foil' peut être tout ce qui est requis du point de vue de modélisation . Les mêmes Statemachines peuvent ensuite être réutilisées pour tester les changements de comportement qui pourraient résulter de la variation de l'état d'exécution. Voici la procédure :

Étape	Action
Créer ou ouvrir diagramme de composants	Ouvrez un diagramme de composant sur lequel travailler. Il peut s'agir du diagramme qui contient votre artefact d'origine.
Sélectionnez l' Statemachine Exécutable à copier	Recherchez maintenant l'artefact Statemachine Exécutable original dans la fenêtre Navigateur .
Créer le nouveau composant	Tout en maintenant la touche Ctrl enfoncée, faites glisser l'artefact d'origine sur votre diagramme . Deux questions vous seront posées. La réponse à la première question est **Object** et à la seconde, **Tout.** Renommez l'Artefact pour le différencier de l'original, puis modifiez ses valeurs de propriété.

Artefact Statemachine Exécutable

Un artefact Statemachine Exécutable est essentiel pour générer Statemachines qui peuvent interagir les unes avec les autres. Il spécifie les objets qui seront impliqués dans une simulation, leur état et la manière dont ils se connectent. L'un des grands avantages de l'utilisation d'artefacts Statemachine Exécutable est que chacune des différentes parties d'un artefact peut représenter une instance d'une Statemachine . Vous pouvez donc configurer des simulations à l'aide de plusieurs instances de chaque Statemachine et observer comment elles interagissent. Un exemple est fourni dans la rubrique d'aide *Exemple d' Statemachine Exécutable* .

Création des Propriétés d'une Statemachine Exécutable

Chaque scénario Statemachine Exécutable implique une ou plusieurs Statemachines . Les Statemachines incluses sont spécifiées par des éléments de propriété UML ; chaque propriété aura un classificateur UML (classe) qui détermine la ou Statemachine incluses pour ce type. Plusieurs types inclus en tant que plusieurs Propriétés peuvent finir par inclure de nombreuses Statemachines , qui sont toutes créées dans le code et initialisées à l'exécution.

Action	Description
Déposez une classe de la fenêtre Navigateur sur l'artefact << Statemachine Exécutable >>	La façon la plus simple de définir des propriétés sur un Statemachine Exécutable est de déposer la classe sur l' Statemachine Exécutable à partir de la fenêtre Navigateur . Dans le dialogue qui s'affiche, sélectionnez l'option permettant de créer une propriété. Vous pouvez ensuite spécifier un nom décrivant la manière dont l' Statemachine Exécutable fera référence à cette propriété. Note : selon vos options, vous devrez peut-être maintenir la touche Ctrl enfoncée pour choisir de créer une propriété. Ce comportement peut être modifié à tout moment en cochant la case « Maintenir la touche Ctrl enfoncée pour afficher cette boîte dialogue ».
Utiliser et connecter plusieurs Propriétés UML	Un Statemachine Exécutable décrit l'interaction de plusieurs Statemachines . Il peut s'agir de différentes instances de la même Statemachine , de différentes Statemachines pour la même instance ou Statemachines complètement différentes de différents types de base. Pour créer plusieurs propriétés qui utiliseront la même Statemachine , déposez la même classe sur l'artefact plusieurs fois. Pour utiliser différents types, déposez différentes classes à partir de la fenêtre Navigateur selon vos besoins.

Définition de l' State initial des Propriétés

Les Statemachines exécuter par un Statemachine Exécutable exécuter toutes dans le contexte de leur propre instance de classe. Un Statemachine Exécutable vous permet de définir l'état initial de chaque instance en attribuant des valeurs de propriété à divers attributs de classe. Par exemple, vous pouvez spécifier l'âge, la taille, le poids ou d'autres propriétés similaires d'un joueur si ces propriétés sont pertinentes pour le scénario en cours exécuter . En procédant ainsi, il est possible de définir des conditions initiales détaillées qui influenceront le déroulement du scénario.

Action	Description
Dialogue Définir les valeurs des propriétés	La dialogue d'attribution de valeurs de propriété peut être ouverte en cliquant avec le bouton droit sur une propriété et en sélectionnant « Fonctionnalités | Définir les valeurs de propriété », ou en utilisant le raccourci clavier Ctrl+Maj+R.
Attribuer une valeur	La dialogue « Définir les valeurs des propriétés » vous permet de définir des valeurs pour tout attribut défini dans la classe d'origine. Pour ce faire, sélectionnez

	la variable, définissez l'opérateur sur « = » et saisissez la valeur requise.

Définir Relations entre Propriétés

En plus de décrire les valeurs à attribuer aux variables appartenant à chaque propriété, un Statemachine Exécutable vous permet de définir comment chaque propriété peut référencer d'autres propriétés en fonction du modèle de classe dont elles sont des instances.

Action	Description
Créer un connecteur	Connectez plusieurs propriétés à l'aide de la relation Connecteur de la boîte à outils Composite.  Vous pouvez également utiliser le Quick Linker pour créer une relation entre deux Propriétés et sélectionner « Connecteur » comme type de relation.
Carte de la classe Modèle	Une fois qu'un connecteur existe entre deux propriétés, vous pouvez le mapper à l'association qu'il représente dans le modèle de classe. Pour ce faire, sélectionnez le connecteur et utilisez le raccourci clavier Ctrl+L. La dialogue « Choisir une association » s'affiche, ce qui permet à la Statemachine générée d'envoyer des signaux à l'instance remplissant le rôle spécifié dans la relation pendant l'exécution.

Génération de code pour Statemachines Exécutables

Le code généré pour un Statemachine Exécutable est basé sur sa propriété de langage. Il peut s'agir de Java, C, C++, C# ou JavaScript . Quel que soit le langage utilisé, Enterprise Architect génère le code approprié, qui est immédiatement prêt à être construit et exécuter . Aucune intervention manuelle n'est nécessaire avant de l' exécuter . En fait, après la génération initiale, tout Statemachine Exécutable peut être généré, construit et exécuté en un clic.

Langue prise en charge

Un Statemachine Exécutable supporte la génération de code pour ces langages de plateforme :

- C/C++ natif de Microsoft
- Microsoft .NET (C#)
- Scriptant (JavaScript)
- Oracle Java (Java)

À partir de la version 14.1 Enterprise Architect , la génération de code est prise en charge sans dépendance vis-à-vis de l'environnement de simulation (compilateurs). Par exemple, si vous n'avez pas installé Visual Studio, vous pouvez toujours générer du code à partir du modèle et l'utiliser dans votre propre projet. les compilateurs sont toujours nécessaires si vous souhaitez simuler des modèles dans Enterprise Architect .

Environnement Simulation (paramètres Compilateur)

Si vous souhaitez simuler le modèle Statemachine Exécutable dans Enterprise Architect , ces plateformes ou compilateurs sont requis pour les langages :

Plateforme linguistique	Exemple de chemin d'accès au cadre
Microsoft natif (C/C++)	C:\Program Files (x86)\Microsoft Visual Studio 12.0 C:\Program Files (x86)\Microsoft Visual Studio\2017\ Professional (ou autres éditions)
Microsoft .NET (C#)	C:\ Windows \Microsoft.NET\Framework\v3.5 (ou supérieur)
Scriptant (JavaScript)	N / A
Oracle Java (Java)	C:\Program Files (x86)\Java\jdk1.7.0_17 (ou supérieur)

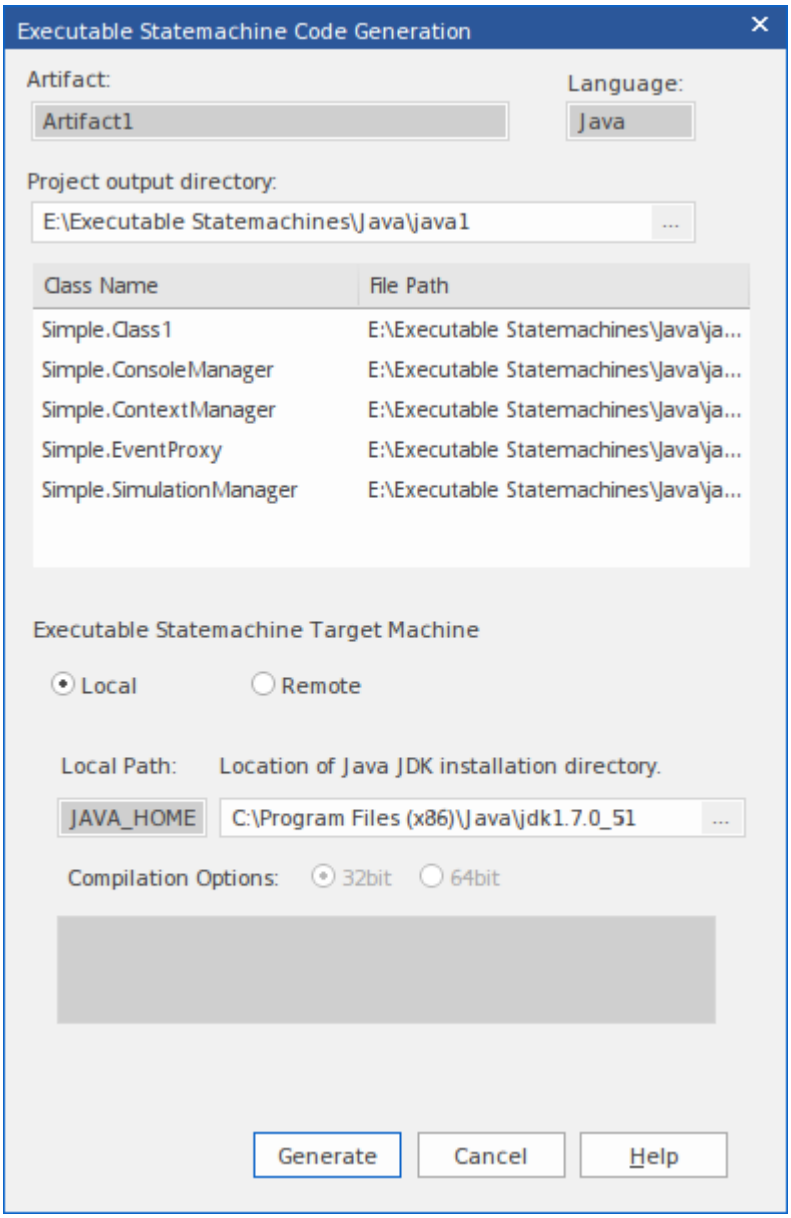
Accéder

Ruban	Simuler > States Exécutables > Statemachine > Générer , Build et Exécuter ou Simuler > States Exécutables > Statemachine > Générer

Génération de code

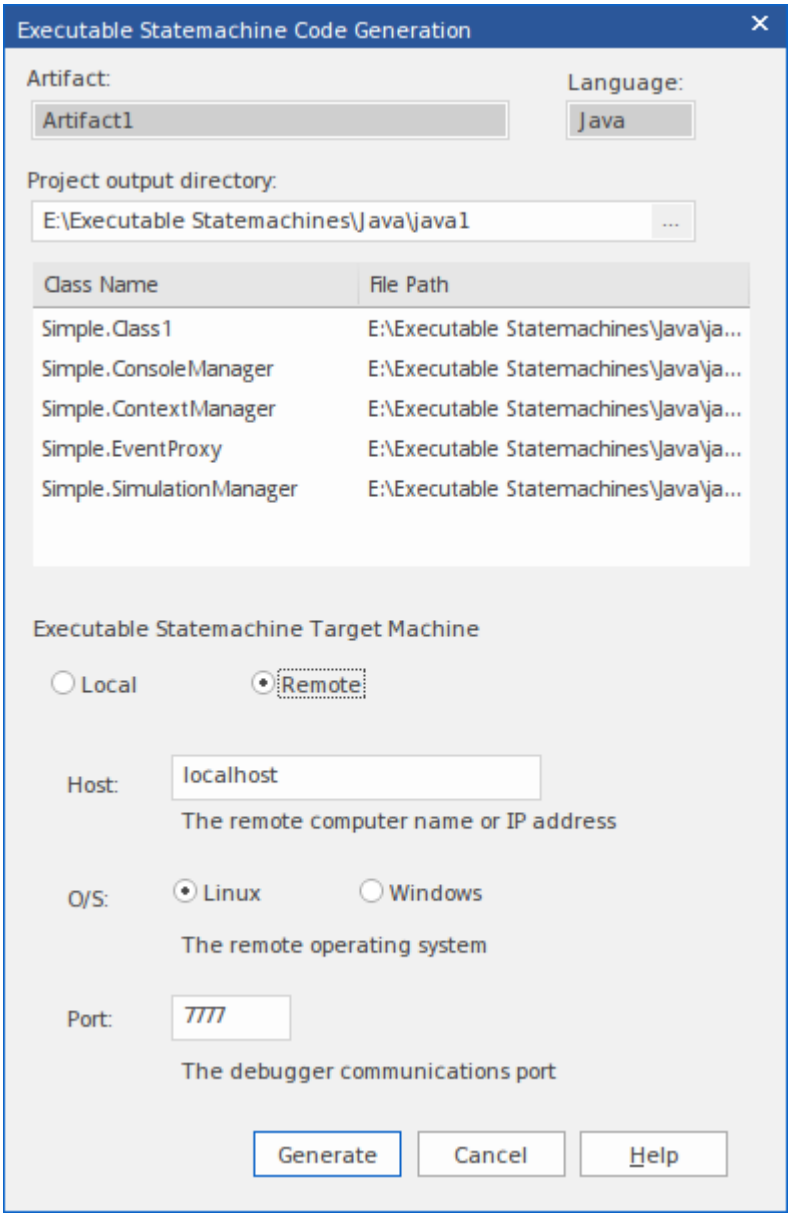
Les options du ruban « Simuler > States Exécutables > Statemachine » fournissent des commandes pour générer du code pour la Statemachine . Sélectionnez d'abord l'artefact Statemachine Exécutable , puis utilisez l'option du ruban pour générer le code. La dialogue « Génération de code de Statemachine exécutable » affichée dépend du langage de code.

Génération de code (Java sur Windows)



Répertoire de sortie du projet	Affiche le répertoire dans lequel les fichiers de code générés seront stockés. Si nécessaire, cliquez sur le bouton à droite du champ pour rechercher et sélectionner un autre répertoire. Les noms des classes générées et leurs chemins d'accès aux fichiers sources s'affichent ensuite.
Machine Statemachine exécutable Machine cible	Sélectionnez l'option « Local ».
JDK Java	Entrez le répertoire d'installation du JDK Java à utiliser.

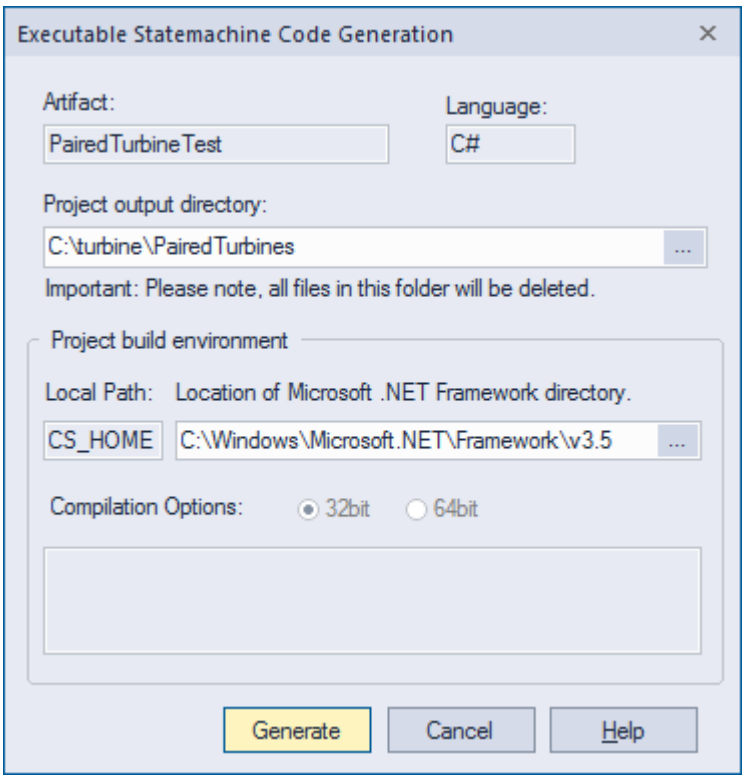
Génération de code (Java sur Linux)



Répertoire de sortie du projet :	Affiche le répertoire dans lequel les fichiers de code générés seront stockés. Si nécessaire, cliquez sur le bouton à droite du champ pour rechercher et sélectionner un autre répertoire. Les noms des classes générées et leurs chemins d'accès aux fichiers sources s'affichent lorsque le chemin est modifié.
Machine Statemachine exécutable Machine cible	Sélectionnez l'option « À distance ».
Système opérateur	Sélectionnez Linux.
Port	Il s'agit du port de débogage à utiliser. Vous trouverez des références à ce numéro

	de port dans les sections ' Déboguer ' et 'DebugRun' du script d'analyse généré.

Génération de code (autres langages)



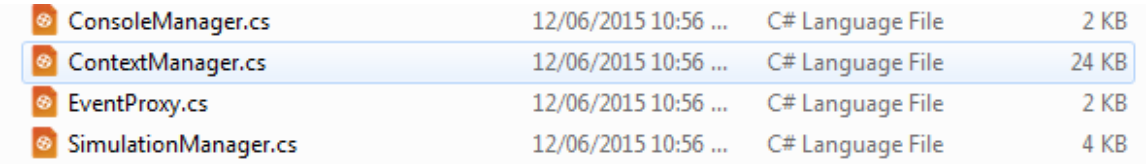
En même temps, la fenêtre Sortie Système s'ouvre sur la page ' Sortie Statemachine Exécutable ', sur laquelle sont affichés les messages de progression, les avertissements ou les erreurs pendant la génération du code.

Dans la dialogue « Génération de code Statemachine Exécutable », les champs « Artefact » et « Langue » affichent le nom de l'élément et la langue de codage tels que définis dans la dialogue « Propriétés » de l'élément.

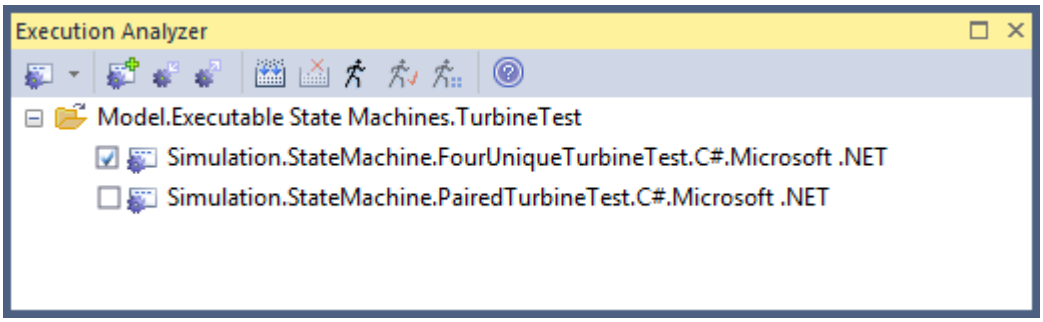
Champ/Option	Description
Répertoire de sortie du projet	Affiche le répertoire dans lequel les fichiers de code générés seront stockés. Si nécessaire, cliquez sur le bouton à droite du champ pour rechercher et sélectionner un autre répertoire.
Environnement de construction du projet	Les champs et informations de ce panneau varient en fonction de la langue définie dans l'élément Artifact et dans le script. Cependant, chaque langue prise en charge fournit une option permettant de définir le chemin d'accès aux frameworks cibles requis pour créer et exécuter le code généré. Des exemples sont présentés dans la section *Langues prises en charge* de cette rubrique. Ce chemin et son ID de chemins locaux sont définis dans la dialogue « Chemins locaux » et affichés ici dans la dialogue « Génération de code Statemachine Exécutable ».

Générer

Cliquez sur ce bouton pour générer le code Statemachine . La génération de code écrasera tous les fichiers existants dans le répertoire de sortie du projet. L'ensemble de fichiers comprendra tous les fichiers requis, y compris ceux de chaque classe référencée par la Statemachine .



Chaque Statemachine Exécutable généré générera également un script Analyseur d'Exécution , qui est le script de configuration pour la construction, l'exécution et le débogage de l' Statemachine Exécutable .



Code du bâtiment

Le code généré par un Statemachine Exécutable peut être construit par Enterprise Architect de trois manières.

Méthode	Description
Ribbon Générer , Build et Exécuter Commande	Pour l' Statemachine Exécutable sélectionné, génère à nouveau l'intégralité de la base de code. Le code source est ensuite compilé et la simulation démarrée.
Commande de création de ruban	Compile le code qui a été généré. Cette fonction peut être utilisée directement après la génération du code, si vous avez apporté des modifications à la procédure de construction (le script Analyzer) ou modifié le code généré d'une manière ou d'une autre.
Analyseur d'Exécution de Script	Le script Analyseur d'Exécution généré inclut une commande permettant de construire le code source. Cela signifie que lorsqu'il est actif, vous pouvez le construire directement en utilisant le raccourci intégré Ctrl+Maj+F12.
Générer le résultat	Lors de la construction, toutes les sorties sont affichées sur la page « Build » de la fenêtre Sortie système. Vous pouvez double-cliquer sur les erreurs du compilateur pour ouvrir un éditeur de code source sur la ligne appropriée.

Exploiter le code existant

Statemachines Exécutables générés et exécutés par Enterprise Architect peuvent exploiter le code existant pour lequel aucun modèle de classe n'existe. Pour ce faire, vous devez créer un élément de classe abstrait nommant uniquement les opérations à appeler dans la base de code externe. Vous devez ensuite créer une généralisation entre cette interface et la classe Statemachine , en ajoutant manuellement les liens requis dans le script Analyzer. Pour Java, vous pouvez ajouter des fichiers .jar au chemin de classe. Pour le code natif, vous pouvez ajouter un .dll au lien.

Débogage de l'exécution des Statemachines Exécutables

La création d' Statemachines Exécutables offre des avantages même après la génération du code. Grâce à l' Analyseur d'Exécution , Enterprise Architect est en mesure de se connecter au code généré. Vous pouvez ainsi déboguer visuellement et vérifier le comportement correct du code ; le même code exactement généré à partir de vos Statemachines , démontré par la simulation et finalement incorporé dans un système réel.

Débogage d'une Statemachine

Être capable de déboguer un Statemachine Exécutable offre des avantages supplémentaires, tels que la possibilité de :

- Interrompre l'exécution de la simulation et de toutes Statemachines en cours d'exécution
- Vue l'état brut de chaque instance Statemachine impliquée dans la simulation
- Vue le code source et Pile d'Appel à tout moment
- Tracez des informations supplémentaires sur l'état d'exécution grâce au placement de points de trace sur les lignes de code source
- Contrôler l'exécution grâce à l'utilisation de points d'action et de points d'arrêt (arrêt en cas d'erreur, par exemple)
- Diagnostiquer les changements de comportement, dus à des changements de code ou modélisation

Si vous avez généré, construit et exécuter un Statemachine Exécutable avec succès, vous pouvez le déboguer ! Le script Analyzer créé pendant le processus de génération est déjà configuré pour fournir le débogage. Pour démarrer le débogage, lancez simplement l'exécution de l' Statemachine Exécutable à l'aide du contrôle Simulation . Cependant, selon la nature du comportement à déboguer, nous définirons probablement d'abord des points d'arrêt.

Interrompre l'exécution lors d'une transition d'état

Comme tout débogueur, nous pouvons utiliser des points d'arrêt pour examiner la Statemachine en cours d'exécution à un point du code. Localisez une classe d'intérêt dans la fenêtre diagramme ou Navigateur et appuyez sur F12 pour afficher le code source. Il est facile de localiser le code des transitions State à partir des conventions de nommage utilisées pendant la génération. Si vous souhaitez effectuer une pause à une transition particulière, localisez la fonction de transition dans l'éditeur et placez un marqueur de point d'arrêt en cliquant dans la marge gauche sur une ligne de la fonction. Lorsque vous exécuter l' Statemachine Exécutable , le débogueur s'arrêtera à cette transition et vous pourrez afficher l'état brut des variables pour toutes Statemachines impliquées.

Interrompre l'exécution conditionnellement

Chaque point d'arrêt peut prendre une condition et une instruction trace. Lorsque le point d'arrêt est rencontré et que la condition est évaluée à True, l'exécution s'arrête. Sinon, l'exécution continue normalement. Vous composez la condition en utilisant les noms des variables brutes et en les comparant à l'aide des opérandes d'égalité standard : < > = >= <=. Par exemple :

(this.m_nCount > 100) et (this.m_ntype == 1)

Pour ajouter une condition à un point d'arrêt que vous avez défini, cliquez-droit sur le point d'arrêt et sélectionnez ' Propriétés '. En cliquant sur le point d'arrêt tout en appuyant sur la touche Ctrl, les propriétés peuvent être rapidement éditées.

Information auxiliaire Traçage

Il est possible de tracer des informations à partir de la Statemachine elle-même en utilisant la clause TRACE dans, par exemple, un *effet* . Le débogage fournit également fonctionnalités de trace appelées Tracepoints. Il s'agit simplement de points d'arrêt qui, au lieu de s'arrêter, impriment des instructions de trace lorsqu'ils sont rencontrés. La sortie est affichée dans la fenêtre Contrôle Simulation . Ils peuvent être utilisés comme une aide au diagnostic pour afficher et prouver la séquence d'événements et l'ordre dans lequel les instances changent d'état.

Visite de la Pile d'Appel

Chaque fois qu'un point d'arrêt est rencontré, la Pile d'Appel est disponible dans le menu Analyseur. Utilisez-la pour déterminer la séquence dans laquelle l'exécution a lieu.

Exécution et Simulation de Statemachines Exécutables

L'une des nombreuses fonctionnalités d' Enterprise Architect est sa capacité à effectuer des simulations. Un Statemachine Exécutable généré et intégré dans Enterprise Architect peut se connecter aux facilités Simulation pour démontrer visuellement l'exécution en direct de l'artefact Statemachine .

Démarrer une Simulation

La barre d'outils de contrôle Simulation fournit un bouton Rechercher que vous pouvez utiliser pour sélectionner l'artefact Statemachine Exécutable à exécuter . Le contrôle conserve une liste déroulante des Statemachines Exécutables les plus récents parmi lesquels vous pouvez faire votre choix. Vous pouvez également utiliser le menu contextuel d'un artefact Statemachine Exécutable lui-même pour lancer la simulation.

Contrôle de la vitesse

Le contrôle Simulation fournit un paramètre de vitesse. Vous pouvez l'utiliser pour ajuster la vitesse à laquelle la simulation s'exécute. La vitesse est représentée par une valeur entre 0 et 100 (une valeur plus élevée est plus rapide). Une valeur de zéro entraînera l'arrêt de la simulation après chaque étape ; cela nécessite d'utiliser les commandes de la barre d'outils pour parcourir manuellement la simulation.

Notation pour States Actif

Au fur et à mesure de l'exécution de l' Statemachine Exécutable , les diagrammes Statemachine concernés s'affichent. L'affichage est mis à jour à la fin de chaque cycle d'étape à étape. Vous remarquerez que seul l' State actif de l'instance qui termine une étape est mis en surbrillance. Les autres States restent grisés.

Il est facile d'identifier quelle instance se trouve dans quel State , car les States sont étiquetés avec le nom de toute instance actuellement dans cet état particulier. Si deux ou plusieurs propriétés d'artefact du même type partagent le même State , l' State aura une étiquette distincte pour chaque nom de propriété.

Générer Diagramme de temps

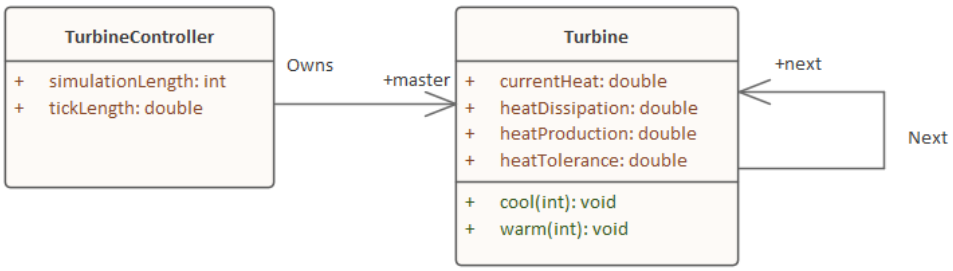
Après avoir terminé la simulation d'un Statemachine Exécutable , vous pouvez générer un diagramme de temps à partir de la sortie. Pour cela :

Dans la barre d'outils de la fenêtre Simulation , cliquez sur « Outils | Générer Diagramme de synchronisation ».

Exemple : Statemachine Exécutable

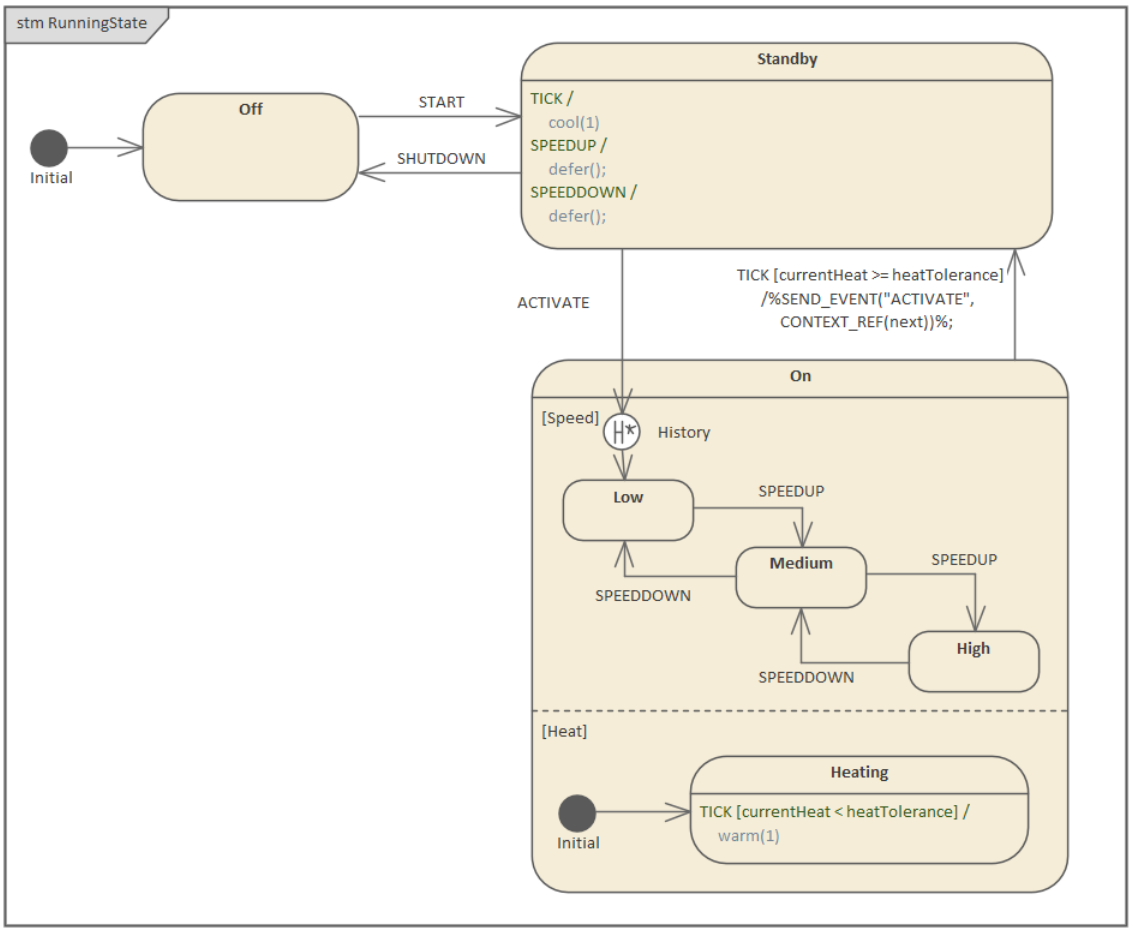
Exemple de classe Modèle

Cette image montre un exemple de modèle de classe utilisé par les Statemachines décrites dans cette rubrique.

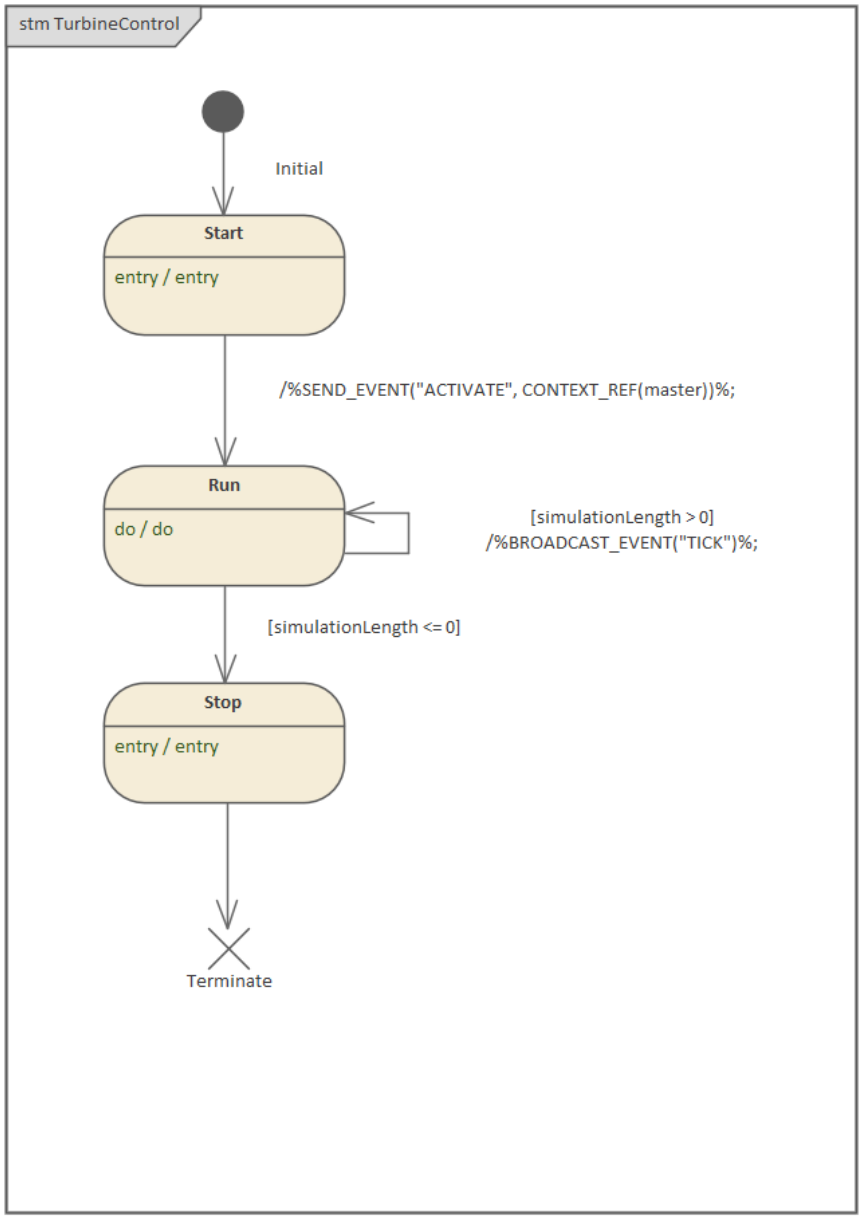


Exemples Statemachines

Ces deux diagrammes montrent les définitions de deux Statemachines . La première référence une autre Statemachine du même type, tandis que la seconde pilote toutes les instances de la première qui existent.

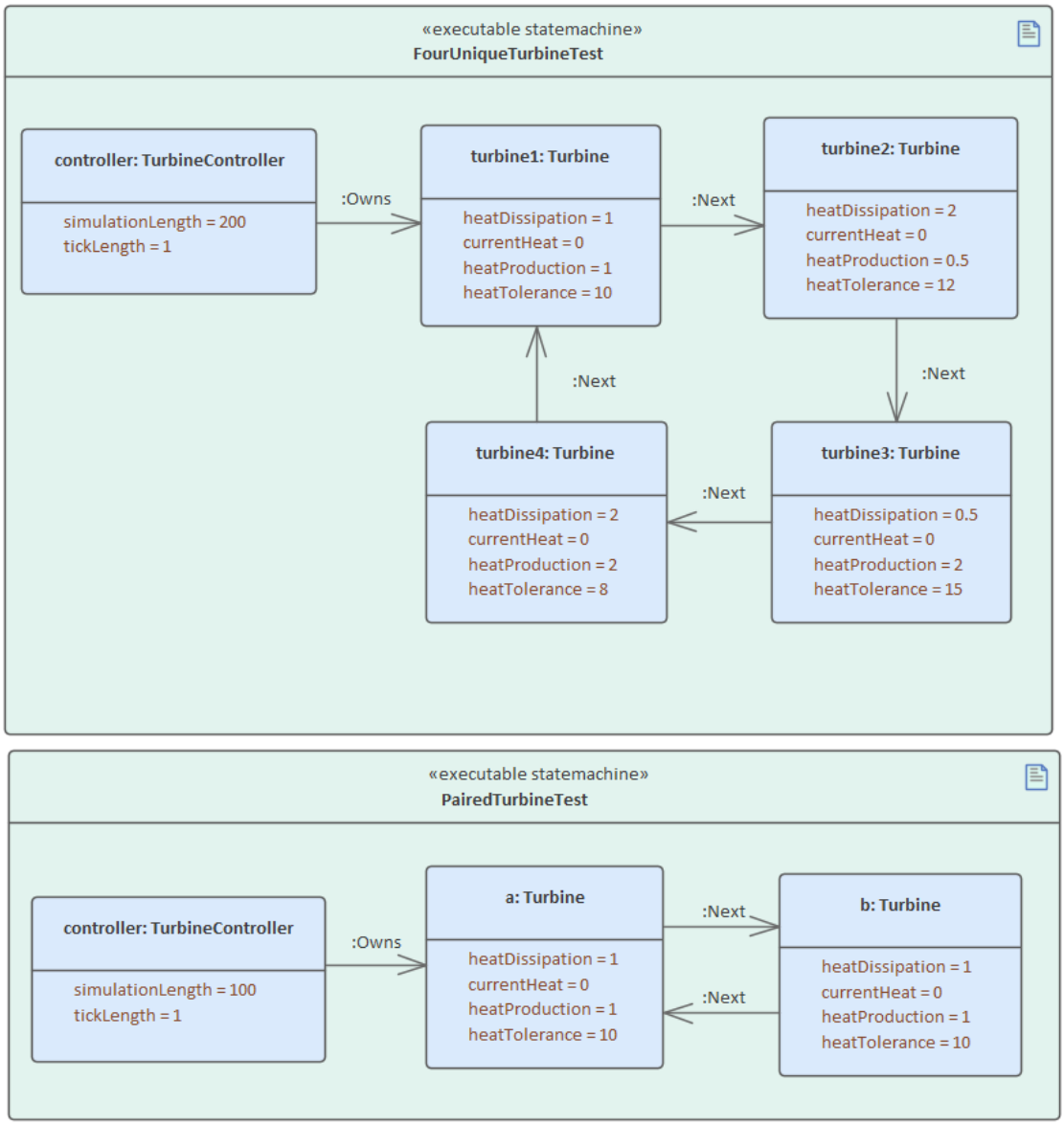


Le contrôleur de niveau supérieur.



Exemples d'artefacts

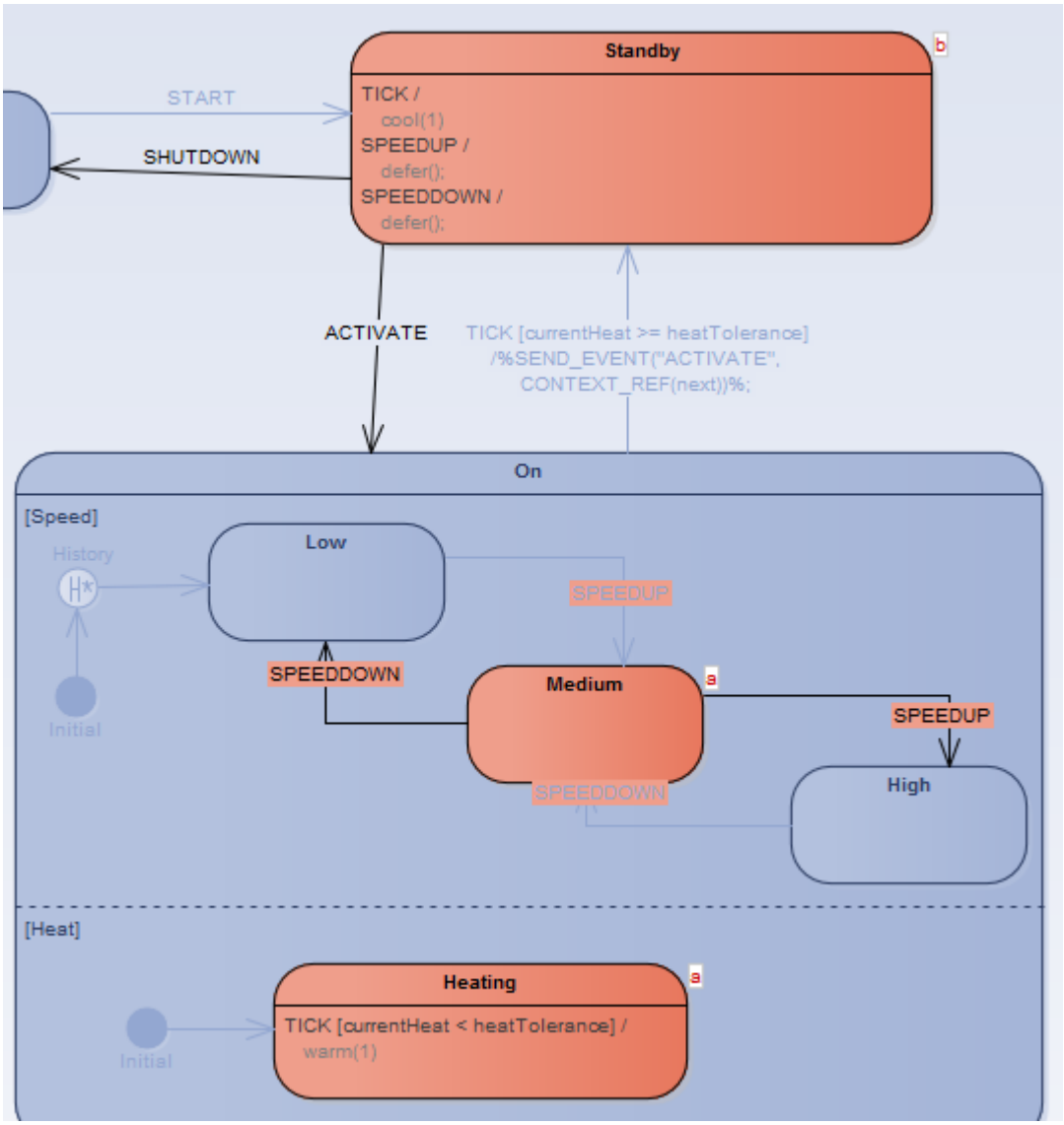
A partir des exemples diagrammes de classes et Statemachine , nous pouvons créer Statemachines Exécutables comme indiqué ici.



Note comment les valeurs de propriété ont été définies pour chaque propriété et les liens entre les éléments identifient les relations qui existent dans le modèle de classe.

Résultats Simulation

Lors de l'exécution d'une simulation, Enterprise Architect met en évidence les States actuellement actifs dans toutes Statemachines . Lorsque plusieurs instances d'une Statemachine existent, il affiche également les noms de chaque instance dans cet State .


Standby
b
START
SHUTDOWN
TICK /
cool(1)
SPEEDUP /
defer();
SPEEDDOWN /
defer();
ACTIVATE
TICK [currentHeat >= heatTolerance]
/%SEND_EVENT("ACTIVATE",
CONTEXT_REF(next))%;
On
[Speed]
History
Low
SPEEDUP
SPEEDDOWN
Medium
a
SPEEDUP
High
SPEEDDOWN
Initial
[Heat]
Heating
a
TICK [currentHeat < heatTolerance] /
warm(1)
Initial

Exemple : Commandes de Simulation

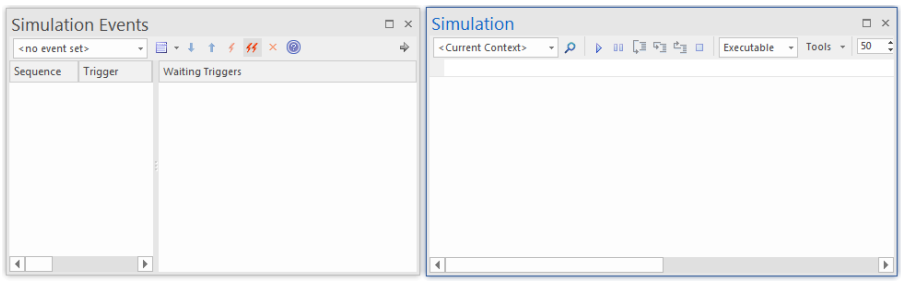
Cet exemple montre comment utiliser la fenêtre Simulation pour observer les messages Trace ou envoyer des commandes pour contrôler une Statemachine . Grâce à cet exemple, vous pouvez examiner :

- Un attribut d'un contexte - la variable membre définie dans la classe, qui est le contexte de la Statemachine ; ces attributs portent des valeurs dans la portée du contexte pour tous les comportements State et les effets de transition, pour accéder et modifier
- Chaque attribut d'un signal - la variable membre définie dans le signal, qui est référencée par un événement et peut servir de paramètre d'événement ; chaque occurrence d'événement de signal peut avoir différentes instances d'un signal
- L'utilisation de la commande « Eval » pour interroger la valeur d'exécution d'un attribut de contexte
- L'utilisation de la commande « Dump » permet de vider le nombre d'événements actifs de l'état actuel ; elle peut également vider l'événement actuel différé dans le pool

Cet exemple est tiré du modèle EAExample :

Exemple Modèle . Simulation de Modèle . Statemachine Exécutable . Commandes de Simulation

Accéder

Ruban	• Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation) • Simuler > Simulation Dynamique > Événements (pour la fenêtre Simulation Événements)  Ces deux fenêtres sont fréquemment utilisées ensemble dans la simulation de Statemachines Exécutables .

Créer un contexte et Statemachine

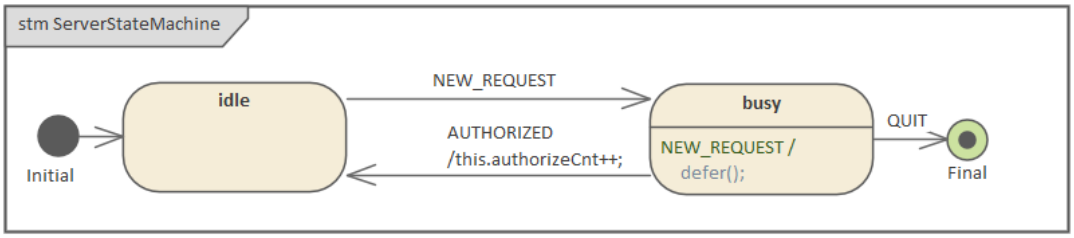
Dans cette section, nous allons créer une classe appelée TransactionServer, qui définit un Statemachine comme son comportement. Nous créons ensuite un Statemachine Exécutable Artifact comme environnement de simulation.

Créer le contexte de la Statemachine

1. Créez un élément de classe appelé *TransactionServer*.

2. Dans cette classe, créez un attribut appelé *authorizeCnt* avec valeur initiale 0.
3. Dans la fenêtre Navigateur , cliquez-droit sur *TransactionServer* et sélectionnez l'option 'Ajouter | Statemachine '.

Créer la Statemachine



1. Créez un pseudo-état initial appelé *Initial* .
2. Transition vers un State appelé *inactif* .
3. Transition vers un State appelé *busy* , avec le déclencheur NEW_REQUEST.
4. Transition:
   - Vers un pseudo-état Final appelé *Final* , avec le déclencheur QUIT
   - Retour au *repos* , avec le déclencheur AUTORISÉ, avec l'Effet 'this.authorizeCnt++;'

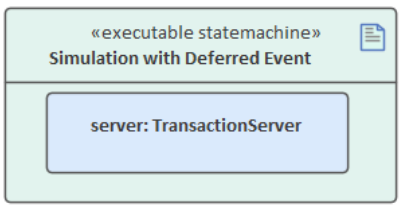
Créer un événement différé pour l' State *occupé*

1. Dessinez une auto-transition pour *être occupé.*
2. Modifiez le « type » de transition en « interne ».
3. Spécifiez le Déclencheur comme étant l'événement que vous souhaitez différer.
4. Dans le champ « Effet », saisissez « defer(); ».

Créer un signal et Attributes

1. Créez un élément Signal appelé *RequestSignal.*
2. Créez un attribut appelé *requestType* avec le type ' int '.
3. Configurez l'événement NEW_REQUEST pour référencer *RequestSignal.*

Créer l'artefact Statemachine Exécutable

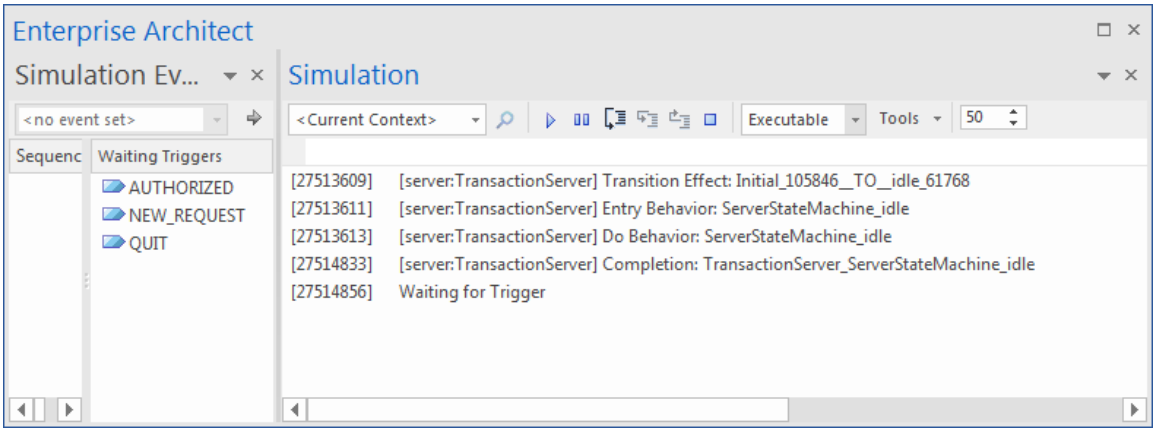


1. Depuis la page « Simulation » de la boîte à outils Diagramme , faites glisser une icône « Statemachine Exécutable » sur le diagramme et appelez l'élément *Simulation avec événement différé.*
2. Ctrl+Glissez l'élément *TransactionServer* depuis la fenêtre Navigateur et déposez-le sur l'Artefact en tant que propriété, avec le *serveur de noms.*
3. Définissez la langue de l'Artefact sur JavaScript , qui ne nécessite pas de compilateur (pour l'exemple ; en production, vous pouvez également utiliser C, C++, C# ou Java, qui support Statemachines Exécutables).
4. Cliquez sur l'artefact et sélectionnez l'option de ruban 'Simulate > States Exécutables > Statemachine > Générer , Build and Exécuter '.

Fenêtre Simulation et commandes

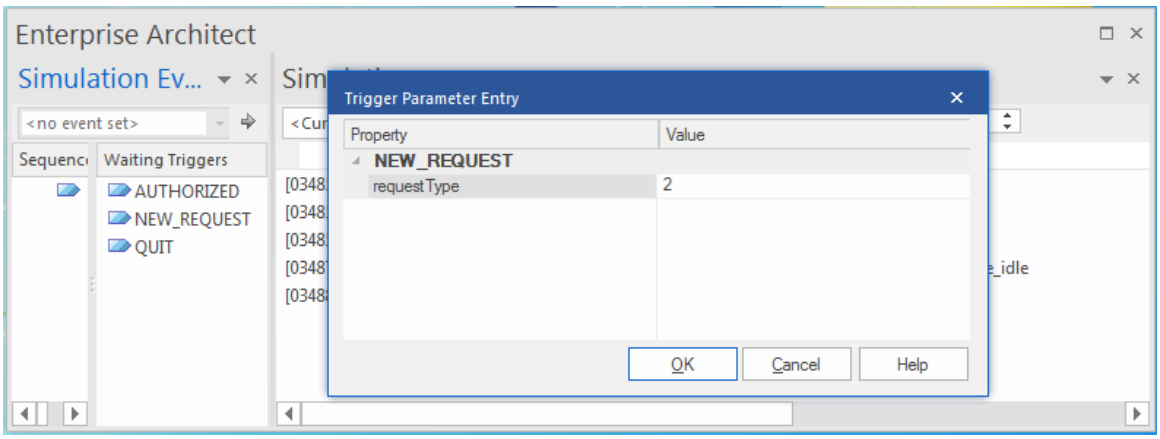
Lorsque la simulation démarre, l'état actuel est *inactif* .

La fenêtre Simulation montre que l'effet de transition, l'entrée et le comportement Do sont terminés pour l'état *inactif* et que la Statemachine attend un déclencheur .



Données d'événement via des valeurs pour Attributes de signal

Pour l'événement de signal de Déclencheur NEW_REQUEST, la dialogue « Entrée du paramètre Déclencheur » s'affiche pour prompt des valeurs pour les attributs répertoriés définis dans le signal *RequestSignal* , référencé par NEW_REQUEST.



Type la valeur '2' et cliquez sur le bouton OK . Les valeurs de l'attribut Signal sont ensuite transmises aux méthodes invoquées telles que les comportements de State et les effets de la Transition.

Ces messages sont affichés dans la fenêtre Simulation :

[03612562] En attente du Déclencheur

[03611358] Commande : **diffusion NEW_REQUEST.RequestSignal(2)**

[03611362] [serveur : TransactionServer] Événement mis en file d'attente : NEW_REQUEST.RequestSignal(requestType : 2)

[03611367] [serveur : TransactionServer] Événement envoyé : NEW_REQUEST.RequestSignal(requestType : 2)

[03611371] [serveur : TransactionServer] Comportement de sortie : ServerStateMachine_idle

[03611381] [serveur : TransactionServer] Effet de transition : idle__TO__busy_61772

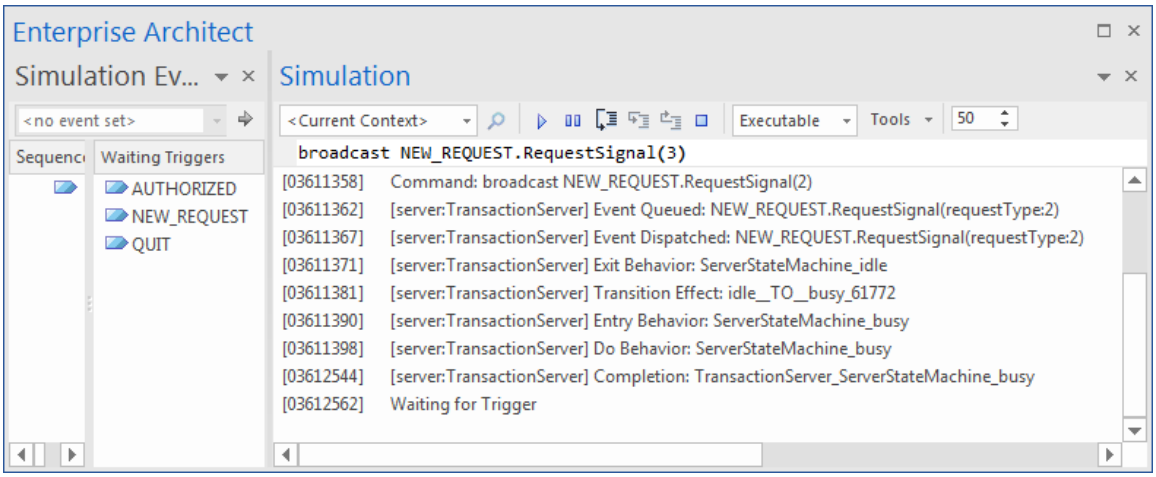
[03611390] [serveur : TransactionServer] Comportement de l'entrée : ServerStateMachine_busy

[03611398] [serveur : TransactionServer] Comportement : ServerStateMachine_busy

[03612544] [serveur : TransactionServer] Achèvement : TransactionServer_ServerStateMachine_busy

[03612562] En attente du Déclencheur

Nous pouvons diffuser des événements en double-cliquant sur l'élément listé dans la fenêtre Simulation Événements . Alternativement, nous pouvons saisir une string de commande dans le champ texte de la fenêtre Simulation (sous la barre d'outils).



[03612562] En attente du Déclencheur

[04460226] Commande : **diffusion NEW_REQUEST.RequestSignal(3)**

[04460233] [serveur : TransactionServer] Événement mis en file d'attente : NEW_REQUEST.RequestSignal(requestType : 3)

[04461081] En attente du Déclencheur

Le message Simulation indique que l'occurrence de l'événement est différée (événement mis en file d'attente, mais non envoyé). Nous pouvons exécuter d'autres commandes à l'aide du champ de texte :

[04655441] En attente du Déclencheur

[04664057] Commande : **diffusion NEW_REQUEST.RequestSignal(6)**

[04664066] [serveur : TransactionServer] Événement mis en file d'attente : NEW_REQUEST.RequestSignal(requestType : 6)

[04664803] En attente du Déclencheur

[04669659] Commande : **diffusion NEW_REQUEST.RequestSignal(5)**

[04669667] [serveur : TransactionServer] Événement mis en file d'attente : NEW_REQUEST.RequestSignal(requestType : 5)

[04670312] En attente du Déclencheur

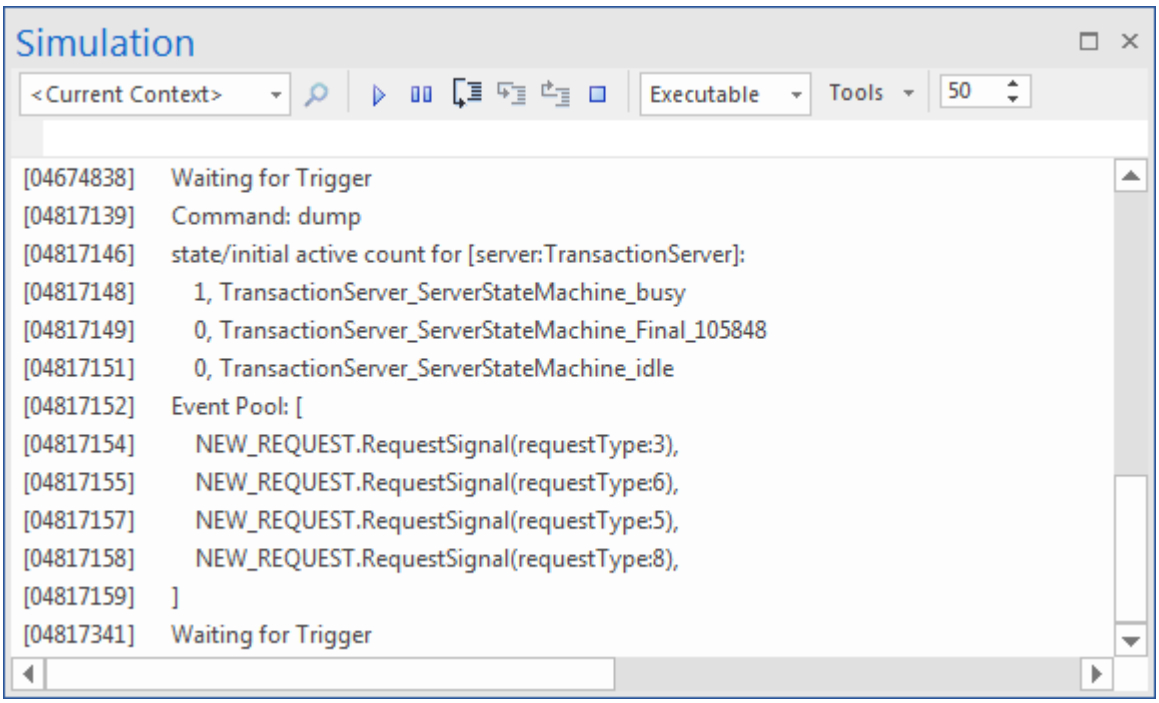
[04674196] Commande : **diffusion NEW_REQUEST.RequestSignal(8)**

[04674204] [serveur : TransactionServer] Événement mis en file d'attente : NEW_REQUEST.RequestSignal(requestType : 8)

[04674838] En attente du Déclencheur

dump : Query « nombre actif » pour un pool State et d'événements

Type *dump* dans le champ de texte ; ces résultats s'affichent :



Dans la section « nombre actif », nous pouvons voir que *occupé* est l'état actif (le nombre actif est 1).

Conseils : Pour un State Composite, le nombre actif est de 1 (pour lui-même) *plus* le nombre de régions actives.

Dans la section « Pool d'événements », nous pouvons voir qu'il existe quatre occurrences d'événements dans la file d'attente des événements. Chaque instance du signal contient des données différentes.

L'ordre des événements dans le pool est l'ordre dans lequel ils sont diffusés.

eval : Query Exécuter Time Valeur du Contexte

Déclencheur AUTORISÉ,

[04817341] En attente du Déclencheur

[05494672] Commande : diffusion AUTORISÉE

[05494678] [serveur : TransactionServer] Événement mis en file d'attente : AUTORISÉ

[05494680] [serveur : TransactionServer] Événement envoyé : AUTHORIZED

[05494686] [serveur : TransactionServer] Comportement de sortie : ServerStateMachine_busy

[05494686] [serveur : TransactionServer] **Effet de transition : busy__TO__idle_61769**

[05494687] [serveur : TransactionServer] Comportement de l'entrée : ServerStateMachine_idle

[05494688] [serveur : TransactionServer] Comportement : ServerStateMachine_idle

[05495835] [serveur : TransactionServer] Achèvement : TransactionServer_ServerStateMachine_idle

[05495842] [serveur : TransactionServer] **Événement envoyé : NEW_REQUEST.RequestSignal(requestType : 3)**

[05495844] [serveur : TransactionServer] Comportement de sortie : ServerStateMachine_idle

[05495846] [serveur : TransactionServer] Effet de transition : idle__TO__busy_61772

[05495847] [serveur : TransactionServer] Comportement de l'entrée : ServerStateMachine_busy

[05495850] [serveur : TransactionServer] Comportement : ServerStateMachine_busy

[05496349] [serveur : TransactionServer] Achèvement : TransactionServer_ServerStateMachine_busy

[05496367] En attente du Déclencheur

- La transition de *l'état occupé* à *l'état inactif* est effectuée, nous nous attendons donc à ce que l'effet soit exécuté
- Un événement est rappelé du pool et envoyé lorsque *l'inactivité* est terminée, ce qui fait que l' *état occupé* devient l'état actif
- Type *dump* et remarquez qu'il reste trois événements dans le pool ; le premier est rappelé et envoyé

[05693348] Pool d'événements : [

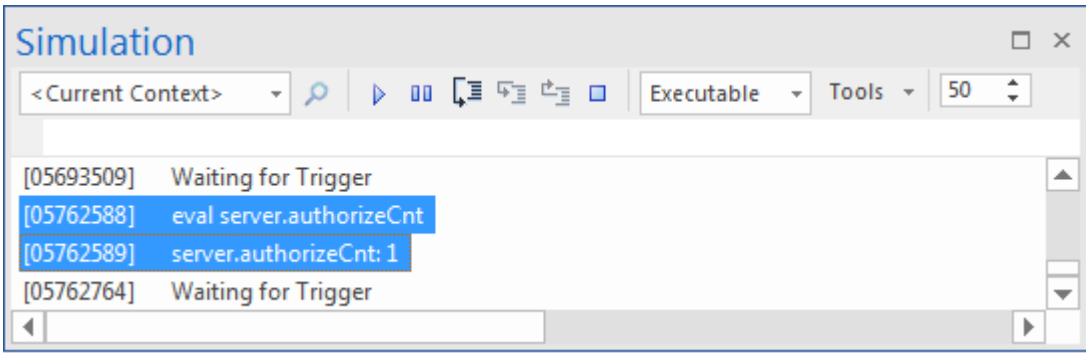
[05693349] NEW_REQUEST.RequestSignal(requestType:6),

[05693351] NEW_REQUEST.RequestSignal(requestType:5),

[05693352] NEW_REQUEST.RequestSignal(requestType:8),

[05693354]]

Type *eval server.authorizeCnt* dans le champ de texte. Ce chiffre indique que la valeur du temps exécuter de 'server.authorizeCnt' est 1.



Déclencheur AUTHORIZED à nouveau. Lorsque le Statemachine est stable à *busy* , il restera deux événements dans le pool. Exécuter *eval server.suthorizeCnt* à nouveau ; la valeur sera 2.

Accéder à la variable membre du contexte à partir du comportement State et de l'effet de transition

Statemachine Exécutable d' Enterprise Architect supporte la simulation pour C, C++, C# , Java et JavaScript .

Pour C et C++, la syntaxe diffère de C# , Java et JavaScript pour accéder aux variables membres du contexte. C et C++ utilisent le pointeur '->' tandis que les autres utilisent simplement '.'; cependant, vous pouvez toujours utiliser *this.variableName* pour accéder aux variables. Enterprise Architect le traduira en this->variableName pour C et C++.

Donc, pour toutes les langues, utilisez simplement ce format pour la simulation :

cette.variableName

Exemples :

Dans l'effet de la transition :

ceci.autoriserCnt++;

Dans le comportement d'entrée, de sortie ou de sortie de certains États :

ceci.foo += ceci.bar;

Note : par défaut, Enterprise Architect remplace uniquement « this-> » par « this » pour C et C++ ; par exemple :

this.foo = this.bar + monObjet.iCount + monPointeur->iCount;

sera traduit par :

ceci->foo = ceci->bar + monObjet.iCount + monPointeur->iCount;

Une liste complète des commandes prises en charge

Étant donné que l'artefact Statemachine Exécutable peut simuler plusieurs contextes ensemble, certaines commandes peuvent spécifier un nom d'instance.

exécuter Statemachine :

Comme chaque contexte peut avoir plusieurs Statemachines , la commande ' exécuter ' peut spécifier une Statemachine avec laquelle démarrer.

- exécuter instance. statemachine
- exécuter tout.tout
- exécuter une instance
- exécuter tous
- exécuter

Par exemple:

exécuter

exécuter tous

exécuter le serveur

exécuter server.myMainStatemachine

diffuser et envoyer l'événement :

- chaîne d'événement de diffusion
- envoyer EventString à l'instance
- envoyer EventString (équivalent à diffuser EventString)

Par exemple:

diffusion Événement1

envoyer l'événement 1 au client

Commande dump :

- décharge
- vidage d'instance

Par exemple:

décharge

serveur de vidage

vider le client

Commande eval :

- évaluer l'instance.variableName

Par exemple:

évaluer client.requestCnt

évaluer le serveur.responseCnt

Commande de sortie :

- sortie

Format de l'EventString :

- EventName.SignalName(liste d'arguments)

Note : la liste des arguments doit correspondre aux attributs définis dans le signal **par ordre** .

Par exemple, si le signal définit deux attributs :

- foo
- bar

Alors ces EventStrings sont valides :

- Événement1.Signal1(10, 5) --------- foo = 10; bar = 5
- Event1.Signal1(10,) --------- foo = 10; bar n'est pas défini
- Event1.Signal1(,5) --------- bar = 10; foo n'est pas défini
- Event1.Signal1(,) --------- foo et bar ne sont pas définis

Si le signal ne contient aucun attribut, nous pouvons simplifier l'EventString comme suit :

- Nom de l'événement

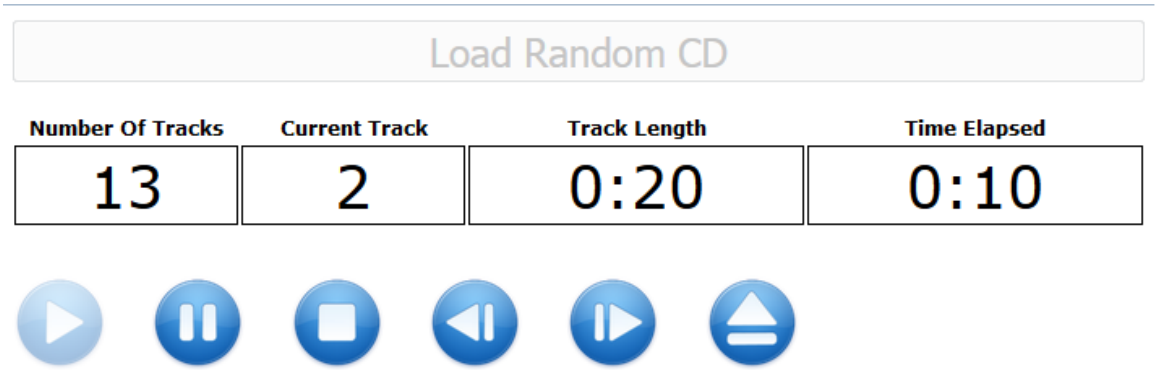
Exemple : Simulation en HTML avec JavaScript

Nous savons déjà que les utilisateurs peuvent modéliser un Statemachine Exécutable et le simuler dans Enterprise Architect avec le code généré. En utilisant les deux exemples *CD Player* et *Regular Expression Parser* , nous allons maintenant vous montrer comment vous pouvez intégrer le code généré à vos projets réels.

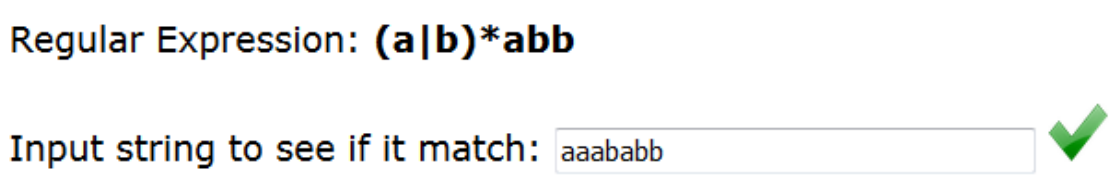
Enterprise Architect fournit deux mécanismes différents permettant au code client d'utiliser une Statemachine :

- Basé sur State Actif - le client peut interroger l'état actif actuel, puis « changer » la logique en fonction du résultat de la requête
- Variable d'exécution basée sur - le client n'agit pas sur l'état actif actuel, mais agit sur la valeur d'exécution des variables définies dans la classe contenant la Statemachine

Dans l'exemple *du lecteur CD* , il y a très peu d'états et beaucoup de boutons sur l'interface graphique, il est donc assez facile d'implémenter l'exemple basé sur le mécanisme State Actif ; nous interrogerons également la valeur d'exécution de la piste actuelle.



Dans l'exemple *Parser d'expressions régulières* , la Statemachine gère tout et une variable membre *bMatch* modifie sa valeur d'exécution lorsque les états changent. Le client n'enregistre pas le nombre d'états présents ni l'état actuellement actif.



Dans ces rubriques, nous démontrons comment modéliser, simuler et intégrer un lecteur CD et un Parser pour une expression régulière spécifiée, étape par étape :

- CD Player
- Regular Expression Parser

Lecteur CD

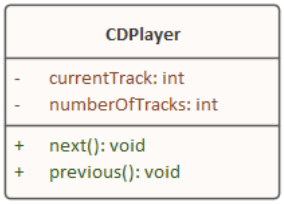
Le comportement d'une application de lecteur CD peut sembler intuitif ; cependant, il existe de nombreuses règles liées au moment où les boutons sont activés et désactivés, à ce qui est affiché dans les champs de texte de la fenêtre et à ce qui se passe lorsque vous fournissez des événements à l'application.

Supposons que notre exemple de lecteur CD possède ces fonctionnalités :

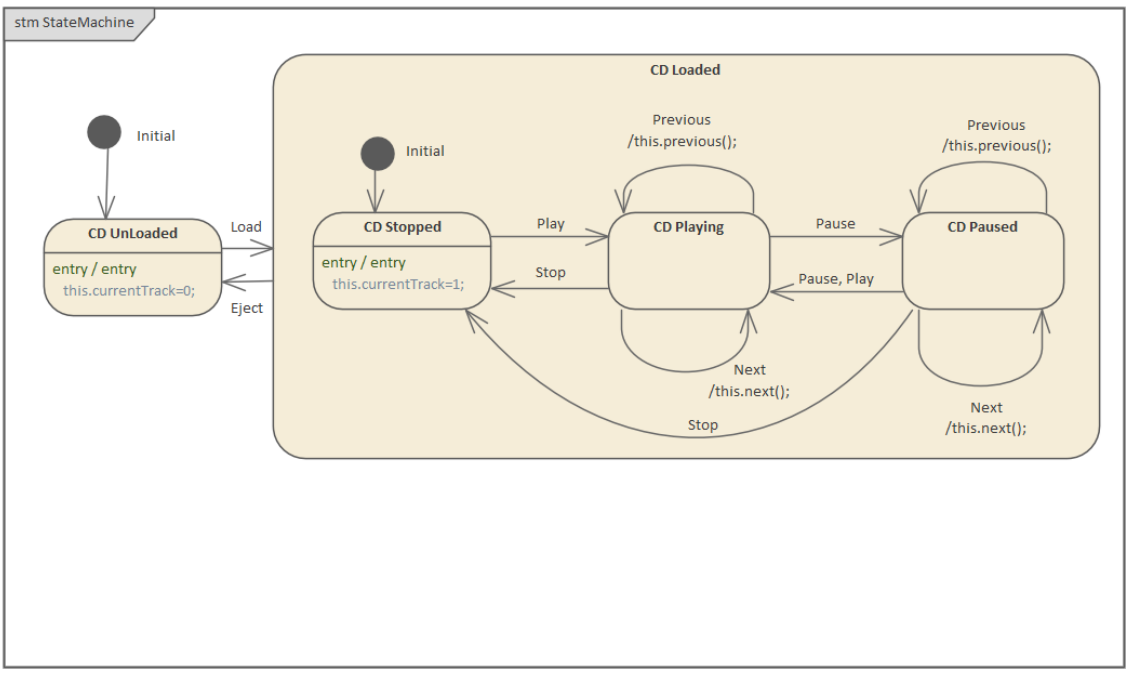
- Boutons - Charger un CD aléatoire, Lecture, Pause, Arrêt, Piste précédente, Piste suivante et Éjecter
- Affichages - Nombre de pistes, piste actuelle, durée de la piste et temps écoulé

Statemachine pour lecteur CD

Une classe *CDPlayer* est définie avec deux attributs : *currentTrack* et *numberOfTracks* .



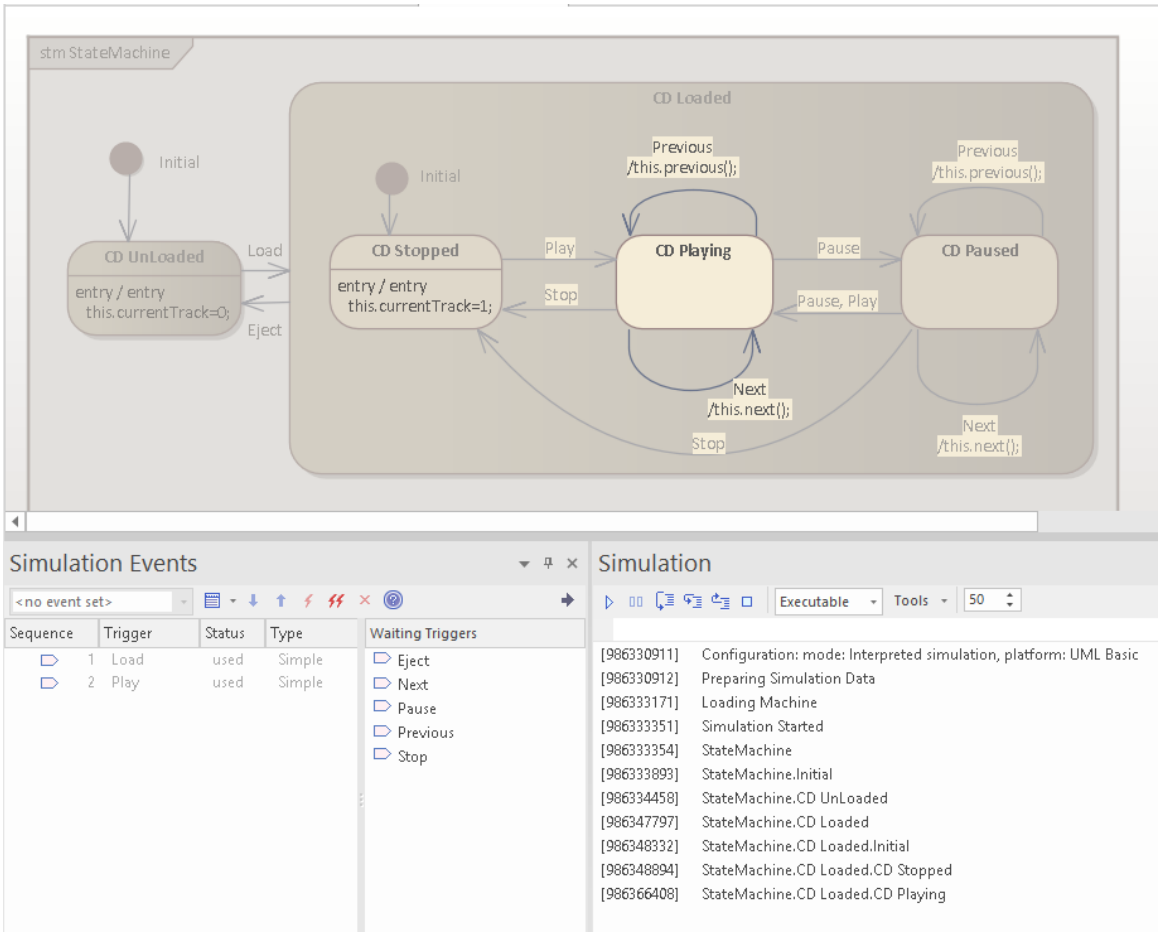
Une Statemachine est utilisée pour décrire les états du lecteur CD :



- Au niveau supérieur, la Statemachine a deux States : *CD non chargé* et *CD chargé*
- *Le CD chargé* peut être composé de trois States simples : *CD arrêté* , *CD en lecture* , *CD en pause*
- Les transitions sont définies avec déclencheurs pour les événements Load, Eject, Play, Pause, Stop, Previous et Next
- Les comportements State et les effets de transition sont définis pour modifier la valeur des attributs définis dans *CDPlayer* ; par exemple, l'événement 'Précédent' déclencheur l'auto-transition (si l'état actuel est *CD en lecture* ou *CD en pause*) et l'effet sera exécuté, ce qui décrémentera la valeur de *currentTrack* ou reviendra à la dernière piste

Nous pouvons créer un artefact Statemachine Exécutable et créer une propriété typant *CDPlayer* , puis simuler le

Statemachine dans Enterprise Architect pour nous assurer que le modèle est correct.



Inspecter le code généré

Enterprise Architect générera ces fichiers dans un dossier que vous avez spécifié :

- Code back-end : CDPlayer.js, ContextManager.js, EventProxy.js
- Code client : ManagerWorker
- Code frontal : statemachineGUI.js, index.html
- Autre code : SimulationManager.js

Déposer	Description
/CDPlayer.js	Ce fichier définit la classe CDPlayer ainsi que ses attributs et ses opérations. Il définit également les Statemachines de la classe avec les comportements State et les effets de transition.
/ContextManager.js	Ce fichier est le gestionnaire abstrait des contextes. Le fichier définit le contenu qui est indépendant des contextes réels, qui sont définis dans la généralisation du *ContextManager* , comme *SimulationManager* et *ManagerWorker* . La simulation (Statemachine Exécutable Artifact) peut impliquer plusieurs contextes ; par exemple, dans une simulation de jeu de tennis, il y aura un *arbitre* typé dans la classe *Arbitre,* et deux joueurs - *joueurA* et *joueurB* - typés dans la classe *Joueur.* La classe *Arbitre* et la classe *Joueur* définiront chacune leur(s) propre(s) Statemachine (s).

/EventProxy.js	Ce fichier définit Événements et les signaux utilisés dans la simulation. Si nous déclenchons un événement avec des arguments, nous modélisons l'événement comme un événement de signal, qui spécifie une classe de signal ; nous définissons ensuite des attributs pour la classe de signal. Chaque occurrence d'événement possède une instance du signal, contenant les valeurs d'exécution spécifiées pour les attributs.
/SimulationManager.js	Ce fichier est destiné à la simulation dans Enterprise Architect .
/html/ManagerWorker.js	Ce fichier sert de couche intermédiaire entre le front-end et le back-end. • Le front-end publie un message pour demander des informations au ManagerWorker • Étant donné que ManagerWorker généralise à partir de ContextManager, il a un accès complet à tous les contextes tels que l'interrogation de l'état actif actuel et l'interrogation de la valeur d'exécution d'une variable • Le ManagerWorker publiera un message sur le front-end avec les données qu'il a récupérées du back-end
/html/statemachineGUI.js	Ce fichier établit la communication entre le front-end et le ManagerWorker, en définissant *stateMachineWorker* . Il : • Définit les fonctions *startStateMachineWebWorker* et *stopStateMachineWebWorker* • Définit les fonctions *onActiveStateResponse* et *onRuntimeValueResponse* avec un code d'espace réservé : //à faire : écrire la logique de l'utilisateur Vous pouvez simplement remplacer ce commentaire par votre logique, comme cela sera démontré plus tard dans ce sujet.
/html/index.html	Ce fichier définit l' Interface Utilisateur , comme les boutons et les entrées permettant de déclencher Événements ou d'afficher des informations. Vous pouvez définir du CSS et JavaScript dans ce fichier.

Personnaliser index.html et statemachineGUI.js

Apportez ces modifications aux fichiers générés :

- Créer des boutons et des affichages
- Créer un style CSS pour formater l'affichage et activer/désactiver les images des boutons
- Créez un ElapseTimeWorker.js pour actualiser l'affichage toutes les secondes
- Créez une fonction TimeElapsed, définissez-la sur Next Track lorsque le temps écoulé atteint la longueur de la piste
- Créer JavaScript comme gestionnaire d'événements du bouton « onclick »
- Une fois qu'un événement est diffusé, demandez l' State actif et valeur d'exécution pour *cdPlayer.currentTrack*
- Lors de l'initialisation, demander l' State actif

Dans statemachineGUI.js, recherchez la fonction *onActiveStateResponse_cdPlayer*

- Dans CDPlayer_StateMachine_CDUnLoaded, désactivez tous les boutons et activez btnLoad
- Dans CDPlayer_StateMachine_CDLoaded_CDStopped, désactivez tous les boutons et activez btnEject et btnPlay
- Dans CDPlayer_StateMachine_CDLoaded_CDPlaying, activez tous les boutons et désactivez btnLoad et btnPlay

- Dans CDPlayer_StateMachine_CDLoaded_CDPaused, activez tous les boutons et désactivez btnLoad

Dans statemachineGUI.js, recherchez la fonction *onRuntimeValueResponse*

- Dans *cdPlayer.currentTrack* , nous mettons à jour l'affichage de la piste actuelle et de sa longueur

L'exemple complet

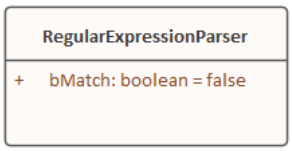
L'exemple est accessible depuis la page « Ressources » du site Web Sparx Systems , en cliquant sur ce lien :
CD Player Simulation

Cliquez sur le bouton Charger un CD aléatoire, puis sur le bouton Démarrer Simulation .

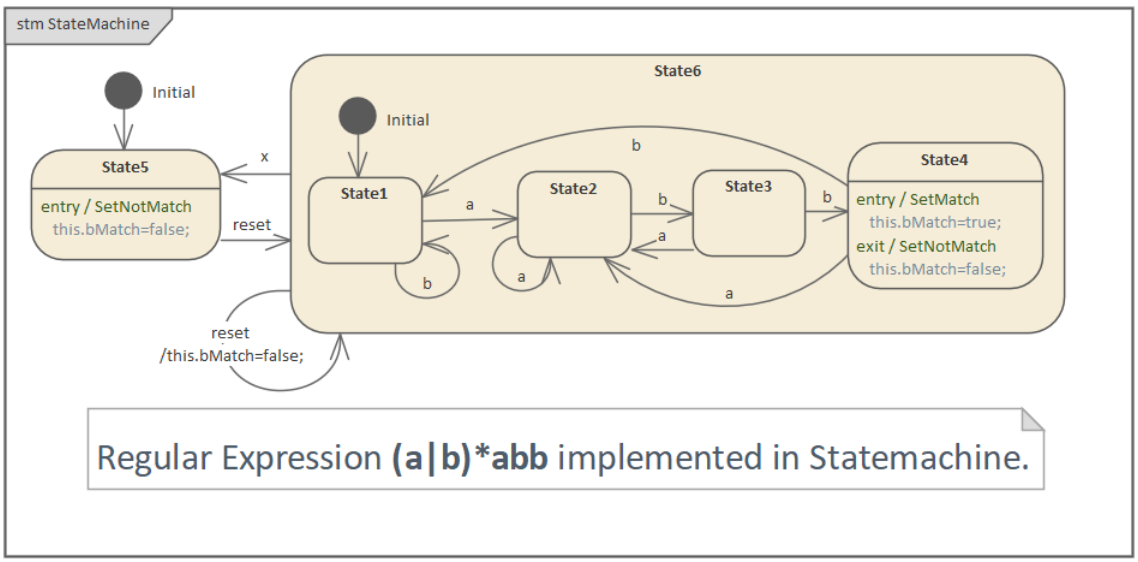
Parser d'expressions régulières

Statemachine pour Parser d'expressions régulières

La classe *RegularExpressionParser* est définie avec un attribut : bMatch.



Une Statemachine est utilisée pour décrire l'expression régulière (a|b)*abb



- Les déclencheurs de transition sont spécifiés comme des événements *a* , *b* , *x* et *reset*
- À l'entrée de l'État 4, bMatch est défini sur True ; à la sortie de l'État 4, bMatch est défini sur False
- À l'entrée de State5, bMatch est défini sur False
- Lors de l'auto-transition de l'état 6, bMatch est défini sur False

Personnaliser index.html et statemachineGUI.js

Apportez ces modifications aux fichiers générés :

- Créez un champ de saisie HTML et une image pour indiquer le résultat
- Créer JavaScript comme gestionnaire d'événements *oninput* du champ
- Créez la fonction « SetResult » pour basculer l'image réussite/échec
- Créez la fonction 'getEventStr', qui renverra 'a' sur ' *a* ' et 'b' sur ' *b* ', mais renverra 'x' sur tout autre caractère
- Lors de l'initialisation, diffuser « *reset* »
- Lors de l'événement de diffusion, demandez la variable d'exécution « regxParser.bMatch »

Dans *statemachineGUI.js* , recherchez la fonction « onRuntimeValueResponse ».

- Dans « regxParser.bMatch », nous recevrons « True » ou « False » et le transmettrons à « SetResult » pour mettre à jour l'image

L'exemple complet

L'exemple est accessible depuis la page « Ressources » du site Web Sparx Systems , en cliquant sur ce lien :

Regular Expression Parser Simulation

Exemple : Entrer d'un State

La sémantique de l’entrée dans un State dépend du type d’ State et de la manière dont on y entre.

Dans tous les cas, le comportement d'entrée de l' State est exécuté (s'il est défini) à l'entrée, mais seulement après la fin de tout comportement d'effet associé à la transition entrante. De plus, si un comportement doActivity est défini pour l' State , ce comportement commence son exécution immédiatement après l'exécution du comportement d'entrée.

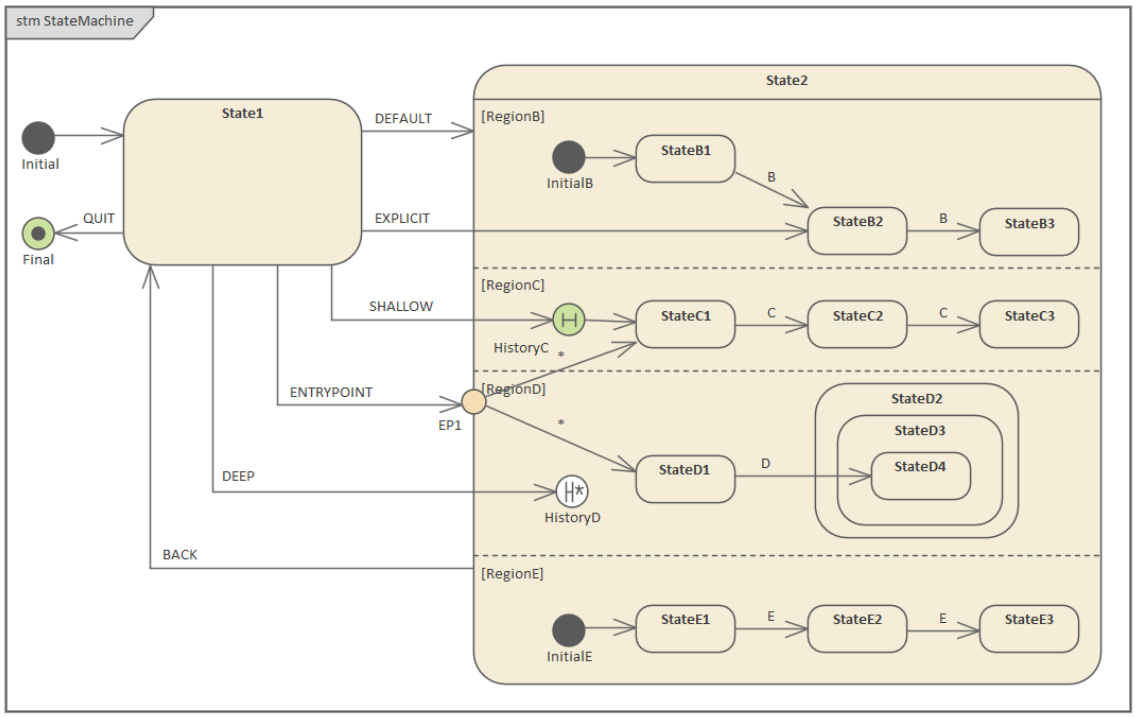
Pour un State composite avec une ou plusieurs régions définies, un certain nombre d'alternatives existent pour chaque région :

- *Entrée par défaut* : Cette situation se produit lorsque l' State composite propriétaire est la cible directe d'une transition ; après avoir exécuté le comportement d'entrée et avoir forgé une éventuelle exécution de comportement doActivity, l'entrée State continue à partir d'un pseudo-état initial via sa transition sortante (appelée transition par défaut de l' State) si elle est définie dans la région
  Si aucun pseudo-état initial n'est défini, cette région ne sera pas active
- *Entrée explicite* : Si la transition entrante ou ses continuations se terminent sur un sous-état directement contenu dans l' State composite propriétaire, alors ce sous-état devient actif et son comportement d'entrée est exécuté après l'exécution du comportement d'entrée de l' State composite contenant
  Cette règle s'applique de manière récursive si la transition se termine sur un sous-état indirect (profondément imbriqué)
- *Entrée d'historique peu profonde* : si la transition entrante se termine sur un pseudo-état d'historique peu profonde de cette région, le sous-état actif devient le sous-état qui était le plus récemment actif (sauf l'état final) avant cette entrée, à moins qu'il ne s'agisse de la première entrée dans cet State ; s'il s'agit de la première entrée dans cet State ou si l'entrée précédente avait atteint un état final, une transition d'historique peu profonde par défaut sera prise si elle est définie, sinon l'entrée State par défaut est appliquée
- *Entrée d'historique profond* : la règle pour ce cas est la même que pour l'historique superficiel, sauf que le pseudo-état cible est de type deepHistory et que la règle est appliquée de manière récursive à tous les niveaux de la configuration State active en dessous de celui-ci
- *Entrée au point d'entrée* : si une transition entre dans l' State composite propriétaire via un pseudo-état entryPoint, alors la transition sortante provenant du point d'entrée et pénétrant dans l' State de cette région est prise ; s'il y a plus de transitions sortantes à partir des points d'entrée, chaque transition doit cibler une région différente et toutes les régions sont activées simultanément

Pour States orthogonaux à plusieurs Régions, si la Transition entre explicitement dans une ou plusieurs Régions (dans le cas d'une Fourche ou d'un point d'entrée), ces Régions sont saisies explicitement et les autres par défaut.

Dans cet exemple, nous démontrons un modèle avec tous ces comportements d’entrée pour un State orthogonal.

Modélisation d'une Statemachine



Contexte de Statemachine

1. Créez un élément Class nommé *MyClass* , qui sert de contexte à la Statemachine .
2. Cliquez-droit sur *MyClass* dans la fenêtre Navigateur et sélectionnez l'option 'Ajouter | Statemachine '.

Statemachine

1. Ajoutez au diagramme un nœud *initial* , un State nommé *State1* , un State nommé *State2* et un élément final nommé *final.*
2. Agrandissez *l'État 2* sur le diagramme , cliquez-droit dessus et sélectionnez l'option « Avancé | Définir les sous-états simultanés », puis définissez *la Région B, la Région C, la Région D* et *la Région E.*
3. Cliquez-droit sur *State2* et sélectionnez l'option 'Nouvel élément enfant | Point d'entrée' pour créer le point d'entrée *EP1* .
4. Dans *la RégionB* , créez les éléments *InitialB* , transition vers *l'ÉtatB1* , transition vers *l'ÉtatB2* , transition vers *l'ÉtatB3* ; toutes les transitions déclenchées par l'événement *B.*
5. Dans *RegionC* , créez les éléments shallow *HistoryC* (cliquez-droit sur le nœud History | Advanced | Deep History | décochez), transition to *StateC1* , transition to *StateC2* , transition to *StateC3* ; toutes les transitions déclenchées par Event *C.*
6. Dans *RegionD* , créer les éléments deep *HistoryD* (cliquez-droit sur le noeud History | Advanced | Deep History | check), transition vers *StateD1* , créer *StateD2* comme parent de *StateD3* , qui est parent de *StateD4* ; transition de *StateD1* à *StateD4* ; déclenchée par l'événement *D.*
7. Dans *RegionE* , créez les éléments *InitialE* , transition vers *StateE1* , transition vers *StateE2* , transition vers *StateE3* ; toutes les transitions déclenchées par l'événement *E.*
8. Dessinez les transitions du point d'entrée *EP1* vers *l'État C1* et *l'État D1.*

Dessiner des transitions pour différents types d'entrées :

1. Entrée par défaut : *État 1* à *État 2* ; déclenché par l'événement DEFAULT.
2. Entrée explicite : *État1* à *ÉtatB2* ; déclenché par l'événement EXPLICIT.
3. Entrée d'historique superficiel : *État 1* vers *Historique C* ; déclenchée par l'événement SHALLOW.
4. Entrée d'historique profond : *État 1* à *Historique D* ; déclenché par l'événement DEEP.
5. Point d'entrée Entrée : *État 1* à *EP1* ; déclenché par l'événement ENTRYPOINT.

Autres transitions :

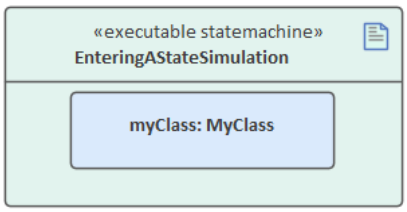
1. Sortie State composite : de *l'État 2* à *l'État 1* ; déclenchée par l'événement BACK.
2. *État 1* à *Final* , déclenché par l'événement QUIT.

Simulation

Artefact

Enterprise Architect supporte C, C++, C# , Java et JavaScript . Nous utilisons JavaScript dans cet exemple car nous n'avons pas besoin d'installer de compilateur. (Pour les autres langages, Visual Studio ou JDK sont requis.)

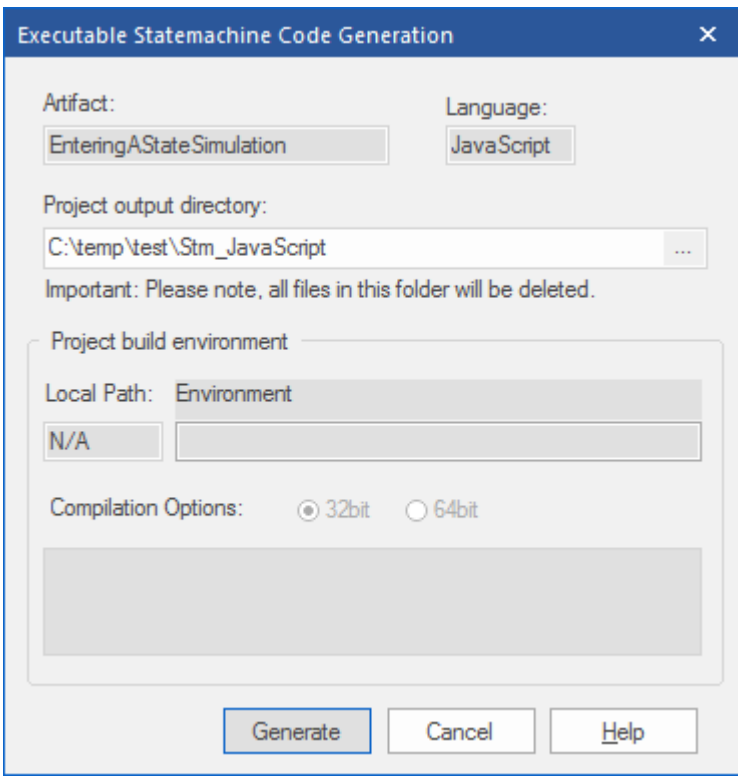
1. Sur la page « Simulation » de la boîte à outils Diagramme , faites glisser l'icône Statemachine Exécutable sur un diagramme et créez un artefact nommé *EnteringAStateSimulation* . Définissez le langage sur JavaScript .
2. Maintenez Ctrl+faites glisser l'élément *MyClass* de la fenêtre Navigateur sur l'artefact *EnteringAStateSimulation* , sélectionnez l'option « Coller comme propriété » et donnez à la propriété le nom *myClass.*



Génération de code

1. Cliquez sur *EnteringAStateSimulation* et sélectionnez l'option de ruban 'Simulate > States Exécutables > Statemachine > Générer , Build and Exécuter '.
2. Spécifiez un répertoire pour le code source généré.

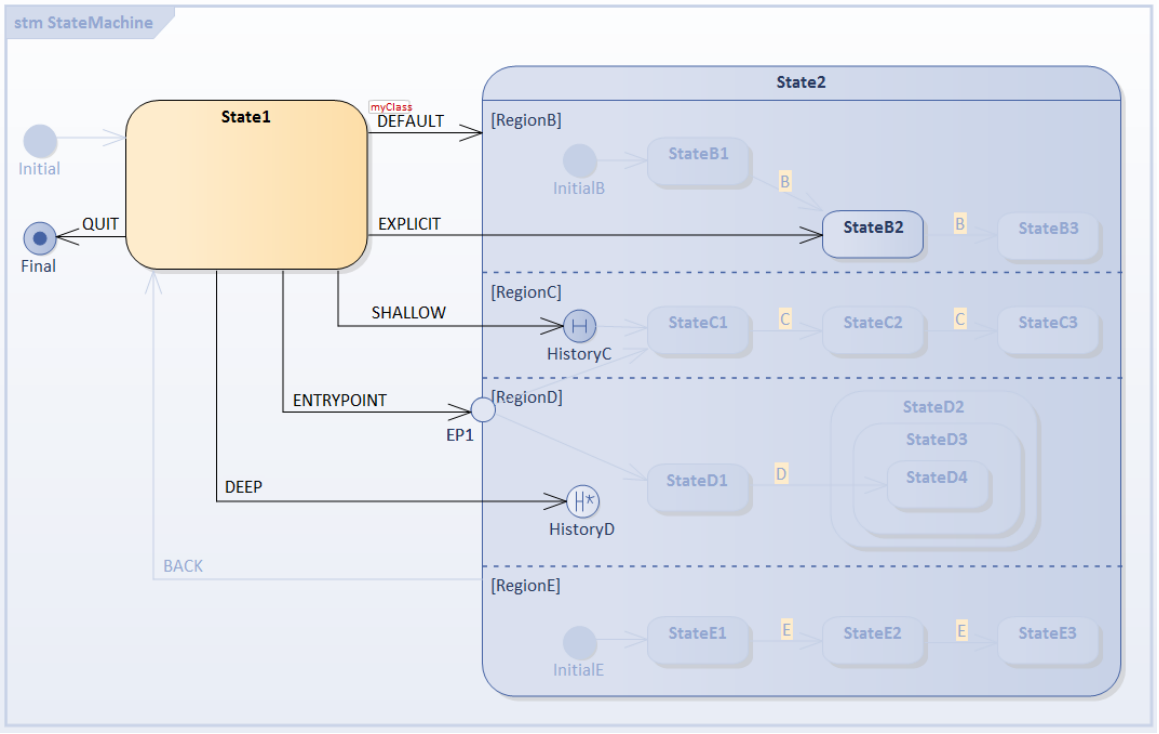
Note : le contenu de ce répertoire sera effacé avant la génération ; assurez-vous de spécifier un répertoire utilisé uniquement à des fins de simulation Statemachine .



Exécuter Simulation

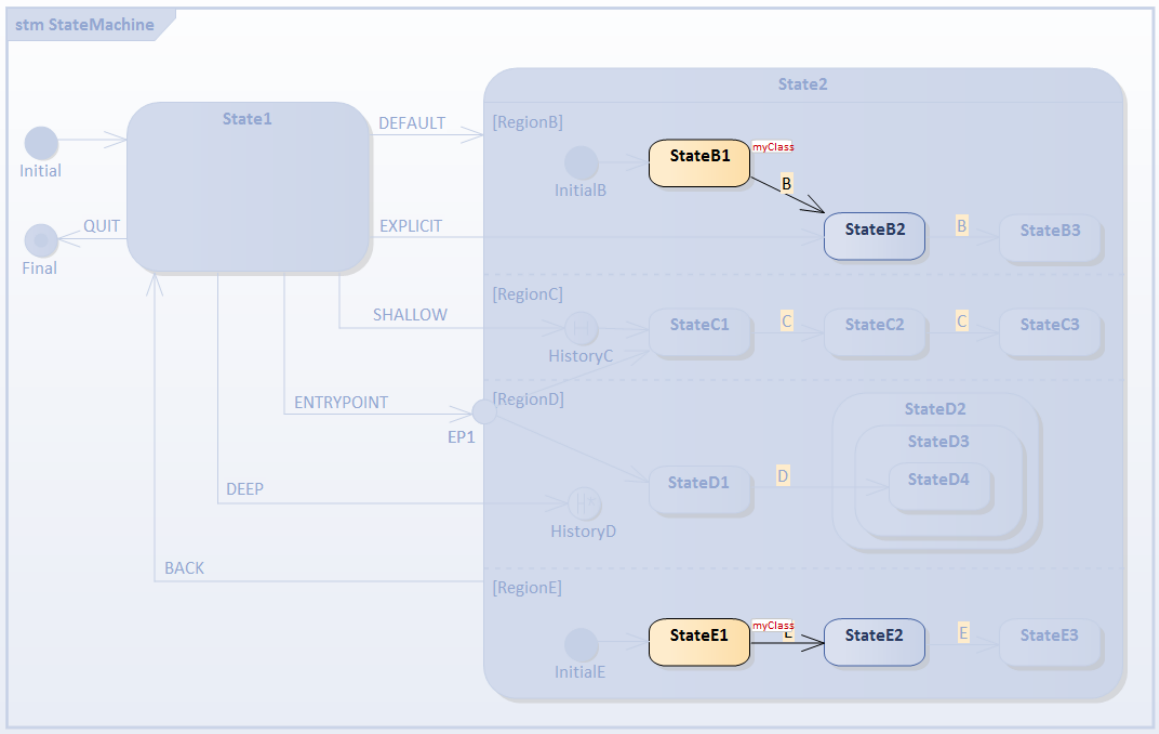
Conseils : Vous pouvez visualiser la séquence de trace d'exécution à partir de la fenêtre Simulation , que vous ouvrez en sélectionnant l'option de ruban 'Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation '

Lorsque la simulation commence, *State1* est actif et Statemachine attend des événements.



Ouvrez la fenêtre Simulation Événements (Déclencheurs) à l'aide de l'option de ruban 'Simuler > Simulation Dynamique > Événements '.

1) Sélectionnez l'entrée par défaut : Déclencheur Séquence [DEFAULT].

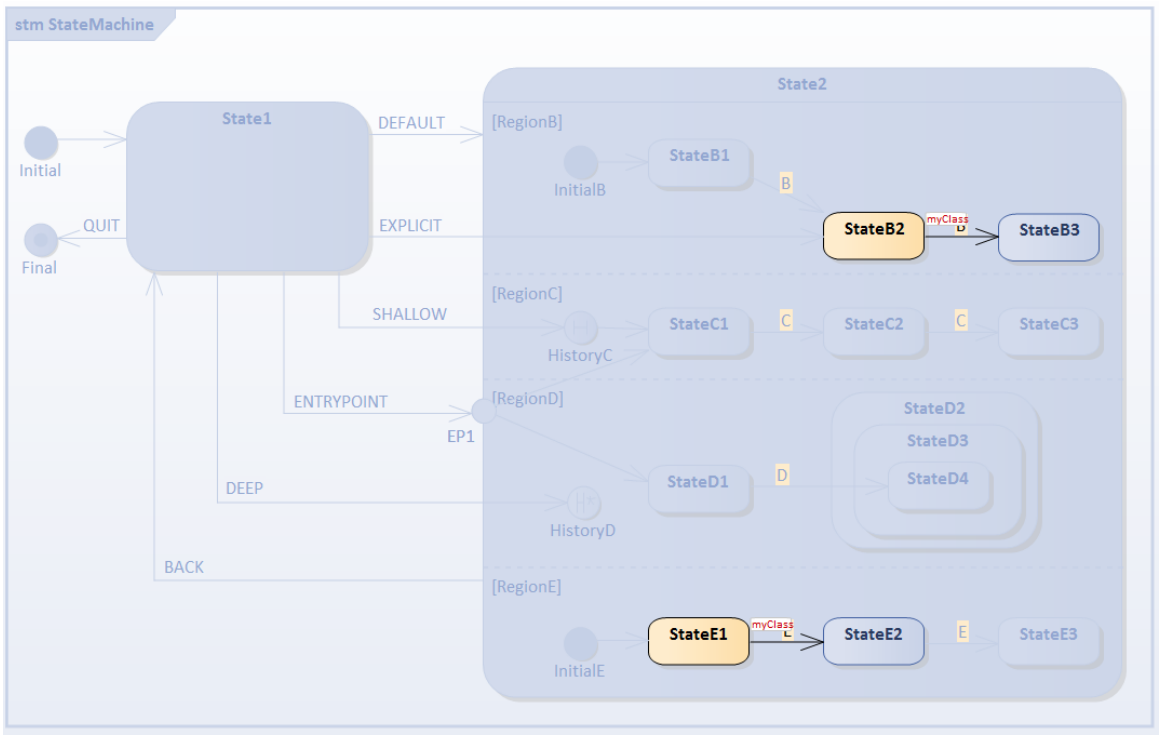


- *La région B* est activée car elle définit *InitialB* ; la transition sortant de celle-ci sera exécutée, *l'état B1* est l'état actif

- *RegionE* est activé car il définit *InitialE* ; la transition sortant de celui-ci sera exécutée, *StateE1* est l'état actif
- *Les régions C* et *D* sont inactives car aucun pseudo-état initial n'a été défini

Sélectionnez le Déclencheur [BACK] pour réinitialiser.

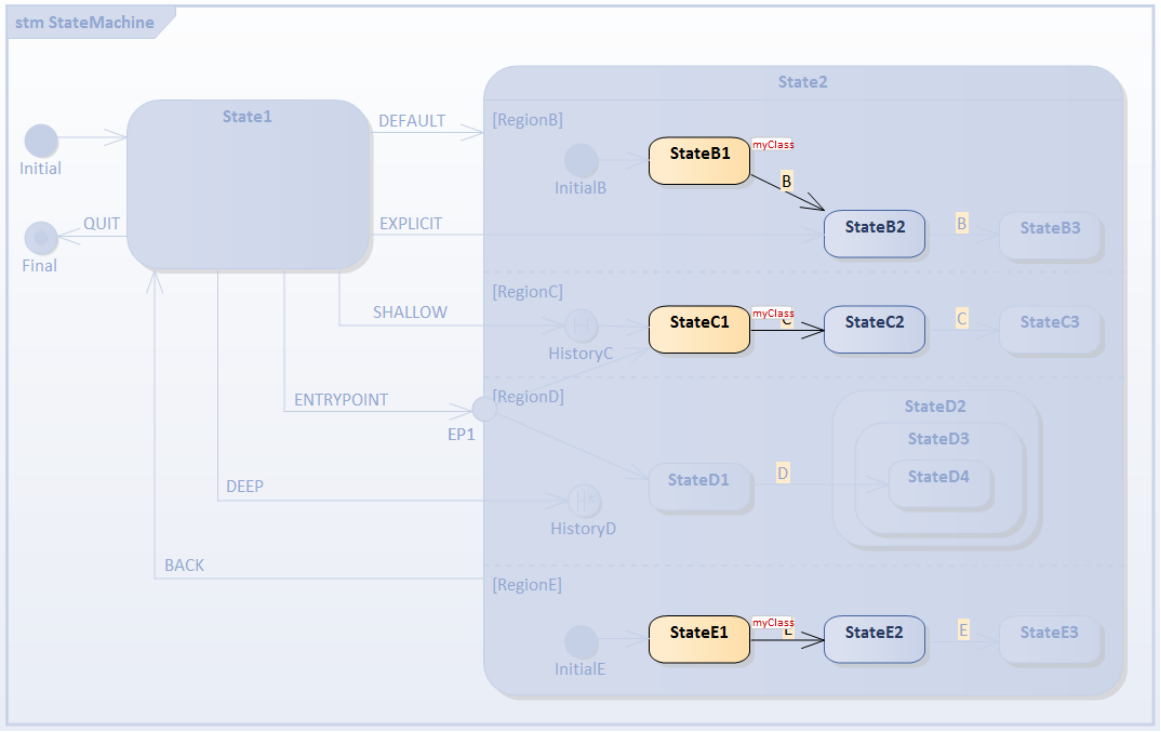
2) Sélectionnez l'entrée explicite : Déclencheur Séquence [EXPLICIT].



- *La région B* est activée car la transition cible le sommet contenu *StateB2*
- *RegionE* est activé car il définit *InitialE* ; la transition sortant de celui-ci sera exécutée, *StateE1* est l'état actif
- *Les régions C* et *D* sont inactives car aucun pseudo-état initial n'a été défini

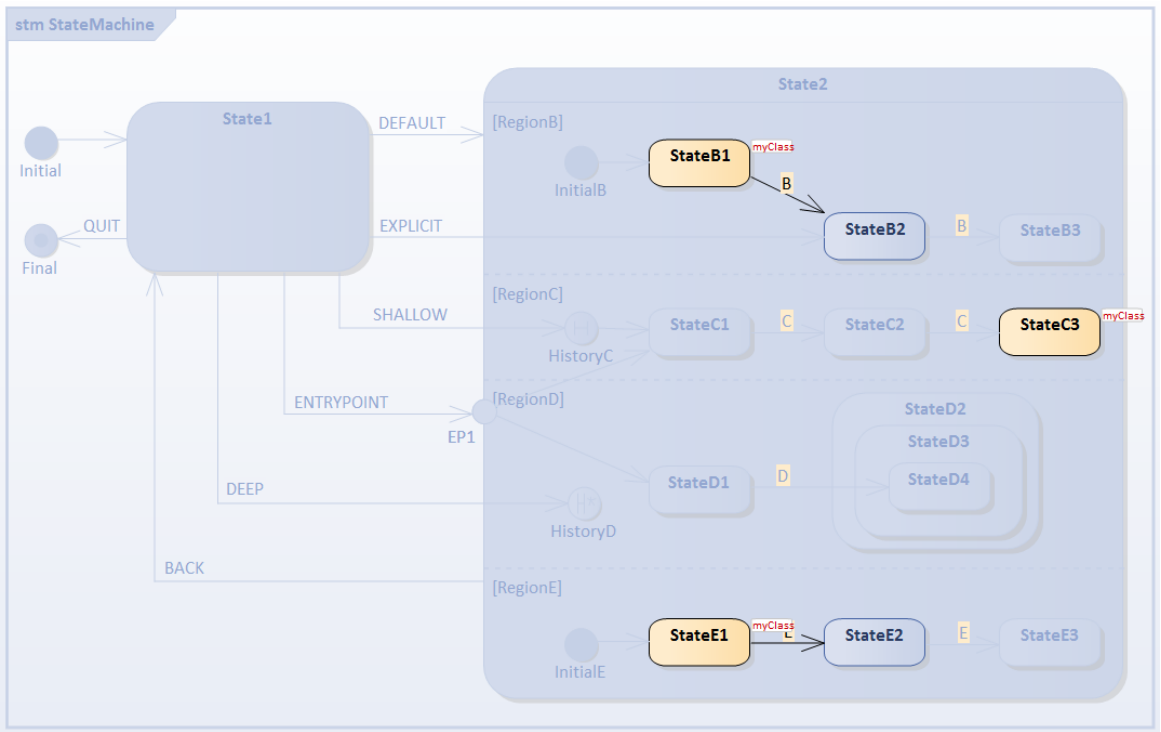
Sélectionnez le Déclencheur [BACK] pour réinitialiser.

3) Sélectionnez la transition historique par défaut : Séquence Déclencheur [SHALLOW].



- *La région C* est activée car la transition cible le sommet contenu *HistoryC* ; puisque cette région est entrée pour la première fois (et que le pseudo-état History n'a rien à « mémoriser »), la transition sortant de *HistoryC* vers *StateC1* est exécutée
- *La région B* est activée car elle définit *InitialB* ; la transition sortant de celle-ci sera exécutée, *l'état B1* est l'état actif
- *RegionE* est activé car il définit *InitialE* ; la transition sortant de celui-ci sera exécutée, *StateE1* est l'état actif
- *La région D* est inactive car aucun pseudo-état initial n'a été défini

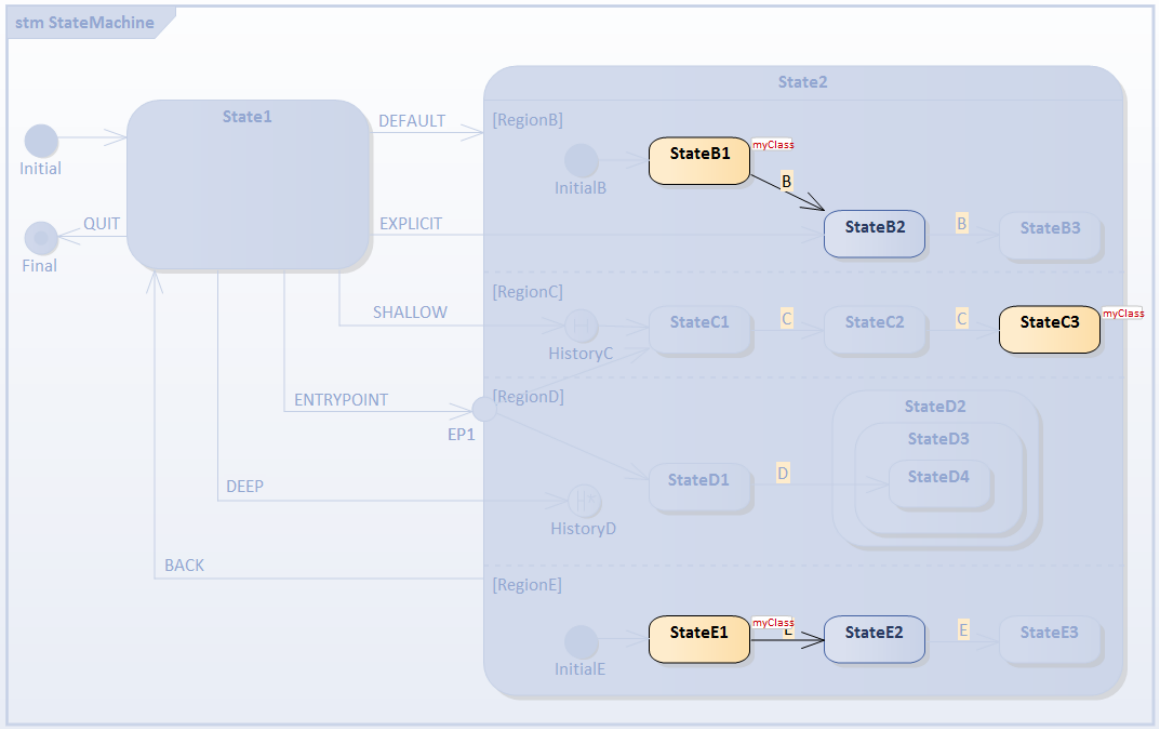
4) Préparez-vous au test de l'entrée d'historique superficielle : Séquence Déclencheur [C, C].



- Nous supposons que le pseudo-état d'historique superficiel *HistoryC* peut se souvenir *de StateC3*

Sélectionnez le Déclencheur [BACK] pour réinitialiser.

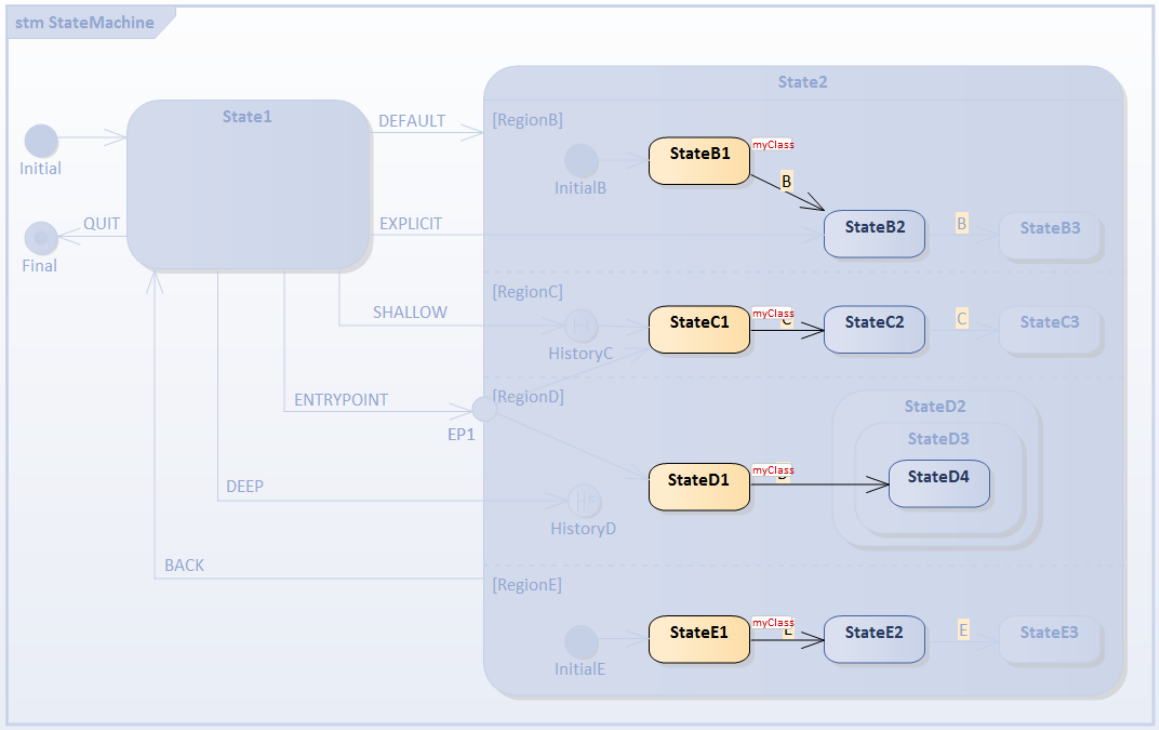
5) Sélectionnez l'entrée d'historique peu profonde : Déclencheur Séquence [SHALLOW].



- Pour *RegionC* , *StateC3* est activé directement

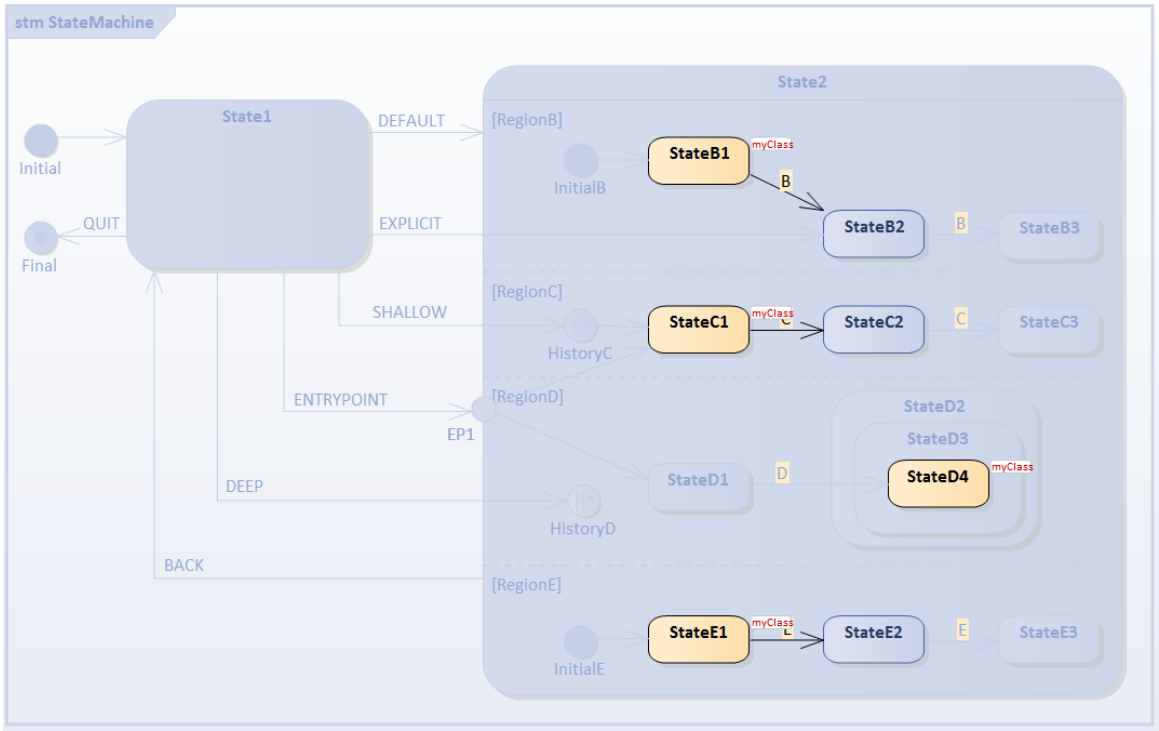
Sélectionnez le Déclencheur [BACK] pour réinitialiser.

6) Sélectionnez l'entrée du point d'entrée : Séquence Déclencheur [ENTRYPOINT].



- *La région C* est activée car la transition depuis *EP1* cible l' *État C1* contenu
- *La région D* est activée car la transition depuis *EP1* cible l' *État D1* contenu
- *La région B* est activée car elle définit *InitialB* ; la transition sortant de celle-ci sera exécutée, *l'état B1* est l'état actif
- *RegionE* est activé car il définit *InitialE* ; la transition sortant de celui-ci sera exécutée, *StateE1* est l'état actif

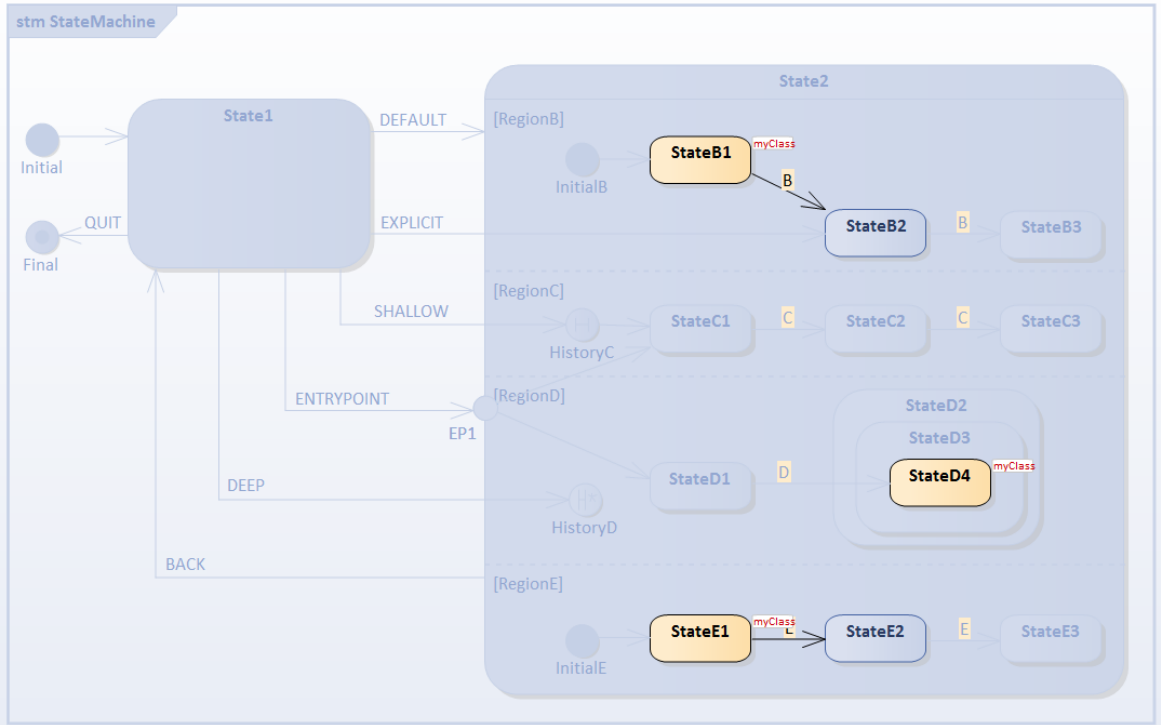
7) Préparez-vous à tester Deep History : Déclencheur Séquence [D].



- Nous supposons que le pseudo-état d'histoire profonde *HistoryD* peut se souvenir *de StateD2* , *StateD3* et *StateD4*

Sélectionnez le Déclencheur [BACK] pour réinitialiser.

8) Sélectionnez l'entrée de l'historique profond : Déclencheur Séquence [DEEP].



- Pour *RegionD* , *StateD2* , *StateD3* et *StateD4* sont saisis ; les traces sont :
  - maClasse[MaClasse].StateMachine_State1 SORTIE

- myClass[MyClass].State1__TO__HistoryD_105793_61752 Effet

- maClasse[MaClasse].StateMachine_State2 ENTRÉE

- maClasse[MaClasse].StateMachine_State2 FAIRE

- myClass[MyClass].InitialE_105787__TO__StateE1_61746 Effet
- maClasse[MaClasse].StateMachine_State2_StateE1 ENTRÉE
- maClasse[MaClasse].StateMachine_State2_StateE1 FAIRE
- myClass[MyClass].InitialB_105785__TO__StateB1_61753 Effet
- maClasse[MaClasse].StateMachine_State2_StateB1 ENTRÉE
- maClasse[MaClasse].StateMachine_State2_StateB1 FAIRE
- maClasse[MaClasse].StateMachine_State2_StateD2 ENTRÉE
- maClasse[MaClasse].StateMachine_State2_StateD2_StateD3 ENTRÉE
- maClasse[MaClasse].StateMachine_State2_StateD2_StateD3_StateD4 ENTRÉE

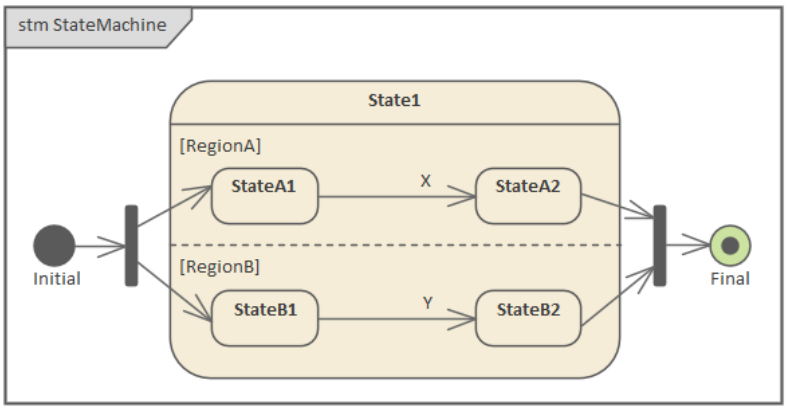
Exemple : Fourche et Joindre

Les pseudo-états de fourche divisent une transition entrante en deux ou plusieurs transitions, se terminant par des sommets dans des régions orthogonales d'un State composite. Les transitions sortant d'un pseudo-état de fourche ne peuvent pas avoir de garde ou de déclencheur , et les comportements d'effet des transitions sortantes individuelles sont, au moins conceptuellement, exécutés simultanément.

Les pseudo-états de jointure sont un sommet cible commun pour deux ou plusieurs transitions provenant de sommets dans différentes régions orthogonales. Les pseudo-états de jointure exécutent une fonction de synchronisation, selon laquelle toutes les transitions entrantes doivent être terminées avant que l'exécution puisse se poursuivre via une transition sortante.

Dans cet exemple, nous démontrons le comportement d'une Statemachine avec des pseudo-états Fourche et Joindre .

Modélisation Statemachine



Contexte de Statemachine

- Créez un élément de classe nommé *MyClass* , qui sert de contexte à une Statemachine
- Cliquez-droit sur *MyClass* dans la fenêtre Navigateur et sélectionnez l'option 'Ajouter | Statemachine '

Statemachine

- Ajoutez un nœud *initial* , une *fourche* , un State nommé *State1* , une *jointure* et un *final* au diagramme
- Agrandir *l'état1* , cliquez-droit dessus sur le diagramme et sélectionnez l'option « Avancé | Définir les sous-états simultanés | Définir » et définissez *la région A* et *la région B*
- Dans *RegionA* , définissez *StateA1* , transition vers *StateA2* , déclenchée par l'événement *X*
- Dans *la région B* , définissez *l'État B1* , transition vers *l'État B2* , déclenchée par l'événement *Y*
- Dessiner d'autres transitions : *Initial* vers *Fork* ; *Fork* vers *StateA1* et *StateB1* ; *StateA2 et StateB2* vers *Join* ; *Join* vers *Final*

Simulation

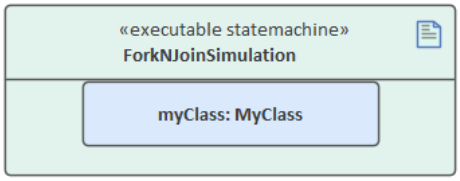
Artefact

Enterprise Architect supporte C, C++, C# , Java et JavaScript ; nous utiliserons JavaScript dans cet exemple car nous n'avons pas besoin d'installer de compilateur (pour les autres langages, Visual Studio ou JDK sont requis).

- Depuis la boîte à outils Diagramme sélectionnez la page « Simulation » et faites glisser l'icône Statemachine Exécutable sur le diagramme pour créer un artefact ; nommez-le *ForkNJoinSimulation* et définissez son champ

« Langue » sur « JavaScript ».

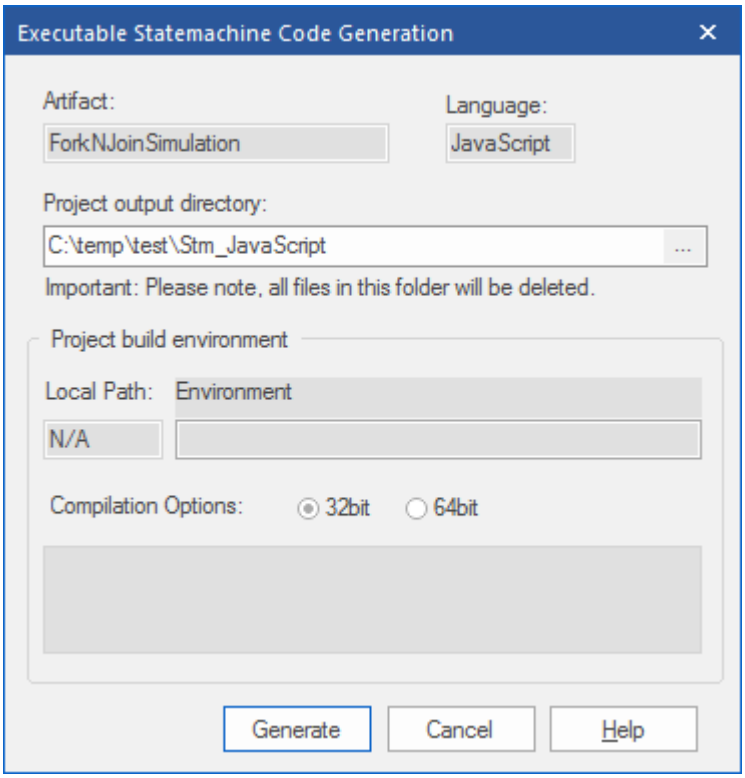
- Ctrl+Drag *MyClass* depuis la fenêtre Navigateur et déposez-le sur l'artefact *ForkNJoinSimulation* en tant que propriété ; donnez-lui le nom *myClass*



Génération de code

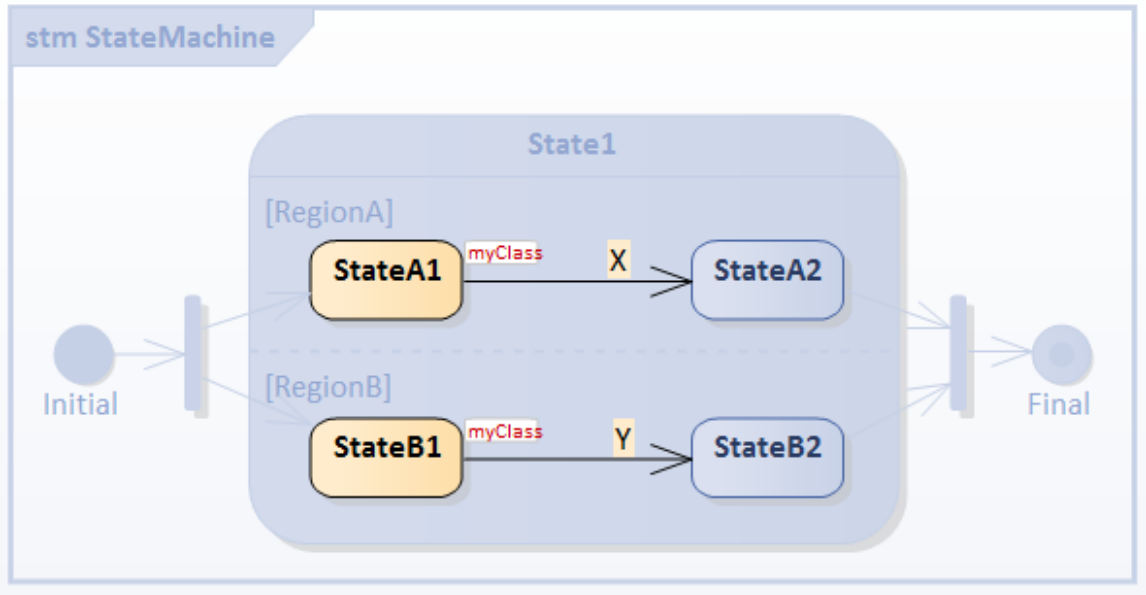
- Cliquez sur *ForkNJoinSimulation* et sélectionnez l'option de ruban 'Simulate > States Exécutables > Statemachine > Générer , Build and Exécuter '
- Spécifiez un répertoire pour le code source généré

Note : le contenu de ce répertoire sera effacé avant la génération ; assurez-vous de pointer vers un répertoire qui existe uniquement à des fins de simulation Statemachine .



Exécuter Simulation

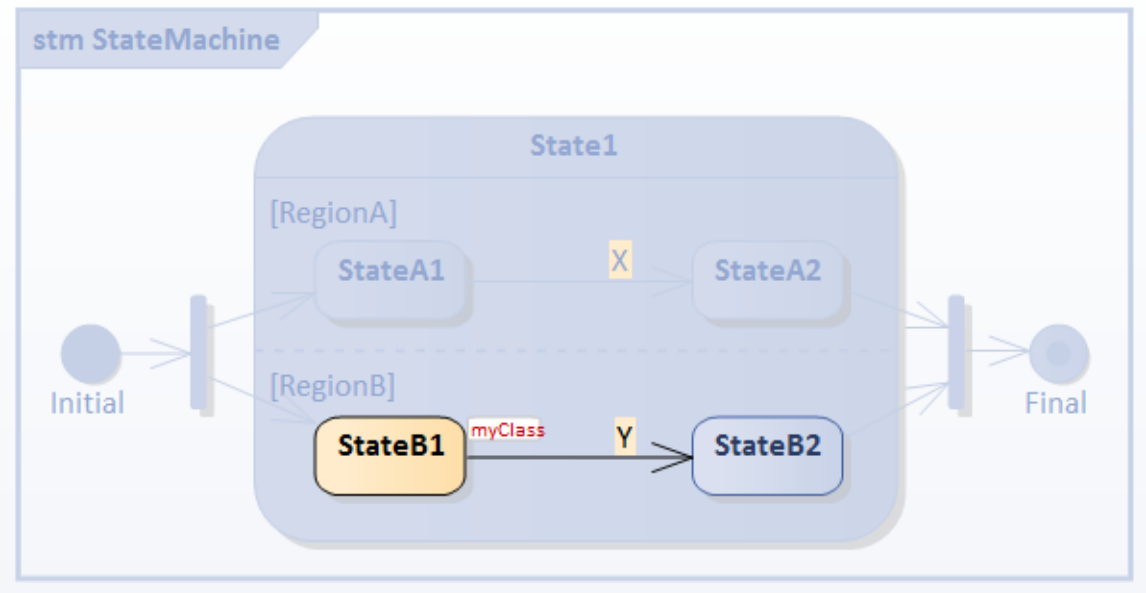
Lorsque la simulation démarre, *State1* , *StateA1* et *StateB1* sont actifs et la Statemachine attend des événements.



Sélectionnez l'option de ruban 'Simuler > Simulation Dynamique > Événements ' pour afficher la fenêtre Simulation Événements .

Lors de l'événement Déclencheur *X* , *StateA1* va sortir et entrer dans *StateA2* ; après que les comportements entry et doActivity ont été exécuter , les événements d'achèvement de *StateA2* sont envoyés et rappelés. Ensuite, la transition de *StateA2* vers le pseudo-état *Join* est activée et parcourue.

Note : *Join* doit attendre que toutes les transitions entrantes soient terminées avant que l'exécution puisse se poursuivre via une transition sortante. Étant donné que la branche de *RegionB* n'est pas terminée (car *StateB1* est toujours actif et en attente de déclencheurs), la transition de *Join* à *Final* ne sera pas exécutée à ce stade. ce moment.



Lors de l'événement Déclencheur *Y* , *StateB1* va sortir et entrer *dans StateB2* ; après que le comportement entry et doActivity a été exécuter , les événements d'achèvement de *StateB2* sont envoyés et rappelés. Ensuite, la transition de *StateB2* vers le pseudo-état *Join* est activée et parcourue. Cela satisfait les critères de tous les transitions entrantes de *Join* étant terminées, la transition de *Join* à *Final* est exécutée. Simulation est terminée.

Conseils : Vous pouvez visualiser la séquence de trace d'exécution depuis la fenêtre Simulation (option du ruban 'Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation ').

myClass[MyClass].Initial_82285__TO__fork_82286_82286_61745 Effet

maClasse[MaClasse].StateMachine_State1 ENTRÉE
maClasse[MaClasse].StateMachine_State1 FAIRE
myClass[MyClass].fork_82286_82286__TO__StateA1_57125 Effet
maClasse[MaClasse].StateMachine_State1_StateA1 ENTRÉE
maClasse[MaClasse].StateMachine_State1_StateA1 FAIRE
myClass[MyClass].fork_82286_82286__TO__StateB1_57126 Effet
maClasse[MaClasse].StateMachine_State1_StateB1 ENTRÉE
maClasse[MaClasse].StateMachine_State1_StateB1 FAIRE
Déclencheur X
maClasse[MaClasse].StateMachine_State1_StateA1 SORTIE
myClass[MyClass].StateA1__TO__StateA2_57135 Effet
maClasse[MaClasse].StateMachine_State1_StateA2 ENTRÉE
maClasse[MaClasse].StateMachine_State1_StateA2 FAIRE
maClasse[MaClasse].StateMachine_State1_StateA2 SORTIE
myClass[MyClass].StateA2__TO__join_82287_82287_57134 Effet
Déclencheur Y
maClasse[MaClasse].StateMachine_State1_StateB1 SORTIE
myClass[MyClass].StateB1__TO__StateB2_57133 Effet
maClasse[MaClasse].StateMachine_State1_StateB2 ENTRÉE
maClasse[MaClasse].StateMachine_State1_StateB2 FAIRE
maClasse[MaClasse].StateMachine_State1_StateB2 SORTIE
myClass[MyClass].StateB2__TO__join_82287_82287_57132 Effet
maClasse[MaClasse].StateMachine_State1 SORTIE
maClasse[MaClasse].join_82287_82287__TO__Final_105754_57130 Effet

Exemple : Motif d'événement différé

Enterprise Architect supporte le Motif d'événement différé.

Pour créer un événement différé dans un State :

1. Créer une auto-transition pour l' State .
2. Modifiez le « type » de transition en « interne ».
3. Spécifiez le Déclencheur comme étant l'événement que vous souhaitez différer.
4. Dans le champ « Effet », saisissez « defer(); ».

Pour simuler :

1. Sélectionnez « Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation ». Sélectionnez également 'Simuler > Simulation Dynamique > Événements ' pour ouvrir la fenêtre Simulation Événements .
2. La fenêtre Événements du Simulateur vous aide à déclencheur des événements ; double-cliquez sur un déclencheur dans la colonne ' Déclencheurs en Attente '.
3. La fenêtre Simulation affiche l'exécution sous forme de texte. Vous pouvez saisir « dump » dans la ligne de commande du simulateur pour afficher le nombre d'événements différés dans la file d'attente. Le résultat peut ressembler à ceci :
   [24850060] Pool d'événements : [NOUVEAU,NOUVEAU,NOUVEAU,NOUVEAU,NOUVEAU,]

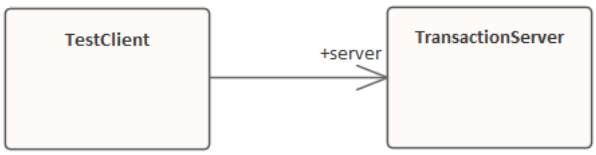
Exemple d'événement différé

Cet exemple montre un modèle utilisant Événements différés et la fenêtre Événements Simulation affichant tous Événements disponibles.

Nous configurons d'abord les contextes (les éléments de classe contenant les Statemachines), les simulons dans un contexte simple et déclenchons l'événement depuis l'extérieur de celui-ci ; puis nous simulons dans un contexte client-serveur avec le mécanisme d'envoi d'événement.

Créer un contexte et Statemachine

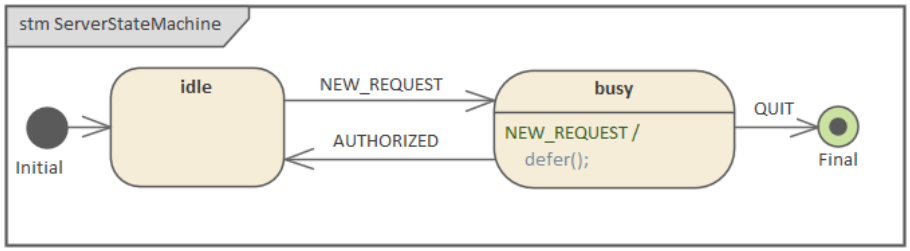
Créer le contexte du serveur



Créez un diagramme de classes et :

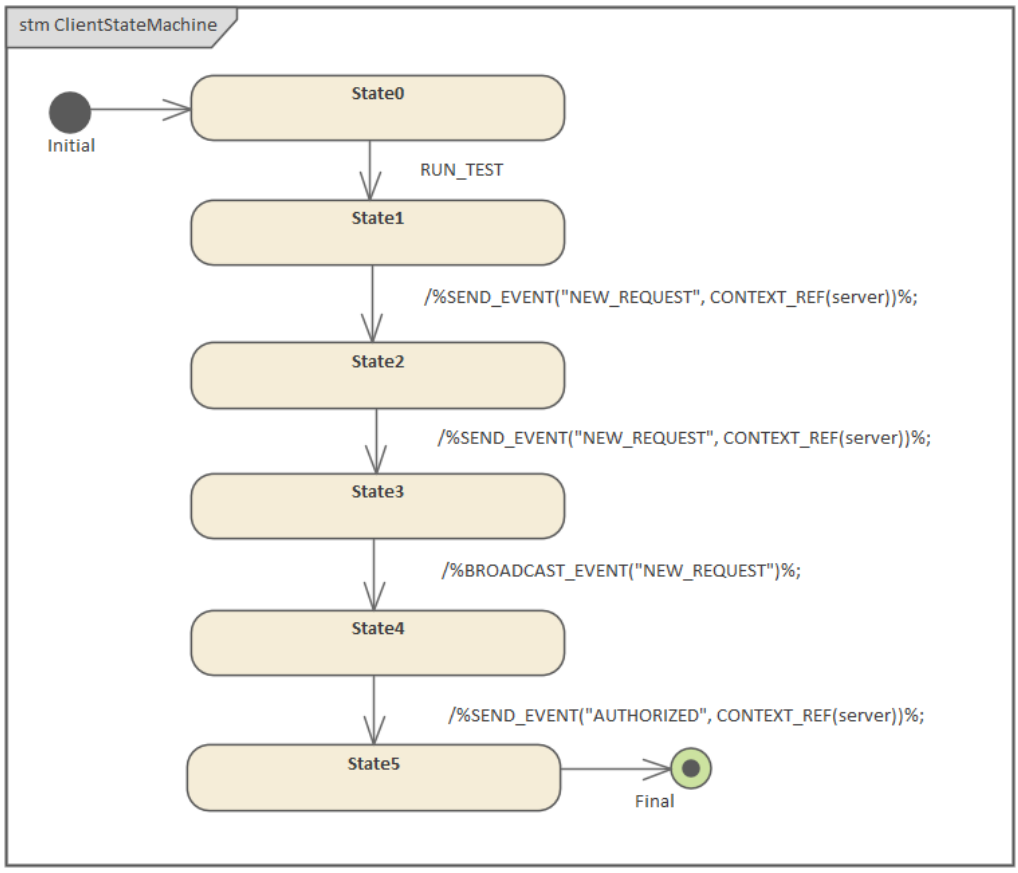
1. Un élément de classe *TransactionServer* , auquel vous ajoutez une Statemachine *ServerStateMachine* .
2. Un élément de classe *TestClient* , auquel vous ajoutez une Statemachine *ClientStateMachine* .
3. Une association de *TestClient* à *TransactionServer* , avec le rôle cible nommé *serveur.*

Modélisation pour *ServerStateMachine*



1. Ajoutez un nœud initial *Initial* au diagramme Statemachine et passez à un State *inactif* .
2. Transition (avec événement NEW_REQUEST comme Déclencheur) vers un State *busy* .
3. Transition (avec événement QUIT comme Déclencheur) vers un State Final Final .
4. Transition (avec événement AUTORISÉ comme Déclencheur) vers *idle* .
5. Transition (avec événement NEW_REQUEST comme Déclencheur et *defer();* comme effet) vers *busy*

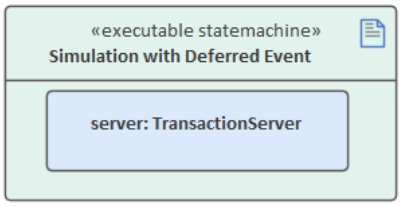
Modélisation pour *ClientStateMachine*



1. Ajoutez un nœud initial *Initial* au diagramme Statemachine et passez à un State *State0* .
2. Transition (avec l'événement RUN_TEST comme déclencheur) vers un State *State1* .
3. Transition (avec effet : %SEND_EVENT("NEW_REQUEST", CONTEXT_REF(serveur))%;) vers un State *State2.*
4. Transition (avec effet : %SEND_EVENT("NEW_REQUEST", CONTEXT_REF(serveur))%;) vers un State *State3* .
5. Transition (avec effet : %BROADCAST_EVENT("NEW_REQUEST")%;) vers un State *State4* .
6. Transition (avec effet : %SEND_EVENT("AUTHORIZED", CONTEXT_REF(serveur))%;) vers un State *State5* .
7. Transition vers un State final *final.*

Simulation dans un contexte simple

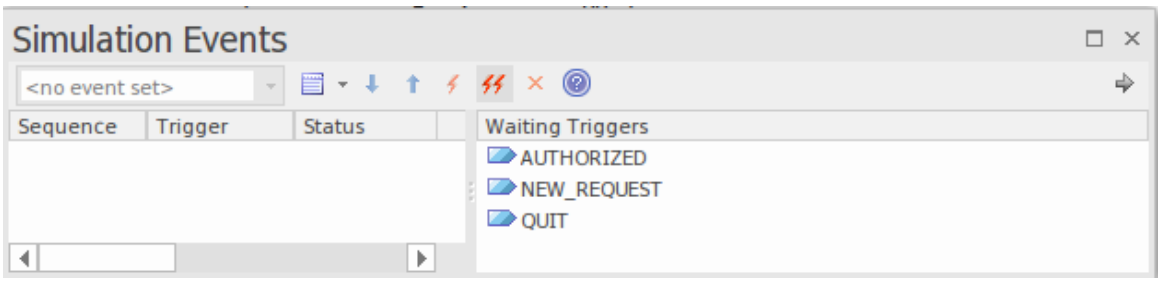
Créer l'artefact Simulation



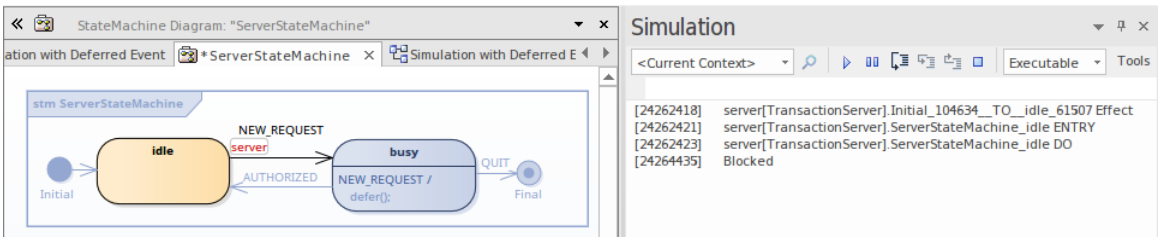
1. Créez un artefact Statemachine Exécutable avec le nom *Simulation avec événement différé* et le champ « Langue » défini sur *JavaScript* .
2. Agrandissez-le, puis appuyez sur Ctrl et faites glisser l'élément *TransactionServer* sur l'artefact et collez-le en tant que propriété avec le *serveur de noms*.

Exécuter la Simulation

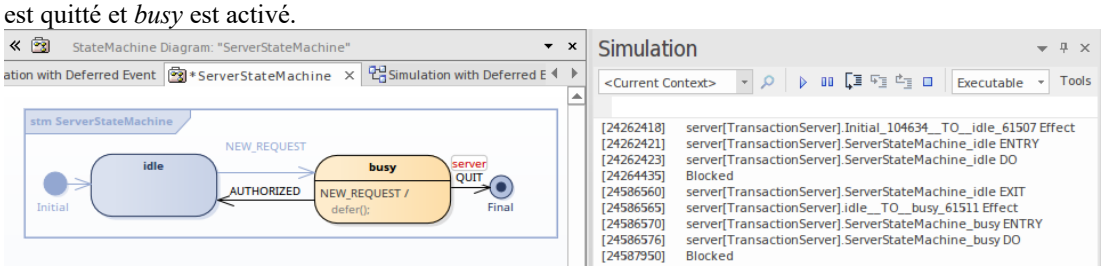
1. Sélectionnez l'Artefact, puis sélectionnez l'option 'Simuler > States Exécutables > Statemachine > Générer , Build et Exécuter ', et spécifiez un répertoire pour votre code (Note : tous les fichiers du répertoire seront supprimés avant le démarrage de la simulation).
2. Cliquez sur le bouton Générer .
3. Sélectionnez l'option « Simuler > Simulation Dynamique > Événements » pour ouvrir la fenêtre Simulation Événement.



Lorsque la simulation démarre, l' *état inactif* sera l'état actif.

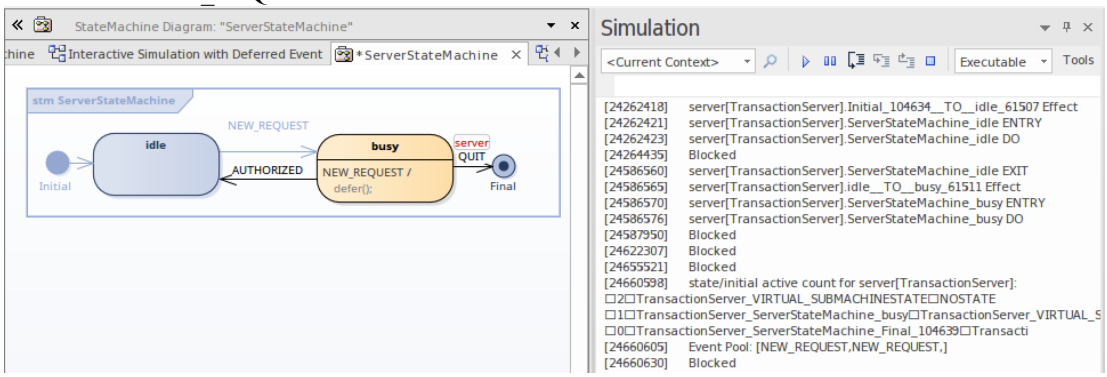


1. Double-cliquez sur NEW_REQUEST dans la fenêtre Simulation Event pour l'exécuter comme le Déclencheur ; *idle* est quitté et *busy* est activé.

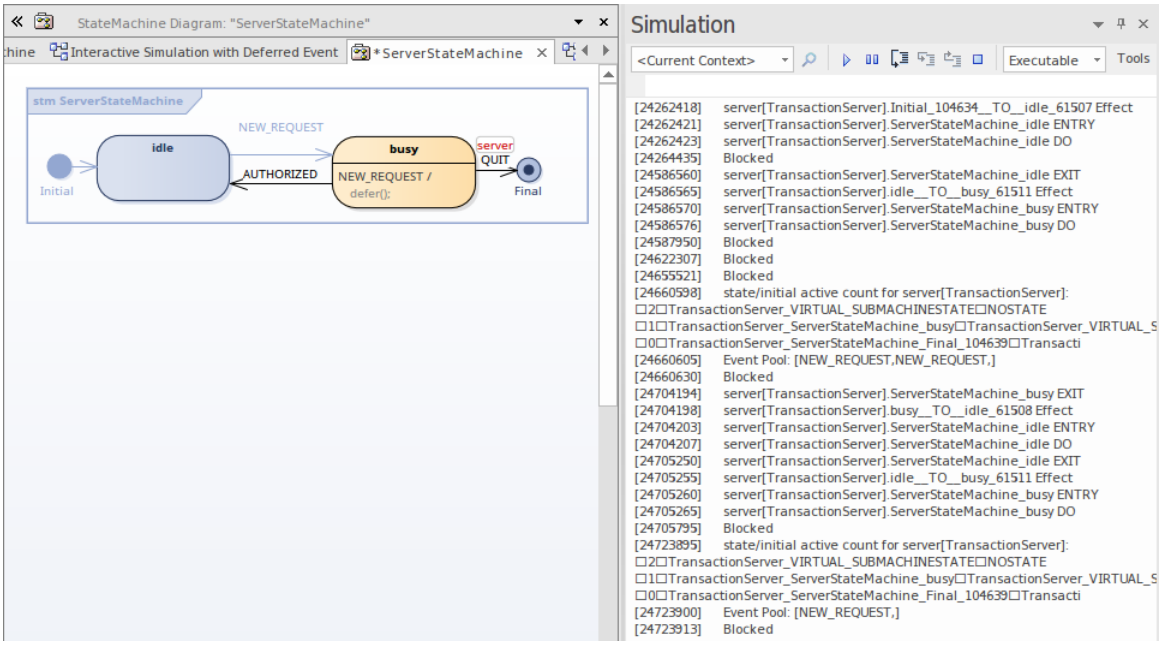


2. Double-cliquez sur NEW_REQUEST dans la fenêtre Événement Simulation pour l'exécuter à nouveau car le Déclencheur ; *busy* reste activé et une instance de NEW_REQUEST est ajoutée dans le pool d'événements.

3. Double-cliquez sur NEW_REQUEST dans la fenêtre Événement Simulation pour l'exécuter une troisième fois pendant que le Déclencheur ; *busy* reste activé, et une instance de NEW_REQUEST est ajoutée dans le pool d'événements.
4. Type *dump* dans la ligne de commande de la fenêtre Simulation ; notez que le pool d'événements comporte deux instances de NEW_REQUEST.

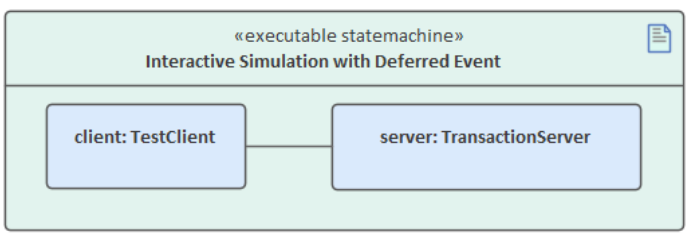


5. Double-cliquez sur AUTORISÉ dans la fenêtre Simulation Event pour l'exécuter en tant que Déclencheur ; ces actions se déroulent :
   - *le mode occupé* est quitté et *le mode inactif* devient actif
   - un événement NEW_REQUEST est récupéré du pool, *l'inactivité* est quittée et *l'occupation* devient active
6. Type *dump* dans la ligne de commande de la fenêtre Simulation ; il n'y a maintenant qu'une seule instance de NEW_REQUEST dans le pool d'événements.



Simulation interactive via envoi/diffusion d'événement

Créer l'artefact Simulation

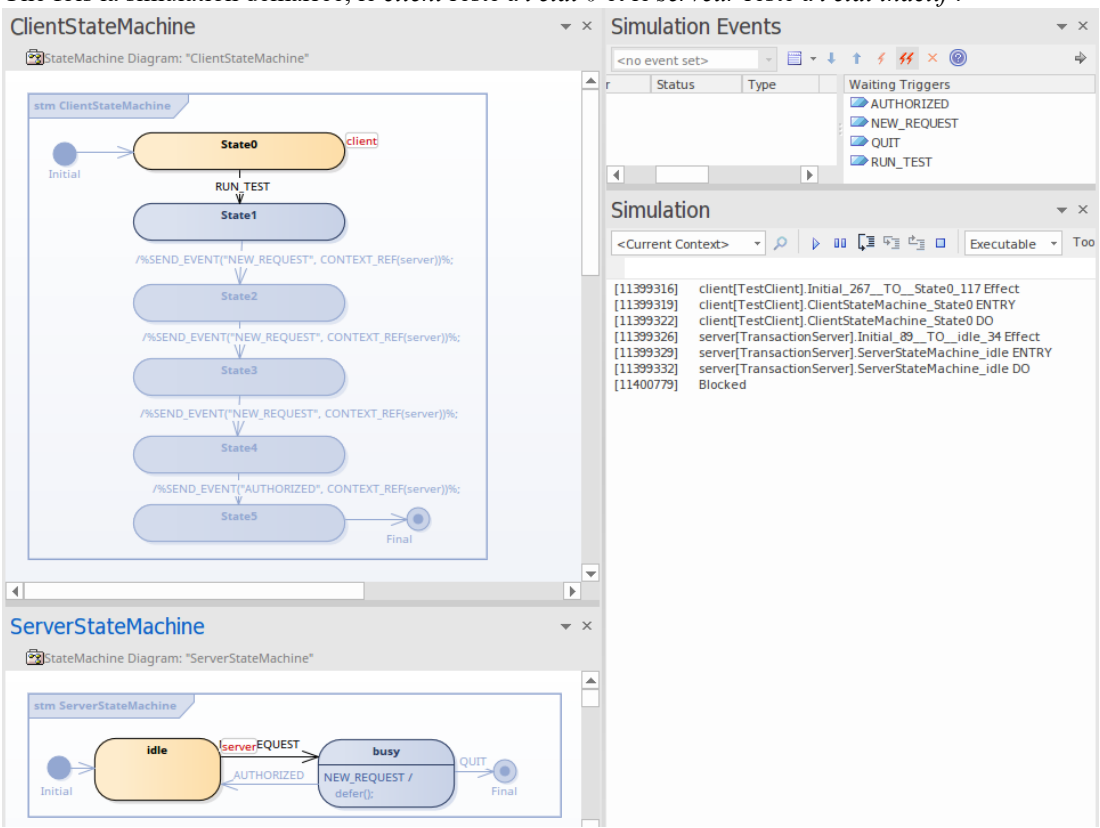


1. Créez un artefact Statemachine Exécutable avec le nom *Simulation interactive avec événement différé* et le champ « Langue » défini sur *JavaScript ;* agrandissez l'élément.
2. Ctrl+faites glisser l'élément *TransactionServer* sur l'artefact et collez-le en tant que propriété avec le *serveur de noms.*
3. Ctrl+Faites glisser l'élément *TestClient* sur l'artefact et collez-le en tant que propriété avec le nom *client* .
4. Créer un connecteur du *client* au *serveur* .
5. Cliquez sur le connecteur et appuyez sur Ctrl+L pour sélectionner l'association de l'élément *TestClient* à l'élément *TransactionServer* .

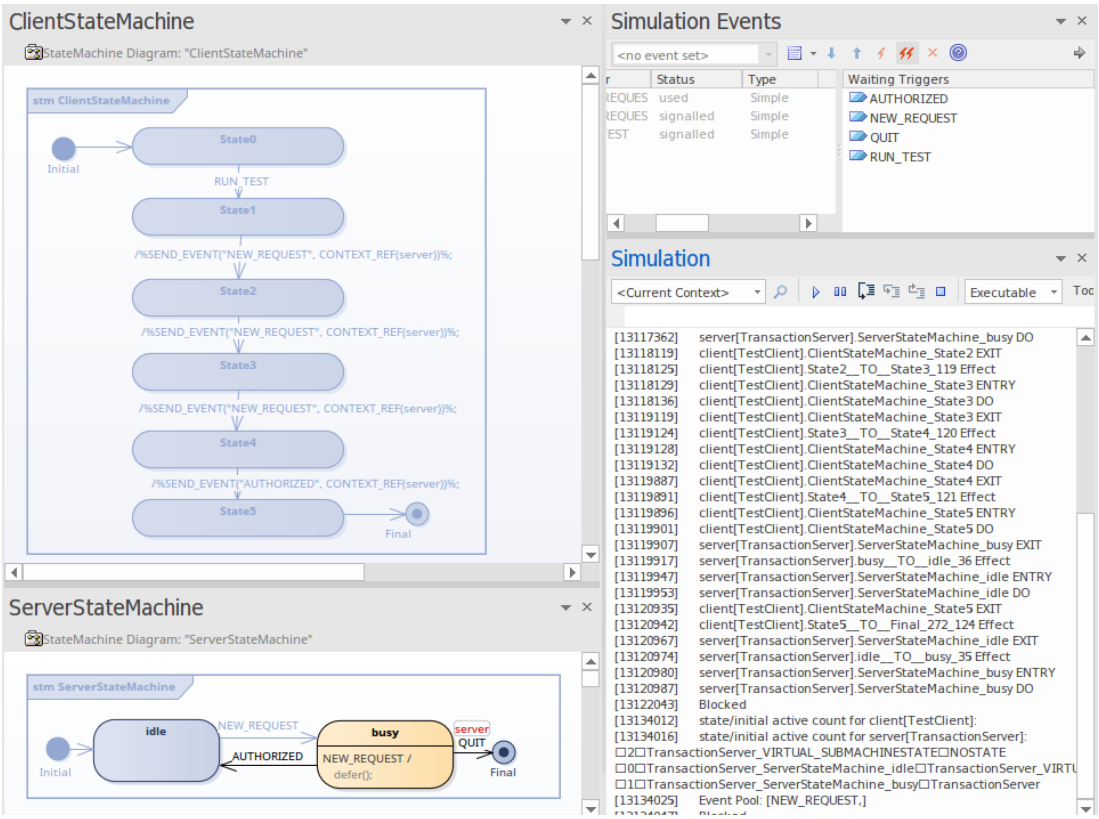
Exécuter Simulation interactive

1. Lancez la simulation de la même manière que pour le contexte simple.

   Une fois la simulation démarrée, le *client* reste à *l'état 0* et le *serveur* reste à *l'état inactif* .



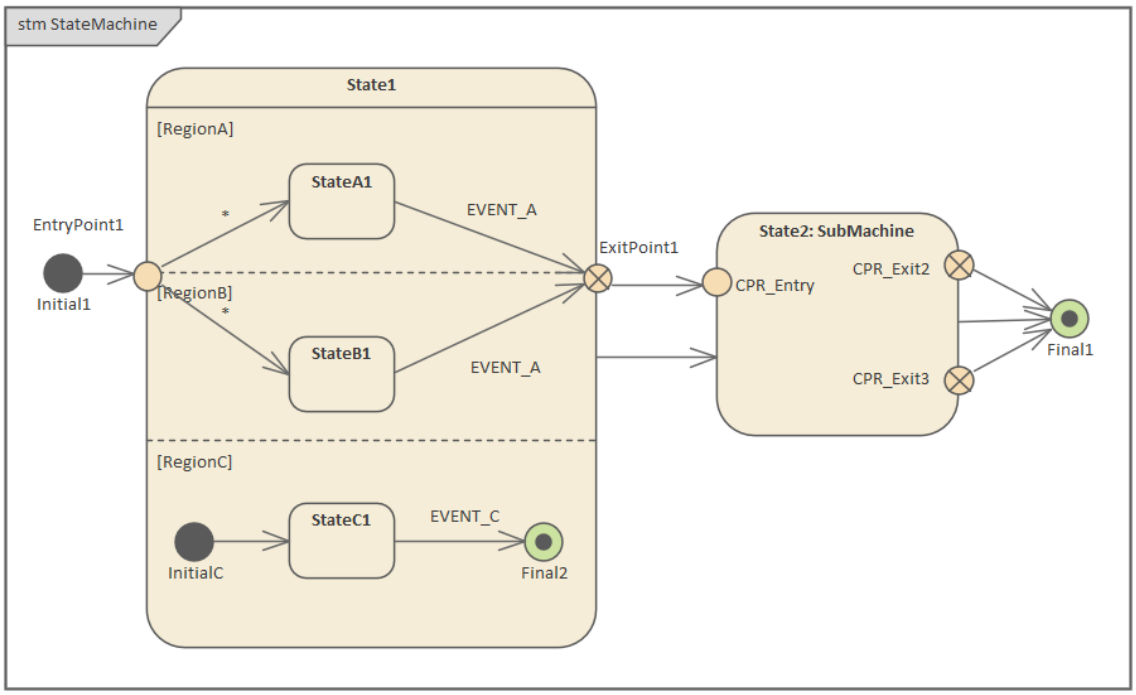
2. Double-cliquez sur RUN_TEST dans la fenêtre Simulation Event pour le déclencheur . L'événement NEW_REQUEST sera déclenché trois fois (par SEND_EVENT et BROADCAST_EVENT) et AUTHORIZED sera déclenché une fois par SEND_EVENT.

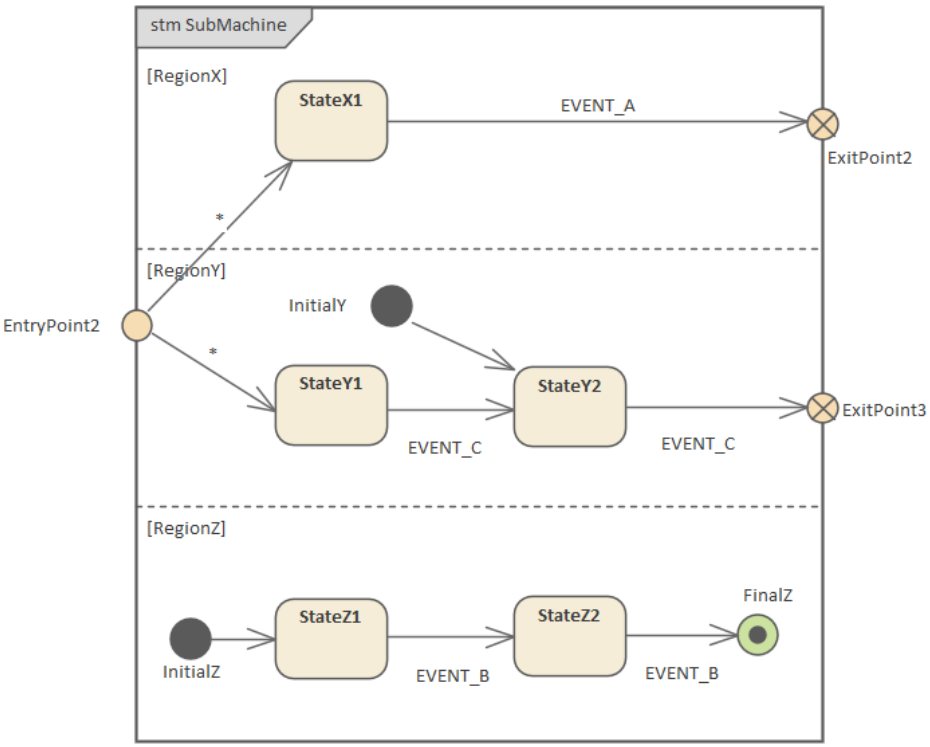


Type *dump* dans la ligne de commande de la fenêtre Simulation . Il reste une instance de NEW_REQUEST dans le pool d'événements. Le résultat correspond à notre test de déclenchement manuel.

Exemple : points d'entrée et de sortie (références de points de connexion)

Enterprise Architect fournit support pour les points d'entrée et de sortie, ainsi que pour les références de points de connexion. Dans cet exemple, nous définissons deux Statemachines pour *MyClass : Statemachine* et *SubMachine.*





- *L'État 1* est un State composite (également appelé State orthogonal car il comporte plusieurs régions) avec trois

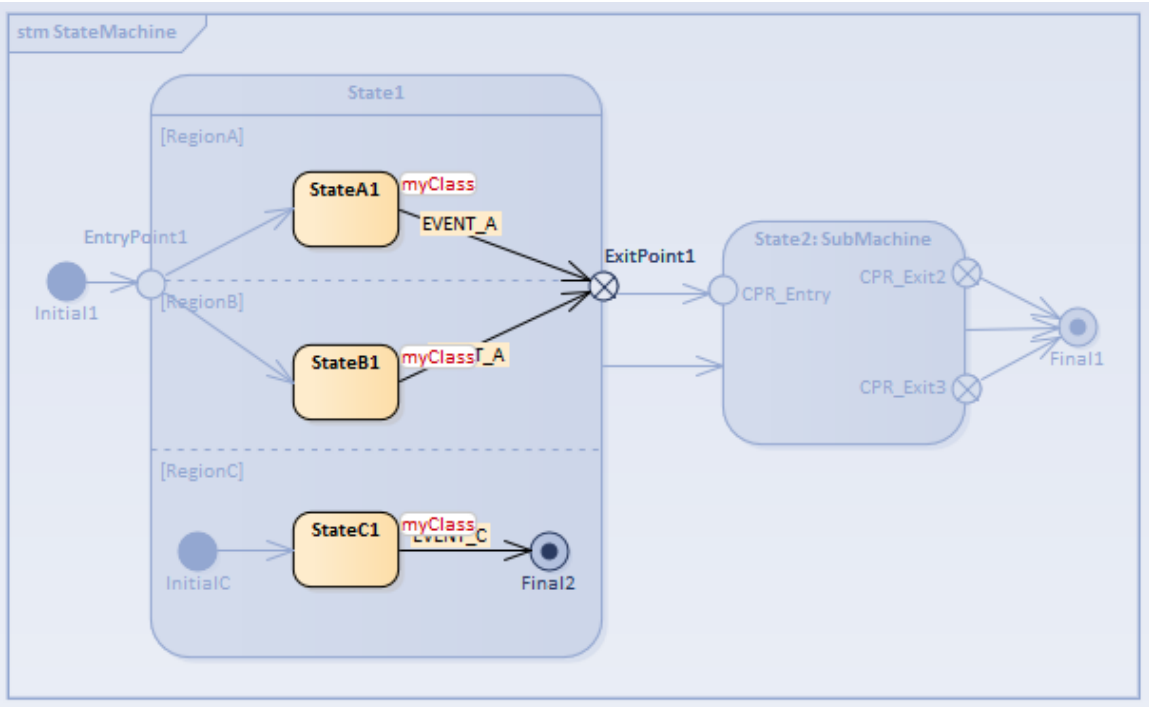
régions : *RégionA* , *RégionB* et *RégionC*

- *State2* est un State de sous-machine appelant *SubMachine* , qui possède trois régions : *RegionX* , *RegionY* et *RegionZ*
- *EntryPoint1* est défini sur *State1* pour activer deux des trois régions ; *EntryPoint2* est défini sur *SubMachine* pour activer deux des trois régions
- *ExitPoint1* est défini sur *State1* ; deux points de sortie *ExitPoint2* et *ExitPoint3* sont définis sur *SubMachine*
- Les références de points de connexion sont définies sur *State2* et se lient aux points d'entrée/sortie de la sous-machine de saisie
- Les nœuds initiaux sont définis pour démontrer l'activation par défaut des régions

Entrer d'un State : Point d'entrée Entrée

Point d'entrée 1* sur *État 1

Lorsqu'une transition ciblant *EntryPoint1* est activée, *State1* est activé suivi des régions contenues.

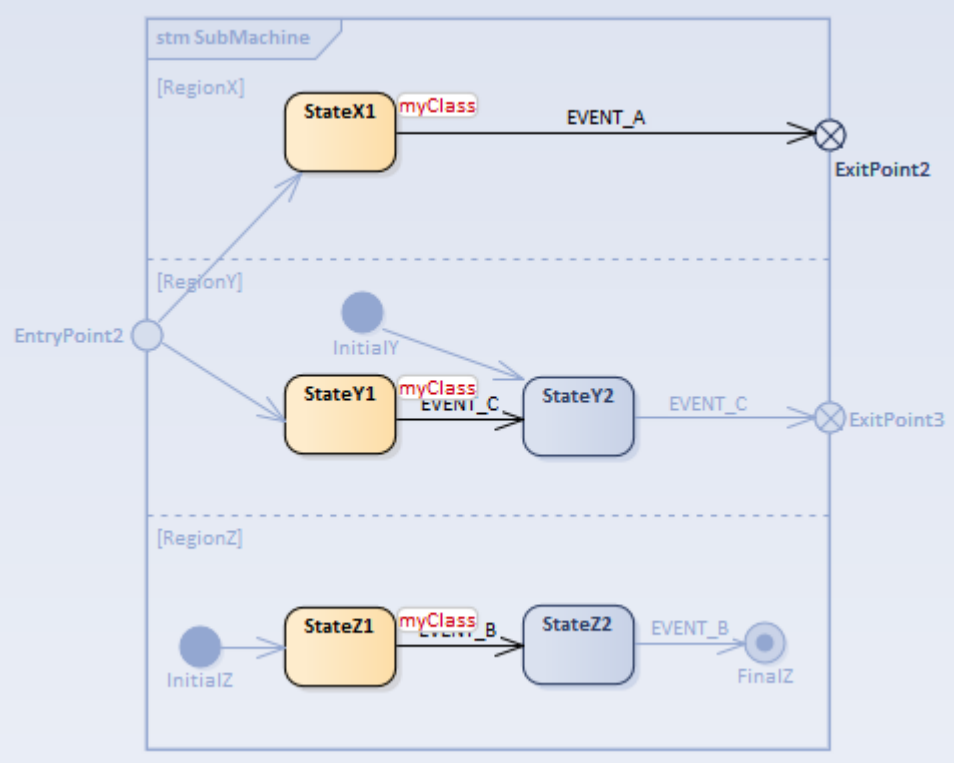


- L'activation explicite se produit pour *RegionA* et *RegionB* , car chacune d'elles est entrée par une transition se terminant sur l'un des sommets contenus dans la région
- L'activation par défaut se produit pour *RegionC* , car elle définit un pseudo-état initial *InitialC* et la transition provenant de l' *InitialC* vers *l'étatC1* démarre l'exécution

EntryPoint2* sur *SubMachine

La Séquence Déclencheur à simuler est : [EVENT_C, EVENT_A].

Lorsqu'une transition ciblant la référence de point de connexion *CPR_Entry* sur *State2* est activée, *State2* est activé, suivi de l'activation de la sous-machine via les points d'entrée de liaison.



- L'activation explicite se produit pour *RegionX* et *RegionY* , car chacune d'elles est entrée par une transition se terminant sur l'un des sommets contenus dans la région - *StateX1* dans *RegionX* , *StateY1* dans *RegionY*
- L'activation par défaut se produit pour *RegionZ* , car elle définit un pseudo-état initial *InitialZ* et la transition provenant d' *InitialZ* vers *StateZ1* démarre l'exécution

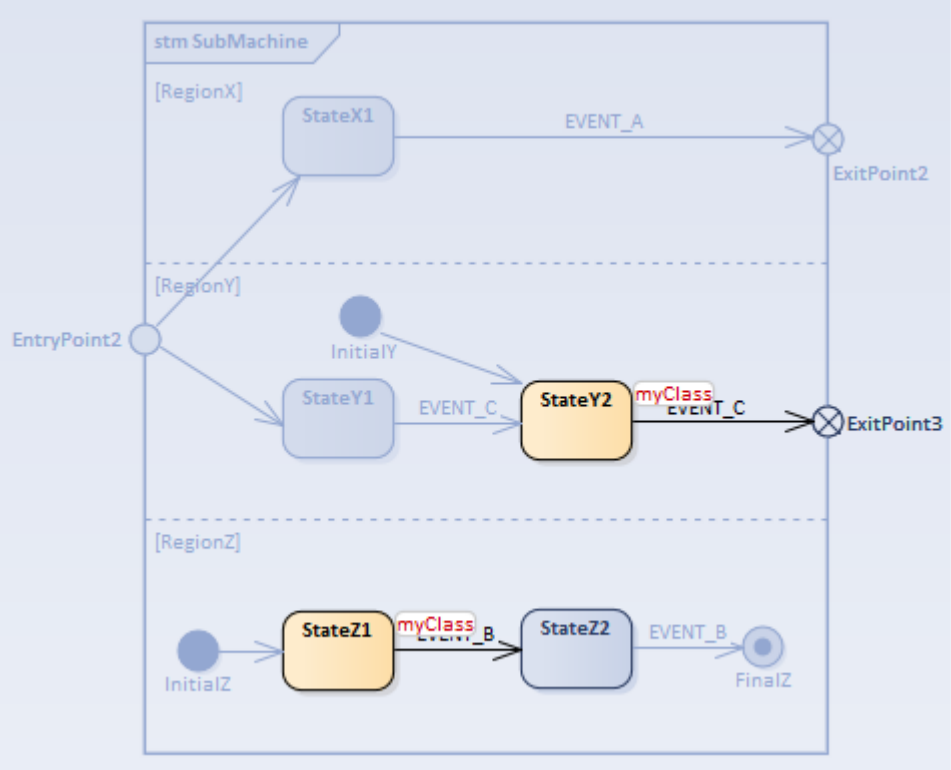
Entrer d'un State : Entrée par défaut

Cette situation se produit lorsque l' State composite est la cible directe d'une transition.

Entrée par défaut de *l'état 2*

La Séquence Déclencheur à simuler est : [EVENT_A, EVENTC].

Lorsqu'une transition ciblant directement *l'État 2* est activée, *l'État 2* est activé, suivi de l'activation par défaut pour toutes les régions de la sous-machine.



- State *de RegionX* est inactif car il ne définit pas de nœud initial
- *La région Y* est activée via *InitialY* et la transition vers *l'état Y2* est exécutée
- *RegionZ* est activé via *InitialZ* et la transition vers *StateZ1* est exécutée

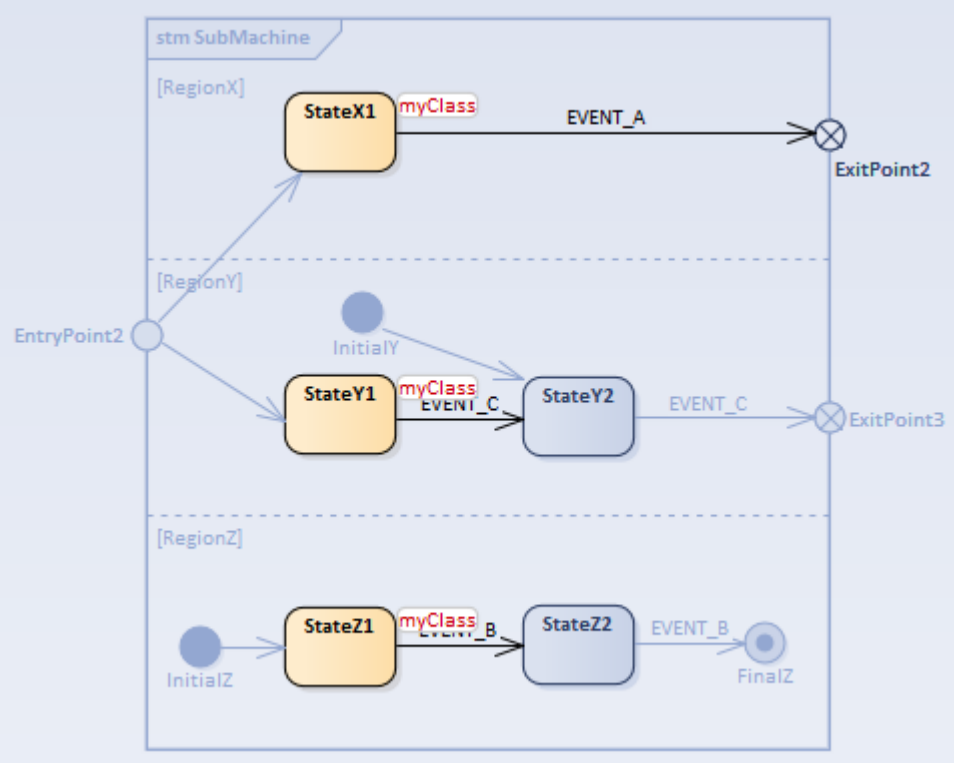
Sortie State

État 1 Sortie

- Séquence Déclencheur [EVENT_C, EVENT_A] : *la région C* est désactivée en premier, puis *la région A* et *la région B* ; une fois le comportement de sortie de *l'état 1* exécuté, la transition sortant de *ExitPoint1* est activée
- Séquence Déclencheur [EVENT_A, EVENT_C] : *les régions A* et *B* sont désactivées en premier, puis *la région C* ; une fois le comportement de sortie de *l'état 1* exécuté, la transition sortant directement de *l'état 1* est activée

Sortie de l'état 2

Déclencheur Séquence [EVENT_C, EVENT_A], donc l'état actuel ressemble à ceci :



- Séquence Déclencheur [EVENT_A, EVENT_C, EVENT_C, EVENT_B, EVENT_B] : *la région X* est désactivée en premier, puis *la région Y* et *la région Z* en dernier ; une fois le comportement de sortie de *l'état 2* exécuté, la transition sortant directement de *l'état 2* est activée
- Séquence Déclencheur [EVENT_A, EVENT_B, EVENT_B, EVENT_C, EVENT_C] : *RegionX* est désactivé en premier, puis *RegionZ* et *RegionY* est le dernier ; une fois le comportement de sortie de *State2* exécuté, la transition sortant de *CPR_Exit3* est activée (*ExitPoint3* sur *SubMachine* est lié à *CPR_Exit3* de *State2*)
- Séquence Déclencheur déclenchement [EVENT_C, EVENT_C, EVENT_B, EVENT_B, EVENT_A] : *la région Y* est désactivée en premier, puis *la région Z* et *la région X* en dernier ; une fois le comportement de sortie de *l'état 2* exécuté, la transition sortant de *CPR_Exit2* est activée (*ExitPoint2* sur *la sous-machine* est lié à *CPR_Exit2* de *l'état 2*)

Exemple : Pseudo-state Historique

L'historique State est un concept pratique associé aux régions d' States composites, selon lequel une région conserve la trace de la configuration dans laquelle se trouvait un State lors de sa dernière sortie. Cela permet de revenir facilement à cette configuration State , si nécessaire, lorsque la région redevient active (par exemple, après être revenue de la gestion d'une interruption), ou s'il existe une transition locale qui revient à son historique.

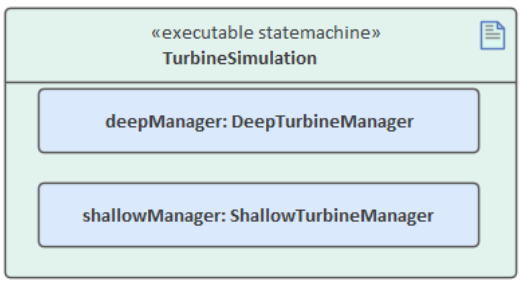
Enterprise Architect supporte deux types de Pseudo-state Historique :

- Historique profond - représentant la configuration complète de State de la visite la plus récente dans la région contenante ; l'effet est le même que si la transition se terminant sur le pseudo-état deepHistory s'était, à la place, terminée sur l' State le plus profond de la configuration de State préservé, y compris l'exécution de tous les comportements d'entrée rencontrés en cours de route
- Historique superficiel - représentant un retour uniquement au sous-état le plus élevé de la configuration State la plus récente, qui est saisi à l'aide de la règle de saisie par défaut

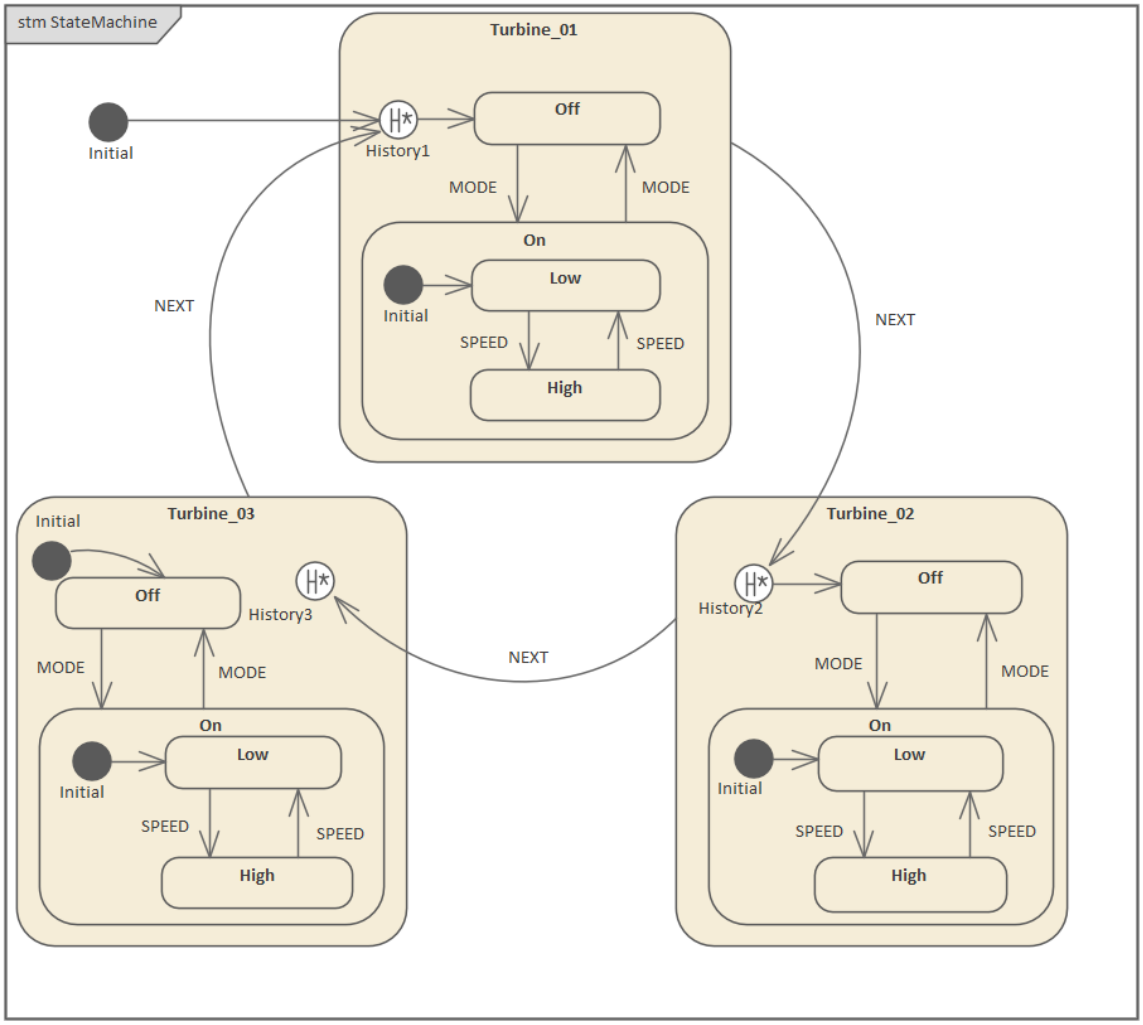
Dans cet exemple, les classes *DeepTurbineManager* et *ShallowTurbineManager* sont exactement les mêmes, sauf que la Statemachine contenue pour la première a un pseudo-état deepHistory et pour la seconde un pseudo-état shallowHistory.

Les deux Statemachines ont trois States composites : *Turbine_01* , *Turbine_02* et *Turbine_03* , chacun ayant States *Off* et *On* et un Pseudo-state Historique dans sa région.

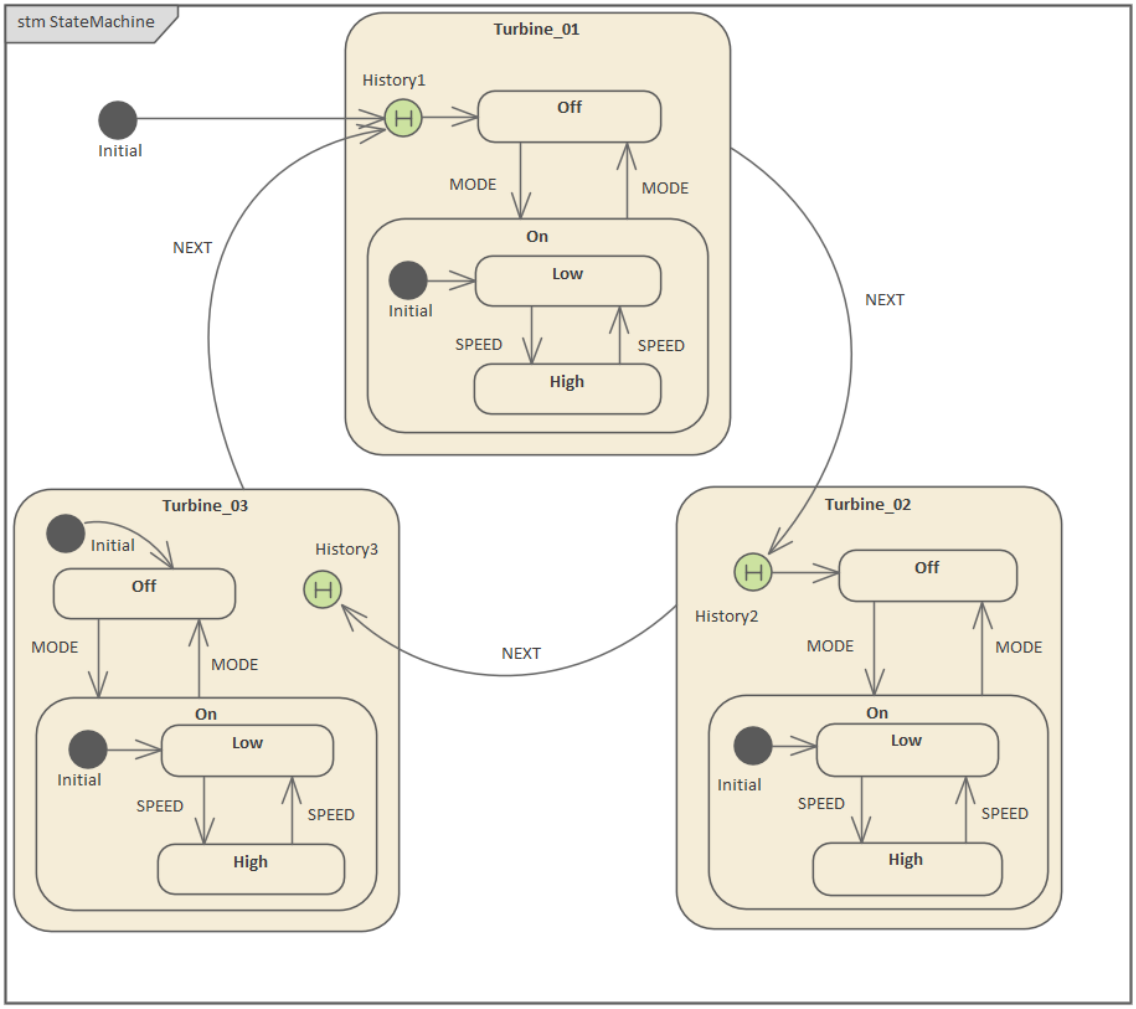
Afin de mieux observer la différence entre l'histoire profonde et l'histoire superficielle, nous exécutons les deux Statemachines dans une seule simulation.



La Statemachine dans *DeepTurbineManager* est illustrée dans ce diagramme :



La Statemachine dans *ShallowTurbineManager* est illustrée dans ce diagramme :

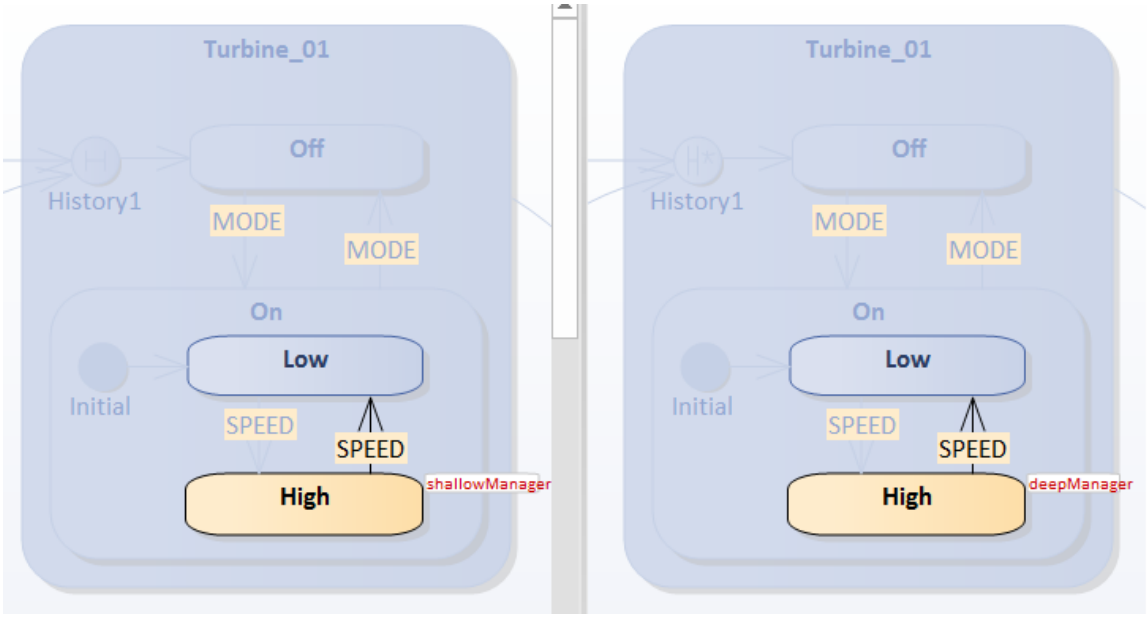


Conseil : Si vous cliquez-droit sur le nœud Historique du diagramme et sélectionnez l'option 'Avancé | Historique profond', vous pouvez basculer le type de Pseudo-state Historique entre superficiel et profond.

Première activation des States

Une fois la simulation démarrée, *Turbine_01* et son sous-état *Off* sont activés.

Séquence Déclencheur : [MODE, SPEED]

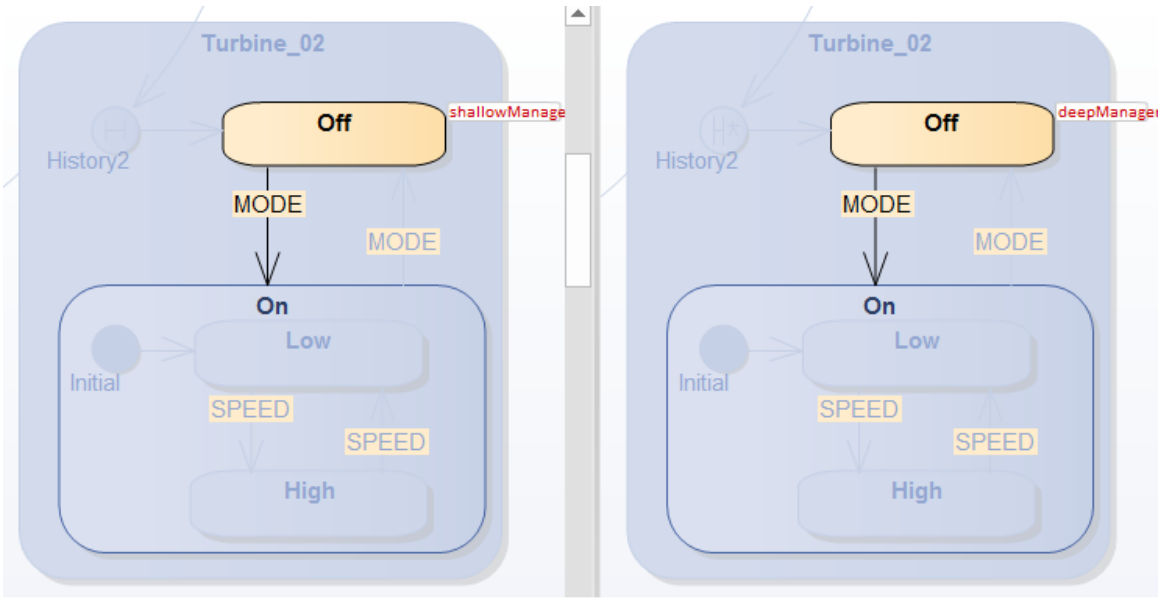


Ensuite, la configuration State actif comprend :

- Turbine_01
- Turbine_01.On
- Turbine_01.En.haut.

Ceci s'applique à la fois *à deepManager* et *à shallowManager.*

Séquence Déclencheur : [SUIVANT]



Cette séquence de traces peut être observée depuis la fenêtre Simulation (Simuler > Simulation Dynamique > Simulateur > Ouvrir la fenêtre Simulation) :

01 peu profondManager [ShallowTurbineManager].StateMachine_Turbine_01_On_High SORTIE

02 peu profondManager [ShallowTurbineManager].StateMachine_Turbine_01_On EXIT

03 ShallowManager[ShallowTurbineManager].StateMachine_Turbine_01 SORTIE

04 effet de ShallowTurbineManager[ShallowTurbineManager].Turbine_01__TO__History2_105720_61730

05 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_02 ENTRÉE

06 gestionnaire peu profond [gestionnaire de turbine peu profond].StateMachine_Turbine_02 DO

07 shallowManager[ShallowTurbineManager].History2_105720__TO__Off_61731 Effet

08 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_02_Off ENTRÉE

09 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_02_Off FAIRE

Note : étant donné que *deepManager* a exactement la même trace que *shallowManager* , la trace de *deepManager* est filtrée à partir de cette séquence.

Nous pouvons apprendre que :

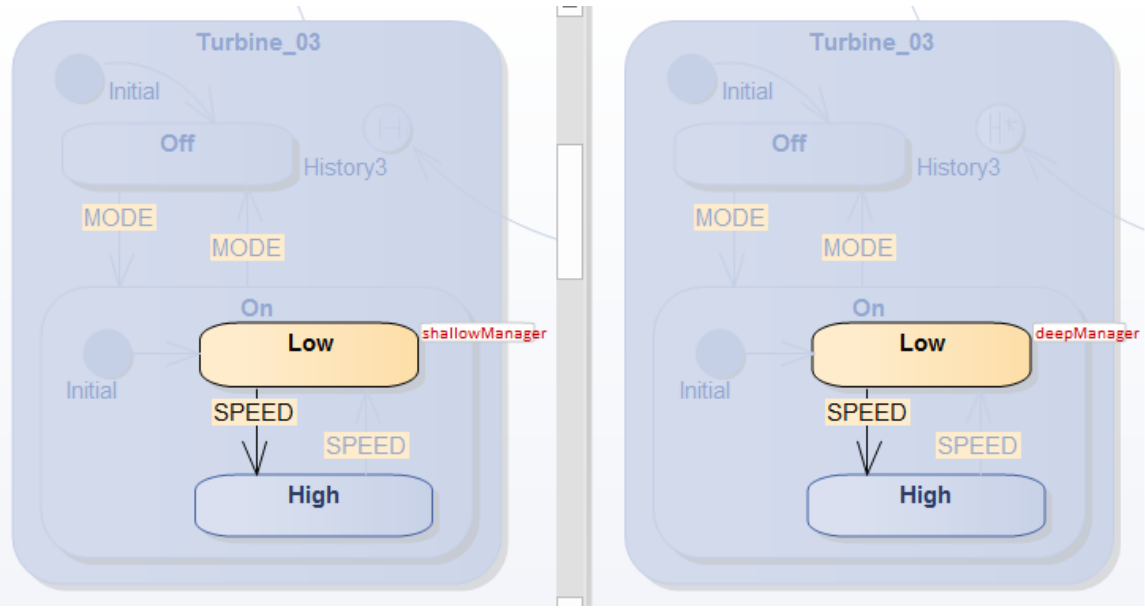
- La sortie d'un State composite commence par l' State le plus interne dans la configuration State actif (voir les lignes 01 à 03 dans la séquence de trace)
- La transition d'historique par défaut n'est prise que si l'exécution mène au nœud Historique (voir ligne 04) et que l' State n'a jamais été actif auparavant (voir ligne 07)

Ensuite, la configuration State actif comprend :

- Turbine_02
- Turbine_02.Arrêt

Ceci s'applique à la fois *à deepManager* et *à shallowManager.*

Séquence Déclencheur : [NEXT, MODE]



Cette séquence de traces peut être observée à partir de la fenêtre Simulation :

Déclencheur [SUIVANT]

01 ShallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off SORTIE

02 Gestionnaire peu profond [Gestionnaire de turbines peu profondes]. StateMachine_Turbine_02 SORTIE

03 effet de ShallowTurbineManager[ShallowTurbineManager].Turbine_02__TO__History3_105713_61725

04 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03 ENTRÉE

05 gestionnaire peu profond [gestionnaire de turbine peu profond].StateMachine_Turbine_03 DO

06 shallowManager[ShallowTurbineManager].Initial_105706__TO__Off_61718 Effet

07 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_Off ENTRÉE

08 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_Off FAIRE

Déclencheur [MODE]

Message omis...

Note : étant donné que *deepManager* a exactement la même trace que *shallowManager,* la trace de *deepManager* est filtrée de cette séquence.

Nous pouvons apprendre que :

- Comme il n'y a pas de transition d'historique par défaut définie pour *History3* , l'entrée par défaut standard de l' State est effectuée ; un nœud initial est trouvé dans la région contenue par *Turbine_03,* donc la transition provenant d' *Initial* est activée (voir ligne 06)

Ensuite, la configuration de l'état actif comprend :

- Turbine_03
- Turbine_03.On
- Turbine_03.Marche.Basse

Ceci s'applique à la fois *à deepManager* et *à shallowManager.*

Entrée dans l'histoire des States

À titre de référence, nous montrons l' instantané de l'historique profond de chaque turbine après sa première activation :

Turbine_01

- Turbine_01.On
- Turbine_01.En.haut.

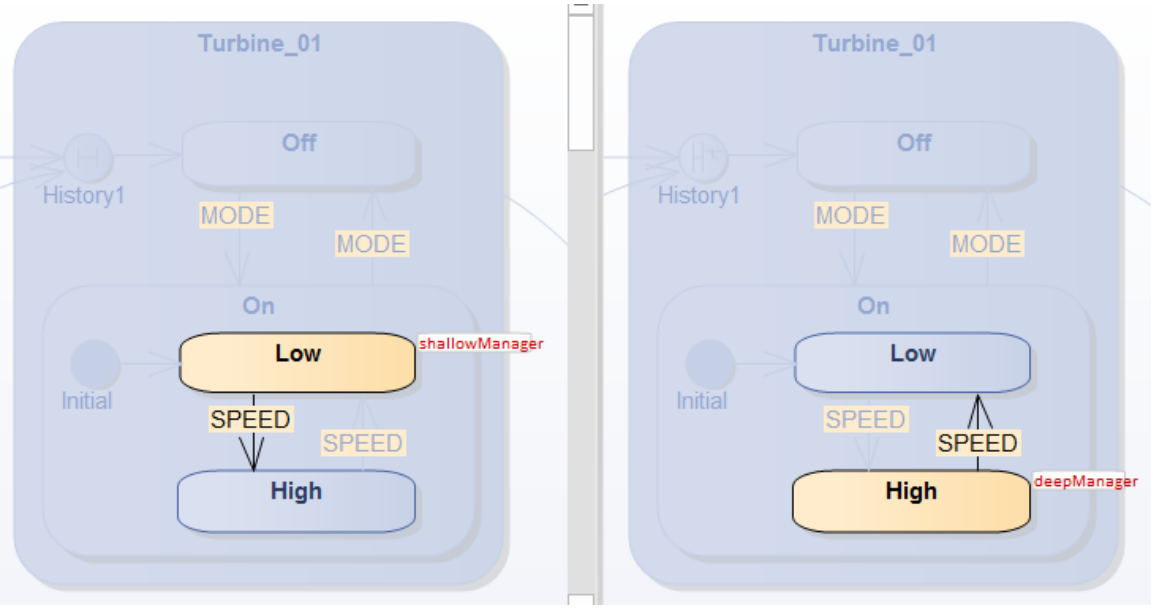
Turbine_02

- Turbine_02.Arrêt

Turbine_03

- Turbine_03.On
- Turbine_03.Marche.Basse

Lorsque nous appuierons sur Déclencheur NEXT, Turbine_01 sera à nouveau activée.



Cette séquence de traces peut être observée à partir de la fenêtre Simulation :

Pour le gestionnaire peu profond :

Déclencheur [SUIVANT]

01 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low SORTIE

02 peu profondManager [ShallowTurbineManager].StateMachine_Turbine_03_On EXIT

03 Gestionnaire peu profond [Gestionnaire de turbines peu profondes]. StateMachine_Turbine_03 SORTIE

04 effet de ShallowTurbineManager[ShallowTurbineManager].Turbine_03__TO__History1_105711_61732

05 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_01 ENTRÉE

06 gestionnaire peu profond [gestionnaire de turbine peu profond].StateMachine_Turbine_01 DO

07 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_01_On ENTRÉE

08 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_01_On FAIRE

09 effet de ShallowTurbineManager[ShallowTurbineManager].Initial_105721__TO__Low_61729

10 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_01_On_Low ENTRÉE

11 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_01_On_Low DO

Nous pouvons apprendre que :

- Le nœud shallowHistory restaure Turbine_01 jusqu'à Turbine_01.On
- Ensuite, la région contenue par Composite State Turbine_01.On sera activée par le nœud *initial* , qui s'est activé à *Low*

Pour deepManager :

Déclencheur [SUIVANT]

01 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On_Low SORTIE

02 deepManager[DeepTurbineManager].StateMachine_Turbine_03_À la sortie

03 deepManager[DeepTurbineManager].StateMachine_Turbine_03 SORTIE

04 deepManager[DeepTurbineManager].Turbine_03__TO__History1_105679_61708 Effet

05 deepManager[DeepTurbineManager].StateMachine_Turbine_01 ENTRÉE

06 deepManager[DeepTurbineManager].StateMachine_Turbine_01 FAIRE

07 deepManager[DeepTurbineManager].StateMachine_Turbine_01_On ENTRÉE

08 deepManager[DeepTurbineManager].StateMachine_Turbine_01_On_High ENTRÉE

Nous pouvons apprendre que :

- Le nœud *deepHistory* restaure Turbine_01 jusqu'à Turbine_01.On.High

Déclencheur [NEXT] pour sortir de Turbine_01 et activer Turbine_02

Les deux *shallowManager* et *deepManager* activent Turbine_02.Off, qui est l'historique instantané lorsqu'ils sont sortis.

Déclencheur [NEXT] pour sortir de la Turbine_02 et activer la Turbine_03

Les *fonctions shallowManager* et *deepManager* activent toutes deux Turbine_03.On.Low. Cependant, les séquences de *ces fonctions* sont différentes.

Pour *le gestionnaire de profondeur* , le *paramètre shallowHistory* ne peut restaurer que jusqu'à Turbine_03.On. Étant donné qu'un nœud *initial* est défini dans Turbine_03.On, la transition provenant d' *Initial* sera activée et Turbine_03.On.Low sera atteinte.

01 ShallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off SORTIE

02 Gestionnaire peu profond [Gestionnaire de turbines peu profondes]. StateMachine_Turbine_02 SORTIE

03 effet de ShallowTurbineManager[ShallowTurbineManager].Turbine_02__TO__History3_105713_61725

04 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03 ENTRÉE

05 gestionnaire peu profond [gestionnaire de turbine peu profond].StateMachine_Turbine_03 DO

06 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_On ENTRÉE

07 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_On FAIRE

08 effet de ShallowTurbineManager[ShallowTurbineManager].Initial_105727__TO__Low_61728

09 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low ENTRÉE

10 peu profondManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low DO

Pour *deepManager* , le *deephistory* peut restaurer jusqu'à Turbine_03.On.Low directement.

01 deepManager[DeepTurbineManager].StateMachine_Turbine_02_Off SORTIE

02 deepManager[DeepTurbineManager].StateMachine_Turbine_02 SORTIE

03 deepManager[DeepTurbineManager].Turbine_02__TO__History3_105680_61701 Effet

04 deepManager[DeepTurbineManager].StateMachine_Turbine_03 ENTRÉE

05 deepManager[DeepTurbineManager].StateMachine_Turbine_03 FAIRE

06 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On ENTRÉE

07 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On_Low ENTRÉE

Macro d'événement : EVENT_PARAMETER

EVENT_PARAMETER est une macro de fonction utilisée pour accéder aux attributs d'une instance de signal, dans le comportement de State , la protection et l'effet de la transition. Cette macro sera étendue au code exécutable selon le langage de simulation.

Accès à la définition de macro par défaut

Ruban | Développer | Code source | Options | Modifier le code Gabarits | Langue | Paramètre d'événement Stm

Format d'utilisation

%EVENT_PARAMETER(Type de signal, Nom d'attribut du signal)%

Par exemple : un signal « MySignal » possède deux attributs, « foo : int » et « bar : int » ; ces cas d'utilisation sont valides :

- Effet de la transition : %EVENT_PARAMETER(MySignal, foo)%
- Comportement de State : %EVENT_PARAMETER(MySignal, bar)%
- Garde de transition : %EVENT_PARAMETER(MySignal, bar)% > 10
- Comportement de State : %EVENT_PARAMETER(MySignal, bar)%++
- Tracez la valeur dans la fenêtre Simulation : %TRACE(EVENT_PARAMETER(MySignal, foo))%

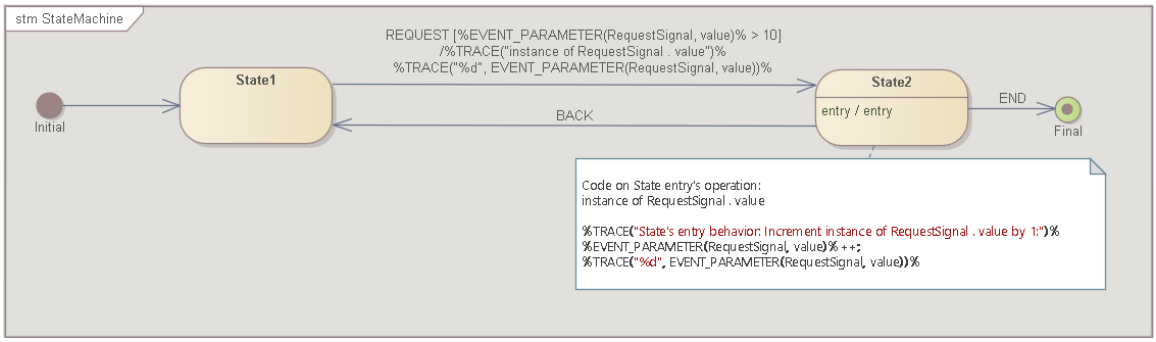
Exemple d'extension macro

Pour le Signal "MySignal" avec attribut " valeur ", Exemples d'expansion de macro pour Transition avec déclencheur du signal :%EVENT_PARAMETER(MySignal, valeur)%

C	((MonSignal*)signal)-> valeur
C++	static_cast<MonSignal*>(signal)-> valeur
C#	((MonSignal)signal). valeur
Java	((EventProxy.MySignal)signal). valeur
JavaScript	signal. valeur

Exemple

Cet exemple montre comment utiliser EVENT_MACRO dans l'effet de Transition, la garde et le comportement de State .



Pendant l'exécution de la simulation,

(1) déclencheur REQUEST et préciser le numéro 1 pour l'attribut valeur .

Étant donné que la condition de garde est fausse, l'état actif restera State1.

(2) déclencheur REQUEST et spécifiez le numéro 11 pour l'attribut valeur .

Étant donné que la condition de garde est vraie, l'état actif passera de l'État 1 à l'État 2 ;

L'effet de transition est exécuté. Ici, nous avons tracé la valeur d'exécution de l'attribut du signal jusqu'à la fenêtre de simulation.

Le comportement de State2 est exécuté. Ici, nous avons incrémenté la valeur d'exécution de l'attribut du signal et l'avons tracée jusqu'à la fenêtre de simulation.

[32608107] [Partie 1 : TransactionServer] Effet de transition : Initial_4019__TO__State1_4420

[32608118] [Partie 1 : TransactionServer] Comportement de l'entrée : StateMachine_State1

[32608124] [Partie 1 : TransactionServer] Comportement : StateMachine_State1

[32608877] [Partie 1 : TransactionServer] Achèvement : TransactionServer_StateMachine_State1

[32608907] En attente du Déclencheur

[32613165] Commande : **diffusion REQUEST.RequestSignal(1)**

[32613214] [Part1:TransactionServer] Événement mis en file d'attente : REQUEST.RequestSignal(valeur :1)

[32613242] [Part1:TransactionServer] Événement envoyé : REQUEST.RequestSignal(valeur : 1)

[32613279] En attente du Déclencheur

[32619541] Commande : **diffusion REQUEST.RequestSignal(11)**

[32619546] [Part1:TransactionServer] Événement mis en file d'attente : REQUEST.RequestSignal(valeur :11)

[32619551] [Part1:TransactionServer] Événement envoyé : REQUEST.RequestSignal(valeur :11)

[32619557] [Partie 1 : TransactionServer] Comportement de sortie : StateMachine_State1

[32619562] [Partie 1 : TransactionServer] Effet de transition : State1__TO__State2_4421

[32619567] **instance de RequestSignal . valeur**

[32619571] **11**

[32619576] [Partie 1 : TransactionServer] Comportement de l'entrée : StateMachine_State2

[32619584] **Comportement de l'entrée State : incrémenter l'instance de RequestSignal. valeur de 1 :**

[32619590] **12**

[32619594] [Partie 1 : TransactionServer] Comportement : StateMachine_State2

[32620168] [Partie 1 : TransactionServer] Achèvement : TransactionServer_StateMachine_State2

[32620211] En attente du Déclencheur

[32622266] Commande : diffusion END

[32622272] [Partie 1 : TransactionServer] Événement mis en file d'attente : FIN

[32622310] [Partie 1 : TransactionServer] Événement envoyé : END

[32622349] [Partie 1 : TransactionServer] Comportement de sortie : StateMachine_State2

[32622359] [Partie 1 : TransactionServer] Effet de transition : State2__TO__Final_4023_4423

[32622896] [Partie 1 : TransactionServer] Achèvement : TransactionServer_VIRTUAL_SUBMACHINESTATE

Limitations et solutions de contournement

Étant donné que l'extension de macro implique un transtypage, il est de la responsabilité de l'utilisateur de s'assurer que le transtypage est valide.

Et voici des solutions de contournement pour certains cas courants dans lesquels le casting de type rencontrera des problèmes dans le modèle.

- **Une transition a plusieurs déclencheurs de différents types de signaux**

Par exemple, une transition a triggerA(SignalA spécifié) et triggerB(SignalB spécifié) ; une macro %EVENT_PARAMETER(SignalA, attributeOfA)% ne fonctionnera pas lorsque cette transition est déclenchée par triggerB.

Nous suggérons de créer deux transitions, une avec déclencheurA (SignalA spécifié) et l'autre avec déclencheurB (SignalB).

- **Un State est la cible de multiples transitions de différents déclencheurs de signaux**

Par exemple, TransitionA et TransitionB ciblent toutes deux MyState, TransitionA est déclenchée par SignalA et TransitionB est déclenchée par SignalB.

Si EVENT_PARAMETER est utilisé dans le code de comportement de l'état, une macro ne pourra pas fonctionner pour les deux cas.

Nous suggérons de déplacer la logique traitant de l'attribut du signal du comportement de State vers l'effet de la transition.

- **Personnaliser gabarit et le code généré**

L'utilisateur peut également modifier le gabarit par défaut pour ajouter l'identification Type d' Exécuter (RTTI) avant la conversion de type. L'utilisateur peut également modifier le code généré après la génération, ce qui peut également satisfaire à l'objectif de simulation après la compilation.

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

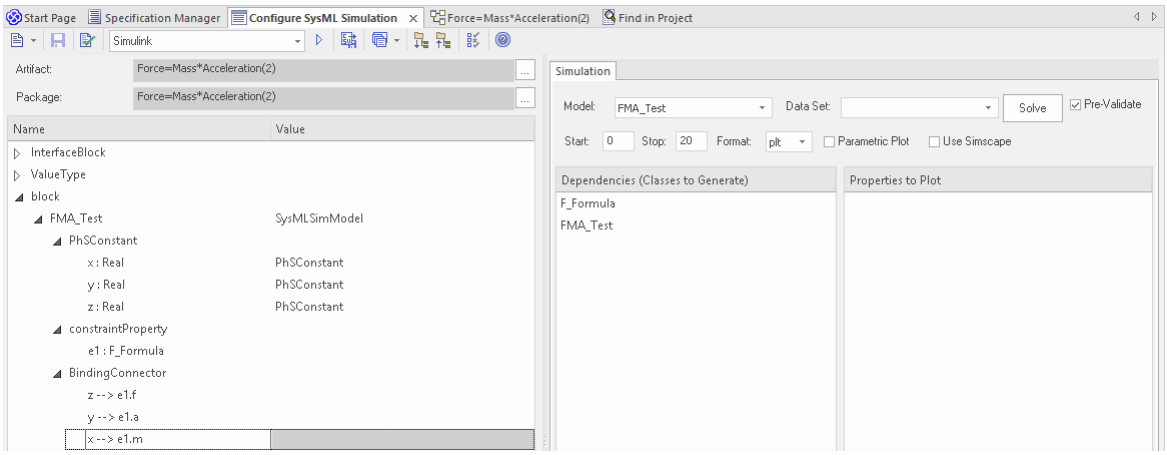
Ces sections décrivent le processus de définition d'un modèle Paramétriques , l'annotation du modèle avec des informations supplémentaires pour piloter une simulation et l'exécution d'une simulation pour générer un graphique des résultats.

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.

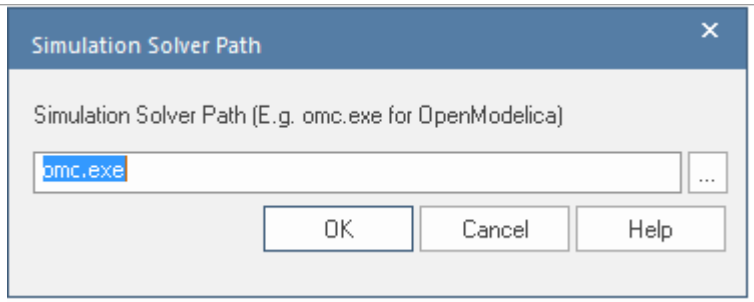


Accéder

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

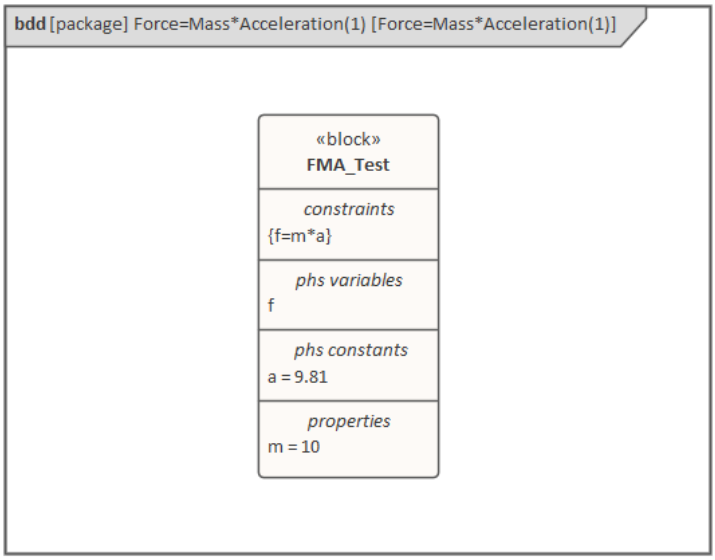
Accéder

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

Définition d'équations de contraintes en ligne sur un Bloc

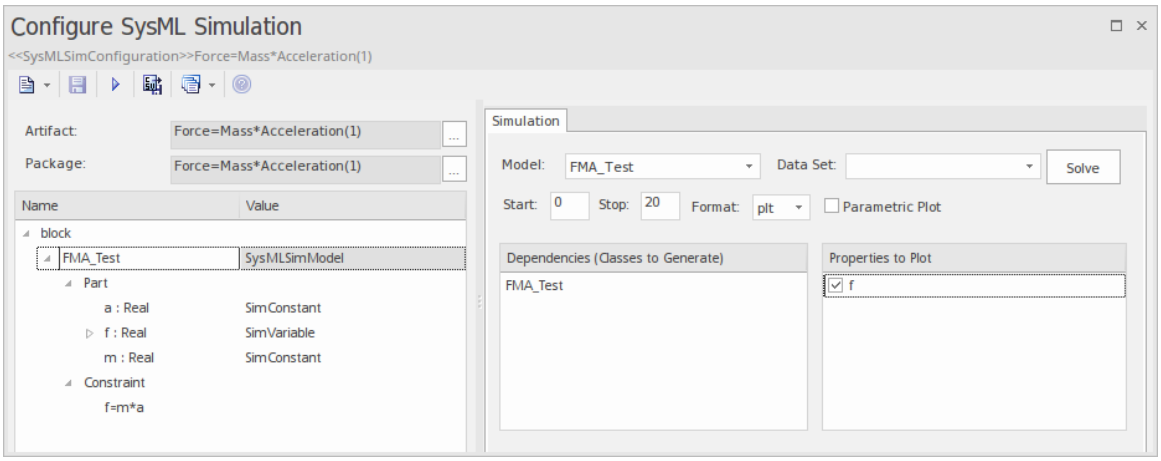
Définir des contraintes directement dans un Bloc est simple et constitue le moyen le plus simple de définir des équations de contraintes.

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.

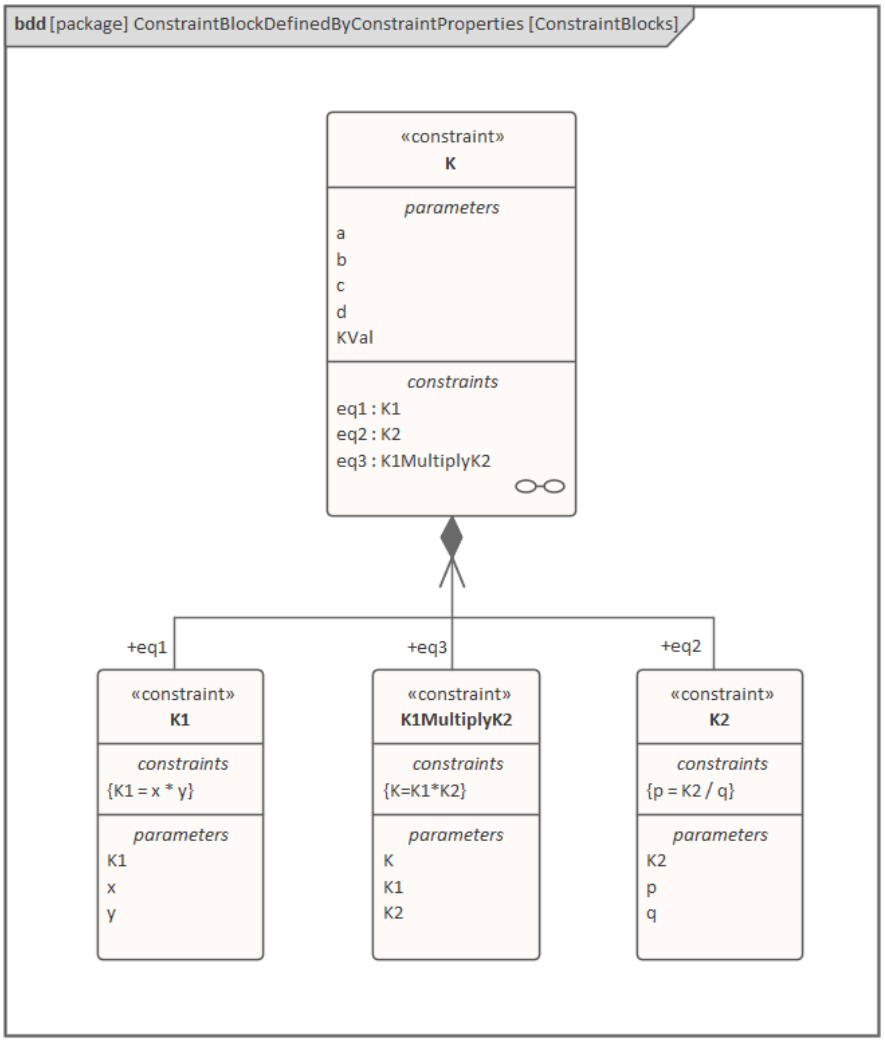


Un graphique doit être tracé avec f = 98,1 (qui vient de 10 * 9,81).

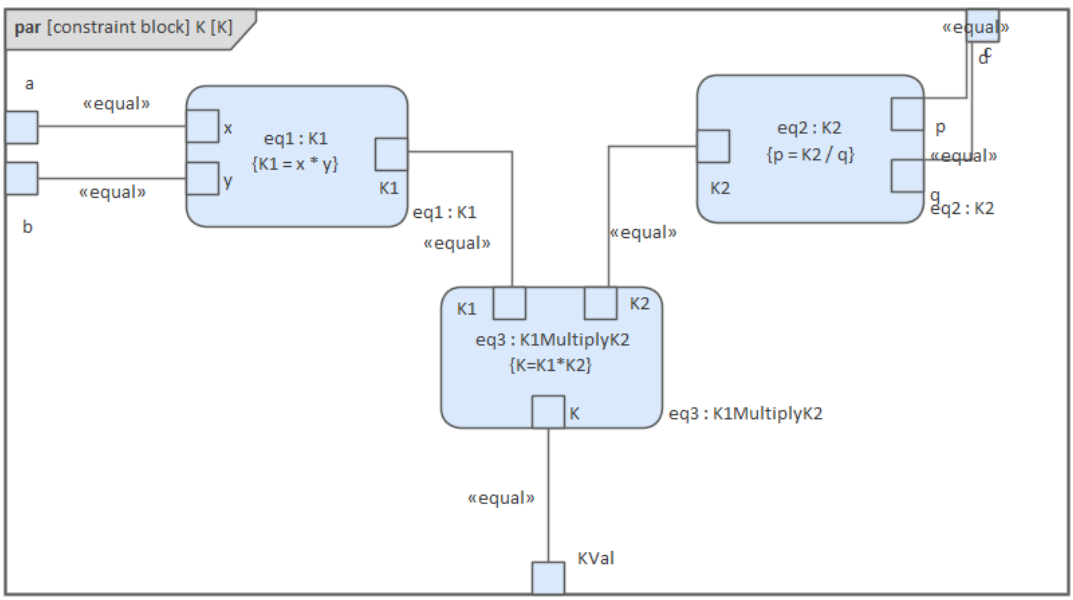
Propriétés de contrainte connectée

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

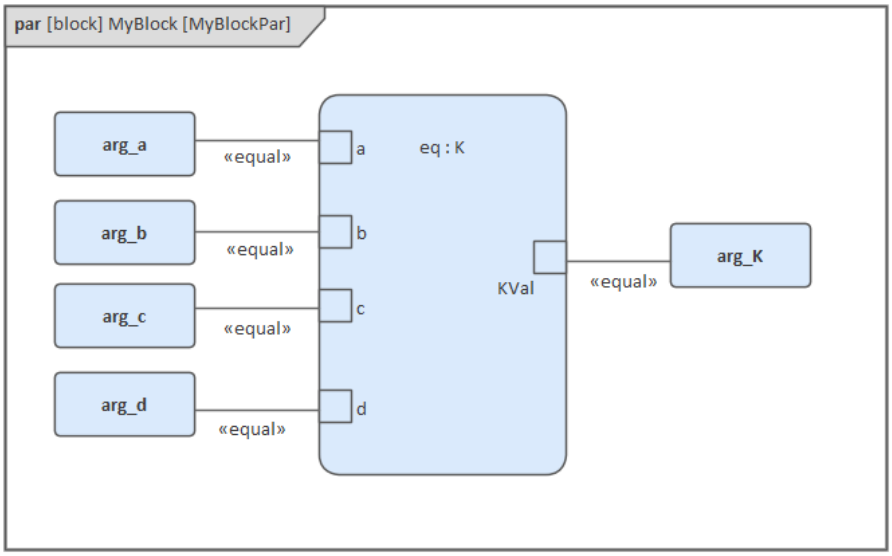
Dans cette figure, ConstraintBlock 'K' définit les paramètres 'a', 'b', 'c', 'd' et 'KVal', ainsi que trois propriétés de contrainte 'eq1', 'eq2' et 'eq3', typées respectivement 'K1', 'K2' et 'K1MultiplyK2'.



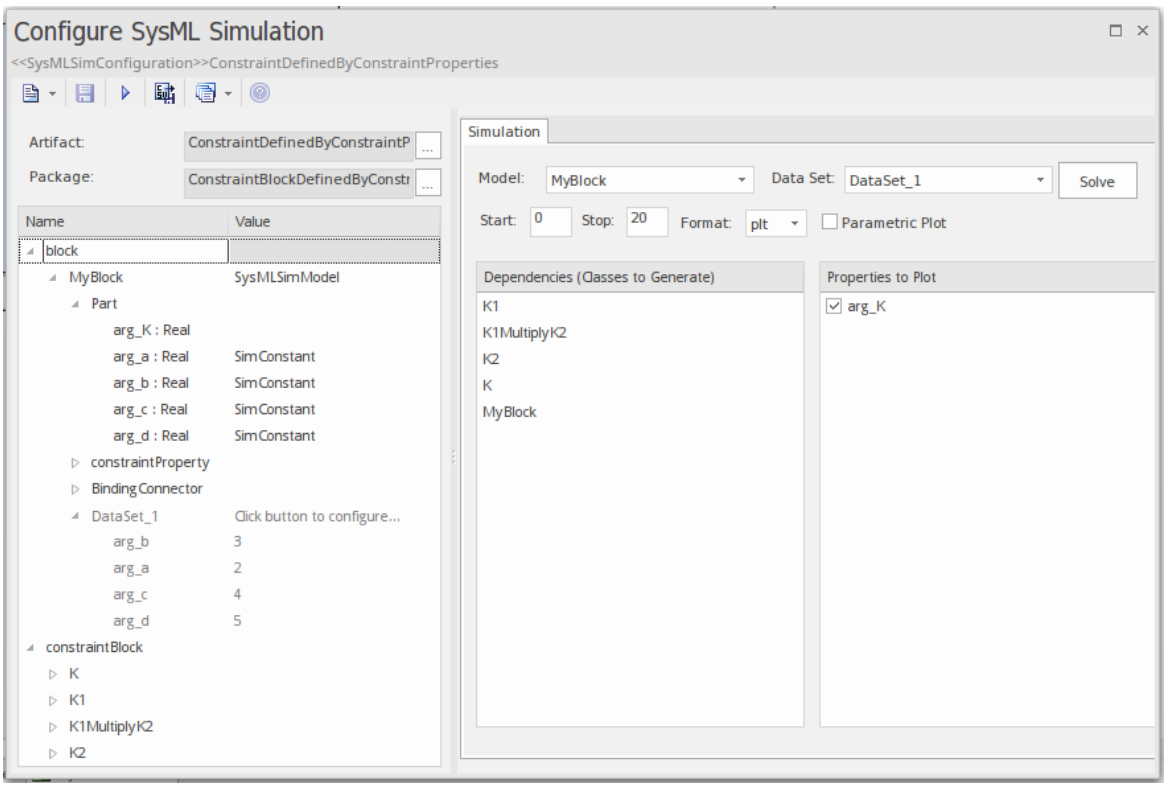
Créez un diagramme Paramétriques dans ConstraintBlock 'K' et connectez les paramètres aux propriétés de contrainte avec des connecteurs de liaison, comme indiqué :



- 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.

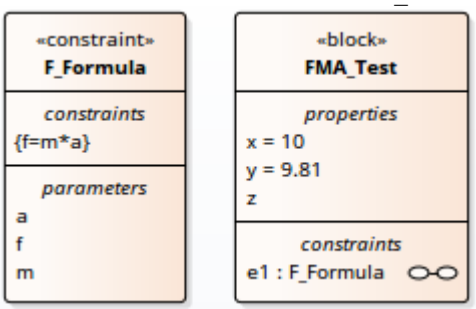
Nous pouvons facilement résoudre K2 comme étant p * q dans cet exemple, mais dans certains exemples complexes, il est extrêmement difficile de résoudre une variable à partir d'une équation ; cependant, Enterprise Architect SysMLSim peut toujours y parvenir.

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

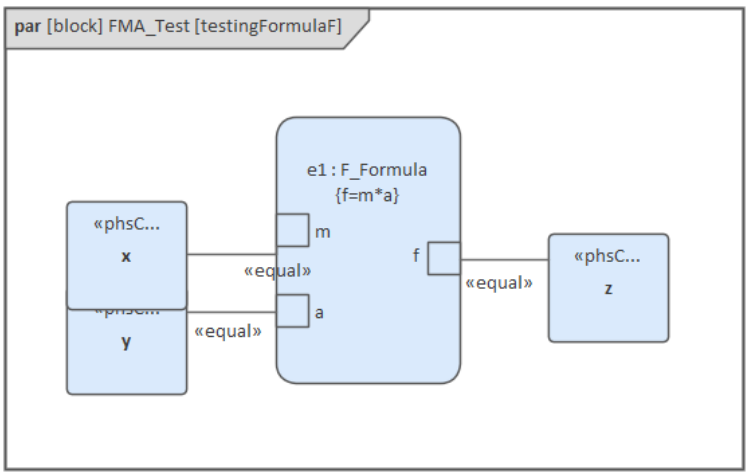
Si une équation est couramment utilisée dans plusieurs blocs, un bloc de contrainte peut être créé pour être utilisé comme propriété de contrainte dans chaque Bloc . Voici les modifications que nous apportons, sur la base de l'exemple précédent :

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



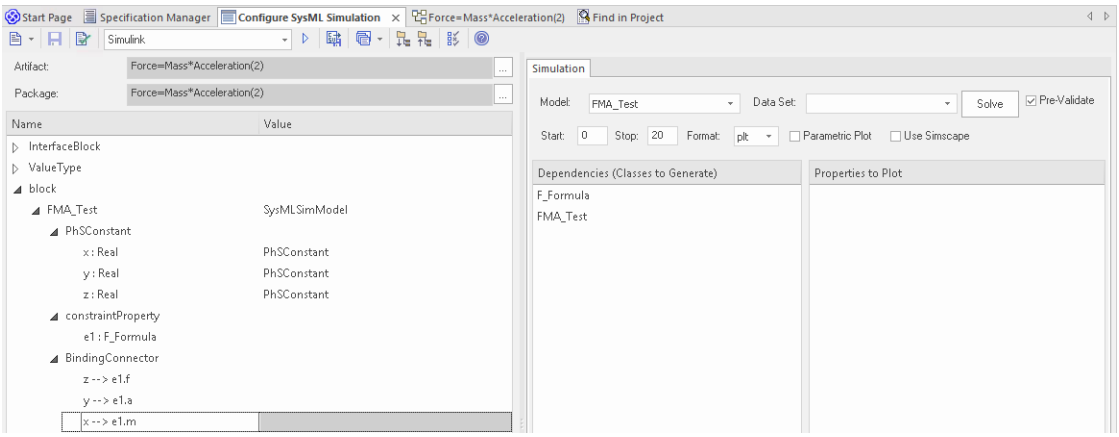
Conseil : Le type primitif 'Real' sera appliqué si les types de propriété sont vides

- 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 :

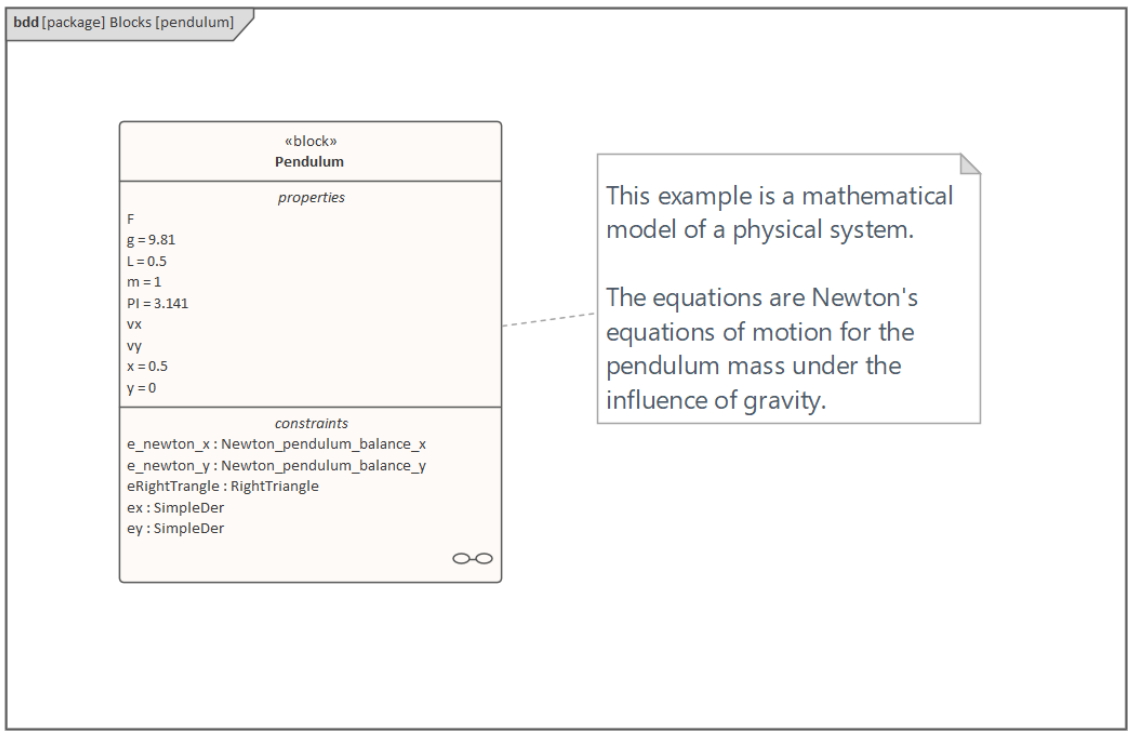
source.pv = résistance.nv;

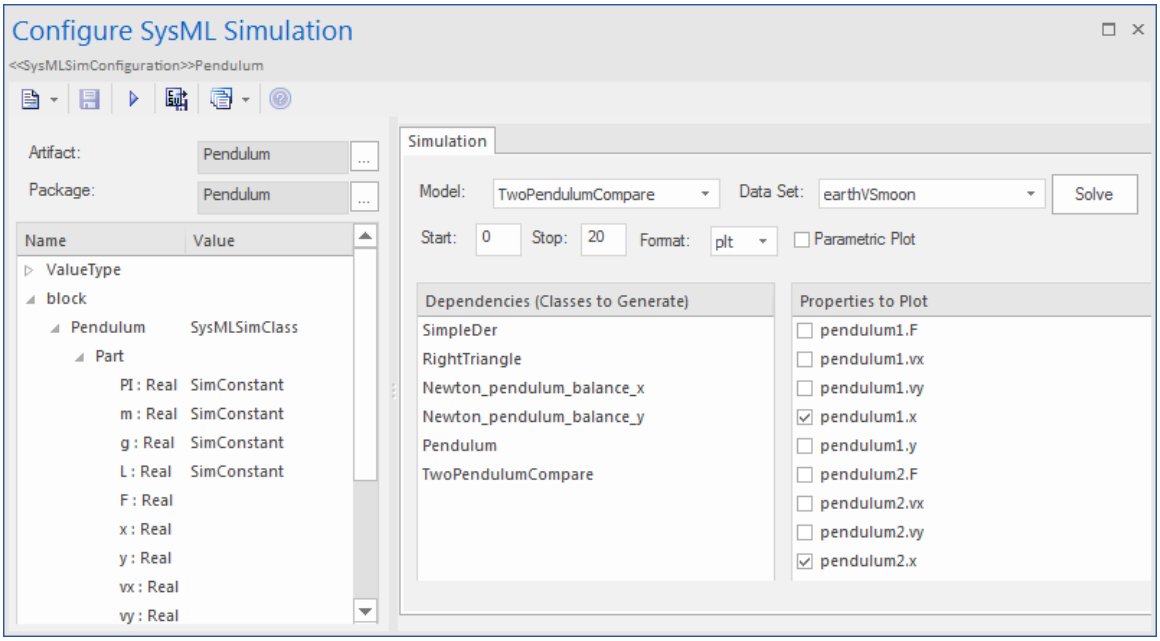
source.pi + résistance.ni = 0;

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 :

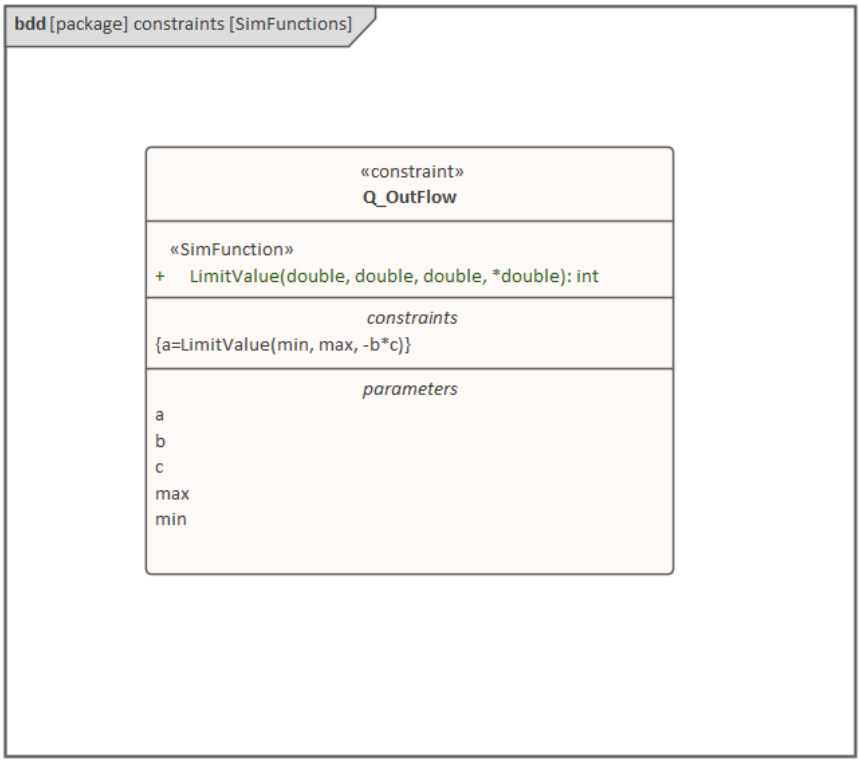
classe Pendule
paramètre PI réel = **3,141 ;**
paramètre Réel m = **1** ;
paramètre réel g = **9,81** ;
paramètre Réel L = **0,5** ;
Réel F;
Réel x **(début=0,5)** ;
Réel y **(début=0)** ;
VX réel;
Vrai vy;
.....
équation
.....
fin du pendule;

- 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;

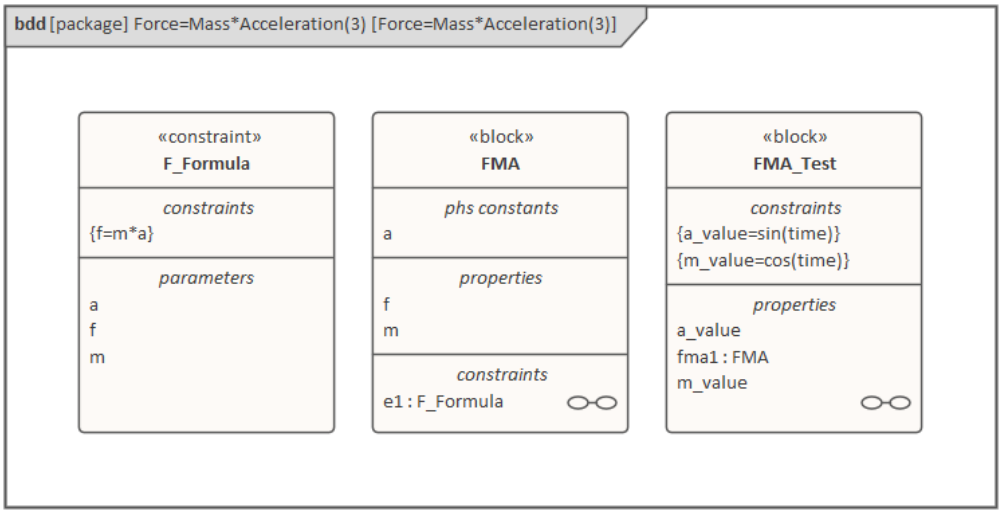
sortie Real pLim;

algorithme

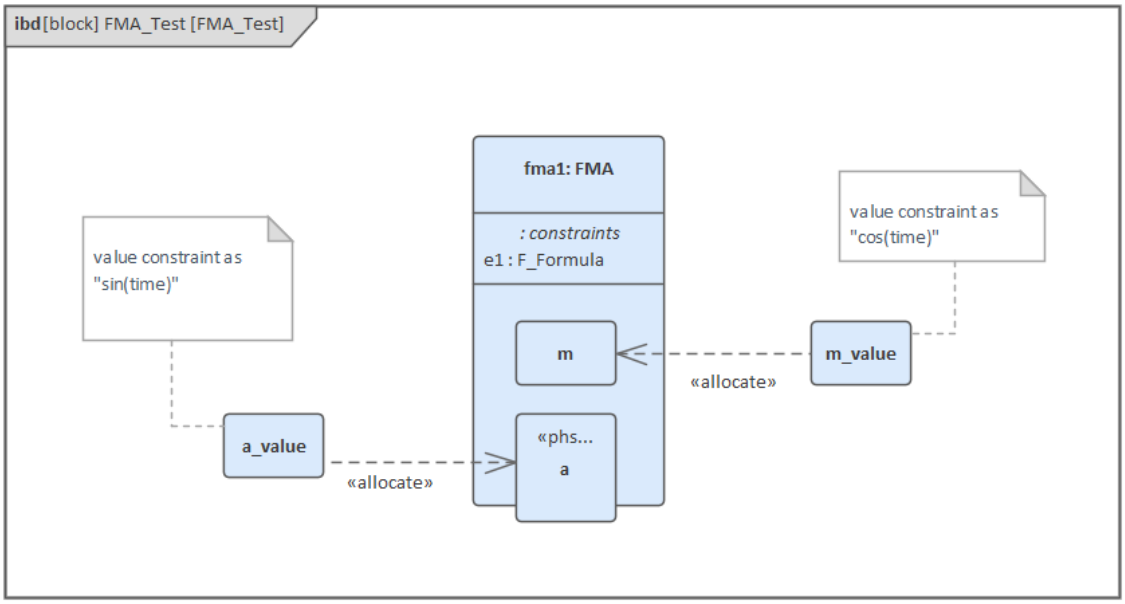
pLim :=
si p > pMax alors
pMax
sinon si p < pMin alors
pMin
autre
p;
fin de LimitValue;

Répartition de la valeur

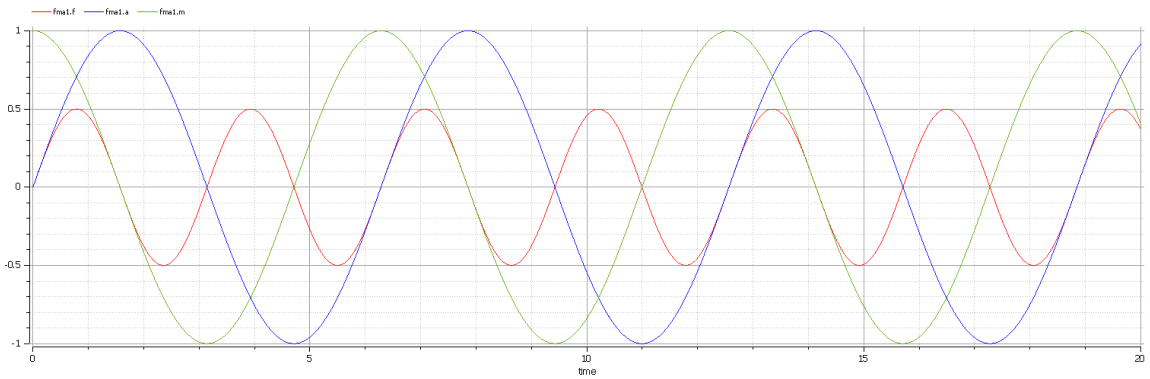
Cette figure montre un modèle simple appelé « Force = Masse * Accélération ».



- 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 »



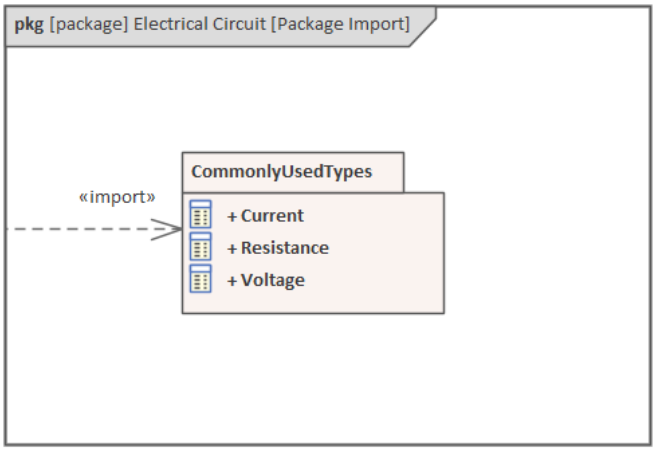
- Cochez les cases « Propriétés à tracer » pour « fma1.a », « fma1.m » et « fma1.f »
- Cliquez sur le bouton Résoudre pour simuler le modèle



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

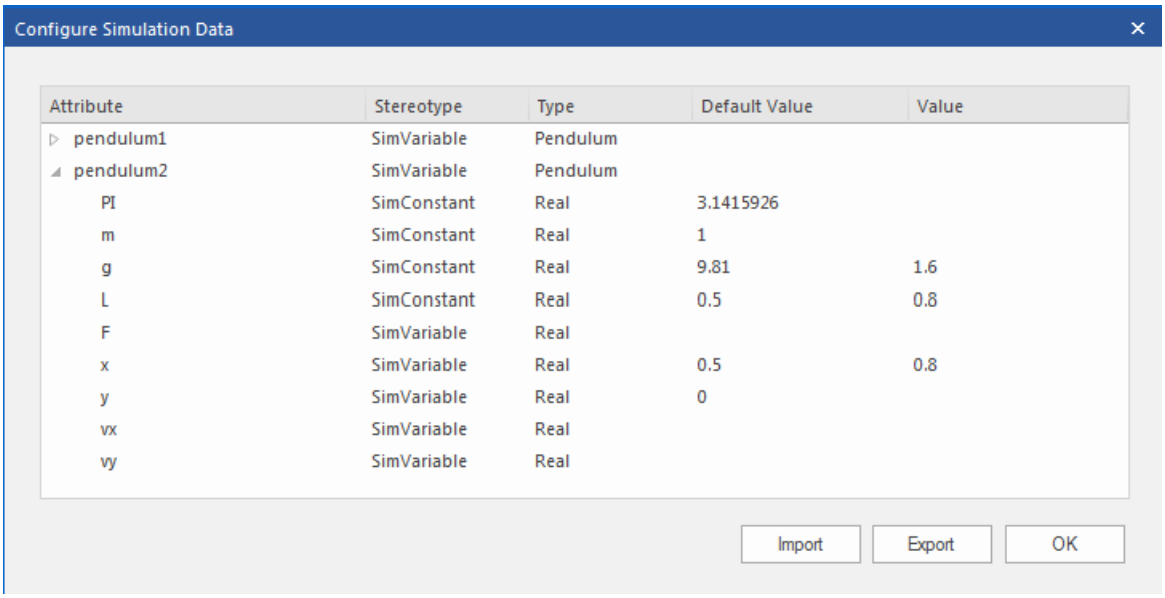
Ruban	Simuler > Comportement du Système > Modelica/Simulink > Gestionnaire de configuration SysMLSim > dans le groupe « bloc » > Colonne Nom > Menu contextuel sur l'élément bloc > Créer un jeu de données Simulation

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

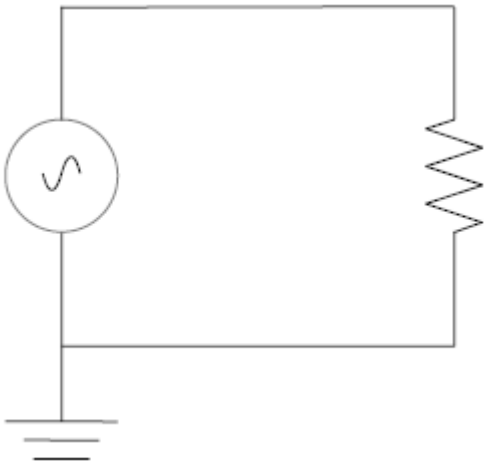
Modèle	Description
Exemple Simulation de circuit électrique	Le premier exemple concerne la simulation du chargement d'un circuit électrique. L'exemple part d'un diagramme de circuit électrique et le convertit en un modèle paramétrique. Le modèle est ensuite simulé et la tension aux bornes source et cible 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

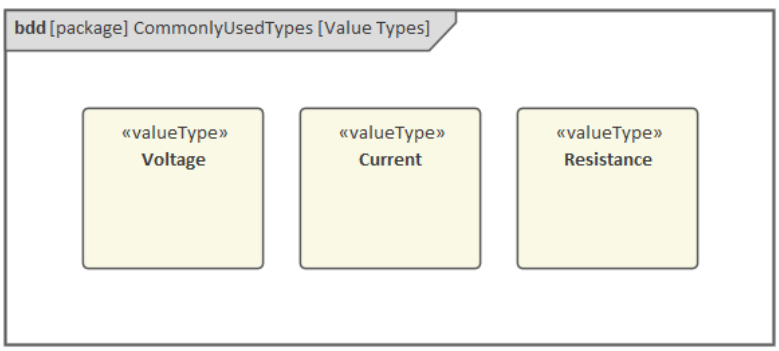
Le circuit électrique que nous allons modéliser, montré ici, utilise une notation de circuit électrique standard.

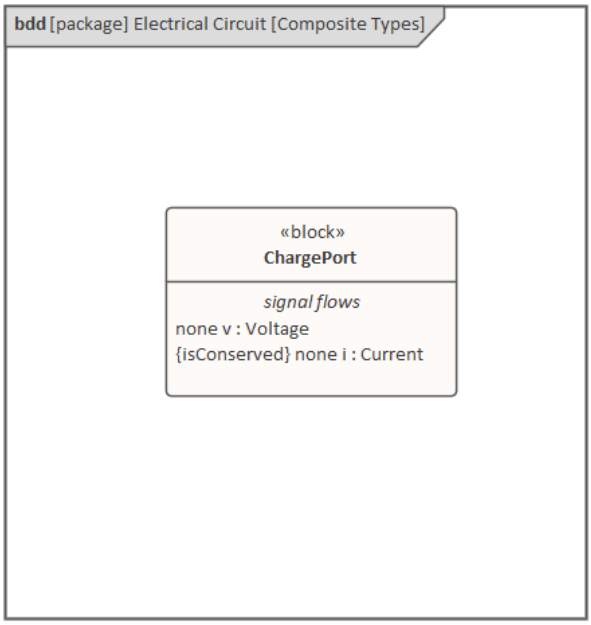


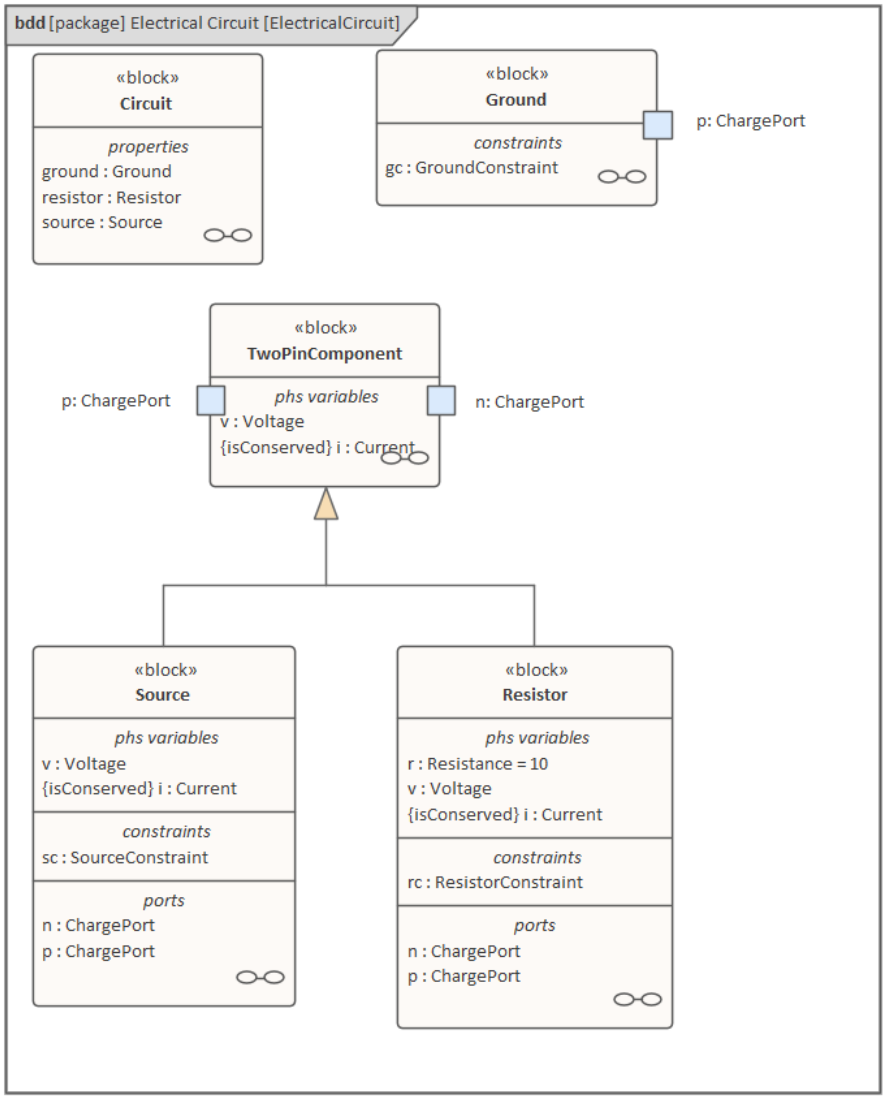
Le circuit comprend une source d'alimentation CA, une terre et une résistance, reliées entre elles par un fil électrique.

Créer Modèle SysML

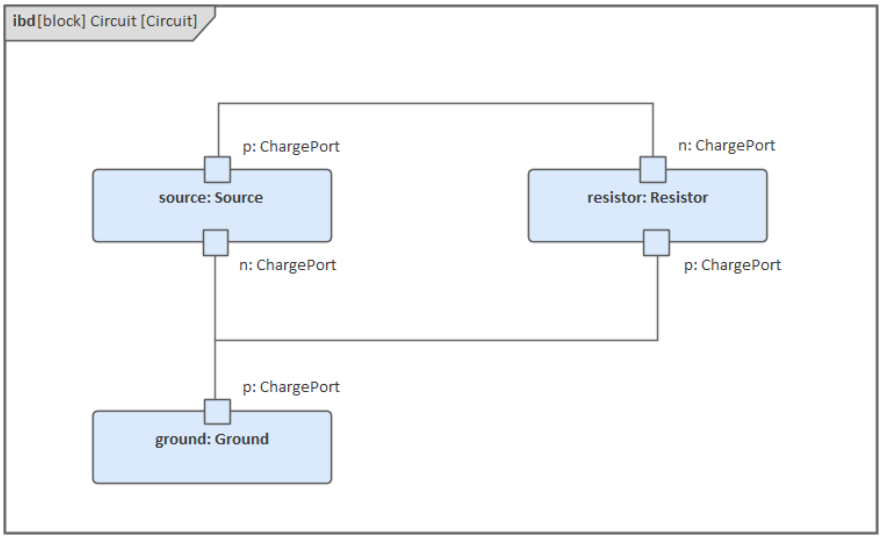
Ce tableau montre comment nous pouvons créer un modèle SysML complet pour représenter le circuit, en commençant par les types de niveau le plus bas et en construisant le modèle une étape à la fois.

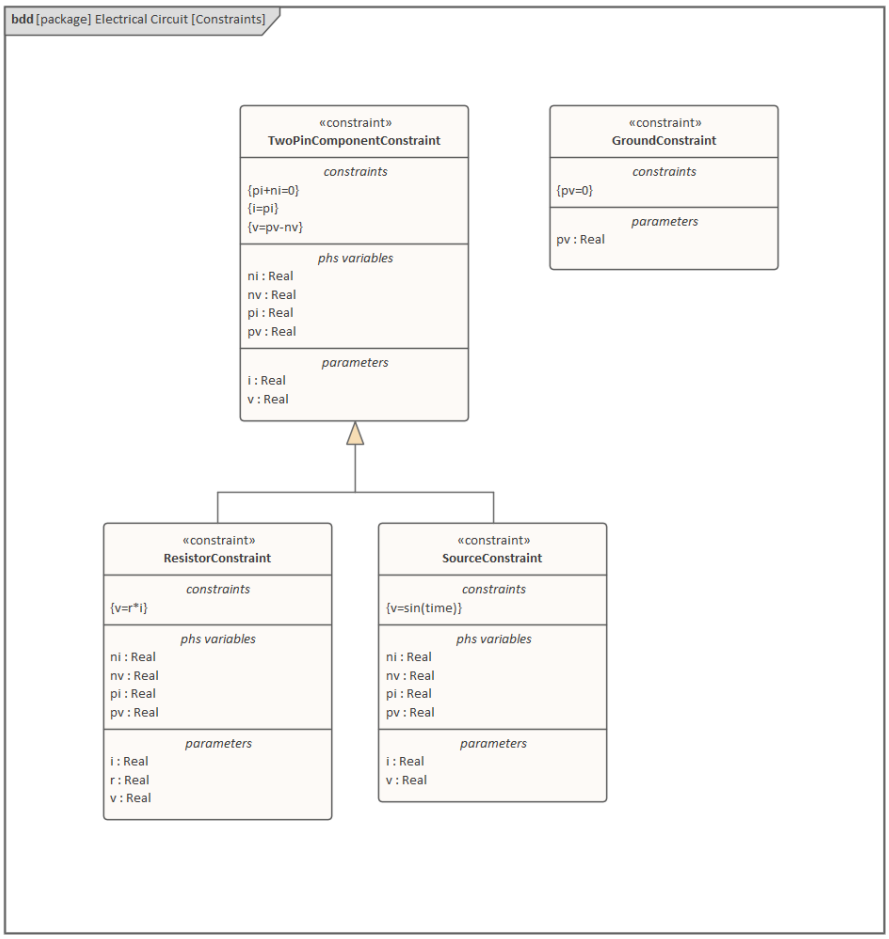
Composant	Action
Types	Définissez les types de valeur pour la tension, le courant et la résistance. Le type d'unité et de quantité n'est pas important pour les besoins de la simulation, mais il serait défini lors de la définition d'un modèle SysML complet. Ces types seront généralisés à partir du type 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.

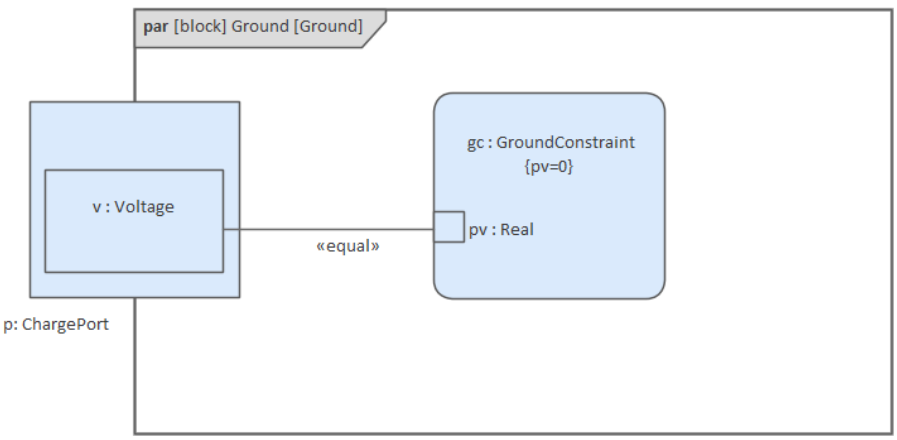
	 Notez que cela suit la même structure que le diagramme de circuit d'origine, mais les symboles de chaque composant ont été remplacés par des propriétés typées par les blocs que nous avons définis.
Contraintes	Les équations définissent des relations mathématiques entre des propriétés numériques. Dans 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

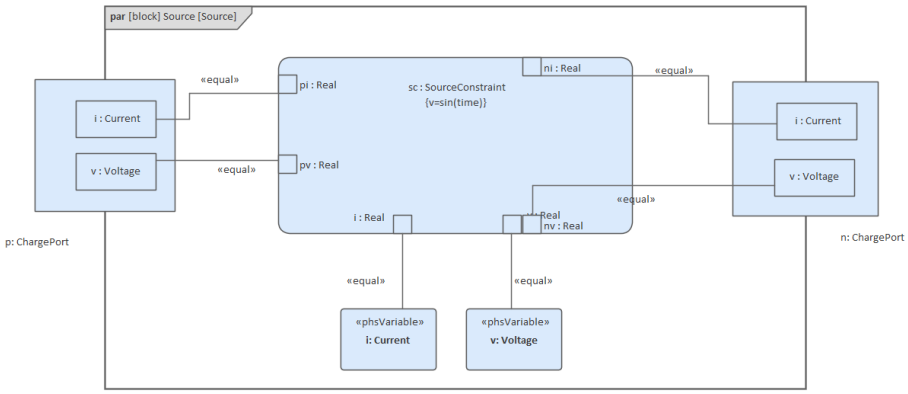
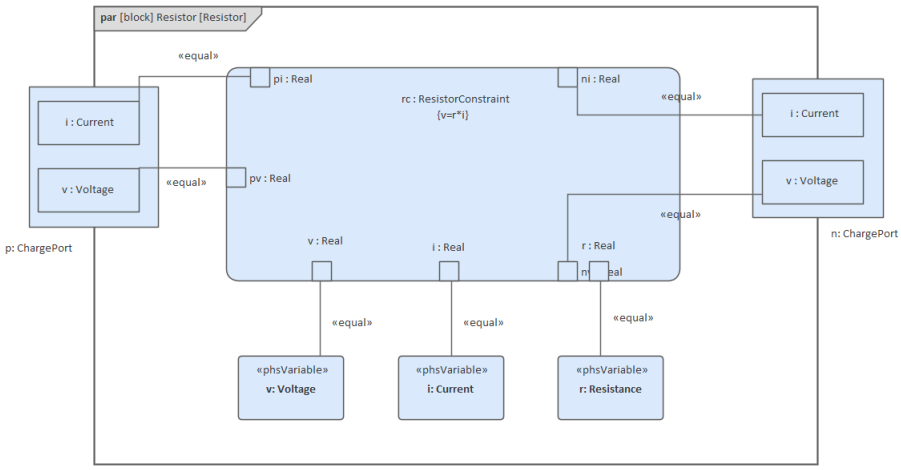
Les valeurs des paramètres de contrainte sont assimilées à des valeurs variables et constantes avec des connecteurs de liaison. Créez des propriétés de contrainte sur chaque Bloc (propriétés typées par ConstraintBlocks) et liez les variables et constantes Bloc aux paramètres de contrainte pour appliquer 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

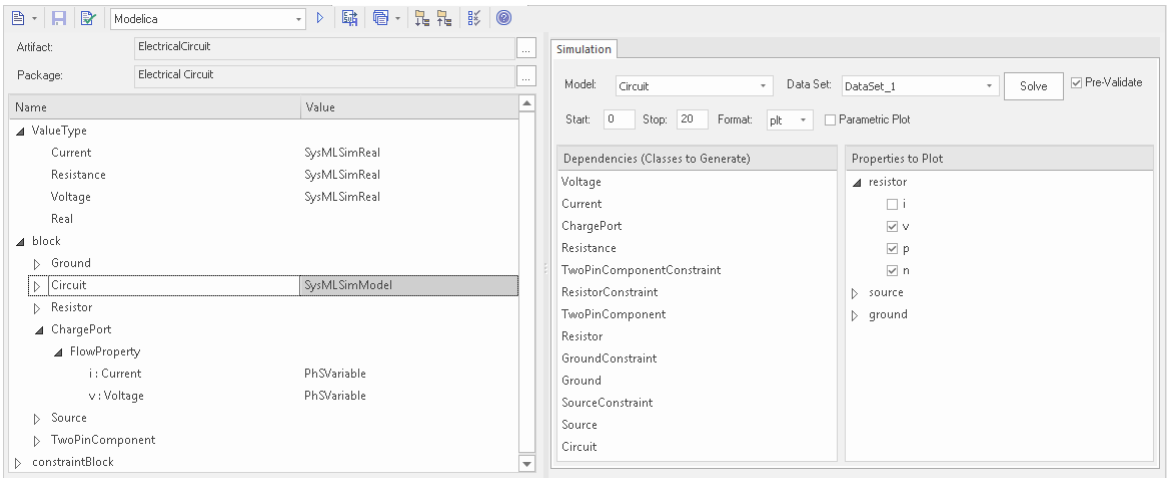
Ce tableau montre les étapes détaillées de la configuration de SysMLSim.

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

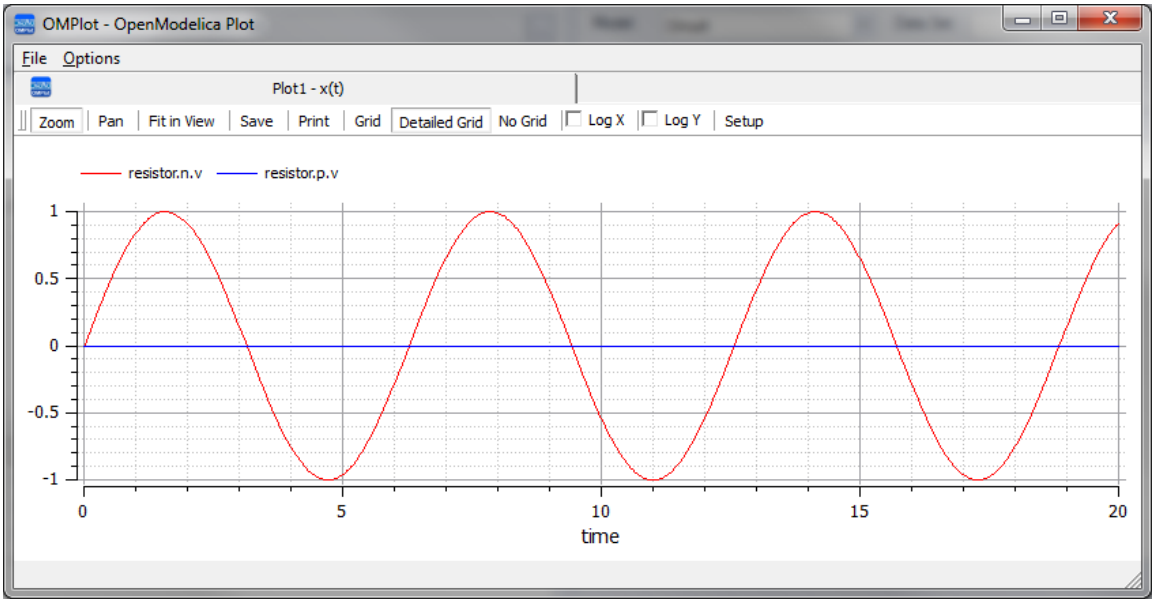
	• Dans la liste déroulante de la première icône de la barre d'outils, sélectionnez « Créer un artefact » et créez l'élément artefact • Sélectionnez le Paquetage qui possède ce modèle SysML
Créer des éléments racines dans Configuration Manager	• Type de valeur • Bloc • Bloc de contrainte
Substitution 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.

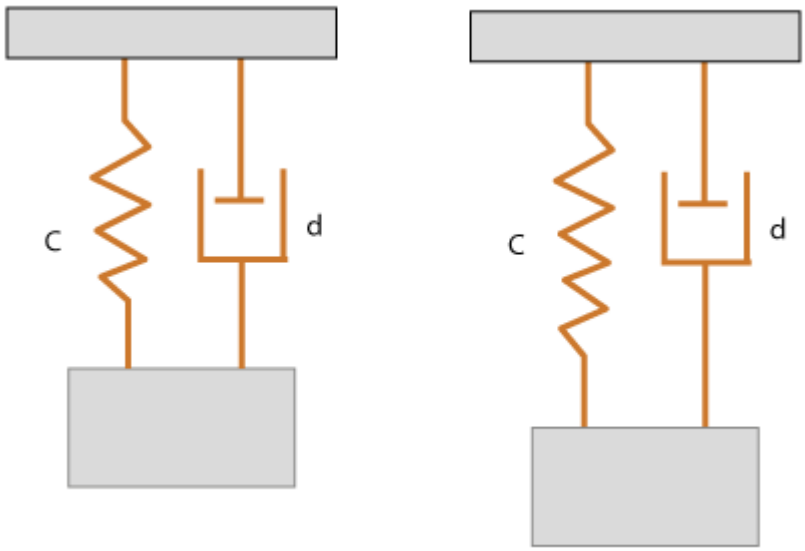

OMPlot - OpenModelica Plot
File Options
Plot1 - x(t)
Zoom Pan Fit in View Save Print Grid Detailed Grid No Grid Log X Log Y Setup
resistor.n.v
resistor.p.v
1
0.5
0
-0.5
-1
0
5
10
15
20
time

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

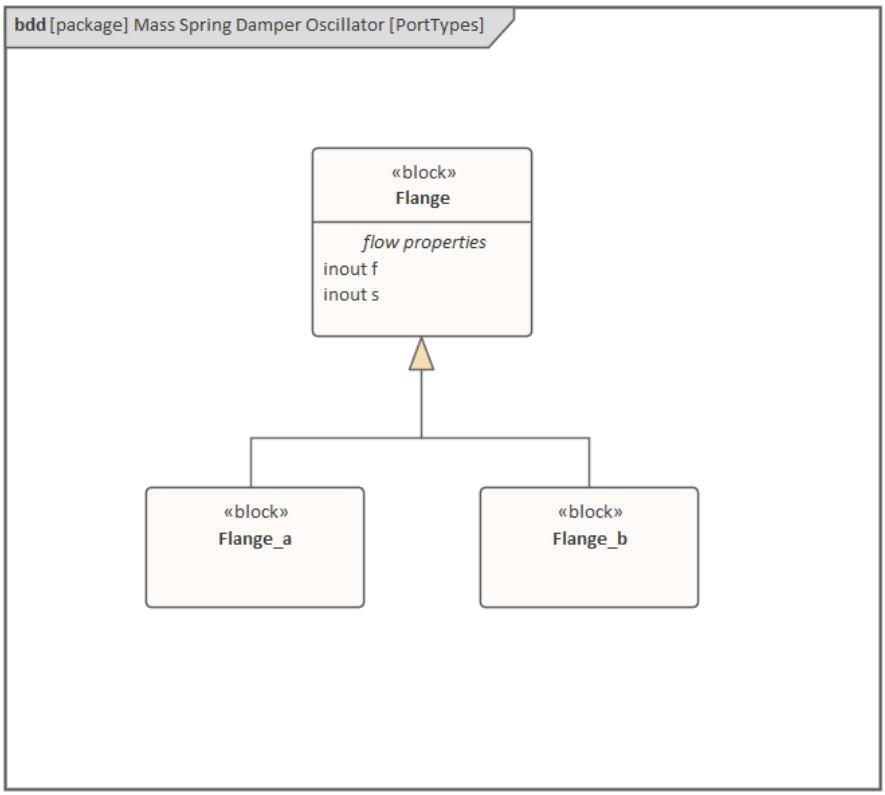
Une masse est suspendue à un ressort et à un amortisseur. Le premier état représenté ici représente le point initial à l'instant = 0, juste au moment où la masse est relâchée. Le deuxième état représente le point final lorsque le corps est au repos et que les forces du ressort sont en équilibre avec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

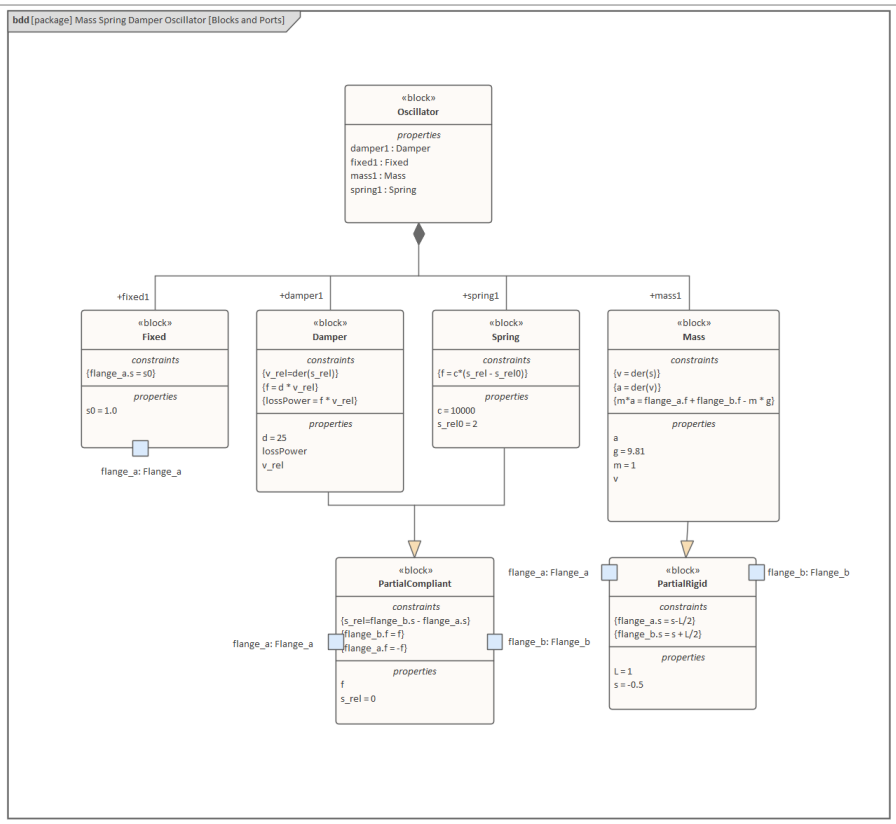


Créer Modèle SysML

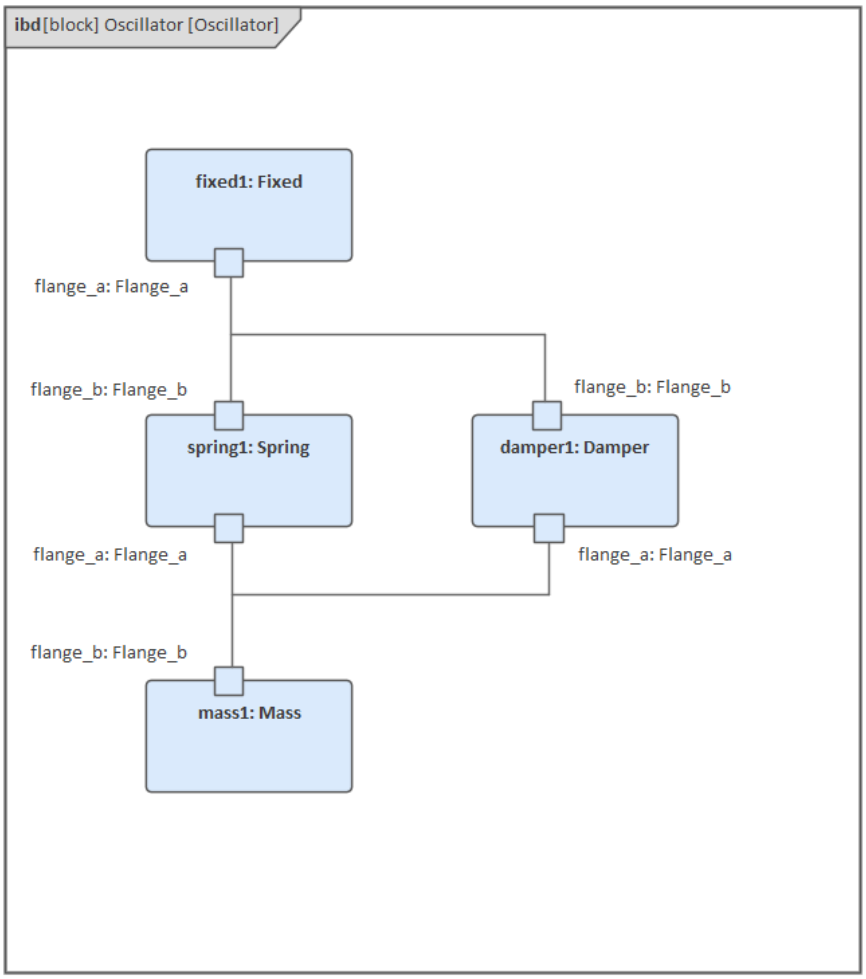
Le modèle MassSpringDamperOscillator dans SysML possède un Bloc principal, l' *Oscillateur* . L'Oscillateur est composé de quatre parties : un *plafond* fixe, un *ressort* , un *amortisseur* et un *corps de masse* . Créez un Bloc pour chacune de ces parties. Les quatre parties du Bloc Oscillateur sont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

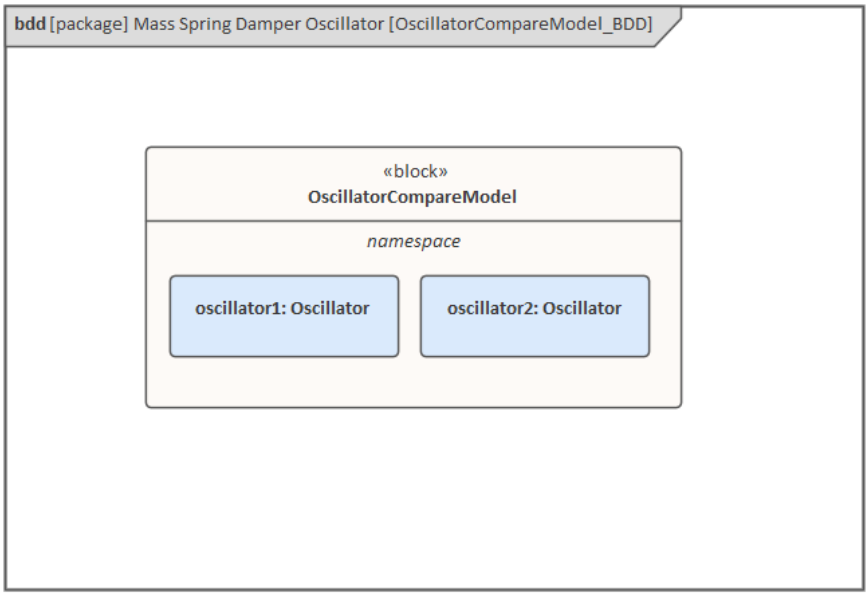
Comparaison de deux plans d'oscillateurs

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 :

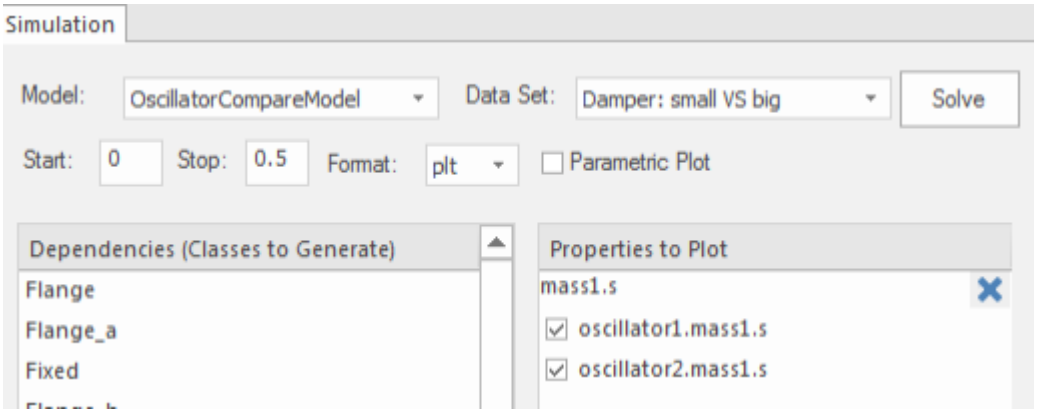
- Amortisseur : petit ou grand
  fournir 'oscillator1.damper1.d' avec la valeur 10 et 'oscillator2.damper1.d' avec la valeur la plus élevée 20
- Amortisseur : non ou oui
  fournir à 'oscillator1.damper1.d la valeur 0 ; ('oscillator2.damper1.d' utilisera la valeur par défaut 25)
- Printemps : petit vs 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

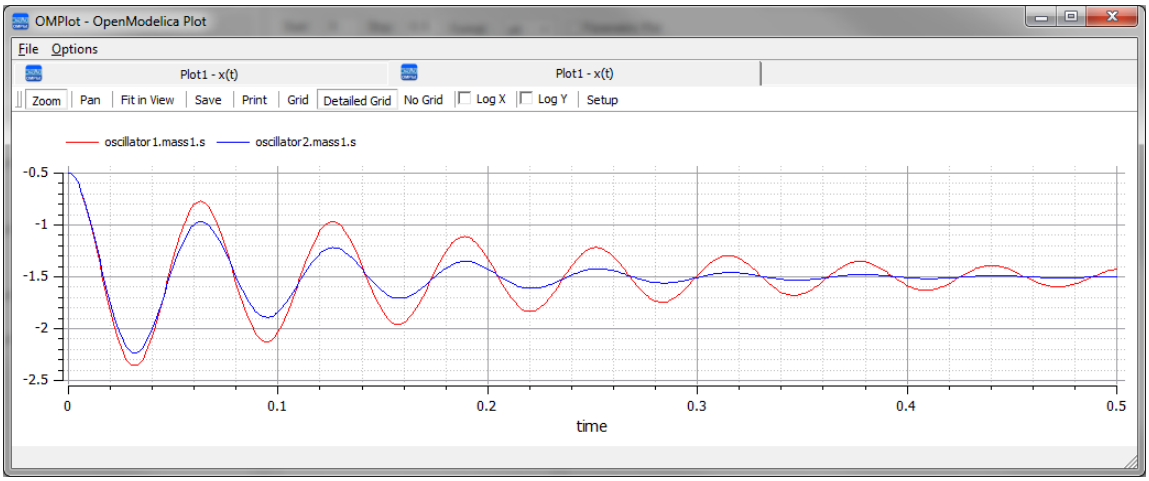
Sur la page « Simulation », sélectionnez « OscillatorCompareModel », tracez « oscillator1.mass1.s » et « oscillator2.mass1.s », puis choisissez l'un des ensembles de données créés et exécuter la simulation.

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.

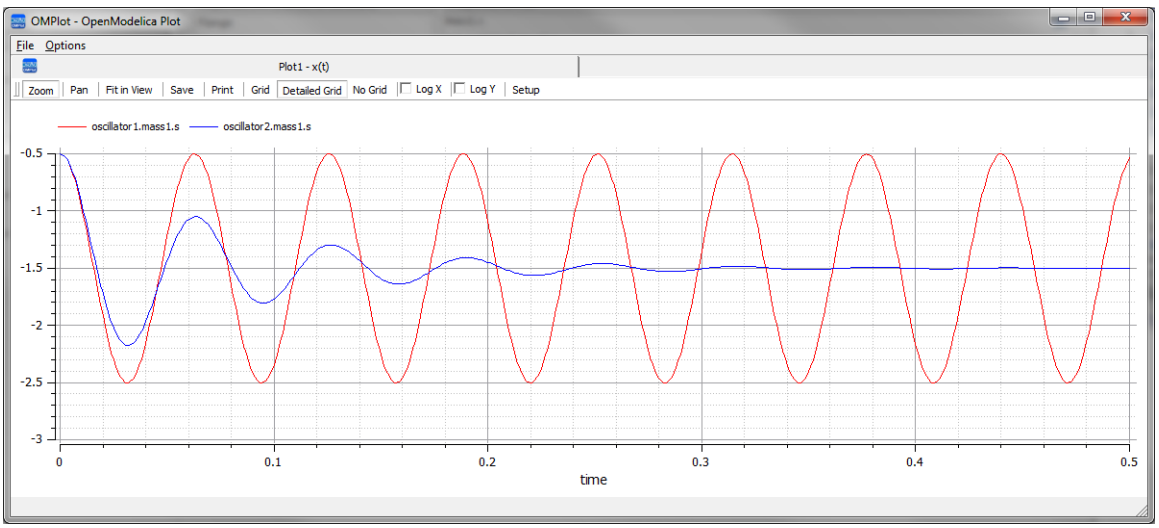


Voici les résultats de la simulation :

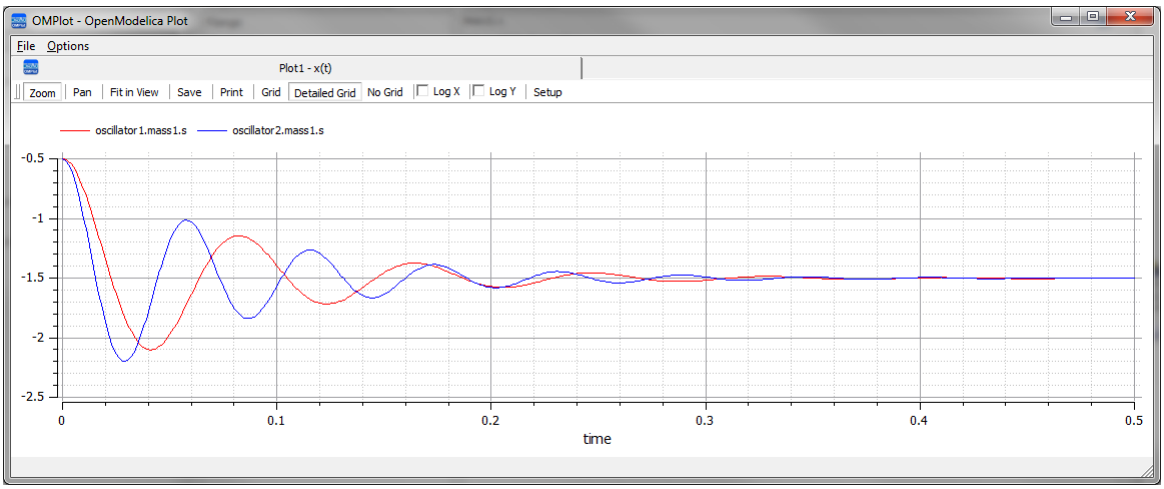
- Amortisseur, petit ou grand : le plus petit amortisseur permet au corps d'osciller davantage



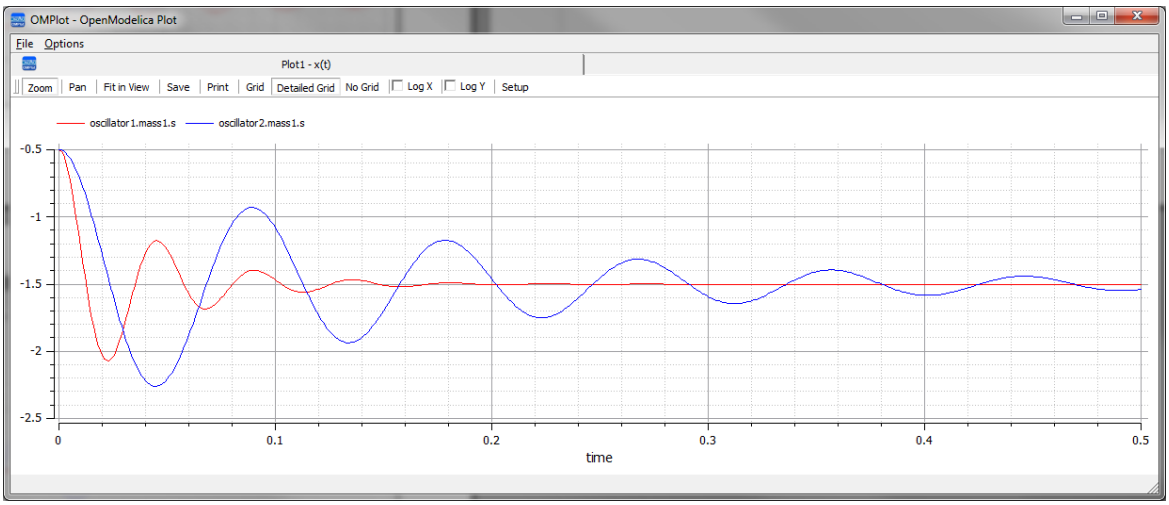
- Amortisseur, non vs oui : l'oscillateur ne s'arrête jamais sans amortisseur



- 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



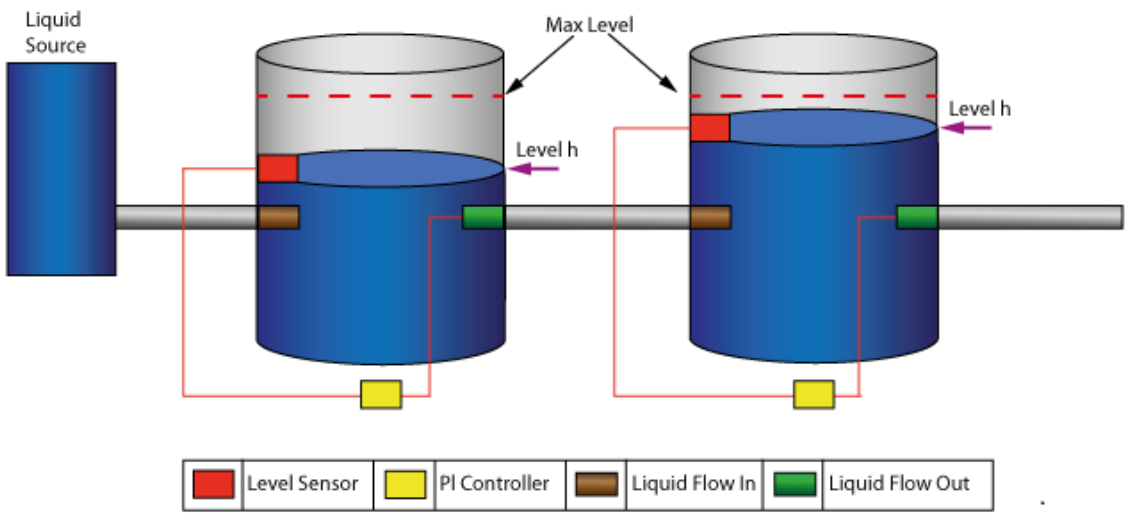
Régulateur de pression du réservoir d'eau

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.

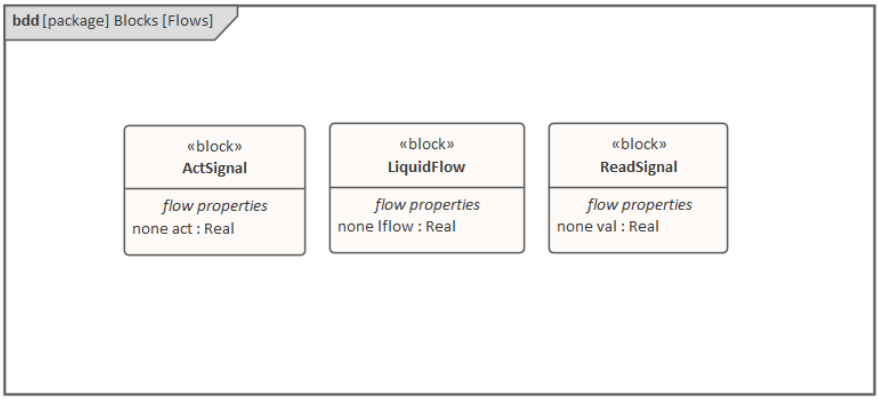
Système en cours de modélisation

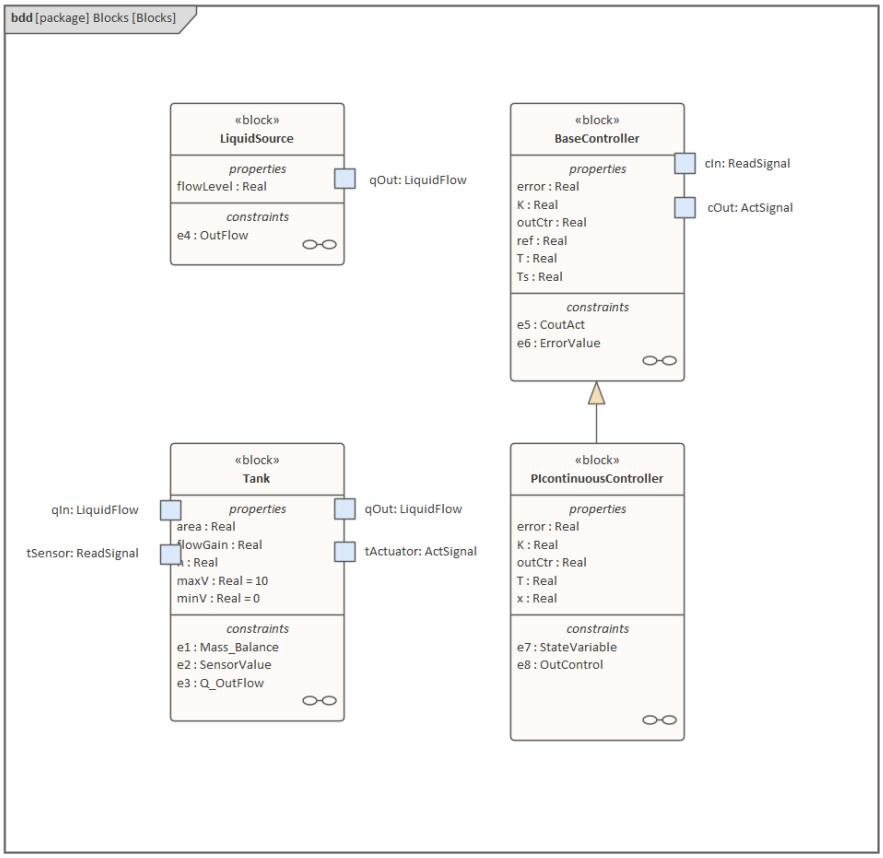
Ce diagramme représente deux réservoirs reliés entre eux et une source d'eau qui remplit le premier réservoir. Chaque réservoir est relié à un contrôleur continu proportionnel-intégral (PI), qui régule le niveau d'eau contenu dans les réservoirs à un niveau de référence. Pendant que la source remplit 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

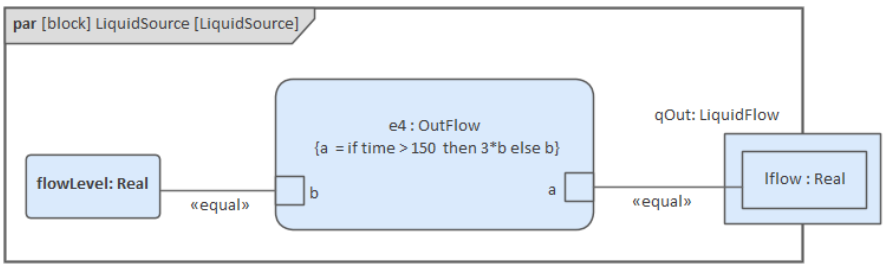
Composant	Discussion
Types de ports	Le char dispose de quatre ports qui sont typés sur ces trois blocs : • ReadSignal : lecture du niveau de liquide ; cette propriété a une unité « m » • ActSignal : Le signal envoyé à l'actionneur pour régler la position de la vanne • LiquidFlow : Le débit de liquide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

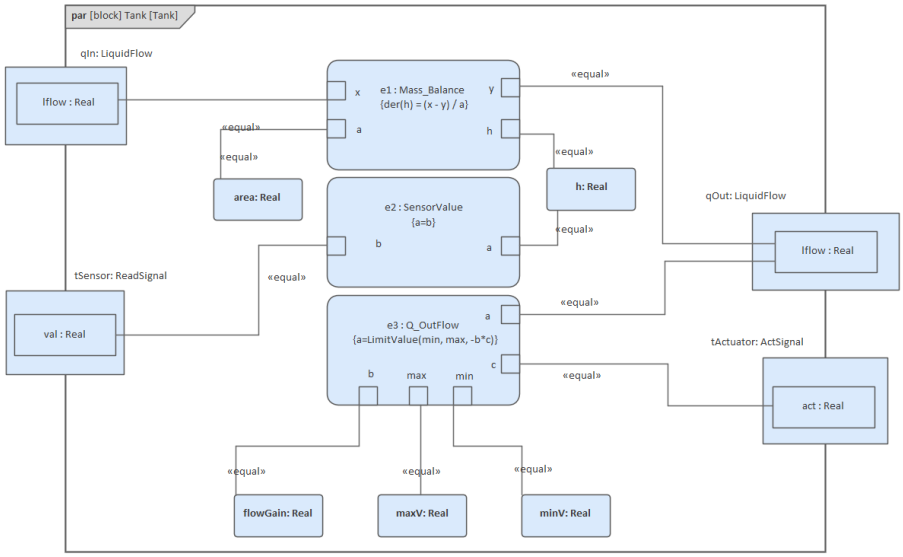
Le débit augmente brusquement à l'instant = 150 jusqu'à un facteur trois du niveau de débit précédent, ce qui crée un problème de contrôle intéressant que le contrôleur du réservoir doit gérer.



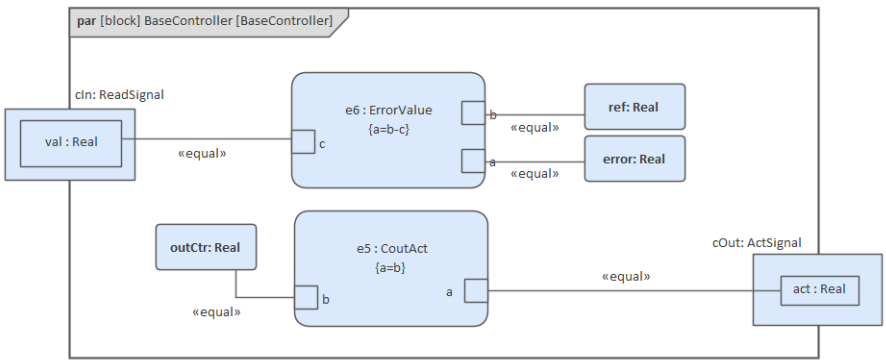
L'équation centrale régulant le comportement du réservoir est l'équation *de bilan massique* .

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.



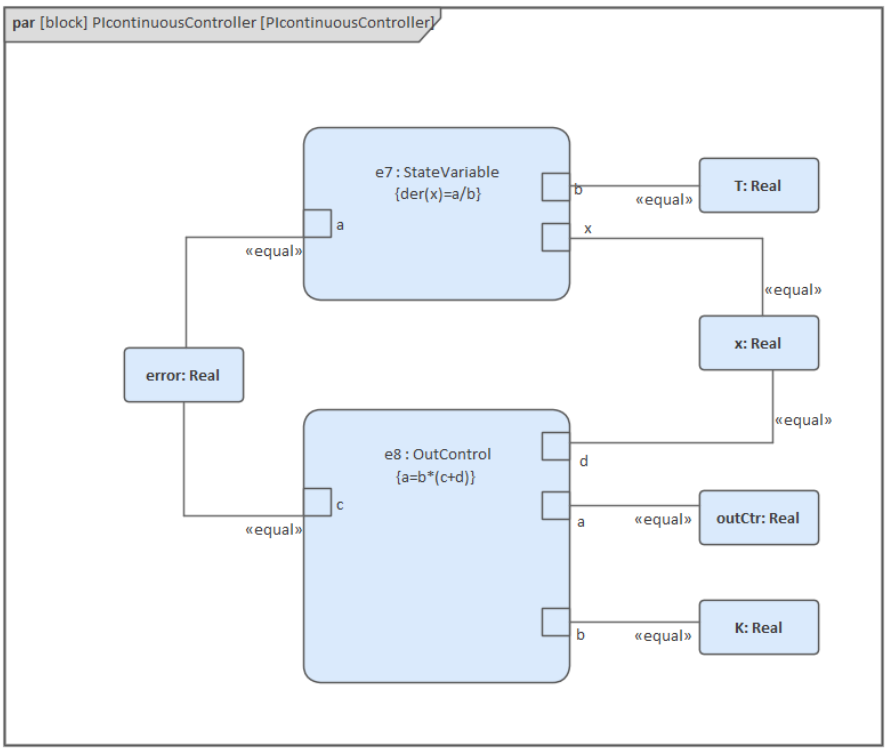
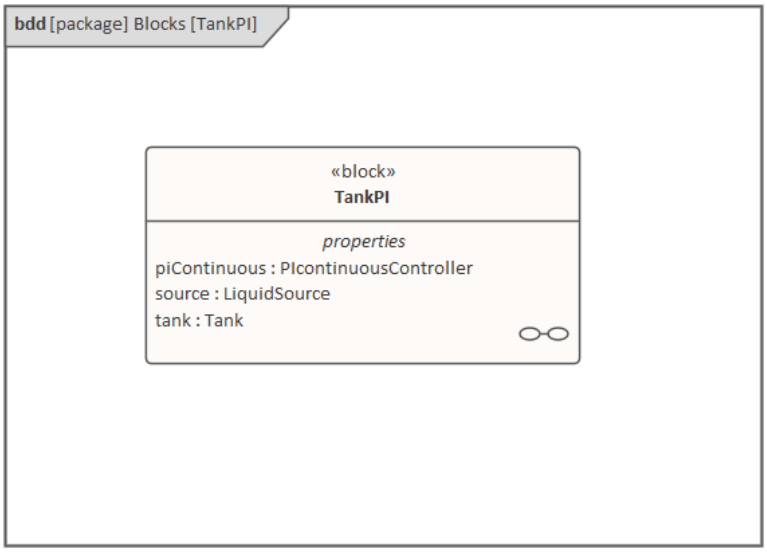
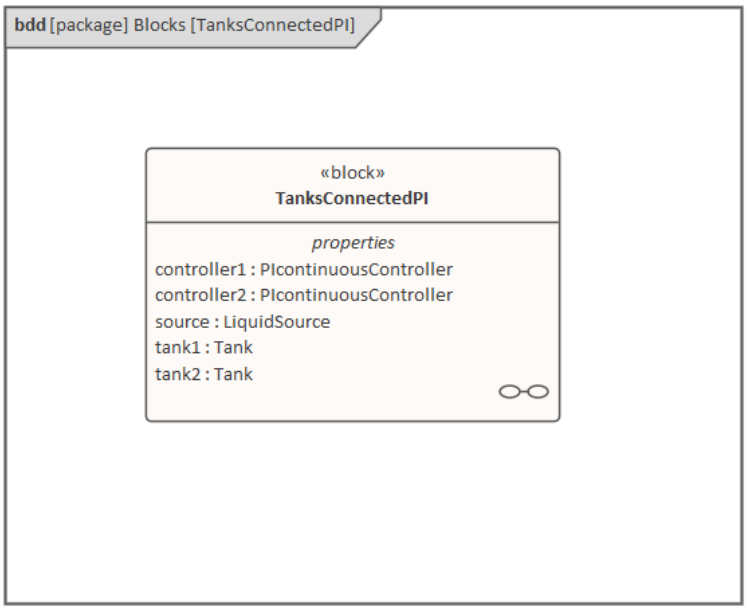


Diagramme Interne de Bloc	Il s'agit du diagramme Bloc interne d'un système avec un seul réservoir.  Il s'agit du diagramme Bloc interne d'un système avec deux réservoirs connectés. 

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 .

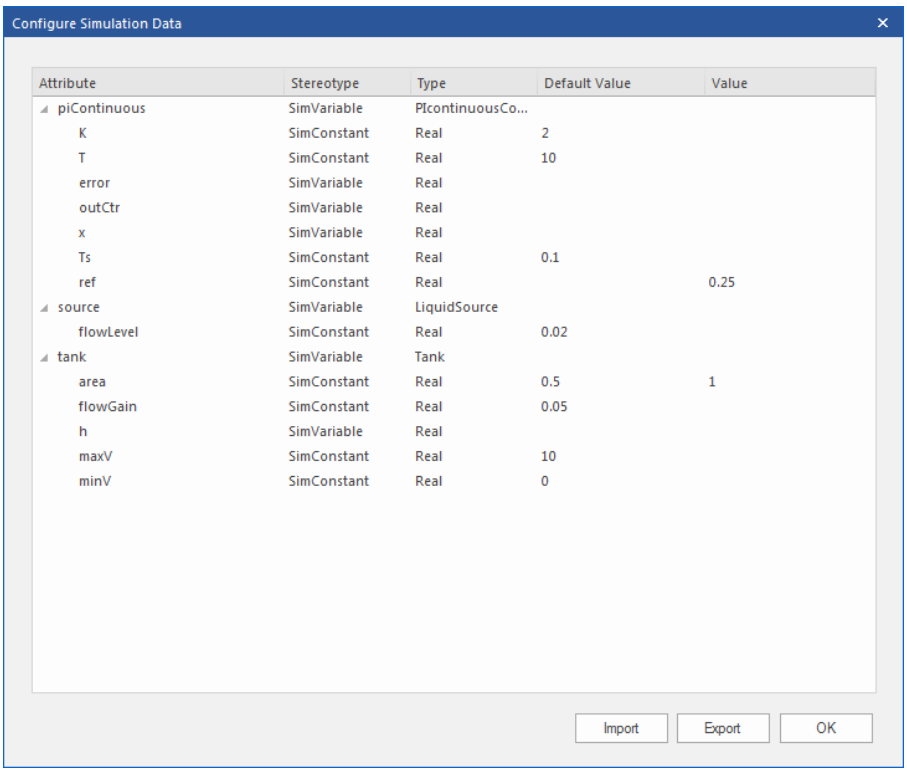
Note : Propriétés non configurées comme 'SimConstant' sont 'SimVariable' par défaut.

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 .

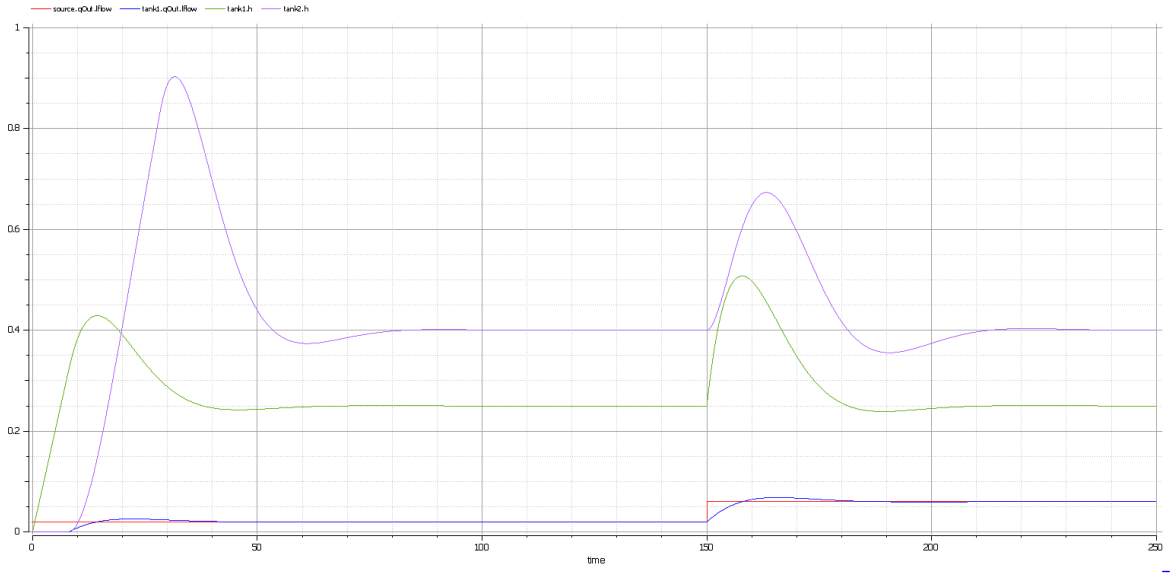
Élément	Ensemble de données
Source liquide	Niveau de débit : 0,02
Réservoir	h.début: 0

	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 :

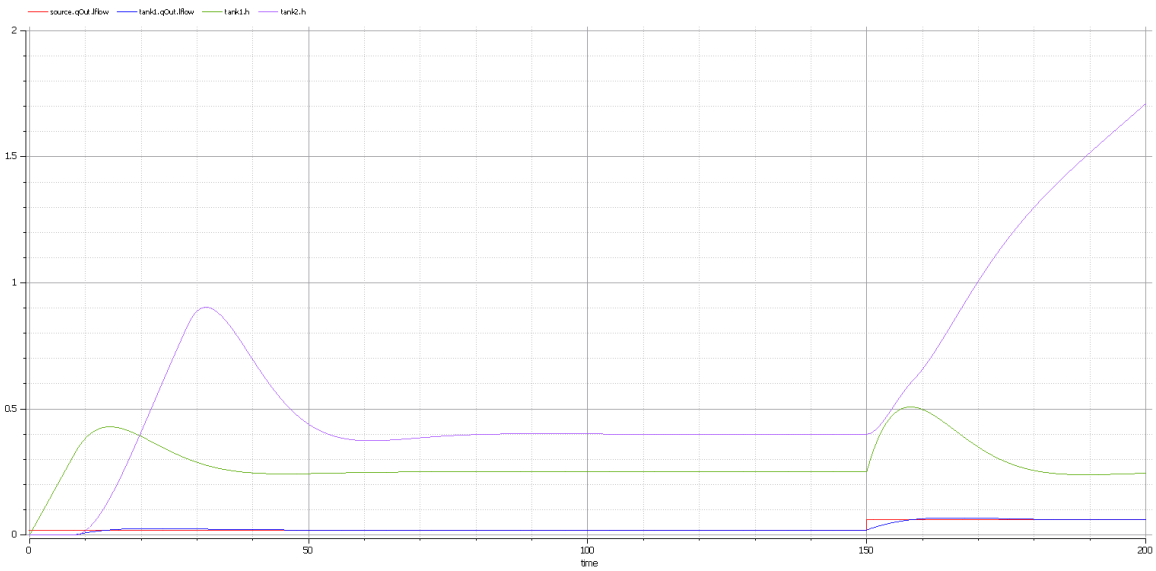
- Le débit du liquide augmente brusquement à l'instant = 150, jusqu'à 0,06 m ^{3} /s, soit un facteur trois du débit précédent (0,02 m ^{3} /s)
- Réservoir 1 régulé à une hauteur de 0,25 et réservoir 2 régulé à une hauteur de 0,4 comme prévu (nous avons défini la valeur du paramètre via l'ensemble de 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.