ENTERPRISE ARCHITECT

用户指南系列

仿真和行为

Author: Sparx Systems
Date: 13/11/2024
Version: 17.0

创建于 ENTERPRISE ARCHITECT

目录

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.

动态模拟

模型仿真通过即时、实时的行为模型执行使您的行为模型栩栩如生。再加上管理触发器、事件、守卫、效果、断点和仿真变量的工具，加上在运行时可视化跟踪执行的能力，模拟器是一种 观察车轮转动"和验证正确性的多功能手段。你的行为模型。借助仿真，您可以探索和测试模型的动态行为。在企业版、统一版和终极版中，您还可以使用JavaScript作为运行时执行语言来评估守卫、效果和其他可编写脚本的行为项。

对触发器、触发器集、嵌套状态、并发性、动态效果和其他高级仿真功能的广泛支持，为构建交互式和工作模型提供了一个卓越的环境，有助于探索、测试和可视化跟踪复杂的业务、软件和系统行为。启用JavaScript后，还可以创建嵌入式 COM 对象，这些对象将执行评估防护和执行效果的工作——允许将仿真绑定到更大的依赖进程集。例如，COM object评估状态转移

上的保护条件转移

可能会查询本地运行的进程，读取和使用一组测试数据，甚至连接到 SOA Web 服务以获取一些当前信息。

由于Enterprise Architect使用动态的、脚本驱动的仿真机制，可以直接分析和使用UML结构，因此在运行仿真之前无需生成中间代码或编译仿真 可执行文件"。这导致了一个非常快速和动态的仿真环境，可以在其中快速进行更改和测试。甚至可以使用仿真控制台窗口实时更新仿真变量。这对于 动态"测试替代分支和条件很有用，无论是在设置的仿真断点处还是在仿真器达到稳定点时（例如，当仿真器被 阻塞"时）。

在Enterprise Architect的专业版中，您可以手动浏览模拟 - 尽管没有JavaScript将执行 - 所以所有选择都是手动决定的。这对于测试行为模型的流程和突出可能的选择和处理路径很有用。在企业版、统一版和终极版中，您可以：

- 动态执行您的行为模型
- 评估用标准JavaScript编写的守卫和效果
- 定义并触发触发器以运行模拟
- 定义和使用触发器集来模拟不同的事件序列
- 自动触发触发器集以模拟复杂的事件历史，无需用户干预
- 即时"更新仿真变量以改变仿真的进行方
- 在仿真期间创建和调用 COM 对象以扩展仿真的范围和输入/输出的可能性
- 运行时检查仿真运行
- 设置脚本 序言"以在执行前定义变量、常量和函数
- 使用具有不同 序言"的脚本仿真各种条件下运行分析器

在统一版和终极版中，还可以模拟 BPMN 模型。

使用模型模拟器，您可以模拟包含行为的概念模型设计的执行。启动仿真时，会对当前模型包进行分析，并触发动态仿真过程来执行模型。

要启动并运行仿真，只需执行以下步骤：

- 编译行为图（状态或活动用于手动或动态执行，序列用于手动交互）
- 可选：加载 仿真仿真工作空间"布局——快速调出所有常用仿真窗口
- 点击模拟器播放按钮

如果图表包含任何外部元素（与图表不在同一个包中的那些），您将必须创建一个从图表包到包含外部元素的包的导入连接器。为此，将两个包从浏览器窗口拖到图表上，然后使用快速链接器箭头在它们之间创建连接器。

仿真概览

方面
模型模拟器概述

使用窗口及相关窗口的仿真及运行仿真
设置仿真并激活仿真脚本
设置和使用仿真断点
仿真物体的使用
不同类型行动在仿真中的使用
用JavaScript进行动态仿真
使用中防护条件的使用
使用中的使用触发器
调用行为和变量
仿真活动返回
仿真结构活动行为
仿真多线程进程
在单独的图表中仿真子流程
执行 BPMN 模拟
仿真Win32对话框行为

平台和可用版本

平台/版	细节
支持的型号和平台	模型模拟器目前支持在仿真平台上执行UML活动交互、状态机模型和业务模型： • UML基础 • BPMN
版支持	模型仿真在Enterprise Architect的各个版本范围内提供不同级别： • 专业-仅限手动仿真 • 企业及以上 - 添加动态JavaScript评估；目前为状态机和活动图启用了JavaScript ；没有为交互启用 • 统一终极-新增BPMN仿真

它看起来如何

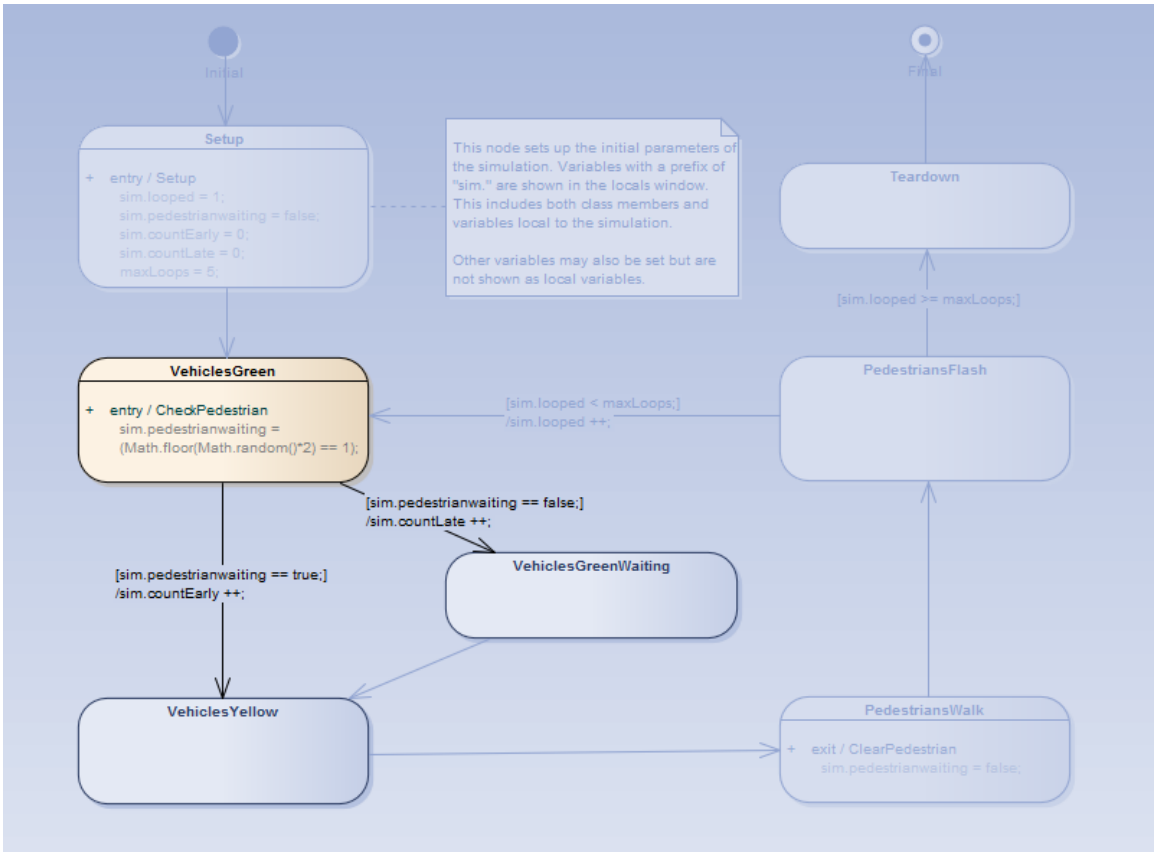
Enterprise Architect在仿真过程中有一种特殊的模型信息展示方式。这有助于聚焦集中在执行或活动节点上。

在仿真过程中， Enterprise Architect将动态跟踪并突出显示模型中的活动节点。如果另一个图表中的节点被激活，该图表将被自动加载并突出显示当前节点。仿真运行时可以修改图表；但是，在当前的仿真结束并开始新的仿真之前，所做的更改不会被识别。

仿真过程中活动节点的突出显示

在此示例中，当前活动节点 (VehiclesGreen) 以正常Enterprise Architect颜色突出显示，并且当前节点的所有可能转换都以全强度呈现。

作为当前活动节点的传出转换的可能目标的元素以半淡化样式呈现，以便它们易于阅读并且与图中的其他元素明显不同。所有其他元素都以完全褪色的样式呈现，以表明它们不是下一步仿真步骤的目标。随着仿真的进行（特别是如果自动运行），这种突出显示有助于将聚焦集中在当前项目及其视觉上下文。



仿真窗口

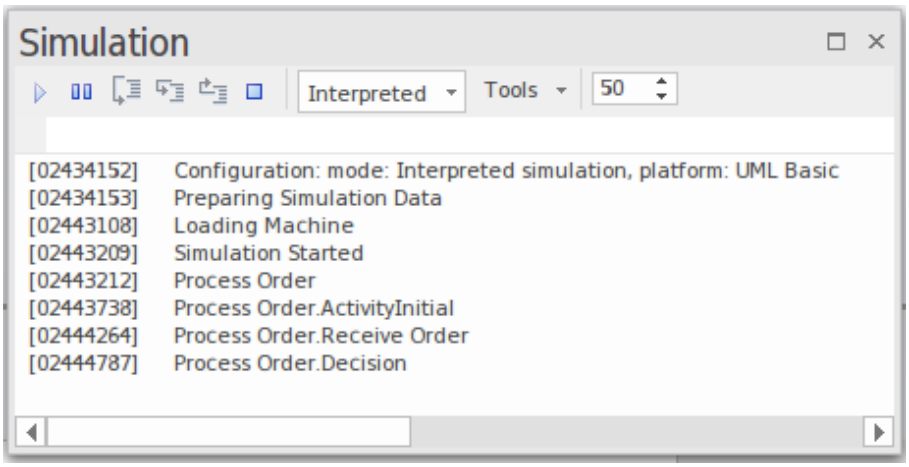
在Enterprise Architect中执行仿真时，可以设置断点、触发执行速度、检查变量执行速度、查看调用堆栈仿真可视化仿真活动节点。

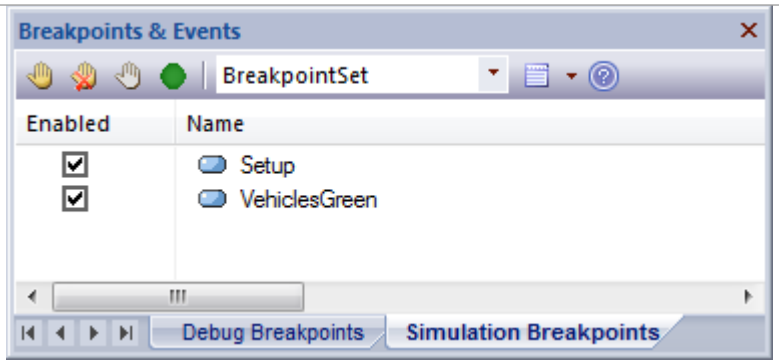
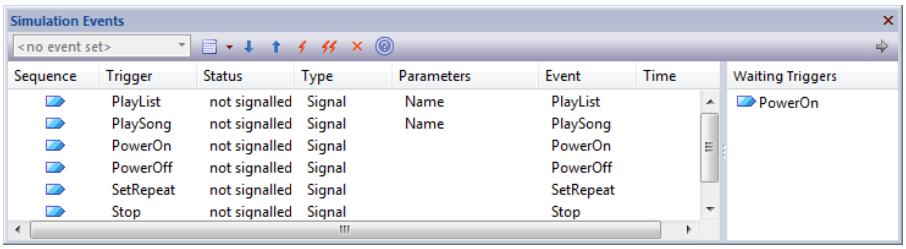
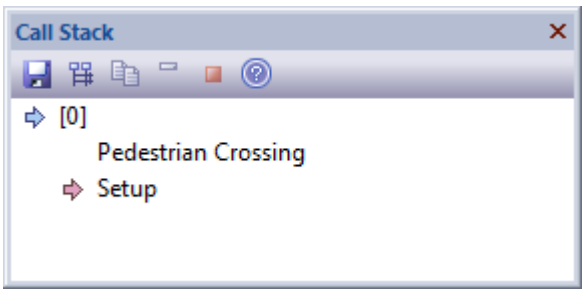
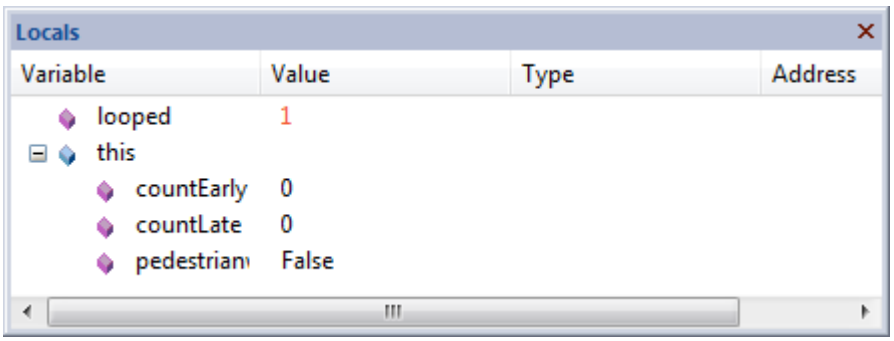
仿真运行时，输出和控制台输入等一些方面是在窗口本身中仿真的，而局部变量和调用堆栈使用标准执行分析器窗口。本主题概述了仿真期间使用的主要窗口。

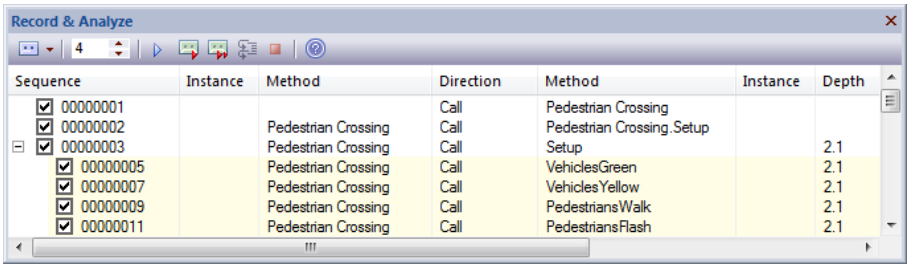
访问

功能区	仿真>动态仿真仿真> 模拟器> 打开仿真窗口

窗口

窗户	目的
执行和控制台	仿真窗口提供启动、停止和步进仿真的主界面。在执行期间，它显示与当前执行步骤和其他重要信息相关的输出。有关工具栏命令的更多信息，请参阅运行模型仿真主题。 注记工具栏正下方的文本输入框。这是控制台输入区域 - 在这里您可以输入简单的JavaScript命令，例如： *this.count = 4;*将名为 "count"的模拟变量动态更改为 4。通过这种方式，您可以在运行时动态影响模拟。 
断点&事件窗口	仿真过程还使用了断点和标记窗口的 仿真断点"选项卡（ 仿真>动态仿真仿真>断点" ）。在这里，您可以在仿真中的特定元素和消息上设置执行断点。详见仿真断点专题。

	
仿真事件窗口	仿真事件窗口（ 仿真>动态仿真>事件”）提供了管理和执行触发器的工具。触发器状态机用于控制过渡的执行。 
调用堆栈窗口	调用堆栈仿真（'仿真>动态仿真>仿真调用堆栈'）显示当前线程和执行上下文的信息。 模拟器支持多线程模拟，并且将为每个活动和暂停的执行线程包括一个线程条目。对于每一个线程，调用堆栈窗口会显示该线程的开始或元素上下文（如状态机元素）加上当前活动的元素。如果当前活动元素是复合活动或状态的入口点，则堆栈还将包括该子上下文中的当前活动元素（以及所有进一步嵌套的活动组合子状态）。 
仿真局部变量窗口	模拟器使用标准的本地窗口窗口（ 仿真>动态仿真仿真>局部变量”）在模拟单步执行或在断点处暂停时显示所有当前模拟变量。注记可以使用动态更新这些变量模拟器控制台。 
记录	在执行模拟期间，会保留所有活动的记录并显示在 记录和分析”窗口中（ 执行 > 工具 > 记录器 > 打开记录器”）。这与可视化执行分析器中正常通话录音

	的工作方式类似。 

设置仿真脚本

您可以使用仿真脚本对仿真的启动方式进行精细控制。在一般情况下，您不需要设置仿真脚本，除非：

- 您想运行需要在仿真开始前初始化变量的解释性仿真；这对于设置全局变量和定义函数很有用
- （在企业版及以上版本中）您不想应用解释 Guards 的默认行为（即，您更喜欢使用手动执行），或者
- 您希望有多种方式运行同一个图表

对于大多数图表，可以简单地通过在开始元素之后的第一个元素或连接器中设置变量来初始化仿真脚本。对于状态图，这是退出初始元素的 Transit 连接器，对于活动模型，这是第一个行动元素。

作为替代方案，您可以使用仿真脚本在仿真开始之前初始化设置。这对于使用多个分析器脚本设置不同的初始值集非常有用，以便您可以在一系列预设条件下运行您的仿真。

要配置仿真脚本包，首先在浏览器窗口、浏览器、图表列表或模型搜索中选择包。然后，您可以使用执行分析器窗口为选定的包添加新脚本。您将使用 执行分析器"对话框的 仿真"页面来配置相关属性。

访问

使用此处概述的方法之一显示执行分析器窗口。

在执行分析器窗口中：

- 找到并双击所需的脚本，然后选择 仿真"页面或
- 点击窗口工具栏中的 并选择 仿真"页面

功能区	开发>源代码>执行分析器>编辑分析器脚本 执行 > 工具 >分析器
上下文菜单	浏览器窗口 |右键单击包|执行分析器
键盘快捷键	Shift+F12

配置仿真脚本

选项	行动
平台	对于UML活动，交互状态机模拟，单击下拉箭头并选择' UML Basic'。 对于 BPMN 图，单击下拉箭头并选择 BPMN"。
入口	单击 按钮并选择： • 仿真的入口点，和 • 活动，交互状态机来模拟 如果不指定入口点，模拟器会尝试遍历整个包。
使用JavaScript评估防护条件	（在企业及更高版本中）不选中复选框以执行手动仿真，您可以在其中选择下一个要转换到的状态以及必须做出决定的点。 选中复选框以执行仿真中影响行为的代码。仿真在这些地方执行JavaScript代码：

	• 状态进入/退出/做操作 • 转移 守卫/效果 • BPMN活动循环条件和序列流条件表达式 除了守卫之外，所有这些都应该是一个或多个有效的JavaScript语句，包括分号。 守卫必须是有效的布尔表达式，也以分号结尾。 当到达仿真断点时，属于 'sim' 或 'this' 成员的变量会在本地窗口窗口中列出。 sim.count = 0;
输入	启用JavaScript后，您可以在此字段中键入脚本命令，这些命令将在仿真运行之前运行。
后处理脚本	使用 Post仿真脚本，您可以在仿真结束后运行JavaScript 。类型在脚本的限定名称中来自模型脚本控件。 例如，如果脚本组 "MyGroup"中有一个名为 "MyScript"的脚本，请输入值 "脚本"。
确定	单击此按钮以保存您的更改。

注记

- 通常所有的仿真元素和关系都在为仿真配置的包内；但是，您可以通过从配置包创建包导入连接器到每个"外部"包来模拟包含来自不同包的元素的图表（或者，对于 BPSim模型，创建从配置包到每个外部元素的依赖连接器）

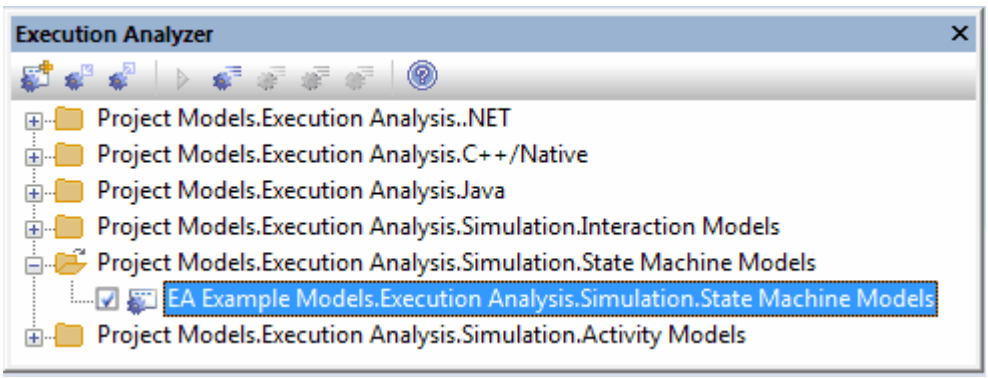
激活仿真脚本

定义仿真参数的模型包配置执行脚本。激活执行脚本的最常见原因是针对一个包配置了多个仿真脚本，并且您想要运行特定的一个。

访问

功能区	执行 > 工具 >分析器 开发>源代码>执行分析器
分析器窗口	点击分析器脚本使其激活
键盘快捷键	Shift+F12

激活仿真脚本执行

节	行动
1	在执行分析器窗口中，选择需要的执行脚本。这使它成为您打开模型的当前默认设置，以便单击仿真运行脚本仿真 
2	单击脚本左侧的复选框以激活它。
3	选择 仿真> 模拟器 > 打开仿真窗口”功能区选项来执行仿真。

运行模型仿真

A仿真逐步执行模型，使您能够验证行为模型的逻辑。当前执行步骤在模型图中自动突出显示，以便于了解仿真过程中发生的各种过程和状态变化。

启动模型仿真有几种方式：

- 当可以模拟活动图表时，主仿真窗口上的运行按钮将通过运行现有脚本或定义新的临时脚本来处理当前图表
- 当活动图无法模拟时，主仿真窗口上的运行按钮将运行活动图的仿真执行分析器脚本
- 通过右键单击执行分析器窗口中的仿真脚本并选择 开始仿真”选项
- 通过右键单击合适的图表并选择 执行仿真”选项之一

执行过程中有视觉提示。仿真运行时， Enterprise Architect会主动突出显示每个执行步骤的每个活动节点。此外，所有传出的转换和控制流将被突出显示，显示可能的前进路径。在可能的前进路径末端的元素将被淡化为一半强度，任何其他剩余元素将 90% 变灰”。这提供了一个非常动态且易 遵循的执行，不断将注意力重新集中在执行上下文上。

访问

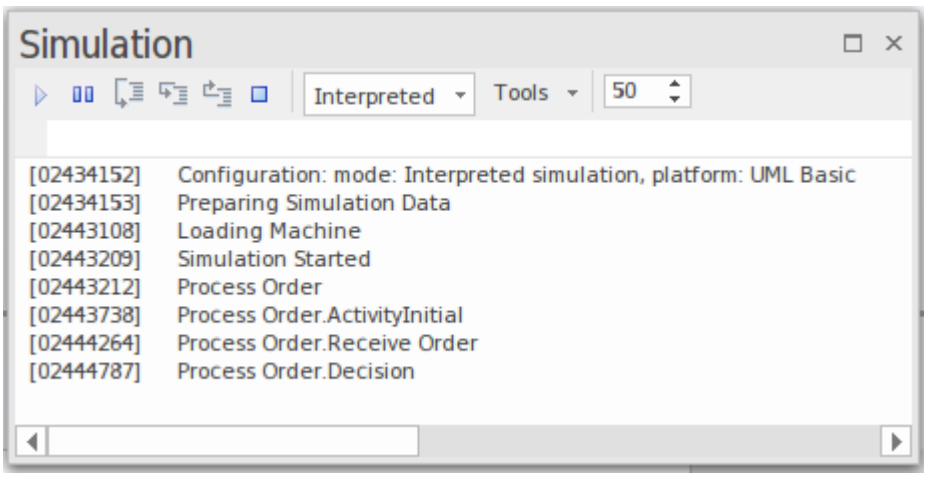
功能区	仿真>动态仿真仿真> 模拟器> 打开仿真窗口 仿真>运行仿真>开始

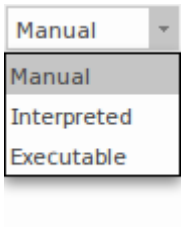
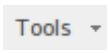
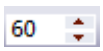
版具体细节

在专业版中，如果在执行中遇到分支，模拟器会提示您选择合适的路径来执行。

在启用了JavaScript的企业版、统一版和终极版中，仿真仿真会自动评估所有防护和效果，并在无需用户干预的情况下动态执行。如果仿真由于没有可能的向前评估为True的路径（或多个路径评估为True ）而阻塞，您可以使用仿真执行窗口的控制台输入即时修改仿真变量。

运行仿真使用工具栏



图标	行动
	为当前图表启动开始，或者，如果当前图表无法模拟，则使用运行的仿真脚本运行仿真。
	暂停仿真。
	当仿真器暂停时，通过跨步、跨步、跨步来控制模拟器在模型仿真中要求的步骤执行。
	停止仿真。
	点击下拉箭头，选择仿真运行类型： • 'Interpreted' - 动态执行仿真（企业版、统一版和终极版） • '手动'-节通过仿真手动（专业版中唯一可用的选项） • 'Executable' - 在可执行状态机上运行仿真时选择
	单击下拉箭头，从选项菜单中选择对仿真脚本和输出执行特定操作的选项，例如编译、运行、生成和视图断点。
	改变仿真的执行率，在 0% 和 100% 之间；在： • 100%，仿真以最快的速度执行 • 0% 模拟器在每条语句处中断执行

注记

- 只有激活了有效的仿真执行脚本，仿真工具才会激活
- 您可以通过在执行分析器窗口中设置仿真脚本的复选框来将其设置为当前默认值

仿真断点

仿真断点和事件窗口的 仿真断点”选项卡可让您中断和检查仿真过程。

当动态执行仿真（在企业统一和终极版本中）时，该过程将自动进行 - 如果您想在某个点停止执行以检查变量、检查调用堆栈或以其他方式与模拟器交互，您可以在一个模型元素的方式与使用一行源代码的方式大致相同。当模拟器到达断点时，执行停止并将控制权返回给Enterprise Architect 。

访问

功能区	仿真>动态仿真仿真>断点>仿真断点

断点

仿真逐步执行模型，使您能够验证您的行为模型的逻辑；仿真在到达定义为断点的元素时停止。

可以定义为断点的UML元素包括行动、活动、状态和大多数其他行为节点，例如决策、初始或终点。

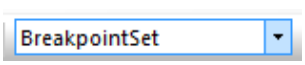
可以定义为断点的UML关系包括交互。

断点存储为给定Enterprise Architect项目的断点集。

元素中包含的、有断点的元素，在仿真过程中，仿真元素左上角附近用绿色圆圈标出。如果仿真没有运行，则不显示绿色圆圈。

启用JavaScript后，所有仿真变量都将显示在本地窗口窗口中，并且可以使用仿真窗口的控制台输入字段（工具栏下方）修改这些仿真变量。

工具栏按钮

物品	描述
	为仿真会话启用当前断点集中定义的所有断点。
	删除在当前断点集中为仿真会话定义的所有断点。
	禁用所有断点 在仿真会话的当前断点集中定义。
	将所选元素或序列消息的断点添加到当前断点集。
	修改选定的断点集以在仿真会话中使用。
	执行断点设置命令： • **新集**：创建一个新的断点集 • **Save As Set** ：以新名称保存当前的断点集

	• **删除选定集**：删除当前断点集 • **删除所有集**：删除为图表保存的所有断点集

仿真中的对象和实例

当一个给定的业务、系统或机械流程执行时，其中的活动和行动可能会生成特定类型的对象并对这些对象执行操作，甚至可能会消耗或销毁它们。您可以使用模拟模型来模拟此类对象的创建、使用和使用，该仿真模型使用类、实例对象、属性、操作和（行动销端口对象节点）等模型元素来表示对象和动作。作为同一过程的一部分，模型还可以在不同阶段创建、作用和销毁几个不同的对象。在仿真中表现模型数据或实物，可以使仿真更准确地反映真实过程。

物件概念

团队	描述
模拟类型	仿真元素的类型，如类、枚举或接口。这些可以是仿真中对象的分类器。
模拟对象	作为 SimType元素实例（由其分类）的object 。
属性	属性元素或指定节点（如 ActivityNode） A属性。
手术	SimType元素或指定节点（例如 ActivityNode） A行为。
端口	A类或物件的端口，一个行动销端口的行动，或object活动的物件节点端口
参数/ 活动参数	操作参数；活动参数，具体来说就是ActivityNodes的参数。
投币口	object中属性A实现。 Slot 有A运行时间值，可以通过运行的运行状态值来初始化。如果这些值不存在，系统将使用属性的初始值。
运行时环境	所有对象都存在于JavaScript运行时环境中，因此您可以使用JavaScript创建或更改模拟对象和模拟变量。
显示变量	所有模拟对象、仿真变量或事件在它们生效时都会在本地窗口窗口中被识别出来。在某些情况下，要显示变量，您可能需要在模型中添加断点以在变量存在时暂停处理。 由于显示了所有对象和变量，存在于模拟之外但对其重要的全局变量（例如定义流程的父类和活动元素）也自动表示为默认object变量。活动的预期输出也是如此，作为返回变量。

在仿真中创建对象

在仿真模型中，您可以创建类并创建它们的实例（全局对象）来表示过程中存在的对象，或者定义行动以在过程中的任何时间点生成一个或多个对象。

在仿真模型中创建对象有三个选项：

- 手动创建物件
- 通过 CreateObject行动元素动态创建物件
- 使用JavaScript函数使用("name") 作为行动元素的 影响”，再次动态创建物件

动态创建一个物件后，您还可以实例化该物件的任何内部对象，例如类上的活动属性，并对该内部object执行操作。

手动创建一个物件

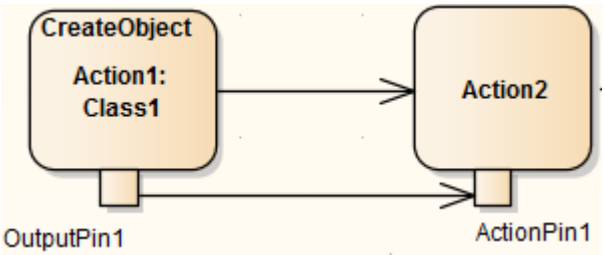
只需通过以下方式在模型的图表上创建一个物件元素：

- 从工具箱的 物件”页面图表一个物件元素并设置其分类器，或
- 从浏览器窗口拖动一个分类器元素并将其作为实例粘贴到图表中

在仿真模型中，您可以自行设置物件属性（例如设置运行状态以重新设置属性的初始值）或动作的行动以作用于物件（例如将其沿流程传递）)并观察仿真中的物件会发生什么。

通过 CreateObject行动创建一个物件

如果您的流程在运行时生成对象，您可以使用仿真行动来模拟这一点。

节	行动
1	在您的活动图表上，从工具箱中图表一个 行动”图标，然后选择 其它| CreateObject 的上下文菜单选项将其定义为 CreateObject行动元素。
2	将 CreateObject行动的分类器设置为物件将成为实例的类。这是在属性窗口 > CreateObjectAction >分类器中设置的，使用 [...] 按钮。
3	在行动上创建一个行动销，种类输出。
4	在处理行动中创建或选择下一个行动序列，并添加一个行动销类输入。 用控制器流量连接器连接两个行动，用控件物件流连接器连接两个行动销。 
5	在图上进行仿真。当创建对象行动被执行时，它创建一个具有分类器属性的物件，并将其存储在其输出销中。物件本身通过物件流连接传递到行动2的输入销，其属性可以在本地窗口窗口中列出，用于仿真。

使用JavaScript创建物件

您还可以在行动元素的 影响”字段中使用JavaScript命令动态创建仿真对象。命令是：

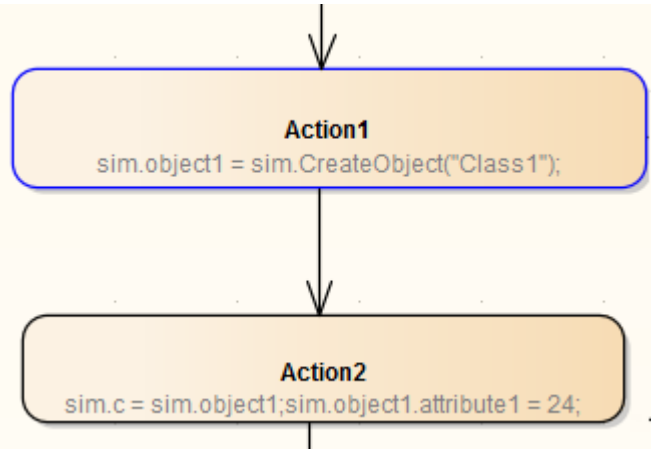
sim.newObject = sim.CreateObject("ClassName");

或者

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

即： 仿真基 类<名称>的物件创作”。分类类将与行动存在于同一个包中。

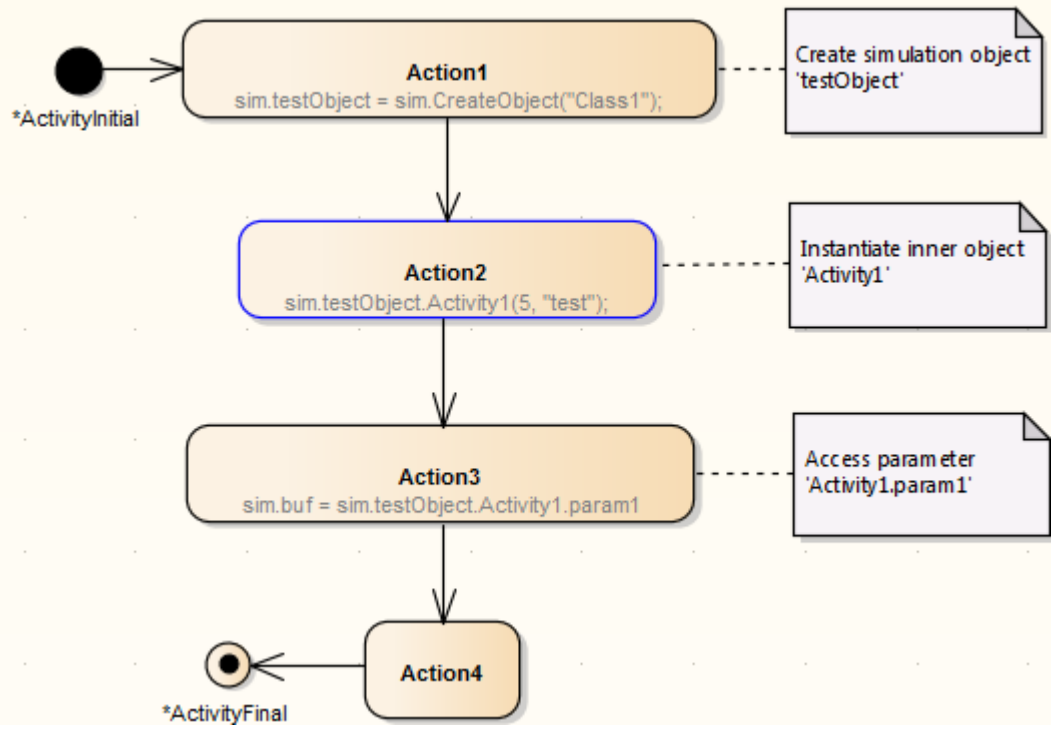
至于CreateObject行动元素，该物件是在仿真过程中创建的，可以被 下游”元素传递和处理。在这个例子中，创建的物件被标识为物件，并且在行动2 中它被访问，并且它的一个属性被赋予了不同的值（也被JavaScript作为行动的影响）。



实例化内部对象

如前所述，您可以使用JavaScript或 CreateObject行动创建一个物件。同样，您可以使用JavaScript或行动来实例化内部对象。

在这个例子中，使用JavaScript ，仿真首先创建了一个基于 Class1 的测试object 。类1有一个活动元素和图表，活动参数1设置为整数 5，活动参数2 设置为string “test”。活动参数1的值被捕获为缓冲区值 ъuf”。


*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

破坏仿真中的物体

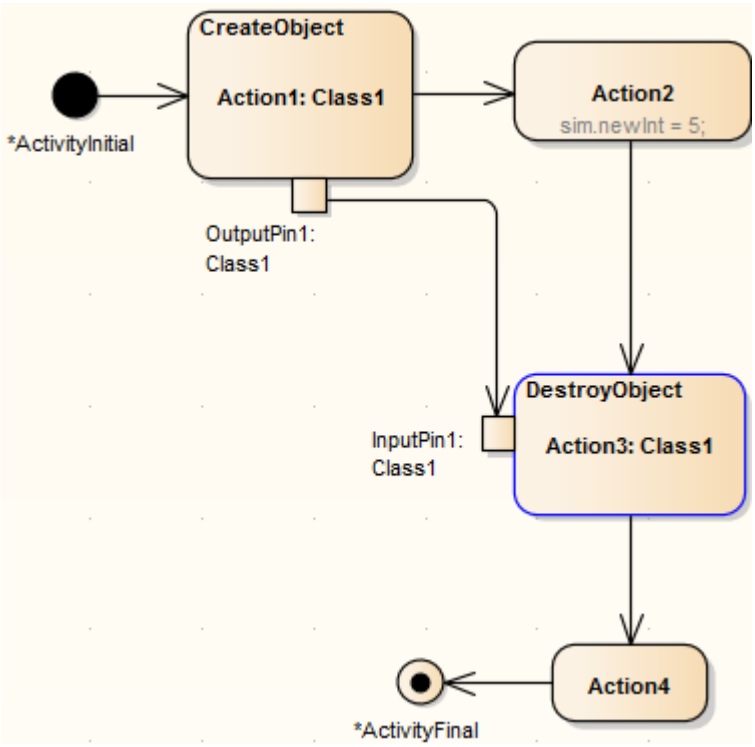
在您的仿真模型中创建或生成对象后，您可以定义行动以在过程中的任何时间点销毁这些对象。所有仿真物品在仿真完成后自动销毁。

销毁仿真模型中的对象有两种选择：

- 通过 DestroyObject行动元素动态销毁 Objects
- 在行动元素中使用JavaScript动态销毁对象

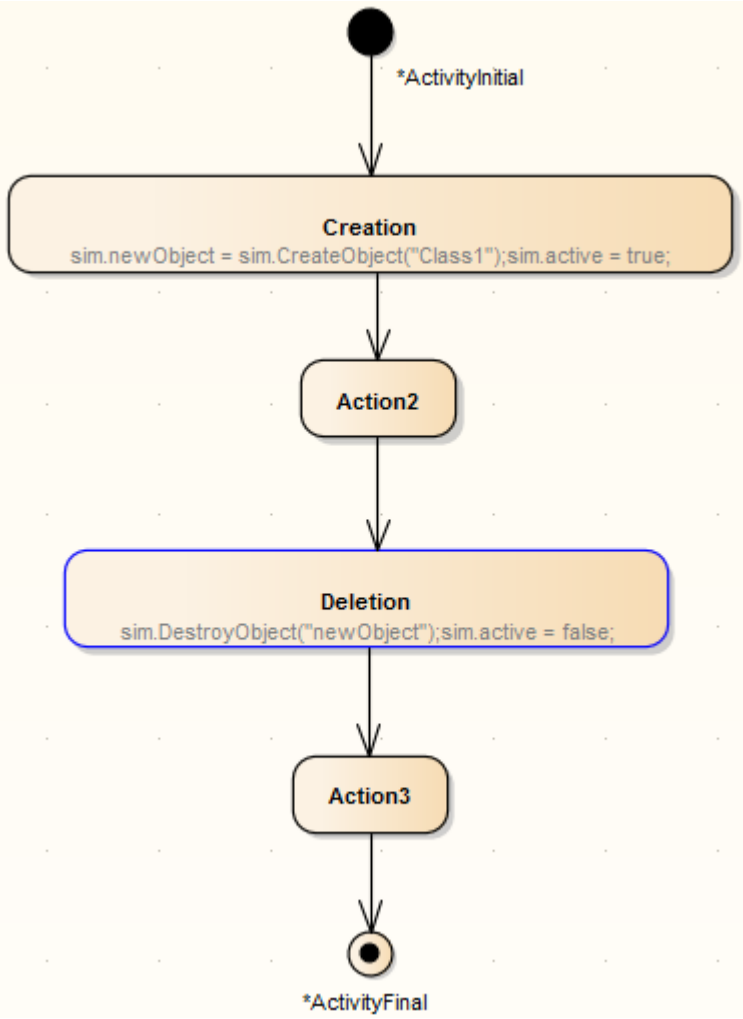
删除的结果可以在局部变量的变化中观察到，在局部窗口上。

通过 DestroyObject行动摧毁物件

节	行动
1	在您的活动图表上，从工具箱中图表一个 行动"图标，然后选择 其它| DestroyObject 的上下文菜单选项将其定义为 DestroyObject行动元素。
2	将 DestroyObject行动的分类器设置为物件是实例的类。 （高级|设置分类器）。 在行动上创建一个行动销，类型输入。
3	将输入行动销连接到来自上一个在行动上操作的行动的物件物件流连接器。在此示例中，对行动进行行动的最后一个物件是创建它的动作。 
4	在图上进行仿真。该过程将物件名称或值作为参数传递到输入行动销。当执行行动动作时，它会从模型中删除具有该名称或值的物件。 示例中，Class1的实例在Action4处理之前被专门销毁，但Action2的结果不受影响。

使用JavaScript销毁一个物件

在行动元素的 属性"对话框中，在 影响"页面的 影响"字段中，键入：
sim.DestroyObject（ 对象名"）
或者
删除 sim.objectFullName
例如：



注记

- 在任何一种情况下，您还可以通过标识执行销毁的行动的物件来销毁全局object （在流程之外创建的对象）；在端口行动的情况下，通过行动物件流连接器将物件名称从物件传递到销上的输入

用JavaScript进行动态仿真

Enterprise Architect的企业版、统一版和终极版提供了使用JavaScript评估仿真时间内的守卫、效果和其他方面的行为的上下文。这为您的状态或活动模型提供了完全自动化、智能的执行，并对断点、执行速度和仿真变量进行了精细控制。

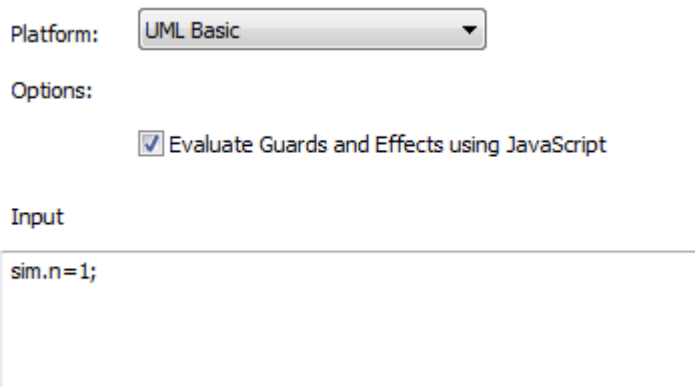
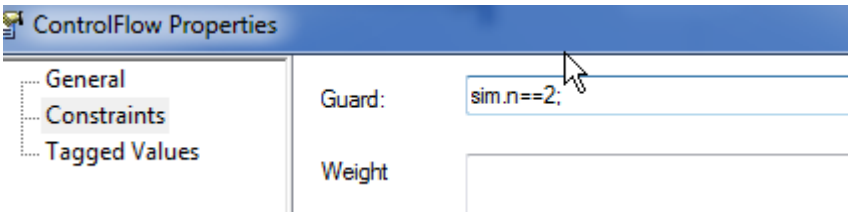
您可以编写使用任何变量的JavaScript 。为了使您能够通过用户界面显示这些变量的值，定义了两个内置对象 - **sim**和**this** - 其成员可以显示在 Local Variables 窗口（也称为本地窗口窗口）中。可以显示的变量示例如下：

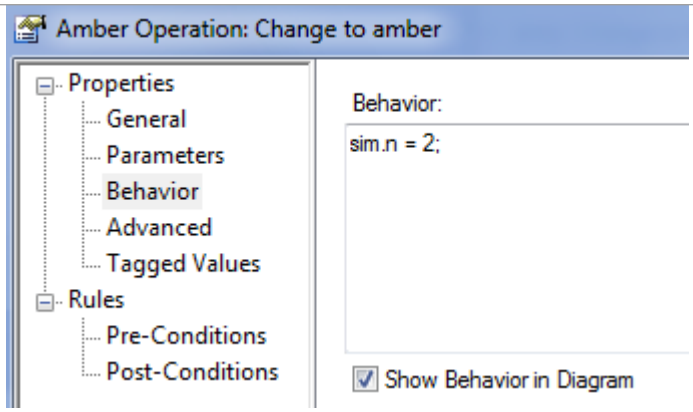
- sim.logger
- sim.顾客.姓名
- 这个.count
- this.账户.金额

推荐的约定是将所有未在 owning类中声明的全局变量或控制变量添加到**sim** object 。相反，将拥有分类器的属性添加到**this** object是正常的。

此处提供了一些使用JavaScript设置仿真行为的位置和方式的示例。 Enterprise Architect附带的 EAExample.eap模型中提供了更多示例。

使用JavaScript

环境	行动
分析器脚本	如果您在执行分析器窗口的 输入”字段中输入J JavaScript代码，该代码将被注入仿真并在仿真开始前执行。这对于建立 COM 变量、全局计数器、函数和其他初始化很有用。 
转移 和控件流卫士	这是仿真功能的主力。由于Enterprise Architect在仿真中评估每个节点的可能前进路径，它会测试 Guards 的传出转换和控制流，并且只有在有一条真实路径可遵循时才会前进 - 否则仿真被认为是 阻塞”和人工干预是必须的。您必须使用 “=”运算符来测试是否相等。 
元素行为	可以为状态定义进入和退出行为。此类代码将在适当的时间执行，并可用于更新仿真变量和进行其他分配。

	 您还可以通过模型中的物件实例和活动来模拟类的行为。
使用 COM	在Enterprise Architect的模拟器中实现JavaScript的一个非常重要的特征是它支持 COM 对象的创建。这提供了将正在运行的仿真与几乎任何其他本地或远程过程连接起来的能力，并且可以根据外部数据影响仿真，或者根据当前仿真状态潜在地改变外部世界的数据或行为（例如，更新机械Enterprise Architect外部的模型或软件仿真）。创建 COM 对象的语法如下所示： this.name="奇偶"; var logger = new COMObject("MySim.Logger"); 记录器.Show(); logger.Log("仿真开始");
使用求解器	Enterprise Architect中Anywhere有JavaScript代码的地方，例如在动态仿真中，您现在都可以使用名为 "Solver"（Solver类）的JavaScript构造与外部工具集成，并直接使用每个工具中的功能来简单直观地执行复杂的数学和图表功能。这些调用可帮助您轻松地在内置JavaScript引擎和每个环境之间交换变量。支持的两个数学库是 MATLAB 和 Octave。 要使用 Solver类，您需要了解首选数学库中可用的函数以及它们使用的参数，如产品文档中所述。 作为JavaScript引擎的一部分，这些 Solver 类也可以立即被插件 访问插件 作家创建基于模型的JavaScript插件。 有关详细信息，请参阅*Octave Solver*、 *MATLAB Solver*和*Solvers*帮助主题。
有信号行动	可以使用JavaScript直接引发信号事件（触发器）。 BroadcastSignal() 命令用于引发可能影响仿真当前状态的命名触发器。例如，此片段引发名为“CancelPressed”的信号（触发器）。 BroadcastSignal("CancelPressed"); 注记记名为 CancelPressed 的触发器必须存在于当前仿真环境中，并且必须具有唯一的名称。 您还可以使用其GUID调用信号。例如： 广播信号（“{996EAF52-6843-41f7-8966-BCAA0ABEC41F}”）；
IS_IN()	如果当前仿真在与传入名称匹配的任何线程中具有活动状态，则 IS_IN（状态）运算符返回True 。例如，为了有条件地控制执行，可以编写如下代码： if (IS_IN("WaitingForInput")) 广播信号（ 取消按下”） 注记该名称在所有上下文中必须是唯一的。

跟踪()	跟踪（语句）方法允许您在仿真中的任意点打印出跟踪语句。这是在执行期间用附加输出信息补充仿真log的极好方法。 JavaScript仿真将截断超过定义的跟踪消息最大长度的字符串。

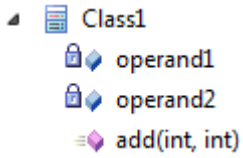
调用行为

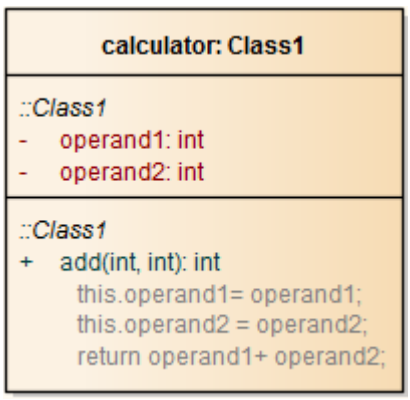
在模拟过程的过程中，您可以在模型中执行类（通过其仿真物件）或活动中定义的行为。在每种情况下，您都使用JavaScript来调用行为。

调用类的行为

模型中A类定义了您要模拟的行为。这种行为在类的操作的行为页面中定义。

例如，类旨在通过操作**add(int , int)**将两个整数相加。在这种情况下，整数是操作的参数，由类、操作数 1 和操作数 2 的属性定义。



节	行动
1	在操作的属性窗口中，选择 行为"选项卡并编辑 行为"字段以将JavaScript仿真对象（ *this*或*sim* ）应用于行为定义。 在示例中： this.operand1=操作数1； this.operand2=操作数2； 返回操作数 1+操作数 2
2	将类拖到您的仿真活动图上并将其粘贴为实例。 在示例中，该物件被称为 计算器"。为清楚起见，此处显示的元素设置为在图表上显示继承的属性和操作以及行为代码。 
3	在仿真图上，对于适当的行动元素，打开 属性"对话框，并在JavaScript中的 影响"页面类型中捕获和模拟物件的行为。 在示例中， JavaScript定义了一个值，该值将通过模拟来自物件的操作行为来提供，就像对两个提供的整数执行的操作一样。那是： sim.result=sim.calculator.add(7,9)
4	运行仿真，并在本地窗口中观察仿真进度。类行为最终体现在仿真结果中。 在示例中：结果 = 16。

调用活动的行为

一个活动元素可以有一个行为，由该元素中的操作定义。举个简单的例子，一个活动可能有一个名为 Get Result 的操作，行为返回 ŏN";。

您可以在活动的子图（即活动的内部）中模拟此行为，并在适当的行动元素的 影响"字段中使用JavaScript语句。在示例中，这可能是：

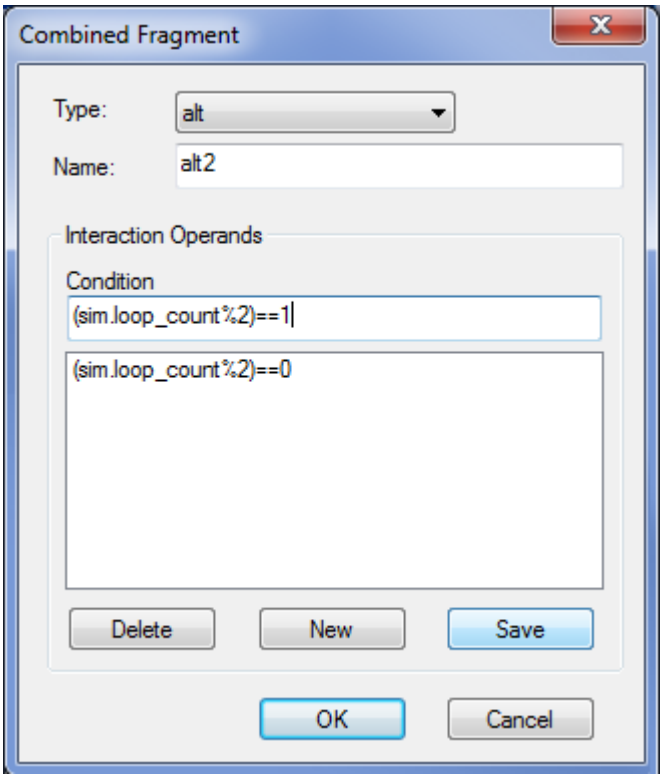
```
sim.result=this.GetResult();
```

该语句调用父活动的操作 GetResult 并将该操作行为的结果分配给 sim.result。您可以在本地窗口窗口中观察仿真的进度和模拟行为的结果，其中（在本例中）最终显示值结果 ŏN"。

交互的条件和信息行为

当你使用一个序列图，你可以模拟条件过程中的情景交互行为，来控制演示过程的仿真。
图表中A信息序列可以链接到操作，因此操作的行为也可以在仿真过程中使用。

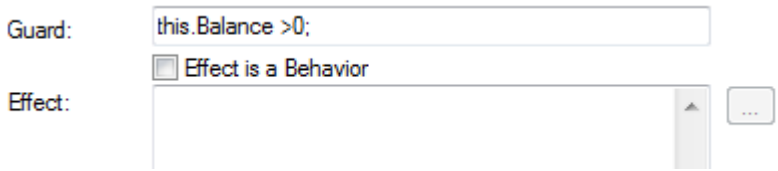
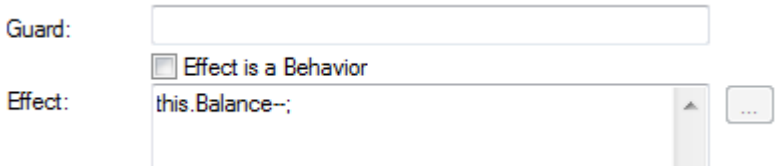
交互条件

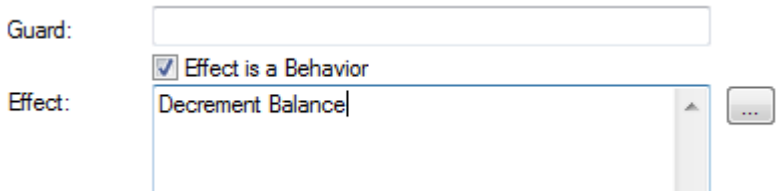
字段/列	描述
操作条件	交互条件是条件语句，只要模拟器确定下一步要走哪条路径，就会被评估。操作数条件通常具有以下特征： • 在 组合片段"对话框中定义 • 用JavaScript • 可以参考仿真过程中定义的变量
添加操作数条件	添加操作数条件： 1. 双击CombinedFragment元素打开 "Combined Fragment"对话框。 2. 单击新建按钮。 3. 在 条件"字段中，键入条件的JavaScript 。 4. 单击保存按钮。 
评估语义	在执行期间，模拟器评估组合条件内的任何操作数条件； CombinedFragment的类型和评估的结果可以决定仿真继续走的路径。

防护条件

防护条件用于控制仿真流程，在仿真过程中执行附加动作或效果。

防护条件

概念	细节
警卫	警卫是条件语句，只要模拟器必须确定下一步要走的路径，就会对其进行评估。守卫通常具有以下特征： • 定义了过渡和控制流，以控制仿真如何进行 • 用JavaScript • 可以参考仿真过程中定义的变量
添加警卫	警卫是在转移 上定义的转移 或选定连接器的 属性"对话框中的 流控件"。守卫条件A是一段JavaScript ，其计算结果为True或False 。例如，这里有一个条件语句，表示当前变量（Balance）大于零。注记使用前缀 *'this'*表示该变量是当前类时间的上下文。 
评估语义	在执行期间，模拟器将检查所有可能的前进路径并评估任何保护条件。该评估可以确定： • A有效的前进路径评估为True ；模拟器将遵循该路径 • 存在两条有效路径；模拟器将阻塞，等待通过控制台窗口进行一些手动输入以解决死锁 • 不存在有效路径；与找到两条路径时的响应相同 -上下文使用控制台窗口等待用户更改执行时间 • 没有路径评估为True但存在默认（未保护路径）；模拟器将采取无人看管的路径
效果	效果是在特殊时间执行的已定义行为： • 进入状态 • 从某个状态退出时 • 从一种状态过渡到另一种状态时（过渡效果） 效果可以是一段JavaScript代码，也可以是当前模拟中对另一个行为元素的调用。
JavaScript效果	JavaScript效果可能类似于A示例，其中 Balance 变量递减： 

调用行为效果	在此示例中，影响是呼叫行为效果。在这种情况下，它调用了一个模型活动，即在别处定义的名为减量平衡的模型活动。然后模拟将进入该图表/行为并继续执行，直到返回到调用影响的点。 
效果的执行顺序	在可能涉及从深度嵌套状态转换到不同父上下文的其他深度嵌套状态的复杂模拟中，重要的是要考虑有关执行顺序的这些规则： • 离开嵌套时间时遇到的所有退出操作（效果）按照嵌套上下文到嵌套最浅的顺序执行 • 接下来执行在转换上定义的所有动作（效果） • 最后，所有的入口效果都是从嵌套最上下文的时间到最深嵌套的时间执行的 所以基本规则是：所有退出动作，然后是所有转换动作，最后是所有进入动作。
关于JavaScript变量的注记	在仿真执行期间要访问和引用的JavaScript变量属于： • sim（例如，sim.pedestrianwaiting）- 通常用于全局仿真变量，或 • this（例如，this.CustomerNumber） - 通常用于引用拥有类属性 这对于让JavaScript引擎知道您指的是仿真变量而不是在例如基本计算期间使用的简单局部变量非常重要。您可以创建任意范围和深度的仿真变量 - 例如，this.customer.name 是合法的限定名称。

触发器

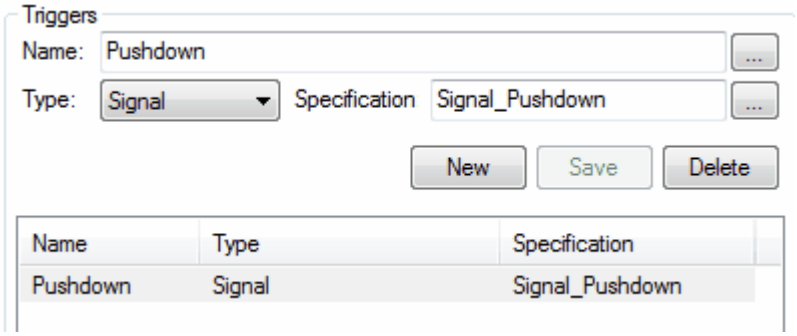
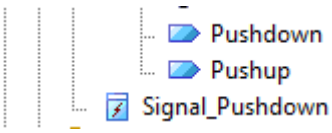
触发器表示可以激活离开当前状态的转换的信号和事件。触发器可能代表真实世界A信号或事件，例如：

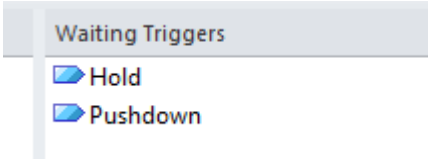
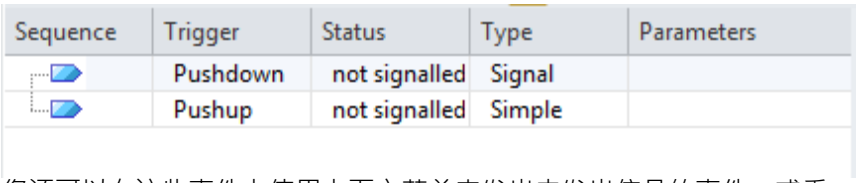
- A按钮被按下
- 收到A消息
- A踏板
- A开关被抛出
- 正在进入或退出A并发区域中的状态

让触发器产生影响

- 必须定义在模拟接收到信号或事件时将触发的转换
- 当前仿真状态或其父级必须具有接受该触发器的传出转换
- 激活的转换必须是无人看守的，或者有一个将评估为True的看守

管理触发器

行动	细节
创建触发器	触发器的元素要么被创建为信号的实例，要么被创建为匿名事件。触发器连接到 转移 "中的转换转移 属性对话框，如下所示。在此示例中，已根据信号 "Signal_Pushdown"定义了一个名为 触发器"的简单示例。 • 省略类型和规范细节会导致简单的匿名触发器。 • 如果需要参数，这些参数在信号上定义并且必须在事件触发时提供  触发器A出现在浏览器窗口的 项目"选项卡中，如下所示： 
使用触发器	触发器通过将它们连接到转换来部署，如前面的示例中一样，并在模拟期间通过根据需要将它们 触发"到正在运行的模拟中来使用。 使用触发器时应考虑以下几点： • A发出有效触发器信号或触发之前，不会发生 触发"转换

	• 当收到触发器时，它将激活依赖于该触发器的所有当前等待转换（即，触发器是广播的） • 对当前子状态的所有父级的所有触发器进行评估；这允许父状态在必要时退出所有子状态 • 一旦在模拟中使用，触发器就会被消耗，如果再次需要，必须重新触发 • 可保存触发器集并手动或自动触发，以方便不同事件模型下的自动模型模拟
射击触发器	触发触发器意味着在当前模拟中发出信号或激活触发器。这可以根据当前仿真的状态和并发性激活零个、一个或多个等待转换。 触发触发器可以通过多种方式实现。最有效的是 等待触发器”列表。 在模型仿真过程中，如果模拟器由于所需的触发器不可用（触发）而陷入僵局，则所有可能的候选触发器列表显示在触发器事件仿真事件窗口的 等待”列表中。  双击此列表中的触发器会将其触发到仿真程序中。其它方式启动触发器包括： 1. 双击事件窗口中的未发出信号的触发器。  您还可以在这些事件上使用上下文菜单来发出未发出信号的事件，或重新发出先前已触发的事件的信号。 2. 使用转移 的上下文菜单转移 需要启动并选择 触发器仿真的信号”菜单选项。

行动行为按类型

你可以通过定义（甚至重新定义）它的类型来改变一个行动元素发起的行为。在仿真中，您可以使用本行动中描述的类型和组中的表来应用和观察许多不同的行为。

行动类型

类型	描述
物件行动	物件行动以特定方式对object进行操作，例如创建、销毁或读取object 。他们包括： • 创建对象 • 销毁对象和 • 阅读自我
可变行动	变量行动在运行时有一个以标记值变量形式与object名称的值关联的变量。它们不仅以object的形式提供变量，还以object的属性（例如属性或端口）的形式提供变量。他们包括： • ReadVariable • 写变量 • ClearVariable • 添加变量值 • 移除变量
结构行动	结构行动作用于结构特征，即活动的属性或对象的分类器的object 。他们包括： • 读取结构特征 • 写结构特征 • 清晰的结构特征 • 添加结构特征值 • 移除结构特征值
调用和接受事件行动	调用和接受事件动作定义了事件的触发器和信号。他们包括： • 发送信号 • 广播信号 • 接受事件 • 发送对象 • 呼叫行为 • 呼叫操作 • 接听电话
杂项行动	行动评估一个值；它必须有一个输入值和一些评估代码作为它的行为或效果。

结构活动仿真

行为模型中更复杂的结构之一是结构活动，它对嵌套结构或评估和执行过程中的一系列动作进行建模。结构活动的评估类型有Conditional节点和Loop节点，你可以很容易地模拟这两种节点。

条件节点

A条件节点基本上由一对或多对测试/体分区组成，每一对被称为一个子句。测试分区由测试条件的活动图元素组成，如果满足该条件，则执行体分区中的进一步活动图元素以产生结果。

如果有一个子句，条件节点要么输出体分区的结果，要么不输出结果。如果有多个子句，则控制从一个测试流向下一个测试，直到满足某个条件并执行一个体分区以产生结果，或者所有测试都失败。

仿真目前支持在属性窗口的 条件"选项卡中使用 确定"复选框设置。其他两个复选框设置被忽略。如果 'Is Assured' 复选框是：

- 选中，至少要满足一个测试，所以执行它的体并输出一个结果；如果没有满足测试并且没有结果输出，则条件节点被阻塞并且处理不能继续超出它
- 未选中，可以满足一个测试并输出一个结果，但是如果没有满足测试并且没有结果输出，超出条件节点仍然可以继续处理

您可以通过键入JavaScript *sim* 来模拟一系列可能的路径和结果。在每个条款的每个分区内的行动元素的 影响"字段中定义或导致特定测试结果和体结果的语句。这些模拟。语句必须标识设置的条件节点、子句和输出销的完整路径。例如，在测试一个人是否有资格成为老年人时：

```
if (sim. Person .age >=65)
sim.AgeCondition.Clause1.Decider1=true；
else
sim.AgeCondition.Clause1.Decider1=false；
```

条件节点称为*AgeCondition* ，测试在*Clause1*中，该测试的 OutputPin 是*Decider1* 。

Loop节点

循环结构活动节点通常表示 While、Repeat 和 For 循环语句A建模等价物。每个 Loop节点有三个分区：

- 设置 - 启动要在循环退出条件中使用的变量；它在进入循环时执行一次
- 测试——定义循环退出条件
- 体- 重复执行，直到测试产生False值

您可以通过将活动图表元素从工具箱页面拖入设置、测试和体分区来定义循环节点。体分区可以包含相当复杂的元素节点结构，定义了 Loop 在这个过程中实际产生的内容。

Loop节点有多个行动销：

- 循环变量（输入）- 要通过循环处理的初始值
- 循环变量（输出）- 执行测试的变化变量
- Decider -测试分区内的一个输出销，每次执行测试后都会检查其值，以确定是否执行循环体
- 体输出-体分区中处理的输出值，为循环的下一次迭代更新循环变量输出销，以及
- Result -测试分区最后一次执行的值（也就是最后一次执行体分区传回来的值）

您可以通过输入JavaScript *sim* 通过循环模拟不同动作和输出的效果。在每个分区内的行动元素的 影响"字段中定义或导致特定测试结果和体结果的语句。这些模拟。语句必须标识正在设置的 Loop节点和输出销的路径。例如，在测试分区中的一个行动中：

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

活动返回价值仿真

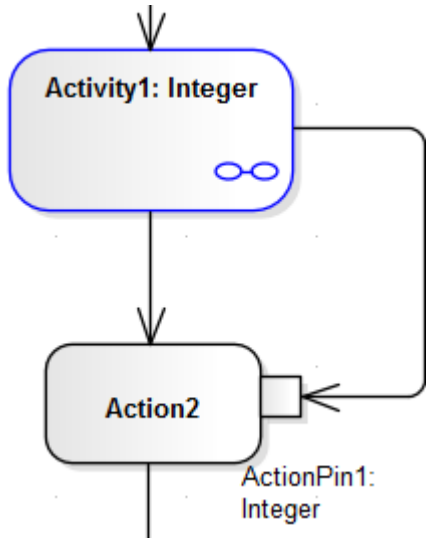
一个活动可能会产生一个返回值作为它所代表的过程的输出。您可以在以下三种情况下模拟该返回值如何传递到流程的下一个阶段：

- 活动只是产生一个返回值，直接传递给下一个行动
- 活动具有一个或多个活动参数 - 在图表上由活动节点表示 - 接受输入值到活动的子活动或保存来自活动行动输出值，并且输出活动参数收集并传递返回值
- 活动由复制活动行为并向前传递返回值的行动实例化

活动返回价值通输出

（这种方法是Enterprise Architect模拟所独有的，模仿活动参数的效果而不必存在。）

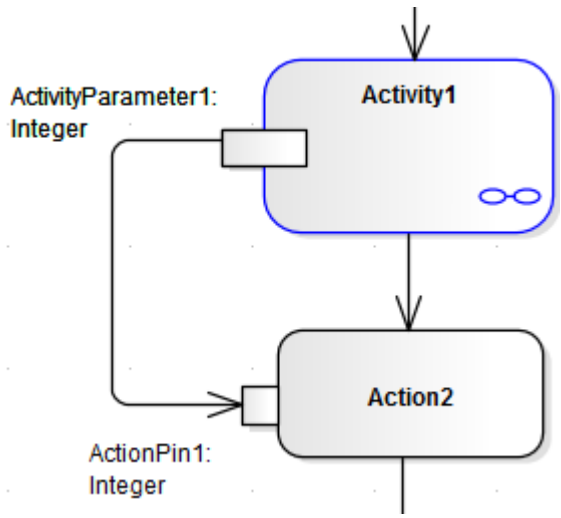
活动有一个返回值，它通过一个行动物件流连接器从活动元素销行动



您可以通过设置一个简单的JavaScript语句来设置活动的子元素中的返回值（例如行动=12;）来模拟这一点，然后运行模拟，在本地窗口中销传递给活动元素的值窗户。

活动参数输出

如果该活动有一个活动参数，则其值传递给相应的活动节点，然后通过一个物件流连接器，行动给过程中下一个活动的输入活动行动销，如图所示：

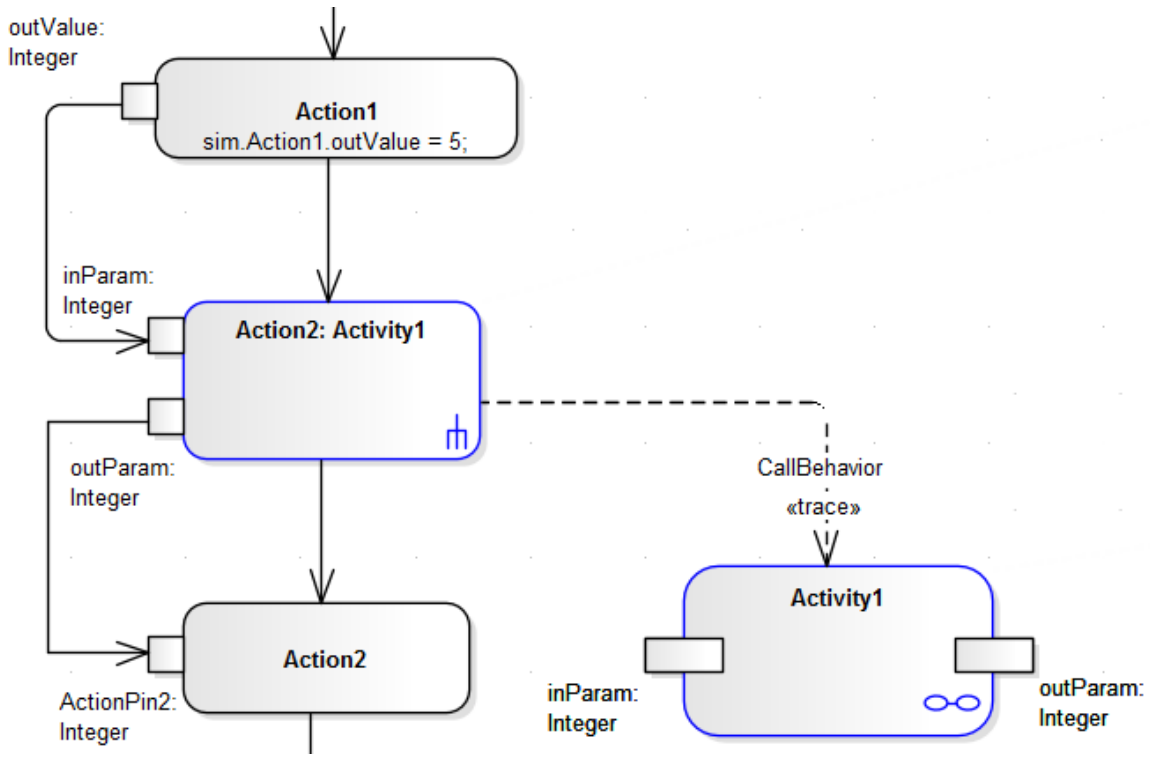


在本地窗口窗口中，您可以观察参数的默认值传递给行动销，也可以在活动的子行动中使用JavaScript来模拟活动中的值的更新。例如：

this.ActivityParameter1=20;

呼叫行为行动

一个活动可能会在一个进程中使用多次，在这种情况下，您可能希望每次都生成一个单独的活动实例。您可以使用行动来创建活动object并执行其行为。输入和输出活动参数与行动上的相应输入和输出行动销（参数）绑定。



当您模拟包含活动的过程部分时，您提供一个输入值（如在行动1中），该输入值传递到行动行动上的输入行动销中，从而创建活动的一个物件。行动执行活动的行为，使用输入行动销作为输入活动参数，使用输出行动销接收返回作为输出活动参数。然后使用行动物件流连接器将活动返回值传递给下一个行动上的行动销。您可以在活动的行动中提供JavaScript语句来作用于输入值并生成返回值，例如：

sim.buf=this.inParam;和

this.outParam=sim.buf + 11：

仿真事件窗口

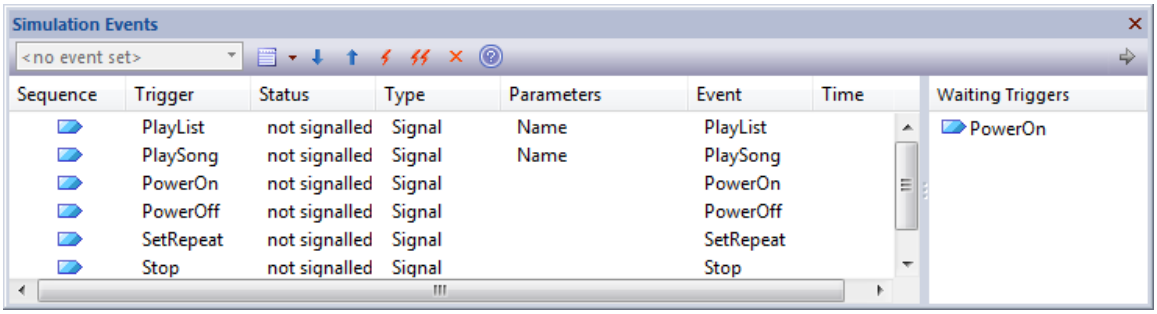
仿真事件窗口是您在仿真中管理触发器和事件集的地方。其主要功能是：

- 为模拟添加、删除和重新排序一组触发器
- 显示当前运行模拟的已触发、丢失和等待事件列表
- 提供选项启动任意任意触发器进入当前模拟
- 提供一个方便的 等待"模拟正在等待的触发器列表
- 保存触发集以供以后在手动和自动仿真中使用
- 接受从浏览器窗口拖入当前列表的触发器
- 在触发前输入等待触发的触发参数

由于在模拟中消耗了触发器，它们的状态和位置记录在仿真事件窗口的主体中。

您可以将触发的触发器的log保存为触发器集或事件集，以便在另一个仿真运行中重新应用，您可以手动或自动执行。有关构建和使用触发器集合的更多信息，请参阅触发器集合和自动触发主题。

此图说明了仿真事件执行过程中的窗口。



访问

功能区	仿真>动态仿真>事件

列详细信息

字段/列	行动
序列	在模拟期间和模拟之后，指示触发或预期触发触发器的序列中的位置。注记如果触发的触发器超出序列，它将被移动到已发出信号的事件部分的底部。
触发器	触发器的名称 -触发器用于启动事件的简单方式。
状态	表示触发器的状态。值可以是： • used - 触发器已被触发并且处理已通过 • lost - 触发器已在列表中触发，但没有效果 • 已发出信号 - 一个或多个转换触发并消耗了触发器 • 未发出信号 - 触发器尚未触发

类型	指示触发器的类型。目前仅支持： • 信号 • （无类型）匿名触发器
参数	对于一个触发器的信号，最初显示信号规范触发所需的参数。例如， 登录"信号可能包括用户名和密码参数——每个触发的调用可以使用不同的参数。 每次模拟触发触发器时，系统都会提示您输入值。当触发器设置为未发出信号时，您还可以直接在列表中编辑值。 参数对于测试模拟中的条件逻辑以及模拟来自模拟外部的各种输入和数据非常有用。
事件	为一个： • 信号的触发器，标识信号规范 • 对于匿名触发器没有任何价值
时间	触发信号的模拟时间。注记这是一个绝对（真实世界）时间，而不是相对模拟事件时间。
等待触发器	触发器可从当前状态中选择的简单状态，包括在单个转换中可能有多个触发器的状态。双击触发器以将其添加为当前事件序列中的下一个触发器并发出信号。 您可以通过单击面板正上方的灰色箭头来显示和隐藏此面板。

工具栏项

选项	行动
	使用此下拉列表来选择和使用先前定义的触发器集。 在运行模拟之前，选择一个先前定义的触发器集以用于下一次模拟运行。您可以通过选择 <no event set> 选项来选择不使用触发器集。
	单击以创建和删除触发器集： • 保存集——将当前触发列表保存为新的触发集；系统提示您输入新集合的名称 • 将集另存为 - 以新集名称创建当前集的副本 • 删除选定集 -删除当前触发集 • 删除所有图表图表-删除当前图表的所有已保存触发器集
	在触发器的触发序列中将所选触发器向下移动一行。 如果所选行下方没有信号触发器，则此选项不可用。
	在触发器的触发序列中将选定的触发器条目向上移动一行。 如果所选行上方没有信号触发器，则此选项不可用。
	单击启动选定的触发器。您也可以通过双击触发触发器。

	单击以打开和关闭自动触发。 自动触发将按顺序触发触发器集中的未发出信号的触发器。如果您的设置与有效的执行路径匹配，则模拟将运行。输出序列或未使用的触发器将 丢失 ”。 断点会暂停自动触发，您需要单击下A触发器才能恢复自动触发模拟。
	从列表中删除选定的触发器。

上下文菜单选项

选项	行动
信号	信号，或触发，选定的未发出信号的触发器。
删除选定	从序列中删除未发出信号的触发器。
重新选择信号	再次触发已使用或已发出信号的触发器。
全部设置为无信号	将所有已使用或已发出信号的触发器设置为未发出信号。
触发器列表	清除窗口中的所有触发器，无论它们的状态如何。

等待触发器

当模拟达到任何状态（对于任何线程）需要状态触发器地继续进行的点时，模拟将被有效暂停并且控制返回给系统。模拟现在正在有效地等待某种形式的事件（现实世界的信号）继续进行。等待触发器列表有助于触发器应该手动发出信号。

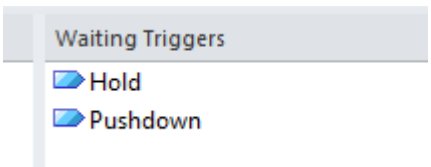
访问

功能区	仿真>动态仿真>事件 右侧窗格列出可用触发器。

仿真事件窗口的触发器名单如下：

- 在每个仿真周期中填充任何触发器的说明，如果发出信号就会立即生效
- 填充离散集（任何重复都不会显示为简单的触发器有效地广播到所有转换）
- 通过双击感兴趣的触发器来激活
- 包括所有可能的触发器——包括那些激活当前嵌套状态的父级的转换

这个例子表明，当前的模拟已经达到了触发器可能影响执行流程的程度。



由于简单的触发器及其影响，该列表可以同样有效地引用这些示例情况中的每一个：

- 单个状态A两个输出转移，分别等待Hold和Pushdown；触发其中之一将激活模拟中的相关转换
- 单个状态A两个或多个可能触发相同的转换，例如安全摄像头被运动检测器、声音检测器或热检测器打开
- 两个（或更多）线程（并发区域）每个都有状态等待保持或下推；触发其中一个触发器将导致线程等待该触发器继续进行，而其他线程将保持阻塞状态
- A状态正在等待其中一个触发器，而父状态正在等待另一个；触发触发器将导致关联的转换被触发，并且子或父相应地继续
- 这些的任意组合

重新信号触发器

可以重新发出简单的信号作为触发器其他触发器实例以发出信号的快捷方式。

访问

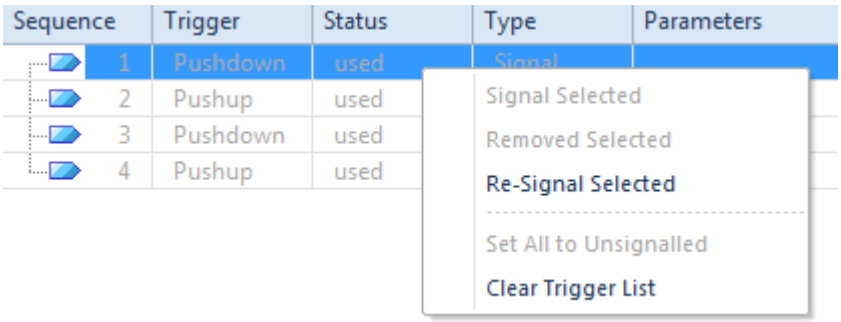
显示仿真事件窗口，然后右键单击该窗口中的触发器并选择 重新发出选定信号”选项。

功能区	仿真>动态仿真>事件>右键单击现有触发器>重新选择信号

触发器列表

仿真事件窗口包含已触发的触发器事件列表。通过右键单击触发器你想再次发出信号的简单，你可以使用时间来上下文重新发出信号。

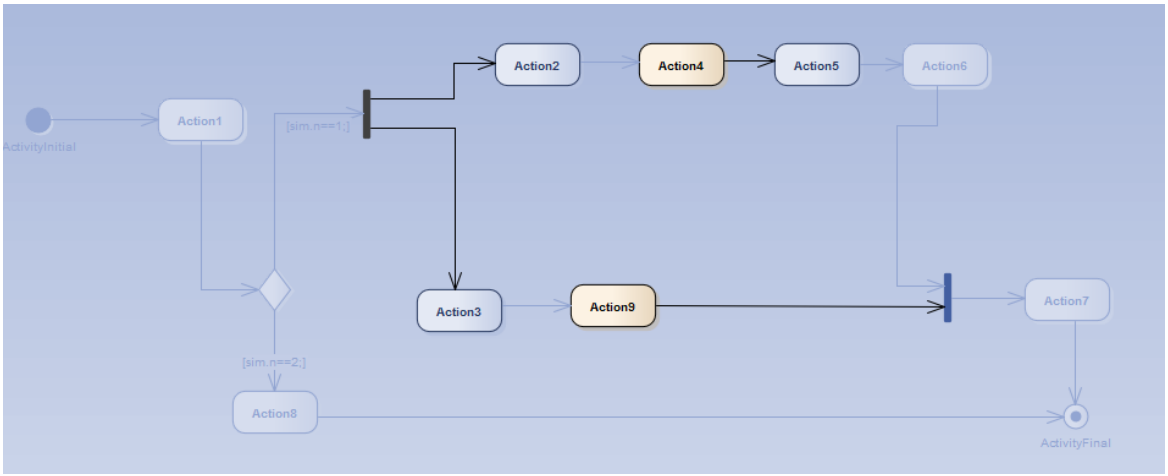
此图像演示了重新发送信号的作用。当重新发出信号时，会制作一个新副本并将其放置在已发出信号的触发器列表的末尾，并在此处再次自动触发。



多线程-分叉和汇合

模型模拟器提供了使用分叉和汇合节点处理多线程模拟的能力。

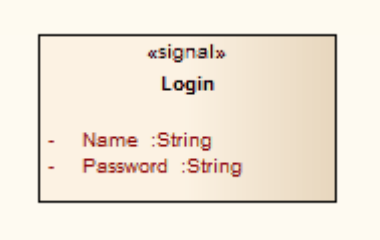
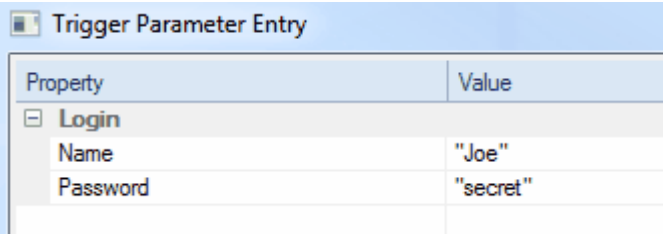
- 在示例中，当前执行点已分叉为两个线程，每个线程都有自己的活动节点
- 随着这个例子的进行，下层分支将在汇合节点等待，直到上层分支完成所有行动
- 一旦两个线程合并为一个线程，仿真将作为一个线程继续进行，直到完成
- 当自动单步执行时，每个线程将被视为在一个模拟 周期"中执行单个步骤 - 尽管当单步执行或在断点时，行为是在线程之间交替执行，因为每个线程接收处理时间
- 注记在示例中， 调用堆栈窗口将显示两个活动线程和一个 p̈a"线程；一旦线程合并，将返回单线程执行
- 另请注记，局部变量在所有线程之间是共享的（全局的）；如果要在线程上仿真私有变量，则必须在每个线程开始时创建新的仿真变量 - 使用现有全局数据预加载此类变量

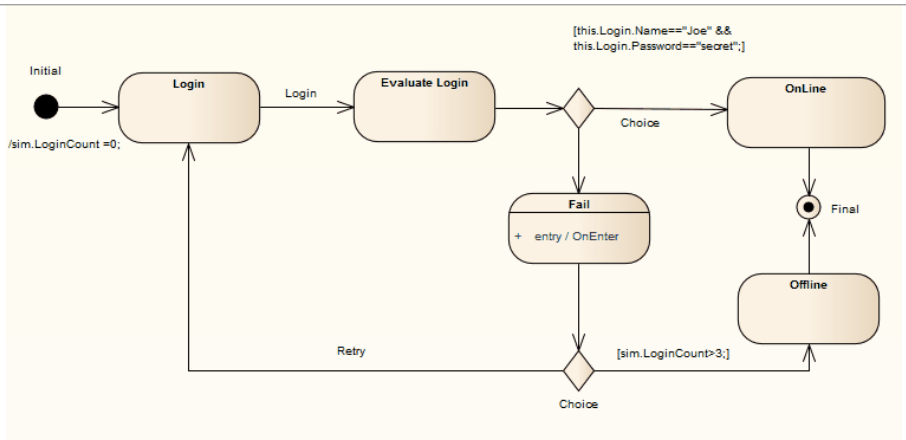


触发器参数

触发器是在触发时与触发器一起传递给模拟的参数。它们允许根据在运行时通过触发的触发器（事件）传递到模拟的变量和数据来指定复杂的行为决策。

参数

参数	细节
介绍	要使用触发参数，您： • 首先创建一个具有适当属性的信号元素，这些属性将在运行时成为您的参数 • 在图表中合适的转换上，创建一个基于之前创建的信号的触发器 • 在运行时，会提示输入合适的参数——然　它们与触发器一起传入
信号	信号元素是A模板或规范，可以从中构建实际的触发器。这个例子有两个参数，一个名称和一个密码。这些将在执行时手动或作为预定义触发器集的一部分填充。 
触发器参数	触发器'提示'要求为每个参数提供合适的值。注记你需要用双引号将字符串括起来，否则解释器会认为你指的是其他变量。 
示例图表	这是一个使用触发参数的示例图。在 Evaluate Login状态下，模拟检查作为触发参数传入的变量，并决定是接受凭据还是拒绝凭据。


[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

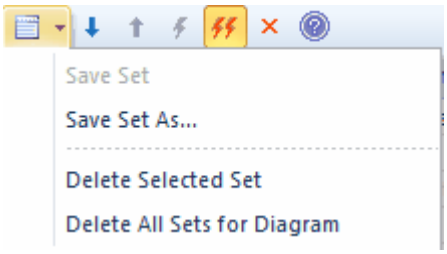
触发器的设置和自动射击

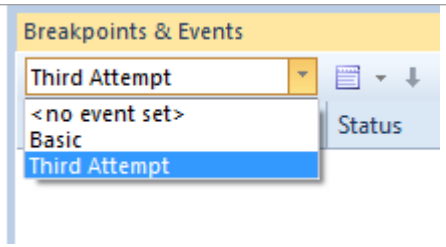
触发器集合是自动化和简化仿真模型的执行、测试和验证的有效手段。通过重复使用触发器集（带或不带参数），可以快速有效地遍历许多模拟场景，无论是手动还是使用 自动触发"工具自动进行。

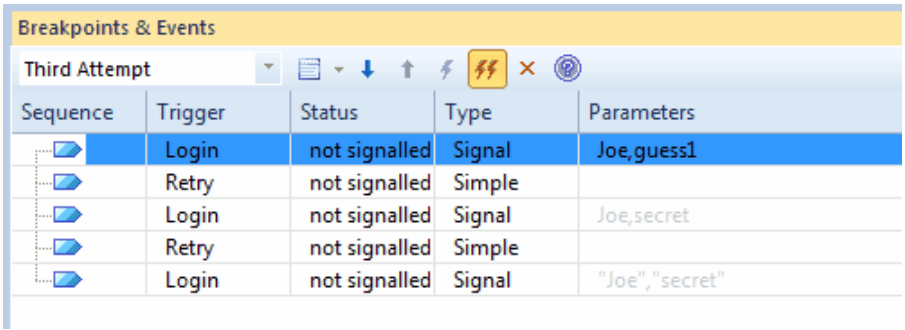
访问

功能区	仿真>动态仿真>事件

关于触发器

方面	细节
触发器套装	• 与关联图一起存储 • 由一组序列的列表触发器 • 可以在必要时包含触发器参数 • 可以根据需要手动双击触发器启动 • 可用作 自动触发"行为的一部分以自动执行 • 由仿真事件窗口管理
管理集	可以通过手动将触发器拖动到活动触发器列表中来创建触发器的集合，然后使用 管理触发器"下拉菜单保存新集合。 还可以将在单个模拟设置期间建立的一组触发器保存为新组。这便于通过模拟创建多个测试路径，基于为每个测试用例保存手动触发的触发器。  您还可以删除一个集合并删除当前图表的所有集合。 也可以加载一个集合，修改参数和/或触发顺序，并用新名称保存集合。这是快速创建一套模拟测试脚本的便捷方法。
使用集合	要使用触发器集，您首先从触发器集下拉列表中按名称选择它，如本示例图像所示。选择后，它会加载带有定义的触发器集的触发器列表窗口。 注记特殊项<*no event set*>表示当前未选择任何集合。在每次模拟开始时，如果选择了一个集合，它将重新加载以进行下一次运行。如果选择<*no event set*>，触发器列表将被清除。

<table>
<tr><td></td><td> 

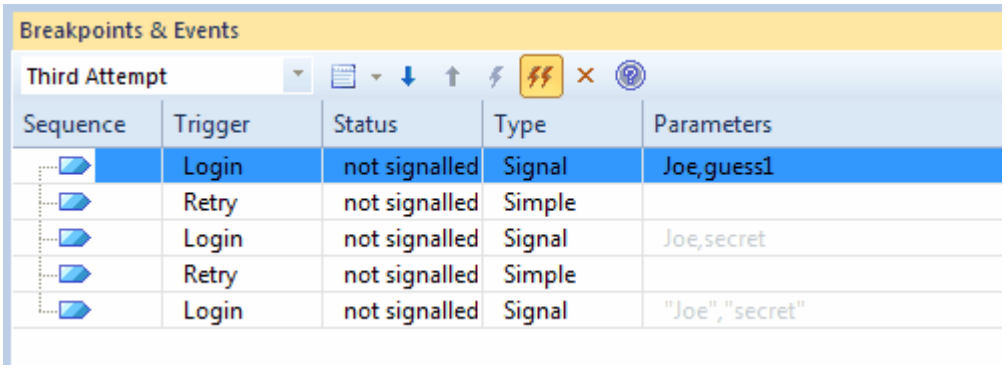
选择触发器集并加载触发器列表后，您有两个选项：
• 根据需要手动触发触发器
• 使用自动触发特征来完全自动化模拟
 
</td></tr>
<tr><td>自动射击</td><td>自动触发是一种简化模拟的便捷方式。加载触发器集后，如果您选择自动触发按钮  ，则Enterprise Architect将在模拟陷入僵局时自动拾取等待的触发器。在实践中，这意味着与模拟路径完全匹配的触发器集将自动运行，无需您的干预。
由于您可以保存任意数量的具有不同路径和触发参数的触发集，因此您可以有效快速地测试和处理许多不同的场景。</td></tr>
<tr><td>自动触发规则</td><td>当模拟在启用自动触发的情况下运行时， Enterprise Architect将等待直到达到模拟 阻塞”或稳定的点，等待一个或多个触发器来推进模拟。届时，列表中第一个未触发的触发器将被拾取并触发到模拟中。结果取决于相关性，也可能取决于触发器的参数。
• 如果触发器与 等待”触发器匹配，则立即消耗它并进行模拟
• 如果触发器不匹配 等待”触发器或可能的父转换，则触发器 丢失”并且模拟保持在当前状态；这对应于一种场景，例如用户连续多次按下 开启”按钮 - 除了第一次按下之外没有其他效果</td></tr>
</table>

使用触发器来仿真一个简单的事件序列

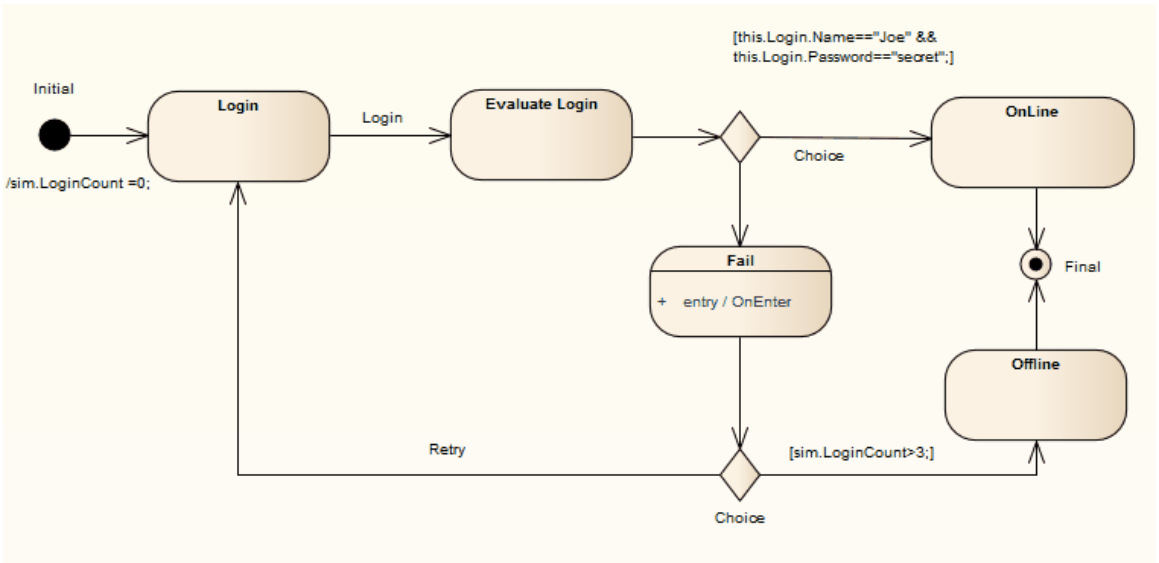
作为触发器集有用性的一个简单示例，请考虑此示例触发器集和关联图。

在此示例中，我们使用用户名和密码模拟了一个简单的 "三击即出"登录过程。成功路径是等待名字 "Joe"和密码 "secret"。 （注记- 引用字符串的参数用引号括起来非常重要，否则解释器认为该名称指的是模拟中的另一个变量。）

- 通过1尝试乔和猜测1 - 失败
- Pass 2 尝试*Joe*和*secret* ，但因为它们指的是变量，而不是字符串 - 这也失败了
- Pass 3显示了引用触发参数的正确方法，模拟成功



这是一个模拟需要用户名和密码对的登录过程的简单图表。



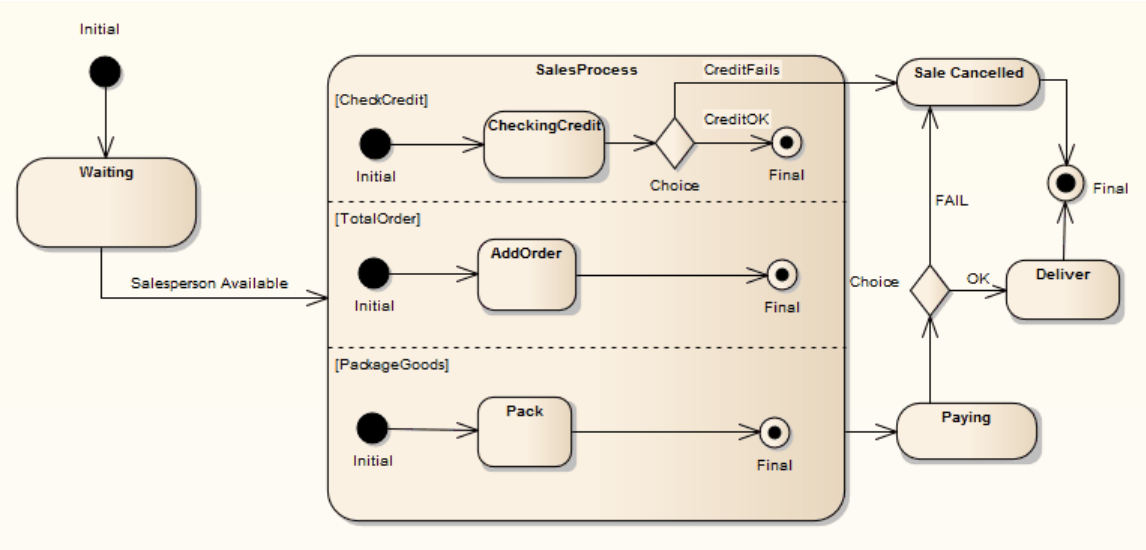
多线程-并发状态区域

状态中的并发区域表示在一个整体父状态中并行发生的状态变化和处理。当一个区域引发事件或修改另一个区域所依赖的模拟变量时，这尤其有用。例如，一个区域可以包含一个模拟计时器，该计时器在设定的时间间隔内引发事件，从而调用其他区域内状态的状态变化。

并发区域与分叉和汇合基本相同，逻辑和处理规则相似。

在示例中：

- 当转换到 SalesProcess 时，每个区域同时激活
- 检查信用，订单总额和所需的货物打包
- 但是，如果选择失败，这会触发到 Sale Cancelled状态的转换；注记当这种情况发生时，整个父状态和所有拥有的区域都会立即退出，而不管它们的处理状态
- 如果Credit选择成功，则区域移动到最终状态，一旦其他区域都达到自己的最终状态，则可以退出父状态

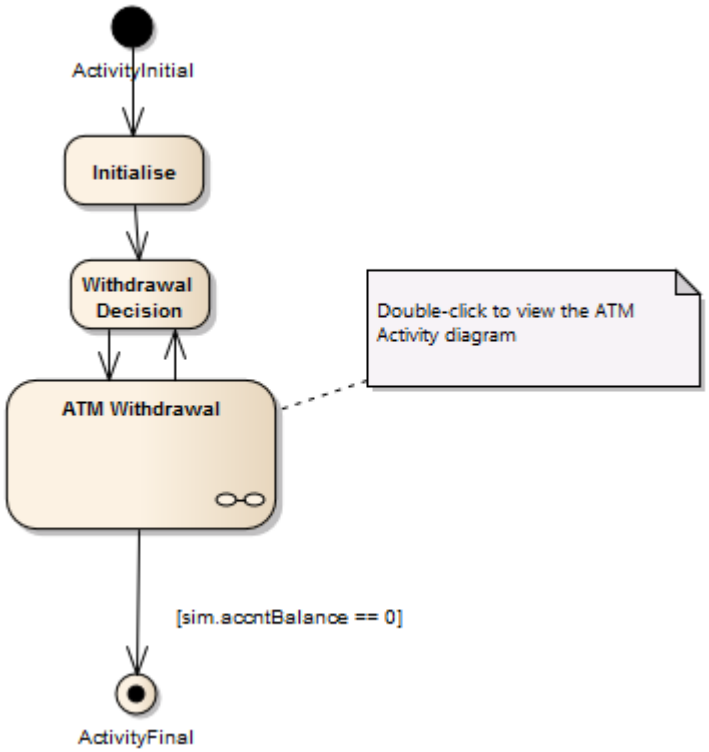


使用复合图表

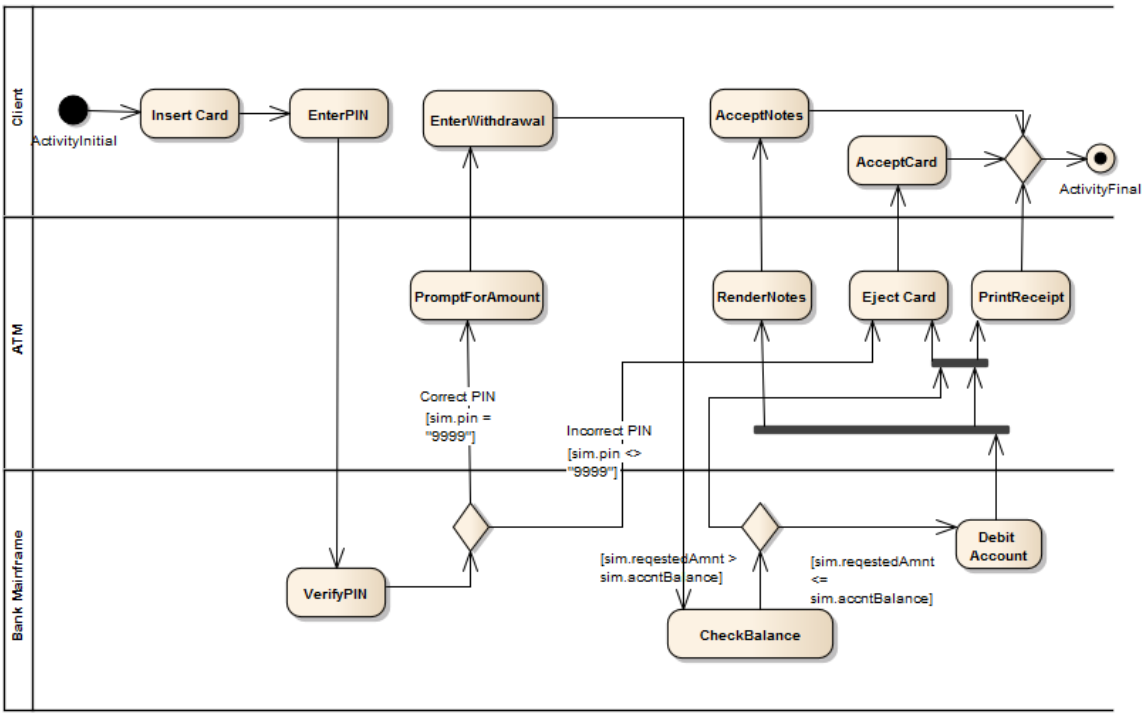
如果要模拟包含在不同图上表示的分支的处理（例如，为了降低主图上的复杂性，或者隐藏仅在异常下操作的处理区域），可以使用复合元素来表示并访问其子复合图上的分支。当您运行模拟并到达复合元素时，它会打开子图并对其进行处理，然后再返回（如果适用）到主处理路径。这是在复杂流程中遵循处理路径的绝佳方法，用复合活动元素表示流程的各个部分，这些元素在各自的子图中扩展了实际处理。您可以有多个复合元素来表示流程的不同阶段或分支。

需要注意的一个方面（这将通过模拟失败来揭示）是让多个线程在不同的图表上同时处理。如果模拟也跟随当前图表上的另一个线程，则模拟无法传递到新图表。

此图提供了 ATM 现金提取流程的概述：



ATM取款活动是一种复合元素。如果您双击它，您将打开并显示子图，这是提款过程的更详细的细分。同样，模拟将打开并处理子图。


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

Win32用户接口仿真

Enterprise Architect支持模拟使用 Win32 ®用户接口配置文件创建的对话框和屏幕，以将用户界面设计与定义的系统行为集成。可以使用行为模型（例如状态机）中的JavaScript命令以编程方式引用和调用对话框，从而提供完全可定制和完全交互式的行为模型执行。

按钮控件可用于广播信号，在单击按钮时触发触发器。信号参数可以从对话框输入字段中填充；例如，捕获并发送用户名和密码以进行评估。

使用 Win32用户接口配置文件设计的对话框（并且与正在执行的行为模型存在于同一包分支中）将在模拟开始时在后台创建为新窗口。可以在设计时通过 Win32用户接口配置文件提供的标记值自定义影响每个对话框和控件的外观和行为的各种属性。

要在模拟过程中通过JavaScript与对话框交互，使用 对话框"模拟级关键字，后跟句点和对话框名称，然后可以访问属性和方法；例如，显示对话框，或设置 编辑控件"的文本值：

对话框.登录.显示=真；

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

例子

要查看 Win32 用户接口仿真示例，请打开用户模型并找到图表：

示例模型>模型仿真>状态机模型>顾客登录>顾客顾客登录

公共属性

这些常用属性和方法在大多数支持的 Win32用户界面控件类型上都可用。

属性/方法	描述
使能够	布尔值 启用或禁用用户交互。
移动到（x，y，宽度，高度）	将窗口移动到指定坐标并设置窗口的高度和宽度。
节目	布尔值 显示或隐藏对话框。当此属性设置为False时，对话框将移出屏幕。
文本	字符串 设置对话框或窗口的标题。

JavaScript函数

函数	描述
广播信号（ string信号）	向模拟事件队列发送信号。 参数：

	• 信号：字符串——要广播的信号的名称
信号（ string信号，数组参数 ）	将带有附加参数的信号发送到模拟事件队列。 参数： • 信号：字符串——要广播的信号的名称 • 参数： Array – 为这个信号提供的附加参数 示例： UIBroadcastSignal("登录",{名称:文本, Password:文本});
ShowInterface (string InterfaceName, boolean Show)	已弃用。请参阅 对话框"控件上的显示属性。例如： 对话框.HelloWorld.Show = true;
InterfaceOperation (string接口名称， string控制名称， string操作名称，[string arg1]，[string arg2])	已弃用。操作可以直接从控件中引用。例如： dialog.HelloWorld.ListControl.InsertItem("测试", 2);
GetInterfaceValue (string InterfaceName, string ControlName, string OperationName,[string arg1],[string arg2])	已弃用.属性可以直接从控件中引用。例如： 文本;

注记

- 控件必须在对话框中；对话框外的任何控件都不会被解释
- 对话框和控件必须在 Win32用户接口图上
- 简单 UI 控件和基本 UI 控件也可以在模拟中使用，但与 Win32 UI 控件相比，功能有限
- 对话框名称和控件名称必须是唯一的；如果存在多个同名控件，则模拟将无法区分它们
- 对话框名称和控件名称中的空格被视为下划线
- 对话框名称和控件名称区分大小写

支持的 Win32 UI 控件

该表标识了Enterprise Architect中用于用户界面设计和模拟的所有 Win32 UI 控件。

访问

功能区	设计>接口>工具箱:图表> 在 查找工具箱项”对话框中指定 用户  - Win32”
键盘快捷键	Ctrl+Shift+3 :  > 在 查找工具箱项”对话框中指定 用户接口- Win32”

Win32 用户界面控件

控件	描述
按钮	按钮控件是在运行时允许用户交互的常用方式；例如，登录屏幕中的确定按钮。 Button A响应单击事件，通过添加 ŏnClick”标记值来定义。 响应点击事件，可以使用按钮，例如，发送信号，在运行时触发启动。 可定制的设计属性： ● 客户端边缘 ● 默认按钮 ● 禁用 ● 平坦的 ● 水平对齐 ● 模态框架 ● 多行 ● 右对齐文本 ● 从右到左阅读顺序 ● 静态边缘 ● 制表位 ● 透明的 ● 垂直对齐 ● 可见的 标记值： ● OnClick – 指定响应此按钮上的单击事件而执行的JavaScript命令 属性： ● 使能够 ● 节目 ● 文本 操作： ● 搬去

选择框	可定制的设计属性： • 自动 • 客户端边缘 • 禁用 • 平坦的 • 水平对齐 • 左文本 • 模态框架 • 多行 • 右对齐文本 • 从右到左阅读顺序 • 静态边缘 • 制表位 • 垂直对齐 • 可见的 标记值： • OnCheck – 指定响应此复选框值的变化而执行的JavaScript命令 属性： • 检查器 - 整数值 [0| 1] • 使能够 • 节目 • 文本
组合框	可定制的设计属性： • 自动 • 客户端边缘 • 数据 – 以分号分隔的string ，用于在运行时填充组合框；例如， 是；否；也许” • 禁用 • 有字符串 • 小写 • 模态框架 • 右对齐文本 • 从右到左阅读顺序 • 种类 • 静态边缘 • 制表位 • 类型 • 大写 • 垂直滚动 • 可见的 操作 • 添加字符串（ string ） • 删除所有 （ ）

	• DeleteItem (number) – 删除指定索引处的项目 • DeleteString (string) – 删除所有匹配string的项目 • 获取计数 () • 获取字符串（数字） • 插入项（数字， string ） • 插入字符串（数字， string ） • SetString（数字， string ） 属性： • 使能够 • 选择 - 当前选定项目的索引 • 节目
对话框	可定制的设计属性： • 绝对对齐 • 应用程序窗口 • 边框 - 仅调整大小或对话框框 • 中心 • 客户端边缘 • 中心鼠标 • 剪辑兄弟姐妹 • 禁用 • 水平滚动条 • 左滚动条 • 本地编辑 • 最大化框 • 最小化框 • 没有激活 • 重叠窗口 • 调色板窗口 • 右对齐文本 • 从右到左阅读顺序 • 设置前景 • 系统菜单 • 系统模态 • 标题栏 • 工具窗口 • 最顶层 • 透明的 • 垂直滚动条 • 可见的 • 窗边 属性： • 使能够 • 节目 • 文本

	操作： • 搬去
编辑控件/富编辑控件	可定制的设计属性 • 对齐文本 • 自动水平滚动 • 自动垂直滚动 • 边界 • 客户端边缘 • 禁用 • 控件（仅限编辑） • 模态框架 • 多行 • 数字 • 密码 • 只读 • 右对齐文本 • 从右到左阅读顺序 • 静态边缘 • 制表位 • 透明的 • 控件（仅限编辑） • 可见的 • 想要返回 属性： • 使能够 • 节目 • 文本
组框	可定制的设计属性： • 客户端边缘 • 禁用 • 平坦的 • 水平对齐 • 模态框架 • 右对齐文本 • 静态边缘 • 制表位 • 可见的 属性： • 使能够 • 节目 • 文本
列表框	可定制的设计属性：

	• 边界 • 客户端边缘 • 禁用无滚动 • 禁用 • 左滚动条 • 模态框架 • 右对齐文本 • 选择 • 种类 • 静态边缘 • 制表位 • 垂直滚动 • 可见的 操作： • 添加字符串（ string ） • 删除所有（ ） • DeleteItem (number) – 删除指定索引处的项目 • DeleteString (string) – 删除所有匹配string的项目 • 获取计数 () • 获取字符串（数字） • 插入项（数字， string ） • 插入字符串（数字， string ） • SetString（数字， string ） 属性： • 使能够 • 选择 - 当前选定项目的索引 • 节目
列表控件	可定制的设计属性： • 结盟 • 始终显示选择 • 边界 • 客户端边缘 • 禁用 • 编辑标签 • 左滚动条 • 模态框架 • 无列标题 • 无滚动 • 单选 • 种类 • 静态边缘 • 制表位 • 视图 • 可见的

	标记值： • Column - 用于初始化此 List控件的列名称和大小的string ，以分号分隔：例如，“Column1;100;Column2;150;” 操作： • 添加字符串（ string ） • 删除所有 （ ） • DeleteItem (number) – 删除指定索引处的项目 • DeleteString (string) – 删除与string匹配的所有项目 • 获取计数 () • GetString（数字，数字） • 插入项（数字， string ） • 插入字符串（数字， string ） • SetString（数字，数字， string ） 属性： • 使能够 • 选择 - 当前选定项目的索引 • 节目
图片控件	初始图片控件标记值可以使用控件图像来图像。将该值设置为模拟可访问的文件名。可以在运行时使用JavaScript中的 ChangeImageFile 方法修改图像。这需要一个要加载的文件名的string参数。 将 图像类型”属性设置为文件的正确类型（位图、增强型元文件或图标）。此设置不能在运行时修改。 可定制的设计属性： • 边界 • 中心图像 • 客户端边缘 • 颜色（框架颜色） • 禁用 • 图像类型 • 模态框架 • Real Size图像 • 静态边缘 • 制表位 • 视图 • 可见的 操作： • ChangeImageFile (string) - 文件名 属性： • 节目
进度控件	可定制的设计属性： • 边界 • 客户端边缘 • 禁用 • 选框

	• 模态框架 • 光滑的 • 静态边缘 • 制表位 • 垂直的 • 可见的 标记值： • Range – 指定此控件的最小值和最大值的string ，用分号分隔：例如，“ 1 ;100” 属性： • 使能够 • 位置 • 范围 • 节目 • 节
单选按钮	可定制的设计属性： • 自动 • 客户端边缘 • 禁用 • 平坦的 • 团体 • 水平对齐 • 左文本 • 模态框架 • 多行 • 静态边缘 • 制表位 • 垂直对齐 • 可见的 标记值： • OnChangeSelection – 指定响应单选按钮选择变化而执行的JavaScript命令 属性： • 检查器 - 整数值 [0| 1] • 使能够 • 选择 - 整数值 • 节目
滑块控件	可定制的设计属性： • 自动打勾 • 边界 • 客户端边缘 • 禁用 • 启用选择范围 • 模态框架

	• 方向 • 观点 • 静态边缘 • 制表位 • 刻度线 • 透明的 • 透明背景 • 工具提示 • 可见的 标记值： • Range – 指定此控件的最小值和最大值的string ，用分号分隔：例如，“ 1 ;100” 属性： • 使能够 • 页面大小 • 位置 • 范围 • 节目
旋转控件	可定制的设计属性： • 结盟 • 方向键 • 自动好友 • 客户端边缘 • 禁用 • 模态框架 • 没有数千 • 方向 • 设置好友整数 • 静态边缘 • 制表位 • 可见的 • 裹 标记值： • Range – 指定此控件的最小值和最大值的string ，用分号分隔：例如，“ 1 ;100” 属性： • 使能够 • 位置 • 范围 • 节目
静态文本/标签	可定制的设计属性： • 对齐文本 • 边界 • 客户端边缘

	• 禁用 • 结束省略号 • 模态框架 • 路径省略号 • 无包装 • 通知 • 路径省略号 • 右对齐文本 • 简单的 • 静态边缘 • 凹陷 • 制表位 • 可见的 • 单词省略号 属性： • 使能够 • 节目 • 文本
选项卡控件	可定制的设计属性： • 边界 • 底部 • 纽扣 • 客户端边缘 • 禁用 • 扁平按钮 • 重点 • 热门曲目 • 模型框架 • 多行 • 右对齐文本 • 静态边缘 • 风格 • 制表位 • 工具提示 • 可见的 标记值： • Tabs – 指定此控件的每个选项卡的名称的string ，以分号分隔：例如，“Tab 1 ;Tab 2;Tab 3;” 属性： • 使能够 • 节目
树控件	可定制的设计属性： • 始终显示选择

	• 边界 • 选择框 • 客户端边缘 • 禁用拖放 • 禁用 • 编辑标签 • 全行选择 • 有按钮 • 有线条 • 水平滚动 • 左滚动条 • 根线 • 模态框架 • 右对齐文本 • 从右到左阅读顺序 • 滚动 • 单 展开 • 静态边缘 • 制表位 • 工具提示 • 曲目选择 • 可见的 操作： • 删除() - 删除指定的TreeItem • InsertItem (string) - 要插入的新树项的虚线路径；将自动创建此虚线路径中尚不存在的任何父项 • InsertString (string) - 见*InsertItem* • TreeItem (string) - 要访问的树项的虚线路径；使用文本属性为该树项设置文本，或使用删除操作从树中删除该项 属性： • 使能够 • 选择 - 包含所选树项的虚线路径的string • 节目 • 文本– 获取或设置指定 TreeItem 的文本 例子： dialog.MyDialog.MyTreeControl.InsertItem("根.Parent.子"); dialog.MyDialog.MyTreeControl.TreeItem("根文本.子").Text="Modified"; dialog.MyDialog.MyTreeControl.Selection = "根.Parent"; dialog.MyDialog.MyTreeControl.TreeItem("根.Parent.Modified").删除();

Win32控件的标记值

可以在设计时通过 Win32用户接口配置文件提供的标记值自定义影响每个 Win32 对话框和控件的外观和行为的各种属性。

标记值

一些控件类型支持添加特殊标记值来修改它们的行为。

按钮、选择框和单选按钮等控件可以对 GUI 事件做出反应并执行JavaScript命令。要允许控件响应事件，请创建一个具有适当名称的新标记值；例如，“OnClick”，然后在值中键入JavaScript命令。

选项卡控件可以使用 选项卡”标记值来定义该控件在模拟时将出现的选项卡。

滑块控件、旋转控件和进度控件可以使用 范围”标记值来定义控件在模拟过程中接受的默认最小值和最大值。

标签	描述
列	**适用于**：列表控件 **使用**：初始化列表控件的列名和宽度。每列名称和宽度用分号隔开；例如，列 1；100；列 2；150；”。
点击	**适用于**：按钮 **使用**：标识为响应 Button 控件上的单击事件而执行的JavaScript命令。
安检	**适用于**：选择框 **使用**：标识为响应选择框控件值的变化而执行的JavaScript命令。
OnChangeSelection	**适用于**：单选按钮 **使用**：标识为响应单选按钮控件值的变化而执行的JavaScript命令。
范围	**适用于**： Slider控件、Spin控件、Progress控件 **使用**：指定控件的默认最小值和最大值，用分号分隔：例如，“ 1 ;100”。
标签	**适用于**：选项卡控件 **使用**指定要为选项卡创建的每个选项卡的名称，以控件分隔：例如，“Tab 1 ;Tab 2;Tab 3;”。

BPMN仿真

BPMN 模拟是一种可视化和验证 BPMN业务流程流程图行为的方法。通过所有当前正在执行的活动以及接下来可以执行的可能活动的可视化指示，您将能够轻松地识别和解决您已建模的流程的潜在问题。

模拟仿真模型类似于模拟标准UML行为模型，除了BPMN：

- 使用一些不同的元素类型（例如网关而不是决策）和
- 对放置在与属性和元素相关联的 标记值”字段中的脚本进行操作，而不是在 “”字段中（如果您愿意，而不是在 执行分析器编译”对话框中）;脚本是用JavaScript的

与BPMN仿真合作

活动	细节
创建BPMN仿真模型	当您创建适合模拟的 BPMN模型时，您会考虑如何表示起点、流程和要测试的条件。
将UML活动与 BPMN 流程进行比较	BPMN业务流程模型的模拟与UML活动图的模拟有许多不同之处。

注记

- BPMN 模拟在Enterprise Architect的统一版和终极版中可用

创建BPMN仿真模型

作为开发仿真模型过程的一部分，请考虑您更喜欢应用执行仿真的三个选项中的哪一个：

- 执行一个模拟脚本来初始化图表的变量——选择'BPMN'作为平台，执行模拟'作为脚本'并选择脚本；然后，您可以在开始模拟之前或在模拟过程中，在元素和连接器的标记值内定义条件和决策作为JavaScript声明
- 不要使用脚本，而是在第一个活动中初始化变量，然后再次修改元素和连接器的标记值中的条件和决策，然后以 "解释"执行模拟；然后您可以在模拟期间重新初始化变量以及条件
- 以 "手动"方式执行模拟，并在每一步手动管理流程和条件

创建适合模拟的 BPMN 图

节	行动
1	从 BPMN 2.0 技术创建业务流程或 BPEL 图。如果您创建 BPEL 图， Enterprise Architect会显示专门的对话框来简化兼容模型的创建。
2	我们建议您创建一个开始事件以清楚地显示您的模拟从哪里开始。事件类型有多种选择；该选择不会影响您的模型的模拟。如果没有定义任何开始事件，则模拟将从没有传入序列流的活动开始。
3	添加正在建模的进程中涉及的所有活动。任务类型有多种选择；该选择不会影响您的模型的模拟。通过指定 Sub-Process 的活动类型并选择 Embedded 或 CallActivity 可以进一步分解活动的行为。还支持标准循环。
4	在您的活动之间添加序列流。在 "BPEL属性"对话框中，您可以输入在遵循序列流之前必须满足的条件 (True)。您还可以将 conditionType 设置为 "Default"，以确保如果所有其他分支都未能满足指定的条件，则将采用此流程。 如果您不使用 BPEL 图，则使用 conditionExpression 和 conditionType标记值。
5	为将导致进程或活动执行路径结束的任何条件添加事件。事件类型有多种选择；其中只有终止类型会影响执行。在具有多个活动节点的模拟中，它会导致整个进程终止，而不仅仅是到达该节点的线程。

注记

- 要包括正在模拟的包之外的包中的活动，请绘制：
- 从包含包包导入连接器
被模拟到每个外部包，或
- 包含图表的包中的依赖连接器
模拟到外部包中的每个活动

初始化变量和条件

对于 BPMN 模拟模型，您可以在执行分析器脚本中初始化变量。您还可以在流程的第一个活动元素的标记值中初始化这些变量，这使您在模拟进行时可以更灵活地添加和更改变量。同样，您可以在序列连接器的标记值中定义适用于流程中各个决策点（网关）的条件和值。

如果你想在你的模拟过程中加入一个用户界面，使用 Win32，你再次使用标记值来识别对话框或提示来显示，在活动元素中，就在处理值或决定的点之前。

对于UML图的模拟，“sim” object和 “this” object内的变量显示在 “局部变量”窗口中。

访问

使用此处概述的方法之一显示属性窗口的 “标签”选项卡。

功能区	探索>门户>窗口>属性>属性>标签
键盘快捷键	Ctrl+2 >属性窗口的 “标签”选项卡

初始化变量

1. 在图表上，单击过程中的第一个活动元素。
2. 在属性窗口的 “Tags”选项卡中，点击脚本 “Value”字段的下拉箭头，选择 “脚本”。
3. 在脚本 “值”字段中，输入适当的JavaScript代码，例如：

   sim.loan=真； sim.status="未定义';

定义条件

1. 在图表上，单击从网关元素发出的序列流连接器。
2. 在属性窗口的 “标签”选项卡中，单击条件类型 “值”字段的下拉箭头，然后选择 “表达式”。
3. 在conditionExpression'Value'字段（<memo>*）点击 … 按钮，显示标记值注记窗口。类型在相应的JavaScript代码中，例如：
   sim.status=="保持"
4. 点击确定按钮。语句文本显示为连接器的标签。

整合Win32用户接口

1. 在图表上，单击代表做出决定的活动元素。
2. 在属性窗口的 “标签”选项卡中，点击 “任务类型”值”字段的下拉箭头，选择 “脚本”。
3. 在 “脚本值”字段中，输入适当的JavaScript代码，例如：
   对话框.屏幕1.显示=真；
   （此语句显示对话框 Screen1。您可以通过将 'Show' 更改为False来临时隐藏对话框。）

UML活动和 BPMN 流程的比较

BPMN模型的执行和模拟与UML活动图的执行和模拟有许多不同之处。这里介绍了相似概念的映射，以及表达系统行为的两种方法之间的差异。

UML活动和 BPMN 流程的比较

UML活动	BPMN业务流程
起点由 Initial节点定义。无法指定活动开始的原因。	起点由开始事件定义。这意味着活动开始的特定原因，尽管它可能是未指定的。
一个活动的基本行为单元是行动元素。 UML提供了许多不同形式的行动，尽管模拟使用了其中的一小部分。	活动活动元素A许多不同的任务类型可用。这些通常描述不同的执行方法（例如手动），而不是发生的情况。
控件用于连接活动图表中A元素。 A区别特征是，除了显式分叉节点外，任何节点都只能遵循单个控件控件。要在 Flow控件上限制流量，请添加一个守卫条件。	A用于连接序列业务流程中的元素。这些与UML活动图的不同之处在于默认采用所有有效的序列流。要限制序列流上的流，请将条件标记值设置为 表达式”，并在条件表达式标记标记值中创建脚本。
决策节点A显式地对正在做出的决策进行模型。当潜在流重新组合为一个时，将使用使用相同语法 A Merge 节点。	当必须选择单个路径时，使用设置为 独占” A网关节点。它还用于再次组合潜在流。可以A方向指定为 会聚”或 发散”以 确选择两种模式。
A分叉节点用于并发执行多个节点，而一个汇合节点，使用相同的语法，用于等待所有传入的流变为可用，并以单个流离开。	设置为 并行” A网关节点用于显式模型多个节点的并发执行。它还用于等待所有传入流变为可用并以单个流离开。可以A方向指定为 会聚”或 发散”以确选择两种模式。
不允许同时执行来自一个节点的UML活动的一些输出。如果您需要这个，您可以在以后添加带有适当 Guard 的控件。	设置为 Inclusive A Gateway 节点用于显式模型，即所有条件为 true 的传出流同时执行的情况。
A需要通过引用外部活动来进一步分解行动时，使用调用行为动作。	活动元素在需要通过引用外部活动进一步分解行为时设置为CallActivity子流程。
活动行动调用行动行为	活动元素被设置为嵌入式子流程，当行为需要进一步分解而不参考外部活动时。

BPSim业务模拟

开放的 BPSim 规范提供了一组丰富的材料，涉及如何为活动或任务配置和分配资源、如何引发事件、决策制定和其他现实世界的能力。一旦根据 BPSim 规范进行配置，业务流程模型（在 BPMN 中构建）可以传递给合适的 BPSim 模拟引擎，并使用运行信息中附加的配置数据，按照 BPMN模型中定义的流程运行。

BPSim 规范非常详细，为感兴趣的建模师和业务战略家提供了前所未有的灵活性，可以将操作信息分配给模型，然后根据从仿真引擎返回的信息评估解决方案的质量。本节详细介绍了配置模型以执行细节时可用的各种屏幕和选项。

Sparx Systems提供支持 BPSim 的模拟器 - **BPSim 执行引擎。**这个插件

与Enterprise Architect中定义的运行和 BPMN 模型集成，提供运行和存储多个模拟结果的能力，并在每个配置的结果集之间进行方便的比较。

BPSim 执行引擎是访问和使用 BPSim 配置功能的先决条件。执行引擎集成了Enterprise Architect的统一和终极版本；在企业版中使用，可以单独购买和安装。

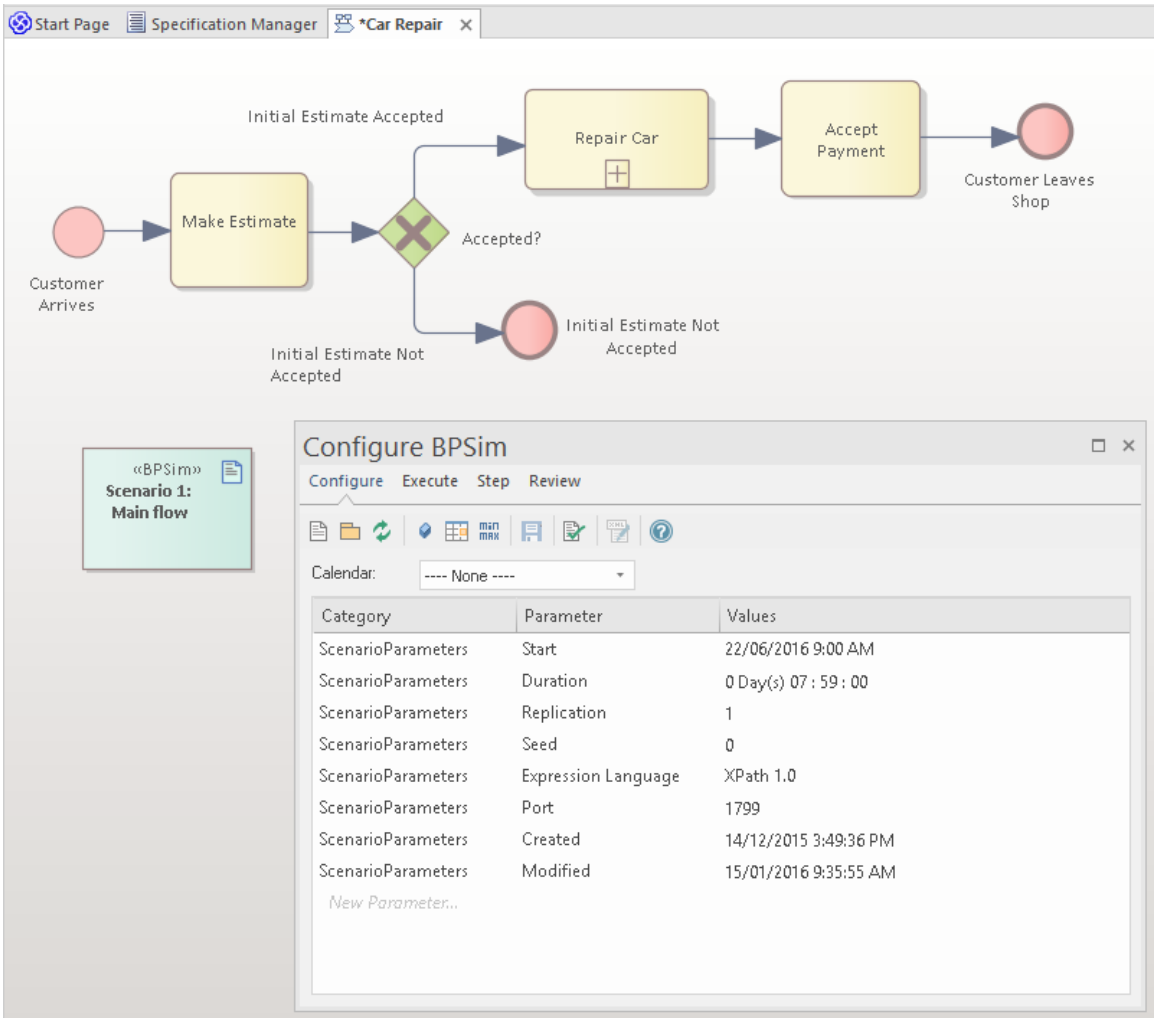
设置好 BPSim 配置后，模拟执行过程会以标准形式导出 BPMN模型及其 BPSim 数据。这确保了对模型的更改始终合并到模拟中。类似地，模型导出过程以可以导入另一个模型并由Sparx Systems BPSim 执行引擎或任何其他符合标准的 BPSim 引擎使用的形式捕获 BPMN模型及其 BPSim 数据。

安装 BPSim

虽然 BP 与Enterprise Architect的统一和终极模拟版集成，但它与企业版分开，并且 - 购买后 - 必须安装在您的系统上。

对于所有三个版本，您必须确保在您的系统上也安装了正确版本的Java Runtime环境(JRE) 和Java Development Kit (JDK)。

BPMN模型与BPMN仿真



配置 BPSim窗口可帮助您定义多个类别的仿真参数，每个类别侧重于仿真配置的一个方面。例如，您将定义：

- ScenarioParameters，定义仿真本身应该如何进行
- 控件参数，它检查活动如何在业务流程中流动，由控件事件的可能性和某些事件的优先级序列
- 时间（时间）参数，检查活动处理中一个或多个阶段的持续时间如何影响业务流程
- 资源参数，检查工人和其他资源的类型和角色的参与、所需数量、成本和可用性

您可以维护多个版本的配置（单独的工件配置也可以维护多个版本）并轻松比较每个配置的差异，以了解如何改变仿真或流程执行的流程例如，您可以建立一个基线配置，然后创建多个 "假设"？改变一个或多个参数的配置。通过仿真引擎运行这些配置后，您可以检查每个结果并确定每个配置的相对优点。此处应用的一个有用原则是一个配置中常见的、未更改的数据由仅包含变化数据的另一种配置简单继承 - 因此，您可以在当前变量集上运行模拟，该模拟利用了标准数据配置同时。

用户可以结合图表和 Charting功能快速改变、模拟和比较业务流程模型的各个方面，并以多种图表格式之一显示模拟之间的差异。

如果您正在跨多个项目工作，您可以在它们之间导出和导入 BPSim 配置。该配置自动携带它所基于的 BPMN 2.0模型。

Enterprise Architect业务流程仿真仿真配置工具基于工作流管理联盟（WfMC）开发的BPSim框架。

注记

- 如果单击图表或浏览器窗口中的业务流程元素或连接器，它将在配置 BPSim窗口中突出显示并选中
- 您模拟的业务流程可以包含多个包中的元素；要在模拟中包含外部元素，您必须创建一个包含 "父"包和包含

外部元素的 外部”包或外部元素本身的包图；创建一个：
- 将包连接器从父包导入到每个外部包，或者
- 从父包到每个外部元素的依赖连接器

安装 BPSim

虽然 BP 与Enterprise Architect的统一和终极模拟集成，但它必须安装在您的系统上。对于这些版本，您必须确保您的系统上也安装了正确版本的Java运行时环境(JRE) 和Java开发工具包 (JDK)。

安装 JDE 和 JDK

要使用Sparx Systems BPSim 执行引擎，您的系统上必须具有Java运行时环境(JRE) 版本1 .7 或更高版本，并且如果您的配置包含任何属性参数，您还必须具有Java开发工具包 (JDK) 版本1 .7 或更高。

除非您的系统上有多个版本的 JRE/JDK，并且您想指定执行引擎应该使用哪个版本，否则您不需要对引擎进行任何进一步的配置。在这种情况下，应用这些环境变量，如下所示：

1. 单击窗口 开始”图标并选择 电脑”选项。
2. 从横幅菜单中，选择 系统属性”选项。
3. 从侧面板中，选择 高级系统设置”选项。
4. 在 系统”属性的 高级”选项卡上，单击 环境变量”按钮。
5. 在 环境变量”对话框的 系统变量”面板中，单击 新建”按钮。
6. 在 系统”对话框中，完成具有显示值的字段：

   对于 JRE：变量名称：MDG_BPSIM_JRE_HOME
   变量值：C:\Program Files\ Java \jre7

   对于 JDK：变量名：MDG_BPSIM_JDK_HOME
   变量值：C:\Program Files\ Java \jdk1.7.0_51
7. 点击确定按钮。
8. 您必须重新启动计算机才能使新变量生效。

配置

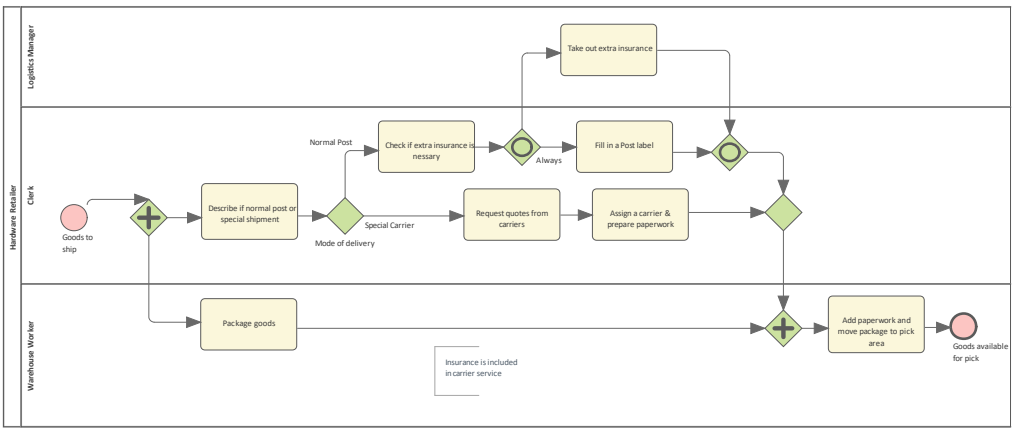
业务流程仿真模型中A元素流程工件（业务流程仿真）中的代表，您可以在与您正在使用的BPMN 相同的项目中的任何包中创建图表。

创建业务流程模型

每个 BPSim 配置都是专门为现有业务流程创建的，并根据 BPMN 中定义的现有业务流程创建。因此，在您使用业务流程仿真管理模型之前，您需要创建或导入该配置所基于的工件模型。

可以在 EAExample模型中找到并处理此示例图，位于：

分析与业务建模> BPMN流程2.0示例> 硬件零售商的进程图表>进程



打造业务流程仿真工件

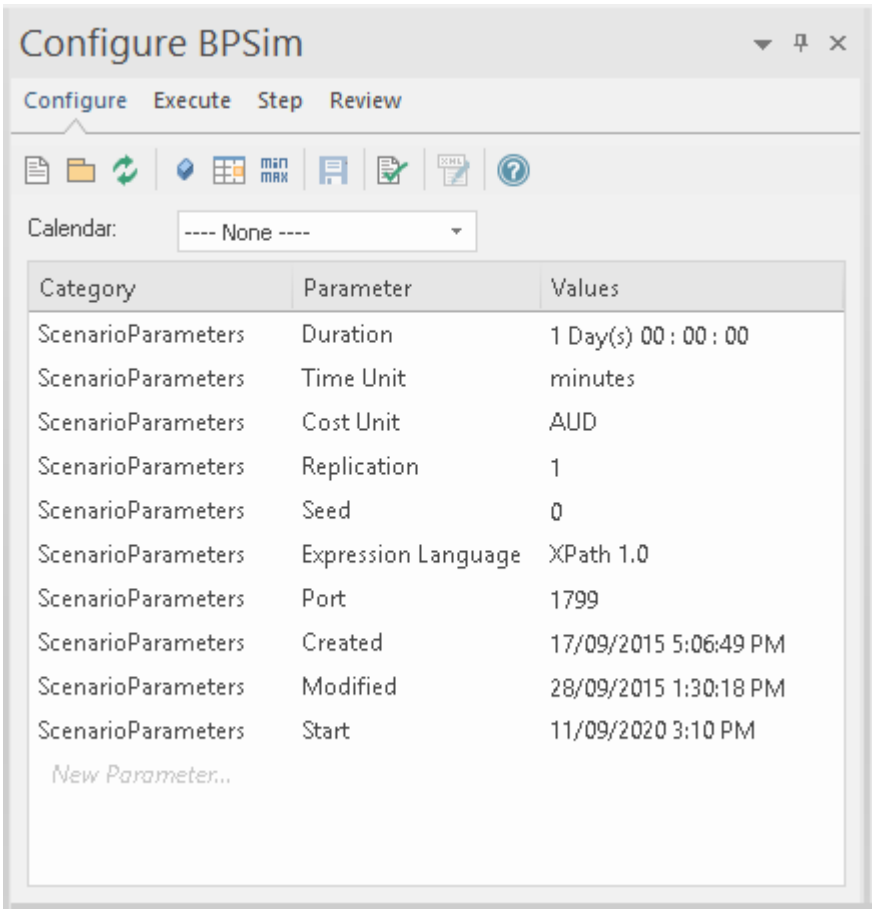
打开图表，在其中创建一个箱子，然后显示图表工具箱工件。展开常用的 仿真”页面，将 业务流程仿真”图标拖到图表上。

当您设置工件时，请考虑是否可以创建一个设备来定义其他工件以定义基本配置的某些方面或其他工件。您将在 变体”工件之间使用概括器和连接器，这样变体将继承您在 基础”工件中定义的数据。这样，您不必在每次工件时都重新定义整个配置。

双击元素并给它一个适当的名称，例如（例如）'Base配置'。

配置 BPSim窗口概览

右键单击设备元素（在图表或浏览器窗口中）并选择 配置 BPSim工件选项。配置 BPSim的窗口工件。



此窗口包含四个选项卡：配置、执行、节和审阅。

- 配置：为每个BPMN元素配置配置参数；定义属性参数、日历和场景参数
- 执行：使用 BPSim 配置执行 BPMN模型
- 节：step over/step in 提供对执行过程的洞察，包括令牌状态、属性值和每个时间/步骤的资源分配
- 审阅/审阅工件（s），生成标准的自定义模拟结果报告

配置- 配置页面

BP工件包将被配置为将加载此包或其子包下的所有 BPMN 元素。默认情况下，包含此工件的包会在加载到此窗口时进行配置。

这个窗口是上下文的。当在图表或浏览器窗口中选择一个元素时，列表将显示该元素的当前配置；此外，值的下拉列表将仅显示元素的可用参数。

当工件是上下文的元素时，列表会显示 ScenarioParameters。

访问

功能区	仿真>流程分析>进程> 打开BPSIM Manager >配置

工具栏选项

选项	描述
	单击此按钮可选择或创建一个 BPSimConfiguration元素。
	单击此按钮为工件按钮设置一个包。本包或其子包下的所有 BPMN 元素都将包括在内。
	单击此按钮可从已配置的包中重新加载 BPMN 元素。例如，当修改了一些 BPMN 元素时，运行此命令重新加载包，以便运行仿真将更改考虑在内。
	单击此按钮可定义属性，该属性可用作 BPMN 元素的属性参数。
	单击此按钮可定义日历，可用于配置元素参数。
	单击此按钮可显示或隐藏 结果请求”列。自定义模拟需要结果请求配置。执行报告将仅包含请求的结果。
	单击此按钮可将配置信息保存到配置 BPSim工件元素。
	单击此按钮以验证 BPMN模型和 BPSim 配置。如果生成错误或警告消息，它们可能会显示在系统输出窗口中。
	单击此按钮以导出具有 BPSim 配置的 BPMN模型。此导出的 BPMN 文件符合 BPMN 和 BPSim 规范，可供第三方 BPSim 执行引擎使用。

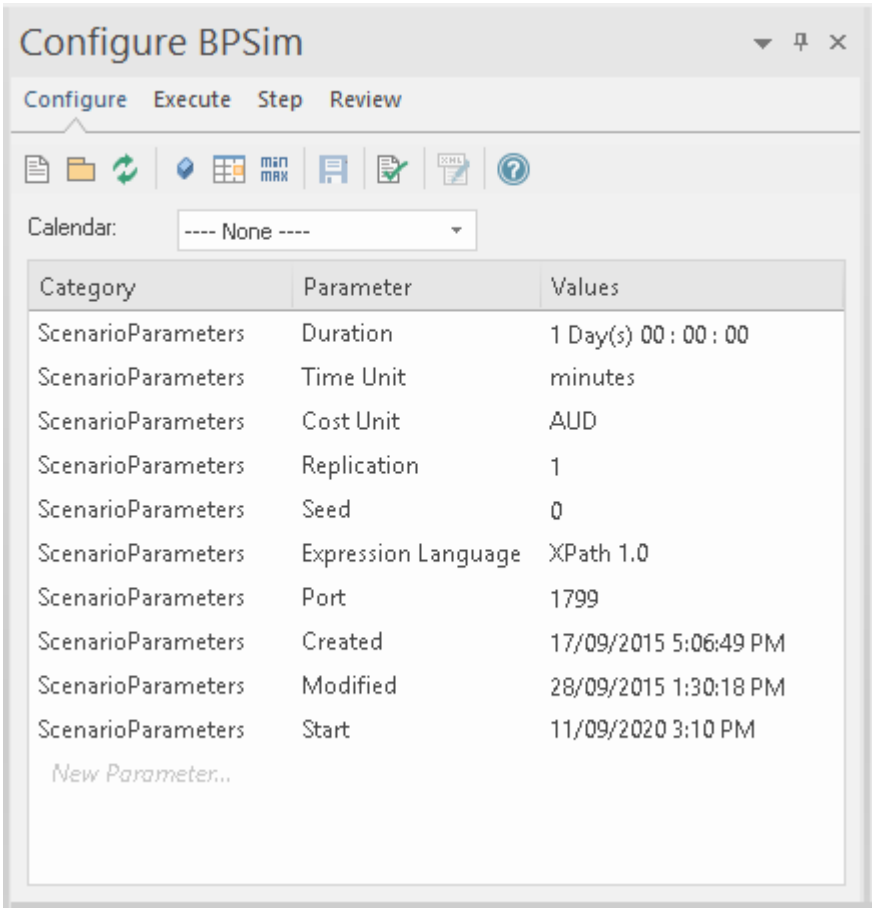
场景参数

A场景是由一组元素参数组成的。场景本身将所有元素使用的参数定义为全局设置。并非元素的所有参数都会

显示，但您可以通过以下方式将它们添加到列表中：

1. 点击新参数 文本，单击下拉箭头并选择 场景参数”。
2. 单击 参数”字段中的下拉箭头，然后从列表中选择参数类型。

注记一旦您为场景添加了所有可能的参数，配置 BPSim窗口将不允许您尝试添加更多参数。



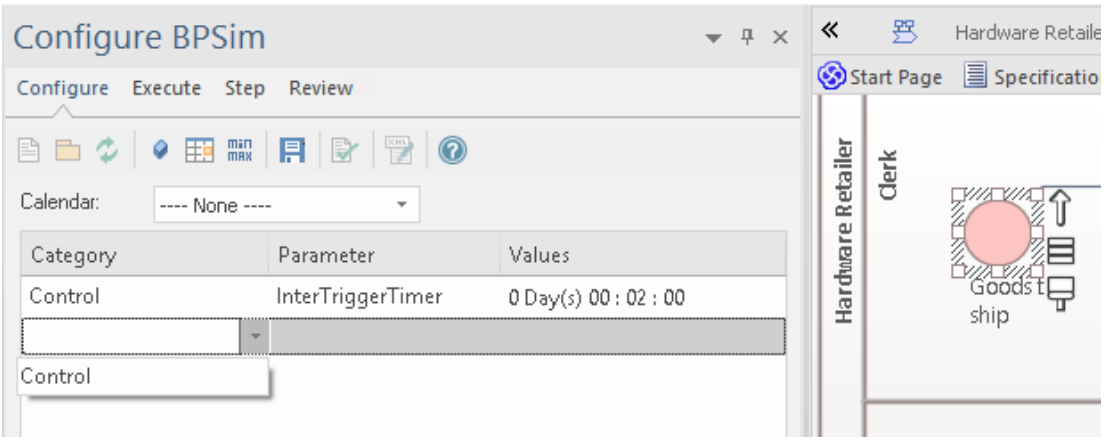
名称	描述
开始	流程开始生效的日期和时间。 您可以通过改写值来编辑它，或者对于日期，通过从下拉日历中进行选择来编辑它。
期间	该过程花费的时间长度。 “持续时间”参数是必需值。它必须足够长以容纳完成模拟；例如，如果一个过程（以及它的模拟）需要三个小时才能完成，则 持续时间”参数必须设置为大于三个小时的值。 您可以通过以 天时：分：秒”格式改写适当的段来编辑它。
时间单位	在此方案中表示时间段的基本单位。除非在本地被覆盖，否则所有表示时间的数值和浮点值都应视为以该单位表示。 您可以通过单击下拉箭头并选择单位来编辑它。
成本单位	流程中记录的任何成本的货币单位。除非在本地被覆盖，否则所有表示成本的数字和浮点值都应视为以该货币代码表示。 您可以通过单击下拉箭头并选择单位缩写来编辑它。
复制	要执行的场景的复制次数。默认为1 。

	您可以通过简单地输入一个值来编辑它。
种子	用于初始化伪随机数生成器A随机种子。 您可以通过简单地输入一个值来编辑它。
表达语言	XPath 1 .0 和Java - XPath 1 .0 是默认语言。如果将Java指定为表达式语言，则必须设置 JDK Home。 您可以通过单击下拉箭头并选择语言来编辑它。
DMN模块	在 BPMN模型中使用业务规则任务时，您可以将这些任务作为 DMN模型来实现。 您可以先创建一个 DMN模型并在Java中生成一个 DMN服务器，然后单击 ... 按钮指定生成的 DMN服务器文件。
JRE主页	Enterprise Architect BPSim 执行引擎在Java环境中运行，因此必须指定 JRE Home。点击 ... 按钮选择一个目录；例如，C:\Program Files\ Java \jre7。 您可以通过再次单击 ... 按钮来浏览目录来编辑它。
JDK 主页	当表达式语言为Java时， Enterprise Architect BPSim 执行引擎将生成Java代码并使用 javac 作为供应商扩展进行编译。所以必须指定一个 JDK Home。使用 ... 按钮选择一个目录（例如 C:\Program Files\ Java \使用）。 您可以通过再次单击 ... 按钮来浏览目录来编辑它。
端口	Enterprise Architect用来与端口执行引擎通信的端口号。默认端口号为 1799。
已创建	只读字段。工件创建时的时间戳。
修改的	只读字段。工件上次修改时间的时间戳。

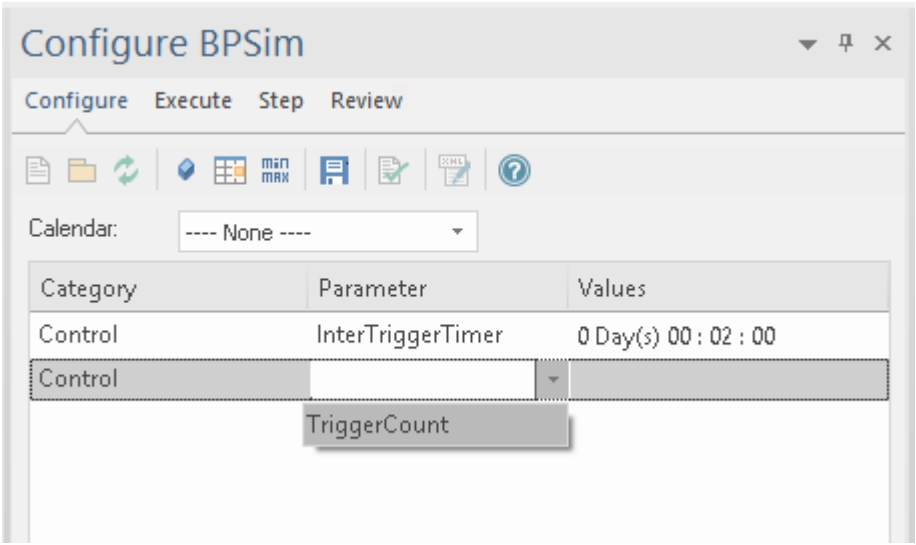
控件参数

要开始为适当的元素控件（例如事件或网关）定义参数：

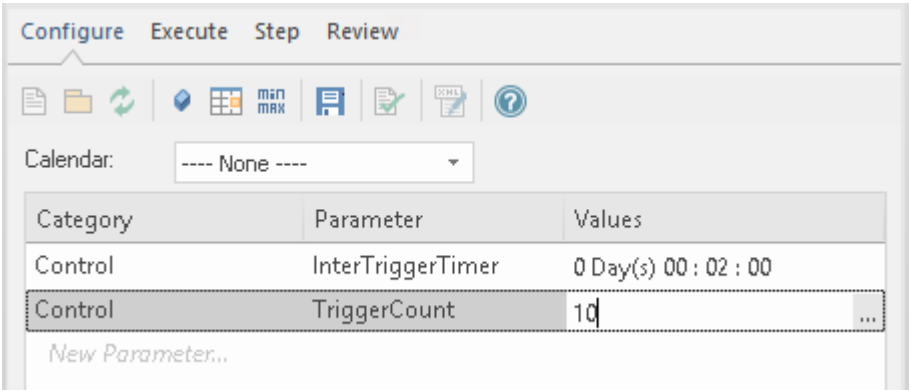
1. 选择图表上的元素，然后单击新参数 文本并在 类别”列中的下拉箭头上，然后选择 控件”。



2. 单击 参数”字段中的下拉箭头，它将显示所选元素的可用未分配参数。



3. 选择适当的参数，然后单击 值"字段；您可以在字段中键入参数值或使用 ... 按钮打开 参数值"对话框。

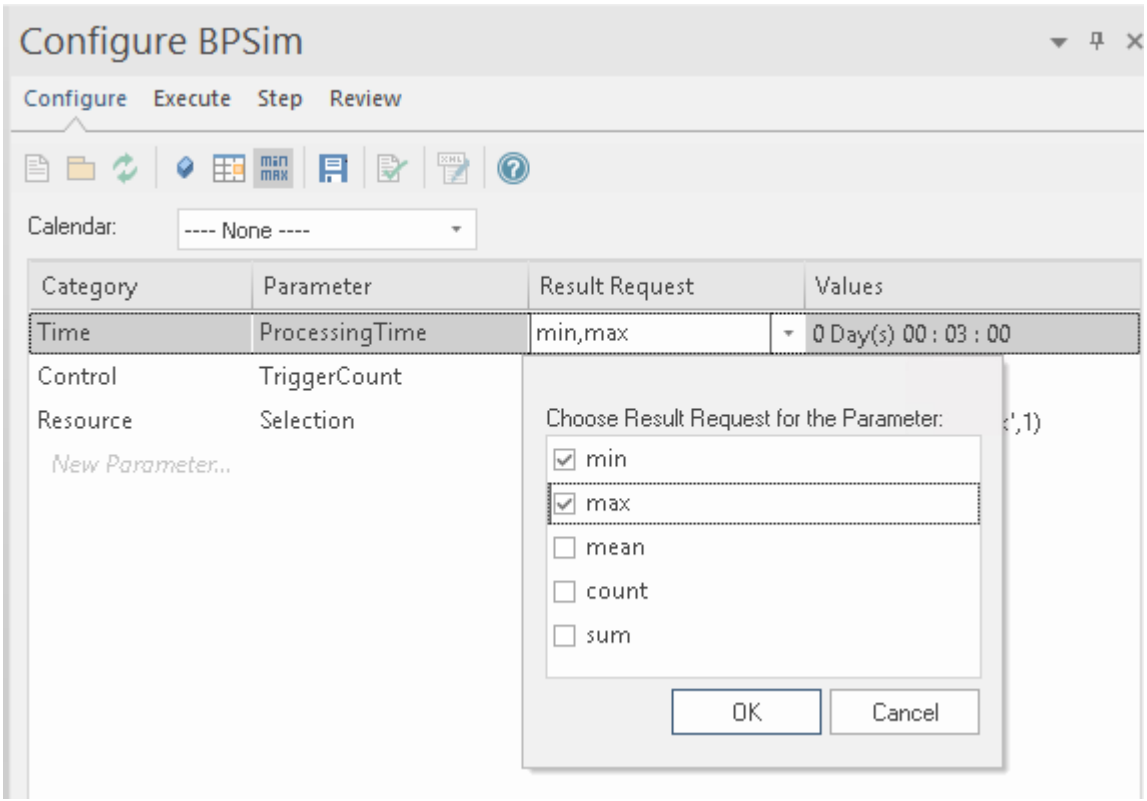


注记该页面只允许您为元素提供适当的参数。一旦您指定了这些参数，这些字段就不允许进一步输入或选择。

时间参数

要开始为适当的元素（例如 BPMN任务）定义时间参数：

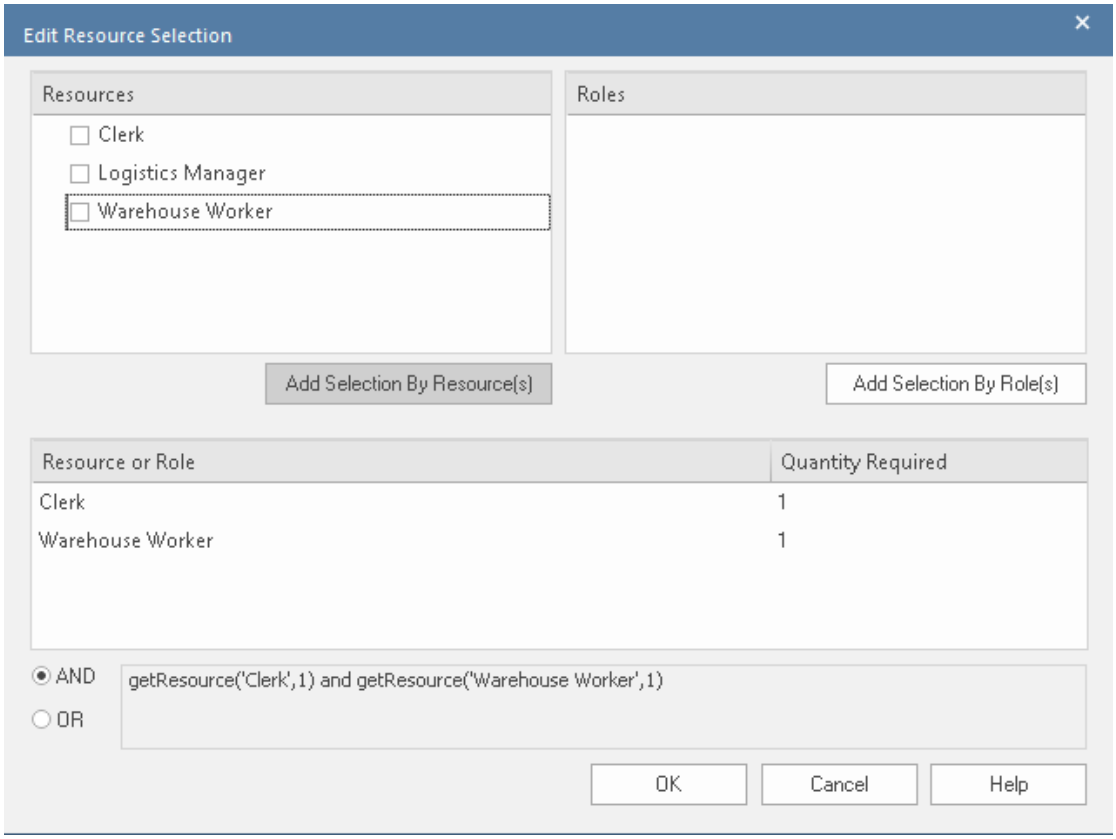
1. 选择图表上的元素。
2. 单击新参数文本和下拉箭头，然后从列表中选择 时间"。
3. 选择 时间"后，单击 参数"字段中的下拉箭头，然后从元素的可用参数中进行选择。
4. 在 值"字段中，输入值或单击 ... 按钮打开 参数值"对话框。
5. 您可以通过单击工具栏上的最小/最大按钮来切换 结果请求"列，以通过要求某些结果来自定义模拟输出



资源参数

要开始为适当的元素（例如 BPMN任务）定义资源参数：

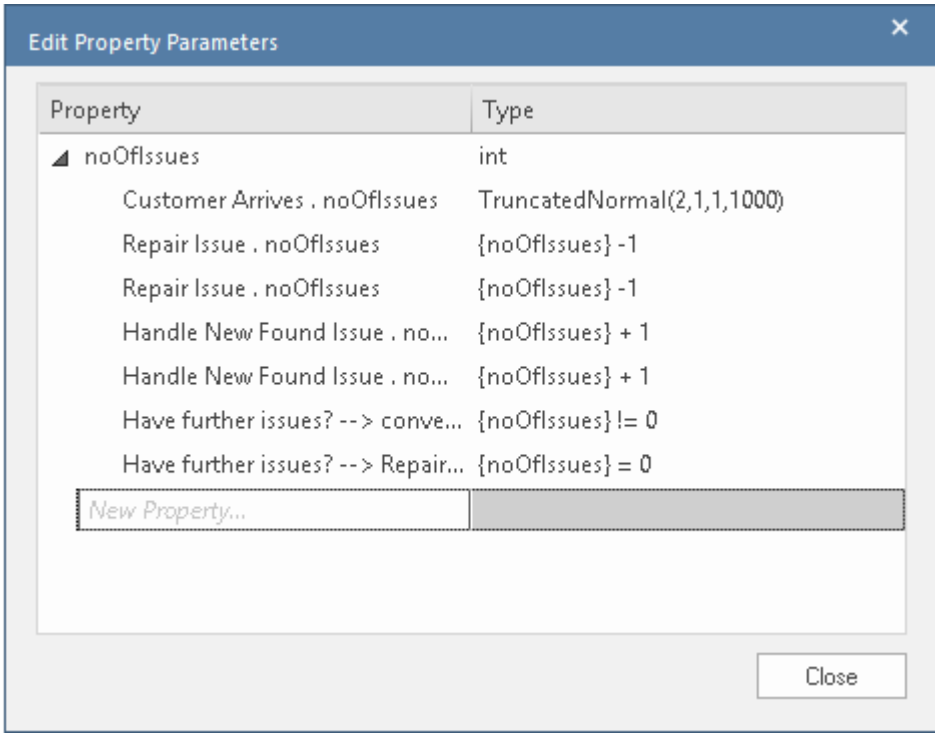
1. 选择图表上的元素。
2. 单击新参数文本和下拉箭头，然后从列表中选择 资源”。
3. 在 参数”字段中，单击下拉箭头并单击列表中的 选择”。
4. 在 值”字段中，单击 ... 按钮以打开 编辑资源选择”对话框。



- 左上面板列出了已定义的 Resource 元素；单击要分配的资源，然后单击 按资源添加选择"按钮将选择移动到 资源或角色"面板
- 右上角的面板列出了为资源元素定义的角色（如果有的话）；单击所需的角色，然后单击 按角色添加选择"按钮将选择移动到 资源或角色"面板
- 每个资源/角色的 需要数量"列默认为1 ；如果需要更大的数量，请用适当的数字改写此值
- 单击相应的单选按钮以将选择的逻辑关系设置为 AND 或 OR
- 资源选择的最终表达式组成并显示在文本字段中
- 单击确定按钮返回到配置 BPSim确定，其中表达式显示在 值"字段中

属性Parameters

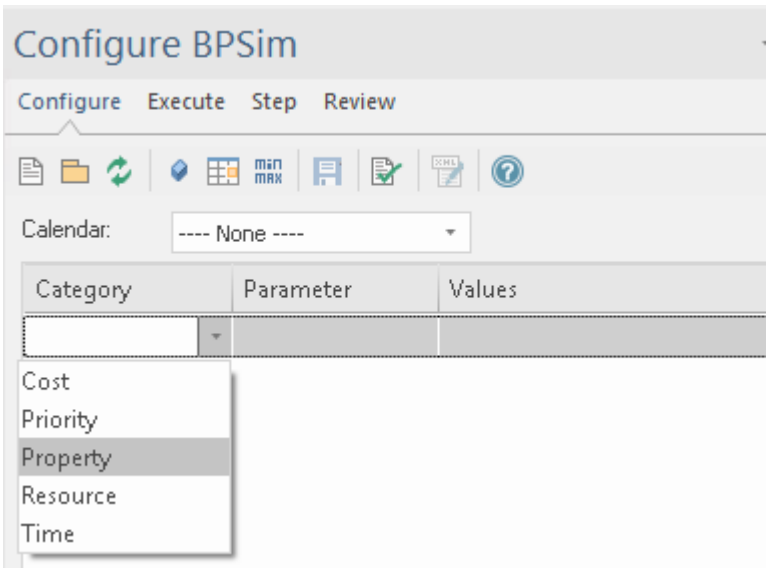
要开始定义属性参数，请单击工具栏上的 按钮。将显示 编辑属性参数"对话框。



列出了定义的属性及其引用。

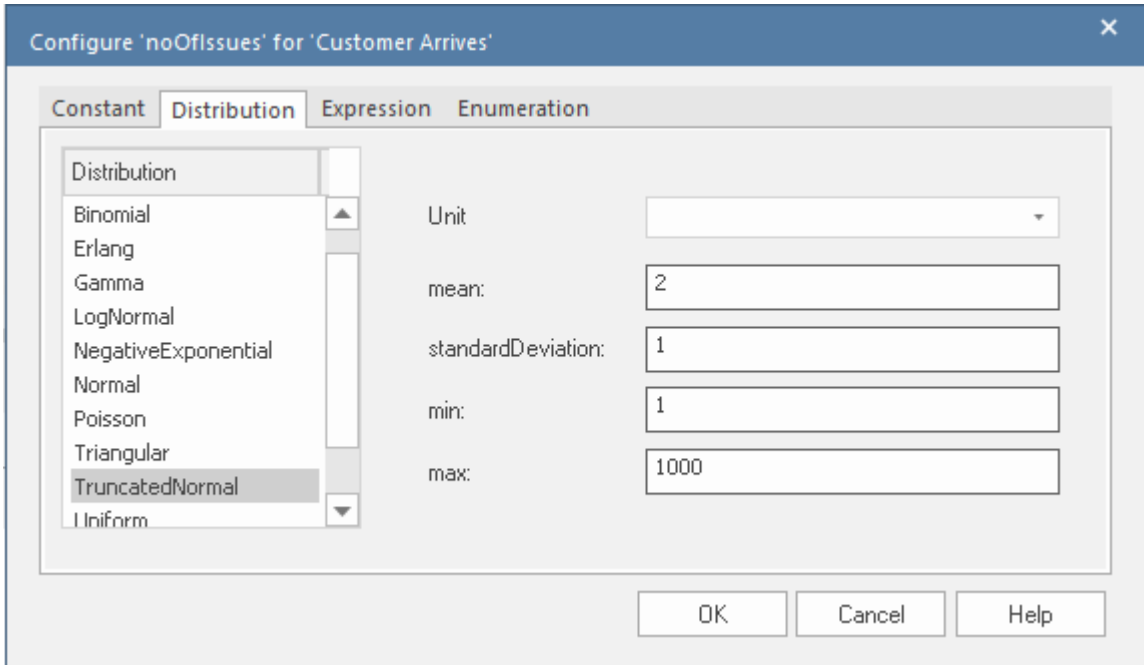
您可以添加新属性、删除选定属性（使用上下文菜单选项）、改写属性名称或为属性选择不同类型。

检查定义的属性后，您可以在 BPMN 元素上设置属性参数。



选择 属性”作为类别，然后单击 参数”字段中的下拉箭头并选择一个属性。

单击 值”列上的 ... 按钮以显示参数值对话框（从属性和父元素命名）。



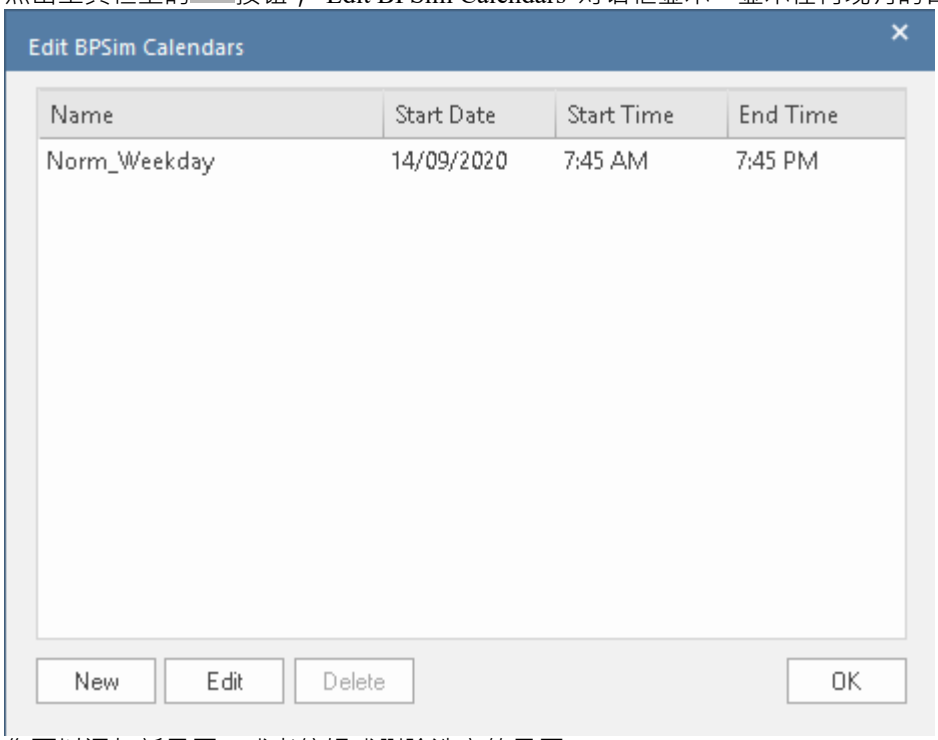
导航到相应的选项卡以选择并定义值和实际值的类型，然后单击确定按钮。该值显示在 值"字段中。

日历

日历可帮助您定义可能影响流程的任意数量的特殊时间段，例如工作日、轮班、假期或定期事件（例如盘点、盘点或审计）。

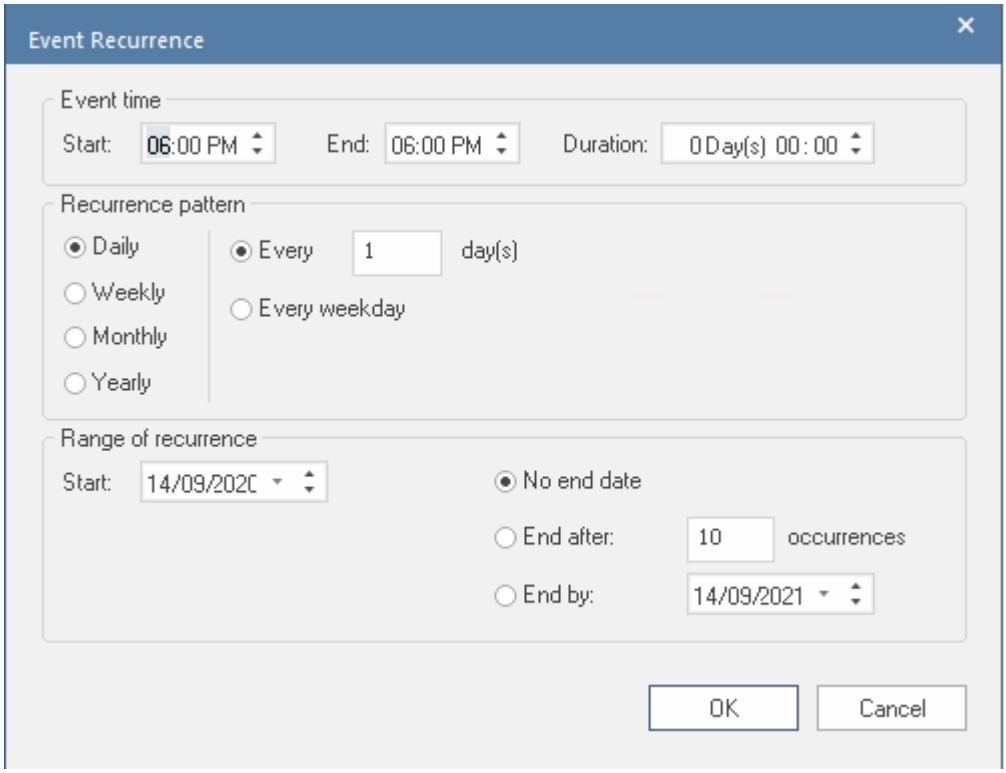
要开始定义日历：

1. 点击工具栏上的按钮； 'Edit BPSim Calendars' 对话框显示，显示任何现有的日历。



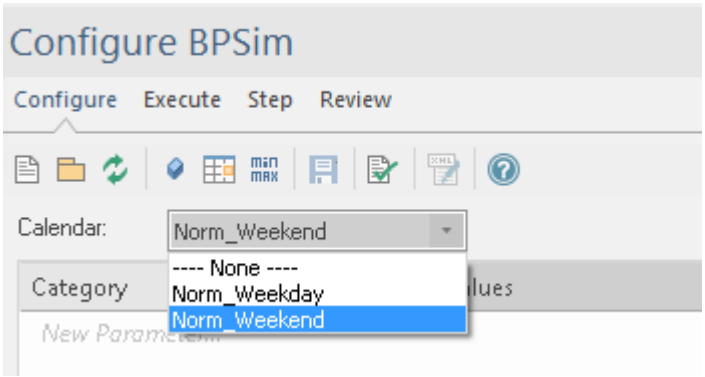
您可以添加新日历，或者编辑或删除选定的日历。

2. 要添加新的日历周期，请单击 新建"按钮以显示 事件重复"对话框。



3. 在 事件时间"面板中， 开始"和 结束"字段均默认为当前时间。 开始"字段是锚点；参考 开始"字段，对 结束"字段或 持续时间"字段的更改会自动更新另一个字段。分别单击每个字段的 时和分钟段（对于 持续时间"字段， 天"段），然后使用 旋转"箭头设置开始时间和结束时间或持续时间时期。
4. 在 重复模式"面板中，选择日历周期重复间隔的单选按钮。每个选项在面板右侧显示一组适当的字段，用于将该时间间隔细化为每天/每周/每月或每两/三/四天/每周/每月、一周中的特定日期或日期或日期一年中的月份、日期或日期。根据需要选择下拉列表中的复选框或值。
5. 在 重复范围"面板中，选择日历周期生效的日期，然后选择适当的单选按钮来定义周期何时停止适用 - 从不、在设定的出现次数之后或在特定日期。您可以从下拉日历中选择结束日期，也可以使用日期每个部分上的 旋转"箭头。
6. 点击确定按钮设置日历周期。

当您定义日历期间时，它们会按照开始日期和/或时间的顺序列出，最早在前。



使用已定义的日历，您可以在选定的日历上配置参数。

验证

为某些 BPMN 元素配置 BPSim 参数后，单击 按钮以运行模拟验证。任何 BPMN 或 BPSim 错误/警告都将显示在系统输出窗口中。根据消息修复问题。

完成此操作后，继续下一个帮助主题：帮助*Execute Page* 。

BPSim - 执行页面

定义配置后，您可以选择进行标准模拟或自定义模拟。执行将生成结果报告和用于重放（逐步执行）模拟的记录列表。

访问

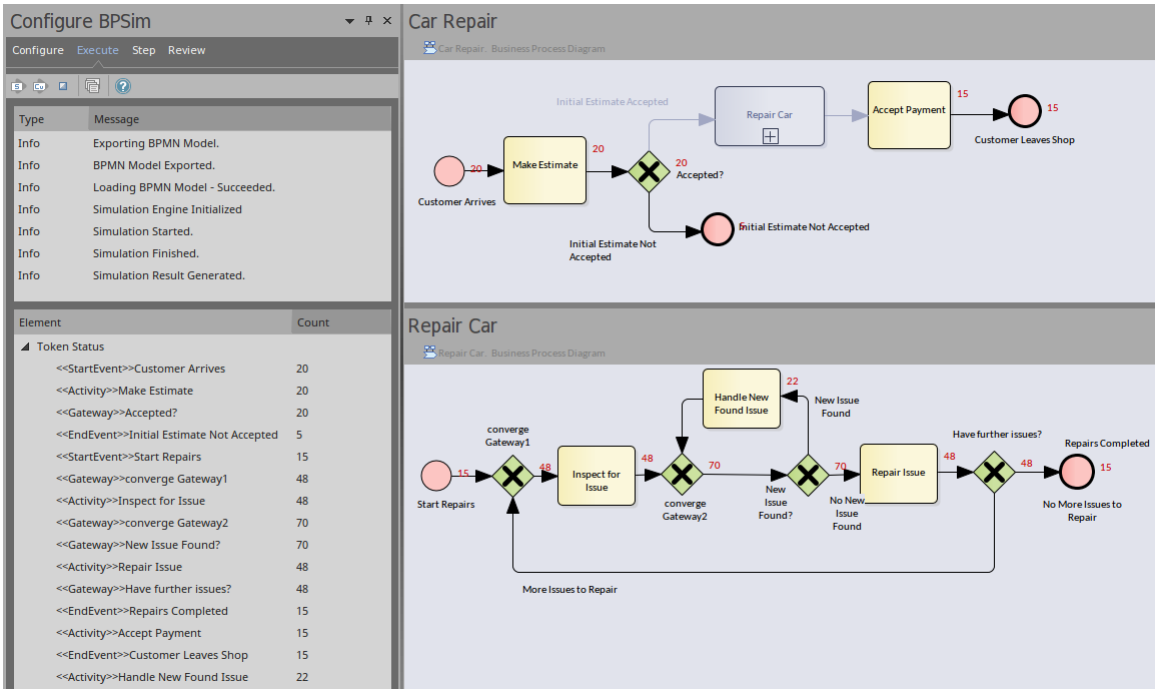
功能区	仿真>过程分析>进程>打开BPSIM Manager>执行页面

工具栏选项

选项	描述
	单击此按钮可使用 BPSim 配置执行 BPMN模型并生成标准报告。
	单击此按钮可使用模型配置执行 BPMN，并根据 "配置"页面上的 "结果请求"设置生成自定义报告。
	单击此按钮可停止执行并退出模拟。
	单击此按钮在 "审阅"页面中打开生成的报告。

执行

当您点击运行仿真按钮或运行自定义仿真按钮时，带有运行配置的BPMN模型将被导出并加载到执行引擎中。



在模拟过程中：

- 令牌状态列表将闪烁运行时值
- 该图将与运行时令牌计数一起闪烁

但是，模拟可能执行得太快而无法看到这一点。如果您使用 节”页面运行模拟，您可以看到这些变化。

在此示例中，流程 汽车修理”和子流程 修理汽车”下的 BPMN 元素在新客户定期到达时被触发。

BPSim -节页

执行成功后，系统会生成一个执行报告，告诉你一般的流程状态，比如（以修车为例）一个任务的平均时间，客户的总等待时间，修复了多少问题。

此外，您可以从各个角度检查过程。例如：

- 从时间戳来看 - 这个过程在上午 9:30 的状态是什么？
- 从代币来看——第三位顾客在店里做了什么？
- 从属性来看属性号车的问题数量是如何减少和增加的？
- 从多个线程 - I可以看到客户走进并在图表上自动模拟吗？
- 从资源中 - 支持人员何时忙或闲？为什么客户要等 40 分钟？

所有这些问题都可以在 节”页面上得到解答。

访问

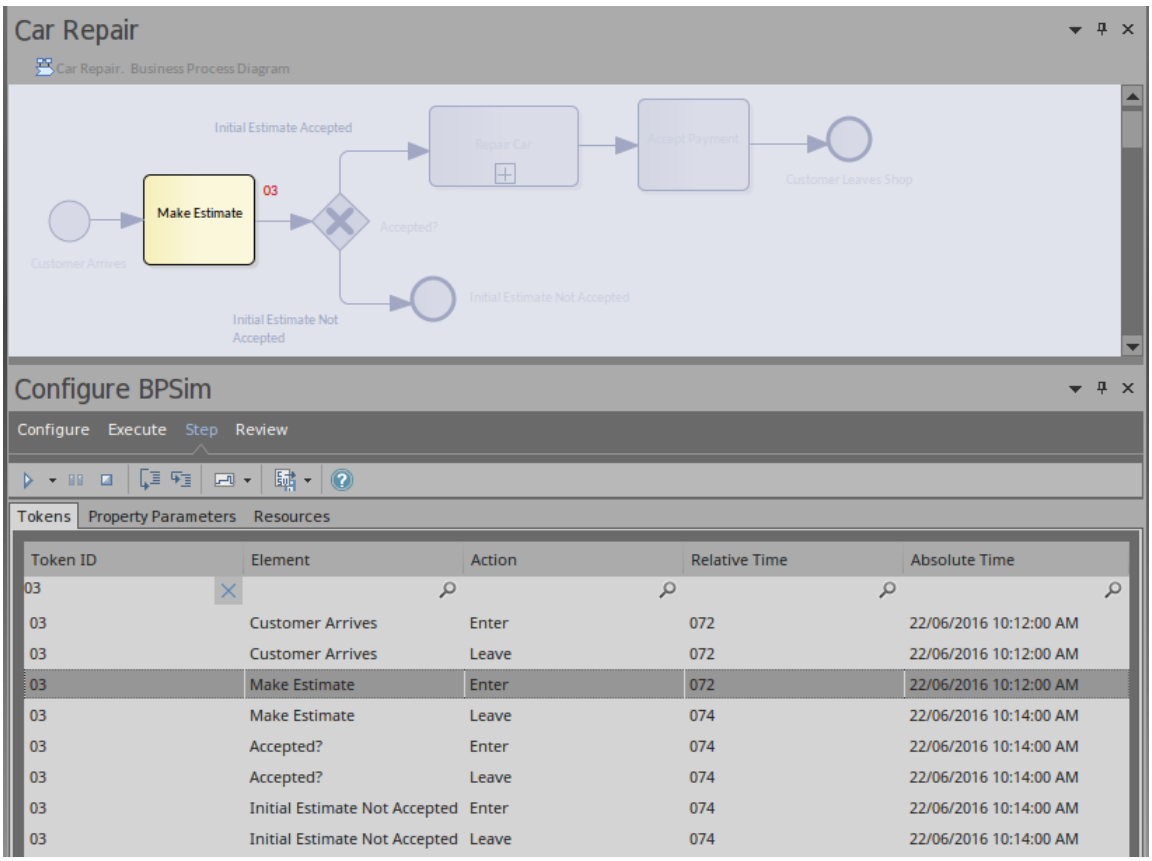
功能区	仿真>流程分析>进程> 打开BPSIM Manager >节页

工具栏选项

选项	描述
	点击此按钮，根据执行结果自动模拟流程。 单击下拉箭头和菜单选项 设置回放速度”，然后将模拟速度调整为正常的倍数。例如，键入 60”使模拟比实际活动快 60 倍；现实生活中的1分钟将在1秒内模拟出来。
	单击此按钮可暂停自动重播模拟。
	单击此按钮可停止模拟。
	单击此按钮可将 节结束”到下一个时间戳。每个 节结束”可以包含多个 步骤”。
	单击此按钮可播放单步。这表示过程中令牌的单次移动。
	单击此按钮为仿真生成一个图表。 您可以从菜单中选择 为每个生成单个时间线”或 为每个令牌生成多个时间线”。请参阅本主题 面的生成图表。
	单击此按钮可将此步骤页面上过滤的记录导出到 CSV 文件。您可以选择从哪个选项卡（ 令牌”、 属性参数”或 资源”）导出数据。

令牌选项卡

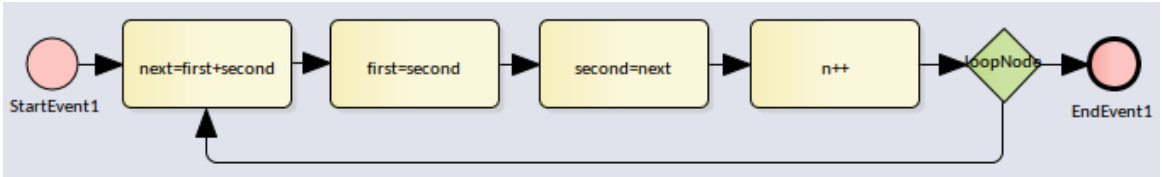
运行执行后，这个页面会填充模拟过程中的token信息；序列的顺序是按照触发时间的顺序。



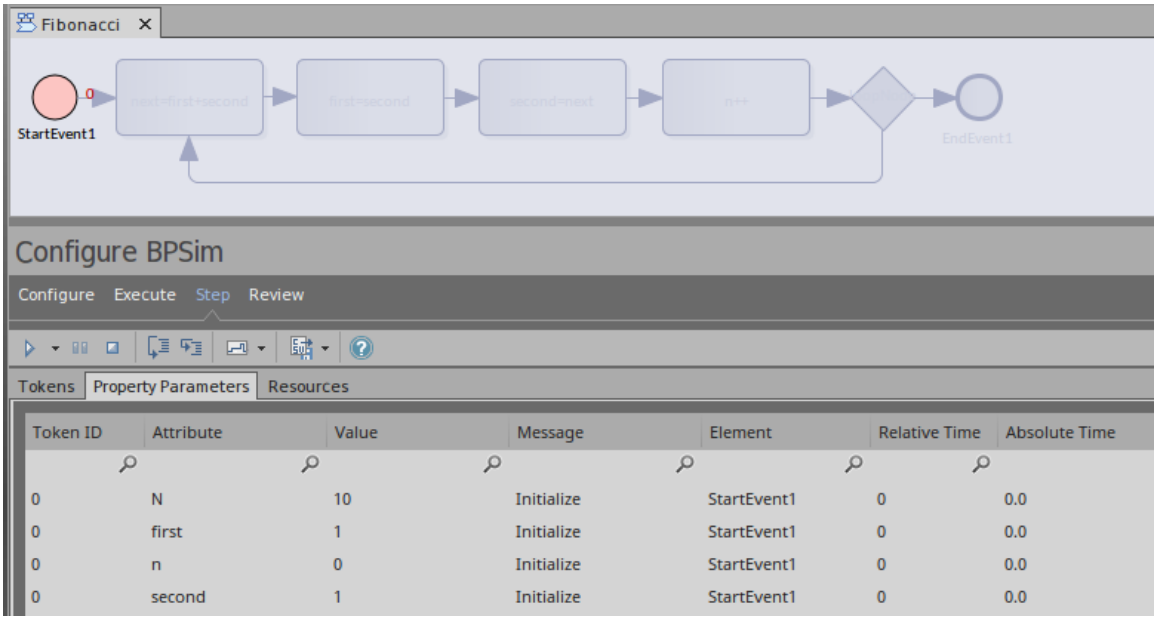
- 使用标题带中的过滤器栏（右键单击列标题并选择 切换过滤器栏”），您可以过滤显示的结果；例如，在 令牌 ID”列中键入 03 将仅显示令牌 03 的记录
- 如果您单击 节in”按钮一次，将播放列表中的一条记录
- 如果您双击记录，模拟将从头开始 节到”该记录
- 如果元素上设置了时间参数，点击节结束按钮将运行到下一个时间事件的最后一条记录
- 播放列表中的记录时，模拟快照将显示在图表上

属性参数选项卡

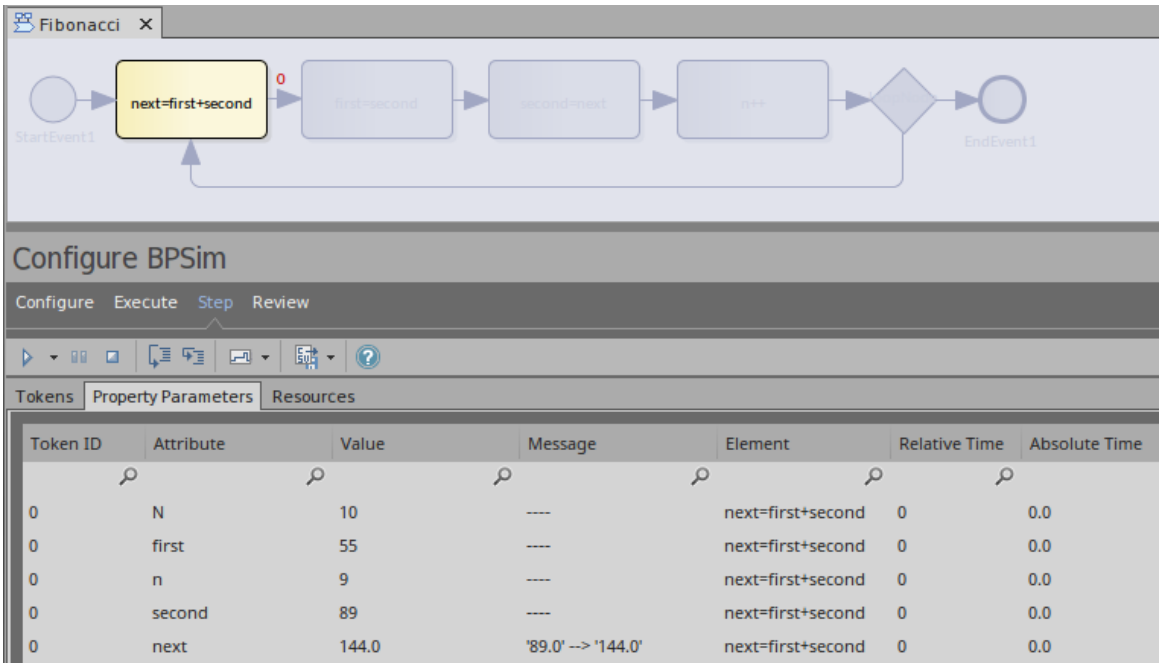
在播放 令牌”选项卡上的记录时， 属性参数”选项卡将显示时间戳属性的运行时间值。
例如，计算斐波那契数的 BPMN 流程可能会以这种方式建模：



在定义好属性参数，为每个元素配置属性参数并执行模型之后，我们就可以进行步骤模拟了：



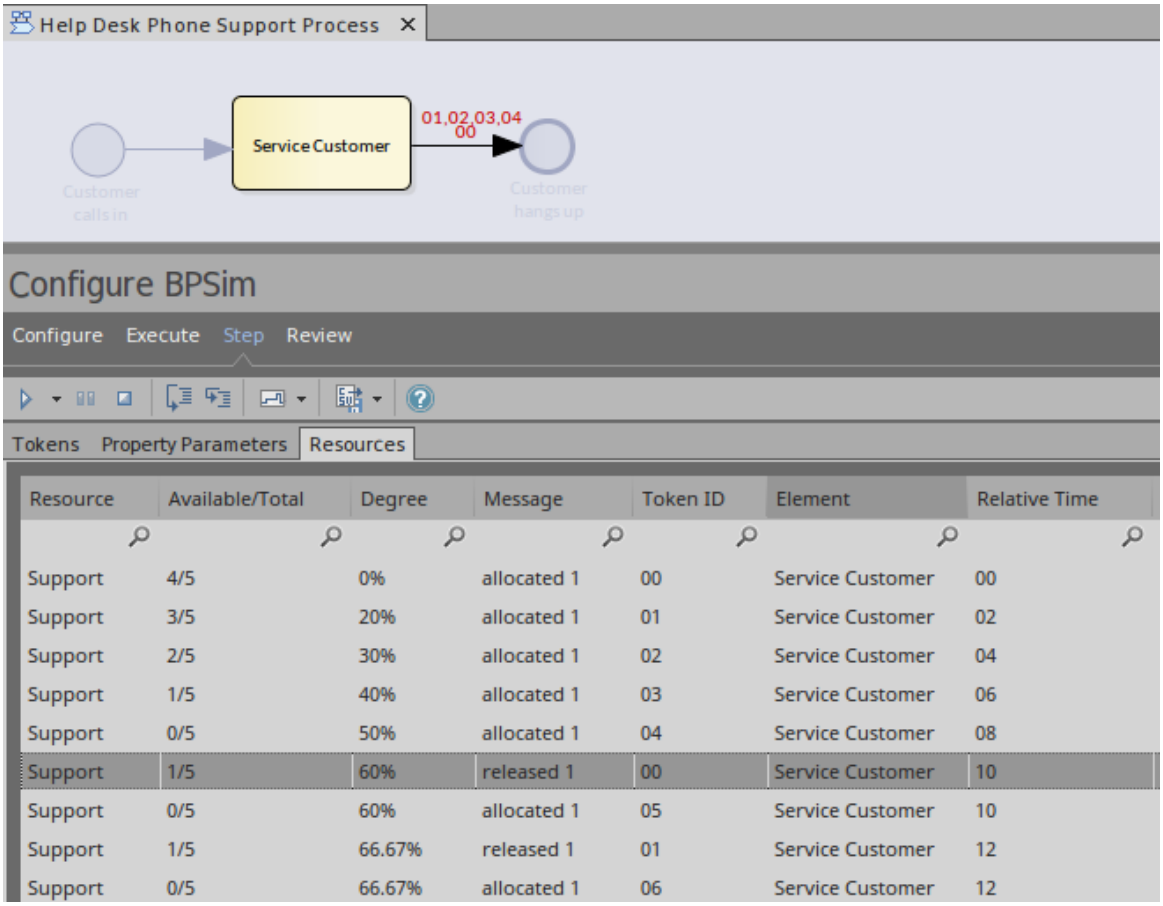
'信息'栏表示初始化了属性' N '、'first'、'n'和'second'。



如果你一直点击节在按钮，列表中的属性将改变它们的值。插图显示，在输入任务 ńext = first + second"时，属性 ńext"的值从 89 变为 144。

资源标签

在播放 令牌"选项卡上的记录时， 资源"选项卡将显示可用的运行时资源、可用资源的数量以及时间戳的分配或释放事件。



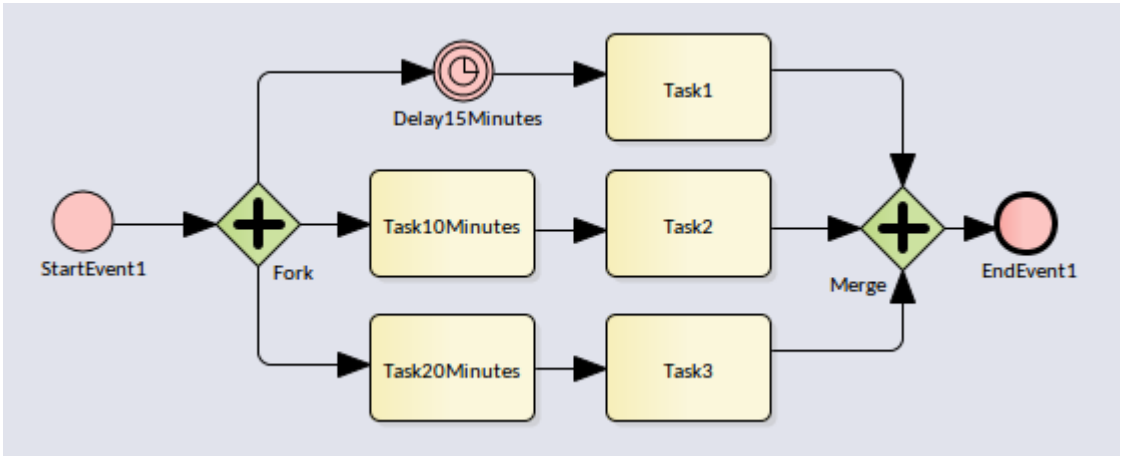
Resource	Available/Total	Degree	Message	Token ID	Element	Relative Time
Support	4/5	0%	allocated 1	00	Service Customer	00
Support	3/5	20%	allocated 1	01	Service Customer	02
Support	2/5	30%	allocated 1	02	Service Customer	04
Support	1/5	40%	allocated 1	03	Service Customer	06
Support	0/5	50%	allocated 1	04	Service Customer	08
Support	1/5	60%	released 1	00	Service Customer	10
Support	0/5	60%	allocated 1	05	Service Customer	10
Support	1/5	66.67%	released 1	01	Service Customer	12
Support	0/5	66.67%	allocated 1	06	Service Customer	12

生成计时图表

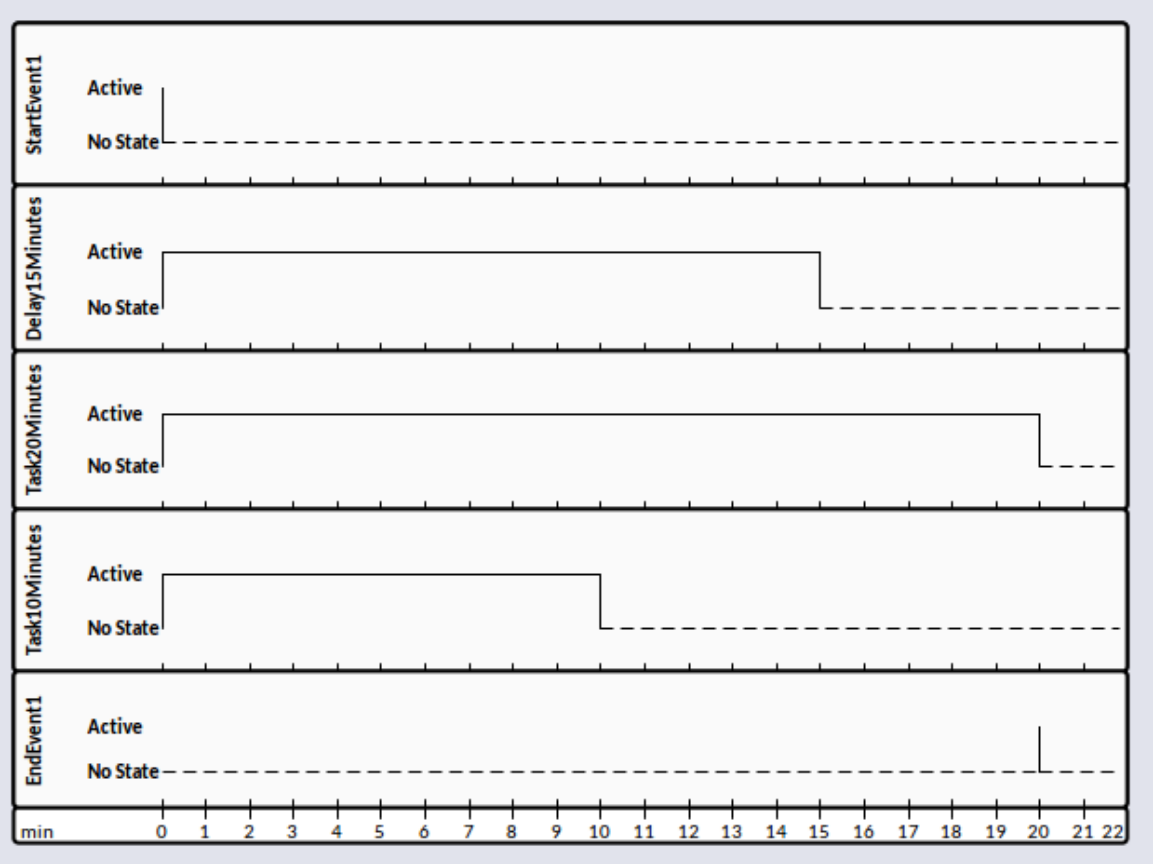
当在 BPMN 元素上配置时间参数时， Enterprise Architect可以为模拟过程生成时序图。

- 为每个令牌生成单个时间线 - 将此选项用于 单线程”流程；也就是说，没有并行网关或事件子进程
- 为每个令牌生成多个时间线 - 在 为每个令牌生成单个时间线”选项不适用的情况下使用此选项

例如：



执行此模型并单击 为每个令牌生成多个时间线”，生成的时序图如下所示：


StartEvent1
Active
No State
Delay15Minutes
Active
No State
Task20Minutes
Active
No State
Task10Minutes
Active
No State
EndEvent1
Active
No State
min
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

BPSim - 审查页面

此审阅页面包含三个选项卡：

- 配置总结
- 标准结果报告
- 自定义结果报告

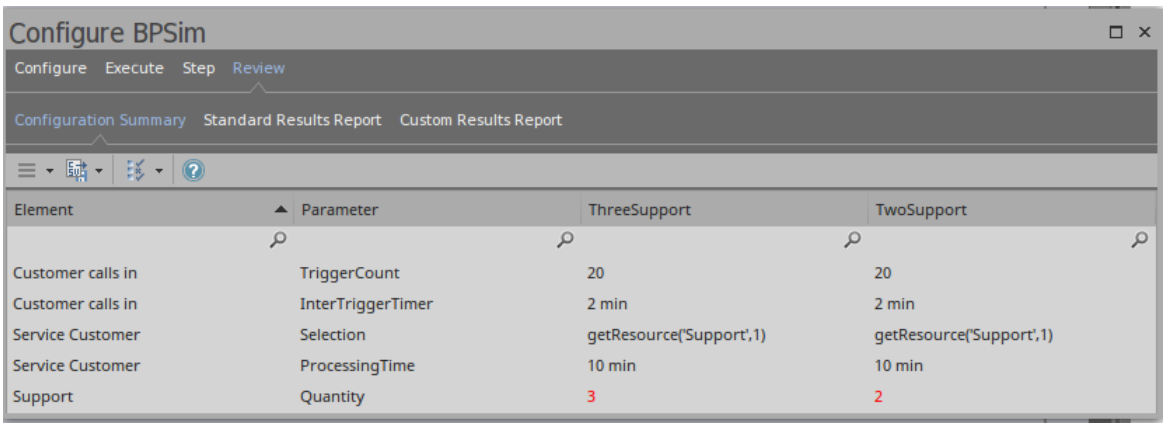
这些选项卡以类似的方式添加一个工件以审阅工件。这使您可以轻松进行假设分析。

访问

功能区	仿真>流程分析>进程>打开BPSIM Manager>审阅

"如果"分析

在服务台支持示例中，我们可以比较两个工件及其相应的结果。



在此图中，我们点击了工具栏中的 图标，并选择了 仅显示不同的项目”选项，以查看更改的参数值所导致的差异。

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

我们看到，当支持人员数量从 3 人减少到 2 人时，等待资源的平均时间从 11.4 分钟增加到 27 分钟。

使用参数值对话框

参数值”对话框可帮助您为整个配置中的各种参数定义值。它支持定义简单的固定值，直至产生派生值的分布和表达式。并非所有类型的值或派生都适用于所有类型的参数。

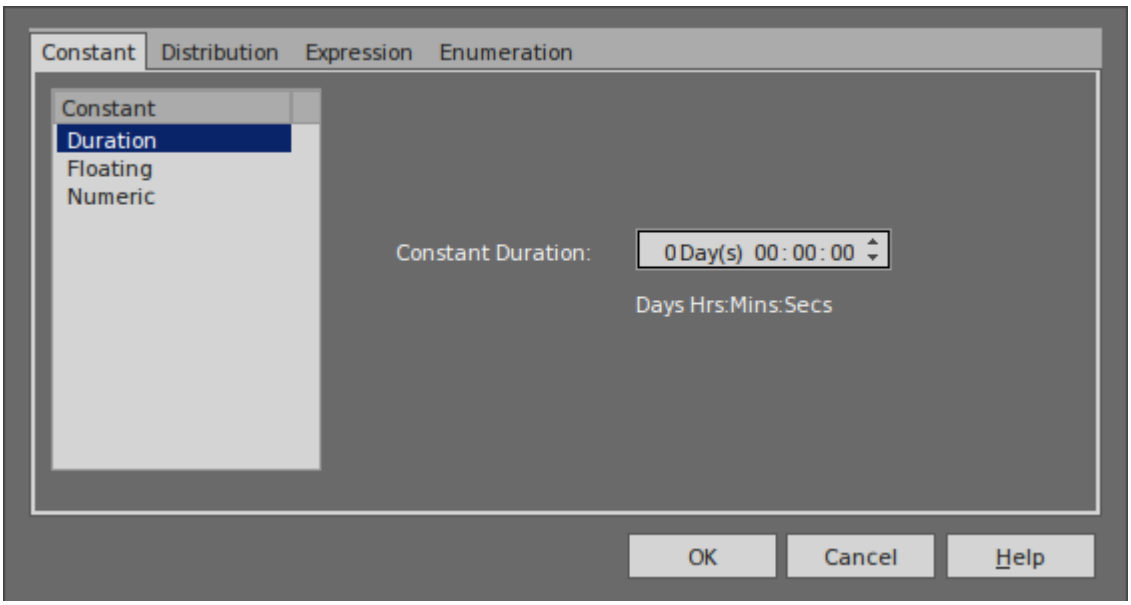
对话框名称取自object名称和正在定义的参数名称；例如，为 'Activity1'配置'Processing'。

访问

将工件加载到配置 BPSim窗口中，在图表或浏览器窗口中选择一个元素，然后在 值”字段中单击  。（如果尚未创建参数，请选择列表中的类别和参数以创建新参数。）

常量选项卡

使用此选项卡为参数定义特定值 - 例如，数字、文本string或时间。



在 常量”面板中，选择常量的类型：

- 漂浮的
- 数字
- 字符串
- 约会时间
- 布尔值，或
- 期间

适当的字段显示在面板的右侧；输入值，如果需要，输入表示值的单位（例如，时间单位或货币单位）。对于某些类型的参数，可以使用下拉列表从中选择一个值。

分布选项卡

在此选项卡上，您可以应用统计抽样方法来获取参数值；对于每种可用的分布类型，将显示相应的字段供您输入分布参数。所有发行版都要求您识别表达单位。

分布参数对于您正在开发的业务流程不是必需的，但（如果您从分布中获取值）是模拟所必需的。

您可以从以下分布类型中进行选择：

- **Beta** - 在短范围内提供 真实"值的连续概率分布，通常为 0 到1
- **Weibull** - 提供 真实"值的连续概率分布，通常用于object寿命分析
- **Gamma** - 提供 真实"值的连续概率分布，可用于对指数分布的随机变量进行建模
- **二项式**- 整数"分布，根据试验次数和某个结果的概率提供值
- **Erlang** - 根据 Erlang *K*值和分布的平均值提供 真实"值
- **正态**- 根据分布的均值和标准差提供 真实"值
- **LogNormal** - 对数呈正态分布的 真实"随机变量的连续概率分布
- **泊松**- 离散（ 整数"）概率分布，表示在固定时间或空间（体积、距离或面积）间隔内独立发生的给定数量事件的概率
- **NegativeExponential** - 根据分布的平均值提供 真实"值
- **三角形**- 根据分布模式和范围的最小值和最大值提供 真实"值
- **TruncatedNormal** - 根据范围的最小值和最大值内的点的平均值和标准差提供 真实"值
- **统一**- 提供范围内最小值和最大值之间的 真实"值

表达式选项卡

在此选项卡上，您键入 XPATH 1 .0 表达式以组合要在运行时处理的显式值、运算符和函数以提供值。表达式的每个属性参数必须用大括号括起来 - {xxx}。

示例1 ：为了表示 c = a + b + 10，我们将此表达式分配给属性 c"：

{a} + {b} + 10

其中 a""和 b""是在属性模型中定义的属性。

示例2：为了表示 $c = t - p * (a - b)^2$ ，我们将这个表达式分配给属性' ：

{t} - {p} * 数学。 pow ({a} - {b}, 2.0)

注记：使用该表达式模拟模型时，请选择 Java "作为语言，以便使用 java 内置的函数数学。 pow （ ）。

枚举选项卡

在 枚举"选项卡上，您可以定义一个枚举以提供一组常量值。您可以从现实世界的历史数据或模型的分析和模拟中获得这些值。每次评估参数时，都会返回下一个枚举值。

定义每个数值时，单击 保存"按钮将其添加到可能值列表中，然后单击 新建"按钮清除准备输入另一个值的数据字段。对于某些类型的枚举值，可能会要求您定义表示值的单位。您可以定义的枚举类型包括：

- 字符串
- 漂浮的
- 数字
- 期间
- 约会时间
- 布尔值

使用 BPSim 执行引擎

BPSim 执行引擎是一个插件

与Enterprise Architect的统一和终极版集成，执行您使用业务流程仿真仿真（功能）功能定义的模拟。 Engine 是访问和使用功能的先决条件。

访问

使用此表中的任工件配置配置窗口打开，单击配置 BPSim并浏览业务流程仿真按钮。

功能区	仿真>过程分析>进程> Open BPSim 管理器 （或流程分析>仿真>进程配置流程> 查找工件配置流程）
上下文菜单	右键点击业务流程仿真元素配置 BPSim工件
其它	用工件BPMN 右键单击业务流程仿真元素仿真|

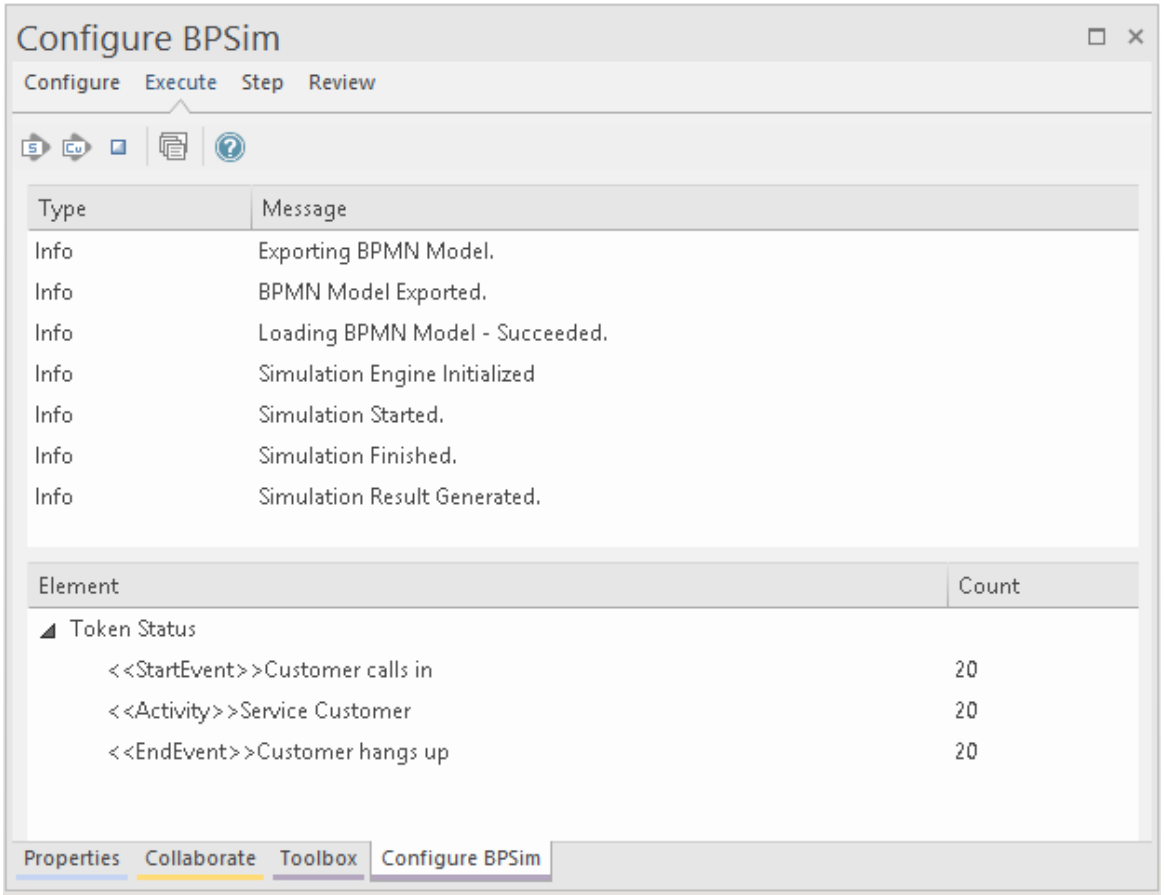
执行和仿真控件

单击 执行”选项卡，然后单击：

- 启动标准仿真或
- 启动定制仿真

这些选项触发相同的处理，除了标准仿真生成模拟中设置的所有内置参数的报告，而定制仿真仅提取您使用配置中的 结果请求”列特别标记的参数的结果.

模拟执行，在对话框顶部显示处理消息，并使用配置中使用的运行时值处理元素和参数。



在模拟过程中，您可以单击图标取消模拟。

将结果写入业务流程元素的工件包。 A <<自定义模拟>>-标准模拟写入<<BP 模拟报告>>-stereotyped工件 Report，而自定义模拟写入<<自定义报告>>-工件。

跟踪属性Values

除了内置参数外，您还可以在配置中定义自己的流程特定属性参数（属性）。模拟完成后，如果您已定义属性参数，则启用 属性”按钮。当您单击此按钮时，将显示 ĐPSim PropertyParameter Values”对话框，您可以通过该对话框跟踪所有属性参数的运行时值如何在业务流程中累积或更改。

审阅仿真

模拟完成处理后，单击 Open Result 按钮。 “BPMN仿真报告视图”选项卡在主工作区打开，显示当前模拟中内置参数的结果（但不显示用户属性的参数）。如果您已经基于相同的业务流程运行了另一个配置的模拟，那么它也会作为附加列显示在报告中。否则，您可以点击工件元素的报表属性，并将其拖到报表配置选项卡上，以比较两个（或更多）下的内置参数。

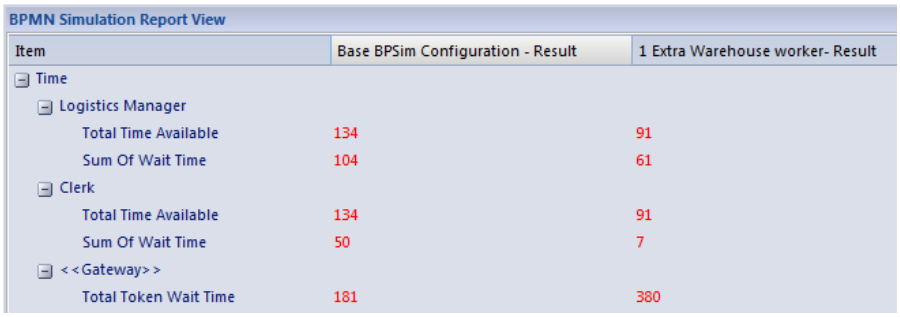
为了方便查看报告中的数据，您可以将 ‘BPMN仿真报告视图”选项卡拖出主视图，使其变为浮动窗口，并将窗口放大到合适的大小。

单击要检查的参数对应的扩展框。您还可以使用右键单击上下文菜单选项公开和过滤信息。

您可以将来自不同模拟的结果之间的特定差异表示为图表。模拟结果>>（<<工件的结果元素）名称必须存在，然后才能设置图表工件。有一种用于标准图表的模板工件和一种用于标准模拟的模型。

BPMN仿真报告选项

选项	描述

全部收缩	选择此选项可将参数层次结构折叠到仅父选项卡名称。
展开全部	选择此选项可将参数层次结构扩展到最低值类型。
仅显示不同的项目	（当您显示两个或更多模拟时。）选择此选项可将显示限制为模拟之间值不同的那些参数。再次单击该选项以取消选择它。
突出显示不同项	（当您显示两个或更多模拟并且它们的某些参数值不同时。）以红色显示不同的参数值。如果您选择 仅显示不同项”选项，此选项将被禁用。 
仅显示非空项	选择此选项可过滤显示以仅显示具有非 0 特定值的参数。
移除模型	（当您选择了特定的结果时，它会在报告中标识模拟。）选择此选项可从报告中删除模拟列。

BPSim执行引擎-仿真语言

BPSim 执行引擎支持在 XPath 1 .0 或Java上进行模拟，其中适当的语言被定义为模拟配置中的表达式语言。它还支持在属性Parameters 中使用流程实例数据，其中实际值仅在执行期间确定。

XPath 1 .0 运算符

这些运算符可用于 BPSim 表达式参数。

操作员	描述
|	Union算子，用于资源获取。 示例： getResource('w1', 1) |获取资源（ 'w2'， 1 ）
+	添加。 示例：4 + 6
-	减法。 示例：6 - 4
*	乘法。 示例：6 * 4
div	分配。 示例：8 div 4
=	平等。 示例：4 = 4（ True ）
!=	不相等。 示例：5 != 3
<	少于。 示例：6 < 9
<=	小于或等于。 示例：x <= 6
>	比...更棒。 示例：9 > 6
>=	大于或等于。 示例：n >= 7
或者	选择。 示例：n = 6 或 n <= 6
和	组合。

	示例：n = 5 和 m < 8
模组	模数除法。 示例：5 模 2
获取属性	获取一个属性值。 示例：getProperty（ 金额”）
获取资源	获取资源分配。 示例： getResource ('w1', 1)

注记

表达式语言可以在配置 BPSim窗口中的 配置”选项卡上进行设置； 'XPath 1 .0' 和 ' Java ' 两个选项可用作 'Expression' 参数的值。

如果选择 “Java ”，则必须将属性 “JDK Home”设置为有效的 JDK 目录。

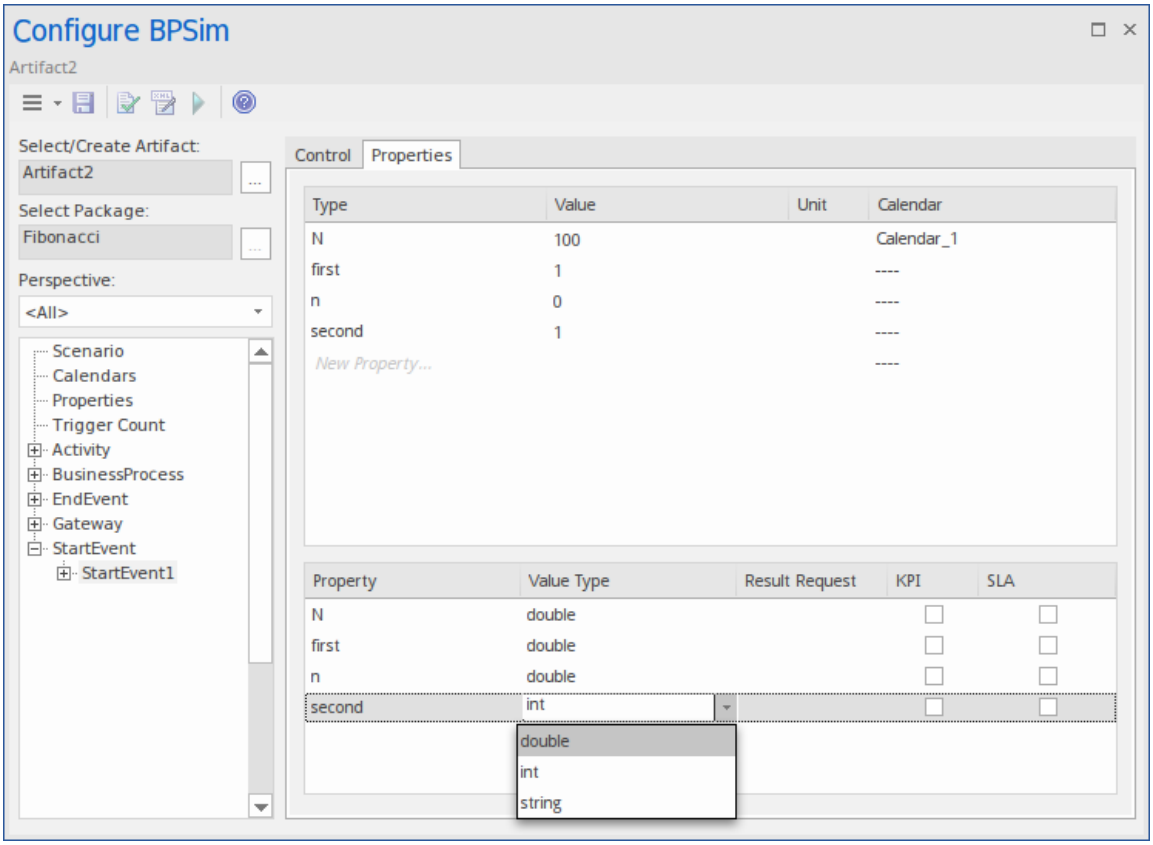
提示：您可以使用 {PropertyParameterName} 作为 getProperty('PropertyParameterName') 的简写形式，这在为表达式编写值时很有用；例如：

{n} < { N } 而不是 getProperty('n') < getProperty(' N ')

getProperty 运算符的缩写形式可用于1 1.0 和Java 。

属性Parameters

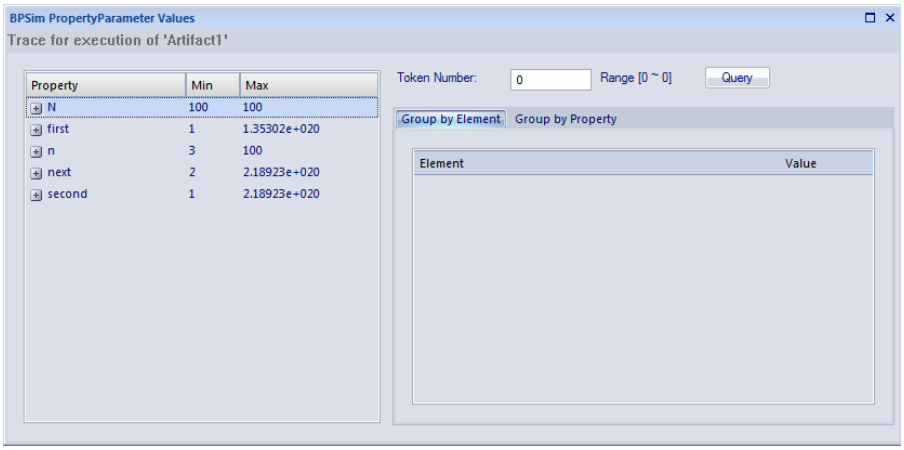
从Enterprise Architect 13.0 版开始，属性参数可以保存流程实例数据，直到运行时间才被赋值。您可以在配置 BPSim属性的 属性”页面定义属性参数类型；支持的类型是 “int ”、 “double”和 “string ”。


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

跟踪属性参数Values

业务流程模拟器 (模型) 可帮助您对业务流程的操作细节进行建模和测试，例如活动和任务的资源分配、事件的干预以及决策点的影响以及在这些点做出的决策。您将这些特定于流程的属性参数或属性添加到属性配置中，当您根据配置在模型上运行模拟时，BPSim 引擎会帮助您捕获每次迭代的属性参数的运行时值模拟，并过滤结果以检查特定的路径或决策点。这使您可以非常详细地了解在特定条件或条件组合下您的业务流程中可能实际发生的情况，以生成结果或显示产生该结果的处理路径。

访问

上下文菜单	右键单击定义的业务流程仿真- 仿真运行仿真配置|工件模拟类型... : 属性 (如果模拟配置不包含任何属性参数，则属性按钮不可用) 

BPSim PropertyParameter Values 对话框字段

选项	行动
属性	此表列出了为进程定义的属性，并显示了整个进程中每个属性的最小和最大可能值。 如果单击某个属性的扩展框，则该表会显示该过程中每个活动或事件 (元素) 的属性的最小值和最大值。
令牌编号	类型要检查的 令牌”的数量；此数字必须在字段右侧显示的范围内。 A令牌”是一个独立的触发器，例如客户或订单进入业务并在审阅下启动业务流程。可以有任意数量的客户或订单，每个客户或订单都与任何其他客户或订单没有任何关系，并且每个客户或订单都可能在业务流程中遵循不同的路线。 如果一个可能的触发事件只有一个可能的实例，例如抛出一个开/关开关，则该令牌被视为0。
查询	单击此按钮以启动对令牌模拟的查询，以填充 按元素分组”和 按属性分组”选项卡。
按元素分组	从属性参数的值如何在选定元素中变化的角度显示结果。该选项卡显示过程中的元素列表，并且对于每个元素，在模拟的每次迭代中应用在元素中的每

	个属性的值。使用标题栏上的 切换过滤器栏”选项，您可以优化显示以仅显示特定属性并查看元素使用它的频率以及使用什么值。
按属性分组	从各个属性的值在整个过程中如何变化的角度来展示结果。该选项卡显示了在流程中应用的属性列表，并且对于每个属性，在流程的每次迭代中每个活动（元素）中的值。

例子

在 EAExample模型中，您可以研究从 BPMN 业务流程模型的模拟生成属性参数信息的两个示例。这些将演示如何根据模型在配置中定义属性参数。您最初可以只对每个示例运行一个模拟，然后按照此处所述检查输出。然后，您可以检查业务流程和配置本身，并更改或添加提供的属性参数。

这些示例在跟踪属性*ParameterValues* - 示例主题中进行了描述。简而言之，它们是：

- 斐波那契”——一个非常简单的递归业务流程，通过十次迭代计算一系列斐波那契数；您可以通过过程的元素看到属性参数在每次迭代中如何递增（在示例模型>模型仿真>示例模型>斐波那契）
- 汽车维修” - 一个更复杂和现实的过程，表示当一系列单独的 步入式”客户将车辆带入汽车维修店进行估计和维修时可能发生的情况（在示例模型>模型仿真>示例模型 > 汽车修复进程）

还有一个时间参数行为的小例子（在示例模型>仿真模型>示例模型>时间参数）。

注记

- 如果 BPSim 配置包含结果请求并对其执行自定义模拟，则 BPMN仿真报告”仅显示配置中请求的内置参数；相反，“BPSim PropertyParameter Values”对话框列出所有属性参数，而不管任何结果请求设置或模拟类型如何

跟踪属性参数Values - 示例

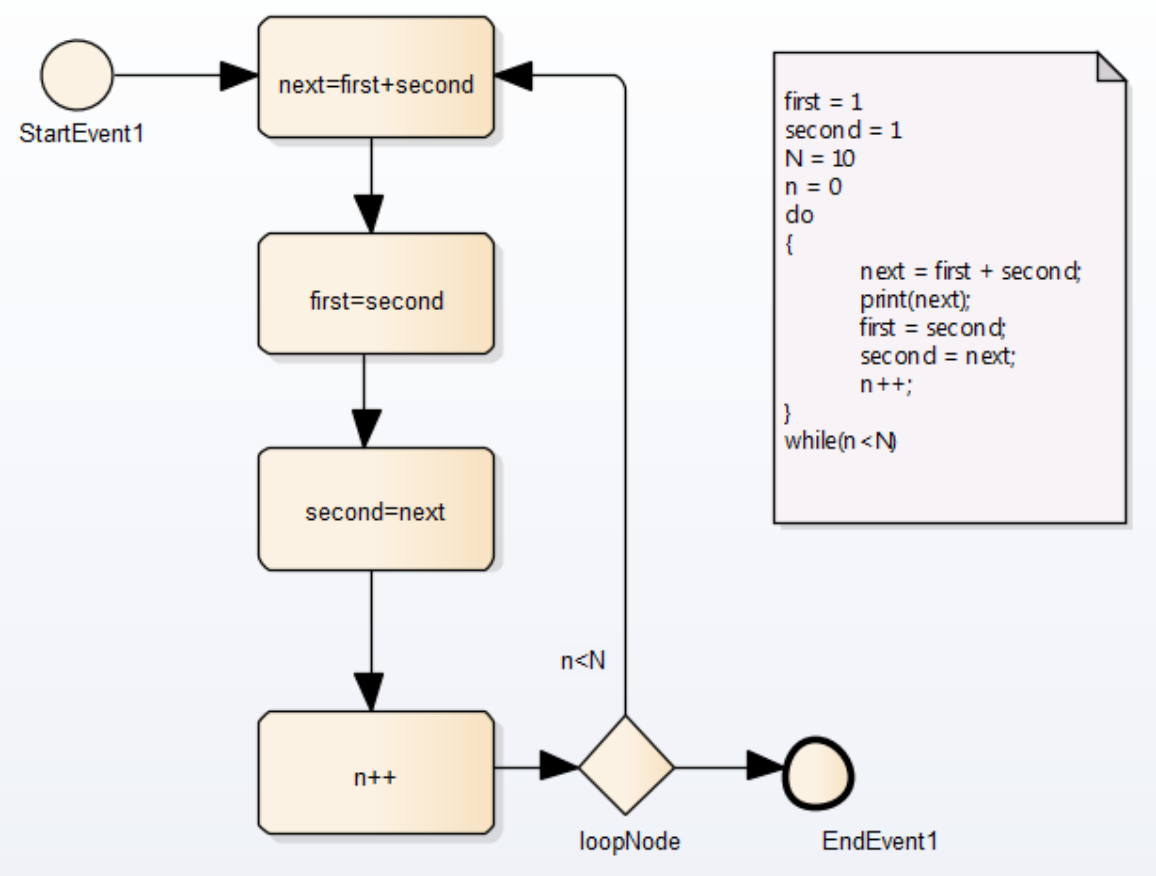
为了帮助您了解从 BPMN 业务流程模型的模拟中生成属性参数信息的功能， Sparx Systems提供了两个示例，您可以在 EAExample模型中探索。这些是：

- 斐波那契过程 - 一个帮助您熟悉参数跟踪功能的非常简单的示例
- 汽车维修过程 - 一个更复杂的示例，您可以操作它来了解如何调查现实生活中的过程

在本主题的结尾部分简要讨论了如何使用包含由 真实"分布初始化的参数的基于整数的过程，以及描述时间参数行为示例的部分。

斐波那契示例

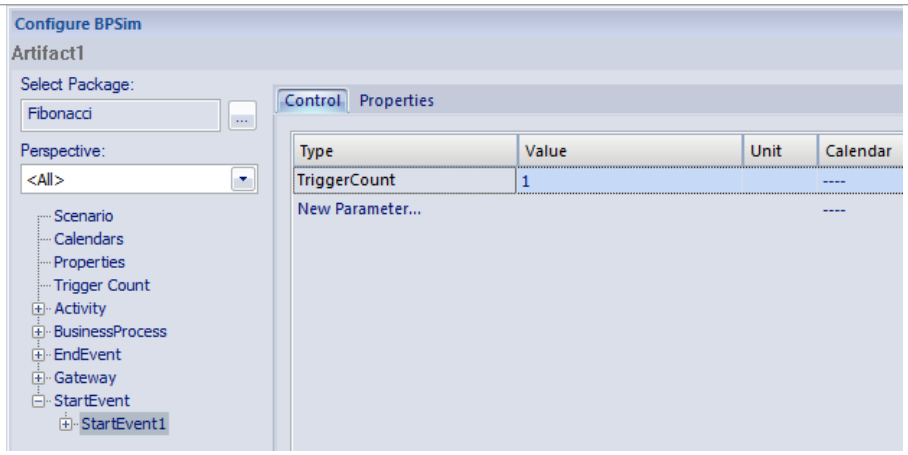
这是一个非常简单的递归业务流程，通过十次迭代计算出一系列斐波那契数；您可以通过流程的元素看到属性参数如何在每次迭代中递增。打开示例模型>模型仿真>示例模型 > 斐波那契。



该过程的伪代码如图上的注记元素所示。语句 'print(next)' 将输出数字系列 2、3、5、8、13、21、34、55、89、144。

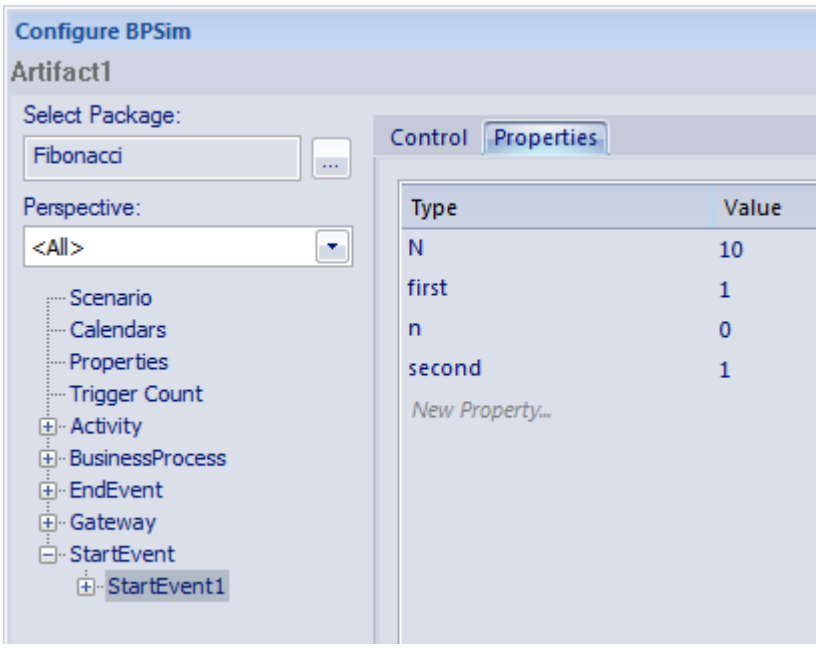
此流程的 BPSim 配置按此处所述进行设置。

节	行动
1	



在 StartEvent 的 无素控件”选项卡上，将 控件”设置为 “1 ”，并在 属性”选项卡上创建并初始化属性：

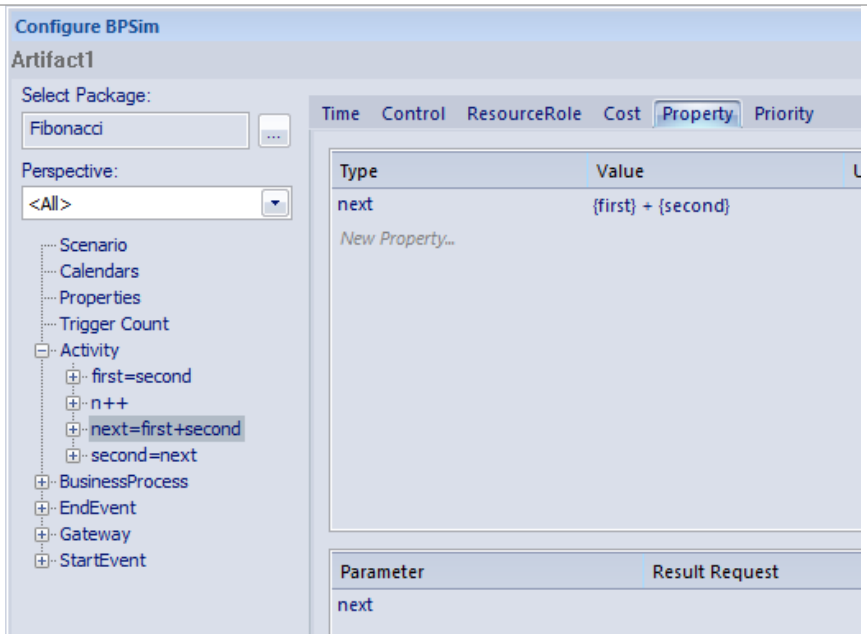
- ' N ' 作为 '10'
- '第一'作为' 1 '
- '第二'作为' 1 '
- 'n' 作为 '0'



2

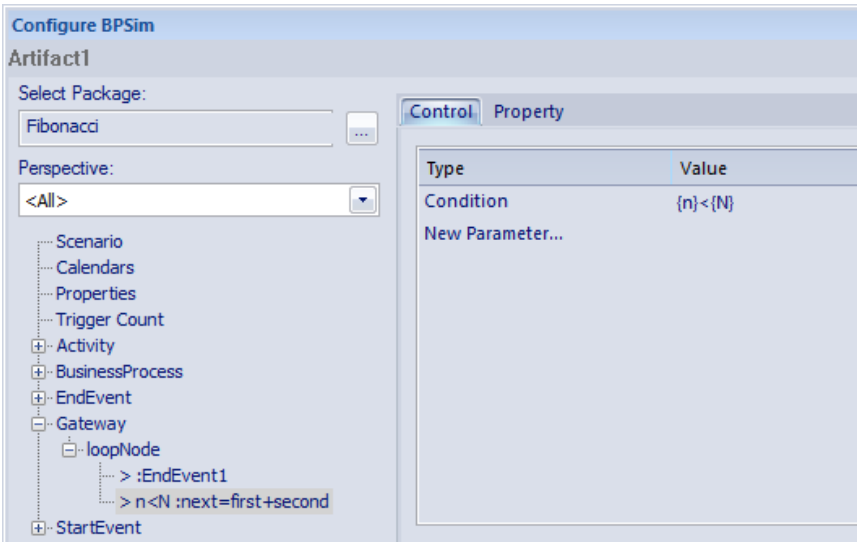
现在在 属性”选项卡上为流程中的每个活动定义属性。注记这些属性的值是从**Expressions**派生的，其组件必须用大括号括起来 - {xxx}。对于活动：

- next=first+second - 设置属性'next'并将值定义为表达式 {first} + {second}

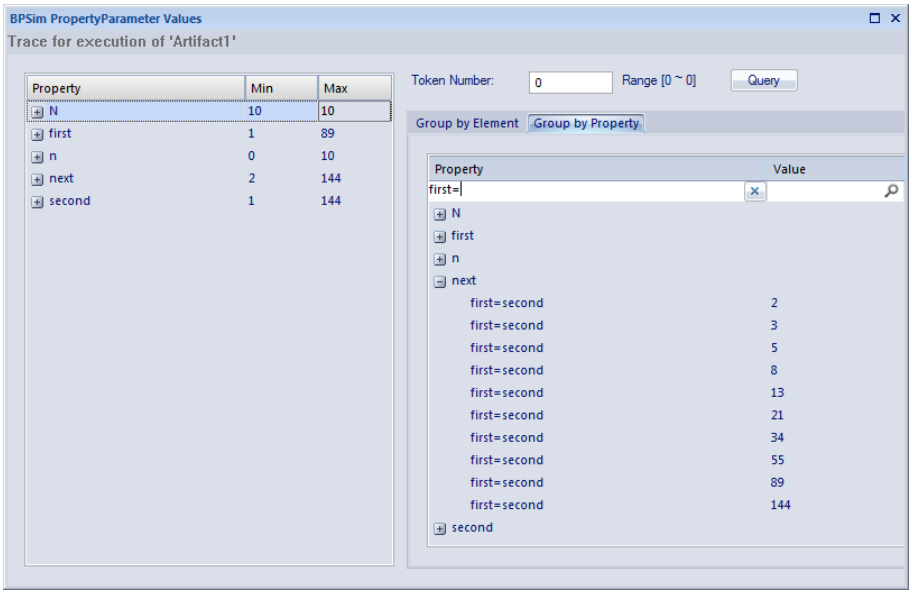
<table>
<tr><td></td><td> 

first=second - 设置属性 “first”并将值定义为表达 {second}
second=next - 设置属性'second'并将值定义为表达式 {next}
n++ - 设置属性'n'并将值定义为表达式 {n} + 1
</td></tr>
<tr><td>3</td><td>在 “条件控件条件元素发出的两个 Sequenceflow 连接器设置 “属性参数”。
扩展网关 | loopNode元素和链接到：

next=first+控件- 将参数设置为 “条件”并将值定义为表达式 {n}<{ N }

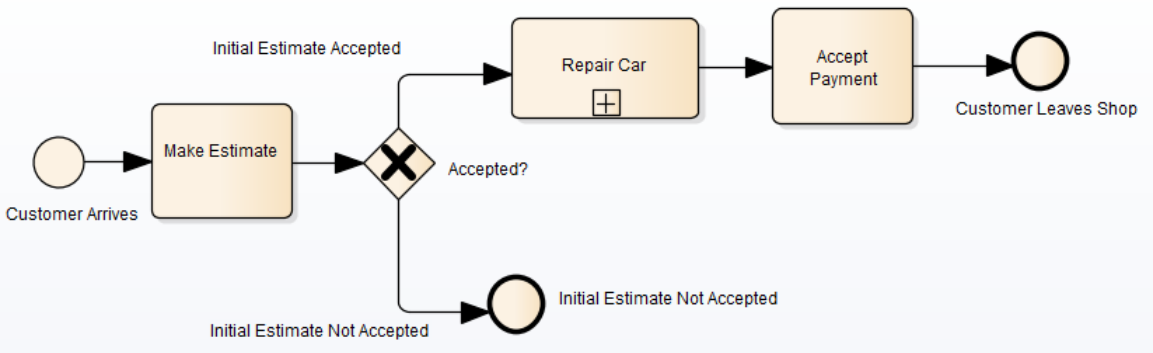


控件- 将控件参数设置为 “条件”并将值定义为表达式 {n}=>{ N }
</td></tr>
<tr><td>4</td><td>完成配置后，单击 “仿真仿真模拟控制器”对话框中的配置 BPSim上的运行按钮，选择一个标准模拟。
模拟完成后，单击 “属性”按钮。
在 “BPSim PropertyParameter Values”对话框中，将 “Token number”字段设置为“0”，然后单击查询按钮。</td></tr>
<tr><td>5</td><td>现在检查在模拟的每次迭代中输入 first=second活动的 “下一个”属性的值。单击 “按属性分组”选项卡并展开 “下一个”项。</td></tr>
</table>

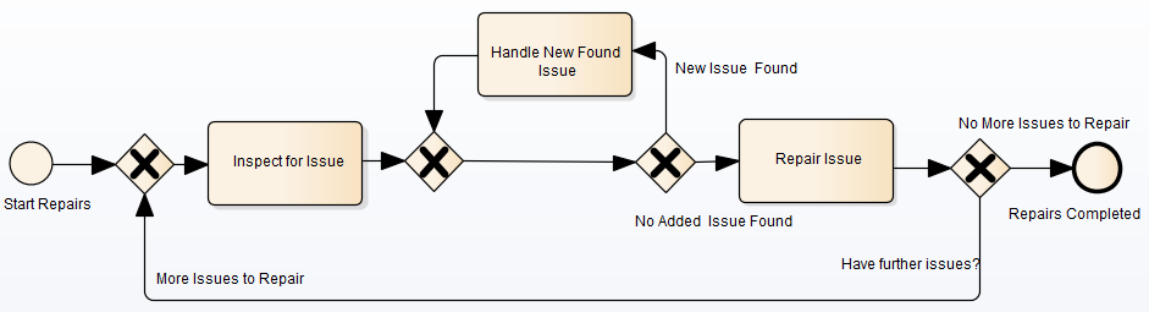
	值列表很长，因此右键单击列标题并选择 "切换过滤器栏"选项。在 "属性"列标题下，键入 "first="。这会过滤列表以仅在输入 first=活动时显示属性参数值。 

汽车维修示例

这个更复杂的示例基于汽车维修过程的真实模型，其中许多个人客户要求进行维修估算，然后继续维修或拒绝继续；您可以看到属性参数如何随着过程中做出的不同决策而变化。打开示例>模型仿真>汽车进程示例模型。整个流程如下图所示：



*Repair Car*活动是一个复合元素，包含以下子流程图：



节	行动
1	在浏览器窗口中，展开 汽车维修进程”包下的 浏览器工件”子包，双击 场景1 ：主流程”。将显示配置 BPSim窗口。在 场景”选项卡上查看 持续时间”字段；这已设置为 2 天 12 小时（即 60 小时）。
2	在窗口左侧的进程层次结构中，展开 开始事件”类别并单击 顾客到达”。选择 控件”选项卡，查看 控件”参数，其值为24分钟；也就是说，每 24 分钟就有一位顾客到达（所以在 60 小时的时间里，有 150 位顾客经过维修店）。 每个客户带着一个或多个需要评估和维修的问题进入维修店。每个客户提出的问题数量可以使用 BPSim 支持的发行版之一随机生成。由于问题编号以离散单位计算（而不是按连续规模测量），我们将使用 整数”分布。如果您为 到达”开始事件选择 顾客属性”选项卡，您将看到 属性”属性值是从平均值为 3 的泊松分布初始化的。
3	现在扩展决定网关 接受？”及其连接器，在流程层次结构中。 'In Estimate概率'有一个控件参数'，设置为0.67。替代的 初始概率连接器未接受”设置为控件。也就是说，我们预计平均有三分之一的问题会被客户撤回或不追究。
4	在此过程中，当评估车辆上的问题时，可能会发现另一个问题。 在网关元素列表中，最后一个 未命名元素”有两个路径： 新问题发现”和 未添加问题”。单击每个这些并查看 控件”选项卡； 'New QuestionFound'的'问题'参数设置为0.25，对于'问题概率'，设置为0.75。因此，平均而言，每报告和评估四个问题，就会发现一个新问题。 新发现问题”路径将流程带到 处理新发现问题”活动，这会将要为当前客户处理的问题数量加1 。展开活动组，然后单击 处理新发现的问题”元素和 属性”选项卡。您将看到此处的属性 “noOfIssues”具有表达 值 {noOfIssues} + **1** 。
5	当车辆出现问题时， 维修”问题活动会从当前客户需要维修的问题数量中扣除1 。单击活动组中的 修复问题”元素和 属性”选项卡。您将看到此处的属性 “noOfIssues”具有表达 值 {noOfIssues} **- 1** 。
6	来自 修复问题”活动的值在 还有其他问题？”中测试。网关。 单击 要修复的更多问题”连接器和 控件”选项卡；遵循此路径的条件参数设置为表达式值 {条件} **> 0** ；流程在 检查问题”活动之前传递到网关。 同样，如果您单击 “No More Issues to Repair”连接器并在 控件”选项卡上，遵循该路径的条件参数设置为表达式值 {条件} **=< 0** ，并且流程传递到 修复”完成'结束事件。 现在您已经检查了流程和配置设置，您可以运行模拟并审阅结果。
7	在配置 BPSim窗口，点击运行按钮，然后在 运行仿真仿真控制器”对话框中，再次点击 “Standard”模拟（但模拟类型对属性参数没有影响） ）。 在 仿真仿真控制器”对话框中，您可以审阅令牌状态（并看到另一个客户在最后一分钟设法进入商店），但很难确切地看到这些汇总数据是如何产生的。单击 属性”按钮可在 属性 PropertyParameter Values”对话框中获取详细的属性参数值信息。
8	对话框的左侧是进程中每个元素的属性参数（属性）的最小值和最大值的摘要。例如，对于 顾客

	到达"元素， 顾客"参数的最小值为 0，最大值为 8，由泊松 (3) 分布生成。 在 令牌编号"字段中，输入一个介于 0 和 150 之间的数字 (N)，为进入维修店的第 N 个客户选择。单击查询按钮以获取该客户在流程中使用的属性参数值。在两个选项卡中的每个选项卡上审阅结果： • 在 按元素分组"选项卡上，查看每个元素中属性的值如何变化；例如，对于客户 24，"noOfIssues"参数由随机分布初始化为 4，并且 检查问题"活动被调用六次，其中三个调用的参数值在循环前被调整为 3到1 ，并且 处理新发现问题"活动被调用两次，参数值都为 3 • 在 按属性分组"选项卡上，查看参数值如何随着流程循环完成活动而变化，从 4 开始，在 3 和 4 之间多次调整，然后在结束时递减到 0
9	根据需要继续探索结果，选择不同的客户（代币）。您还可以返回到 BPSim 配置并更改参数初始化并添加新参数，或更改决策点，以试验该过程。

在基于整数的过程的模拟中响应实数

在某些情况下，当流程中的活动使用整数运行时，或者当您想查看强制整数值对流程的影响时，您可能需要使用返回 实数"的分布来生成属性参数值。

在这种情况下应用的一种机制是设置条件以避免绝对数字。因此，例如，您可能有一个递减1的计数器，它被初始化为一个 实数"。如果将条件设置为 'value==0'（等于 0）或 'value !=0'（不等于 0），这两个条件可能永远不会分别为True或始终为True ，从而导致无限循环。为避免这种情况，您将使用以下运算符：

'值 > 0'

'值 < 0'

'值> = 0'

'值 <= 0'

另一种机制是编辑BPSim引擎使用的代码模板，截取提供给特定参数的实数并将其替换为整数，如图所示：

1. 选择 开发 >源代码 > 选项 > 编辑代码模板"功能区选项。
2. 在代码模板编辑器的 语言"字段中，单击下拉箭头并选择 MDGBPSimExecutionEngineExtension"。
3. 在 (Java) 模板列表中，单击 MDGBPSimExecutionEngineExtension Compute Value"。模板内容显示在 模板"面板中。
4. 找到这一行：

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

   将其更改为：

   %**如果**bpsimPropertyParameterName == "noOfIssues" 或 bpsimPropertyParameterName == "noOfVisitors"%
   双 %bpsimPropertyParameterName% = (int) distribution.next();
   //double %bpsimPropertyParameterName% = Math. ceil (distribution.next());
   //double %bpsimPropertyParameterName% = Math. floor （distribution.next（ ））；
   //double %bpsimPropertyParameterName% = Math. round （distribution.next()）；
   % **else** %
   double %bpsimPropertyParameterName% = (double) distribution.next();
   %**结束如果**%
5. 将属性参数名称替换为您自己的属性参数。
6. 单击保存按钮，关闭代码模板编辑器并重新加载项目。

如前所述，对于每个指定的参数，代码模板将简单地将任何由分布初始化的 实数"替换为整数。如果您愿意，可以改用注释行之一：

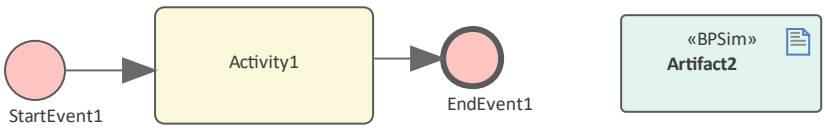
- 数学。 ceil () 将获取 真实"数字并将其转换为下一个最大整数

- 数学。 floor () 将采用 "实数"并将其转换为下一个最小整数
- 数学。 round () 将取 "实数"并根据它是否大于或小于 n.5 向上或向下round入

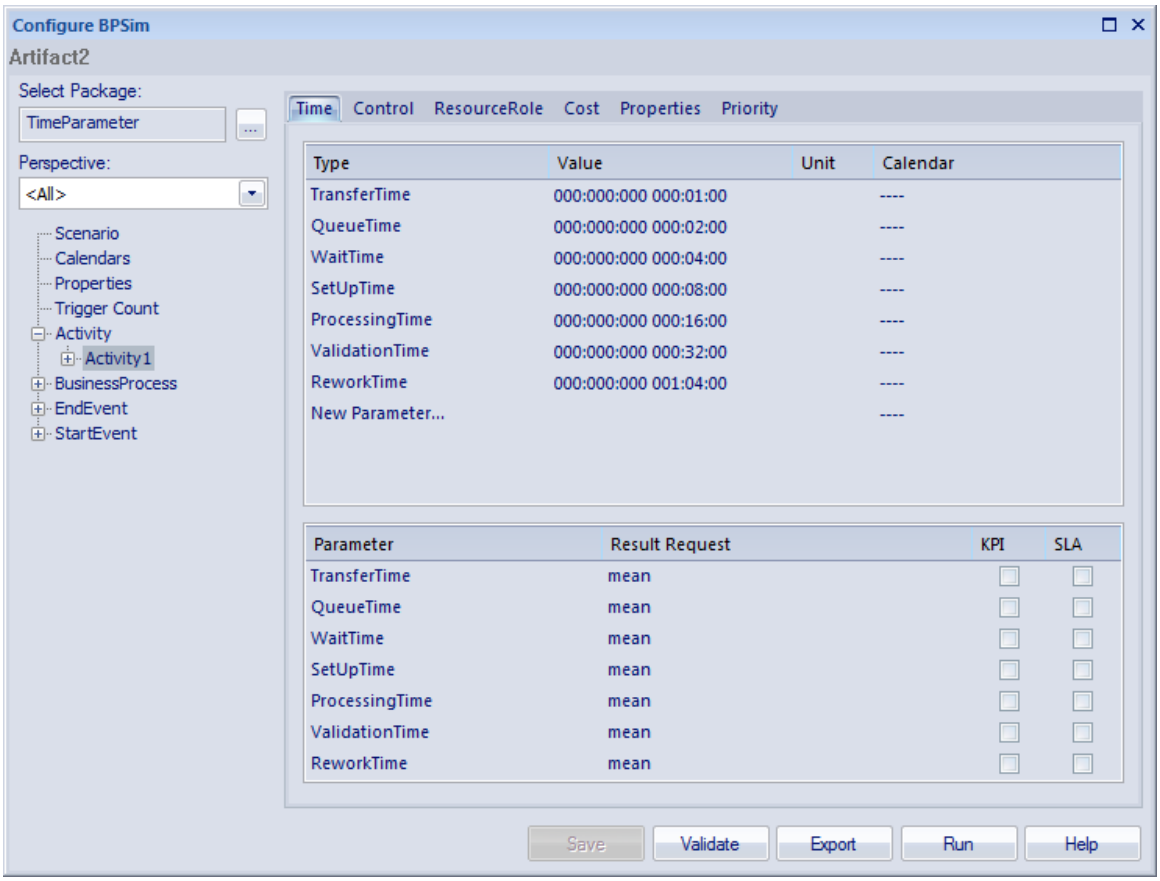
时间参数行为

在 BPSim 配置中，您可以为一个活动设置多个时间参数，例如队列时间和等待时间。您还可以为每个自定义模拟设置结果请求。然而，BPSim 模拟引擎将这些参数组合成一个单一的 "处理时间"量。

考虑示例模型中的简单模型时间参数（示例模型>模型仿真>示例模型>时间参数），如下图所示：

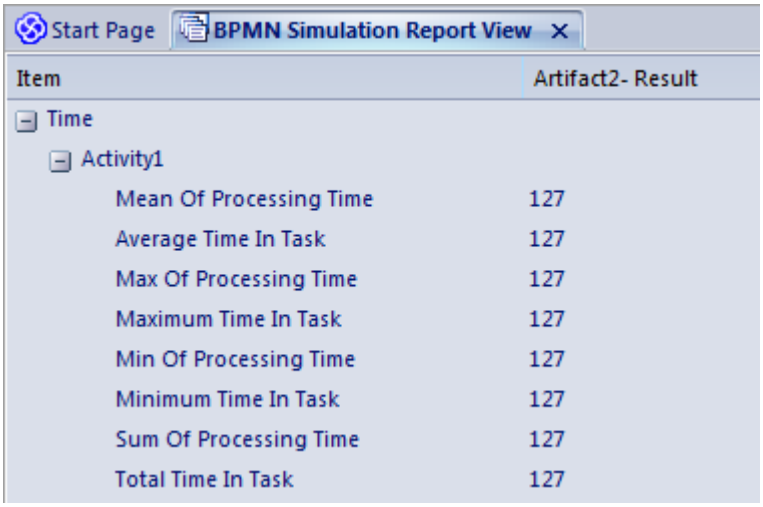


如果双击 Artifact2元素，将显示配置 BPSim窗口。单击图中的 Activity1元素以展开活动组，在对话框左侧的层次结构中选择 Activity1，并显示配置中元素的第一个选项卡 "时间"，如图所示。



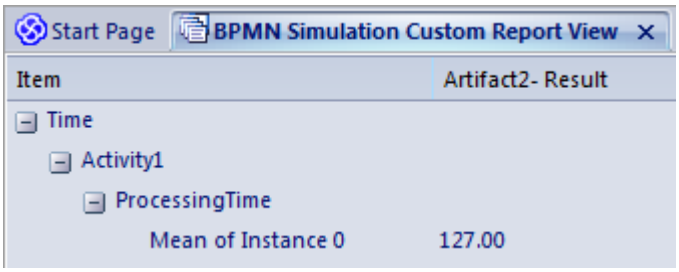
注记在上面的面板中有七个系统提供的 "时间"参数，它们的初始值依次为 - 1和 64 分钟（64 分钟为1小时）和 4 分钟）。注记在下面的面板中，每一个都有一个参数的平均运行时间值的结果请求。

点击运行按钮，在 "运行仿真控制器"对话框中点击运行按钮，选择 "标准仿真"。模拟被配置为循环通过该过程一次。模拟完成后，点击 "打开结果"按钮，在 "BPMN仿真报告视图"上单击鼠标右键，选择 "仅显示非空项"选项。对于设置了参数的 Activity1元素，这将为您提供以下结果：



所有这些导出的结果都是 127 分钟，即原始七个 时间"参数的初始值之和。各个参数不单独处理。

如果您返回 仿真仿真控制器运行"对话框并单击 仿真仿真运行"按钮，这次选择 仿真"， 打开结果"按钮将显示"BPMN仿真报告视图"。在配置中，结果请求是针对七个参数的平均值。在模拟的报告视图中，您只能看到单个聚合参数 ProcessingTime 的平均值为 127 分钟。



比较 BPSim 配置

当您开发 BPSim 配置时，您可以在运行模拟并观察这些设置和所选设置更改的影响之前定义要设置的范围广泛的参数。为了便于管理多个 假设”场景，建议您创建原始配置的副本（作为工件元素和设置更改）并在副本中进行设置。

创建配置变体的A有用功能是应用继承，即您不打算改变的数据和参数保存在一个配置中，而只有您更改的那些参数保存在另一个配置中。 变量”配置使用（继承）基本配置中保存的公共数据，因此您不必在 变量”配置中重新创建该公共数据。

然后，您可以在更改的配置和原始 运行”上运行模拟，并比较模拟报告以查看运行时变量发生了什么差异，然后运行并显示配置的比较以查看参数设置的哪些变化导致了这些运行时差异。

通过在原始配置和副本配置下运行模拟，比较结果和导致结果的更改，并相应地修改模型，您可以在简化您正在开发的业务流程方面实现高度控制。

访问

上下文菜单	在图表或浏览器窗口中|右键点击业务流程仿真工件|显示配置>审阅选项卡>配置摘要选项卡

配置摘要选项卡

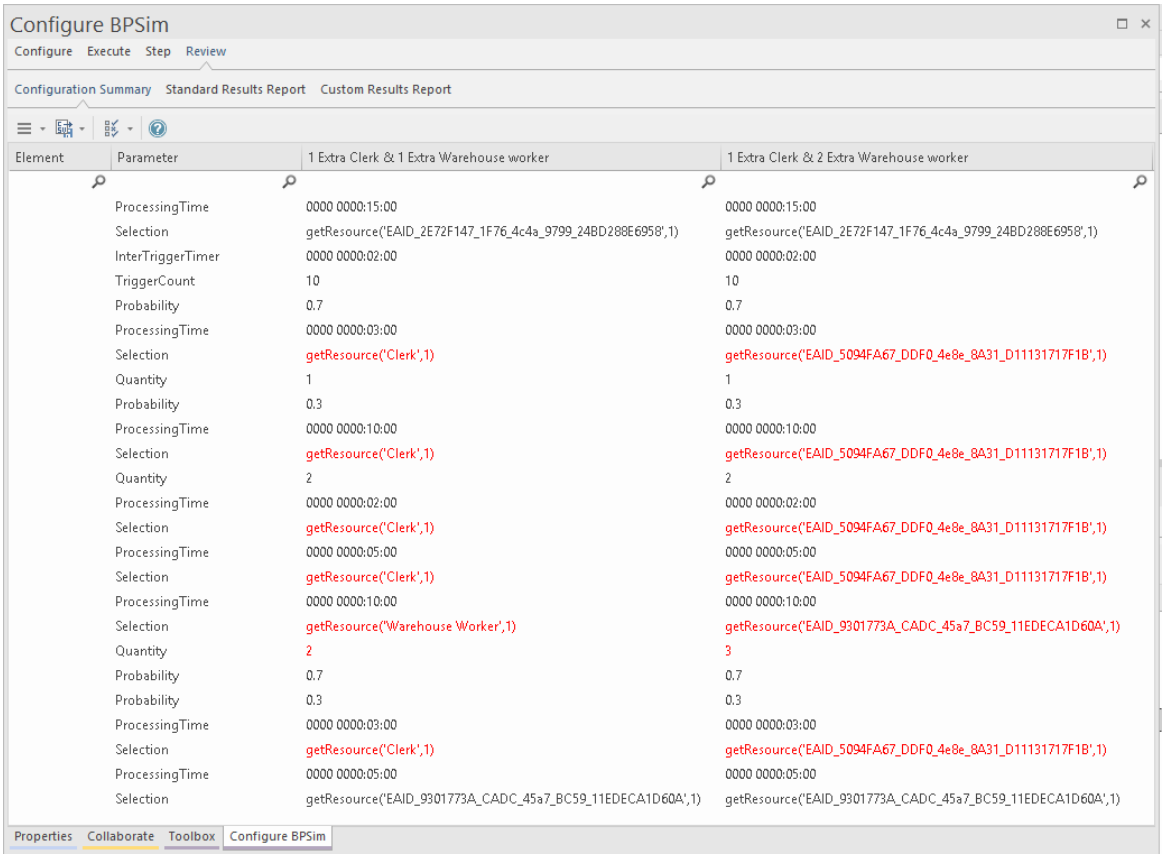
此视图最初显示所选配置中具有值的参数，并在配置名称下的列中显示为这些参数设置的值。如果窗口停靠，您可以通过将选项卡拖离工作区以成为浮动显示，并将显示扩展到全屏大小，从而更轻松地审阅结果。

要比较两个或更多配置，请单击另一个配置工件（或单击另一个浏览器元素中的一个或一个）

- 将其拖到报告视图或
- 选择 显示配置”选项

您还可以在 配置摘要”选项卡上的  图标，单击 添加配置配置”选项并浏览并选择 工件元素”对话框上的 元素”。

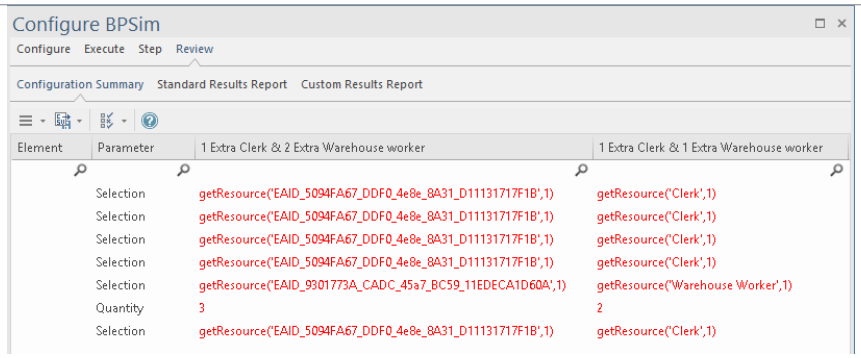
参数层次结构现在包含任何附加配置，并且其参数值显示在原始配置的列中，参数值的名称位于该工件标题的左侧。



您可以使用工具栏和右键单击上下文菜单中的可用选项来审阅和操作报告上的信息。

BPMN仿真配置汇总选项

选项	描述
	选择此工具栏选项以显示选项的简短菜单： • 添加配置- 显示 无素选择”对话框，您可以通过该对话框浏览另一个工件并选择那些已经在页面旁边显示其参数的 BPSim 配置 • 重新加载 - 刷新显示并将列更改返回到原始宽度和位置 • 清除列表 - 从显示中清除选定的工件及其参数 • 删除报告名称；从页面中清除报告名称工件参数为页面上显示的每个工件提供的选项之一
	单击此工具栏图标可将此页面上的记录导出为 CSV 文件或 XML 文件。将显示 另存为”浏览器，您可以从中选择保存文件的位置。
	显示两个选项，用于过滤页面上显示的信息；当显示两个或更多配置时，将启用这两个选项： • 仅显示不同项- 选择此选项将显示限制为仅显示配置之间值不同的参数；再次单击该选项可取消选择 • 突出显示不同项-（当两个或多个配置具有不同的参数值时）以红色显示不同的参数值；如果您已选择 仅显示不同项”选项，则此选项将被禁用


Configure BPSim
Configure Execute Step Review
Configuration Summary Standard Results Report Custom Results Report
Element Parameter 1 Extra Clerk & 2 Extra Warehouse worker 1 Extra Clerk & 1 Extra Warehouse worker
Selection getResource('EAID_5094FA67_DDF0_4e8e_8A31_D11131717F1B',1) getResource('Clerk',1)
Selection getResource('EAID_5094FA67_DDF0_4e8e_8A31_D11131717F1B',1) getResource('Clerk',1)
Selection getResource('EAID_5094FA67_DDF0_4e8e_8A31_D11131717F1B',1) getResource('Clerk',1)
Selection getResource('EAID_5094FA67_DDF0_4e8e_8A31_D11131717F1B',1) getResource('Clerk',1)
Selection getResource('EAID_9301773A_CADC_45a7_BC59_11EDECA1D60A',1) getResource('Warehouse Worker',1)
Quantity 3 2
Selection getResource('EAID_5094FA67_DDF0_4e8e_8A31_D11131717F1B',1) getResource('Clerk',1)

BPSim 图表

图表工具箱的 图表”页面提供了两个图标，专门用于生成反映从图表模拟中选择的结果的图表。这些是：

- 图表结果图表 - 生成反映一系列标准图表模拟的选定结果的图表
- 图表自定义结果图表 - 生成反映一系列自定义图表模拟结果的图表

至于其他维度的工件，图表图表都可以快速配置为以时间折线图或 3 维图表图表变化形式显示模拟结果。

先决条件

要填充从业务流程仿真工件的工件中创建的结果，您可以选择在模拟过程中显示的每个结果创建配置。因此，必须首先执行初始模拟，工件生成报告。

访问

使用本表中工具箱图表 图表”页面。

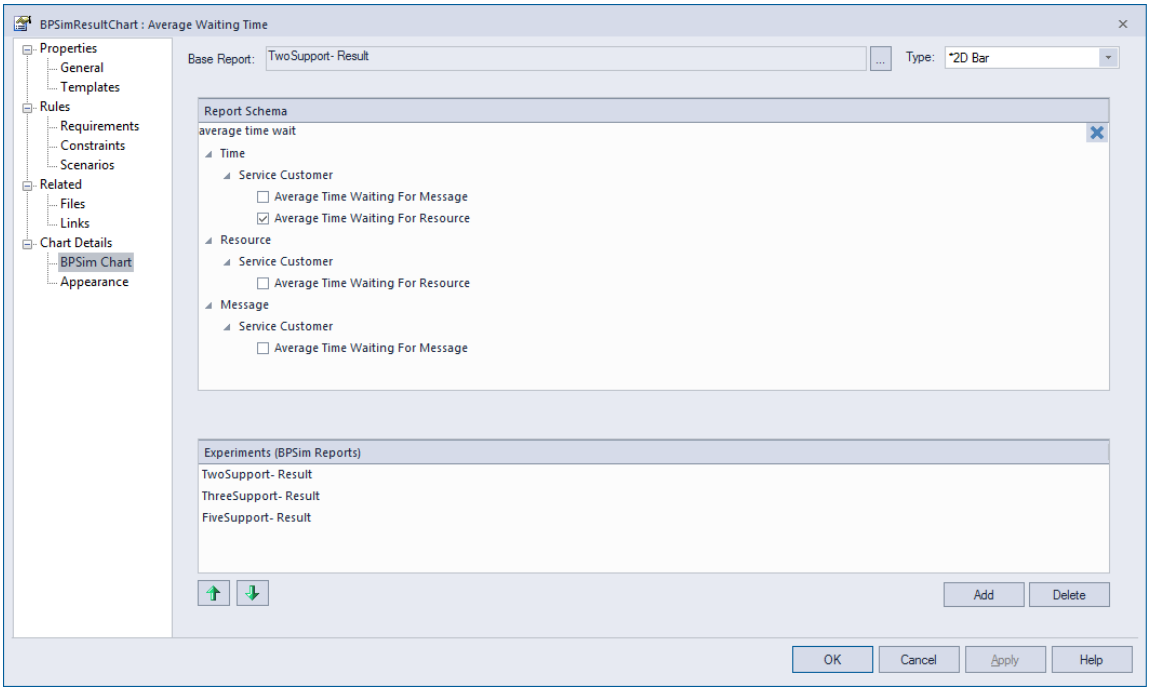
然后，将新的图表<type>工件图标拖到图表上——创建了一个新图表元素图标。

双击新的图形元素打开 属性”对话框，显示 图表图表”页面。

功能区	设计>图表>工具箱> 图表
键盘快捷键	Ctrl+Shift+3 |图表
其它	您可以通过单击 » « 图表工具箱显示或隐藏图形图表视图。

选择要在图表中显示的结果

填写 属性”对话框的 完全图表”页面上的图表。



选项	行动
基础报告	单击 在<工件>>报表 ... 上选择”按钮显示 选择报表工件”对话框，并选择其他模拟结果将与之比较的模拟结果的报表。该结果作为 实验”工件中的第一个添加到要比较的报告结果列表中
类型	单击下拉箭头并选择要显示结果的图表的格式类型 - 2D Bar、3D Bar 或 Line。指定要比较的报告参数后，您可以选择对话框的 外观”页面并定义图表或时间线图的外观。
报告架构	根据需要展开层次结构并选中每个属性的复选框以显示在图表上。每个属性将由图表上的单独一行或一组图表表示。通常你会为每个object选择相似的对象（比如不同的资源）和相同的单个属性（比如资源的利用率）。你有大量的属性可以检查和比较，但是在同一个图表上的任何一个以上都会使图表难以阅读。
实验（BPSim 报告）	此模拟报告结果（如工件Result 面板将列出使用您选择的要比较的图表的模拟报告结果），在图表中显示它们选择的图表的序列通常，基础报告在列表中保持在第一位，其参数的结果显示在图表的左侧。如果要更改序列，请单击结果/工件名称并单击相应的向上向下绿色箭头按钮。 要添加结果名称，请单击 将结果名称添加到列表”工件从 <工件>> 选择工件”对话框中选择更多按钮。

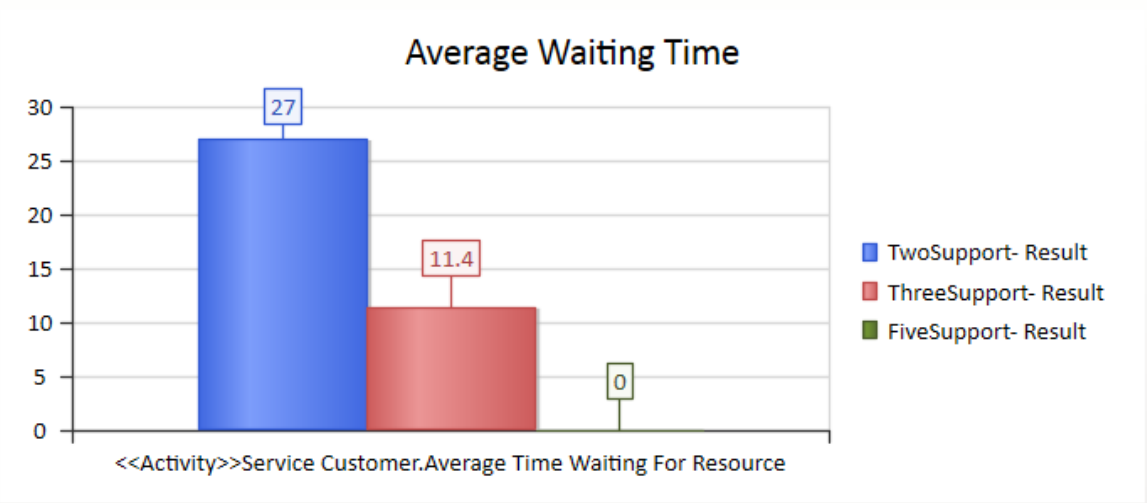
A图表示例

在电话支持示例中，我们创建了三个服务台工件来分析 我们需要多少支持资源才能以经济的方式通过电话回答客户的问题？”

我们从两个支持资源开始，然后尝试了三个和五个资源。模拟后，我们有一个基于不同配置的 BPSim 报告：TwoSupport-Result、ThreeSupport-Result 和 FiveSupport-Result。

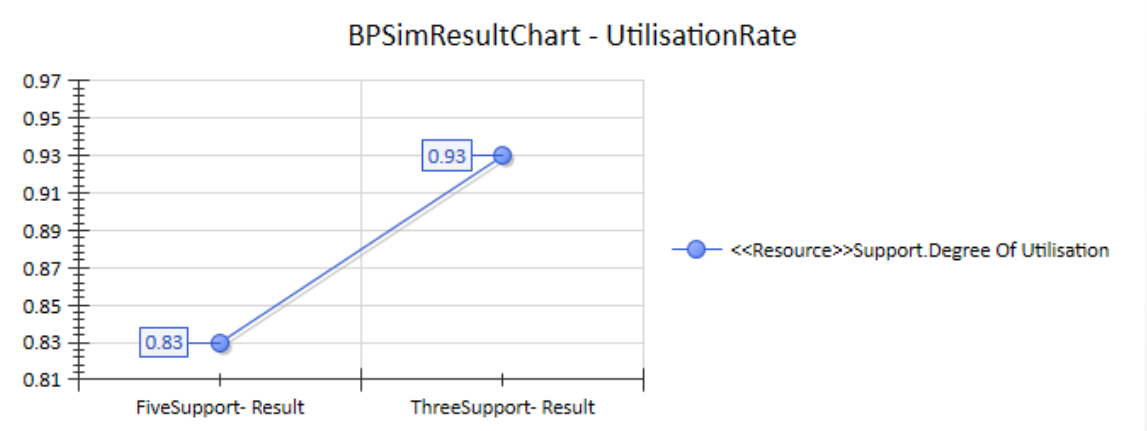
以下是创建图表以比较客户等待支持的平均时间的步骤：

1. 在图表上创建一个图表结果图表，并将其命名为平均等待时间。
2. 双击 Figure 打开属性窗口，然后打开 图表图表”选项卡。
3. 单击[...]按钮选择一个基本报告，我们从中定义要在图表中使用的模式（图例）。选择 双支持结果”。
4. 选择此架构：
   -时间|服务顾客|平均时间等待资源
5. 单击 Add 按钮添加另外两个 BPSim 报告：“ThreeSupport-Result”和 FiveSupport-Result”。
6. 单击确定按钮，然后调整图表元素的大小。这个图表给了我们直接的信息。



这些是创建一个图表来比较支持的使用程度的步骤：

1. 在图表上创建一个图表Result Figure，并将其命名为*Utilization Rate*。
2. 双击 Figure 打开属性窗口，然后打开 图表图表”选项卡。
3. 单击[...]按钮并选择一个 Base Report，从中定义要在图表中使用的模式（图例）。选择 双支持结果”。
4. 选择此架构：
   - 资源 |支持 |利用度
5. 单击 Add 按钮添加另外两个 BPSim 报告：“ThreeSupport-Result”和 FiveSupport-Result”。
6. 单击确定按钮并调整图表元素的大小。该图表提供了具体信息。



BPSim 示例

本节提供了几个 BPMN 建模、配置和仿真结果分析的示例。

这些示例都可以从 EAExample模型中访问。

要运行示例模拟，您必须安装运行执行引擎和Java运行时环境(JRE) 1 .7 或更高版本。要使用属性参数，您还必须安装Java开发工具包 (JDK) 版本1 .7 或更高版本。

订餐协作版本1

在此示例中，我们创建了一个非常简单的模型来模拟客户与餐厅之间就餐单进行的通信。

对于客户的流程：

1. A向餐厅发送消息以订餐。
2. 客户将等待交货。
   如果60分钟内没有送达，他们会打电话给餐厅，然后继续等待。
3. 交货后，客户将享用晚餐。

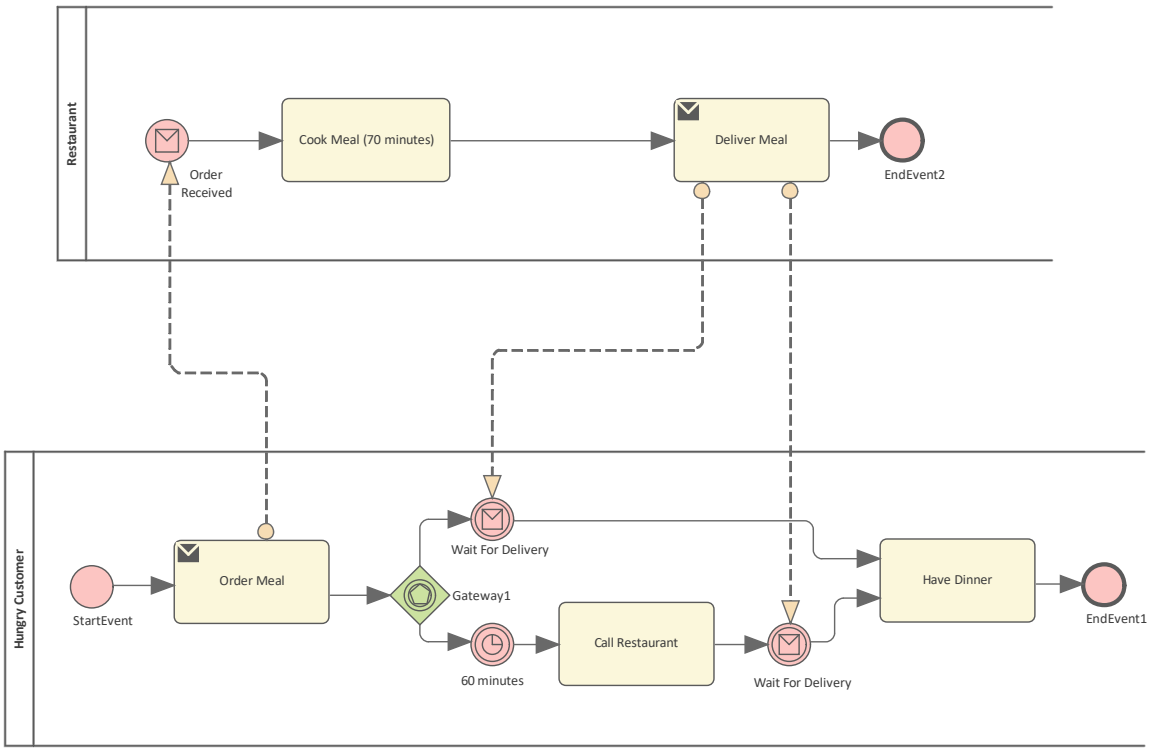
对于餐厅的流程：

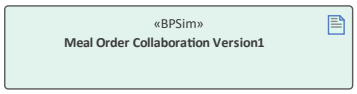
1. 该过程从收到客户的订餐单开始。
2. 烹饪时间可由用户设定。这允许用不同的事件时间进行实验，例如 30 分钟、70 分钟
3. 餐厅送餐并完成流程。

创建 BPMN模型

要设置可用于此 BPSim 模拟的 BPMN模型，您：

- 创建一个带有 2 个池的协作模型
- 在每个泳池中为每个过程创建一个元素
- 将元素与流程通信的信息流连接起来
- 包括用于设置模拟细节的 BPSim工件。

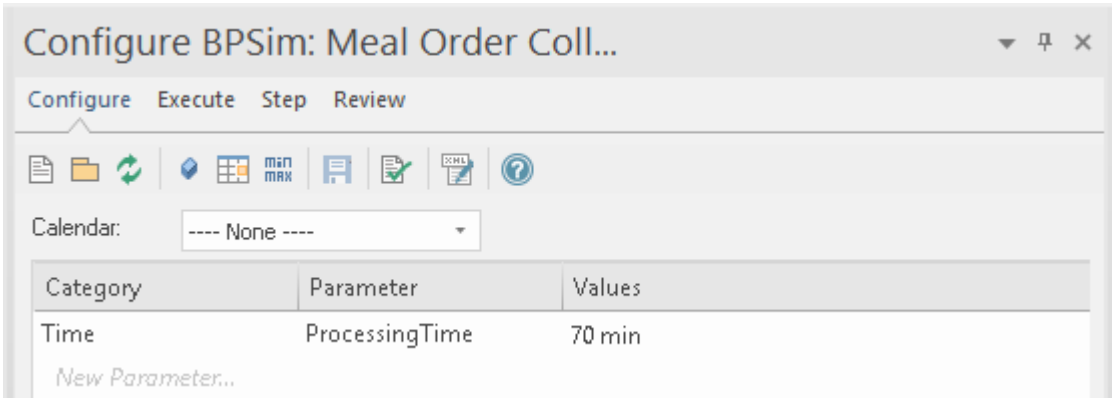




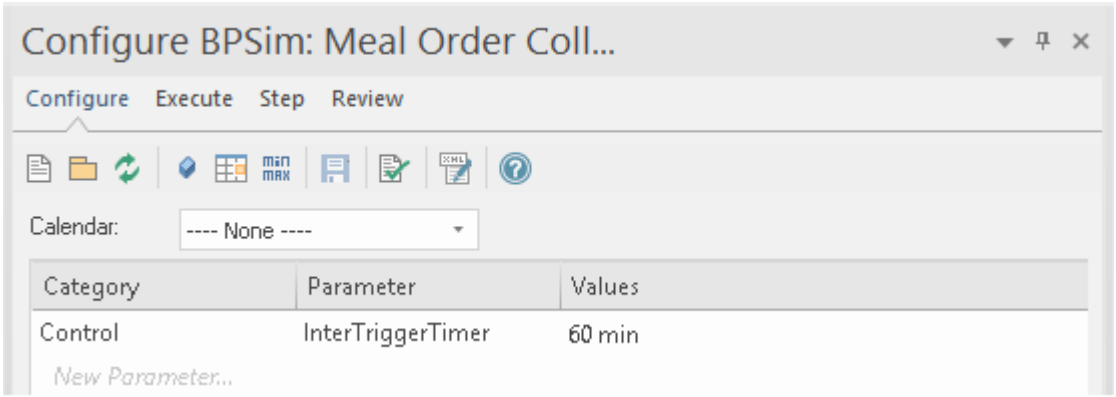
配置 BPSim

在此示例中，我们配置以下 BPSim 参数：

- 设置饿了么顾客中StartEvent的TriggerCount为1
- 将 Cook Meal 的 ProcessingTime 设置为 70 分钟



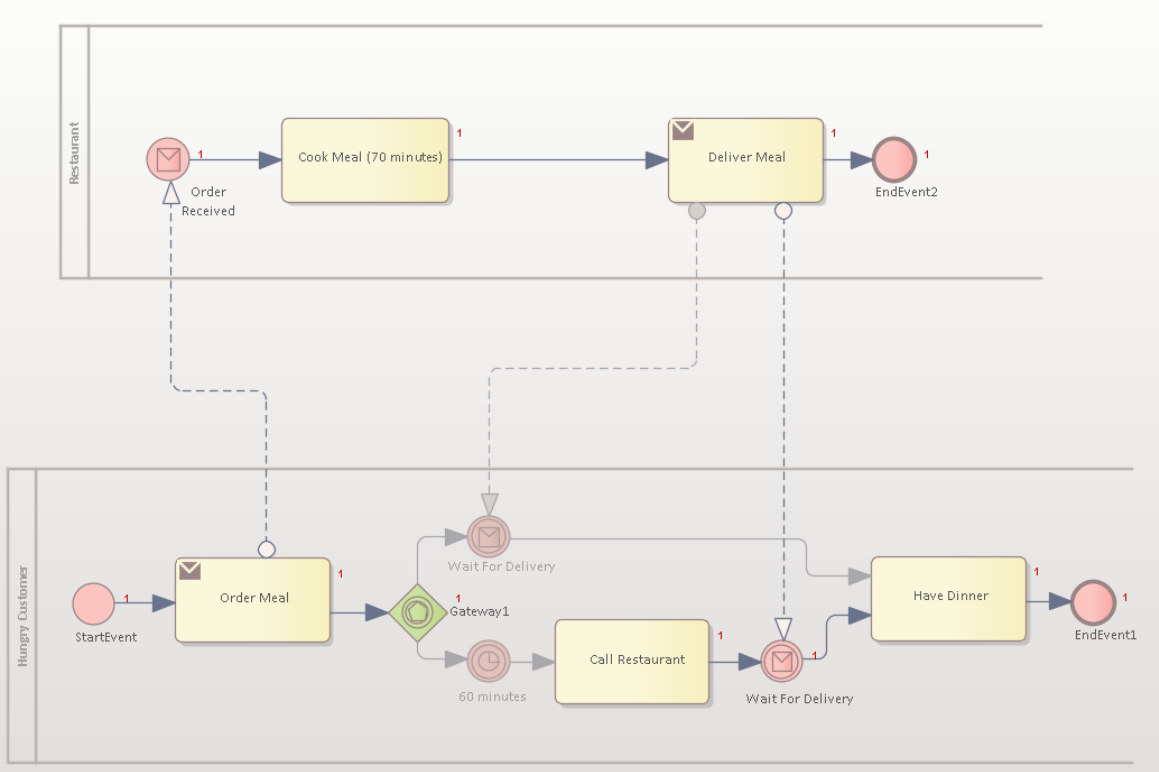
- 将中间事件的 InterTriggerTimer 设置为 60 分钟



- 其他BPSim参数的默认设置，这里是一个配置列表，你可以查看through审阅>配置Summary

仿真

确保 Config BPSim 窗口已打开（仿真>进程分析> BPSIM > 打开 BPSIM 管理器）。
导航到执行选项卡并运行标准仿真：



当 Cooking Meal 任务耗时 70 分钟时，独家事件网关由 60 分钟计时器事件触发。

如果我们将 BPSim 设置更改为任务：烹饪膳食：处理时间从 70 分钟到 30 分钟，消息事件*Wait For Delivery*将触发 Exclusive事件Gateway，并且根本不会激活调用餐厅任务。

订餐协作版本2

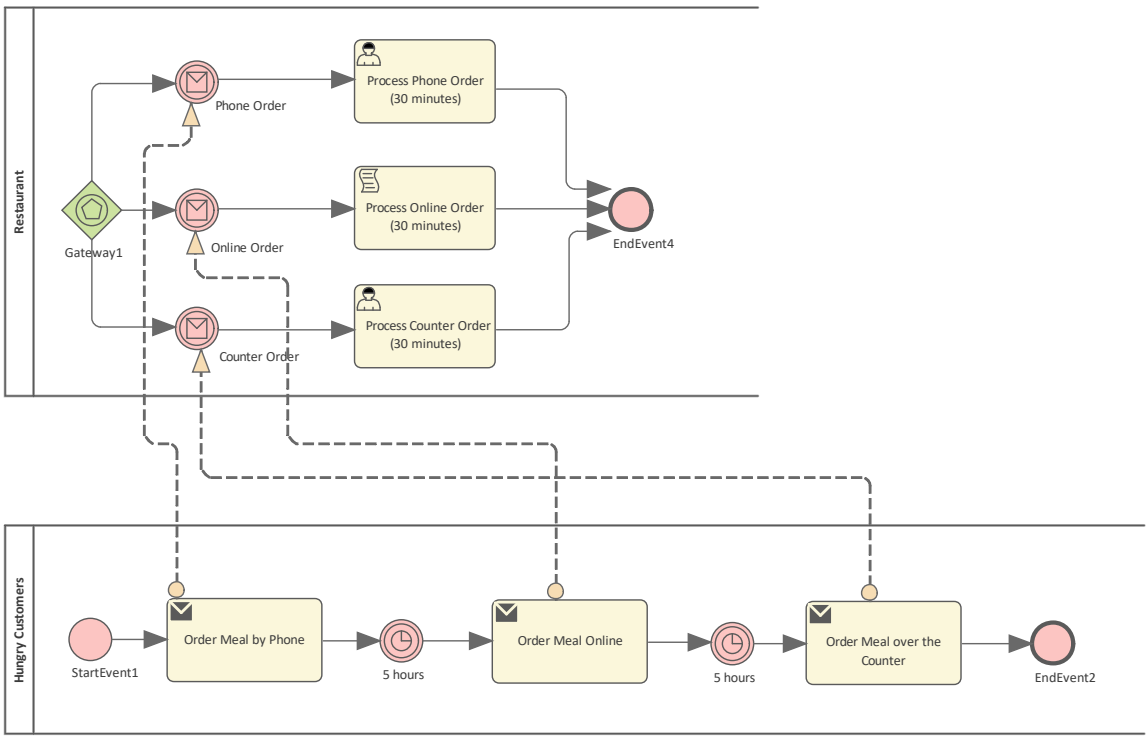
在此示例中，我们创建了一个模型来模拟客户与餐厅之间就餐单进行的通信。客户通过电话、在线和柜台下三种不同的订单。

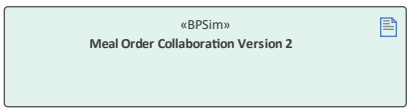
创建 BPMN模型

要设置可用于此 BPSim 模拟的 BPMN模型，您：

- 创建一个带有 2 个池的协作模型
- 在每个池中为每个进程创建一个元素
- 创建描述进程通信的信息流
- 包括用于设置模拟细节的 BPSim工件。

请注记： Event-Exclusive gateway 的*instantiate*属性被设置为**true** ，这意味着进程是由这个开始事件*Gateway*启动的。

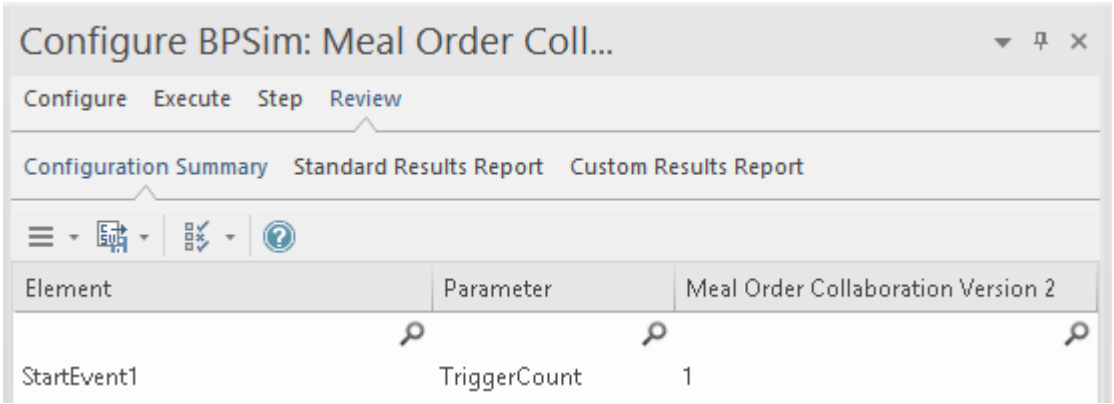




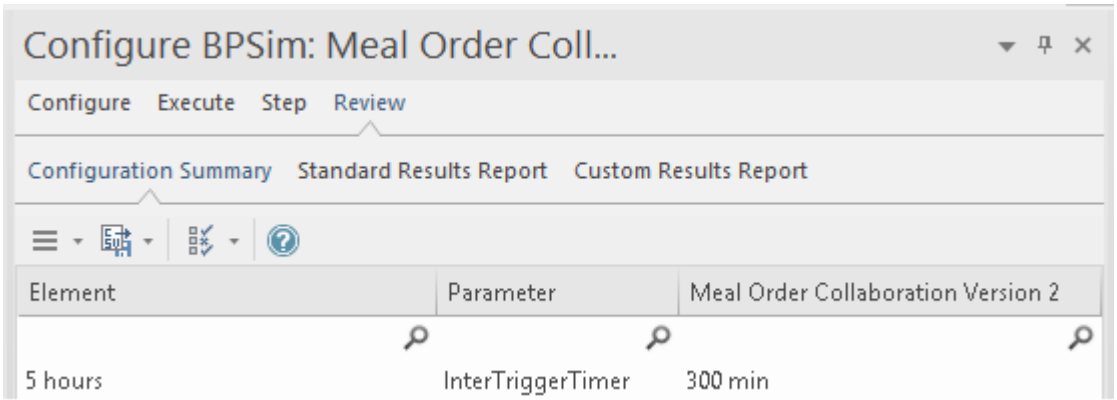
配置 BPSim

在此示例中，我们配置以下 BPSim 参数：

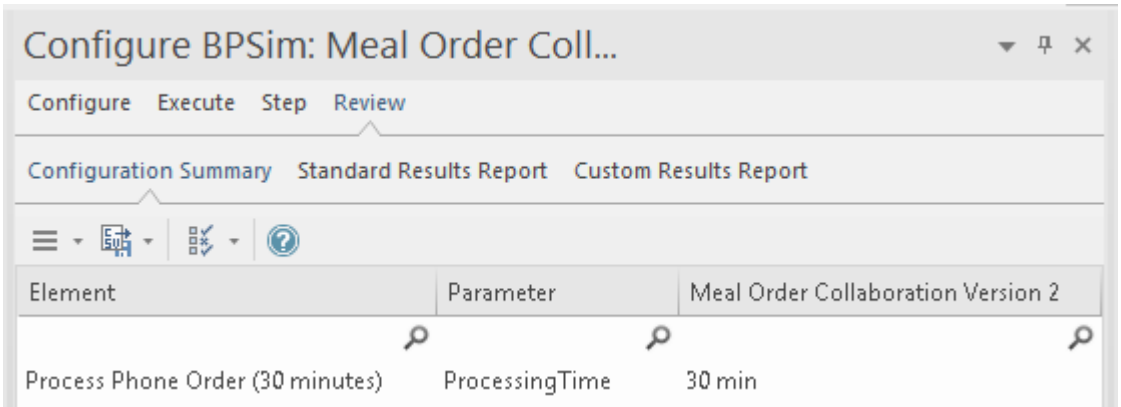
- 对于开始事件，将参数 TriggerCount 设置为值1



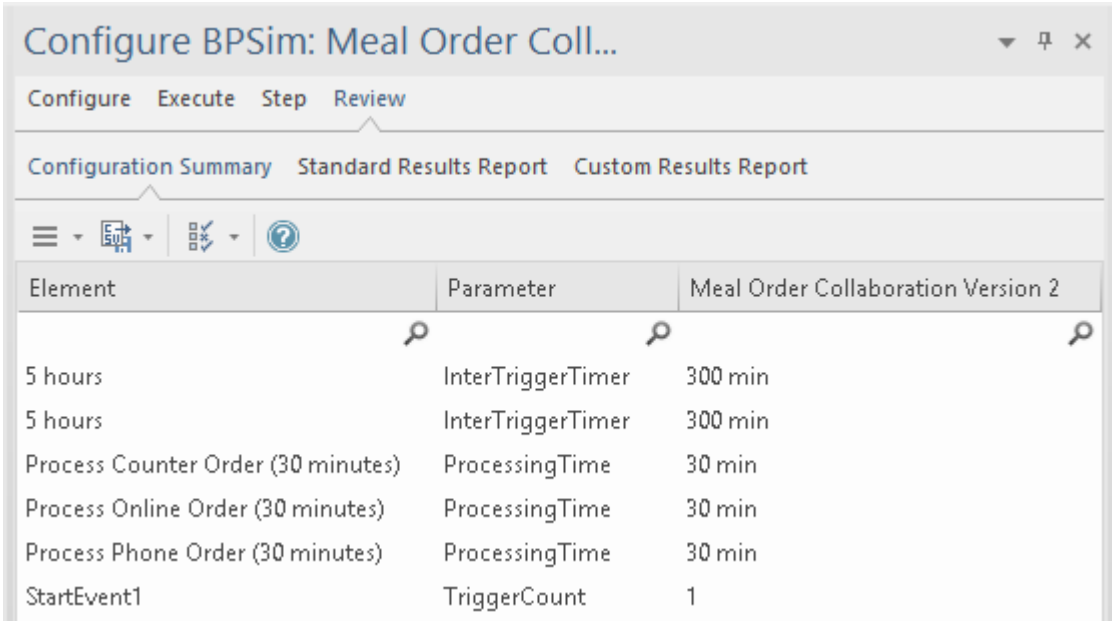
- 对于中间事件中的两个 5 小时，将参数 InterTriggerTimer 设置为值 300 分钟



- 对于 Restaurant 中的三个任务，将参数 ProcessingTime 设置为值 30 分钟

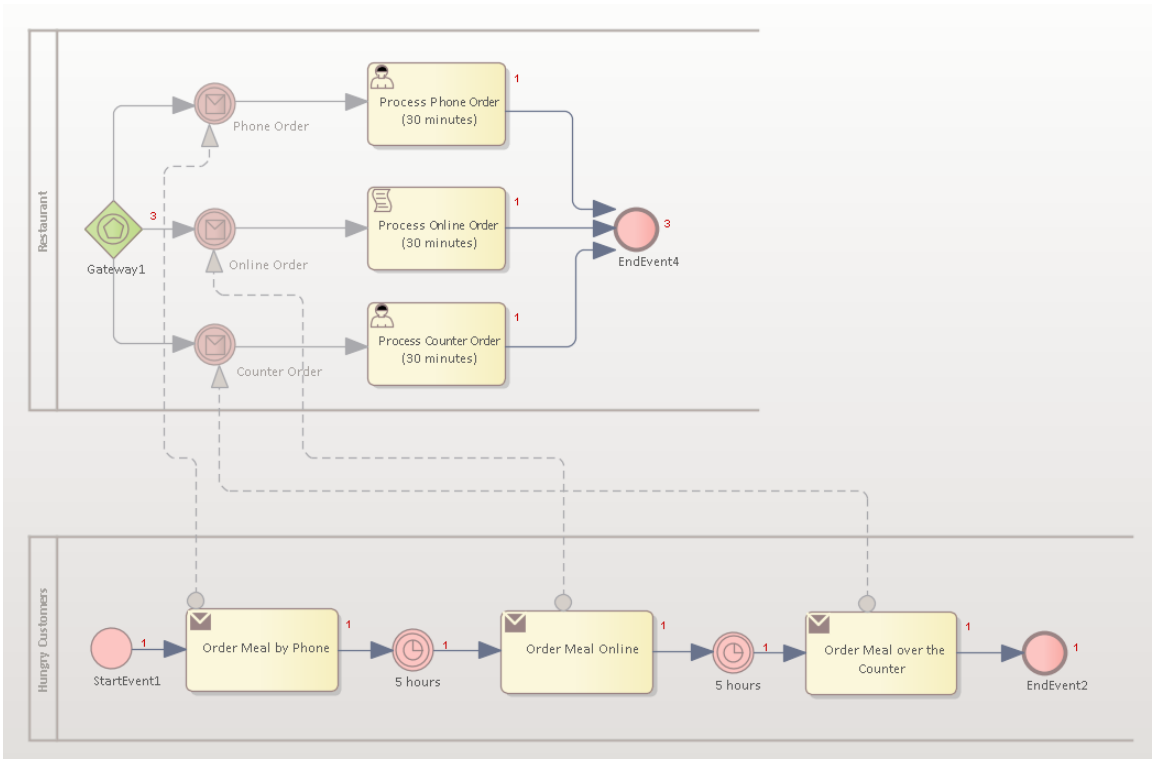


- 其他BPSim参数的默认设置，这里是一个配置列表，你可以查看through审阅>配置Summary



仿真

导航到执行选项卡并运行标准仿真：



*Restaurant*进程被 Event-Exxlusive 网关通过 3 条不同的消息实例化了 3 次。

服务台电话支持仿真

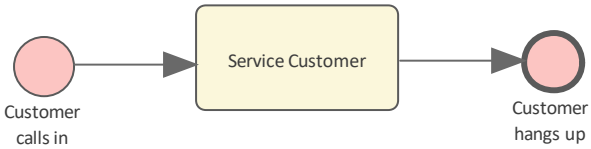
在本例中，我们创建了一个非常简单的模型来模拟服务台电话支持过程。

我们设置了一个资源有限的场景，并且必须将请求放入等待队列中以获取资源。然后我们尝试使用假设分析在客户的等待时间和资源数量之间寻求平衡点。

首先，我们一步一步地模拟这个过程，从一个简单的参数设置开始，而不是用笔和纸计算，然后用模型进行验证。之后，我们会进行假设分析，以帮助经理做出决定。

创建 BPMN模型

模型本身很简单，由一个开始事件、一个任务和一个结束事件组成。



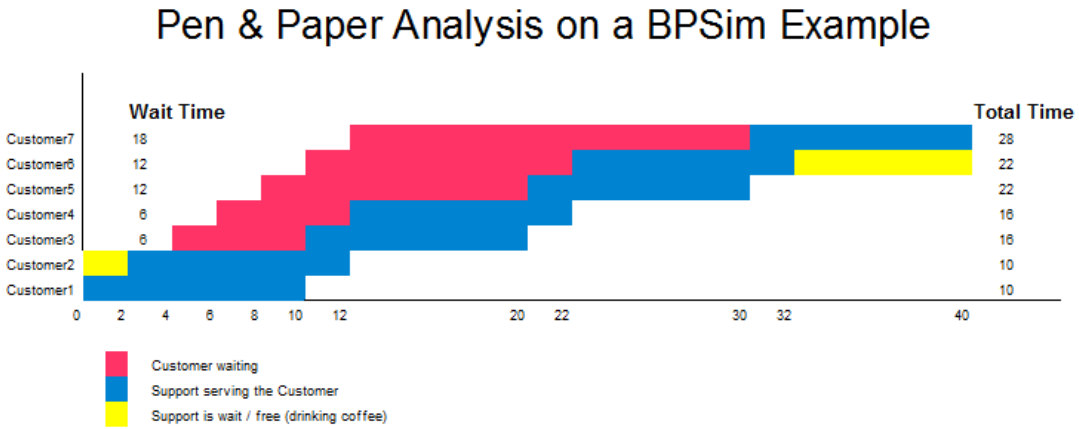
- 创建一个名为 顾客" 的开始事件
- 将序列流添加到名为*Service*顾客的目标抽象任务
- 将一个序列流添加到一个名为 顾客挂断"的目标结束事件

创建一个名为*Support* 的 *BPMN2.0* 资源；该元素将在配置中使用。

笔和纸分析

我们将用纸笔来分析这个案例：

- 7 位客户每 2 分钟打来一次电话
- 2个支持资源可用
- 每次服务需要 10 分钟



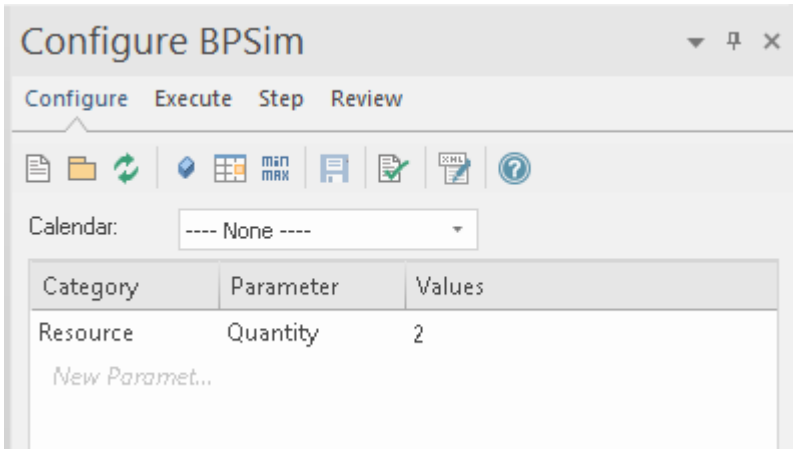
Now we can perform some calculations using a pen and paper based approach:

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%	

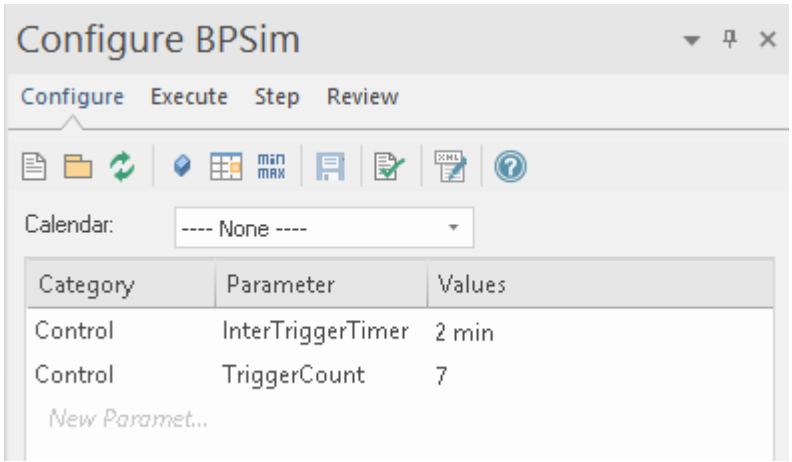
从这个结果可以看出，对于这样一个简单的模型，在应用资源约束的情况下，已经是一个非常复杂的计算了。当流程扩展并应用更多约束时，用笔和纸分析流程将很快变得不可能。我们将演示 BPSim 如何提供帮助。

配置

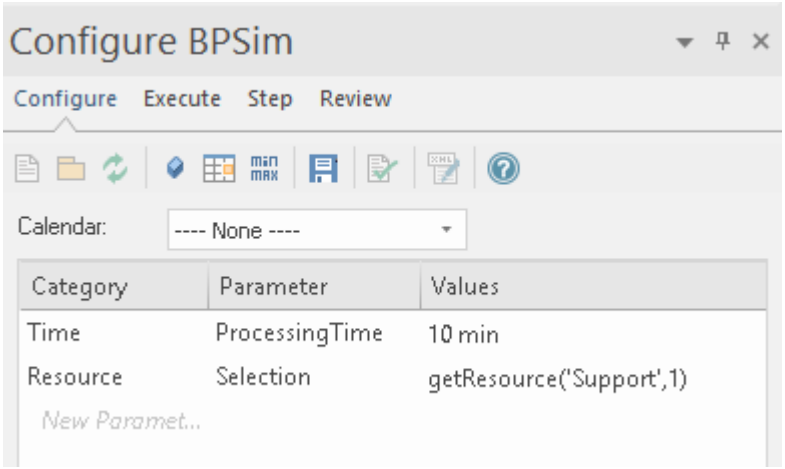
1. 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'）。
2. 单击 业务  "图标和 加新业务流程"按钮，并在命名为 已命名的工件*& Paper*分析客户"的 7上创建业务流程仿真。
3. 单击  图标并浏览并选择包含相应 BPMN 2.0模型的包。
4. 打开模型图并单击名为 支持"的资源元素。
5. 在窗口的 类别"列中，单击 新参数"下拉箭头并选择 资源"，然后单击 参数"下拉箭头并选择 数量"，然后在 值"中字段类型 2"。
6. 单击  图标。



1. 在图中，单击 顾客呼入"开始事件元素。
2. 在窗口的 类别"列中单击 新参数"下拉箭头并选择 控件"。
3. 单击 参数"下拉箭头并选择 InterTriggerTimer"。
4. 在 Value"字段中，单击 ... 按钮，选择 Constant"选项卡和 Numeric"，在 Constant Numeric"字段中输入 "2"并在 TimeUnit"字段中选择 minutes"，然后单击确定按钮。
5. 重复步骤 2 和 3，在 参数"字段中选择 TriggerCount"，并在 值"字段中键入 7"。
6. 单击 图标。

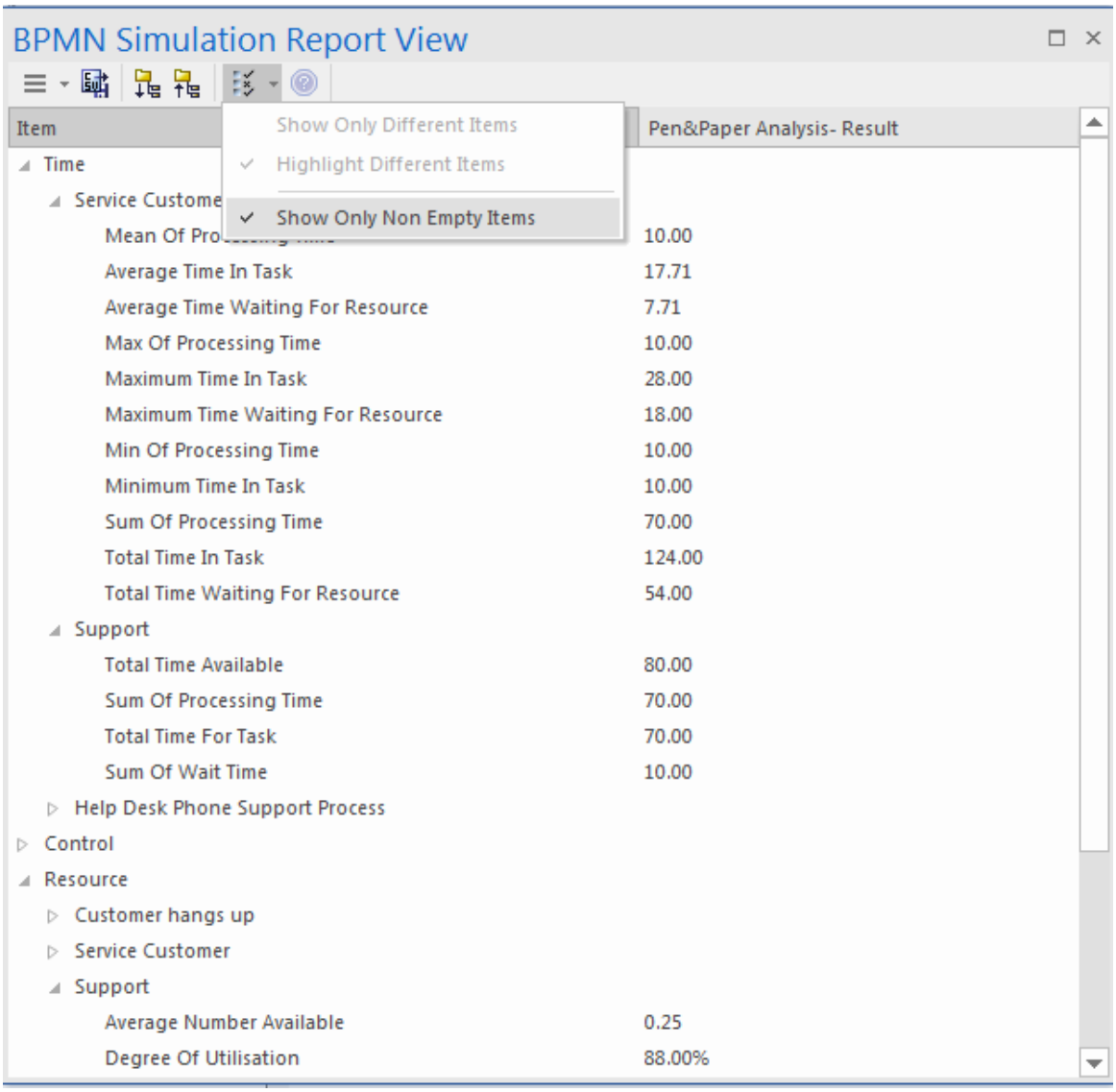


1. 在图中，单击 服务顾客"活动元素。
2. 在窗口的 类别"列中，单击 新参数"下拉箭头并选择 时间"。
3. 单击 参数"下拉箭头并选择 ProcessingTime"。
4. 在 Value"字段中，单击 ... 按钮，选择 Constant"选项卡和 Numeric"，在 Constant Numeric"字段中输入 "10"并在 TimeUnit"字段中选择 minutes"，然后单击确定按钮。
5. 在窗口的 类别"列中，单击 新参数"下拉箭头并选择 资源"。
6. 单击 参数"下拉箭头并选择 选择"。
7. 在 值"字段中，单击 ... 按钮，选择 支持"（您在元素中创建的资源的模型）并单击 按资源添加选择"按钮 将 支持"移动到 "资源或角色"列。
8. 在 需要数量"栏中输入1 。
9. 点击确定按钮。在 值"字段中，将显示自动生成的表达式*bpsim:getResource('Support', 1)* 。
10. 单击 图标。



运行仿真

1. 在配置 BPSim窗口中，单击 执行"选项卡和工具栏中的 图标。
2. 模拟完成 完成以打开 结果"选项卡的 配置摘要"选项卡



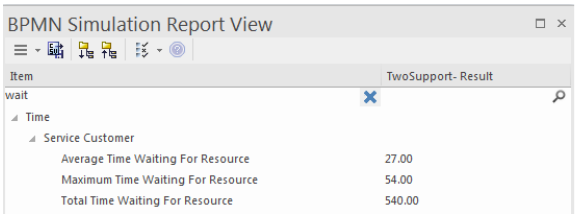
结果与笔和纸分析相匹配。

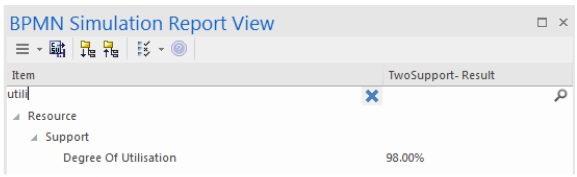
仿真- 2个支持资源为20个客户

您可以通过复制现有的工件配置来创建新的业务流程仿真工程师。复制*Pen &*分析客户元素并按 Ctrl+Shift+V 粘贴，将新元素命名为*TwoSupport* 。

1. 双击*TwoSupport*打开配置 BPSim窗口；你可以看到所有的配置都保留在复制的源中
2. 在图中，单击 顾客呼入"开始事件元素。
3. 在窗口的 类别"列中，单击 控件"- 触发器计数"的 值"字段，然后将值更改为 20"。
4. 单击 图标。

运行仿真并分析结果





从报告中，您可以看到：

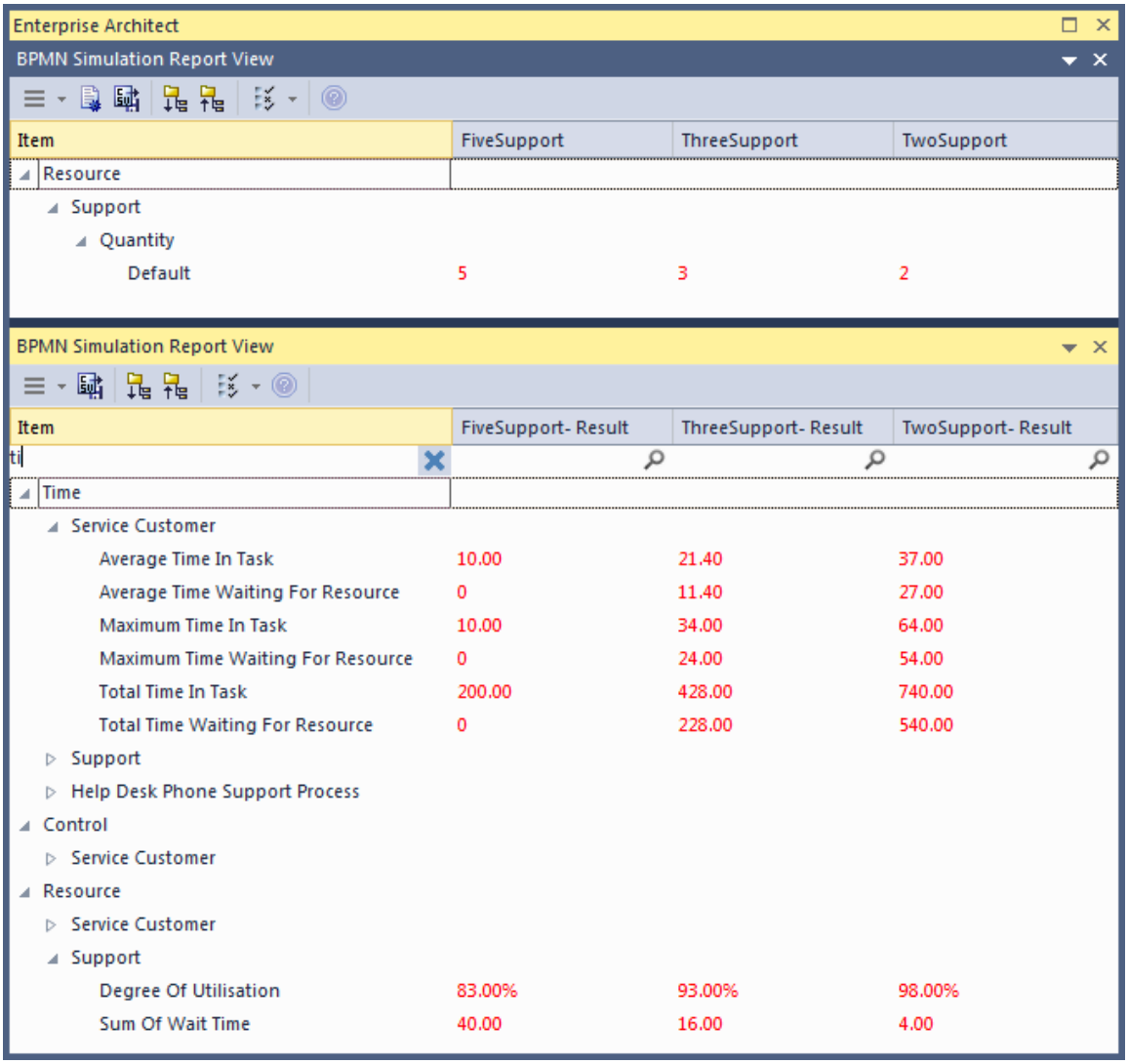
- 平均等待资源时间"为 27 分钟， 最大时间等待资源"为 54 分钟
- 两个支持资源 - 他们忙吗？如果他们不是，我们可能不得不改变流程以使用他们所有的时间并减少客户的等待时间；但是， 利用率"为 98%，这表明资源几乎没有空闲时间

假设" l有更多员工？比较 2 支持资源与 3 和 5 支持资源

1. 复制*TwoSupport*并按 Ctrl+Shift+V 进行粘贴，将新元素命名为*ThreeSupport*。
2. 双击*ThreeSupport*打开 配置 BPSim "对话框。
3. 在图中，单击 支持"资源元素。
4. 在配置 BPSim窗口中，在 资源"- 数量"的 值"字段中，键入 3"。
1. 再次复制*TwoSupport*并按 Ctrl+Shift+V 进行粘贴，将这个新元素命名为*FiveSupport*。
2. 双击*FiveSupport*以打开 配置 BPSim "对话框。
3. 在图中，单击 支持"资源元素。
4. 在配置 BPSim窗口中，在 资源"- 数量"的 值"字段中，键入 5"。

运行模拟并进行比较；在浏览器窗口中：

1. Ctrl+单击*TwoSupport* 、 *ThreeSupport*和*FiveSupport* ，然后右键单击并选择 显示 BPSim配置"选项。
2. Ctrl+单击*TwoSupport-Result* 、 *ThreeSupport-Result*和*FiveSupport-Result* ，然后右键单击并选择 显示 BPSim报告"选项。



提示：

- 单击 按钮和两个视图的 仅显示不同项”选项
- 您可以将视图停靠在一起，以便它们提供直接比较：这些是由配置中的这些差异引起的结果差异
- 切换过滤器栏以过滤您感兴趣的项目

分析

停靠的比较视图显示配置差异和相应的结果差异。

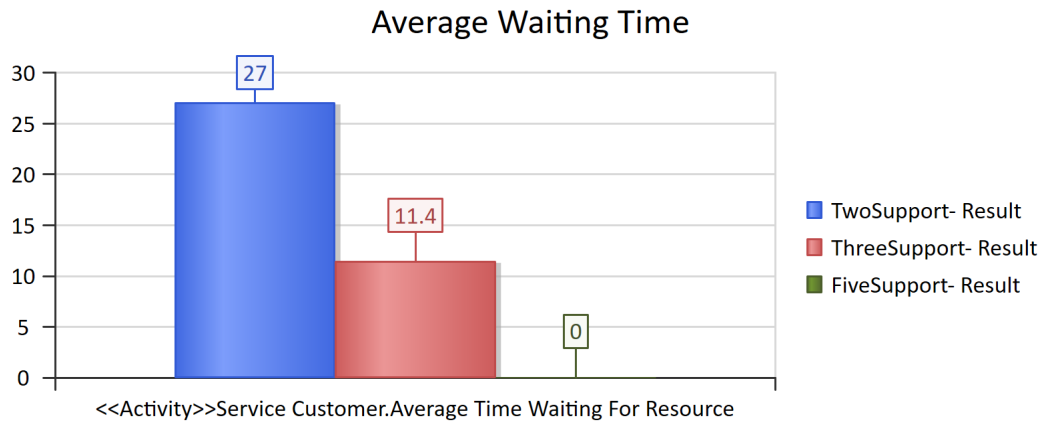
- 客户的等待时间从 27 分钟（2 个支持资源）下降到 11.4 分钟（3 个支持资源）并进一步下降到 0 分钟（5 个支持资源）
- 使用度”从 98%（2 个支持资源）下降到 93%（3 个支持资源）并进一步下降到 83%（5 个支持资源）

客户很可能会对 5 个支持资源感到满意；但是，成本可能超出预算。因此，3 个或可能 4 个支持资源可能是这种情况下的平衡点。尝试复制业务流程仿真工件和模拟配置和运行支持资源之一。

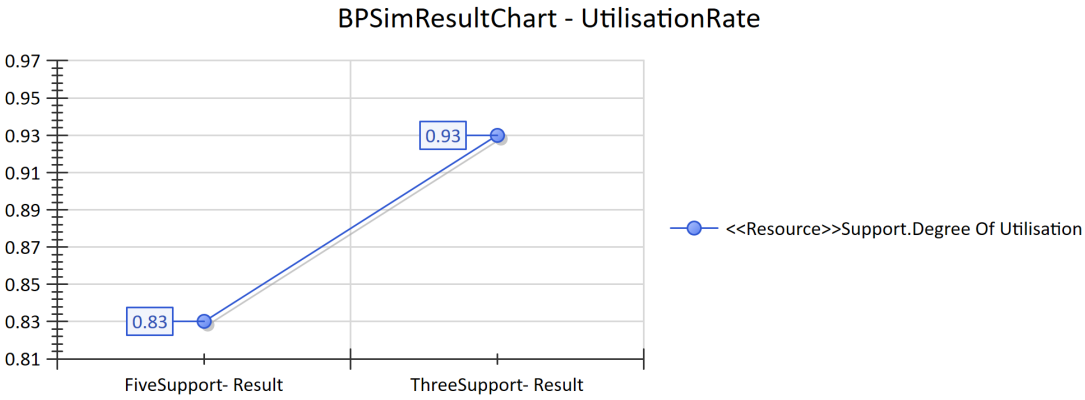
用图表显示结果

1. 将工具箱中的图表Result Figure' 图标拖到工具箱中，然后在工具上创建图表Result工件；称之为平均等待时间。

2. 右键单击工件并选择 属性”选项以元素 属性”对话框；点击 图表图表”页面。
3. 单击…按钮并选择一个基本报告，从中定义要在图表中使用的架构（图例）；选择*TwoSupport*-结果。
4. 选择模式 时间” | '服务顾客' | '平均等待资源的时间'。
5. 单击添加按钮添加另外两个 BPSim 报告： *ThreeSupport-Result*和*FiveSupport-Result*
6. 点击确定按钮，调整图表元素的大小；这个图表给了我们非常直接的信息



1. 在工件上创建另一个图表结果显示，称为*Utilization Rate*。
2. 双击属性以显示元素对话框，然后单击 工件图表”选项卡。
3. 单击…按钮并选择一个 Base Report，从中定义要在图表中使用的模式（图例）；选择*TwoSupport*-结果。
4. 选择架构 资源”| '支持' | 利用程度”。
5. 单击添加按钮添加另外两个 BPSim 报告： *ThreeSupport-Result*和*FiveSupport-Result*。
6. 单击确定按钮并调整图表元素的大小。



基于日历的服务台电话支持仿真

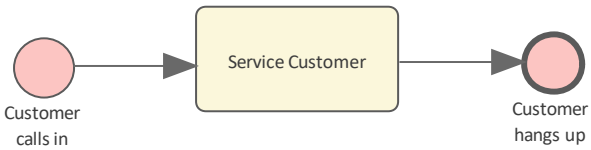
在这个例子中，我们创建了一个非常简单的模型来模拟服务台电话支持过程，基于日历设置。我们假设：

- 客户在工作日和周末以不同的时间间隔致电
- 工作日和周末的处理时间不同
- 工作日和周末有不同数量的支持资源

我们模型一步地模拟这个过程，然后创建日历并配置业务流程模拟，简单到可以用纸笔计算。之后，我们运行模拟以将该结果与笔和纸分析进行比较。

创建 BPMN模型

模型本身很简单，由一个开始事件、一个任务和一个结束事件组成。

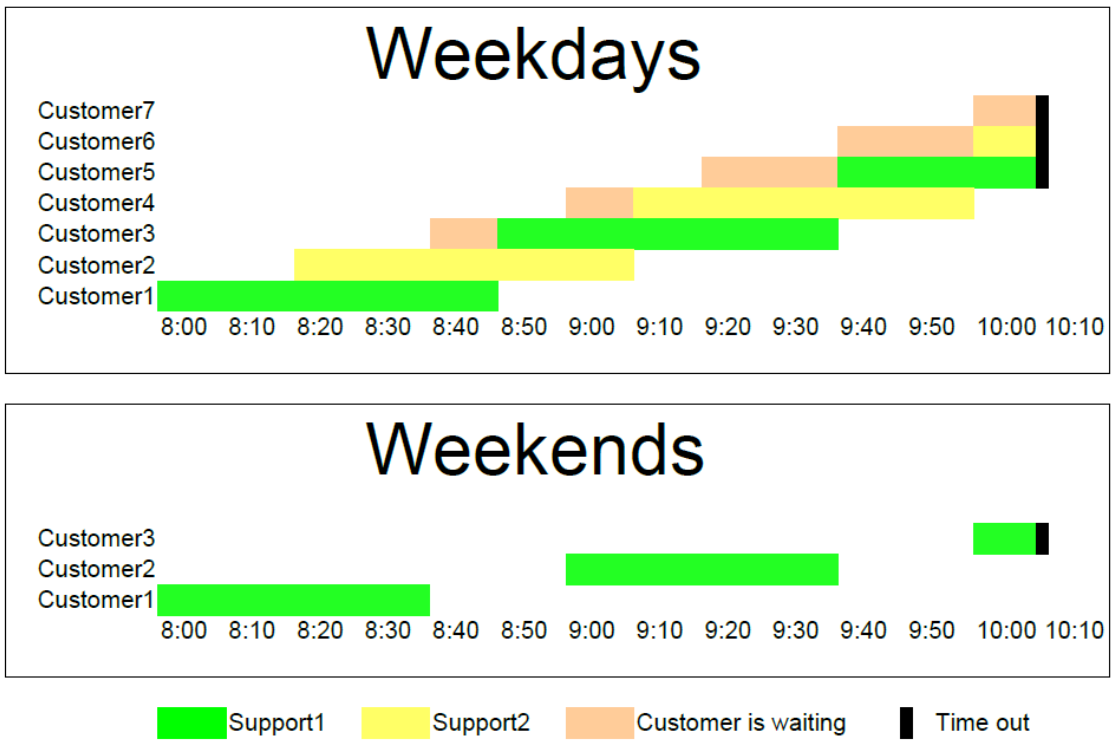


1. 创建一个开始事件顾客呼入。
2. 添加一个序列目标抽象任务活动服务顾客。
3. 将一个序列流添加到目标结束事件的顾客挂断。
4. 创建一个名为*Support* 的 *BPMN2.0* 资源。
5. 在*Service*顾客中创建一个元素::ResourceRole，将其命名为*support*并将标签*resourceRef*设置为 Resource *ElementSupport*的名称。

笔和纸分析

我们可以用纸笔来分析这个案例：

- 模拟时间为 2 小时 10 分钟，从上午 8:00 到上午 10:10
- 工作日每 20 分钟就有A顾客打来电话
- 周末每 60 分钟就有A顾客打来电话
- 平日为每位客户服务需要 50 分钟
- 周末为每位顾客服务需要40分钟
- 平日有2个支持资源
- 周末有1支持资源



从这个结果来看，当应用资源限制时，对于这样一个简单的模型，计算是相当复杂的。

在工作日

- 7 位客户在 2 小时 10 分钟内每隔 20 分钟打一次电话
- 4个客户电话正常挂断
- 2 客户通话因超时而中断
- 1客户电话无人接听
- Support1连续工作130分钟，Support2连续工作110分钟

在周末

- 3 位客户在 2 小时 10 分钟内每隔 60 分钟致电一次
- 2个客户电话正常挂断
- 1客户通话因超时而中断
- Support1 工作了 90 分钟，每 40 分钟一次，通话间隔 20 分钟

现在我们将了解 BPSim 如何提供帮助。

配置

在本节中，我们首先创建日历，然后设置持续时间和开始参数。

对于元素参数，您可以为给定参数指定一个或多个日历。但是，**如果为参数值设置了任何日历，则必须存在默认值（未指定任何日历）**，否则模拟将无法工作。

单击配置 BPSim窗口工具栏上的  按钮将自动为您检查此约束。

任务	行动
创建工件并设置包	1. 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'）。

	2. 创建一个业务流程仿真辅助工件，命名为基于进程的流程仿真。 3. 选择包含对应BPMN 2.0模型的包。 4. 打开包含要模拟的模型的图表。
日历	1. 在配置 BPSim窗口的 "配置"选项卡上，单击工具栏中的  图标。将显示 "编辑 BPSim 日历"对话框。 2. 单击 "新建"按钮以显示 "事件重复"对话框并按照此处所述完成字段，以创建日历。（您将创建两个日历。） 3. 在 "事件时间"面板中，将 "开始"设置为上午 08:00，将 "结束"设置为下午 5:00。 4. 在 "重复模式"面板中选择 "每周"并选中从 "星期一"到 "星期五"的复选框。 5. 在 "重复范围"面板中，将 "开始"设置为 "2020 年 2 月 11 日"，然后选择 "无结束日期"。 6. 点击确定按钮。系统会提示您输入日历名称；用 "工作日"改写*Calendar_1*，然后单击确定按钮。 7. 再次单击新建按钮并使用这些值重复步骤 3 到 6： - '开始' - 08:00AM - '结束' - 05:00PM - "每周" - '周六和周日' - "开始"到 "2020 年 7 月 11 日"和 "无结束日期" - 用 "周末"改写*Calendar_2* 8. 点击确定按钮。
期间	在图表上，单击 "基于仿真进程流程配置 BPSim工件窗口的 "配置"选项卡上，将 "日历"字段设置为 "----无----"，创建或编辑此场景参数： • 持续时间 - 常量值为 0000 002:10:00，表示 0 天 2 小时 10 分钟
顾客到来	在图表上，单击 StartEvent 中的顾客调用，然后在配置 BPSim配置的 "配置"选项卡上，创建或编辑控件参数： • InterTriggerTimer - 值：0 00:00:00， "日历"字段设置为 "----None----"（此默认值是必需的） • InterTriggerTimer - 值：0 00:20:00， "日历"字段设置为 "工作日" • InterTriggerTimer - 值：0 01:00:00， "日历"字段设置为 "周末"
处理时间	在图表上，单击服务顾客活动，然后在配置 BPSim配置的 "配置"选项卡上，创建或编辑此时间参数： • ProcessingTime - 值：0 00:00:00， "日历"字段设置为 "----None----"（此默认值是必需的） • ProcessingTime - 值：0 00:50:00， "日历"字段设置为 "工作日" • ProcessingTime - 值：0 00:40:00， "日历"字段设置为 "周末"
资源	在图表上，单击支持资源，然后在配置 BPSim配置的 "配置"选项卡上，创建或编辑此资源参数 • 数量 - 值：0；日历， "日历"字段设置为 "----无----"（此默认值是必需的） • 数量 - 价值：2；日历， "日历"字段设置为 "工作日" • 数量 - 价值： 1 ；日历， "日历"字段设置为 "周末"
资源选择（分配）	在图表上，单击服务顾客活动，然后在配置 BPSim配置的 "配置"选项卡上，将日历字段设置为 "----None----"，检查 "值"字段是否为资源参数 "选择"设置为：

	bpsim::getResource('Support', 1)作为表达式 该表达式默认从您的 BPMN模型中加载。您可以为任务的资源选择做一些高级配置。

运行仿真

平日

1. 单击 日历"字段并选择 工作日"。
2. 单击 执行"选项卡和  工具栏图标。

生成A名为*Calendar Based Support*仿真进程的文件。此报告文件包含工作日模拟的结果，该结果显示在 标准结果报告"选项卡上的 配置 BPSim审阅窗口的 审阅"选项卡上。

周末

1. 单击 日历"字段并选择 周末"。
2. 单击 执行"选项卡和  工具栏图标。

基于日历的流程仿真支持 -进程文件已更新以显示周末模拟的结果，并显示在 标准结果报告"选项卡上配置 BPSim审阅的 审阅"选项卡上。

在每种情况下，用笔和纸检查结果文件和我们的分析之间的匹配。

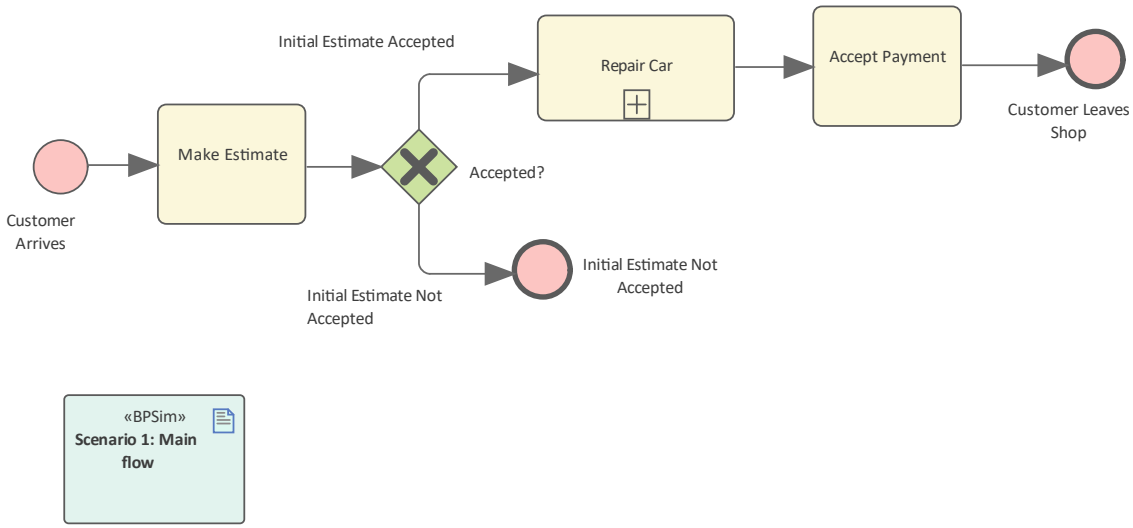
汽车维修进程

此示例模拟汽车维修店的流程。 BPSim 配置：

- 使用由分布初始化的属性参数为每个客户生成随机数量的问题
- 应用概率来模拟：
  - 是否接受初步估计
  - 修复过程中是否会发现新问题
- 在每个任务中增加或减少属性参数的值
- 将属性参数的值用于从网关传出的序列的条件
- 模拟给定开始和持续时间的客户到达

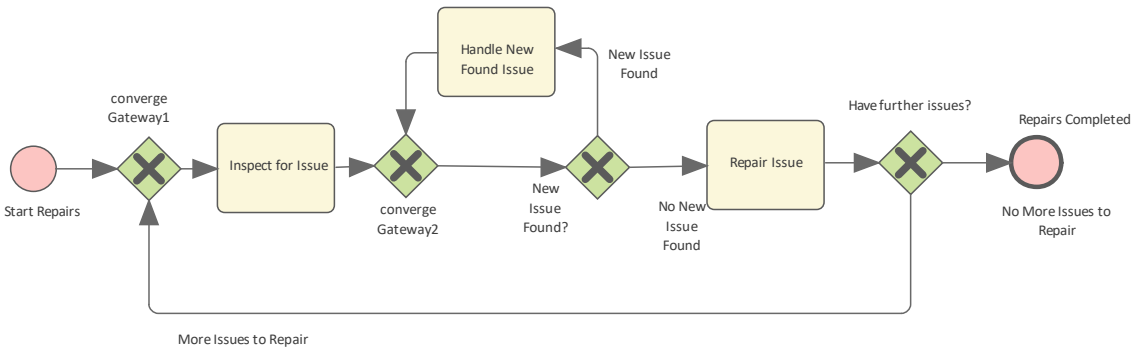
创建 BPMN模型

创建主进程



1. 创建一个开始事件顾客到达。
2. 将序列流添加到目标抽象任务活动进行估计。
3. 将序列添加到目标 Exclusive Gateway *Accepted?*。
4. 将序列流添加到：
   A未接受目标结束事件初始估计
   - A目标子流程修理车
5. 从*Repair Car*中，将序列流添加到目标抽象任务活动接受付款。
6. 将序列流添加到目标结束事件顾客离开商店。

创建子流程修理汽车



1. 创建一个开始事件开始。
2. 将序列流添加到目标 Exclusive Gateway会聚 *Gateway1*。
3. 将一个序列添加到一个抽象任务活动中检查问题。
4. 将序列流添加到独占网关会聚网关 *2*。
5. 将序列添加到独占网关*New*问题*?*。
6. 将序列流添加到：
   - A目标抽象任务活动处理新发现的问题，然后添加一个流回汇聚序列
   - A目标抽象任务修复问题，然后添加一个序列一个target Exclusive Gateway还有其他问题吗？
7. 从网关还有问题吗？将序列流添加到：
   - 目标结束事件修复完成
   -汇聚网关1

配置 BPSim

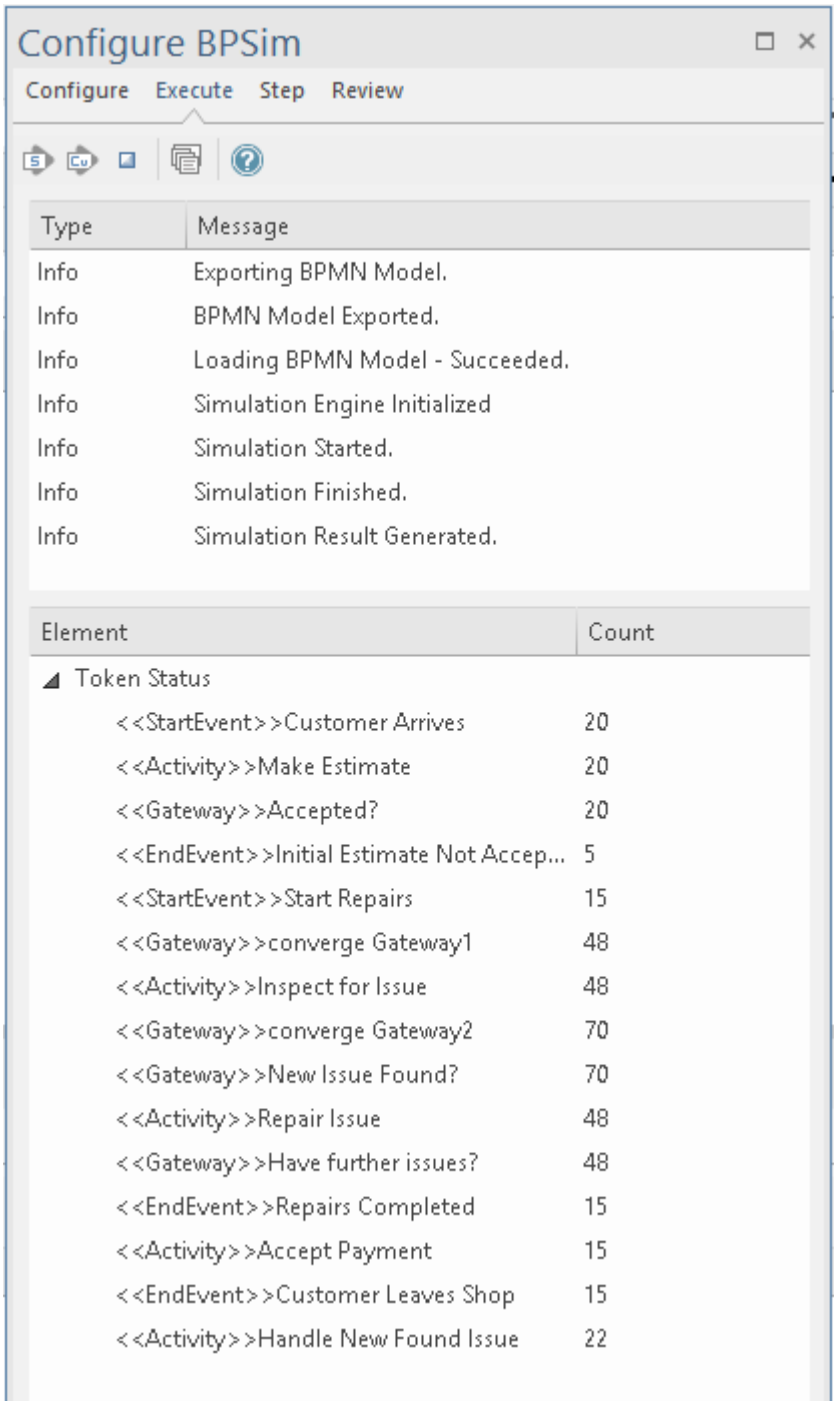
任务	描述
工件和包	1. 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'）。 2. 创建一个业务流程仿真工件的场景*1* ：主流程。 3. 选择包含对应BPMN 2.0模型的包。
开始和持续时间	我们将模拟一家汽车维修店的流程，该店的营业时间为上午 9:00 至下午 5:00，即 8 小时。我们还假设下午 4:50 之后走进的顾客当天不会得到服务。因此模拟开始时间为上午 9:00，持续时间为 7 小时 50 分钟。 在 业务流程仿真”图上，单击 汽车维修名称”，在 配置 BPSim窗口中，单击 “主*1* ”和 汽车维修名称”，更新这些工件： • 开始- 用任何日期（格式为 dd/mm/yyyy）改写 值”字段并将时间部分更改为 上午 9:00” • 持续时间 - 单击 值”字段中的[...]按钮并将其设置为 0 07:50:00”的恒定持续时间
顾客来了	我们将模拟每 24 分钟到达的客户。 第一个客户在上午 9:00 到达，最后一个在下午 4:36 到达（下午 5:00 到达的客户今天将不会得到服务，因为这受 持续时间”设置的限制）。 用笔和纸，我们可以计算出服务了 20 位客户（上午 9:00 到下午 4:36 = 456 分钟；客户数量为 456/24 + 1 = 19 + 1 = 20）。稍后我们将通过仿真结果验证这

	一点。 在 "Car Repair"图表上，单击开始事件元素顾客，然后在配置 BPSim开始中： 1. 单击新参数下拉箭头，然后选择 "控件"。 2. 单击 "参数"下拉箭头并选择 "InterTriggerTimer"。 3. 在 "值"字段中单击...按钮并设置常数数值 "24 分钟"。单击确定按钮和保存工具栏图标。
属性Parameters	我们假设每个客户的汽车最初可能有不同数量的问题。这可以使用随机数生成器来反映。 BPSim 提供了许多发行版以满足您的需求。 在此示例中，我们使用截断正态分布来初始化属性。任务修复问题和处理新发现问题将分别递减和递增*noOfIssues*的值。 1. 在 "汽车维修"图表上，单击开始事件顾客到达。 2. 在配置 BPSim配置的 "配置"选项卡上，单击新参数下拉箭头并创建一个名为*noOfIssues*的属性参数。 3. 在 "值"字段中单击...按钮；显示 "CustomerArrives"对话框的 "配置"配置。 4. 单击 "分布"选项卡并选择 "截断正常"；在田野里： - '平均值'，输入'2' - '标准偏差'，输入' 1 ' - 'Min'，输入' 1 ' - "最大"，输入 "1000" **重要注记：**诸如'TruncatedNormal'之类的分布，返回一个浮点值，但该属性用作整数。设置属性的类型很重要，尤其是在进行相等测试时的条件表达式中。例如，条件表达式*getProperty('noOfIssues') = 0*几乎永远不会满足，因为*noOfIssues*是由浮点分布初始化的。 **提示：如何自定义属性的类型** 创建属性并设置值后，单击工具栏上的图标，然后单击图标以显示 "属性参数"对话框。在属性的 "类型"字段中，单击下拉箭头并选择值 "int"而不是默认的 "double"。 1. 在 "维修汽车"图表上，单击活动维修问题。 2. 在配置 BPSim配置的 "配置"选项卡上，单击新参数下拉箭头并创建一个名为*noOfIssues*的属性参数。 3. 在 "值"字段中单击...按钮。显示 "修复问题"对话框的 "配置"配置。 4. 单击 "表达式"选项卡，然后在 "表达式"字段中输入*{noOfIssues} -1* ；点击确定按钮。 5. 在 "修理汽车"图表上，单击活动处理新发现问题。 6. 在配置 BPSim配置的 "配置"选项卡上，单击新参数下拉箭头并创建一个名为*noOfIssues*的属性参数。 7. 在 "值"字段中单击...按钮。显示 "处理新发现的问题"对话框的 "配置"配置。 8. 单击 "表达式"选项卡，然后在 "表达式"字段中输入*{noOfIssues} +1* ；点击确定按钮。
概率上的序列流	我们估计三分之一的客户不会接受最初的维修估价，剩下的两个会接受。我们还估计，四分之一的维修会发现新问题，其余三个维修不会发现新问题。 在 "汽车维修"图上，参考网关元素接受了吗？.点击： • 初始接受序列流和配置 BPSim窗口中单击参数下拉箭头，并创建一个名

	为 概率参数"的控件参数；在 值"字段中输入 0.67" • *Initial Not Accepted*序列流，在配置 BPSim中点击参数-down箭头，并创建一个控件参数，名为'参数概率'；在 值"字段中输入 0.33" 在 汽车维修"图上，请参阅网关元素新发现问题？ .点击： • *No More to Repair* SequenceFlow 和配置 BPSim序列中单击参数下拉箭头，并创建一个控件 参数参数"的概率；在 值"字段中输入 0.75" • 修复序列流的更多问题，并在配置 BPSim窗口中单击参数下拉箭头，并创建一个名为 参数概率"的控件参数；在 值"字段中输入 0.25"
条件序列条件	我们使用表达式返回一个布尔值作为序列流的条件，这在流的逻辑中起着关键作用。 在 维修汽车"图表上，请参阅还有其他问题？网关元素。点击： • 修复序列流的更多问题，并在配置 BPSim窗口中单击新参数下拉箭头，并创建一个名为 条件"的控件参数；在 值"字段中单击 按钮，单击 表达 "选项卡并在 表达 "字段中键入**{noOfIssues} != 0** • *No More to Repair* SequenceFlow 并在配置 BPSim序列单击参数向下箭头，并创建一个名为 参数条件"的控件参数；在 值"字段中单击 按钮，单击 表达式"选项卡并在 表达式"字段中键入**{noOfIssues} = 0** 注记：网关的所有传出转换都应包含 100% 的逻辑；例如，您不会输入**{noOfIssues} > 10**和**{noOfIssues} < 5**作为条件表达式，因为**[5, 10]**范围内的值不会被任何传出序列流处理。

运行仿真

1. 在配置 BPSim窗口中，单击 执行"选项卡和工具栏中的 图标。
2. 当模拟完成时，执行选项卡提供类似于以下的结果：



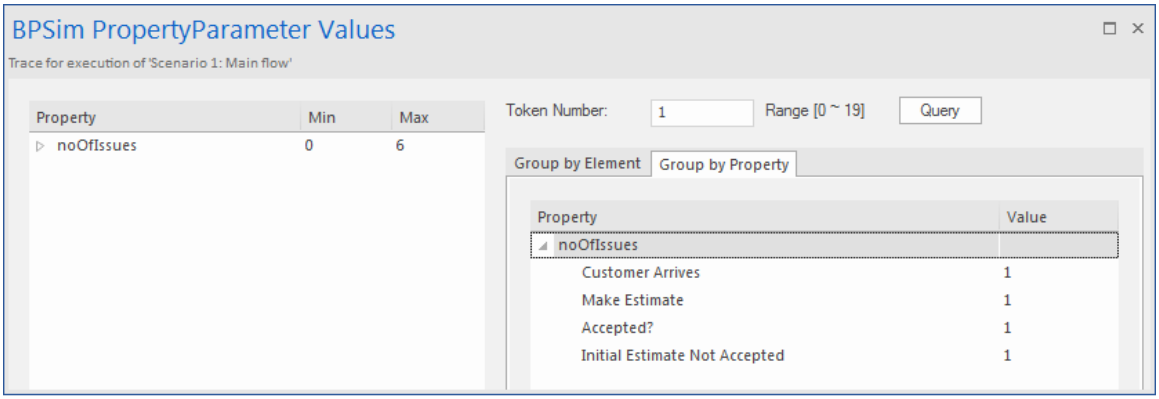
代币分析

- 20 位客户到达，与我们手动计算的数量相匹配（请参阅配置 *BPSim*顾客表的顾客到达）
- 20 位客户中有 8 位不接受初步估价，而 20 位客户中有 12 位接受并修理了他们的汽车；这些数字大致符合1 /3 和 2/3 的概率
- 64 个令牌通过了网关新问题*Found*？ ，其中19个有新问题，45个没有；这些数字大致符合1 /4 和 3/4 的概率

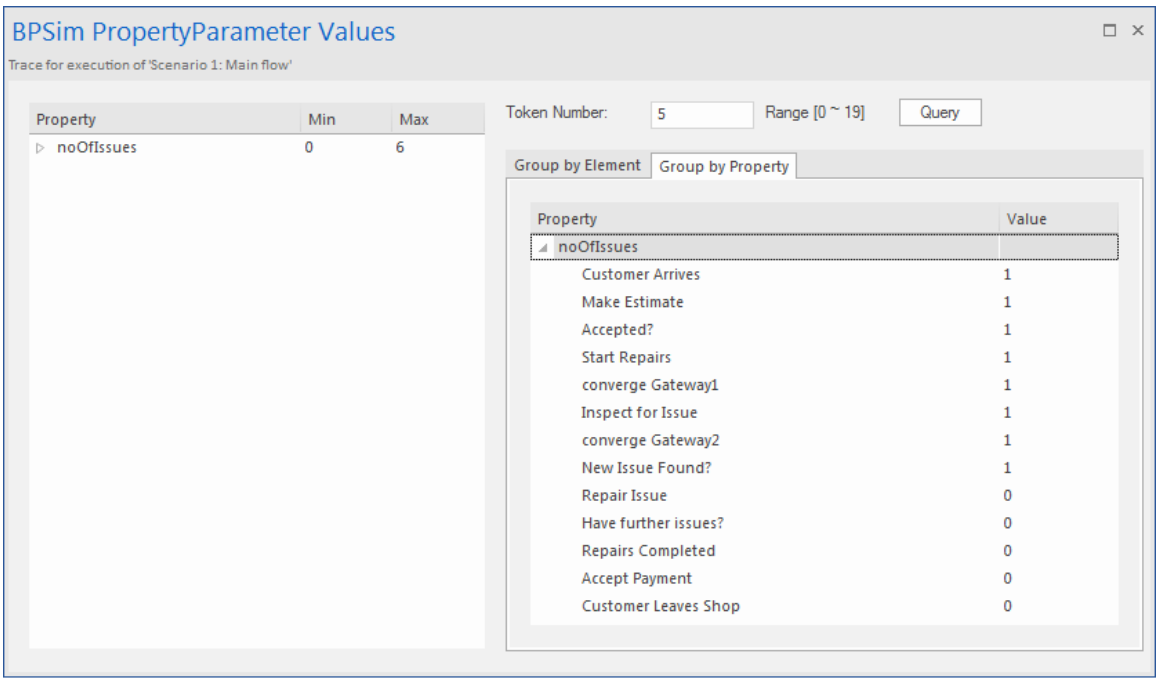
个人客户分析

单击工具栏上的 按钮以打开 "BPSim PropertyParameter Values"对话框。由 有 20 个客户（令牌），您可以在 "令牌编号"字段中输入一个介于 0 和 19 之间的值，然后单击查询按钮进行一些分析：

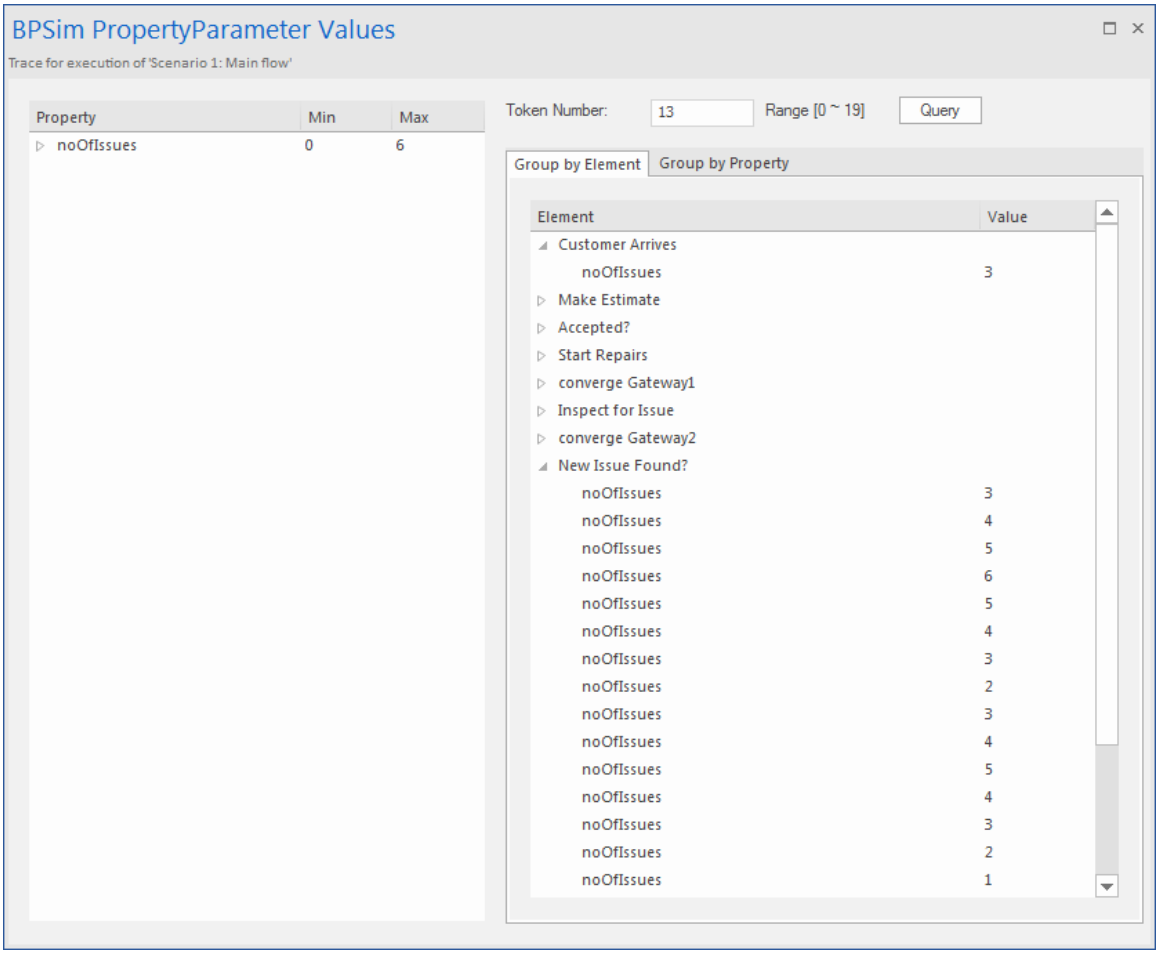
- 该客户不接受初始估算，如 "按属性分组"选项卡上所示：



- 该客户的汽车只有一个问题，已修复：



- 该客户的汽车有三个已知问题，在维修过程中发现了三个其他问题，因此总共修复了六个问题（可能这是一辆非常旧的汽车）；切换到 按元素分组”选项卡：


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

BPMN2.0事件示例

事件是在进程的过程中发生的事情。事件影响进程的流程，通常有原因或影响，一般要求或允许反应。例如，一个活动的开始，一个活动的结束，一个文档状态的变化，或者一个信息的到来，都可以被认为是事件。

事件允许描述 事件驱动"进程。在这些进程中，有三种主要类型的事件：

- 开始事件指示进程将从哪里开始
- 事件，表示一个进程的路径将在哪里结束
- 中间事件，指示在进程的开始和结束之间发生某事的位置

在这三种类型中，事件可以是两种子类型之一：

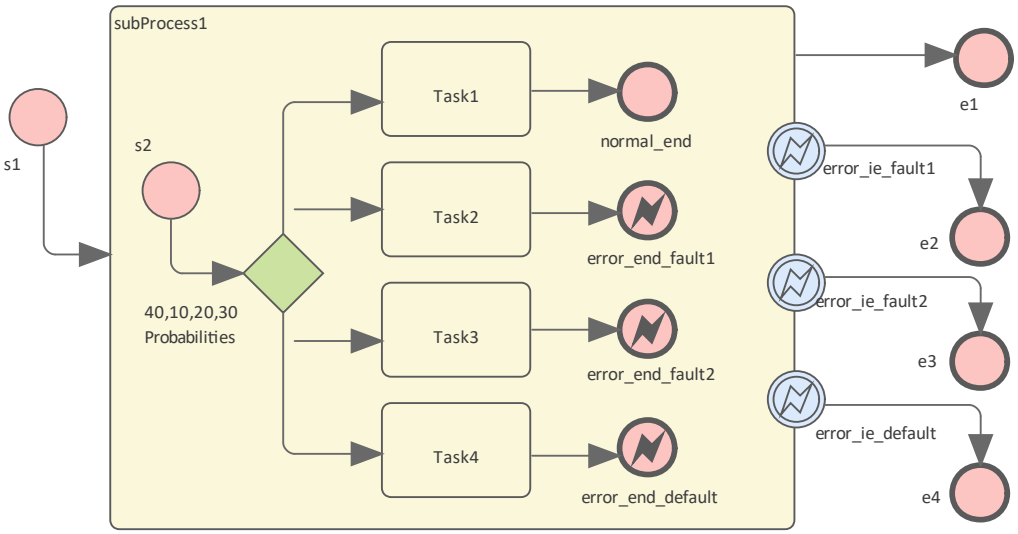
- 捕获触发器的事件- 所有开始事件和一些中间事件都在捕获事件
- 抛出 Result 的事件- 所有事件和一些中间事件都在抛出最终可能被另一个事件捕获的事件

在本节中，我们提供的示例说明了许多常用的 BPMN 2.0 事件。在每个示例中，我们都提供了逐步的 BPMN 建模和 BPSim 配置说明，并对仿真结果进行了深入分析。 EAExample模型中提供了所有示例。

错误事件

当中间错误事件连接到活动的边界时，它就成为异常流的一部分。当令牌导致在正常流程中引发故障名称时触发该事件，并进入错误结束事件。

创建 BPMN模型



创建主进程

- 创建一个开始事件*s1*
- 添加一个序列到一个目标活动元素*subProcess1*；放大活动并右键单击，选择 是展开的"选项，然后打开 属性"对话框并将 类型"设置为 子进程"
- 将序列流添加到目标结束事件元素*e1* （'类型'设置为'无'）
- 创建三个中间事件，将元素从工具箱边界；从即时菜单中选择 边缘安装"和 错误"：
  - *error_ie_fault1* ;将序列流添加到目标 EndEvent元素*e2* （ 类型"设置为 无"）
  - *error_ie_fault2* ;将序列流添加到目标 EndEvent 元素*e3* （'类型'设置为'无'）
  - *error_ie_default* ;将序列流添加到目标 EndEvent elemenr *e4* （'类型'设置为'无'）

创建子流程

在*subProcess1*活动中：

- 创建一个开始事件*s2* , 'Standalone' 并将 '类型' 设置为 'None'
- 为目标网关元素创建一个序列，设置为 独占"，名称为 *"40,10,20,30 Probabilities"*
- 对类型为'abstractTask'的四个目标Activiy元素创建序列流，称为：
  - *Task1* ，并将序列流添加到名为*normal_end*的目标 EndEvent ， 类型"设置为 无"
  - *Task2* ，并将序列流添加到名为*error_end_fault1* 的目标 *EndEvent* 中，"类型"设置为 错误"
  - *Task3* ，并将序列流添加到名为*error_end_fault2* 的目标 *EndEvent* 中， 类型"设置为 错误"
  - *Task4* ，并将序列流添加到名为*error_end_default* 的目标 *EndEvent* 中， 类型"设置为 '错误'

创建 BPMN2.0::Error 元素

创建错误元素*Fault1*和*Fault2* ，它们将被事件用作错误代码。

- 双击*error_end_fault1*元素，在 属性"对话框的 BPMN2.0"选项卡中，找到 errorRef"标签
- 在 值"字段中，单击 ... 按钮并浏览到包含此模型的包

- 单击 加新"按钮，在 名称"字段中输入名称*Fault1* ，然后单击 保存"按钮
- 再次单击加新按钮，在 名称"字段中输入名称*Fault2* ，然后单击保存按钮
- 单击确定按钮，然后再次单击下一个确定按钮

为错误代码设置事件

- 双击*error_end_fault1*元素，在 属性"对话框的 BPMN2.0"选项卡中，找到 errorRef"标签
- 在 值"字段中，单击  按钮并浏览到包含此模型的包
- 单击*Fault1* ，然后单击确定按钮，然后再次单击确定按钮。

对这些元素执行相同的操作：

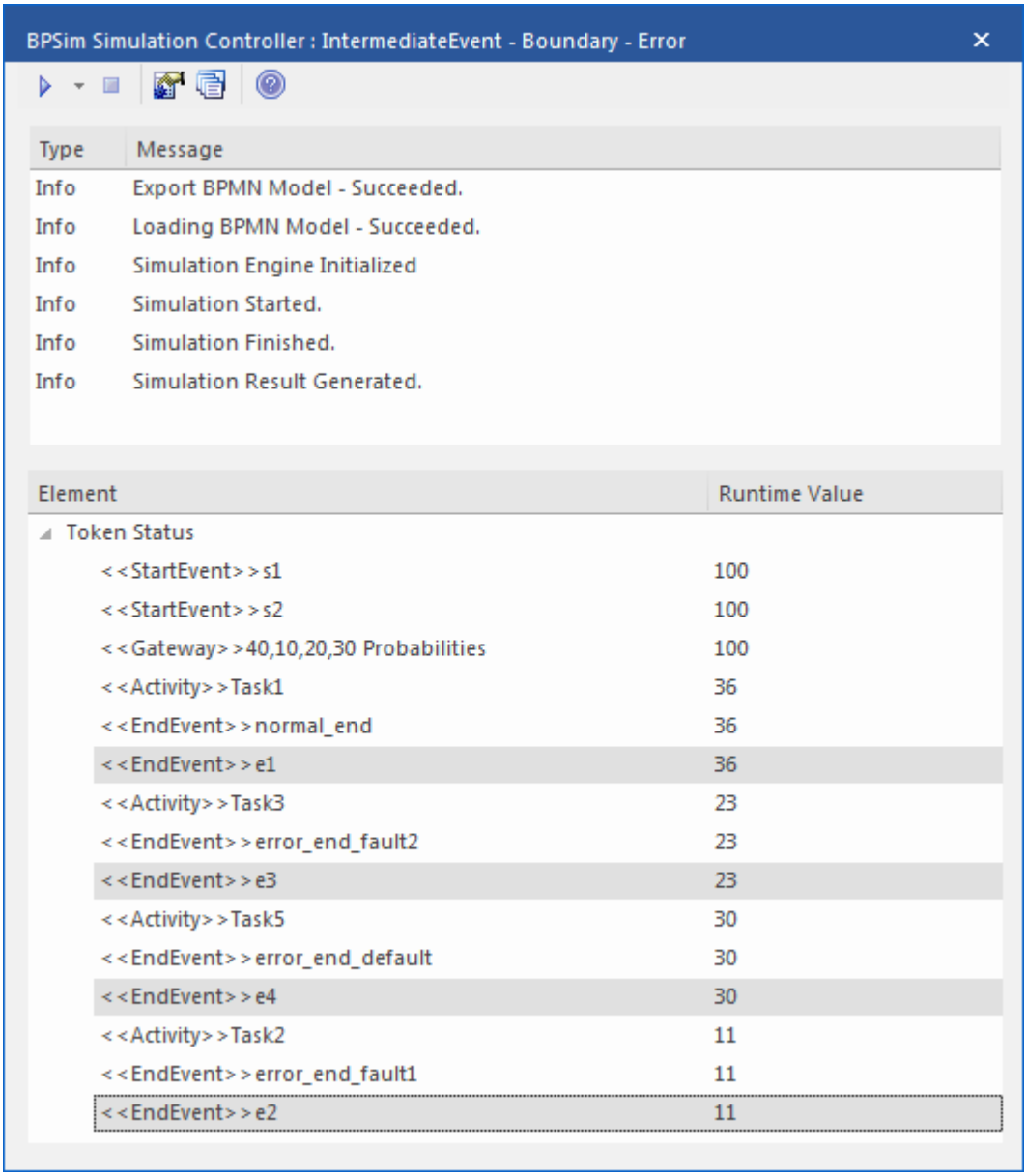
- *error_end_fault2* ，点击*Fault2*
- *error_ie_fault1* ，点击*Fault1*
- *error_ie_fault2* ，点击*Fault2*

配置 BPSim

物件	行动
工件&包	• 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'） • 创建它的工件名称，单击 边界- 资源包"中的 Intermediate - 资源包"（在 "  事件/创建工件包"字段中选择 父元素包"，然后单击 加新按钮"，然后单击 新元素包"保存按钮和确定按钮） 然后所有的 BPMN 元素将被加载到配置 BPSim窗口中。
s1	• 从配置 BPSim窗口左侧的树中，展开 StartEvent"并单击 s1" • 在 控件"选项卡的 新建参数..."字段中，单击下拉箭头并选择 触发器计数" • 在 值"字段中，输入 100"
概率	从配置 BPSim窗口左侧的树中，展开 网关	40,10,20,30 概率'。 提示：您也可以浮动配置 *BPSim*窗口，然后点击*BPMN*图上的元素或连接器；配置 *BPSim*窗口中的元素将被自动选中。 对于每个任务元素，在 控件"选项卡中单击 新建参数"下拉箭头并选择 概率"，然后在 值"字段中键入相应的值： • 对于 Task1 类型 0.4" • 对于任务 2，键入 '0。 1 ' • 对于 Task3 类型 0.2" • 对于 Task4 类型 0.3"

运行仿真

- 在 配置 BPSim "对话框工具 上，单击 运行"图标打开 运行控制器"对话框
- 点击运行按钮，选择 标准仿真"
- 模拟结果如下：



分析：

从概率流出的序列上的序列集的概率分别为1和0.3。

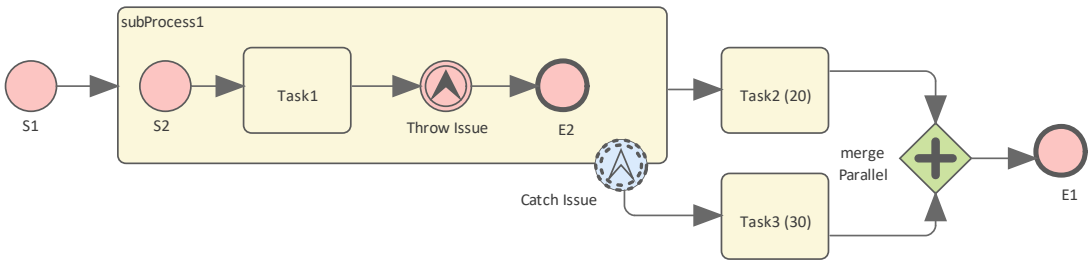
- 100 次传球中有 36 次在*normal_end*完成，流向*e1*
- 100 次通过中有 11 次在*error_end_fault1*完成，由 ErrorRef Fault1 触发*error_ie_fault1* ，异常流向*e2*
- 100 次中的 23 次在*error_end_fault2*完成，由 ErrorRef *Fault2*触发*error_ie_fault2* ，异常流向*e3*
- 100 次传递中有 30 次在*error_end_default*完成，这触发了*error_ie_default*因为它们没有设置 ErrorRef 并且异常流向*e4*

36、11、23、30这三个数字加起来就是100，在*s1*中设置为*TriggerCount*，所以它们匹配100%的概率

升级事件

在 BPMN 中，Escalation 是 Error 的非中断对应物，具有类似的 throw-catch 行为。然而，与 Error 不同的是，从活动中退出的正常流程和异常流程是并行路径，而不是替代路径。

创建 BPMN模型



创建主进程

- 创建一个开始事件*S1*
- 将序列流添加到目标活动*subProcess1*；放大活动并右键单击，选择 是展开的"选项，然后打开 属性"对话框并将 类型"设置为 子进程"
- 将一个序列流添加到目标 abstractTask活动元素*Task2 (20)*（打开 属性"对话框并将 类型"字段设置为 抽象任务"）
- 将一个序列添加到目标并行网关元素合并并行（打开 属性"对话框并将 类型"字段设置为 并行"）
- 将序列流添加到目标事件
- 在*subProcess1* 上，添加边界非中断 EscalationEventCatch问题（将 事件"图标拖到事件上，并从即时菜单中选择 Edge Mounted"和 Escalation"；双击元素以显示 属性" '对话框并添加名称，然后在'类型'字段中选择'边界非中断>升级'）
- 将序列流添加到目标 abstractTask活动元素*Task3 (30)*（打开 属性"对话框并将 类型"字段设置为 抽象任务"）
- 添加一个序列到目标元素合并并行

创建子流程

- 在*subProcess1*内（或下），创建一个开始事件*S2*
- 将序列流添加到目标 abstractTask活动元素*Task1*（打开 属性"对话框并将 类型"字段设置为 抽象任务"）
- 将序列流添加到目标事件Escalation IntermediateEventThrow问题（打开 属性"对话框并在 类型"字段中选择 抛出 > 升级"）
- 将序列流添加到目标结束事件*E2*

创建 BPMN2.0::Escalation 元素

在工具箱中，展开 图表2.0 Types"页面，将 Escalation"图标拖到图表上，并将元素命名为*Escalation1* ；这将被事件用作升级代码。

为升级代码设置事件：

- 双击*Throw*问题并在 escalationRef 标记的 值"字段中单击 ... 图标并找到并选择*Escalation1*
- 双击*Catch*问题并再次在 escalationRef 标签的 值"字段中单击 ... 图标并找到并选择*Escalation1*

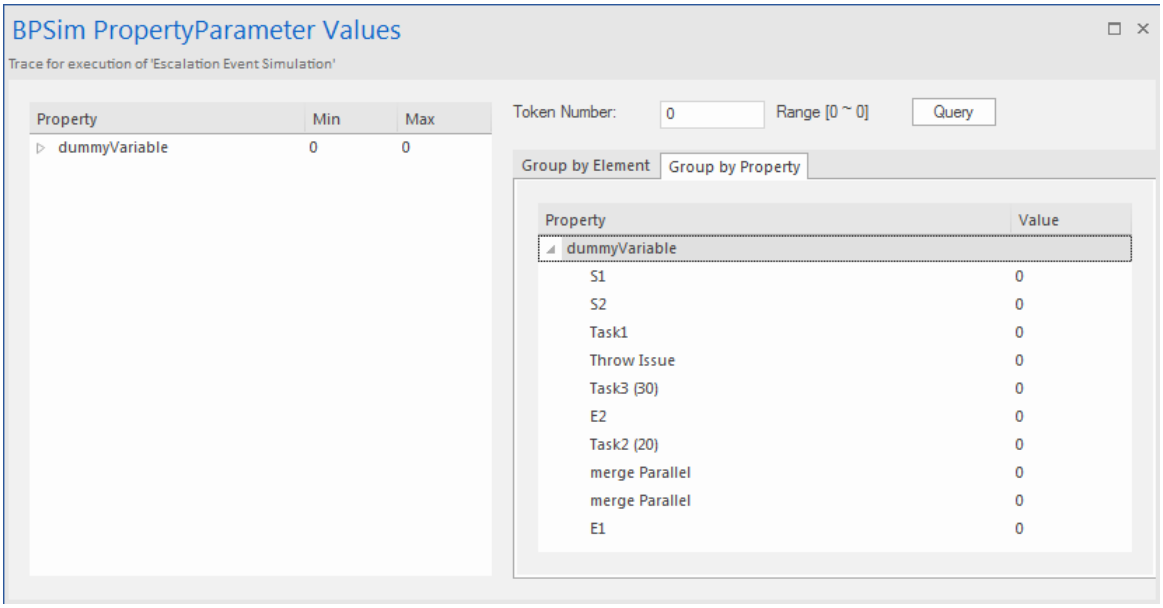
（活动退出的异常流程是并行的。）

配置 BPSim

任务	行动
工件&包	• 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'） • 创建加新事件添加工件名称的 仿真事件"（在 选择/创建工件包名称"字段中，单击 选择/创建父元素名称"并  按钮，然后键入按钮，然后键入 保存"和确定按钮） 然后所有的 BPMN 元素将被加载到配置 BPSim窗口中。
触发器的开始事件计数	• 从配置 BPSim窗口左侧的树中，展开 StartEvent"并单击*S1* • 在 控件"选项卡的 新建参数..."字段中，单击下拉箭头并选择 触发器计数" • 在 值"字段中，输入 1"
处理时间	• 在左侧树中展开 活动"并单击*Task2 (20)* ；在 处理时间"的 值"字段中输入 20"，在 单位"字段中输入 s"（20 秒） • 点击*Task3*（*30*）；同样，将 'ProcessingTime' 设置为 30 秒
dummyVariable 用于跟踪	为了显示给定标记的准确轨迹，您必须在*S1*上设置一个虚拟变量。 • 在左侧层次结构中单击*S1* ，然后在 属性"选项卡上用变量名称覆盖新属性文本（例如 属性"） • 在 值"字段中，单击  按钮，然后在 <<StartEvent>>S1 : <变量名称>"对话框中单击 数字"并键入 常量数字"值 0"；点击确定按钮

运行仿真

- 在 配置 BPSim "对话框工具 上，单击 运行"图标打开 运行仿真控制器"对话框
- 点击 运行"图标下拉箭头并选择 标准仿真"
- 仿真完成后，单击工具栏上的  按钮，显示 BPSim PropertyParameter Values"对话框
- 单击 查询属性"按钮和 按属性分组"选项卡，然后展开 dummyVariable"（或您分配给变量的名称）



分析：

与*Error*不同，从*subProcess1*退出的正常流程和异常流程不是替代路径，而是并行的。这个特征很容易从trace中发现：

- 在*Task3 (30)*开始后， *E2*和*Task2 (20)*仍然被遍历
- 遍历*mergeParallel*两次后到达*E1*

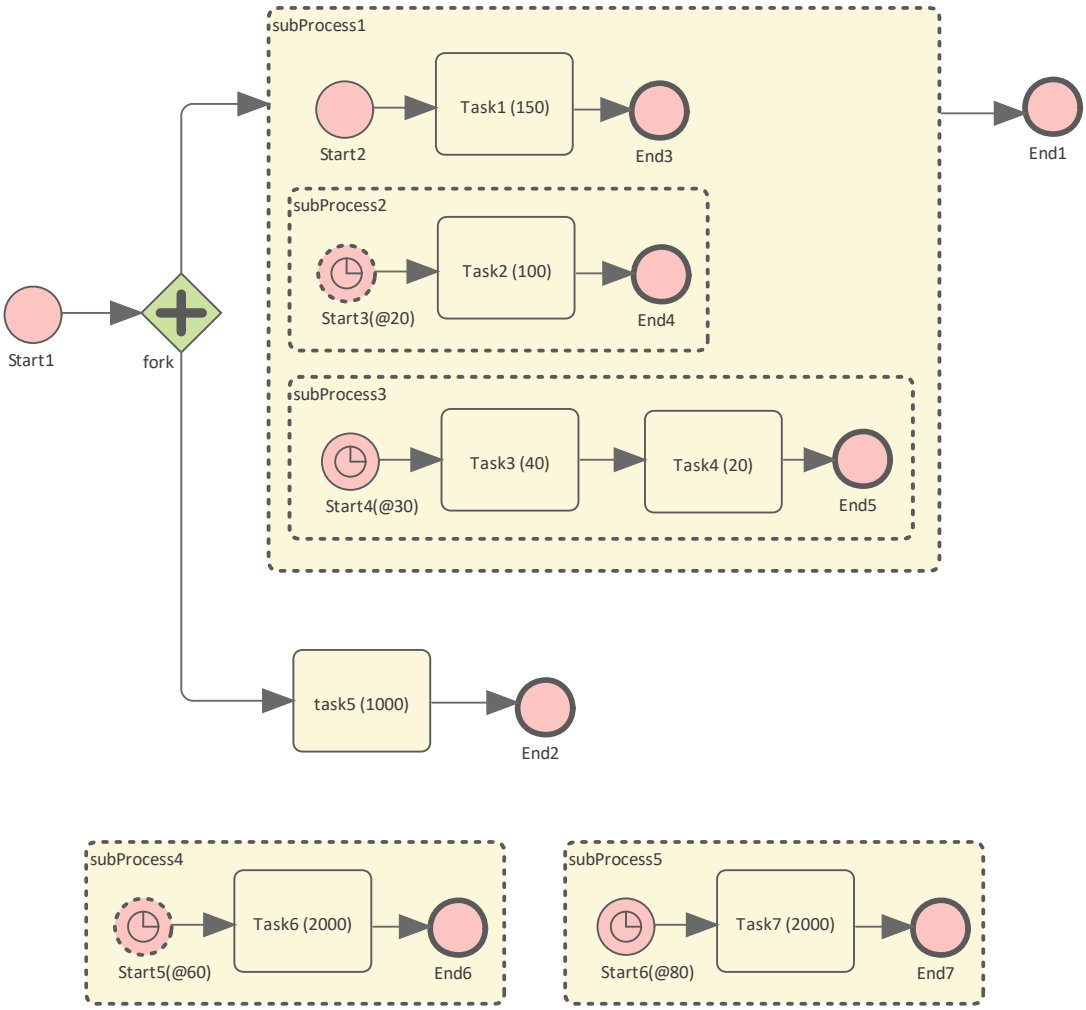
事件子流程

事件子流程使您的系统能够在给上下文流程或进程的时间内处理事件。一个事件子流程总是以一个开始事件开始；它不是由正常的控制流实例化，而是仅在触发相关的开始事件时实例化。事件子流程是自包含的，不得连接到子流程中的其余序列流。

- 如果设置了其开始事件的 isInterrupting 属性，则事件子进程取消执行封闭子进程
- 如果未设置 isInterrupting 属性，则封闭子流程的执行与事件子流程并行继续

在这个例子中，我们演示了Interrupting 和Non-事件Sub-Processes 如何影响封闭的Sub-Process 和进程的生命线。

创建 BPMN模型



主要流程模型

- 创建一个 StartEvent *Start1*
- 将序列流添加到目标并行网关元素分叉
- 添加一个序列
  - 一个子流程*subProcess1* ，并从中添加一个序列*S*目标结束事件元素
  结束*1*
    - 一个抽象任务*Task5* ，并从中添加一个序列流到目标结束事件元素*End2*

关于如何模型事件子流程的提示

- 将 活动-业务流程”工具箱中的活动拖到图表上
- 双击活动以显示 属性”对话框，并在 类型”字段中选择 孚进程”；将 'triggeredByEvent' 设置为 'true' 并点击确定按钮
- 右键单击元素并选择 是展开的”选项；这将在左上角显示元素名称

为主进程模型事件子进程

- 创建事件子流程*subProcess4*
  - 创建事件开始*(@60)* ，然后双击它以显示 属性”对话框
  并且，在 类型”字段中，选择 事件子进程非中断 > 定时器”；点击确定按钮
  - 将序列流添加到目标抽象任务活动*Task6(2000)*
  - 将序列流添加到目标结束事件元素*End6*
- 创建事件子流程*subProcess5*
  - 创建事件开始*(@80)* ，然后双击它以显示 属性”对话框
  并且，在 类型”字段中，选择 事件子进程中断 > 定时器”；点击确定按钮
  - 将序列流添加到目标抽象任务活动*Task7(2000)*
  -将序列流添加到目标结束事件元素*End7*

模型子流程 subProcess1 和封闭的事件子流程

- 创建一个 StartEvent *Start2*
  -将序列流添加到目标抽象任务活动*S (150)*
  -添加一个 流向目标的序列事件
- 创建一个事件子流程*subProcess2*
  - 创建事件开始*(@20)* ，然后双击它以显示 属性”对话框
  并且，在 类型”字段中，选择 事件子进程非中断 > 定时器”
  - 将序列流添加到目标抽象任务活动*Task2(100)*
  -将序列流添加到目标结束事件元素*End4*
- 创建一个事件子流程*subProcess3*
  - 创建事件开始*(@30)* ，然后双击它以显示 属性”对话框
  并且，在 类型”字段中，选择 事件子进程中断 > 定时器”
  - 将序列流添加到目标抽象任务活动*Task3(40)*
  - 将序列流添加到目标抽象任务活动*Task4(20)*
  - 将序列流添加到目标结束事件元素*End5*

配置 BPSim

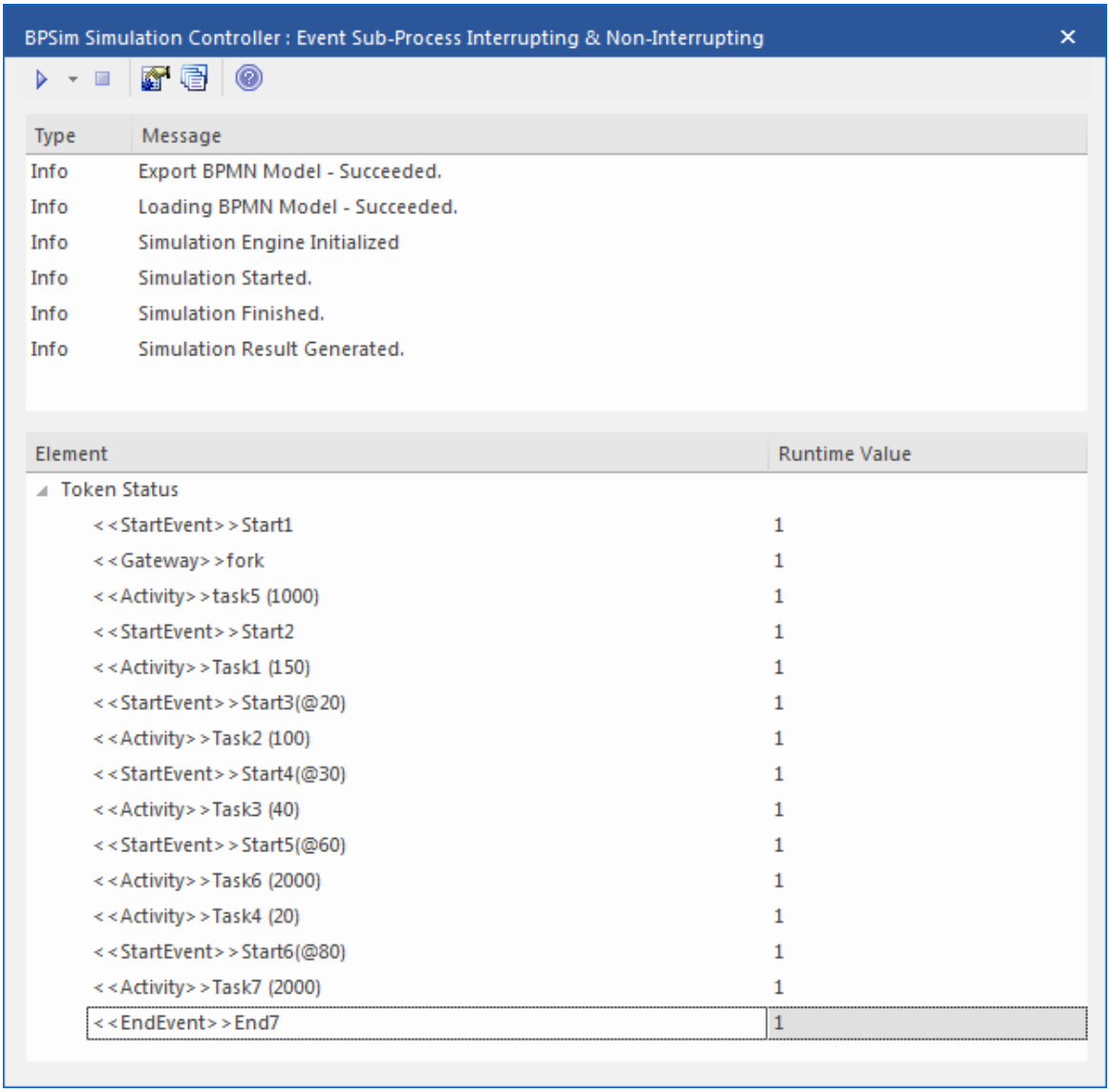
使用这个表，我们在配置包中创建工件并配置每个元素的参数值。

任务	行动
创造工件	• 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'） • 创建加新按钮工件'事件子进程和Non-Interrupting'（在'选择/创建工件器名称'字段中，  并单击它的包级名称，然后在元素上输入点击保存按钮和确定按钮） 然后所有的 BPMN 元素将被加载到配置 BPSim窗口中。
开始用于事件子进程中的开始事件	从 配置 BPSim ”对话框左侧的树中，展开 StartEvent”。 对于此处列出的每个元素，在 控件”选项卡上单击 新参数...”字段中的下拉箭头并选择参数“InterTriggerTimer”。单击 值”字段中的  按钮以打开 参数 ”对话框并选择 常量>数字”，然后输入值并选择 秒”。 • 开始 3(@20)：20 秒 • 开始 4(@30)：30 秒

	• 开始 5(@60)：60 秒 • 开始6（@80）：80 秒
任务的处理时间	从配置 BPSim窗口左侧的树中，展开 活动"。 对于此处列出的每个元素，在 时间"选项卡上单击 新参数..."字段中的下拉箭头并选择参数 处理时间"。单击 值"字段中的  按钮以打开 参数"对话框并选择 常量>数字"，然后输入值并选择 秒"。 • 任务 1 (150)：150 秒 • 任务 2 (100)：100 秒 • 任务 3 (40)：40 秒 • 任务 4 (20)：20 秒 • 任务 5 (1000)：1000 秒 • 任务 6 (2000)：2000 秒 • 任务 7 (2000)：2000 秒

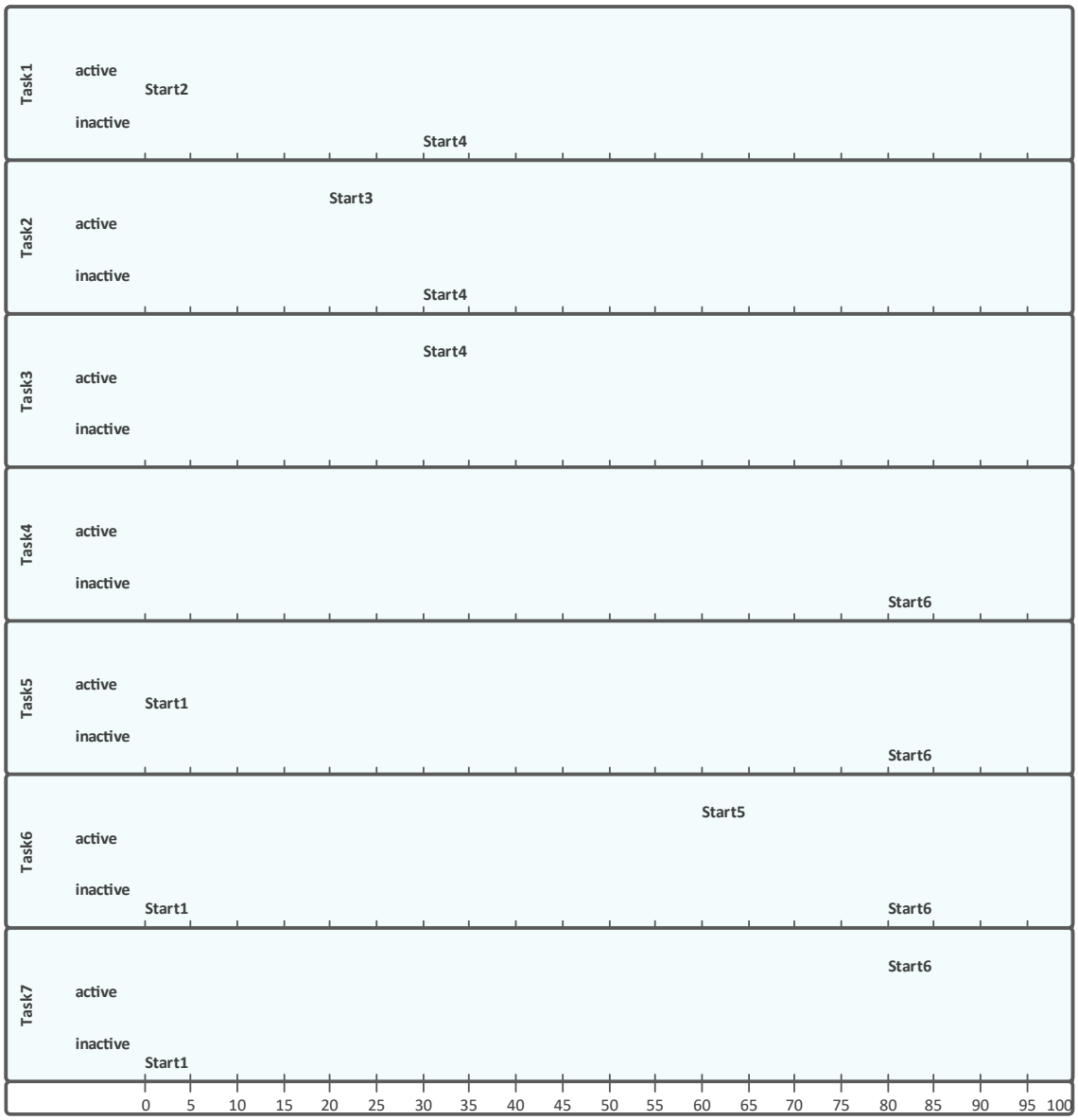
运行仿真

- 在 配置 BPSim "对话框工具 上，单击 运行"图标打开 运行仿真控制器"对话框
- 点击 运行"图标下拉箭头并选择 标准仿真"



分析

阅读结果，可能并不完全直截了当地看到发生了什么；但是，如果我们在时序图中为每个任务绘制生命线，它就会变得更加清晰。



- 事件*(@20)*是非中断的，它没有在 20 秒时停止*Task1*
- 事件*(@30)*正在中断，它在 30 秒时停止了*Task1*和*Task2* ；它没有影响*Task5*因为*Task5*的封闭进程（主进程）级别高于*Start4*的封闭子进程（ *subProcess1* ）
- *EventStart5(@60)*是非中断的，它在 60 秒时启动了事件，而不影响*Task3*或*Task5*
- *EventStart6(@80)*正在中断，它在 80 秒时启动了 *Task7*事件并中断了与其封闭进程相同或更低级别的正在运行的任务（ *Task4* 、 *Task5* 、 *Task6* ）
- 只达到预期的*End7*

带有链接事件的斐波那契数生成器

链接事件是连接进程A两个部分的机制。链接事件可用于：

- 创建循环场景，作为 进程级别"中的通用 Go To 对象
- 避免长序列流线；成对的链接事件可以用作跨多个页面打印进程的 页面"连接器

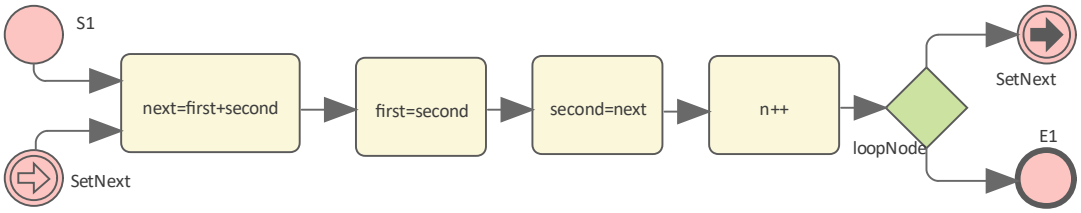
事件的使用仅限于单个进程级别（即不能链接父进程和子进程）。

可以有多个源链接事件，但只能有一个目标链接事件。

- 目标链接事件标记未填充，从源链接中 捕获"
- 源链接事件标记被填充以 扔"到目标链接

元素执行引擎在运行仿真时，源-目标链接事件是通过 ElementNAME 配对的，所以不能为空。

创建 BPMN模型



- 创建一个 StartEvent *S1*
- 添加一个序列目标abstractTask活动元素*next=first+second*（打开 属性"对话框并将 类型"字段设置为 抽象任务"）
- 将序列流添加到目标 abstractTask 活动元素*first=second*
- 将序列流添加到目标 abstractTask活动元素*second=next*
- 将序列流添加到目标 abstractTask活动元素*n++*
- 将序列流添加到目标独占网关元素*loopNode*（在即时菜单上，选择 独占"）
- 将一个序列流添加到这些目标元素中的每一个：
  - A Throwing Link 中间事件元素*SetNext*（打开 属性"对话框并设置 '类型'字段到'投掷>链接'）和
  - 结束事件元素*E1*
- 创建捕获链接中间事件元素*SetNext*（打开 属性"对话框并将 类型"字段设置为 捕获>链接"）
- 添加一个序列到目标元素*next=first+second*

配置 BPSim

我们将使用属性参数来定义序列流如何形成一个循环，在该循环期间将生成一个斐波那契数。循环机制是通过一对事件实现的。

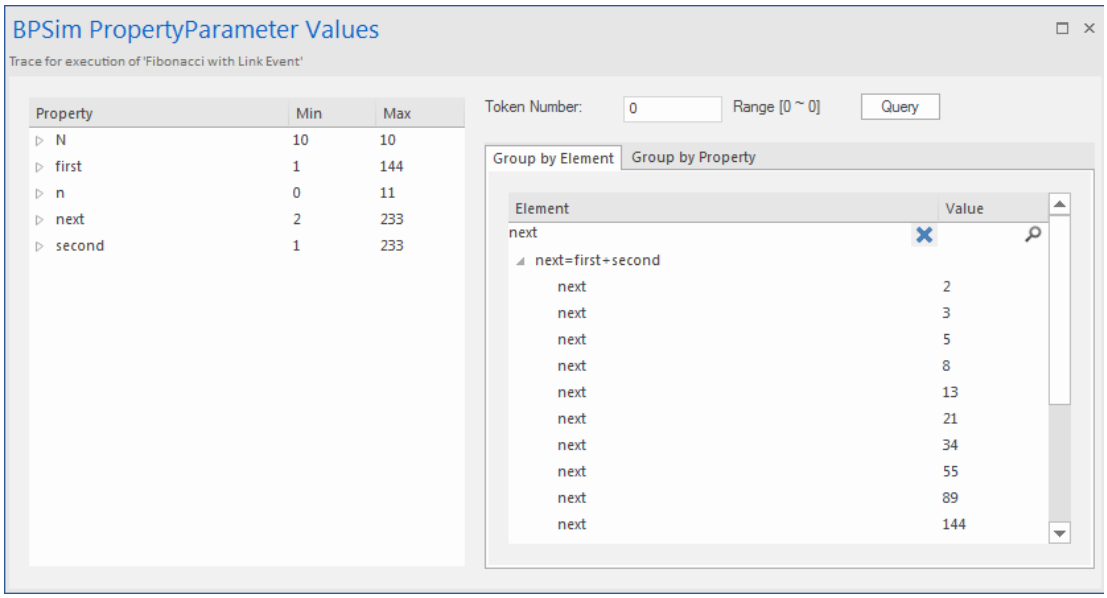
打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'）

任务	行动
元素：S1	在左侧的元素列表中，展开开始事件组并单击*S1*。 单击 控件选项卡"和 新参数"下拉箭头；选择 触发器计数"。 在 值"字段中输入 '1"。

	单击 属性"选项卡 改写新属性文本以创建这些属性： • N -并在 值"字段中输入 f0"作为要生成的斐波那契数的总数 • 首先-在 值"字段中输入 "1 " • 第二-在 值"字段中输入 "1 " • n -并在 值"字段中输入 0"作为第*n*个新的斐波那契数
元素：next=first+second	在元素类型列表中，展开活动组并单击*next=first+second* 。 单击 属性"选项卡并用 下一步"覆盖新属性文本。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 "{first}+{second}"。 点击确定按钮。
元素：first=second	在元素类型列表中，在活动组中单击*first=second* 。 单击 属性"选项卡并用 第一个"改写新属性文本。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 "{second}"。 点击确定按钮。
元素: second=next	在元素类型列表中，在活动组中单击*second=next* 。 单击 属性"选项卡并用 第二个"覆盖新属性文本。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 {next}"。 点击确定按钮。
元素：n++	在元素类型列表中，在活动组中单击*n++* 。 单击 属性"选项卡并用 n"改写新属性文本。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 {n}+1"。 点击确定按钮。
网关条件	在元素列表中，展开 Gateway 组和 LoopNode元素，然后单击*SetNext* 。 单击 控件"选项卡和 新参数"下拉箭头，然后选择 条件"。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 {n} <={ N }"。 点击确定按钮。 现在点击*E1* 。 单击 控件"选项卡和 新参数"下拉箭头，然后选择 条件"。 在 值"字段中，单击…按钮，单击 表达式"选项卡并输入表达式 {n} > { N }"。 点击确定按钮。

运行仿真

- 在 配置 BPSim ”对话框的工具 中，单击 迭行”图标； 仿真仿真控制器”对话框显示
- 点击 迭行”图标下拉箭头并选择 标准仿真”
- 模拟完成后，点击工具  完成； “BPSim PropertyParameter Values”对话框显示
- 单击查询按钮和 按元素分组”选项卡，然后展开 下一步=第一+第二”；列出所有属性的快照值
- 应用过滤器 下一步”（右键单击列表标题，选择 切换过滤器栏”并在 无素”标题下键入 下一步”）；结果将类似于此图像：



再生成十个斐波那契数：

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

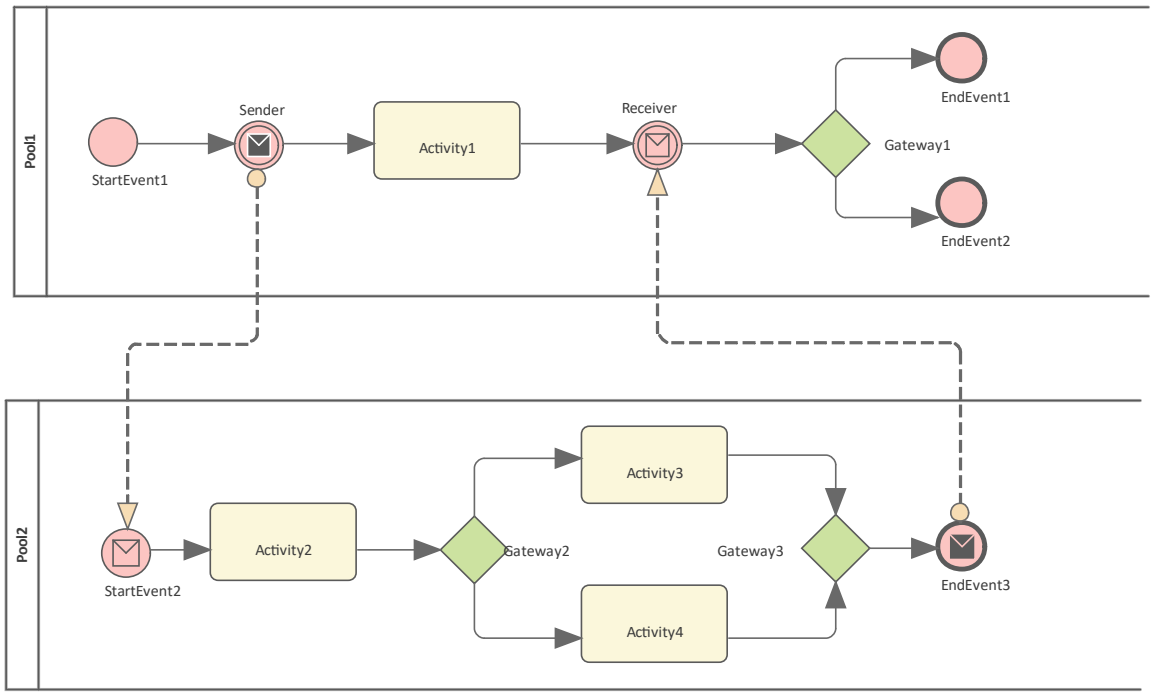
信息事件

在正常序列流中使用时，信息事件可用于发送或接收消息。

- 向参与者发送消息时，复制所有属性参数的值；一旦发送消息，令牌将继续沿着序列流
- 接收消息时，当收到消息时触发该事件。

这个例子展示了信息事件的特征。我们将首先创建 BPMN模型，然后配置运行并运行仿真。

创建 BPMN模型



序列

池1

- 令牌从 StartEvent1 开始
- 收到令牌后，Sender（一个 Throwing Intermediate信息事件）创建一条消息并将当前属性值复制到该消息中
- 发件人将消息发送给 收件人"参与者（Pool2，StartEvent2）
- 发送者沿着它的序列流转发令牌，直到接收者
- 令牌在 Receiver 处等待，直到消息到达

池2

- StartEvent2 收到消息并启动令牌
- StartEvent2 复制消息中的值并将其设置在令牌中
- StartEvent2 沿其序列流转发令牌，直到 EndEvent3
- EndEvent3 创建一条消息并将当前属性值复制到该消息中
- EndEvent3 将消息发送给 "To"参　者（Pool1，Receiver）

池 1 续

- 等待的 Receiver 被唤醒，属性值从到达的消息中更新

创建图表

- 创建 BPMN 2.0协作图
- 选择 "在新的 CollaborationModel 中创建此图表"选项
- 通过将 "泳池"图标从工具箱拖到图表上来创建*Pool1*和*Pool2*

池1内

- 创建一个类型为 "None"的开始事件，命名为*StartEvent1*
- 向目标 "Throwing信息"类型的中间事件添加一个序列流，称为*Sender*
- 添加一个序列流到'abstract'类型的目标活动，称为*Activity1*
- 将序列流添加到目标：
  - 类型为 "无"的结束事件，称为*EndEvent1*
  - 类型为 "无"的结束事件，称为*EndEvent2*

池2内

- 创建一个 "信息"类型的开始事件，称为*StartEvent2*
- 添加一个序列流到'abstract'类型的目标活动，称为*Activity2*
- 将一个序列流添加到 "独占"类型的目标网关，称为*Gateway2*
- 将序列流添加到目标：
  - "抽象"类型的活动，称为*Activity3*
  - "抽象"类型的活动，称为*Activity4*
- 将*Activity3*和*Activity4*中的序列流添加到 "Exclusive"类型的目标网关，称为*Gateway3*
- 在 "信息"类型的结束事件中添加一个序列流，称为*EndEvent3*

信息流

- 添加一个从*Sender*到*StartEvent2*的信息流
- 将*EndEvent3*中的信息流添加到*Receiver*

配置 BPSim

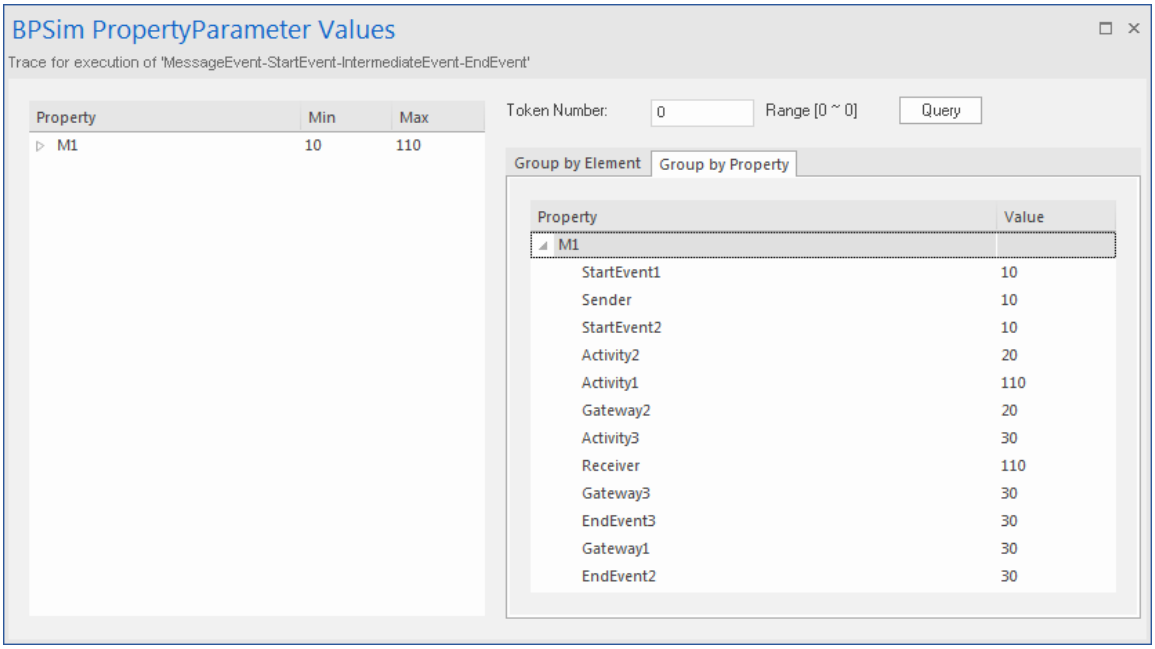
为了显示信息流携带值的能力，我们创建了一个属性参数'M1'并在每个活动中更改其值。然后我们使用 "M1"的值作为序列流条件表达式的一部分。

任务	描述
创建工件包	• 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'） •  事件/创建工件"字段中的 "事件/创建按钮"并创建一个工件 "消息事件-开始事件-中间事件-结束事件"的浏览器 • 在 "选择包"字段中选择包含模型的包 模型中的所有 BPMN 元素都加载到配置 BPSim模型中。
属性值	我们将在*StartEvent1*处为属性参数"M1"赋予初始值 10。然后我们在令牌流经流程时更改值，并且值在参与者之间复制。 在对话框左侧的元素列表中： • 展开 "StartEvent"组，单击*StartEvent1*并在 "属性"选项卡上，用 "M1"覆盖新属性；在 "值"字段中单击  按钮并选择 "常量"和 "数字"，然后在 "常量数字"字段中键入 "0" • 展开 "活动"组，单击*Activity1*和 "属性"选项卡，并用 "M1"覆盖新属性；在

	值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 “{M1} + 100” • 单击*Activity2*和 属性”选项卡，然后用 M1”覆盖新属性；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} + 10” • 单击*Activity3*和 属性”选项卡，然后用 M1”覆盖新属性；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} + 10” • 单击*Activity4*和 属性”选项卡，并用 M1”覆盖新属性；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} + 1 ” 提示：*{PropertyName}* 的格式是 *getProperty("PropertyName")* 的一种方便的缩写形式。
控件参数	在这个模拟中，我们只需要一个令牌来评估模型的行为。 • 在展开的 StartEvent”组中，单击*StartEvent1*和 控件”选项卡；单击 新参数”下拉箭头并选择 触发器计数”，然后输入 1 ”的 值” 现在为网关的传出序列流设置条件。在对话框左侧的元素列表中，展开 网关”组： • 展开*Gateway1* ，单击*EndEvent1*并在 控件”选项卡上，然后单击 新参数”下拉箭头并选择 条件”；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} >= 50” • 单击*EndEvent2*并在 控件”选项卡上，然后单击 新参数”下拉箭头并选择 条件”；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} < 50” • 展开*Gateway2* ，单击*Activity3*并在 控件”选项卡上，然后单击 新参数”下拉箭头并选择 条件”；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} >= 15” • 单击 控件”控件卡，然后单击 新参数”下拉箭头并选择 条件”；在 值”字段中单击…按钮并选择 表达式”，然后在 表达式”字段中键入 {M1} < 15”

运行仿真

- 在 配置 BPSim ”对话框工具 上，点击运行按钮； 仿真仿真控制器”对话框显示
- 点击 迤行”图标下拉箭头，选择 标准仿真”；模拟开始
- 模拟完成后，点击 按钮； 属性参数值”对话框显示，在模拟过程中跟踪属性值
- 在 令牌编号”字段中输入 0”，然后单击查询按钮和 按属性分组”选项卡



分析

由于*Activity1*的 'ProcessingTime' 被设置为分布值，结果是：

- [Process1] 在*Pool1.StartEvent1*之后的 'M1' 值为 '10'，正如预期的那样
- *[Process2] *Pool2.StartEvent2*的 'M1' 值为 '10';此值来自*Pool1.Sender*发送的消息

现在实际上有两个'M1' - *Process1.M1*和*Process2.M1*

- [Process2] *Pool2.Activity2*增加*Process2.M1* 10； *[进程2.M1 == 20]*
- 【Process1】 *Pool1.Activity1*增加*Process1.M1* 100； [过程*1.M1 == 110]*
- [Process2] 评估条件表达式；当 '20 > 15' 时，令牌将流向*Activity3 [Process2.M1 == 20]*
- [Process2] *Pool2.Activity3*增加*Process2.M1* 10； *[进程2.M1 == 30]*
- [Process1] *Pool1.Receiver*已到达并等待*[Process1.M1 == 110]*
- [Process2] *Pool2.Gateway3*作为 Merge 节点，继续*EndEvent3 [Process2.M1 == 30]*
- *[Process1] *Pool1.Receiver* 被消息唤醒（携带*M1 == 30* ）并且*Process1.M1*的值从 110 变为 30
- [Process1] 评估条件表达式；作为 '30 < 50'，令牌将流向*EndEvent2 [Process1.M1 == 30]*

注记

- 标有星号 (*) 的行是信息流的效果
- 定义了流程中的顺序；然而，两个进程之间的顺序并不总是可预测的
- 投掷信息事件分叉另一个进程；捕获信息用作线程同步

信号事件

A信号事件通过发布-订阅集成提供功能 投掷者"和 捕手"的功能。 A投掷者"将广播信号而不是将其发送给特定进程；侦听该特定事件的任何进程都可以使用信号开始事件触发新实例。

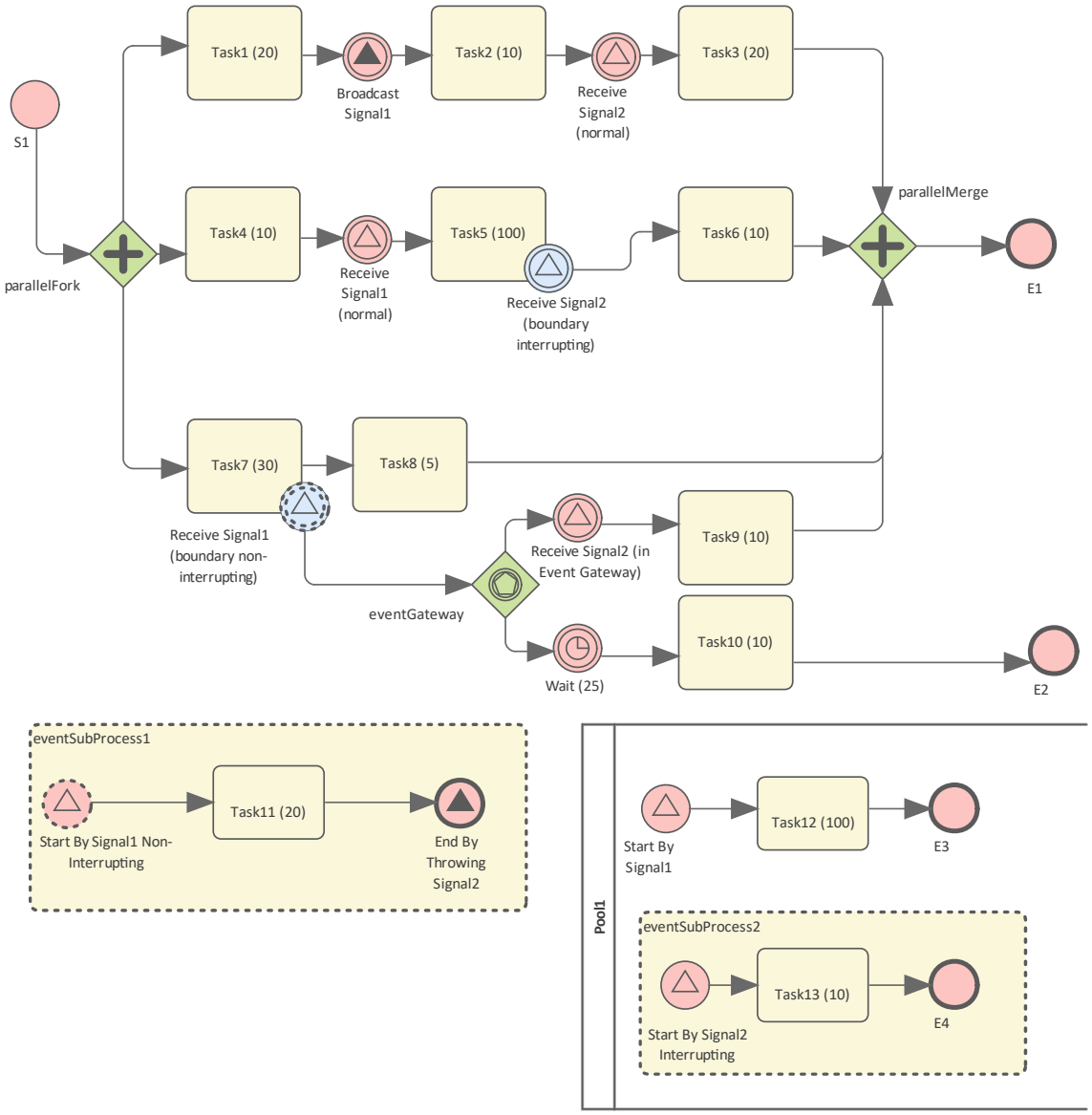
信号可以A抛出中间事件或抛出结束事件中抛出，并且可以在开始事件或捕获中间事件（包括边界信号事件）中捕获。

在此示例中，我们通过 BPSim 参数设置演示了这些信号事件及其对任务生命线的影响。

- 开始信号事件：
  -在顶层进程（Pool1）中通过 *Signal1*开始
  开始*Signal2*中断事件子流程*eventSubProcess2*
  开始*Signal1* Non Interrupting in event subprocess *eventSubprocess1*
- 抛出中间信号事件：
  -广播信号*1*
- 捕获中间信号事件：
  -接收信号*1*（正常）
  -接收信号*2*（正常）
  -接收信号*2*（边界中断）
  -接收信号 *1*（边界不中断）
  -接收信号 *2*（在事件网关中）
- 结束信号事件：
  -以抛出 *Signal2* 结束

创建 **BPMN**模型

为了演示通过协作信号事件跨进程进行通信的能力，我们创建了一个具有主泳池和另一个泳池（信号）中的进程的模型。



创建协作和主流程

创建一个新的协作协作图*CollaborationForTestingSignalEvents* ，（选择 在名为 BP 的新协作模型中创建此图表"选项）。右键单击图表名称窗口并选择 封装进程浏览器"选项。

创建了A泳池和一个流程*BusinessProcess_PoolMain* ，并使用自动值设置这些标签：

- *CollaborationForTestingSignalEvents.mainPool*设置为*PoolMain*
- *PoolMain.processRef*设置为*BusinessProcess_PoolMain*

为主流程创建元素

创建一个开始事件*S1*并将一个序列流添加到一个分叉Parallel Gateway *parallelFork*

将序列流添加到：

- 一个任务*(20)*然后添加这个序列流链：
  - 到一个 Throwing 中间信号事件广播 *Signal1*
  - 然后到一个任务*(10)*
  -然后到一个捕获中间信号事件接收*Signal2*（正常）
  - 然后到一个抽象任务*(20)*
  - 然后到合并并行网关*parallelMerge*

- 然后结束事件*E1*

- 一个任务*(10)*然后添加这个序列流链：
  - 到一个捕获的中间信号事件接收信号接收正常）
  - 然后到一个任务*(100)*，在它上面创建一个边界捕获中间信号事件接收*Signal2*（边界中断）
  - 然后到一个抽象任务*(10)*
  - 然后到前面的 Merge Parallel Gateway *parallelMerge*
- 一个任务*(30)* ，然后添加这个序列流链：
  - 抽象任务*8 (5)*
  - 然后到前面的 Merge Parallel Gateway *parallelMerge*

在边界*(30)*上，创建一个事件非中断捕获中间信号接收信号接收（边界非中断）。将序列流添加到事件网关 *eventGateway* ，并将序列流添加到：

- A捕获中间信号事件事件接收*2*（在网关中），然后这个序列流：
  - 抽象任务*9 (10)*
    - 然后到前面的 Merge Parallel Gateway *parallelMerge*
- A捕获中间定时器事件*Wait (25)* ，然后是这个序列流链：
  - 抽象任务*10 (10)*
  - 然后结束事件*E2*

在主进程中创建事件子进程（由非中断的开始信号事件触发）

- 创建一个活动*eventSubProcess1* ，并在其 属性"对话框中，将 类型"字段设置为*subProcess*并将属性 "triggeredByEvent"更改为*true*
- 在*eventSubProcess1*中，通过开始开始-*Interrupting*创建一个开始事件开始，并在其 属性"对话框中，将 类型"字段设置为事件-*Process Non-Interrupting* >信号
- 将序列流添加到目标任务*(20)*
- 通过 *Throwing Signal2*将序列流添加到目标事件，并在元素 属性"对话框中，将 类型"字段设置为信号

创建另一个进程

- 从工具中，将 泳池"图标拖放到图表上，并将元素池命名为工具箱
- 在浏览器窗口中右击*Pool1* ，选择 封装进程"选项；创建了一个流程*BusinessProcess_Pool1*并将标签 "Pool1.processRef"设置为*BusinessProcess_Pool1*

为*Pool1*创建主进程

- 通过开始信号一个信号开始事件开始
- 将序列流添加到目标任务*(100)*
- 将序列流添加到目标结束事件*E3*

创建Event子进程事件*Pool1*

- 创建一个活动*eventSubProcess1* ，并在 属性"对话框中，将 类型"字段设置为*subProcess* ；将属性 "triggeredByEvent"更改为*true*
- 在*eventSubProcess2*中，通过 *Signal2*开始创建一个开始事件开始，并在 属性"对话框中，将 类型"字段设置为事件子进程中断>信号
- 将序列流添加到目标任务*(10)*
- 将序列流添加到目标结束事件*E4*

创建信号元素并配置信号事件

在 BPMN 2.0工具箱中，展开 BPMN 2.0 - Types"页面并将 信号"图标拖到图表上；命名 *ElementSignal1*元素再次将图标拖到图表上以创建*Signal2* 。这些是根元素（所有进程都可以使用），因此它们将直接在模型包下创建。

双击每个信号事件元素，然后在 信号"标签的 值"字段中，单击 ... 按钮并浏览到适当的信号元素。

提示：或者，您可以将浏览器窗口中的信号元素拖放到图表中的事件元素上；将显示一个上下文菜单，您可以从中选择 设置信号参考"选项。

- 将 signalRef 设置为 "signal1"：
  -广播信号*1*
  -顶级进程（开始）中的*Signal1*开始
  -在事件子进程*eventSubprocess1*中通过开始不中断
  -接收信号*1*（正常）
  -接收信号 *1*（边界不中断）
- 将 signalRef 设置为 "signal2"：
  开始*Signal2*中断事件子进程*eventSubProcess2*
  -接收信号*2*（正常）
  -接收信号*2*（边界中断）
  -接收信号 *2*（在事件网关中）

配置 BPSim

在本节中，我们创建配置工件，指定模型包并配置每个元素的值。

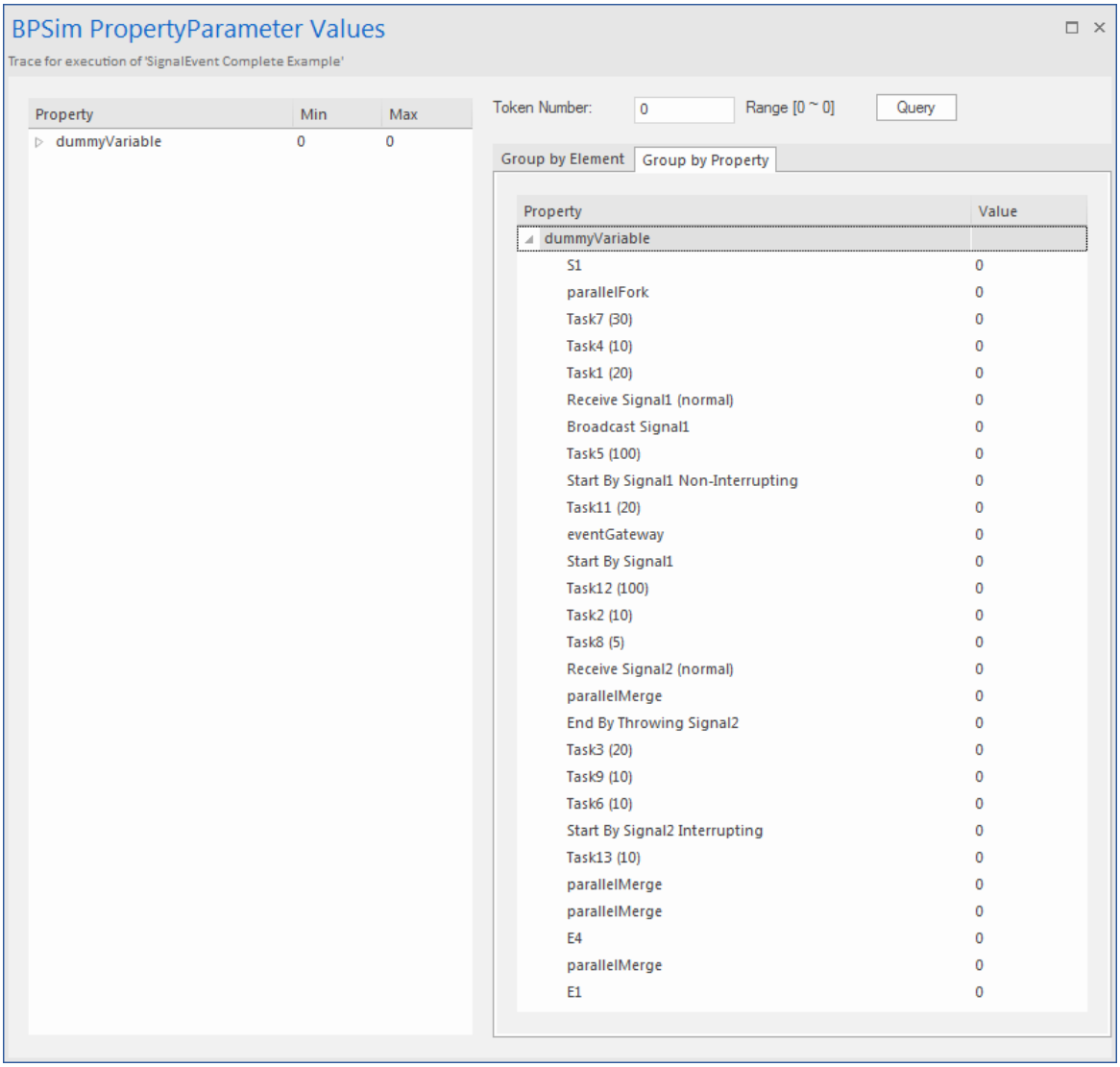
配置非常简单，因为所有信号事件都不需要任何信号配置。我们所要做的就是设置任务的处理时间，这样我们就可以观察进程、线程和任务是如何启动和中断的。

任务	描述
设置配置	• 打开配置 BPSim窗口（'仿真>过程分析>进程>打开BPSim 管理器'） • 创建加新元素完全工件示例'选择/创建工件…按钮并单击父包并单击按钮，然后在示例按钮上输入，然后键入单击保存按钮和确定按钮） 然后所有的 BPMN 元素将被加载到配置 BPSim窗口中。
非信号事件	• 在对话框左侧的元素列表中，展开 "StartEvent"组，然后单击*S1*和 控件"选项卡；单击 新参数"下拉箭头并选择 触发器计数"，然后在 值"字段中键入 "1 " • 展开 "IntermediateEvent"组，然后单击*Wait (25)*和 控件"选项卡；单击 新参数"下拉箭头并选择 "InterTriggerTimer"，然后单击 值"字段中的…按钮；选择 "Constant"和 "Numeric"，在 "Constant Numeric"字段中输入 "25"，在 "TimeUnit"字段中输入 "seconds"
进程虚拟变量	模拟控制器显示一个列表，显示每个元素的运行时令牌计数。例如，在模拟中，有 4 个令牌通过了*Gateway*元素。这对于某些统计和分析非常有用。但是，它没有显示在模拟期间何时遍历了*parallelMerge* 。为了获得单个令牌的准确跟踪，我们使用属性跟踪实用程序，它依赖于属性参数。所以我们创建了一个虚拟参数。 在 配置"对话框中，展开 "业务流程"组。 • 单击*BusinessProcess_Main*和 属性"选项卡，然后用属性改写属性；在 值"字段中，单击…按钮并单击 常量"和 数字"，然后在 常量数字"字段中键入 "0" • 单击*BusinessProcess_Pool1*并执行与*BusinessProcess_Main*完全相同的操作
任务处理时间	展开 活动"组和此处列出的每个任务元素：选择 时间"选项卡，单击 新参数"下拉箭头并选择 处理时间"，然后单击 值"列上的…按钮，选择 常量"和 "数字"，在 常量数字"字段中键入指示的值，然后在 时间单位"字段中选择 秒"。 • 任务 1 (20)：20 秒 • 任务 2 (10)：10 秒

	• 任务 3 (20)：20 秒 • 任务 4 (10)：10 秒 • 任务 5 (100)：100 秒 • 任务 6 (10)：10 秒 • 任务 7 (30)：30 秒 • 任务 8 (5)：5 秒 • 任务 9 (10)：10 秒 • 任务 10 (10)：10 秒 • 任务 11 (20)：20 秒 • 任务 12 (100)：100 秒 • 任务 13 (10)：10 秒

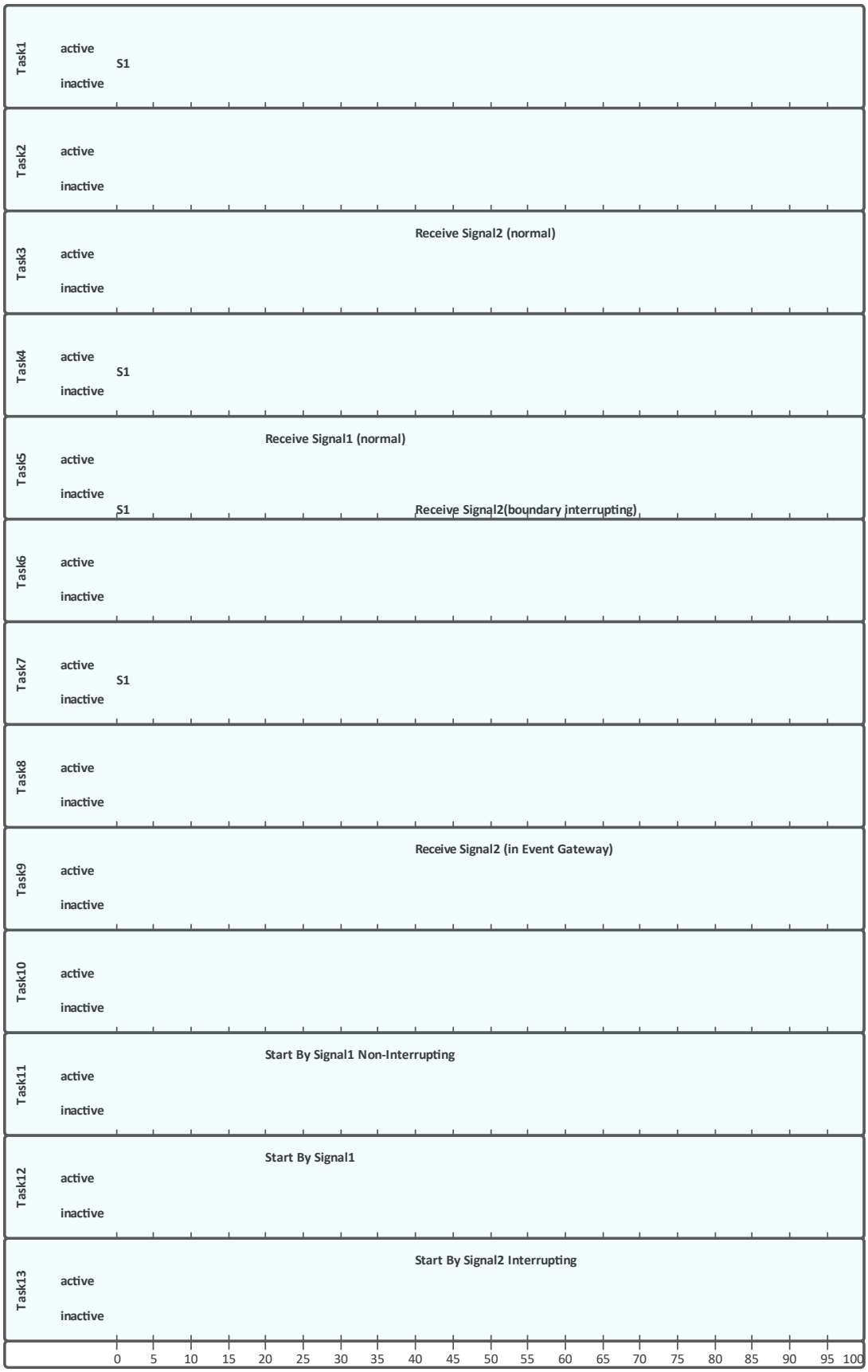
运行仿真

- 在 配置 BPSim ″对话框工具 上，单击 迤行″图标打开 迤行仿真控制器″对话框
- 点击 迤行″图标下拉箭头并选择 标准仿真″
- 仿真完成后，单击工具栏上的  按钮，显示 BPSim PropertyParameter Values″对话框
- 单击 查询属性″按钮和 按属性分组″选项卡，然后展开 dummyVariable”



分析

从模拟的直接结果来看，发生了什么可能并不明显；但是，如果我们为每项任务画出生命线，就会变得非常清晰。



- *Task1*、*Task4*和*Task7*并行启动

- *Task2*在*Task1*完成后立即开始（没有在投掷事件处停止）
- 在 20 秒时， *Signal1*由 Throwing IntermediateEventBroadcast事件广播，并且：
  -接收信号 *1*（正常）被激活，任务 *5*开始
  开始*Signal1 Non-Interrupting*已激活，并且*eventSubProcess1*中的*Task11*已启动
  开始*Signal1*已激活，并且*Pool1*中的 *Task12*已启动
- 在 40 秒时，事件*By Throwing Signal2*广播了 Signal2，并且：
  -接收信号 *2*（正常）被激活并且任务 *3*开始
  - *Task5*被中断， *Task6*开始
  -接收信号 *2*（在事件网关中）被激活并且任务 *9*开始
  -开始通过 *Signal2* 中断被激活，并且：
  > *Pool1*中的主进程被中断， *Task12*停止
  > *eventSubProcess2*中的*Task13* 已启动
- 50 秒到达*E4*时， *BusinessProcess_Pool1*中的*eventSubProcess2*完成
- 在 60 秒到达*E1*时， *BusinessProcess_MainPool*完成
- Intermediate事件*(25)*没有被激活，因为网关中的信号事件首先被激活；结果， *Task10*从未启动

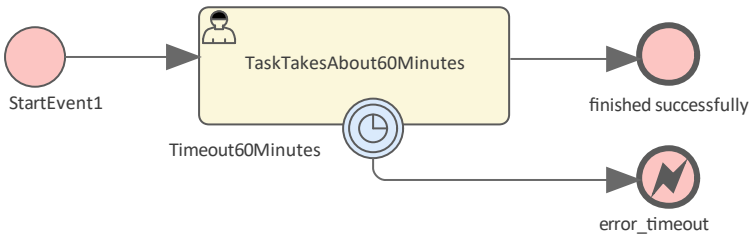
注记：每个任务的实际运行时间可以从生成的 BPSimReport元素中观察到，通过：

1. 双击 <<BPSimReport>>元素。
2. 扩展 时间"组。
3. 扩展任务元素。
4. 检查 任务总时间"。

例如*ElementTask5 (100)* ，虽然我们设置它的processingTime为 100 秒，但**Total时间是 20 秒，任务**元素秒时被接收信号 *2*（边界中断）中断。

边界事件

创建 BPMN模型



- 创建一个开始事件*StartEvent1*
- 给目标用户添加一个序列流任务*TaskTakesAbout60Minutes*
- 将序列流添加到目标结束事件成功完成
- 创建一个中间事件，将工具箱中的图标拖放到工具箱上；从自动菜单中选择 'Edge-Mounted' 和 'Timer'，然后调用元素
- 将序列流添加到目标事件(Error) *error_timeout*

配置 BPSim

在本节中，我们创建配置工件，识别父包并设置每个元素的参数值。

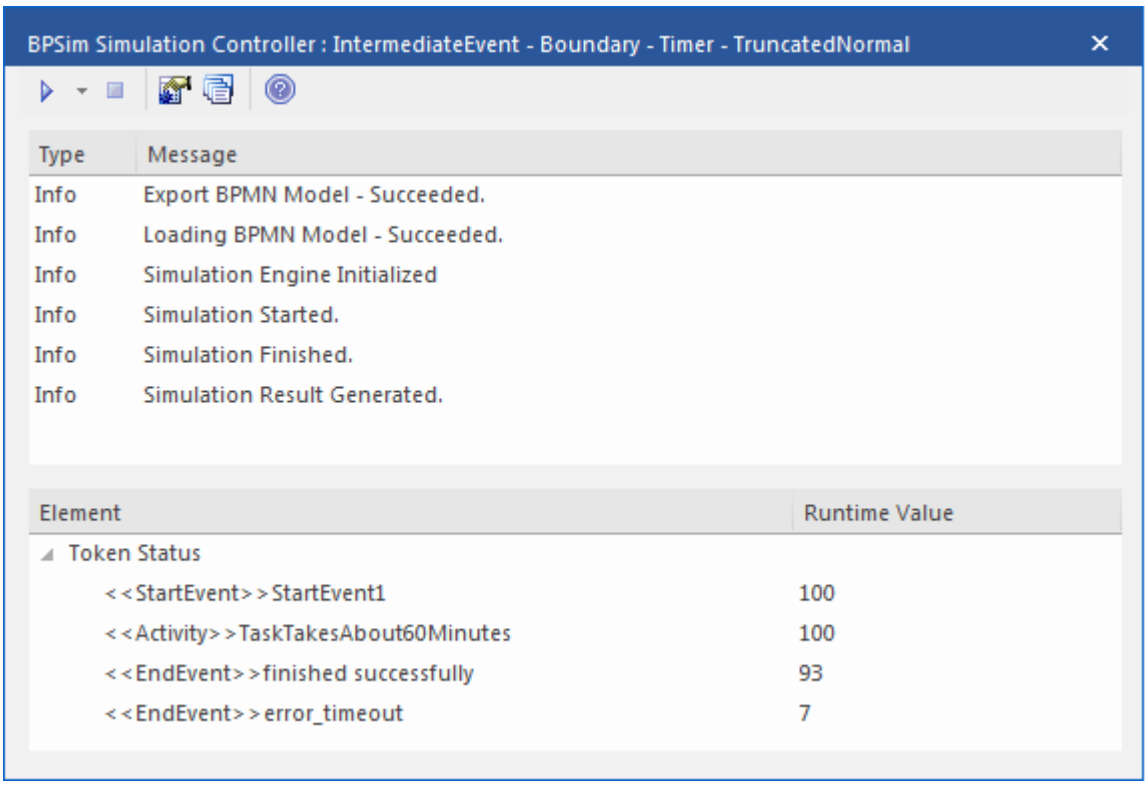
对象	行动
创建工件包	• 打开 配置 BPSim ”对话框（ 仿真>流程分析>仿真>进程BPSim 管理器”） • '在'创建新包元素名称'字段中的'创建资源包/工件'字段中选择 'Intermediate'边界工件，然后单击[...]按钮，选择其加新组件，单击按钮，然后单击在保存按钮和确定按钮上） 然后所有的 BPMN 元素将被加载到 配置 BPSim ”对话框中。
开始事件1	在对话框左侧的元素列表中，展开 StartEvent”组并单击*StartEvent1*。 • 单击 控件”选项卡 • 单击 新参数”下拉箭头并选择 TriggerCount” • 在 值”字段中输入 100”
任务大约需要60分钟	在对话框左侧的元素列表中，展开 活动”组并单击 *TaskTakesAbout60Minutes*。 • 点击 时间”标签 • 单击 新参数”下拉箭头并选择 ProcessingTime” • 在 值”字段中单击[...]按钮并选择 分布”和 截断正常” • 在 平均值”字段中输入 50” • 在 标准偏差”字段中输入 10” • 在 Min”字段中输入 0” • 在 最大”字段中输入 1000”

	• 点击确定按钮
超时60分钟	在对话框左侧的元素列表中，展开 "IntermediateEvent"组并单击 *Timeout60Minutes*。 • 单击 "控件"选项卡 • 单击 "新参数"下拉箭头并选择 "InterTriggerTimer" • 将值设置为 "000:000:000 001:00:00"（即1小时）

运行仿真

- 在配置 BPSim窗口工具栏上，单击 "运行"图标，打开 "运行仿真控制器"对话框
- 点击 "运行"图标下拉箭头并选择 "标准仿真"
- 仿真完成后，单击工具栏上的  按钮，显示 "BPSim PropertyParameter Values"对话框
- 单击 "查询属性"按钮和 "按属性分组"选项卡，然后展开 "dummyVariable"

在模拟中，我们得到以下结果：



分析

由于*TaskTakesAbout60Minutes*的ProcessingTime被设置为一个分布值，结果是：

- 100 人中有 93 人在1小时内完成，所以正常流程成功完成才生效
- 100 个中有 7 个在超过1小时内完成，因此*error_timeout*的异常流程生效

其它配置

示例中还有另外两个业务流程仿真管理文件夹，分别设置了工件的常量值50分钟和80分钟，其他设置保持不变。

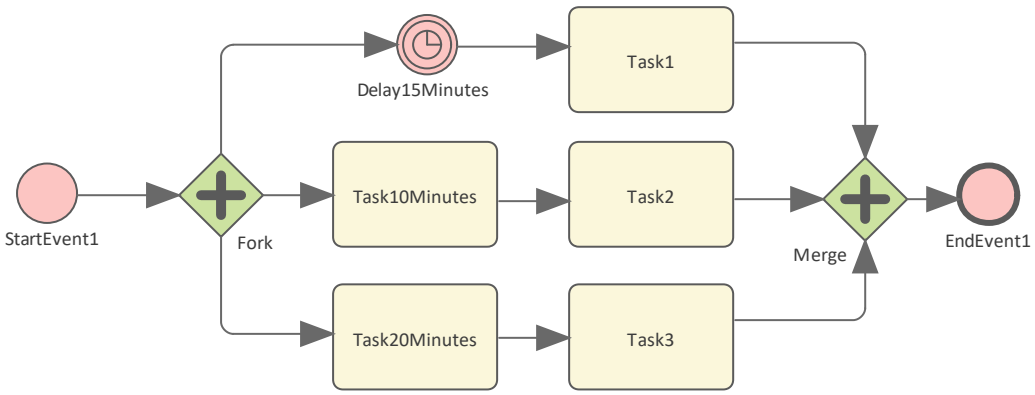
在这两个上运行工件：

- 配置为 50 分钟的处理时间始终以正常流程完成
- 配置为 80 分钟的处理时间始终在异常流中完成

事件独立中间事件

当 Timer事件在正常序列流中用作独立元素时，它充当延迟机制。

创建 BPMN模型



- 创建一个名为开始的启动事件
- 向名为分叉的目标并行网关添加一个序列
- 将序列流添加到：
  - A名为*Delay15Minutes*的独立定时器中间事件，并由此产生一个序列流到一个名为*Task1*的活动
    - 一个名为*Task10Minutes*的活动，从那个序列流到一个名为*Task2*的活动
    - 一个名为*Task20Minutes*的活动，从那个序列流到一个名为*Task3*的活动
- 从*Task1* 、 *Task2*和*Task3*创建序列称为*Merge*的合并并行网关
- 将序列流添加到名为*EndEvent1* 的目标 *EndEvent*

配置 BPSim

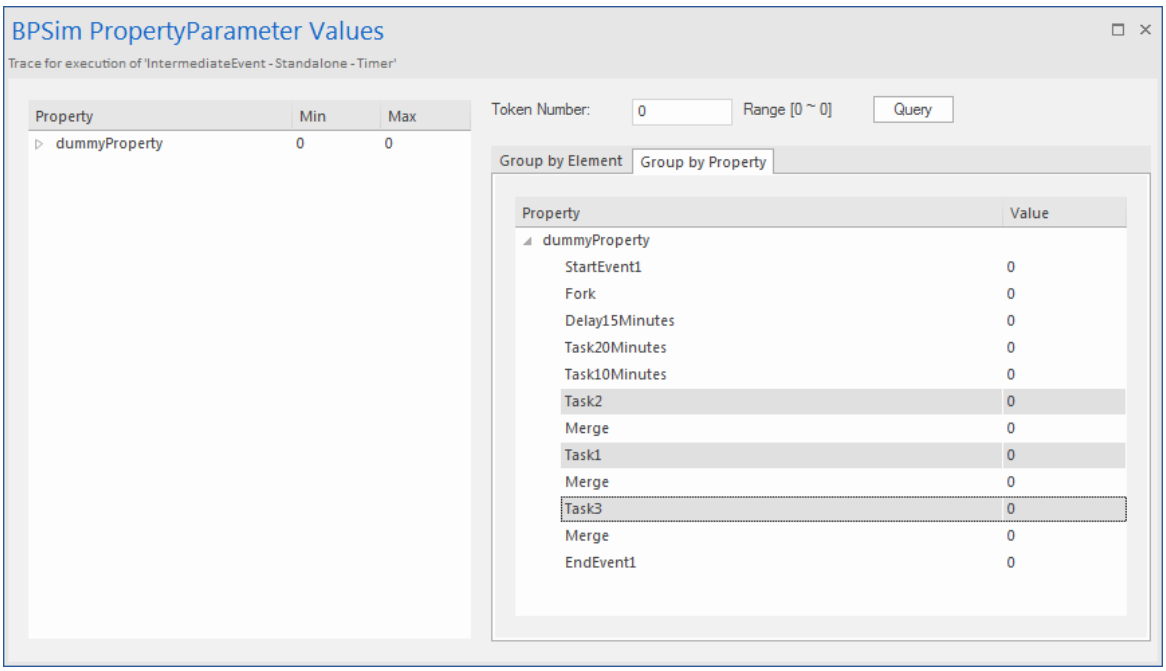
在本节中，我们创建配置工件，指定模型包并配置每个元素的值。

物件	行动
创建工件包	• 打开 配置 BPSim "对话框（ 仿真>流程分析>仿真>进程BPSim 管理器"） • 创建一个工件名称，然后单击 "IntermediaEvent"按钮，在 选择/创建工件 "字段中单击 选择/创建独立"..."按钮并选择其父包，然后在元素名称中输入 加新"并单击保存按钮和确定按钮） 然后所有的 BPMN 元素将被加载到 配置 BPSim "对话框中。
开始事件1	• 在对话框左侧的元素列表中，展开 "StartEvent"，然后单击*StartEvent1*和 控件"组 • 单击 新参数"下拉箭头并选择 触发器计数"，然后在 值"字段中键入 "1" • 单击 属性"选项卡 • 用*dummyProperty*改写属性；在 值"字段中，单击"..."按钮并单击 常量"和 数字"，然后在 常量数字"字段中键入 "0"

	使用此属性， 属性跟踪”对话框将能够在模拟期间显示序列的元素。
延迟15分钟	• 在左侧的元素列表中，展开 IntermediateEvent”组，然后单击 *Delay15Minutes*并在 控件”选项卡上 • 单击 新参数”下拉箭头并选择 InterTriggerTimer”，然后将 值”字段设置为 15 分钟（“000:000:000 000:15:00”）
任务10分钟	• 在对话框左侧的元素列表中，展开 活动”组，然后单击*Task10Minutes*和 时间”选项卡 • 单击 新参数”下拉箭头并选择 处理时间”，然后将 值”字段设置为 10 分钟（“000:000:000 000:10:00”）
任务20分钟	• 在对话框左侧的元素列表中，展开 活动”组，然后单击*Task20Minutes*和 时间”选项卡 • 单击 新参数”下拉箭头并选择 处理时间”，然后将 值”字段设置为 20 分钟（“000:000:000 000:20:00”）

运行仿真

- 在 配置 BPSim ”对话框工具 上，单击 运行”图标打开 运行仿真控制器”对话框
- 点击 运行”图标下拉箭头并选择 标准仿真”
- 仿真完成后，单击工具栏上的  按钮，显示 BPSim PropertyParameter Values”对话框
- 单击查询按钮和 按属性分组”选项卡



分析

分叉并行网关将同时激活传出序列流（顺序未定义且不重要）。但是，我们希望任务的顺序是准确的：

- 任务*2*
- 任务*1*

- 任务*3*

此顺序由在两个活动 (ProcessingTime) 和计时器中间事件(InterTriggerTimer) 上设置的 BPSim 参数确定。“ 列 PropertyParameter Values″对话框中显示的 列确认*Task2*领先于*Task1* ，后者领先于*Task3* 。

画墙进程仿真(调用活动)

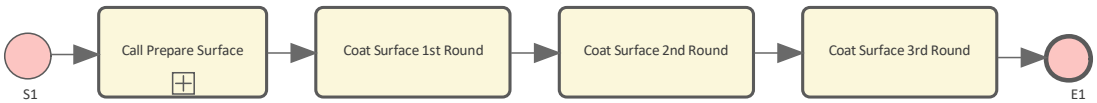
这是一个模拟墙壁粉刷过程的简单示例。我们将主要过程定义为准备表面，然后将其绘制 3 次。准备表面进一步分为打磨和清洁等任务。

我们假设涂上三层油漆中的每一层都是相同的过程，只是在每层油漆上随机花费的时间可能不同。

创建 BPMN模型

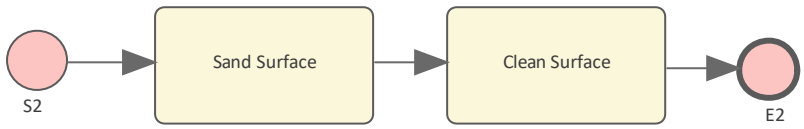
该模拟对两个过程进行。

主要流程 - Paint Wall进程



1. 创建一个名为*S1*的开始事件。
2. 将一个序列流添加到一个名为调用*Prepare Surface* 的目标调用调用。
3. 将序列流添加到名为*Coat Surface 1st Round* 的目标 *callGlobalTaskActivity*。
4. 将一个序列流添加到名为*Coat Surface 2nd Round* 的目标 *callGlobalTaskActivity*。
5. 将一个序列流添加到名为*Coat Surface 3rd Round* 的目标 *callGlobalTaskActivity*。
6. 将序列流添加到名为*E1*的目标结束事件。

重用流程进程Surface流程



1. 创建一个名为*S2*的开始事件。
2. 将一个序列流添加到一个名为*Sand Surface*的目标抽象任务中。
3. 将一个序列流添加到一个称为清理表面的目标抽象任务。
4. 将序列流添加到名为*E2*的目标结束事件。

设置任务用流程来调用活动

1. 创建一个名为*Coat Surface*的全局任务活动。
2. 双击*Coat Surface 1st Round*、*Coat Surface 2nd Round*和*Coat Surface 3rd Round*中的每一个，并将标签 称为ActivityRef"设置为*Coat Surface*。
   提示：也可以从浏览器窗口中将全局任务 涂层表面"拖放到调用活动元素上，单击上下文菜单上的 设置调用活动引用"选项。
3. 双击调用*Prepare* ，设置 调用ActivityRef"标签为*Prepare Surface*进程。
   提示：您也可以从浏览器窗口中将 Prepare Surface进程元素"进程拖放到调用活动中，单击上下文菜单上的"Set调用"选项。

配置 BPSim

1. 打开 配置 BPSim "对话框（ 仿真>流程分析>仿真>进程BPSim 管理器"）。
2. 点击图标仿真创建一个业务流程仿真管理名工件名

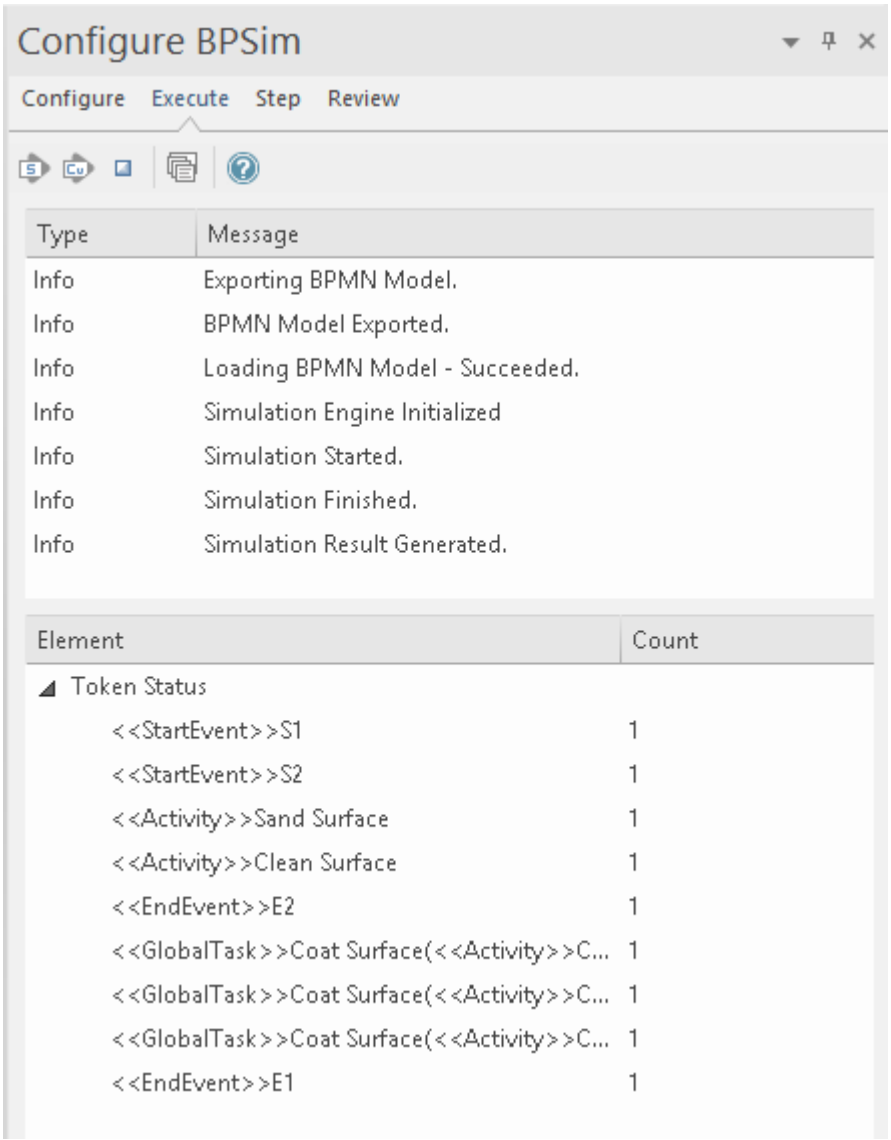
3. 单击图标并选择包含相应 BPMN 2.0模型的包。

物件	活动
固定缩放时间	1. 打开 "Prepare Surface"进程图。 2. 单击活动沙面，在配置 BPSim窗口中，单击新参数下拉箭头并创建一个名为 "处理时间"的时间参数。 3. 在 "值"字段中，将设置更改为 0 00:30:00（即 30 分钟）。 4. 单击活动清理，然后在配置 BPSim窗口中，单击新参数下拉箭头并创建一个名为 "处理时间"的时间。 5. 在 "值"字段中，将设置更改为 0 00:10:00（即 10 分钟）。 6. 单击图标。
随机涂层时间	1. 在浏览器窗口中，单击全局任务活动涂层表面。 2. 在配置 BPSim窗口中，单击新参数下拉箭头并创建一个名为 "处理时间"的时间参数。 3. 在 "值"字段中单击按钮，在参数对话框中单击 "分布"选项卡和 "泊松"。 4. 在 "平均值"字段中输入 "10"，然后点击确定按钮。 5. 单击图标。 使用此设置，泊松分布生成的随机数的平均值为 10。如果您愿意，可以选择其他类型的分布。
S1 上的触发计数	在 "Paint Wall"进程图中，单击开始事件*S1*。 1. 在配置 BPSim窗口中，单击参数parameter 下拉箭头并创建一个名为 "控件"的控件参数。 2. 在 "值"字段中输入 "1"。 3. 单击图标。

运行仿真

1. 在配置 BPSim窗口的 "执行"选项卡上，单击图标。

当模拟完成时，它会提供类似于以下的结果：



流量分析

对于在*S1*上启动的唯一令牌，我们可以从配置 BPSim窗口的 执行"选项卡中看到流程是如何发展的：

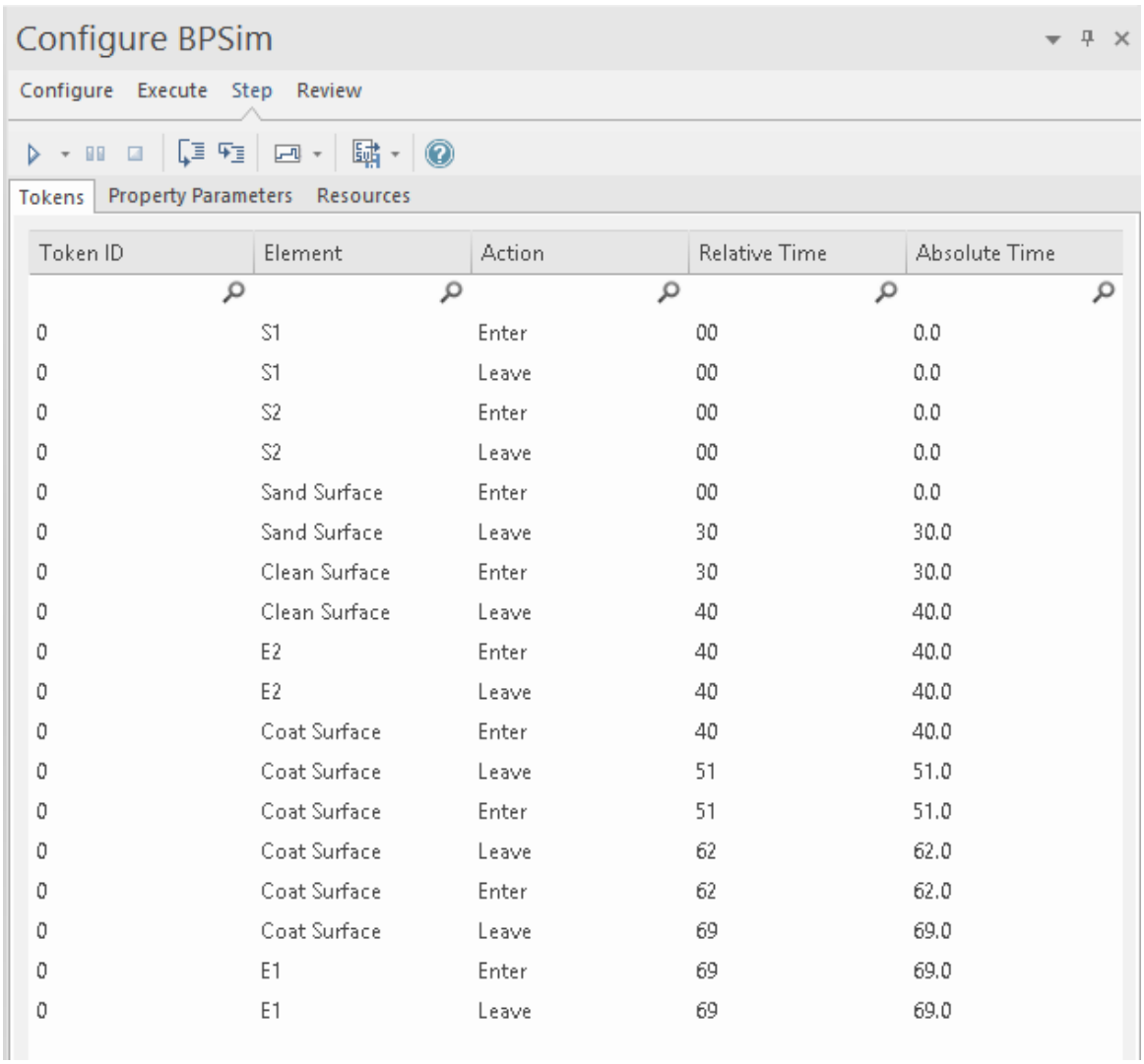
Element	Count
Token Status	
<<StartEvent>>S1	1
<<StartEvent>>S2	1
<<Activity>>Sand Surface	1
<<Activity>>Clean Surface	1
<<EndEvent>>E2	1
<<GlobalTask>>Coat Surface(<<Activity>>Coat Surface 1st Round)	1
<<GlobalTask>>Coat Surface(<<Activity>>Coat Surface 2nd Round)	1
<<GlobalTask>>Coat Surface(<<Activity>>Coat Surface 3rd Round)	1
<<EndEvent>>E1	1

- 当到达callProcessActivity时，被调用的进程被激活；所以我们有*S2 ~ E2*

- 当到达 callGlobalTaskActivity 时，被调用的任务 被激活 - 符号为： *GlobalTask name*（被称为活动名称）；
  全球*Coat Surface*被调用了 3 次：
  -涂层表面（涂层表面第一轮）
  - 涂层表面（涂层表面第二轮）
  - 涂层表面（涂层表面第三轮）

时间分析

单击配置 BPSim窗口的 步骤”选项卡，然后单击类似于下图的 令牌”选项卡：



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

您可以按原样检查列表中的时间，但为了使过程更容易，在 行动”列的过滤器栏字段中键入 离开”以仅显示该列中包含该文本string的记录。

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

报告显示如图所示，我们可以进行这样的分析：

- 调用活动调用准备表面耗时 40 分钟，由沙面（30 分钟）和清理表面（10 分钟）组成，定义为
- 涂层表面（第一轮）耗时 11 分钟；涂层表面（第 *2* 轮）耗时 11 分钟；涂层表面（第 *3* 轮）耗时 7 分钟 - 数字 11、11、7 由 Poisson(10) 分布随机生成；这里重要的是任务的每个调用实例都有自己的值
- 涂层表面从所有实例中收集的总时间：11 + 11 + 7 = 29
- *Paint Wall*进程的求和处理时间为69分钟，由四个调用活动组成：40 + 11 + 11 + 7 = 69

定制仿真

我们可以在 BPMN 元素上配置一个 "结果请求"来自定义模拟报告，以便我们只报告我们感兴趣的参数。

配置结果请求

1. 在 "Paint Wall"进程图中，单击 "活动*Coat Surface 1st Round*"。
2. 在配置 BPSim窗口中，单击新参数下拉箭头并创建一个名为 "ProcessingTime"的时间参数。
3. 单击  工具栏图标。 "结果请求"列显示在 "参数"列的右侧；单击下拉箭头并选择 "总和"复选框。点击确定按钮。
4. 在 "值"字段中输入 "1"。
5. 单击  图标。
6. 重复步骤1到 5 进行活动调用准备表面、涂层表面第 *2* 轮、涂层表面第 *3* 轮
- 展开*Paint*进程流程组的 "业务流程"并重复这些步骤

运行仿真

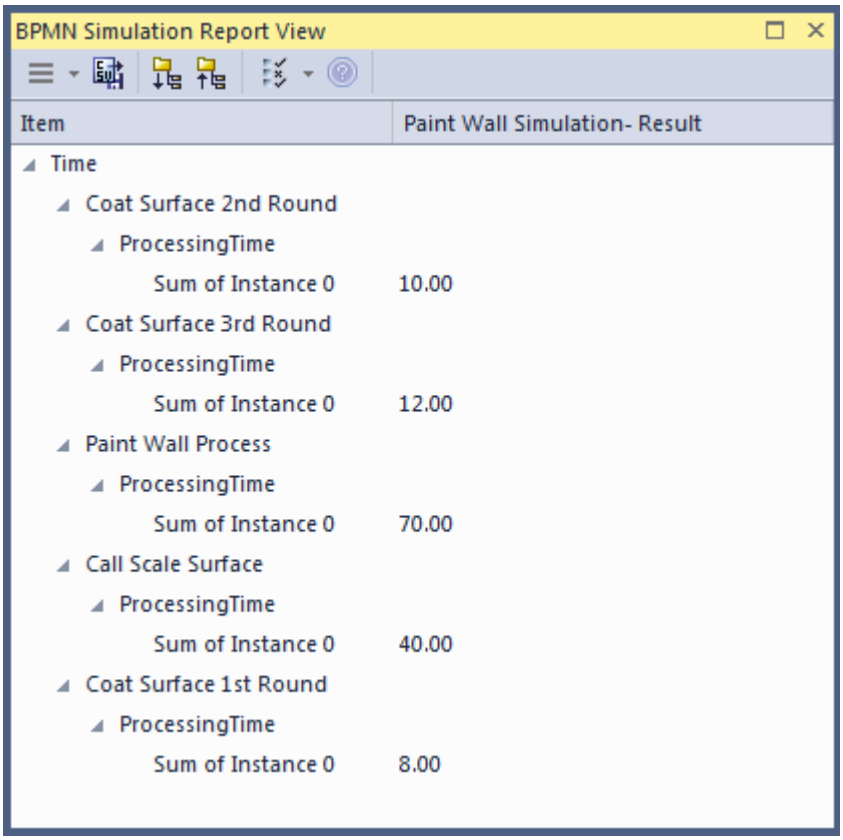
- 在 "配置 BPSim"对话框工具 上，点击运行按钮； "仿真仿真控制器"对话框显示
- 单击运行按钮下拉箭头，在本例中选择 "定制仿真"

流量分析

流量分析与标准仿真完全相同。

时间分析

在 仿真仿真  控制器； “BPMN仿真报告视图”显示。



时间分析与标准仿真相同；但是，该报告仅包含我们要求的 总和”结果。

注记：目前，在时间分析中，我们不能在被调用进程本身或被调用进程包含的活动上请求ProcessingTime。如果您有此要求，请使用标准仿真。

BPSim 成本参数

BPSim 1 .0 提供了设置成本参数和从过程模拟实验接收成本统计数据的方法。 BPSim 提供了一个框架来根据两个参数确定**可变**成本，这两个参数都与模拟过程中执行的活动水平有关。这些参数是：

- 完成成本（BPSim 规范中的 固定成本”）- 每次操作完成时产生的成本；该成本可以与任务、进程、子流程、调用活动或资源元素相关
- 时间成本（BPSim 规范中的 单位成本”）——任务、流程、子流程、调用活动或资源在一段时间内忙碌时产生的成本

活动、资源和流程支持成本参数。上：

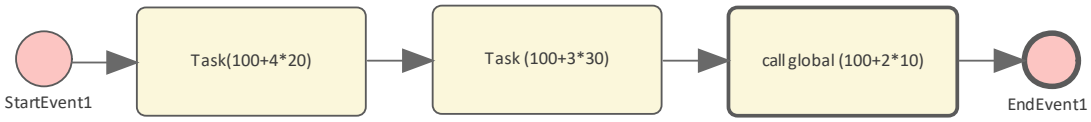
- 活动结束时都会产生活动、完成成本和时间成本（单位成本 *时间）
- 每当每个涉及的资源完成A活动时，都会产生资源、完成成本和时间成本
- A进程，完成成本和时间都是在一个进程完成的时候产生的

BPSim 不支持那些无需模拟即可知道的成本——例如，总体劳动力雇佣成本。

成本参数的配置和模拟通过两个例子来演示：

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

在活动上设置成本参数



创建 BPMN模型（活动）

1. 在浏览器窗口中，创建一个*StartEvent1* 、一个*GlobalTask1* 、两个 AbstractTask 和一个*EndEvent1*。
2. Ctrl+拖动浏览器窗口中的元素到图表上，将*GlobalTask* 1 粘贴为调用（调用活动），称为*call global (100+2*10)*。
3. 给元素命名并将它们与序列流连接起来；应该调用两个 AbstractTasks：
   -任务（*100+3*30*）和
   -任务（*100+4*20*）。

配置

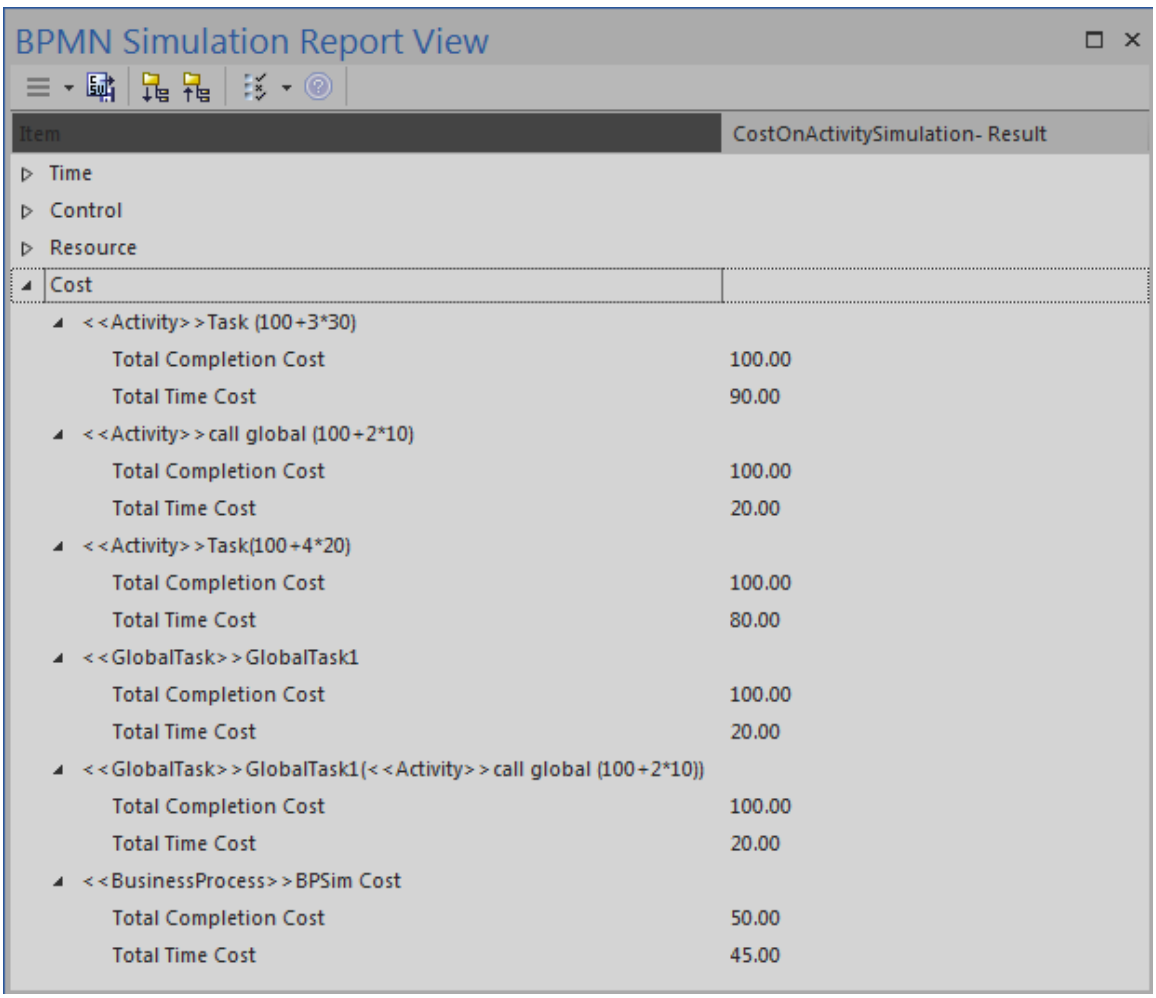
在图中创建业务流程仿真配置配置 BPSim工件选项。设置配置以链接到包含 BPMN模型元素的包，并按照指示配置这些 BPSim 参数。

参数	设置
场景参数	1. 点击配置工件和参数，对于 Scenario"TimeUnit'时间点击'Value' 下拉箭头 'hours'。 2. 在 持续时间"参数的 值"字段中，将值设置为 "0001 00:00:00"（ 1天）。 此时间单位用于计算时间成本（时间成本 = 单位成本 *时间），因此请确保单位成本基于正确的时间单位。
控件参数	1. 在图表上，单击*StartEvent1*。 2. 单击新参数下拉箭头并选择 控件"。 3. 在 参数"字段中，单击下拉箭头并选择 "TriggerCount"。 4. 在 值"字段中输入 "1 "。
时间参数	1. 在图表上单击任务（*100+4*20*） 。 2. 单击新参数下拉箭头并选择 时间"。 3. 单击 参数"下拉箭头并选择 "ProcessingTime"。 4. 在 值"字段中，将值设置为 "000:000:000 004:00:00"（4 小时）。 5. 单击图表上的任务（*100+3*30*）并重复步骤 2、3 和 4，将 值"字段设置为 "000:000:000 003:00:00"（3 小时）。 6. 单击图表上的*GlobalTask* 1 并重复步骤 2、3 和 4，将 值"字段设置为 "000:000:000 002:00:00"（2 小时）。
成本参数	1. 在图表上，单击任务*(100+4*20)* 。 2. 单击新参数下拉箭头并选择 成本"。 3. 在 参数"字段中单击下拉箭头并依次选择： - "FixedCost"，然后在 "Value"字段中单击 ... 按钮，选择

	常量”选项卡和 浮动”，以及 常量浮动”字段 输入 1'00”；点击确定按钮 - 'UnitCost' - 做同样的事情，将'Constant Floating' 字段设置为'20'。 4. 在图表上，单击任务（*100+3*30*）并重复步骤 2 和 3，设置： - '固定成本' 到 '100 - 单位成本”到 30”。 5. 在图表上，单击*GlobalTask1*并重复步骤 2 和 3，设置： - '固定成本' 到 '100 - 'UnitCost' 到 '10'。 6. 在图表上，单击*BPSim Cost*并重复步骤 2 和 3，设置： - '固定成本' 到 '50' - 'UnitCost' 到 '5'。

仿真

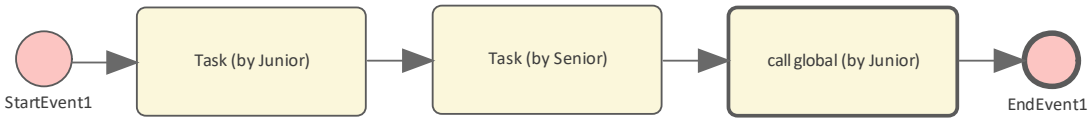
1. 在 配置 BPSim ”对话框中，单击 执行”选项卡。
2. 单击 按钮。
3. 模拟完成后，单击 审阅”选项卡和 标准结果报告”选项卡。
4. 过滤器通过单击 按钮并选择 仅显示非空项”选项来过滤报告。



分析

活动	分析
任务(100+4*20)	• Total Completion Cost 为 100，与 BPSim 中的 FixedCost (100) 设置相匹配 • 时间为 80，计算方式为 ProcessingTime (4 hours) * UnitCost (20/hour)
任务(100+3*30)	• Total Completion Cost 为 100，与 BPSim 中的 FixedCost (100) 设置相匹配 • 时间为 90，计算为 ProcessingTime (3 hours) * UnitCost (30/hour)
调用全局（100+2*10）	• Total Completion Cost 为 100，与 BPSim 中*GlobalTask* 1 设置的 FixedCost (100) 匹配 • 时间为 20，计算为 GlobalTask1 上的 ProcessingTime (2 hours) * *UnitCost* (10/hour)
BPSim 成本流程	• Total Completion Cost 为 50，与 BPSim 中的 FixedCost (50) 设置相匹配 • 时间为 45，计算为所有任务的 Total ProcessingTime (4 + 3 + 2 = 9 hours) * UnitCost (5/hour)

在资源上设置成本参数



创建BPMN模型(资源)

1. 在浏览器窗口中创建一个*StartEvent1* 、一个*GlobalTask1* 、两个名为 Task任务*by Junior)*和*Task(by Senior*任务的 abstractTask 以及一个*EndEvent1*。
2. Ctrl+将浏览器窗口中的元素拖到图表上，将*GlobalTask* 1 粘贴为名为*call global (by Junior)*的 Invocation (调用活动)。
3. 将元素与序列流连接起来。
4. 创建两个 BPMN2.0 资源元素：初级开发人员和高级开发人员。

配置

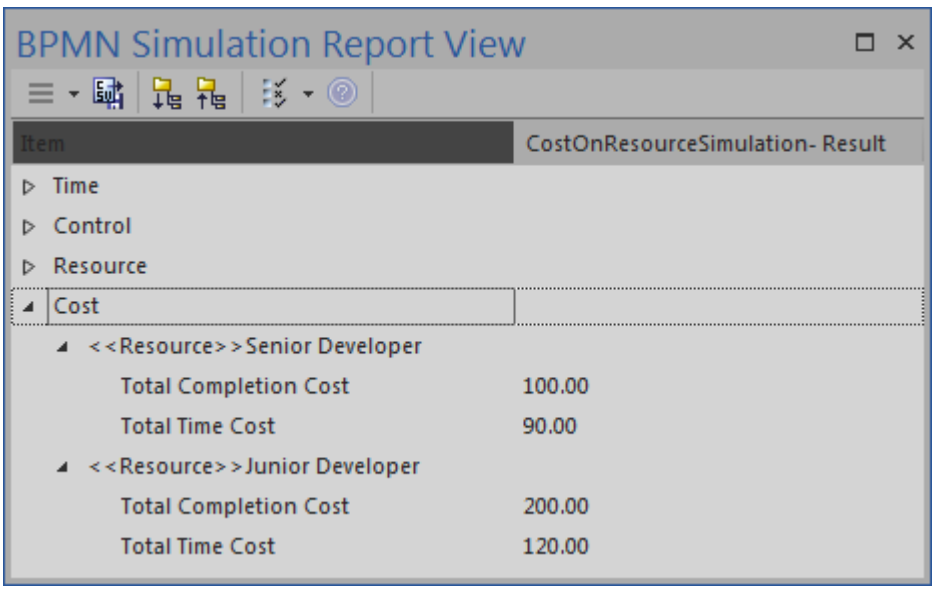
在图中创建业务流程仿真配置 BPSim工件选项，然后将配置设置为链接到包含BPMN元素的包，并按照指示配置这些模型参数。

参数	环境
场景参数	1. 点击配置工件和参数，对于 Scenario"TimeUnit'时间点击'Value' 下拉箭头 'hours'。 2. 在 持续时间"参数的 值"字段中，将值设置为 "0001 00:00:00"（ 1天 ）。 此时间单位用于计算时间成本（时间成本 = 单位成本 *时间），因此请确保单位成本基于正确的时间单位。
控件参数	1. 在图表上，单击*StartEvent1*。 2. 单击新参数下拉箭头并选择 控件"。 3. 在 参数"字段中，单击下拉箭头并选择 "TriggerCount"。 4. 在 值"字段中输入 "1 "。
时间参数	1. 在图表上点击任务*(by Junior)* 。 2. 单击新参数下拉箭头并选择 时间"。 3. 单击 参数"下拉箭头并选择 "ProcessingTime"。 4. 在 值"字段中，将值设置为 "000:000:000 004:00:00"（ 4 小时 ）。 5. 单击图表上的任务（按高级）并重复步骤 2、3 和 4，将 值"字段设置为 "000:000:000 003:00:00"（ 3 小时 ）。 6. 单击图表上的*GlobalTask* 1 并重复步骤 2、3 和 4，将 值"字段设置为 "000:000:000 002:00:00"（ 2 小时 ）。
资源参数	1. 在图表上，单击初级开发人员资源。 2. 单击新参数下拉箭头并选择 资源"。 3. 单击 参数"下拉箭头并选择 选择"。

	4. 在 值”字段中，单击 按钮以打开 编辑资源选择”对话框。 5. 单击 初级开发人员”，然后单击 按资源添加选择”按钮将选择移动到 资源或角色”面板。 6. 需要数量”列默认为 1”；用 10”改写这个值。 7. 单击 AND 单选按钮以设置逻辑关系；资源选择的最终表达式组成并显示在文本字段中。 8. 单击确定按钮返回到配置 BPSim确定，其中表达式显示在 值”字段中。 9. 单击高级开发人员资源并重复步骤 2 到 8，在 需要数量”字段中键入 “5”。
成本参数	1. 在图表上单击*Junior Developer* 。 2. 单击新参数下拉箭头并选择 资源”。 3. 单击 参数”下拉箭头并依次选择： - “FixedCost”，然后在 Value”字段中单击 按钮，选择 常量”选项卡和 浮动”，然后在 常量浮动”中 字段类型 100”和 货币单位”字段类型 AUD”；点击 确定按钮 - 'UnitCost' - 做同样的事情，将'Constant Floating' 字段设置为'20'。 4. 在图表上单击高级开发人员并重复步骤 2 和 3，设置： - '固定成本' 到 '100' - 单位成本”到 30”。

仿真

1. 在 配置 BPSim ”对话框中，单击 执行”选项卡。
2. 单击 按钮。
3. 模拟完成后，单击 审阅”选项卡和 标准结果报告”选项卡。
4. 过滤器通过单击 按钮并选择 仅显示非空项”选项来过滤报告。



分析

资源	结果
初级开发人员	• 总完成成本为 "200"，计算方式为 FixedCost (100) * 涉及的活动数量 (2) • 时间为 "120"，计算为 ProcessingTime (4 + 2 = 6 hours) * UnitCost (20/hour)
高级开发人员	• 总完成成本为 "100"，计算方式为 FixedCost (100) * 涉及的活动数量 (1) • 时间为 '90'，计算为 ProcessingTime (3 hours) * UnitCost (30/hour)

导出一个导出配置

当您在模型中定义了配置后，您可以将其导出为 XMI 文件以导入到其他项目中。配置所基于的BPMN 2.0模型也随配置一起导出。当您将 XMI 文件导入另一个项目时，模型将绑定到相应的配置。

访问

上下文菜单	在图表上点击浏览器窗口或业务流程仿真管理窗口或右键-导出工件配置
其它	配置 BPSim窗口工具栏| 导出icon

发布模型包

导出 BPSim 配置及其模型的过程使用 "发布模型包"对话框将模型发布到 XMI 文件。

选项	描述
包	默认为包含业务流程仿真工件包的名称。
文件名	类型输入或浏览（单击 图标）要导出模型的文件路径和 XML 文件名。
XML类型	选择 "BPMN 2.0 XML"。
导出	单击此按钮可导出配置和 BPMN 2.0模型。当 "进度"字段中显示确认消息时，导出完成。
格式XML输出	默认为选中；离开选中。
视图XML	如果要检查导出的 XML，请单击此按钮。

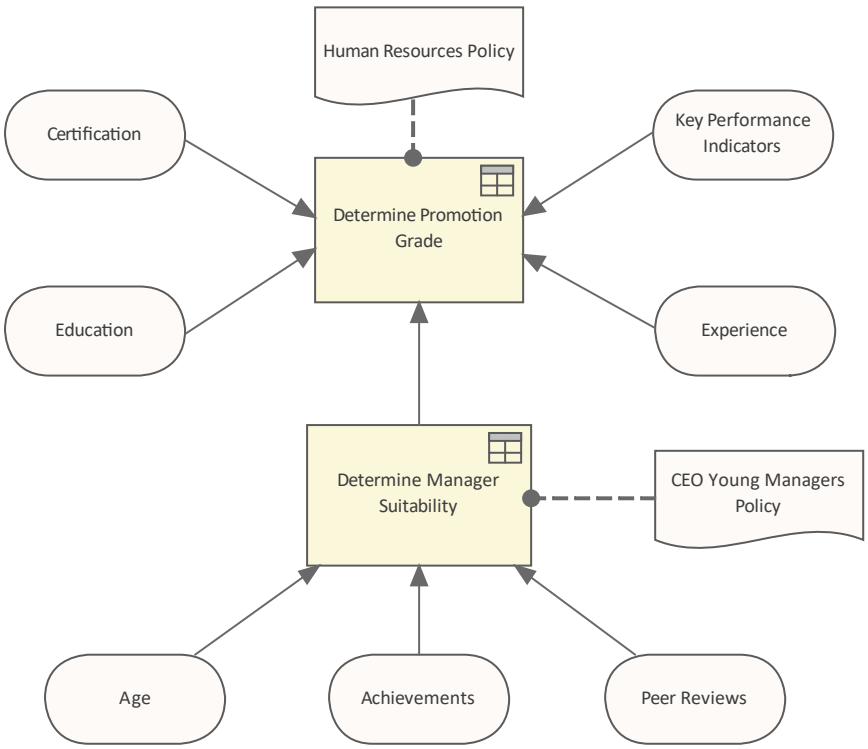
注记

- 要将模型从 XMI 导入到新项目中，请在新项目中选择目标包，然后选择 "发布>模型交换>导入>本地文件"或 "导入>XMI文件"功能区选项

Decision Model and Notation (DMN)

创建和仿真企业决策的详细模型

组织面临着日益艰难的运营环境，来自现有和新市场参与者的激烈且往往不可预测的竞争、政府和行业法规的变化以及客户群社会结构的剧变。组织在此上下文的决定对于组织的成功及其在这些未知的公司水域中引导安全路径的能力至关重要。使用企业架构师模型和表示法 (决策)特征，您不仅可以对组织做出的决策进行建模，还可以从这些模型中进行运行，以根据示例数据集预测结果。这种语言的力量在于，业务人员可以很容易地理解和使用简单但富有表现力的决策需求图，这些图详细说明了细节是什么、决策的输入是什么以及预期的输出是什么。可以通过多种方式记录规则，包括易于定义的决策库表。一旦完成这些图表以及随附的输入数据示例，就可以模拟以显示决策的结果。



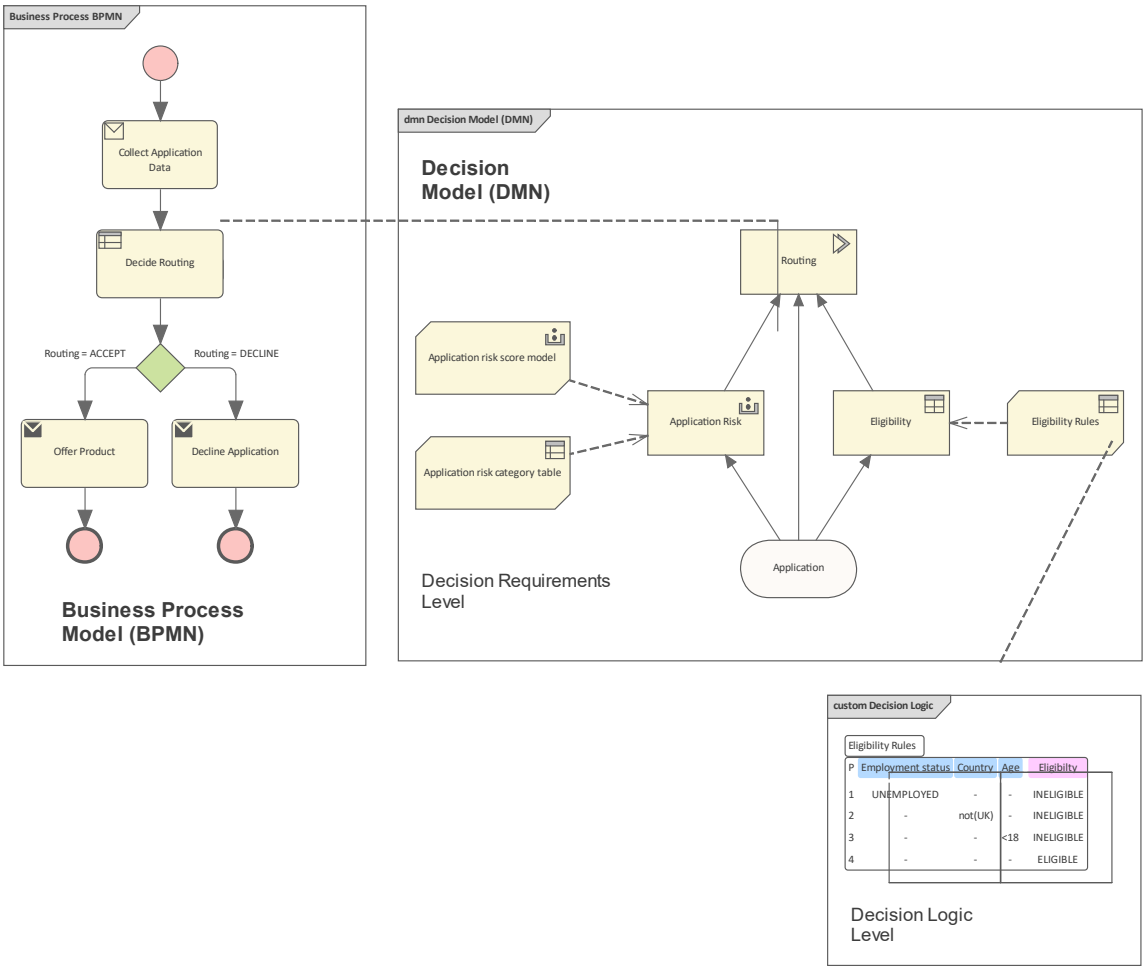
决策需求图显示了具有决策业务知识模型的决策和包括另一个决策在内的许多输入。

一旦企业定义、模拟和测试了这些模型，技术人员和工程师就可以改进这些模型，并直接从模型中自动生成包括编程代码在内的软件工件，从而减少解释错误的可能性并缩短实施时间。

什么是 DMN？

DMN 旨在提供业务流程模型和决策逻辑模型之间的桥梁：

- 业务流程模型将定义业务流程中需要进行决策的任务
- 决策需求图表将定义在这些任务中要做出的决策、它们的相互关系以及它们对决策逻辑的要求
- 决策逻辑将在足够的细节中定义所需的决策，以允许验证和/或自动化



综合起来，决策需求图和决策逻辑允许您通过在细节中指定在流程任务中执行的决策制定来补充业务决策模型的完成决策模型。

DMN 提供了跨越决策需求和决策逻辑建模的结构。

- 对于决策需求建模，它定义了决策需求图表（决策）的概念，包括一组元素及其连接规则，以及相应的符号：决策需求图表（DRD）。
- 对于决策逻辑建模，它提供了一种称为 FEEL 的语言，用于定义和组合决策表、计算、if/then/ else逻辑、简单的数据结构以及来自Java和 PMML 的外部定义逻辑，并形成具有正式定义语义的可执行表达式。

在Enterprise Architect中使用 DMN 的好处

建模使用 DMN 的决策过程允许您将复杂的决策过程记录、指定和分析为相互关联的决策、业务规则、数据集和知识源的系统。通过这样做，您可以将高度复杂的决策过程分解为支持决策和输入数据的网络。这有助于更轻松地理解整个流程，支持流程重构并简化验证流程的任务，让您可以轻松验证构成整个流程的各个步骤。

当您使用 DMN 在Enterprise Architect中构建决策模型时，您可以运行该模型的模拟来验证模型的正确性。验证您的模型后，您可以生成Java 、 JavaScript 、C++ 或 C# 的 DMN 模块。生成的 DMN 模块可以与Enterprise Architect BPSim 执行引擎、执行可执行状态机一起使用，或者在您正在实施的单独软件系统中使用。

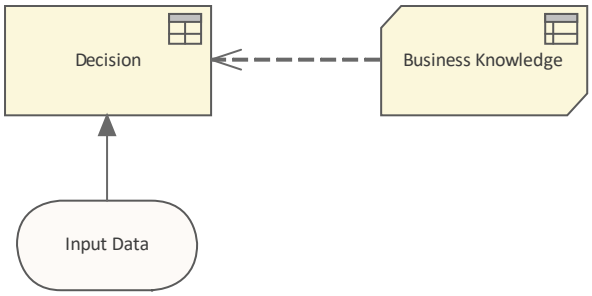
Enterprise Architect还提供了 测试模块”功能，这是一个将 DMN 与 BPMN 集成的预处理。目的是生成 BPMN2.0::DataObject 元素，然后使用这些元素验证指定的目标决策是否已通过 DMN 模块正确评估。然后，您通过加载数据对象并将 DMN 模块决策分配给 BPSim属性来配置 BPSim。

此特征在Enterprise Architect的统一版和终极版中可用，从 15.0 版开始。

决策需求图

DMN 决策需求模型由一个或多个决策需求图表(决策) 中描述的决策需求图 (DRG) 组成。建模的元素是决策、业务知识领域、业务知识来源、输入数据和决策服务。

DRG 是由由需求连接的元素组成A图，并且是自包含的，因为 DRG 中任何决策的所有建模需求（其信息、知识和权限的直接来源）都存在于同一个 DRG 中。将 DRG 的完成定义与呈现它的任何特定视图的 DRD 区分开来是很重要的，这可能是部分显示或过滤显示。



开始

决策模型和符号（DMN）是由物件管理组（OMG）发布和管理的标准。

本主题的部分内容已逐字使用或自由改编自 *DMN* 规范，该规范可在 *OMG DMN* 网页 *(https)* 上找到。 *DMN* 及其功能*A*完整描述可以在 *OMG* 网站上找到。

DMN 的目的是提供模型决策所需的结构，以便组织决策可以很容易地在图表中描述，由业务分析师准确定义，并且（可选）自动化。它还旨在促进组织之间决策模型的共享和交换。

选择蓝图

Enterprise Architect将工具的广泛特征划分为蓝图，确保您可以聚焦于特定任务并使用您需要的工具，而不会分散其他特征的注意力。要使用决策模型和符号特征，您首先需要选择此蓝图：

 <perspective name> >需求>决策建模

设置蓝图确保决策模型和符号图、它们的工具箱和蓝图的其他特征将默认可用。

示例图表

示例图提供了对该主题的可视化介绍，并允许您查看在指定或描述决策建模方式时创建的一些重要元素和连接器。决策图将引入需求决策表、知识来源、输入日期等要素。 Enterprise Architect的大部分功能依赖于模拟或 迄行"决策模型以及基 不同数据集预测结果的能力。此功能将在后面的主题中进行描述，但从创建决策需求图开始。

使用 DMN建模

本主题向您介绍创建决策模型所需的最重要元素。这包括创建决策需求图，描述决策如何相关以及每个决策可能包含哪些输入，包括其他决策。您将了解最重要的元素，包括：决策、业务模型、输入数据项定义、数据集和决策服务。

代码生成和测试模块

本主题向您介绍语言的主要概念，包括其结构、架构以及用于创建决策模型和表示法 (DMN) 模型的元素和连接器。了解语言的意图和结构将有助于分析师创建有意义且富有成效的决策模型。

集成BPSim进行仿真

Enterprise Architect允许运行（模拟）决策模型，这使您可以可视化决策的结果。除了这个核心功能之外，您还可以将决策模型与 BPSim 集成，决策是用于模拟 BPMN 图的模拟引擎。在本主题中，您将通过多种不同的方式将 DMN 与 BPSIm 集成。

集成到UML类元素

在本主题中，您将了解将 DMN模型与UML类元素集成的过程。 DMN Module 可以与UML类元素集成，因此从

类元素生成的代码可以重用 DMN Module 并且结构良好

导入 DMN XML

本主题描述如何从不同的Enterprise Architect存储库或其他 DMN 兼容工具导入 DMN XML 文件。开放标准的承诺之一是能够在不同工具之间共享模型。 Enterprise Architect经常成为建模的首选工具，因为它支持的特征和标准的广度。这允许决策模型与战略、需求业务、流程、软件实施元素等相关。

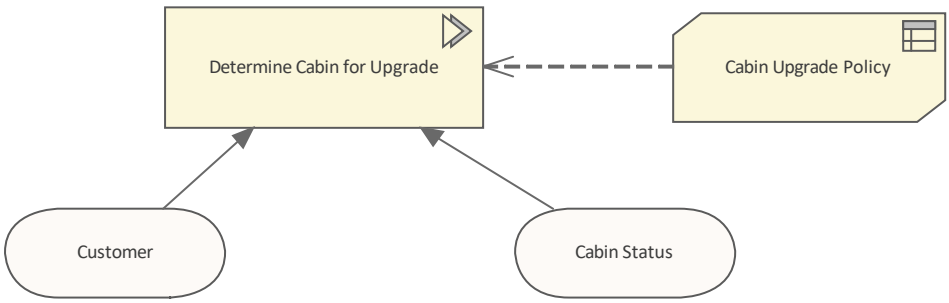
更多信息

本节提供了指向其他主题和资源的有用链接，您在使用决策模型和符号工具特征时可能会发现它们很有用。

示例图表

想象一下，您是一名航空公司预订人员，在一家繁忙的国内航空公司的值机柜台工作。让飞机准时起飞至关重要，因为延误可能会导致机场管制员收取费用，需要在较低的高度飞行，从而增加燃料成本以及其他处罚。

您的屏幕上会出现来自主管A消息，说明经济舱已超额预订；您需要将部分乘客升舱至业务或类——但应该选择哪些乘客，升舱至哪个舱位？ A做出决定，但应该考虑哪些因素？这可以使用决策需求图记录在决策模型中。



这很有帮助，但忙碌的值机员仍然需要权衡所有因素并做出公正的决定。心怀不满的乘客是否应优先于金级常旅客，或者是否应优先考虑特定乘客连接国际航班的事实。这些 规则"都可以记录在决策决策表中，明确哪些乘客应该升舱，哪些舱位：业务或类。这将使决策变得更加容易，并且可以在总部制定、商定和检查规则的一致性。在这个例子中，我们保持简单，并使用了两个因素：首先是乘客上个月的航班数量，其次是客舱超额预订的程度。

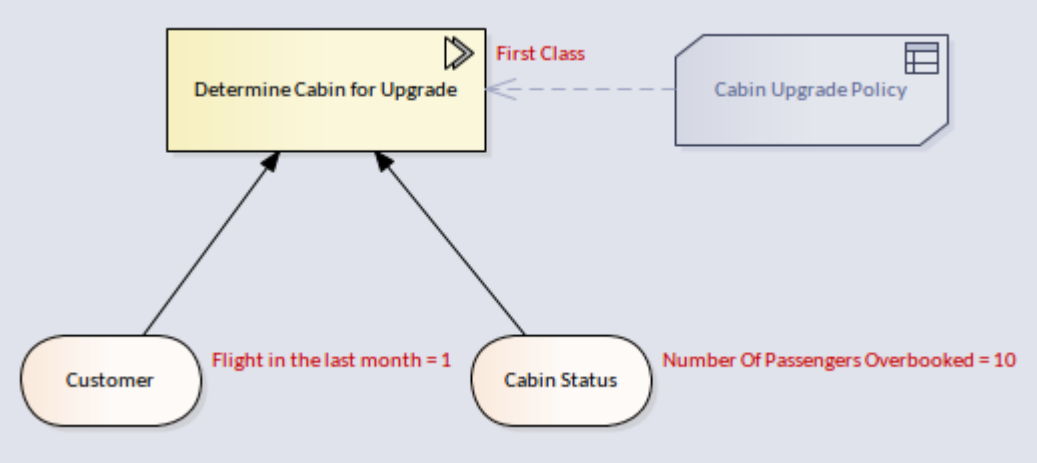
Cabin Upgrade Policy | Input Parameter Values for Simulation

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

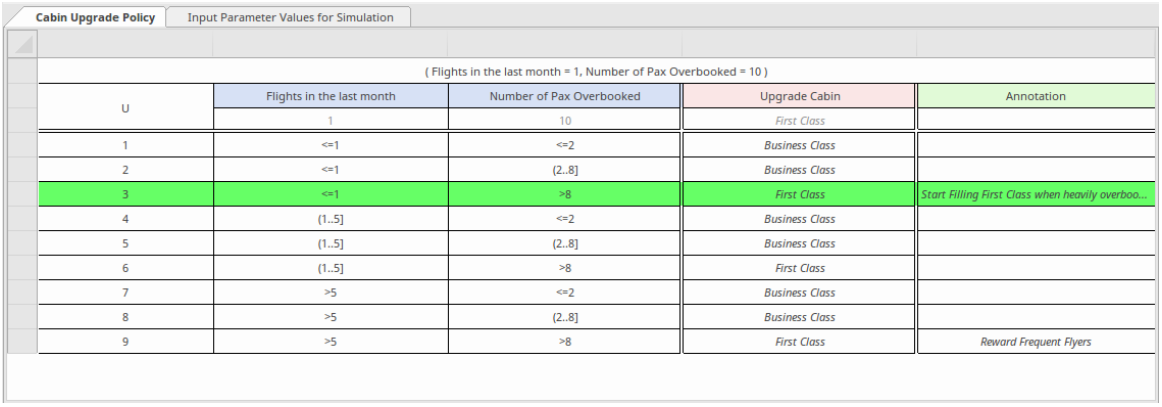
该表分为列和行。共有三种类型的列：做出决策所需的输入、应用规则的结果和注释的输出。

这再次非常有帮助，但仍然需要忙碌的源人员能够获取在决策表中找到正确行所需的所有必需信息。即使所有这些信息都可用，错误的决定仍然可能是由于人为错误选择表中的错误行而导致的。

幸运的是，决策模型可以自动生成并生成可以由应用程序执行的编程代码。所以我们忙碌的值机员不需要做任何事情或做出任何决定；当他或她正在为乘客办理登机手续时，如果特定乘客有权升级，它将在计算机屏幕上可见。在下图中，模型已被模拟，因此业务和技术人员可以同意模型已正确定义。在生成将在运行系统中运行并将结果显示给最终用户的编程代码之前，可以使用任意数量的用户定义数据集来测试模型。



在开发模型时，业务或技术用户可以逐步完成模拟，系统将向用户显示决策决策表中的哪一行被触发以确定输出。这在由多个决策组成的模型中非常有用。



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

管理升级决策的规则经常发生变化。例如，营销部门可能决定要奖励乘坐长途航班的乘客。可以更改决策需求图以包括新的输入、修改的决策表和重新生成的编程代码。一旦更改被推送到机场系统，合适的乘客将自动升级。签到人员仍然可以在培训和简报会议期间查看决策表以了解规则。

创建决策模型

在我们描述的模型示例建模决策我们展示了如何使用决策决策表对决策进行建模，其中决策结果是通过在表中找到输入值与表匹配的行来确定的正在考虑的值，给出特定的输出结果。

我们现在将通过逐步完成为 Airline Cabin Upgrade 示例创建决策模型的过程来了解如何在Enterprise Architect中创建这样的模型。

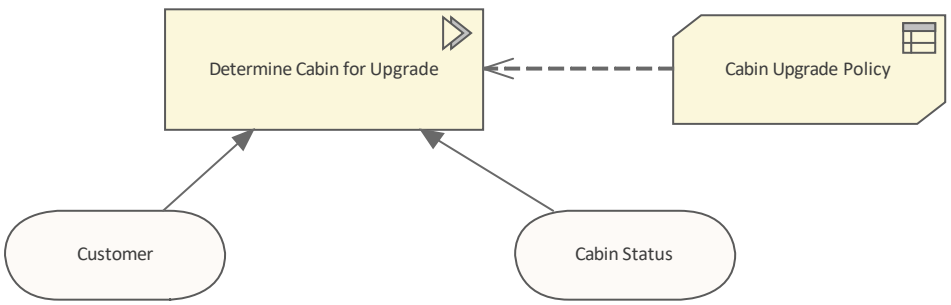
此示例中涉及许多模型元素，例如输入数据元素、用于描述输入数据（定义数据类型）的项目定义、决策元素以及保存决策决策表的业务知识模型元素定义。

创建决策需求图表

这些步骤将指导您创建一个简单的决策需求图表(DRD)。在此示例中，我们将从头开始创建模型，而不是使用模型生成器中的模式。

节	描述
1	选择视角 需求|决策建模”。 （显示模型生成器对话框，但在本例中我们不会使用它。）
2	创建一个新的 DMN 图。将其名称 机舱升级”。
3	使用图表工具箱，在图表上放置决策元素。选择 调用”作为类型——我们将使用该元素 调用”来自业务知识模型元素的决策。名称元素 确定升级舱”。
4	在图表上放置一个名称元素。将这个元素为 顾客”。
5	在图上放置另一个元素元素。名称为 客舱状态”。
6	在图上放置一个业务知识模型元素模型。选择类型 决策决策表”。名称为 元素升级政策”。
7	从决策 确定升级客舱”到输入数据 顾客”，划一个 信息需求”连接器。
8	从决策 确定升级客舱”到输入数据 划状态”的 信息需求”连接器。
9	从决定 确定升级客舱”到BKM 客舱升级政策”，划一个 知识需求”连接器。

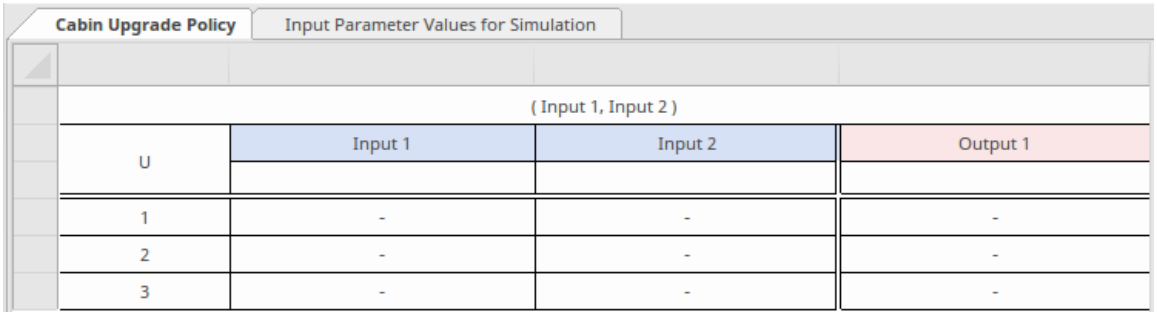
在这个阶段，我们应该有一个简单的 DRD，如下所示：



我们现在可以指定构成此模型的每个元素的详细信息。

定义决策表

双击业务知识模型元素 舱室升级政策”，弹出 DMN 表达式”窗口，显示为空的决策表。这是我们定义客舱升级政策规则的地方。



默认情况下，新的决策表创建有两个输入列和一个输出列、一个标题行和三个空规则行。

表格中最左边的一列显示了表命中策略，并为规则编号。默认情况下，命中策略是 ' U ' 代表 'Unique'。这是我们将用于示例的策略，因此您无需更改此列标题。

有关命中策略的更多信息，请参阅决策决策表命中策略帮助。

决策表输入和输出的名称和定义类型

节	描述
1	在 DMN 表达式”窗口的工具栏上，单击 编辑参数”按钮 。 将显示 编辑参数”对话框。
2	将参数名称 Input 1 ”替换为 Num of Pax Overbooked”。 如有必要，单击 类型”下拉箭头并将此参数的类型设置为 数字”。
3	将参数名称 'Input 2' 替换为 'Num of Flights in Last Month by Pass'。 将此参数的类型也设置为 数字”。 关闭 编辑参数”对话框。
4	编辑将为列1评估的输入表达式。 选择标题单元（包含文本 Input 1 ”）然后再次单击或按 F2 进入 编辑”模式。选择所有单元文本，然后按空格键。显示输入参数列表。单击 超额预订人数”，然后按“Enter”。第1列的表达式设置为 'Num of Pax Overbooked'。 **注记**：为每一列计算的输入表达式通常只使用相应的输入参数；但是，您可以使用复杂的表达式。
5	右键单击第1列表达式并检查其数据类型是否设置为 数字”。
6	编辑将为第 2 列计算的输入表达式。 选择所有文本，然后按空格键。显示输入参数列表。选择 通票上个月的航班数量”，然后按“Enter”。

	第 2 列的表达式设置为 通行证上个月的航班数量”。
7	右键单击第 2 列表达式并将其数据类型设置为 数字”。
8	编辑决策表输出的名称。 将 输出”替换为 Upgrade Cabin”，然后1 “Enter”。
9	设置决策输出的数据类型。 右键单击输出列标题并选择 string ”。
10	设置决策输出的允许值。 在输出列单元正下方（但在第1行上方）的单元中，定义输出的允许值。输入 业务类，第一类”。 **注记**：由于数据类型已指定为 string ”，因此不需要在值周围加引号。

定义决策表的规则

在单元表中输入值以匹配此图像。

Cabin Upgrade Policy | Input Parameter Values for Simulation

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

单击一个单元以选择它，然后再次单击以对其进行编辑。

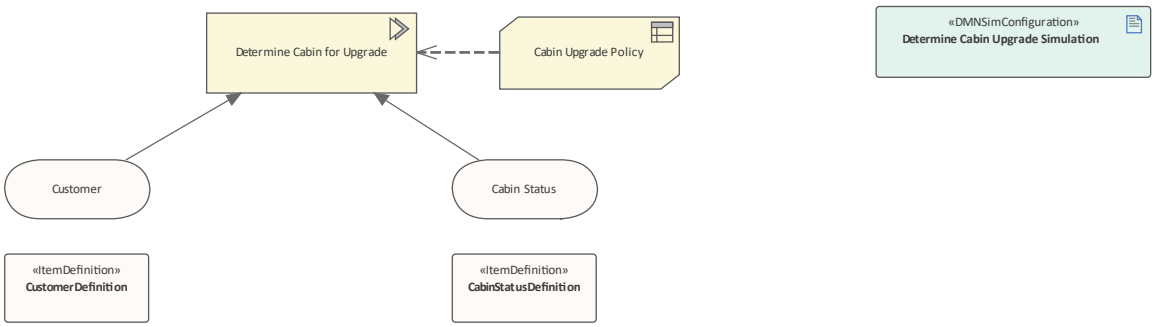
您可以通过选择要复制的行来复制和粘贴现有规则（Shift+单击添加到选择中），右键单击并选择 复制”，然后右键单击并选择 附加”。

完成规则编辑后，单击保存按钮 。

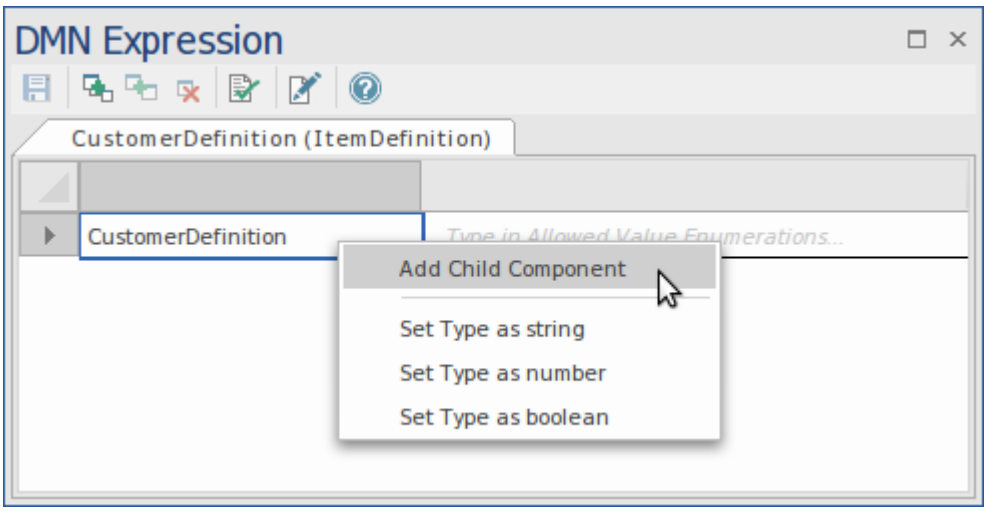
最后，单击验证按钮 ，以检查规则表中的错误。

创建 ItemDefinition元素

将两个 ItemDefinition 元素添加到图表中，每个 InputData 元素一个。名称一个元素 CustomerDefinition”，另一个元素 CabinStatusDefinition”。



双击名为 "CustomerDefinition"的 ItemDefinition 以编辑定义。显示DMN 表达式窗口。

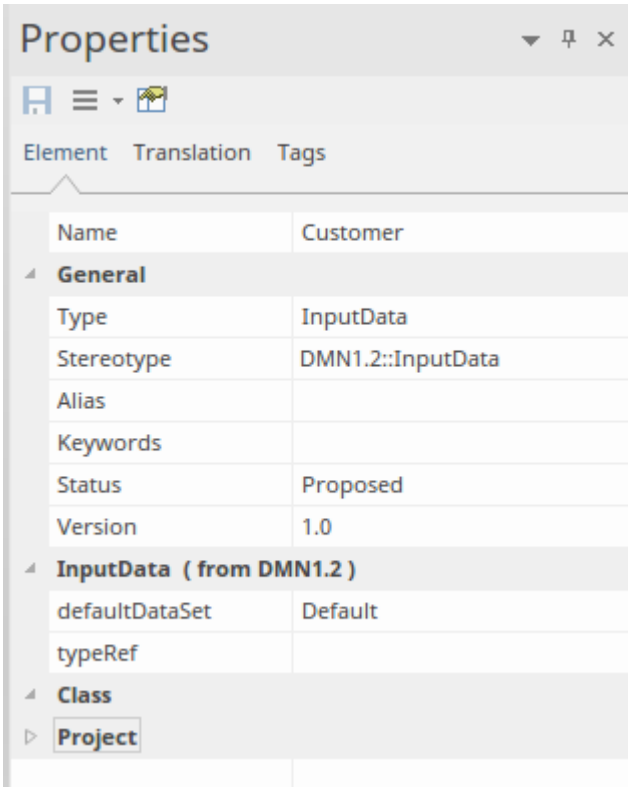


右键单击单元 "客户定义"并选择 "添加子部件"。用 "上个月的航班数量"改写子组件的名称，用 "数字"改写其数据类型。单击 "保存"按钮保存更改，然后关闭窗口。

同样，双击名为 "CabinStatusDefinition"的 ItemDefinition，添加一个名为 "Num of Pax Overbooked"的子组件，并将其数据类型设置为 "number"。保存更改并关闭窗口。

为每个 InputData元素指定数据类型

选择 InputData元素'顾客'。在属性窗口中，选择属性 "属性"并单击 ... 按钮。



选择顾客'顾客定义' 作为类型。点击 确定”。

同样，指定 客舱状态定义”作为 状态”的类型。

指定决策元素的输入

双击决策元素'Determine Cabin for Upgrade'

在DMN 表达式窗口中，找到第一列中包含文本 Num of Pax Overbooked”的表行。单击该行第二列的单元，然后按空格键。 A可能的输入值列表。选择 客舱状态”。超额预订人数”，然后按 Enter”。选择写入单元。

对第二个表行 上个月的航班数量”重复此过程，选择 顾客。上个月的航班数量”。

单击保存按钮。

单击验证按钮。

定义数据集

您的决策模型的 正确性”可以通过使用一系列代表性数据集运行模拟来测试，以验证模型在所有情况下都能产生正确的结果。

您可以使用一系列数据值创建具有不同名称的大量数据集。您可以将其中一个数据集设置为默认值。

我们现在将为每个现在元素创建一个数据集。

节	描述
1	双击 InputData元素 顾客”。 显示DMN 表达式窗口。
2	

	在DMN 表达式窗口中，单击 编辑数据集”按钮  。 将显示 编辑数据集”窗口。
3	单击  按钮。 创建A新的数据集。
4	如果您愿意，请覆盖数据集的名称。 将类型保留为 数字”。输入一个值，例如 3。 点击保存图标和确定按钮。
5	对 InputData 的 客舱状态”重复此操作。输入一个值，例如 4。

添加一个工件物业

仿真配置'模拟箱'工件in the图表的工具箱将其中之一也拖放到图表上。

双击它以在 仿真”选项卡上打开 DMN仿真窗口。

在仿真窗口中，您可以运行已完成的决策决策模型的模拟。您还可以执行验证、生成代码和生成测试模块。

节	描述
1	在此窗口的工具栏中找到编辑字段。
2	单击此字段中的下拉箭头。 A列表显示了与决策配置相关的包中的所有决策和决策工件。在这种情况下，确定升级客舱”是列表中的唯一项目。
3	单击 确定要升级的客舱”。
4	窗口的主体现在显示 InputData 元素和可用作所选决策输入的决策结果。 单击保存按钮。
5	使用 值”  运行使用选定的数据集运行模拟。

决策需求图表

决策需求图（DRG）和决策需求图（决策）决策的要素是决策、业务知识模型、输入数据、知识源和决策服务。这些元素之间的依赖关系表达了三种需求——信息、知识和权威。

决策需求图表的组成部分

本表总结了决策需求图的所有组成部分的符号。

部件	描述
决策	决策元素表示从大量输入（输入数据或决策）中确定输出A行为，使用以文字表达式、决策表、调用或盒装上下文表示的决策逻辑。
业务知识模型	业务知识模型是指A函数为代表的可复用的决策逻辑模块，包括零个、一个或多个参数。
服务决策服务（扩展）	A可以决策可重用的决策，这些决策在内部调用 - 例如，由另一个决策决策或决策业务知识模型- 或在外部 - 例如，由进程流程调用。 A好的做法是使用图表来描述单个扩展的决策服务。
服务决策服务（折叠）	如果一个决策服务元素是一个可调用的元素，通过调用逻辑将知识需求与其他元素连接起来，我们可以隐藏决策服务的细节，以聚焦在大图的决策层次上。
输入数据	输入数据元素表示用作一个或多个决策的输入的信息。
项目定义	项定义用于定义决策模型中使用的数据项的类型和结构。它主要由 Input Data 元素引用，作为预期输入的数据类型和结构的基础。也可以参考它来设置输出的结构。 项目定义包含数据集，这些数据集提供了在执行各种模拟时有用的值集。
知识源	知识源元素表示业务知识模型或决策A决策。
信息需求	信息需求表示输入数据或决策输出被用作决策的决策。
知识需求	知识需求表示对业务知识模型或决策服务A调用。
权威需求	权威需求表示一个 DRG元素对另一个 DRG元素的依赖性，该 DRG 元素源指导或知识的来源。

决策表达式编辑器

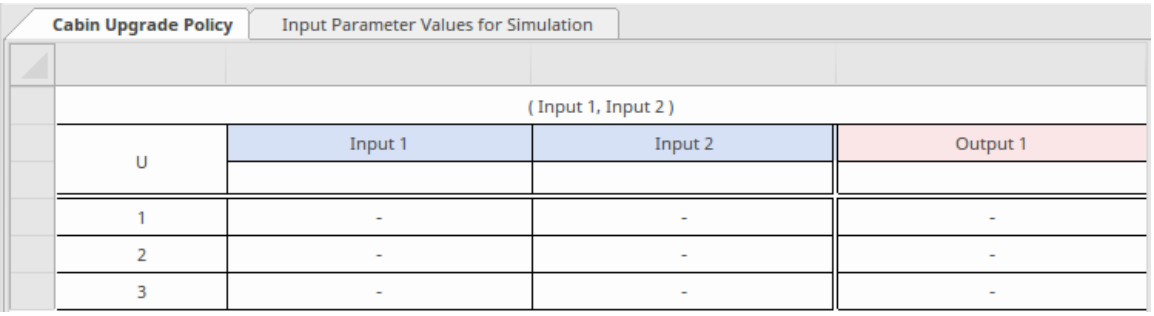
DMN 表达式编辑器是您将在其中定义、审阅和更新模型中大多数不同类型 DMN元素的详细信息的窗口。首先，它用于编辑决策元素和业务知识模型（BKM）元素的价值表达。

A决策元素和决策元素使用的四种类型的值表达中的每一种，都会显示不同版本的DMN 表达式编辑器。对于 BKM 元素，还提供了第二个窗口选项卡，用于定义调用 BKM 时使用的输入和输出参数。

还存在两个附加版本的DMN 表达式编辑器以支持编辑 ItemDefinition 和 InputData 元素。

显示的工具栏和窗口内容的布局取决于当前选择的 DMN元素的类型，以及在适用的情况下定义的值表达式的类型。

此图显示了用于定义决策表的DMN 表达式编辑器的版本。在这种情况下，底层元素是一个 BusinessKnowledgeModel，因此决策逻辑由其他元素 "调用"，输入和输出通过参数传递。



DMN 表达式编辑器对每个元素和表达式类型的特征的详细解释在本主题的子帮助主题中提供。

访问

图表	双击图表上的 DMN元素。 显示元素及其表达式类型对应的DMN 表达式编辑器窗口。

值表达式

本表总结了四种不同类型的价值表达，并参考了详细说明每种类型的帮助主题。

类型和图标	描述
决策表	决策表是A组相关输入和输出表达式的表格表示，组织成规则，指示哪个输出条目适用于一组特定的输入条目。
文字表达	文字表达式A决策逻辑指定为描述输出值如何从其输入值派生的文本表达式。为了支持模拟和执行，文字表达式可以使用JavaScript函数。
盒装上下文	盒装上下文是上下文条目A集合，由（名称，值）对组成，每对都有一个结果值。 上下文条目提供了一种将复杂的表达式分解为一系列简单表达式的方法，提供可以在后续上下文条目中使用的中间结果。
调用	调用调用另一个模型元素（业务知识模型或决策服务）以提供决策结果。调用定义了传递给 "被调用"元素的参数，为上下文其决策逻辑提供时间。然后

	将决策结果传递回 调用"元素。

ItemDefinition 和 InputData元素

元素	描述
项目定义	ItemDefinition 元素用于定义数据结构，并可选择限制数据允许值的范围。ItemDefinitions 的范围可以从简单的单一类型到复杂的结构化类型。ItemDefinitions 用于指定 InputData 元素的类型以及输入参数。
输入数据	决策元素用于为决策元素提供输入。 InputData元素的数据类型是使用 ItemDefinition元素定义的。数据集也可以定义为 ItemDefinition 的一部分，然后 InputData元素可以指定运行模拟时要使用的数据集。

决策表

决策表是A组相关输入和输出表达式的表格表示，组织成规则，指示哪个输出条目适用于一组特定的输入条目。

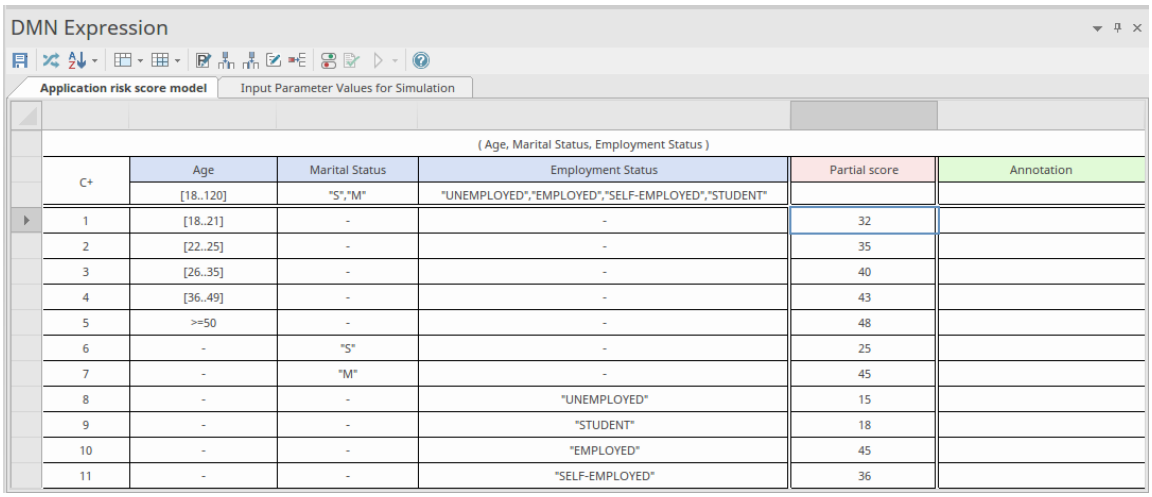
决策表由决策和决策业务知识模型元素类型提供支持。它们由图上元素右上角的 图标表示。

访问

图表	在图表上，双击决策元素或 BusinessKnowledgeModel元素。 显示DMN 表达式窗口，显示所选元素的详细信息。

概述

此图显示了决策表的DMN 表达式窗口。



(Age, Marital Status, Employment Status)

C+	Age	Marital Status	Employment Status	Partial score	Annotation
	[18..120]	"S","M"	"UNEMPLOYED","EMPLOYED","SELF-EMPLOYED","STUDENT"		
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	

决策表包括A

- 指定如何应用规则的库表命中策略（C+、 U 、 A 、P 等）
- 规则A （ 1等），其中每个规则行包含特定的输入条目和相应的输出条目
- 输入子句A （在蓝色标题下），定义为通常涉及一个或多个输入值的表达式
- 输出子句A （在粉红色标题下），定义对应于一组特定输入的输出
- 您可以通过窗口工具栏添加的每个规则的可选注释（在绿色标题下）

输入子句由一个表达式和一个可选的允许值列表（列标题下方的行）组成。很多时候，表达式只是一个未经修改的输入值；但是，它也可能是一个涉及多个输入值的表达式，或者可能是一个条件语句，例如 Application风险Score > 100”。允许的值适用 表达式结果，而不是使用的输入值。

每个输出子句由一个标识符（名称）和该子句允许值的可选列表组成。

决策表应包含确定输出所需的所有输入——并且仅包含那些输入。

在确定应用哪些规则时，输入子句中定义的表达式会针对给定的输入进行评估，然后使用表达式结果来查找具有匹配输入条目的规则。

如果DMN 表达式窗口不够宽或不够深，无法显示所有列和行，您可以使用滚动条来访问隐藏的内容，或者将边框拖出以增加每列的宽度。 输入”和 输出”列宽最初是相同的，但您可以通过在表内或在选项卡名称正下方的

灰色栏中拖动列边框来独立调整每个列宽。

决策表编辑器工具栏

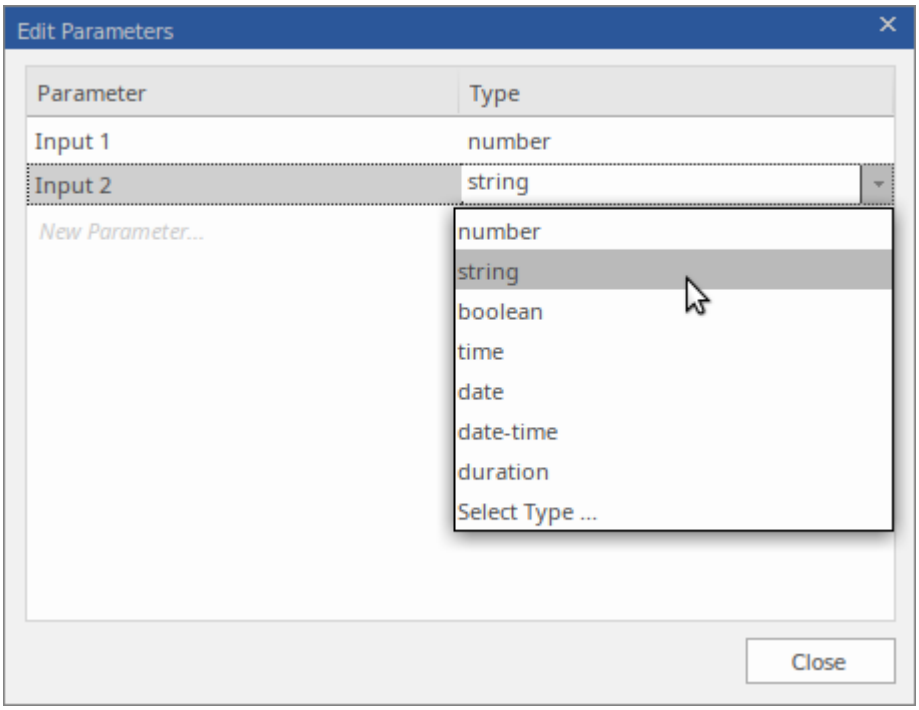
选择决策表后，可以通过窗口顶部的工具栏访问DMN 表达式窗口中可用的特征，如下所示：

有关更多详细信息，请参阅决策表编辑器帮助主题的工具栏帮助

参数

在业务知识模型（BKM）元素的情况下，参数用于传递调用元素提供的输入值。使用输入参数评估 BKM 的决策逻辑，并将结果返回给调用元素。默认情况下，使用两个输入参数 "Input 1"和 "Input 2"创建 BKM元素。

单击DMN 表达式窗口工具栏中的  图标以显示 "编辑参数"对话框。



您可以在此处更改参数名称、重新排列序列、设置其数据类型、创建附加参数或删除现有参数。

命中策略

右键单击 "命中策略指示器"，然后从弹出菜单中选择所需的命中策略目标。各种库表命中策略在细节决策表命中策略帮助中详细描述。

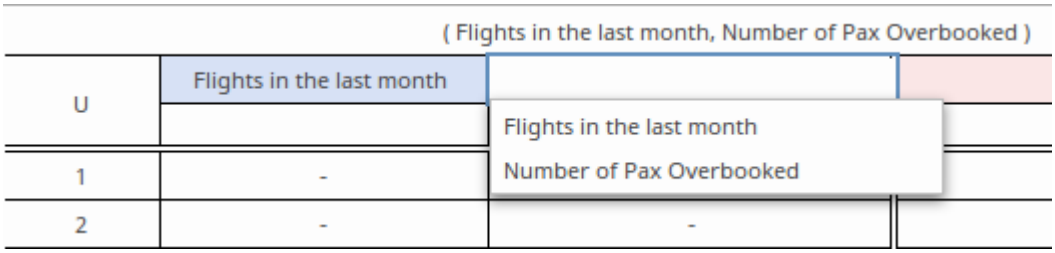
输入子句

决策表的输入子句被定义为一个表达式。很多时候，表达式只是一个未经修改的输入值；但是，它也可以是一

个涉及多个输入值的表达式，或者可以定义为条件语句，例如 风险> 100”。允许的值适用 表达式结果而不是使用的输入值，因此，值的类型应与表达式结果的类型相匹配。

决策表是用两个默认的输入子句创建的， 输入1 ”和 输入 2”。这两个子句的数据类型都是 数字”。在DMN 表达式窗口中，输入子句在决策表上显示为列标题。要修改输入子句，请单击列标题以选择单元，然后再次单击或按 F2 进行编辑。

编辑输入子句时支持自动补全。这意味着，对于决策元素，任何连接到决策元素的输入都可以从列表中进行选择。类似地，对于业务知识模型元素，调用参数可用于从列表中选择。有关详细信息，请参阅帮助*DMN* 表达式自动完成帮助主题。



要向表添加其他输入条目列，请单击DMN 表达式窗口工具栏上的决策表图标。

要从表中删除输入列，请在不需要的输入列中单击鼠标右键，然后从弹出菜单中选择 删除输入列”选项。

可以通过将列拖放到新位置来重新排列表中输入列的顺序。 （将表列最顶部未标记的单元所需位置。）

允许值

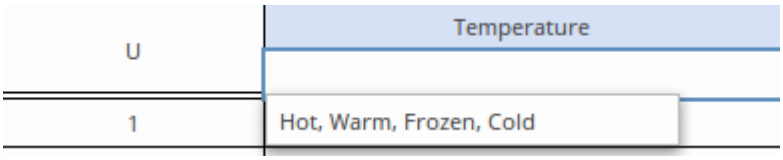
在定义 输入”或 输出”列时，该列的第二行定义了允许值。这是列中的一个可选单元，但对于澄清其下方行中的条目很有用。运行验证时，将检查允许值单元的每个单元，以确保它们符合此单元中的表达式。

此单元中使用的表达式取决于 输入”或 输出”列的键入方式。例如：

- 数字 - [18 ..35]
- 字符串- 高”、 低”、 中”
- 布尔值 - 真、 false
- 不明确的 - '-'

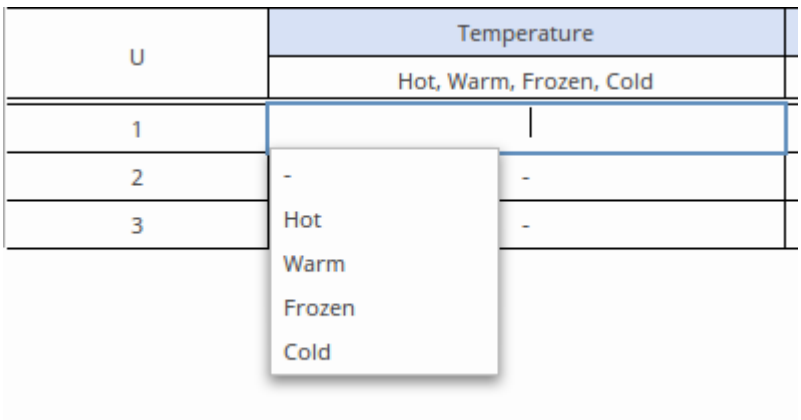
快速填充允许值

这里引用的输入/输出表达式可以是一个简单的值，也可以是一个复杂的FEEL表达式；但是，如果它与 ItemDefinition 的 允许值”字段直接相关，则按下S速度条将启用快速填充选项来设置ItemDefinition中定义的 允许值”（通常通过 InputData元素引用）。



快速填充行

一旦定义了 允许值”字段，以及限制在表中定义规则时可以使用的值， 允许值”字段还为用户提供了快速填充选项。这在规则单元中通过按下S速度条并选择所需项目来调用：



更多详情请参见帮助主题帮助*DMN* 表达式自动完成。

输出子句

输出子句由名称、数据类型和可选的允许值列表组成。要修改输出子句，请单击列单元以选择单元，然后再次单击或按 F2 进行编辑。

要向决策表。

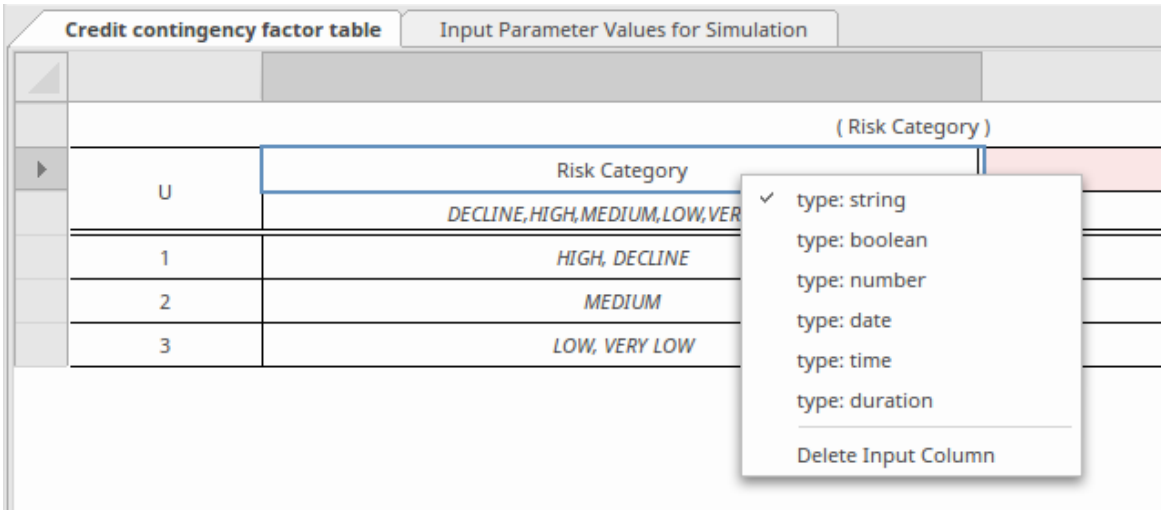
要从表中删除输出列，请在不需要的输出列中单击鼠标右键，然后从弹出菜单中选择选项 删除输出列”。

可以通过将列表到新位置来重新排列表中列的顺序。 （将表列最顶部的未标记单元所需位置。）

输入/输出子句的数据类型

为了使模拟工作，为所有输入和输出子句设置数据类型至关重要。 'number' 类型的子句支持范围、间隙和重叠验证，但如果未指定类型，则无法执行验证。 C++、C# 和Java等类型语言的代码生成要求指定数据类型。当数据类型指定为' string '时，不需要将每个string文字括在引号中。如果类型已声明，字符串值以斜体显示。

要设置数据类型，请右键单击输入或输出列标题，然后从列表中选择所需的类型。



定义决策表规则

决策表规则是通过在一个表行的单元内指定输入条目和相应的输出条目来定义的。对于 数字"数据类型，可以将输入条目指定为单个值或数字范围，例如 "10"、 "100"或 "[2..8)"。 （定义数字范围时，使用round表示不包括边界数字；使用方括号表示包括边界数字。）输出条目应指定每个单元的单个值。

通过单击工具栏中的图标，可以将其他规则附加到规则列表中。通过右键单击规则并从弹出菜单中选择 删除规则行"选项，可以从表中删除不需要的规则。

可以通过首先选择规则来复制和粘贴现有表（使用 "Ctrl+单击"从选择中添加/删除），然后使用菜单选项 复制规则到剪贴板"和 粘贴规则从剪贴板"到执行复制和粘贴。然后可以通过选择和编辑单个单元来修改复制的规则。

如果为string或布尔表达式设置了 允许值"字段，则空格键可用于显示可供选择的值列表，如前面的 允许值 - 快速填充行"部分所示。

规则也可以在表中排序，方法是：

- 单击工具栏上的图标，然后选择 按输入排序"或 按输出排序"，或
- 右键单击表中的单个规则，然后从弹出菜单中选择 上移规则"或 下移规则"选项

为了确定选择哪些表行进行输出，由输入子句定义的表达式针对给定的输入进行评估，然后将表达式的结果与表行的输入条目进行比较。如果表达式结果与表行的输入条目匹配，则选择该行进行输出。

决策决策表的 命中策略"决定了匹配行如何使用决策表来产生其输出。

规则格式

您可以选择 - 使用工具栏图标 - 以三种格式之一显示决策表，如下所示。

Rule-as-Row 格式，其中规则沿行开发，输入、输出和注释设置在列中：

(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*

Rule-as-Column 格式，其中规则在列下展开，输入、输出和注释沿行设置：

(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

Rule-as-Crosstab 格式，其中规则由定义为一组行和列组合的输入形成，输出设置在相交单元中。（注记这种格式隐藏了 注释"字段）：

(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*				

在模拟结束时，在交叉决策表中，相关的输入条目和输出条目被突出显示。例如，在此模拟处理中，订单大小小于或等于 10顾客适用交货的客户业务输出了 0.10 的折扣。

交叉表设置

在 Rule-as-Crosstab 格式中，由于输入形成行和列，输出位于交叉点，因此设置值的步骤与其他两种格式略有不同。

1. 要添加其他类型的输入，请右键单击输入列标题并选择 添加输入”选项。系统会提示您输入输入名称；输入被添加为当前列字段下的一组字段。
   要从列中删除输入类型，请右键单击其字段集并选择 删除输入”选项。输入的名称及其字段集将从列标题中删除。
2. 要添加其他类型的输出，请右键单击窗口左上角的输出块，然后选择 添加输出”选项。系统会提示您输入输出名称；名称将添加到输出块，并在窗口主体中的每个单元中添加一个新行。
   要删除输出类型，请右键单击窗口左上角输出块中的类型名称，然后选择 删除输出”选项。网格中的输出名称及其字段将被删除。
3. 您可以在输入类型之间旋转以选择一种作为行标题。右键单击该行，然后单击 选择输入作为行标题”选项。这将显示输入类型的列表；单击要用作行标题的类型；其他类型组合在列中。
4. 要将值条目行或列添加到输入，请右键单击当前行或列，然后根据需要选择 添加输入条目行”或 添加输入条目列”选项。显示输入条目名称A提示；当您输入此信息时，相应的行或列将添加到决策表中。
   要删除值条目行或列，请右键单击它并选择 删除输入条目行”或 删除输入条目列”选项。从表中删除选定的行或列。
5. 在输入列中，每一行都匹配一种输入。如果要将一种输入类型的行移动到另一种类型的上方或下方，请右键单击它并选择 上移输入”或 下移输入”选项。上下文菜单仅提供可操作的选项，因此无法将最后一行向下移动，因此未列出 向下移动输入”选项。

决策表编辑器工具栏

This表provides descriptions of the特征accessible in the DMN 表达式window when a决策表is selected.

工具栏选项

图标	描述
	保存对当前选择的决策或 BusinessKnowledgeModel元素的更改。
	在 Rule-as-Row、Rule-as-Column 和 Rule-as-Crosstab 之间切换决策表的视图。或者，单击下拉箭头并选择您需要的格式。
	单击 按输入排序”以按输入列对规则进行排序；单击 按输出排序”以按输出列对规则进行排序。可以拖放列来组织排序顺序。
	相邻规则的合并单元，其中输入条目的内容相同。您可以编辑合并后的内容。单元。在模拟过程中，合并的项目会突出显示。
	单元先前已合并的输入条目。
	显示 编辑参数”窗口，您可以在其中指定调用 BusinessKnowledgeModel元素的决策逻辑时传递的参数的名称和数据类型。
	将输入列附加到决策表。
	将输出列附加到决策表。
	在表格后面添加一个注释栏（带有绿色标题单元），可以在其中记录表的注记或对规则的评论。 （参见前面的 Rule-as-Row/Rule-as-Column 行中的插图。）如果需要，您可以添加多个注释列，在每个标题单元中键入适当的列标题。 要删除注释列，请右键单击它并选择 删除注释列”选项。
	将规则附加到决策表。
	显示或隐藏 输入”和 输出”列的允许值字段。 为输入或输出定义的允许值将用于验证和自动完成编辑。
	执行决策表的验证。 Enterprise Architect将执行一系列验证，以帮助您发现决策表中的任何错误。
	在为 BusinessKnowledgeModel元素定义决策表时启用此按钮。 选择 参数for仿真”选项卡，完成字段，然后单击此按钮。测试结果将显示在决策表上，显示输入和输出的运行时值，并突出显示有效规则。 您可以使用此功能对 BusinessKnowledgeModel元素进行单元测试，而无需指定其上下文。 此工具栏按钮A许多菜单选项可用。有关详细信息，请参阅帮助主题仿真帮助

	仿真模型

目标决策表命中策略

命中策略规定了规则重叠情况下的决策决策表结果。特定决策决策表单元中的单个字符表示表的类型，并明确地反映决策逻辑。

命中策略：

- **U**性：不可能有重叠，所有规则都是不相交的；只能匹配一条规则（这是默认设置）
- **A** ny：可能存在重叠，但所有匹配规则显示每个输出的输出条目相同，因此可以使用任何匹配
- **优先**级：可以匹配多个规则，输出条目不同；该策略返回输出优先级最高的匹配规则
- **第一**：可以匹配多个（重叠）规则，具有不同的输出条目；返回规则顺序的第一个命中

多重命中策略：

- **O**顺序：按输出优先级降序返回所有命中
- **R**顺序：按规则顺序返回所有命中
- **收集**：以任意顺序返回所有命中；可以添加一个运算符（'+'、'<'、'>'、'#'）来对输出应用一个简单的函数

收集运算符是：

- + (sum)：决策表的结果是所有不同输出的总和
- < (min)：决策表的结果是所有输出中的最小值
- > (max)：决策表的结果是所有输出中的最大值
- # (count)：决策表的结果是不同输出的数量

示例命中策略的示例

独特"的命中策略是决策决策表中最流行的类型，所有规则都是不相交的。

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*

示例命中策略的示例

在带有 优先级"命中策略的表中，可以匹配多个规则，并具有不同的输出条目。该策略返回输出优先级最高的

匹配规则。

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

注记：允许值列表用于定义输出优先级。在这里，允许的值被列为 INELIGIBLE 和 ELIGIBLE； INELIGIBLE 被定义为具有比 ELIGIBLE 更高的优先级。

一种可能的模拟结果可能类似于：

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

匹配规则被突出显示，但选择规则 2 的输出，因为 INELIGIBLE 的优先级高于 ELIGIBLE。

示例-Sum命中策略的示例

对于具有 收集和”（C+）命中策略的决策决策表，决策决策表的结果是所有不同输出的总和。

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

在此示例中，输出 Partial Score 计算为 43 + 45 + 45 = 133

决策表验证

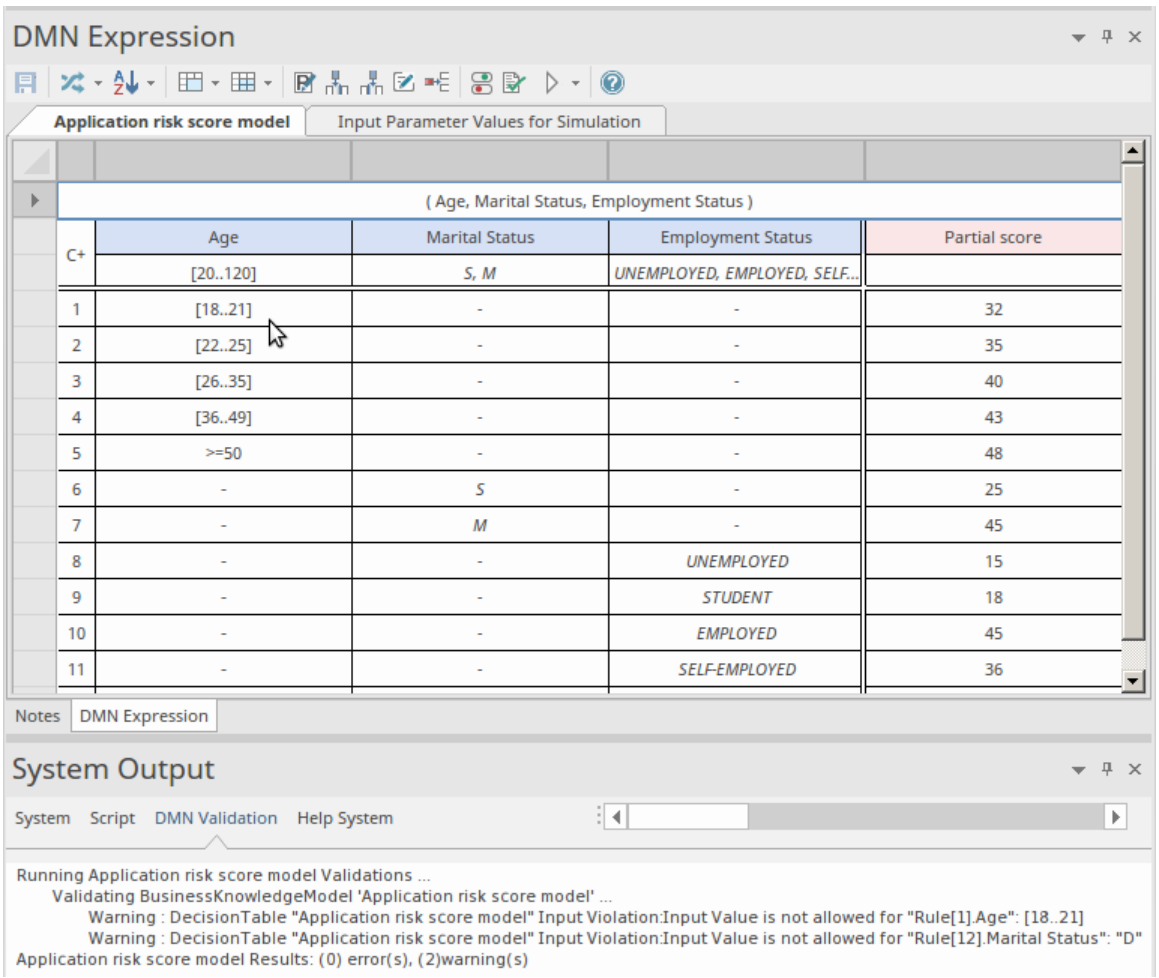
决策决策表是用于表达决策逻辑的最常见和最有用A DMN 表达式之一。然而，对决策表建模也可能很复杂，特别是如果多个输入子句子被组合用于许多决策表规则。如本主题所述， Enterprise Architect提供了验证决策表的功能。

访问

DMN 表达式窗口	仿真>决策分析> DMN > DMN 表达式：验证按钮
DMN仿真窗口	仿真>决策分析>DMN>打开DMN仿真>配置：验证按钮

条目超出范围检测

为决策表的输入子句和输出子句定义 允许值”是一种很好的做法。 允许值”列表用 对表规则的输入和输出条目值进行范围检查。



在这个例子中：

- 'Age' 输入子句定义了 [20..120] 的范围；但是，规则1的输入条目指定了 [18..21] 的范围；这超出了允许值的

范围，因此规则1被报告为无效

- 'Marital状态' 子句将其允许值定义为' S , M '的枚举；规则 12 指定值 “D”，因此该规则也被报告为无效

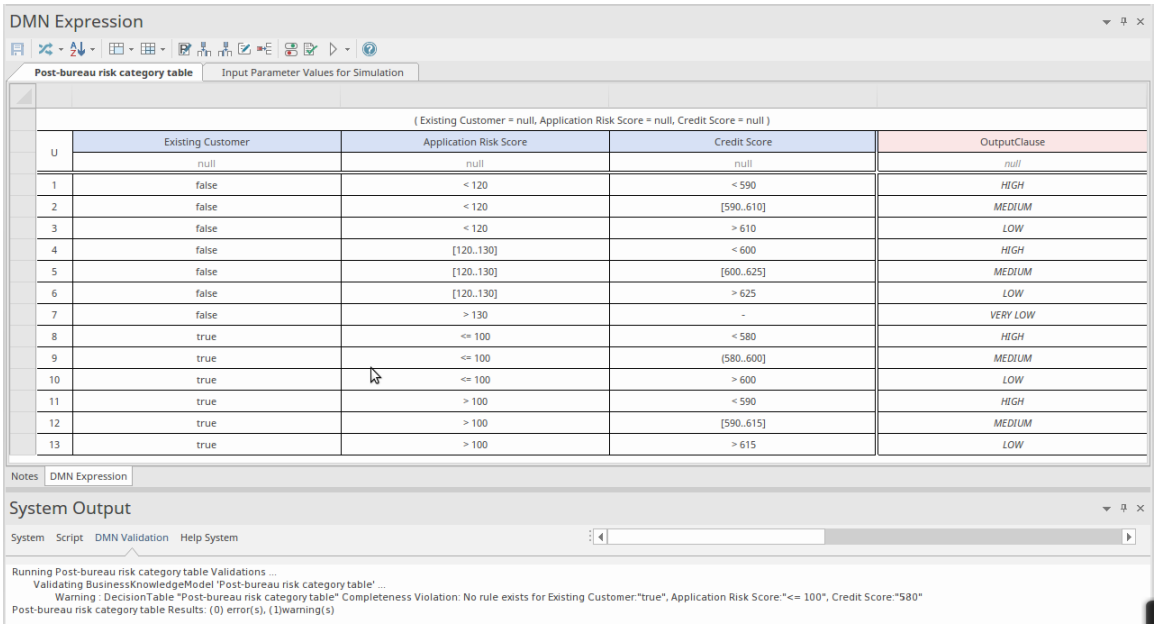
这些问题可以通过更新 “允许值”或修改无效规则的输入条目来纠正，具体取决于实际的业务规则。

完整性检测 - 报告规则中的空白

决策表规则的差距意味着，给定输入值的组合，没有规则是匹配的。这表明可能缺少某些逻辑或规则（除非定义了默认输出）。

当决策表包含许多指定数量范围的规则时，很难直观地发现差距，并且编写和运行详尽的测试用例非常耗时。

例如：



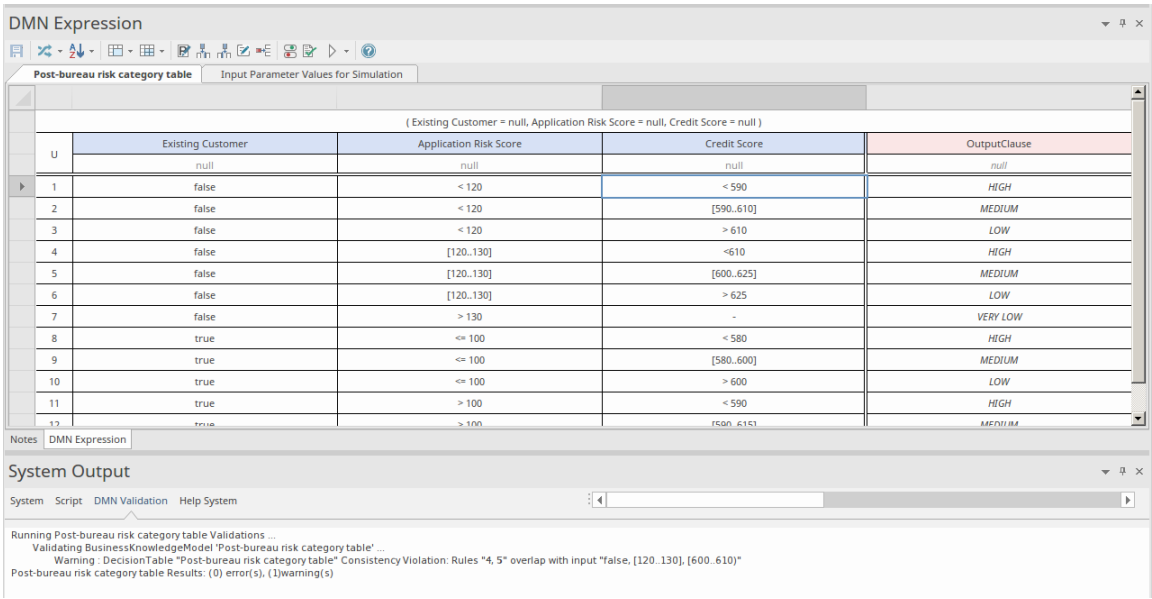
验证报告规则中的差距。仔细检查发现规则 9 中有错误。输入条目(580..600] 应该是[580..600]。

Unique命中策略的规则重叠检测

当规则重叠时，对于给定的输入值组合，匹配多个规则。如果决策决策表将其命中策略指定为 “唯一”，则属于违规行为。

当决策表包含许多指定数量范围的规则时，很难直观地发现差距，并且编写和运行详尽的测试用例非常耗时。

例如：



验证报告规则中有重叠，涉及规则 4 和 5。仔细检查发现重叠存在于第三个输入 信用评分”中，其中 “610” “[600..625]”重叠。您可以通过将规则 4 更改为 “600”或将规则 5 更改为 [“610..625]”来纠正此问题，以反映实际的业务规则。

文字表达

Literal Expression 是 DMN 表达式A最简单形式；它通常定义为单行语句或 if-else 条件块。 Literal Expression 是一种在决策元素和业务知识模型(决策) 元素中使用的价值表达。随着表达式变得越来越复杂，您可能更喜欢 Boxed Context，或者为了提高可读性，您可以将一些逻辑封装为 DMN 库中的函数。

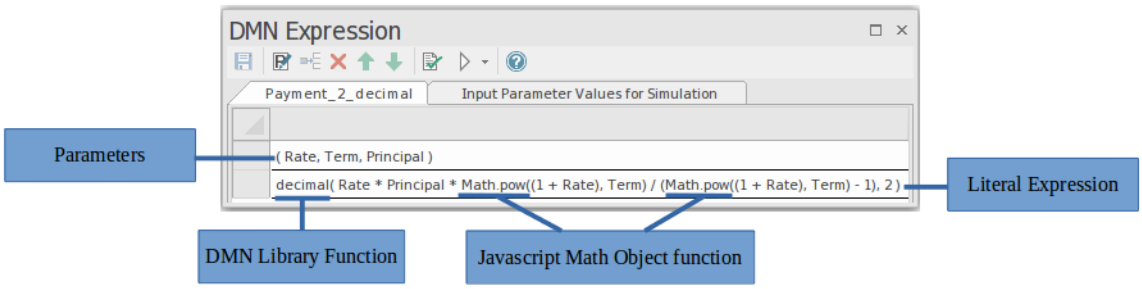
决策或 BKM元素右上角的 =X 图标表示它是作为*Literal Expression*实现的。

访问

图表	在图表上，双击决策元素或 BusinessKnowledgeModel元素。 DMN 表达式编辑器窗口显示所选元素的详细信息。

概述

此图像显示DMN 表达式编辑器窗口，它显示为文字表达式。



文字表达式是决策逻辑的文本表示。它描述了如何使用数学和逻辑运算从输入值导出输出值。

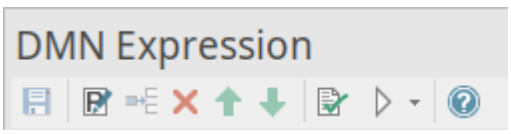
表达式编辑器窗口将 Literal Expression 显示为一个表，其中包含两个关键行：

- 参数：定义表达式中使用的输入参数
- 文字表达式：定义表达式的公式的地方 - 这定义了决策的输出

为了支持模拟和执行，文字表达式可以使用JavaScript全局函数或JavaScript object函数。用户还可以创建 DMN 库函数以在表达式中使用。

文字表达式编辑器的工具栏

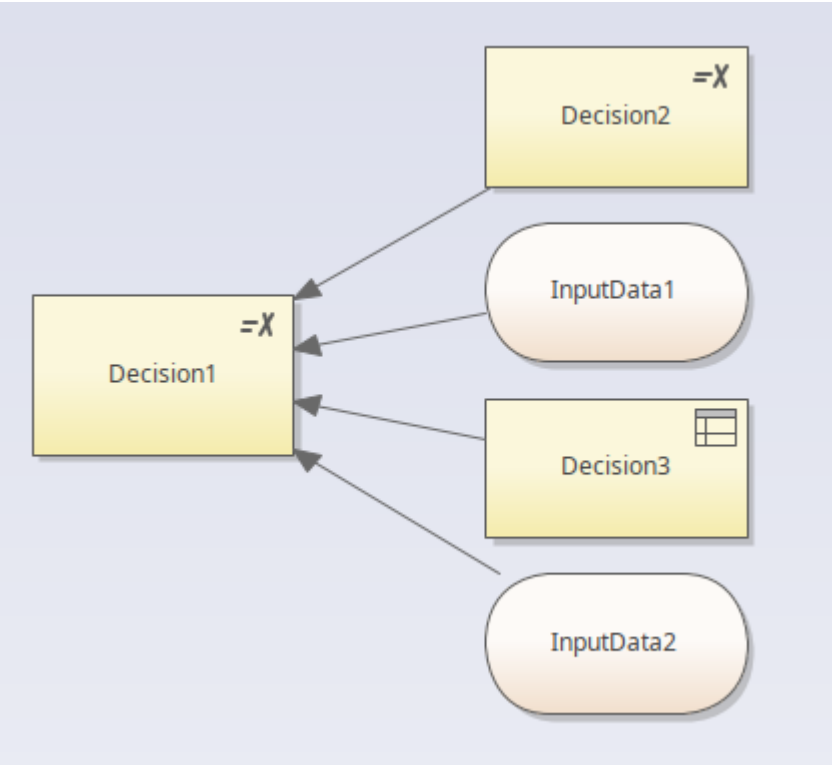
选择文字表达式后， DMN 表达式窗口中可访问的布局特征为：



更多详情请参考帮助主题 Toolbar for帮助*Expression Editor* 。

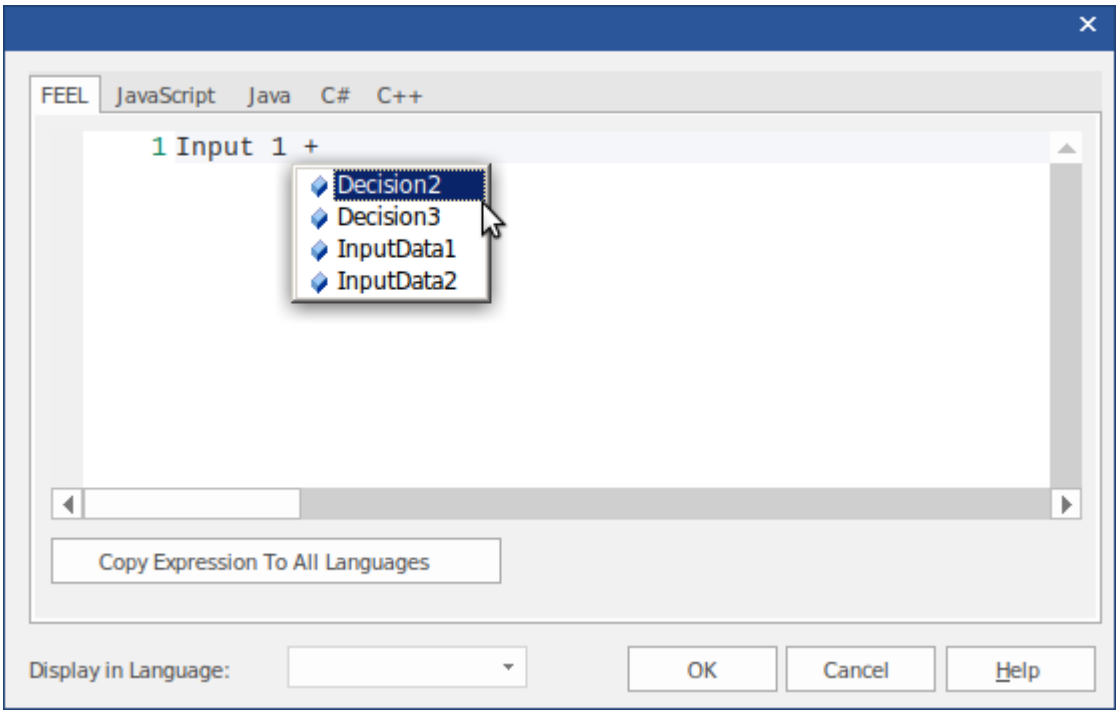
表情编辑器和智能感知支持

根据《足够友好表达语言》（FEEL）语言规范，参数名称可以包含空格，这样表达起来更容易阅读。Enterprise Architect还提供智能感知。支持编辑表达式，减少输入和错误



给定一个如图所示的决策层次结构，当编辑 "Decision1"的表达 时，"Decision1"的输入——即 "Decision2"、"Decision3"、 "InputData1"和 "InputData2"——将通过智能感知获得编辑。

通过右键单击DMN 表达式窗口的 "表达式"行，然后选择菜单选项 "编辑表达式..."，将显示表达式代码编辑器对话框。按 Ctrl+Space 显示智能感知菜单：



- 对于 "决策"元素，将显示决策的所有输入
- 对于业务知识模型（BKM）元素，将显示所有输入参数

DMN模型可以生成为JavaScript 、 Java 、C#或C++的源代码；由于某些语言可能对某些表达式有不同的语法，Enterprise Architect为每种语言提供了语言覆盖页面。如果没有为语言指定覆盖代码，则将使用为 FEEL 语言定义的表达式。

在生成的代码中，变量名中的空格将被替换为下划线。

文字表达式编辑器的工具栏

选择文字表达式后， DMN 表达式窗口会显示特定于该表达式类型的工具栏。

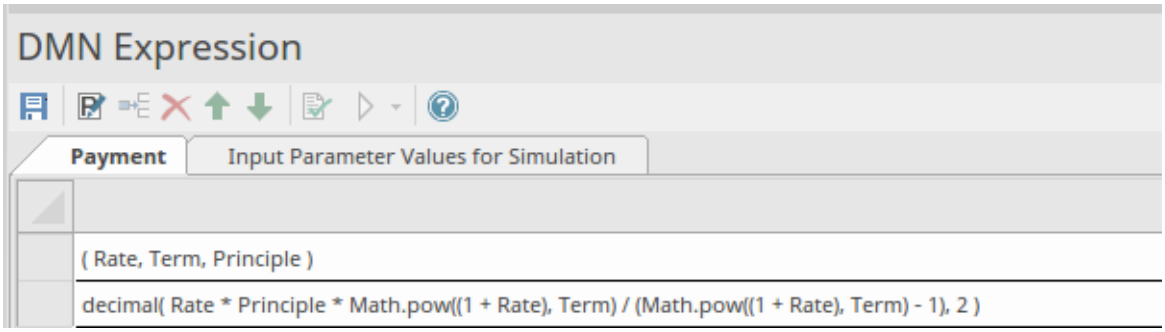
工具栏选项

此表提供了在选择文字表达式时可从DMN 表达式窗口中的工具栏访问的特征描述。

选项	描述
	单击此按钮可将配置保存到当前决策或 BusinessKnowledgeModel。
	单击此按钮可编辑业务知识模型的参数。
	对文字表达式禁用此选项。
	对文字表达式禁用此选项。
	对文字表达式禁用此选项。
	对文字表达式禁用此选项。
	单击此按钮以执行文字表达式的验证。 Enterprise Architect将执行一系列验证以帮助您定位表达式中的任何错误。
	当为 BusinessKnowledgeModel元素定义文字表达式时，将启用此按钮。

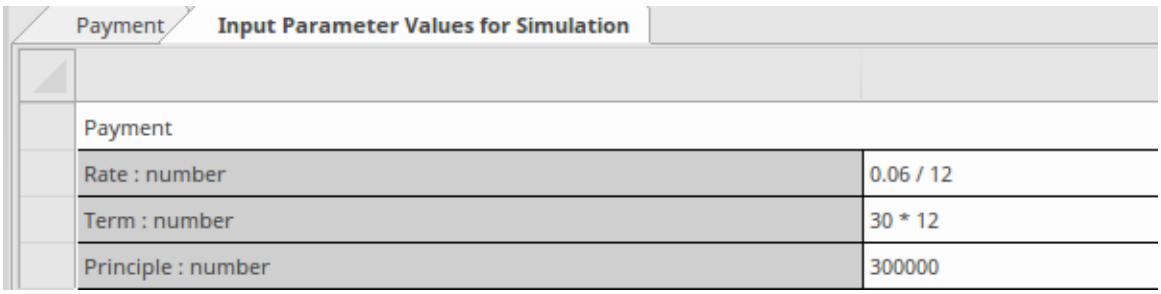
示例-偿还贷款

该业务知识模型（BKM）支付实现为文字表达。

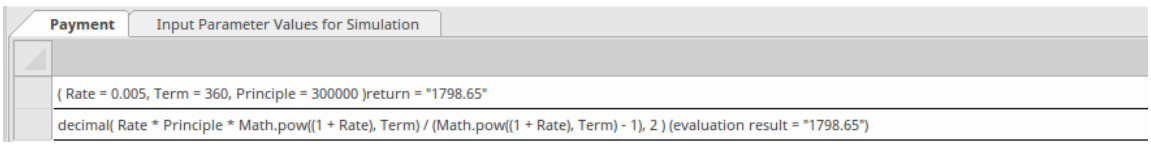


- BKM 定义了三个参数：Rate、团队和 Principle

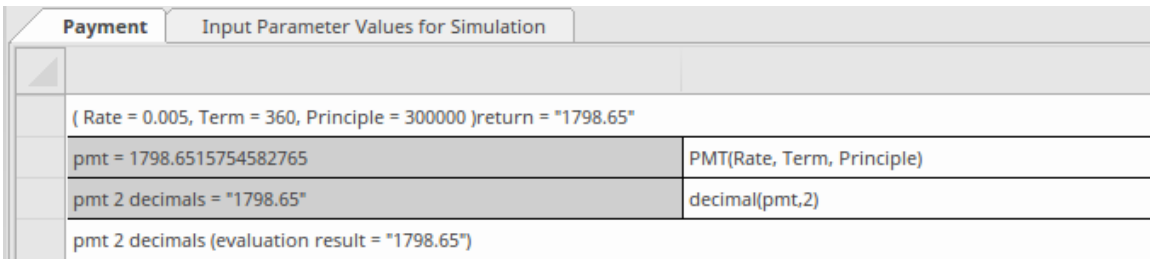
设置输入参数的值并评估模型：



- 将显示运行时参数值；例如，速率 = 00.005
- BKM 的结果将通过字面表达式求值，并在声明行显示值；例如，返回 = 1798.65

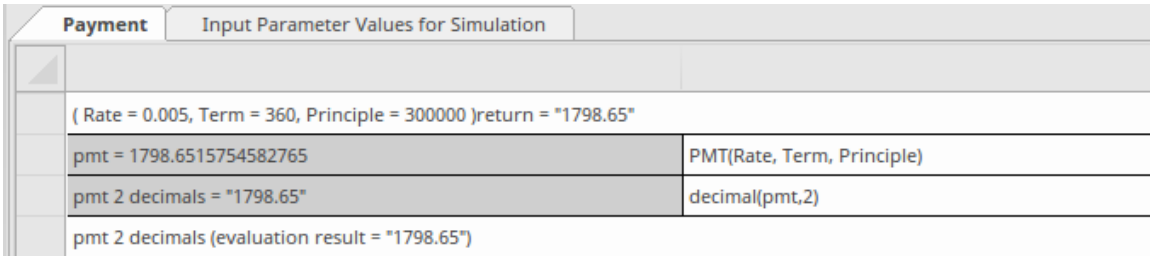


虽然这个公式可以写成一行，但是相当复杂。我们可以使用内置函数和盒装上下文重构此模型以提高可读性：



- Boxed Context 定义了两个变量表达式配对条目；这些变量用作 局部变量”，可以在以后的表达式中使用
- 返回值：表达式可以使用'局部变量'的值
- Boxed Context 中的任何表达式都可以使用在可定制模板 - *DMN* 库中定义的内置函数；例如，此示例中使用了函数 PMT(...) 和 decimal(...)

模拟结果与 Literal Expression 完全相同：


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")

盒装上下文

上下文Context 是时间条目A集合，以表的形式呈现，后跟最终结果表达式。

这些上下文条目由一个与值表达式配对的变量组成，可以被认为是中间结果。这允许将复杂的表达式分解为一系列简单的表达式，最终结果以更简单的形式进行评估。

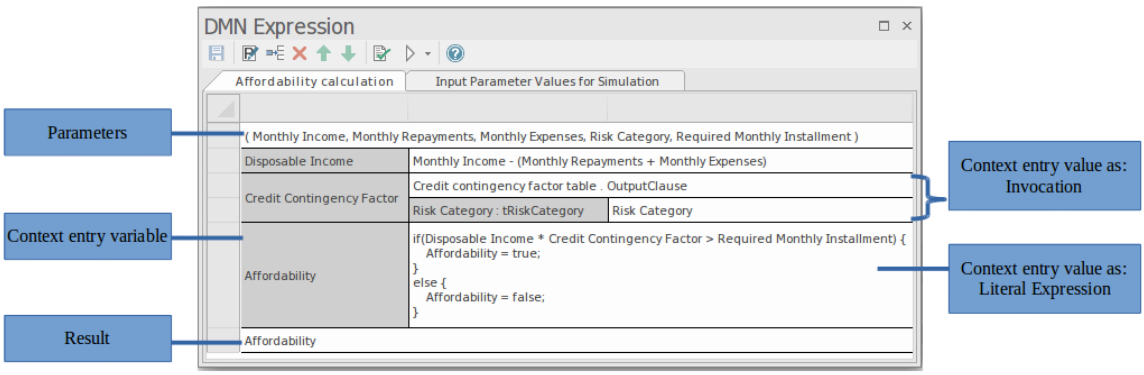
决策和决策业务知识模型元素类型都支持盒装上下文类型。它由 图标表示。

访问

图表	在图表上，双击决策元素或 BKM元素。 随即显示DMN 表达式编辑器窗口，显示所选元素的详细信息。

概述

此图像显示了为盒装上下文表达式显示的DMN 表达式编辑器窗口。

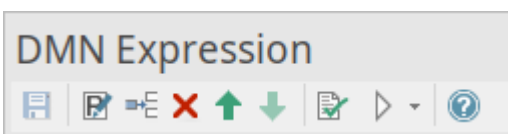


上下文Context 是时间条目A集合，以表的形式呈现，后跟最终结果表达式。每个上下文条目由一个变量和一个值表达式组成。该变量可以被认为是一个中间结果，它可以在任何后续上下文条目的值表达式中使用。上下文条目的值表达式可以是文字表达式或调用，并且可以使用任何可用的输入，例如参数（到 BKM元素）、InputData 或决策结果，以及任何先前定义上下文变量。

Boxed Context 表达式的最终结果是通过依次处理每个上下文条目、评估值表达式并将其结果分配给变量，然后最终评估结果表达式来确定的。结果表达式也可以使用任何输入或局部变量，但必须计算以提供结果。

盒装上下文编辑器的工具栏

选择布局表达式时， DMN 表达式窗口中可访问的地形特征为：

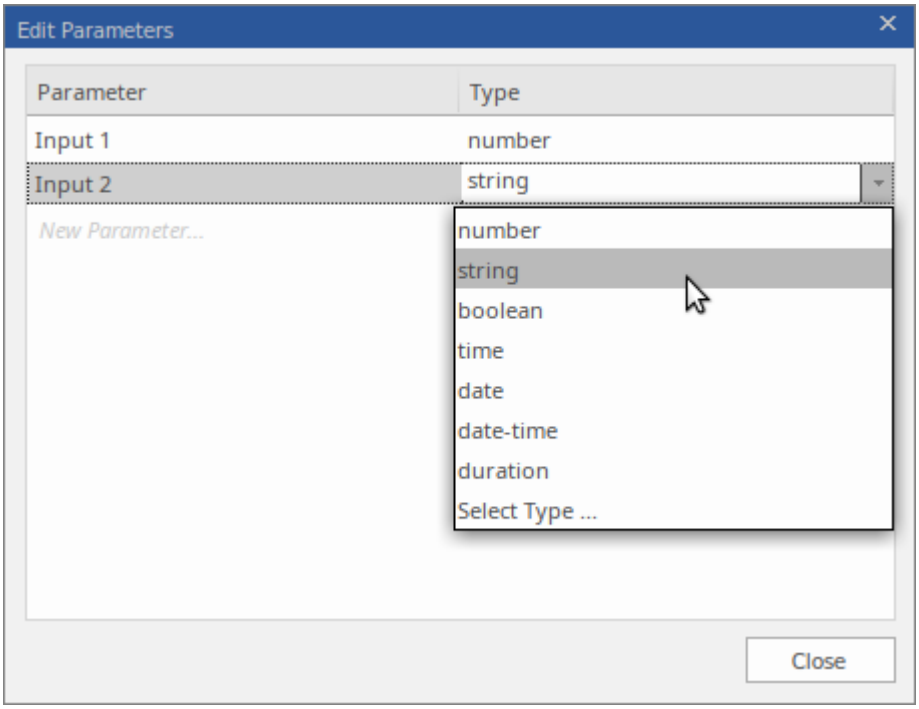


更多详情请参考帮助主题 帮助*Context Editor* 的工具栏"。

指定参数

对于 BusinessKnowledgeModel 元素，参数用于传递调用元素提供的输入值。使用输入参数评估 BKM 的决策逻辑，并将结果返回给调用元素。默认情况下，使用两个输入参数 "Input 1 "和 "Input 2"创建 BKM元素。

单击DMN 表达式窗口工具栏中的  图标，显示 "编辑参数"窗口。



您可以在此处更改参数名称、设置其数据类型、创建附加参数或删除现有参数。

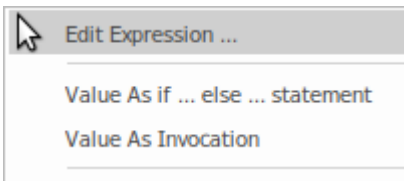
指定上下文条目

每个上下文条目都包含一个变量-表达式对。

变量名可以是您喜欢的任何文本，甚至可以包含空格。要编辑变量名称，点击单元将其选中，然后再次点击或按 F2 进入编辑模式。要退出编辑模式，请单击其他位置或按 Enter 键。

一般，没有必要为表达式或变量指定数据类型 - 类型将从值中推断出来。但是，如果您打算为Java 、C++ 或 C# 等编译语言生成代码，则必须指定所有上下文条目变量的类型。

上下文条目的值表达可以是文字表达或调用，并且可以利用任何可用的输入，例如参数（对业务知识模型元素）、输入数据或决策结果，以及任何先前定义上下文变量.右键单击表达式单元会显示一个弹出菜单，该菜单提供用于显示表达式代码编辑器或将值表达式设置为 If-Else 语句或调用的选项。



您还可以通过在表达式中直接输入文本来编辑值单元。

有关如何指定文字表达式或调用的更多信息，请参阅帮助这些主题的帮助主题。

盒装上下文编辑器的工具栏

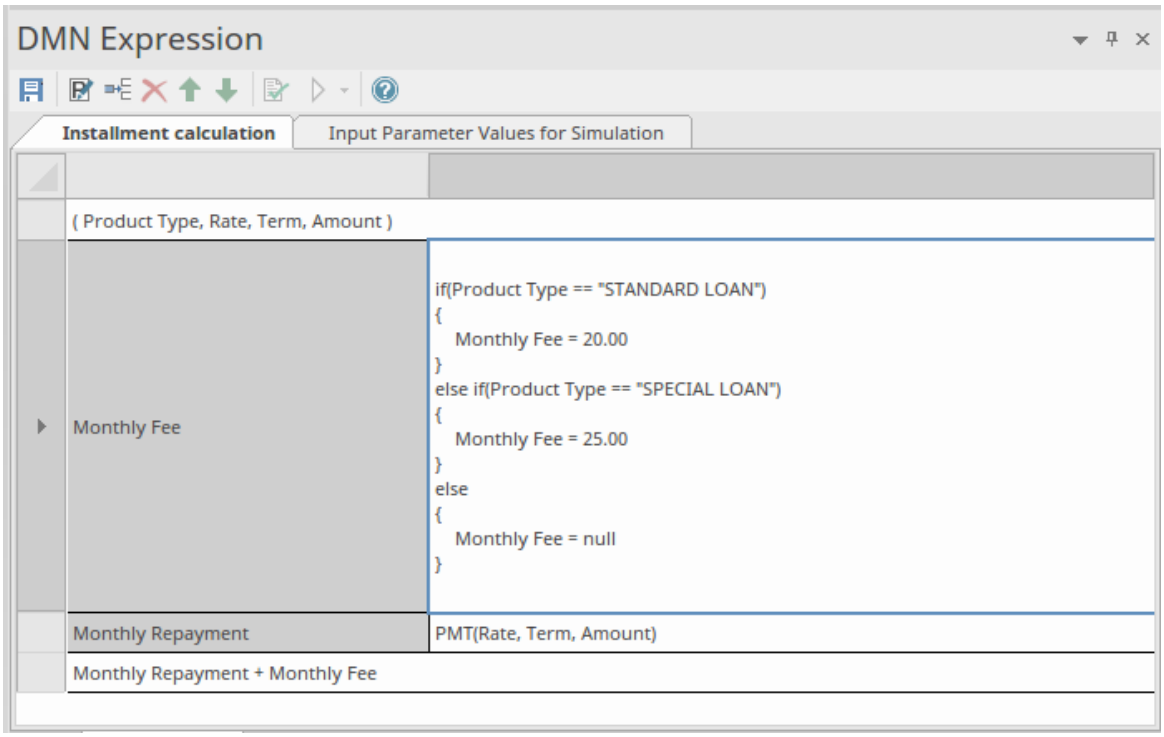
此表提供了在选择盒装上下文时在DMN 表达式窗口中可访问的特征的描述。

工具栏选项

此工具栏适用于盒装上下文。

选项	描述
	保存对当前选择的决策或 BusinessKnowledgeModel元素的更改。
	显示 编辑参数"窗口，您可以在其中指定调用 BusinessKnowledgeModel元素的决策逻辑时传递的每个参数的名称和数据类型。
	创建一个新的上下文条目并将其附加到上下文列表中。
	删除当前选定的上下文条目。
	将当前选定的上下文条目在列表中上移一位。
	将当前选定的上下文条目在列表中下移一位。
	执行BoxedContext的验证。 Enterprise Architect将执行一系列验证，以帮助您发现BoxedContext定义中的任何错误。
	该按钮在为知识业务知识模型元素定义业务决策决策表时启用。 选择 参数for仿真"选项卡，完成字段，然后单击此按钮。测试结果将显示在决策表上，显示输入和输出的运行时值，并突出显示有效规则。 您可以使用此功能对 BusinessKnowledgeModel元素进行单元测试，而无需指定其上下文。 此工具栏按钮A许多菜单选项可用。有关详细信息，请参阅帮助主题仿真帮助仿真模型

示例-贷款分期计算



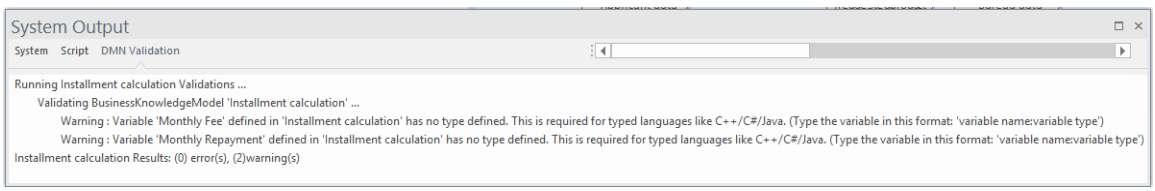
业务知识模型（BKM）分期计算以盒装上下文的形式实现。

- BKM 定义了四个参数：类型、Rate、团队和 Amount
- Boxed Context 定义了两个变量-表达式对条目；这些变量充当 局部变量”，可以在以后的表达式中使用
- 返回值：表达式可以使用'局部变量'的值
- 盒装上下文中的任何表达式都可以使用内置函数，这些函数在可定制的模板 - *DMN* 库中定义；本例中使用了函数 PMT(...) 和 decimal(...)

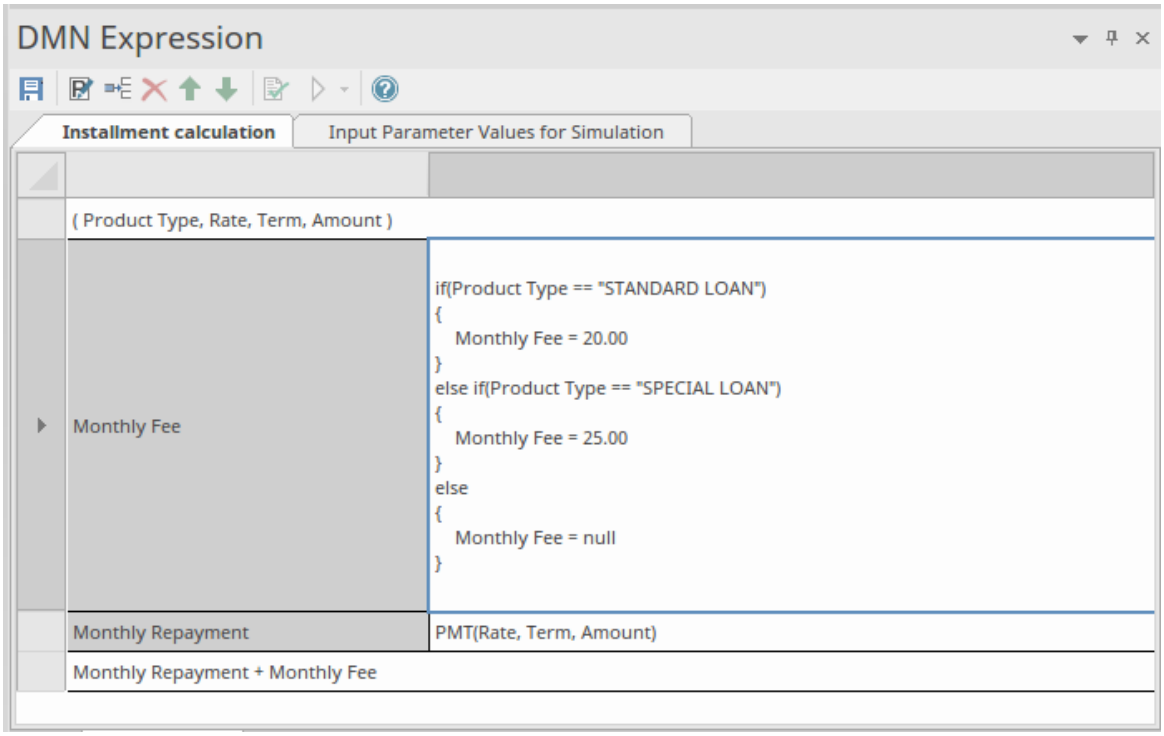
为上下文条目变量指定类型

一般，表达式和变量不必指定类型，类型是从提供的值推断出来的。此特征一般由JavaScript支持，用于Enterprise Architect的 DMN仿真。

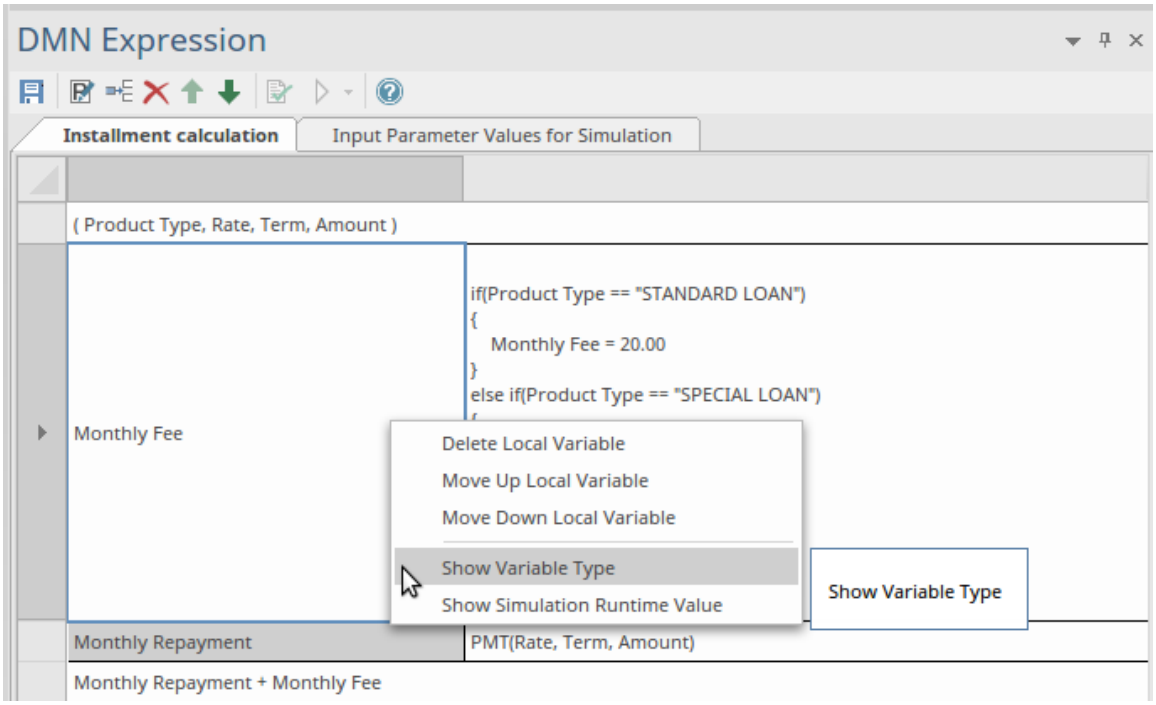
但是，如果您想将 DMN模型中的代码生成为Java 、C++ 或 C# 等编译语言，则必须为每个上下文条目变量指定类型。否则，如果您验证模型，您将看到如下警告：



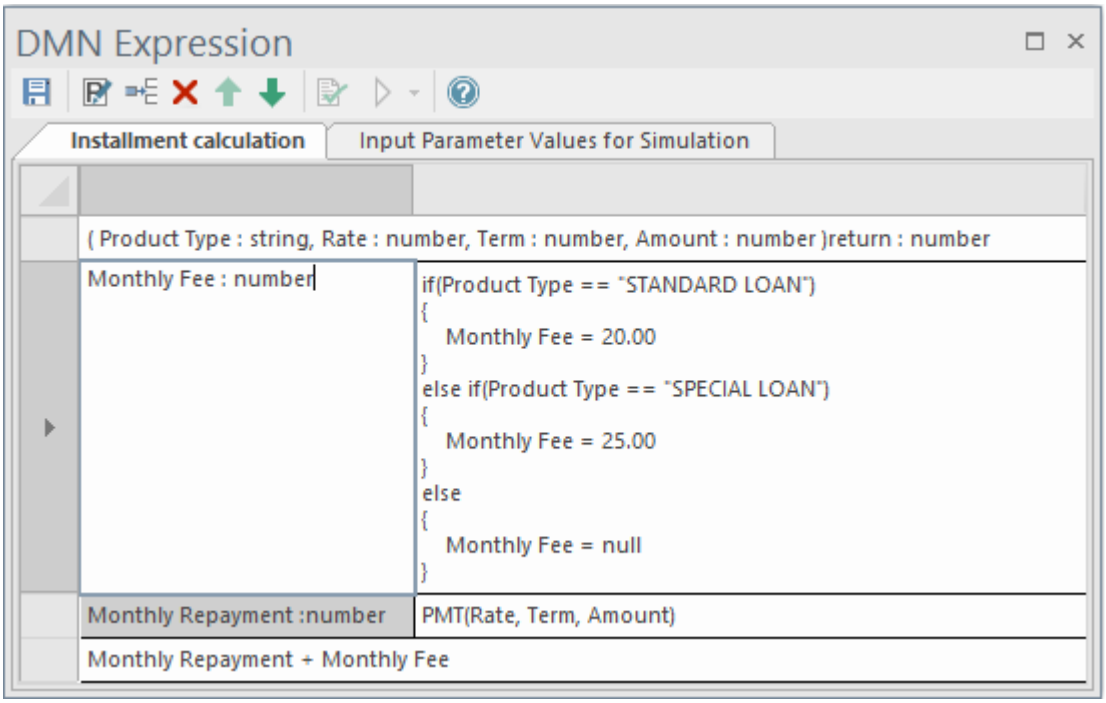
右键单击此模型中的上下文条目变量（月费、月还款）。



选择 显示变量类型”选项。



现在输入变量类型，将其附加到变量名称并用冒号分隔，如此处所示。

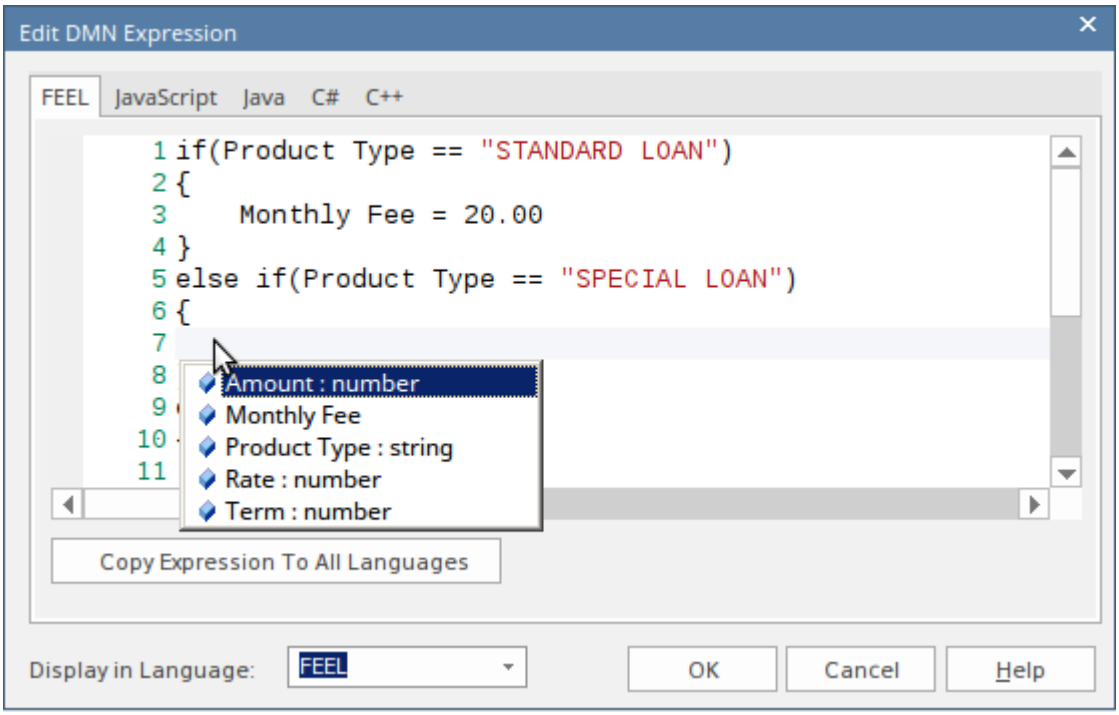


然后单击工具栏上的 Save 按钮保存表达式，然后单击 按钮再次验证模型。

表情编辑器和智能感知支持

根据 FEEL 语言规范，参数和上下文条目的变量名称可以包含空格。这个特征使表达式易于阅读。为了帮助您用更少的输入和更少的错误编辑表达式， Enterprise Architect智能感知了对编辑表达式的支持：

要编辑表达式，请右键单击表达式（在右侧字段中）并选择 编辑表达式”菜单选项。将显示 表达式”对话框。单击并按 Ctrl+Space 以显示所需的智能感知菜单行：



- 将包含所有早于当前的上下文条目变量（不包括晚于当前的上下文条目变量）

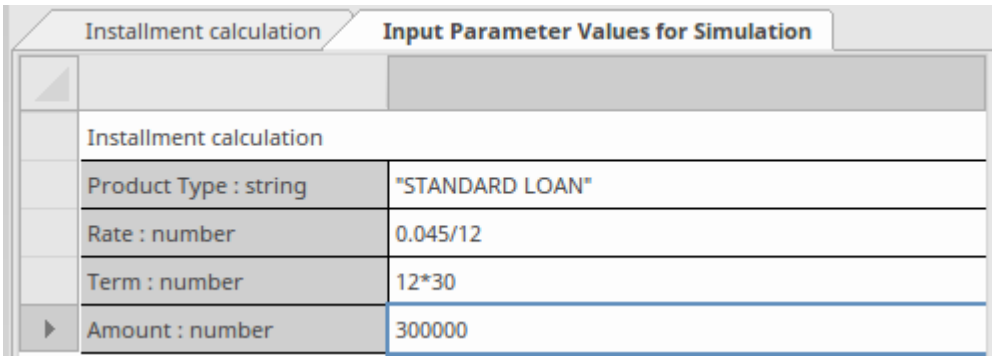
- 对于业务知识模型（BKM），将包含所有参数
- 对于决策，将包括所有必需的决策

DMN模型可以生成为JavaScript 、 Java 、C# 和 C++ 的源代码。由于某些语言可能对某些表达式有不同的语法， Enterprise Architect为每种语言提供了语言覆盖页面。如果没有为语言指定覆盖代码，则将使用为 FEEL 语言定义的表达式。

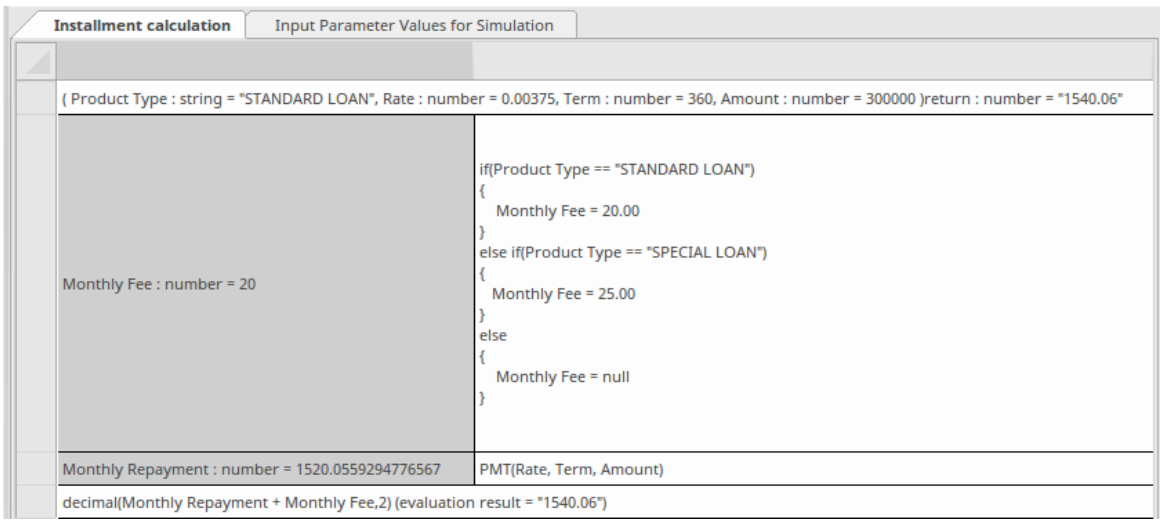
在生成的代码中，变量名中的空格将被替换为下划线。

业务知识模型仿真

选择 用于仿真的输入参数值”选项卡并完成每个字段。



单击保存按钮，然后单击工具栏上的仿真按钮；测试结果将显示在 Boxed Context 表达式中。



- 将显示运行时参数值；例如， 速率 = 0.00375”
- 将显示 上下文条目”变量的运行时值；例如， 每月还款额 = 1520.06”
- 业务知识模型（BKM）的结果将根据最后一个条目和申报行显示的值进行评估；例如，'return = 1540.06'

您可以使用此功能对 BKM 进行单元测试，而无需了解上下文，以便稍后可以由决策或另一个 BKM 调用。

盒装清单

DMN 盒装列表是A决策元素，其中包含盒装表达式列表。这些项目在DMN 表达式窗口中以垂直列表的形式排列。

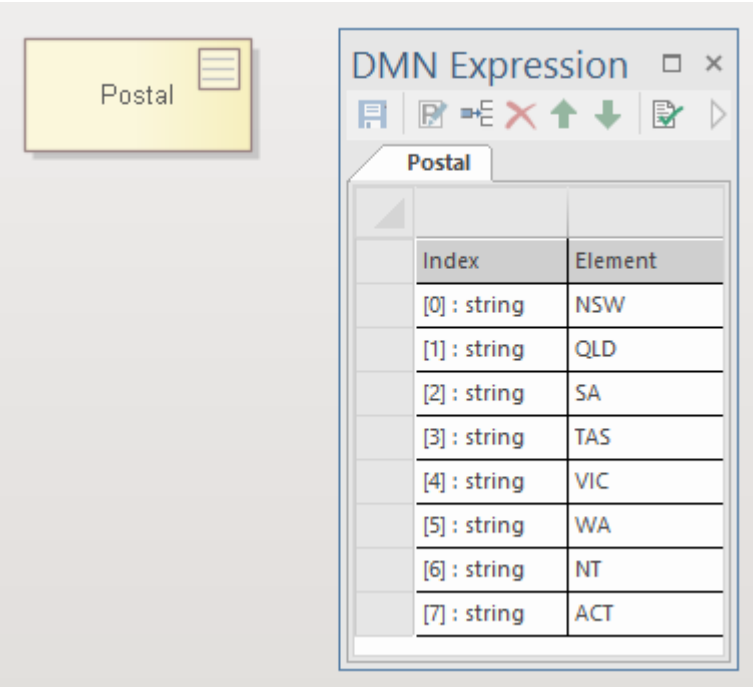
盒装A通常与包含在相关决策元素中的*for* 循环表达式结合使用。 *for* 循环表达式用于遍历盒装列表中的每一行，将列表的元素字段绑定到相应的变量，并在范围内评估表达式。 *for* 循环的输出是一个列表，其中包含每个单独迭代的表达式评估。

访问

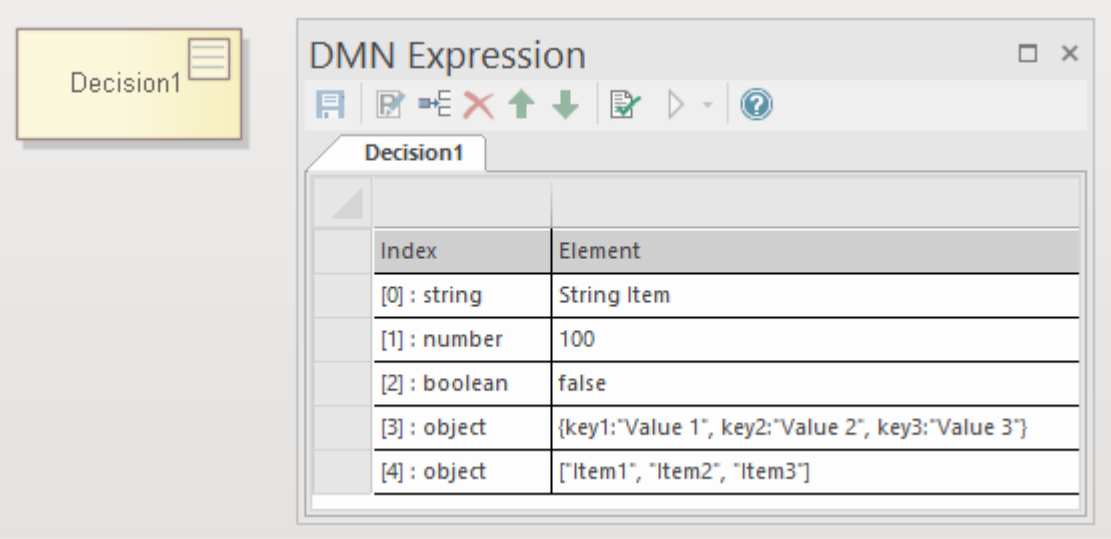
图表工具箱	将工具箱中的工具箱元素或 BKM元素拖到决策图上，然后从弹出的表达式菜单中选择 列表”。 双击DMN元素；显示DMN 表达式窗口，显示所选元素的详细信息。
属性	右键单击图表上的 DMN决策或 BKM元素，然后选择 属性|属性”菜单选项。 在 常规”页面上，选择 标签”选项卡，然后在 表达式类型”值字段中单击下拉箭头并选择 列表”。点击确定按钮。 双击DMN元素；显示DMN 表达式窗口，显示所选元素的详细信息。

概述

盒装列表通常用作枚举，其中所有列表项都属于同一类型。



将盒装列表用作数据集合也很常见，其中每个项目可能具有不同的类型。

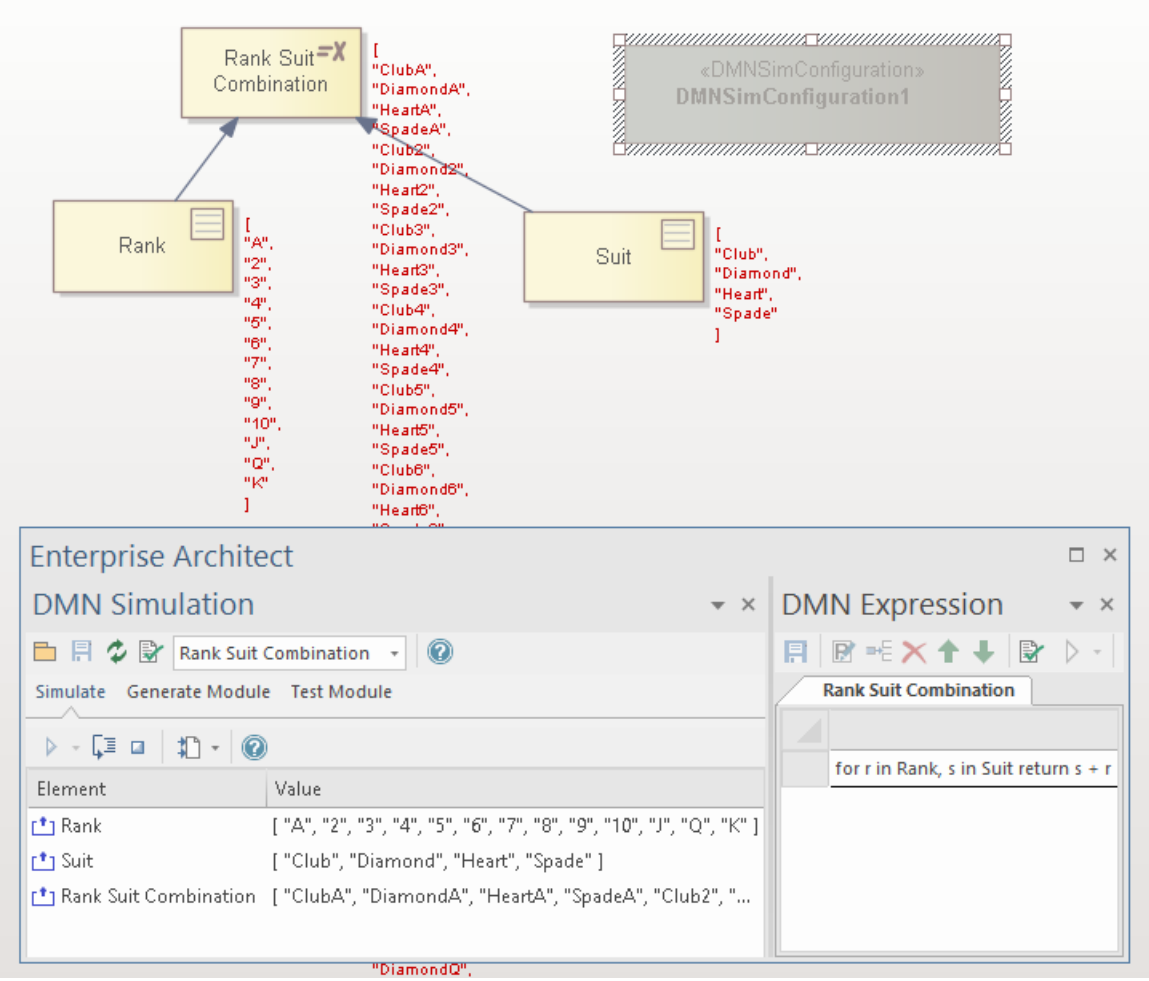


编辑盒装列表

DMN 表达式窗口有一个工具栏，提供 加新列表项"、 删除现有列表项"和 向上或向下移动项"选项。右键单击列表项以显示上下文菜单选项，用于设置列表项的类型 - string 、数字、布尔值或object 。

示例- 扑克等级和花色

在这个例子中，我们有三个决策：Rank、Suit 和 Rank Suit Combination



- 决策表示为一个项列表，包含从 "A "到 "K "的 13 个项目
- 决策表示为带有 4项的盒装列表： 俱乐部"、 钻石"、 心"和 黑桃"
- 决策Suit Combination 表示为带有*for*循环的文字表达式： *for r in Rank, s in Suit return s + r*

当在同一个*for*循环表达式中定义多个迭代上下文时，生成的迭代是迭代上下文元素的叉积。迭代顺序是从内迭代上下文到外迭代上下文。

在此示例中，Rank（13 项）和 Suit（4 项）的叉积是 13 * 4 = 52 个元素的列表。

关系

DMN决策关系元素提供了A方便的简写方法，用于在 DMN 图中定义相关值的列表。决策关系就像A有列和行的关系表。网格的标题显示每列的名称。每行显示对应列的值集。

访问

工具箱	创建决策关系： • 确保蓝图设置为：需求>决策建模 • 从工具箱的图表Components 页面，将决策元素或 BKM元素拖到决策图上 • 从弹出的表达式菜单中选择 关系” • 双击 DMN元素以显示DMN 表达式窗口。

或者，您可以将现有的决策决策或 BKM元素更改为决策关系类型。去做这个：

- 右键单击图表上的 DMN决策或 BKM元素，然后选择 属性|属性”菜单选项
- 在 常规”页面上，选择 标签”选项卡，在 表达式类型”值字段中单击下拉箭头并选择 关系”；点击确定按钮

概述

DMN关系决策类型是包含值行的垂直列表。使用决策关系的A关键是使用*For* 循环遍历值行的方法。例如，可以在相关的文字表达式元素决策中将决策循环定义为处理决策关系中的行的公式。

编辑 DMN 关系

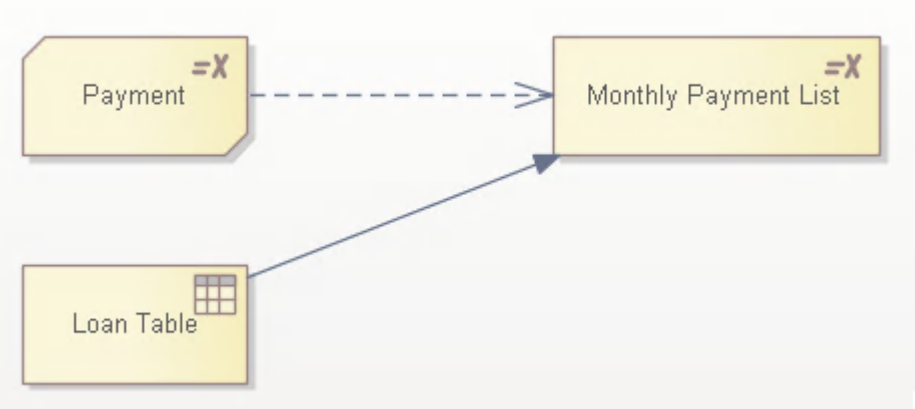
DMN 表达式窗口有一个工具栏，提供添加新行、删除现有行以及向上或向下移动选定行的选项。

你也可以：

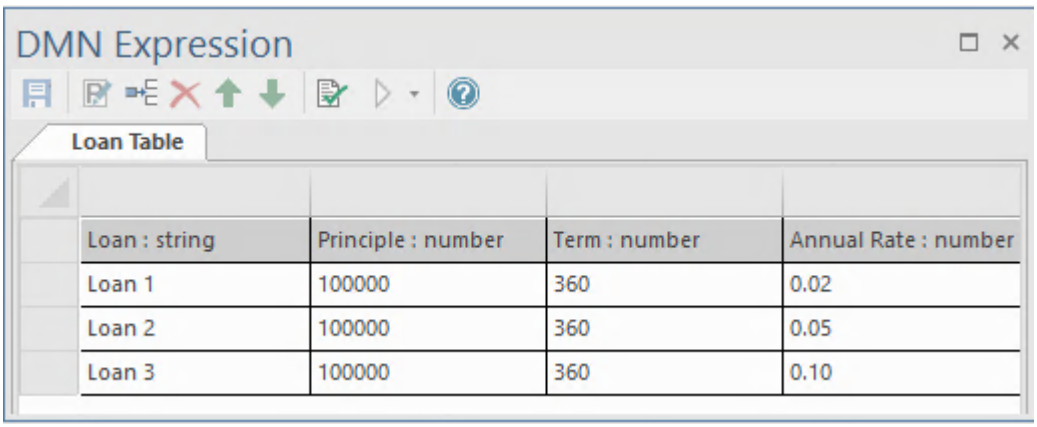
- 拖动网格标题以重新定位列。
- 右键单击单元以显示上下文菜单选项，用于将列的类型设置为：' string '、'number'、'boolean'或' object '。

示例-贷款库表

在这个例子中，我们有两个决策—— 贷款库表”和 每月付款清单”——以及一个业务知识模型—— 付款”。

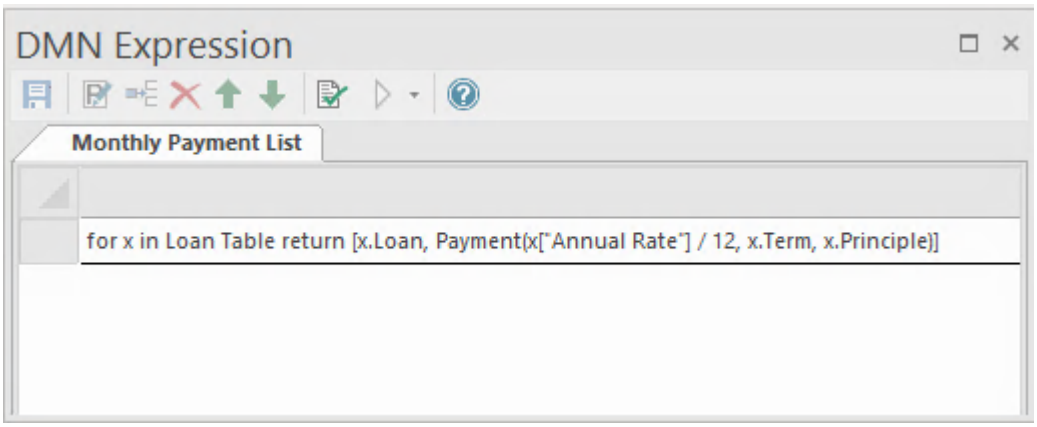


贷款库表"是一个以关系实施的决策，有四列： 贷款"、 原则"、 团队"和 年利率"。



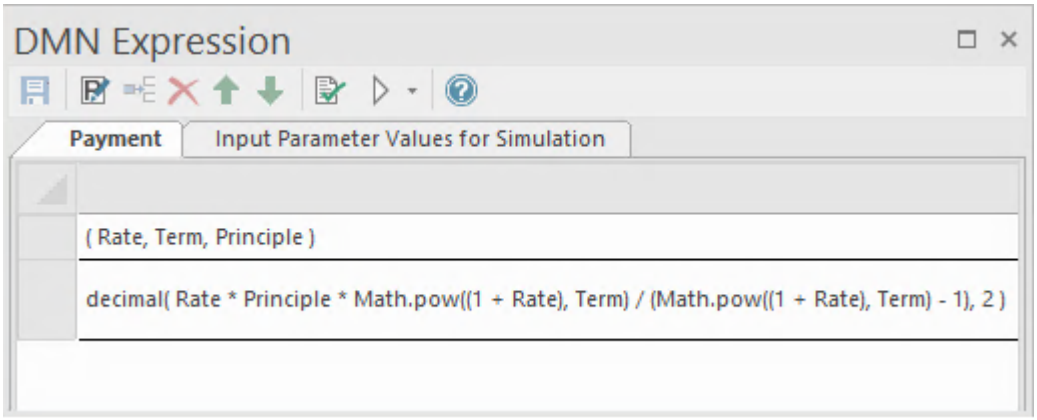
Loan : string	Principle : number	Term : number	Annual Rate : number
Loan 1	100000	360	0.02
Loan 2	100000	360	0.05
Loan 3	100000	360	0.10

每月付款清单"是一种决策，以带有*for*循环的文字表达式实现：

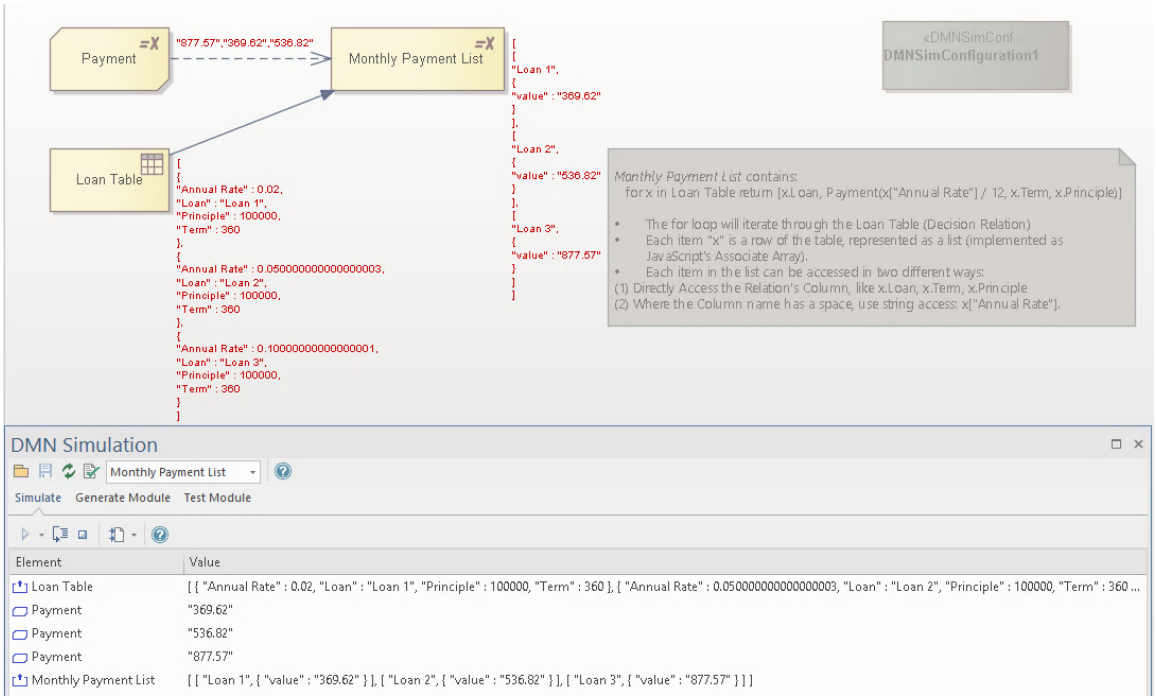


*for*循环将遍历 贷款库表"：

- 每个项目 'x' 是表的一行，表示为一个列表
- 对每一行项目 x"，调用业务知识模型 支付"与行列表中的项目
- 可以通过两种不同的方式访问列表中的每个项目：
  (1) 直接访问Relation的列，如x.Loan,团队, x.Principle
  (2) 列名有空格的地方，使用string访问：x["Annual Rate"]



仿真时，运行时值显示在仿真窗口和元素旁边的图表中；您可以单击一个步骤来查看模拟过程。



调用

调用是为评估上下文业务知识模型主体提供时间的参数绑定的容器。调用有两种常见用例：

- 将输入数据绑定到业务知识模型
- 将参数或上下文入口变量绑定到业务知识模型

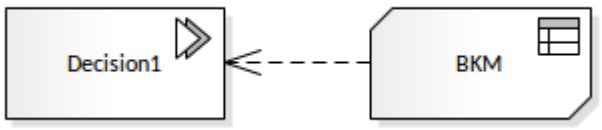
本帮助主题的子主题中提供了每个示例。

访问

图表	双击适当的决策元素或 BKM元素。 显示DMN 表达式窗口，显示所选元素的详细信息。

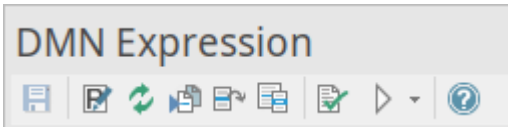
概述

调用是一种适用于决策要素和决策业务知识模型要素的价值表达类型。它是一个表格表示，一个决策或另一个业务知识模型调用一个可调用元素（一个决策业务知识模型或一个决策服务）中定义的决策逻辑。



调用编辑器工具栏

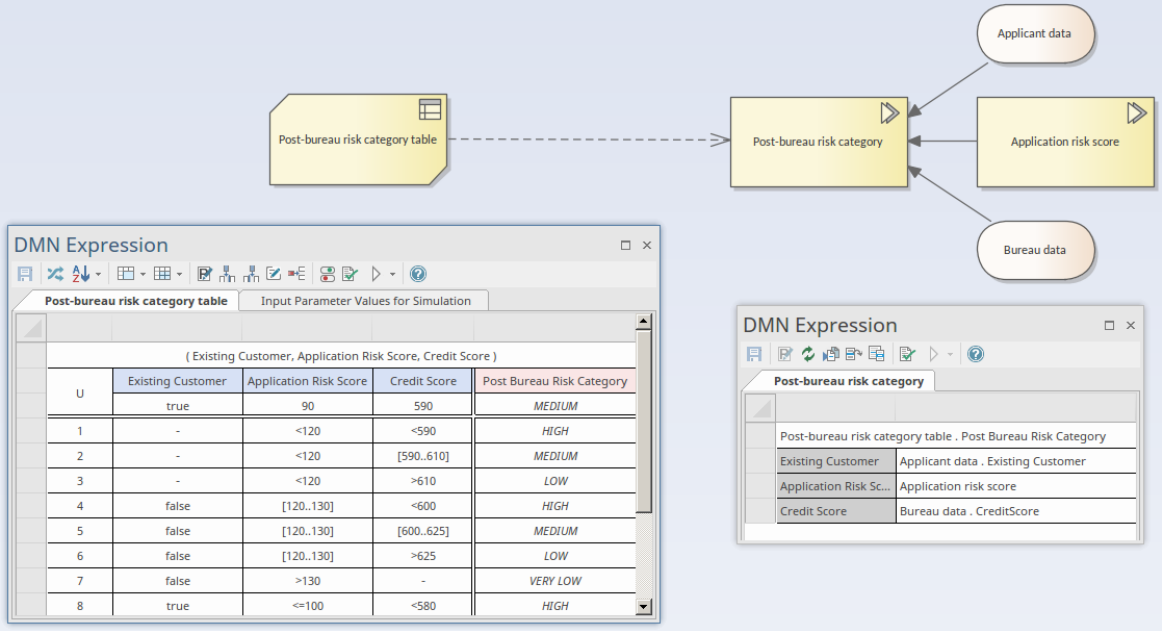
选择一个调用后，可以从功能DMN 表达式窗口的工具栏访问许多处理它的功能：



更多详细信息请参考帮助主题 调用编辑器工具栏”。

绑定

调用的参数上下文提供了评估可调用元素主体的时间。

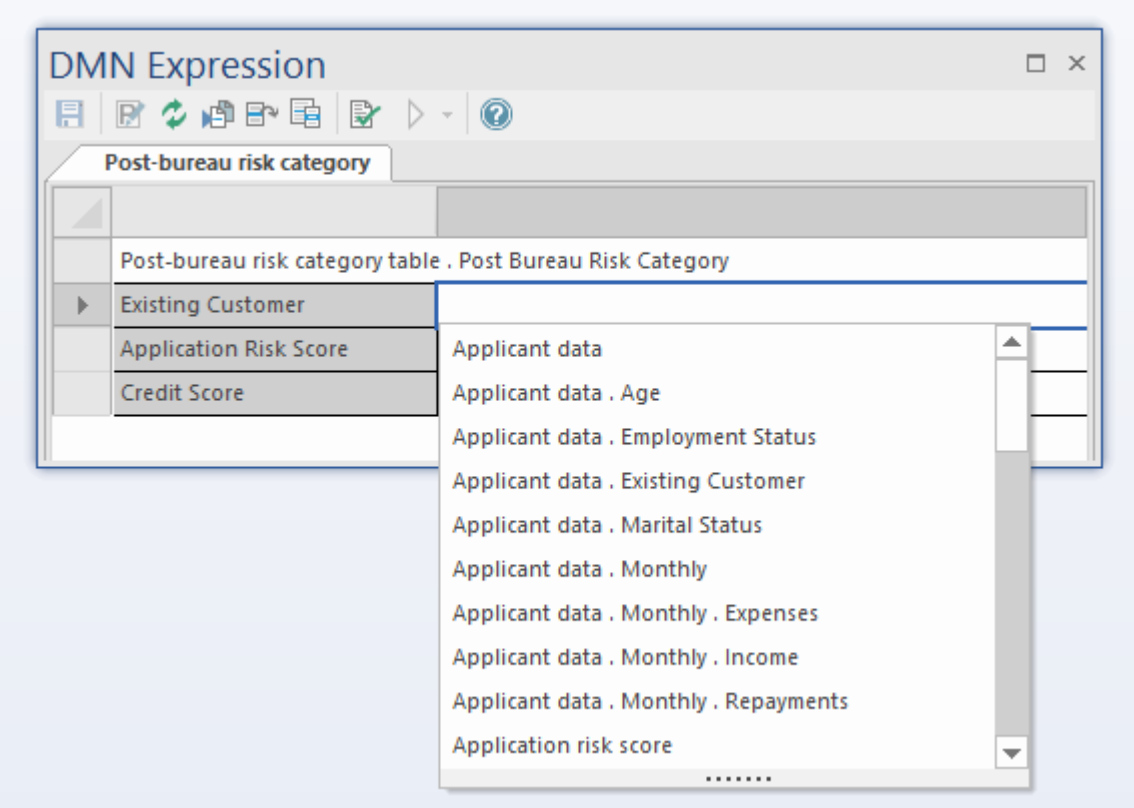


在这个例子中：

- 将'Post-bureau risk category'表示为与决策业务知识模型 "局后风险类别表"相关联的调用，以决策决策表的形式实现
- 决策决策'Post-bureau risk category'是来自两个输入数据元素和一个决策要素的三个信息需求连接器的目标
- 绑定列表将输入值绑定到业务知识模型的参数
- 调用还指定了请求的 "OutputClause"；在决策表定义了多个输出子句的情况下，调用必须明确请求输出子句作为表达式的结果

输入

可以通过在字段中按空格键来设置来自其他决策和 InputData 元素的输入：



输出

由于Invocation只能调用一个业务知识模型，输出由业务知识模型输出定义。

调用编辑器工具栏

选择调用表达式时， DMN 表达式窗口工具栏提供特定于该表达式类型的选项。

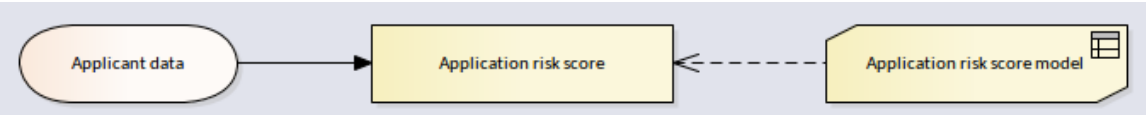
工具栏选项

此表提供了在选择调用时在DMN 表达式窗口中可访问的特征的描述。

选项	描述
	单击此按钮可将配置保存到当前决策或 BusinessKnowledgeModel。
	单击此按钮可编辑业务知识模型的参数。
	适用于调用值表达式，适用于决策元素和业务知识模型(BKM) 元素。 单击此按钮可与调用的 BKM 同步。例如，如果 BKM 更改名称、参数、输出或类型，请单击此按钮以同步这些更改。
	适用于调用值表达式，适用于决策元素和业务知识模型(BKM) 元素。 单击此按钮可设置或更改 BKM 作为调用。
	适用于调用值表达式，适用于决策元素和业务知识模型(BKM) 元素。 单击此按钮可在DMN 表达式窗口中打开调用的 BKM。
	适用于调用值表达式，适用于决策元素和业务知识模型(BKM) 元素。 当一个BKM被实现为决策表时，它可以定义多个输出子句；对此 BKM 的调用可能必须指定请求的输出。 单击此按钮可在上下文菜单中列出所有可用的输出；检查当前配置的输出。
	执行调用的验证。 Enterprise Architect将执行一系列验证以帮助您定位调用定义中的任何错误。
	此按钮在为业务知识模型定义调用时启用。 选择 用于仿真的输入参数值”选项卡，完成字段并单击此按钮。测试结果将显示在决策表上，显示输入和输出的运行时值，并突出显示有效规则。 您可以使用此功能对业务知识模型进行单元测试，而无需知道上下文和稍后由决策或其他业务知识模型调用。 此工具栏按钮可使用菜单选项。有关详细信息，请参阅仿真*DMN*模型帮助主题。

示例1 - 将输入数据绑定到业务知识模型

可以A模型模式创建完整示例（在功能区中，选择 仿真>决策分析>决策>应用蓝图>决策决策>决策与决策：创建模型”）。



在此示例中，输入数据申请人数据被键入到申请人数据定义中，该定义具有三个组件。

Applicant data : Applicant data Definition

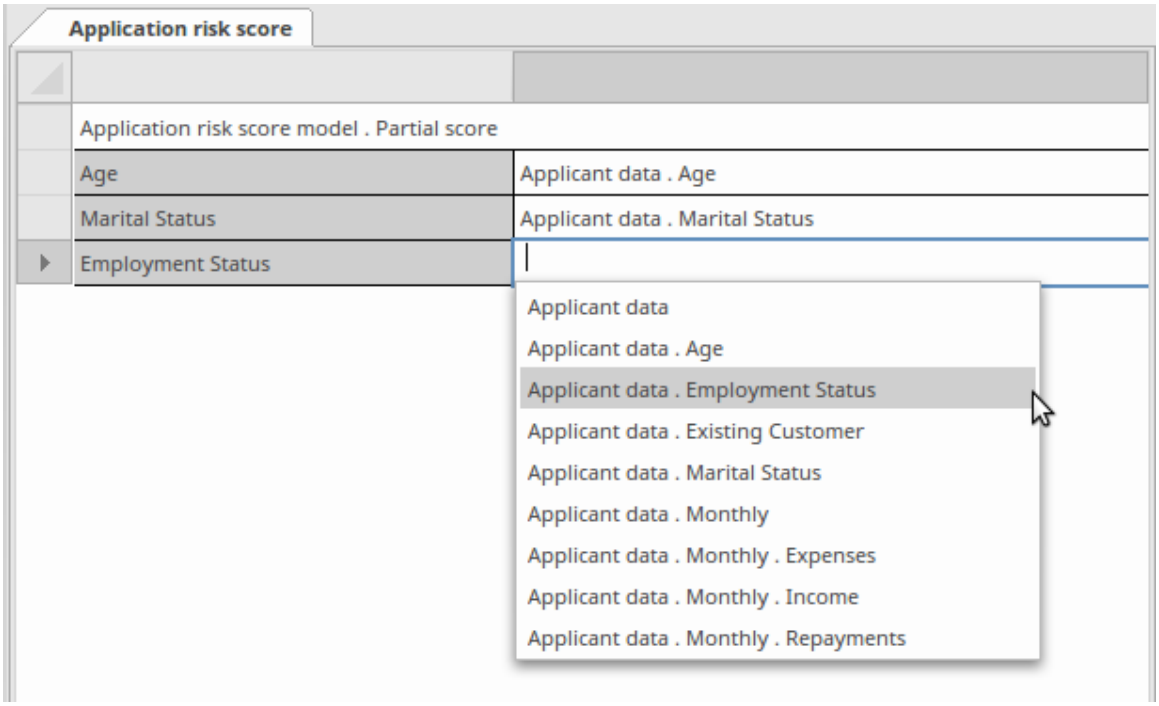
Applicant data Definition	Age : number	40
	Employment Status : string	"EMPLOYED"
	Marital Status : string	"M"

业务知识模型应用风险评分模型实现为三进一出的决策决策表。

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

决策应用程序风险评分被实现为将输入数据的 叶”组件绑定到 BKM 参数的调用。

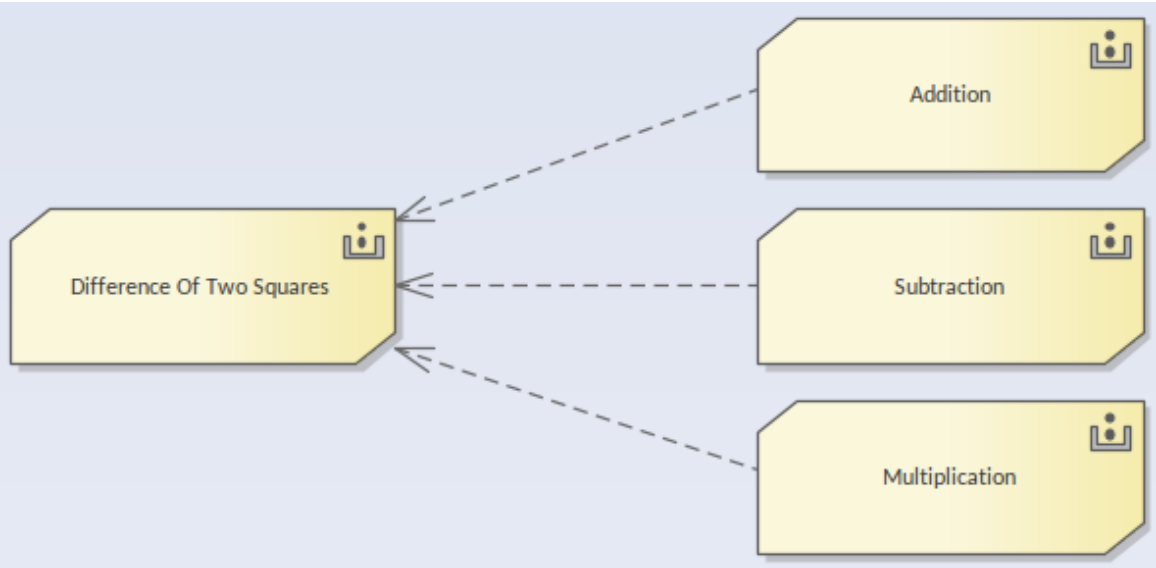


为了使绑定更容易，绑定表达式支持自动完成。

该模式的文档中提供了完整的建模和仿真说明。

示例2 - 将上下文条目变量绑定到业务知识模型

可以A模型模式创建完整示例（在功能区中，选择 仿真>决策分析>决策>应用蓝图>DMN业务知识模型示例>业务知识模型调用：创建模型）。



在本例中，业务知识模型（BKM）两平方差值实现为盒装上下文：

- *ab* 的变量 *sum*通过将参数*a*和*b*绑定到 BKM *Addition*来实现为调用
- *ab* 的变量差异通过绑定参数*a*和*b*到 BKM减法实现为调用
- 通过将局部变量 ab 的*sum* 和 *ab*的差值绑定到 BKM乘法来实现变量的平方差

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		

为了使绑定更容易，绑定表达式支持自动完成。

该模式的文档中提供了完整的建模和仿真说明。

编辑DMN 表达式对话框

无素编辑DMN 表达式”对话框用于设置盒装内容、调用和文字表达式类型中的表达式。它提供智能感知支持，用于构建基于 FEEL 语法的表达式以及可用于模型生成的代码语言。

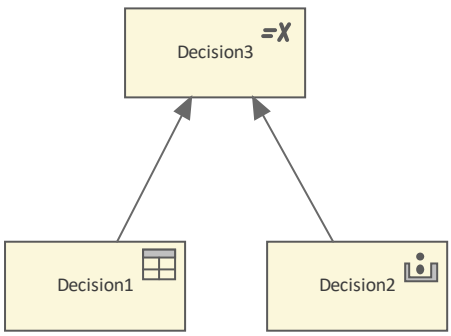
DMN 表达式表达编辑和智能感知支持

为了帮助您减少输入和错误， Enterprise Architect提供了智能感知表达式编辑支持来编辑表达式。

注记根据 FEEL 语言规范，参数和上下文条目变量名称可以包含空格。此特征旨在使每个表达式易于阅读。

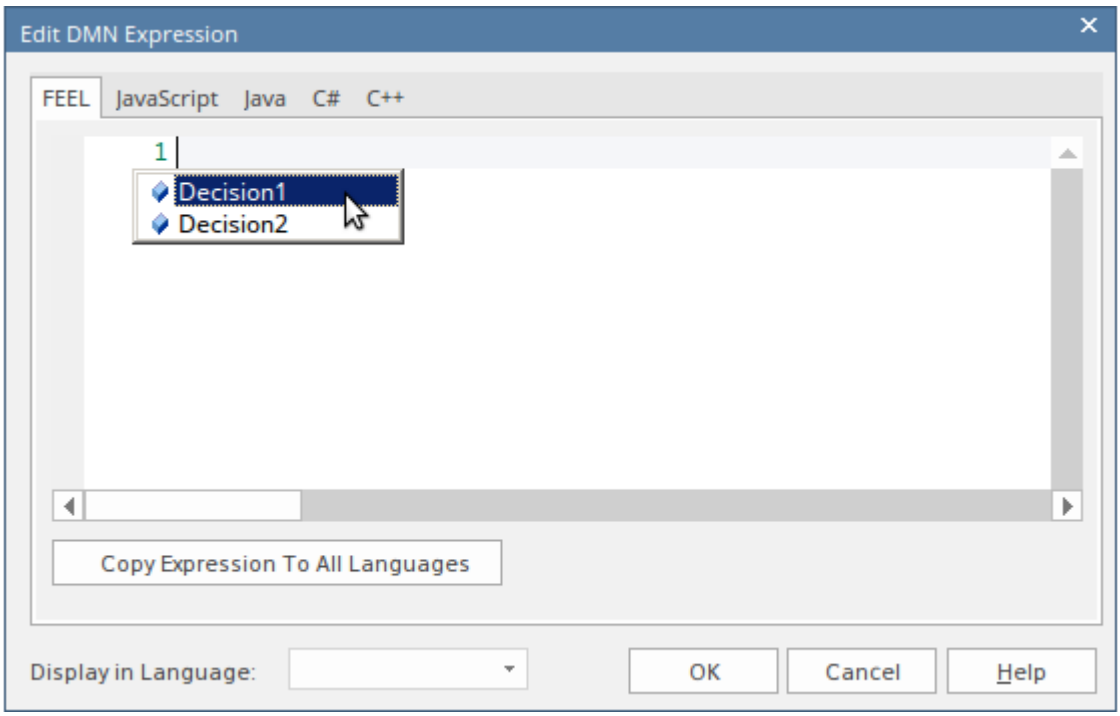
例子

鉴于此决策层次结构，“Decision3”中的表达　能够使用来自两个引用决策的输出。



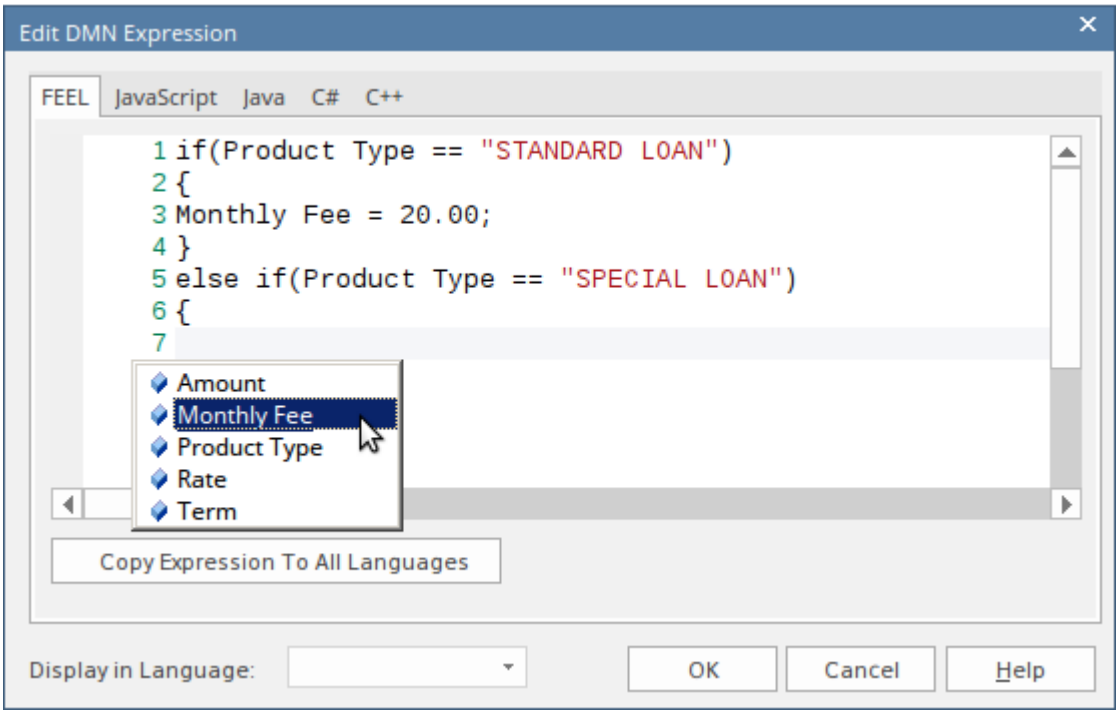
要打开 编辑DMN 表达式”对话框：

1. 双击图中的决策元素，显示DMN 表达式窗口。
2. 右键单击表达式行并选择菜单选项 编辑表达式”。将显示 编辑DMN 表达式”对话框。
3. 单击 a 并按 Ctrl+空格键以显示智能感知菜单：



- 对于 BusinessKnowledgeModel 表达式，将包含所有参数
- 对于决策表达，将包括所有必需的决策
- 将包括早于当前变量的所有上下文条目变量（不包括晚于当前变量的上下文条目变量）

在此示例中，编辑 BKM 盒装上下文表达式，输入参数显示在智能感知菜单中：



语言选择

DMN模型可以生成为JavaScript 、 Java 、C# 或 C++ 的源代码。由于语言之间的语法不同， Enterprise Architect为每种语言提供了语言覆盖页面。如果没有为语言指定覆盖代码，则将使用为 FEEL 语言定义的表达式。

注记：在生成的代码中，变量名中的空格会被替换为下划线。

DMN 表达式验证

DMN 定义了很多表达式，例如 FunctionDefinition、DecisionTable、Boxed Context、Invocation 和 Literal Expression。这些表达式的参数、参数和逻辑主要由 文本”实现。

为了使建模更容易和更可靠， Enterprise Architect提供了两个特征：自动完成和验证。

- 验证：识别由拼写错误、逻辑不完整、不一致等引起的建模错误
- 自动完成：您可以从枚举列表中选择一个文本string ，而不是在

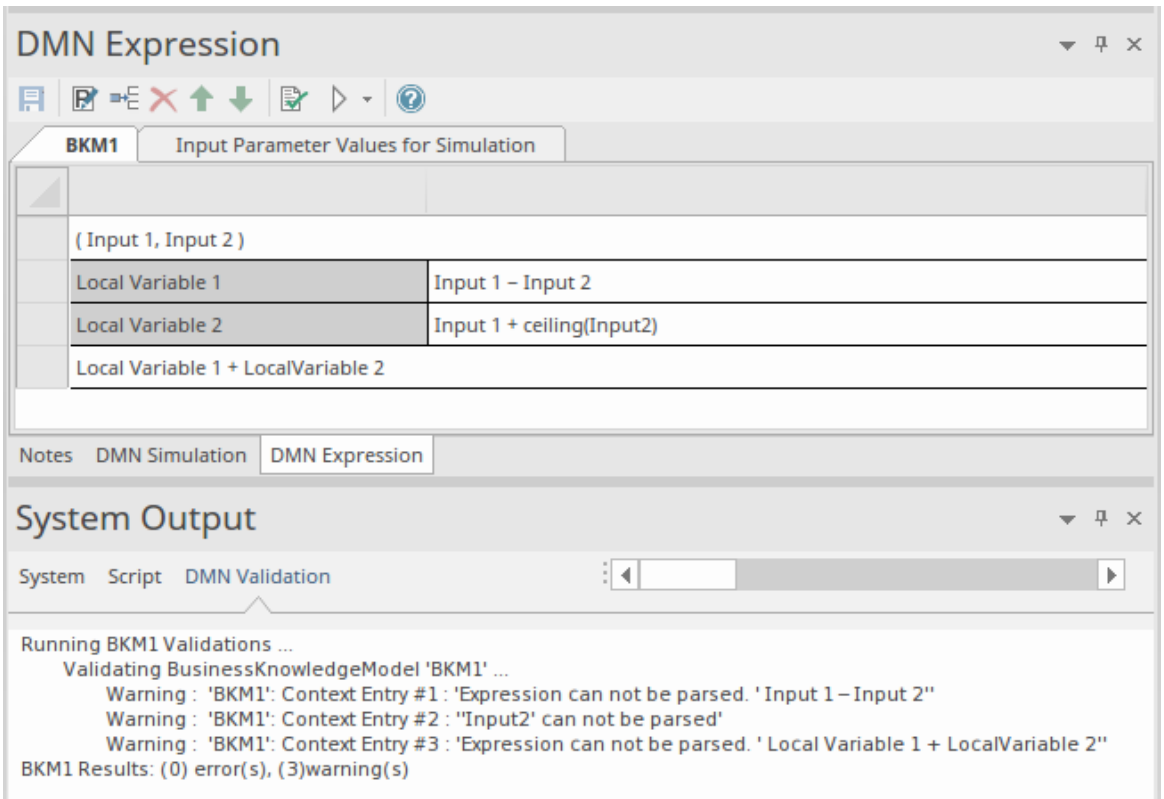
在本主题中，我们将向您展示如何验证DMN 表达式。

访问

DMN 表达式窗口	仿真>决策分析> DMN > DMN 表达式：验证按钮
DMN仿真窗口	仿真>决策分析>DMN>打开DMN仿真>仿真：验证图标

公共验证

变量名称Validation



在本例中，Boxed Context业务知识模型BKM1定义了两个参数 输入1 ”和 输入2”，以及两个局部变量 局部变量1 ”和 局部变量2”。表达 已经过验证，结果输出到系统输出窗口的 DMN Validation”选项卡。

- 上下文条目 #1 失败，因为存在印刷错误；它应该是运算符'-'，但用户在'-'中键入或复制
- Context Entry #2 失败，因为 'Input' 和数字 2 之间没有空格；注记函数'ceiling()' 是在 DMN 库中定义的，因

此可以成功解析

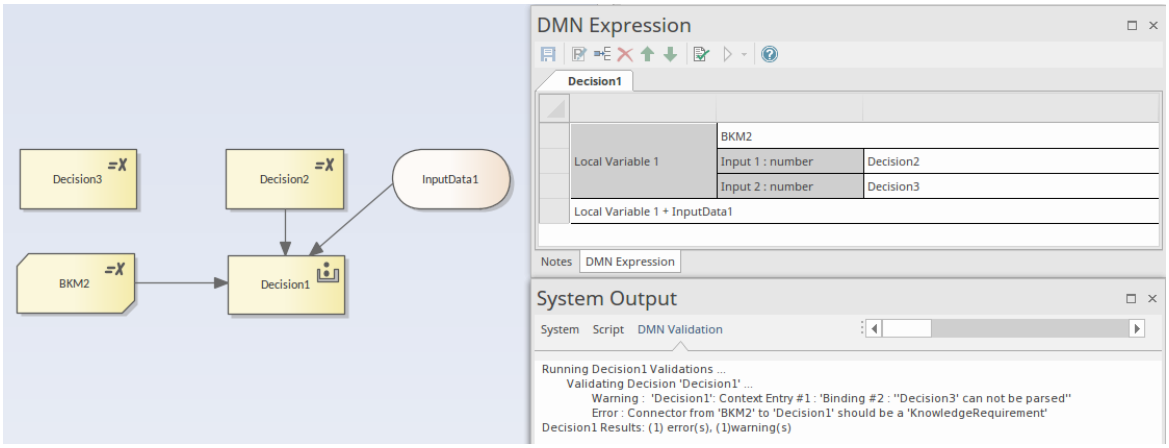
- 上下文条目 #3 失败，因为 本地”和 变量”之间没有空格

很难直观地识别这些类型的错误。运行验证可以帮助识别错误，然后您可以轻松地执行更正。

依赖验证

A决策可能需要其他决策、输入数据和业务知识模型；这些关系由 InformationRequirement 和 KnowledgeRequirement 连接器标识。

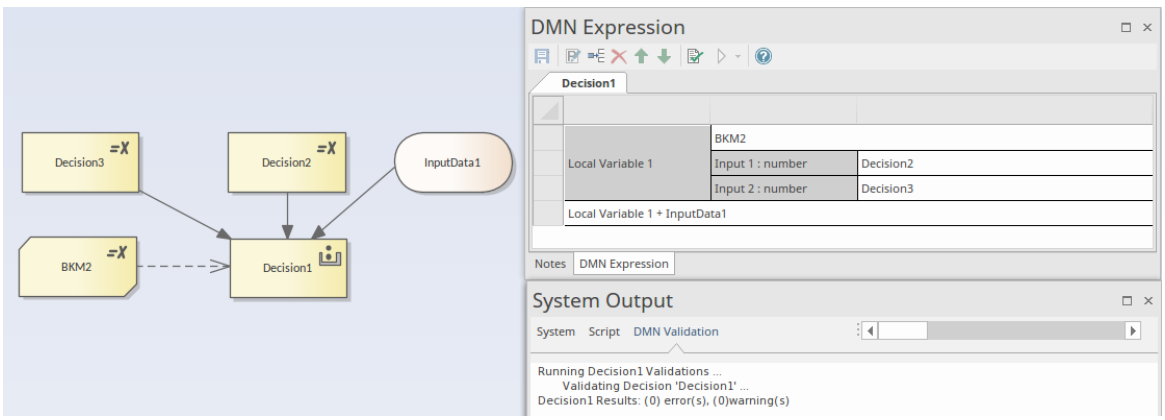
当图表变得复杂时，很可能缺少某些连接器或使用了错误的连接器类型。



在本例中，单击验证按钮， Enterprise Architect将显示：

- 'Decision3' 由'Decision1' 通过绑定到被调用的BKM2 的参数来使用；但是，它没有定义 - 缺少 InformationRequirement 连接器
- “Decision1”中定义的调用无效；从 “BKM2”到 “Decision1”的连接器类型应该是 KnowledgeRequirement

解决这些问题后，再次运行验证：



DMN 表达式自动完成

DMN 定义了很多表达式，例如 FunctionDefinition、DecisionTable、Boxed Context、Invocation 和 Literal Expression。这些表达式的参数、参数和逻辑主要由文本实现。

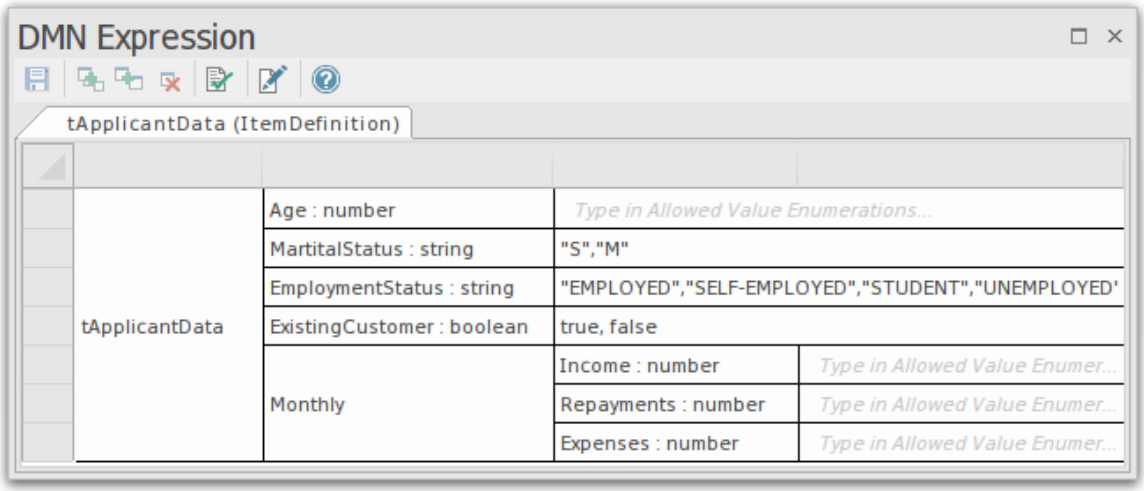
为了使建模简单可靠， Enterprise Architect提供了自动完成功能，帮助提供：

- ItemDefinition 的允许值
- 决策表的输入/输出条目
- 信息需求

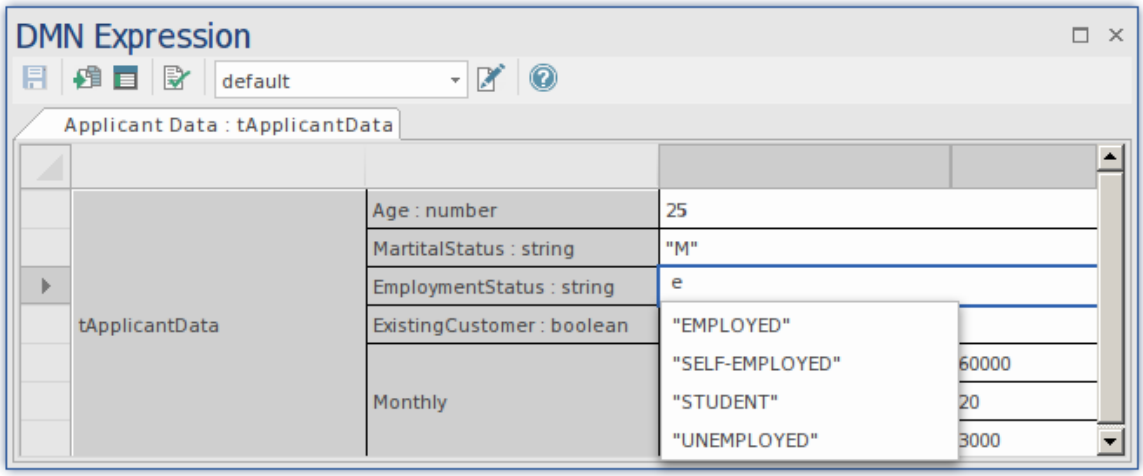
ItemDefinition 的允许值

这个想法是在 ItemDefinition 中定义允许的值枚举，然后在请求这些值时组成一个列表以供选择。

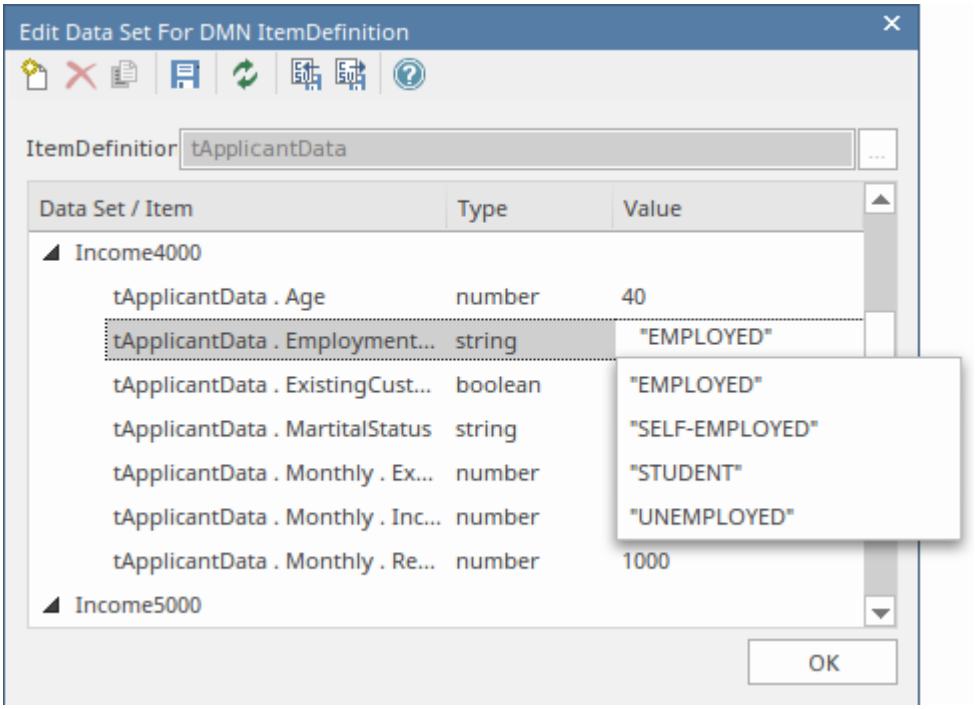
在本例中，ItemDefinition '申请人数据。就业状态'定义了允许值的枚举。



编辑为此 ItemDefinition 键入的 InputData 的值时，按键盘上的空格键以显示可供选择的值列表。

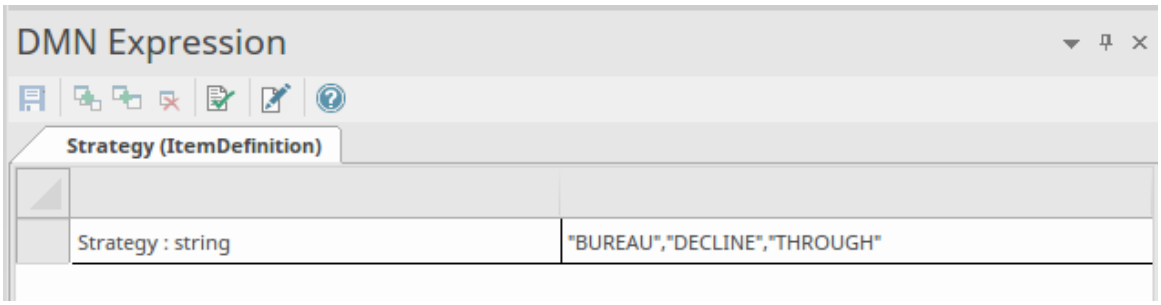


我们还可以为 InputData 定义多个数据集，因为自动完成特征在此对话框中可用。

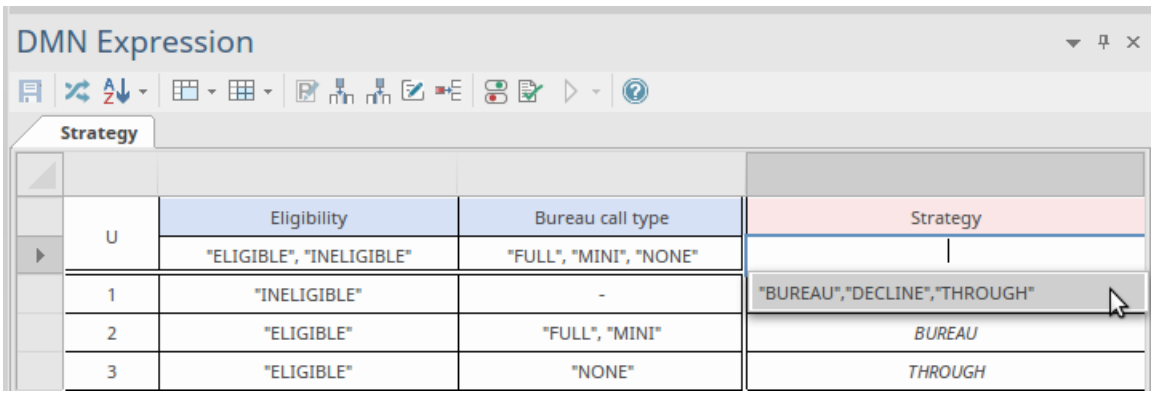


决策表的输入/输出条目

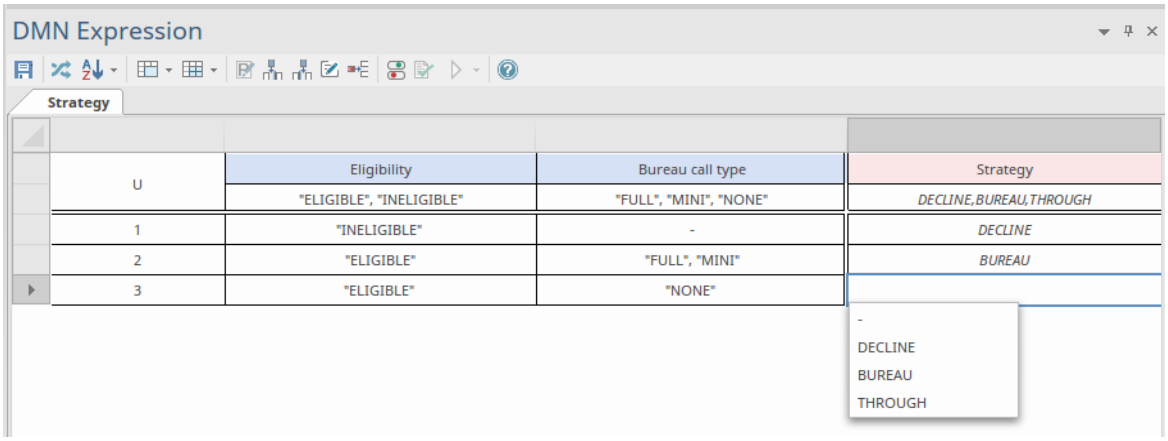
以 'Strategy' ItemDefinition 为例：



我们可以通过选择快速填写决策表的 允许值”字段：



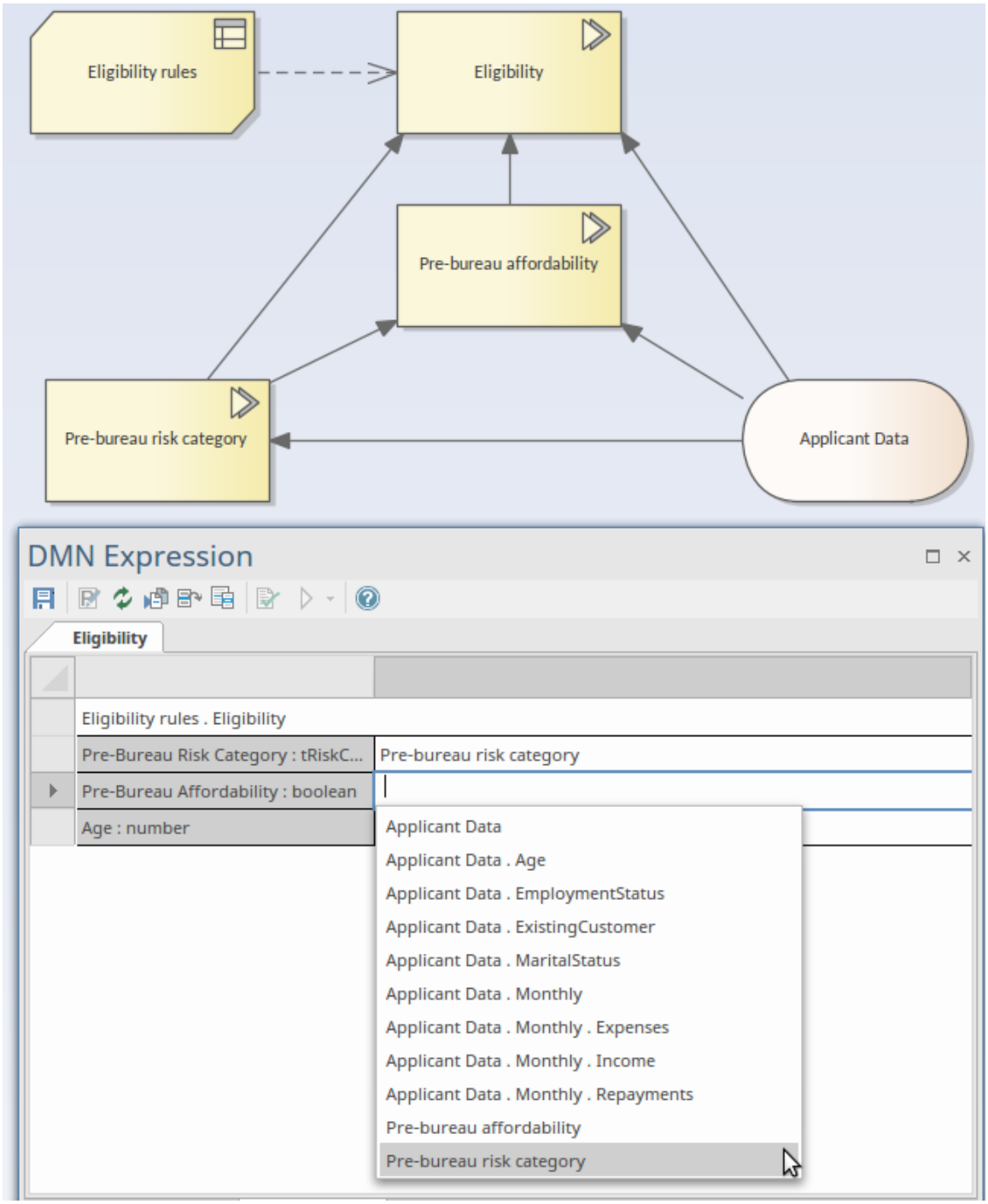
然后我们可以通过选择快速填写决策表规则：



注记：默认的 -”表示 未定义”。

信息需求

在决策层次结构中，决策可能会访问所需的决策和输入数据；这些必需的元素构成了可供决策使用的变量列表。

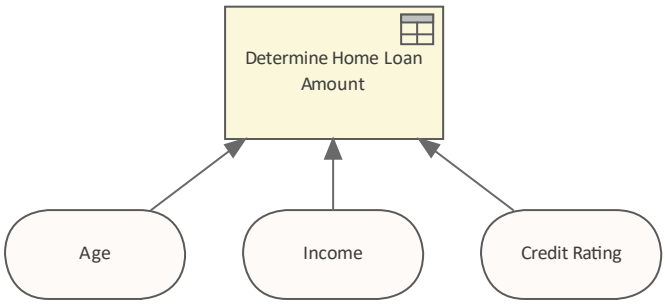


在这个例子中，决策 资格"需要两个决策—— 局前风险类别"和 局前负担能力"——以及一个输入数据项 申请人数据"。

为调用的 BusinessKnowledgeModel 'Eligibility rules' 设置绑定值时，会提示选择一个自动完成列表。在此列表中，有子决策名称 - 输入数据的叶组件。使用此特征，您可以轻松设置调用。

使用 DMN建模

本主题向您介绍创建决策模型所需的最重要元素。正如前面所讨论的，决策模型有两个基本部分，即决策需求图和决策逻辑。创建决策图是直截了当的，可能练习中最繁重的部分将解开组织做出决策的方式以及这些决策的输入是什么。图表通常包含链接在一起的决策，描述一个决策可以为另一个决策提供输入等事实。



决策的需求图显示了使用决策表的决策输入。

决策的逻辑使用许多设备来描述，但最常用和最容易理解的形式是决策决策表。决策表像电子表格一样包含行和列，并涵盖所有可能的输入组合以产生许多输出。例如，如果申请人年龄大于 21 岁且小于 65 岁，并且年收入 60,000 美元且信用评级良好，那么银行将借给他们 300,000 美元用于家贷款。

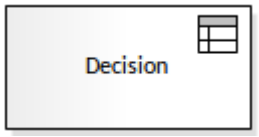
Determine Home Loan Amount

U	Age	Income	Credit Rating	Loan Amount
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-

显示三个输入和一个输出A决策表，将添加行来定义规则。

决策

决策元素用于基于A或多个输入来评估输出。确定输出的逻辑要么在该决策元素中定义，要么调用与决策决策相关的决策业务知识模型中包含的决策逻辑。



输入

决策可以A任意数量的输入，包括在元素中定义输入值的选项。最常见的输入是使用 Input Data元素。

输出

A决策可以有零个或一个输出。输出可以是复杂的数据集。

值表达式

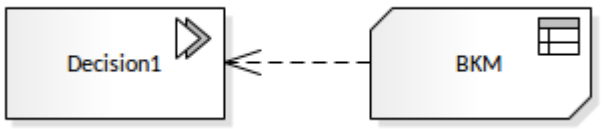
决策元素的输出是决策表达式确定的。值表达式包含元素的决策逻辑，可以采用以下四种形式之一：决策决策表、文字表达式、调用或盒装上下文。值表达式是使用DMN 表达式编辑器定义和编辑的，它根据所使用的表达式类型显示四种格式之一。

当显示在图表上时，决策元素会在右上角显示一个图标，指示它正在使用哪种类型的价值表达。

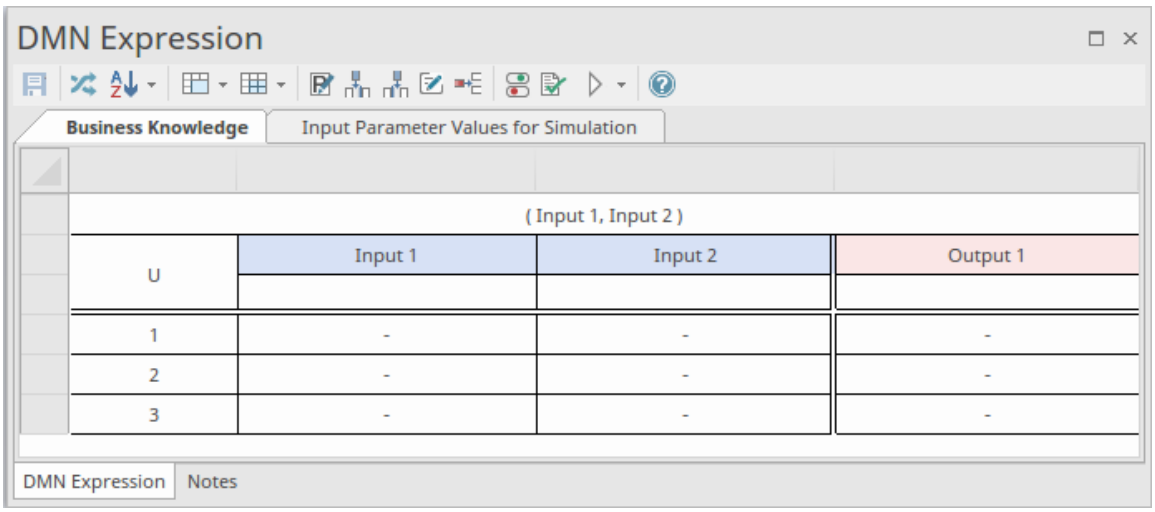
类型	描述
	决策表是A组相关输入和输出表达式的表格表示，组织成规则，指示哪个输出条目适用于一组特定的输入条目。
	文字表达式是 DMN 表达式A最简单形式。它通常定义为单行语句或 if-else 条件块。
	决策调用需要A知识需求连接器来引用业务知识模型元素。决策元素只包含为评估业务知识模型（上下文）提供时间的参数，或者所有从部件返回的结果都可以设置为作为决策的输出决策。
	盒装上下文是上下文条目A集合。每个上下文条目由一个变量和一个表达式组成。上下文也有一个结果值。

业务知识模型

A业务知识模型(BKM)元素代表了一个可重用的决策逻辑。通常，它连接到调用决策并传递一组输入的决策元素。 BKM 使用其内部逻辑评估传递回决策的输出。



除非 BKM 处理固定值，否则它通常需要定义一组输入参数以及输出的定义。使用DMN 表达式窗口定义参数和决策逻辑。



输入和输出

在决策模型中使用时，BKM 必须通过 KnowledgeRequirement 连接到决策或另一个决策，通过它接收其输入。输入参数使用图标定义。这些可以设置为使用 ItemDefinition 定义的简单类型或复杂类型。输入参数的命名会影响值表达式中的命名。

输出

BKM 输出是A KnowledgeRequirement 实现的，它必须是决策或另一个 BKM 的输入。输出定义使用：

- 文字表达式的图标
- 用于决策表、盒装内容和调用的DMN 表达式表中的输出列。

输出可以是使用 ItemDefinition 定义的简单类型或复杂类型。

值表达式

为了定义评估输出的方法，基于决策逻辑，业务知识模型（BKM）元素包含一个值表达式。这是使用DMN 表达式窗口定义和编辑的，它有四种格式，格式由您要使用的值表达式的类型确定。

BKM元素可以使用这些结构为值表达式设置。每个都在模型中显示，带有一个图标。

类型	描述
	决策表是A组相关输入和输出表达式的表格表示，组织成规则，指示哪个输出条目适用于一组特定的输入条目。
	文字表达式是 DMN 表达式A最简单形式。它通常定义为单行语句或 if-else 条件块。
	决策调用要求A知识需求连接器来引用业务模型元素。它仅包含为评估上下文模型提供时间的参数。
	盒装上下文是上下文条目A集合。每个上下文条目由一个变量和一个表达式组成。上下文也有一个结果值。

验证和测试

为确保 BKM元素能够产生正确的输出，可以使用验证图标  。 BKM 也可以作为A单元进行测试，以确保它可以使用仿真  按钮进行操作。有关更多详细信息，请参阅用于仿真帮助的仿真参数值主题。

BKM 参数

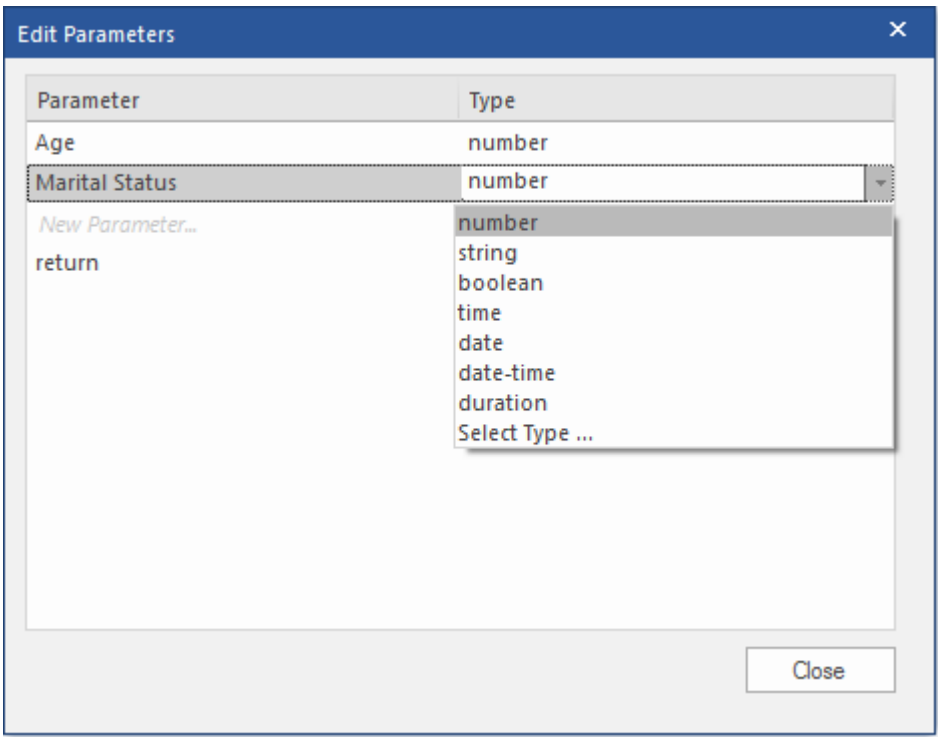
业务知识模型（BKM）以函数定义A形式实现，以参数和DMN表达式为主体（如决策表、盒装上下文或文字表达式）。

由于函数旨在独立运行，并由决策或其他 BKM 调用，因此有必要定义任何输入参数。此外，对于文字表达式，您必须定义输出参数。

定义任何输入参数时，您可以使用默认值设置它们以进行测试。创建 BKM 后，要验证它是否正常运行，您可以基于这些默认值运行模拟。

业务知识模型参数

要打开 编辑参数”对话框，请在DMN 表达式窗口中单击编辑参数按钮 ：



注记：这是包含返回类型的文字表达式的示例。

编辑参数

您可以对参数执行以下操作：

行动	描述
	通过在 新参数...”行中键入来添加新参数。
	通过在单元中就地编辑来修改现有参数的名称。
	使用上下文菜单删除现有参数。
	单击类型以启用下拉菜单。从下拉列表中选择参数的类型。

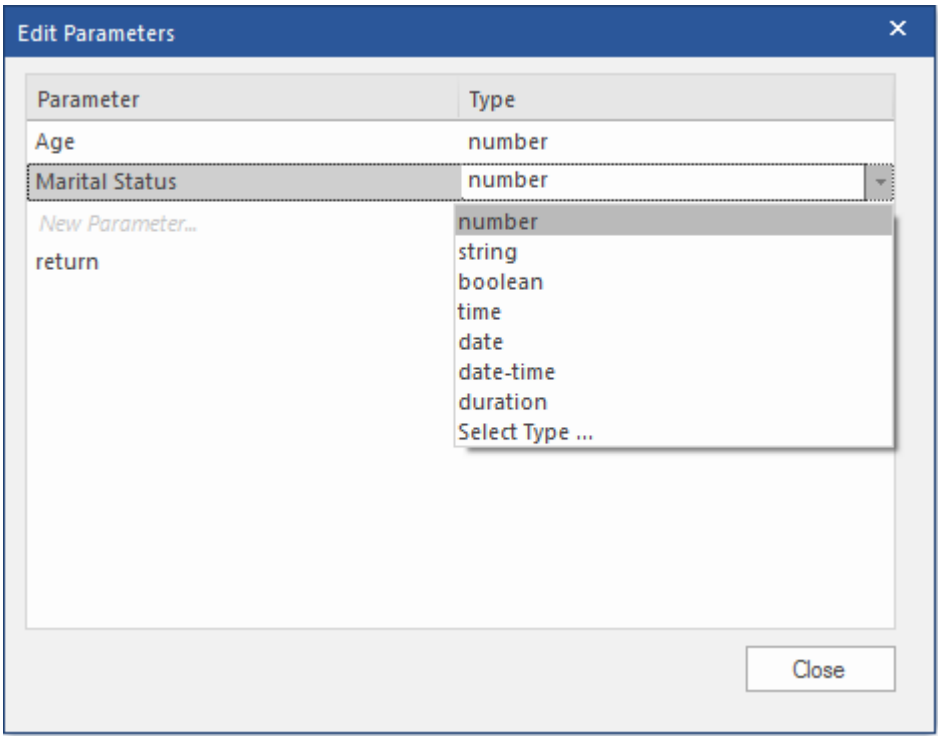
	设置项目定义类型 更改参数的类型时，可以选择从参数中选择预定义类型。选项是 选择类型 ...”。选择此选项 ，它将打开一个对话框以选择 ItemDefinition。

用于仿真的输入参数值

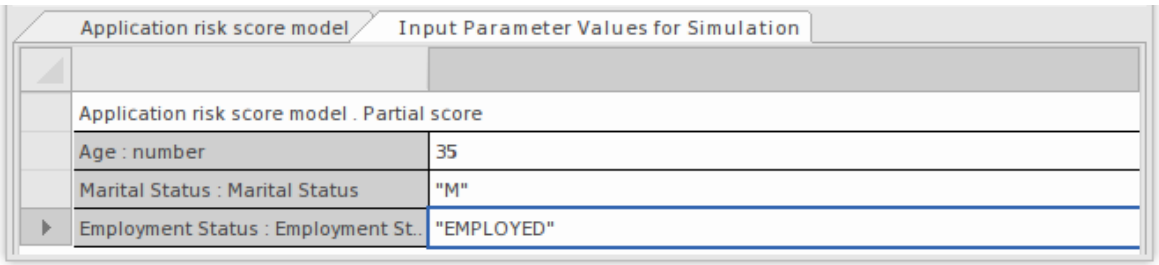
由于业务知识模型是独立的，因此可以通过提供一组默认值作为其参数的输入来执行模拟 单元测试”。这些值可以在DMN 表达式窗口的仿真输入参数值选项卡中定义。

业务知识模型参数(BKM)

使用工具栏上的编辑参数按钮从DMN 表达式窗口访问 BKM 的参数：



这些参数A默认值集（可用于 BKM 模拟）在 用于仿真的输入参数值”选项卡中定义 在DMN 表达式窗口上：



设置好这些参数后，BKM就可以使用仿真按钮进行测试。

仿真例子

这是使用输入参数值进行仿真的两个示例。

类型	描述
决策表	基于输入参数仿真选项卡中设置的值的 BKM决策表元素的示例模拟。

文字表达	基于 仿真仿真参数值" 选项卡中设置的值的 BKM 文字表达式元素的示例模拟。

决策表仿真示例

本节中描述的示例业务知识模型(BKM) 可从模型构建器 (Ctrl+Shift+M) 中获取。在模型中选择一个主机包，调用模型构建器，然后从蓝图下拉菜单中选择 需求|决策建模”。

要访问本节使用的示例：

- 为 DMN决策|”创建一个模 完全示例A
- 在浏览器窗口中导航至A完全示例|业务模型”

它也可以在Enterprise Architect示例模型(EAExample) 中找到：

- 在浏览器窗口中导航至 分析和业务建模> DMN Examples > A完全示例>业务模型”

双击 资格规则”元素以在DMN 表达式窗口中打开 BKM

当为业务知识模型创建决策表时，我们可以通过绑定一些值来测试这个BKM：

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*

我们可以提供如下测试值：

Eligibility rules | **Input Parameter Values for Simulation**

Eligibility rules . Eligibility	
Pre-Bureau Affordability	true
Pre-Bureau Risk Category	"VERY LOW"
Age	16

单击工具栏上的仿真按钮 即可获得此结果：

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*

- 运行时参数值将取代模拟模式下的 允许值”
- 有效规则已突出显示

- 因为此决策表命中策略是 P（优先级），所以最终结果由输出值的顺序决定；因为 'INELIGIBLE' 和 'ELIGIBLE' 是输出值并且 'INELIGIBLE' 位于 'ELIGIBLE' 之前，所以规则 #3 将给出最终结果并且该申请人为 'INELIGIBLE'。

文字表达仿真示例

本节中描述的业务知识模型(BKM) 可从模型构建器 (Ctrl+Shift+M) 中获取。在模型中选择一个主机包，调用模型构建器，然后从蓝图下拉菜单中选择 需求|决策建模”。

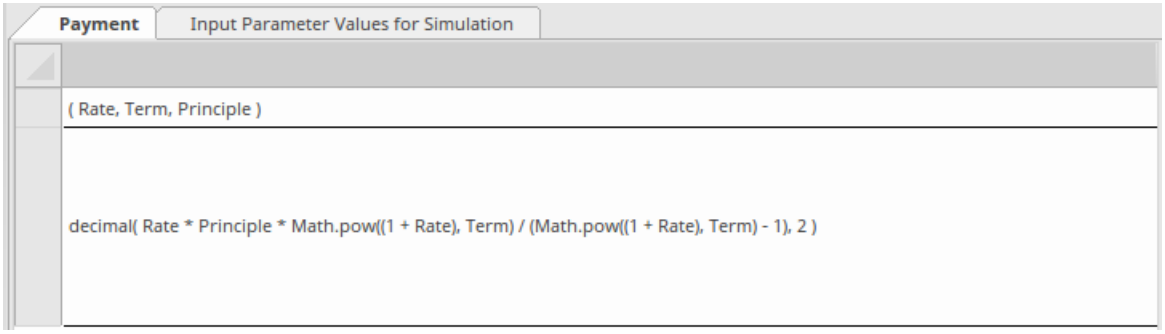
要访问本节使用的示例：

- 创建 DMN业务知识模型>业务知识模型字面表达”的模式
- 在浏览器窗口中导航至 业务知识模型文字 > 付款”

它也可以在Enterprise Architect示例模型(EAExample) 中找到：

- 在浏览器窗口中导航至 模型仿真> DMN模型>业务知识模型>业务知识模型Literal Expression”

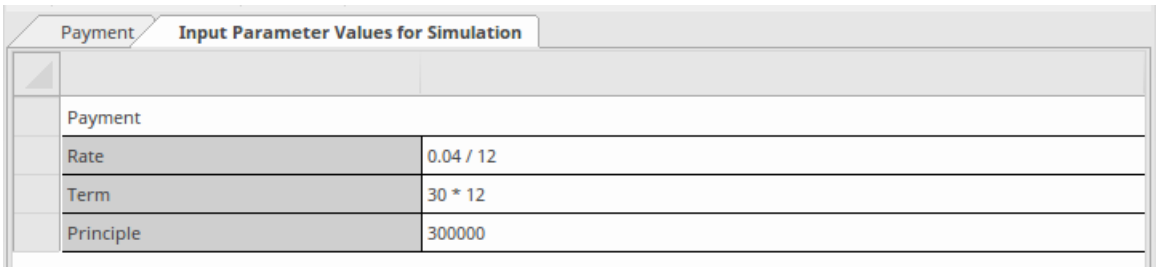
双击 Payment”元素以在DMN 表达式窗口中打开 BKM。



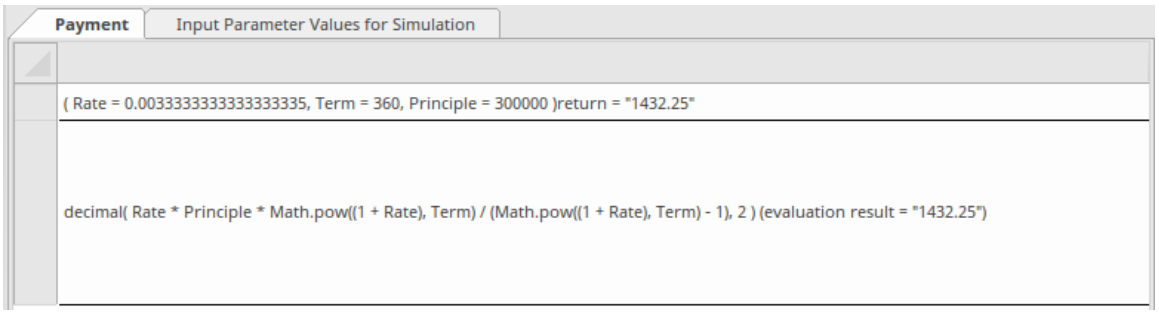
与决策表类似，以盒装表达式实现的业务知识模型也可以进行测试。

以 还款”元素为例，该BKM将根据利率、期限数和本金计算每月还款额。

我们可以提供如下测试值：



点击工具栏上 仿真，得到如下结果：



运行时参数和返回值将显示为等号 =”，后跟运行时值。此值还显示为其父图表上元素的标签。

在这个例子中，假设年利率为 4%，期限为 30 年，本金为 300,000 美元，则每月还款额为 1,432.25 美元。

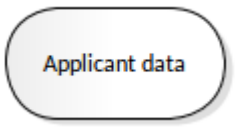
注记： DMN 库已经定义了 PMT函数；此示例主要演示了文字表达式的工作原理以及如何使用一组参数对其进行测试。

输入数据

InputData元素用于将源自模型之外的一组值输入到决策中。该组值用于评估决策。它从 ItemDefinition 派生其类型和一组值。

概述

InputData 元素是通过将 Input Data 图标从工具箱拖到 DMN 图上来创建的。



InputData元素的名称必须唯一，不能与其他决策模型、决策、业务知识模型、决策服务或导入决策模型中的名称重复。

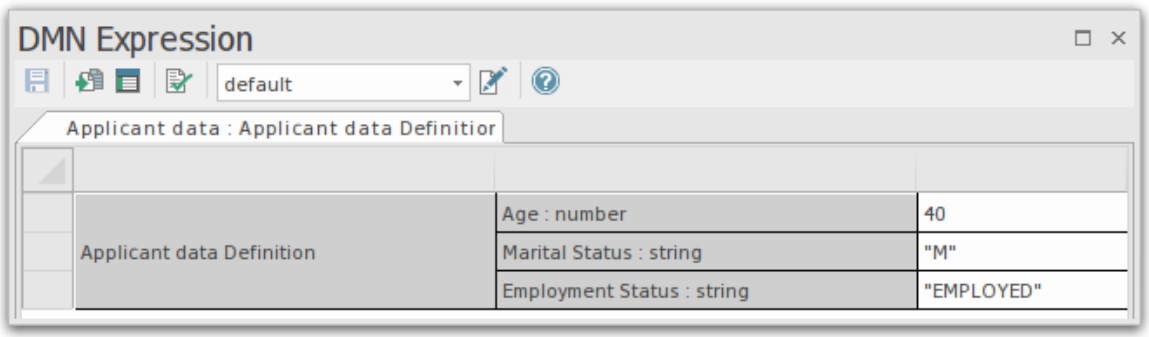
引用 ItemDefinition

数据的结构以及 InputData元素的值集在 ItemDefinition元素中定义。 A DMN InputData元素必须由 ItemDefinition 引用（键入）：

- 单击 InputData元素或DMN 表达式窗口上的图标
- 选择 InputData元素并按 Ctrl+L 从对话框中选择 ItemDefinition

InputData属性

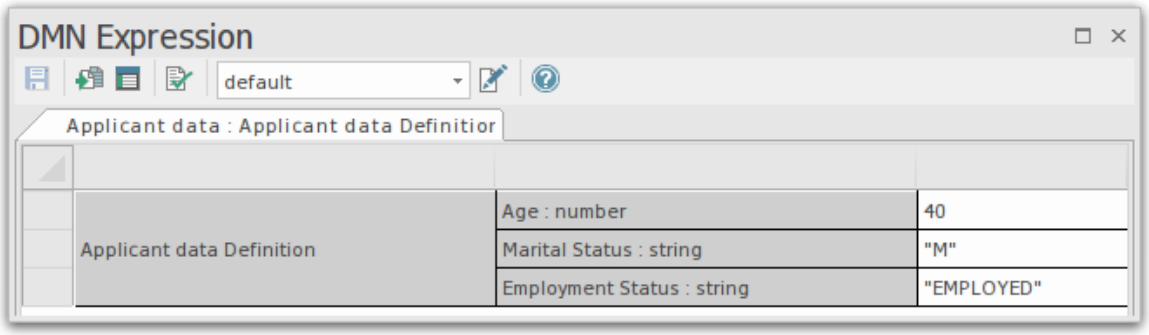
InputData元素的属性可通过DMN 表达式窗口访问。双击 InputData元素以打开此窗口。



DMN 表达式窗口提供了数据结构的视图以及对可在模拟中使用的数据集的访问。

InputData DMN 表达式

DMN 表达式窗口提供了 InputData 数据结构的视图、更改项值的选项以及对可在模拟中使用的数据集的访问。



访问

功能区	仿真>决策分析>DMN> DMN 表达式，然后选择/创建一个InputData
其它	双击一个 DMN InputData元素

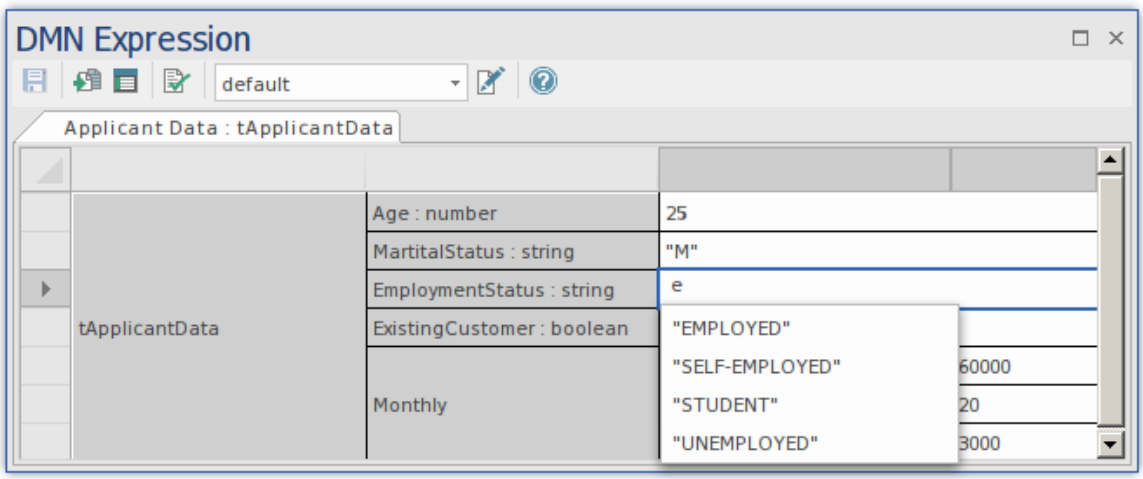
工具栏选项

选项	描述
	将配置保存到当前 InputData元素。
	通过选择对 ItemDefinition 的引用来设置 InputData 的类型。
	打开此 InputData 作为其类型定义引用的 ItemDefinition元素。
	运行 InputData 的验证。 Enterprise Architect将执行一系列验证以帮助您识别 InputData 中的错误。
	用于选择引用此 InputData 的 ItemDefinition 中定义的数据集的选项。
	打开用于编辑此输入数据的数据集的对话框。每个 InputData 可以定义多个数据集。有了这个特征，DMN仿真可以通过选择不同的数据集来快速测试决策的结果。

自动完成

如果 InputData 有一个定义了 允许值"的字段，则可以通过选择该字段，按空格键，然后从下拉列表中选择一个

选项来填充该字段。



数据集

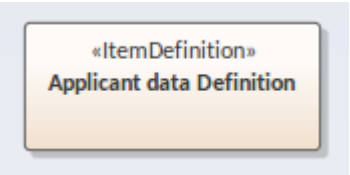
数据集在 InputData元素引用的 ItemDefinition 中定义。使用工具栏下拉菜单，您可以从 ItemDefinition 中选择一个数据集。选择一组后，您可以更改项目的值。您还可以通过使用  图标打开编辑数据集窗口来添加新数据集。

项目定义

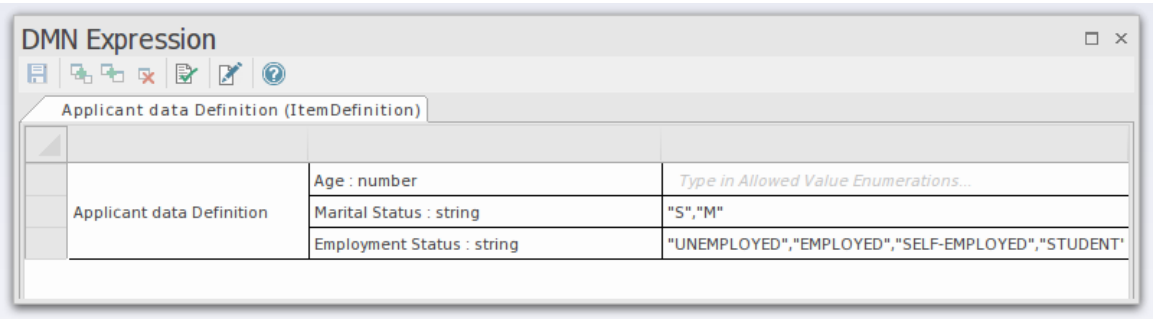
创建决策模型的基础是定义模型中使用的数据项的结构。 ItemDefinition 用于定义输入数据的结构，并可选择限制数据允许值的范围。 ItemDefinitions 的范围可以从简单的单一类型到复杂的结构化类型。

概述

ItemDefinition 元素是通过将 Item Definition 图标从 DMN工具箱页面拖到 DMN 图上来创建的。



通过DMN 表达式窗口访问属性元素的核心属性。



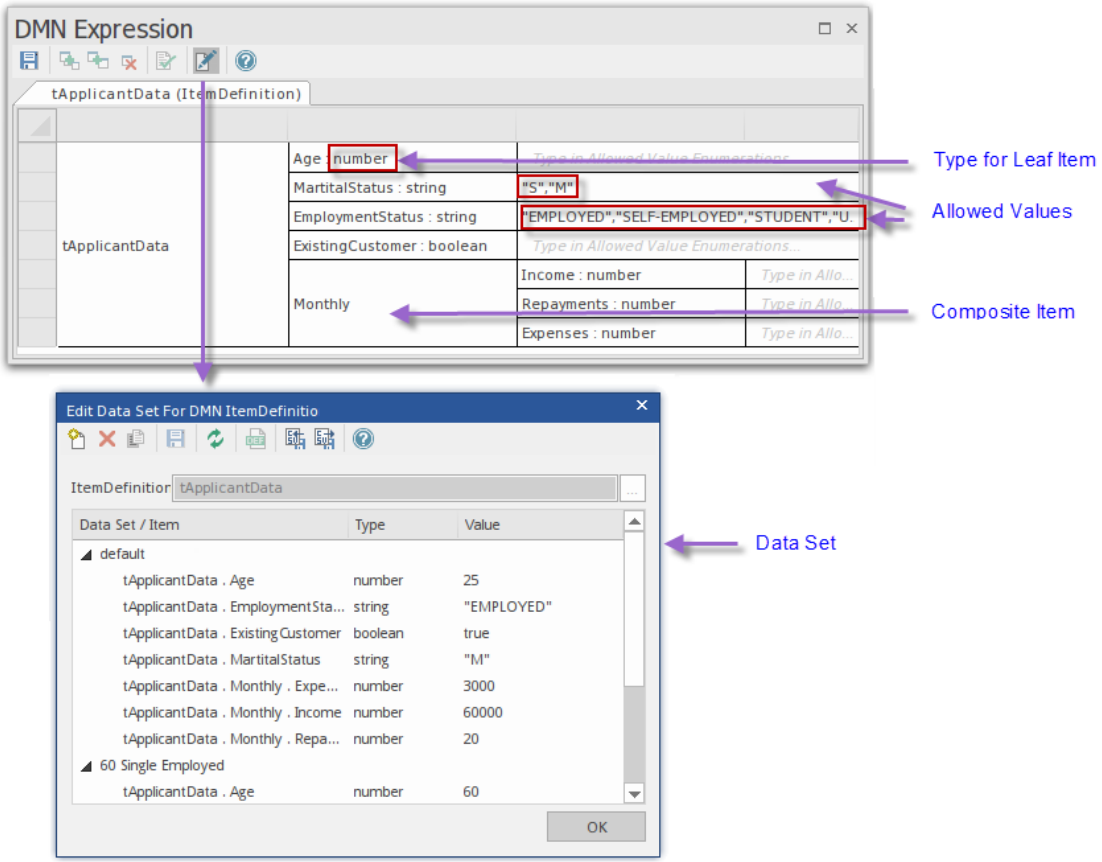
访问

打开 ItemDefinition元素的DMN 表达式窗口：

功能区	仿真>决策分析>DMN> DMN 表达式，然后选择或创建一个决策
其它	双击 DMN ItemDefinition

DMN 表达式和数据集

此图是DMN 表达式窗口的概述，显示了复杂的数据项以及数据定义中使用的关键字段的布局。包括使用此 ItemDefinition 定义的数据集的视图。数据集是符合 ItemDefinition A数据 实例”，其中包含要在 DMN 模拟中使用的一组值。



由于项目定义是模型中的基础元素，因此建议在继续在模型中使用它们之前对其进行验证。这将确保在创建复杂模型的过程中及早解决任何问题。

有关设置帮助的更多详细信息，请参阅这些帮助主题：

- *DMN* 项目定义、数据集和输入数据
- 部件
- *ItemDefinition* 允许值
- *DMN* 表达式自动完成
- *DMN* 表达式验证

项目定义工具栏

此表提供了在选择 ItemDefinition元素时在DMN 表达式窗口中可访问的特征的描述。

工具栏选项

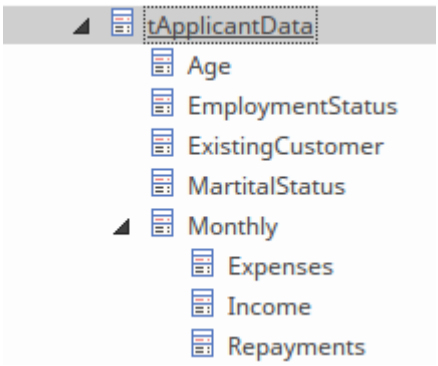
选项	描述
	保存当前 ItemDefinition 的配置。
	创建一个新的数据组件作为选定组件的子组件。
	创建一个新的数据组件作为选定组件的兄弟。
	删除选定的数据组件。
	验证 ItemDefinition； Enterprise Architect将执行一系列验证以帮助您识别 ItemDefinition 中的任何错误。
	打开 编辑数据集”对话框，您可以在其中创建和编辑 ItemDefinition 的 实例”以供 InputData 元素使用。

项目定义和数据集

决策描述了决策模型中使用的数据项的类型和结构。它作为决策元素、决策元素和业务知识模型参数的数据类型定义。 ItemDefinition 还可以定义数据集，这些数据集提供用于 DMN 模拟的值集。在不同数据集之间切换提供了使用决策模型进行 假设"分析的能力。

项目定义结构

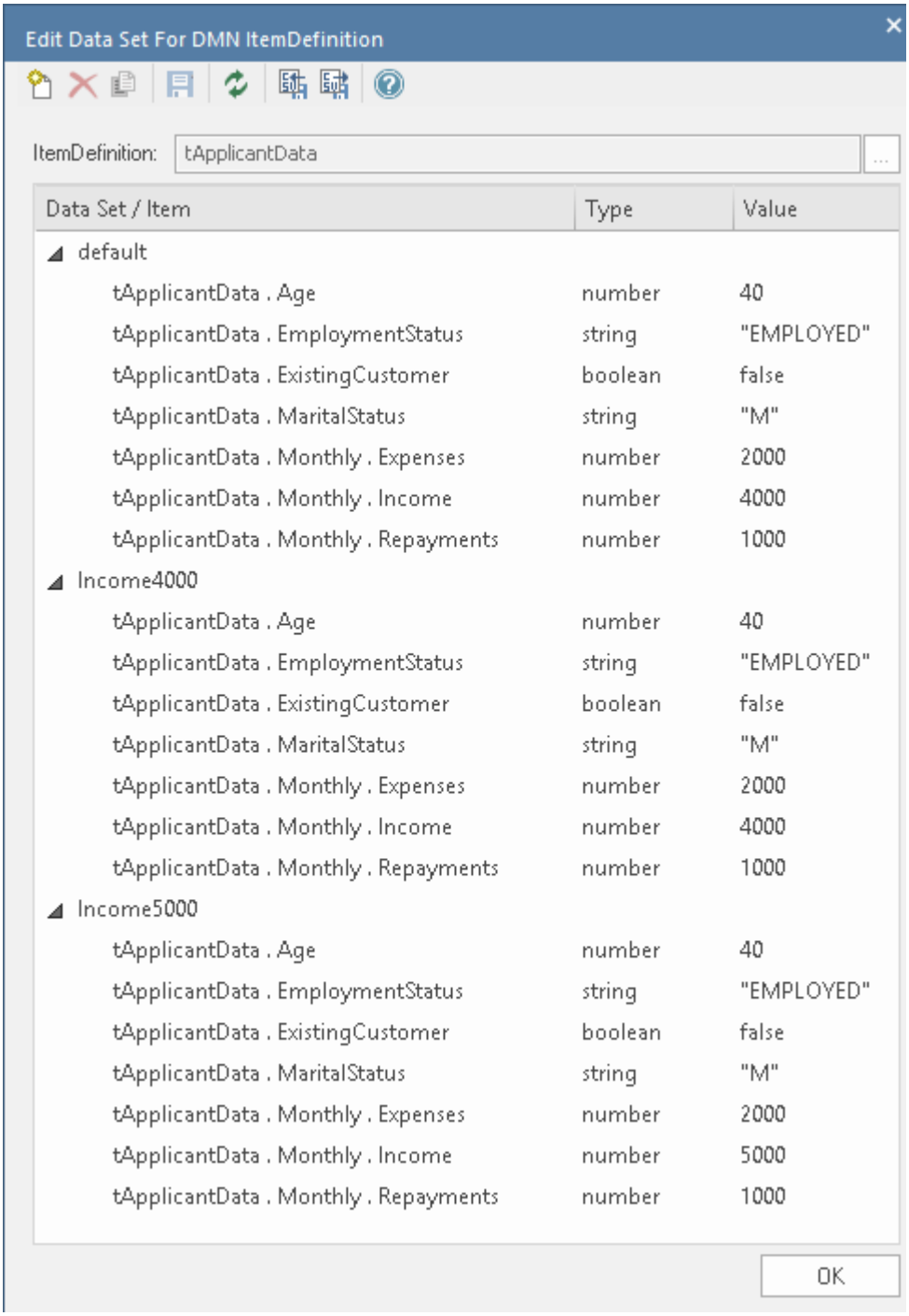
复杂A ItemDefinition 由嵌套元素组成。例如， *tApplicantData*的结构如下：



tApplicantData ItemDefinition 示例是五个子项的复合类型。 每月"由三个子项（费用、收入和还款）组成。叶组件（非复合）将具有原始类型，例如数字、 string或布尔值。

数据集

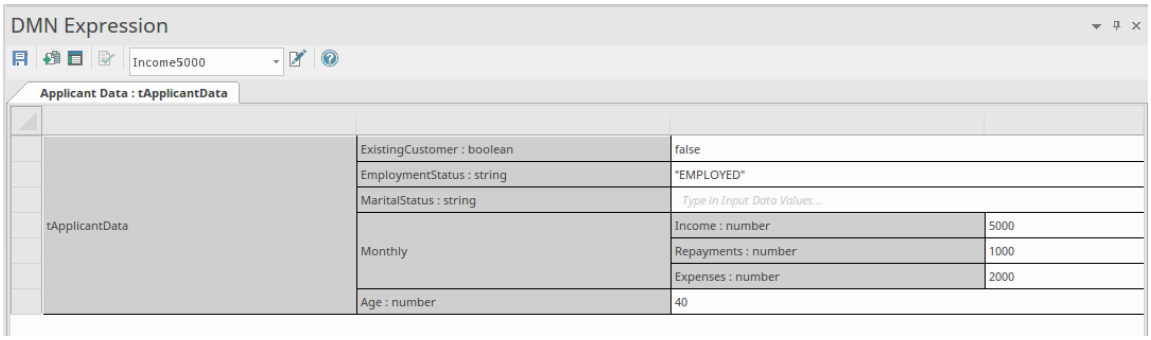
可以使用工具栏上的  图标查看和编辑 ItemDefinition 的数据集。通过 编辑数据集"对话框，您可以添加、删除和复制数据集。还支持数据集的 CSV 导入和导出。



如示例所示，tApplicantData 的*ItemDefinition*定义了三个数据集：

- 默认
- 收入4000
- 收入5000

每个数据集都可以在输入到 ItemDefinition 的 InputData元素中查看。例如，'Applicant Data' InputData元素被键入到'tApplicantData' ItemDefinition。此处所示的 申请者数据"的DMN 表达式窗口根据窗口工具栏下拉列表中选择的数据集（本例中为*Income5000* ）显示数据值。



将引用设置为参考

DMN InputData元素设置为由 ItemDefinition 引用（键入），使用以下任A ：

- InputData元素或DMN 表达式窗口上的 图标
- 选择 InputData元素并按 Ctrl+L 从对话框中选择 ItemDefinition

还有其他使用 ItemDefinitions 的情况；例如，在为 BKM 中的输入参数或决策表中的输出参数设置类型时。

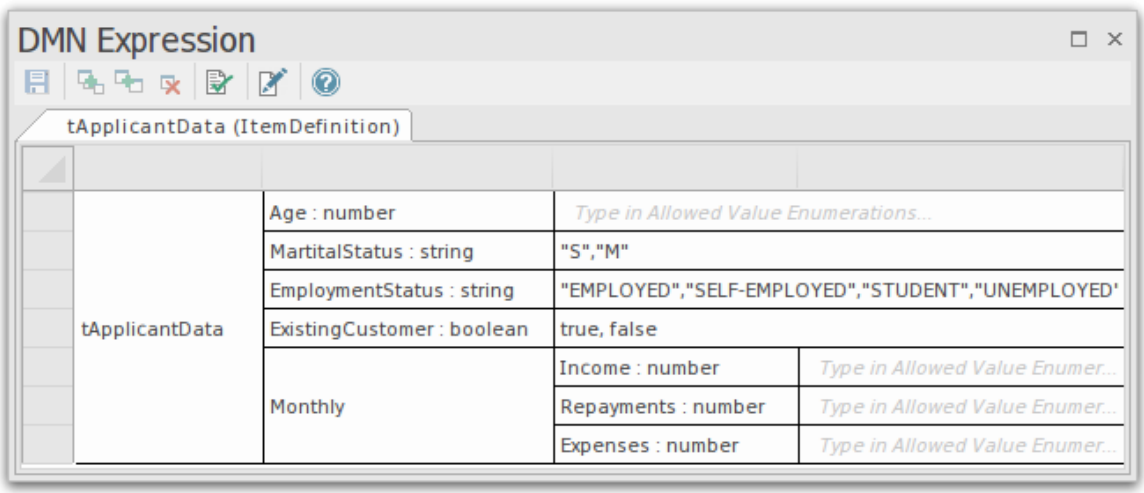
部件

一个 ItemDefinition元素可以定义为一个组件树，它只包含以下之一：

- A类型或
- A组组合元素

在这个组件树中，如果组件是没有子组件的 叶子”，则必须将其设置为内置类型。如果 ItemDefinition 具有子组件，则将这些子/叶组件设置为内置类型。

例如， *Applicant Data*和*Monthly*是组合，而*Age*和*Expenses*是设置为内置类型的叶子：



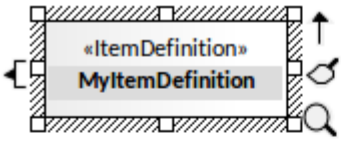
FEEL 语言具有以下内置类型：

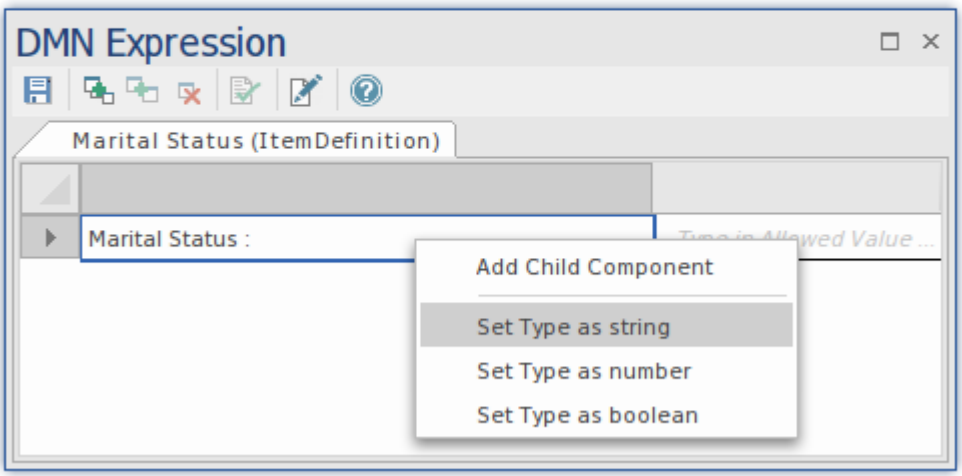
- **数字**
- **string**
- **布尔值**
- 天数和持续时间
- 年和月持续时间
- 时间
- 日期和时间

注记： Enterprise Architect支持 'number'、' string ' 和 'boolean' 进行模拟。

要为 叶子”ItemDefinition 设置类型，您可以使用以下三种方法之一：

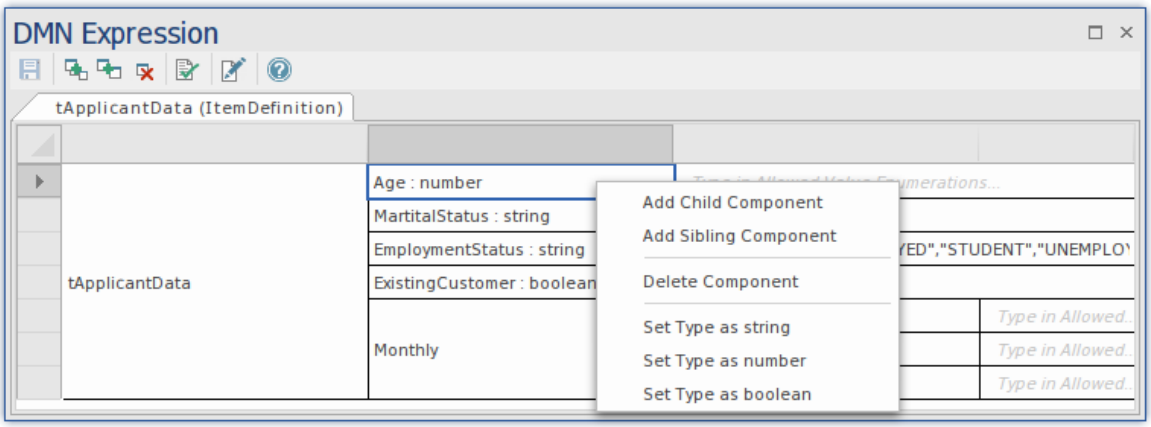
- 在DMN 表达式窗口中选择合适的上下文菜单选项（推荐）





- 类型': string ', ':boolean' or ':number' 在单元DMN 表达式窗口中的名称后面
- 类型' string '、'boolean'或'number'作为类型窗口中属性'Type'的值

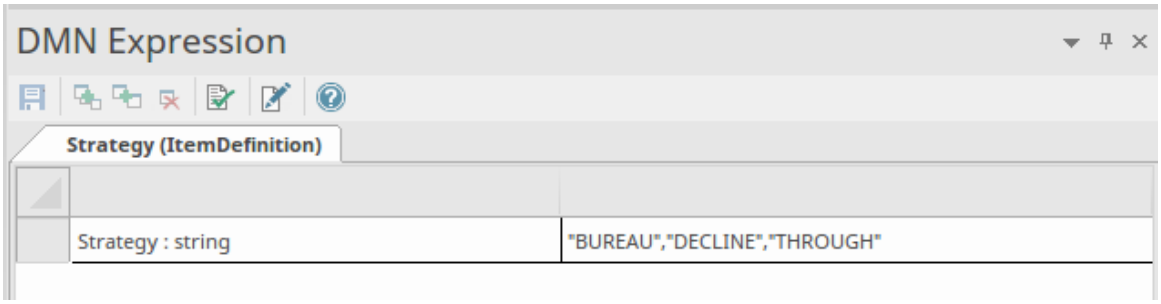
对于复合上下文，时间菜单还提供了创建子组件或同级组件或删除所选项目的选项：



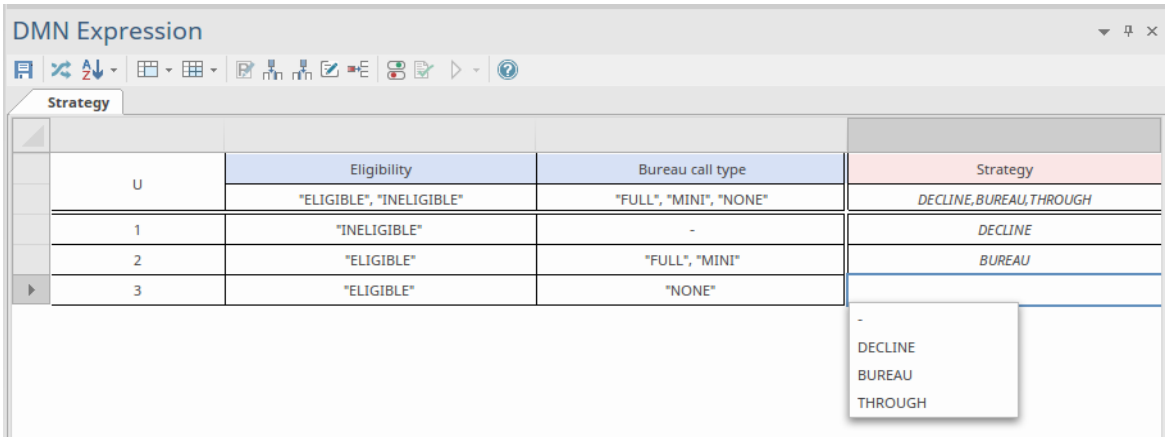
允许的值枚举

在为决策定义数据输入时，通常希望限制输入的允许值集。例如，您可能希望将婚姻状态的允许值限制为两个选项， 单身"和 已婚"。

您可以为 ItemDefinition 的任何叶组件指定允许的值。最初，叶组件的数据字段包含*Allowed Value Enumerations*中的文本类型。您只需使用允许的值输入此文本。例如，ItemDefinition *Strategy*具有三个允许值 - BUREAU、DECLINE 和 THROUGH。



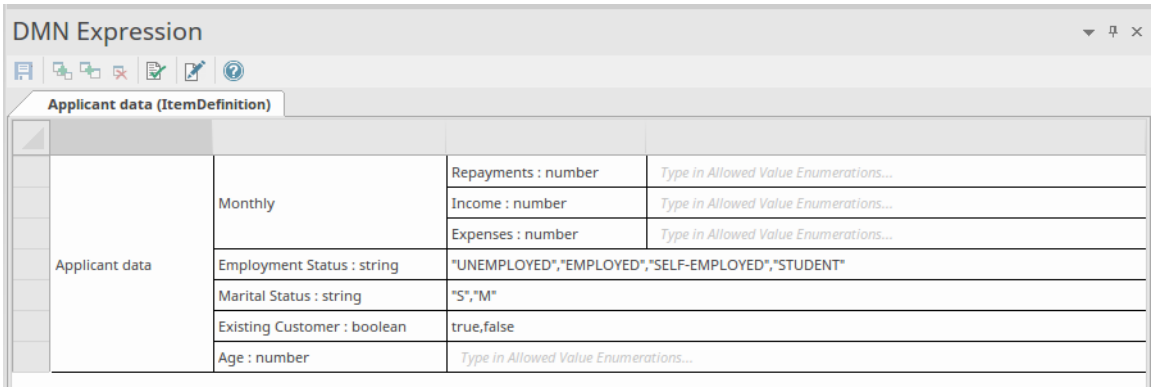
允许值枚举也用于支持自动完成。当为 InputData元素或引用已定义允许值的 ItemDefinition 的输入参数指定值时，用户只需按空格键并从列表中选择一个值。



您还可以通过键入要输入的选项的第一个字母来自动完成。

决策表的输入参数和输出子句也支持允许值的规范。这限制了在表中定义规则时可以使用的值，但也允许用户通过按空格键然后选择所需项目来快速填充规则。

A复杂的 ItemDefinition 可以包含许多 Allowed Value Enumerations；例如：



数据集

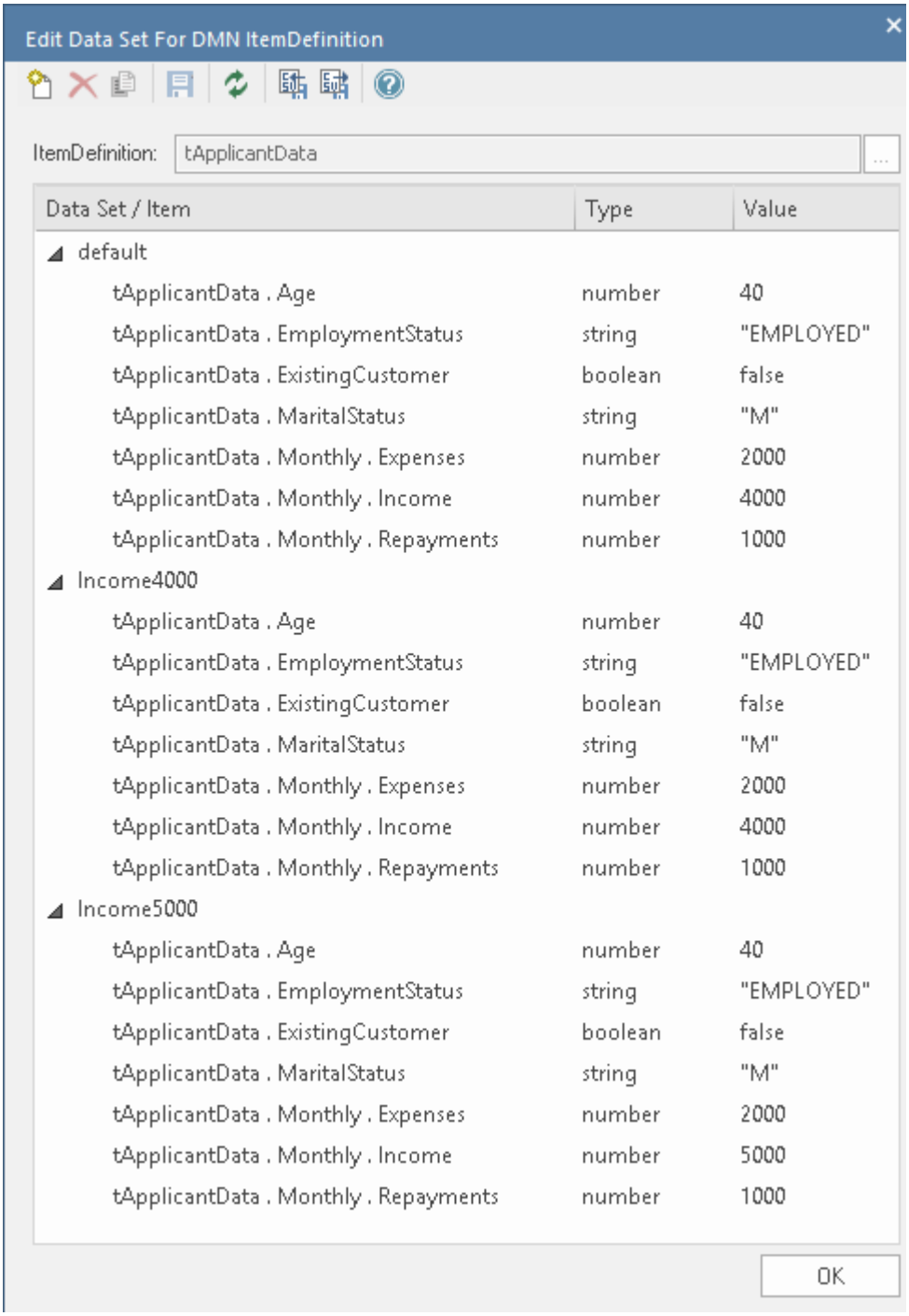
由 ItemDefinition 类型化的每个 InputData元素都有一组组件，并且可以定义多个数据集来为这些组件提供不同的值集。有了这个特征，执行 DMN仿真的用户可以通过选择不同的数据集来快速测试决策的结果。数据集与 ItemDefinition 相关联并基于 ItemDefinition，但您也可以通过 InputData元素处理它们。

您可以使用 编辑数据集"对话框添加或更新数据集，您可以从DMN 表达式窗口为 ItemDefinition 或 InputData元素调用该对话框。最初， 编辑数据集"对话框显示一组没有值的组件，其名称为 默认"。您可以不设置任何值，也可以提供值；无论哪种方式，您都可以将其用作模板来复制新数据集。您不能删除 默认"数据集。

当您在DMN 表达式窗口中访问 InputData元素时， 默认"数据集中的值将根据元素的组件显示。然 ，您可以单击工具栏中的下拉箭头并从列表中选择任何其他数据集。请注记，如果您保持 默认"数据集不变，您可以创建一个重复的 默认"数据集并分配值到它，并且 默认"设置将在您最初访问 InputData元素时提供值。

您可以复制和删除您创建的任何其他数据集，将数据集导出到 CSV 文件并从 CSV 文件中导入它们。

注记，如果您创建一个数据集但不输入值，则在您关闭对话框时它会被丢弃。



访问

功能区	仿真>决策分析>DMN> DMN 表达式>点击InputData项： 图标
其它	在图表中，双击 DMN InputData元素： 图标。

工具栏选项

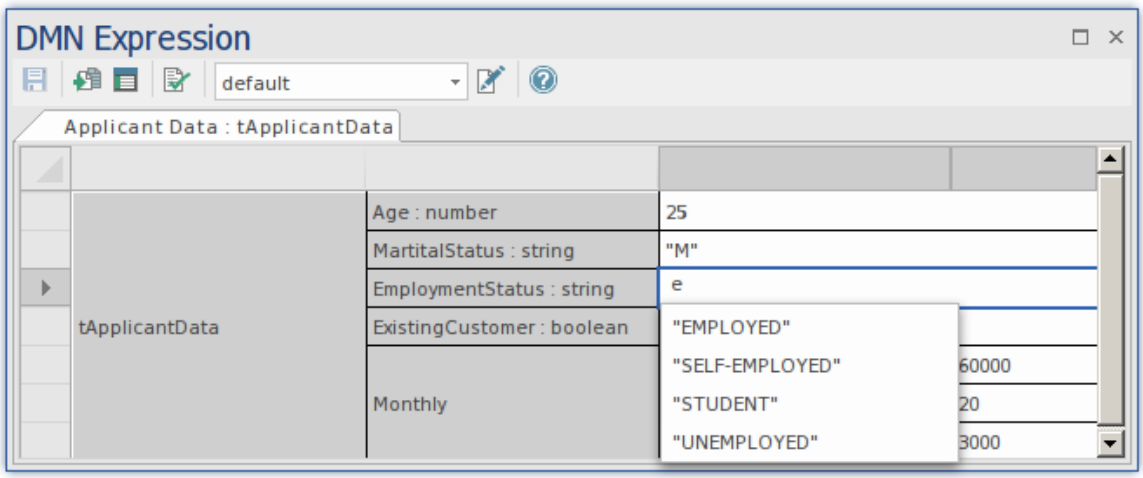
选项	描述
	单击此按钮可创建新数据集。
	单击此按钮可删除选定的数据集。
	单击此按钮可复制选定的数据集。
	单击此按钮可将数据集保存到 InputData。
	单击此按钮可重新加载 InputData 的数据集。
	单击此按钮可从 CSV 文件导入数据集。
	单击此按钮可将数据集导出到 CSV 文件。

使用 DataObjects 交换数据集

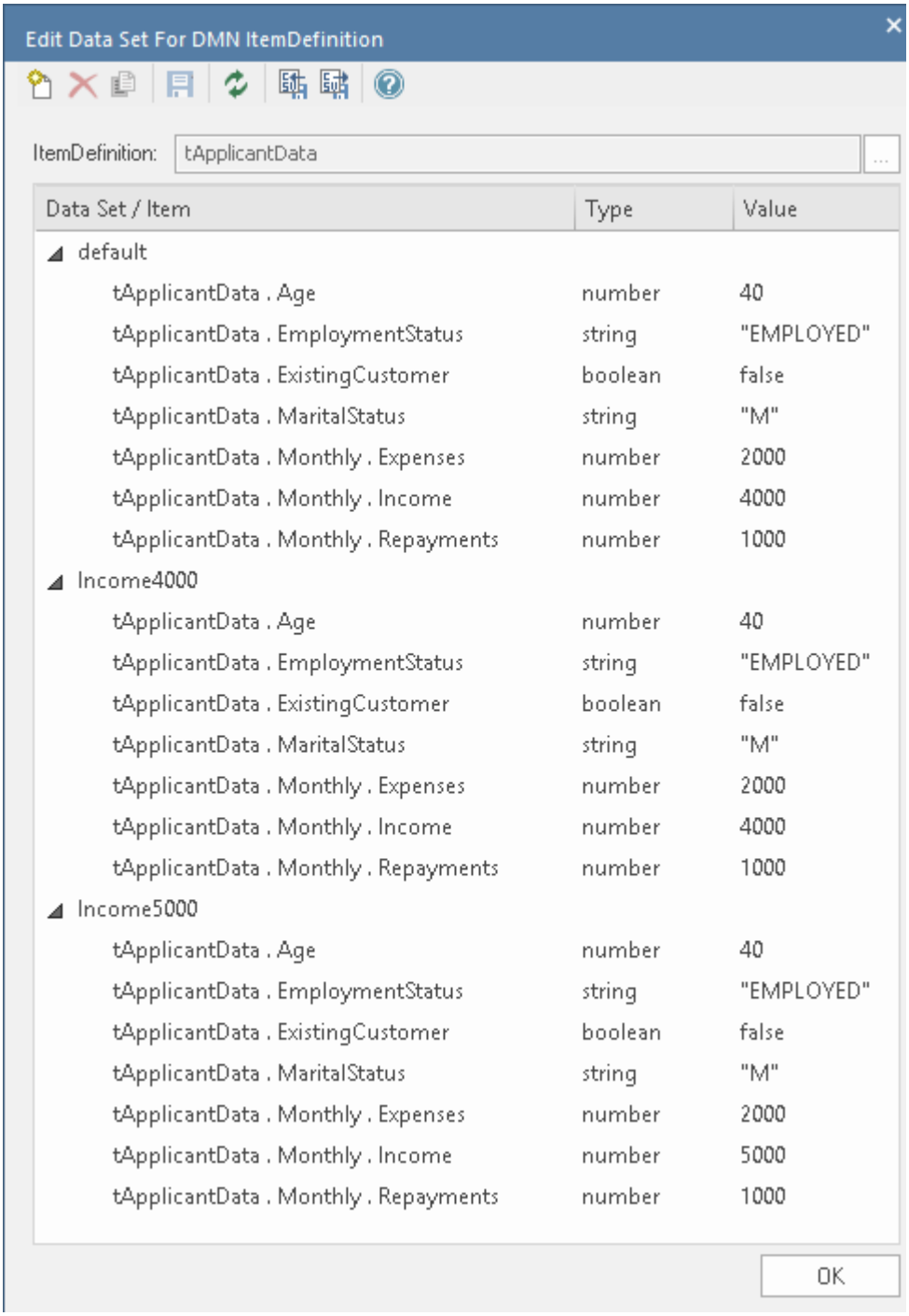
在测试由DMN模型生成的代码时，或者在模拟调用DMN模型的业务流程模型和表示法（BPMN）模型时，您需要一种交换数据集的方法。例如，在 DMN模型的 BPMN 调用中，BPMN DataObject 用于存储将传递给它正在调用的 DMN模型的变量集。此 DataObject 需要填充适合 DMN InputData 数据结构的数据，以准备传递给该 InputData object 。在测试从 DMN模型生成的代码时，使用相同的 BPMN DataObject。

本主题描述从 DMN 数据集创建 BPMN 数据对象的过程。

数据集存储A DMN InputData元素中，可以使用DMN 表达式窗口上的  图标进行访问。



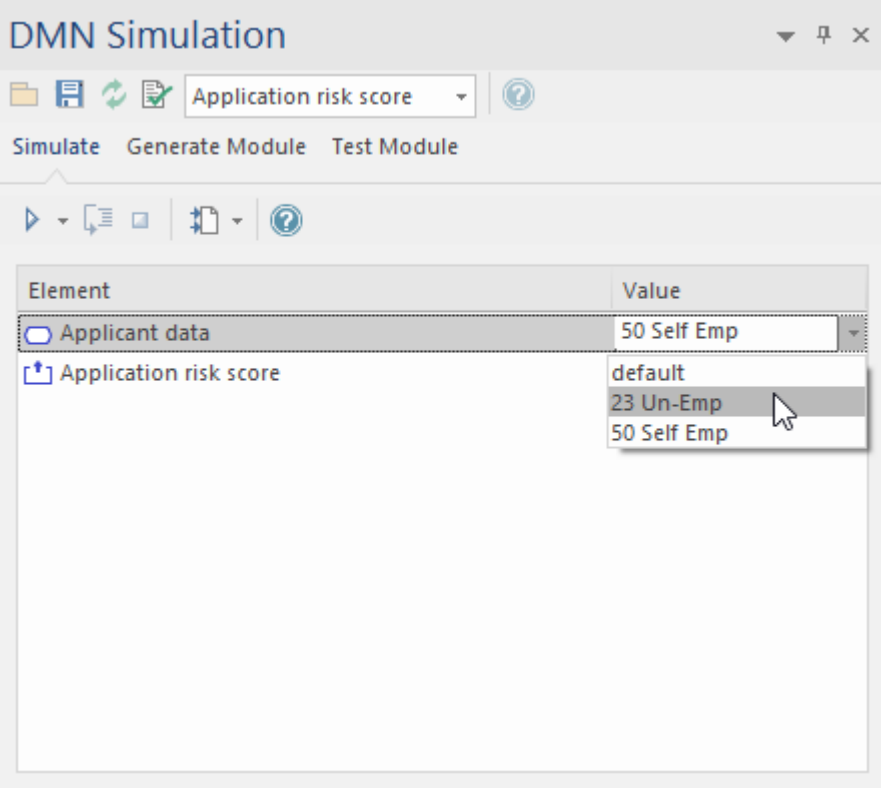
这将打开 InputData 的 “编辑数据集”对话框，其中可以包含多组值：



将数据集传输到数据对象有两个选项：

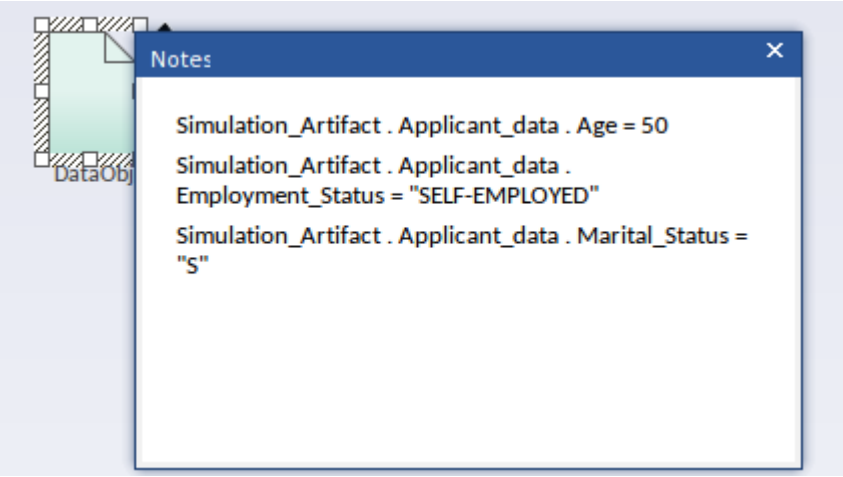
1 .直接的

- 在浏览器窗口的一个包下创建一个BPMN DataObject。
- 打开DMN仿真窗口



- 从 值”下拉列表中选择数据集
- 单击 DMN仿真窗口上的图标；这将打开 选择元素”对话框
- 选择 BPMN DataObject元素
- 点击确定按钮

现在可以在 DataObject 的注记中查看数据集。

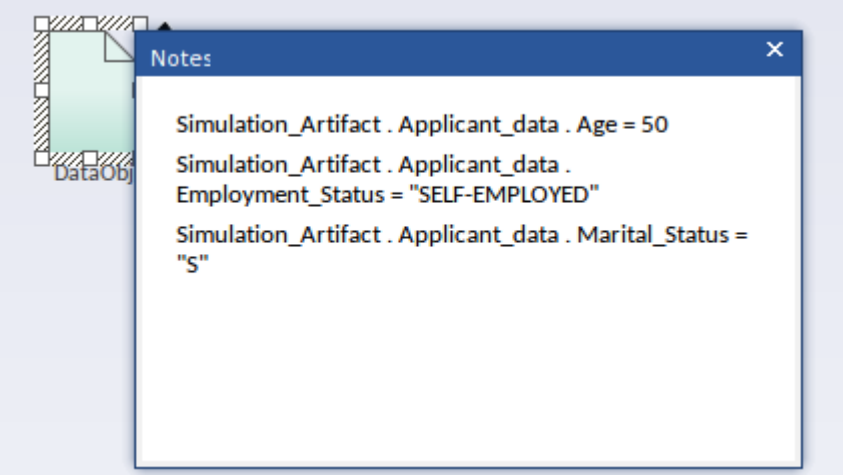


2. 手册

要手动交换此数据集：

- 打开 InputData元素的DMN 表达式窗口
- 单击编辑数据集图标；这将打开 编辑数据集”对话框
- 使用 CSV导出的使用图标将这些详细信息到文件

CSV 文件中的文本可以作为文本添加到 BPMN DataObject元素的注记中。


DataObj
Notes
Simulation_Artifact . Applicant_data . Age = 50
Simulation_Artifact . Applicant_data . Employment_Status = "SELF-EMPLOYED"
Simulation_Artifact . Applicant_data . Marital_Status = "S"

服务决策服务

本主题的部分内容已逐字使用或自由改编自 *DMN* 规范，该规范可在以下网址获得： *https://www.omg.org/spec/DMN* 。该站点包含对 *DMN* 及其功能的完整描述。

决策决策A决策模型中的一个或多个决策公开为可重用元素，该元素可能由决策模型决策模型任务在外部调用。

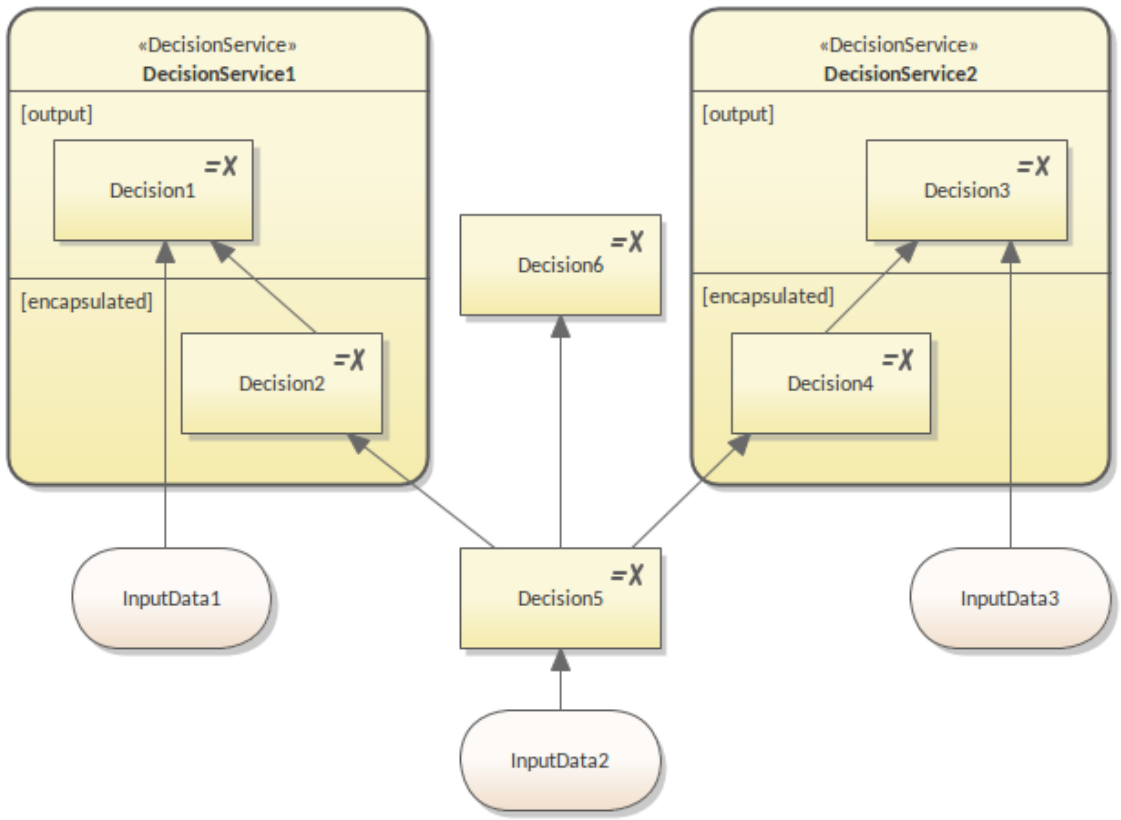
当使用必要的输入数据和输入决策调用决策服务时，它会返回公开决策的输出。

接口服务的决策

决策服务的接口包括：

- 输入数据 - 封装决策所需的所有输入数据的实例
- 输入决策 - 所有输入决策结果的实例
- 输出决策 - 使用提供的输入决策和输入数据评估（至少）所有输出决策的结果

当使用必要的输入数据和输入决策调用决策服务时，它会返回公开决策的输出。



该图显示了一个决策模型，包括六个决策和三个输入数据项。

对于DecisionService1，：

- 输出决定是{Decision1}
- 输入决策是 {Decision5}，并且
- 输入数据是 {InputData1}

由于Decision1需要Decision2，而Decision2并不作为输入提供给服务，因此服务也必须封装Decision2；因此封装的决策是{Decision1，Decision2}。

从图中可以明显看出，DecisionService1 的任何决策都不需要 Decision6、Decision3、Decision4 和 InputData3。

InputData2 呢？虽然Decision5 需要，DecisionService1 需要InputData2，但DecisionService1 实际上不需要InputData2。这是因为 Decision5 被定义为输入决策。从决策服务的角度来看，我们忽略了输入决策所需的任何决策或输入决策。

对于DecisionService2，：

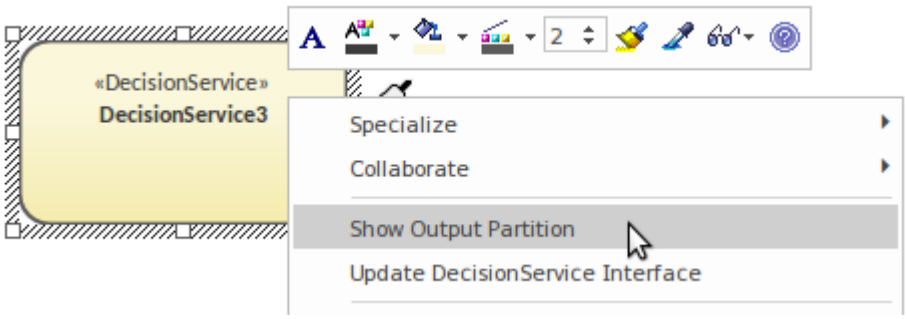
- 输出决定是{Decision3}
- 输入决策是 {Decision5}，并且
- 输入数据是 {InputData3}

由于Decision3需要Decision4，而Decision4并不作为输入提供给服务，因此服务也必须封装Decision4；因此封装的决策是{Decision3, Decision4}。

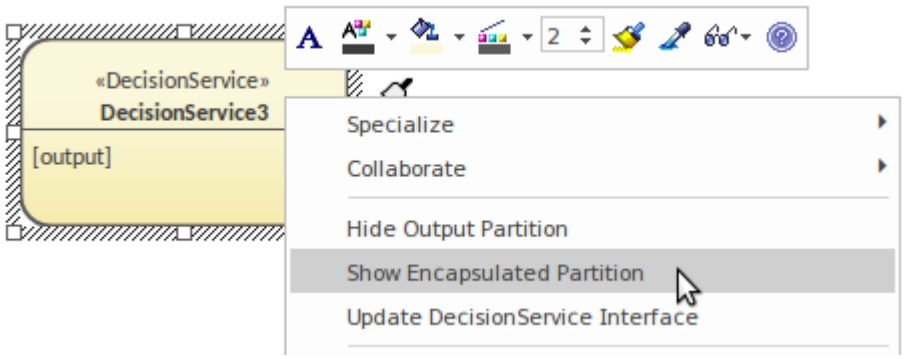
为每个决策服务创建单独的图表是一种很好的做法。这样，图将只包含决策的接口元素和封装的决策；不相关的元素不会出现在图表上。

建模决策服务

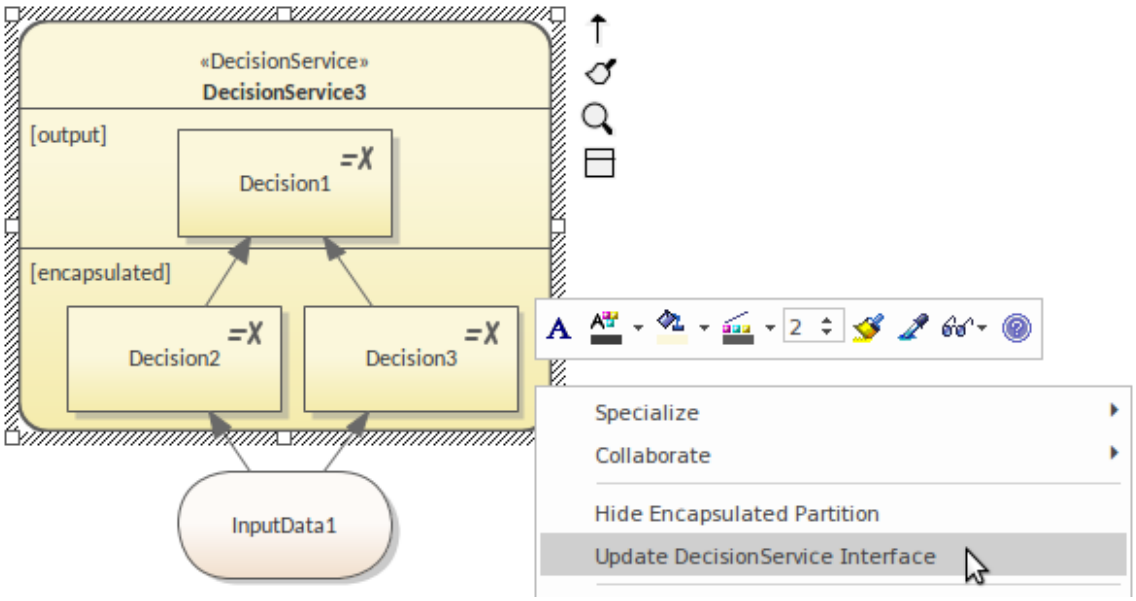
我们可以从图表工具箱的决策页面创建决策服务元素，并从上下文菜单切换 [输出] 和 [封装] 分区。



当显示 [output] 分区时，您只能显示 [encapsulated] 分区。



将决策和输入数据放入正确的分区后，您必须运行上下文菜单中的 更新决策服务接口”命令来更新模型。



重要提示：为了使 DMN 模拟正常工作，请在以下情况下更新决策服务接口：

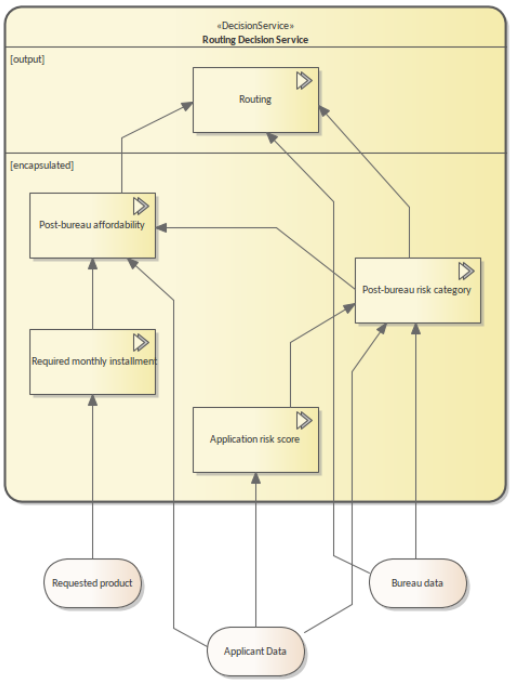
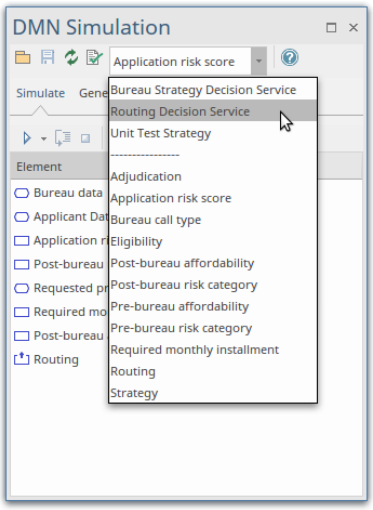
- 显示/隐藏决策服务分区
- 向决策服务添加决策
- 从决策服务中删除决策
- 在分区之间移动决策
- 添加/删除决策服务输入：输入数据或输入决策

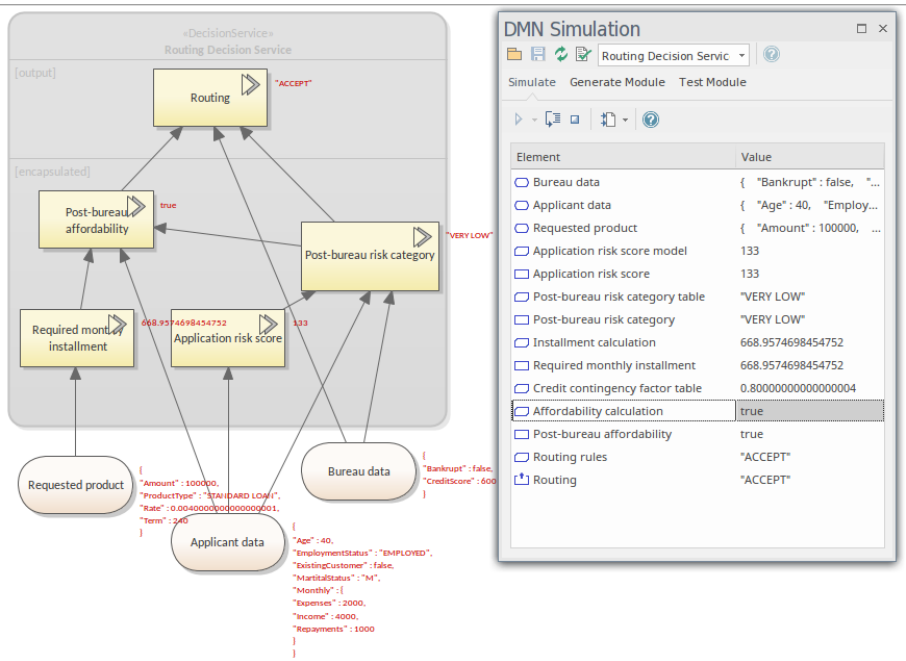
仿真决策服务

可以对决策服务执行模型模拟。

决策仿真

要对决策服务执行模型模拟，请执行以下步骤：

节	描述
1	将 仿真工具组件"页面上的 工具箱组件"元素工具的 配置组件"图表中，在工件仿真窗口中双击打开。   默认情况下，所有决策服务元素和每个单一决策都会在对话框工具栏的下拉字段中列出以供选择。
2	选择运行模拟的决策服务元素。在示例中，我们选择了 路由决策服务"，因此在模拟列表中加载了三个输入数据项和五个封装决策（包括一个输出决策）。 重要：此列表来自决策服务的内部数据；每当决策服务模型图发生更改时，请确保运行上下文菜单中的 更新决策服务接口"命令。通过单击 DMN仿真窗口工具栏上的 刷新"图标（左起第三个）重新加载决策模型。
3	输入数据和决策的执行顺序正确。例如， 应用风险评分"将在'Post-bureau risk category' 、 'Post bureau affordability'和'Routing'之前执行。对于每个输入数据元素，单击 值"字段中的下拉箭头并选择数据集以提供输入数据值。 验证输入数据和决策，并使用DMN 表达式窗口进行任何必要的更正。 在 DMN仿真窗口上，单击保存图标和工具栏上的 按钮。
4	运行时执行结果显示在列表和图表中。您也可以单击工具栏上的 "Step-through"图标来调试 DMN模型。



A好的做法是在调试时保持DMN 表达式窗口打开。运行（如决策表、盒装上下文、状态表达式或调用）的运行时间将显示决策决策封装或调用的决策业务知识模型的逻辑细节。

代码生成和测试模块

创建并模拟决策模型后，您可以生成Java 、 JavaScript 、C++ 或 C# 的 DMN 模块。该 DMN 模块可以与 Enterprise Architect BPSim 执行引擎、可执行状态机或您自己的项目一起使用。

Enterprise Architect还提供了一个 测试模块”页面，这是一个将 DMN 与 BPMN 集成的预处理。其概念是提供一个或多个决策::DataObject 元素，然后测试是否可以正确评估指定的目标决策。

如果出现任何错误或异常，您可以创建分析器来调试脚本模块和测试客户端的代码。

在这个 测试模块”过程之后， Enterprise Architect保证 BPMN2.0::DataObject 元素可以很好地与 DMN 模块一起工作。

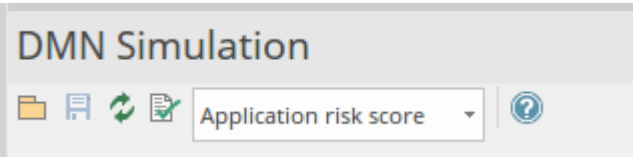
然后，您通过加载数据对象并将 DMN 模块决策分配给 BPSim属性来配置 BPSim，这将进一步用作从网关传出的序列流的条件。

访问

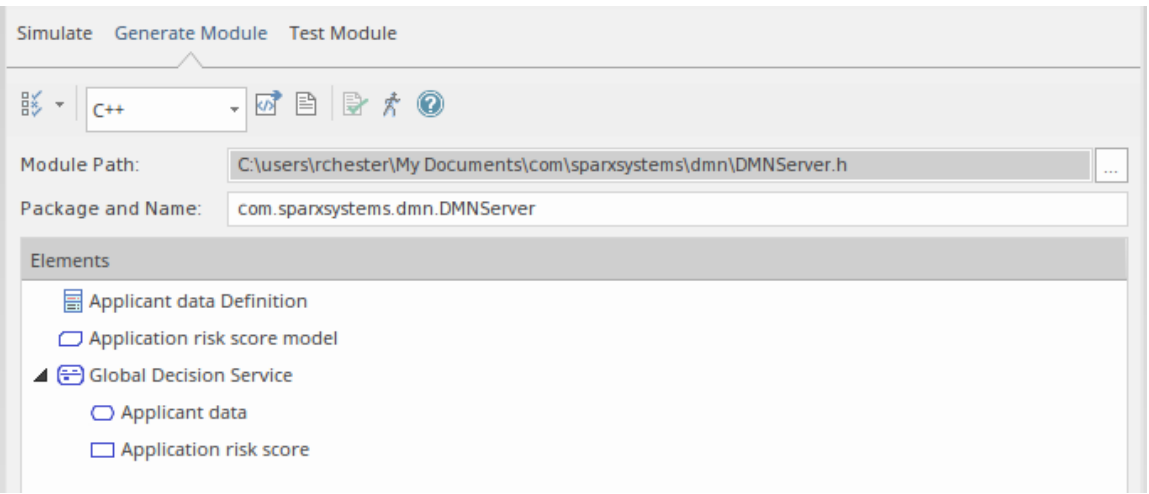
功能区	仿真>决策分析>DMN>Open DMN仿真>生成模块

DMN 模块：代码生成

在 DMN仿真窗口上，在工具栏的数据输入字段中选择要从中生成模块的 DMN 结构。



单击 生成模块”选项卡，然后按住 Ctrl 并单击要生成到服务器的 DMN 元素的名称。



在选项卡工具栏的数据输入字段中，选择要生成的语言，然后在 模块路径”字段中单击 ... 图标并浏览到生成模块的路径位置（注记，对于Java ，路径有以匹配包结构）。

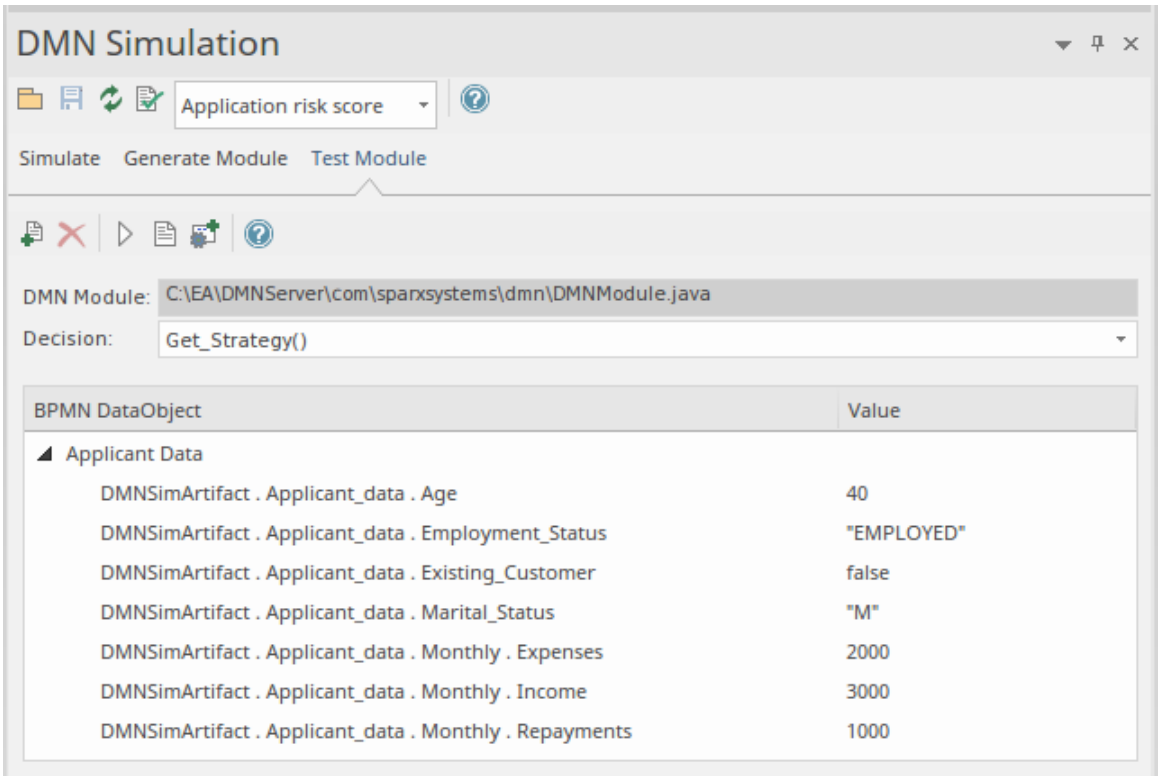
单击生成按钮 生成。

生成完成后，单击完成按钮打开 模块的 测试模块”选项卡。

DMN服务器：测试模块

当您使用 按钮选择 测试模块”选项卡时，“DMN 模块”字段将自动填充您最近在 生成模块”选项卡上生成的模块的生成 DMN服务器路径。如有必要，在 决策”字段中单击下拉箭头并选择所需的决策。

单击工具栏中的添加 BPMN 数据对象按钮 ( ) 并选择一个或多个 (Ctrl+单击) BPMN2.0 数据对象以添加到主面板的列表中。



现在单击工具栏上的运行按钮。在系统输出窗口中，此消息表明 DMN服务器和 BPMN2.0 DataObject 可以很好地相互配合以评估所选决策：

为 *DMN*服务器运行测试客户端...

dmnServer.Application_risk_score：*133.0*

结果：*133.0*

运行成功完成。

如果有错误，通过单击分析器工具栏按钮创建 脚本并使用该脚本修复问题。

重要提示：建议在将 DMNServer.java 与Enterprise Architect BPSim 执行引擎集成之前执行此 测试模块”步骤。请参阅将 *DMN* 模块集成到仿真以获取仿真帮助帮助。

代码生成和连接到 BPMN

- 生成Java 、 JavaScript 、C++ 或 C# 中的 DMN服务器
- Java版运行服务器的运行/调试测试
- 将 DMN服务器连接到Enterprise Architect BPSim 执行引擎

公共错误和解决方案

- 变量类型：由于 DMN 模型使用 FEEL 语言（用JavaScript仿真），变量类型不是强制性的；但是，当为编译的语言生成代码时，您必须键入一个变量 - 有上下文菜单选项和标记值用于设置变量的类型
- 由于 DMN 表达式允许使用空格，因此为了阐明复合输入数据，“.”之前和之 必须有一个空格。在表达式中；例如， 申请人数据。年龄”有效，而 申请人数据。年龄”无效
  注记使用自动完成特征时不会出现此问题
- 运行验证将帮助您定位大部分建模问题；在模拟和代码生成之前执行此操作

注记

- 用Java编译需要对目标目录有完全的读写权限；如果模块路径设置为 C:”或 C:\Program Files (x86)”，编译将失败

集成BPSim进行仿真

DMN 的优势在于它能够通过决策需求图来描述业务需求，并将复杂的逻辑封装在多种表达方式中，例如决策表和 Boxed Context。

同样，BPMN 的优势在于它能够用任务和事件的序列流来描述业务流程，或者用信息流来描述流程的协作。

决策需求图在业务流程模型和决策逻辑模型之间架起了一座桥梁：

- 业务流程模型定义业务流程中需要决策的任务
- 决策需求图定义了在这些任务中要做出的决策、它们的相互关系以及它们对决策逻辑的要求
- 决策逻辑以足够的细节定义所需的决策，以允许验证和/或自动化

决策提供了一个完成决策模型，通过在细节中指定在流程任务中执行的决策来补充业务流程模型。

本主题中演示的两个示例可以从以下位置访问：

- EA示例模型|模型仿真| BPSim 模型
- 蓝图|业务建模| BP模拟 | BPSim 案例研究

BPSim 表达式使用 DMN模型有两种方式：

- DMN 的决策服务 - 由贷款申请进程演示
- DMN 的 BusinessKnowledgeModel - 由交付成本计算证明

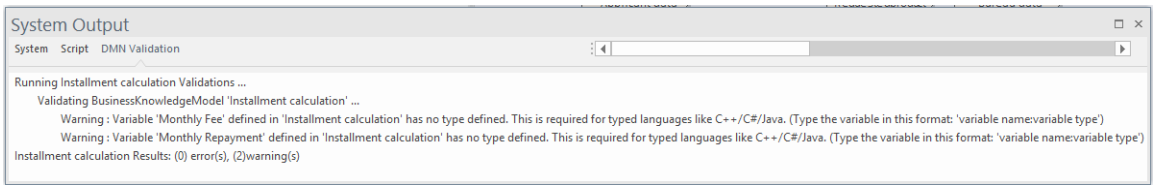
将 DMN模型与 BPSim模型集成的过程包括：

- DMN模型验证、仿真、代码生成和生成模块的测试
- 设置从 BP工件到工件使用依赖关系
- 从 DMN 数据集中生成或更新 BPMN 数据对象
- 在 BPSim 中创建属性参数以用于从网关流出的任务和序列
- 将 DMN 接口绑定到属性Parameters

针对Java等编译语言的 DMN模型验证

当您创建 DMN模型并在Enterprise Architect中对其进行仿真时，驱动仿真的代码是JavaScript ；这意味着不需要显式键入变量（变量类型是从分配给它的值推断出来的）。

但是，对于 C++、C#、 Java等语言，编译器会报一个变量没有类型的错误。

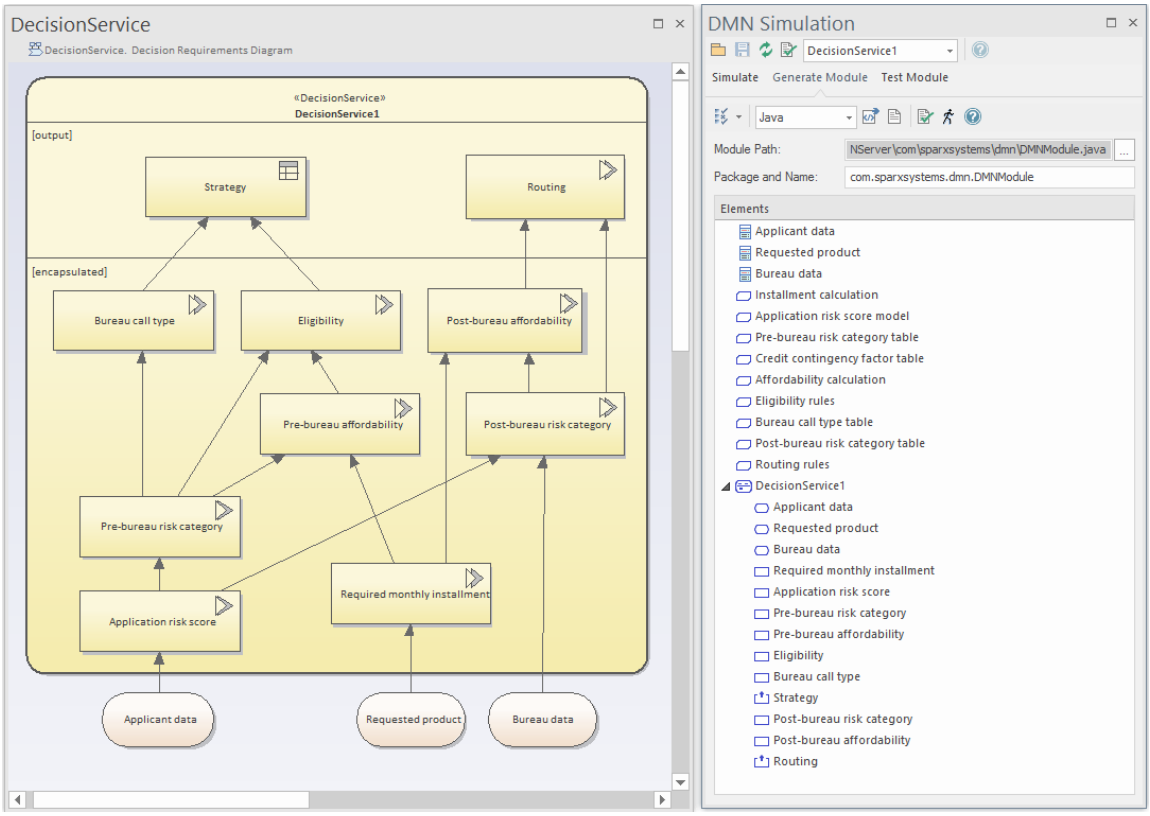


要生成这些语言，您必须在模型上运行验证并使用结果来查找需要其类型集的变量。例如：

- 业务知识模型参数——选择要在DMN 表达式窗口中查看的BKM元素，点击第二个按钮打开 参数”对话框，指定参数类型
- 决策类型 - 选择决策元素，打开属性窗口，对于属性'variableType'从'Value'字段中选择
- 决策表输入/输出- 在决策表输入/输出子句上，右键显示输出上下文菜单并选择类型
- Boxed Context 变量 - 请参阅Boxed Context帮助主题

Java中的DMN代码生成

在使用验证修复任何变量类型问题后，我们可以进入 DMN仿真窗口中的 生成模块”页面。

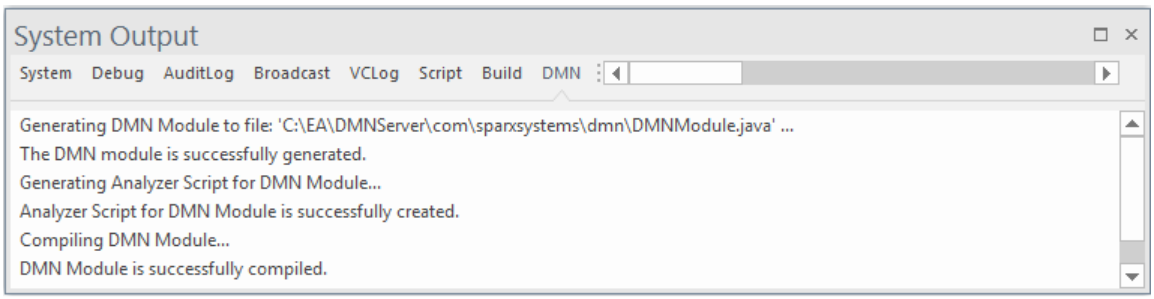


- 在顶部工具栏数据输入字段中选择*DecisionService1* ； *DecisionService1*中涉及的所有元素现在都将包含在列表中
- 项目定义和业务知识模型是全局元素
- 输入数据和决策被封装在 DecisionService元素中
- 支持的语言有C++、C#、 Java和JavaScript ；注记对于JavaScript生成的.js文件与模拟脚本相同（ 仿真”选项卡|运行按钮下拉菜单|生成新脚本（脚本窗口））除了省略了仿真相关的代码
- 对于Java · 模块路径”值必须 包结构匹配；在本例中，必须将 DMNModule.java 生成到一个目录以形成一个以 \"com\sparxsystems\dmn\DMNModule.java”结尾的文件路径 - 您必须手动为现在创建目录结构

单击工具栏上的生成代码按钮 ()。这个例子将使用Java ；但是，C++ 和 C# 是相同的。执行这些操作：

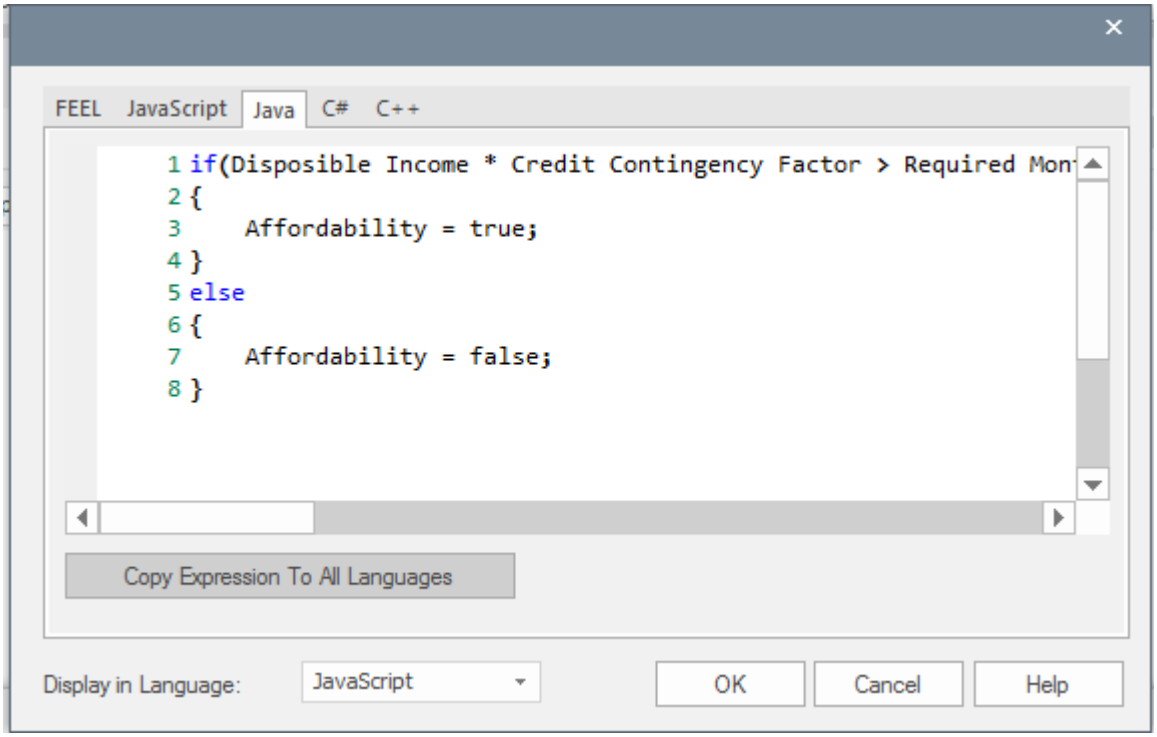
- .java 文件生成到指定的路径
- 为这个工件创建的分析器编译脚本)
- 这个分析器脚本编译脚本被执行
- 在系统输出窗口中报告进度消息

如果模型有效，此过程将返回消息：



如果出现编译错误，您可以通过点击工具栏上  按钮旁边的  按钮打开生成的.java文件，手动修复问题，并使用生成的脚本编译，直到成功。

编译失败的一个常见原因是语言可以有不同的表达式语法。您可能需要为一种语言提供一个值以覆盖默认值（右键单击 DMN Literal Expression | Edit Expression ）。

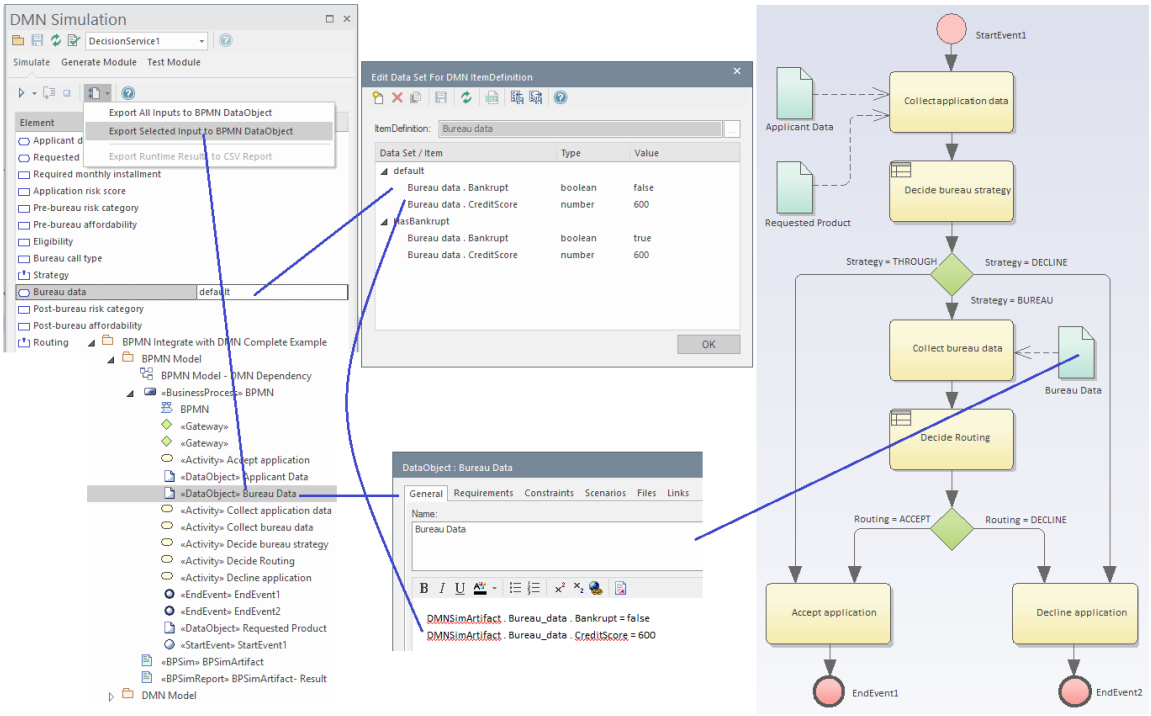


在外部使用之前测试DMN Modules

将模型生成到 java 代码并成功编译后，我们现在想要：

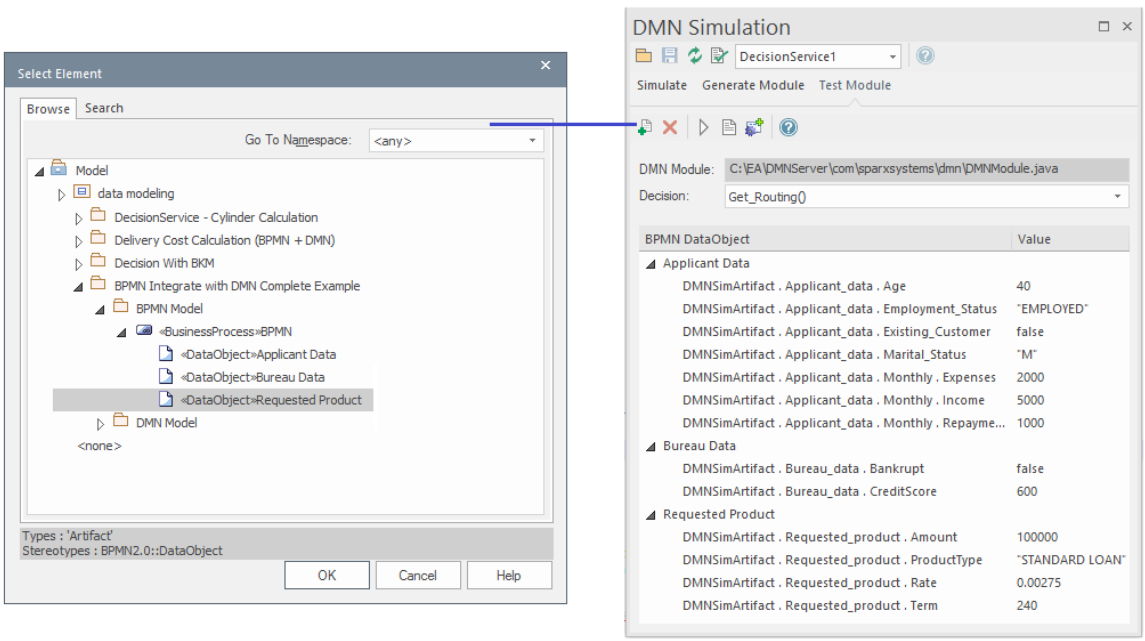
- 测试这个模块的正确性
- 为其提供输入
- 获取输出决策值

生成BPMN数据对象

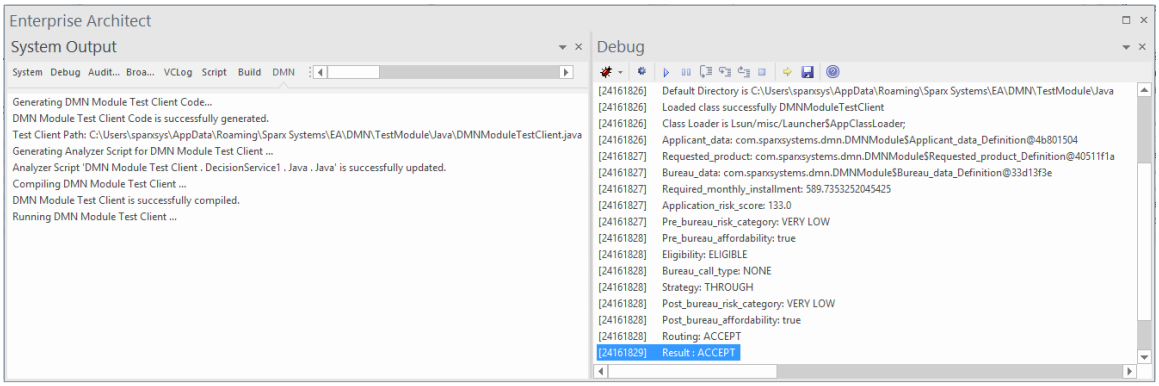


所选数据集携带的数据将生成到BPMN DataObject的 注记”字段。

- 单击 按钮（ 生成模块”选项卡工具 右侧第二个）打开 测试模块”选项卡



- 单击工具栏上的 以选择输入 BPMN DataObject 元素
- 从 决策”组合框中选择可用的输出，例如决策()，然后单击工具栏上的运行按钮



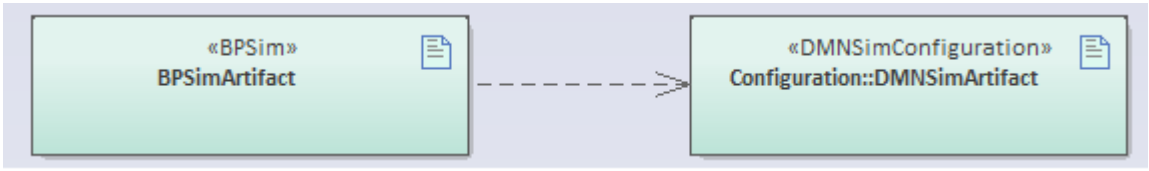
执行结果将显示在调试窗口中。也可以打开测试模块文件，在行上设置断点，在DMN模块中调试，进行行级调试。

我们强烈建议您使用此窗口测试您的 DMN 模块，以确保 DMN 模块在给定输入（来自 BPMN 数据对象）下正常工作，并且可以成功计算输出结果。

注记模块路径：工件保存在 DMN 的 'Filepath属性中。

现在，是时候将 DMN 模块与 BPSim模型集成了。

第一步是在工件和工件仿真之间建立使用依赖关系。



注记：如果需要， A工件可以使用多个 DMN 模块。这只是简单地将所有工件支持的图表放在这个图表上，并从该图表中绘制出从 BP 到每个工件仿真的连接工件。

这些帮助主题提供了使用这些方法的两个示例。看：

- 示例：将 *DMN* 决策服务集成到决策数据物件和属性参数中
- 示例：将 *DMN*业务知识模型集成到属性参数中

示例：将 DMN 决策服务集成到决策数据物件和属性参数中

BPSim 的模型构建器中提供了将 DMN决策服务集成到 BPSim模型中的示例。

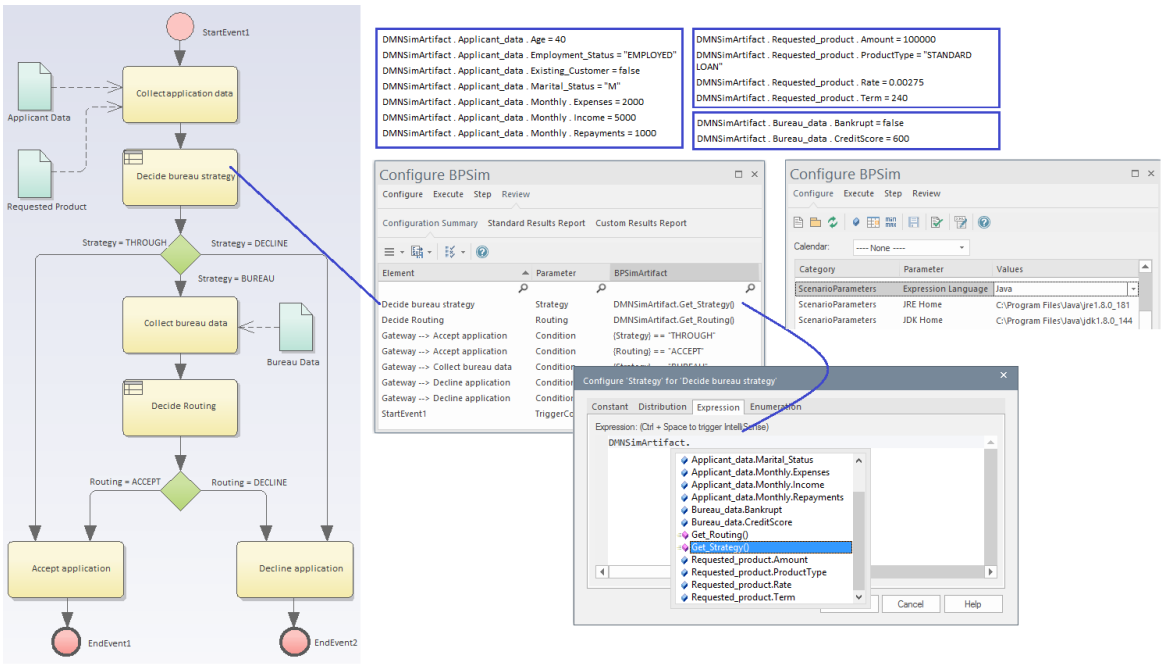
要访问它：

- 将蓝图设置为业务建模> *BPSim*。将显示模型构建器对话框。
- 从 BPSim 案例研究组中选择*BPMN* 与 *DMN* 集成完全示例
- 单击创建模型按钮。

这将创建 BPMN 和 DMN 模型，配置为模拟从 BPMN模型对 DMN模型调用。

注记：为了集成 DMN 模块，表达式语言必须使用Java ，并且必须正确配置 JRE 和 JDK（ Java的最低版本为1 ）。请参阅帮助主题BPSim Business Simulations中的安装 *BPSim* 执行引擎BPSim Business Simulations 。

在此 BPMN 图中，有三个 DataObjects（浅绿色）连接到 BPMN 活动。这些数据对象元素携带从 DMN仿真窗口生成的输入数据。



- 当模拟运行时，它将在模拟令牌通过时自动加载连接到任务的所有数据对象
- 第二个业务规则任务 决定局策略”配置为将属性 Strategy”设置为值 DMNSimArtifact.Get_Strategy()”；您无需输入此内容 - 按 Ctrl+Space 键可帮助您编辑表达式

设置完成后，点击 执行”标签，模拟模型，即可查看报告，或进入 节”页面对BPSim模型进行分步调试。

示例：将 DMN业务知识模型集成到属性参数中

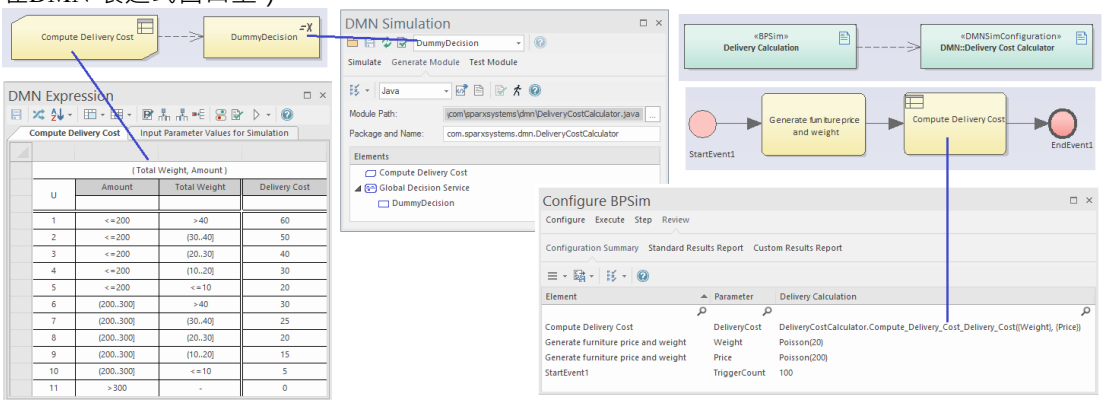
在某些情况下，您可能只想设计一个决策表以在 BPMN模型中使用。如果是这样，则无需经历创建决策服务、决策、输入数据甚至项目定义的过程，因为可以直接连接业务知识模型（BKM）。

BPSim 的模型生成器中提供了将 DMN BKM 集成到 BPSim模型的示例。

要访问它：

- 将蓝图设置为业务建模> *BPSim*。将显示模型构建器对话框。
- 从 BPSim 案例研究组中选择*BPMN* 与 *DMN* 集成 - 交付成本计算
- 单击创建模型按钮

1. 使用参数创建一个简单的业务知识模型作为决策表（您还可以创建其他表达式，例如盒装上下文或文字表达式），然后对逻辑（输入子句、输出子句、规则）模型并对其进行测试（ 输入参数值"用 仿真选项卡在DMN 表达式窗口上）。



2. 使用知识需求连接器将 BKM 连接到一个决策。此决策用作多个 BKM 函数的组名；您只需在表达式中输入一个数字（如 "0"）。例如，如果您只想使用五个 BKM 生成Java代码（考虑到您的模型可能有一百多个 BKM），您可以将这五个 BKM 连接到一个决策并在 DMN仿真窗口中选择此决策，然后所有五个 BKM 都将自动包含在内。
3. 生成Java代码并且（假设一切正确）编译将成功。
4. 在BPSim配置中，我们只需使用智能感知来构建任务 计算运输成本'的表达式。

在这个例子中， 生成家具价格和重量"任务将为属性 重量"和 价格"生成 机值，然后 计算交货成本"任务会将该值传递给业务知识模型，并将结果带回属性 "DeliveryCost"。

您现在可以执行模拟，并逐步执行调试过程来观察属性值的变化等。

集成到UML类元素

创建并模拟决策模型后，您可以生成Java 、 JavaScript 、C++ 或 C# 的 DMN 模块并对其进行测试。

DMN Module 可以与UML类元素集成，因此从类元素生成的代码可以重用 DMN Module 并且结构良好。由于类元素是一个状态机，在与DMN Module集成后，可以可执行状态机模拟通用地使用DMN Module的强大功能。

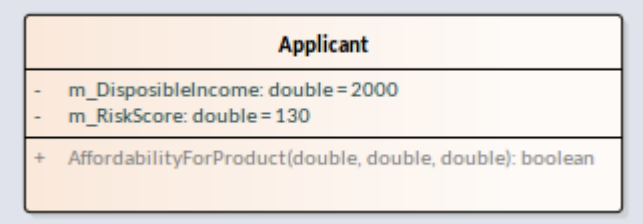
在本主题中，我们将解释将 DMN模型与UML类元素集成的过程，考虑到：

- 类元素的要求
- DMN 模型
- DMN捆绑到类&智能感知
- 类元素上的代码生成

类元素的需求

假设我们有一个类*Applicant* ，其操作*AffordabilityForProduct*评估申请人是否能够负担得起贷款产品。

A简化的模型类似于：



类*Applicant*包含两个属性，实际上是根据申请人的月收入、支出、现有还款、年龄和就业状态等更基础的数据计算得出的。

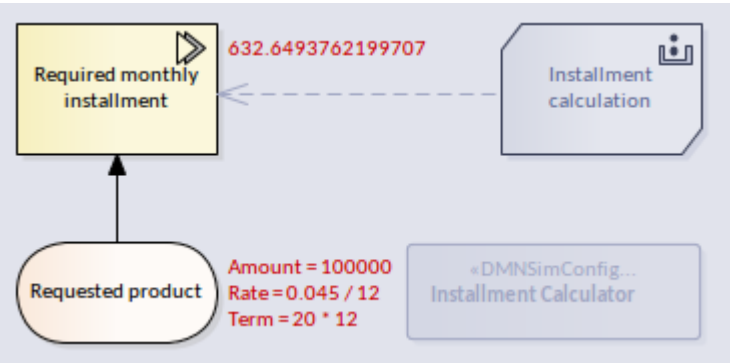
然而，在本例中，我们通过跳过这些步骤并直接提供可支配收入和风险评分来简化模型。

DMN 模型

在这个例子中，我们有两个不相交的 DMN 模型来展示一个UML类可以集成多个 DMN 模型。

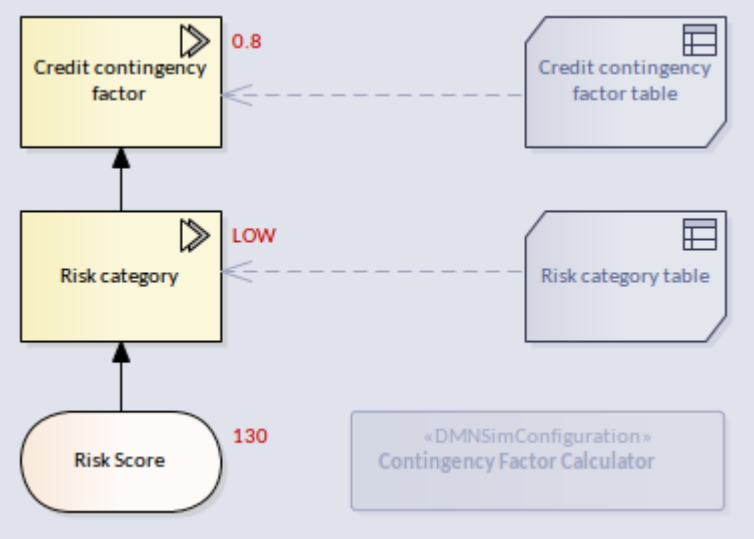
分期付款计算器

此 DMN模型根据金额、费率和条款计算每月还款额。它由输入数据、决策和决策业务知识模型组成。



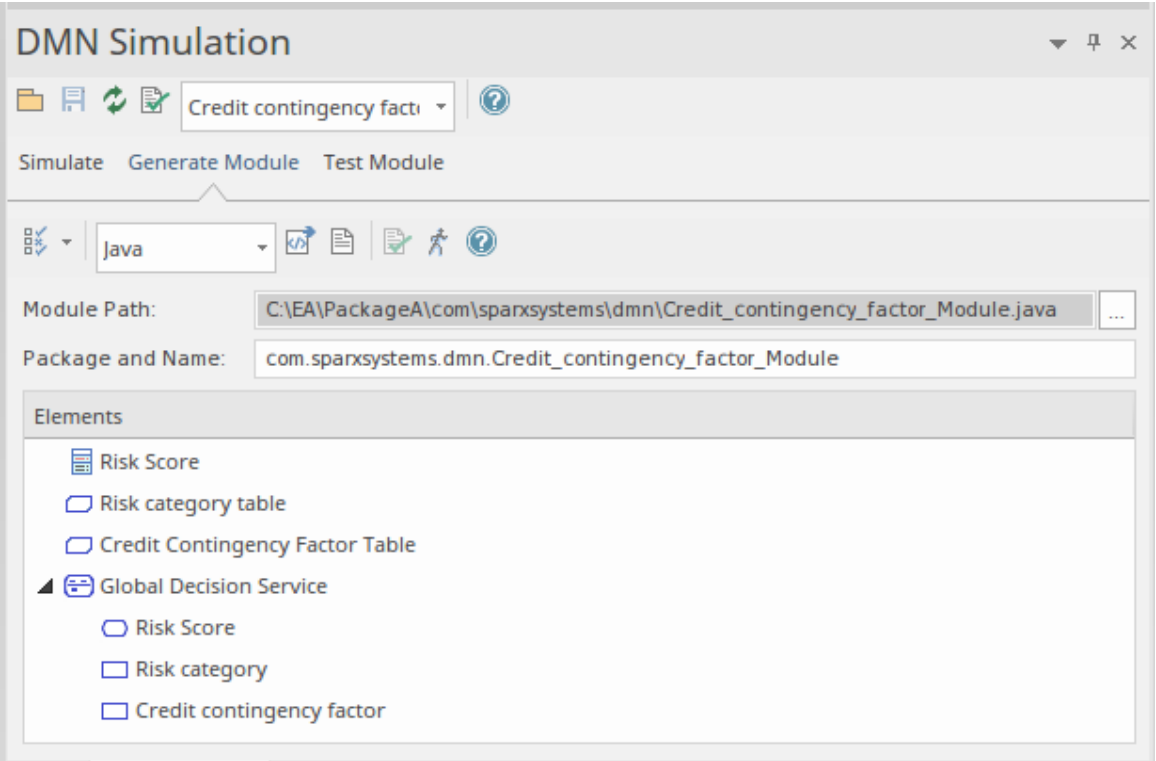
信用应急因子计算器

该 DMN模型根据申请人的风险评分计算信用或有因素。它由一个InputData、两个Decision和两个业务模型组成。



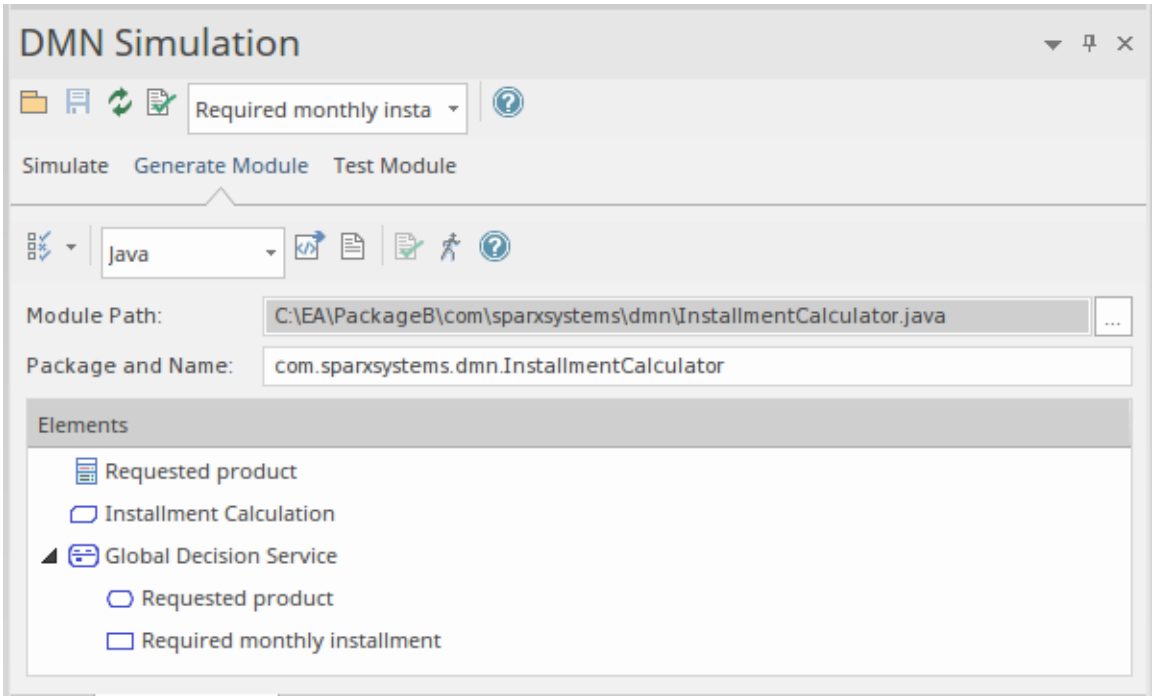
注记：在本例中，我们聚焦关注如何将 DMN 模块集成到类元素中； DMN 元素的细节在此不作描述。

为两个 DMN 模型生成代码

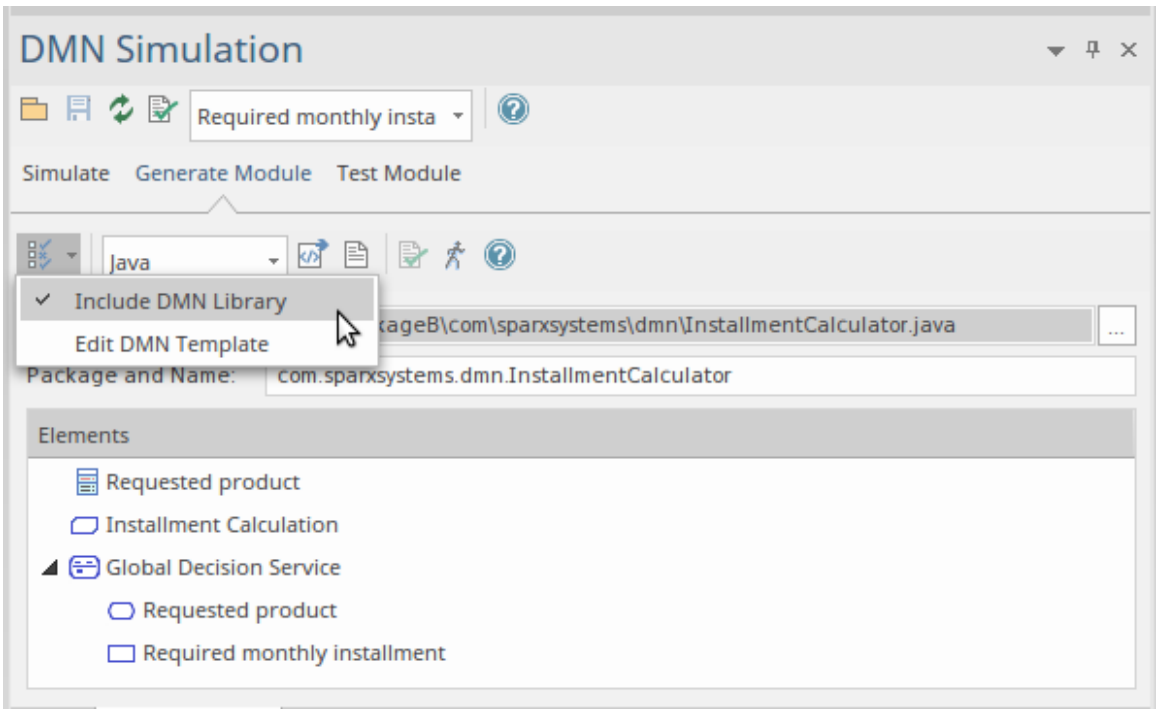


单击生成代码图标，并检查您是否可以在系统输出窗口的 "DMN"选项卡中看到此string ：

DMN 模块已成功编译。



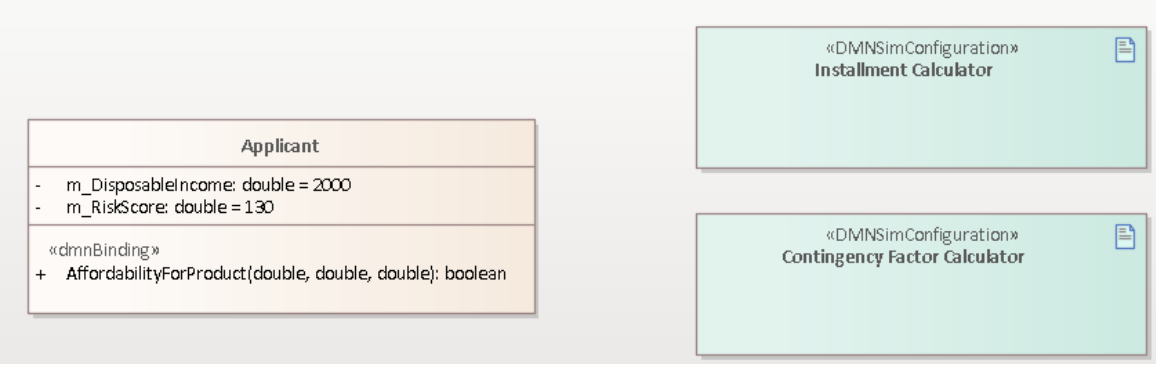
注记：由于本模型使用内置函数PMT，因此必须包含 DMN 库：



单击生成代码图标，并检查您是否可以在系统输出窗口的 "DMN" 页面中看到此string ：
DMN 模块已成功编译。

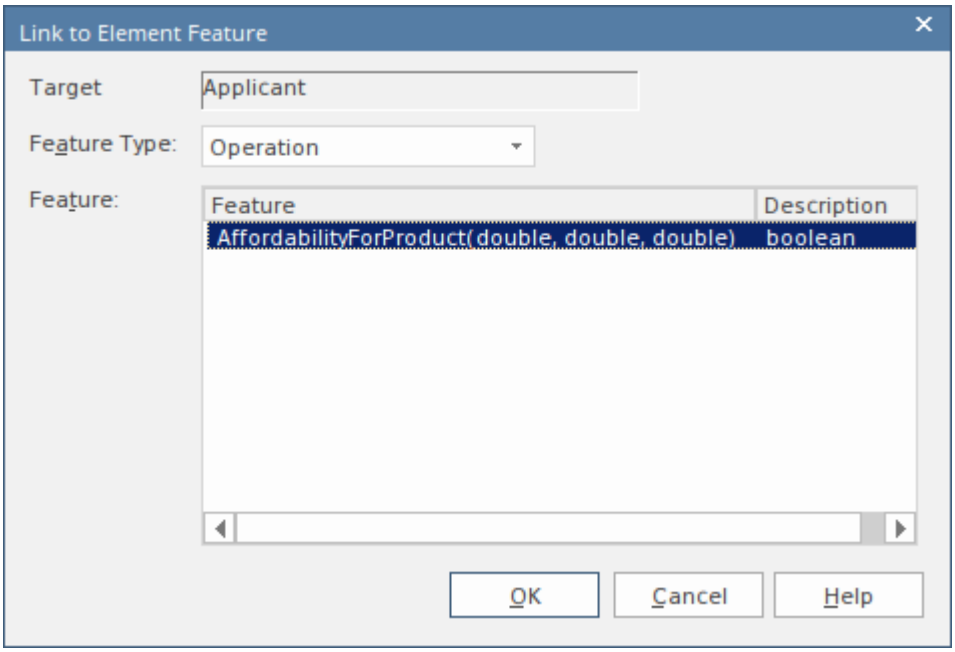
DMN捆绑到类&智能感知

把两个工件放在类图上。



使用快速链接器创建一个从类应用程序到每个工件依赖关系的快速链接器。

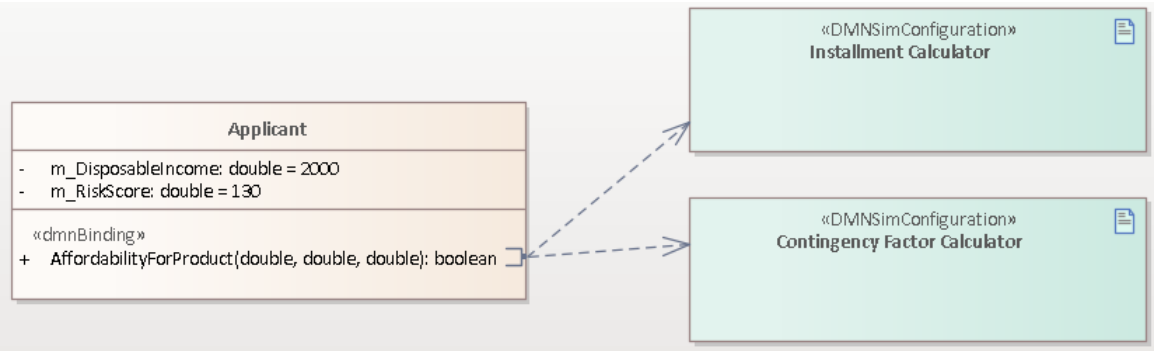
在创建连接器时，会出现一个对话框，提示您选择要绑定到 DMN 模块的操作。



DMN模块绑定操作时：

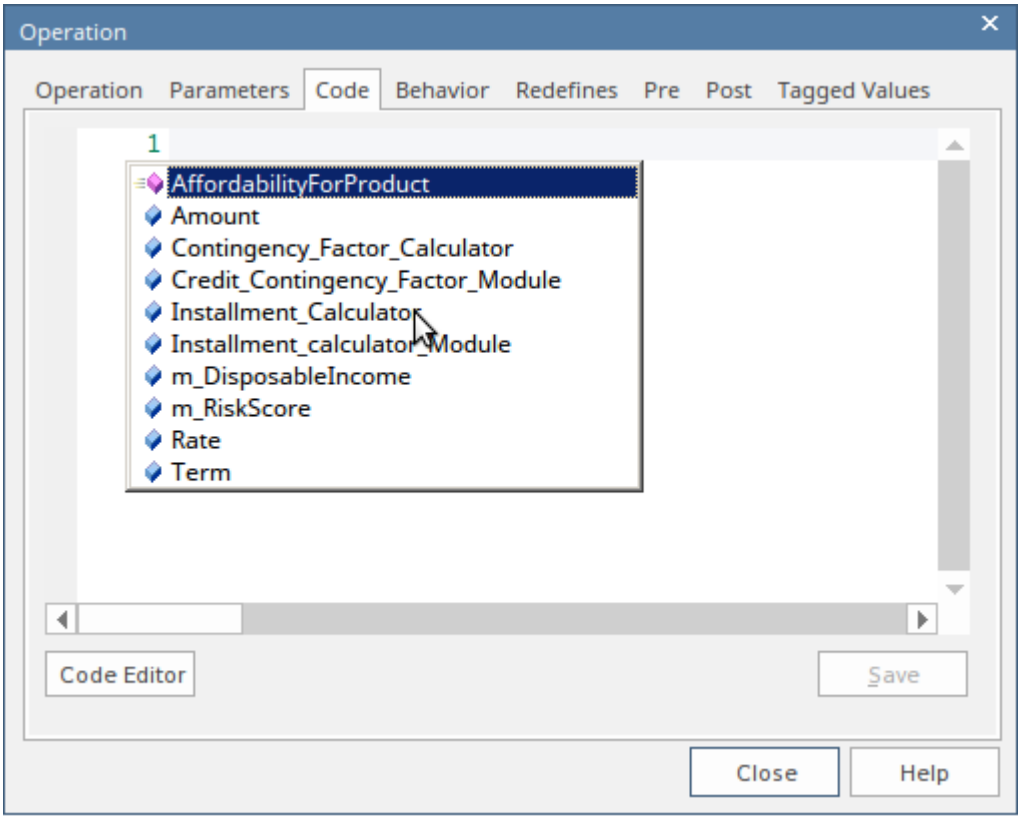
- 该操作采用构造型 <<dmnBinding>>
- 依赖连接器链接到操作

多个工件操作可以绑定到同一个操作。



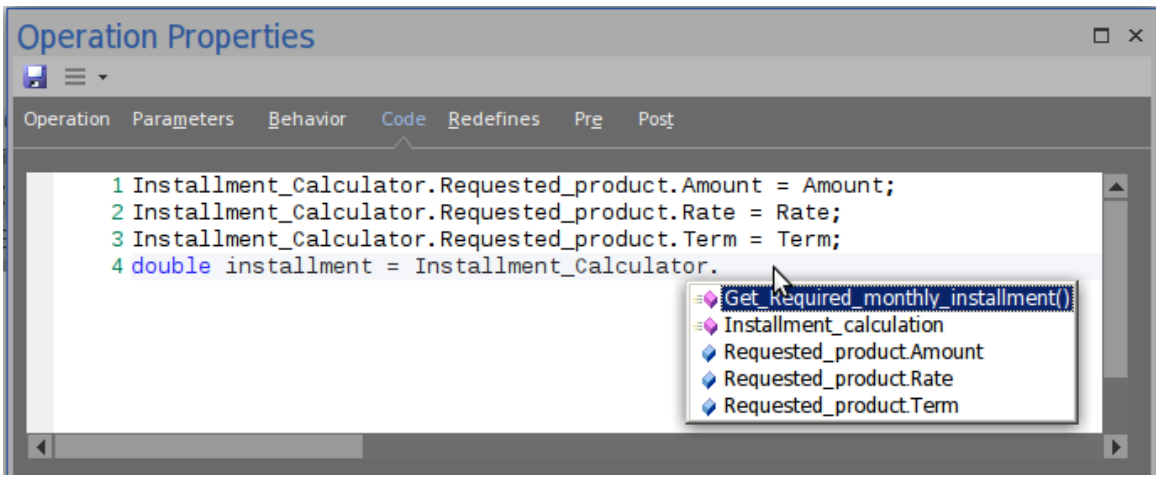
在DMN Bindings之后，智能感知操作的代码编辑器将支持DMN Modules。要触发智能感知，请使用以下组合键：

- Ctrl+空格 - 在大多数情况下
- Ctrl+Shift+Space - 当 Ctrl+Space 在括号 '(' 之后不起作用时；例如，函数的参数，或在 'If' 条件的括号内



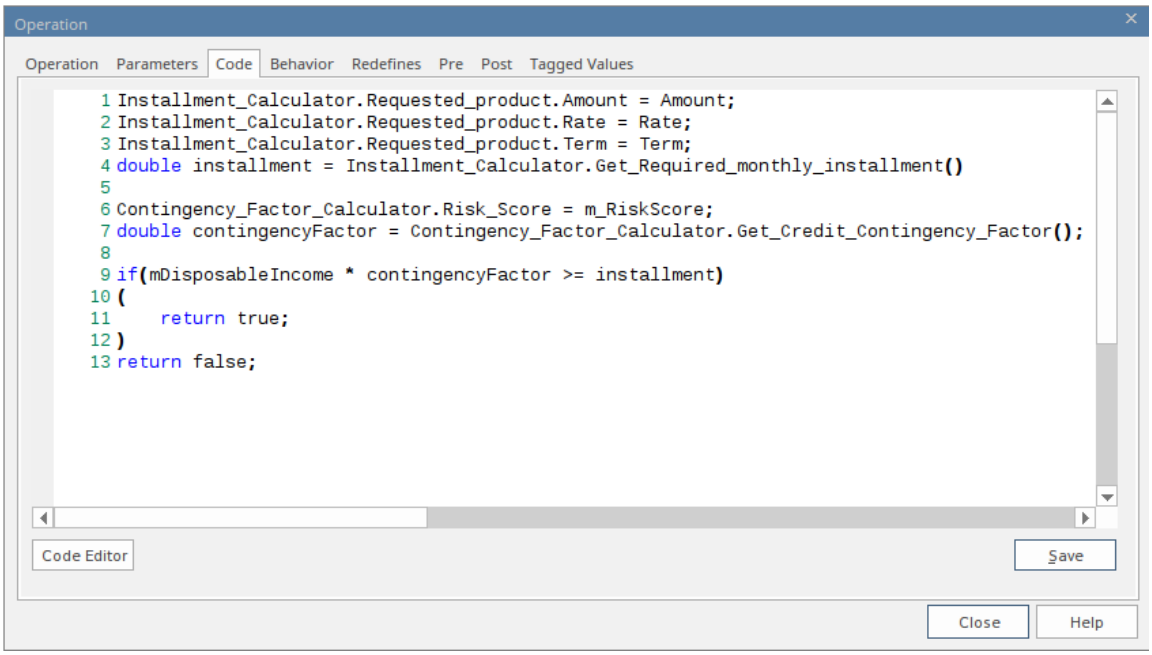
- 将列出类属性 - m_RiskScore、m_DisposableIncome
- 将列出操作参数 - 金额、费率、团队
- 操作将被列出 - AffordabilityForProduct
- 将列出所有绑定的 DMN 模块 - Contingency_Factor_Calculator、Installment_Calculator

使用智能感知支持编写代码非常容易。在访问 DMN 模块时，所有的输入数据、决策和业务模型将被列出以供选择。



此插图显示我们正在从 Installment_Calculator 中选择 "Get_Required_monthly_installment()"。

这是操作的最终实现。



类的代码生成（使用 DMN集成）

'生成代码生成' 生成类代码：

```
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
```

- DMN 模块作为类的属性生成
- dmnBinding 操作的代码已更新

注记：无论生成选项是 'Overwrite' 还是 'Synchronize'，如果操作的代码具有构造型 'dmnBinding'，都会更新。

导入 DMN XML

Enterprise Architect支持 DMN 1的导入。 1或1 .2 XML 文件到项目中，同时包含模型语义和图表交换信息。

访问

在浏览器窗口中，选择要导入 XML 文件的包。然后使用此处列出的方法之一从 DMN 1打开 导入包”。 1 XML'对话框。

功能区	发布>模型交换>导入>DMN 1 。 1
键盘快捷键	Ctrl+Alt+I :其他XML 格式 >其它1 。 1

导入DMN 1 . 1 XML

节,节	行动,行动
1	在 文件名”字段中，输入源文件路径和名称，或单击  图标定位并选择文件。
2	单击导入按钮将文件导入包中。

从 OMG导入示例

1. 在此链接下载 zip 文件并将其解压缩到您的文件管理器。
2. 浏览文件夹示例*/Chapter 11/* 。
3. 单击文件*Chapter 11*示例并将其作为 DMN 1导入。 1格式文件。

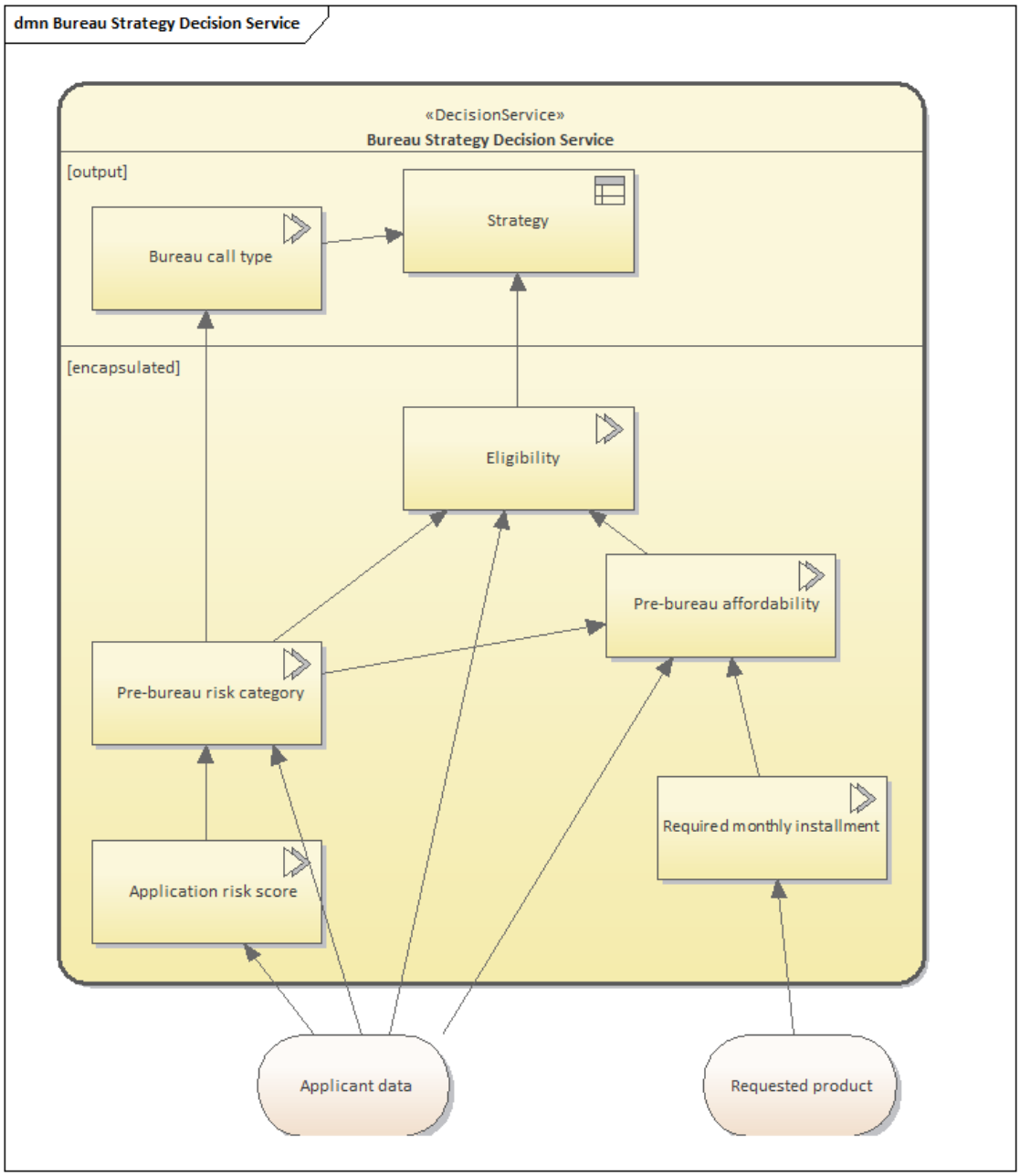
导入这些图表以显示模型的不同视角：

- 所有自动化决策的 DRD
- 用于审阅应用决策点的 DRD
- 决定路由决策点的 DRD
- 决策局战略决策点的 DRD

导入这些图以定义决策服务：

- 局战略决策服务
- 路由决策服务

此处显示了 局战略决策服务”图。它有两个输入数据元素（申请人数据、请求的产品）、两个输出决策（局呼叫类型、策略）和五个封装决策。注记，图中未显示调用的业务模型。



为了从模型生成生产代码，您可能需要运行验证和模拟以确保导入的模型具有正确的表达式。

1. 在任何列出的图表上创建一个工件Sim配置，然后双击它以在 DMN仿真窗口中打开它。
2. 决策服务和决策列在目标下拉字段中。指定目标后，窗口中会列出所有必需的元素。
3. 单击 验证”按钮（工具栏上的第 4 个）。如果显示任何错误或警告消息，我们建议您在执行模拟之前按照错误或警告说明的指示修复问题。
4. 为输入提供适当的值，然后运行仿真或逐步调试模型。

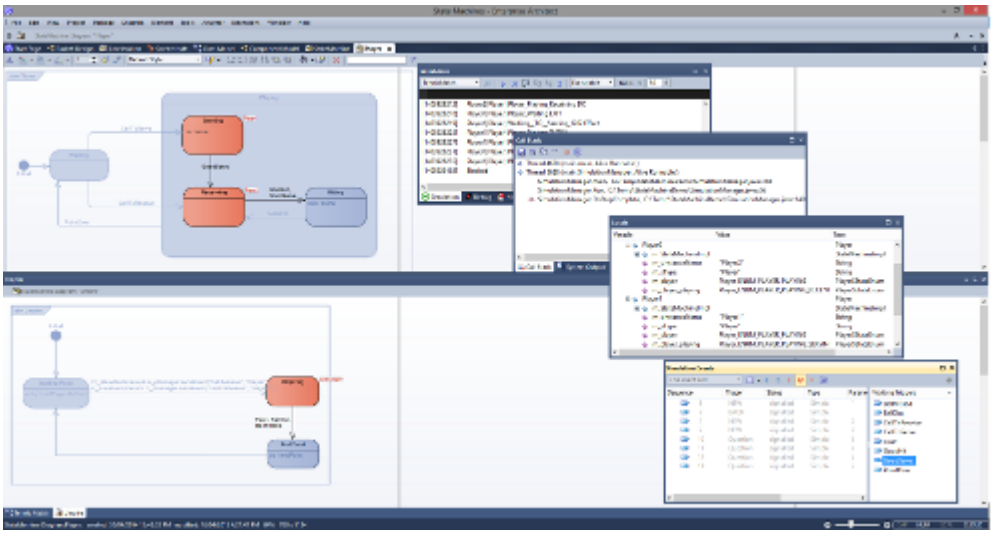
注记： “Bureau Strategy DecisionService”示例也可在决策模型中找到。从 开始”图中，选择 业务建模 > DMN 示例 > 局战略决策服务”。

更多信息

在Enterprise Architect中， Decision Model and Notation (DMN)特征起着至关重要的作用：提供有效建模决策的基本结构。此功能允许在工具中的图表中直观地描绘组织决策，使业务分析师能够准确定义决策模型。Enterprise Enterprise Architect支持这些决策模型的可选自动化，从而简化决策流程。

Enterprise Architect的 DMN 功能促进了组织之间无缝共享和交换决策模型。这种互操作性确保决策模型可以轻松进行交流和协作，从而促进不同团队和利益相关者之间的高效决策。

可执行状态机



可执行状态机提供了一种快速生成、执行和模拟复杂状态模型的方法。与使用Enterprise Architect的仿真引擎动态模拟状态图相比，多平台可执行状态机提供了完成的语言特定实现，可以形成多个软件的行为“引擎”。执行的可视化基于与仿真功能的无缝集成。模型的演变现在提出了更少的编码挑战。代码生成、编译和执行由Enterprise Architect负责。对于那些有特殊要求的人，每种语言都提供了一组代码模板。模板可以由您自定义，以您认为合适的任何方式定制生成的代码。

这些主题向您介绍建模可执行状态机的基础知识，并帮助您了解如何生成和模拟它们。

Enterprise Architect统一版和终极版支持可执行状态机的创建、使用和生成代码

建造和执行状态机概述

建造和使用可执行状态机非常简单，但确实需要一点计划和一些有关如何将不同组件链接起来以构建有效执行模型的知识。幸运的是，您不必花费数小时来获得正确的模型并修复编译错误，然后您就可以开始可视化您的设计。

在勾勒出模型的广泛机制后，您可以在几分钟内生成代码来驱动、编译、执行和可视化它。这些要点总结了开始执行和模拟状态机所需的内容。

功能	描述
编译类和状态模型	第一个任务是构建描述要构建的实体和行为的标准UML类和状态模型。对您的模型感兴趣的每一类都应该有自己的状态机，描述控制其整体行为的各种状态和转换。
创造一个可执行状态机的工件	一旦你已经建立了类模型和状态，就该设计可执行状态机的工件了。这将描述所涉及的类和对象，以及它们的初始属性和关系。正是链接多个对象的脚本决定了它们在运行时将如何通信。请注记，可以在一个可执行状态机工件或多个对象绑定在一起，并将其绑定为单个类的实例。这些将在运行时具有自己的状态和行为，并且可以在必要时进行交互。
生成代码并编译	无论您使用JavaScript 、C++、 Java还是 C#， Enterprise Architect的工程能力都为您提供了一个有效的工具，允许您随时重新生成可执行文件，并且不会丢失您可能制作的任何定制代码。这是项目生命周期中的主要优势。可能还值得注意的是，生成的整个代码库是独立且可移植的。代码绝不会与模拟引擎使用的任何基础设施相结合。
执行状态机	那么我们如何看待这些状态机的行为呢？一种方法是为每个平台构建代码

	库，将其集成到一个或多个系统中，在可能的几个部署场景中 就地"检查行为。或者我们可以使用Enterprise Architect执行它。无论是Java 、 JavaScript 、C、C++ 还是 C#， Enterprise Architect都会负责创建运行时、模型的托管、其行为的执行以及所有状态机的再现。
可视化状态机	可执行状态机可视化集成了Enterprise Architect的仿真工具。观察图表上发生的状态转换以及针对哪个object 。轻松识别共享相同状态的对象。重要的是，这些行为在多个平台上保持一致。您还可以控制机器的运行速度，以更好地了解事件的时间线。
调试状态机	当状态应该改变但不改变时，当转换不应该被启用但被启用时，当行为 - 简而言之 - 不受欢迎并且从模型中没有立即显现时，我们可以转向调试。Enterprise Architect的可视化执行分析器附带了 ExecutableStateMachine 代码生成支持的所有语言的调试器。调试提供了许多好处，其中之一可能是验证/证实附加到状态机中的行为的代码，以确保它被实际反映在执行过程中。

建模可执行状态机

可执行状态机模型和计算机模型所需的大部分工作是基于标准 UML 的类状态建模，尽管必须遵守一些约定以确保形成良好的代码库。唯一新奇的构造工件就是用元素来构成可执行状态机实例或场景的配置。工件：用于指定详细信息

- 代码语言（ JavaScript 、C#、 Java 、C++ 包括 C ）
- 场景中涉及的类和状态机
- 包括运行状态的实例规范；注记这可能包括同一状态机的多个实例，例如在网球比赛模拟中使用了两次 球员”类

可执行状态机基本建模工具和对象

这些是构建可执行状态机时使用的主要建模元素。

元素类型	描述
类和类图表	类定义了与被建模的状态机相关的object类型。例如，在一个简单的网球比赛场景中，您可以为球员、比赛、击球和裁判中的每一个定义一个类。每个都有自己的状态机，并且在运行时将由每个相关实体的object实例表示。有关类和类图的更多信息，请参阅*UML*建模指南。
状态机	对于您定义的将在场景中具有动态行为的每个类，您通常会定义一个或多个UML状态机。每个状态机将确定适用于拥有类的一个方面的合法的基于状态的行为。例如，可以有一个状态机代表玩家的情绪状态，一个跟踪他当前的健康和能量水平，一个代表他的输赢状态。所有这些状态机都会在状态机场景开始执行时被初始化并启动。
可执行状态机工件	这种刻板工件是用来指定可执行状态机的参与者、配置的核心元素和启动条件。从场景的角度来看，它用于确定涉及哪些（类）实例，它们可能会触发器发送和发送什么事件，以及它们在什么启动条件下运行。 从配置方面工件，你可以用来建立一个分析器脚本的链接，该分析器脚本将确定输出目录、代码语言、编译脚本等。右键单击工件设备将允许您生成、构建、编译和可视化您的机器的状态机执行。

状态机支持的构造

此表详细说明了支持的状态机造以及与每种类型相关的任何限制或一般约束。

构建	描述
状态机	• 简单状态机：状态机有一个区域 • 正交状态机：状态机包含多个区域 顶级区域（状态机所有）激活语义： **默认激活**：状态机开始执行时。 **入口入口**：从入口过渡到包含区域中的顶点。 • 注记*1* ：每个拥有区域的状态机，入口只有一个转移 从入口点到该区域内的区域 • 注记*2*：这个状态机可以被一个子机状态引用——连接点引用应该在子机

	中状态为转换的源/目标；连接点引用表示在状态机中定义并由子机状态引用的*Entry*/出口的用法 **多状态机**：浏览器窗口中的列出顺序决定执行顺序。 • 当涉及到一个子机状态时，类下可能有多个状态机 • 使用浏览器窗口工具栏中的上移或下移箭头调整状态机的顺序；置顶的将设置为主状态机 **不支持** • 协议状态机 • 状态机重新定义
状态	• 简单状态：没有内部顶点或过渡 • 复合状态：只包含一个区域 • 正交状态：包含多个区域 • 子机状态：指整个状态机
复合状态Entry	• 默认条目 • 显式输入 • 浅历史条目 • 深度历史条目 • 入口
子状态	• 子状态和嵌套子状态 进入和退出语义，其中转换包括多个嵌套级别的状态，将服从嵌套行为（例如 OnEntry 和 OnExit）的正确执行。
过渡支持	• 外部转移 • 本地转移 • 内部转移 （画一个自我转移 并改变转移 对内部友好） • 完成转移 和完成事件 • 转移 警卫 • 复合过渡 • 触发优先级和选择算法 有关更多详细信息，请参阅*OMG UML*规范。
触发器的事件	可执行状态机仅支持 Signals 的事件处理。 要使用调用、计时或更改事件类型，您必须定义一个外部机制以基于这些事件生成信号。
信号	属性可以在 Signals 中定义；属性的值可以用作转移 中的事件参数转移 防护条件。 例如，这是 C++ 中转换效果中的代码集： 如果（信号->signalEnum == ENUM_SIGNAL2）

	{
	int xVal = ((Signal2*)signal)->myVal;
	}
	Signal2 生成如下代码：
	类 Signal2：公共信号{
	上市：
	信号2(){};
	Signal2(std::vector<String>& lstArguments);
	int myVal;
	};
	注记：更多细节可以通过生成一个可执行状态机并参考生成的 "EventProxy"文件找到。

最初的	Initial Pseudostate 表示区域的起点。最多是一个转移 的源转移 ;在一个区域中最多可以有一个 Initial Vertex。
区域	**默认激活和显式激活：** 转换终止于包含状态： • 如果在区域中定义了初始伪状态：**默认激活** • 如果没有定义初始 Pseudostate，则区域将保持非活动状态，并且包含状态被视为简单状态 • 如果转换终止于区域包含的顶点之一：**显式激活**，导致其所有正交区域的默认激活，除非这些区域也显式输入（多个正交区域可以通过源自的转换显式并行显式输入同分叉伪态） 例如，如果为正交状态定义了三个区域，并且*RegionA*和*RegionB*具有 Initial Pseudostate，则明确激活*RegionC* 。默认激活适用于*RegionA*和*RegionB* ；包含状态将具有三个活动区域。
选择	当复合转换遍历达到此伪状态时，动态评估所有传出转换的守卫条件约束。
连接点	静态条件分支：在执行任何复合转换之前评估保护约束。
分叉/汇合	非线程，每个活动区域交替移动一步，基于完成事件池机制。
入口点/出口点节点	非线程用于正交状态或正交状态机；每个活动区域交替移动一个步骤，基于完成事件池机制。
历史节点	• 状态：代表其所属状态的最近活动状态配置 • ShallowHistory：代表其包含状态的最近活跃的子态状态，而不是那个子状态的子态
延期事件	划一个自我转移 , 并改变转移 *kind* to类型();'在过渡的 影响"字段中。
连接点参考	A点参考表示子机状态引用的状态机中定义的条目/出口的使用（作为子机状态的一部分）。子机状态的连接点引用可以作为Transitions的源和目标。它们代表进入或退出子机状态所引用的状态机。

状态行为	状态 "entry"、 状态"和 "exit"行为被定义为对状态的操作。默认情况下，您将用于每个行为的代码键入到 行为操作的属性窗口的 代码"面板。请注记，您可以通过自定义生成模板将其更改为将代码键入 行为"面板。 生成的 返行"行为将在继续之前运行完成。代码与其他入口行为不并发；'doActivity' 行为被实现为 'execute in order序列entry' 行为。

对其它/类中的行为的引用

如果子机状态引用了当前上下文或类之外的行为元素，则必须在当前上下文类中添加一个<<import>>连接器到容器上下文类。例如：

Class1中的子机状态S1指Class2中的状态机ST2

因此，我们将 <<import>> 连接器从 Class1 添加到 Class2，以便可执行状态机生成代码以正确生成子机状态S1的代码。 （在类1上，单击快速链接箭头并拖动到类2，然后从连接器类型菜单中选择 导入"。 ）

再可执行状态机工件

您可以使用单个工件文件创建组件的多个模型或版本。工件一个电阻器，举例来说，可以重复使用箔电阻器和绕线电阻器这很可能是相似对象的情况，尽管它们由相同的分类器表示，但通常表现出不同的运行状态。从建模的角度来看， A名为 "esistorType"的属性取值 "wire"而不是 "foil"可能是所需的全部。然 可以重新使用相同的状态机来测试可能由于运行状态的变化而导致的行为变化。这是过程：

节	行动
创建或打开部件图	打开要处理的部件图。这可能是包含您的原始工件。
选择要复制的可执行状态机	现在在浏览器窗口中找到工件可执行状态机。
创建新部件	在按住 Ctrl 键的同时，将原始工件拖到您的图表上。系统将提示您两个问题。 第一个答案是**物件**，第二个答案是**全部。**重工件属性并将其与原始属性区分开来，然后继续更改其属性。

可执行状态机工件

一个可执行状态机工件产生状态机交互的关键它指定将参与模拟的对象、它们的状态以及它们如何连接。 A较大的状态状态机可执行状态机是在一个工件中使用多个部件的多个工件来表示一个实例，因此您可以设置模拟每个状态机的多个实例并观察它们如何交互。示例示例可执行状态机帮助中提供了一个示例。

创建一个可执行状态机的属性

每个可执行状态机场景都涉及一个或多个状态机。包含的状态机由UML属性元素指定；每个属性都会有一个UML分类器（类）来确定该类型包含的状态机。作为多个属性包含的多种类型最终可以包含许多状态机，这些状态机都是在代码中创建并在执行时初始化的。

行动	描述
从浏览器可执行状态机>>的窗口上工件一个类	最简单的方法是在可执行状态机可执行状态机从类浏览器窗口中定义属性。在显示的对话框中，选择创建属性的选项。您可以指定一个名称来描述该属性可执行状态机将如何引用此属性。 注记：根据您的选择，您可能需要按住 Ctrl 键来选择创建属性。可以使用 按住 Ctrl 以显示此对话框"复选框随时更改此行为。
使用和连接多个UML属性	一个可执行状态机描述了多个状态机的交互。这些可以是同一状态机的不同实例，同一实例的不同状态机，或者来自不同基本类型的完全不同的状态机。要创建多个属性，将使用相同的状态在相同的状态机上类工件。要使用不同的类型，请根据需要从浏览器窗口中删除不同的类。

定义属性的初始状态

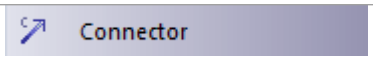
运行的状态机，可执行状态机在上下文运行的情况下类。可执行状态机允许您通过将属性值分配给各种类属性来定义每个实例的初始状态。例如，如果这些属性与正在运行的场景相关，您可以指定玩家的年龄、身高、体重或类似属性。通过这样做，可以设置详细的初始条件，这些条件将影响场景的发展方式。

行动	描述
设置属性值对话框	可以通过右键单击属性并选择 特征|"来打开用 分配属性值的对话框。设置属性值'，或使用键盘快捷键 Ctrl+Shift+R。
赋值	设置属性值"对话框允许您为原始类中定义的任何属性定义值。为此，请选择变量，将运算符设置为 ="并输入所需的值。

定义属性之间的关系

除了要分配给每个属性所拥有的值之外，属性可执行状态机还允许您根据其他属性如何引用它们作为实例的类模型来定义每个属性的引用方式。

行动	描述
创建连接器	使用复合工具箱中的连接器关系连接多个属性。

	 或者，使用快速链接器在两个属性之间创建关系并选择 连接器”作为关系类型。
映射到类模型	一旦两个属性之间存在连接器，您可以将其映射回它在类模型中表示的关联。为此，请选择连接器并使用键盘快捷键 Ctrl+L。将显示 选择一个关联”对话框，它允许生成的状态机在执行期间向填充关系中指定的角色的实例发送信号。

可执行状态机代码生成

为一种可执行状态机生成的代码是基于它的属性。这可能是Java 、C、C++、C# 或JavaScript 。无论是哪种语言， Enterprise Architect都会生成适当的代码，这些代码可以立即构建和运行。在运行它之前不需要手动干预。事实上，在初始生成后，任何一个可执行状态机都可以通过点击按钮生成、构建和执行。

支持的语言

可执行状态机支持以下平台语言的代码生成：

- 微软本机 C/C++
- 微软.NET (C#)
- 脚本(JavaScript)
- 甲骨文Java （ Java ）

从Enterprise Architect Release 14. 1开始，支持代码生成而不依赖于模拟环境（编译器）。例如，如果您没有安装 Visual Studio，您仍然可以从模型生成代码并在您自己的项目中使用它。如果您想在Enterprise Architect中模拟模型，仍然需要编译器。

仿真环境（编译器设置）

如果您想在Enterprise Architect中模拟可执行状态机模型，以下语言需要这些平台或编译器：

语言平台	框架路径示例
微软本机 (C/C++)	C:\Program 文件 (x86)\Microsoft Visual Studio 12.0 C:\Program Files (x86)\Microsoft Visual Studio\2017\专业(或其他版本)
微软.NET (C#)	C:\窗口\Microsoft.NET\Framework\v3.5（或更高版本）
脚本(JavaScript)	A N
甲骨文Java （ Java ）	C:\Program Files (x86)\ Java \jdk1.7.0_17（或更高版本）

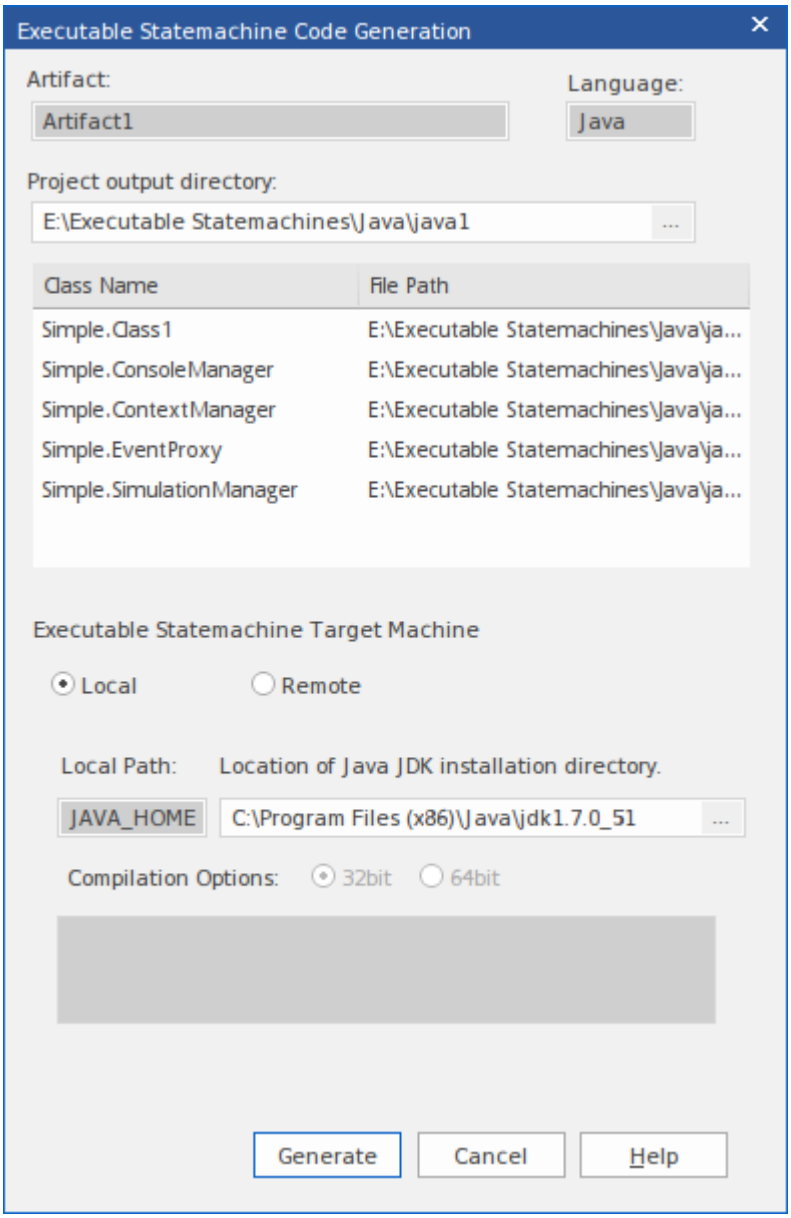
访问

功能区	仿真> 可执行状态> 状态机>生成、编译和运行或 仿真> 可执行状态> 状态机>生成

生成代码

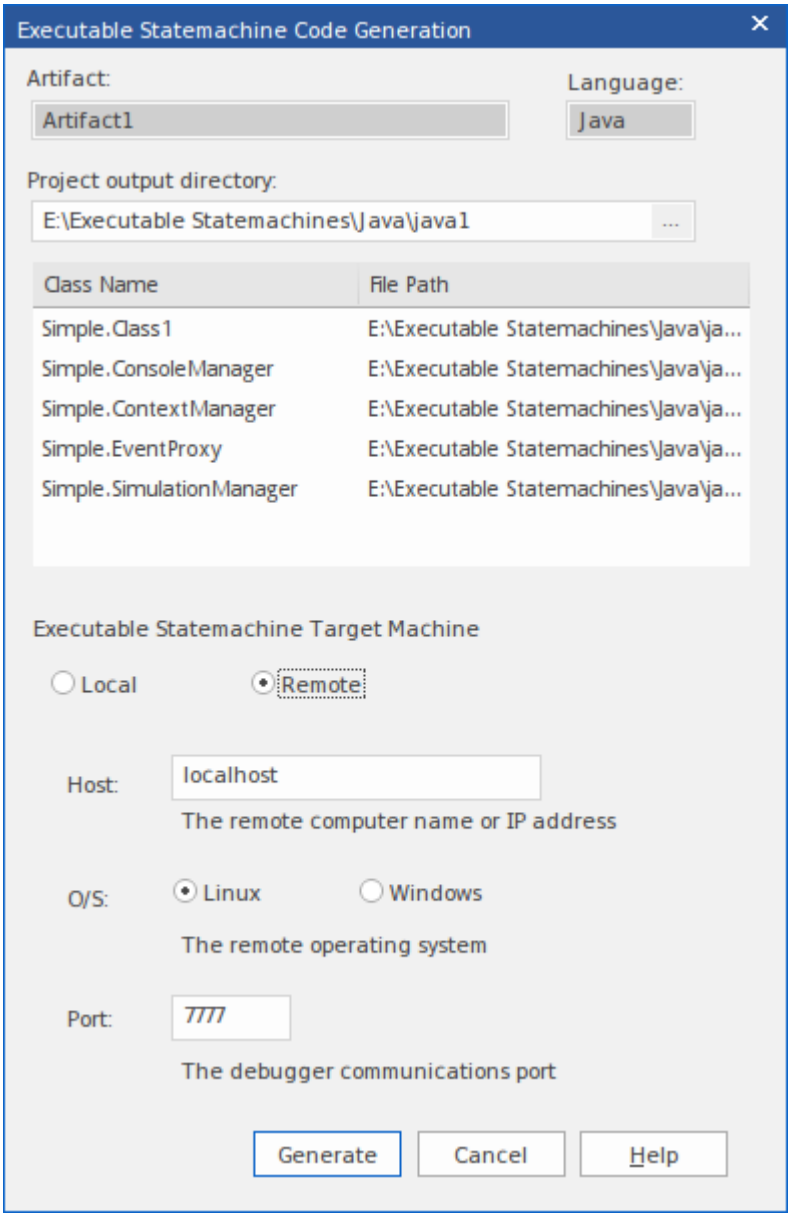
仿真> 可执行状态 > 状态机”功能区选项提供了用于为状态机生成代码的命令。工件选择可执行状态机，然后使用功能区选项生成显示的 可执行状态机代码生成”对话框取决于代码语言。

生成代码（ Java on窗口 ）



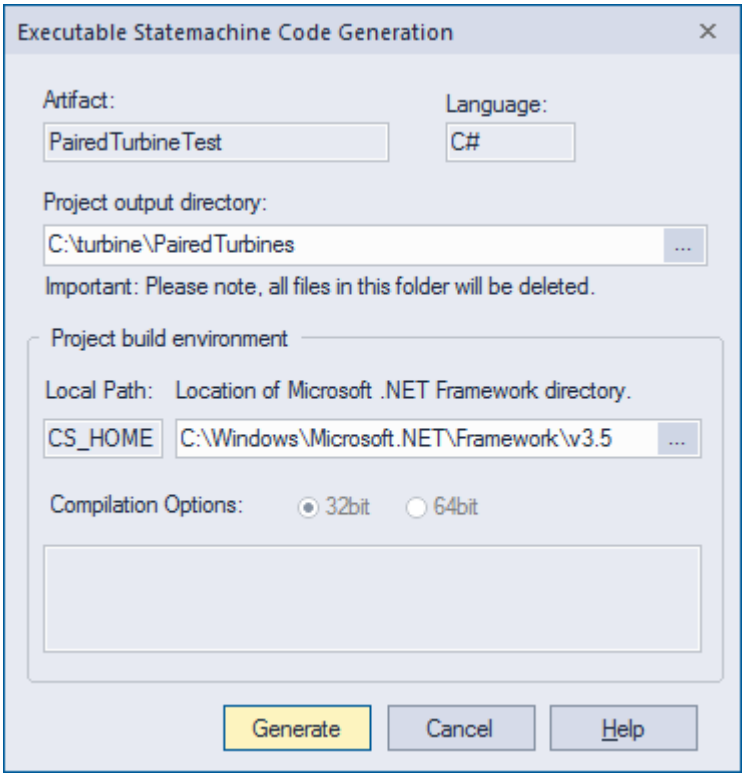
项目输出目录	显示将存储生成的代码文件的目录。如有必要，单击字段右侧的按钮以浏览并选择不同的目录。生成的类的名称和它们的源文件路径显示在这之后。
可执行状态机目标机器	选择 本地”选项。
Java JDK	输入要使用的Java JDK的安装目录。

生成代码（ Linux 上的Java ）



项目输出目录：	显示将存储生成的代码文件的目录。如有必要，单击字段右侧的按钮以浏览并选择不同的目录。路径更改时会显示生成的类的名称及其源文件路径。
可执行状态机目标机器	选择 远程”选项。
系统	选择 Linux。
端口	这是要使用的调试器端口。您将在分析器脚本生成的 调试”和 分析器”部分找到对这个端口号的引用。

生成代码（其它语言）



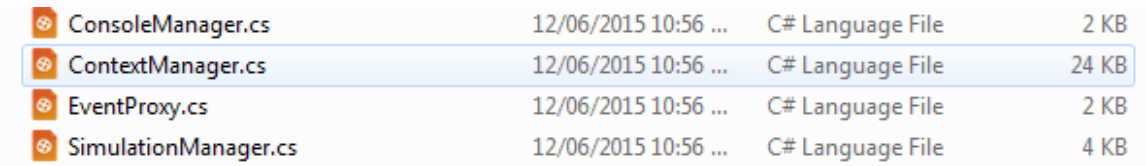
同时在 系统输出可执行状态机输出"页面打开窗口，在代码生成过程中显示哪些进度信息、警告或错误输出。

在 可执行状态机"字段和 属性名称"对话框中的 代码工件"字段显示元素的 属性"对话框中元素的 属性名称"和编码语言。

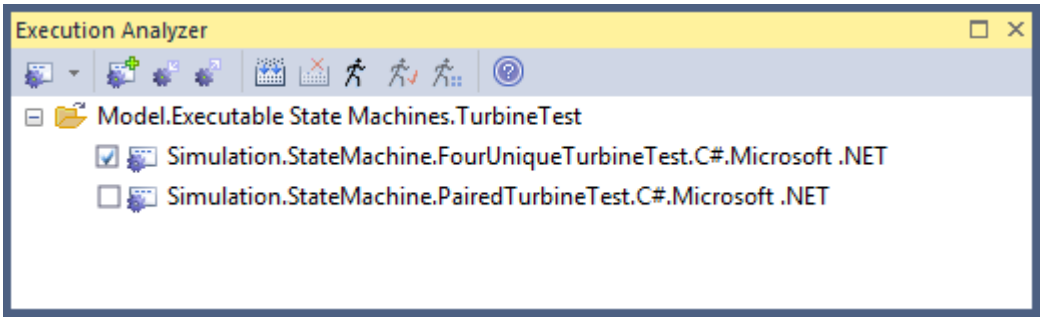
字段/选项	描述
项目输出目录	显示将存储生成的代码文件的目录。如有必要，单击字段右侧的按钮以浏览并选择不同的目录。
项目构建环境	此面板中的字段因信息元素和脚本中定义的工件而异。但是，每种支持的语言都提供了一个选项来定义构建和运行生成的代码所需的目标框架的路径；示例显示在本主题的支持的语言部分。 此路径的本地路径 ID 是在 本地路径"对话框中定义的，此处显示在 状态机代码生成"对话框和 可执行状态机代码生成"对话框中。

生成

点击此按钮生成状态机代码。代码生成将覆盖项目输出目录中的任何现有文件。文件集将包括所有必需的文件，包括状态机引用的每个类的文件。



每个可执行状态机还会生成一个执行分析器脚本，该脚本是用于构建、运行和调试运行可执行状态机的配置脚本。



建造代码

Enterprise Architect可执行状态机的代码可以通过以下三种方式之一进行构建。

方法	描述
功能区生成、编译和运行命令	再次对选中的生成可执行状态机，生成整个代码库。然后编译源代码并开始模拟。
功能区编译命令	编译已生成的代码。这可以在生成代码后直接使用，如果您对构建过程（分析器脚本进行了更改或以某种方式修改了生成的代码。
执行分析器脚本	生成的执行分析器脚本包含一个构建源代码的命令。这意味着当它处于活动状态时，您可以直接使用内置快捷键 Ctrl+Shift+F12 进行构建。
编译输出	构建时，所有输出都显示在系统输出窗口的 编译”页面上。您可以双击任何编译器错误以在相应行打开源编辑器。

利用现有代码

由Enterprise Architect生成并执行的可执行状态机可以利用没有类模型的现有代码。为此，您将创建一个抽象类元素，仅命名要在外部代码库中调用的操作。然后，您将在此接口和状态机类之间创建一个脚本，在分析器中手动添加所需的链接。对于Java ，您可以将 .jar 文件添加到类路径。对于本机代码，您可以将 .dll 添加到链接中。

可执行状态机的调试执行

可执行状态机的创建甚至在代码生成之后也提供了好处。使用执行分析器， Enterprise Architect能够连接到生成的代码。因此，您可以直观地调试和验证代码的正确行为；从您的状态机生成的完全相同的代码，由模拟演示并最终整合到现实世界的系统中。

调试状态机

能够调试可执行状态机会带来额外的好处，例如能够：

- 中断仿真的执行和所有正在执行的状态机
- 视图模拟中涉及的每个状态机实例的原始状态
- 在任意时间点视图源代码和调用堆栈
- 通过在源代码行上放置跟踪点来跟踪有关执行状态的其他信息
- 通过使用操作点和控件来执行控件（例如，错误时中断）
- 诊断由于代码或建模更改引起的行为变化

如果你已经生成、构建和运行成功了一个可执行状态机，你就可以调试它了！生成过程中创建的分析器脚本配置为提供调试功能。要开始调试，只需使用仿真控件开始运行可执行状态机控件。然而，根据被调试行为的性质，我们可能会先设置一些断点。

在状态转换中中断执行

像任何调试器一样，我们可以使用断点来检查代码中某个点的执行状态机。在图表或浏览器窗口中找到感兴趣的类，然后按 F12 查看源代码。从生成期间使用的命名约定很容易找到状态转换的代码。如果您想在特定转换处中断，请在编辑器中找到转换函数，然后通过单击函数内一行的左边距来放置断点标记。当你运行可执行状态机时，调试器会在这个状态机停止，你可以查看任何涉及到的状态机的原始状态。

有条件地中断执行

每个断点都可以采用条件和跟踪语句。当遇到断点并且条件评估为True时，执行将停止。否则执行将照常继续。您可以使用原始变量的名称构成条件，并使用标准相等操作数比较它们：< > = >= <=。例如：

(this.m_nCount > 100) 和 (this.m_ntype == 1)

要将条件添加到您设置的断点，请右键单击断点并选择 厲性”。按住 Ctrl 键的同时单击断点，可以快速编辑属性。

追踪辅助信息

可以使用 TRACE 子句从状态机本身内部跟踪信息，例如效果。调试还提供称为跟踪点的跟踪特征。这些只是断点，而不是中断，在遇到它们时打印跟踪语句。输出显示在仿真控件窗口中。它们可以用作诊断辅助来显示和序列事件的顺序以及实例更改状态的顺序。

观看调用堆栈

每当遇到断点时，调用堆栈序列都可以使用 .使用分析器来确定执行的位置

可执行状态机的执行与仿真

Enterprise Architect的众多特征之一是其执行模拟的能力。 Enterprise Architect ，可以挂接到可执行状态机器仿真，直观地展示状态机工件功能实时执行。

开始仿真

仿真控件提供了一个搜索按钮，您可以使用该工具工件来运行可执行状态机。控件维护一个最近可执行状态机下拉列表供您选择。您也可以使用上下文工件的可执行状态机菜单来启动模拟。

控制速度

仿真控件提供了速度设置。您可以使用它来调整模拟执行的速率。速度表示为 0 到 100 之间的值（值越大越快）。 A值将导致模拟在每一步后停止；这需要使用工具栏控件手动逐步完成模拟。

活动状态的符号

随着可执行状态机的执行，显示相关状态机图。显示在每个步骤到完成周期结束时更新。您会注意到，仅突出显示完成步骤的实例的活动状态。其他州仍然黯淡无光。

很容易识别哪个实例处于哪个状态，因为状态标有当前处于该特定状态的任何实例的名称。如果相同类型的两个或多个工件状态将具有相同的属性状态，则每个属性名称都有一个单独的标签。

生成计时图表

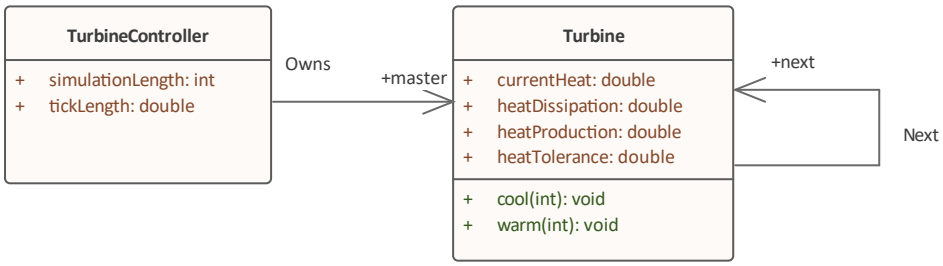
完成计算机的可执行状态机后，可以从输出中生成时序图。去做这个：

在仿真窗口工具栏中，单击 "工具|生成图表"。

示例：可执行状态机

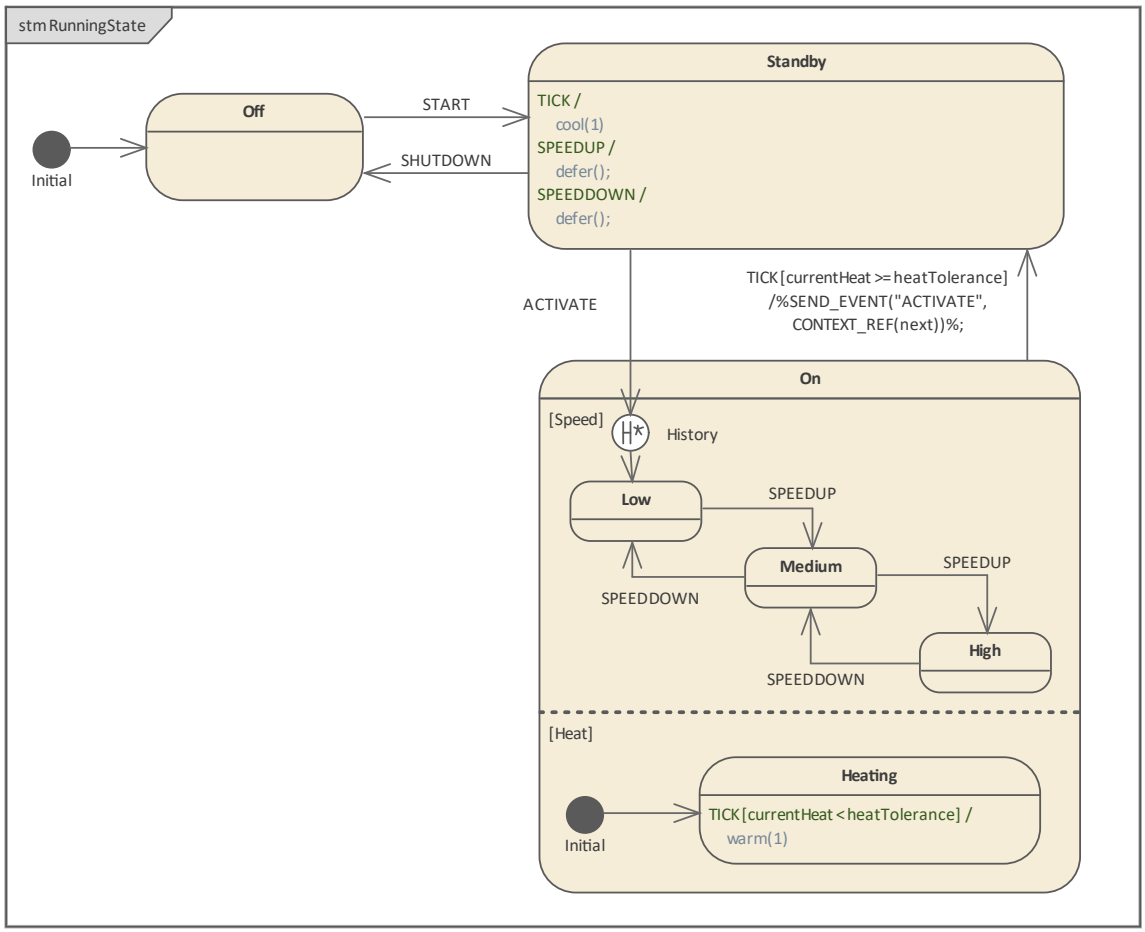
示例类模型

此图显示了本主题中描述的状态机使用的示例类模型。

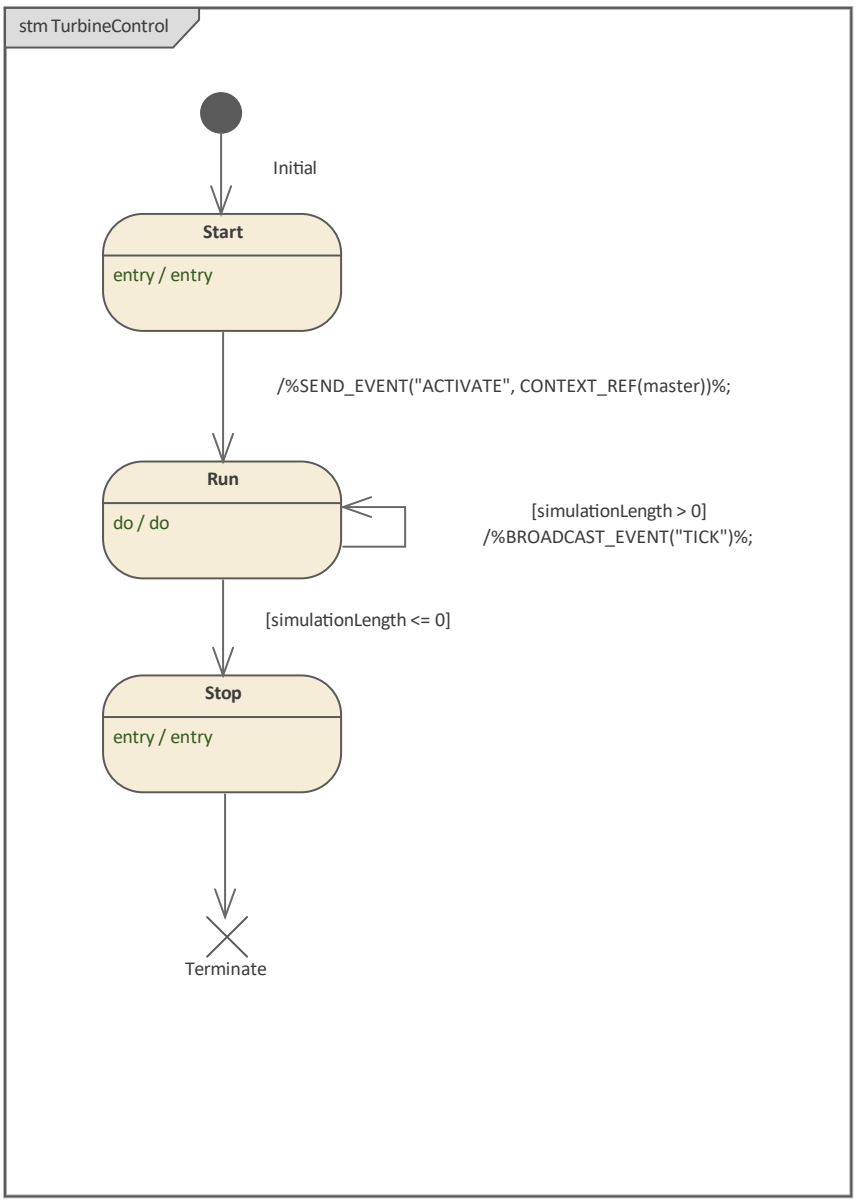


示例状态机

这两张图展示了两种状态机的定义。第一个引用另一个相同类型的状态机，而第二个驱动第一个存在的任何实例。

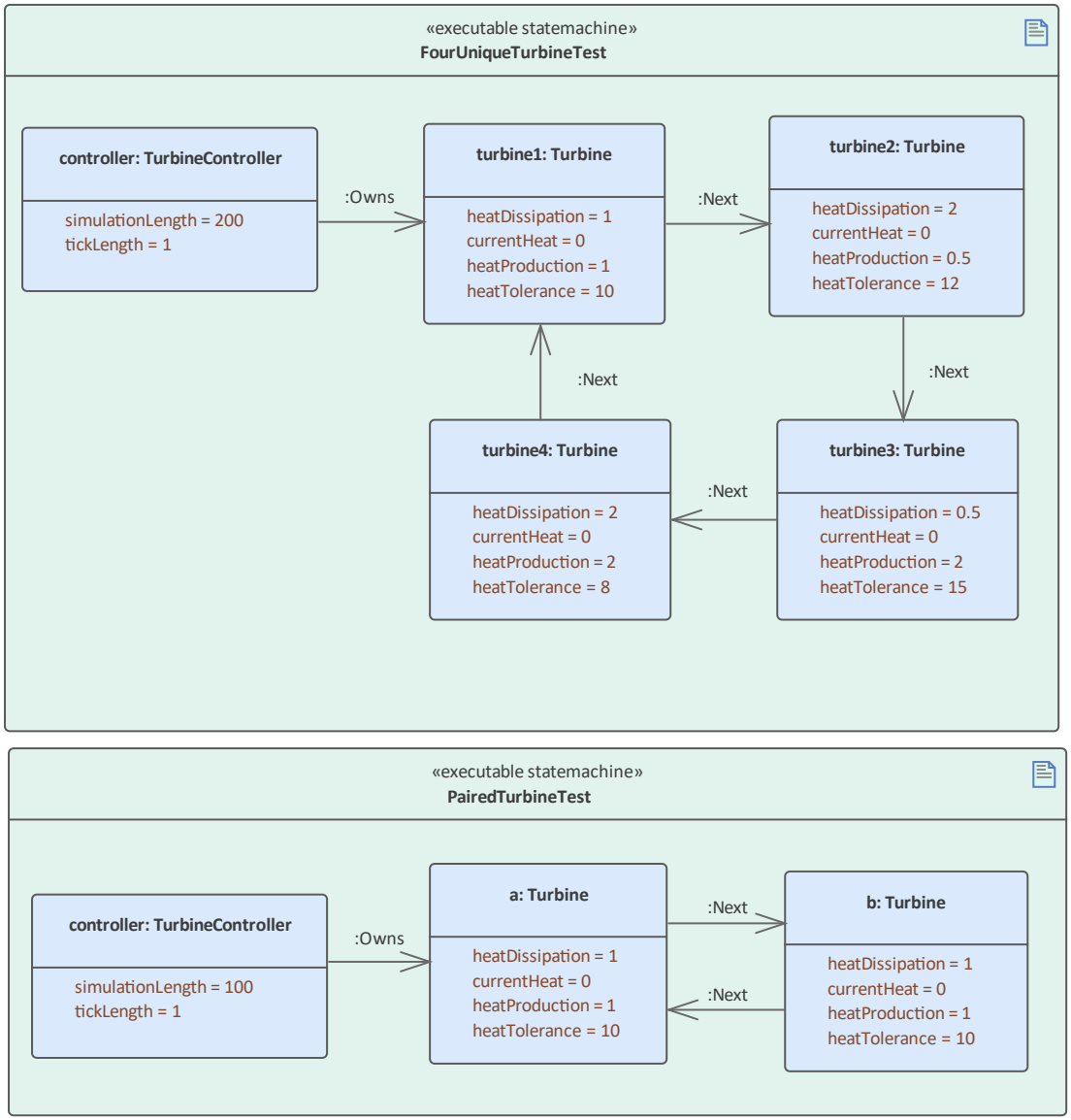


顶级控制器。



示例工件

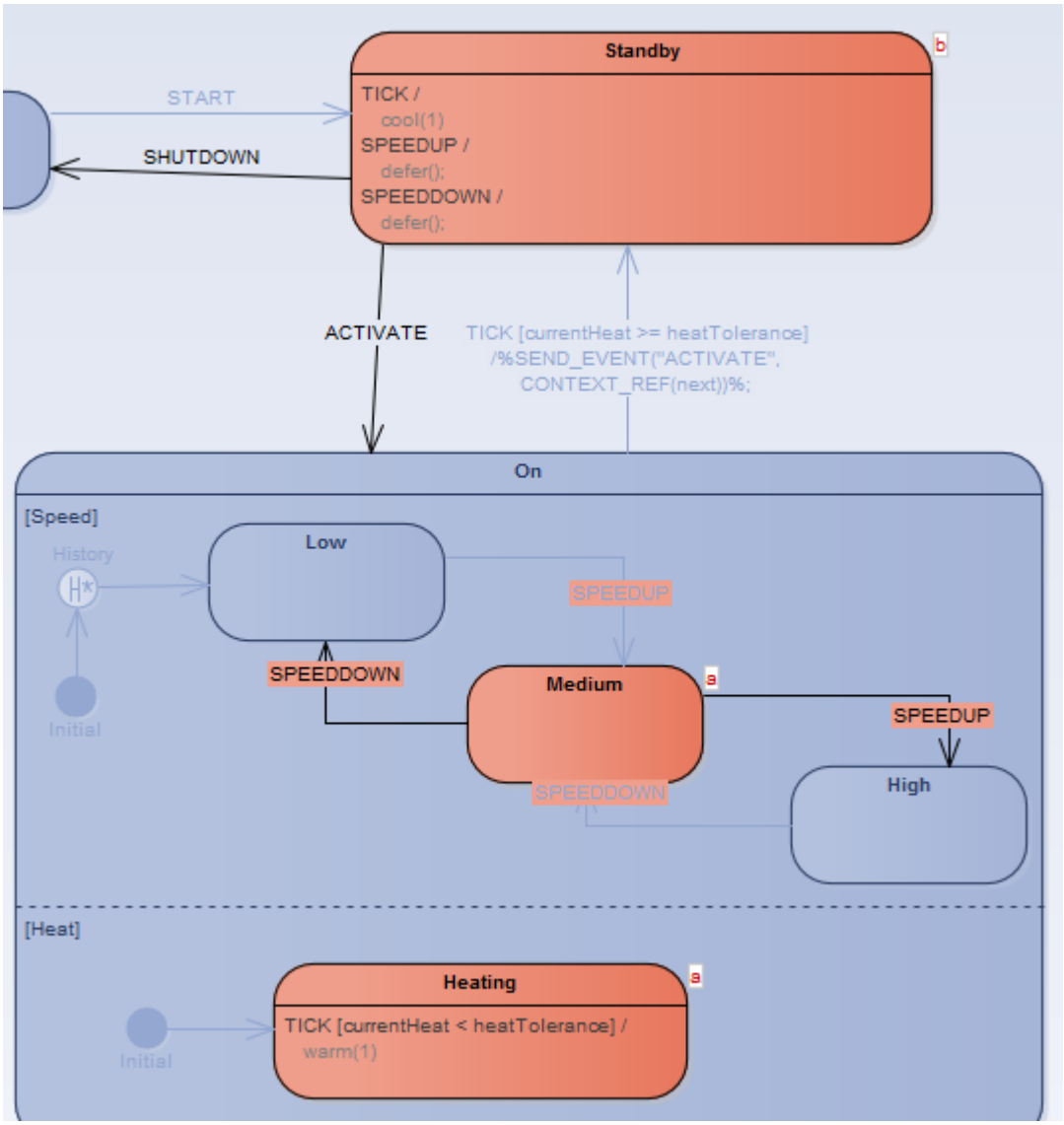
从示例类和状态机图中，我们可以创建如下所示可执行状态机。



注记如何为每个属性设置属性值，元素之间的链接标识类模型中存在的关系。

仿真结果

运行模拟时， Enterprise Architect将突出显示任何状态机中的当前活动状态。在一个状态机的多个实例存在的情况下，它还将显示该状态下每个实例的状态。


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

示例：仿真命令

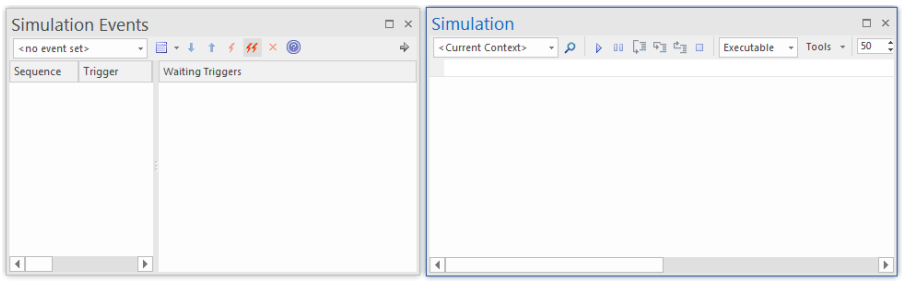
这个例子演示了我们如何使用仿真窗口来观察跟踪消息或发送命令来控制状态机。通过示例，您可以检查：

- 一个上下文的属性——类中定义的成员变量，即状态机的上下文；这些属性在上下文范围内携带值，用于所有状态行为和过渡效果，以访问和修改
- 一个信号的每个属性——信号中定义的成员变量，被一个事件引用，可以作为一个事件参数；每个信号事件出现可能有一个信号的不同实例
- 使用'Eval'命令查询上下文属性的运行时值
- 使用 "转储"命令转储当前状态的活动计数；它还可以转储池中延迟的当前事件

此示例取自 EAExample模型：

示例模型.模型仿真.可执行状态机.仿真命令

访问

功能区	• 仿真>动态仿真仿真> 模拟器> 打开仿真窗口） • 仿真>动态仿真>事件（用于仿真事件窗口）  这两个窗口在可执行状态机的模拟中经常一起使用。

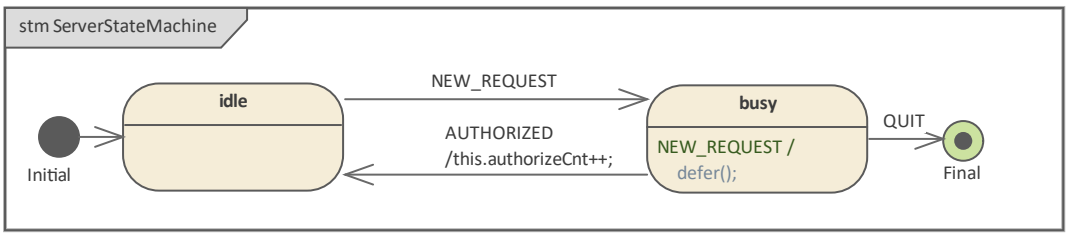
创建上下文和状态机

在本节中，我们将创建一个称为 TransactionServer 的类，它将状态机定义为其行为。然后我们创建一个可执行状态机工件模拟环境。

创建状态机的Context

1. 创建一个名为*TransactionServer*的类元素。
2. 在这个类中，创建一个名为*authorizeCnt*的属性，初始值为 0。
3. 在浏览器窗口中，右键单击*TransactionServer*并选择 "添加|状态机"选项。

创建状态机



1. 创建一个名为*Initial*的 Initial 伪状态。
2. 转移
   到了一种状态空闲的状态。
3. 转移
   到一个称为忙的状态，触发状态。
4. 转移
   ：
   - 到一个称为终点的终点伪状态，使用触发器 QUIT
   - 回到空闲状态，触发 AUTHORIZED，影响'影响++;'

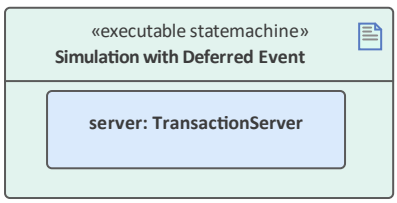
为**忙状态创建一个延迟事件**

1. 为忙碌划自我过渡。
2. 更改向 内部"过渡的 种类"。
3. 简单地触发器要延迟的事件。
4. 在 影响"字段中，输入 "defer();"。

创建信号和属性

1. 创建一个称为*RequestSignal*的信号元素。
2. 创建一个名为*requestType*的属性，类型为 "int "。
3. 配置事件配置以引用*RequestSignal*。

创造可执行状态机工件



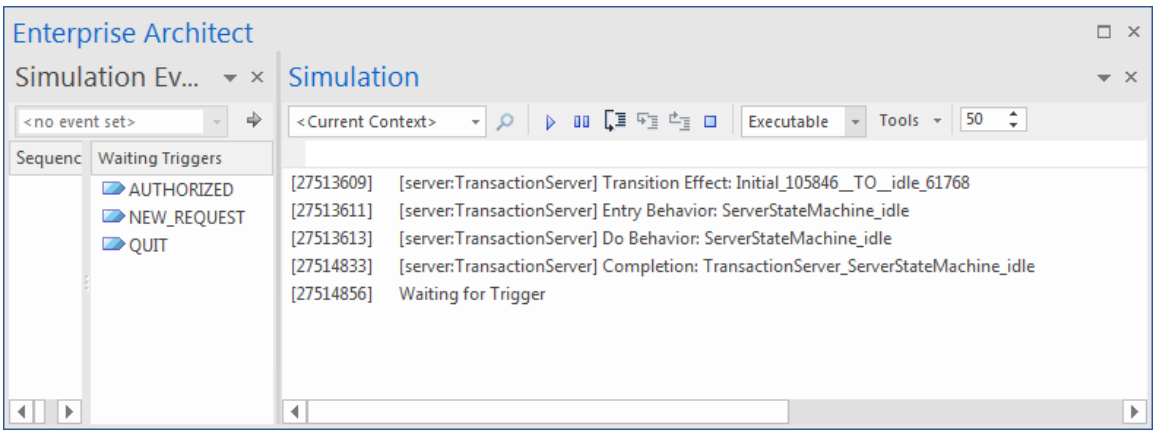
1. 在工具箱的页面中，将 无素仿真可执行状态机"图标图表图表上，并调用仿真事件。
2. Ctrl 从浏览器窗口拖动事务元素并将其工件到作为属性服务器的名称服务器上。
3. 将设备的JavaScript工件例如；在生产中，您还可以使用 C、C++、C# 或Java ，它们也支持可执行状态机）。
4. 点击工件仿真> 执行状态 > 执行状态 > 状态机生成,编译和运行功能区选项

仿真窗口和命令

模拟开始时， *idle*是当前状态。

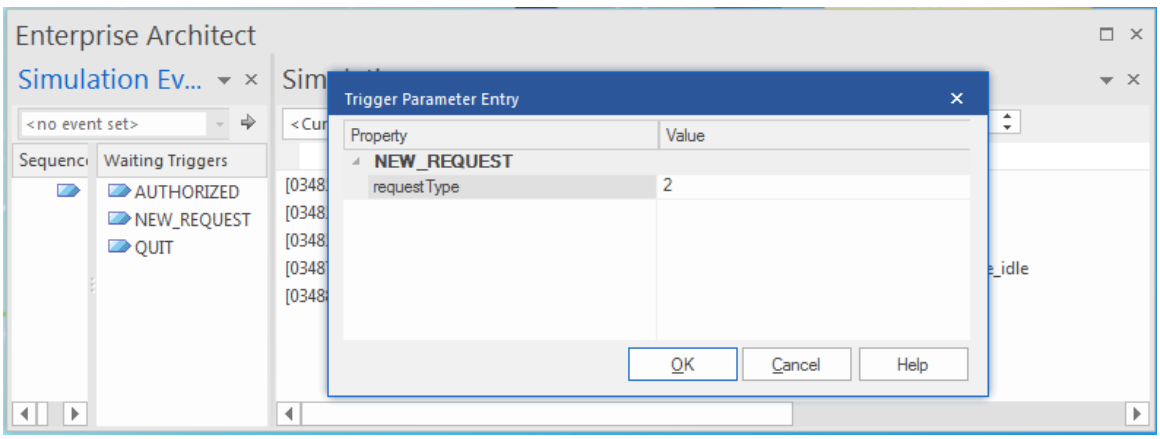
仿真窗口显示，转移

状态空闲的影响、进入和执行行为已完成，状态机正在等待触发。



通过信号属性值的事件数据

对于简单的信号事件触发器， 触发器参数Entry”对话框将显示用 提示在信号请求信号中定义的列出的属性的值，由信号引用。



类型值 2”并单击确定按钮。然 将信号属性值传递给调用的方法，例如状态的行为和转移的影响。

这些消息输出到仿真窗口：

[触发器] 等待简单

[03611358] 命令：**广播 NEW_REQUEST.RequestSignal(2)**

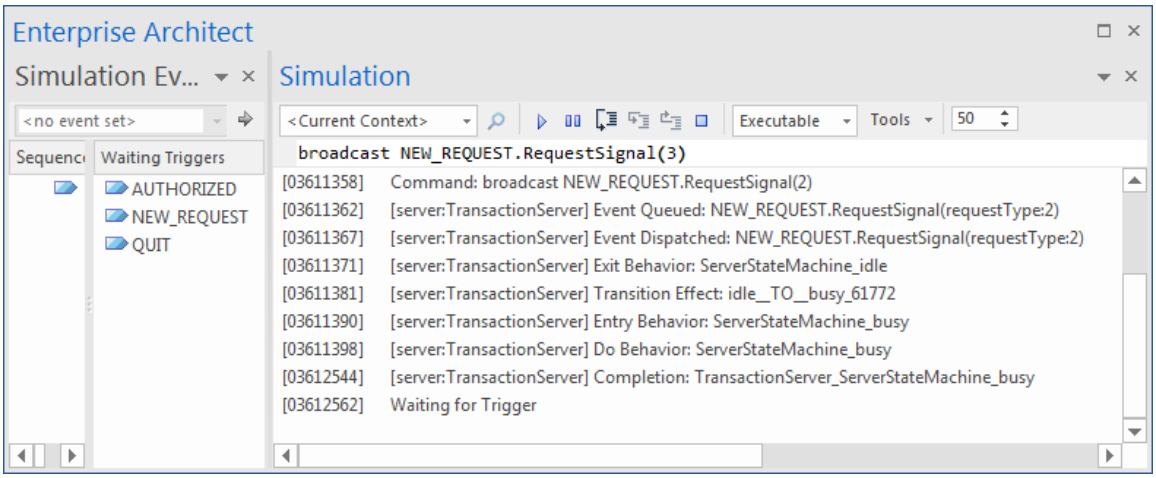
[03611362] [服务器：TransactionServer]事件排队：NEW_REQUEST.RequestSignal（requestType：2）

[03611367] [服务器：TransactionServer]事件调度：NEW_REQUEST.RequestSignal（requestType：2）

[03611371] [server:TransactionServer] 退出行为：ServerStateMachine_idle

[03611381] [服务器：TransactionServer]转移
影响：影响
[03611390] [server:TransactionServer] 入口行为：ServerStateMachine_busy
[03611398] [server:TransactionServer]行为：ServerStateMachine_busy
[03612544] [服务器：TransactionServer] 完成：TransactionServer_ServerStateMachine_busy
[触发器] 等待简单

我们可以通过双击仿真事件窗口中列出的项目来播放事件。或者，我们可以在仿真窗口（工具栏下方）的文本字段中键入命令string 。

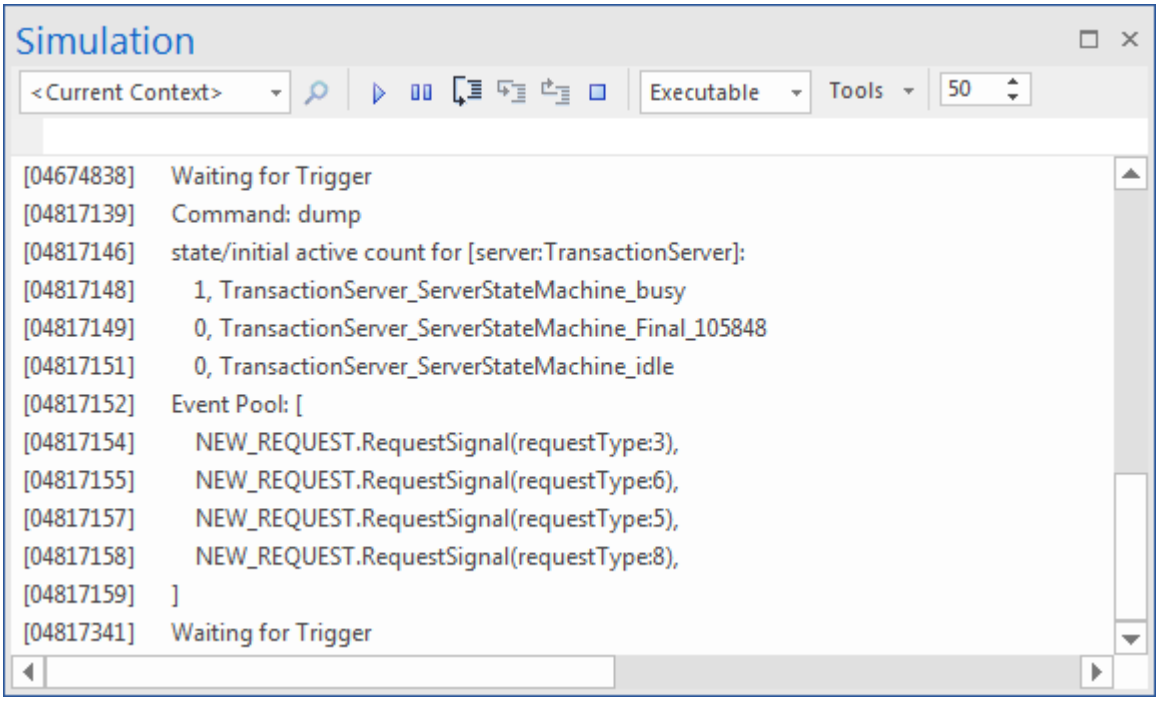


[触发器] 等待简单
[04460226] 命令：**广播 NEW_REQUEST.RequestSignal(3)**
[04460233] [服务器：TransactionServer]事件排队：NEW_REQUEST.RequestSignal（requestType：3）
[触发器] 等待简单

仿真事件消息表明事件发生被推迟（排队，但未调度）。我们可以使用文本字段运行更多命令：
[触发器] 等待简单
[04664057] 命令：**广播 NEW_REQUEST.RequestSignal(6)**
[04664066] [服务器：TransactionServer]事件排队：NEW_REQUEST.RequestSignal（requestType：6）
[触发器] 等待简单
[04669659] 命令：**广播 NEW_REQUEST.RequestSignal(5)**
[04669667] [服务器：TransactionServer]事件排队：NEW_REQUEST.RequestSignal（requestType：5）
[触发器] 等待简单
[04674196] 命令：**广播 NEW_REQUEST.RequestSignal(8)**
[04674204] [服务器：TransactionServer]事件排队：NEW_REQUEST.RequestSignal（requestType：8）
[触发器] 等待简单

dump:查询状态和事件的 活动泳池”

文本字段中的类型转储；这些结果显示：



从 活动计数”部分，我们可以看到忙碌是活动状态（活动计数为1 ）。

提示：对于复合状态，活动计数为1 （对自身）加上活动区域的数量。

从 事件泳池”部分，我们可以看到事件队列中有四个事件发生。信号的每个实例都携带不同的数据。池中事件的顺序是它们被广播的顺序。

eval：查询运行时间Context的值

触发器授权，

[触发器] 等待简单

[05494672] 命令：广播已授权

[05494678] [服务器：TransactionServer]事件排队：已授权

[05494680] [服务器：TransactionServer]事件调度：授权

[05494686] [server:TransactionServer] 退出行为：ServerStateMachine_busy

[05494686] [服务器：TransactionServer]**转移**

影响：忙影响

[05494687] [server:TransactionServer] 入口行为：ServerStateMachine_idle

[05494688] [server:TransactionServer]行为：ServerStateMachine_idle

[05495835] [服务器：TransactionServer] 完成：TransactionServer_ServerStateMachine_idle

[05495842] [服务器：TransactionServer]**事件调度：NEW_REQUEST.RequestSignal（requestType：3）**

[05495844] [server:TransactionServer] 退出行为：ServerStateMachine_idle

[05495846] [服务器：TransactionServer]转移

影响：影响

[05495847] [server:TransactionServer] 入口行为：ServerStateMachine_busy

[05495850] [server:TransactionServer]行为：ServerStateMachine_busy

[05496349] [服务器：TransactionServer] 完成：TransactionServer_ServerStateMachine_busy

[触发器] 等待简单

- 从忙到空闲的转换已经完成，所以我们期望执行效果
- 空闲完成时从池中回调一个事件并调度，使忙碌状态变为活动状态
- 类型转储，注意池中还剩下三个事件；第一个被召回和发送

[泳池]事件泳池：[

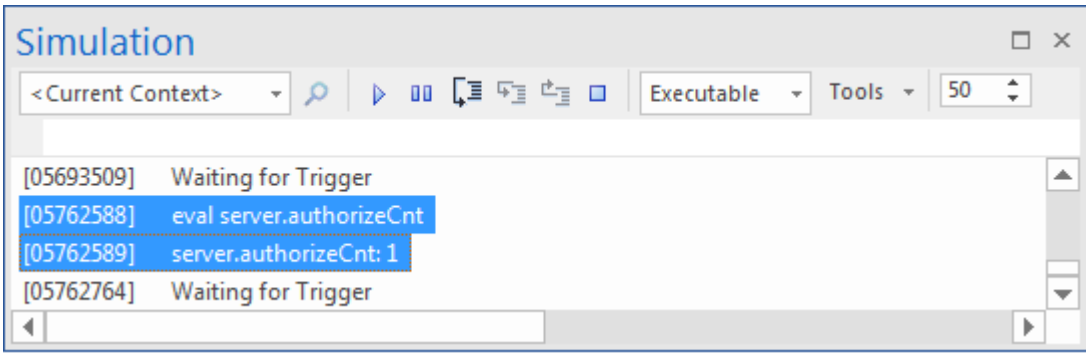
[05693349] NEW_REQUEST.RequestSignal(requestType:6),

[05693351] NEW_REQUEST.RequestSignal(requestType:5),

[05693352] NEW_REQUEST.RequestSignal(requestType:8),

[05693354]]

文本字段中的类型*eval server.authorizeCnt* 。该图表示 '运行' 的运行时间值为1 。



触发器再次授权。当状态机稳定在*busy*时，池中会剩下两个事件。再次运行*eval server.suthorizeCnt* ；该值为 2。

从状态行为和转移

访问上下文的成员变量转移

影响

Enterprise Architect的可执行状态机，支持模拟C、C++、C# Java和JavaScript 。

对于 C 和 C++，访问上下文的成员变量的语法与 C#、 Java和JavaScript不同。 C 和 C++ 使用指针 '->' 而其他的只是使用 '.'；但是，您始终可以使用*this.variableName*来访问变量。 Enterprise Architect将把它翻译成 this->variableName 用于 C 和 C++。

因此，对于所有语言，只需使用以下格式进行模拟：

this.variableName

例子：

在过渡效果中：

this.authorizeCnt++;

在某些状态的进入、做或退出行为：

this.foo += this.bar;

注记：默认情况下，对于 C 和 C++， Enterprise Architect仅将 'this->' 替换为 'this'；例如：

this.foo = this.bar + myObject.iCount + myPointer->iCount;

将被翻译成：

this->foo = this->bar + myObject.iCount + myPointer->iCount;

支持的命令A完全列表

由于多个可执行状态机工件一起模拟一些上下文，因此可以指定一个实例名称。

运行状态机：
由于每个上下文可以有多个状态机， 迩行"命令可以指定一个状态机的机器。

- 运行instance.statemachine
- 运行
- 运行实例
- 运行
- 运行

例如：
运行
运行
运行服务器
运行

广播和发送事件：

- 广播事件字符串
- 将 EventString 发送到实例
- 发送EventString（相当于广播EventString）

例如：
广播事件1
将 Event1 发送给客户端

转储命令：

- 倾倒
- 转储实例

例如：
倾倒
转储服务器
转储客户端

评估命令：

- eval instance.variableName

例如：
评估 client.requestCnt
评估 server.responseCnt

退出命令：

- 出口

EventString 的格式：

- EventName.SignalName（参数列表）

注记：参数列表应与信号中**按顺序**定义的属性相匹配。

例如，如果信号定义了两个属性：

- 富
- 酒吧

那么这些 EventStrings 是有效的：

- Event1.Signal1(10, 5) --------- foo = 10;酒吧 = 5
- Event1.Signal1(10,) --------- foo = 10;酒吧未定义
- Event1.Signal1(,5) --------- bar = 10; foo未定义
- Event1.Signal1(,) --------- foo 和 bar 都没有定义

如果信号不包含任何属性，我们可以将 EventString 简化为：

- 事件名称

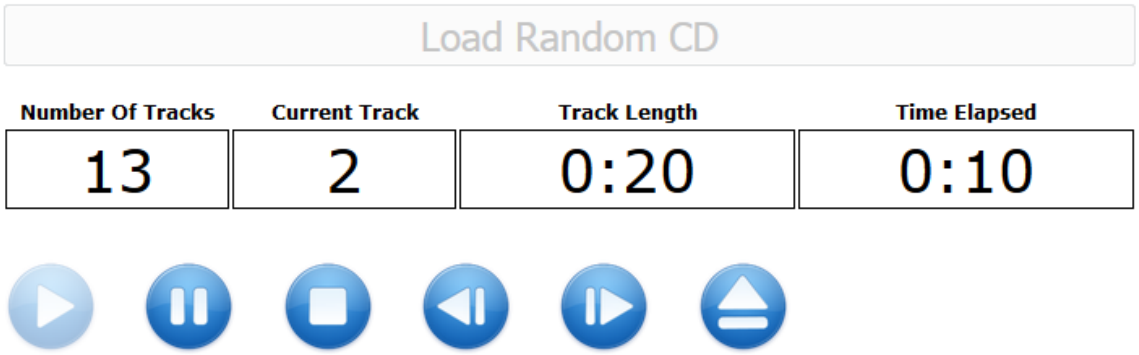
示例：在 HTML 中使用JavaScript进行仿真

我们已经知道用户可以模型模拟可执行状态机并在Enterprise Architect中使用生成的代码进行模拟。使用*CD* 播放器和正则表达式解析器这两个示例，我们现在将现在如何将生成的代码与实际项目集成。

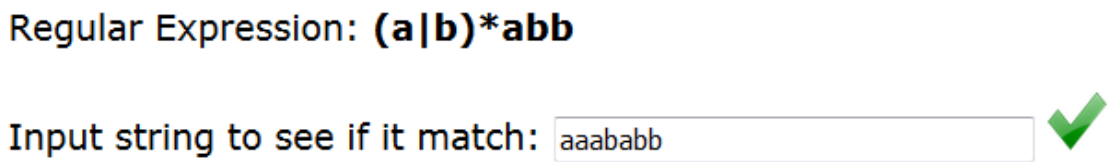
Enterprise Architect为客户端代码使用状态机提供了两种不同的机制：

- 状态- 客户端可以查询当前活动状态，然后根据查询结果 切换"逻辑
- Runtime Variable Based - 客户端不作用于当前活动状态，但作用于包含状态机的类中定义的变量的运行时值

在*CD*播放器的例子中，GUI上的状态很少，按钮很多，所以基于Active状态的例子很容易实现；我们还将查询当前轨道的运行时值。



在正则表达式解析器示例中，状态机处理所有事情，并且成员变量*bMatch*在状态更改时更改其运行时值。客户端不会注册有多少状态或当前处于活动状态的状态。



在这些主题中，我们将逐步演示如何为指定的正则表达式模型、模拟和集成 CD 播放器和解析器：

- CD Player
- Regular Expression Parser

激光唱机

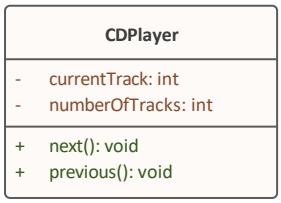
CD 播放器应用程序的行为可能看起来很直观；但是，有许多规则与按钮何时启用和禁用、窗口的文本字段中显示的内容以及向应用程序提供事件时发生的情况有关。

假设我们的示例 CD 播放器具有以下特征：

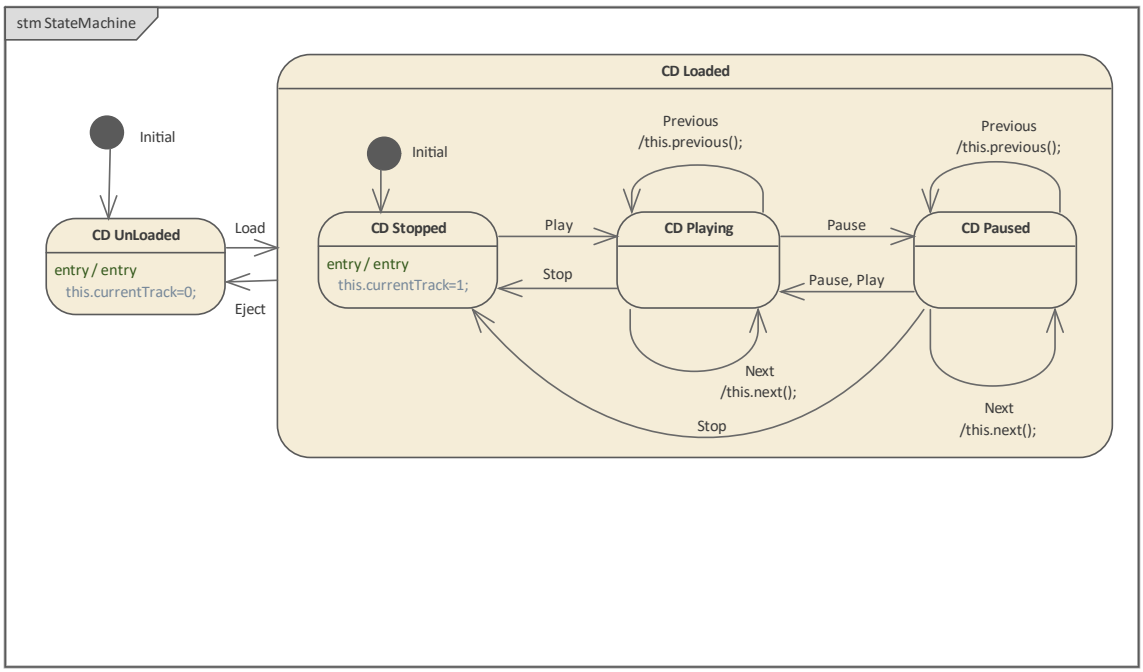
- 按钮 - 加载随机 CD、播放、暂停、停止、上一曲目、下一曲目和弹出
- 显示 - 曲目数、当前曲目、曲目长度和已播放时间

CD播放器状态机

A类*CDPlayer*定义了两个属性： *currentTrack*和*numberOfTracks* 。

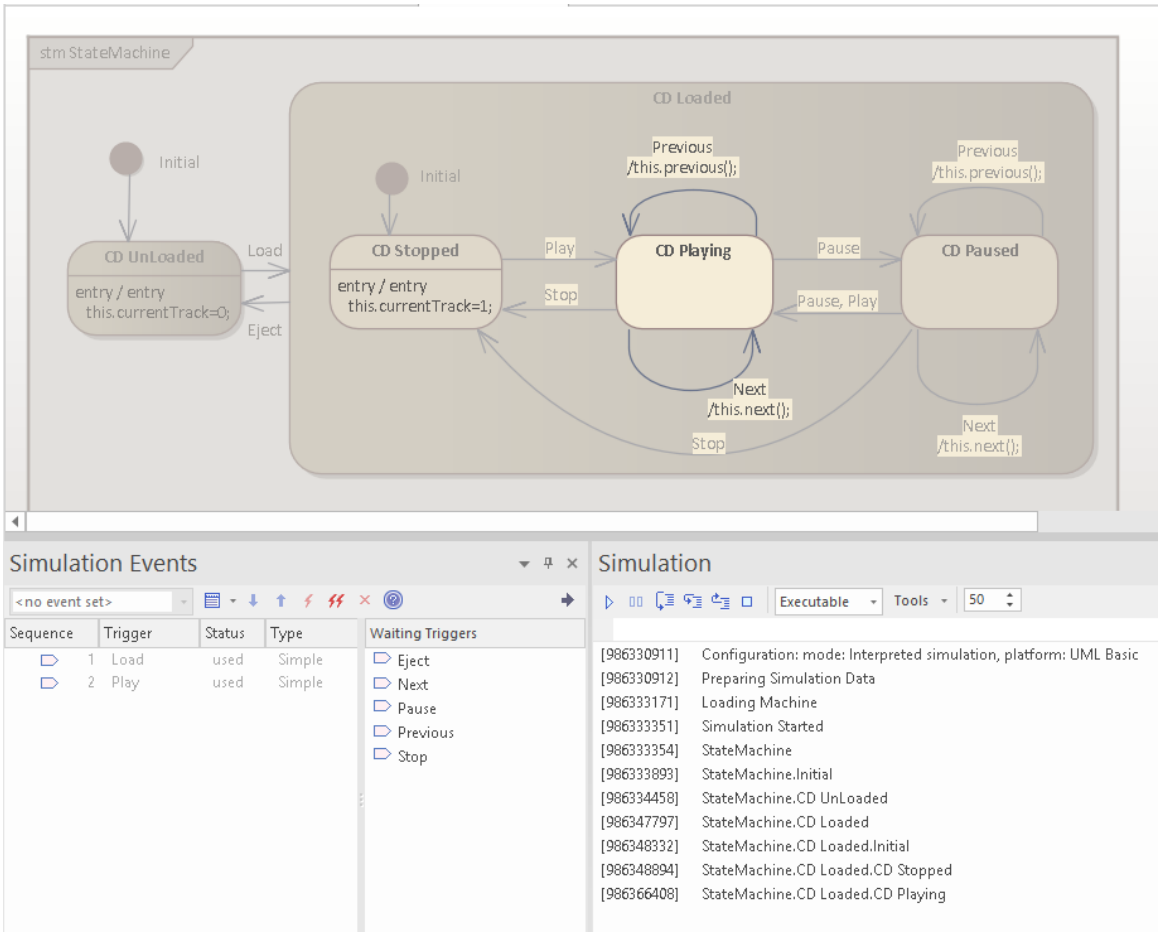


状态机用于描述 CD 播放器A状态：



- 在更高的层次上，状态机有两种状态： *CD UnLoaded*和*CD Loaded*
- *CD Loaded*可以由三个简单状态组成： *CD Stopped* 、 *CD Playing* 、 *CD Paused*
- 转换定义为事件 Load、Eject、Play、Pause、Stop、Previous 和 Next 的触发器
- 定义状态behaviors和效果状态定义的属性值；例如，'Previous' 事件将触发自转换（如果当前状态是*CD Playing*或*CD Paused* ）并执行影响，这将减少*currentTrack*的值或换行到最后一个轨道

我们可以创建一个状态状态机的属性，然后在Enterprise Architect可执行状态机中模拟该工件的类型，以确保该模型是正确的。



检查生成的代码

Enterprise Architect将在您指定的文件夹中生成这些文件：

- 后端代码：CDPlayer.js、ContextManager.js、EventProxy.js
- 客户端代码：ManagerWorker
- 前端代码：statemachineGUI.js、index.html
- 其它代码：SimulationManager.js

文件	描述
/CDPlayer.js	该文件定义了类CDPlayer 及其属性和操作。它还用状态行为和过渡效果定义了类的状态机。
/ContextManager.js	该文件是上下文的抽象管理器。该文件定义了与实际上下文无关的内容，这些内容是在*ContextManager*的泛化中定义的，例如*SimulationManager*和*ManagerWorker* 。 模拟（可执行状态机）可以涉及多个工件；例如，在网球比赛模拟中，将有一个裁判输入类*Umpire*，两个玩家——*playerA*和*playerB*——输入类。类和*ClassPlayer*类定义自己的状态机。
/EventProxy.js	该文件定义了仿真中使用的事件和信号。 如果我们用参数引发一个事件，我们模型事件为一个信号事件，它指定一个信号类；然后我们为信号类定义属性。每个事件发生都有一个信号实例，带

	有为属性指定的运行信号。
/SimulationManager.js	该文件用于Enterprise Architect中的模拟。
/html/ManagerWorker.js	该文件充当前端和后端之间的中间层。 • 前端发消息向ManagerWorker请求信息 • 由于ManagerWorker是从ContextManager泛化而来的，所以它可以完全访问所有上下文，例如查询当前活动状态和查询变量的运行时值 • ManagerWorker 将向前端发送一条消息，其中包含从后端检索到的数据
/html/statemachineGUI.js	该文件通过定义*stateMachineWorker*来建立前端和 ManagerWorker 之间的通信。它： • 定义函数*startStateMachineWebWorker*和*stopStateMachineWebWorker* • 使用占位符代码定义函数*onActiveStateResponse*和*onRuntimeValueResponse*： //to do: 写用户的逻辑 您可以简单地用您的逻辑替换此评论，本主题稍后将演示
/html/index.html	这定义了 HTML用户接口，例如用于引发事件或显示信息的按钮和输入。您可以在此文件中定义 CSS 和JavaScript 。

自定义 index.html 和 statemachineGUI.js

对生成的文件进行以下更改：

- 创建按钮和显示
- 创建 CSS 样式以格式化显示并启用/禁用按钮图像
- 创建一个 ElapseTimeWorker.js 以每秒刷新一次显示
- 创建一个TimeElapsed函数，当经过的时间达到轨道长度时设置为Next Track
- 创建JavaScript作为按钮 'onclick' 事件处理程序
- 广播事件后，请求状态的活动状态和运行时值
- 初始化时，请求活动状态

在 statemachineGUI.js 中找到函数*onActiveStateResponse_cdPlayer*

- 在 CDPlayer_StateMachine_CDUnLoaded 中，禁用所有按钮并启用 btnLoad
- 在 CDPlayer_StateMachine_CDLoaded_CDStopped 中，禁用所有按钮并启用 btnEject 和 btnPlay
- 在 CDPlayer_StateMachine_CDLoaded_CDPlaying 中，启用所有按钮并禁用 btnLoad 和 btnPlay
- 在 CDPlayer_StateMachine_CDLoaded_CDPaused 中，启用所有按钮并禁用 btnLoad

在 statemachineGUI.js 中找到函数*onRuntimeValueResponse*

- 在*cdPlayer.currentTrack*中，我们更新当前曲目和曲目长度的显示

完全的示例

通过单击此链接，可以从Sparx Systems网站的 资源"页面访问该示例：

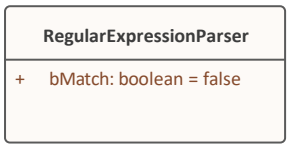
CD播放器仿真

单击加载随机 CD 按钮，然后单击开始仿真按钮。

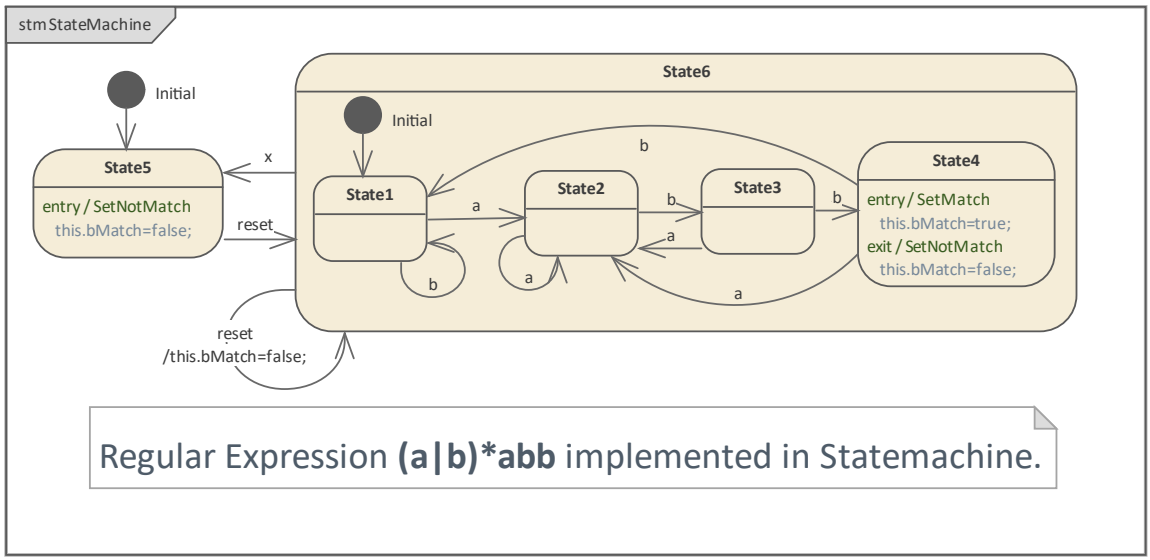
正则表达式解析器

正则表达式状态机解析器

类*RegularExpressionParser*定义了一个属性：bMatch。



状态机用于描述正则表达式 (A |b)*abb



- 转换触发器被指定为事件*a* 、 *b* 、 *x*和*reset*
- 在进入 State4 时，bMatch 设置为True ；从 State4 退出时，bMatch 设置为False
- 在进入 State5 时，bMatch 设置为False
- 在 State6 的自我转换中，bMatch 设置为False

自定义 index.html 和 statemachineGUI.js

对生成的文件进行以下更改：

- 创建一个 HTML 输入字段和一个图像来指示结果
- 创建JavaScript作为字段的*oninput*事件处理程序
- 创建函数 "SetResult "以切换通过/失败图像
- 创建函数 "getEventStr"，它将在 "a" 上返回 "a"，在 "b" 上返回 "b"，但在任何其他字符上返回 "x"
- 初始化时，广播 "重置"
- 在广播事件上，请求运行时变量 "regxParser.bMatch"

在*statemachineGUI.js*中，找到函数'onRuntimeValueResponse'。

- 在 "regxParser.bMatch" 中，我们将收到 "True "或 "False "并将其传递给 "SetResult"以更新图像

完全的示例

通过单击此链接，可以从Sparx Systems网站的 资源”页面访问该示例：
正则表达式解析器仿真

示例：进入状态

进入状态的状态取决于状态的状态和进入的方式。

在所有情况下，状态的进入行为在状态时执行（如果已定义），但仅在与传入转移

相关联的任何效果行为之后转移

完成了。此外，如果为状态定义了行为，则该行为在执行该条目行为之后立即开始执行。

对于定义了一个或多个区域的复合状态，每个区域存在许多替代方案：

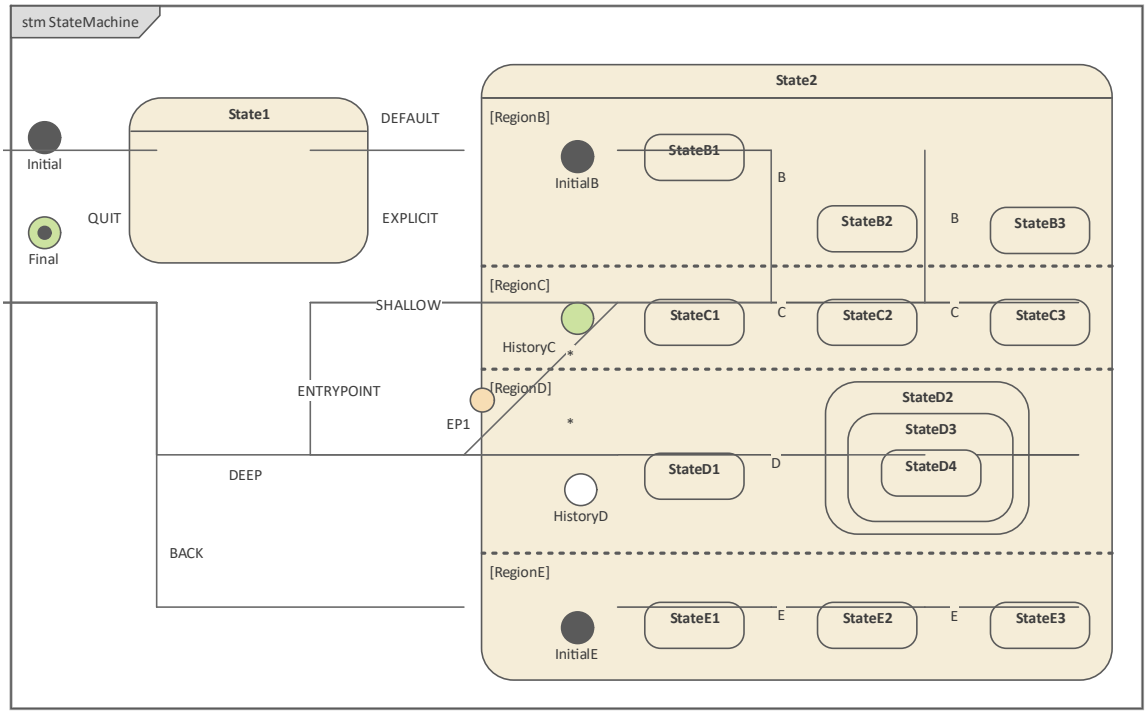
- 默认条目：当拥有复合状态是转移
  的直接目标时会发生这种情况转移
  ;在执行 entry行为并分叉一个可能的行为执行之后，状态entry 通过它的输出转移
  从一个初始的 Pseudostate 继续转移
  （称为默认转移
  的状态），如果它在区域中定义
  如果没有定义初始 Pseudostate，则此区域将不会处于活动状态
- 显式入口：如果传入转移
  或者它的延续终止于拥有复合状态的直接包含的子态状态，然后该子态状态变为活动状态，并且在包含复合状态的行为执行之后执行其条目行为
  如果转移
  ，则此规则递归适用转移
  以间接（深度嵌套）子态终止
- 浅历史入口：如果传入转移
  终止于该区域的一个区域Pseudostate，激活的子态状态成为在该进入之前最近处于活动状态的子态状态（除了 FinalState），除非这是进入该状态的第一个条目；如果它是第一个进入这个状态的条目，之前的条目已经到达了一个终点或一个默认的浅历史转移
  如果定义了就取，否则应用默认状态入口
- 深度历史条目：这种情况的规则与浅层历史的规则相同，只是目标伪状态的类型为 deepHistory 并且该规则递归地应用于此之下的活动状态配置中的所有级别
- 入口点入口：如果一个转移
  通过一个状态Pseudostate进入拥有复合状态，然后向外转移
  取自入口点，渗透到该区域内的状态；如果从入口点有更多的传出转换，则每个转移
  必须定位到不同的区域并且所有区域同时激活

对于具有多个区域的正交状态，如果转移

显式输入一个或多个区域（在分叉或入口点的情况下），这些区域显式输入，其他区域默认输入。

在这个例子中，我们演示了一个具有正交状态的所有这些入口行为的模型。

建模状态机



状态机的上下文

1. 创建一个名为*MyClass*的类元素，作为状态机的上下文。
2. 在浏览器窗口中右键单击*MyClass*并选择 添加|状态机"选项。

状态机

1. 在图中添加一个*Initial*节点、一个名为状态的状态、一个名为 *State2*的状态和一个名为 final 的终点元素。
2. 放大图上的*State2* ，右键单击它并选择'高级|定义 Concurrent Substates 的选项，定义*RegionB*、*RegionC*、*RegionD*和*RegionE*。
3. 右键单击*State2*并选择 New子元素|入口"选项以创建入口*EP1* 。
4. 在*RegionB*中，创建元素*InitialB* ，过渡到*StateB1* ，过渡到*StateB2* ，过渡到*StateB3* ；由事件触发的所有转换*B*
5. 在*RegionC*中，创建元素 shallow *HistoryC* （右键单击 History 节点 |高级| Deep History | 取消选中），过渡到*StateC1* ，过渡到*StateC2* ，过渡到*StateC3* ；事件*C*触发的所有转换。
6. 在*RegionD*中，创建元素 deep *HistoryD* （右键单击 History 节点 |高级| Deep History | 勾选），过渡到*StateD1* ，创建*StateD2*作为 StateD3 的父级，它是*StateD4*的父级；从*StateD1*到*StateD4*的转换；由事件*D*触发。
7. 在*RegionE*中，创建元素*InitialE* ，过渡到*StateE1* ，过渡到*StateE2* ，过渡到*StateE3* ；由事件触发的所有转换*E*
8. 从入口*EP1*到*StateC1*和划的划线过渡。

不同条目类型的划转换：

1. 默认条目： *State1*到*State2* ；由事件DEFAULT 触发。
2. 显式条目： *State1*到*StateB2* ；由事件EXPLICIT 触发。
3. 浅历史条目：从*State1*到*HistoryC* ；由事件SHALLOW 触发。
4. 深度历史条目： *State1*到*HistoryD* ；由事件DEEP 触发。
5. 入口条目： *State1*到*EP1* ；由事件ENTRYPOINT 触发。

其它转换：

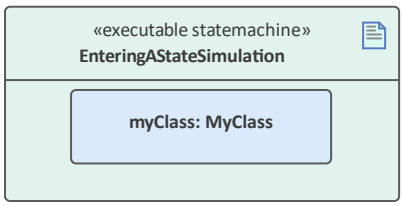
1. 复合状态Exit：从*State2*到*State1* ；由事件BACK 触发。
2. *State1*到终点，由事件QUIT 触发。

仿真

工件

Enterprise Architect支持 C、C++、C#、 Java和JavaScript 。我们在此示例中使用JavaScript ，因为我们不需要安装编译器。 （对于其他语言，需要 Visual Studio 或 JDK。 ）

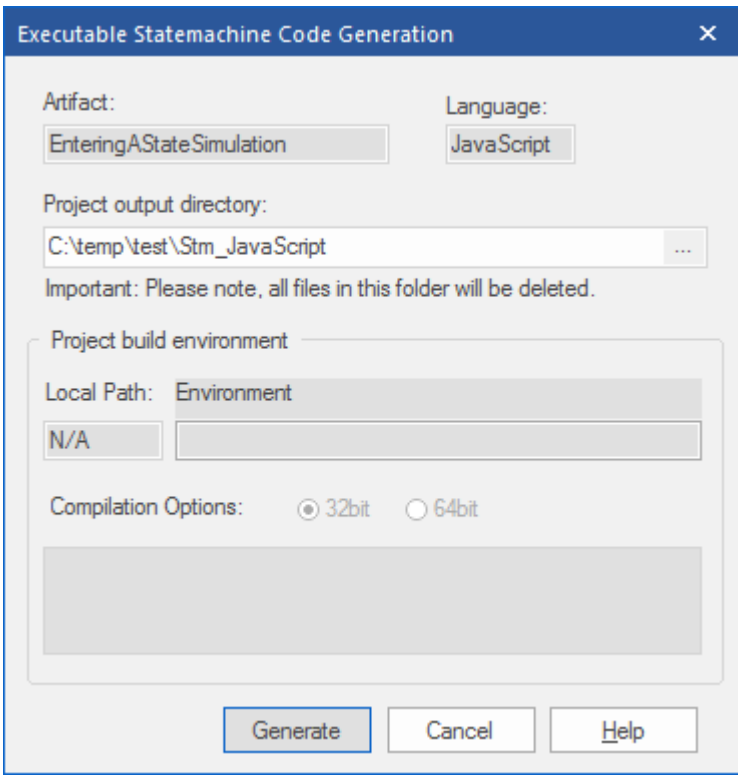
1. 在工具图表的 仿真工具"页面上，将 输入和可执行状态机"的图表工具箱一个工件 状态模拟"的仿真工具上。将语言设置为JavaScript 。
2. Ctrl+ 将*MyClass*从元素浏览器窗口输入到工件属性浏览器中，选择 粘贴作为属性"选项并将名称命名为属性。



代码生成

1. 单击*EnteringAStateSimulation*并选择 仿真> 可执行状态 > 状态机 >生成、编译和运行"功能区选项。
2. 为生成的源代码指定一个目录。

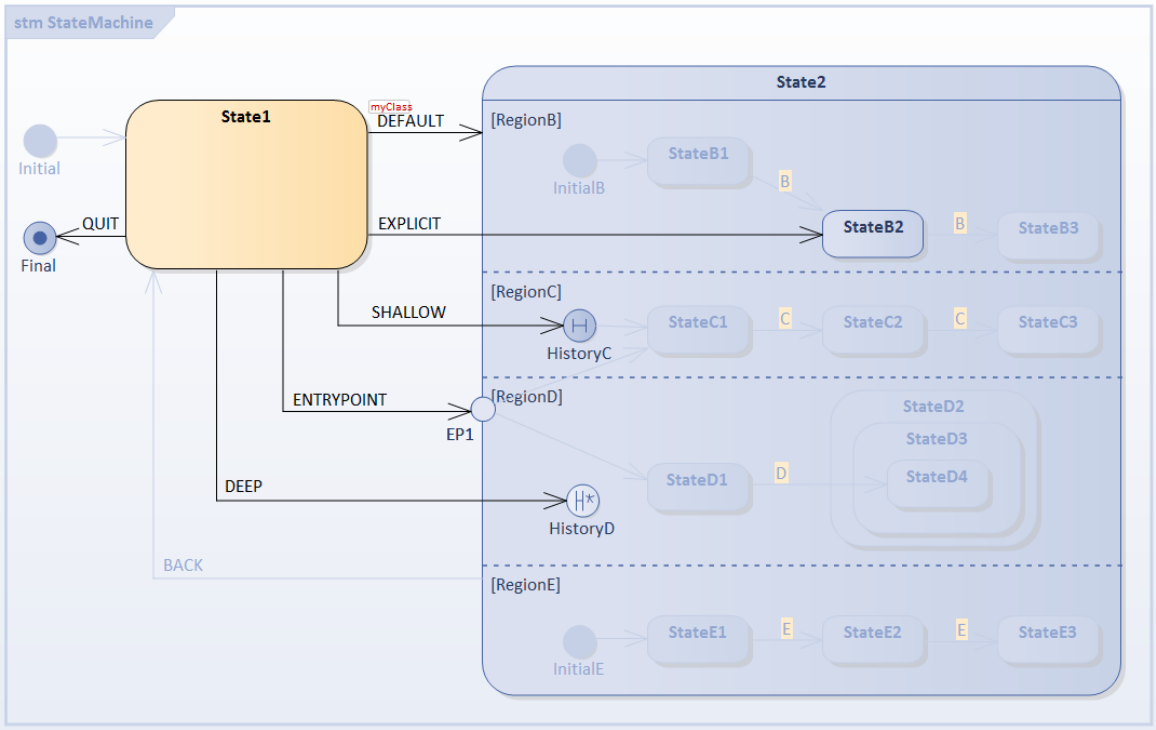
注记：该目录的内容在生成前会被清除；确保您指定的目录仅用于状态机模拟目的。



运行仿真

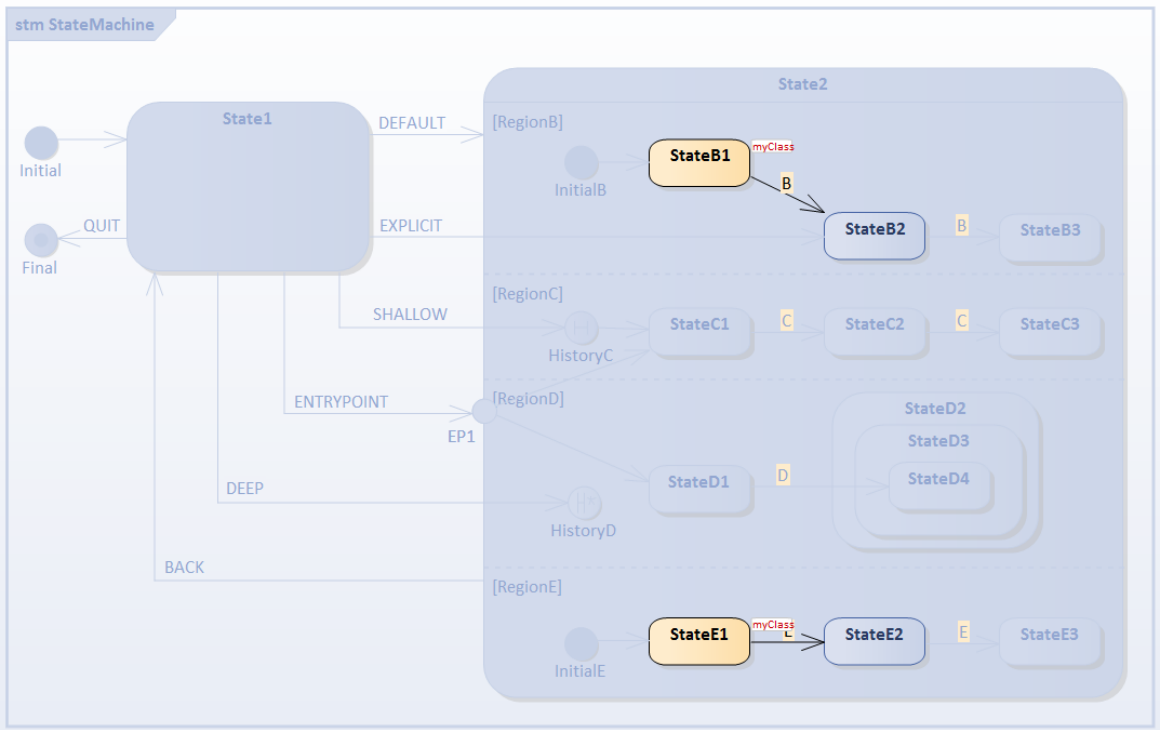
温馨提示：您可以在仿真窗口中查看执行序列，您可以通过选择 仿真>动态仿真仿真>模拟器>打开仿真窗口"功能区选项打开该窗口

模拟开始时， *State1*处于活动状态，状态机正在等待事件。



使用 “仿真动态仿真>事件”功能区选项打开仿真事件（触发器）窗口。

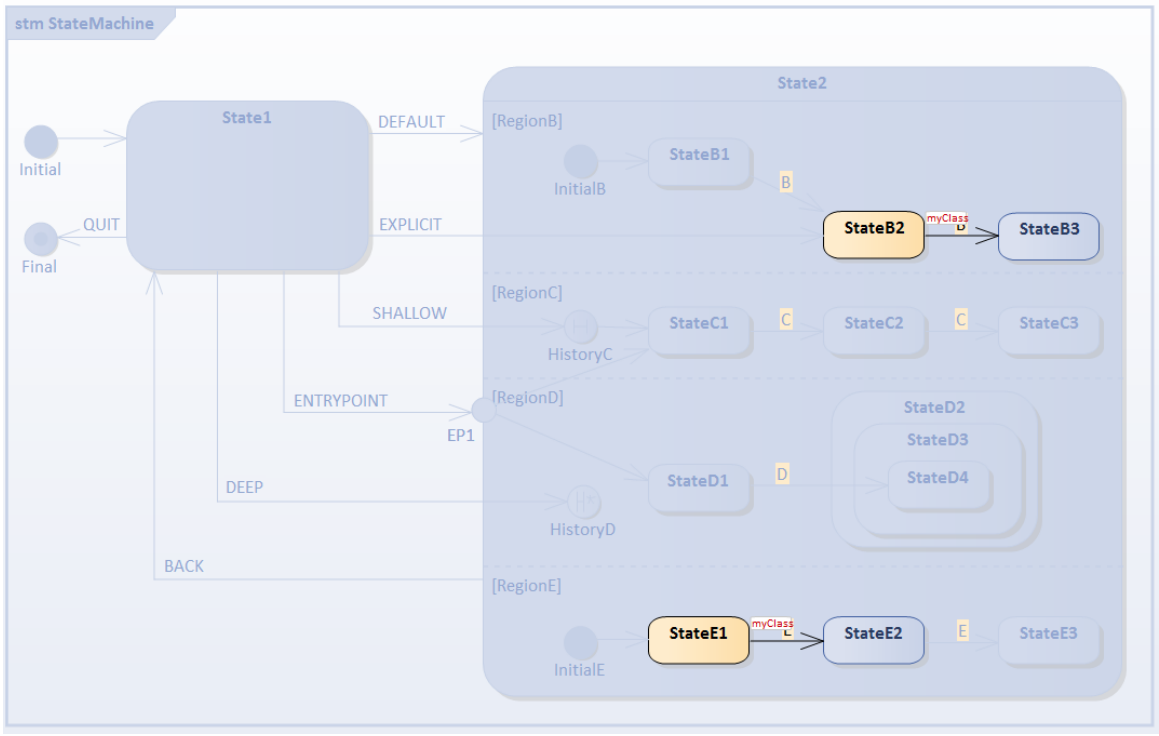
1) 选择默认条目：触发器序列[DEFAULT]。



- *RegionB*被激活，因为它定义了*InitialB* ；从它发出的转换将被执行， *StateB1*是活动状态
- *RegionE*被激活，因为它定义了*InitialE* ；从它发出的转换将被执行， *StateE1*是活动状态
- *RegionC*和*RegionD*处于非活动状态，因为没有定义初始伪状态

选择触发器的[BACK]来重置。

2) 选择显式条目：触发器序列[EXPLICIT]。

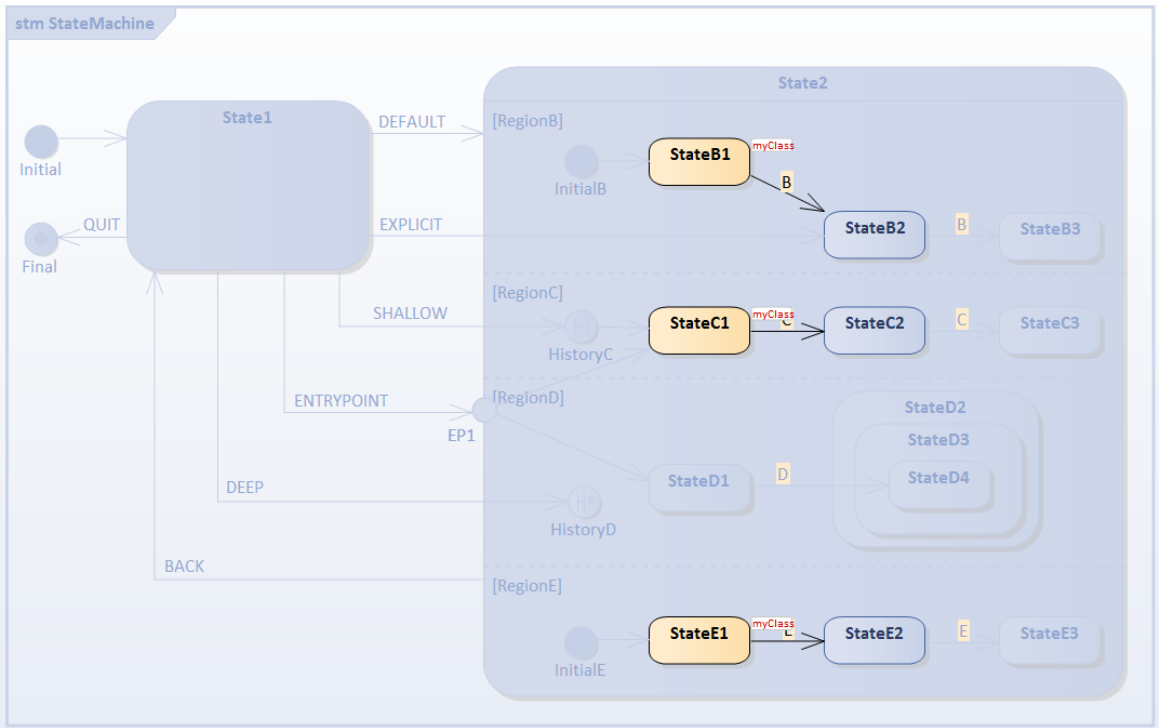


- *RegionB*被激活，因为过渡以包含的顶点*StateB2*为目标
- *RegionE*被激活，因为它定义了*InitialE* ；从它发出的转换将被执行， *StateE1*是活动状态
- *RegionC*和*RegionD*处于非活动状态，因为没有定义初始伪状态

选择触发器的[BACK]来重置。

3) 选择默认历史转移

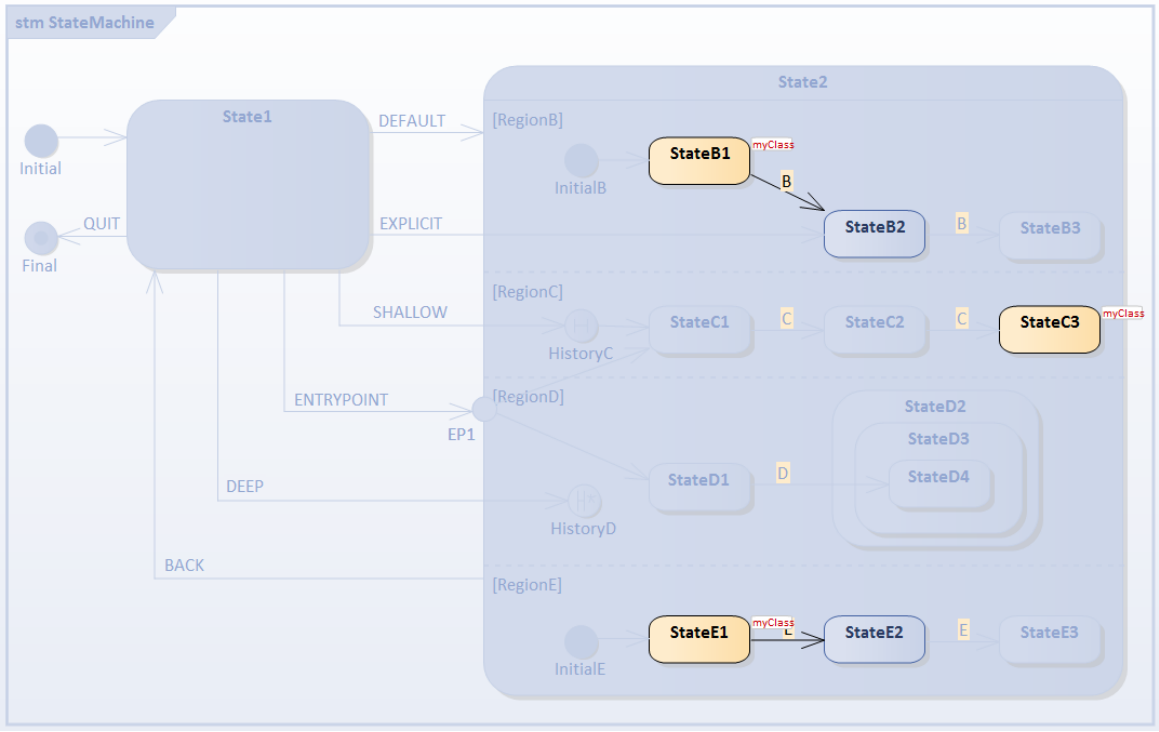
:简单的序列[触发器]



- *RegionC*被激活，因为过渡以包含的顶点*HistoryC*为目标；因为这个区域是第一次进入（并且历史伪状态没有什么要 记住"的），所以从历史C到状态*C1*的转换被执行

- *RegionB*被激活，因为它定义了*InitialB* ；从它发出的转换将被执行， *StateB1*是活动状态
- *RegionE*被激活，因为它定义了*InitialE* ；从它发出的转换将被执行， *StateE1*是活动状态
- *RegionD*处于非活动状态，因为未定义初始伪状态

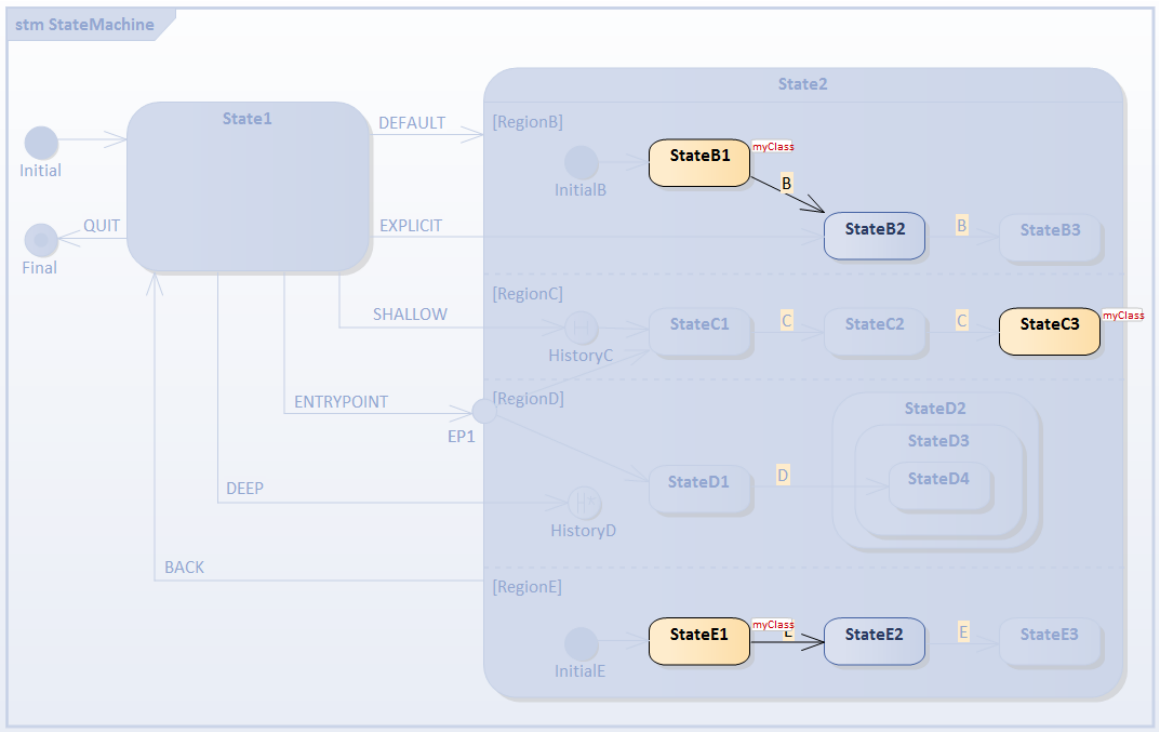
4) 准备测试Shallow History Entry：触发器序列[C, C]。



- 我们假设浅历史伪状态*HistoryC*可以记住*StateC3*

选择触发器的[BACK]来重置。

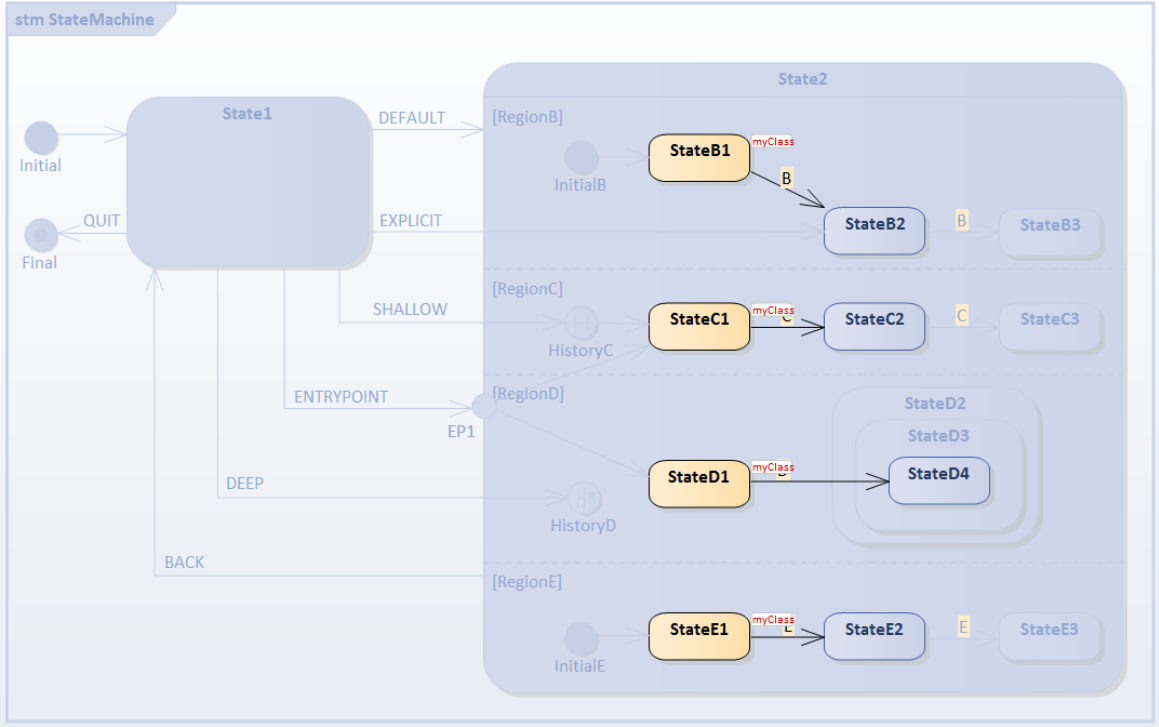
5) 选择Shallow History Entry：简单序列[触发器]。



- 对于*RegionC* ，直接激活*StateC3*

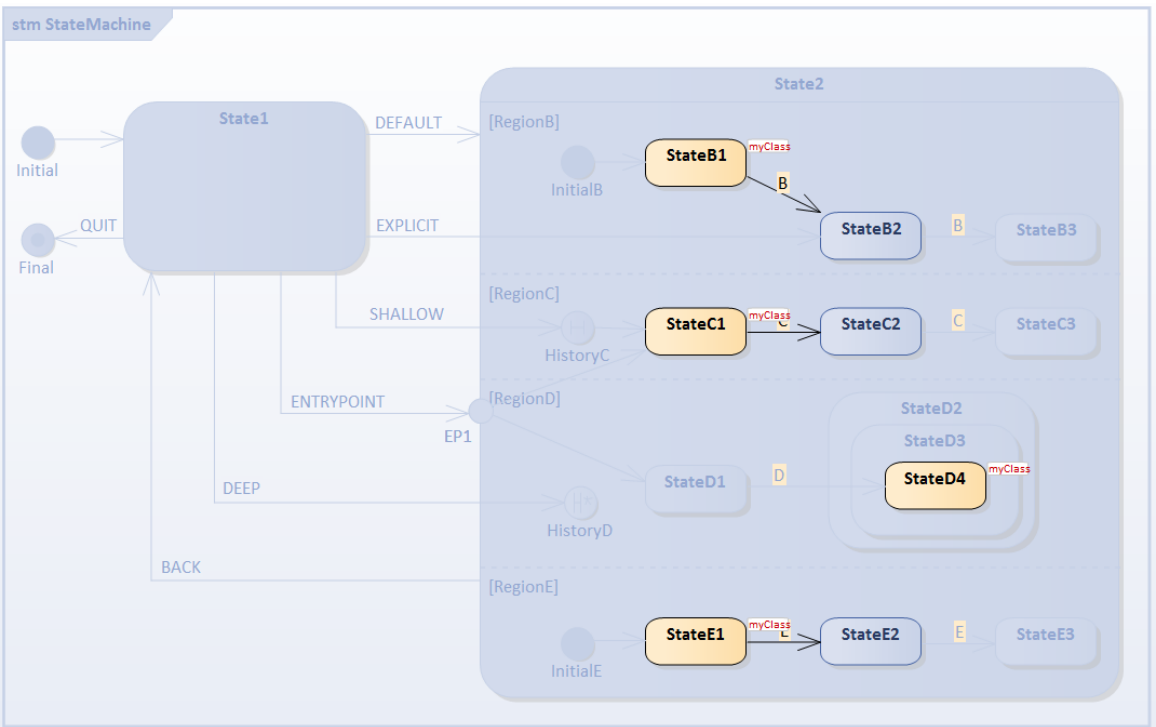
选择触发器的[BACK]来重置。

6) 选择入口条目：简单的序列[触发器]。



- *RegionC*被激活，因为从*EP1*的转换以包含的*StateC1*为目标
- *RegionD*被激活，因为从*EP1*的转换以包含的*StateD1*为目标
- *RegionB*被激活，因为它定义了*InitialB* ；从它发出的转换将被执行， *StateB1*是活动状态
- *RegionE*被激活，因为它定义了*InitialE* ；从它发出的转换将被执行， *StateE1*是活动状态

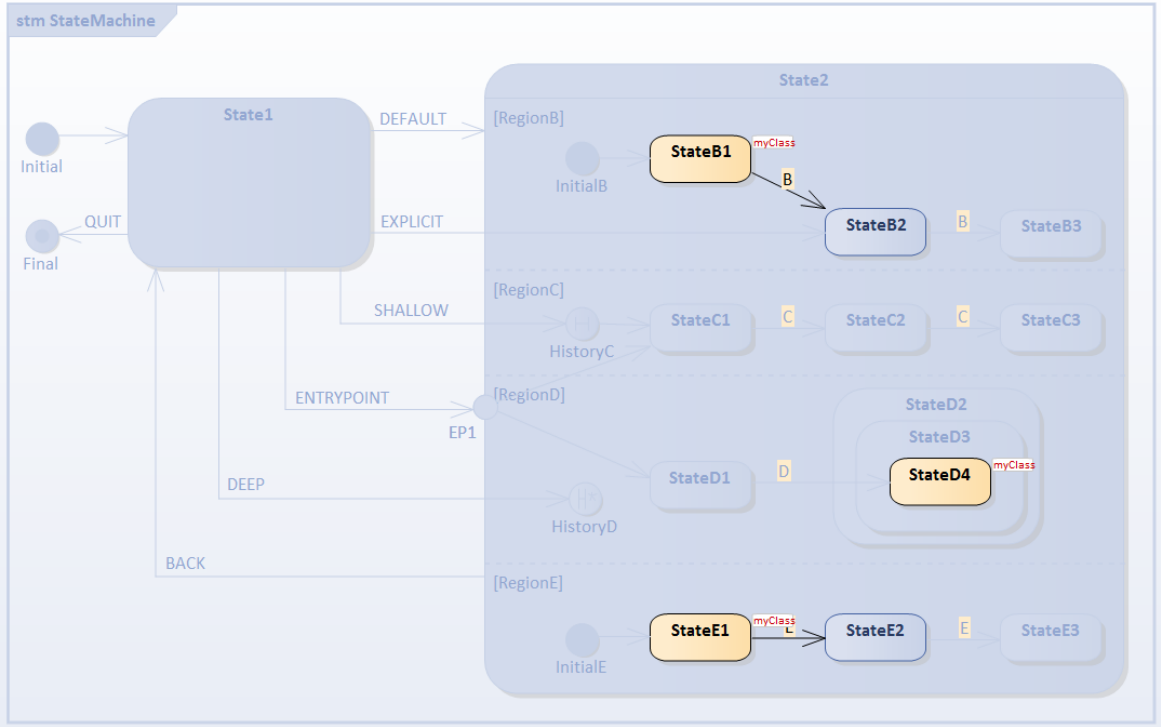
7) 准备测试Deep History：触发器序列[D]。



- 我们假设深度历史伪状态*HistoryD*可以记住*StateD2* 、 *StateD3*和*StateD4*

选择触发器的[BACK]来重置。

8) 选择深度历史条目：触发器序列[DEEP]。



- 对于*RegionD* ，输入*StateD2* 、 *StateD3*和*StateD4* ；痕迹是：
  - myClass[MyClass].StateMachine_State1 退出

- myClass[MyClass]影响
- myClass[MyClass].StateMachine_State2 ENTRY
- myClass[MyClass].StateMachine_State2 DO
- myClass[MyClass].InitialE_105787__TO__StateE1_61746影响
- myClass[MyClass].StateMachine_State2_StateE1 条目
- myClass[MyClass].StateMachine_State2_StateE1 DO
- myClass[MyClass].InitialB_105785__TO__StateB1_61753影响
- myClass[MyClass].StateMachine_State2_StateB1 条目
- myClass[MyClass].StateMachine_State2_StateB1 DO
- myClass[MyClass].StateMachine_State2_StateD2 条目
- myClass[MyClass].StateMachine_State2_StateD2_StateD3 条目
- myClass[MyClass].StateMachine_State2_StateD2_StateD3_StateD4 条目

示例：分叉和汇合

分叉伪状态分裂一个传入的转移

成两个或多个Transitions，终止于一个复合状态的正交区域中的Vertices。从分叉伪状态传出的转换不能有保护或触发器，并且各个传出转换的影响行为至少在概念上是同时执行的。

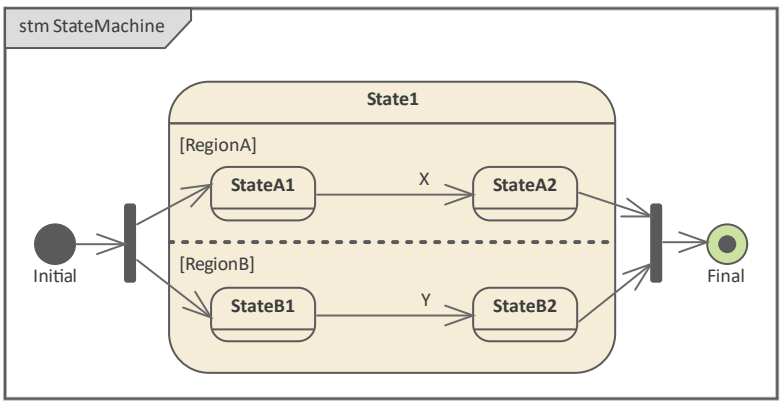
汇合函数合伪状态是来自不同正交区域中的顶点的两个或多个转换的公共目标顶点。汇合伪状态执行同步，因此所有传入的转换必须完成才能通过传出的转移

继续执行转移

.

在这个例子中，我们演示了具有分叉和汇合伪状态的状态机的行为。

建模状态机



状态机的上下文

- 创建一个名为*MyClass*的类元素，作为状态机的上下文
- 在浏览器窗口中右键单击*MyClass*并选择 "添加|状态机"选项

状态机

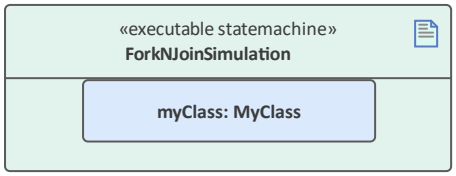
- 在图中添加一个初始节点、一个分叉、一个名为*State*的状态、一个汇合和一个终点
- 放大*State1* ，右键单击它 在图表上选择'高级|定义并发子状态 |定义'选项并定义*RegionA*和*RegionB*
- 在*RegionA*中，定义*StateA1* ，转换到*StateA2* ，由事件*X*触发
- 在*RegionB*中，定义*StateB1* ，转换到*StateB2* ，由事件*Y*触发
- 绘制其他转换：初始到分叉；分叉到划和*StateB1* ； *StateA2* 和 *StateB2*到汇合；汇合到终点

仿真

工件

Enterprise Architect支持 C、C++、C#、 Java和JavaScript ；我们将在此示例中使用JavaScript ，因为我们不需要安装编译器（对于其他语言，需要 Visual Studio 或 JDK ）。

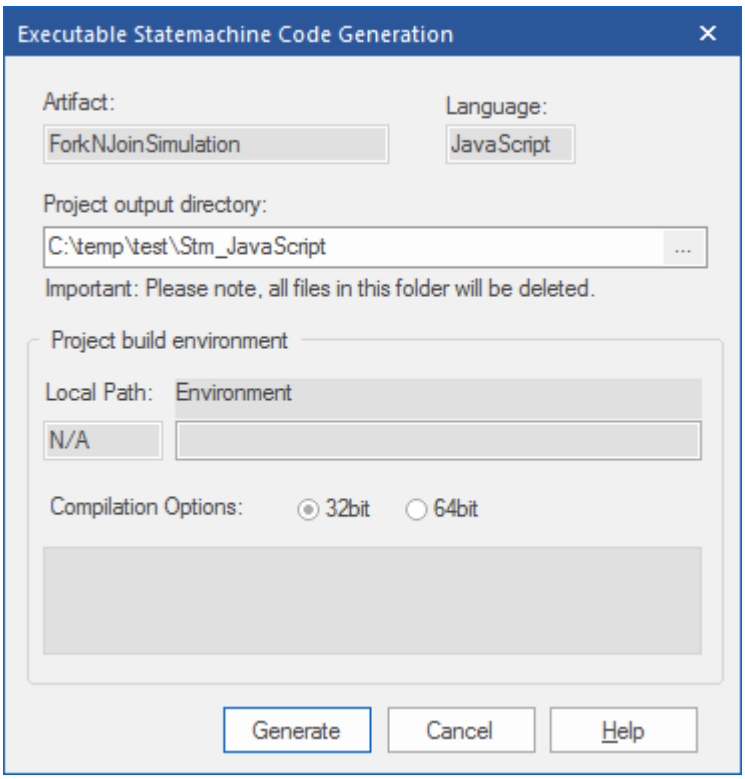
- 从图表中选择 "工具箱仿真页面，然后将绘图工具拖到可执行状态机上，创建一个工件；将其命名为*ForkNJoinSimulation*并将其 "语言"字段设置为 "JavaScript "
- 从浏览器窗口 Ctrl+ 将*MyClass*拖放到属性*ForkNJoinSimulation D*工件；将其命名为*myClass*



代码生成

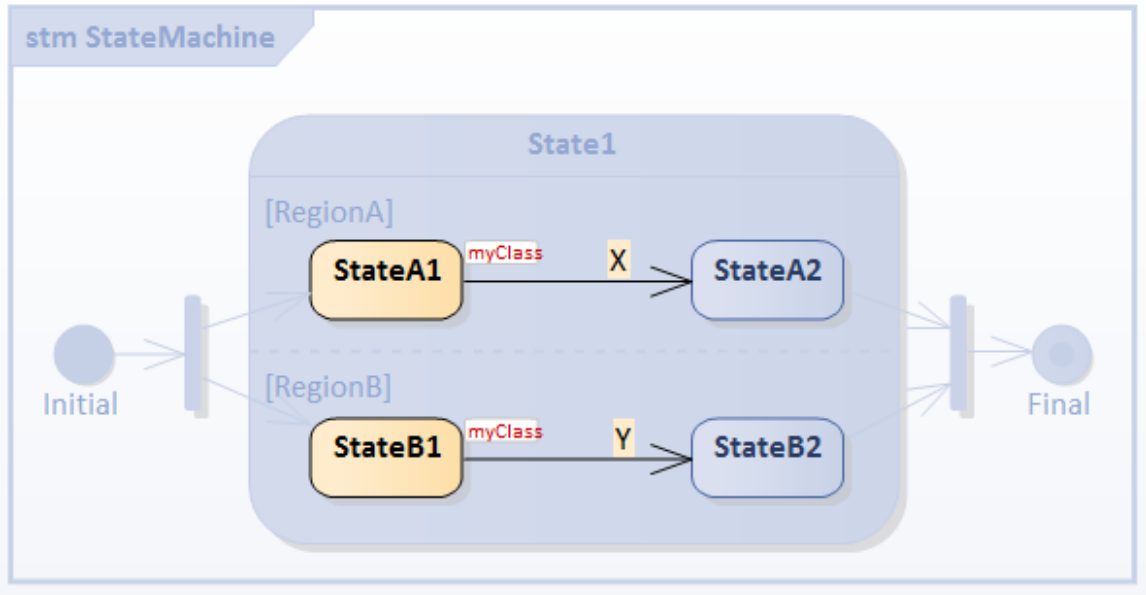
- 单击*ForkNJoinSimulation*并选择 仿真> 可执行状态 > 状态机 >生成、编译和运行”功能区选项
- 为生成的源代码指定目录

注记：该目录的内容在生成前会被清除；确保您指向的目录仅用于状态机模拟目的。



运行仿真

当仿真开始时， *State1* 、 *StateA1*和*StateB1*处于活动状态，状态机正在等待事件。



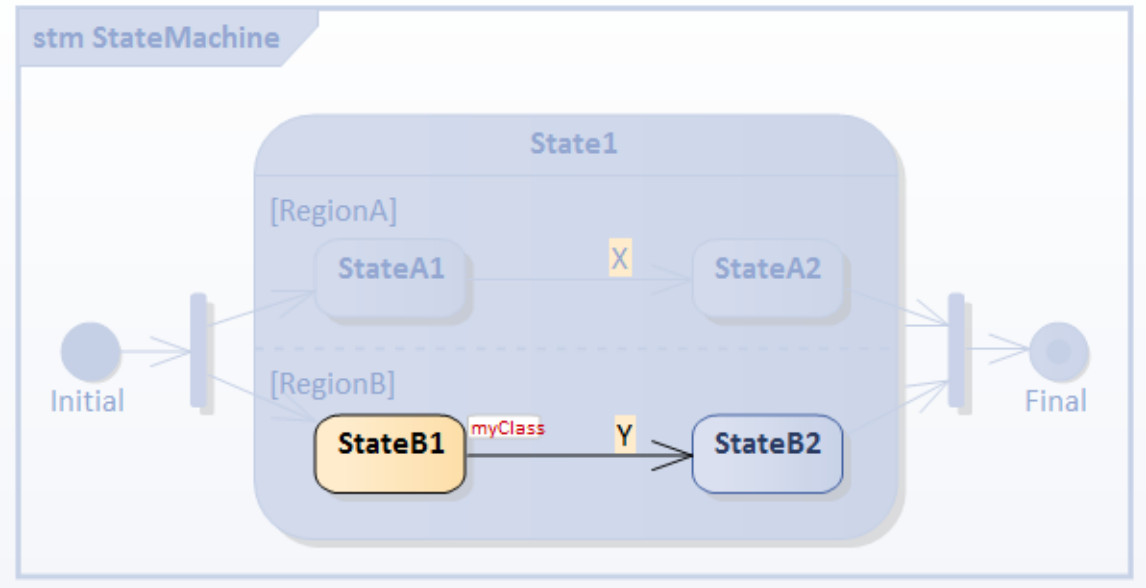
选择 '仿真>动态仿真>事件"功能区选项，显示仿真事件窗口。

简单来说，在事件*X*中，触发器会退出并进入*StateA2* ；在 entry 和 doActivity 行为运行后，运行的完成事件被调度和回调。然后启用并遍历从*StateA2*到汇合伪状态的转换。

注记：汇合必须等待所有传入的转换完成，然后才能通过传出的转移

继续执行转移

.由于*RegionB*的分支没有完成（因为*StateB1*仍然处于活动状态，等待触发），此时不会执行从汇合终端到终点的转换。



简单来说，在事件*Y*时，触发器将退出并进入*StateB2* ；在 entry 和 doActivity 行为运行后，运行的完成事件被调度和回调。然后启用并遍历从*StateB2*到汇合伪状态的转换。满足所有传入的过渡都已完成的条件，则执行从汇合到汇合终点的过渡。仿真结束。

提示：您可以从仿真窗口（'仿真>动态仿真仿真>模拟器>打开仿真窗口'功能区选项）查看执行轨迹序列。

myClass[MyClass].Initial_82285__TO__fork_82286_82286_61745影响

myClass[MyClass].StateMachine_State1 条目

myClass[MyClass].StateMachine_State1 DO
myClass[MyClass]影响
myClass[MyClass].StateMachine_State1_StateA1 条目
myClass[MyClass].StateMachine_State1_StateA1 DO
myClass[MyClass]影响
myClass[MyClass].StateMachine_State1_StateB1 条目
myClass[MyClass].StateMachine_State1_StateB1 DO
触发器X
myClass[MyClass].StateMachine_State1_StateA1 退出
myClass[MyClass]影响
myClass[MyClass].StateMachine_State1_StateA2 条目
myClass[MyClass].StateMachine_State1_StateA2 DO
myClass[MyClass].StateMachine_State1_StateA2 退出
myClass[MyClass]影响
触发器
myClass[MyClass].StateMachine_State1_StateB1 退出
myClass[MyClass]影响
myClass[MyClass].StateMachine_State1_StateB2 条目
myClass[MyClass].StateMachine_State1_StateB2 DO
myClass[MyClass].StateMachine_State1_StateB2 退出
myClass[MyClass]影响
myClass[MyClass].StateMachine_State1 退出
myClass[MyClass]影响

示例：延迟事件模式

Enterprise Architect支持延迟事件模式。

在一个状态下创建一个延迟事件：

1. 为状态创建一个自我转换。
2. 更改向 ‟内部”过渡的 ‟种类”。
3. 简单地触发器要延迟的事件。
4. 在 ‟影响”字段中，输入 ‟defer();”。

仿真：

1. 选择 ‟仿真>动态仿真仿真>模拟器>打开仿真窗口”。同时选择 ‟仿真>动态仿真>事件”打开仿真事件窗口。
2. 模拟器事件窗口帮助您触发事件；双击 ‟等待触发器”列中的触发器。
3. 仿真窗口以文本形式显示执行过程。您可以在 Simulator 命令行中键入 ‟dump”以显示队列中延迟了多少事件；输出可能类似于：
   泳池]事件泳池：[NEW,NEW,NEW,NEW,NEW,]

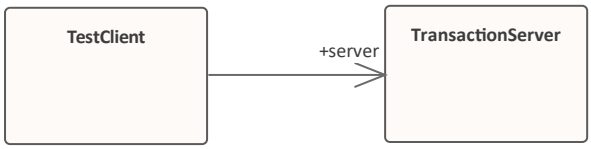
延迟事件示例

此示例显示了使用延迟事件的模型，以及显示所有可用事件的仿真事件窗口。

我们首先设置上下文（包含状态机的类元素），在一个简单的上下文模拟它们并从外部引发事件；然后用上下文事件机制模拟一个客户端-服务器时间。

创建上下文和状态机

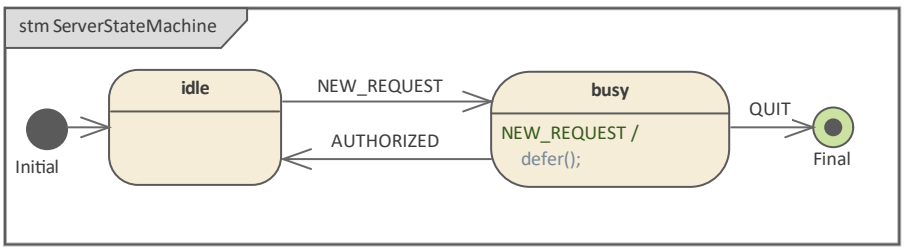
创建服务器上下文



创建类图并：

1. A类元素*TransactionServer* ，你添加一个状态机*ServerStateMachine* 。
2. A类元素*TestClient* ，向其中添加状态机*ClientStateMachine* 。
3. 从*TestClient*到*TransactionServer*的关联，目标角色名为*server*。

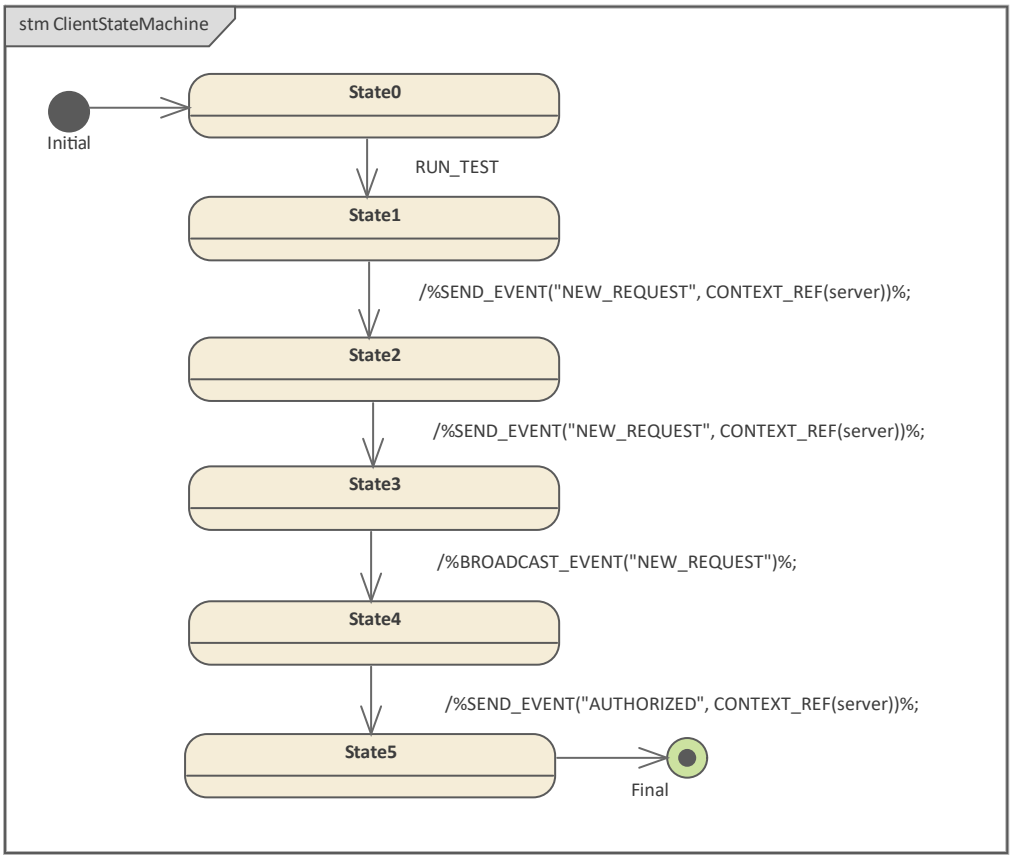
***ServerStateMachine*建模**



1. 在状态机图中添加一个 Initial节点*Initial* ，并转换到一个状态*idle* 。

2. 转移
   （以事件触发器为简单）到状态忙。
3. 转移
   （以事件为触发器）终点状态终点
4. 转移
   （与事件 AUTHORIZED 一样触发器）到*idle* 。
5. 转移
   （以事件触发器作为简单和*defer();*作为效果）到忙

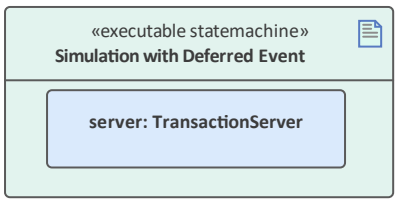
***ClientStateMachine*建模**



1. 在状态机图中添加一个 Initial节点*Initial* ，并转换到一个状态。
2. 转移
   （以事件*RUN_TEST* 作为触发器）到 *stateState1*状态
3. 转移
   （效果：%SEND_EVENT("NEW_REQUEST", CONTEXT_REF(server))%;) 到状态*State2*。
4. 转移
   （效果： %SEND_EVENT(" *NEW_REQUEST", CONTEXT_REF(server))%;)* 到 *stateState3*状态
5. 转移
   （效果： %BROADCAST_EVENT(" *NEW_REQUEST")%;)* 到 *stateState4*状态
6. 转移
   （效果： %SEND_EVENT("AUTHORIZED", CONTEXT_REF(server))%;)到 *stateState5*状态
7. 转移
   到终点状态终点

仿真在一个简单的上下文

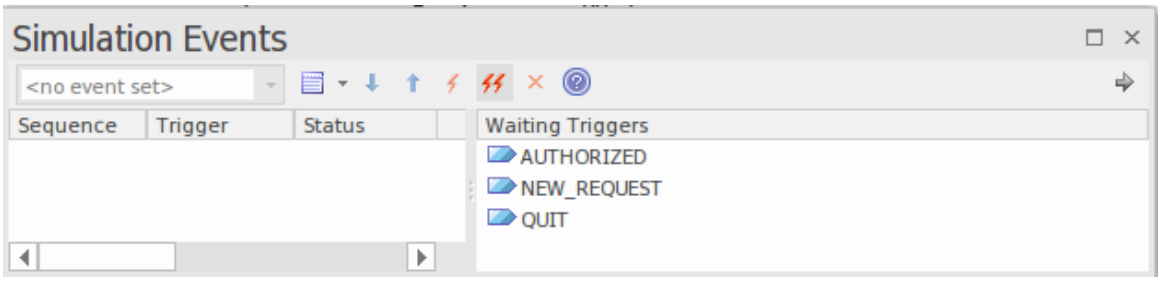
创造仿真工件



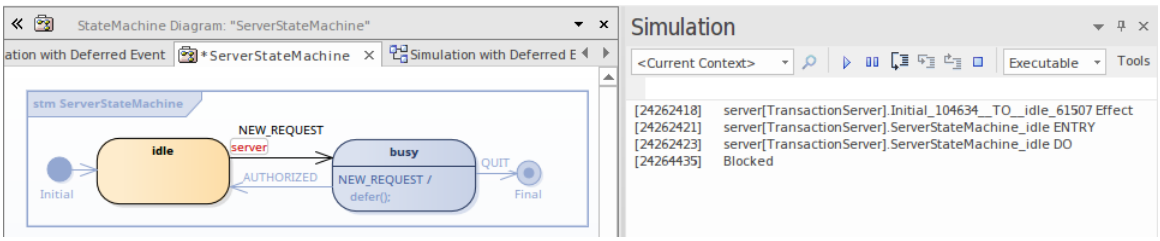
1. 工件and with the name仿真事件的可执行状态机将 语言"字段设置为*JavaScript*
2. 放大 Ctrl+元素将它拖到事务服务器上，然后将其作为属性与名称服务器工件粘贴。

运行仿真

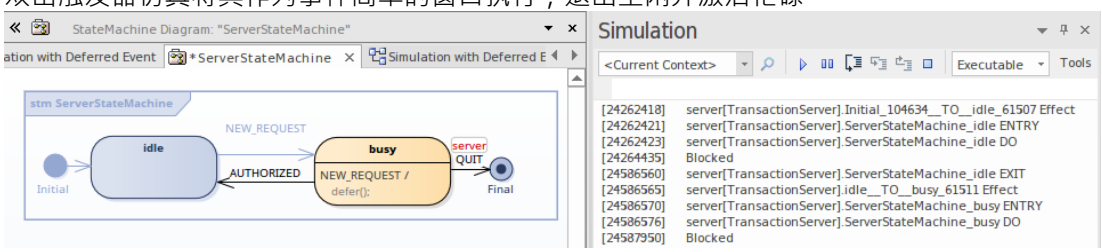
1. '选择工件编译仿真生成运行指定一个目录为你的（注记：目录中的所有文件将在模拟开始之前）。
2. 点击生成按钮。
3. 选择 仿真>动态仿真>事件"选项打开仿真事件窗口。



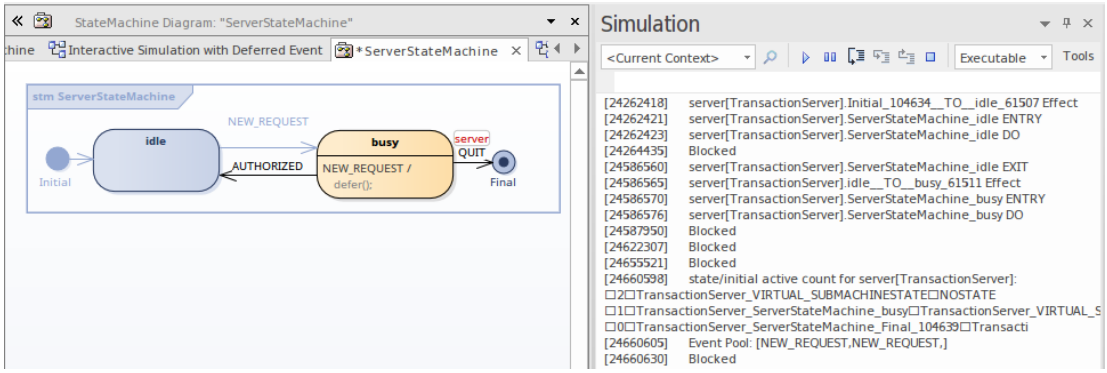
模拟开始时， *idle*将是活动状态。



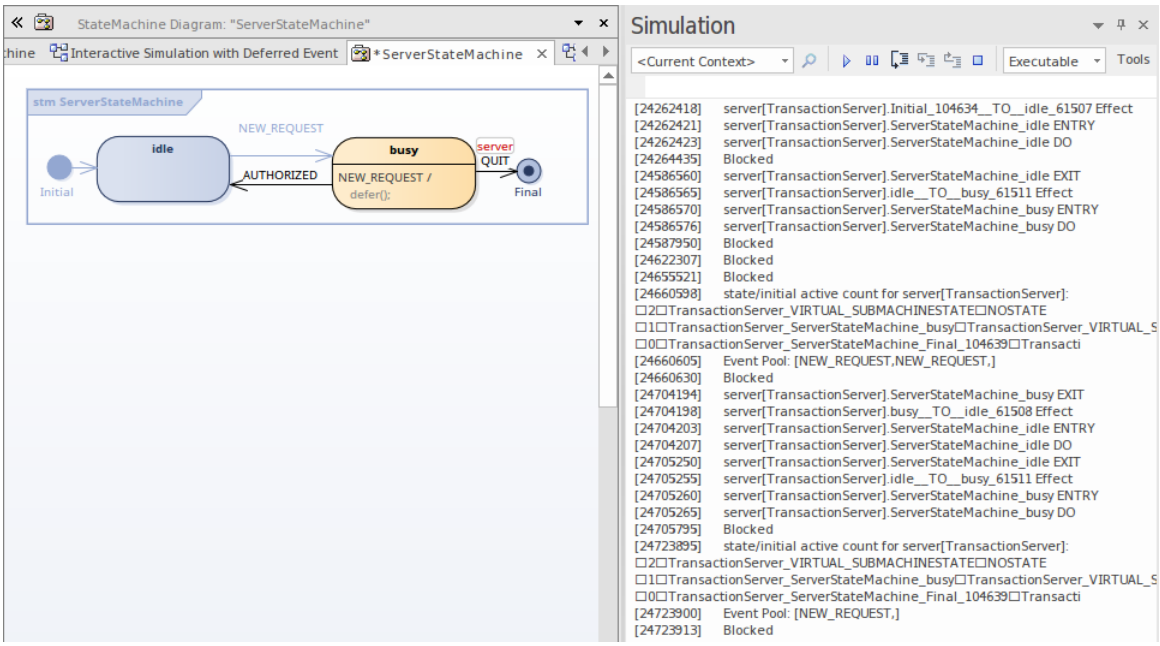
1. 双击触发器仿真将其作为事件简单的窗口执行；退出空闲并激活忙碌。



2. 双击仿真窗口中的仿真，再次像事件一样触发器； *busy*保持激活状态，并且泳池的实例被附加到事件池中。
3. 双击仿真事件窗口中的仿真，第三次执行触发器事件； *busy*保持激活状态，并且泳池的实例被附加到事件池中。
4. 仿真窗口命令行中的类型转储；请注意，事件池有两个 NEW_REQUEST 实例。

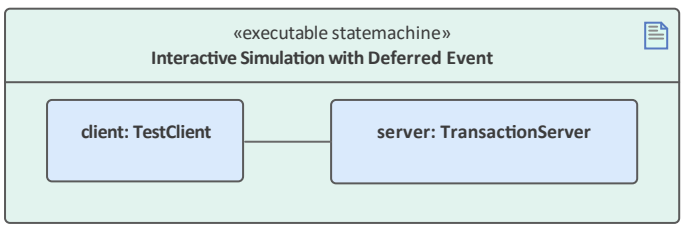


5. 双击触发器仿真中的AUTHORIZED，将其作为事件模拟窗口执行；这些行动发生：
   -*busy*退出， *idle*变为活动状态
   - 从池中检索 NEW_REQUEST 事件，退出空闲并且忙碌变为活动状态
6. 仿真窗口命令行中的类型转储；现在事件泳池只有一个现在实例。



通过发送/广播事件进行交互模拟

创造仿真工件

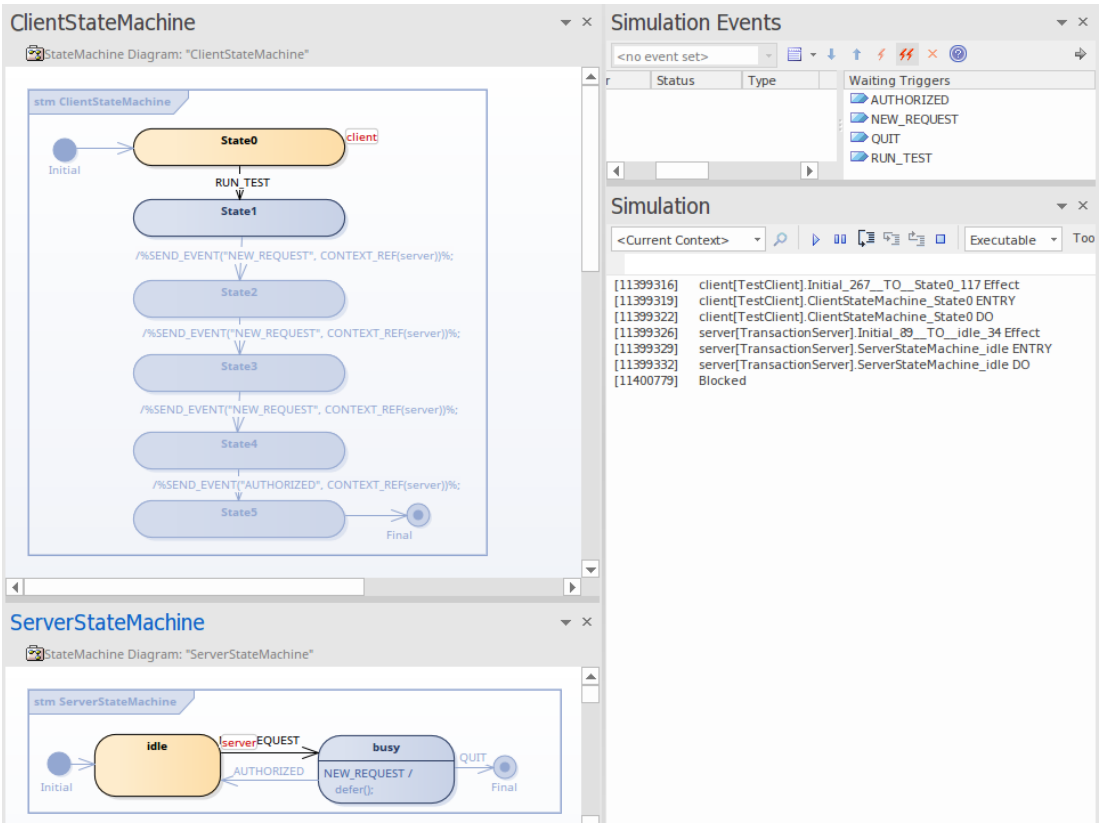


1. 使用名称创建事件仿真状态并可执行状态机到工件*JavaScript* 语言"字段集的名称放大元素。
2. Ctrl+D 将*TransactionServer*元素工件属性上，并将其粘贴为名称服务器。
3. Ctrl+工件将*TestClient*元素拖到属性上，并将其粘贴为名称*client* 。
4. 创建从客户端到服务器的连接器。
5. 单击连接器并按 Ctrl+L 以选择从*TestClient*元素到*TransactionServer*元素的关联。

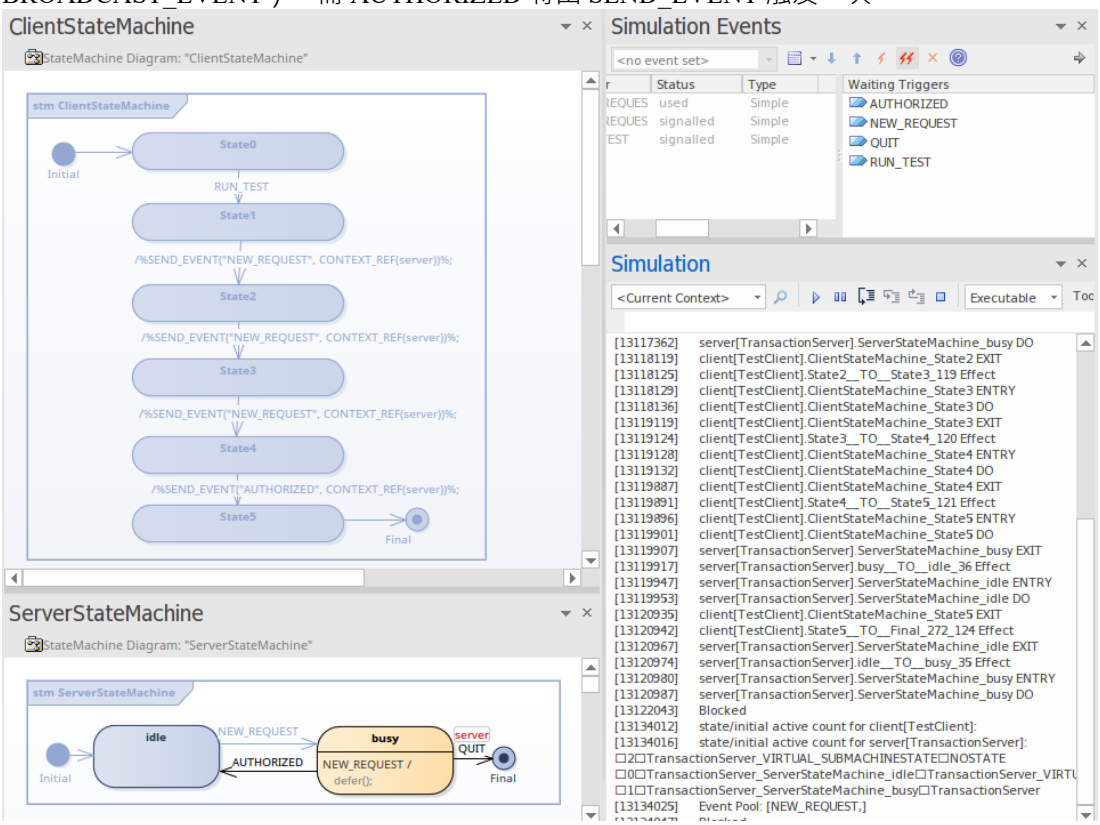
运行互动仿真

1. 以与简单上下文相同的方式启动模拟。

   模拟开始后，客户端保持在*State0*而服务器保持在空闲状态。



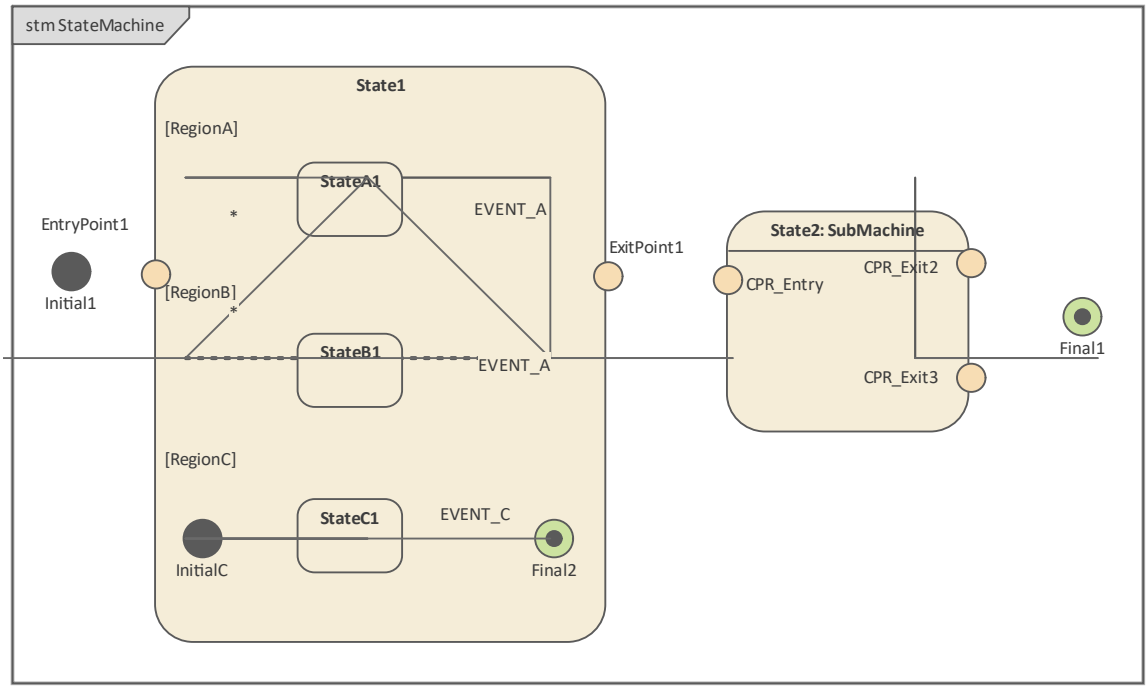
2. 双击仿真事件窗口中的 RUN_TEST 即可触发。事件 NEW_REQUEST 将被触发 3 次（由 SEND_EVENT 和 BROADCAST_EVENT），而 AUTHORIZED 将由 SEND_EVENT 触发一次。

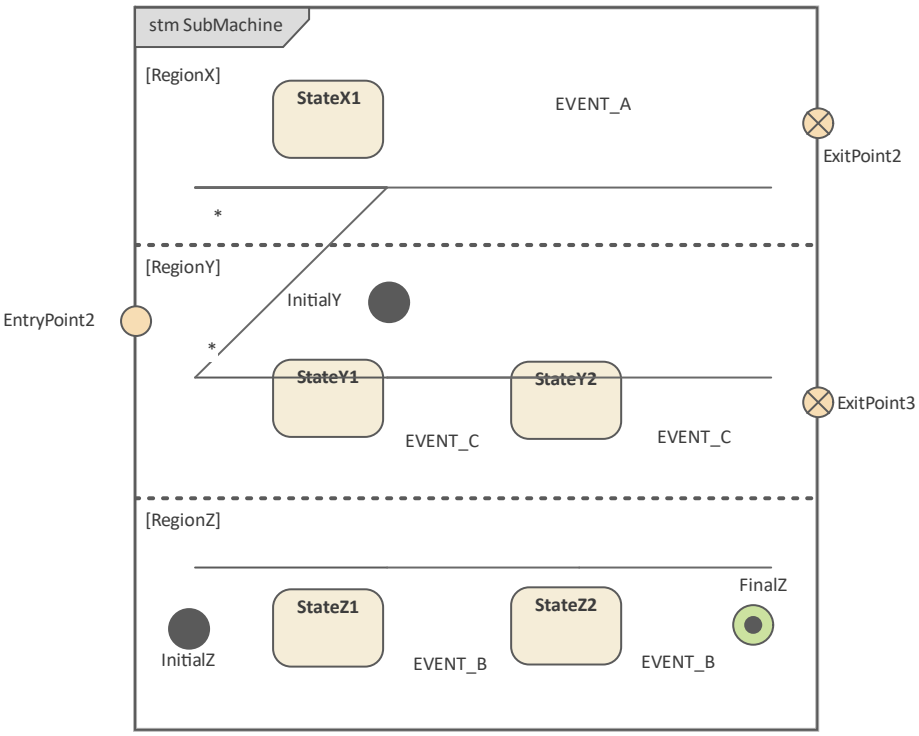


仿真窗口命令行中的类型转储，事件池中只剩下一个泳池实例。结果与我们的手动触发测试相匹配。

示例：入口和出口点（连接点参考）

Enterprise Architect提供对入口和出口点以及连接点引用的支持。在这个例子中，我们为*MyClass* 定义了两个状态机——状态机状态机*SubMachine*。





- *State1*是一个复合状态（也称为正交状态，因为它有多个区域），有三个区域： *RegionA* 、 *RegionB*和*RegionC*
- *State2*是调用*SubMachine*的 SubMachine状态，它有三个区域： *RegionX* 、 *RegionY*和*RegionZ*
- 在*State1*上定义*EntryPoint1* ，激活三个区域中的两个；在*SubMachine*上定义*EntryPoint2*以激活三个区域中的

两个

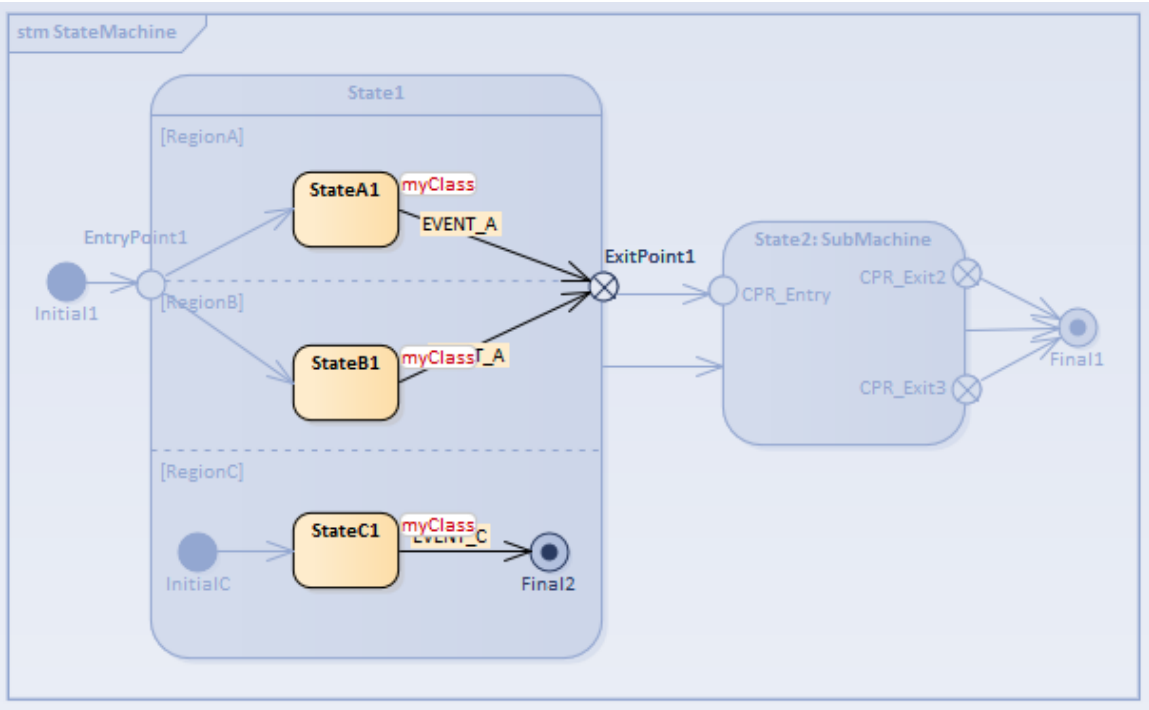
- *ExitPoint1*在*State1*上定义；在*SubMachine*上定义了两个出口点*ExitPoint2*和*ExitPoint3*
- 连接点引用在*State2*上定义并绑定到输入 SubMachine 的入口/出口点
- 定义初始节点以演示区域的默认区域

进入状态：入口进入

状态1 上的入口点 1

当一个转移

以*EntryPoint1*为目标已启用， *State1*被激活，然后是包含的区域。



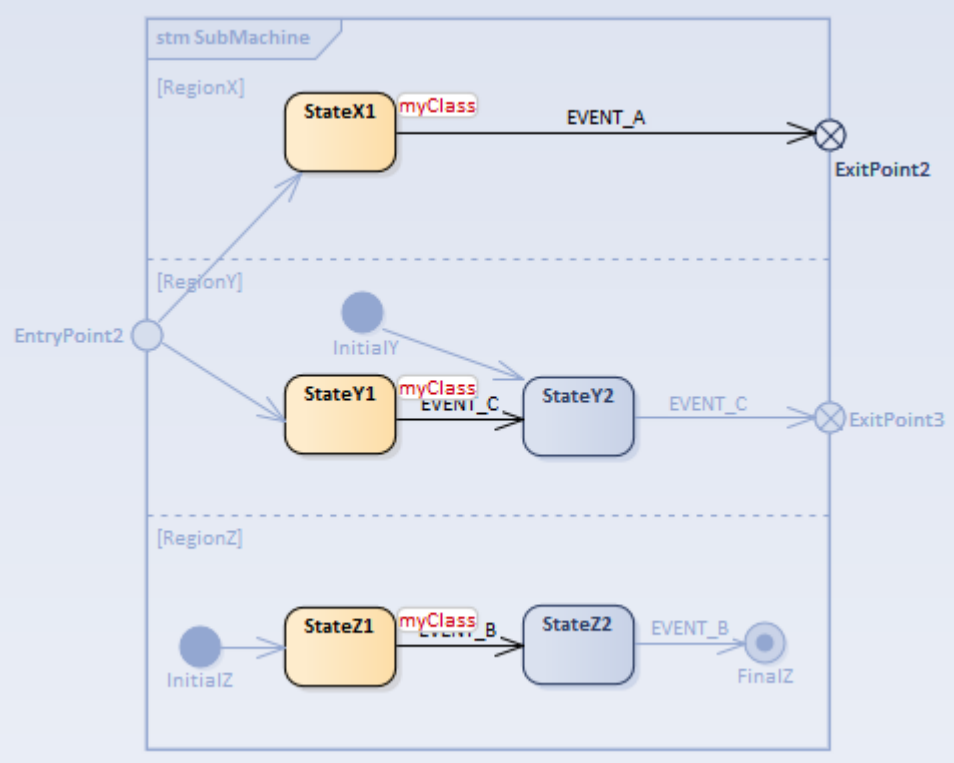
- *RegionA*和*RegionB*发生显式激活，因为它们中的每一个都由转移
  输入转移
  终止于区域包含的顶点之一
- *RegionC*发生默认激活，因为它定义了一个 Initial 伪状态*InitialC*和转移
  从*InitialC*到*StateC1*开始执行

SubMachine*上的*EntryPoint2

要模拟的序列是：[简单的触发器，EVENT_A]。

当一个转移

针对连接点参考*CPR_Entry* on *State2*被启用， *State2*被激活，随后通过绑定入口点激活SubMachine。



- 显式激活*RegionX*和*RegionY* ，因为它们中的每一个都由转移
  输入转移
  终止于区域包含的顶点之一 - *RegionX*中的*StateX1* ， *RegionY*中的*StateY1*
- *RegionZ*发生默认激活，因为它定义了 Initial 伪状态 *InitialZ*和转移
  从*InitialZ*到*StateZ1*开始执行

进入状态：默认进入

当复合状态是转移

的直接目标时，就会出现这种情况转移

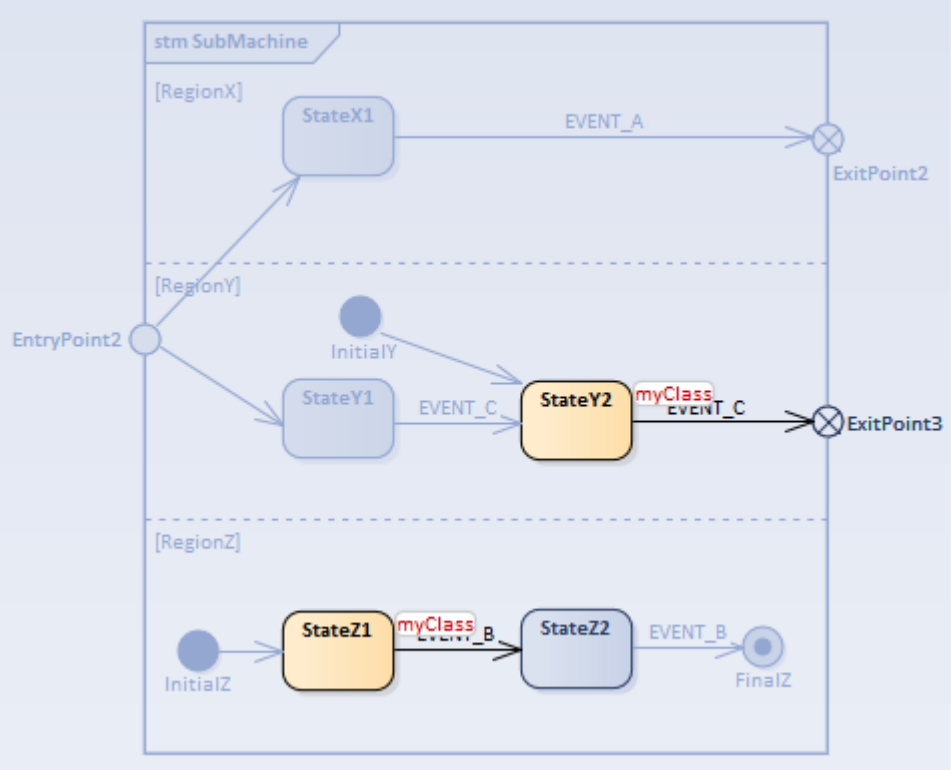
.

***State2*的默认入口**

要模拟的序列是：[简单的触发器，EVENTC]。

当一个转移

直接针对*State2*启用， *State2*被激活，随后默认激活区域的所有区域。



- *RegionX*的状态是不活跃的，因为它没有定义一个Initial节点
- *RegionY*通过*InitialY*和转移
  激活转移
  到*StateY2*被执行
- *RegionZ*通过*InitialZ*和转移
  激活转移
  到*StateZ1*被执行

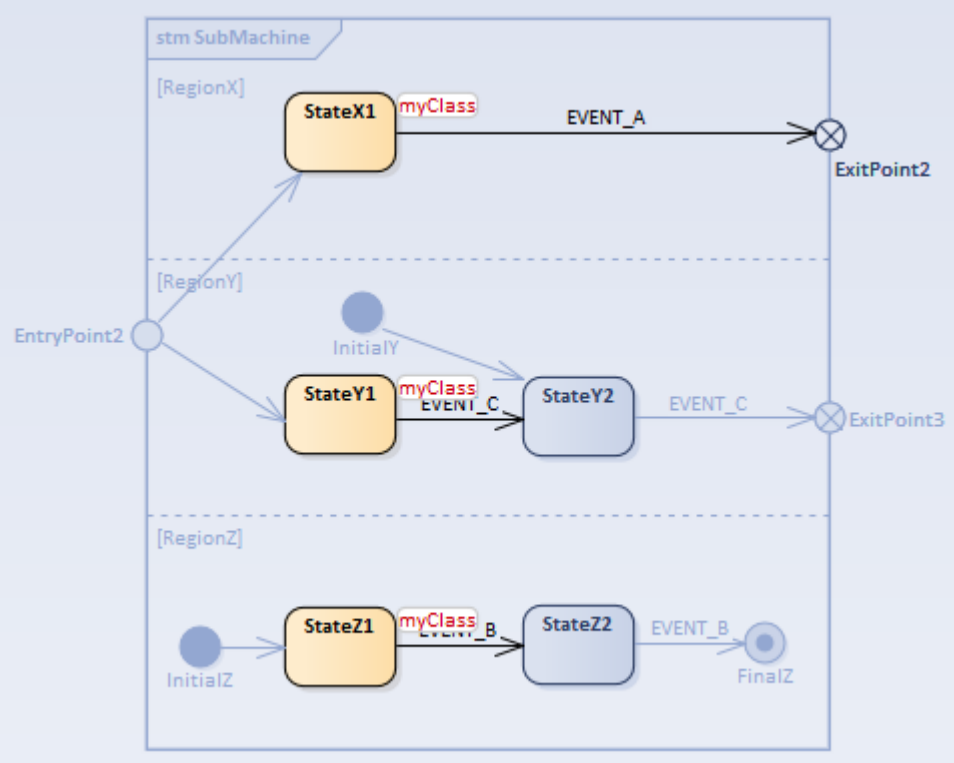
状态

State1 出口

- 触发器序列[EVENT_C, EVENT_A]：先去激活RegionC，再去激活*RegionA*和*RegionB* ； *State1*的退出行为执行后，转移
  从*ExitPoint1*传出已启用
- 触发器序列[EVENT_A, EVENT_C]：先去激活*RegionA*和*RegionB* ，再去激活*RegionC* ； *State1*的退出行为执行后，转移
  直接从*State1*传出已启用

State2 退出

触发器序列[EVENT_C, EVENT_A]，所以当前状态类似这样：



- 简单的触发器[EVENT_A,序列, EVENT_C, EVENT_B, EVENT_B]：先去激活RegionX，再去*RegionY* ，最后去*RegionZ* ； *State2*的退出行为执行后，转移
  直接从*State2*传出已启用
- 简单的触发器[EVENT_A,序列, EVENT_B, EVENT_C, EVENT_C]：先去激活RegionX，再去*RegionZ* ，最后去*RegionY* ； *State2*的退出行为执行后，转移
  启用从*CPR_Exit3*传出（ *SubMachine*上的*ExitPoint3*绑定到*State2*的*CPR_Exit3* ）
- 简单的触发器[EVENT_C,序列, EVENT_B, EVENT_B, EVENT_A]：先去激活*RegionY* ，然后去*RegionZ* ，最后去RegionX； *State2*的退出行为执行后，转移
  从*CPR_Exit2*传出已启用（ *SubMachine*上的*ExitPoint2*绑定到*State2*的*CPR_Exit2* ）

示例：历史伪状态

状态历史是一个与复合状态区域相关联的方便概念，其中区域跟踪状态上次退出时所处的配置。如有必要，当区域下一个变为活动状态时（例如，从处理中断返回后），或者如果存在本地转移

，这允许轻松返回到该状态配置转移

回到它的历史。

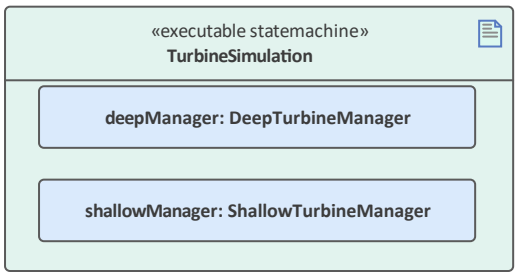
Enterprise Architect支持两种类型的历史伪状态：

- Deep History - 代表最近访问包含区域的完整状态配置；效果和转移
  一样转移
  终止于状态Pseudostate 相反，终止于保留状态配置的最内层状态，包括执行沿途遇到的所有入口行为
- 浅历史 - 表示仅返回最近状态配置的最顶层子状态，使用默认输入规则输入

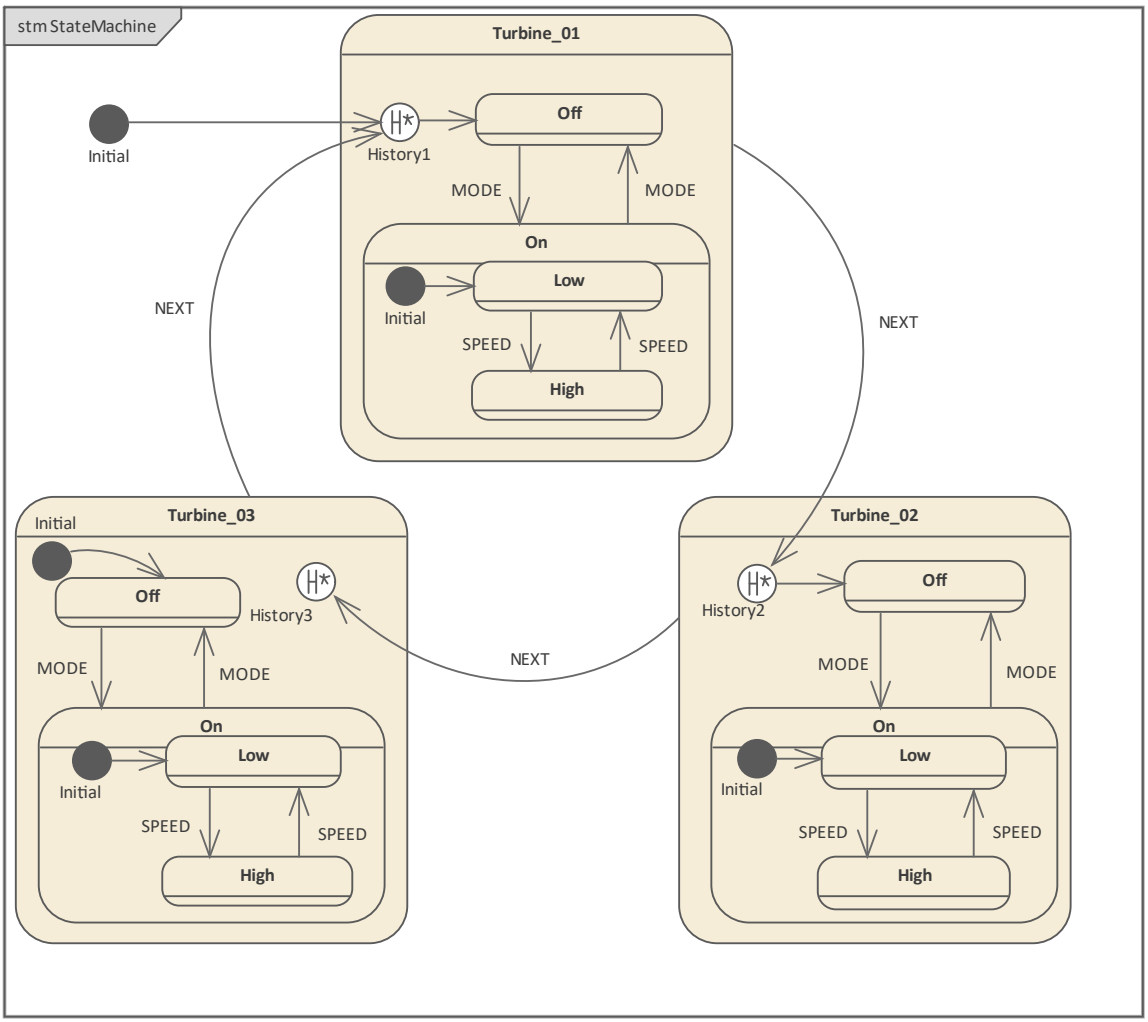
在此示例中， *DeepTurbineManager*和*ShallowTurbineManager*类完全相同，只是第一个包含的状态机具有 deepHistory Pseudostate，而第二个包含的状态机具有 shallowHistory Pseudostate。

两个状态机都有三个复合状态： *Turbine_01* 、 *Turbine_02*和*Turbine_03* ，每个状态机在其区域中都有*Off*和*On*状态以及 History Pseudostate。

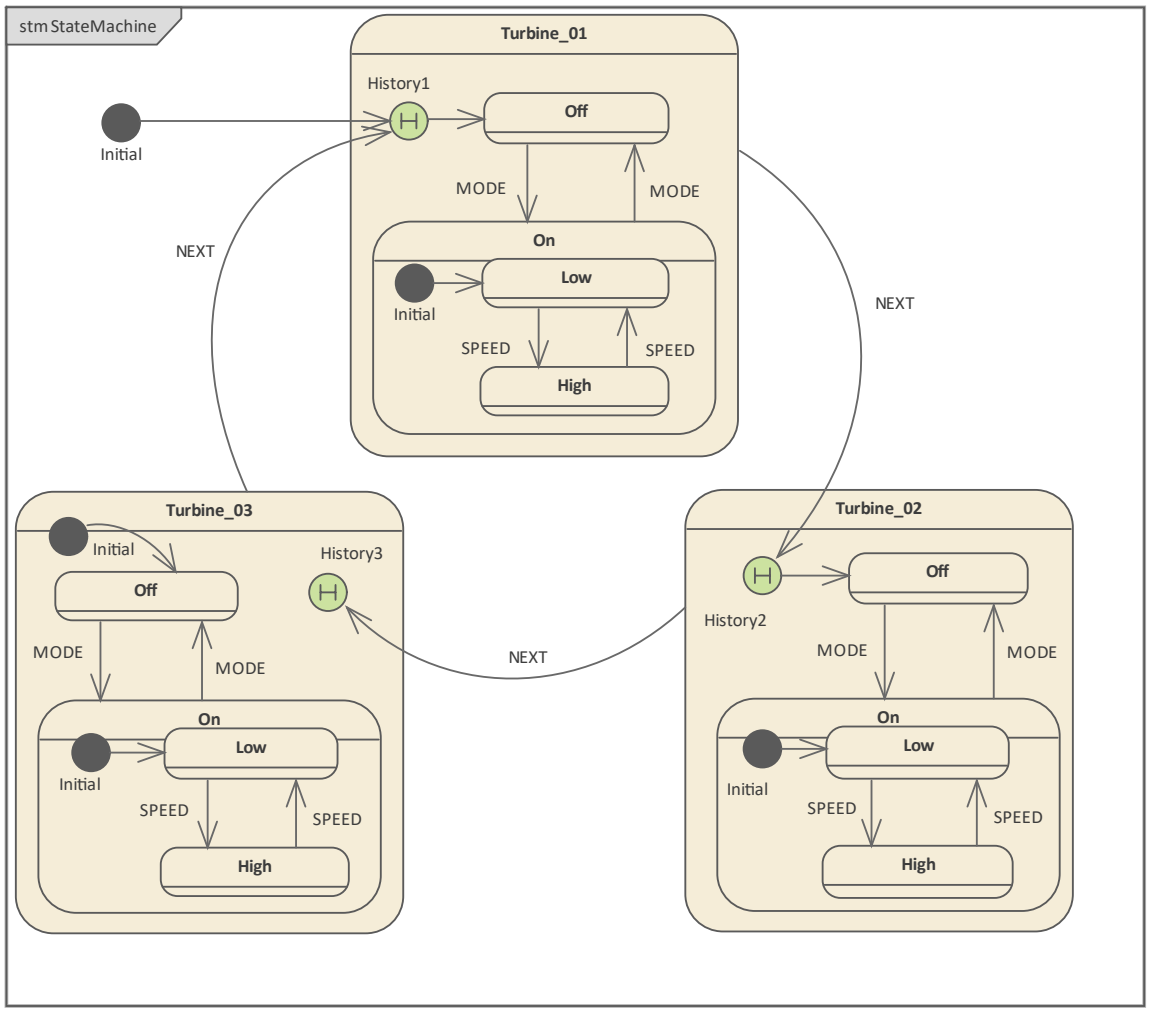
为了更好地观察 Deep History 和 Shallow History 的区别，我们在一次模拟中执行了两个状态机。



*DeepTurbineManager*中的状态机如下图所示：



*ShallowTurbineManager*中的状态机如下图所示：

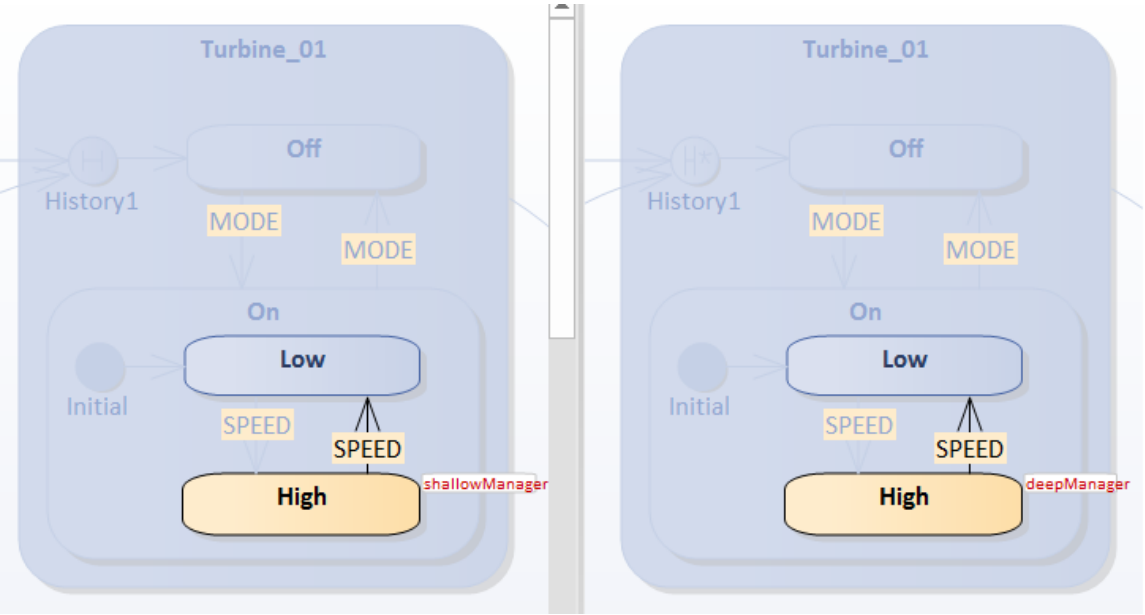


提示：如果您右键单击图表上的历史节点并选择'高级|深度历史选项，您可以在浅层和深层之间切换历史伪状态的类型。

第一时间激活状态

仿真开始后， *Turbine_01*及其子状态*Off*被激活。

触发器：[MODE,序列]

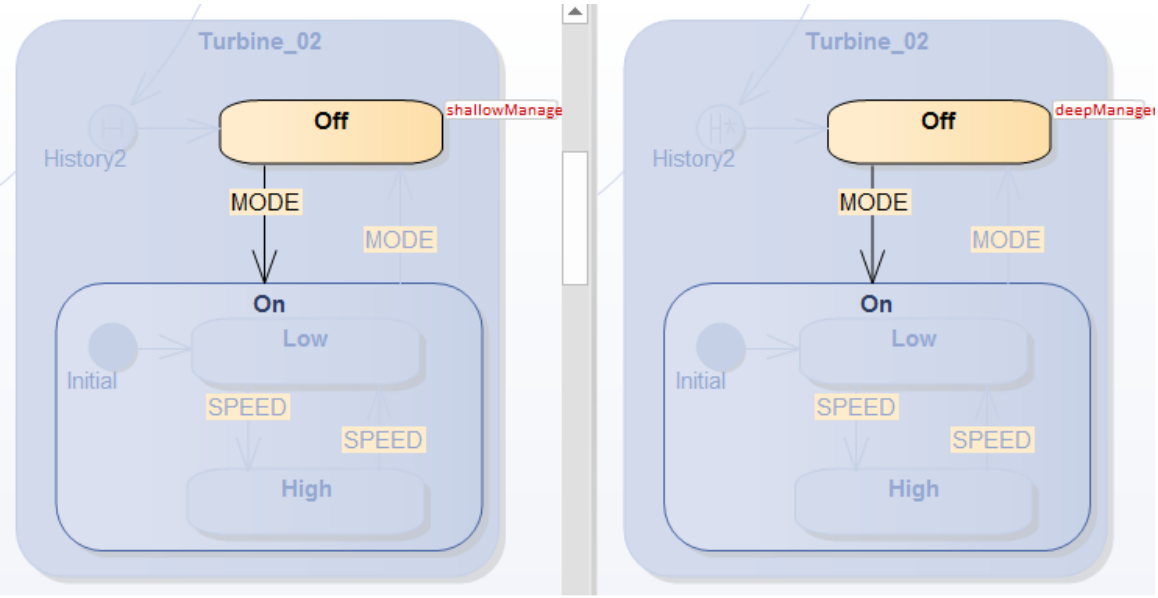


那么活动状态配置包括：

- 涡轮_01
- Turbine_01.On
- Turbine_01.On.High

这适用于*deepManager*和*shallowManager*。

触发器：[序列]



从仿真窗口（仿真>动态仿真仿真>模拟器>打开仿真窗口）可以观察到这条序列：

01 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On_High 退出
02 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On EXIT
03 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01 退出

04 shallowManager[ShallowTurbineManager].Turbine_01__TO__History2_105720_61730影响

05 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02 ENTRY

06 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02 DO

07 shallowManager[ShallowTurbineManager].History2_105720__TO__Off_61731影响

08 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off ENTRY

09 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off DO

注记：由于*deepManager*和 shallowManager 的 trace 完全一样，所以*deepManager*的*trace*是从这个序列中过滤掉的。

我们可以了解到：

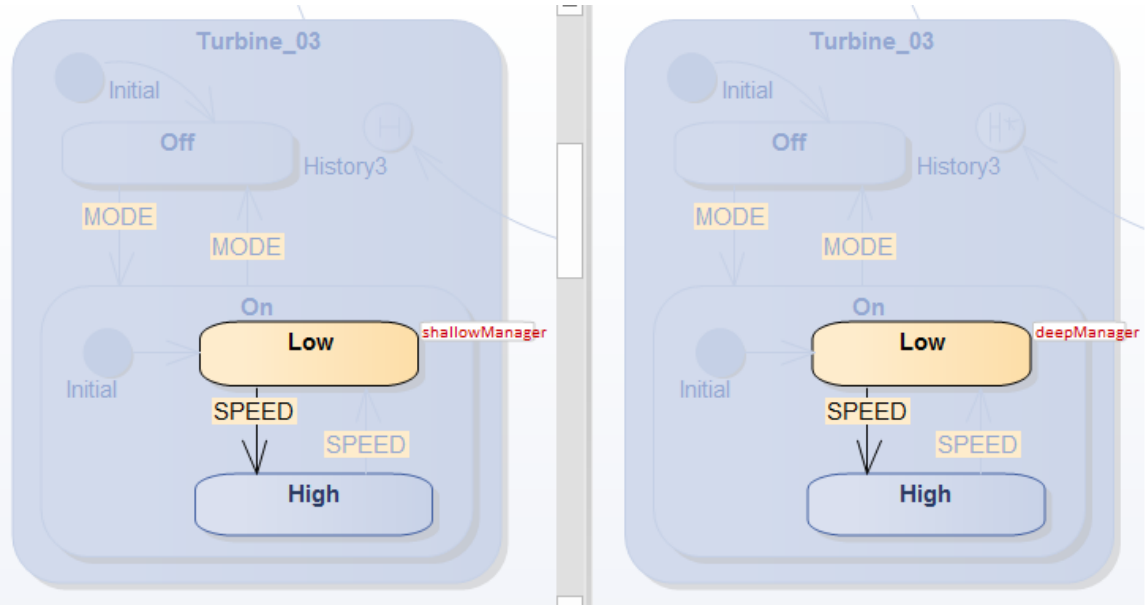
- 退出复合状态从活动状态配置中的最状态开始（参见跟踪序列中的第 01 - 03 行）
- 默认历史转移
  仅当执行导致历史节点（参见第 04 行）并且该状态之前从未处于活动状态（参见第 07 行）时才会执行

那么活动状态配置包括：

- 涡轮_02
- Turbine_02.Off

这适用于*deepManager*和*shallowManager*。

触发器：[序列,MODE]



从仿真窗口可以观察到这个序列：

触发器[下一个]

01 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off 退出

02 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02 退出

03 shallowManager[ShallowTurbineManager].Turbine_02__TO__History3_105713_61725影响

04 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03 ENTRY

05 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03 DO

06 shallowManager[ShallowTurbineManager].Initial_105706__TO__Off_61718影响

07 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_Off ENTRY

08 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_Off DO

触发器[模式]

信息省略...

注记：由于*deepManager*和 shallowManager 的 trace 完全一样，所以*deepManager*的*trace*是从这个序列中过滤掉的。

我们可以了解到：

- 由于没有默认历史转移
  为*History3*定义，执行状态的标准默认输入；在区域所包含的区域中发现了一个 Initial 节点，因此转移源自*Initial*已启用（参见第 06 行）

那么活动状态配置包括：

- 涡轮_03
- Turbine_03.On
- Turbine_03.On.Low

这适用于*deepManager*和*shallowManager*。

国家的历史条目

作为参考，我们展示了每个涡轮机在第一次激活后的深度历史快照：

涡轮_01

- Turbine_01.On
- Turbine_01.On.High

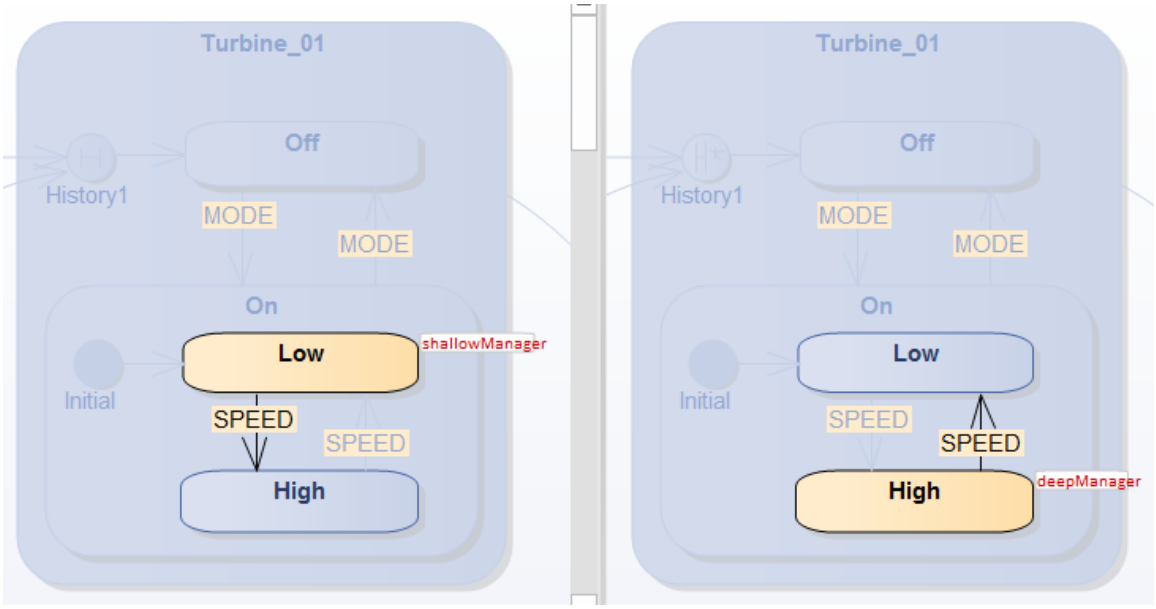
涡轮_02

- Turbine_02.Off

涡轮_03

- Turbine_03.On
- Turbine_03.On.Low

当我们进一步简单地 NEXT 时，触发器将再次被激活。



从仿真窗口可以观察到这个序列：

对于浅层管理器：

触发器[下一个]

01 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low 退出

02 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On EXIT

03 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03 退出

04 shallowManager[ShallowTurbineManager].Turbine_03__TO__History1_105711_61732影响

05 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01 ENTRY

06 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01 DO

07 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On ENTRY

08 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On DO

09 shallowManager[ShallowTurbineManager].Initial_105721__TO__Low_61729影响

10 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On_Low ENTRY

11 shallowManager[ShallowTurbineManager].StateMachine_Turbine_01_On_Low DO

我们可以了解到：

- shallowHistory 节点将 Turbine_01 恢复到 Turbine_01.On
- 那么复合状态区域所包含的区域将被*Initial*节点激活，在*Low*激活

对于 deepManager：

触发器[下一个]

01 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On_Low 退出

02 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On EXIT

03 deepManager[DeepTurbineManager].StateMachine_Turbine_03 退出

04 deepManager[DeepTurbineManager].Turbine_03__TO__History1_105679_61708影响

05 deepManager[DeepTurbineManager].StateMachine_Turbine_01 条目

06 deepManager[DeepTurbineManager].StateMachine_Turbine_01 DO

07 deepManager[DeepTurbineManager].StateMachine_Turbine_01_On ENTRY

08 deepManager[DeepTurbineManager].StateMachine_Turbine_01_On_High ENTRY

我们可以了解到：

- *deepHistory*节点将 Turbine_01 恢复到 Turbine_01.On.High

触发器[NEXT] 退出 Turbine_01 并激活 Turbine_02

*shallowManager*和*deepManager*都激活了 Turbine_02.Off，这是它们退出时的 History 快照。

触发器[NEXT]退出Turbine_02并激活Turbine_03

*shallowManager*和*deepManager*都激活 Turbine_03.On.Low。但是， *shallowManager*和*deepManager*的顺序是不同的。

对于*shallowManager* ， *shallowHistory*只能恢复到 Turbine_03.On。由于在 Turbine_03.On 中定义了*Initial*节点，因此转移

源自*Initial*的将被启用并达到 Turbine_03.On.Low。

01 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02_Off 退出

02 shallowManager[ShallowTurbineManager].StateMachine_Turbine_02 退出

03 shallowManager[ShallowTurbineManager].Turbine_02__TO__History3_105713_61725影响

04 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03 ENTRY

05 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03 DO

06 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On ENTRY

07 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On DO

08 shallowManager[ShallowTurbineManager].Initial_105727__TO__Low_61728影响

09 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low ENTRY

10 shallowManager[ShallowTurbineManager].StateMachine_Turbine_03_On_Low DO

对于*deepManager* ， *deephistory*可以直接恢复到 Turbine_03.On.Low。

01 deepManager[DeepTurbineManager].StateMachine_Turbine_02_Off 退出

02 deepManager[DeepTurbineManager].StateMachine_Turbine_02 退出

03 deepManager[DeepTurbineManager].Turbine_02__TO__History3_105680_61701影响

04 deepManager[DeepTurbineManager].StateMachine_Turbine_03 ENTRY

05 deepManager[DeepTurbineManager].StateMachine_Turbine_03 DO

06 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On ENTRY

07 deepManager[DeepTurbineManager].StateMachine_Turbine_03_On_Low ENTRY

事件宏：EVENT_PARAMETER

EVENT_PARAMETER 是一个函数宏，用于访问信号实例的属性，在状态的行为中，转移的守护和效果。该宏将根据仿真语言扩展为可执行代码。

访问默认宏定义

功能区|开发 |源代码 |选项 |编辑代码模板|语言 | Stm事件参数

用途格式

%EVENT_PARAMETER（信号类型，信号属性名称）%

例如： A信号'MySignal'有两个属性，'foo: int '和'bar: int '；这些用例是有效的：

- 转移
  的效果：%EVENT_PARAMETER(MySignal, foo)%
- 状态的行为：%EVENT_PARAMETER(MySignal, bar)%
- 转移
  的守卫：%EVENT_PARAMETER(MySignal, bar)% > 10
- 状态的行为：%EVENT_PARAMETER(MySignal, bar)%++
- 跟踪值到仿真窗口：%TRACE(EVENT_PARAMETER(MySignal, 富))%

宏扩展示例

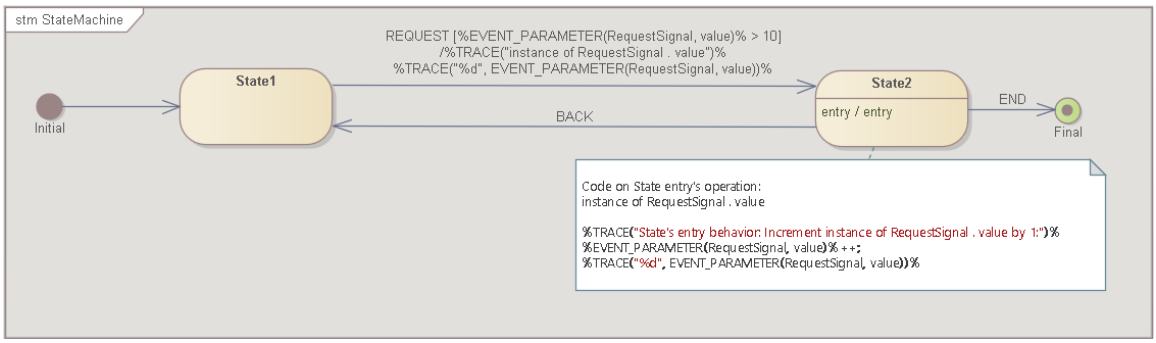
对于带有属性 "value"的信号 "MySignal"，转移
的宏扩展示例转移
带信号触发：%EVENT_PARAMETER(MySignal, value)%

C	((MySignal*)信号)->值
C++	static_cast<MySignal*>(signal)->value
C #	((MySignal)信号).value
Java	((EventProxy.MySignal)信号).value
JavaScript	信号值

示例

这个例子演示了如何在转移

中使用 EVENT_MACRO转移
的效果、守卫和状态的行为。



在运行模拟时，
(1) 触发 REQUEST 并为属性值指定数字1 。
由于守卫的条件为false ，因此活动状态将保持为 State1。
(2) 触发 REQUEST 并为属性值指定数字 11。
由于守卫的条件为真，活动状态将从State1变为State2；
转移
的效果被执行。在这里，我们将信号属性的运行时值跟踪到模拟窗口。
State2 的行为被执行。在这里，我们增加了信号属性的运行时间值并将其跟踪到模拟窗口。

[32608107] [Part1:TransactionServer]转移
影响：影响
[32608118] [Part1:TransactionServer] 入口行为：StateMachine_State1
[32608124] [Part1:TransactionServer]行为：StateMachine_State1
[32608877] [第 1 部分：TransactionServer] 完成：TransactionServer_StateMachine_State1
[触发器] 等待简单
[32613165] 命令：**广播 REQUEST.RequestSignal(1)**
[32613214] [Part1：TransactionServer]事件排队：REQUEST.RequestSignal（值： 1 ）
[32613242] [Part1:TransactionServer]事件调度：REQUEST.RequestSignal（值： 1 ）
[触发器] 等待简单
[32619541] 命令：**广播 REQUEST.RequestSignal(11)**
[32619546] [Part1：TransactionServer]事件排队：REQUEST.RequestSignal（值：11 ）
[32619551] [Part1：TransactionServer]事件调度：REQUEST.RequestSignal（值：11 ）
[32619557] [Part1:TransactionServer] 退出行为：StateMachine_State1
[32619562] [Part1:TransactionServer]转移
影响：影响
[32619567] **RequestSignal 的实例。价值**
[32619571] **11**
[32619576] [Part1:TransactionServer] 入口行为：StateMachine_State2
[32619584]**状态的进入行为：增加 RequestSignal 的实例。值1 ：**
[32619590] **12**
[32619594] [Part1:TransactionServer]行为：StateMachine_State2

[32620168] [第 1 部分：TransactionServer] 完成：TransactionServer_StateMachine_State2
[触发器] 等待简单
[32622266] 命令：广播结束
[32622272] [Part1：TransactionServer]事件排队：END
[32622310] [Part1：TransactionServer]事件调度：END
[32622349] [Part1:TransactionServer] 退出行为：StateMachine_State2
[32622359] [Part1:TransactionServer]转移
影响：影响
[32622896] [第 1 部分：TransactionServer] 完成：TransactionServer_VIRTUAL_SUBMACHINESTATE

限制和解决方法

由于宏扩展涉及类型转换，因此用户有责任确保类型转换有效。
以下是模型中类型转换会遇到问题的一些常见情况的解决方法。

- **一个转移**
  具有不同信号类型的多个触发器

例如，一个转移
有triggerA（指定SignalA）和triggerB（指定SignalB）； A这个转移
时，宏 %EVENT_PARAMETER(SignalA, attributeOfA)% 将不起作用转移
由 triggerB 触发。
我们建议创建两个转换，一个使用 triggerA（指定 SignalA），另一个使用 triggerB（SignalB）。

- **一个状态是不同信号触发的多个Transition的目标**

例如，TransitionA 和 TransitionB 都针对 MyState，TransitionA 由 SignalA 触发，TransitionB 由 SignalB 触发。
如果在状态的行为代码中使用了 EVENT_PARAMETER，则一个宏将无法同时适用于两种情况。
我们建议将处理信号属性的逻辑从状态的行为转移到转移
的效果。

- **自定义模板和生成的代码**

用户也可以更改默认模板，在类型转换前添加运行时间类型标识（RTTI）。用户也可以在生成后修改生成的代码，也可以满足编译后的模拟目的。

SysML参数仿真

Enterprise Architect提供与 OpenModelica 和 MATLAB Simulink 的集成，以支持对 SysML模型在不同情况下的行为方式进行快速而可靠的评估。

OpenModelica 库是提供许多有用类型、函数和模型的综合资源。在Enterprise Architect中创建 SysML 模型时，您可以参考这些库中可用的资源。

Enterprise Architect的 MATLAB 集成通过 MATLAB API 进行连接，允许您的Enterprise Architect仿真和其他脚本根据任何可用的 MATLAB 函数和表达式的值进行操作。您可以通过 Solver类调用 MATLAB，或将您的模型导出到 MATLAB Simulink、Simscape 和/或状态流。

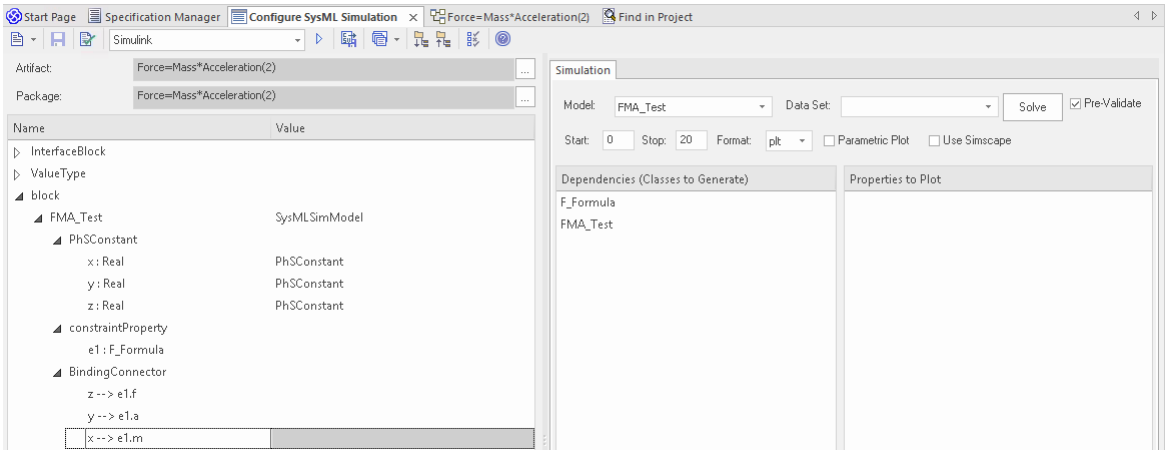
SysML仿真特征

这些部分描述了定义参数模型、使用附加信息对模型进行注释以驱动模拟以及运行模拟以生成结果图的过程。

部分	描述
SysML 参数模型简介	SysML 参数模型支持关键系统参数的工程分析，包括评估关键指标，如性能、可靠性和其他物理特性。这些模型通过捕获基于复杂数学关系的可执行约束，将需求模型与系统设计模型相结合。参数图是专门的内部块图，可帮助建模者将行为和结构模型与工程分析模型（例如性能、可靠性和质量属性）结合起来。 有关 SysML 参数模型概念的更多信息，请参阅 OMG SysML 官方网站及其链接资源。
创建参数模型	概述开发用于仿真的 SysML模型元素、在配置SysML仿真窗口中配置这些元素以及观察仿真结果。
工件适合	Enterprise Architect帮助您扩展 SysML 参数模型的实用性，方法是使用允许模拟模型的额外信息对它们进行注释。然后将生成的模型生成为可以使用 MATLAB Simulink 或 OpenModelica 求解（模拟）的模型。 模拟属性针对您的仿真模型进行工件。这保留了您的原始模型并支持针对单个 SysML模型配置的多个模拟。仿真工件在工件的工具箱上找到
用户接口	SysML 模拟的用户界面在配置模拟仿真主题中进行了描述。
使用数据集进行模型分析	使用仿真配置，SysML块可以有多个针对它定义的数据集。这允许在SysML模型的模拟上运行可重复的变化。
SysPhS 标准支持	*SysPhS* 标准是交互物理和辅助信号流仿真的 *SysML*扩展。它定义了在SysML模型和 Modelica模型或 Simulink/Simscape模型之间转换的标准方法，为共享仿真提供了一种更简单的基于模型的方法。请参阅*SysPhS* 标准支持帮助帮助。
例子	为了帮助您了解如何创建和模拟 SysML 参数模型，我们提供了三个示例来说明三个不同的领域。所有三个示例都使用 OpenModelica 库。 *SysML*仿真示例主题中描述了这些示例以及您可以从中学到的东西。

配置SysML仿真

配置SysML仿真窗口是一个界面，您可以通过该界面提供运行时参数以执行 SysML模型的模拟。工件中的模拟基于元素定义的模拟配置。

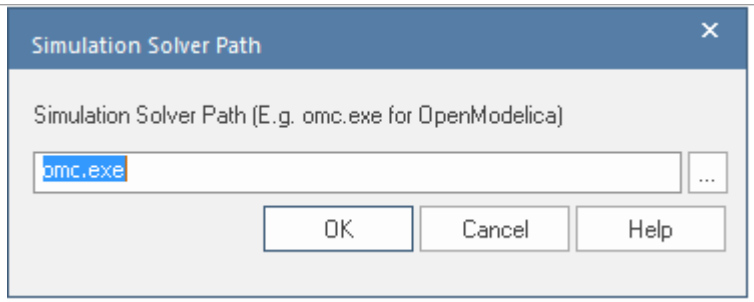


访问

功能区	仿真>系统行为> Modelica/Simulink >配置管理器
其它	双击具有工件构造型的属性。

工具栏选项

选项	描述
	单击下拉箭头并从以下选项中进行选择： • （如果尚未选择工件配置，则从一个工件配置中选择一个配置和负载） • 创建工件— 创建新的工件或选择并加载现有的配置 • 选择包- 选择一个包以扫描 SysML 元素以配置模拟 • Reload — 重新加载配置管理器并更改当前包 • 配置仿真求解器 - 显示 仿真求解器路径”对话框，您可以在其中键入或浏览要使用的求解器的路径。对于 MATLAB/Simulink，将自动检测路径，因此只有在自动检测存在问题或安装了多个版本的 MATLAB 时才需要更改此设置。

	
	单击此按钮将配置保存到当前工件。
	单击此图标现在专门针对现在配置验证模型。验证结果显示在系统输出窗口的 SysML仿真”选项卡中。您还可以选择一个选项以在执行每个模拟之前自动预验证模型。请参阅仿真标签表中的 预验证”选项。
	单击此图标可展开窗口 名称”列中层次结构中的每个项目。
	单击此图标可折叠窗口 名称”列中模型层次结构中所有展开的项目。
	单击此图标可显示可在模拟中抑制的object类型列表。单击要抑制的每个object的复选框，或单击 全部”按钮以选择要抑制的所有项目。 您还可以使用 选项”列顶部的过滤器栏来仅显示名称中具有指定字母或文本string的项目。
	单击下拉箭头并选择正在运行仿真的应用程序，例如运行或 Simulink。
	单击此按钮可生成、编译和运行当前配置，并显示结果。
	仿真后，以 plt、mat 或 csv 格式生成结果文件。也就是说，使用文件名： • ModelName_res.mat（OpenModelica 的默认值） • ModelName_res.plt 或 • 模型名称_res.csv 单击此按钮指定Enterprise Architect将复制结果文件的目录。
	单击此按钮可从以下选项中进行选择： • 运行Last Code - 执行最近生成的代码 • 生成代码——生成编译或运行代码 • Open仿真— 打开将生成 OpenModelica 或 Simulink 代码的目录 • 编辑模板——使用代码模板编辑器自定义为 OpenModelica 或 Simulink 生成的代码

仿真工件模型

字段	行动

工件	单击…并选择现有的工件图标，然后创建一个新的工件。
包	如果您有一个现有的工件配置模型包，该字段指定了与该属性相关联的默认工件。 否则，单击…图标并浏览并选择包含 SysML模型的包以进行仿真配置。您必须在选择包之前指定（或创建）工件。

包对象

本表讨论了将在配置SysML仿真窗口的 名称”列下列出的 SysML模型中的object类型，这些对象将在模拟中进行处理。每个object类型展开以列出该类型的命名对象，以及需要在 值”列中配置的每个object的属性。

很多级别的object类型、名称和属性都不需要配置，因此对应的 Value”字段不接受输入。如果输入合适且被接受，则在字段的右端会显示一个下拉箭头；当您单击此箭头时，将显示一个可能值的简短列表以供选择。某些值（例如部件的 部件”）添加更多的参数层和属性，您可以单击…按钮再次选择和设置参数的值。对于数据集，输入对话框允许您输入或导入值，例如初始值或默认值；请参阅模型分析使用数据集帮助主题。

元素类型	行为
值类型	ValueType 元素要么从原始类型泛化，要么被 SysMLSimReal 替代以进行仿真。
块	映射到 SysMLSimClass 或 SysMLSimModel 元素的块元素支持数据集的创建。如果您在上下文中定义了多个数据集（可以泛化），则必须将其中一个标识为默认值（使用时间菜单选项 设置为默认数据集”）。 由于 SysMLSimModel 可能是模拟的顶级元素，并且不会被概括，如果您定义了多个数据集，则在模拟期间选择要使用的数据集。
属性	指定常量或变量及其设置的首选方法是在属性本身上使用 SysPhS 构造型 PhSConstant 和 PhSVariable。 PhSVariable 构造型具有内置属性*isContinuous*、*isConserved*和*changeCycle*。 该属性将列在 PhSConstant 或 PhSVariable 下，并且该值不能更改。 也可以在配置SysML仿真窗口中定义设置。在这种情况下，它们将列在 属性”下。 块内的属性可以配置为 SimConstants 或 SimVariables。对于 SimVariable，您可以配置以下属性： • isContinuous — 确定属性值是连续变化（'true'，默认值）还是离散变化（'false'） • 属性— 确定属性的值是否保留（'true'）（'false'，默认值）；在为物理相互作用建模时，相互作用包括保守物理物质的交换，例如电流、力或流体流动 • changeCycle — 指定离散属性值更改的时间间隔；默认值为 0” - changeCycle 只能设置为 0 以外的值 isContinuous = 'false' - changeCycle 的值必须为正或等于 0
端口	无需配置。
模拟函数	函数被创建为块或约束块中的操作，原型为 SimFunction”。 配置仿真窗口无需配置。

概括	无需配置。
捆绑连接器	将属性绑定到约束属性的参数。 无需配置；但是，如果属性不同，系统会提供同步它们的选项。
连接器	连接两个端口。 配置仿真窗口无需配置。但是，您可能必须通过确定属性属性是否应该设置为 "False"（对于潜在的，以便建立相等耦合）或 "True"（对于流/保守的属性）来配置端口类型的属性, 从而建立和到零的耦合)。
约束块	无需配置。

仿真标签

此表描述了配置SysML仿真窗口上的 "仿真"选项卡的字段。

字段	行动
模型	单击下拉箭头并选择模拟的顶级节点（SysMLSimModel元素）。该列表填充有定义为顶级模型节点的块的名称。
数据集	单击下拉箭头并选择所选模型的数据集。
预验证	选中此复选框可在执行模型的每次模拟之前自动验证模型。
开始	类型在开始模拟之前的初始等待时间，以秒为单位（默认值为 0）。
停止	类型模拟将执行的秒数。
格式	单击下拉箭头并选择 "plt"、 "csv"或 "mat"作为结果文件的格 ，其他工具可能会使用这些格式。
参数图	• 选中此复选框以在 y 轴上绘制图例A ，在 x 轴上图例B • 取消选中复选框以在 y 轴上绘制图例，在 x 轴上绘制时间 注记：选中复选框后，您必须选择两个要绘制的属性。
使用Simscape	（如果选定的数学工具是 Simulink）如果您还想在 Simscape 中处理仿真，请选中该复选框。
依赖项	列出模拟该模型必须生成的类型。
属性的属性	提供与模拟有关的变量属性列表。选中要绘制的每个属性对应的复选框。

创建参数模型

在本主题中，我们将讨论如何开发用于仿真的 SysML模型元素（假设已有 SysML 建模知识），在配置SysML仿真窗口中配置这些元素，并观察在一些不同定义和建模方法下的仿真结果。本章提供的 SysML仿真示例中的图表和屏幕快照说明了这些要点。

创建参数模型时，您可以应用以下三种方法之一来定义约束方程：

- 在块元素上定义内联约束方程
- 创建可重用的约束块，以及
- 使用连接约束属性

您还将考虑：

- 物理交互中的流动
- 默认值和初始值
- 仿真函数
- 价值分配，以及
- 包导入

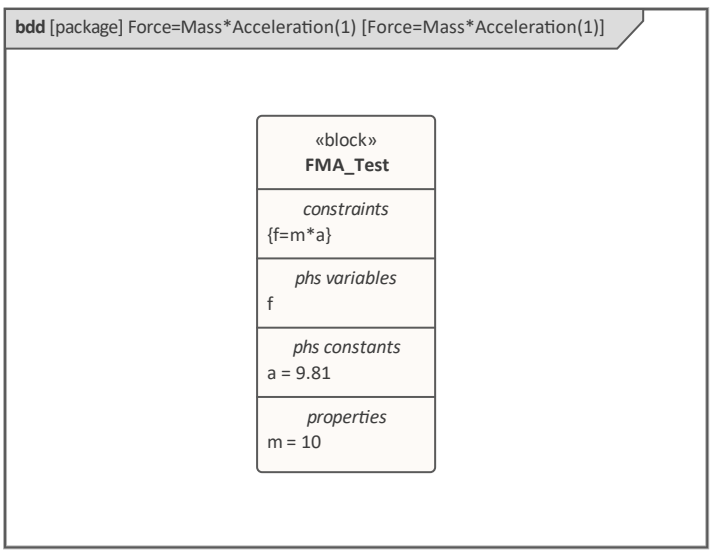
访问

功能区	仿真>系统行为> Modelica/Simulink >配置管理器

在块上定义内联约束方程

直接在块中定义约束很简单，也是定义约束方程的最简单方法。

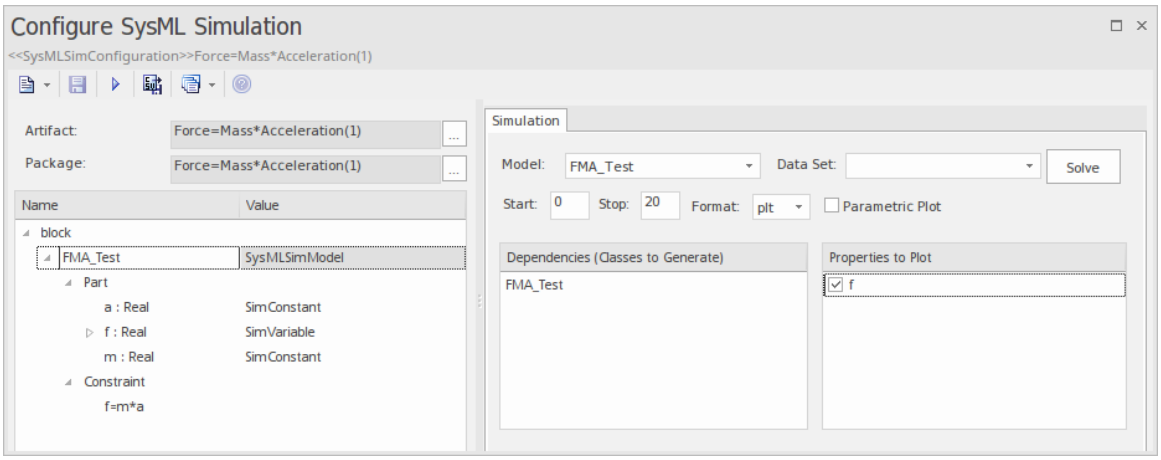
在这个图中，约束'f = m * a'被定义在一个块元素。



提示：您可以在一个块中定义多个约束。

1. 创建一个SysMLSim配置工具工件=Mass*Acceleration(1)'并将其指向包'FMA_Test'
2. 对于 "FMA_Test"，在 "值"列中设置 "SysMLSimModel"。

3. 对于 á"、 ḿ"和 f"部分，在 值"列中：将 á"和 ḿ"设置为 ƤhSConstant"，（可选）将 f"设置为 "PhSVariable"。
4. 在 仿真"选项卡的 要绘制的属性"面板中，选中 f"复选框。
5. 单击求解按钮运行模拟。

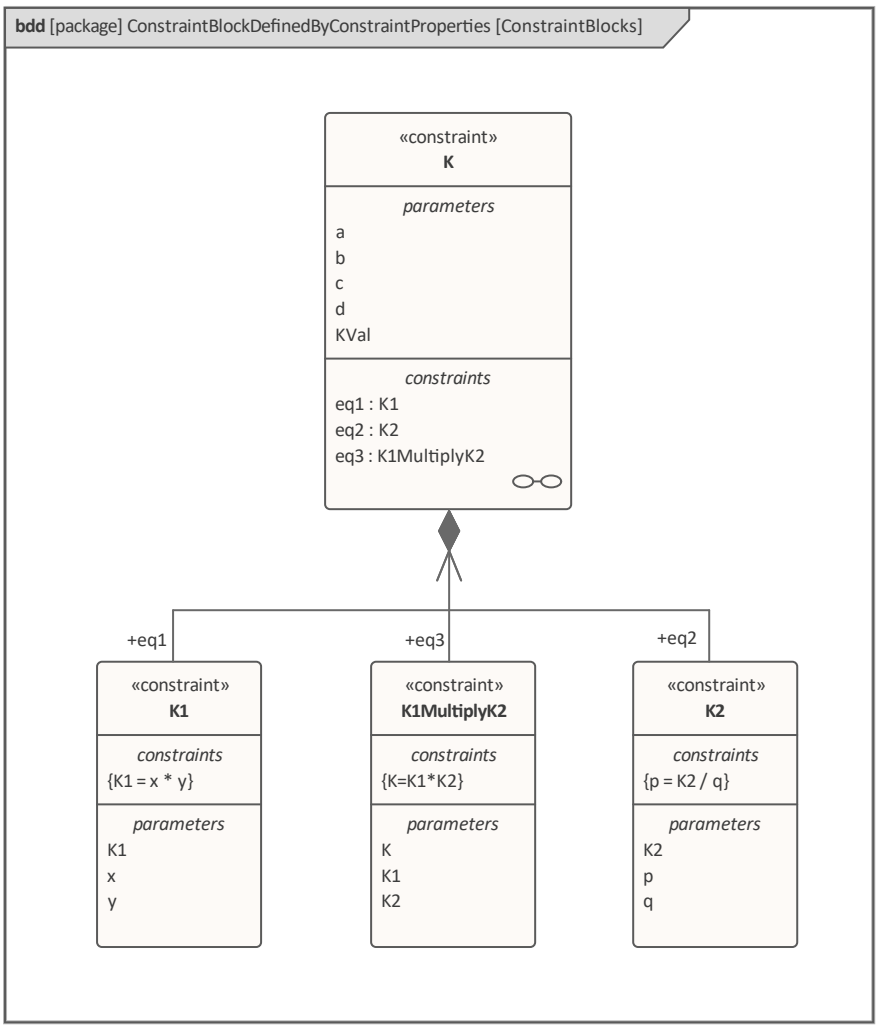


应A f = 98. 1 （来自 10 * 9.81）绘制图表。

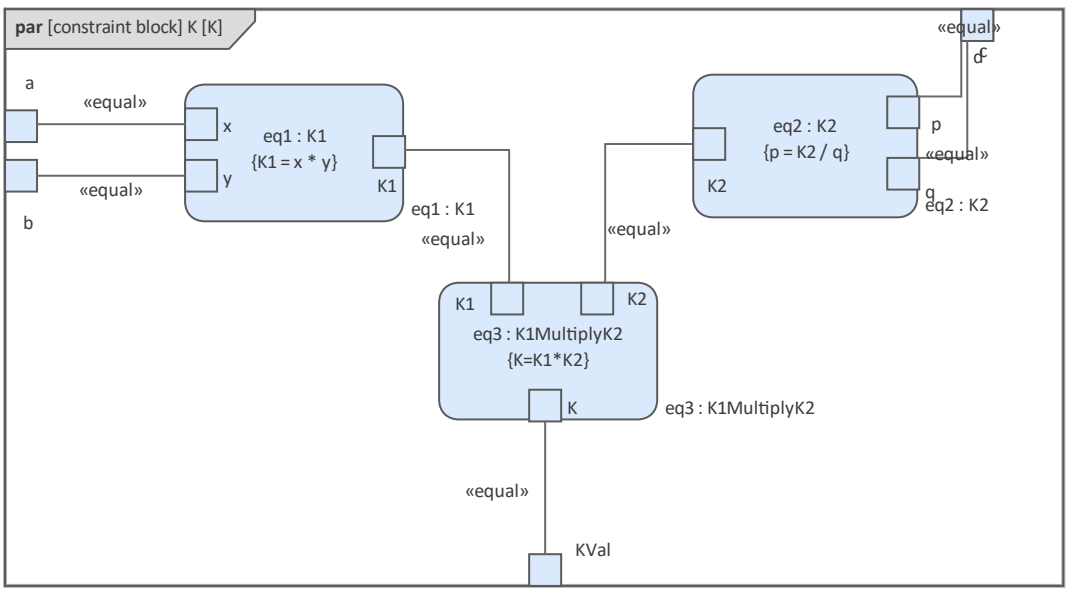
连接约束属性

在 SysML 中，ConstraintBlocks 中存在的约束属性可用于在定义约束时提供更大的灵活性。

在这个图中，ConstraintBlock ' K ' 定义了参数'a'、'b'、'c'、'd'和'属性'，以及三个约束'eq1'、'eq2'和'eq3'，类型为 'K1 '、'K2' 和 'K1MultiplyK2' 分别。

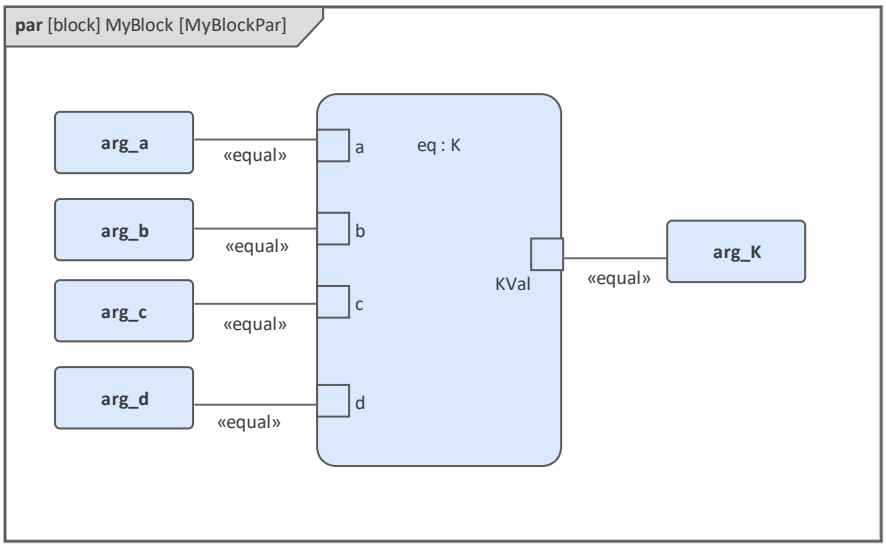


在 ConstraintBlock ' K ' 中创建一个参数图，并使用捆绑连接器将参数连接到约束属性，如图所示：

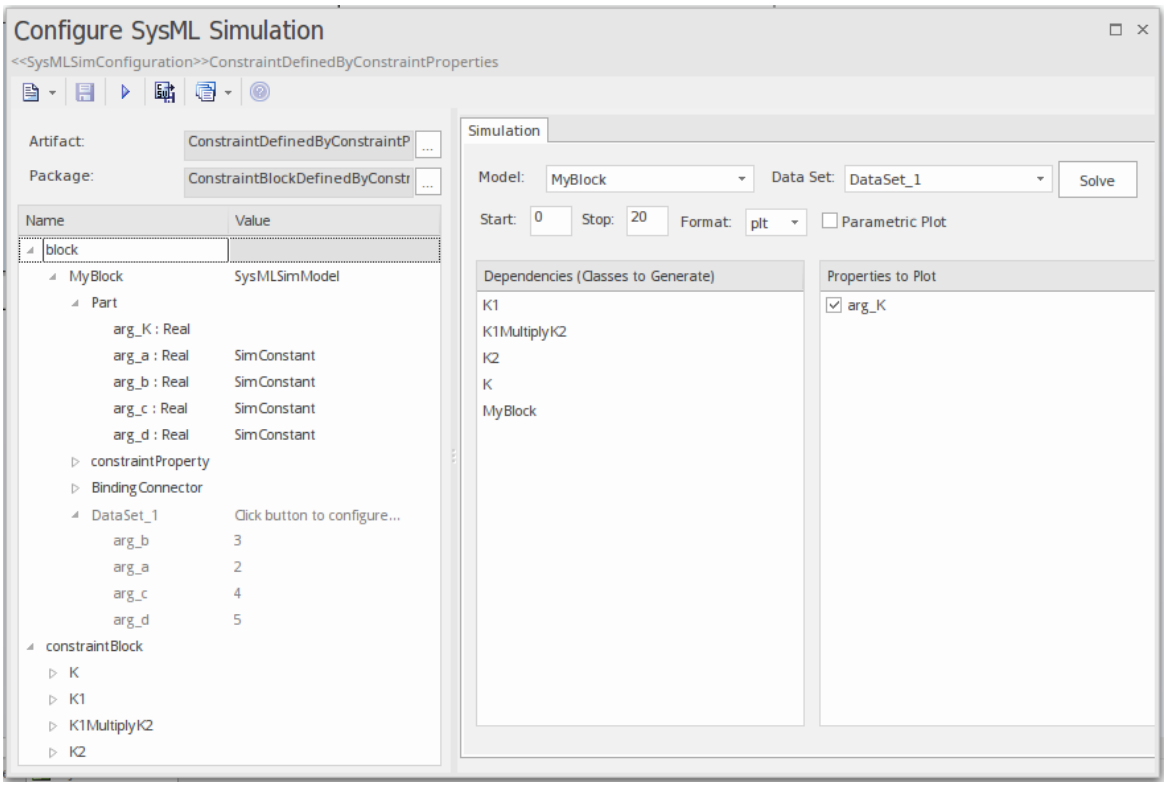


- 创建具有五个属性（零件）的模型属性
- 为属性创建一个约束属性 'eq' 并显示参数

- 将属性绑定到参数



- 在数据集中提供值 (arg_a = 2, arg_b = 3, arg_c = 4, arg_d = 5)
- 在 "配置SysML仿真"对话框中，将 "模型"设置为 "MyBlock"，将 "数据集"设置为 "DataSet_1"
- 在 "propertyto Plot"面板中，选中 "属性"复选框
- 点击 Solve 按钮运行模拟



将计算并绘制结果 120（计算为 2 * 3 * 4 * 5）。这与我们用笔和纸进行扩展时相同： K = K1 * K2 = (x*y) * (p*q)，然后绑定值 (2 * 3) * (4 * 5)；我们得到 120。

有趣的是，我们有意将 K2 的方程定义为 "p = K2 / q"，这个例子仍然有效。

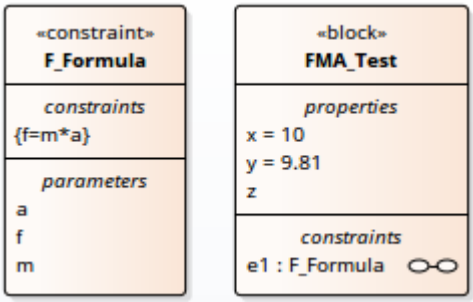
在这个例子中，我们可以很容易地将 K2 求解为 p * q，但在一些复杂的例子中，从方程中求解变量是非常困难的；但是， Enterprise Architect SysMLSim 仍然可以做到这一点。

总之，该示例向您展示了如何通过构造约束属性来定义具有更大灵活性的 ConstraintBlock。虽然我们只在 ConstraintBlock 中演示了一层，但这种机制将适用于任意级别的复杂模型。

创建可重用的约束块

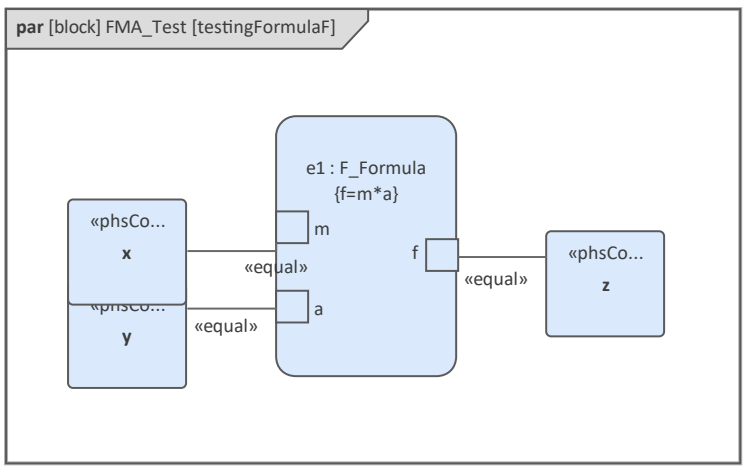
如果一个方程在许多块中通用，则可以创建一个 ConstraintBlock 用作每个块中的约束属性。这些是我们根据前面的示例所做的更改：

- 创建一个 ConstraintBlock元素'F_Formula' 具有三个参数 'a'、'm' 和 'f'，以及一个约束 'f = m * a'

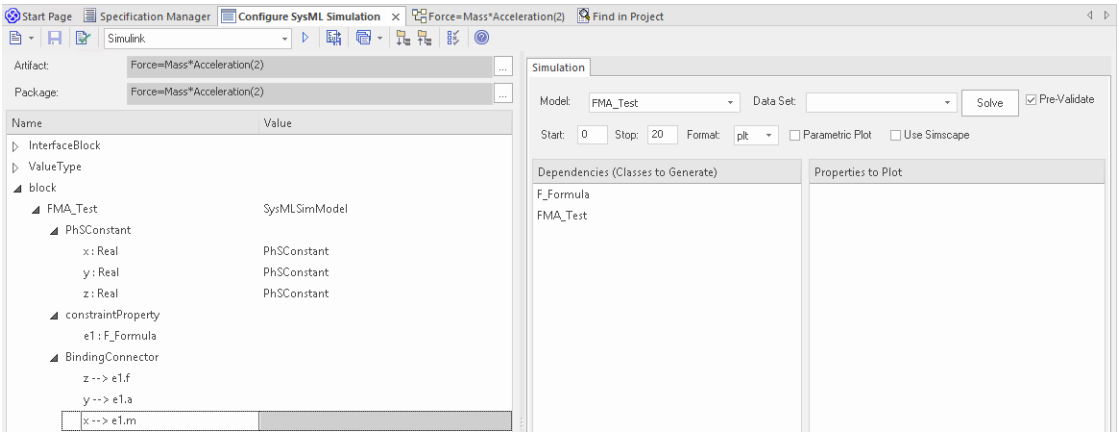


提示：如果属性类型为空，将应用原始类型 *'Real'*

- 用'x'、'y'和'z'三个属性创建一个块'属性'，并分别赋予'x'和'y'默认值'10'和'9.81'
- 在 "FMA_Test" 中创建一个参数图，显示属性 "x"、"y" 和 "z"
- 创建一个 ConstraintProperty 'e1' 类型为 'F_Formula' 并显示参数
- 在'x-m'、'y-a'和'f-z'之间划捆绑连接器，如图所示：



- 在元素中创建一个工件并对其进行配置，如图所示：
  - 在 "值" 列中，将 "FMA_Test" 设置为 "SysMLSimModel"
  - 在 "值" 列中，将 "x" 和 "y" 设置为 "PhSConstant"
  - 在 "要绘制的属性" 面板中，选中 "Z" 复选框
  - 单击求解按钮运行模拟



应A f = 98. 1 （来自 10 * 9.81）绘制图表。

物理交互中的流动

在对物理相互作用进行建模时，应将电流、力、扭矩和流速等守恒物理物质的交换建模为流，并且应将流变量设置为属性 "isConserved"。

连接建立两种不同类型的耦合，取决于流属性是潜在的（默认）还是流（守恒）：

- 等式耦合，用于潜在（也称为努力）属性
- 和零耦合，用于流（守恒）属性；例如，根据电学领域的基尔霍夫电流定律，电荷守恒使所有电荷流入一个和为零的点

在 "ElectricalCircuit"示例的生成 OpenModelica 代码中：

连接器充电端口

流动电流 i； // 如果 'isConserved' = true 将生成流关键字

电压v；

结束充电端口；

电路模型

源；

电阻电阻器；

接地；

方程

连接（源.p，电阻器.n）；

连接（接地.p，源.n）；

连接（电阻器.p，源.n）；

结束电路；

每个连接方程实际上扩展为两个方程（ChargePort 中定义了两个属性），一个用于等式耦合，另一个用于和零耦合：

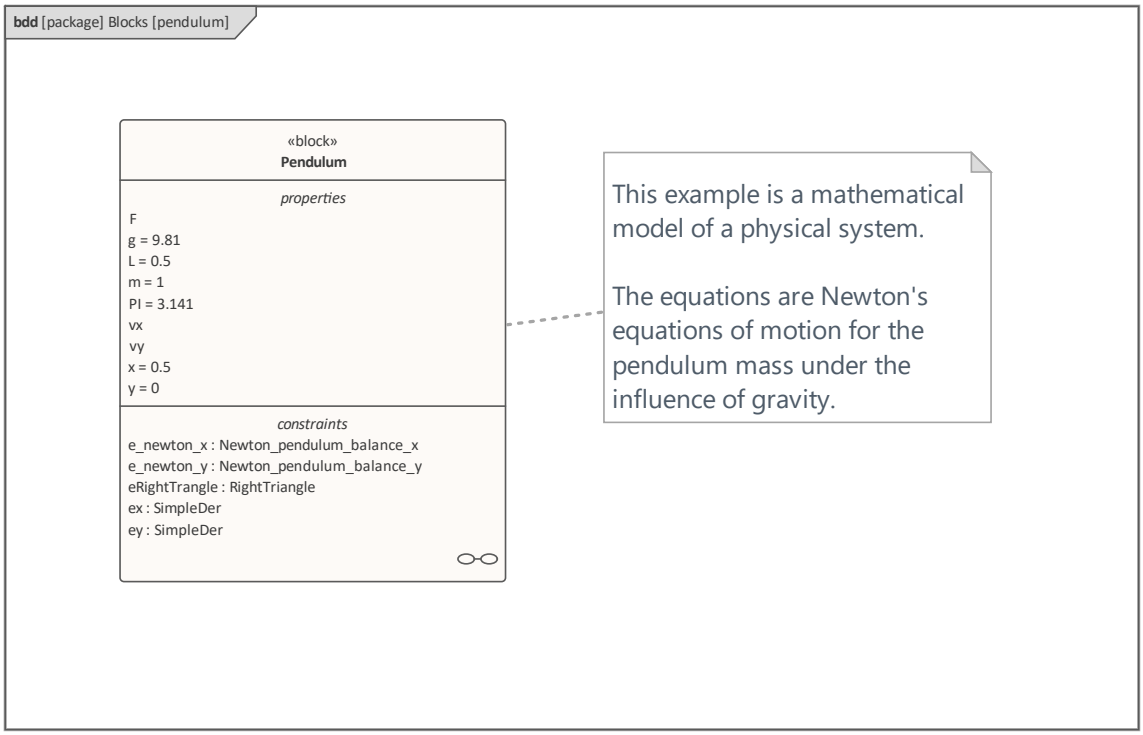
源.pv = 电阻器.nv;

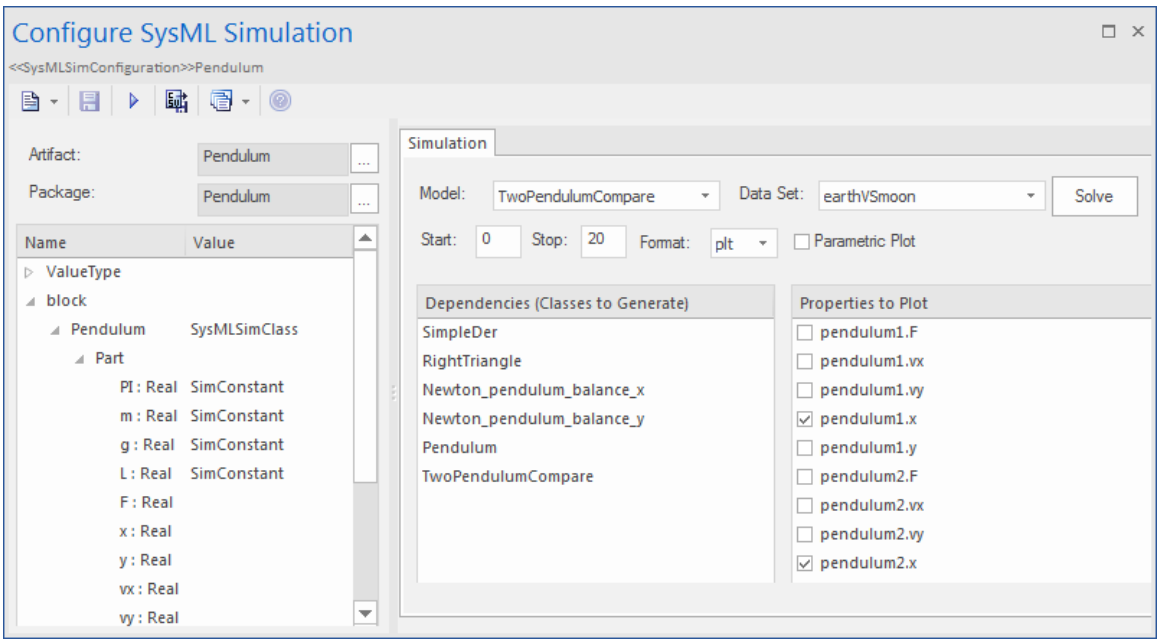
源.pi + 电阻.ni = 0;

默认值和初始值

如果在 SysML属性元素（ 属性”对话框 > 属性”页面 > 初始”字段）中定义了初始值，则它们可以作为 PhSConstant 的默认值或 PhSVariable 的初始值加载。

在这个 Pendulum 示例中，我们为属性 g”、 L”、 m”、 PI”、 x”和 y”提供了初始值，如图左侧所示。由于“PI”（数学常数）、 m”（摆锤的质量）、 g”（重力因子）和 L”（摆锤的长度）在模拟过程中不会发生变化，因此将它们设置为 PhSConstant”。





生成的 Modelica 代码如下所示：

类摆
参数 Real PI = **3.141；**
参数 Real m = **1** ;
参数 Real g = **9.81** ;
参数 Real L = **0.5** ;
实F；
实数 x **(start=0.5)** ;
实 y **(start=0)** ;
真正的vx；
真正的 vy;
……
方程
……
结束摆；

- 属性'PI'、'm'、'g'和' L '是常数，并作为声明方程生成
- 属性 x‴和 y‴是可变的；它们的起始值分别为 0.5 和 0，初始值作为修改生成

仿真函数

仿真函数是编写复杂逻辑A有用工具，并且易于用于约束。本节介绍 TankPI 示例中的一个函数。
在 ConstraintBlock 'Q_OutFlow' 中，函数'LimitValue' 被定义并在约束中使用。

- 在块或约束块上，创建一个操作（本例中为 L‴imitValue”）并打开特征窗口的 操作”选项卡

- 给操作定型 "SimFunction"
- 定义参数并将方向设置为 "输入/输出"

提示：多个参数可以定义为'out'，调用者取值的格式为：

*(out1, out2, out3) = function_name(in1, in2, in3, in4, ...); //*方程形式

*(out1, out2, out3) := function_name(in1, in2, in3, in4, ...); //*报表形式

- 在属性窗口的 "代码"选项卡的文本字段中定义函数体，如图：

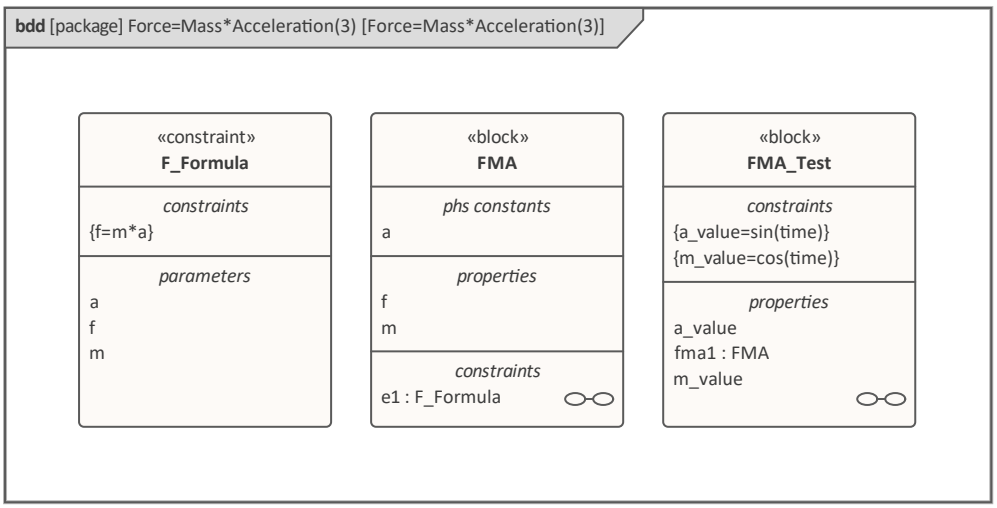
```
pLim :=
如果 p > pMax 那么
最大
else如果 p < pMin 那么
pMin
else
p;
```

生成代码时， Enterprise Architect将收集所有在 ConstraintBlocks 和 Blocks 中定义为 "SimFunction"的操作，然后生成类似于以下内容的代码：

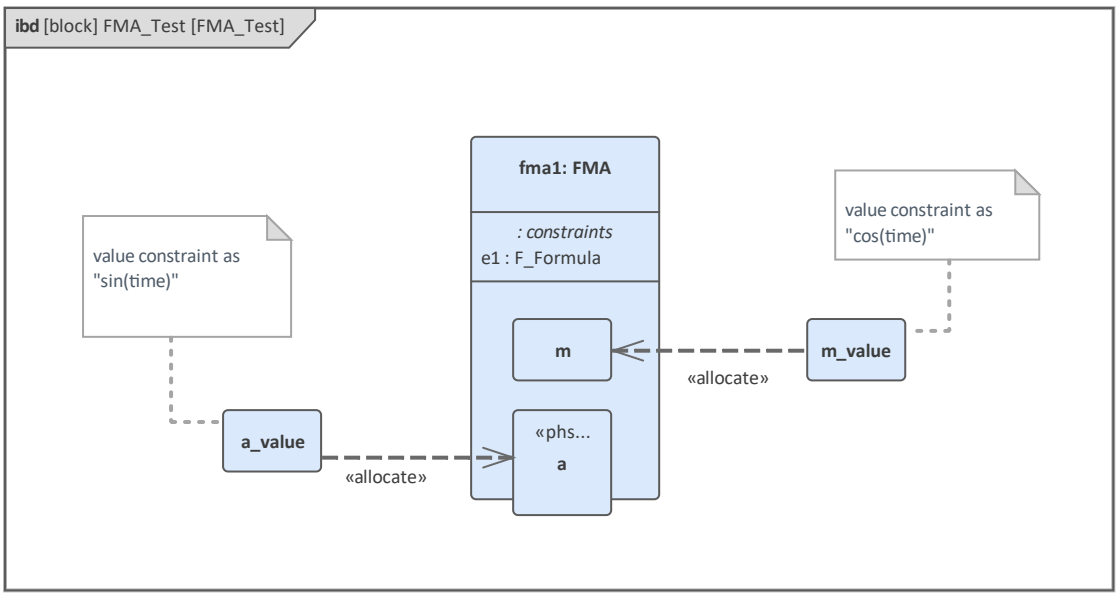
```
函数极限值
输入Real pMin；
输入真实 pMax；
输入实数 p;
输出实 pLim；
算法
pLim :=
如果 p > pMax 那么
最大
else如果 p < pMin 那么
pMin
else
p;
结束极限值；
```

价值分配

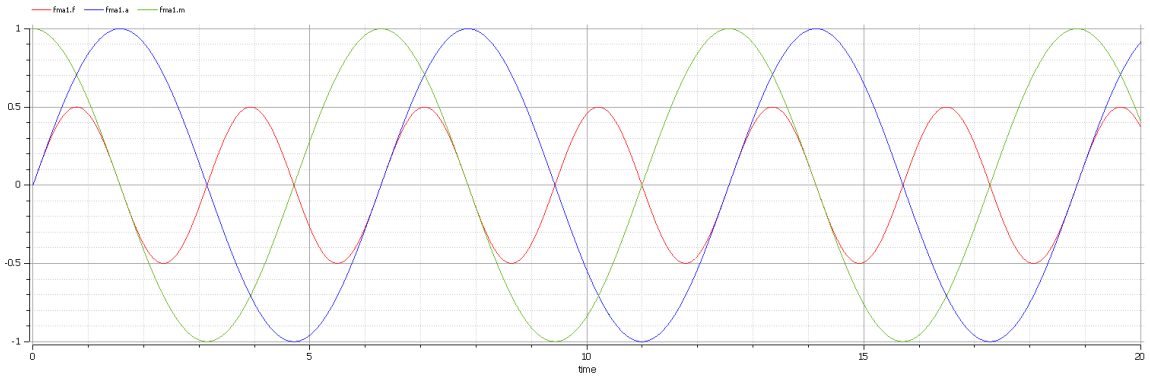
该图显示了一个名为 "力=质量*加速度"的简单模型。



- 块 f"MA" A属性 á"、 f"和 ḿ"以及约束属性 é1"建模，输入约束块 f"_Formula"
- 块'FMA'的属性没有设置任何初始值，并且属性'a'、'f'和'm'都是可变的，所以它们的值变化取决于它们被模拟的环境
- 创建一个块'FMA_Test'作为SysMLSimModel并添加属性'fma1'来测试块'FMA'的行为
- 约束'a_value' 为' sin (time)'
- 约束'm_value'为' cos (time)'
- 划分配连接器将值从环境分配到模型'FMA'



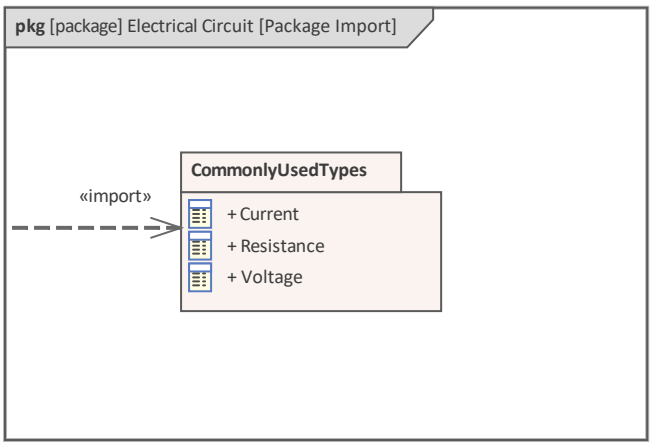
- 选中针对 f"ma1.a"、 f"ma1.m"和 f"ma1.f"的 要绘制的属性"复选框
- 单击求解按钮以模拟模型



包导入

工件选择一个包的元素（如Blocks、ConstrainBlocks和Value Types）如果模拟依赖于不属于这个包的元素，比如Reusable库， Enterprise Architect在包元素之间提供了一个导入连接器来满足这个需求。

在电气电路示例中，工件被配置为 "ElectricalCircuit"包，其中包含模拟所需的几乎所有元素但是，一些属性被键入为值类型，例如 "电压"、 "电流"和 "电阻"，这些值类型在多个 SysML 模型中常用，因此放置在单个 SysML 模型之外的一个名为 "CommonlyUsedTypes"的包中。如果使用导入连接器导入此包，则导入包中的所有元素都将出现在导入配置管理器中。



使用数据集进行模型分析

参数模型中使用的每个 SysML块都可以在仿真配置中定义多个数据集。这允许使用相同的 SysML模型进行可重复的模拟变化。

块A键入为 SysMLSimModel（不能被概括或构成组合的一部分的顶级节点）或 SysMLSimClass（可以被概括或构成组合的一部分的较低级别的元素）。在 SysMLSimModel元素上运行模拟时，如果您定义了多个数据集，则可以指定要使用的数据集。但是，如果模拟中的类有多个数据集，则您无法选择在模拟期间使用哪一个，因此必须将一个数据集标识为该类的默认值。

访问

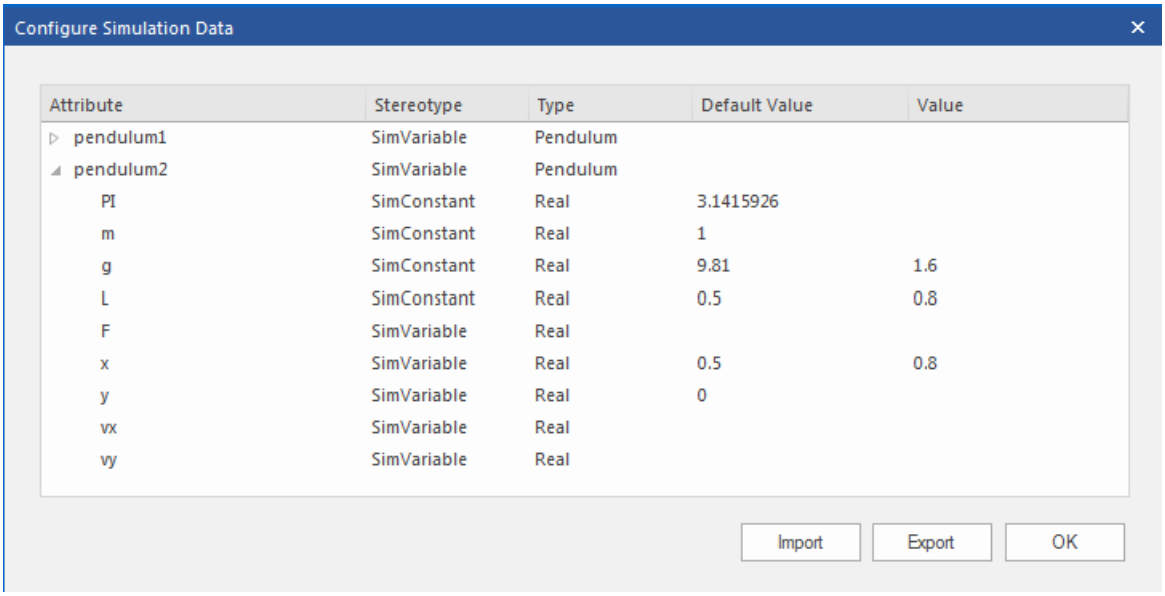
功能区	仿真>系统行为>Modelica/Simulink>配置管理器>在 块"组>名称列>块元素上下文菜单>创建仿真数据集

数据集管理

任务	行动
创造	要创建新数据集，请右键单击块名称并选择 创建仿真数据集"选项。数据集被添加到块名称下方的组件列表的末尾。单击 ... 按钮在 配置仿真数据"对话框中设置数据集（参见配置仿真数据表）。
复制	要复制现有数据集作为创建新数据集的基础，请右键单击数据集名称并选择 复制"选项。重复的数据集将添加到块名称下方的组件列表的末尾。单击 ... 按钮以编辑 配置仿真数据"对话框中的数据集（请参阅配置仿真数据表）。
删除	要删除不再需要的数据集，请右键单击数据集并选择 删除数据集"选项。
默认设置	要设置 SysMLSimClass 在用作属性类型或继承（以及当有多个数据集时）使用的默认数据集，请右键单击数据集并选择 设置为默认值"选项。默认数据集的名称以粗体突出显示。模型使用的属性将使用此默认配置，除非模型明确覆盖它们。

配置仿真数据

此对话框主要用于提供信息。您可以直接添加或更改数据的唯一列是 值"列。



柱子	描述
属性	属性”列提供了正在编辑的块中所有属性的树视图。
构造型	对于每个属性， 构造型”列标识它是否已配置为模拟期间的常量，或者其值预计会随时间变化的变量。
类型	类型”列描述了用 模拟此属性的类型。它可以是原始类型（例如 Real”）或对模型中包含的块的引用。属性引用块将显示由它们下面的引用块指定的子属性。
默认值	如果未提供覆盖， 默认值”列显示将在模拟中使用的值。这可以来自 SysML模型中的 初始值”字段或来自父类型的默认数据集。
价值	值”列允许您覆盖每个原始值的默认值。
导出/导入	单击这些按钮可使用电子表格等外部应用程序修改当前数据集中的值，然后将它们重新导入列表。

SysML仿真实例

本节为每个阶段提供了一个工作示例：为域创建 SysML模型，对其进行仿真，并评估仿真结果。这些示例应用了前面主题中讨论的信息。

例子

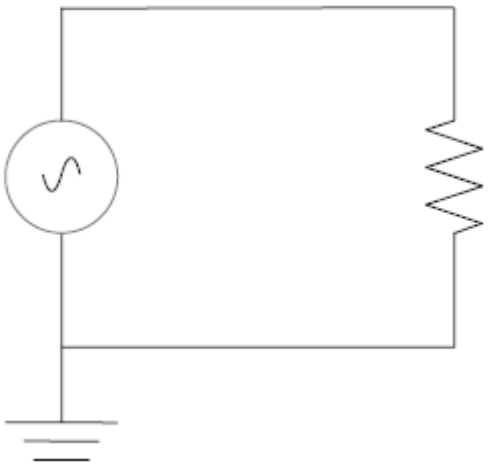
模型	描述
电路仿真示例	第一个例子是加载电路的模拟。该示例从电路图开始，并将其转换为参数模型。然后对模型进行仿真，评估电阻器源端和目标端的电压并将其与预期值进行比较。
质量弹簧阻尼振荡器仿真示例	第二个例子使用一个简单的物理模型来演示质量-弹簧-阻尼系统的振荡行为。
水箱压力调节器	最后一个示例显示了两个水箱的水位，其中水在它们之间分配。我们首先模拟一个平衡良好的系统，然后我们模拟一个水将从第二个水箱溢出的系统。

电路仿真示例

在本例中，我们为一个简单的电路创建 SysML 参数模型，然后使用参数仿真来预测和绘制该电路的行为。

图表

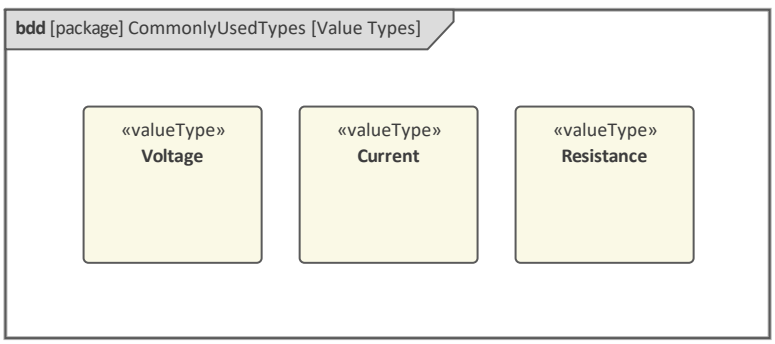
我们将要学习的电路模型（此处显示）使用标准电路符号。

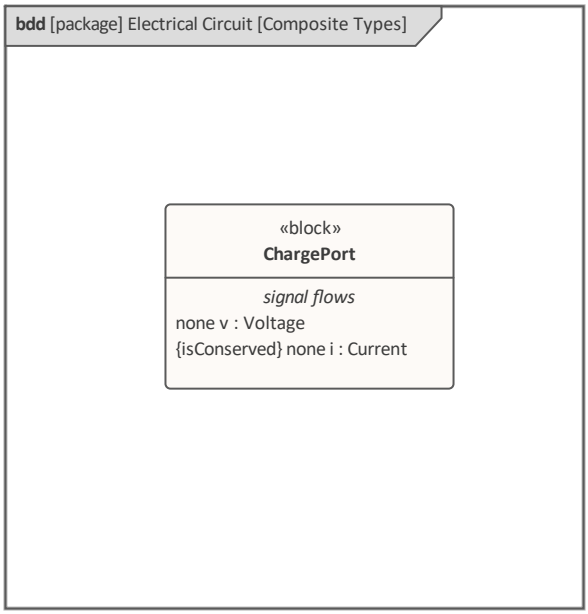


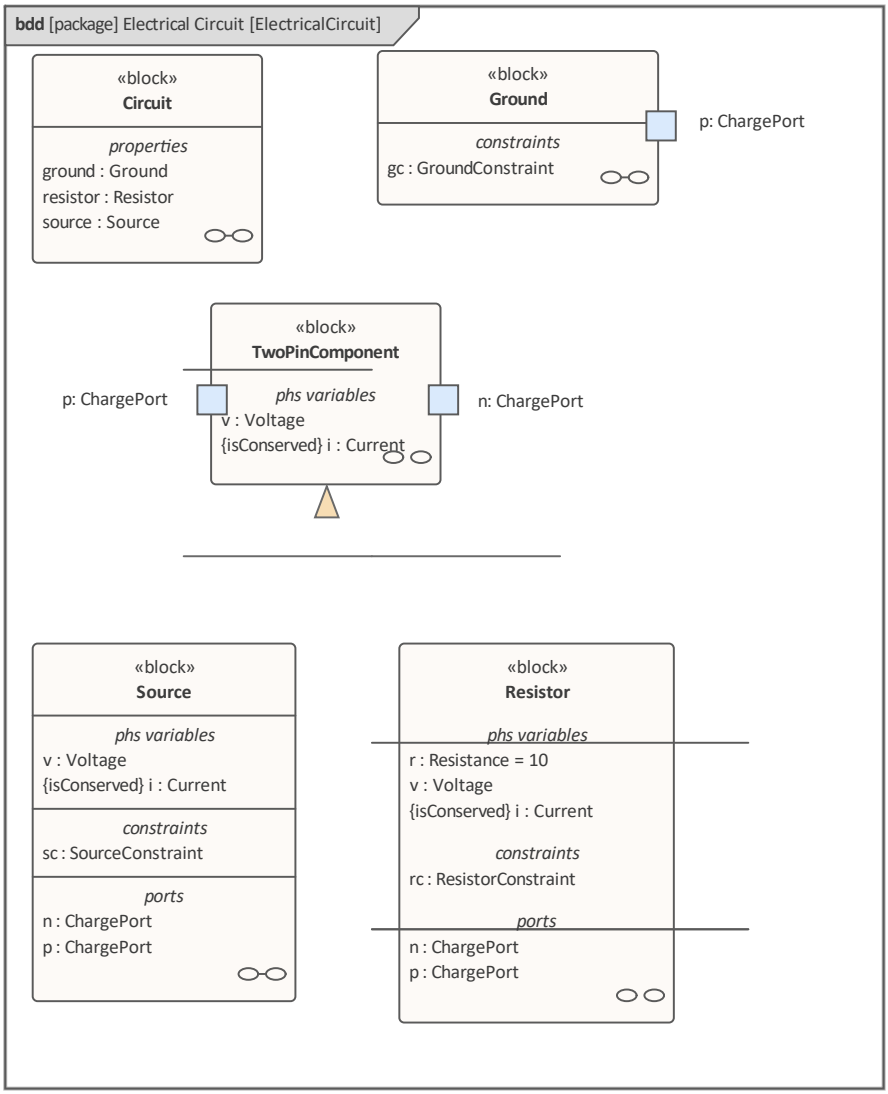
该电路包括交流电源、源和电阻器，通过电线相互连接。

创建 SysML模型

这张表显示了我们如何构建一个完整的完成模型来表示电路，从最低级别的类型开始，一步一步地构建模型。

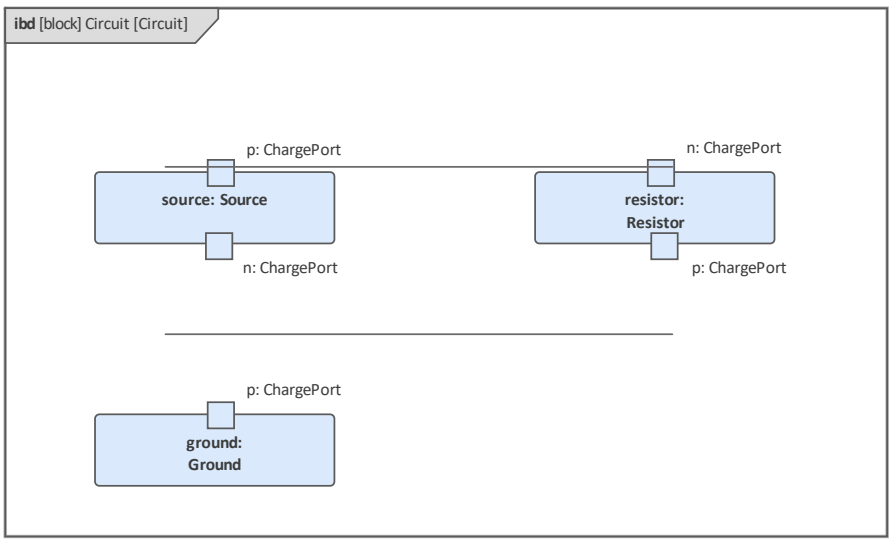
部件	行动
类型	定义电压、电流和电阻的值类型。单位和数量种类对于模拟而言并不重要，但如果定义一个完整的完成模型则需要设置。这些类型将从原始类型“Real”泛化。在其他模型中，您可以选择将值类型映射到与模型分开的相应仿真类型。  此外，定义一个名为 ChargePort 的复合类型 (块)，其中包括电流和电压属性。这种类型允许我们表示组件之间连接器处的电能。

	
块	在 SysML 中，电路和每个组件都将表示为块。 在块定义图 (图表) 中，创建一个电路块。该电路由三部分组成：源、地和电阻。这些部分属于不同的类型，具有不同的行为。 为每个零件类型创建一个块。电路块的三个部分通过端口连接，代表电气引脚。源和电阻有一个正极和一个负极引脚。地只有一个引脚，它是正极。电流（电荷）通过引脚传输。创建一个带有两个端口（引脚）的抽象块“TwoPinComponent”。这两个端口分别命名为 p”（正）和 n”（负），它们的类型为 ChargePort. 此图显示了 BDD，包括电路块、接地、TwoPinComponent、源和电阻器。

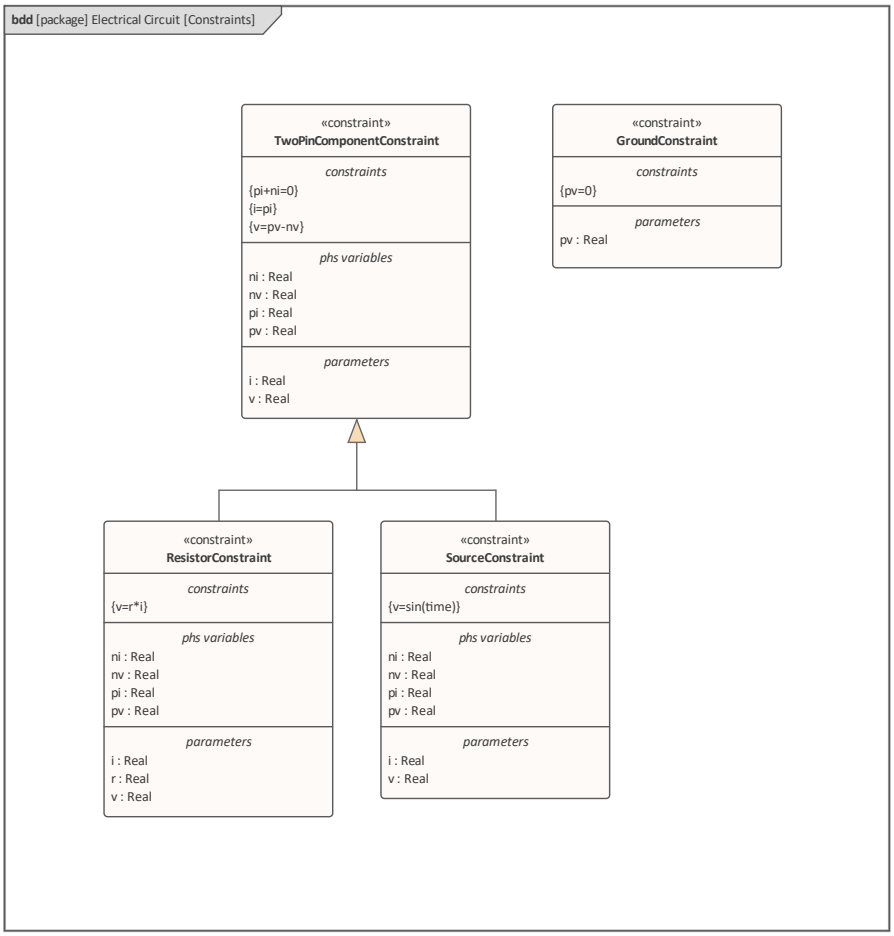


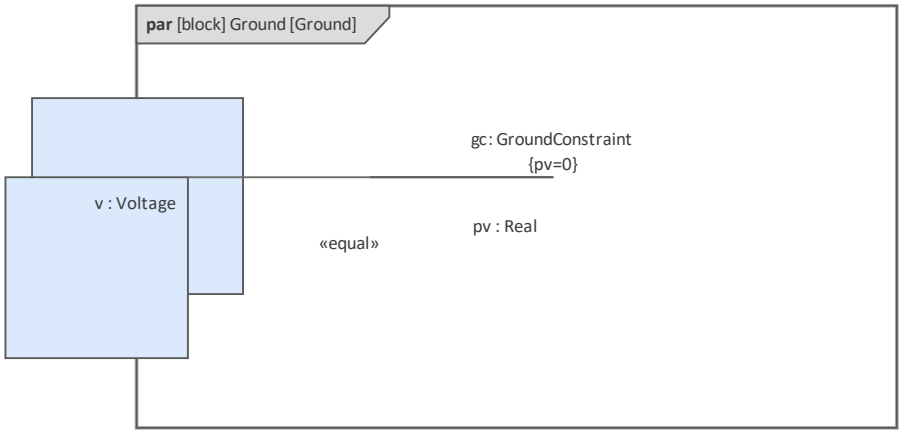
内部结构

为电路创建一个内部块图（图表）。为源、电阻和接地添加属性，由相应的块键入。用连接器连接端口。源的正极引脚连接到电阻器的负极引脚。Resistor 的正极引脚连接到源的负极引脚。地也连接到源的负极引脚。



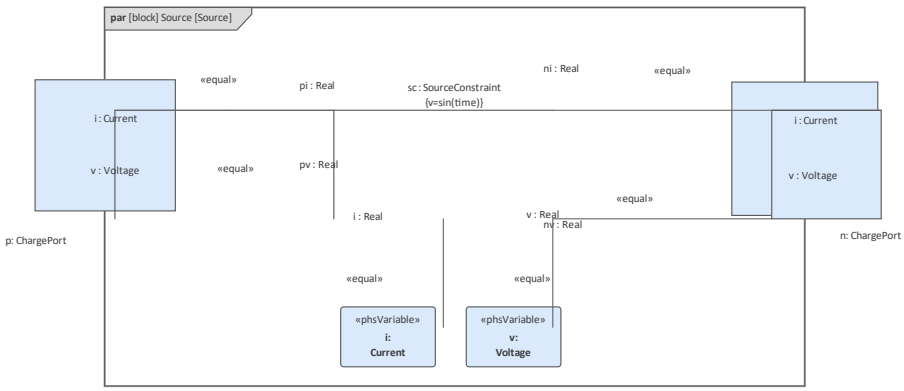
请注意，这遵循与原始电路图相同的结构，但每个组件的符号已替换为由我

	们定义的块键入的属性。
约束	方程定义数值属性之间的数学关系。在 SysML 中，方程表示为 ConstraintBlocks 中的约束。 ConstraintBlocks 的参数对应于块的 PhSVariables 和 PhSConstants（本例中为'i'、'v'、'r'），以及端口类型中存在的端口（'pv'、'pi'、'nv '，在本例中为 'ni'）。 创建一个 ConstraintBlock 'TwoPinComponentConstraint' 来定义源和电阻器共有的参数和方程。该等式应状态组件的电压等于正负引脚电压之差。组件的电流等于通过正极引脚的电流。通过两个引脚的电流之和必须加起来为零（一个是另一个的负数）。接地约束表明接地引脚上的电压为零。源约束将电压定义为sine ，以当前仿真时间为参数。此图显示了这些约束如何在 BDD 中呈现。 
绑定	约束参数的值等同于具有绑定连接器的可变值和常数值。在每个块上创建约束属性（属性类型为 ConstraintBlocks），并将块变量和常量绑定到参数，以将约束约束块。这些图分别显示了接地、源和电阻的绑定。 对于地面约束，将 gc.pv 绑定到 pv



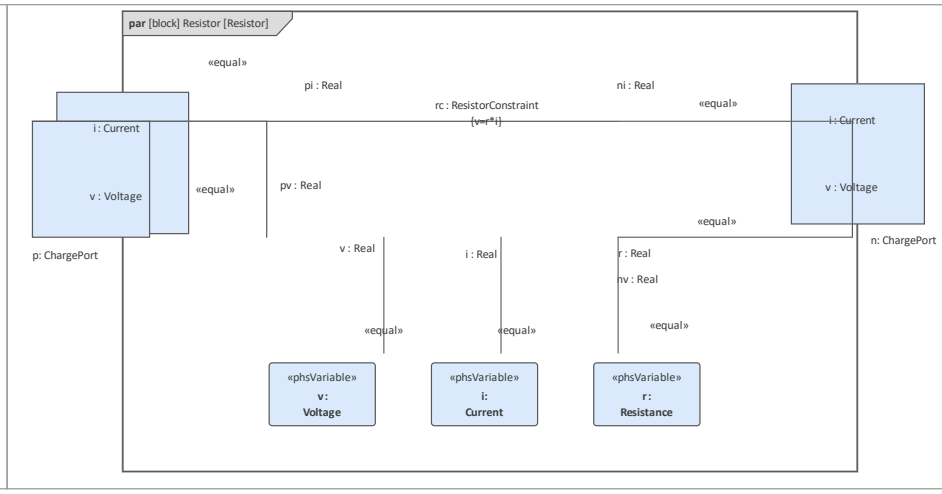
对于源约束，绑定：

- sc.pi 到 pi
- sc.pv 到 pv
- sc.v 到 v
- sc.i到我
- sc.ni 到 ni 和
- sc.nv 到 nv



对于电阻器约束，绑定：

- rc.pi 到 pi
- rc.pv 到 pv
- rc.v 到 v
- rc.i 到我
- rc.ni 到 ni
- rc.nv 到 nv 和
- rc.r 到 r



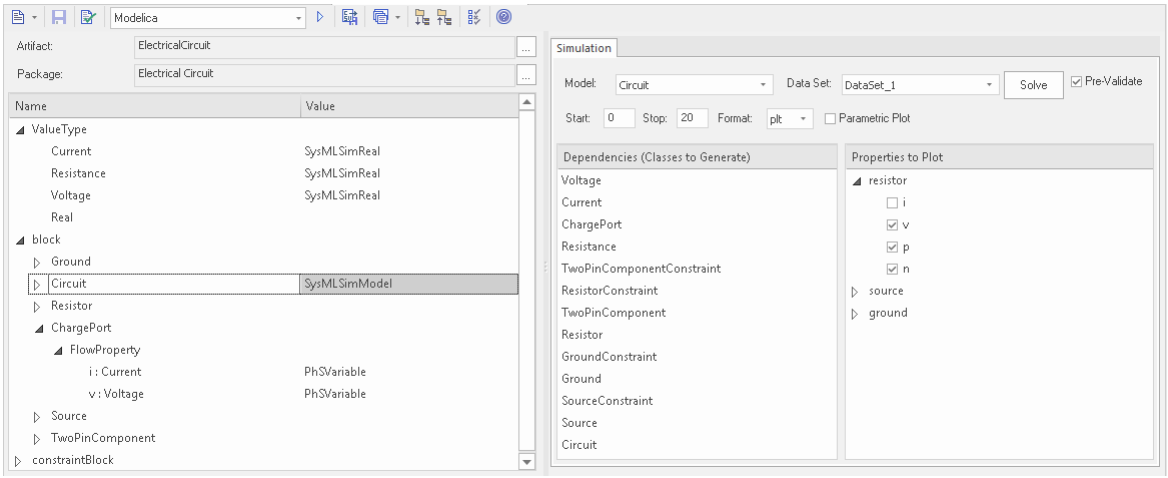
配置仿真行为

此表显示了 SysMLSim 配置的详细步骤。

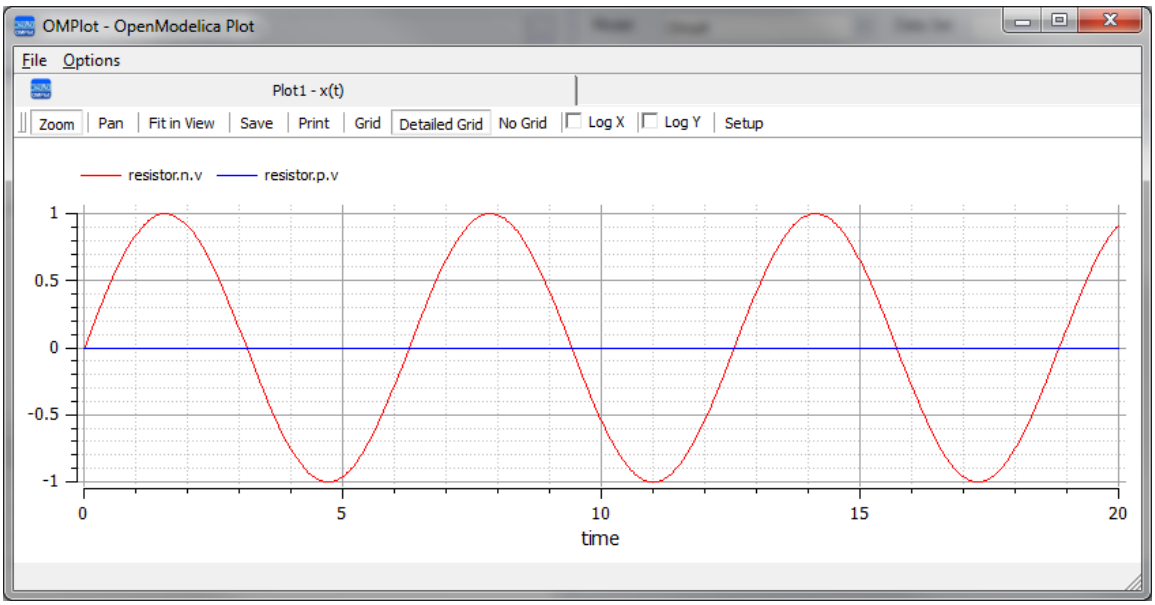
节	行动
工件适合	• 选择 仿真>系统行为> Modelica/Simulink >配置管理器” • 从第一个下拉菜单中，选择工具栏创建工具工件元素创建工件 • 选择拥有此 SysML模型的包
在配置管理器中创建根元素	• 值类型 • 块 • 约束块
值类型替换	展开 ValueType 并为 Current、Resistance 和 Voltage 中的每一个从 “Value”组合框中选择 “SysMLSimReal”。
将属性设置为流	• 将 块”扩展到 ChargePort |流属性 | i ：当前并从 值”组合框中选择 “SimVariable” • 对于 无素”，单击 … 按钮以打开 “ElementConfigurations”对话框 • 将 “isConserved”设置为 “True ”
SysMLSimModel	这就是我们要模拟的模型：将块'Circuit'设置为'SysMLSimModel'。

运行仿真

在 仿真”页面中，选中 “resistor.n”和 “resistor.p”复选框进行绘图，然后单击 求解”按钮。



如图所示，绘制了两个图例 "resistor.n"和 "resistor.p"。

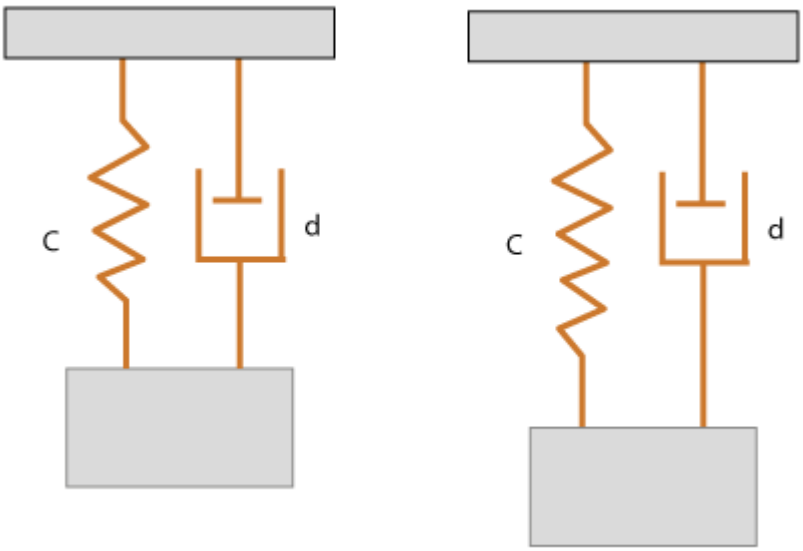


质量弹簧阻尼振荡器仿真示例

在本节中，我们将为由质量、弹簧和阻尼器组成的简单振荡器创建 SysML 参数模型，然后使用参数模拟来预测和绘制该机械系统的行为。最后，我们通过比较通过数据集提供不同参数值的两个振荡器来执行假设分析。

正在建模的系统

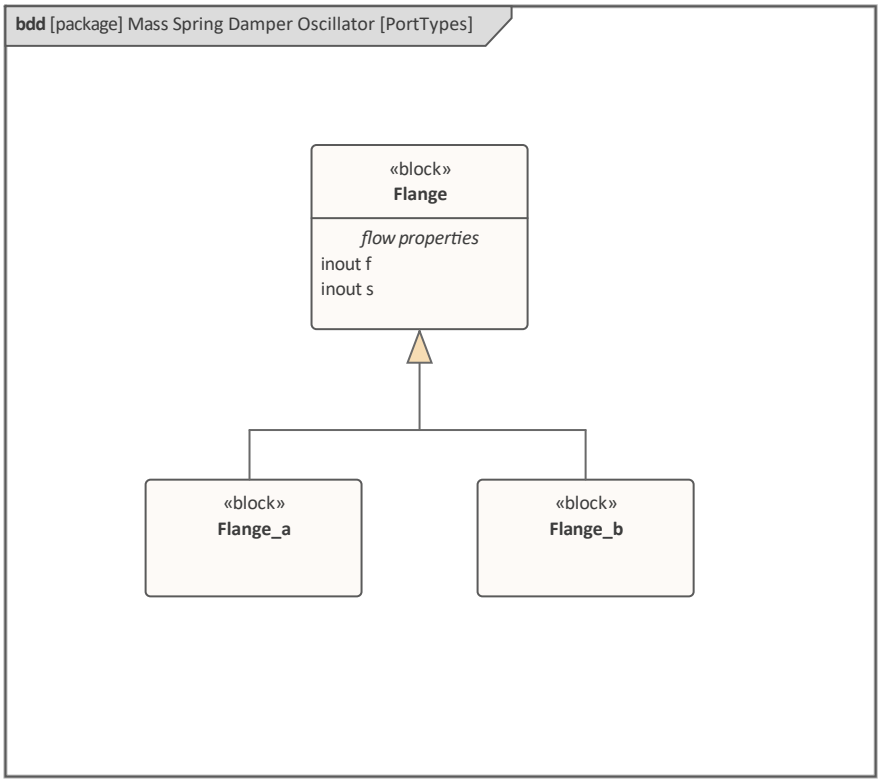
质量悬挂A弹簧和阻尼器上。此处显示的第一个状态表示时间=0 时的初始点，就在质量释放时。第二个状态代表当身体静止并且弹簧力与重力平衡时的最终点。

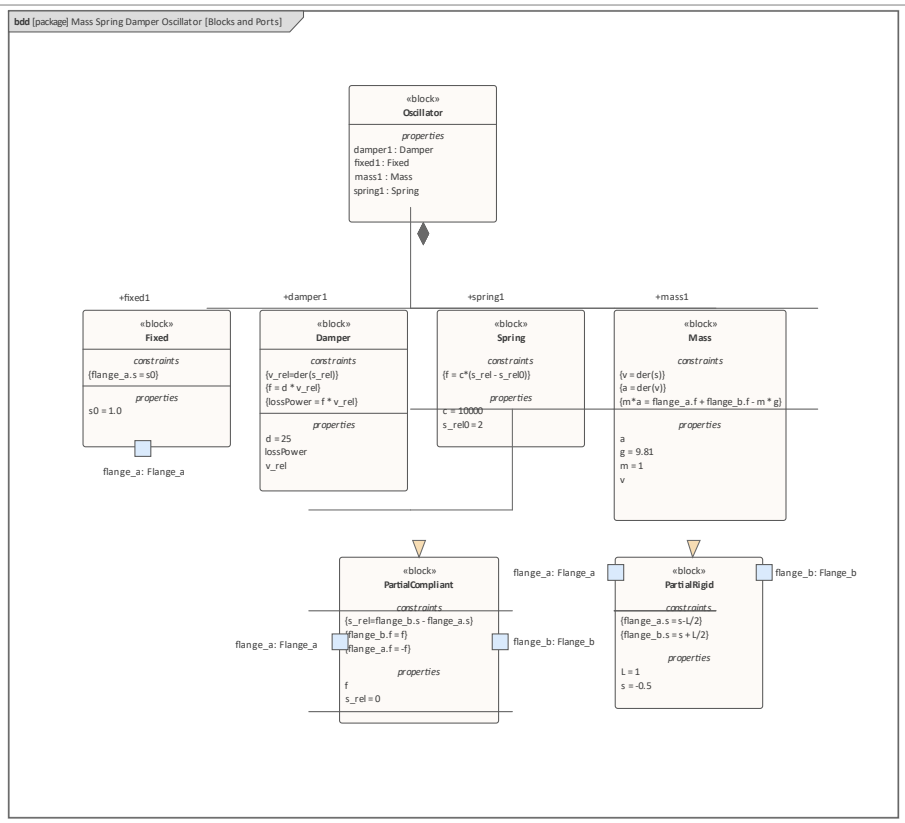


创建 SysML模型

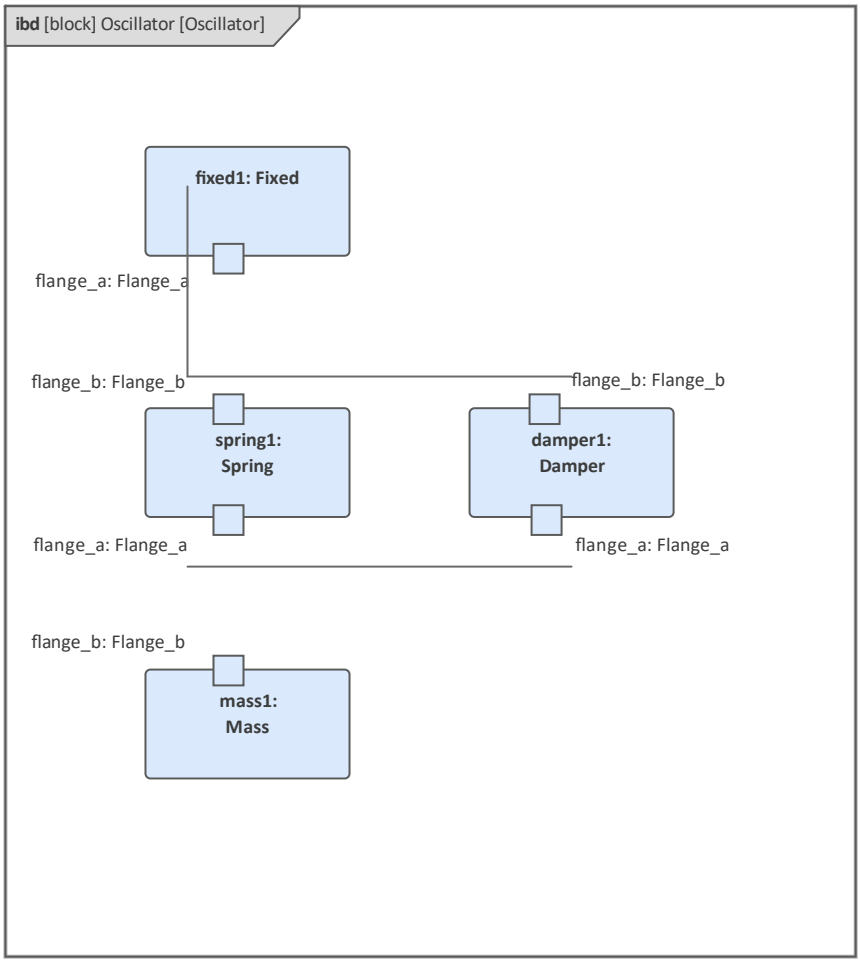
SysML 中的块模型有一个主要模块，即振荡器。振荡器有四个部分：固定顶板、弹簧、阻尼器和质量体。为这些部分中的每一个创建一个块。 Oscillator块的四个部分通过端口连接，代表机械法兰。

成分	描述
端口类型	用于 1D 过渡机械域中法兰的块 'Flange_a' 和 'Flange_b' 相同，但作用略有不同，有点类似于电气域中 PositivePin 和 NegativePin 的作用。力通过法兰传递。所以 flow 属性*Flange.f*的属性*isConserved*应该设置为True 。

	
块和端口	• 创建块 弹簧”、 阻尼器”、 质量”和 固定”以分别表示弹簧、阻尼器、质量体和天花板 • 创建一个块'PartialCompliant'，有两个端口（法兰），命名为'flange_a'和'flange_b'——它们分别是Flange_a和Flange_b类型； 'Spring' 和 'Damper' 块从 'PartialCompliant' 概括 • 创建一个具有两个端口（法兰）的块 PartialRigid”，命名为 flange_a”和“flange_b”——它们分别是 Flange_a 和 Flange_b 类型； 'Mass'块从'PartialRigid'概括 • 为天花板创建一个只有一个法兰的块'Fixed'，它的端口只有'flange_a'类型为 Flange_a



内部结构	为 "振荡器"创建内部块图 (IBD)。为固定天花板、弹簧、阻尼器和质量体添加属性，由相应的 Blocks 键入。用连接器连接端口。 • 将 "fixed1"的 "flange_a"连接到 "spring1"的 "flange_b" • 将 "damper1"的 "flange_b"连接到 "spring1"的 "flange_b" • 将 "damper1"的 "flange_a"连接到 "spring1"的 "flange_a" • 将 "spring1"的 "flange_a"连接到 "mass1"的 "flange_b"

	
约束	为简单起见，我们直接在块元素中定义约束；您可以选择定义约束块，在块中使用属性，并将它们的参数绑定到块的属性。

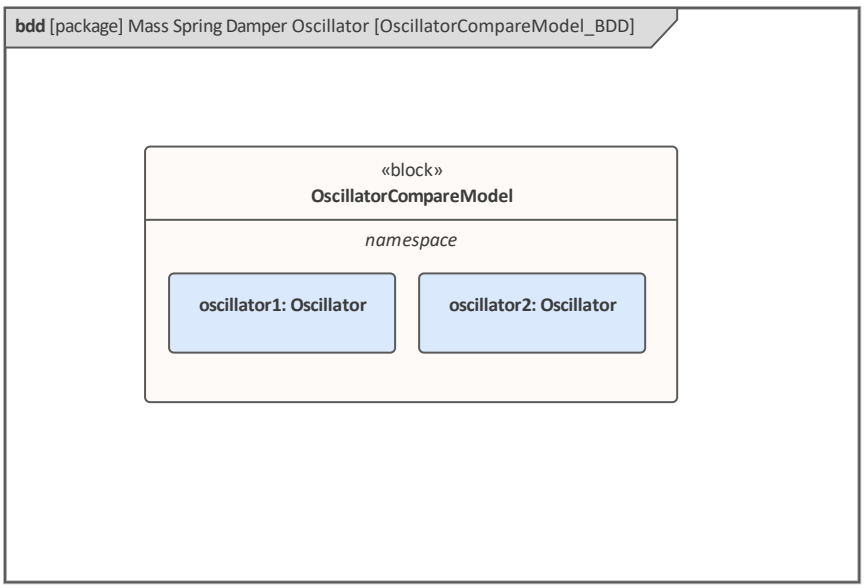
两个振荡器比较计划

在我们对模型进行建模后，我们想做一些假设分析。例如：

- 具有不同阻尼器的两个振荡器有什么区别？
- 如果没有阻尼器怎么办？
- 两个不同弹簧的振荡器有什么区别？
- 两个不同质量的振荡器有什么区别？

以下是创建比较模型的步骤：

- 创建一个名为 "OscillatorCompareModel"的块
- 为 "OscillatorCompareModel"创建两个属性，称为振荡器1 和振荡器*2*，并使用块振荡器键入它们



设置数据集并运行仿真

创建一个配置工件其分配给这个包。然后创建这些数据集：

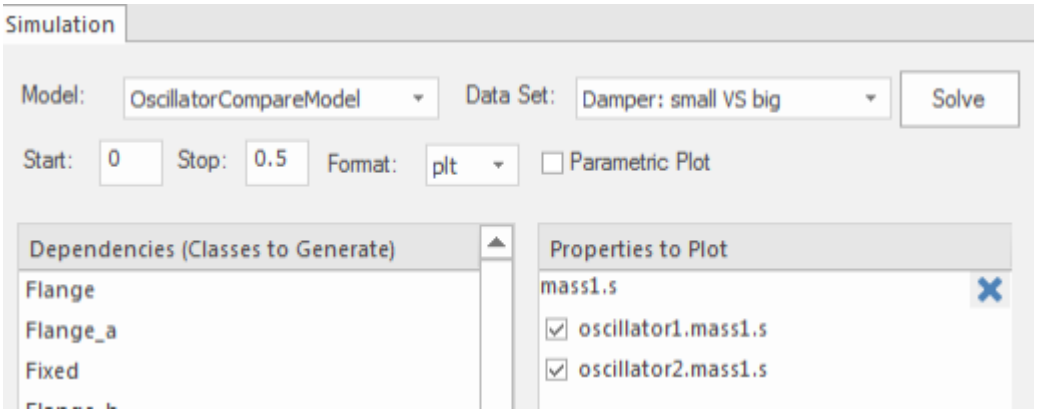
- 阻尼器：小与大
  为 'oscillator1.damper1.d' 提供值 10，为 'oscillator2.damper1.d' 提供较大的值 20
- 阻尼器：否 vs 是
  为 'oscillator1.damper1.d 提供值 0;（'oscillator2.damper1.d' 将使用默认值 25）
- 弹簧：小与大
  提供 'oscillator1.spring1.c' 的值 6000 和 'oscillator2.spring1.c' 的较大值 12000
- 质量：轻与重
  为 "oscillator1.mass1.m"提供值 0.5，为 "oscillator2.mass1.m"提供较大的值 2

配置的页面如下所示：

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

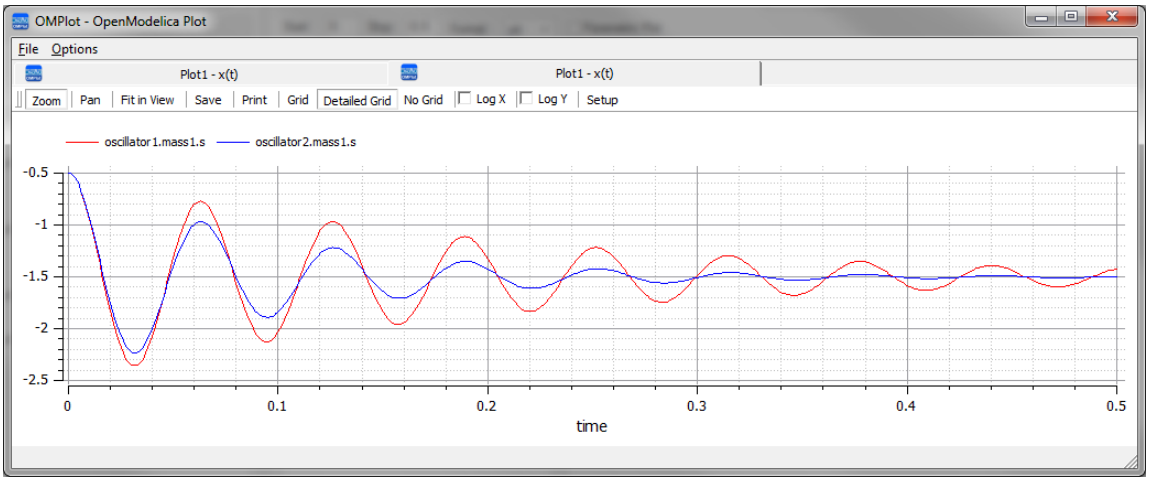
在 "仿真"页面上，选择 "OscillatorCompareModel"，绘制 "oscillator1.mass1.s"和 "oscillator2.mass1.s"，然后选择一个创建的数据集并运行仿真。

提示：如果绘图列表中的属性过多，您可以使用列表标题上的时间菜单切换过滤器栏，然后在本例中键入 “上下文”。

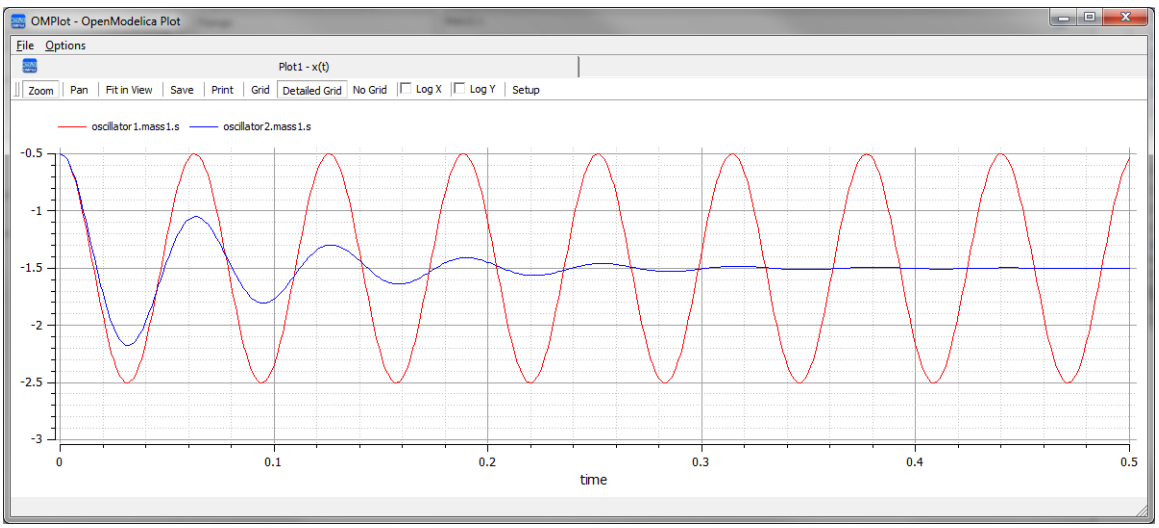


这些是模拟结果：

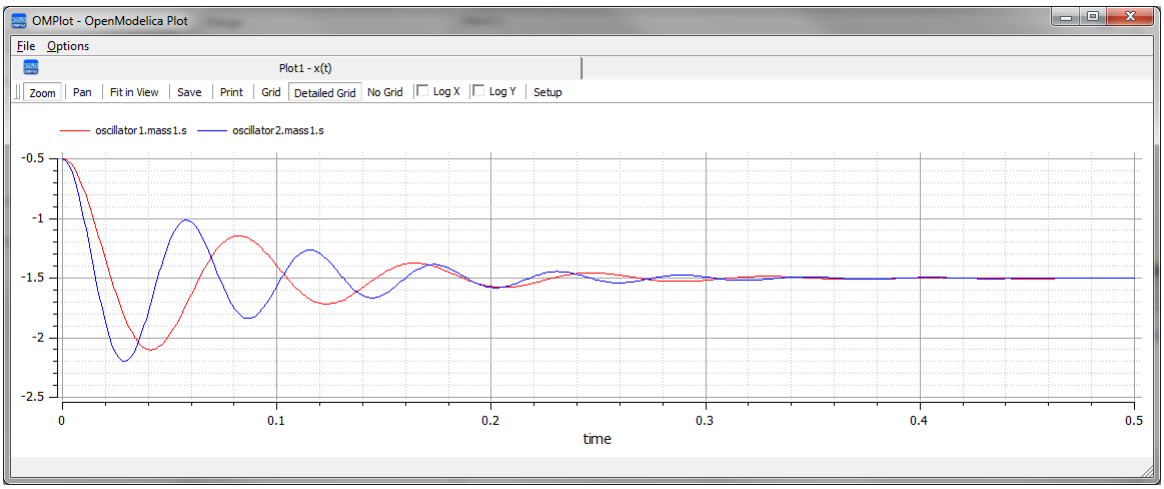
- 阻尼器，小与大：阻尼器越小，车身振动越大



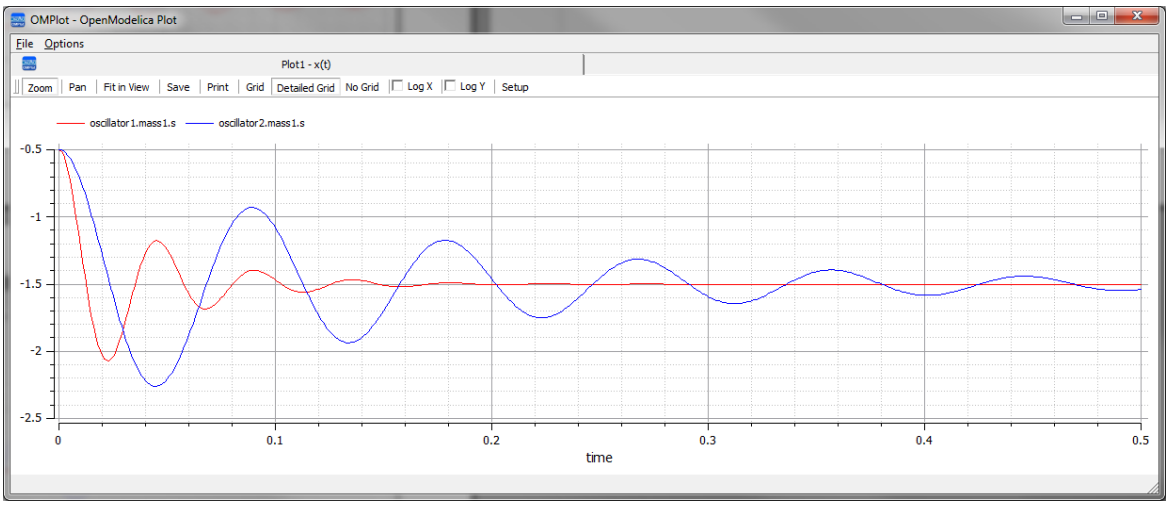
- 阻尼器，否 vs 是：振荡器在没有阻尼器的情况下永远不会停止



- 弹簧，小与大：“c”较　的弹簧会摆动得更慢



- 质量，轻与重：质量较小的object会更快地振动并更快地调节



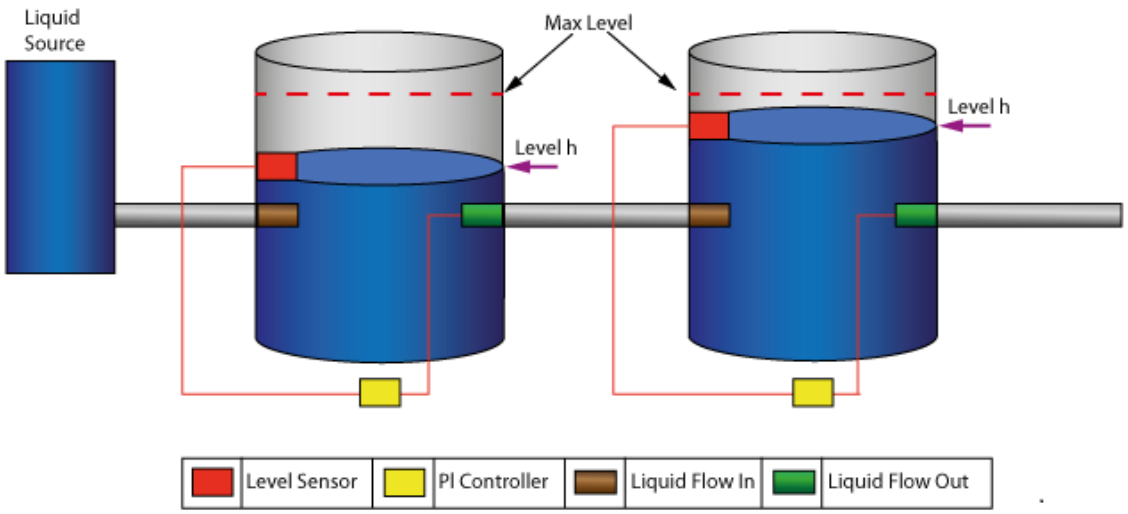
水箱压力调节器

在本节中，我们将介绍为水箱压力调节器创建 SysML 参数模型，该模型由两个连接的水箱、一个源和两个控制器组成，每个控制器监控水位并控制阀门以调节系统.

我们将解释 SysML模型，创建它并设置 SysMLSim 配置。然后我们将使用运行仿真。

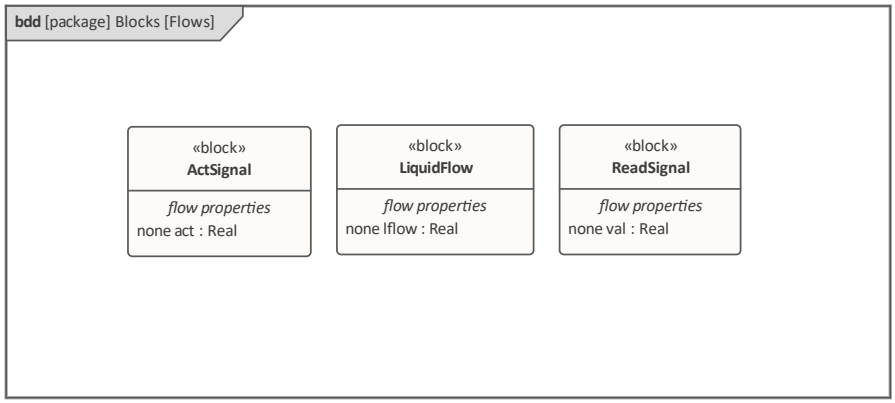
正在建模的系统

此图描绘了连接在一起的两个源，以及填充第一个水箱的水源。每个水箱都有一个与其相连的比例积分 (PI) 连续控制器，它将水箱中的水位调节到参考水位。当源向第一个水箱注水时，PI 连续控制器根据水箱的实际水位调节水箱的流出量。第一个水箱中的水流入第二个水箱，PI 连续控制器也试图对其进行调节。这是一个自然的、非特定领域的物理问题。



创建 SysML模型

部件	讨论
端口类型	坦克有四个端口，它们分别输入到这三个块中： • ReadSignal：读取液位；这有一个单位为 ḿ”的属性 v̋al” • ActSignal：用于设置阀门位置的执行器信号 • LiquidFlow：入口或出口处的液体流量；这有一个属性 l̋flow”，单位为 “m ³ /s”



块定义图表

LiquidSource：进入水箱的水一定来自某个地方，因此我们在水箱系统中有一个液体源组件，属性*flowLevel*的单位为 m̋³/s"。 A 端口"键入为 端口"。

水箱：水箱通过端口连接到控制器和液体源。

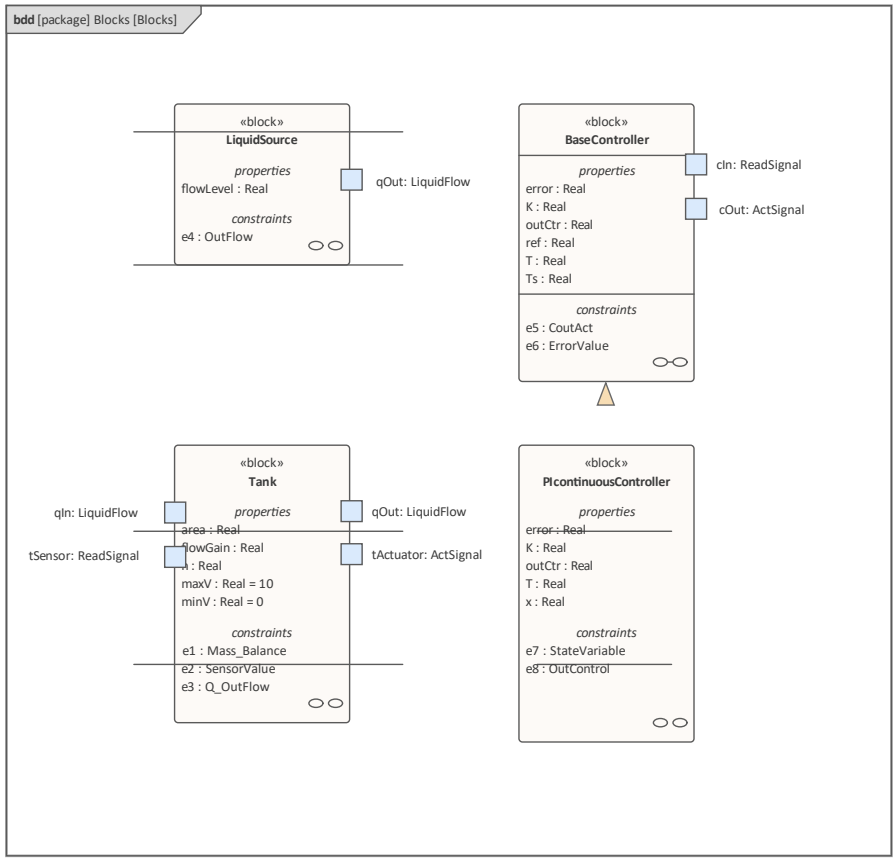
- 每个 Tank 有四个端口：
  - qIn：用于输入流
  - qOut：用于输出流
  - tSensor：用于提供液位测量
  - tActuator：用于设置阀门在出口处的位置
  坦克
- 属性：
  - 体积（单位='m³'）：罐的容量，参与质量平衡
  方程
  - h（单位='m'）：水位，参与质量平衡
  方程;它的值由传感器读取
  - flowGain（单位='m³/s'）：输出流量与阀门有关
  按流量增益定位
  - minV, maxV：输出阀流量的限制

BaseController：这个块可以是 PI Continuous控制器和 PI Discrete控制器的父级或祖先。

- 端口：
  - cIn：输入传感器电平
  -控件：控制执行器
- 属性：
  - Ts（单位='s'）：离散样本之间的时间周期（未使用
  在这个例子中）
  - K ：增益因子
  - T (unit = 's'): 控制器的时间常数
  - 参考：参考水平
  - 误差：参考水平与实际水平之间的差异
  水位，从传感器获得
  - outCtr：到执行器的控制信号，用于控制阀门
  位置

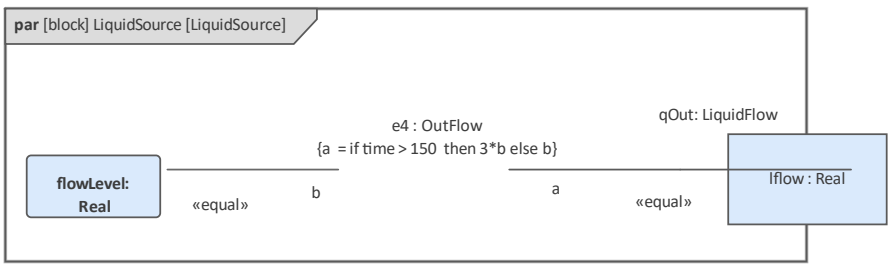
PIcontinuousController：从 BaseController 专门化

- 属性：
  -x：控制器状态变量



约束块

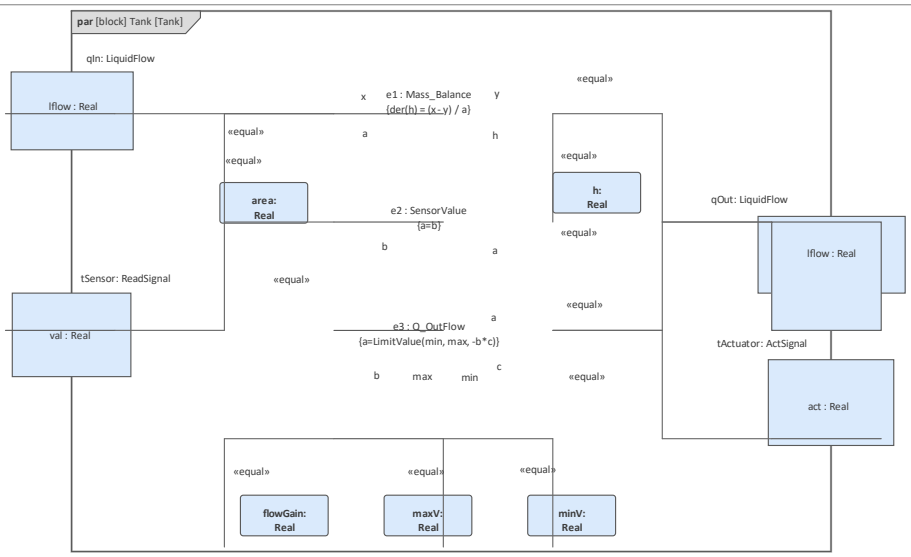
流量在时间=150 时急剧增加至先前流量水平的三倍，这产生了一个有趣的控制问题，罐的控制器必须处理。



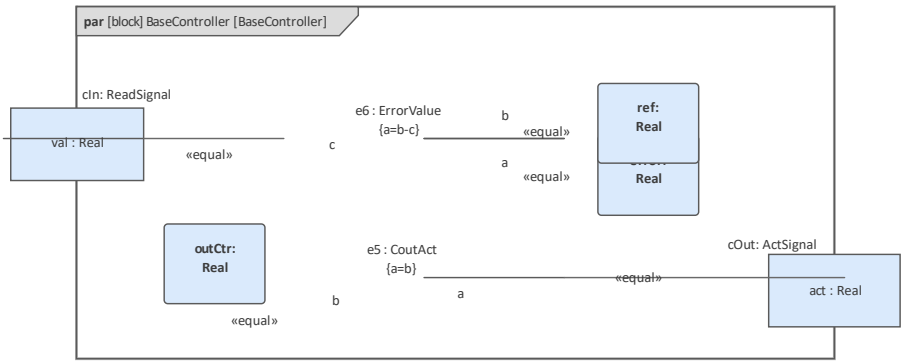
调节罐性能的中心方程是质量平衡方程。

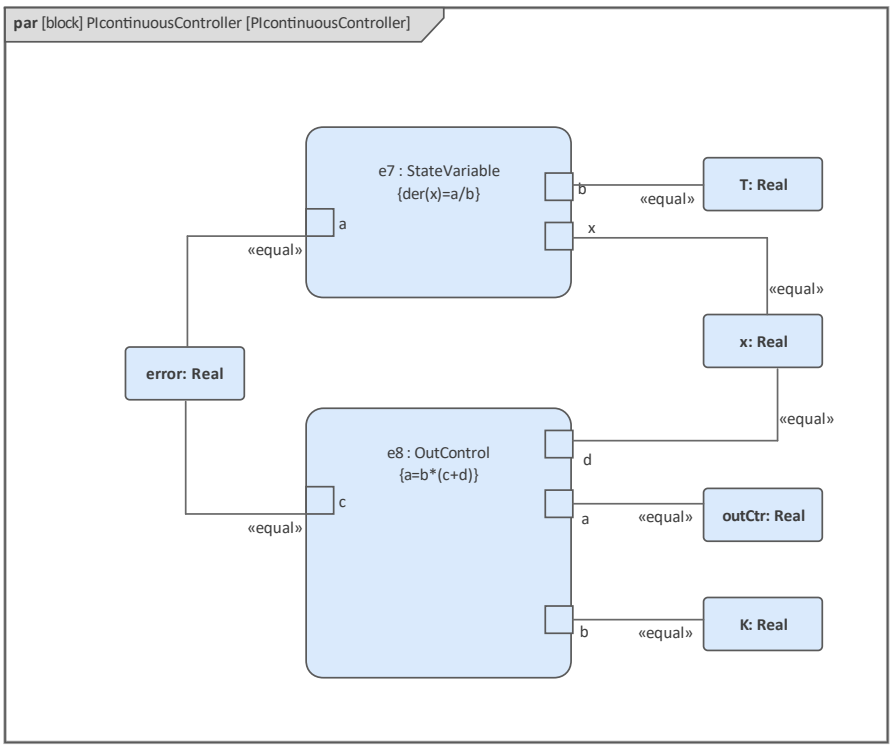
输出流量通过 "flowGain"参数与阀门位置相关。

传感器只是读取水箱的液位。

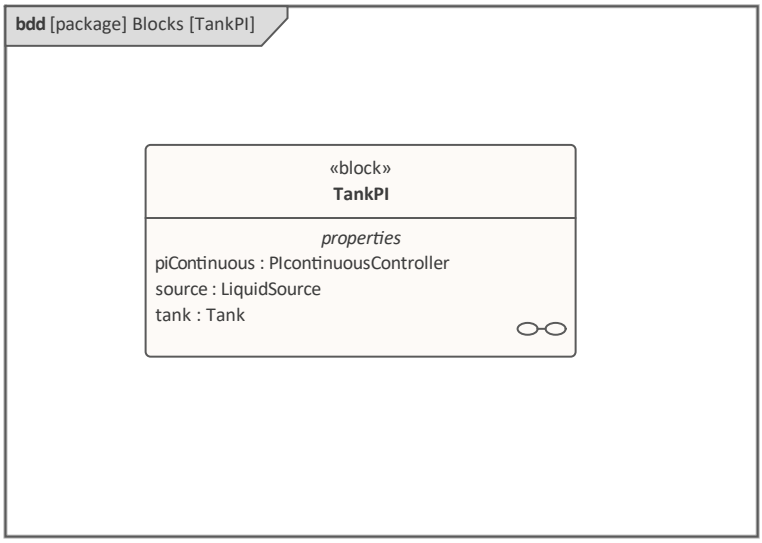


这些图中说明了为 "BaseController"和 "PIcontinuousController"定义的约束。

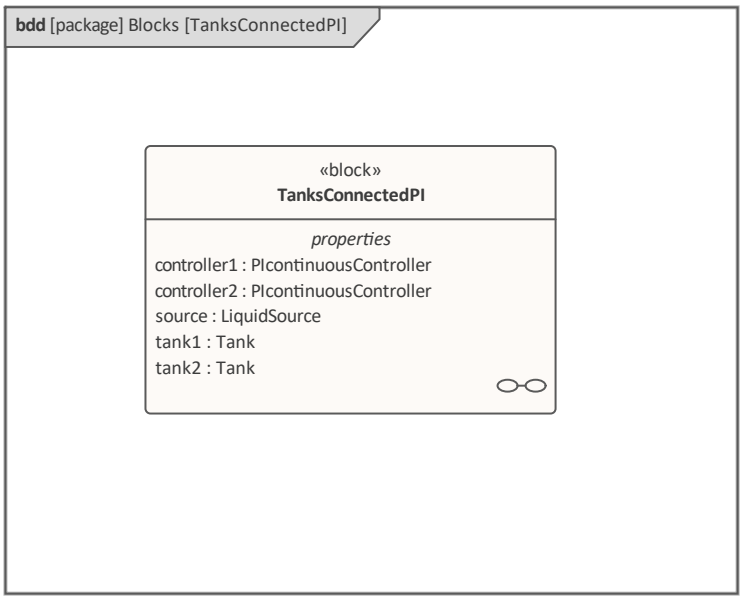




内部块图表	这是具有单个水箱的系统的内部块图。



这是具有两个连接罐的系统的内部块图。



运行仿真

由于*TankPI*和*TanksConnectedPI*被定义为 "SysMLSimModel"，因此它们将列在 "仿真"页面的 "模型"组合框中。

选择*TanksConnectedPI* ，并观察这些 GUI 发生的变化：

- "数据集"组合框：将填充*TanksConnectedPI*中定义的所有数据集
- 'Dependencies' 列表：将自动收集 TanksConnectedPI 直接或间接引用的所有 Blocks、约束、 *SimFunctions*和 ValueTypes（这些元素将生成为 OpenModelica 代码）
- '属性Plot'：将收集一长串'叶子'变量属性（即它们没有属性）；您可以选择一个或几个进行模拟，属性将显示在图例中

工件和配置

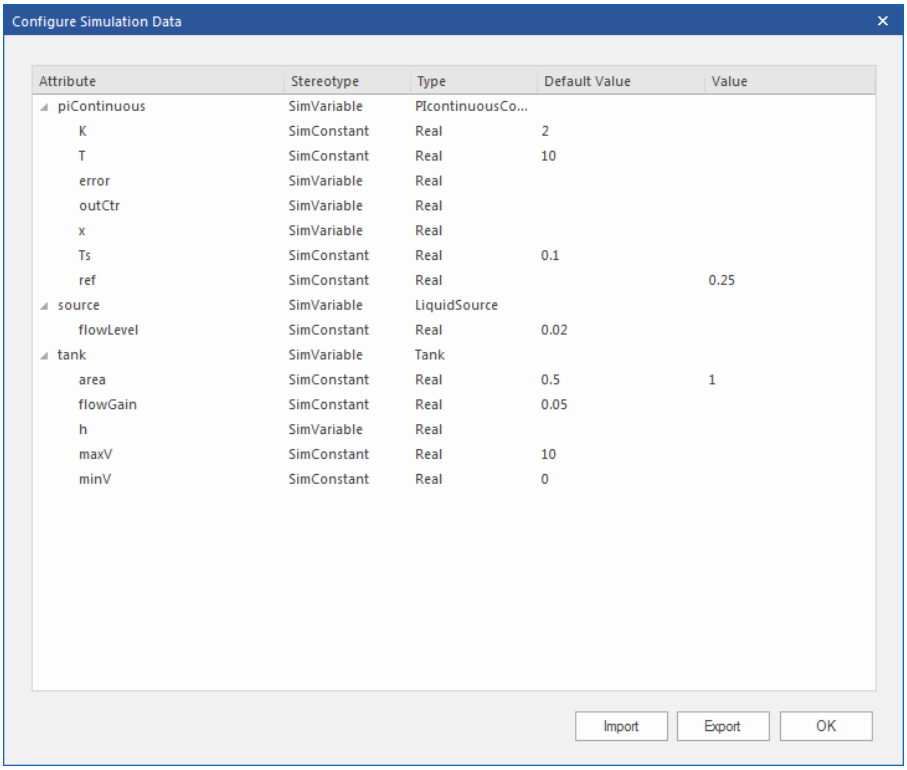
选择 仿真>系统行为> Modelica/Simulink >配置管理器”
包中的元素将被加载到配置管理器中。
配置这些块及其属性，如本表所示。
注记：未配置为 属性”的属性默认为 SimVariable”。

块	属性
液体源	配置为 配置”。 属性配置： • flowLevel：设置为 SimConstant”
坦克	配置为 配置”。 属性配置： • 区域：设置为 SimConstant” • flowGain：设置为 SimConstant” • maxV：设置为 SimConstant” • minV：设置为 SimConstant”
基本控制器	配置为 配置”。 属性配置： • K ：设置为 SimConstant” • T ：设置为 SimConstant” • Ts：设置为 SimConstant” • 参考：设置为 SimConstant”
PI连续控制器	配置为 配置”。
坦克PI	配置为 配置”。
TanksConnectedPI	配置为 配置”。

设置数据集

右键单击每个元素，选择 创建仿真数据集”选项，然后配置数据集，如本表所示。

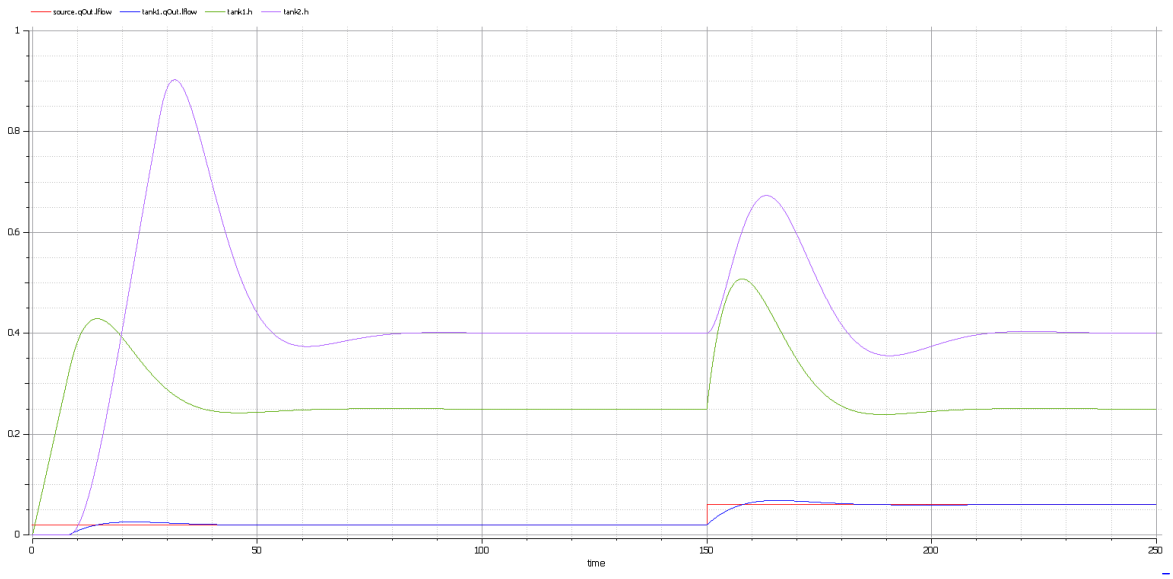
元素	数据集
液体源	流量等级：0.02
坦克	h.开始：0 流量增益：0.05 面积：0.5 最大V：10 最小伏特：0
基本控制器	T ：10

	K : 2 TS: 0. 1
PI连续控制器	无需配置。 默认情况下，特定块将使用超级块的默认数据集中配置的值。
坦克PI	这里有趣的是可以在 配置仿真数据”对话框中加载默认值。例如，我们在每个块元素上配置为默认数据集的值被加载为属性的默认值。单击每行上的图标可将属性的内部结构扩展到任意深度。  点击确定按钮，返回配置管理器。然后配置这些值： • tank.area: 1这会覆盖 Tank块数据集中定义的默认值 0.5 • piContinuous.ref：0.25
TanksConnectedPI	• 控制器1.ref：0.25 • 控制器2.ref：0.4

仿真分析1

选择这些变量并单击求解按钮。这个情节应该提示：

- 源.qOut.lflow
- tank1.qOut.lflow
- 坦克1.h
- 坦克2.h



以下是对结果的分析：

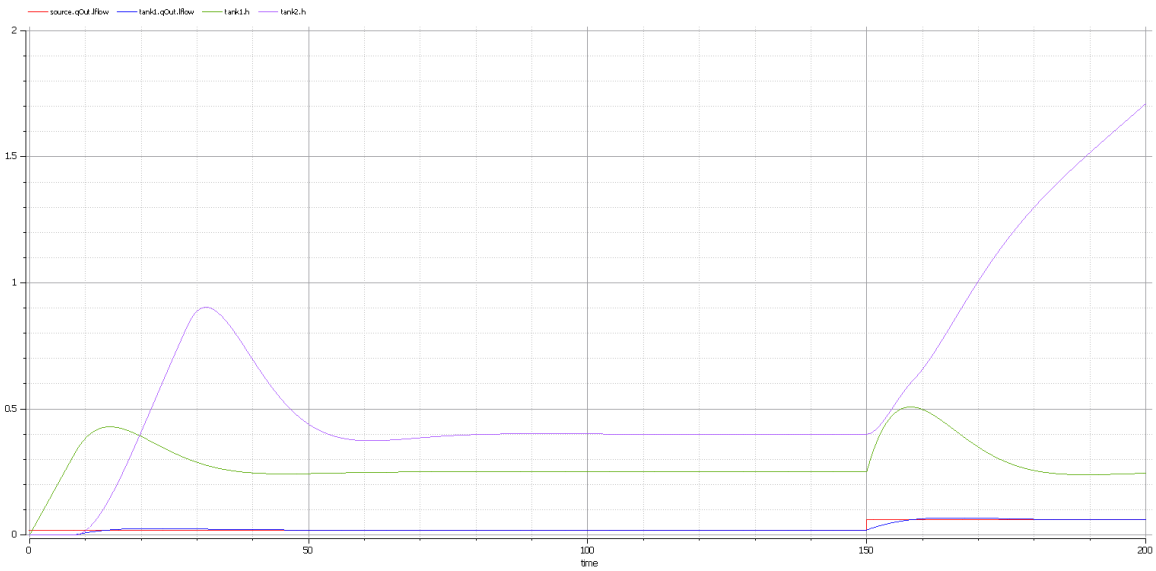
- 液体流量在时间=150 时急剧增加，达到 0.06 m^3/s，是之前流量 (0.02 m^3/s) 的三倍
- Tank1 调节高度 0.25，tank2 调节高度 0.4 符合预期（我们通过数据集设置参数值）
- 在模拟过程中，tank1 和 tank2 都进行了两次调节；首次调节流量0.02 m^3/s；第二次调节流量 0.06 m^3/s
- 在 tank1 有任何流量之前 Tank2 是空的

仿真分析2

在示例中，我们将坦克的属性 "minV"和 "maxV"分别设置为值 0 和 10。在现实世界中，10 m^3/s 的流量需要在水箱上安装一个非常大的阀门。

如果我们将 "maxV"的值更改为 0.05 m^3/s 会发生什么？基于之前的模型，我们可能会做出这些改变：

- 在 TanksConnectedPI 的现有 "DataSet_1"上，右键单击并选择 "复制数据集"，然后重命名为 "Tank2WithLimitValveSize"
- 单击按钮进行配置，展开 "tank2"并在 "属性"的 "值"列中键入 "0.05"
- 在 "仿真"页面上选择 "Tank2WithLimitValveSize"并绘制属性
- 单击求解按钮执行模拟



以下是对结果的分析：

- 我们的更改仅适用于 tank2； tank1 可以像以前一样在 0.02 m^3/s 和 0.06 m^3/s 上调节
- 当源流量为0.02 m^3/s时，tank2可以像以前一样调节
- 然而，当源流量增加到 0.06 m^3/s 时，阀门太小，无法让流出流量与流入流量匹配；唯一的结果是tank2的水位升高
- 然后由用户来解决这个问题；例如，换一个更大的阀门，减少源流量或制作一个额外的阀门

总之，此示例显示了如何通过复制现有 DataSet 来调整参数值。