ENTERPRISE ARCHITECT

用户指南系列

MDG技术

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

创建于 ENTERPRISE ARCHITECT

目录

MDG 技术 5
指定需要MDG 技术 6
与MDG 技术合作 8
管理MDG 技术 9
访问远程MDG 技术 11
导入MDG 技术模型 12
扩展 - MDG 技术 14
MDG 技术SDK 16
定义建模语言 17
开发Profiles 19
创建构造型Profiles 20
创建配置文件包 21
添加构造型和元类 23
创建构造型扩展非 UML 对象 26
在另一个配置文件中重新定义构造型 28
定义构造型标记值 30
将枚举添加到构造型 31
定义结构化标记值 33
使用标记值连接器 36
使用预定义的标签类型 37
使用预定义的标签类型（传统Profiles ） 40
定义构造型约束 41
添加形状脚本 42
设置默认外观 44
特殊属性 45
将构造型定义为元类型 50
定义多重刻板印象级别 51
定义实例的创建 52
定义复合元素 54
定义子图表类型 55
定义标签分组 57
介绍元模型视图 59
内置元模型图表视图 60
自定义元模型图表视图 64
定义元模型约束 69
元约束连接器上的约束 74
元模型约束和快速链接器 80
快速链接器 82
快速链接器定义格式 83
关系库表 87
快速链接器示例 89
隐藏默认快速链接器设置 91
快速链接器物件名称 92
将快速链接器定义添加到配置文件 95
导出一个配置文件 96
保存配置文件选项 98
浏览器-资源中的UML Profiles 99

浏览器-导入UML Profiles Into资源 100
MDG 技术-创造 101
使用配置文件Helpers 102
使用配置文件Helpers 创建构造型Profiles文件 104
使用配置文件Helpers 添加构造型和元类 106
编辑构造型元素 109
使用配置文件帮助器创建图表Profiles文件 110
使用配置文件帮助器创建工具箱Profiles文件 112
使用配置文件Helpers 创建隐藏的子菜单 116
创建MDG 技术文件 118
添加配置文件 120
添加模式 121
添加一个图表配置文件 122
添加工具箱配置文件 123
添加标记值类型 124
添加代码模块 125
定义代码选项 126
添加数据库数据类型 127
添加 MDA 转换 128
添加文档报告模板 129
添加链接文档模板 130
添加图片 131
添加脚本 132
添加工作区布局 133
添加模型视图 134
添加模型搜索 135
使用 MTS 文件 136
创建工具箱Profiles 137
创建工具箱Profiles 138
工具箱页面属性 141
创建隐藏的子菜单 142
将图标分配给工具箱项 144
覆盖默认工具箱 146
工具箱页面中使用的元素 148
工具箱页面中使用的连接器 151
创建自定义图表Profiles 153
内置图表类型 155
属性值 - styleex & pdata 156
设置技术元素图片 158
定义验证配置 159
合并模型Builder模板 160
添加导入/导出脚本 162
部署MDG 技术 164
形状脚本 165
开始形状脚本 166
形状编辑器 168
编写脚本 169
形状属性 172
绘图方法 175
颜色查询 181
条件分支 182

查询方法 183
显示元素/连接器属性 185
子形状 189
为元素添加自定义分区 190
显示复合图表 195
保留名称 199
语法文法 200
示例脚本 202
标记值类型 213
从预定义类型标记值类型 214
预定义的结构化类型 215
创建自定义屏蔽标记值类型 221
创建参考标记值 223
预定义参考数据类型 224

MDG 技术

MDG 技术是一种工具，用于提供对商业可用技术或您自己创建的技术资源的访问。这些资源包括范围广泛的功能和工具，例如UML Profiles 、代码模块、脚本、模式、图像、标记值类型、报告模板、链接文档模板和工具箱页面。

使用Enterprise Architect ，您可以开发基于标准UML规范的模型，并且可以使用 UML 支持的机制（如标记值、构造型、 Profiles和设计模式）扩展核心UML结构。这些功能在Enterprise Architect核心技术中，您可以激活和使用进一步的模型驱动生成 (MDG)技术，这些技术要么与系统集成，要么可从外部位置获得。

如果您的系统或工作领域需要进一步的专业化，作为技术开发人员，您可以使用Enterprise Architect开发您自己的定制建模语言和解决方案。

获取和使用技术

技术源
核心技术Enterprise Architect本身包含： • 基础 UML 2 技术作为UML 2.5 结构和行为建模的实现，以及 • 核心扩展技术，应用配置文件和刻板印象来提供需求、用户界面和数据建模等方面的扩展数据建模
Enterprise Architect安装目录 MDGTechnologies 子文件夹中包含其他技术。
您可以将外部来源的技术导入 APPDATA 文件夹 (%APPDATA%\ Sparx Systems \EA\MDGTechnologies) 以供您自己使用，或导入浏览器窗口的 资源”选项卡以供其他项目用户访问。
您可以将技术转移到 MDGTechnologies 子文件夹中；这些技术在您重新启动Enterprise Architect时可用（在 Vista/窗口系统上，您可能需要增加访问权限才能执行此操作）。
您可以从Enterprise Architect访问和激活远程系统文件夹或网站中的MDG 技术。
技术开发人员可以通过 MDGTechnologies 子文件夹或远程文件夹或网站创建新的MDG 技术并将其部署到项目团队。
要查看Enterprise Architect中可用的技术并激活您需要的技术，请使用 MDG 技术”对话框（ 特定>技术> 管理技术”功能区选项）。 使MDG 技术可用后，您可以管理它们对用户的可用性，并且可以与他们一起工作。 您还可以功能或禁用Enterprise Architect 基础 UML 2”和 核心扩展”技术和功能，以便您可以将Enterprise Architect功能和特征专门应用于一个或多个选定的MDG 技术。

指定需要MDG 技术

当您有必须使用某些MDG 技术的模型时，模型管理员可以配置系统以在加载过程中检查这些技术是否可用且处于活动状态，然后才能真正打开模型。您可以在 管理模型选项"对话框的 'MDG 技术"部分中识别技术。如果技术是：

- 需要且未安装在用户的机器上，该用户将无法打开模型
- 需要且可用，但未启用，系统可配置为自动启用该技术
- 具体不在本模型中使用，但可用并启用，系统可配置自动禁用该技术

管理员因此可以确保正确的操作环境在模型中工作，以便所有用户具有相同的视图并使用相同的模型功能（或者，至少没有使用错误的工具和创建结构其他用户无法使用）。

您可以在需要某些技术但其他技术可由用户自行决定使用的情况下使用 宽松"模型，或者在需要某些技术而所有其他技术被阻止的情况下使用 受限"模型。

访问

功能区	设置 >模型> 选项 > MDG 技术

选择需要技术

选项	行动
技术	审阅您当前可以访问的MDG 技术，按字母顺序列出。这些技术可能内置在Enterprise Architect中，由插件 提供插件 或从导入的目录或 URL。
需要	对于模型管理员，请针对打开模型之前必须可用的每项技术选中此复选框。 下次用户尝试打开模型时， Enterprise Architect将在允许访问模型之前检查所选技术在用户系统上是否可用。如果没有安装所需的技术， Enterprise Architect将不会打开模型。 此外，如果标记为需要的技术可用但未启用，系统将自动启用该模型；在用户可能访问的任何其他模型中，该技术仍将被禁用。
禁用	所有复选框默认为未选中，允许使用该技术。 选中模型中明确不得使用的每种技术的复选框。如果该技术可用并启用，系统会在模型中自动禁用它。它仍将在用户可能访问的其他模型中启用。
全部	单击此按钮以选中列表中每个技术的 需要"复选框。
没有任何	单击此按钮可清除列表中所有选中的 需要"复选框。

注记

- 在Enterprise Architect的企业版、统一版和终极版中，如果启用了安全性，您必须具有 配置项目要求”权限才能选中或清除技术对应的 需要”和 禁用”复选框

与MDG 技术合作

可以启用 "MDG 技术"对话框中列出的任何MDG 技术，这使得它们的界面配置文件和工具箱页面可供您使用。

当您启用MDG 技术时，任何特定于技术的图表类型都将添加到 "新图表"对话框列表中，并且该技术的图表工具箱页面将添加到通过工具箱的搜索功能可用的工具箱中。

如果将MDG 技术设置为 "Active"，它将成为模型的主要技术。一次只能激活一个技术。该技术的验证配置已设置，虽然常见的工具箱页面始终可见，但该技术的工具箱页面会覆盖任何并行的Enterprise Architect工具箱页面；例如，ICONIX 的 "类"页面会覆盖Enterprise Architect的 "类"页面。

您可以创建特定于技术的图表，并以与标准Enterprise Architect图表相同的方式使用元素和连接器填充它们。

管理MDG 技术

您使用 管理MDG 技术”对话框来管理项目可访问且可供项目用户使用的MDG 技术。该对话框列出了项目访问的多个位置中保存的技术，例如 APPDATA 文件夹和Enterprise Architect安装目录。您可以根据需要将这些技术设置为可用或禁用。 MDG 技术部署为 .xml 文件。

访问

功能区	特定>技术> 管理技术

配置技术可用性

选项	行动
技术	按字母顺序列出项目当前可访问的所有MDG 技术。 如果单击技术名称，对话框的右上方面板将显示该技术： ● 名称 ● 版本号 ● 徽标（如果已定义），以及 ● 部署的 XML 文件的位置，可以是： - Enterprise Architect内部 - 一个扩展 - 在安装目录中（只是文件名） - 在 APPDATA 文件夹中（文件名后跟（在 APPDATA 中）） - 在模型中 右下方面板显示技术描述，在许多情况下提供制造商的网站地址和支持联系人。
启用	针对您希望在项目中使用的每种技术选中此复选框。当启用MDG 技术时： ● 该技术已添加到默认工具工具栏的 配置文件”字段中的可用选项列表中，以便您可以应用MDG 技术的接口配置文件 ● 至少一组MDG 技术的工具箱页面会自动添加到图表工具箱；您可以通过 查找工具箱项”对话框访问添加的工具箱页面 ● 任何MDG技术特定的图表模板都添加到 新图表”对话框中以供选择；选择后，这些显示特定于图表的工具箱页面 清除针对 a技术的复选框以使其对项目用户不可用。 如果您禁用正在使用的MDG 技术，其工具箱页面、图表类型和快速链接将在用户界面中的图表工具箱、默认工具工具栏、图表和 新图表”对话框中省略。
全部	单击此按钮以选中对话框中列出的每个技术的 启用”复选框。
没有任何	单击此按钮可清除对话框中列出的每个技术的 启用”复选框。 如果单击此按钮，滚动到列表顶部并选择 基础 UML 2技术”和 核心扩展”复选框以重新启用 UML ”和 扩展”工具箱页面和图表类型。

设置为活动	将技术设置为 Active 使该技术成为Enterprise Architect的默认界面，并且可以： • 使用特定于活动技术的页面覆盖各种工具箱页面（包括来自其他技术的页面） • 在另一个配置文件中重新定义原型，添加新标签并删除或修改现有标签，而原型在所有其他方面的行为就好像它是原始原型一样 如果您的首选技术不使用覆盖和重新定义，则无需将其设置为 Active。 选择并突出显示您的首选技术，然后单击设置活动按钮。这会在 技术"面板中的技术名称旁显示一个星号，并选择默认工具工具栏的 配置文件"字段中的技术。如果MDG 技术尚未启用，此按钮也将启用它。
高级	单击此按钮可将MDG 技术添加到Enterprise Architect远程的文件夹和网站中。
消除	（仅对直接导入模型的技术启用。） 单击此按钮可从列表、浏览器窗口的 资源"选项卡和模型中删除选定的技术。
确定	单击此按钮关闭对话框，保存更改并使其生效。
取消	单击此按钮可关闭对话框并中止您所做的更改。

注记

- 如果您更改了MDG 技术的 启用"设置，或者如果您更改了外部路径列表，请单击确定按钮以重新加载所有启用的技术；您无需重新启动Enterprise Architect即可使更改生效
- 要专门使用选定的MDG 技术或少数技术，您可以仅启用这些技术（可能将其中一项设置为 Active），然后取消选中 基础 UML 2技术"复选框（如果需要，还可以 核心扩展"复选框）
- 对已添加新标记值且定型名称一致的配置文件的更新，则可以同步这些新的或更改的标记值；更多详情请参见同步标记值和约束帮助主题

访问远程MDG 技术

当您处理您的模型时，您可以使用系统本地的MDG 技术技术或者您可以访问您在远离系统的文件夹和网站中识别的技术。您基本上将这些远程技术为 书签”以供继续使用，然后在您不想再使用它们时删除链接。

访问

功能区	特定>技术> 管理技术：高级

注记

- 要删除 MDG 技术MDG 技术-高级”对话框中列出的 MDG 技术，请单击文件夹路径或 URL，然后单击 删除”按钮；路径或 URL 被删除

指定远程MDG 技术

节	行动
1	在 MDG 技术-高级”对话框中，单击 添加”按钮。 A简短的上下文菜单显示，提供： • '添加路径' • '添加网址'
2	要在目录文件夹中指定MDG 技术，请选择 添加路径”选项。 将显示 浏览文件夹”对话框。 浏览MDG 技术文件夹，点击它，然后点击确定按钮；转到第 4 步。
3	要在网站上指定MDG 技术，请选择 添加 URL”选项。 将显示 输入”对话框。 在 输入值”字段中，键入或复制并粘贴MDG 技术URL，然后单击确定按钮。
4	MDG 技术的文件夹路径或 URL 显示在路径面板中。 该技术可用

导入MDG 技术模型

如果您找到或创建了对您的项目有用的MDG 技术，您可以将其导入到项目中：

- 仅供您自己使用；也就是说，将技术导入工作站上的 %APPDATA%\ Sparx Systems \EA\MDGTechnologies 文件夹，或者
- 对模型的所有用户可用，通过模型浏览器窗口的 资源”选项卡

要导入MDG 技术，您必须有一个合适的MDG 技术XML 文件。如果MDG 技术包括对任何元文件的引用，它们应该与MDG 技术XML 文件位于同一目录中。

与模式MDG 技术一起提供的模型必须在Enterprise Architect安装目录的模型模式目录中都有相关的模式XML 文件和一个包含模型描述的相同文件名的模式文件。

启动时， Enterprise Architect扫描 APPDATA 文件夹和Enterprise Architect安装目录模型子文件夹中的技术文件，通过 ‘MDG 技术”对话框和浏览器窗口的 资源”选项卡使它们可用导入到 APPDATA 文件夹的.技术由文本 “Location:技术.xml”指示。模型模　需要单独导入到用户系统的模型模式目录中。

访问

功能区	特定>技术>发布技术>导入MDG 技术
上下文菜单	在浏览器窗口的 资源”选项卡中 |右键单击MDG 技术文件夹|导入技术

导入一项技术

节	行动
1	在 导入MDG 技术”对话框的 文件名”字段中，键入要导入的MDG 技术文件的路径和文件名，或使用…按钮进行浏览。 当您输入文件名时， MDG 技术名称和版本会显示在 技术”和 版本”字段中，任何注记都会显示在 注记”字段中。
2	为您要执行的导入类型选择适当的单选按钮： • 导入模型 • 导入用户
3	点击确定按钮。 • （如果您选择了 导入到用户”选项）如果 APPDATA 文件夹还不存在， Enterprise Architect会创建它 • 如果MDG 技术已经存在， Enterprise Architect会显示一个覆盖现有版本并导入新版本的提示 一旦导入到 APPDATA完成，您必须重新启动Enterprise Architect ；然后在 ‘MDG 技术MDG 技术”对话框中列出 MDG 技术。

注记

- 要删除已添加到 APPDATA 的MDG 技术，请在 %APPDATA%\ Sparx Systems \EA\MDGTechnologies 文件夹中找到相应的 XML 文件并将其删除
- 考虑到一些MDG 技术可能很大并且可能在工作站上施加一些延迟，因为它们在每次用户连接到模型时加载
- 要从浏览器窗口和模型的 资源”选项卡中删除MDG 技术，可以：
  - 右键单击技术名称并选择 删除技术”菜单选项，或
  - 单击 管理MDG 技术”对话框中的技术名称，然后单击
  删除按钮

扩展 - MDG 技术

Enterprise Architect是其建模功能的一系列模型驱动生成 (MDG) 扩展的核心，使用更专业的利基框架和配置文件。

扩展功能

扩展
许多技术已经与Enterprise Architect A程序集成，包括： • ArchiMate • BPEL • BPMN • 数据流图 • Eriksson-Penker 扩展 • ICONIX • 思维导图 • SoaML • SOMF 2. 1 • 策略建模 • 系统建模语言(SysML) • Eclipse 的MDG链接 • Visual Studio.NET 的MDG链接
Enterprise Architect提供以下支持： • 从外部系统文件或网站下载MDG 技术，或 • 使用Enterprise Architect MDG 技术向导轻松创建您自己的
Sparx Systems还销售一些MDG产品： MDG 技术为： • 扎克曼框架 • The Open Group架构框架(TOGAF) • 统一架构框架（UAF），原为 DoDAF 和 MODAF 的统一配置文件（UPDM） • 数据分布服务(DDS) • Python（ Enterprise Architect版本 4.5 到 5.0；集成在更高版本中）（*免费产品！*） • CORBA（* 免费产品！*） • Java Beans（* 免费产品！*） • 测试（*免费产品！*） MDG集成对于： • 日食 3.3 • 视觉工作室 2005、2008 和 2012 MDG链接 • Microsoft Visio（* 免费产品！*） • IBM Rational软件架构师（原 Telelogic） DOORS

随着时间的推移，此列表正在扩展以包括更多产品。
Sparx Systems提供Enterprise Architect的扩展版本，为系统工程和业务工程提供更大的支持。 这些版本包含几个列出的MDG 技术和其他插件。
有关可用插件的最新列表和每个产品的介绍，包括定价、购买和下载选项的详细信息，请参阅Sparx Systems网站。 当您购买其中一个插件时，您会收到一个或多个许可证密钥以及有关获取、安装和注册产品的说明。

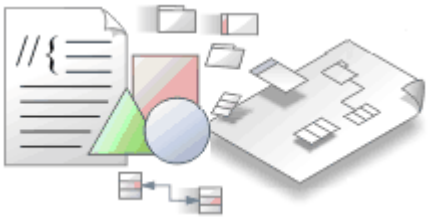
MDG 技术SDK

Enterprise Architect是一个多功能工具，具有数百个内置特征，并支持开箱即用的各种建模标准。它还提供了一系列有用的扩展机制。 Enterprise Architect软件开发工具包 (SDK) 包含扩展核心UML以支持特定领域、平台或方法的建模的机制。 Enterprise Architect和其他合作伙伴组织提供商业上可用的模型驱动生成 (MDG)技术，但任何人都可以免费使用 SDK 创建新的配置文件并将其作为MDG 技术分发。例如，您可能在安全工程领域工作，并使用特定的结构来模型您的领域和使用的方法。例如，您可以使用Enterprise Architect创建新元素来表示故障事件、故障模式和任何其他特定于域的实体。一旦配置文件完成，就可以将其捆绑到MDG 技术中，然后在您的组织内本地使用或分发给整个行业。

注记

- 在开发您的技术时，您需要熟悉核心系统和扩展机制的建模结构和概念，因为它们影响并被您为其设计技术的人使用；即本用户指南的建模部分中描述的系统

定义建模语言



如果您想执行更专业的建模，您可以扩展基本的UML建模元素及其使用来开发您自己的建模语言或解决方案。A简单的方法是开发和部署MDG 技术，它可以包含许多专门的Profiles和一系列其他机制，为您的定制解决方案提供最广泛的范围。

扩展功能

功能	描述
MDG 技术	MDG 技术是一种工具，用于提供对商业可用技术或您自己创建的技术资源的访问。这些资源包括范围广泛的功能和工具，例如UML Profiles 、代码模块、脚本、模式、图像、标记值类型、报告模板、链接文档模板和工具箱页面。
Profiles	Profiles是一种扩展UML的方法；您可以使用它们来构建特定领域的模型。 配置文件是扩展或应用于元素、属性、方法和连接器的附加构造型和标记值A集合，它们一起描述一些特定的建模问题并促进该域中的建模构造。
构造型	构造型是一种内在的机制，用于在逻辑上扩展或改变元素模型的含义、显示和句法。不同的模型元素具有与之相关的不同标准构造型。 当您自定义自己的构造型时，同样的原则也适用，或者通过“UML类型”对话框来限定现有类型的元素，或者作为扩展特定元类的元素来定义新的元素类型。
设计模式	模式是可以从一组一般建模场景（即参数化协作）中抽象出来的协作对象/类的组。 它们通常描述如何解决抽象问题，并且是实现重用和构建健壮性的极好方法。
形状脚本	形状脚本脚本是A脚本，它将自定义形状和方向应用于元素或连接器，以代替该object的标准UML表示法。每个脚本都与一个特定的原型相关联，并为每个具有该原型的object绘制。 在您重新定义标准UML object的属性以创建新object的地方，您也可以将新形状应用于object 。
标记值类型	您可以使用标记值向模型元素添加更多属性。您可以在三个级别应用它们： • 作为与元素模型相关的标准标记值 • 作为基于标准标记值类型的定制标记值 • 作为基于自定义标记值标记值类型的自定义标记值
代码模板框架	在Enterprise Architect中，您可以通过自定义控制这些操作的模板来修改生成或转换代码的方式，包括为行为模型生成代码。您还可以将这些模板合并到一项技术中，以将自定义生成和转换添加到该技术的功能中。

开发Profiles

Profiles是扩展的集合，基于应用于UML元素、连接器和特征的构造型。构造型可以具有专门定义标记值的属性，从而进一步扩展构造型元素或连接器的特性。 Profiles存储为具有特定格式的 XML 文件；要应用配置文件的扩展，您将其 XML 文件添加为MDG 技术的组件，并部署该技术；那是：

1. 创建一个模型来开发MDG 技术，并在其中创建一个配置文件包，您可以在其中定义您的配置文件
2. 将配置文件另存为具有特定格式的 XML 文件。
3. 使用MDG 技术创建向导将 XML 文件调用为MDG 技术技术。
4. 在您的系统上部署MDG 技术（以及配置文件）。

创建构造型Profiles

当您创建一个配置文件来定义一个新的建模解决方案时，您最初会创建一个带有 «profile» 原型的包。然后考虑需要创建的模型元素（以及因此的构造型元素）的数量。如果您要创建：

- A的构造型元素，可以在配置文件包内的单个子图上进行管理，并将图保存为配置文件
- 大量A构造型元素，在方便的情况下在尽可能多的子图上创建它们（如果您愿意，每个图一个构造型）并将包保存为配置文件

每个构造型元素至少扩展一个元类元素。构造型元素使用配置文件名作为它们的命名空间。创建配置文件后，您可以将其合并到MDG 技术中。

创建配置文件并将其应用到模型的过程包括多个步骤。仅当您希望配置文件将特定含义、显示、外观或语法应用于模型元素类型时，这些步骤中的某些步骤才是必需的。

创建配置文件

节	描述
1	在技术开发模型中创建配置文件包。
2	将构造型和元类元素添加到配置文件包的子图中。
3	为构造型标记值。
4	定义构造型元素的约束。
5	添加枚举元素以定义构造型上标记值的下拉元素。
6	为构造型元素添加形状脚本。
7	为每个原型模型元素设置默认外观。
8	在配置文件中包含快速链接器定义。
9	将包或图表另存为配置文件，然后导出。
10	将配置文件合并到MDG 技术中并部署该技术。

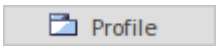
注记

- A配置文件包可以包含多个图表和许多元素和连接器，但不能包含其他包；不要在配置文件中使用嵌套包
- 如果您正在创建配置文件以构成MDG 技术的一部分，请注记您在单独的Profiles中为该技术定义了特殊的工具箱页面和图表

创建配置文件包

创建UML配置文件以定义新模型元素的第一步是创建一个包，该包在您的技术开发模型中具有构造型«profile»。

工具箱图标



访问

创建一个新的包图，然后显示图表工具箱打开 配置文件"页面。
使用此处列出的方法之一来访问工具箱的 配置文件"图表。

功能区	设计>  工具箱图表查找工具箱项"对话框并指定 配置文件"
键盘快捷键	Ctrl+Shift+3 :  显示 查找工具箱项"对话框并指定 配置文件"
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。

创建配置文件包

节	描述
1	在 New图表"对话框中，单击 Select From"字段中的 UML Structural"和 包"字段中的 图表"。 点击确定按钮。新图表在 图形"图表视图中打开。
2	打开工具箱的 配置文件"页面（点击图表显示 查找  工具箱"对话框并指定 配置文件"）。
3	将 配置文件"项拖到包图上。 将显示 新模型包"对话框。
4	在 包名称"字段中，输入配置文件的名称并选中 自动添加新图表"复选框。 点击确定按钮。将显示 新图表"对话框。
5	在 名称"字段中，输入图表名称，然后在 Select From"字段中单击 UML Structural"，在 类"字段中单击 图表"。
6	点击确定按钮。 系统创建一个带有原型«profile»的包和一个子类图。 根据您的系统设置，可能会显示包的 属性"对话框。如有必要，您可以添加要分配给包的任何基

	本包详细信息，例如版本、相或注记。
7	在图上，双击配置文件包打开子图。 您现在使用此子图将构造型元素添加到配置文件中。

添加构造型和元类

当您扩展UML以开发特定于域的工具集时，您首先为要定制的构造型创建一个配置文件包。此包至少有一个子类图，您在此子图上指定：

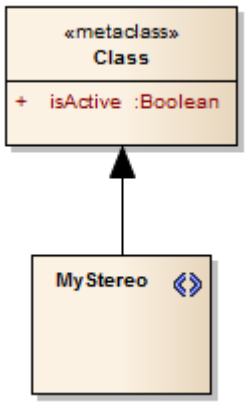
- 您正在扩展的object类型，由 Metaclass 元素表示，以及
- 每个object的扩展方式，由构造型元素表示

您可以使用一系列其他工具来限定构造型对元类的影响，包括：

- 形状类型中的构造型
- 由标记构造型元素中的属性定义的标记值
- 结构化标记值类，使用构造型元素标记中的属性定义
- 枚举，使用构造型元素中的属性定义
- 标记值连接器，用于识别以构造型生成的元素中的标记值的可能值
- 关于构造型元素的约束
- 特殊属性，定义原型元素的特定默认行为，例如元素的初始大小和颜色
- 修改构造型元素的默认外观

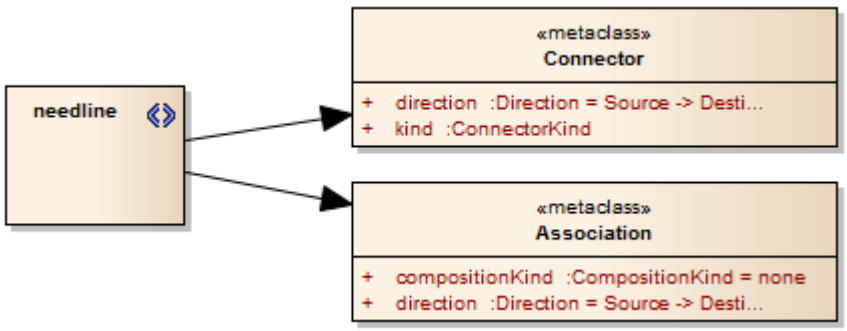
将元类和构造型添加到配置文件

节	描述
1	打开配置文件包的子图。
2	将元类元素从工具箱的 配置文件"页面工具箱图表上。 将显示 扩展元类"对话框，列出您可以扩展的object类型，即： ● 核心UML元素、属性和操作 ● 核心连接器 ● 抽象元类型，例如行动类型、ConnectorEnd 和门，以及 ● 构造型 在 核心元素"选项卡上，您可以通过选中 包含扩展"复选框来包含系统定义的扩展元素集，例如 ActivityRegion 和更改用户。 在 构造型"选项卡上，要指定包含要扩展的构造型的技术，请单击顶部字段中的下拉箭头并选择技术名称。
3	滚动所选列表并勾选一种或多种object类型以进行扩展。 如果要选择选项卡上的所有对象，请单击全部按钮。
4	点击确定按钮。 对于您选择的每个复选框，都会在图表上创建一个新的元类元素。
5	将构造型元素从工具箱拖到图表上。 如果没有显示 属性"对话框，请双击图表上的元素。
6	在名称字段中，输入构造型的名称。
7	点击确定按钮。

8	单击工具箱中的扩展关系并将连接从构造型元素拖动到它将扩展的元类元素。
9	您的图表现在类似于此示例： 
10	或者，您可以现在添加到您的构造型元素： • 构造型标签 • 枚举标签 • 结构化标记值 • 标记值连接器 • 特殊属性 • 约束和/或 • 形状脚本 您还可以根据需要定义元素或连接器的默认外观。

注记

- 如果您打算扩展大量的模型元素，而不是将它们全部放在一个图表上，您可以在配置文件包下创建额外的子类图表，并将不同类型的元类元素添加到不同的图表中；在这种情况下，您将包保存为配置文件，而不是单个图表
- 如果您想要一个构造型扩展多个元类，请创建一个构造型元素，并为每个元类元素创建一个扩展连接器，如下所示：



- 构造型元素必须具有唯一的名称，但元类元素可以具有相同的名称（例如，可以有多个行动元类，每个具有不同的行动属性）

创建构造型扩展非 UML 对象

配置文件通常A扩展核心UML object类型来创建您自己的建模语言或技术来定义；但是，您也可以扩展由其他现有技术（如 ArchiMate、BPMN 或 SysML）定义的非 UML 对象。

扩展非 UML object允许从现有构造型继承属性，即：

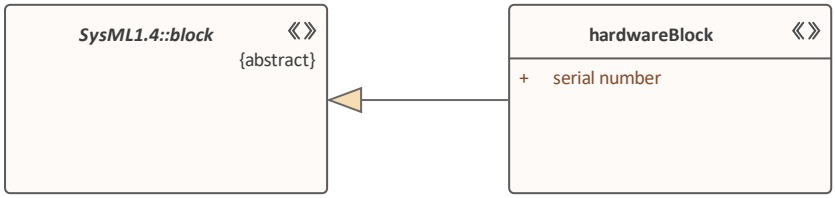
- 标记值
- 形状脚本
- 构造型颜色和
- 元类型属性

创建一个构造型扩展一个非 UML物件

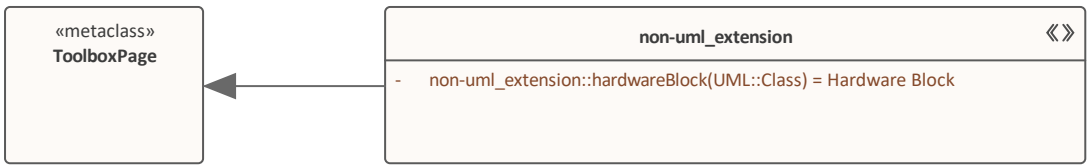
节	描述
1	在浏览器窗口中，找到带有<<profile>>构造型包并打开其子图。 如果您没有现有的<<profile>>包，请使用模型生成器中的 管理 | MDG 技术生成器”选项来创建新技术，然后从新创建的<<profile>>包中打开图表。
2	将 无类”图标从 工具箱”的 配置文件”页面图表图表上。 将显示 扩展元类”对话框。
3	选择 构造型”选项卡。
4	从下拉列表中，选择要扩展的配置文件（例如，“SysML1.4”）并选中要扩展的非 UML构造型旁边的复选框（例如， 块”）。 点击确定按钮。 适当的构造型元素被添加到配置文件图中。
5	通过从图表工具箱中拖动 添加构造型配置文件帮助器”来添加新的构造型工具箱 这将是扩展在步骤 4 中添加到图表中的非 UML 类型的自定义构造型类型。 完成后，构造型元素和元类元素将显示在配置文件图上。
6	从第 5划中添加的自定义概括连接或在第 4 步中添加的非 UML构造型构造型元素
7	将图表另存为配置文件。
8	定义一个工具箱配置文件，其中包含每个构造型的项目。
9	将保存的Profiles合并到MDG 技术中。

示例构造型配置文件

此示例显示定义构造型<<hardwareBlock>> 的构造型配置文件。 <<hardwareBlock>> 原型与 SysML块具有泛化关系，来自 SysML MDG 技术。这意味着在任何允许使用 SysML块的地方都可以使用 hardwareBlock。



这个例子展示了一个工具箱如何允许创建一个hardwareBlock。注记扩展总是原始原型扩展的UML类型。它绝不是对刻板印象的参考。



注记

- 当使用形状脚本来自定义构造型的外观时，您可以使用形状脚本() 方法来呈现为正在扩展的非 UML object定义的形状
- 如果您将任何元类元素属性添加到您的构造型，或者如果您想使用配置文件帮助器来创建一个工具箱配置文件，那么您的构造型类必须扩展一个元类以及专门化一个构造型

在另一个配置文件中重新定义构造型

如果您想在另一个配置文件中重新定义构造型，添加新标签并删除或修改现有标签，而构造型在所有其他方面的行为就好像它是原始构造型一样，您可以使用此处描述的重定义关系。

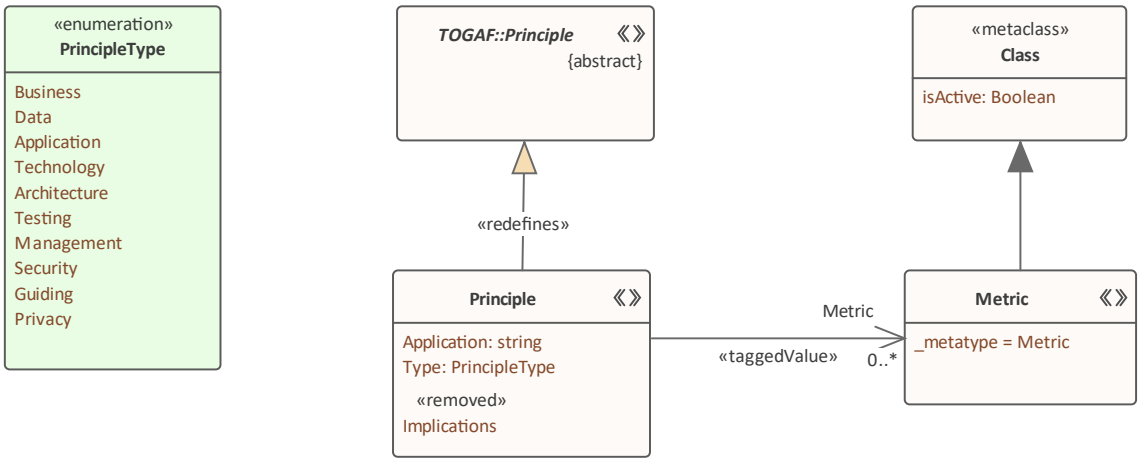
应用重新定义关系

节	行动
1	创建一个构造型元素，其名称与您正在重新定义的构造型的完全限定名称相同。请参阅示例中的“TOGAF::Principle”。 将此构造型元素设置为 抽象”。
2	创建具有相同名称的构造型元素，而不是完全限定的。请参阅示例中的 原理”。 划a 将关系从重新定义的刻板印象重新定义为重新定义的刻板印象。
3	到： • 从重新定义的原型中**删除**一个标签，将图表工具箱的 配置文件”页面中的 已删除属性”属性拖放到重新定义的原型上。使用与要删除的标签相同的名称该属性。这将创建一个具有原型 <<removed>> 的新属性，从而防止从原始原型继承标记值；请参阅示例中的 含义” 使用我们的配置文件创建的 A ”元素将以各种方 充当 TOGAF 原则元素，但不会有通常的 含义”标签 • 向重新定义的构造型**添加**新标签，只需为重新定义的构造型提供标签即可；在示例中，新的 应用程 ”标签不是由 TOGAF 配置文件提供的，但会显示为 • **修改**重新定义的构造型中的现有标记值类型，为重新定义的构造型提供一个与要修改的标签同名但类型不同的标签；在示例中， 类型”是来自 TOGAF 配置文件的枚举，但我们已为其提供了一组修改后的枚举文字，“Metric”是 TOGAF 配置文件中的纯文本标签，但我们已将其重新定义为引用新 Metric”构造型的 RefGUIDList 标签
4	在将配置文件保存并部署在MDG 技术中之后，用户可以通过将技术设置为 活动”来指定创建的任何重新定义的元素都应该使用活动技术中的重新定义来创建。

示例图表

This diagram demonstrates a more complex scenario for extending a non-UML type. It demonstrates the capability of using a Redefines relationship to declare that this should behave like it is the original Principle element from TOGAF.

It also demonstrates how to remove and override Tagged Values defined in the original profile. Elements created with this stereotype will:
1. Not contain an Implications tag
2. Provide different options for Type from the base profile.
3. Allow metric definitions to be re-used by replacing the plain text with a RefGUIDList to a new Metric stereotype
4. Add a new simple tag "Application" that appears in the TOGAF::Principle group



Optionally, an end user can specify that creating a TOGAF::Principle should actually create an instance of Principle from this profile. They do that by setting this profile as Active (which can only be done for a single technology.)

定义构造型标记值

您可以通过添加各种类型的元素标记值来为构造型定义额外的元信息，您将其标识为构造型的属性。最简单的标记值是那些在 值"字段中键入纯文本的值。

对于更复杂的标记值，例如枚举和结构化标记值，请参阅以下主题：

- 将枚举添加到构造型
- 定义结构化标记值
- 使用预定义的标签类型定义构造型标签

访问

使用此处概述的方法之一显示特征窗口的 属性"页面。

功能区	设计>元素>编辑>特征>属性
上下文菜单	在浏览器窗口或图表中|右键单击元素|特征|属性
键盘快捷键	F9 或 Ctrl+5 > 属性

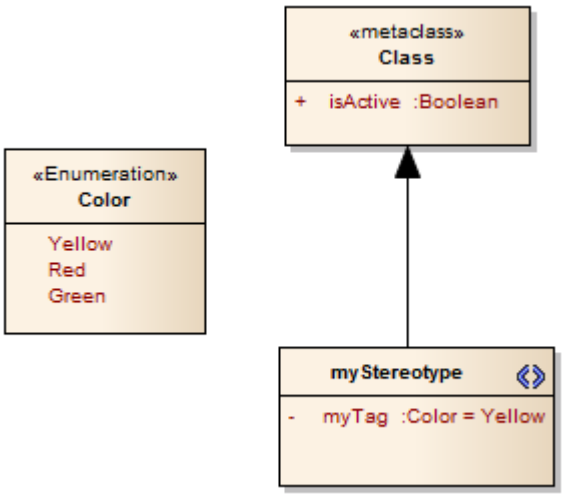
为构造型元素标记值

字段	行动
名称	用新属性/标签的名称改写新属性文本。
类型	默认为int 。如有必要，单击下拉箭头并选择不同的属性类型。
范围	默认为私人。如有必要，单击下拉箭头并选择不同的范围值。
构造型	如果需要属性构造型，请单击  图标并从 构造型选择器"对话框中搜索和/或选择构造型。
别名	如有必要，输入属性/标签的别名。
初始值	（可选。）类型属性/标签的初始值。

将枚举添加到构造型

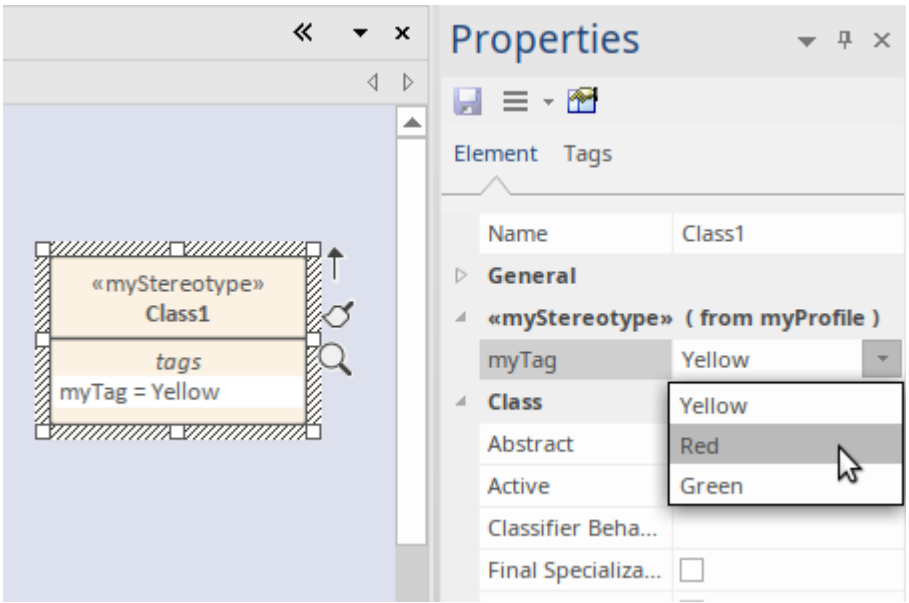
枚举元素可用于生成与构造型元素关联的标记值的下拉列表。在属性窗口的 标签”选项卡中显示列表和选定的值。

继主题定义排序标记值构造型之后，此示例说明如何使用枚举 颜色”为 myTag”提供值的下拉列表（ 黄色”、 红色”、 绿色”）对元素的价值'我的标记值印象'。



给构造型添加一个枚举

节	描述
1	打开配置文件包图。 在这张图上，我们应该已经有了元素<<metaclass>>类和原型元素'myStereotype'。
2	在工具箱中，找到并选择 配置文件”页面。
3	将 枚举”图标从工具箱拖到图表上。
4	如果尚未显示，请打开 属性”对话框。 功能区：'设计>元素>属性>常规>属性'（或按对话框+2）
5	在 名称”字段中，输入新枚举元素的名称。
6	如果尚未显示，请在 属性”页面打开特征窗口： 功能区：'开始>所有窗口>>属性>元素特征>属性
7	在 名称”字段中，输入枚举属性的名称（例如， 黄色”），然后按 Enter”。
8	单击*New Attribute*文本并输入下一个枚举属性的名称。对其他属性重复此步骤，以定义下拉列表的其他值。
9	右键单击构造型元素 myStereotype”并选择 特征>属性”选项。 原型的特征窗口显示在 属性”页面上。

10	在 名称"字段中输入属性的名称。
11	在 类型"字段中，单击下拉箭头并单击 类型"选项，然后从 选择 <Item>"对话框中浏览并选择枚举元素的名称。
12	在 初始"字段中输入定义默认值的必需枚举属性的名称。
13	单击关闭按钮。 您现在已经生成了一个下拉列表，用于在属性窗口的 标签"选项卡中设置标签的值。使用配置文件时，使用构造型创建的元素的标记值可能如下所示： 

定义结构化标记值

如果要定义一个包含多个组成部分的属性，例如地址，可以使用结构化标记值。这由一组相关的简单标记值序列，它们一起定义了属性。例如，街道地址的结构化标记值标记值有以下组成部分：

物业编号 - 448

街道 - 我的街道

城镇 - 克雷斯维克

区号 - 3363

当您最初在属性窗口或元素的标签隔间中显示它时，标签的值显示在一个string中，例如：

448，我的街，克雷斯维克，3363

然后，您可以展开结构标记值以列出组件标记名称和值。

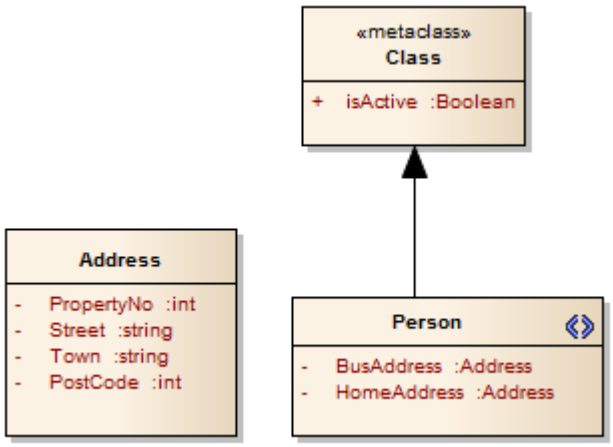
您使用非定型类在配置文件中创建结构化标记值。任何由构造型元素拥有的由此类类型键入的配置文件中的属性都将定义结构化标记值。

创建结构化的标记值类

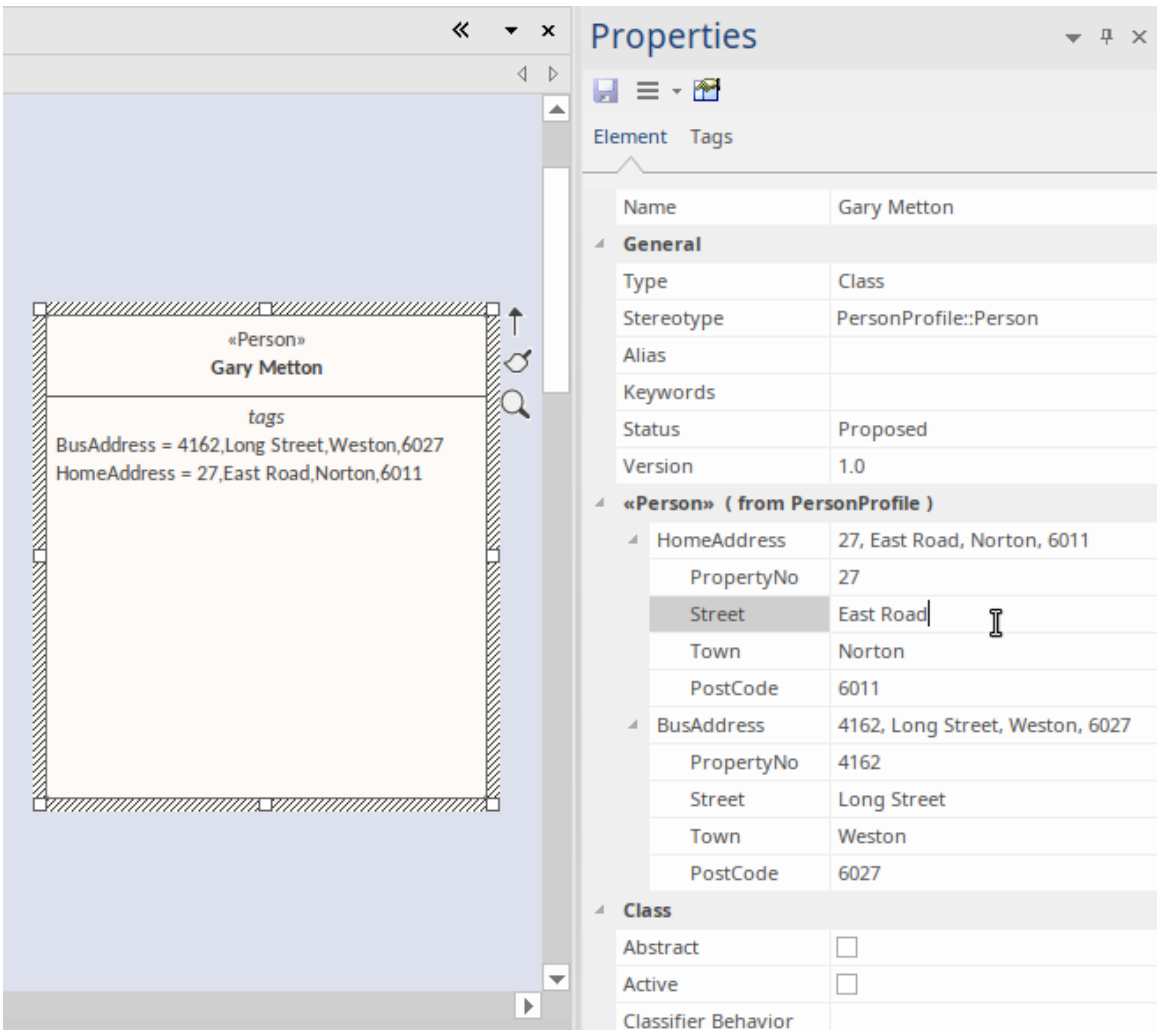
节	描述
1	在您的配置文件包中，打开子类图。
2	在工具箱中，找到并选择 类"页面。
3	将一个类项从工具箱拖到图表上。 如果没有显示 属性"对话框，请双击图表上的元素。
4	在 名称"字段中，输入新类元素的名称。
5	单击 详细信息"选项卡和 属性"按钮。 特征窗口显示，显示 属性"页面。
6	在 名称"字段中，输入结构化标签属性的名称（例如*PropertyNo* ）。
7	在 类型"字段中，单击下拉箭头并选择适当的类型（例如 "int "或 "string "）。
8	单击*New Attribute*文本，然后对每个剩余的组件标记属性（例如：Street、Town、AreaCode）重复步骤 6 到 8。
9	定义完所有组件标签后，点击构造型元素；特征窗口显示在 属性"页面上，用于构造型。
10	在 名称"字段中输入属性的名称（例如："HomeAddress"）。
11	在 类型"字段中单击 ... 按钮并从 选择<项目>"对话框中选择结构化标记值类元素的名称，作为属性的分类器。 您现在已经为从配置文件的这一部分派生的任何元素生成了要在属性窗口中维护的结构化标记值的组件。
12	继续定义配置文件，然后将图表或包保存为配置文件，然后将其导出以供使用或将其添加到MDG

	技术文件中。

示例



当这些元素作为配置文件导入模型时，定义了一个可以应用于类元素的 'Person "原型。此构造型允许您在应用构造型的元素中以结构化标记值的形式输入家和业务地址详细信息。



注记

- 通过配置文件应用结构标记值的过程是通过插件
  应用标记值的替代方法插件
  播送;查看了解更多主题
- 构成一个结构的标记值标记值必须简单；备注标记值不能包含在结构化标记值中

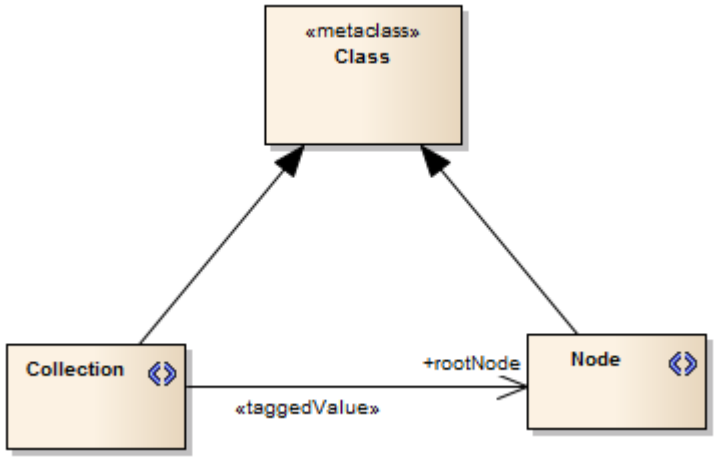
使用标记值连接器

创建配置文件时A常见情况是一个原型的实例需要引用应用了另一个原型的元素。例如，一个定义了 Collection 的元素可能有一个名为 rootNode 的标记值来标识该 Collection 的根，这将是一个具有构造型 <<Node>> 的类。

在属性窗口中，用户点击根节点标记值的选择按钮（ ... ）；当 选择<Item>"对话框出现时，用户可以定位当前模型中的所有节点，并选择其中一个元素作为标签的值。

为此，您可以使用工具箱 配置文件"页面中的标记值连接器。 A标记值连接器定义了一个标记类型（即 RefGUID ）属于源原型的标记值；标记值名称是该连接器的目标角色的名称，并且标记值仅限于引用具有目标元素的构造型的元素。

此图演示了如何使用连接器来表示示例。 A配置文件定义了两个原型：«Collection» 和 «Node»（它们都扩展了 Metaclass类）。 «Collection»构造型拥有一个标记值连接器，其目标角色为rootNode，指向«Node»构造型。您在连接器 属性"对话框的 角色"页面上输入目标角色名称。



注记

- 标记值连接器也可以直接与元类元素链接，以识别基本UML元素类型；例如：如果目标是参与者元类，当您选择识别特定目标时， 选择 <item>"对话框将列出基 参与者元的所有元素
- 此外，连接器可以链接到元素类型组的元类，即分类器和属性；如果连接器目标是元类：
  -分类器，当您选择识别特定目标元素时， 选择<item>"对话框将
  列出所有企业架构师定义的分类器类型，例如类和部件
  -属性，当您选择识别特定目标元素时， 选择 <item>"对话框将列出
  列出端口、部件和属性元素

使用预定义的标签类型

标记值定义了一个模型元素具有的范围广泛的属性和特征，而这些属性中的一些是复杂的或结构化的值。例如，您可能希望您的用户在上限和下限之间选择一个值（使用 旋转"箭头）、设置日期和时间、从调色板中选择一种颜色，或者通过核对表进行操作。

您可以从许多预定义的简单标记值类型和过滤器中的任何一个创建这些复杂的标记值，其中一些可能是您自己创建的，使用工具箱的 配置文件"页面中的 图表类型"元素。

使用数据类型元素的巨大优势在于它可以帮助您定义特定于profile的标记标记值，因此您可以在不同的profile中创建具有相同名称的不同类型的标记值，而不会在运行MDG 技术时产生冲突那些配置文件。

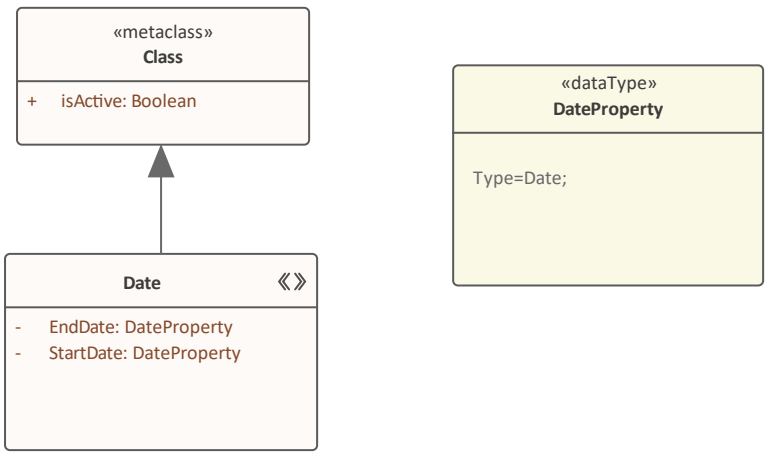
建议使用此方法从预定义的简单标记值类型和过滤器创建复杂的标记值配置文件。出于维护现有配置文件的目的，仍支持在UML类型对话框中定义全局标记值类型的原始方法；请参阅*With Predefined Tag Types (Legacy Profiles)*帮助主题。

将值分配给构造型标记值

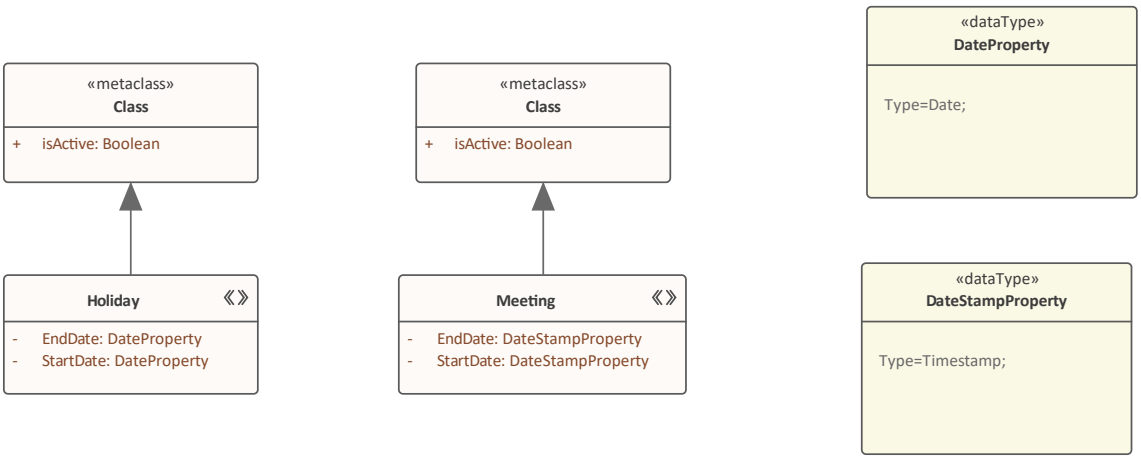
创建结构化标记值后，您可以将其分配给构造型元素，方法与简单标记值相同，方法是在构造型元素中使用标记值类型的名称创建一个属性。

示例

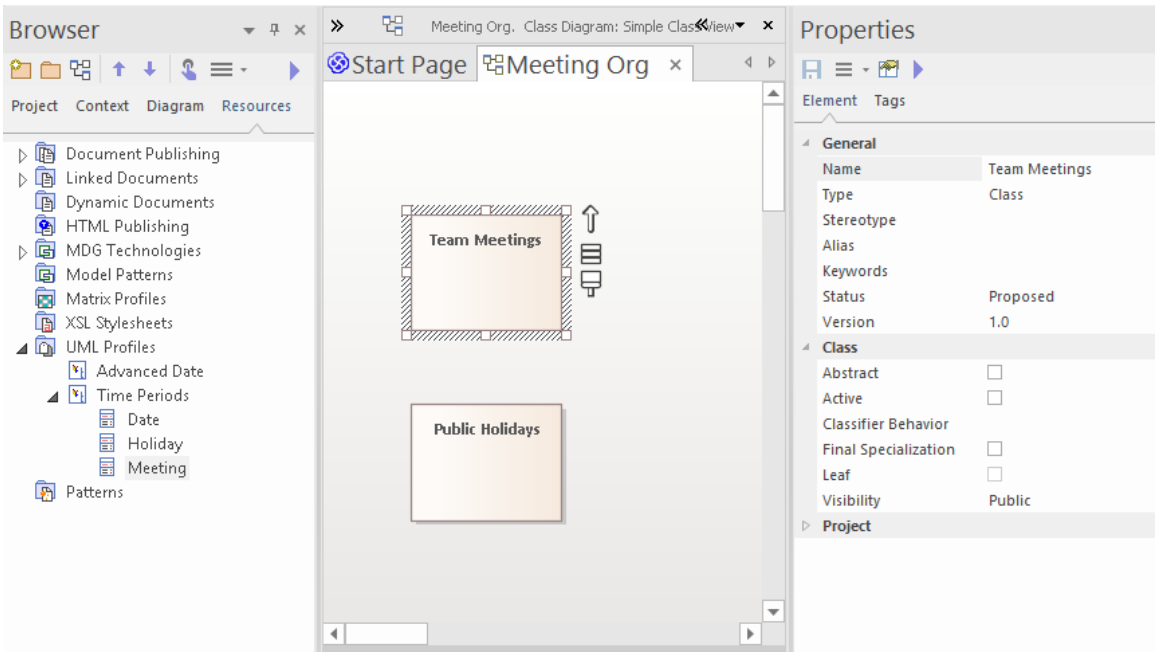
考虑 开始日期"和 结束日期"标记值的例子。使用 DataType元素，您将在标记值的注记中定义一个类型- '标记值' 帮助标记值类型（使用*Predefined Structured Types*帮助中列出的定义），并从任意数量引用该类型构造型元素中的属性 - 例如 开始日期"和 结束日期"。此定义及其引用对象与父包之外的任何其他定义无关。



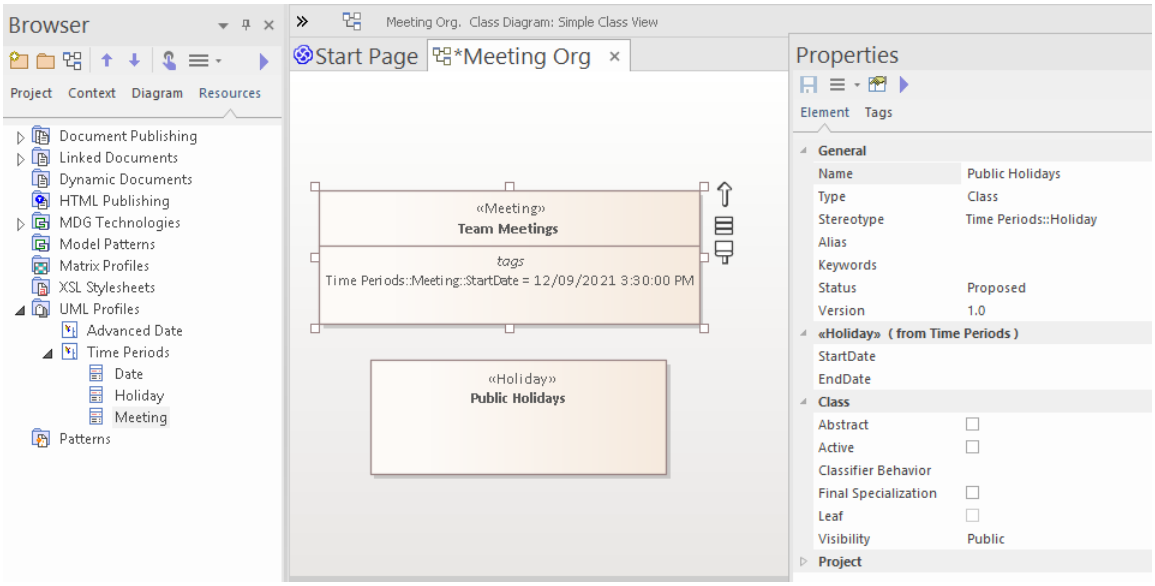
扩展此示例，假设您有名为 'Holiday' 和 'Meeting' 的构造型，并且两者都有 StartDate 和结束日期属性。但是，"Holiday"使用 DateProperty"，定义为 Type=Date;"，而 Meeting"可以使用 DateTimeProperty"，定义为 "Type=Timestamp;"。



将配置文件导入用户模型时（作为独立配置文件或作为MDG 技术的组件），可以将构造型应用于新元素或现有元素，并添加标记值类型（并在创建时可用）'标记值'对话框下拉菜单中的更多标签）。在此插图中，时间周期配置文件已导入模型的 资源"选项卡，将用于定制图表上存在的两个标准类元素。

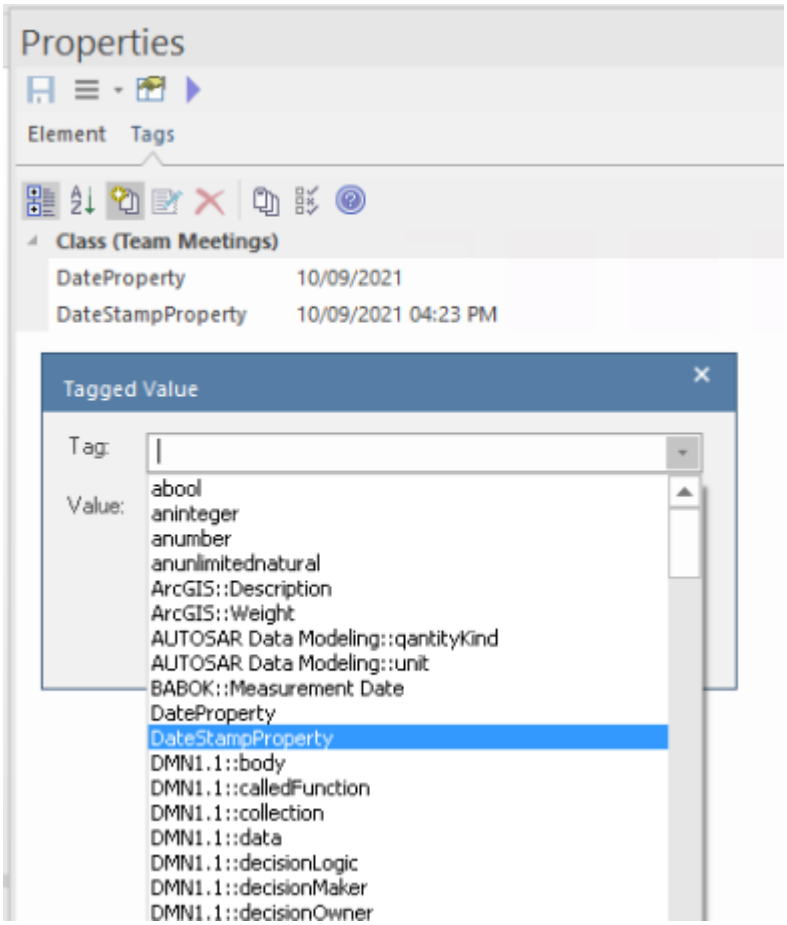


'Meeting' 配置文件元素现在是 Ctrl+拖动到 Team Meetings类，而 'Holiday' 配置元素是 Ctrl+拖动到 Public Holidays类。结果是两个类元素都采用了适当的构造型，这些构造型显示在图表中的元素上以及属性窗口的 元素"选项卡上的 构造型"字段中。另请注意， 元素"选项卡上有一个构造型组，列出了为构造型定义的标签。



对于团队会议元素，我们刚刚在元素的 开始日期"字段中输入了一个值，该值立即显示在图表上元素的 标签"隔间中。

如果您需要将构造型标签添加到其他类元素，一旦导入配置文件，您可以通过每个元素的属性窗口的 标签"选项卡执行此操作，选择最适合该元素的格式。注记如何此示例中两种格式不同，其中插入了两种数据类型。



使用预定义的标签类型（传统Profiles ）

标记值定义了一个模型元素具有的范围广泛的属性和特征，而这些属性中的一些是复杂的或结构化的值。例如，您可能希望您的用户在上限和下限之间选择一个值（使用 旋转"箭头）、设置日期和时间、从调色板中选择一种颜色，或者通过核对表进行操作。

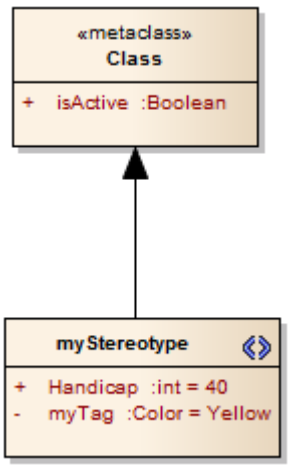
您可以使用 "UML类型"对话框的 标记值类型"页面 - '设置>参考，从许多预定义的简单标记值类型和过滤器中的任何一个创建这些复杂的标记值，其中一些可能是您自己创建的> UML类型 >标记值类型'。您在标记值的注记中指定标记值标记值类型（使用*Predefined Structured Types*帮助主题中列出的定义）。

请注记，此方法支持维护现有配置文件和MDG 技术，但对于新的或不完整的配置文件，我们建议您使用类型-请参阅*With Predefined Tag Types*帮助主题。

使用数据类型元素可以帮助您定义特定于配置文件的标记标记值，因此您可以在不同的配置文件中创建不同类型的同名标记值标记值而不会在运行从这些配置文件派生的MDG 技术时发生冲突。 标记值类型"页面适用整个模型，所有同名标签必须为同一类型。这可能会妨碍您在模型中创建多个配置文件，当他们使用这些全局标签并且其中一个或多个发生冲突时，无论是在技术开发模型中还是在启用该技术的任何模型中。

将值分配给构造型标记值

创建结构化标记值后，您可以将其分配给构造型元素，方法与简单标记值相同，方法是在构造构造型元素中创建一个具有标记值类型名称的属性。例如，要使标记值'Handicap' 出现在构造型中，请创建名为'Handicap' 的属性。根据标签类型，您可以通过为属性赋予初始值来设置标签的默认值。



定义构造型约束

如果需要定义构造型元素运行和存在的条件和规则，可以通过在元素上设置约束来实现。典型的约束是前置条件和后置条件，它们指示在创建或访问元素之前必须为真的事物以及在元素被销毁或其动作完成之后必须为真的事物。

您可以使用 分隔可见性”函数直接在图表上显示元素的约束。

访问

选择构造型元素，然后使用本表中列出的任何方法显示 属性”对话框的 约束”页面。

功能区	设计>元素>职责>约束
上下文菜单	右键单击元素|属性|职责 >约束
键盘快捷键	Shift+Alt+C
其它	双击构造型元素>约束

为构造型定义约束

字段/按钮	描述
新的	单击此按钮可清除准备创建新约束的字段。
约束	类型约束的值。
类型	单击下拉箭头并选择适当的类型（前置条件、后置条件或不变量）。
状态	单击下拉箭头并选择适当的状态。
注记	类型所需的任何附加信息。
节省	单击此按钮以保存约束数据。
确定	单击此按钮可关闭对话框。

添加形状脚本

UML元素和连接器在形状、颜色和标签方面都具有标准外观。可以通过多种方式更改元素类型或连接器的外观，使用形状脚本来定义要施加在默认或主形状上的确切特征。如果要标准化外观以应用于许多元素，请将形状脚本脚本附加到UML配置文件（例如 MDG元素MDG 技术UML配置文件）中的构造型属性。

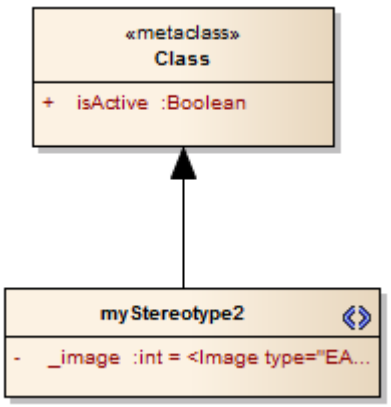
访问

对于在UML配置文件中定义构造型的元素，定义一个名为 形状脚本”的属性，该属性将指定形状脚本。
通过单击 形状脚本”属性的 初始值”字段中的浏览图标来显示形状脚本编辑器。

功能区	设计>元素>特征>属性>[定义或选择属性'_image']>单击 初始值”字段中的 ... 。
上下文菜单	右键单击构造型元素|特征|属性 | <定义或选择属性'_image'> |单击 初始值”字段中的 ...
键盘快捷键	F9 | <定义或选择属性'_image'>] |单击 初始值”字段中的 ...

将形状脚本脚本添加到构造型元素

构造型元素现在类似于这个例子：



节	描述
1	在 名称”字段中，输入 _“image”。
2	单击 初始值”字段旁边的 ... 按钮。 将显示 形状编辑器”对话框。
3	在 形状编辑器”对话框中输入形状脚本。 写完形状脚本后，单击确定按钮，然后单击关闭按钮。

注记

- 你的形状脚本可能包含外部定义的图像；在这种情况下，形状脚本将包含 image 方法，指定以技术名称为前缀的图像文件名
- 如果您正在为关联类创建形状脚本，请注记该形状脚本适用于类部分和关联部分；因此，您可能必须在 shape main 中包含测试元素类型的逻辑，以便您可以为类和关联提供单独的绘图说明

  这种逻辑在以下情况下是不必要的：
  - 形状源或形状目标，它们被类忽略，或者
  - 装饰形状，被关联忽略
- 您还可以临时将形状脚本应用于元素，将形状脚本附加到“UML类型”对话框（ 设置 >参考> UML类型”）上定义的形状脚本型

设置默认外观

如果您想为构造型元素或连接器定义一个简单的默认外观，您可以选择定义它的构造型元素并设置任何或全部：

- 背景/填充颜色
- 边框颜色
- 边框线宽，或
- 字体颜色

要设置这些，请使用 默认外观”对话框。

访问

上下文菜单	元素右键|外观|默认外观
键盘快捷键	F4

注记

- 当您保存定义原型元素和连接器的配置文件时，选择 保存UML配置文件”对话框上的 颜色和外观”复选框

特殊属性

可以定义原型元素模型的许多特殊特征和行为，例如在浏览器窗口和图表工具箱中表示它的图标，与原型关联的任何图像文件的默认位置，图表中的元素，或者外观是否由形状脚本定义。您可以在您的配置文件中定义这些特征，使用可应用于以下任一项的特殊属性：

- 构造型元素或
- 元类元素，指的是扩展它们的原型

访问

功能区	设计>元素>特征>属性
上下文菜单	右键单击元素|特征|属性
键盘快捷键	F9

设置属性

字段/按钮	描述
名称	类型属性的名称（在这些库表中列出）。
最初的	类型或选择属性的初始值。
关	单击此按钮可关闭对话框。

构造型元素属性

属性	意义
_defaultAttributeType	定义从图表工具箱创建的新属性的默认类型。在扩展属性元类的构造型元素中使用this，并将 初始值"字段设置为所需的属性。 如果您不提供此选项，系统将创建默认类型为**int**的属性。
图标	包含在浏览器窗口中由构造型定义的所有元素旁边显示的 16x16 像素图标的位图文件位置。这不适用于包元素。该图标也自动用作图表工具箱图像，只要列出了原型元素。 对于透明背景，您可以使用浅灰色 - RGB (192,192,192)。 要使该属性正常工作，还要设置 _metatype 属性。
_图片	标识一个形状脚本脚本定义，在 初始值"字段中为其创建脚本。

	要使该属性生效，您需要在保存配置文件时设置 图像"选项。
_instanceMode	已弃用。
_instanceOwner	已弃用。
_instanceType	将对话框中 粘贴as"字段的第二个选项修改为： • 作为元素的实例(ProfileName::<<stereotype>>) <<stereotype>> 值在属性的 初始值"字段中定义，并对应于使用 _"metatype"属性赋予原型元素的元类型。
_元类型	将原型定义为元类型，以便隐藏元素作为自定义的原型元素的身份。
_scriptlet	在显示图表之前定义一个脚本来设置和自定义此原型的元素。
_sizeY	设置元素的初始高度，以像素为单位，在 100% 缩放。 要使此属性生效，您需要在保存配置文件时设置 无素大小"选项。注记：不能将元素设置为小于其默认最小高度。
_sizeX	设置元素的初始宽度，以像素为单位，在 100% 缩放。 要使此属性生效，您需要在保存配置文件时设置 无素大小"选项。注记：不能将元素设置为小于其默认最小宽度。
_严格	定义一个原型元素可以应用多个原型的程度。

元类元素属性

属性	意义
_AttInh	如果设置为1 ，则在每个新的定型元素模型上将 继承特征：显示属性"复选框设置为选中。
_AttPkg	如果设置为1 ，则在每个新的原型模型元素上将 属性可见性：包"复选框设置为选中。
_AttPri	如果设置为1 ，则在每个新的原型模型元素上将 属性可见性：私有"复选框设置为选中。
_AttPro	如果设置为1 ，则在每个新的原型模型元素上将 属性可见性：受保护"复选框设置为选中。
_AttPub	如果设置为1 ，则在每个新的原型模型元素上将 属性可见性：公共"复选框设置为选中。
组成种类	当应用于关联时，定义源端或目标端是聚合还是复合。允许的值为： • 没有任何 • 聚合源 • 聚合在目标

	• 复合源 • 复合目标
_ConInh	如果设置为1 ，则在每个新的定型模型元素上将 显示元素隔间：继承约束"复选框设置为选中。
_约束	如果设置为1 ，则在每个新的定型模型上将 显示元素隔间：元素约束"复选框设置为选中。
_dbtype	当与扩展数据库建模原型（例如<<EAUML::table>>）的原型一起使用时，将为新元素设置默认数据库语言。
_defaultDiagramType	定义元素组合时创建的子图的类型。
方向	当任何类型的连接器元类元素从 配置文件"工具箱页面拖到图表上时自动创建。您可以为此属性设置一个值，而不是使用**_SourceNavigability**或**_TargetNavigability**属性。
_gentype	设置非数据库元素的默认代码生成语言。
_HideMetaclassIcon	如果您要扩展一个将以矩形表示法显示的元素并且您不希望它在右上角显示 无类"图标，则设置为True 。影响需求、组件和使用案例等元素。
_HideStype	通过为每个新的刻板模型元素设置 隐藏刻板特征"过滤器，将 初始值"字段设置为以逗号分隔的刻板印象列表以隐藏这些刻板印象。
_HideUmlLinks	如果您使用元模型来创建您的快速链接器定义并且您想从您的原型源元素的快速链接器中排除UML快速链接器定义，则设置为True 。（否则， UML快速链接器定义将从基本的UML元类继承。）
_常见	如果您不希望在创建目标元素时在快速链接器中提供关系，请将其设置为True 。
_isVertical	为构造型 ActivityPartition 设置为True以使默认活动分区方向垂直。
_lineStyle	设置原型连接器的线型；属性的 初始值"可以是以下之一： • 直接的 • 汽车 • 风俗 • 贝塞尔曲线 • 树H（水平） • 树V（垂直） • treeLH（横向横向） • treeLV（横向垂直） • 正交S（正交、方角） • 正交R（正交、圆角）
_makeComposite	使每个原型元素在创建时成为复合元素。
_含义向后	从目标读到源时，关系A自然语言含义。例如，<<Flow>>关系可能将 _MeaningBackwards 设置为 "flows from"。用 可追溯性窗口和其他地方。

_意义转发	从源读到目标时关系A自然语言含义。例如，<<Flow>>关系可能将 _MeaningForwards 设置为 "Flows to"。用 可追溯性窗口和其他地方。
_OpInh	如果设置为1 ，则在每个新的定型模型元素上将 继承的特征：显示操作"复选框设置为选中。
_OpPkg	如果设置为1 ，则在每个新的原型模型上将 操作可见性：元素包"复选框设置为选中。
_OpPri	如果设置为1 ，则在每个新的定型模型元素上将 操作可见性：私有"复选框设置为选中。
_OpPro	如果设置为1 ，则在每个新的定型模型元素上将 操作可见性：受保护"复选框设置为选中。
_OpPub	如果设置为1 ，则在每个新的原型模型元素上将 "Operation Visibility: Public"复选框设置为选中。
_PType	如果设置为1 ，则将在每个新立体元素上选中的 显示元素类型（仅端口模型或部件）"复选框设置为。
_树脂	如果设置为1 ，则在每个新的定型模型元素上将 显示元素隔间：继承的职责"复选框设置为选中。
_责任	如果设置为1 ，则在每个新的原型模型上将 显示元素分隔：元素需求"复选框设置为选中。
_运行状态	如果设置为任何非空白值，则在每个新的定型模型元素上将 隐藏当前图中的物件运行状态"复选框设置为选中。 要显示运行状态，请省略此属性或给它一个空白值。
_SourceAggregation	已弃用。请参阅**compositionKind** 。
_SourceMultiplicity	设置源元素的多重性，例如1 ..* 或 0.. 1 。
_SourceNavigability	如果连接器不可导航，则将此属性设置为 "不可导航"。 如果其他值更合适，请使用**方向**属性。
_subtype 属性	指定每次从工具箱创建具有构造型的元素时用于生成弹出子菜单的标记值的完全限定名称。 标记值是一个枚举，子菜单由每个枚举文字的命令组成。使用子菜单中选择的任何命令初始化标记值；如果没有选择（例如，如果用户单击子菜单），则默认值将照常使用。 例如，如果您创建 BPMN 2活动元素，则会显示一个子菜单，其中列出了任务类型，例如 "BusinessRule"、 "Manual"和 "接收"。选择其中一个值将其设置为taskType标记值。 值实际上是活动标记值的子类型；在 BPMN 2 配置文件中，在格式 profile::stereotype::tag 中，活动原型的 subtypeProperty 将是： BPMN2.0::活动::taskType。
_标签	如果设置为1 ，则在每个新的原型模型元素 显示元素分隔：标签"复选框设置为选中。

_tagGroupings	将标记值映射到属性窗口的 标记”选项卡中显示的标记组中，格式为： 标记名1=组名1；标记名2=组名2； 此功能目前仅适用于object类型，不适用于属性等其他类型。
_tagGroups	定义所需组的逗号分隔列表，按照它们在属性窗口的 标签”选项卡中的显示顺序。例如： 组名1、组名2、组名3 此功能目前仅适用于object类型，不适用于属性等其他类型。
_tagGroupStates	将属性窗口 Tags”选项卡中显示的属性映射到打开或关闭状态，形式为： 组名1=打开；组名2=关闭； 此功能目前仅适用于object类型，不适用于属性等其他类型。
_TagInh	如果设置为1 ，则在每个新的原型模型元素上将 显示元素分隔：继承的标签”复选框设置为选中。
_TargetAggregation	已弃用。请参阅**compositionKind** 。
_TargetMultiplicity	设置目标元素的多重性，例如1 ..* 或 0.. 1 。
_TargetNavigability	如果连接器不可导航，请将此属性设置为不可导航。 如果其他值更合适，请使用**方向**属性。
_UCRect	（仅适用于具有不同矩形符号的元素类型，或具有评估 'rectanglenotation'属性的形状脚本的元素，其中可以包括通常不具有矩形符号的元素类型。 ） 如果设置为1 ，最初显示矩形表示法中的元素。如果设置为 0，则最初以标准符号显示元素。

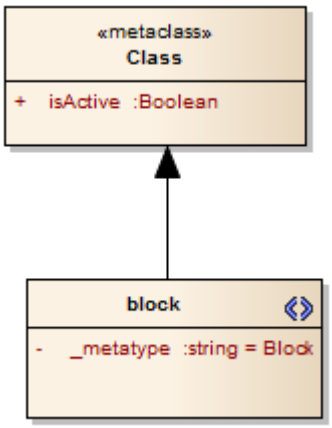
注记

- 如果属性设置为1以打开特征，则将其设置为 0 会关闭特征

将构造型定义为元类型

如果您想将自定义元素的身份隐藏为原型UML元素，您可以在定义它的构造型元素中设置 _metatype 特殊属性。 _metatype 属性还使自定义元素类型出现在只有Enterprise Architect的内置类型通常会出现的上下文中；例如，在关系矩阵的元素类型列表中。

在这个来自 SysML 的示例中，块被定义为扩展UML类的构造型元素。



但是，SysML 用户对UML类不感兴趣，只对 SysML 块感兴趣。如果您将 _metatype 属性设置为块，则从该构造型创建的任何元素，虽然在大多数情况下的行为方式与构造型类相同，但将：

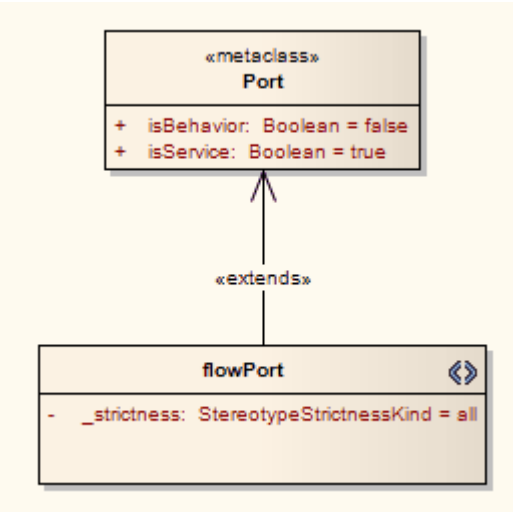
- 显示块<名称> 而不是类<名称> 作为其 属性"对话框的标题
- 在创建时自动编号为 Block1 而不是 Class1，并且
- 在整个Enterprise Architect的许多其他上下文中显示为块而不是类

定义多重刻板印象级别

一个元素可以有多个应用到它的刻板印象。您可以通过在定义构造型元素中创建 _strictness 特殊属性来定义可以应用多个构造型的级别。属性的类型是 StereotypeStrictnessKind，在 初始值"字段中具有四个值之一：

- profile，它指出不能从同一个配置文件中给这个类型的元素一个以上的构造型
- 技术，它指出这种类型的元素不能从同一技术中获得多个刻板印象
- all，它表明这种类型的元素根本不能有多个构造型，或者
- none，这是默认行为，并声明对使用多个构造型没有限制

此示例来自 SysML，并表明 «flowPort» 不能应用任何其他构造型。

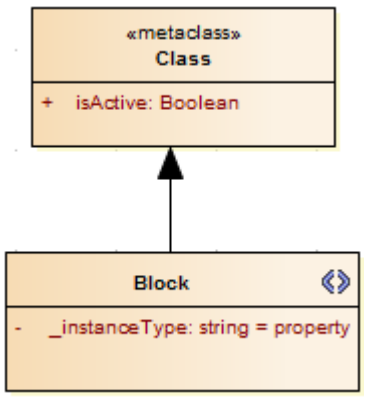


定义实例的创建

A原型元素可以是从它创建的实例的分类器。您可以通过向定义构造型添加特殊属性来定义如何从该构造型元素创建实例。属性修改 粘贴As"对话框上的文本，当一个原型元素被拖出浏览器窗口到图表上时，该对话框会显示。

属性

这个来自 SysML 的示例显示了可能创建的 SysML块元素的任何实例的定义。



当用户将 SysML块元素从浏览器窗口拖到图表上时，系统会检查 _instanceType 属性值并在 SysML配置文件中搜索具有匹配 _metatype 属性值的元素模板，并从中生成实例。通过示例定义，您将获得具有 属性"构造型的块元素。

属性	意义
_instanceMode	已弃用 将对话框中 粘贴as"字段的第二个选项修改为： • 实例（<元素类型>）或 • 属性（物件） 文本由属性的 初始值"字段的值（ 实例"或 属性"）确定。 如果未应用该属性，则该选项默认为 实例"。
_instanceOwner	已弃用 将对话框中 粘贴as"字段的第二个选项修改为： • 作为 <元素> 的实例 文本由属性的 初始值"字段的值确定，例如 块"。 如果未应用该属性，则该选项默认为 元素"。
_instanceType	将 粘贴as"对话框中 粘贴as"字段的第二个选项修改为： • 作为元素的实例(ProfileName::<<stereotype>>) <<stereotype>> 值在属性的 初始值"字段中定义，并对应于使用 "metatype"属性赋予原型元素的元类型。 注记您可以使用 _instanceType 属性或元约束来定义实例的创建。区别在于： 1. 在 粘贴As"对话框中，元约束允许您定义多个实例原型，而 _instanceType 则没有。多个实例都列出来了；这是一个非常有用的特征。 2. 元约束（当前）对 _instanceType 所做的 转换为实例"命令没有任何影

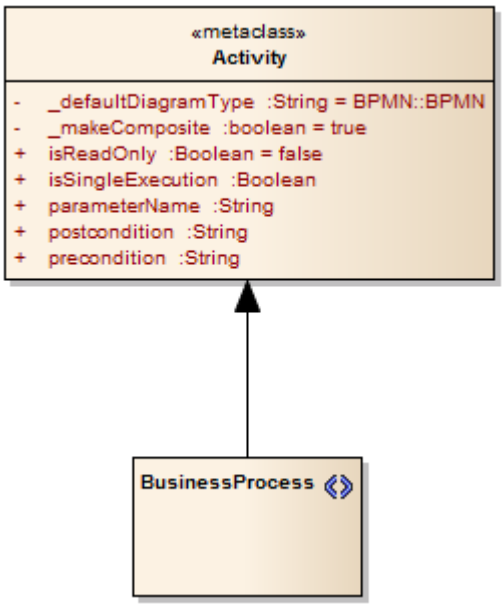
	响。

定义复合元素

原型元素A自动创建为复合元素。您可以使用特殊属性定义这一点，以及组合的子图是否属于特定类型。

要定义元素是否总是在创建时复合，您将 _makeComposite 特殊属性应用于适当的元类元素（而不是原型元素）。构造A类在创建时默认没有子图，因此您使用 _makeComposite 属性来触发子图的创建。对于原型组合，子图是元类通常的默认图类型；您可以使用 _defaultDiagramType 特殊属性更改子图表类型，以识别首选图表类型，

来自 BPMN 的这个示例表明，BusinessProcess元素始终创建为具有 BPMN 自定义子图的复合元素。

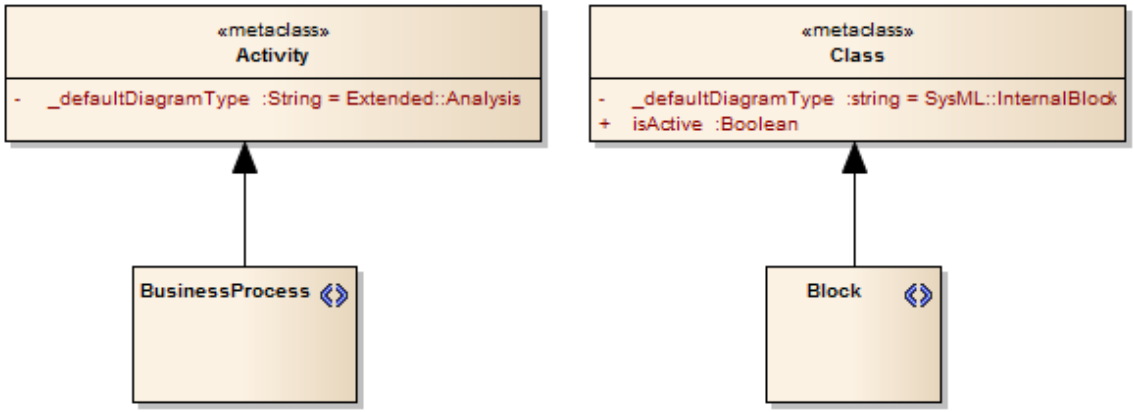


定义子图表类型

如果您将构造型元素类型定义为组合，则其子图类型最初与您扩展的元类元素的默认类型相同。您可以使用 _defaultDiagramType 特殊属性将图表类型更改为任何其他内置UML或扩展类型，或任何您自己的自定义图表类型。由于图类型默认来自 Metaclass元素，因此您可以在该 Metaclass元素（而不是构造型元素）上设置属性以更改默认值。

您在属性的 初始值"字段中标识子图表类型。内置UML和扩展图类型的实际值列在*Initial Values*部分中。如果要设置自定义图表类型，请在图表类型名称前加上图表配置文件名称和 ::"。图表配置文件名称是您保存配置文件时为其指定的名称，默认为配置文件包或配置文件图表的名称。我们建议图表配置文件名称基于技术名称。您还可以使用包的 _defaultDiagramType 属性，扩展包元类元素。

这些示例展示了一个 «BusinessProcess»活动，当制作一个复合元素时，它会自动创建一个分析图，以及一个创建 SysML InternalBlock 自定义图的 «block» 构造型。



初始值

这些字符串可用于 _defaultDiagramType 的 初始值"字段，以识别内置UML和扩展图类型：

- UML行为::用例
- UML行为::活动
- UML行为：状态机
- UML行为::通讯
- UML行为::序列
- UML行为::时间
- UML行为：：交互
- UML结构::包
- UML结构::类
- UML结构::物件
- UML Structural::复合结构
- UML Structural::部件
- UML结构::部署
- 扩展::自定义
- 扩展::需求
- 扩展::维护
- 扩展::分析
- 扩展::用户接口

- 扩展::数据建模
- 扩展::模型文档。

注记

- 尽管我们建议自定义图表类型的图表配置文件名称基于技术名称，但属性前缀并不是对技术名称的直接引用

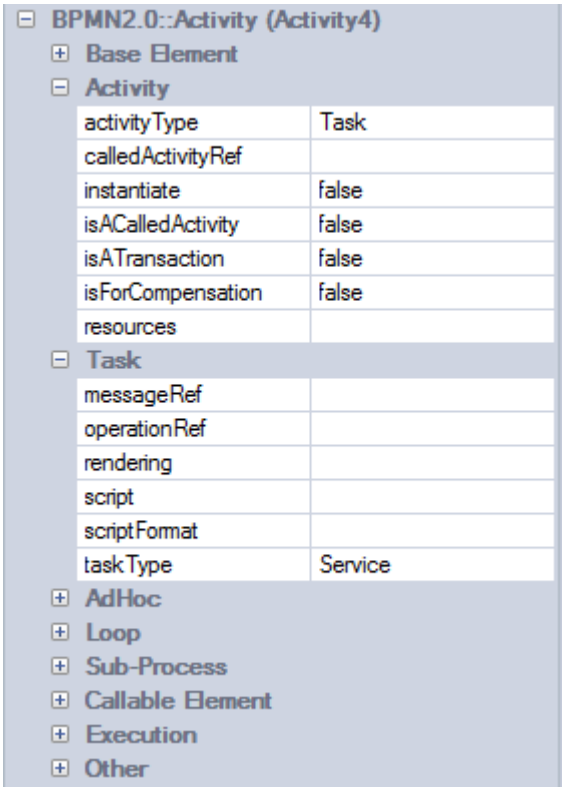
定义标签分组

在配置文件中开发一个定型元素时，可能会定义大量的标记值。例如，BPMN 2.0配置文件中的一个BPMN活动元素有30个标记值。默认情况下，在元素的属性窗口的 "Tags"选项卡中，这些标记值最初都将按字母顺序显示，如果它们恰好具有按字母顺序遥远的名称，则可能会拆分相关标签。为了将相关标签保持在一起并控制最初显示哪些标签，在 BPMN 2.0配置文件中，标记值已被分组。您可以应用相同的解决方案，使用元类中的三个标记分组特殊属性，这些属性由构造型元素元素，其中标记被定义为属性。

您可以使用以下方式应用分组：

- _tagGroups 定义组名
- _tagGroupings 定义哪些标签进入每个组
- _tagGroupStates 定义哪些标记组最初在属性窗口的 标记"选项卡中展开，哪些标记组被折叠

BPMN 2.0活动元素属性窗口的 标签"选项卡最初显示如下：



活动元类属性

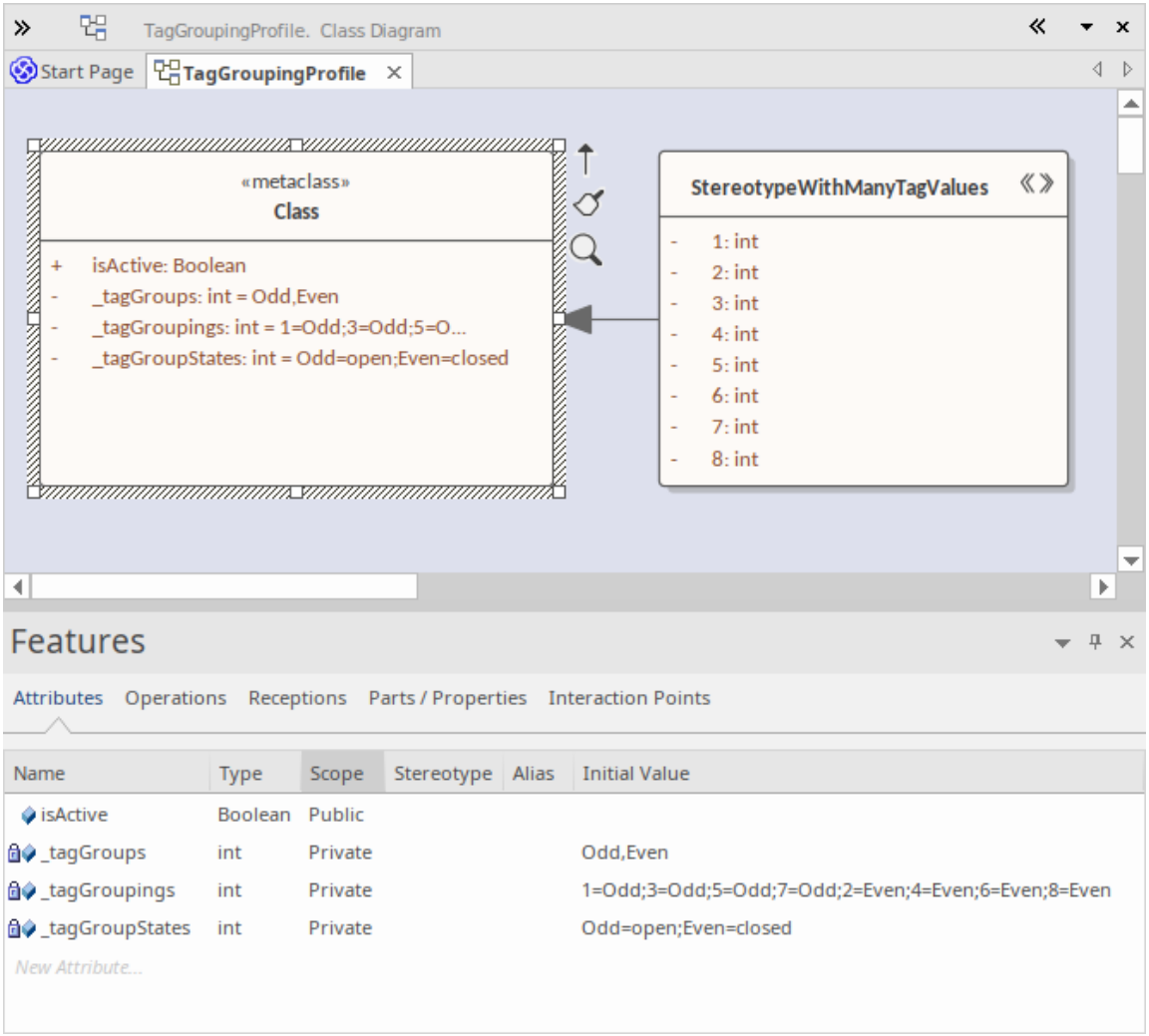
为了实现 BPMN 2.0活动标记值的显示，技术开发人员在活动元类元素中定义了特殊属性，如下所示：

属性	价值观
_tagGroups	Base元素,活动,任务,AdHoc,Loop,Sub-Process,Callable元素,其它,其他
_tagGroupings	auditing=基本元素；categoryValue=基本元素；documentation=基本元素；monitoring=基本元素；activityType=Activity;calledActivityRef=Activity;instantiate=Activity;isACalledActivity=Activity;isATransaction=Activity;isForCompensation=Activity;resources=Activity;messageRef=任务；操作引用=任务；渲染=任务；脚本=任务；scriptFormat=任务；taskType=任务；adHoc=AdHoc；adHocOrdering=AdHoc；...（等等）

_tagGroupStates	基本元素=关闭；活动=打开；任务=打开；AdHoc=关闭；循环=关闭；子流程=关闭；可调用元素=关闭；执行=关闭；其他=关闭

示例

此处显示的是如何使用标签分组属性的简单示例。



注记

- 此功能目前仅适用于object类型，不适用于属性等其他类型

介绍元模型视图

Enterprise Architect包括一个非常有效和灵活的系统定义和用户定义元模型的视图系统。视图系统提供高度集中的图表，将可用的元素和连接数量限制为仅用于完成特定任务所需的核心。例如，类图上的层次结构视图可能会将唯一可用的元素限制为 类"，并将唯一的连接器限制为 继承"。

使用视图系统来指导建模调色板和可用关系，您将构建紧凑且有目的的图表，这些图表仅使用当前建模上下文中所需的元素。消除噪音和减少可用构造集是确保设计解决预期目的并避免可能对模型的可读性和正确性产生负面影响的无关元素的好方法。

元模型视图

类别	描述
系统	Enterprise Architect提供多种内置视图，可满足众多建模场景和领域的需求。许多模型模式都预设了视图，而 图表生成器"对话框包含许多派生图表视图，可扩展和完善基本图表类型的功能。
风俗	除了在Enterprise Architect中使用系统定义的基于元模型的视图外，还可以创建自己的元模型并轻松地将它们添加到当前模型中，然后您和其他建模者可以根据需要将它们应用于各种图表。例如，您可以定义一个特定的元模型集来满足您组织中需求建模的需求，然后要求所有需求图都使用该元模型视图。

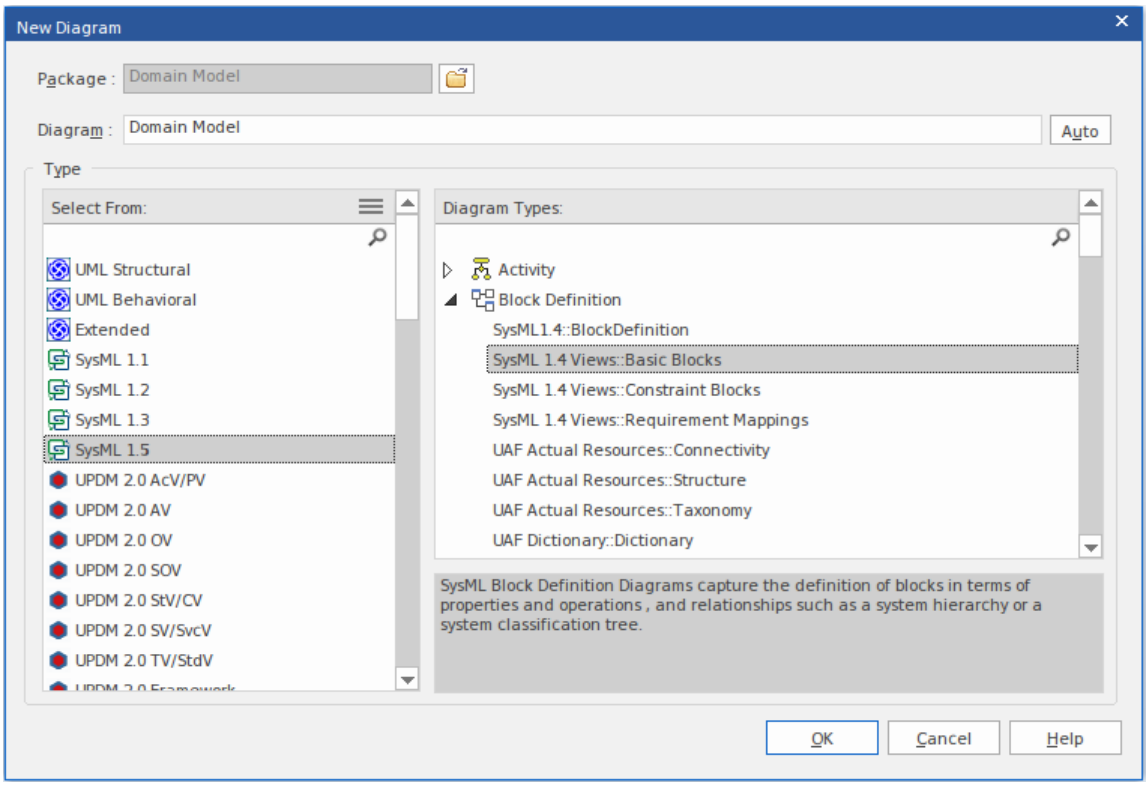
视图系统功能

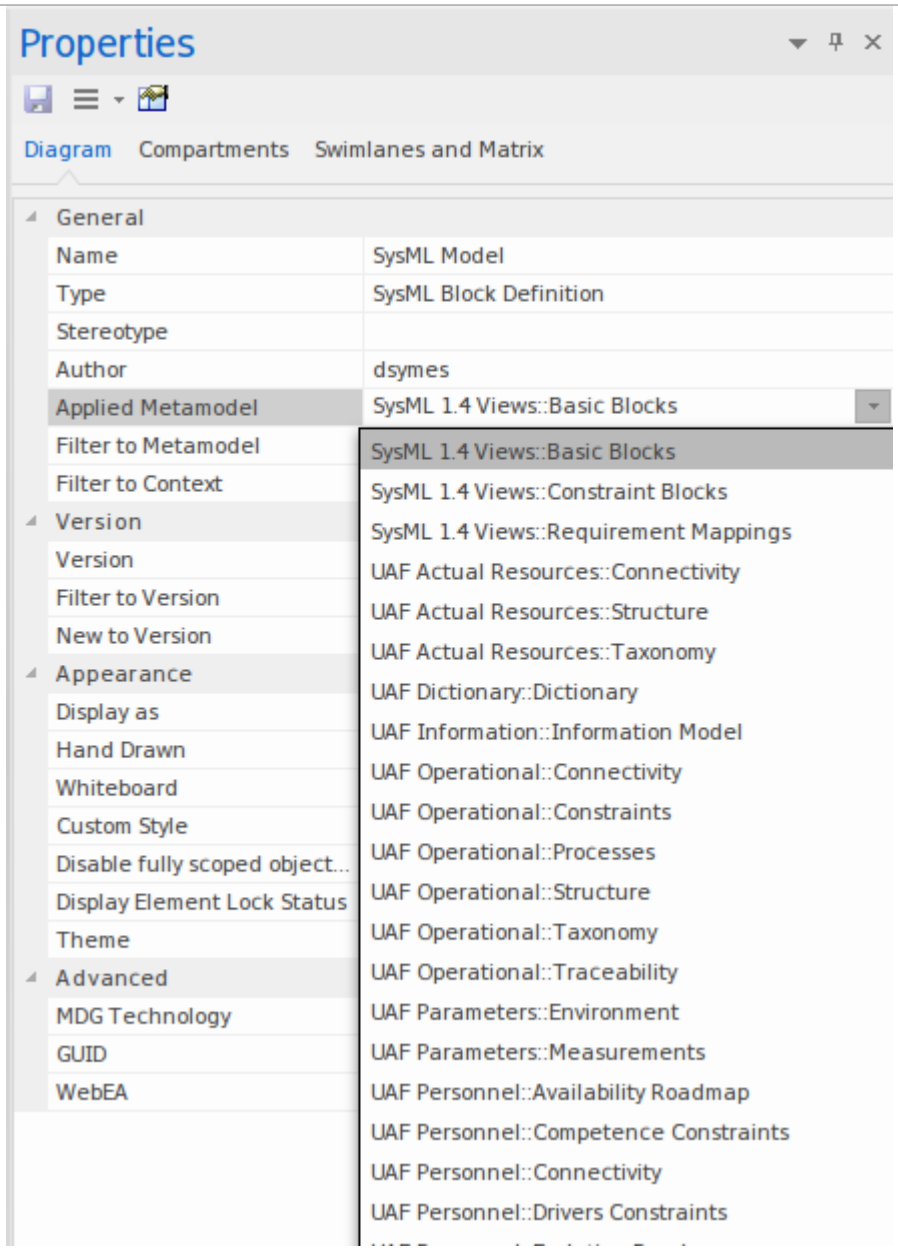
功能	描述
图表过滤器	除了限制可用的调色板外，视图系统还允许建模者启用图表过滤器，该过滤器将使不属于当前视图集的任何元素变灰。这允许建模者更正其模型中不符合所选视图目的的任何部分，或者过滤掉需要存在但不构成当前建模目标一部分的元素。
图表属性	图表的 属性"对话框包括当前所选图表类型的可用视图的下拉列表。选择其中一个视图将减少可用构造的调色板并限制快速链接器中的条目。建模者可以轻松激活视图，甚至在必要时移除视图 - 实际模型内容不会改变。
图表视图	图表构建器"对话框包括许多不同的视图，这些视图为UML 、SysML、BPMN 和 UAF 等图表类型提供不同的调色板集和聚焦目标。如果您的目标是对没有高级特征的简单活动图进行建模，那么UML活动图部分下的简单活动视图可能是比使用完整活动图集更好的选择。

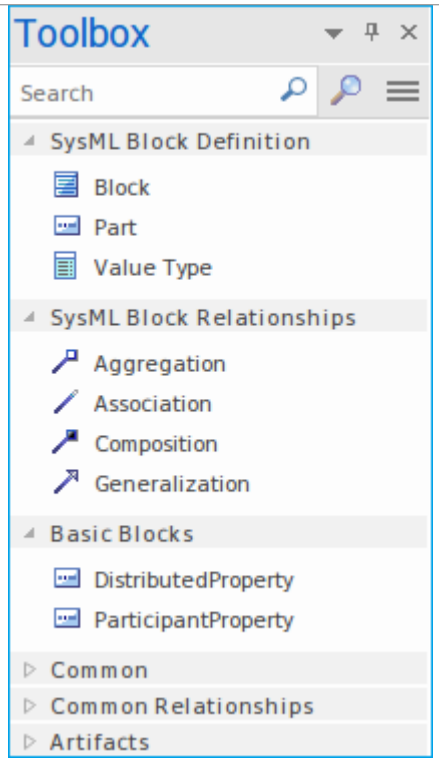
内置元模型图表视图

新图表”对话框包括许多不同的视图，为UML 、SysML、BPMN 和 UAF 等图表类型提供不同的调色板集和聚焦目标。例如，如果您的目标是对没有高级特征的简单 SysML块定义图进行建模，则 图表1 .5块定义图”部分下的 基本块视图”可能比使用完整块更好的选择定义图集。此示例用于在本主题的过程中提供值。

使用图表视图

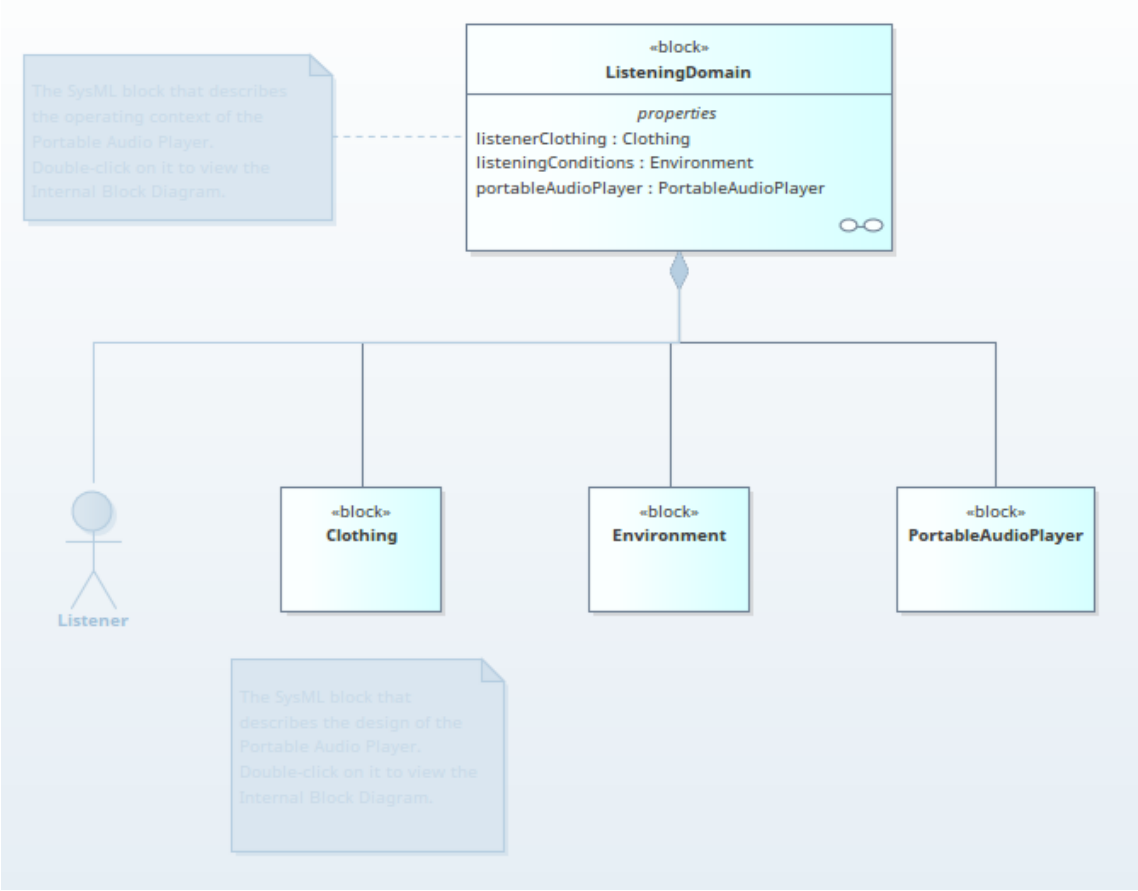
节	行动
1	在浏览器窗口中，单击要放置图表的包或元素。 打开 新图表”对话框，选择 SysML 1 .4视图:: Basic Blocks”并点击确定按钮来创建图表。 
2	在创建图表的属性窗口中， 应用元模型”字段将显示应用的图表视图。您还可以单击此字段中的下拉箭头，然后从列表中选择另一个可用的图表视图。

	
3	在图表工具箱中，与图表视图关联的受限元素和关系集将是可见的。



更改 应用元模型”选项列表中的图表视图将更改工具箱中的元素和关系。

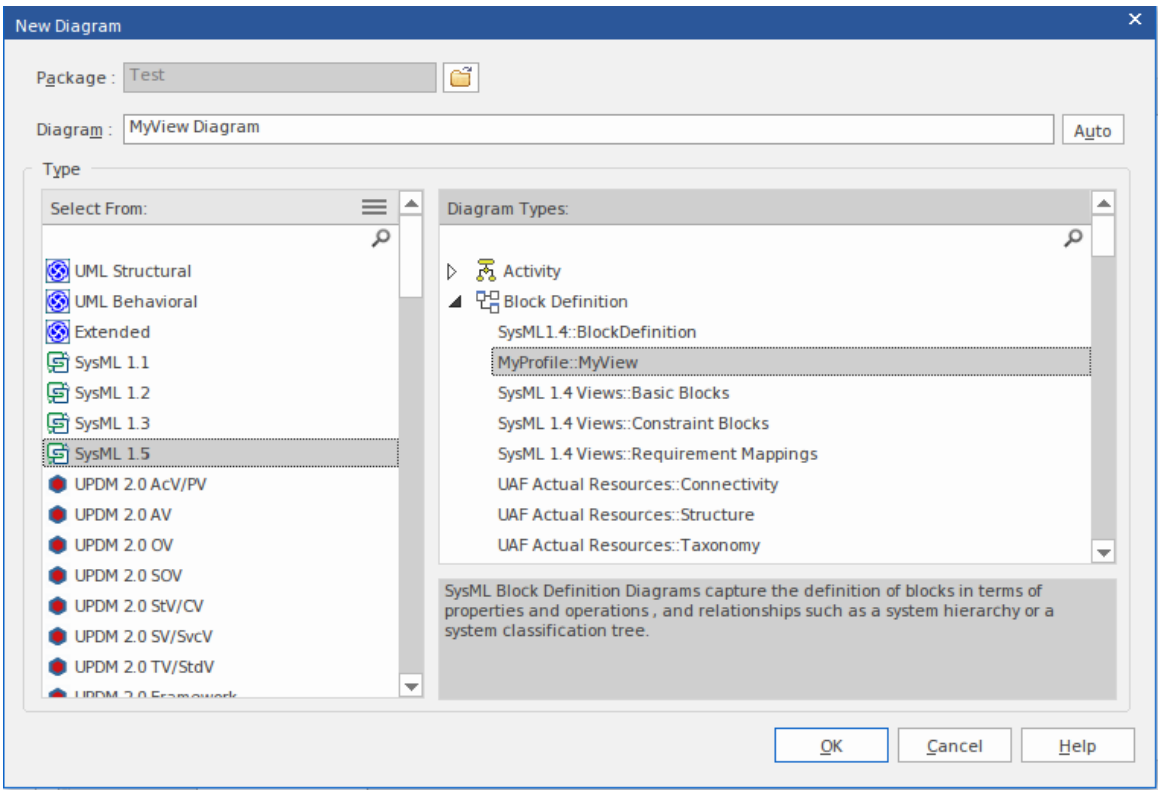
4 在属性窗口中选择 对滤器到元模型”选项将使不属于当前图表视图集的任何元素变灰。这使您可以更正模型中不符合所选视图目的的任何部分，或过滤掉可能需要存在但不构成当前建模目标一部分的元素。



自定义元模型图表视图

Enterprise Architect具有广泛的内置图表视图，但您也可以创建自己的元模型来定义自定义图表视图。例如，您可以定义一个特定的元模型来满足您组织中需求建模的需求，然后要求所有需求图表都使用该图表视图而不是内置的需求图表视图。您可以快速将图表视图添加到当前模型，您或其他建模者可以将它们应用于您的图表。

作为说明，假设您决定在您的项目中提供一个新的 SysML 1 .4块定义图视图，称为 "MyView"。用户将通过 "New块图表类型。

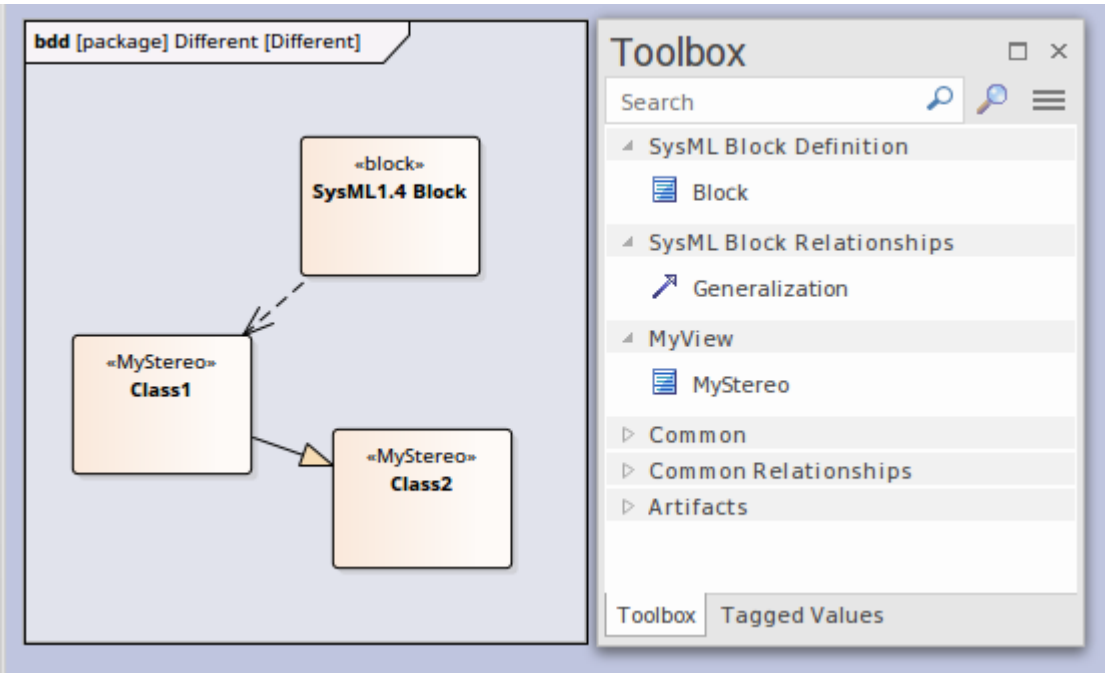


图表视图的完全扩展名称反映了父配置文件名称 (MyProfile) 和视图名称 (MyView) - 因此是 "MyProfile::MyView"。您可以调用示例视图1 1视图的视图。

如果您使用配置文件名称 "UML "扩展UML基本图类型，则等效视图名称可能是诸如 "UML ::视图"之类的名称。

用户选择示例图视图来创建一个非常简单的 SysML 1 .4块图，它可以具有：

- 两种元素：
  - SysML 1 .4块元素（ SysML 1 .4 技术的扩展类 ）
  - 您在新元模型 "MyView"中定义的 MyStereo元素
  具有原型 MyStereo 的类
- 一种标准的概括块连接器（ 与标准UML概括相同 ）



图表视图使元素和连接器可从工具箱（如图所示）和快速链接器中使用。

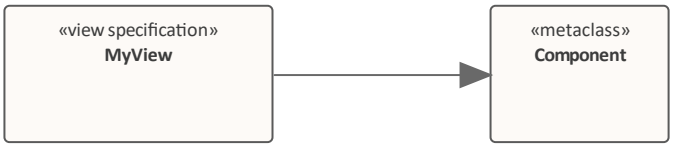
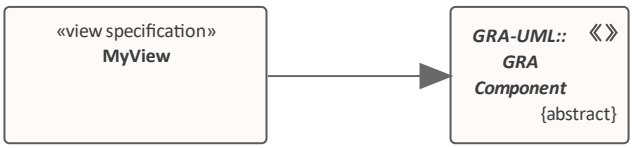
配置文件中的表配置文件自定义图表视图解释了如何创建定义新图表视图的元模型，最后以 MyView 示例结束。

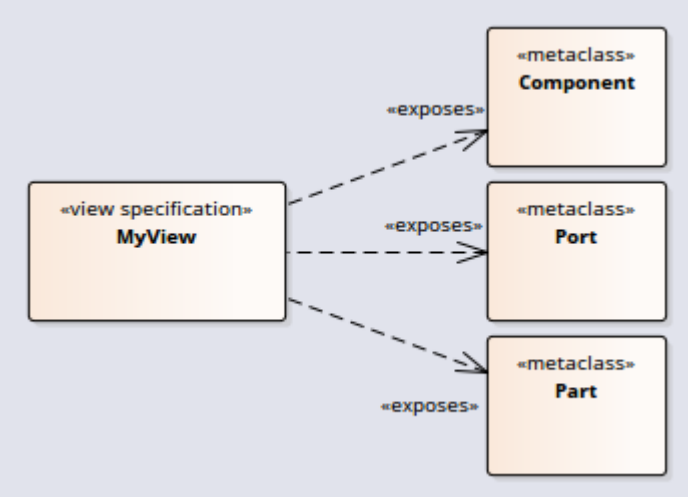
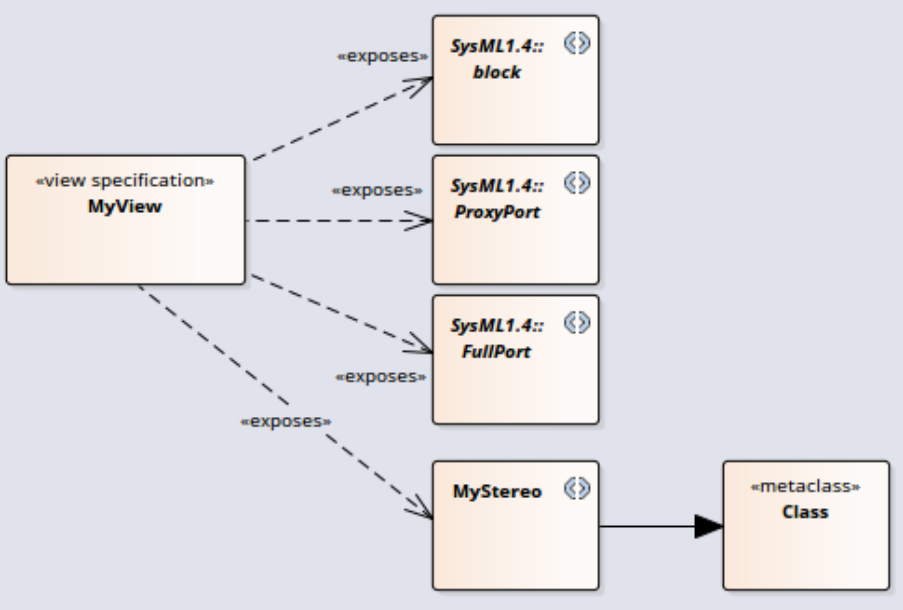
访问

功能区	设计>图表>工具箱： >配置文件>元模型
键盘快捷键	Ctrl+Shift+3： >配置文件>元模型

在配置文件中创建自定义图表视图

手术	行动
创建配置文件图	在您的配置文件包中，创建一个新包图，然后在图表工具箱中打开 配置文件”页面（选择 设计>图表>工具箱”功能区选项，然后单击 并选择 配置文件”）。 将 配置文件”图标拖到图表上并将其命名为 MyProfile”，选择添加名称为“MyView”的子类图表，然后打开它。 展开工具箱中的 无模型”页面并注记： • '视图'元素，您可以使用它来创建自定义视图 • '工具箱' 连接器，用于指定与自定义图表关联的工具箱页面的视图
添加视图规格	在配置文件中，您使用 视图规范”原型元素将新的自定义图表视图为现有内置或原型图的扩展。 将 视图规范”图标拖到配置文件图上，并为元素命名；在我们的示例中，

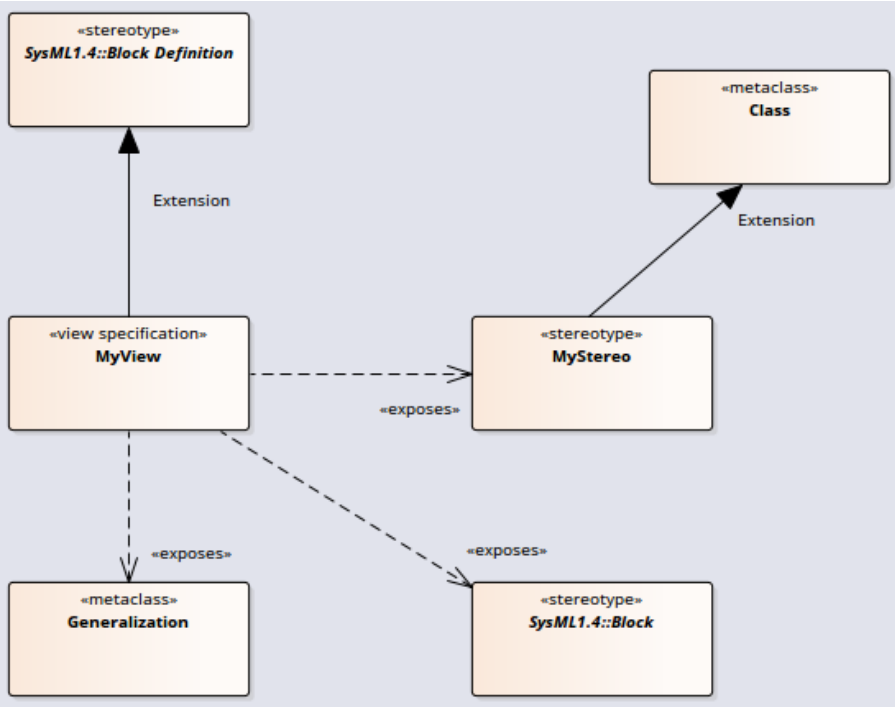
	“MyView”。 定义新视图时首先要考虑的是它应该适用于哪种图表类型。接下来的两行显示了如何为UML图和配置文件图定义视图。 在这两种情况下，单击 扩展”图标并从视图规范拖动到图表类型元素，以创建扩展连接器。
扩展UML图表类型	要扩展基本的UML图类型，将 类”图标从工具箱拖到图上，然后在属性窗口中，给出元素： • 图表类型的确切名称（在*Built-in*图表帮助主题中列出），例如“Logical”（用于类图），以及 • 刻板印象<<元类>> 此示例显示之前创建的 MyView”，扩展了UML部件图。  结果是在 新图表”对话框中，在UML部件图表类型下添加了一个额外的视图。
扩展 Profiled图表类型	要扩展分析图表类型，例如 BPMN 或 SysML 图表类型，请将 构造型”图标拖到图表上，并为构造型元素提供图表类型的完全限定名称。 因为这是对外部原型的引用，所以它也应该被标记为 Abstract 以防止它被导出到配置文件中。为此，显示属性窗口，展开 高级”部分并选中 摘要”复选框。 此示例显示之前创建的 MyView”，扩展了 GRA-UML部件图表类型。  结果是 新建图表”对话框将显示我们在 GRA-UML 组件图下定义的视图。 注记：如果您不知道要扩展的图表类型的完全限定名称，请查询 API 以获取“Metatype”字段。在JavaScript控制台中，您可以使用： ？获取当前图表()。元类型 或者，在浏览器中选择图表，然后在停靠的属性窗口中查看，它将在MDG 技术下列出。
在图表视图视图工具箱中工具箱对象	工具箱连接器将object添加到图表视图的工具箱页面。对于要添加到图表视图工具箱页面的每个元素和连接器，您将 定义元素”拖到图表上，然后单击工具箱 配置文件”页面中的 公开”图标并将光标从将视图规范元素添加到 定义元素”以创建连接器。 定义元素的类型取决于您是公开基本UML元素还是原型元素，如下两行所示。
暴露UML元素类型	如果您在自定义图表视图中使用基本UML元素或连接器，那么对于每个元素或连接器： 1. 将工具箱 配置文件”页面中的 无类”图标拖到图表上，并为其提供其代表的基本元素或连接器类型的名称，然后 2. 在视图规范元素和元类元素之间添加视图连接器 例如：

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>暴露已分析的元素类型</td><td>如果您在图表视图中定义一个新的原型object ，或者使用已经在其他配置文件中定义的原型元素，那么对于每个元素或连接器：
1. 将 构造型"图标从工具箱 配置文件"页面拖到图表上，并为元素指定其代表的构造型元素或连接器的名称
2. 如果构造型在另一个配置文件中定义，展开属性窗口的 高级"部分并选中 抽象"复选框
3. 如果在这里定义了构造构造型，则将构造型扩展的基本元素添加到图中，并在构造型和基本元素之间创建扩展连接器
4. 在视图规范元素和构造型元素之间添加视图连接器
例如：
 
</td></tr>
<tr><td>完成示例</td><td>参考表中前面的行，在 MyView类图（MyProfile 图的子图）上：
1. 创建元素规范视图。
2. 创建构造型元素SysML1.4::块并将其设置为Abstract。
3. 使用扩展连接器将视图规范连接到视图::块。
4. 创建一个元素的元概括。
5. 创建一个名为块::block 的构造型元素并将其设置为 Abstract。
6. 创建一个名为 MyStereo 的构造型元素和一个名为UML类的元类元素，并</td></tr>
</table>

使用扩展连接器将构造型连接到元类。

7. 将元素视图规范元素连接到元素、概括::块元素和元素，每个元素都有一个 Exposes 连接器。

此插图代表您创建的图表：



当您完成您的图表视图时，您可能会决定应该使用特定类型的连接器将一种类型的元素连接到相同类型或其他类型的元素。您可以使用元关系连接器来定义这一点，如定义元模型约束帮助主题中所述。

保存视图规格图。您现在可以将其作为其父配置文件的一部分添加到MDG 技术文件中；您将父配置文件添加到 "MDG 技术向导 -配置文件文件选择"页面。请参阅添加配置文件帮助主题。

定义元模型约束

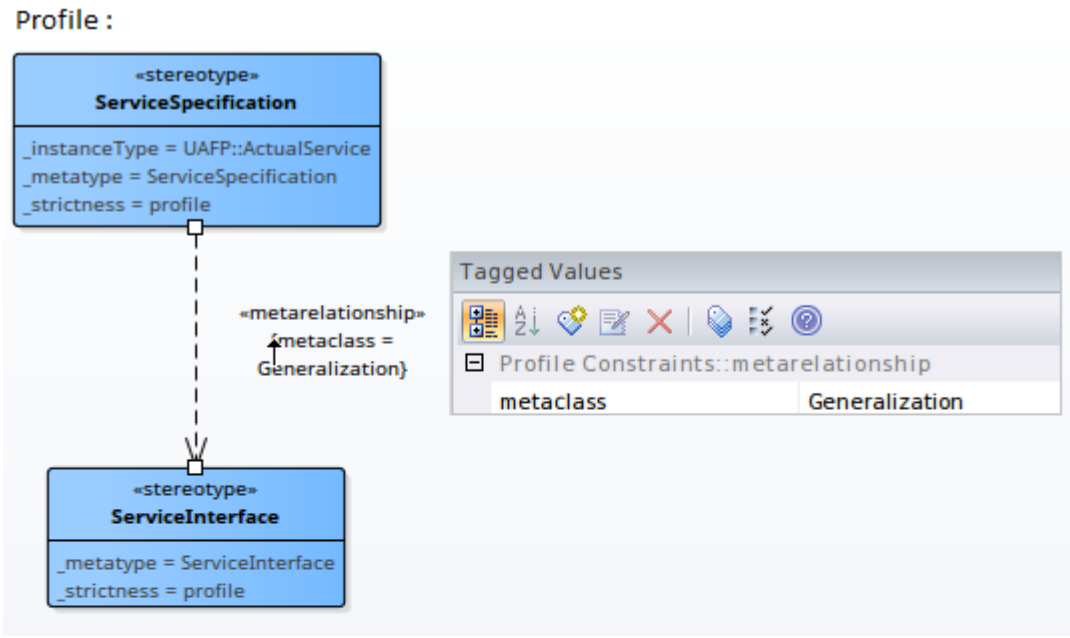
当扩展UML以开发特定于域的配置文件时， Enterprise Architect允许您指定约束以限制可以从构造型绘制的连接器，使用快速链接器或工具箱。这些约束是使用 配置文件"工具箱的 无模型"页面下的关系定义的。

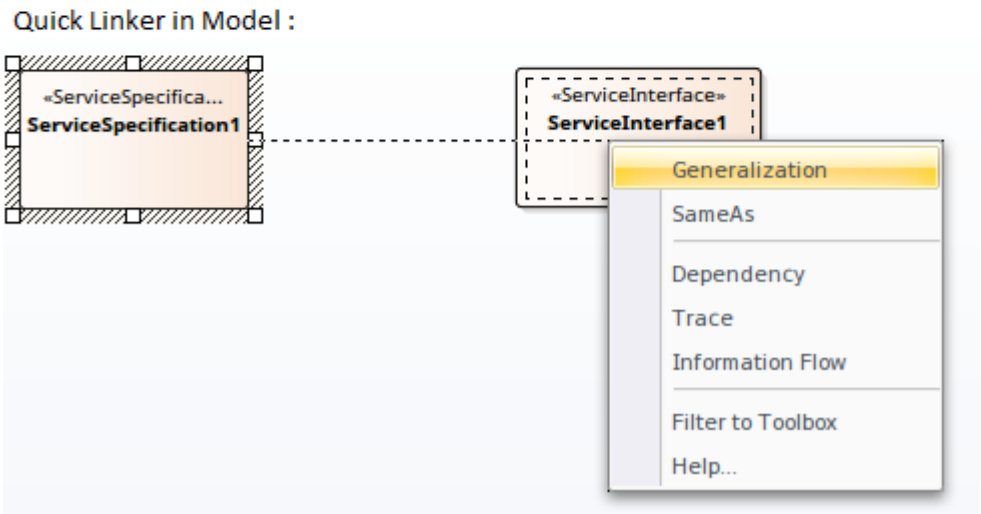
访问

功能区	设计>图表>工具箱： >配置文件
键盘快捷键	Ctrl+Shift+3

将元模型约束添加到配置文件

物品	细节
元关系	两个构造型之间A 无关系"连接器用 指定这两个构造型之间的有效UML连接器。 UML连接器的名称应该设置在«metarelationship» 连接器上的标记 无类"中。

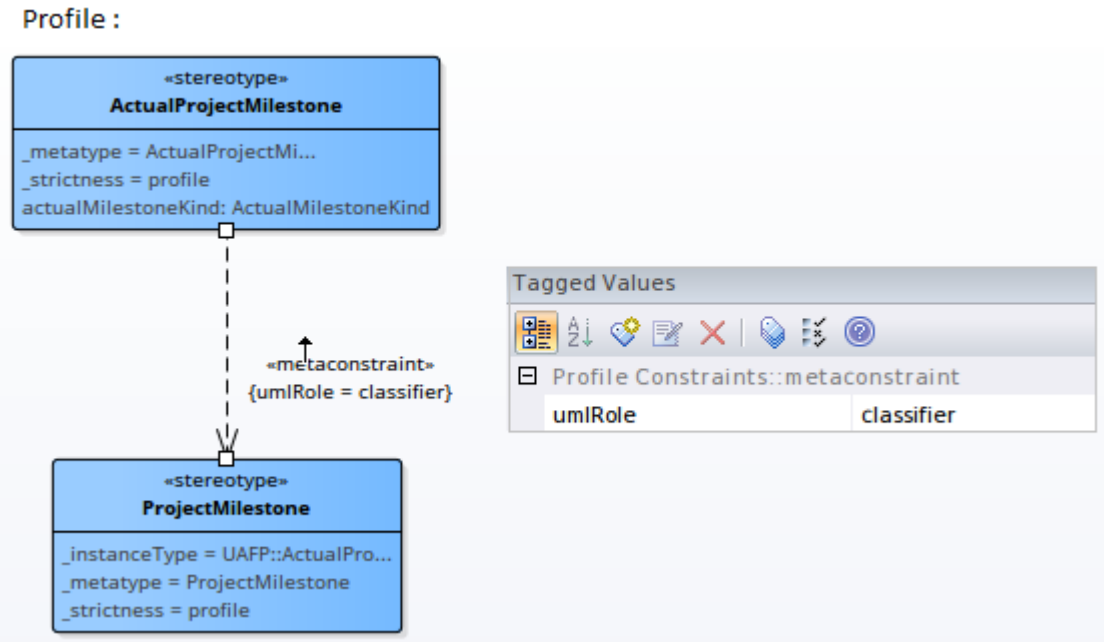




在配置文件示例中，从 ServiceSpecification 到 ServiceInterface 绘制了一个 «metarelationship» 连接器，并且UML连接器的名称在连接器的属性窗口的 "Tags"选项卡中指定。

将此配置文件导入模型后， Enterprise Architect将在使用 Quick Linker 绘制 ServiceSpecification 和 ServiceInterface 之间的关系时显示UML连接器。

元约束	两个构造型之间A «metaconstraint» 连接器用于指定这两个构造型之间的约束。 应在 Meta-Constraint 连接器上的标记 "umlRole"中设置约束。



在配置文件示例中，从 ActualProjectMilestone 到 ProjectMilestone 绘制了一个 «metaconstraint» 连接器，并且在连接器的标记值中的标记 "umlRole" 上将约束指定为分类器。

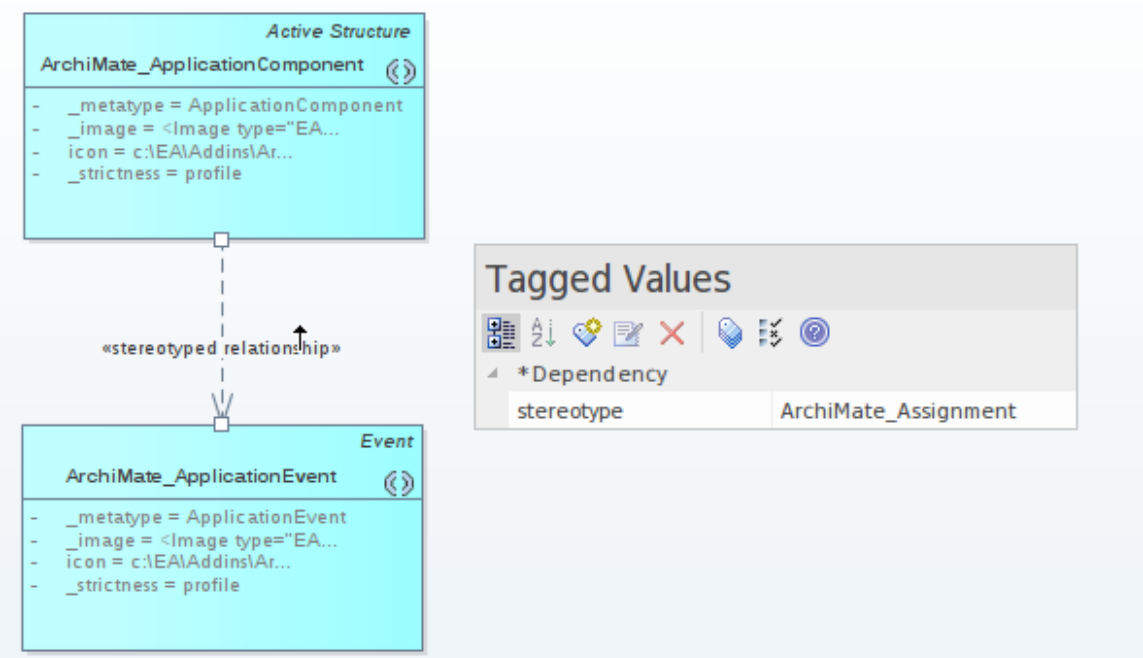
将此配置文件导入模型后， Enterprise Architect将在为 ActualProjectMilestone元素分配分类器时仅显示 ProjectMilestone 原型元素。

标签 "umlRole"的约束值包括：

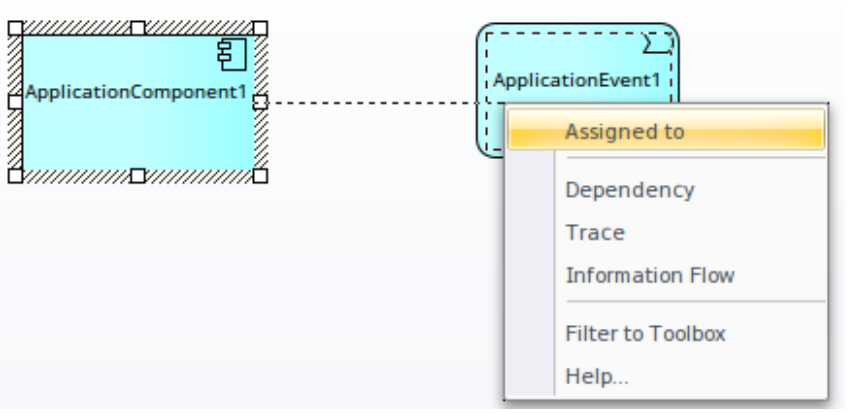
- 分类器——将源构造型元素的分类器限制为目标构造型元素
- type – 将源构造型元素的类型限制为目标构造型元素
- 行为 - 将源构造型元素的行为限制为目标构造型元素
- 传递 - 将源构造型元素的传递元素限制为目标构造型元素
- slot - 将源构造型元素的槽限制为目标构造型元素
- client/源元素- 将连接器的源限制为目标构造型
- provider/target/end[1].role/informationTarget - 将连接器的目标限制为目标构造型元素
- implementationConnector/realizingActivityEdge/realizingMessage - 限制可以实现信息流的关系
- typedElement/instanceSpecification – 当作为分类器从浏览器窗口中删除时，此约束将类型限制为目标构造型元素
- owner/class/activity/owningInstance – 将这个元素的容器限制为目标构造型元素；此约束用于创建嵌入式元素，并为快速链接器验证嵌套验证模型验证
- ownElement/ownedAttribute/ownedOperation/ownedParameter/ownedPort——限制源构造型元素可以拥有的元素/属性/操作/参数/端口；此约束通常用于验证嵌套模型验证
- annotatedElement/constrainedElement – 将注记连接器的目标限制为目标构造型元素

刻板关系	您可以在两个构造型或元类之间使用 "stereotyped relationship"连接器来指定这些元素的实例之间的有效构造型连接器。 指定关系时，如果所引用的关系是在定义规则的配置文件中定义的，则可以将属性型属性设置为仅该构造型的名称。但是，如果关系是在另一个概要文件中定义的，则必须使用与定义构造型的位置相对应的完全限定构造型名称。

Profile :



Quick Linker in Model :



在配置文件示例中，从 ApplicationComponent 到 ApplicationEvent 绘制了一个«stereotyped relationship»连接器，并且在连接器的标记值中将关系的构造型设置为 Assignment"。

将此配置文件导入模型后， Enterprise Architect将在使用快速链接器绘制 ApplicationComponent 和 ApplicationEvent 之间的关系时显示 巴分配"选项。

特殊元类

您可以将连接器的源指定为所有特殊形式的超类，并将目标指定为特殊元类，该元类在使用时指定与实际元类的关系。您可以使用这些术语之一作为具有构造型 «metaclass» 的类元素的元素名称。

物品	细节
源 .元	目标元素必须与源中定义的精确原型相匹配。

类型	
源.metatype.一般	目标元素可以匹配源中使用的确切构造型，以及任何具体的（isAbstract=false）广义构造型。
源.metatype.specific	目标元素可以匹配源中使用的确切原型，以及任何具体的 (isAbstract=false) 专门的原型。
源.metatype.both	目标元素可以匹配源中使用的确切刻板印象，以及任何具体的（isAbstract=false）概括或专门的刻板印象。
<profile_name>::*	将“profile_name>”替换为配置文件的名称；这将扩展为给定配置文件中所有具体原型的列表。
<none>	当你想防止源元素从它的超类型继承指定的连接器时使用这个元类名称。

元约束连接器上的约束

在创建特定领域的配置文件时， Enterprise Architect允许您指定相关构造型之间的约束。例如，您可以限制可以在 Stereotyped元素上设置为分类器的元素。

在 配置文件"工具箱的 无模型"页面上，两个构造型之间A元约束连接器用于指定两个构造型之间的约束。应在 Meta-Constraint 连接器上的标记 ű mlRole"中设置约束。

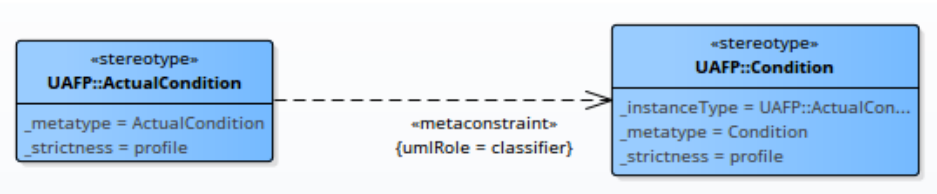
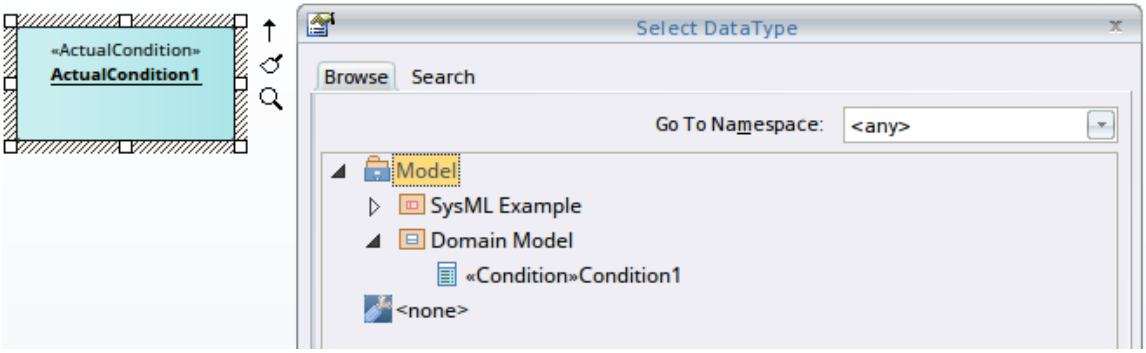
访问

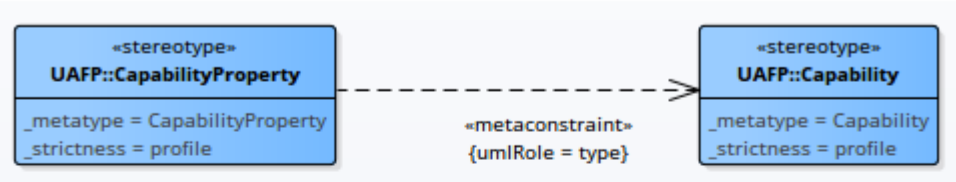
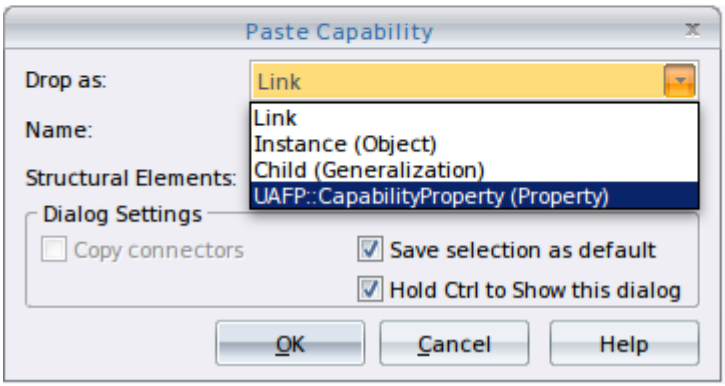
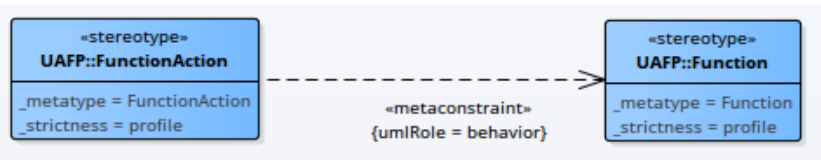
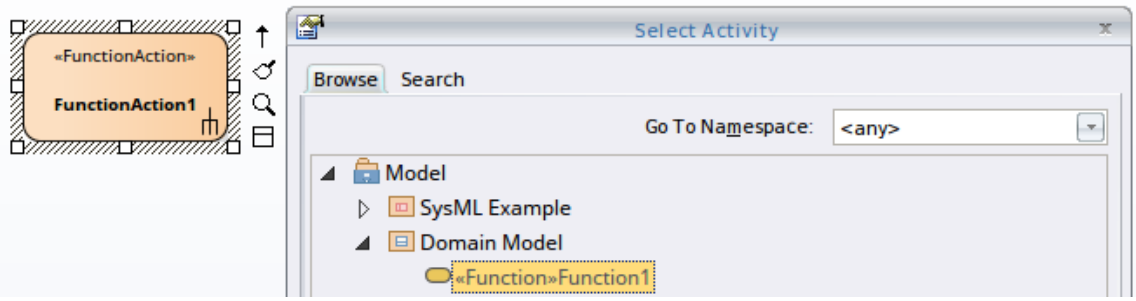
功能区	设计>图表>工具箱： >配置文件>元模型
键盘快捷键	Ctrl+Shift+3： >配置文件>元模型

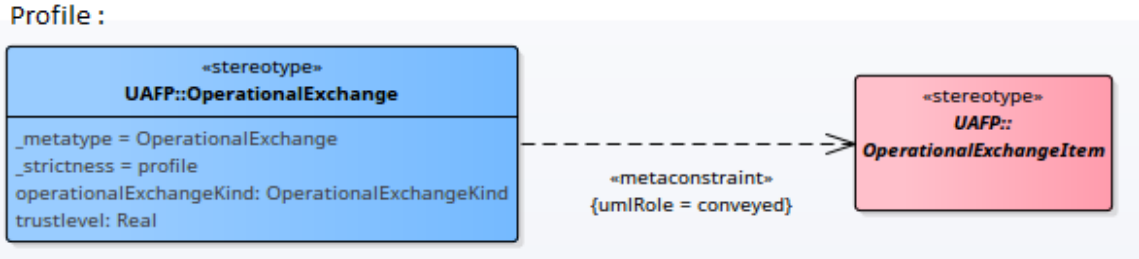
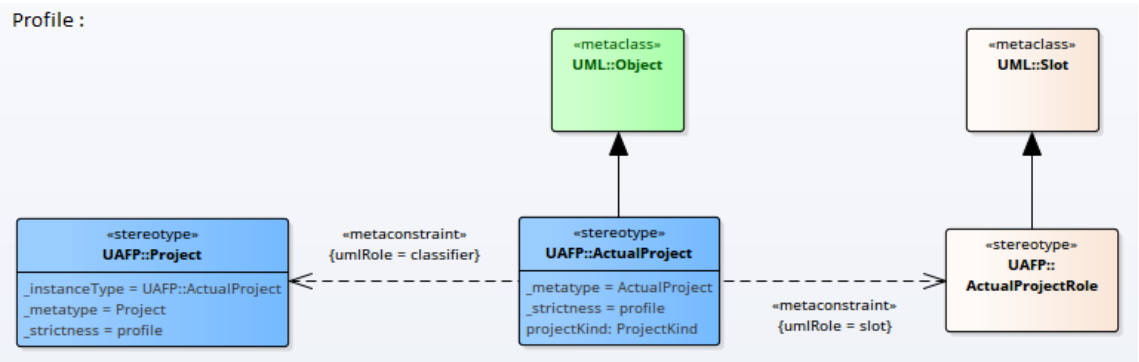
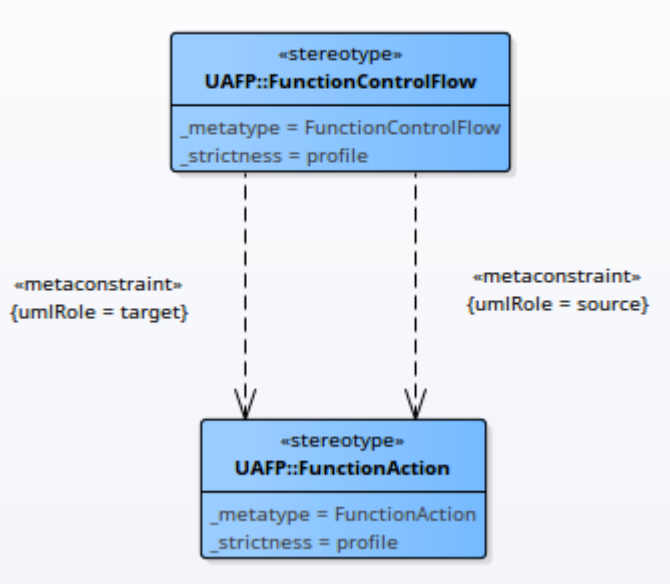
标签 ű mlRole"的约束值

（注意：下表显示了标签 ű mlRole"的所有可接受的约束值。这些值区分大 写，应按表中所示输入。）

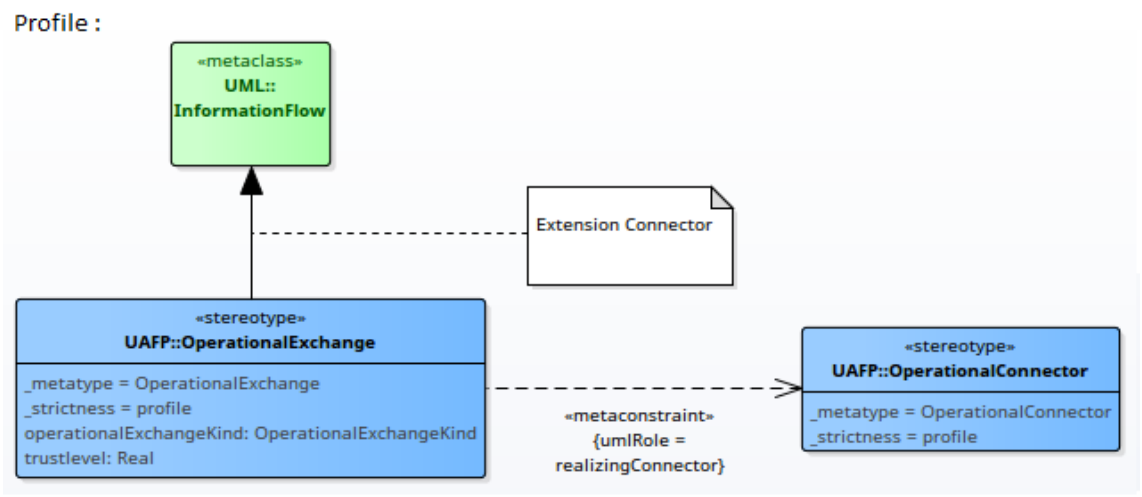
Meta-Constraint 连接器上标签 ű mlRole"的约束值为：

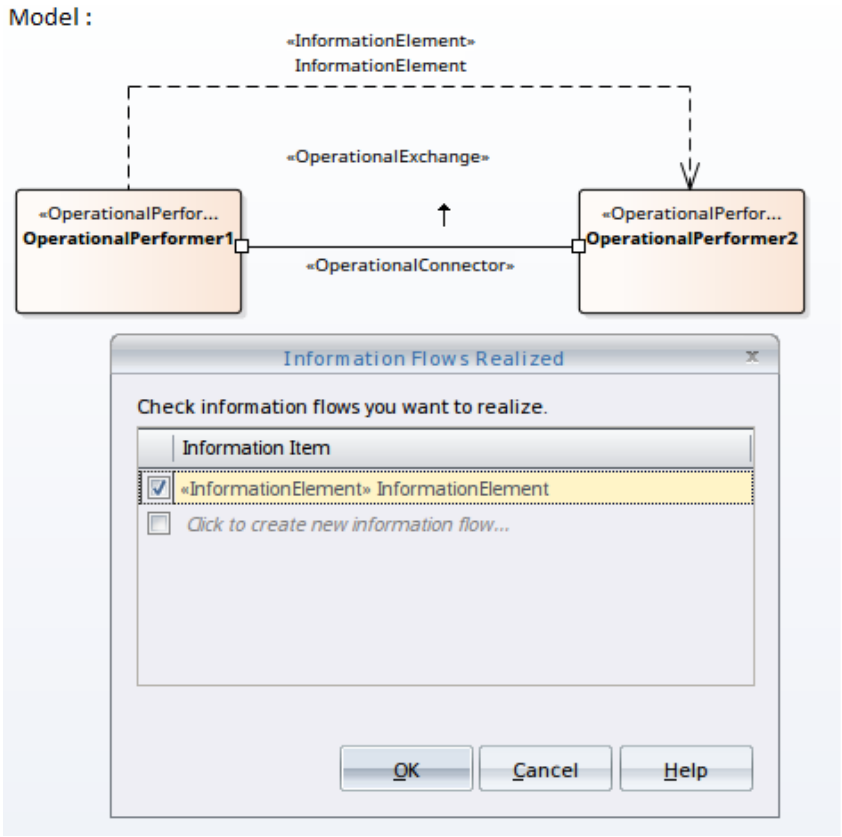
约束	描述
分类器	设置此约束以将源构造型元素的分类器限制为目标构造型元素。 Profile :  Model :  在配置文件示例中，Meta-Constraint 连接器从构造型 ActualCondition 绘制到条件，并且在连接器的标记值列表中的标记 ű mlRole"上将约束指定为 分类器"。这意味着只有 条件"原型元素可以设置为 ActualCondition 原型元素的分类器。 将此配置文件导入模型后， Enterprise Architect将在为条件原型元素设置 DataType 时仅在 选择数据类型"对话框中显示条件原型元素。

类型	设置此约束以在按住 Ctrl 键的同时将目标构造型元素从浏览器窗口拖放到图表中时指定其类型。 Profile :  Model :  在配置文件示例中，Meta-Constraint 连接器从原型 CapabilityProperty 绘制到 Capability，并且该约束在连接器属性窗口的 "Tags"选项卡中的标记 "umlRole"上指定为 "type"。 将此配置文件导入模型后，当按住 Ctrl 键并从浏览器窗口将 Capability 原型化元素拖放到图表中时， 粘贴<item>"对话框将显示 CapabilityProperty 作为选项之一'放下成'清单。
行为	设置此约束以将源构造型元素的行为限制为与目标构造型元素相同。 Profile :  Model :  在配置文件示例中，元约束连接器从构造型 FunctionAction 绘制到函数，并且约束在连接器的属性窗口的 标记"选项卡中的标记 "umlRole"上指定为 行为"。这意味着只有一个 函数"原型元素可以被设置为一个功能动作原型元素的分类器。 将此配置文件导入模型后， Enterprise Architect在设置 FunctionAction 原型元素的行为时，将在 选择活动"对话框中仅显示函数原型元素。
传达	设置此约束以限制可以在扩展信息项信息流连接器的构造型上传达的信息。

	Profile :  在配置文件示例中，元约束连接器从构造型 OperationalExchange 绘制到属性 OperationalExchangeItem，并且在连接器窗口的 标记"选项卡中的标记 属性"上将约束指定为 传递"。这意味着当绘制 OperationalExchange 连接器时，可以在连接器上传递的信息项被限制为 OperationalExchangeItem 原型元素。
投币口	设置此约束以将构造型元素的槽限制为目标构造型元素。  在配置文件示例中，从构造型 ActualProject 到 ActualProjectRole 绘制了一个 Meta-Constraint 连接器，并且在连接器的标记值中的标记 'umlRole' 上将约束指定为 'slot'。注记构造型 'ActualProject' 扩展了UML物件并可以对刻板印象 项目"进行分类。在模型中创建项目元素的实例规范时（通过在按住 Ctrl 键的同时将其从浏览器窗口拖放到图表中）： • 创建的实例规范将被定型为 ActualProject • 'Project' 原型元素中的任何属性都将被创建为实例规范中的原型属性'属性'
客户/源/结束[0].角色/信息来源	设置此模型验证约束以限制 Stereotyped 连接器的起始元素。 Profile :  在配置文件示例中，从构造型 "unctionControlFlow"到 "unctionAction"绘制了一个元约束连接器，并且在连接器的标记值中的标记 "mlRole"上将约束指定为 源"。这意味着当绘制

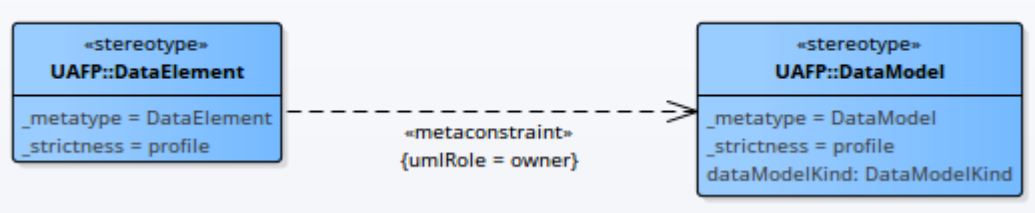
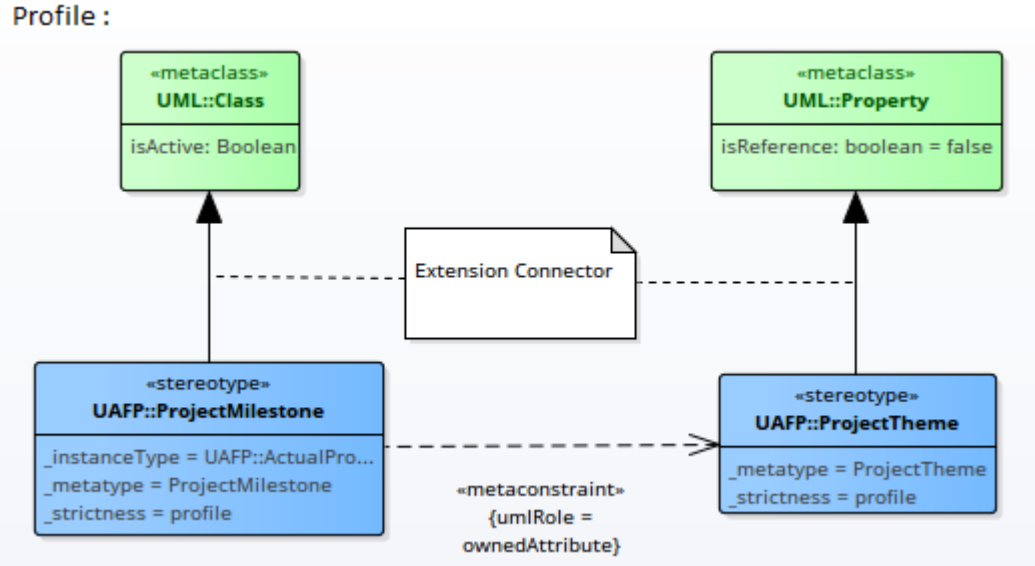
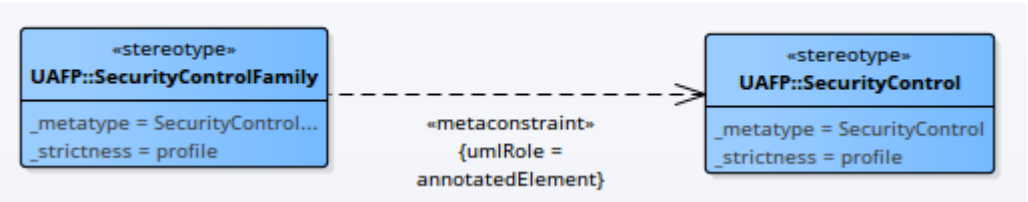
	FunctionControlFlow 连接器时，源元素应该是 FunctionAction 原型元素。否则， Enterprise Architect将在执行模型验证时标记错误。
供应商/目标/结束[1].角色/信息目标	设置此模型验证约束以限制 Stereotyped 连接器的目标元素。
实现连接器/实现 Activity Edge/实现消息	设置此约束来限制可以实现信息流连接器的关系。





在配置文件示例中，Meta-Constraint 连接器从构造型 OperationalExchange（它扩展了UML信息流元类）绘制到 OperationalConnector，并且在连接器标记值中的标记 "umlRole" 上将约束指定为 "realizingConnector"。这意味着在绘制 OperationalConnector 连接器时，可以在此连接器上实现的信息流连接器可以是 OperationalExchange 原型连接器。

类型化元素/实例规范	当作为分类器从浏览器窗口中删除时，此约束将可用类型限制为目标构造型元素。
所有者/班级	设置此约束以将元素的容器/所有者限制为目标构造型元素。用于嵌入元素规则链接器并用于创建此快速嵌套模型验证。

/ 活动 / 拥有实例	Profile :  在配置文件示例中，Meta-Constraint 连接器从原型 DataElement 绘制到数据模型，并且在连接器的标记值中的标记 "umlRole"上将约束指定为 "所有者"。这意味着 DataElement 原型元素可以是数据模型原型元素的子元素。换言之，只有数据模型可以包含/拥有模型中的数据元素。
拥有元素 / 拥有属性 / 拥有运营 / 拥有参数 / 拥有港口	设置这个约束来限制源构造型元素可以拥有的元素/属性/操作/参数/端口。此约束通常用于验证嵌套模型验证。 Profile :  在配置文件示例中，Meta-Constraint 连接器从原型 ProjectMilestone 绘制到 ProjectTheme，并且在连接器的标记值中的标记 "umlRole"上将约束指定为 "ownedAttribute"。这意味着 ProjectMilestone 原型元素可以在模型中包含 "ProjectTheme"原型属性。
注释元素 / 约束元素	设置此模型验证约束以限制 NoteLink 连接器的目标。 Profile :  在配置文件示例中，Meta-Constraint 连接器从构造型 SecurityControlFamily 绘制到 SecurityControl，并且在连接器的标记值中的标记 "umlRole"上将约束指定为 "annotatedElement"。 当配置文件被导入模型时，来自SecurityControlFamily原型化元素的NoteLink连接器的目标应该是SecurityControl原型化元素。否则， Enterprise Architect将在执行模型验证时标记错误。

元模型约束和快速链接器

当您拖动快速链接器箭头以创建与另一个元素的关系时，将显示可用连接器类型的菜单以及 - 如果在图表上未选择目标元素- 将显示可用元素类型的菜单。本主题中的库表显示了连接器名称和元素类型的来源，当您提供或未提供元模型约束属性的值时。

规则过滤

元模型约束主要定义了哪些连接是有效的。快速链接器是根据这些有效关系构建的，然后以多种方式过滤，以便向用户呈现相关关系。

物品	细节
工具箱过滤	默认情况下，对于所有新图表，快速链接器提供的元素和关系被限制为与工具箱中可用的类型相匹配。 这可以由用户在图表上通过选择图表的完全视图或取消选中快速链接器菜单中的 对滤器到工具箱”选项来更改。
公共关系	当还需要创建新元素时，不会将使用关系属性定义的关系作为建议提供。 这些UML关系在与元关系一起使用时包括此行为： • 抽象 • 依赖 • 信息流 • 实现 • 用途

连接器标签

此表标识 Quick Linker 可以从中检索名称以显示在可用连接器类型的菜单中的点。

物品	细节
意义向前和意义向后	具有在 _MeaningForwards 和 _MeaningBackwards属性中定义的值的构造型将使用这些值来描述快速链接器菜单中的连接器。 注记：如果没有为构造型定义_MeaningBackwards，快速链接器将提供一个选项来创建反向或反向的关系。
元类型名称	当未定义 名称”属性时，具有在构造型属性中定义的值的构造型将使用这些值来描述快速链接器菜单中的连接器。
构造型名称	如果未定义 _MeaningForwards、_MeaningBackwards 或 _Metatype 值，则构造型名称将用作关系的菜单标签。
元类名称	当使用元关系连接器在您的原型之间包含UML关系时，您无法控制用于关系的标签。当这些关系在UML元素之间可用时，快速链接器将使用相同的标签。

元素标签

当您将快速链接器拖到空白处时，菜单会显示可用的目标元素类型。此表标识快速链接器从何处检索名称以显示在可用元素的菜单中。

物品	细节
元类型名称	具有在构造型属性中定义的值的构造型将使用这些值来描述快速链接器菜单中的元素。
构造型名称	如果没有定义 _MeaningForwards、_MeaningBackwards 或元类型值，则构造型的名称将用作元素的菜单标签。
元类名称	当使用 Metarelationship 连接器或 Stereotypedrelationship 连接器将您的原型链接到UML元素时，您无法控制用于元素的标签。快速链接器将使用与在UML下连接这些元素时使用的标签相同的标签。

快速链接器

当用户在图表上创建新元素和连接器时，他们可以使用快速链接器箭头来简化流程，该箭头显示可以从选定元素发出的常见连接器列表以及每个连接器可以连接的常见元素列表至。这些列表源自快速链接器定义，它是一个逗号分隔值 (CSV) 格式文件。

作为配置文件的一部分，您可以使用自己的定义添加或替换内置快速链接器定义。这些可以来自：

- A Quick Link Definition Format CSV 通过将 CSV 文本添加到配置文件中的配置文件中，与配置文件集成（在配置配置文件图上添加文档工件元素首选方法） - 请参阅*Quick Linker Definition Format*帮助主题
- A定义元模型图视图，包括一组元模型约束，这些约束定义哪些类型的元素器通过什么类型的连接器连接（第二个首选方法） - 请参阅引入元模型视图和定义元模型约束帮助主题）
- A库表关系配置文件，您还可以通过将元素文本添加到文档工件关系图与配置文件关系图的元数据图来集成配置文件（最好只用于实现不一定对应于配置文件的复杂规则的复杂规则） - 请参阅关系库表帮助主题

注记

- 快速链接器定义背后的理念不是提供有效或合法连接的完成列表，而是提供给定上下文的最常见连接的简短且方便的列表

快速链接器定义格式

为了替换或更改用户从图表上的一个配置文件元素中拖动快速链接器箭头时显示的快速链接器菜单，您可以创建或编辑相应的快速链接器定义。这是一个逗号分隔值 (CSV) 文本文件，由记录（行）组成，每条记录由表中定义的 23 个逗号分隔字段组成。

其中一些字段定义菜单命令，而另一些则充当过滤器，如果不满足过滤条件，则忽略该条目。

快速链接A定义可以包括注释： Enterprise Architect忽略所有前两个字符 // 的行。字段值中的引号 (" ") 不是必需的。

快速链接器定义的每条记录代表快速链接器菜单上的单个条目组合；即，对于选定的源元素，特定的连接器类型和特定的目标元素类型。从满足过滤器的所有行中填充A菜单；也就是说，第一个菜单列出了所有定义的对源元素类型合法有效的连接器，第二个菜单列出了所有对源元素和连接器类型组合合法有效的目标元素。

快速链接器定义字段

柱子	标题（作为指导意见输入）
A	源元素类型 描述：标识配置文件中的有效源元素。 如果连接器被拖离这种类型的元素，则评估该行。否则，该行将被忽略。 如果源是另一个连接器，则在连接器类型前加上单词'link：'；例如， 链接：控制流”。
B	源构造型过滤器 描述：标识源元素基本类型的构造型（例如，事件源元素可以是普通事件、开始事件、中间事件或结束事件型元素）。构造型可以是完全限定的构造型或当前配置文件中构造型的名称。 如果设置，并且如果连接器被拖离此构造型的元素，则评估该行。否则，该行将被忽略。
C	目标元素类型 描述：标识配置文件中的有效目标元素。要表明目标元素可以是抽象UML元类的任何特化，请将前缀 @”添加到元类名称；例如，“@Classifier”、“@NamedElement”。 如果设置，并且如果连接器被拖到这种类型的元素上，则评估该行。 如果为空白，并且如果将连接器拖到图表上的空白区域，则评估该行。 否则，该行将被忽略。 如果目标是另一个连接器，请在连接器类型前加上 链接：”一词；例如， 链接：控制流”。
D	目标构造型过滤器 描述：标识目标元素基本类型的构造型。 如果已设置，并且目标元素类型也已设置，并且连接器被拖到此构造型的元素上，则将评估该行。否则，将忽略该行。 如果没有设置，并且如果连接器被拖到目标元素类型的非构造型元素上，则评估该行。否则，忽略该行。
E	图表过滤器 描述：包含图表类型的包含列表或独占列表，这限制了可以在其上创建指定

	连接器的图表。 • 每个图表名称都以分号结尾；例如： 协作；物件；习惯； • 可以使用完全限定的图表类型 (DiagramProfile::DiagramType) 引用来自 MDG 技术的自定义图表类型；例如： BPMN0::业务流程;BPMN2::Choreography;BPMN2::协作;协作; • 作为图表配置文件中所有图表类型的简写，您可以使用 *"通配符，它必须以图表配置文件 ID 开头；例如： BPMN2.0::*; • 每个排除的图表名称前面都有一个感叹号；例如： ！序列; 此列覆盖快速链接器的 对滤器到工具箱"设置，该设置在图表上默认启用。要强制连接器在所有图表上可见，您可以排除不存在的图表类型。例如： !TBFilter 注记：执行图表过滤器的首选机制是现在工具箱过滤器。这会根据当前图表自动显示相关连接器类型，包括图表类型，因为它们在未来由其他技术定义。
F	新元素类型 描述：定义将连接器拖入开放空间时要创建的元素类型，前提是 创建元素"字段设置为True 。 此值不能是连接器类型。
G	新元素构造型 描述：定义将连接器拖入开放空间时要创建的元素原型的类型，前提是 创建元素"字段设置为True 。这可以是完全限定的构造型，也可以是当前配置文件中构造型的名称。
H	新链接类型 描述：定义要创建的连接器类型，如果 'Create Link' 也设置为True 。
I	新的链接构造型 描述：定义创建的连接器的构造型，如果 'Create Link' 也设置为True 。将快速链接器记录添加到内置类型时需要此字段。构造型可以是完全限定的构造型，或者是当前配置文件中构造型的名称。
J	新链接方向 描述：定义连接器方向，可以是： • 定向（总是创建一个从源到目标的关联） • from（总是从目标到源创建一个关联） • 无向（总是创建一个未指定方向的关联） • 双向（始终创建双向关联），或 • to（根据 关联方向"字段的值创建有向或无向关联） 并非所有这些都适用于所有连接器类型；例如，你不能创建一个概括的你。
K	新链接标题 描述：如果正在创建新连接器但不是新元素，则定义要在 快速链接器"菜单中显示的文本。
L	新链接和元素标题

	描述：定义在创建新连接器和新元素时要在 快速链接器”菜单中显示的文本。
M	创建链接 描述：如果设置为True ，则会创建一个新的连接器；留空以停止创建连接器。
N	创建元素 描述：如果设置为True并且连接器被拖到图表上的空白区域，则会导致创建新元素。 留空以停止创建元素。这会覆盖 目标元素类型”和 目标构造型过滤器”的值。
O	禁止自连接器 描述：如果自连接器对这种连接器无效，则设置为True ；否则将此字段留空。
磷	ST过滤器+独有 没有从 Metatype 继承 描述：设置为True以指示具有构造型 源构造型过滤器”的类型 源元素类型”的元素不显示等效未构造型元素的快速链接器定义。 如果 源构造型过滤器”字段（ B列）为空，则忽略该字段。
Q	菜单组 描述：指示在其中创建菜单项的子菜单的名称。 此列仅在创建新元素时适用；也就是说，用户正在从一个元素拖动到图表上的空白区域，或者在目标元素上拖动以创建一个新的嵌入元素。
R	复杂程度 描述：包含标识复杂功能的数字位掩码值。 • **0** = 没有复杂的功能 • **4** = 强制空白源刻板印象；除非源元素没有刻板印象，否则将跳过该行 • **8** = 强制空白目标定型；除非目标元素没有刻板印象，否则将跳过该行 • **16** = 将 源构造型过滤器”列（ B列）中的值视为源名称过滤器 • **32** = 将 目标构造型过滤器”列（ D列）中的值视为目标名称过滤器，并使用 新元素构造型”列（G 列）中的值作为新的名称创建元素 • **64** = 将 源构造型过滤器”列（ B列）中的值视为源分类器名称过滤器 • **128** = 将 目标构造型过滤器”列（ D列）中的值视为目标分类器名称过滤器，并使用 新元素构造型”列（G 列）中的值作为新创建元素的分类器，如果当前模型中不存在该名称的元素，则创建一个额外的新元素 这些值可以加在一起以组合功能；例如， **192**结合了**64**和**128**的功能。
S	目标必须是父母 描述：如果菜单项只应在从子元素拖到其父元素时出现，则设置为True ；例如，从一个端口到它的包含类。否则将此字段留空。
T	嵌入元素 描述：设置为True以将正在创建的元素嵌入到目标元素中；否则将此字段留空。

U	在分隔符 LEAF 之前 描述：设置为True以在此条目下方的 快速链接器”菜单中添加菜单项分隔符；否则将此字段留空。
V	先于分隔符 GROUP 描述：设置为True以向 快速链接器”子菜单添加菜单项组分隔符；否则将此字段留空。
W	虚拟柱 描述：根据您使用的电子表格应用程序，此列可能需要每个单元中的值，以强制 CSV 导出以正确处理尾随空白值。

关系库表

指定元素之间的快速链接器链接的另一种方法是使用关系表，您最初使用 Microsoft ™ Excel 等电子表格应用程序将其创建为 CSV 文件。创建填充文件后，您将其导入到个人资料中的文档工件元素中。

此方法导致的行为等同于在您的配置文件中描述的构造型之间使用构造型关系连接器。

在大多数情况下，我们建议使用在快速链接器定义格式中定义链接的原始方法，或在元模型视图中建模关系，而不是使用这种关系库表方法。但是，为了实现不一定对应于定义的元模型的复杂关系规则，支持此方法。

格式

关系表的格式基于 ArchiMate 规范中使用的格式，增加了两行将名称映射到构造型。根据以下格式指南设置表：

部分	描述
连接器别名	定义中的第一行提供了映射到完全限定的连接器原型的单字母连接器标识符列表。例如： **a** =ArchiMate3::ArchiMate_Access; **c** =ArchiMate3::ArchiMate_Composition; 也就是说，在文件主体中， **a**表示 ArchiMate 3 ArchiMate访问连接器， **c**表示 ArchiMate 3 ArchiMate组合连接器。
元素别名	定义中的第二行提供了映射到完全限定元素原型的标识符列表。例如： 评估=ArchiMate3::ArchiMate_Assessment；约束 =ArchiMate3::ArchiMate_Constraint； 也就是说，在文件主体中，“Assessment”指的是 ArchiMate 3 ArchiMate 评估元素。
源元素	定义中的第三行列出了针对第二行中的标识符定义的所有可能的源元素。这些是表中的列标题。例如： ,评估,约束,
目标元素	第一列，从第四行开始，列出了根据第二行中的标识符定义的所有可能的目标元素。这些是表中的行标题。
链接定义	行和列交叉处的单元标识在源和目标元素之间有效的连接器，使用在第1行定义的单字母标识符。例如： **S n** o，表示A列类型的元素可以通过Specialization、 **Composition** I Aggregation、 **Influence**和Ass **o** ciation连接器连接到该行类型的元素

将库关系库表添加到配置文件

节	讨论
1	打开包含配置文件的构造型元素的配置文件配置文件子图。
2	选择 文档图表”工具箱（点击  工具箱项目”并指定 工具箱项目”，并在 文档工具”对话框中

	拖拽一个 文档工件元素"到图表上）将此元素命名为 关系表"。
3	双击元素打开链接文档编辑器；取消模板名称的提示。
4	在文本编辑器中打开您的记事本文件，然后将内容复制到文档工件中，并以链接文档的形式元素到文档中。 保存并关闭文档。
5	继续处理配置文件直到它完成，然后保存它。 QuickLink 定义与配置文件一起保存，并在将配置文件（在其MDG 技术内）导入另一个模型时进行处理和应用。 A技术可以包含多个Profiles ，因此具有多个快速链接定义，每个配置文件一个。

快速链接器示例

如果要创建快速链接器定义，最简单的方法是在电子表格中进行设置，每个菜单项定义跨行构建，如下例所示：

	A	B	C	D	E	F	G	H	I	J	K
1	//Source Element Type	Source ST filter	Target Element Type	Target ST Filter	Diagram Filter	New Element Type	New Element ST	New Link Type	New Link ST	New Link Direction	New Link Caption
2	Class	quick				Component		Dependency		to	
3	Class	quick				Component		Dependency		from	
4	Class	quick	Component					Dependency		to	Dependency to
5	Class	quick	Component					Dependency		from	Dependency from
6	Class	quick	Port					Dependency		to	Dependency to
7	Class	quick	Port					Dependency		from	Dependency from
8	Class	quick	Component			Port		Dependency		to	
9	Class	quick	Component			Port		Dependency		from	
10											

	L	M	N	O	P	Q	R	S	T	U	V	W
1	New Link & Element Caption	Create Link	Create Element	Disallow Self connector	Exclusive to ST Filter & No inherit from metatype	Menu Group	Complexity Level	Target Must Be Parent	Embed element	Preceeds Separator LEAF	Preceeds Separator GROUP	DUMMY COLUMN
2	Dependency to	TRUE	TRUE	TRUE	TRUE	Component	0					
3	Dependency from	TRUE	TRUE	TRUE	TRUE	Component	0			TRUE		
4		TRUE		TRUE	TRUE		0					
5		TRUE		TRUE	TRUE		0			TRUE		
6		TRUE		TRUE	TRUE		0					
7		TRUE		TRUE	TRUE		0			TRUE		
8	Dependency to	TRUE	TRUE	TRUE	TRUE	Port	0		TRUE			
9	Dependency from	TRUE	TRUE	TRUE	TRUE	Port	0		TRUE	TRUE		
10												

该示例的第一行是标识列标题的注释行。随后的几行定义了具有构造型 «quick» 的类元素的连接器/目标元素选项。当连接器被拖离这种类型的元素时，您希望用户创建一个到或来自部件元素的依赖关系。当他们将连接器拖到现有的端口或部件元素上时，您希望该组件具有依赖关系，或者在部件的情况部件，您希望用户能够创建嵌入的端口元素。

这些要求在快速链接器定义文件的八条记录中定义：

1. 对新部件的依赖
2. 来自新元件的部件
3. 对现有元件的部件
4. 来自现有元件的部件
5. 对现有端口的依赖
6. 来自现有端口的依赖
7. 对现有组件的部件，创建新的端口
8. 依赖现有部件，创建新端口

记录保存到此 CSV 文件：

//源元素类型,源ST过滤器,目标元素类型,目标ST过滤器,图表过滤器,新元素类型,新元素ST,新链接,新链接类型,新链接方向,新链接标题,新链接 &元素标题，创建链接，创建元素，禁止自我连接器，ST过滤器+不继承元类型，菜单组，复杂性级别，目标必须是父级，嵌入元素，前置分隔符LEAF，前置分隔符组，DUMMY COLUMN

类,quick部件,,Dependency,,to,,Dependency to,TRUE,TRUE,TRUE,TRUE,部件,0,,,,,

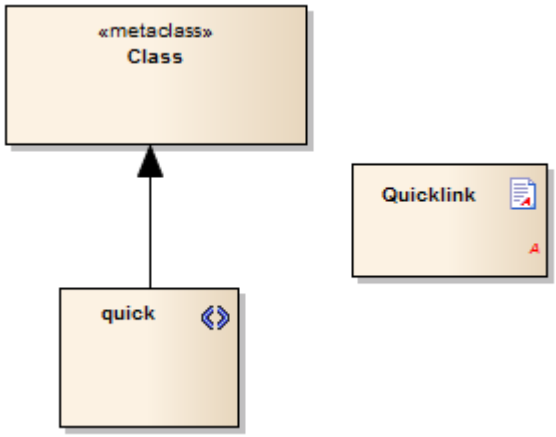
类,quick部件,,Dependency,,from,,Dependency from,TRUE,TRUE,TRUE,TRUE,部件,0,,,TRUE,,

类,quick,部件Dependency,,to,Dependency to,,TRUE,,TRUE,TRUE,,0,,,,,

类,quick,部件Dependency,,from,Dependency from,,TRUE,,TRUE,TRUE,,0,,,TRUE,,

类,quick,端口,,,,,Dependency,,to,Dependency to,,TRUE,,TRUE,TRUE,,0,,,,,,
类,quick,端口,,,,,Dependency,from,Dependency from,,TRUE,,TRUE,TRUE,,0,,,TRUE,,
类,部件,组件,,端口,,依赖,,to,,依赖到,TRUE,TRUE,TRUE,端口,0,TRUE,,,
类,部件,组件,,端口,,依赖,,来自,,依赖来自,TRUE,TRUE,TRUE,端口,0,TRUE,TRUE,,

如果您想测试效果，您可以将此配置文件剪切并粘贴到 QuickLink文档工件中的元素中。



隐藏默认快速链接器设置

如果您为元素创建自己的快速链接器定义，您可能希望在给定的源和目标元素之间隐藏默认的UML快速链接器选项。如何执行此操作取决于您是使用元模型定义方法还是电子表格定义方法来定义快速链接器链接。

元模型法

在每个源原型元素的 <<metaclass>>元素中，添加属性 _HideUmlLinks 设置为 "True"，这样以该原型作为源元素的快速链接将不包括从基本UML元类继承的快速链接。

电子表格法

首先，您可以通过在定义 CSV 文件中根据需要在每一行上将 "Exclusive to stereotype"过滤器标志（列 P）设置为True来隐藏默认的UML快速链接器选项。

或者，您可能希望隐藏默认的快速链接器选项，而无需替换自定义选项。例如，通常如果您没有为一个«quick»类定义到另一个 «quick»类的任何快速链接，快速链接器箭头会显示一个类到另一个类的默认快速链接。要覆盖此行为，请创建一个快速链接器定义，您可以在其中设置：

- 源元素类型（ A栏）
- 源构造型过滤器（ B栏）
- 目标元素类型（C栏）
- 目标构造型过滤器（ D列）
- 新的链接类型（H 列）到<none>
- 刻板印象独有 + 不从 Metatype（P 列）继承为 TRUE

尝试将此行添加到快速链接器示例中：

类,quick,接口,,,,, <none> ,,,,,,,,TRUE,,0,,,,,

通过定义中的这一行，当快速链接从«快速»类拖动到接口元素时，默认的类到接口快速链接被隐藏。

注记 "Exclusive to stereotype"过滤器隐藏了所有没有此过滤器集的上下文敏感关系，这将在定义了源元素原型的地方生效。

快速链接器物件名称

创建快速链接器定义文件时，您使用一系列基本元素和连接器类型来标识：

- 源元素类型（ A栏）
- 目标元素类型（C栏）
- 新元素类型（F 列）和
- 新链接类型（H 列）

这些然后由您在定义中指定的构造型限定。您可以使用的基本元素和连接器类型在此处标识。

物件类型名称

物件組	物件类型
元素类型	行动 行动销 活动 活动参数 活动分区 参与者 工件 边界 中央缓冲区节点 更改 选择状态 类 协作 部件 数据类型 决策 深度历史状态 部署规范 设备 图门 实体 入口点 入口状态 执行环境 出口点 退出状态 扩展节点 扩展区域 特征 最终活动

	图形界面元素 历史状态 信息项 初始活动 初始状态 交互发生 接口 问题 可中断活动区域 连接状态 合并节点 消息端点 n元关联 节点 物件 物件节点 包 部件 端口 原始类型 提供的接口 接收 必需接口 需求 屏幕 发送 序列 信号 状态 州生命线 状态机 同步_H 同步_V 同步状态 UMLD图 用例 价值生命线
连接器类型	抽象 聚合 关联 关联类 通讯路径 组合 连接器链接

	控制流 代表链接 依赖 部署 扩展 概括 信息流 接口链接 显现 嵌套 对象流 包导入 包合并 实现 重新定义 序列 状态流 替代 模板绑定 UC扩展 UC包括 用途 用例

将快速链接器定义添加到配置文件

将配置文件快速链接器定义设置为 CSV 文件后，您可以将它们合并到配置文件中。要执行此操作，请将文档工件的文件内容复制到与配置文件的构造型元素相同的图表中。

在配置文件中添加定义

节	讨论
1	打开包含配置文件的构造型元素的配置文件配置文件子图。
2	选择 "文档图表"工具箱（点击  工具箱项目"并指定 "工具箱项目"，并在 "文档工具"对话框中拖拽一个 "文档工件元素"到图表上）将此元素命名为 "QuickLink"。
3	双击元素打开链接文档编辑器；取消模板名称的提示。
4	在文本编辑器中打开，然后将内容复制到您的 CSV文档工件中，然后将其作为元素本链接文档。 保存并关闭文档。
5	继续处理配置文件直到它完成，然后保存它。 QuickLink 定义与配置文件一起保存，并在将配置文件（在其MDG 技术内）导入另一个模型时进行处理和应用。 A技术可以包含多个Profiles ，因此具有多个快速链接定义，每个配置文件一个。

导出一个配置文件

创建配置文件、定义构造型元素并添加所需的任何标记值、形状脚本、约束和快速链接器定义后，您可以将配置文件保存（导出）到磁盘。然后可以将配置文件与MDG 技术集成并部署到其他模型以供使用。

保存配置文件

节	描述
1	如果你的配置文件是： • 单个配置文件分布在同A配置文件包中的多个图表上，在浏览器窗口中找到配置文件包，然后选择 特定>技术>发布技术>将包发布为UML配置文件”功能区选项 • 同一配置文件包中的多个Profiles文件之一，单击配置文件图背景中的任意位置，然后选择功能区选项 设计>图表>管理>另存为配置文件”或 特定>技术>发布技术>发布”图表为UML配置文件' • 配置文件包中A单个图表，单击配置文件图表背景中的任意位置，然后选择功能区选项 设计>图表>管理>另存为配置文件”或 ⺌特定技术>发布技术>图表为” UML配置文件' 将显示 保存UML配置文件”对话框。
2	单击  按钮，然后选择 XML配置文件的目标目录路径。 如有必要，编辑配置文件名，但不要删除 .xml 扩展名。
3	在 配置文件类型”字段中，使用默认值 EA (UML)2.X”（或者，如有必要，单击下拉箭头并选择此值）。 注记：如果此字段显示为灰色，则表示您要从中导出配置文件的包没有 <<profile>> 构造型。您必须提供该构造型的包或将配置文件图和/或元素转移到具有该构造型的另一个包。
4	为配置文件中定义的所有构造型设置所需的导出选项： • 元素- 选中复选框以导出元素属性 • 颜色和外观-（如果从图表中保存配置文件则启用；如果从浏览器窗口中的包中保存则禁用）选中该复选框以导出背景颜色、边框颜色、文本颜色和边框粗细属性 • 图像选中复选框以导出元文件图像 • 代码模板- 选中复选框以导出代码模板（如果存在）
5	单击 保存”按钮将配置文件保存到磁盘。

避免配置文件名称和 ID 冲突

每个配置文件都应该有一个唯一的名称和 ID。配置文件名称是在保存配置文件时指定的，而 ID 是从用于保存配置文件的图表或包的GUID派生的。为避免名称和 ID 冲突：

- 创建多个Profiles时，为每个配置文件使用新图表或包
- 保存Profiles时，输入唯一的配置文件名

在启动Enterprise Architect或启用MDG 技术时，如果检测到重复的配置文件名称或重复的配置文件ID，则会在系统输出窗口中显示警告。

注记

- 为了快速测试配置文件，您可以将 XML 文件单独导入浏览器窗口的 资源”选项卡中；对于最终部署，将配置文件合并到MDG 技术中

保存配置文件选项

保存配置文件时，可以从其父包或配置文件图中保存它，具体取决于配置文件是否为：

- 单个配置文件分布在同A配置文件包中的多个图表上，这通常是构造型配置文件的情况
- 同一个配置文件包内的多个Profiles之一；例如，创建多个工具箱配置文件时
- 配置文件包内A单图

访问

功能区	设计>图表>管理>另存为配置文件 特定>技术> 发布技术> 将图表发布为UML配置文件 特定>技术> 发布技术> 将包发布为UML配置文件

选项比较

从图表保存	保存来自包
配置文件采用图表名称。	配置文件取包名。 注记：包和图表名称不一定相同，但如果将它们设置为相同或非常相似，可以避免很多混乱。 例如：带有图表 GL1、GL2、GL3 的包GL。
配置文件取图表的注记。	配置文件取包的注记。 注记：图表注记在配置文件定义中可能很重要，例如工具箱Profiles 。 请参阅Create Toolbox Profiles
您可以从图表object中获取默认大小和外观（包括替代图像）。	您不能从图表object中获取默认大小和外观。 您可以使用 _sizeX、_sizeY 和 _image属性，但没有等效的默认颜色。 注记：
此选项可以更快。	此选项可能会慢得多。 注记：之所以出现差异，是因为图表对象保存在内存中，而浏览器窗口元素没有。 仅当配置文件很大并且您使用慢速网络连接到远程存储库时，这才可能成为问题。

浏览器-资源中的UML Profiles

浏览器窗口的 资源”选项卡包含一个树结构，其中包含包括UML Profiles在内的一系列项目的条目。 UML Profiles节点最初不包含条目；为了能够使用 资源”选项卡中的Profiles ，您必须将它们从外部 XML 文件导入到项目中。

配置文件中的项表示构造型。这些可以通过多种方式应用于UML元素；例如，适用于的刻板印象：

- 类和接口等元素可以直接从 资源”选项卡拖到当前图表中，自动创建原型元素；或者，可以将它们拖到现有元素上，自动将它们应用到元素
- 属性可以拖放到宿主元素上（例如类）；一个刻板的属性会自动添加到元素的特征列表中
- 操作与应用于属性的操作相同；拖放到主机元素上以添加原型操作
- 通过在浏览器关联的 资源”选项卡中选择它们，然后单击图表中的开始元素并拖动到结尾元素（与添加普通连接器）；添加了一个定型连接器
- 可以通过将连接器末端元素拖动到图表中的关联末端来添加关联末端

浏览器-导入UML Profiles Into资源

Profiles以 XML 文件的形式存在，可以导入到任何项目中，为特定领域提供量身定制的建模结构。 Sparx Systems网站上A许多配置文件XML 文件可供您使用，用于导入您的模型。您还可以导入自己创建的配置文件XML 文件。如果配置文件包含对任何元文件的引用，请将这些元文件复制到与配置文件XML 文件相同的目录中。

访问

功能区	开始> 所有窗口>设计> 浏览> 浏览>资源> 右击' UML Profiles ' 文件夹>导入配置文件
键盘快捷键	Alt+6 |右击“UML Profiles”文件夹|导入配置文件

导入一个配置文件

字段/按钮	行动
文件名	单击...按钮并找到要导入的 XML配置文件文件。
元素尺寸	选中复选框以导入配置文件中定义的所有构造型的元素大小属性。
色彩与外观	选中复选框以导入配置文件中定义的所有构造型的颜色（背景、边框和字体）和外观（边框粗细）属性。
替代图像	选中复选框以导入配置文件中定义的所有构造型的元文件图像。
代码模板	选中复选框为配置文件中定义的所有构造型导入代码模板（如果存在）。
覆盖现有模板	对于配置文件中定义的所有构造型，选中复选框以覆盖当前项目中定义的任何现有代码模板。
导入	单击此按钮将配置文件添加到UML Profiles文件夹。 如果配置文件已经存在，则会显示一个提示，让您覆盖现有版本并导入新版本。 导入完成后，配置文件就可以使用了。

MDG 技术-创造

如果您想访问和使用与Enterprise Architect中的特定技术相关的资源，您可以使用模型驱动生成 (MDG)技术来实现。管理员或个人用户有多种选择可以将现有的MDG 技术与Enterprise Architect一起使用。技术开发人员还可以开发新的MDG 技术，并根据需要将它们部署到项目团队，为您的工作领域或环境提供量身定制的解决方案。

使用配置文件Helpers

MDG技术和Profiles是使用Enterprise Architect中的图表和元素开发的。这些图表和元素使用特定的属性和属性来确定生成的MDG 技术的内容和行为。配置文件帮助者协助创建新的MDG技术，这些配置文件类型：

- 构造型Profiles
- 工具箱Profiles及
- 图表Profiles

配置文件助手由两个部分组成：

- 模型中的MDG 技术构建器模板，为创建新的MDG 技术提供了起点
- 配置文件"工具箱中的配置文件帮助项，提供了简化构造型、工具箱和图表Profiles创建的对话框

访问

选择一个要添加MDG 技术生成器模板的包，然后使用这里概述的方法之一显示模型生成器对话框。

功能区	开始> 个人>模型构建器 设计>包>模型生成器
上下文菜单	右键点击包|模型（模式库）
键盘快捷键	Ctrl+Shift+M
其它	浏览器窗口标题栏菜单|模型（图案库）

创造新的MDG 技术

节	描述
1	在 模型生成器"对话框中，单击<视角名称>按钮并选择 管理| MDG 技术生成器"。 在 "MDG 技术构建器"组中选择 基本模板"模式。 单击创建模型按钮。提示显示技术名称的A 。
2	输入新MDG 技术的名称，然后单击确定按钮。 这将创建一个基本的包模板和示例元素，可将其用作创建MDG 技术的起点。该模板包括三个包，每个包都具有与技术相同的名称，但具有与配置文件类型相对应的不同构造型配置文件他们定义： • <<profile>> -用于定义包含构造型用户的配置文件包将应用于元素 • <<图表配置文件>> -描述用户将创建的图表类型的配置文件包 • <<工具箱配置文件>> -描述要在工具箱中显示的元素的配置文件包
3	在每个包中，打开图表并参考提供的示例元素，将其他项目添加到配置文件中。 配置文件工具箱包含一个配置文件帮助图标页面，当将其拖到图表上时，可以帮助您创建和填充各种Profiles的元素。

4	将这些Profiles中的每一个保存到磁盘。
5	将保存的Profiles合并到MDG 技术中。

使用配置文件Helpers 创建构造型Profiles文件

在创建技术以提供特定于领域的工具集时，典型的起点是定义您要提供的每个元素、连接器、特征和结构组件。这些由配置文件定义。

配置文件中定义的所有构造型要么是Enterprise Architect定义的核心UML对象（元类）的扩展，要么是其他现有Profiles和技术定义的非 UML 对象（构造型）的扩展。

完成配置文件的开发后，将其保存到外部 XML 文件中，然后合并到MDG 技术中以进行最终部署。

配置文件中定义的每个构造型都会修改它扩展的元类或构造型的行为。这些修改可能包括：

- 提供附加属性标记值
- 约束来定义适用于每个构造型的条件和规则
- A形状脚本来自定义新object的整体外观
- 更改object A默认外观，例如背景、边框和字体颜色
- 快速链接器定义，提供每个构造型中最常见的连接类型列表
- 定义新object的特定外观和行为的特殊属性，包括初始元素大小和浏览器窗口图标

创建UML配置文件

节	描述
1	在浏览器窗口中，找到具有 <<profile>> 构造型的包并打开其子图。 如果您没有现有的<<profile>>包，请使用模型生成器中的 管理 | MDG 技术生成器”蓝图来创建新技术，然后从新创建的<<profile>>包中打开图表。
2	（可选）如果您希望构造型元素包含引用预定义标签类型的标记值，则可以在配置文件图表上的数据类型元素中定义这些标签类型。在数据类型元素的注记中包含标记值类型定义，例如“Type=Memo;”或 类型=RefGUID；” 如果您希望构造型元素包含具有多个预定义值的下拉列表的标记值，则每组值必须由配置文件图表上的枚举元素定义。 如果您希望构造型元素包含结构化标记值以提供一组复合信息，则每个结构必须由配置文件图表上的类元素定义。 在为构造型定义这些标记值类型之前，枚举和类元素必须存在；您可以在此时创建元素，也可以稍后将这些标记值添加到您的构造型中。
3	通过从图表工具箱中拖动 添加构造型配置文件帮助器”来添加新的构造型工具箱由 添加构造型配置文件帮助器”打开的对话框将允许您为您的构造型指定各种一般属性、标记值和形状脚本脚本。
4	（可选）为构造型定义约束。
5	（可选）设置构造型的默认外观。
6	对要创建的每个新构造型元素重复步骤 3 到 5。
7	（可选）将快速链接器定义添加到配置文件。
8	将包保存为配置文件。 保存配置文件时，使用的名称应与配置文件包的名称一致；这对于工具箱配置文件中的引用正确执行函数是必要的

9	将配置文件合并到MDG 技术中。

注记

- A配置文件包不能包含其他包；不要在配置文件中添加任何其他包

使用配置文件Helpers 添加构造型和元类

您可以在配置文件中定义构造型以扩展：

- 核心UML对象（在Enterprise Architect中预定义的元类），或
- 由其他Profiles和技术定义的对象（构造型）（例如在 ArchiMate 或 SysML 中定义的对象）

构造型可以通过多种方式扩展元类：

- 一个构造型扩展一个元类，用于一种object类型的特定定义
- 一个构造型扩展多个元类，其中定义适用于多个object类型 - 例如以相同的方式修改一个类和一个物件
- 多个构造型扩展一个元类，您在其中创建相同基础object类型的多个变体；例如，定义关联连接器的类型，代表父母、兄弟、祖父母、叔叔/阿姨和表亲关系

将元类和构造型添加到配置文件

节	描述
1	如果您要扩展由现有配置文件或技术定义的非 UML 类型，请遵循创建构造型扩展非 *UML* 对象帮助帮助中描述的过程。
2	在浏览器窗口中，找到带有 <<profile>>构造型的包并打开它的子图。
3	将图形工具箱的 配置文件助手”页面中的 添加构造型”工具箱图表图表上。 将显示 添加构造型”对话框。
4	在 名称”字段中，输入构造型名称（这也是新建模object的名称）。
5	通过单击 类型”下拉箭头选择这些object组之一： • 元素扩展——创建一个扩展元素的构造型 • 连接器扩展-创建扩展连接器的构造型 • 抽象元类 - 创建扩展结构或行为修饰符的构造构造型 • 元类扩展- 创建一个构造型，它扩展了已经存在于您的模型中（并且很可能在您当前正在使用的图表中）的元类
6	单击添加元类按钮。 将显示 扩展元类”对话框，其中显示与在步骤 5 中选择的object组关联的object类型列表。 从列表中选择要扩展的元类，然后单击确定按钮。 如果您在步骤 5 中选择了 无类扩展”，则会显示 选择配置文件元素浏览器/搜索”对话框；搜索并选择现有的 Metaclass元素以使用此构造型进行扩展。 元类名称被添加到 扩展”字段。
7	如果您想扩展多个具有构造型的元类，请再次单击 添加元类”按钮并选择下一个要扩展的object类型。您可以对尽可能多的元类重复此操作，并使用此构造型进行扩展。 要从 扩展”列表中删除选定的元类，请单击 删除”按钮。
8	审阅 构造型”面板中的可用属性。这些属性修改构造型的行为。 要应用属性，请单击 值”字段并键入或选择适当的值。 当您选择一个属性字段时，属性效果的描述会显示在 构造型”面板的底部。

	仅提供要应用于此构造型的属性的值。
9	单击 扩展"字段审阅类的名称，并在 无类"面板中查看可用的属性。这些属性根据特定于被扩展的元类的选项进一步修改构造型的行为。 要应用属性，请单击 值"字段并键入或选择适当的值。 当您选择一个属性字段时，属性效果的描述会显示在 无类"面板的底部。 不要为您不想应用于此构造型的任何属性提供值。 如果要扩展多个元类，请单击 扩展"字段中的下一个元类名称并审阅该object类型的属性。
10	单击下一步按钮。显示 定义标记值"页面。
11	在 属性"面板中，右击可显示上下文菜单，其中包含创建和分组不同类型的标记值的选项。这些选项包括： ● 添加标记值：创建一个简单的标记值- 显示一个提示标记值名称。添加名称，点击确定按钮，在 属性"栏显示名称；要设置默认值，请在 默认值"字段中输入 ● 添加专业标记值： -枚举：创建一个枚举标记值，基于 在一个现有的枚举元素上 - 预定义：从一个预定义的标记值类型中选择一个 列表，并在 默认值"字段中，输入或选择一个首字母 必要时的价值 - Structured：创建一个由Structured标记值组成的结构 其他几个简单的标记值，由现有的类型 类元素 -参考：创建一个标记值，用户可以使用它 定位并引用使用指定创建的元素 构造型（ RefGUID标记值的一种形式）；在创造这个时， 您必须选择定义的现有构造型元素 刻板印象 -参考列表：创建一个用户用来标记值 可以定位和引用使用 a 创建的元素列表 指定的构造型（RefGUIDList标注标记值的一种形式）； 在创建它时，您必须选择现有的构造型元素 定义了刻板印象 ● 编辑标记值名称：显示一个简单的提示，您可以在其中改写当前名称以更正或更改它 ● 创建标签组：在元类元素中创建标签组，通过它来组织你在构造型标记值中创建的元素 ● 将标签移动到组（当您右键单击现有的标记值时显示）：显示 移动标签到组"对话框，您可以在该对话框中选择现有的标签组以包含选定的标记值 ● 删除分组：删除选中的标签组，使其成员标记值列在 属性"列的末尾 ● 删除构造：从列表和列表中删除选定的标记值构造型
12	单击下一步按钮。将显示 定义形状脚本"页面。 形状脚本A用于定义构造型形状的外观。要包含一个形状脚本，请单击编辑按钮。 显示形状编辑器窗口。使用这个编辑器创建你的形状脚本。 完成脚本创建后，点击确定按钮。由形状脚本定义的图像显示在 预览"面板中。 注记：为使形状脚本生效，保存配置文件时必须选择 图像"选项。 或者，您可以在创建构造型元素后为模型object定义简单的默认外观（背景颜色、线条颜色）。
13	单击完成按钮。构造型元素和元类元素现在显示在配置文件图上。
14	您现在可以： ● 对要创建的其他每个构造型元素重复步骤 2 到 13

	• 使用配置文件帮助器编辑您定义的构造型（并通过它，元类）元素属性 • 将约束添加到您的构造型元素中 • 如果尚未设置形状，那么您现在可以定义object的默认外观（背景颜色、线条颜色） • 为配置文件中的原型元素和连接器设置快速链接器定义

注记

- 如果您打算扩展大量的模型元素，而不是将它们全部放在一个图表上，您可以在<<profile>>包下创建额外的子类图，并将不同类型的元类元素添加到不同的图表中；在这种情况下，您将包保存为配置文件，而不是单个图表
- 构造型元素必须具有唯一的名称，但元类元素可以具有相同的名称（例如，可以有多个行动元类，每个具有不同的行动属性）
- 如果您在构造型元素中有多个标记值，并且您已将它们分配给组，您可以在属性窗口的 标签”选项卡中定义哪些组默认为展开（打开），哪些默认为关闭;在 属性”页面打开元类的特征窗口，并添加属性 _tagGroupStates，初始值为 <groupname>=closed;<groupname>=closed;<groupname>=open; ...

编辑构造型元素

如果要添加或更正配置文件中的构造型或元类元素的属性，可以使用标准功能编辑它，例如元素 属性"对话框和 标签"选项卡。但是，您也可以通过配置文件帮助器的 构造型属性"对话框更新构造型元素，并且还可以通过构造型更新构造型扩展的 Metaclass 元素。

您通过其他方式（例如通过元素 属性"对话框）对元素所做的任何更改都会反映在配置文件帮助器的内容中。

访问

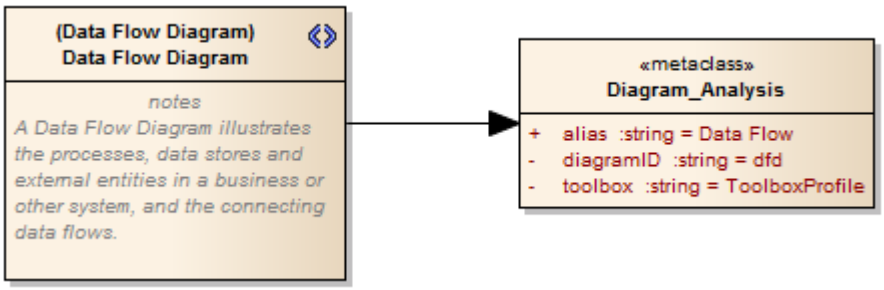
上下文菜单	右键单击构造型元素|使用配置文件助手编辑

编辑构造型元素

节	描述
1	构造型属性"对话框默认为 常规"选项卡。在此选项卡上，您可以： ● 更改构造型元素名称 ● 添加更多元类元素以通过这个构造型元素进行扩展 ● 添加或更改构造型元素的属性值 ● 为每个 Metaclass元素的属性添加或更改值
2	点击 标记值"标签。在此选项卡上，您可以： ● 编辑标签的默认值 ● 添加一系列类型之一的新标签 ● 创建标签组 ● 将标签分配或重新分配给组 ● 删除标签组 ● 从构造型中删除标记值
3	单击 形状脚本"选项卡。在此选项卡上，您可以： ● 添加一个形状脚本（如果不存在） ● 使用形状编辑器编辑现有的形状脚本
4	编辑完构造型元素后，单击确定按钮。 配置文件类图重新显示，已编辑的元素显示您所做的更改。

使用配置文件帮助器创建图表Profiles文件

当您开发MDG 技术时，可以创建扩展图表类型并将它们作为自定义图表Profiles在您的MDG 技术中。例如，您可以创建一个图表图形配置文件分析，将 DFD 图定义为内置图的扩展，如下所示：



添加图表扩展”配置文件帮助器可以帮助您定义图表配置文件，添加必要的元素并赋予它们适当的属性以定义生成的自定义图表类型的功能。

创建扩展图表类型

节	行动
1	如果您还没有这样做，请使用模型生成器的 管理 | MDG 技术生成器”蓝图来创建一组用于定义Profiles的包。 在浏览器窗口中，找到具有 <<diagram profile>> 构造型的包并打开其子图。
2	将工具箱的 配置文件助手”页面中的 添加图表扩展”项拖到图表上。 将显示 添加图表扩展”对话框。
3	在 名称”字段中，输入自定义图表类型的名称。
4	在 扩展类型”字段中，单击下拉箭头并选择自定义图表类型将扩展的内置图表类型。
5	在 描述”字段中输入图表用途的简要说明。 当用户在 New图表”对话框中选择此图表类型时，此描述将显示在对话框的右下方。
6	在 属性”窗格中输入这些字段的值： • 别名：定义在图表标题栏上显示在单词 图表”之前的图表类型；例如： 块图表 • 框架 ID：定义将出现在图表框架标签中的图表类型 • 框架格式字符串：输入一个包含替换宏的string ，用于定义框架标题，可以带或不带 () 等附加分隔符；可以使用的宏是： - #DGMALIAS# - #DGMID# - #DGMNAME# - #DGMNAMEFULL# - #DGMOWNERNAME# - #DGMOWNERNAMEFULL# - #DGMOWNERTYPE# - #DGMSTEREO# - #DGMTYPE# • 工具箱配置文件：点击下拉箭头，选择定义所需工具箱配置文件的图表类型（保存配置文件

	时输入的名称）；每次打开这种类型的工具箱都会自动打开工具箱 • 泳道：定义将在图表上显示的泳道；例如： 车道=2；方向=水平；车道1=标题1；车道2=标题2； （其中*Lanes*可以是任何值，但*Lane<n>*值的数量必须等于*Lanes*的值；方向可以省略，在这种情况下泳道默认为垂直）
7	属性”窗格中的其余字段可用 自定义图表的默认选项。将不会应用任何留空的属性。 当用户选择一个字段时，属性效果的描述会显示在 属性”窗格的底部。
8	点击确定按钮。适当的构造型和元类元素被添加到图中。
9	对要包含在图表配置文件中的每个图表扩展重复步骤 2 到 8。
10	将图表另存为配置文件。
11	将配置文件合并到MDG 技术中。

注记

- 添加图表扩展后，您可以通过右键单击图表上的适当构造型元素并选择 使用配置文件帮助器编辑”来再次修改其属性

使用配置文件帮助器创建工具箱Profiles文件

在MDG 技术中，您可以创建多个工具箱Profiles 。每个工具箱配置文件定义一个工具箱。工具箱由A或多个可展开/可折叠的区域组成，称为工具箱页面。

创建工具箱配置文件

节	行动
1	如果尚未创建用于定义Profiles的包组，请使用模型生成器的 管理 | MDG 技术生成器"蓝图来创建该组。 在浏览器窗口中，找到具有 <<toolbox profile>> 构造型的包并打开其子图。
2	将 配置文件助手"工具箱页面中的 创建自定义工具箱"项拖到图表上。 将显示 选择工具箱配置文件包"对话框。
3	选择步骤1中提到的具有 <<toolbox profile>> 构造型的包。 点击确定按钮。将显示 创建工具箱页面"对话框。
4	在 工具箱名称"字段中输入您的工具箱页面的名称。 这是在使用图形图表中的项目搜索功能时为工具箱页面显示的工具箱。
5	在 描述"字段中输入工具箱的描述。 此描述充当工具箱的默认工具工具箱，除非您如步骤 10 中所述为工具箱页面定义特定的工具提示。
6	点击确定按钮。 将创建并显示用于定义工具箱的图表。
7	（可选）将项目从工具箱拖到图表上时，该项目通常会创建元素或连接器。 也可以有一个工具箱项目，当拖到图表上时，将提供可供选择的项目选择。这被称为隐藏的子菜单。 如果您希望您的工具箱包含一个或多个隐藏的子菜单，您应该在继续执行此页面上的步骤之前定义这些子菜单。
8	您现在可以定义将出现在工具箱上的一个或多个工具箱页面。 将 配置文件助手"工具箱页面中的 添加工具箱页面"项拖到图表上。 将显示 添加工具箱页面"对话框。
9	在 名称"字段中，输入工具箱页面的名称。 这是将显示在相应工具箱页面的标题栏中的文本。
10	在 工具提示"字段中，输入相应工具箱页面的工具提示。
11	在这种情况下， 图标"字段将被禁用。该字段仅在定义隐藏的子菜单工具箱页面时使用。
12	这些选项可用于确定工具箱页面的外观和功能。启用时： • 'Images Only'：显示工具箱页面，图标旁边没有文本标签

	• 'Is Hidden'：将工具箱页面定义为隐藏的子菜单 • '公共'：工具箱页面在您的技术处于活动状态时对所有定义的工具箱都是通用的；页面最初显示为折叠状态 • 'Is Collapsed'：工具箱页面最初是最小化的
13	您现在可以定义要添加到工具箱的项目。 单击 添加”按钮右侧的向下箭头。选择以下选项之一： • '添加构造型'：为当前模型的UML配置文件中定义的构造型添加工具箱项；此配置文件必须包含在MDG 技术中的工具箱配置文件中 选择此选项后，将显示 选择配置文件元素”对话框；使用它来选择要添加的构造型元素（如果需要，在单击多个元素时按住 Ctrl 键） • '添加内置类型'： -元素：为UML元素类型添加一个工具箱项 选择此选项后，将显示 创建新工具箱项”对话框；在 别名”字段中，键入出现在工具箱项目上的标签，然后单击确定按钮 然后显示 选择元类”对话框；选择要添加到您的工具箱的UML元素类型，然后点击确定按钮 - '连接器'：为UML连接器类型添加工具箱项 选择此选项后，将显示 创建新工具箱项”对话框；在 别名”字段中，键入出现在工具箱项目上的标签，然后单击确定按钮 然后显示 选择元类”对话框；选择要添加到您的工具箱的UML连接器类型，然后点击确定按钮 • '添加隐藏工具箱'：添加隐藏工具箱子菜单项；使用此选项之前必须定义隐藏的工具箱 选择此选项后，将显示 创建新工具箱项”对话框；在 别名”字段中，键入要出现在工具箱项目上的标签，然后单击确定按钮 然后显示 选择隐藏的工具箱构造型”对话框；选择要添加到您的工具箱中的隐藏工具箱，然后单击确定按钮 • '加新项目'：添加一个工具箱项目，只有一个别名 仅此选项不会创建功能工具箱项；以这种方式添加的工具箱项，以后必须通过工具箱项列表进行修改 单击 添加”按钮，而不是单击下拉箭头，与选择 添加构造型”选项相同。
14	（可选）定义一个工具箱项目，该项目将从外部MDG 技术创建项目。例如，添加一个创建工具箱块元素的工具箱项。 1. 单击 添加”按钮右侧的向下箭头。 2. 选择 加新物品”选项。 将显示 创建新工具箱项”对话框。 3. 在 别名”字段中，键入要出现在工具箱项目上的标签，然后单击确定按钮。 工具箱项将被添加到 工具箱项”列表中。 4. 在此工具箱项目的 构造型”字段中，键入： 配置文件::构造型(UML ::BaseUMLType) -配置文件是定义构造型的配置文件的名称 -构造型是此工具箱项将创建的构造型/元类型的名称 - *BaseUMLType*是非 UML object的基本UML类型 例如，要在工具箱中包含 SysML块，您可以键入： SysML1.3::块(UML ::类) 5. 要识别配置文件：构造型string ，请创建要包含在工具箱中的类型的元素（例如，SysML 1 .3块），然后选择元素并显示属性窗口。 此元素的任何预定义标签都将分组在配置文件::构造型标题下；例如，SysML 1 .3块的标签被分组在 SysML1.3::块下。 Enterprise Architect中的所有非 UML 对象都是UML类型的扩展。您可以通过删除元素的构造型来显示元素的基本UML类型。例如，创建一个构造型块，然后使用属性窗口删除该块元素的构造类型。元素类型将从块变为类。

15	（可选）创建一个工具箱项目，将模式拖放到图表上。 1. 单击 添加"按钮右侧的向下箭头。 2. 选择 加新物品"选项。 将显示 创建新工具箱项"对话框。 3. 在 别名"字段中，键入要出现在工具箱项目上的标签，然后单击确定按钮。 4. 工具箱项将被添加到 工具箱项"列表中。 5. 在此工具箱项目的 构造型"字段中，键入： TechnologyID::PatternName(UMLPattern) - *TechnologyID*是在MDG 技术创建向导中输入的技术ID - *PatternName*是保存模式时输入的名称；例如： BusFramework::Builder(UMLPattern) 如果您想避免显示 添加模式"对话框，请将 (UMLPattern) 替换为 (UMLPatternSilent)。 6. 要在自定义工具箱（例如 GoF模式）中定义基于模型的模式，请创建一个具有以下格式名称的属性： PatternCategory::PatternName(UMLPattern) 例如： GoF::Mediator（UMLP 模式）
16	添加工具箱项后，它将出现在 工具箱项"列表中。您可以选择为工具箱项目添加自定义图标图像。 图标图像必须是 16x16 像素的位图文件；对于透明背景，使用浅灰色 - RGB(192,192,192)。 设置工具箱项目的图标： 1. 在 工具箱项"列表中找到该项目，然后在 工具箱图标"列中单击。 2. 单击此列中的  按钮。将显示 选择工具箱图标"对话框。 3. 找到图像文件并单击 打开"按钮。
17	对要添加到工具箱页面的每个项目重复步骤 13 到 16。 要删除工具箱项目，请在 工具箱项"列表中选择它，然后单击删除按钮。 添加所有适当的工具箱项目后，单击确定按钮。构造型A元素添加到您的工具箱配置文件图中。
18	对要包含在工具箱中的每个工具箱页面重复步骤 8 到 17。
19	通过单击打开图表的背景并选择任一功能区选项来保存工具箱配置文件： • 设计>图表>管理>另存为配置文件或 • 特定>技术> 发布技术> 将图表发布为UML配置文件
20	将配置文件合并到MDG 技术中。

注记

- 可以A右键单击工具箱配置文件图表上的适当构造型元素并选择 使用配置文件帮助器编辑"选项来修改工具箱页面
- 为工具箱页面指定名称时，请注意 无素"是保留字；如果使用了 无素"一词，则不会出现在相应工具箱页面的标题栏中
- 序列中工具箱页面的工具箱由配置文件图或配置文件包序列它们的构造型元素的顺序决定；如果您从以下位置创建并保存配置文件：
-图表，工具箱页面序列是由构造型元素的Z-order决定的
该图 -构造型元素的 Z 序数越高，越往下

工具箱其工具箱页面放置；如果你改变构造型元素的 Z 顺序
图它改变了元素页面在工具箱中的位置
-包在浏览器窗口中，工具箱的页面序列由浏览器的列表顺序决定
包中的构造型元素 - 第一个列出元素的工具箱页面位于顶部
工具箱；如果你重新排序浏览器窗口中的元素，你会产生相同的结果
工具箱中页面的重新排序

使用配置文件Helpers 创建隐藏的子菜单

当您创建工具箱项目时，其中一些可能非常相似，因为它们基于相同类型的元类。例如，有许多不同类型的行动元素。您可以创建一个 基础”工具箱项目，并从子菜单中提供一个变体选项，而不是用每个变体填充工具箱页面，当基础项目被拖到图表上时会显示该子菜单。

定义一个隐藏的子菜单

节	行动
1	如果您尚未这样做，请创建并显示您将用于定义工具箱，如使用配置文件帮助器创建工具箱 *Profiles*文件的步骤1至 6 中所述。
2	将 配置文件助手”工具箱页面中的 添加工具箱页面”项拖到图表上。 将显示 添加工具箱页面”对话框。
3	在 名称”字段中，输入子菜单工具箱项的名称。
4	在这种情况下， 工具提示”字段可以留空。
5	选中 隐藏”复选框。 'Images Only'、'公共' 和 'Is Collapsed' 复选框应保持未选中状态。
6	选择 隐藏”复选框后， 图标”字段应变为活动状态。您可以选择为子菜单工具箱项添加自定义图标图像。 图标图像必须是 16x16 像素的位图文件；对于透明背景，使用浅灰色 - RGB(192,192,192)。 要设置子菜单工具箱项目的图标，请单击 图标”字段右侧的文件夹图标。选择图像文件并单击 打开”按钮。
7	您现在可以将元素和连接器等项目添加到子菜单。 单击添加按钮右侧的向下箭头，然后选择以下选项之一： • '添加构造型'：为当前模型的UML配置文件中定义的构造型添加工具箱项；此配置文件必须包含在MDG 技术中的工具箱配置文件中 选择此选项后，将显示 选择配置文件元素”对话框；使用它来选择要添加的构造型 • '添加内置类型'： -元素：为UML元素类型添加一个工具箱项 选择此选项后，将显示 创建新工具箱项”对话框；在 别名”字段中，键入出现在工具箱项目上的标签，然后单击确定按钮 然后显示 选择元类”对话框；选择要添加到您的工具箱的UML元素类型，然后点击确定按钮 -连接器：为UML连接器类型添加工具箱项 选择此选项后，将显示 创建新工具箱项”对话框；在 别名”字段中，键入出现在工具箱项目上的标签，然后单击确定按钮 然后显示 选择元类”对话框；选择要添加到您的工具箱的UML连接器类型，然后点击确定按钮 • '添加隐藏工具箱'：添加隐藏工具箱子菜单项；创建 隐藏工具箱”子菜单本身时不要使用此选项 • '加新项目'：添加一个工具箱项目，只有一个别名 仅此选项不会创建功能工具箱项；以这种方式添加的工具箱项必须稍后通过 工具箱项”列表进行修改

	单击 添加”按钮，而不是单击下拉箭头，与选择 添加构造型”选项相同。
8	（可选）添加工具箱项后，它会出现在 工具箱项”列表中，您可以为该项添加自定义图标图像。图标图像必须是 16x16 像素的位图文件；对于透明背景，使用浅灰色 - RGB(192,192,192)。 要设置工具箱项目的图标，请在 工具箱项”列表中找到该项目，然后单击 工具箱图标”列。单击此列中的 ... 按钮。将显示 选择工具箱图标”对话框。找到图像文件并单击 打开”按钮。
9	对要添加到子菜单的每个项目重复步骤 7 和 8。 要删除工具箱项目，请从 工具箱项”列表中选择它，然后单击删除按钮。 添加所有适当的子菜单项后，单击确定按钮。构造型A元素添加到您的工具箱配置文件图中。
10	对要创建的每个工具箱子菜单重复步骤 2 到 9。
11	之前创建的子菜单现在可以作为项目包含在工具箱页面中。

注记

- A通过右键单击工具箱配置文件图表上的适当构造型元素并选择 使用配置文件助手编辑”选项来修改子菜单

创建MDG 技术文件

创建MDG 技术文件时，可以包含范围广泛的功能和工具，包括UML Profiles 、代码模块、脚本、模式、图像、标记值类型、报告模板、链接文档模板和工具箱建造。使用MDG 技术创建向导将这些按逻辑序列放入MDG 技术文件很容易。

访问

功能区	特定>技术>发布技术>生成MDG 技术

创建MDG 技术文件

节	描述
1	选择 "生成MDG 技术文件"选项。 显示MDG 技术创建向导屏幕。
2	单击下一步按钮。 MDG 技术向导提示您： • 基于新的MDG 技术选择 (MTS) 文件创建MDG 技术文件 • 基于现有 MTS 文件创建MDG 技术文件，或 • 不使用任何 MTS 文件 MTS 文件存储您在创建MDG 技术期间定义的选定选项；如果您使用 MTS 文件，您可以稍后修改它以添加或删除MDG 技术中的特定项目，这是推荐的过程。
3	选择适当的 MTS 文件选项。 单击下一步按钮。 如果您选择了 MTS 文件， MDG 技术向导会提示您将更改保存在现有 MTS 文件中或保存到新的 MTS 文件中；这使您能够在现有 MTS 文件的基础上创建修改，同时保留原始文件。
4	如有必要，输入或浏览所需的文件路径和名称。 单击下一步按钮。 将显示 "MDG 技术向导 - 创建"对话框。
5	完全此屏幕上的字段： • 文件名 -类型或选择MDG 技术文件的路径和文件名；此文件的文件扩展名为 .xml • ID -类型MDG 技术文件的唯一引用，最长 12 个字符 • 版本类型MDG 技术文件的版本号 • 图标——（可选）类型或选择包含技术图标的图形文件的路径和文件名；该图标是 16 位或 24 位颜色深度、16x16 位图图像，显示在 "MDG 技术"对话框左侧的技术列表中 • Logo - （可选）类型或选择包含技术标志的图形文件的路径和文件名；徽标是 16 位或 24 位色深、64x64 或 100x100 位图图像，显示在 "MDG 技术"对话框右上角的显示窗格中 • URL - （可选）如果您有任何网站产品信息可能对使用此技术的用户有所帮助，请在此字段中

	键入或粘贴 URL • 支持 - （可选）如果您有任何可能对本技术用户有帮助的基于 Web 或其他支持功能，请在此字段中键入或粘贴联系地址 • 注记-类型MDG 技术功能的简短说明
6	单击下一步按钮。 MDG 技术向导 - 内容屏幕显示。
7	选中要包含在MDG 技术文件中的每个项目的复选框。 当您选择了要包含的所有项目的复选框后，单击 "下一步"按钮。 每个选择都会运行特定的对话框，以定义要包含在MDG 技术中的特定项目。
8	完成响应您的选择而显示的对话框，当所有对话框都完成后，单击下一步按钮。 显示 "MDG 技术向导 - 完成"屏幕，提供有关MDG 技术文件中包含的项目的信息。
9	如果您使用过 MTS 文件并想要更新它，请选中 "保存到 MTS"复选框。
10	如果您对所选择的项目感到满意，请单击 "完成"按钮。 如果需要，您现在可以编辑 MTS 文件以添加更多项目，例如： • 验证模型配置 • 模型生成器模板(模式) 编辑完 MTS 文件并重新生成技术(.xml) 文件后，您可以添加另一个 "脚本"部分以包含包XMI导出和/或导入脚本。保存已编辑的技术文件。 要使Enterprise Architect模型可以访问MDG 技术.xml 文件，您必须将技术文件路径添加到 "MDG技术-高级"对话框（通过单击 "MDG技术"对话框上的高级按钮，通过 "特定>技术>管理技术"功能区选项进行访问）。

添加配置文件

创建MDG 技术文件时，您可以包含一个或多个符合UML 2.5 标准的配置文件，这些Profiles已定义用于创建新类型的模型元素。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将Profiles添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在该节中选中 "Profiles "复选框。 将显示 "MDG 技术向导 -配置文件文件选择"页面。
2	在 "目录"字段中，导航到包含所需配置文件或Profiles的目录。 配置文件文件会自动列在 "可用文件"面板中。
3	要单独选择每个所需的配置文件，请突出显示 "可用文件"列表中的配置文件，然后单击 --> 按钮。 文件名显示在 "选定文件"列表中。 或者： 要选择每个可用的配置文件，请单击 -->> 按钮，然后通过选择它并单击 <-- 按钮返回每个您不想要的配置文件。 • Profiles在此对话框中选择工具箱Profiles或图表；这会在 .mts 文件中生成冲突的命令 • 确保您确实包含您的UML Profiles
4	单击下一步按钮继续。

添加模式

创建MDG 技术文件时，您可以在浏览器窗口的 资源”选项卡中包含您想要提供的特殊设计模式，如果您愿意，也可以在技术工具箱页面中。您之前已经发布过这些模式。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将设计模式添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤进行操作，直到第 6 节（包括第 6节），在该节中选中 模式”复选框。 显示 MDG 技术向导-模式文件”选择页面。
2	在 目录”字段中，导航到包含所需模式XML 文件的目录。 模式文件会自动列在 可用文件”面板中。
3	要单独选择每个所需模式，请在 可用文件”列表中突出显示该模式，然后单击 --> 按钮。 文件名显示在 选定文件”列表中。 或者，要选择所有可用模式，请单击 -->> 按钮，然后通过选择它并单击 <-- 按钮返回每个您不想要的模式。
4	单击下一步按钮继续。

添加一个图表配置文件

创建MDG 技术文件时，您可以包含已定义的图表Profiles以生成新类型的图表。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将图表文件到MDG 技术Profiles

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在该节中选择 图表类型”复选框。 ' MDG 技术向导 -图表' 页面显示。
2	在 目录”字段中，导航到包含所需图表的Profiles 。 目录中的Profiles会自动列在 可用文件”面板中。
3	要单独选择每个所需的图表配置文件，请在 可用文件”列表中突出显示文件名，然后单击 --> 按钮。 文件名显示在 选定文件”列表中。 或者，要选择所有可用的Profiles Profiles如果它们都是图表），请单击 -->> 按钮，然后通过选择它并单击 <-- 按钮来返回您不想要的每一个。
4	单击下一步按钮继续。

添加工具箱配置文件

创建MDG 技术文件时，您可以包含您创建的图表工具箱定义，以提供工具箱页面以支持自定义图表。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将工具箱Profiles添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中 "工具箱"复选框。 将显示 "MDG 技术向导 - 工具箱"页面。
2	在 "目录"字段中，导航到包含所需工具箱Profiles的目录。 配置文件文件会自动列在 "可用文件"面板中。
3	要单独选择每个所需工具箱配置文件，请在 "可用文件"列表中突出显示文件名，然后单击 --> 按钮。 文件名显示在 "选定文件"列表中。 或者，要选择所有可用的Profiles （如果它们都是工具箱Profiles ），请单击 -->> 按钮，然后通过选择它并单击 <-- 按钮返回您不想要的每一个。
4	单击下一步按钮继续。

添加标记值类型

在创建MDG 技术文件时，可以包含标记值类型，技术用户可以从中创建特定领域的标记值。您可以使用两种方法：

- 在配置文件图表上的数据类型元素中定义标记值类型，如使用预定义标签类型帮助主题（推荐）中所述，或者
- 直接在MDG 技术向导中添加标记值类型，如此处所述。

访问

功能区	特定>技术>发布技术>生成MDG 技术

在MDG 技术标记值类型技术文件中添加值类型

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在此选中 标记值类型”复选框。 显示 ‘MDG 技术向导 -标记值类型”页面。
2	要单独选择每个所需的标记值类型，请在 可用标记值”列表中突出显示名称，然后单击--> 按钮。 名称显示在 已选择的标记值”列表中，标记值类型的名称、描述和注记显示在页面底部的面板中。 或者，要选择所有可用的标记值类型，请单击 -->> 按钮，然后通过选择并单击 <-- 按钮返回您不想要的每一个。
3	单击下一步按钮继续。

添加代码模块

创建MDG 技术文件时，您可以包含已为其设置代码模板和数据类型的代码模块。这些模块可以用于修改系统默认语言，也可以用于您使用代码模板和代码模板编辑器自己定义的语言。在编辑器中为新语言设置代码模板之前，您必须为该语言定义至少一种数据类型。您还可以指定语言的代码选项，这是数据类型或代码模板未解决的附加设置；它们保存在一个 XML 文档中，该文档包含在模块的MDG 技术文件中。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将代码模块添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中 代码模块”复选框。 ' MDG 技术向导 - 代码模块' 页面显示，列出了您当前项目中定义的代码模块。
2	单击要包含在技术中的每个代码模块的复选框（ 产品”、 数据类型”、 代码语法”和 代码模板”）。
3	如果您已为选定模块创建了代码选项 XML 文档，请单击该模块的 代码选项”列中的  按钮。将显示A浏览器，您可以通过它找到并选择 XML 文档。
4	单击下一步按钮继续。

定义代码选项

在修改现有编程语言的代码生成模板或定义新的编程语言时，还有一些仅在构建MDG 技术时可用的附加选项。这些附加选项会影响Enterprise Architect如何处理该语言的代码生成和逆向工程。这些选项是使用 XML 文件指定的，该文件是使用您首选的文本编辑器创建的。

XML 文档中的根节点名为根。子节点被命名为 CodeOption。每个 CodeOption 包含一个 name 属性，对应于可用代码选项之一的名称。每个节点的主体都包含选项值。例如：

<代码选项>

<CodeOption name="DefaultExtension">.h</CodeOption>

<CodeOption name="HasImplementation">true</CodeOption>

<CodeOption name="ImplementationExtension">.cpp</CodeOption>

<CodeOption name="Editor">C:\Window\notepad.exe</窗口>

</代码选项>

支持的代码选项

代码选项	描述
构造函数名称	用作构造函数的函数的名称。由 classHasConstructor 代码模板宏使用。
CopyConstructorName	用作复制构造函数的函数的名称。由 classHasCopyConstructor 代码模板宏使用。
默认扩展	生成代码时的默认扩展名。
默认源目录	Enterprise Architect生成新文件的默认路径。
析构函数名称	用作析构函数的函数的名称。由 classHasDestructor 代码模板宏使用。
编辑	用于编辑该语言源的外部编辑器。
有实现	指定此语言的代码生成是否同时生成源文件和实现文件。
实施扩展	Enterprise Architect用来生成实现文件的扩展。
实施路径	从源文件生成实现文件的相对路径。
包路径分隔符	使用 packagePath 宏与代码模板时用于分隔包名的分隔符。

注记

- 一旦可以在模型中使用一种语言（通过导入和激活MDG 技术），您可以在 首选项"对话框中显示和编辑代码选项（“>开始外观>首选项>首选项”）

添加数据库数据类型

在创建MDG 技术文件时，您可以包含 DDL 文件，这些文件为您打算通过您的技术使用并已为其设置数据类型的每个数据库定义数据库数据类型。在为新的数据库类型设置 DDL 文件之前，您必须为该数据库定义至少一种数据类型。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将数据库数据类型添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 7 节（包括第 7节），在该节中选中 "DDL 文件"复选框。"MDG 技术向导 - DDL 文件"页面显示，列出了当前项目中可用的数据库类型。
2	单击要在技术中包含 DDL 文件的每种数据库类型的复选框。如果模型中存在 DDL 的数据类型，也选择相应的 "数据类型"复选框。
3	单击下一步按钮继续。

添加 MDA 转换

创建MDG 技术文件时，您可以包含您在模型中创建或修改的任何 MDA变换模板，并且您希望将其部署为技术的一部分。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将 MDA变换添加到模板MDG 技术文件中

节	描述
1	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中 "MDA 转换"复选框。 ' MDG 技术向导 - 转换模块' 页面显示，列出了您系统上可用的 MDA 转换模板。
2	单击要添加到MDG 技术中的每个转换模板名称旁边的复选框。
3	单击下一步按钮继续。

添加文档报告模板

创建MDG 技术文件时，您可以包含用户定义的文档报告模板。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将报告模板添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在该节中选中“RTF模板”复选框。将显示“MDG 技术向导 - RTF 报告模板”对话框。
2	对于当前模型中可用的每个所需的用户定义报告模板，选中模板名称旁边的复选框。
3	单击下一步按钮继续。

添加链接文档模板

创建MDG 技术文件时，您可以包含链接文档模板。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将链接文档模板添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在该节中选中 "链接文档模板"复选框。 将显示 "MDG 技术向导 - 链接文档模板"对话框。
2	对于当前模型中可用的每个所需文档模板，选择模板名称旁边的复选框。
3	单击下一步按钮继续。

添加图片

创建MDG 技术文件时，您可以合并要在部署该技术的所有模型中使用的图像。这些图像必须已经存在于正在开发该技术的模型中；您可以使用图像管理器上的添加加新按钮将图像导入此模型。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将图像添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中 图像”复选框。 将显示 "MDG 技术向导-图像选择”对话框。
2	对于当前模型中可用的每个所需模型图像，选中图像名称旁边的复选框。 当您选择复选框时，每个图像A预览都会显示在对话框的右侧。
3	单击下一步按钮继续。

添加脚本

创建MDG 技术文件时，您可以包含在模型中创建的脚本。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将脚本添加到MDG 技术文件

节	描述
1	按照创建*MDG* 技术主题中的步骤直到第 6 节（包括第 6节），在该节中选中 脚本”复选框。 将显示 “MDG 技术向导 - 脚本”对话框。
2	对于当前模型中可用的每个所需脚本，选择脚本名称旁边的复选框。
3	单击下一步按钮继续。

注记

- 此功能在Enterprise Architect的企业统一版和终极版中可用

添加工作区布局

在开发MDG 技术文件时，您可以包含用户定义的工作空间布局。工作空间布局是工具栏和窗口的排列，适合于需求管理和代码工程等工作领域。工作区布局自动打开并组织所有工具以适应您使用系统的方式。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将工作区布局添加到MDG 技术文件中

节	描述
1	在您的模型中，创建要包含在您的技术中的工作区布局。
2	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中“工作区布局”复选框。 将显示“MDG 技术向导 -工作区布局”对话框，列出可供您使用的用户定义的工作区布局。
3	对于您要合并到技术中的每个工作空间布局，选择布局名称旁边的复选框。
4	单击下一步按钮继续。

添加模型视图

在开发MDG 技术文件时，您可以包含用户定义的模型视图。模型视图基于从模型中提取特定信息的搜索，以提供模型的不同视角和 切入点”。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将模型视图添加到MDG 技术文件

节	描述
1	在您的模型中，创建您想要包含在您的技术中的模型视图。
2	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在该节中选中 模型视图”复选框。 ' MDG 技术向导-模型视图'对话框显示，列出当前模型中可用的用户定义视图。
3	对于要合并到技术中的每个模型视图，选择视图名称旁边的复选框。
4	单击下一步按钮继续。

注记

- 技术视图不存储收藏包，只存储视图
- 如果您包含运行您定义的搜索的模型视图，您还必须在您的MDG 技术中包含这些搜索

添加模型搜索

在开发MDG 技术文件时，您可以包含用户定义的模型搜索。您可以使用模型搜索功能设置这些搜索，在#
, 在查询生成器中或作为插件
，然后将它们链接到您的MDG 技术中。

访问

功能区	特定>技术>发布技术>生成MDG 技术

将模型搜索添加到MDG 技术文件

节	描述
1	在您的模型中，创建您想要包含在您的技术中的模型搜索。
2	按照创建*MDG* 技术主题中的步骤直至第 6 节（包括第 6节），在此您选择 模型搜索”复选框。显示 "MDG 技术向导 -模型搜索”对话框，列出当前模型中可用的用户定义搜索。
3	对于您要合并到技术中的每个模型搜索，选择搜索名称旁边的复选框。
4	单击下一步按钮继续。

注记

- 如果您使用自定义#
  搜索， #
  必须包含 ea_guid AS CLASSGUID 和object类型
- 如果您包含运行您定义的搜索的模型视图，您还必须在您的MDG 技术中包含这些搜索

使用 MTS 文件

当您使用MDG 技术向导创建MDG 技术文件时，您可以选择存储您在MDG 技术选择 (.mts) 文件中定义的所有选项和结构。这将捕获您在技术向导中输入的所有信息，因此您不必再次输入。如果您使用 .mts 文件，则可以随后对其进行编辑以更改您在生成技术文件时选择的特征，并添加或删除其他高级特征。

访问

功能区	特定>技术>发布技术>生成MDG 技术

管理 .MTS 文件

行动	描述
创建一个 .MTS文件	要创建 .mts 文件，请启动并通过MDG 技术向导工作；在第二页上，选择 创建新的 MTS 文件”选项。
.MTS文件的高级选项	完成MDG 技术向导并设置 .mts 文件后，您可以单独添加： • 模型验证配置 • 模板模型 首先为模型验证配置和模型模板定义 XMI，然后在文本编辑器中打开 .mts 文件并复制 </ MDG .Selections> 行之前的验证和/或模板描述。 保存 .mts 文件。
更新MDG 技术	再次启动MDG 技术向导，但这次在第二页上选择 打开现有 MTS 文件”选项并指定您一直在处理的 .mts 文件的文件路径。 单击下一步按钮，直到向导完成；您的MDG 技术.xml 文件已更新。

注记

- 使用向导和 .mts 文件创建您的MDG 技术后，您可以通过技术.xml 文件添加导入和导出脚本

创建工具箱Profiles

作为MDG 技术的一项功能，您可能希望提供图表工具箱，以访问您在该技术中创建的任何元素和连接器。您在特定的Profiles中定义这些工具箱页面，每个配置文件定义元素和连接器工具箱页面，这些页面为图表类型打开或可选择。

创建自定义工具箱

节	行动
1	创建一组工具箱Profiles ，其中包含生成工具箱页面所需的定义。
2	酌情编辑定义以： ● 包括隐藏的子菜单 ● 覆盖默认工具箱 ● 更改工具箱项目的默认图标
3	创建一个 .mts 文件，其中包含有关如何构建您的MDG 技术的说明，并在该技术中包含工具箱 Profiles 。

创建工具箱Profiles

在MDG 技术中，您可以创建多个工具箱Profiles 。每个工具箱配置文件都包含确定打开图表时显示哪些页面的定义，或者通过从工具箱中的图表功能中进行工具箱，或者通过打开或创建链接到工具箱配置文件。

工具箱配置文件Errors

当您使用图表MDG 技术中定义的工具箱时，可能会显示某些错误消息。此表解释了这些错误消息的含义。

信息	意义
缺少基本类型 <名称>	例如：'缺少基本类型：'SysML1.3::块' 不扩展' UML ::状态' 基本类型要么丢失，要么与扩展元素类型不对应（在示例中，SysML::块实际上扩展了UML ::类）。
未找到 ID <名称> 的配置文件	此错误消息可能意味着无法找到配置文件，或者包含配置文件的MDG 技术已被禁用（使用 专业化>技术>管理”进行检查）。
在配置文件 <name> 中找不到构造型 <name>	例如：在配置文件 “SysML 1 .2”中没有找到原型 “ProxyPort”。 此消息表明所需的构造型与它应该包含的配置文件不匹配。在示例中，SysML1.2 没有 ProxyPorts，因此构造型应该是 “FlowPort”或配置文件 “SysML 1 ” .3'。
未知/非法基类型：<name>	显示此消息可能有多种原因。例如： • 未知/非法基类型： UML ::Capability - 显示是因为没有 “Capability”这样的UML元类 • 未知/非法基本类型：SysML 1 .3::块显示是因为您试图从另一个配置文件扩展构造型，在本例中是来自 SysML 1 .3 配置文件的 <<Block>>；您必须扩展与您专门扩展的构造型相同的东西（在本例中为 “UML ::类”）

创建工具箱配置文件

节	行动
1	在配置文件包中，创建一个具有适当名称的类图，以便您以后参考它，例如 MyClassDiagram。
2	双击图表背景以显示图表 属性”对话框，并在 注记”字段中为图表指定别名和以下格式的描述： Alias=MyClass;Notes=类图的结构元素；
3	在图表上，创建一个名为 ToolboxPage 的元类元素。
4	为要在您的工具箱中创建的每个工具箱页面创建构造型元素，例如 MyClassElements 和 MyClassRelationships。 双击每个元素以显示 属性”对话框，并在 别名”字段中键入要在工具箱页面的标题栏中显示的文本，例如我的类或我的类关系。 在每个元素的 注记”字段中，输入相应工具箱页面的工具提示；例如，'元素类图表' 或 '关系类图表'。

	在每个构造型元素和 ToolboxPage 元类元素之间创建扩展连接器。
5	在每个构造型元素中，按 F9 并为该元素定义的页面中的每个工具箱项创建一个属性。 每个属性的名称是要删除的元素或连接器的名称，包括元素的命名空间；例如， UML ::包、UML ::类和UML ::接口。您可能不想在您的工具箱中显示包含诸如UML ::包或UML ::类之类的文本的名称，因此请给属性一个 初始值”，例如包或类。 工具箱项目的显示顺序与其在序列中的属性元素，因此请使用特征窗口 属性”页面中的属性排序选项来定义工具箱页面中图标的顺序。 在来自您自己的技术的元素或连接器的属性名称中，使用您的配置文件名称作为命名空间，然后在项目名称后面加上您正在扩展的元素或连接器类型，在括号中（以识别Enterprise Architect要创建什么类型的object ）；例如，一个 SysML块元素会显示为： SysML::块(UML ::类) 许多元素和连接器可以扩展以在工具箱中使用。
6	要定义工具箱项以将设计模式拖放到图表上，请将该属性命名为： 我的技术ID::我的模式（UMLPattern） 'MyTechnologyID' 是技术的 ID（不是名称），'MyPattern' 是要放弃的模式的名称；例如： 总线框架::Builder(UMLPattern) 如果您想避免显示 Add模式”对话框，请将（UMLPattern）替换为（UMLPatternSilent）。 要在自定义工具箱（如 GoF模式）中定义基于模型的模式，请创建一个具有以下格式名称的属性： 模式类别::模式名称（UMLPattern） 例如： GoF::Mediator（UMLPattern）
7	定义您需要修改工具箱页面显示的任何属性，例如工具箱页面是最小化还是不显示项目名称（标签）。
8	要保存工具箱配置文件，请单击打开图表的背景并选择任一功能区选项： • 设计>图表>管理>另存为配置文件或 • 特定>技术> 发布技术> 将图表发布为UML配置文件

注记

- 为工具箱页面分配别名时， 无素”是保留字；如果使用了 无素”一词，则不会出现在相应工具箱页面的标题栏中
- 每个配置文件元素合并到一个MDG工具箱页面中，启用一个上下文菜单选项来同步从它创建的所有对象的标记值和约束
- 序列中的工具箱页面由配置文件图或配置文件包序列的构造型元素的工具箱决定；如果您从以下位置创建并保存配置文件：
  -图表，工具箱页面序列由构造型元素的Z-order决定
  在图中 -构造型元素的 Z 序号越低（越接近1 ）（越接近
  它是图表的 表面”），工具箱的工具箱页面放置得越往下；
  如果改变图中构造型元素的 Z 顺序，它会改变构造型元素的位置
  工具箱上的元素页面
  -包在浏览器窗口中，工具箱的页面序列由浏览器的列表顺序决定
  包中的构造型元素——第一个列出元素的工具箱页面位于
  工具箱顶部；如果你重新排序浏览器窗口中的元素，你会产生相同的结果
  工具箱中页面的重新排序

工具箱页面属性

当您创建构造型元素来定义MDG 技术中的工具箱页面时，您可以添加许多属性来控制页面本身在图表工具箱中的行为方式。构造型元素可以是扩展 ToolboxPage 元类的几个元素之一。

您可以添加的属性包括：

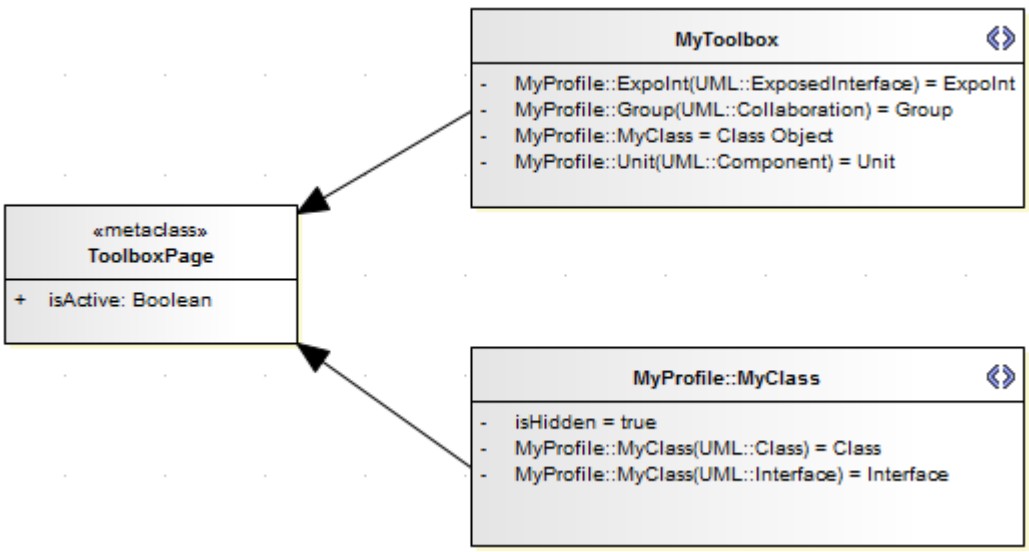
- 图标 - 请参阅Assign Icons To Toolbox Items
- ImagesOnly - 如果将 初始值”设置为 true，则工具箱页面显示时图标旁边不会显示文本标签
- isCollapsed - 如果将 初始值”设置为 true，则工具箱页面最初处于最小化状态
- isCommon - 如果将初始值设置为 true，则只要来自相同技术的另一个工具箱页面是当前工具箱页面，就会显示此公共工具箱页面
- isHidden - 请参阅Create Hidden Sub-Menus

创建隐藏的子菜单

当您在工具箱页面上创建项目时，其中一些可能非常相似并且基于相同类型的元类。例如，有许多不同类型的活动元素，在行动2.0 中，您可以创建每种类型的事件元素，无论是独立的还是边缘安装在另一个元素上。您可以创建一个 基础"工具箱项目并从子菜单中提供变体选项，而不是用每个变体填充工具箱页面，当基础项目被拖到图表上时会显示该变体，但在其他情况下会被隐藏。这种技术对于可以应用于多个元类的 消除歧义"构造型非常有用。

在子菜单中，您只定义变体类型（对于行动元素列表）。但是，如果变体还为其定义了 ToolboxItemImage，则该图标将显示在子菜单中的变体名称旁边（对于 BPMN 2.0事件）。您还可以使用此方法专门定义将应用于子菜单选项的图标。

定义一个隐藏的子菜单

节	行动
1	在与 ToolboxPage 元类相同的图表上创建构造型元素，名称以配置文件名为前缀（这是强制性的）。例如： 我的个人资料::我的班级 该名称不得与任何其他配置文件中存在的任何外部构造型的名称匹配。 子菜单元素可以有一个别名。
2	在这个子菜单构造型元素中，创建属性 isHidden ，其初始值为True 。 对于每个子菜单项，添加一个属性来标识该项。将 初始值"设置为要在菜单中显示的名称。例如，如果 «MyClass» 构造型可以应用于UML类或UML接口，这两个选项的属性将是： MyProfile::MyClass(UML ::类) 初始值 =类 MyProfile::MyClass(UML ::接口) 初始值=接口
3	创建第二个构造型元素并定义一个与子菜单构造型元素同名的属性，并使用要在工具箱项中显示的文本的初始值。例如： MyProfile::MyClass =类物件 像工具箱一样为工具箱中的其余项目定义附加属性。
4	在每个构造型元素和 ToolboxPage 元类元素之间创建 <<Extension>>关系，如图所示。 

	使用该配置文件时，当类物件项从工具箱拖到图表上时，隐藏菜单显示类或接口选择；在选择时，元素被放到图表上。
5	如果没有从现有定义中为工具箱项目分配图标，并且您想要显示一个，请将图像定义为 ToolboxItemImage 图标。

将图标分配给工具箱项

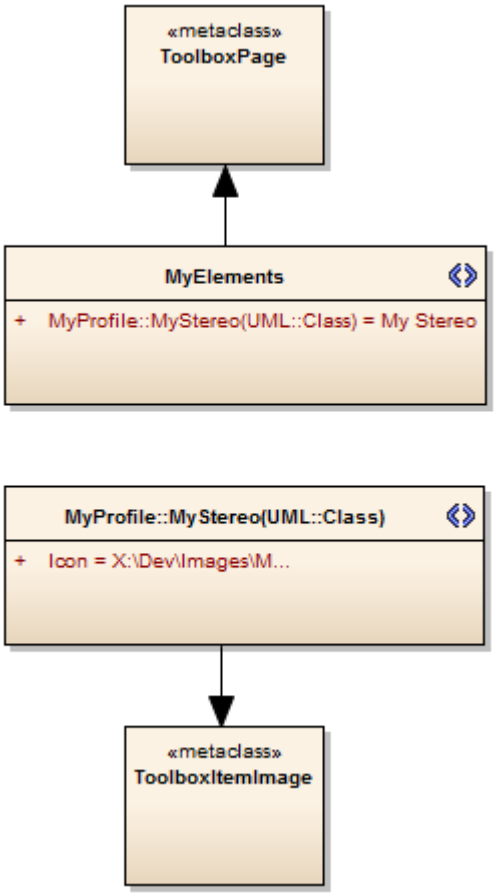
当您创建一个原型模型元素来定义一个在图表工具箱页面中表示的元素或连接器时，您可以定义显示在浏览器窗口中的元素名称和工具箱中的元素或连接器类型的图像工具箱页，通过将特殊属性图标分配给构造型元素。

可以通过扩展 ToolboxItemImage 元类来覆盖或替换工具箱项的此图像定义，该过程通常是可选的。但是，如果你想在隐藏的子菜单上显示一个图标，你必须使用这个方法；系统选择 ToolboxItemImage 定义作为隐藏菜单项的图标。

如果您不使用图标属性或 ToolboxItemImage 元类来定义工具箱图标，则图像默认为用于已扩展的标准UML模型元素的图像。如果没有这样的图像，则图标使用系统默认的通用 "工具箱项"图像。

扩展 **ToolboxItemImage** 元类

节	行动
1	在与工具箱项目相同的工具箱配置文件中创建一个新的构造型元素。
2	为构造型元素赋予与为其分配图像的元素相同的名称；例如： MyProfile::MyStereo(UML ::类)
3	为构造型元素赋予特殊属性 Icon with Initial Value 设置为要使用的图像的完整路径和文件名。 图标图像是一个 16x16 像素的位图文件；对于透明背景，使用浅灰色 - RGB(192,192,192)。
4	创建一个名为 ToolboxItemImage 的元类元素，并创建从构造型元素到此元类的扩展关联。


«metaclass»
ToolboxPage
MyElements
+ MyProfile::MyStereo(UML::Class) = My Stereo
MyProfile::MyStereo(UML::Class)
+ Icon = X:\Dev\Images\M...
«metaclass»
ToolboxItemImage

覆盖默认工具箱

当您创建一种内置图表类型的图表时，系统会根据相应的默认工具箱配置文件显示一个图表工具箱页面。如果您自定义了图表类型，它仍然会为您扩展的基本图表类型应用系统默认工具箱页面，除非您使用您自己创建的替代工具箱页面覆盖该默认设置。例如，您可能有自己的UML ::类工具箱页面版本，您希望在每次打开类图时显示，当您的技术处于活动状态时。

注记要让默认工具箱页面被您的MDG 技术中的自定义工具箱页面覆盖， MDG 技术必须设置为 活动”。 (特定>技术>管理技术”，然后选中您的MDG 技术名称对应的复选框，然后单击 设置活动”按钮。)

访问

要将系统默认工具箱替换为您自己的工具箱：

使用此处概述的方法之一显示工具箱配置文件图的 属性”对话框，并显示 常规”选项卡。

然后，在 注记”字段中键入 RedefinedToolbox 子句。

例如：

类类的结构元素；

这表明此配置文件定义的工具箱将系统工具箱UML ::类替换为所有UML类图的默认工具箱。

功能区	设计>图表>管理>属性>常规
上下文菜单	工具箱配置文件图表右击|属性|一般的

可被覆盖的系统默认工具箱页面名称

- UML ::活动
- UML ::类
- UML ::通讯
- UML ::部件
- UML ::复合
- UML ::部署
- UML交互
- UML ::元模型
- UML ::物件
- UML ::配置文件
- UML ::状态
- UML ::时序
- UML ::用例
- 扩展::分析
- 扩展::自定义
- 扩展::数据建模
- 扩展::维护
- 扩展::需求

- 扩展::用户界面
- 扩展::WSDL
- 扩展::XMLSchema

工具箱页面中使用的元素

当您为您的MDG 技术创建工具箱页面时，您可以合并标准UML元素和通过扩展UML元素创建的新元素。您在工具箱配置文件中定义要使用的元素。该表列出了您用来识别的名称：

- 要包含在工具箱页面中的标准元素或
- 您正在扩展的标准元素以定义要包含在工具箱页面中的新元素

您在工具箱页面构造型元素中列出的每个名称前面都有命名空间UML ::。括号中的文本表示默认工具箱页面中显示的标签名称，这与UML :: 语句文本有任何不同。

工具箱页面定义的元素名称

- 行动
- 行动销
- 活动
- ActivityFinal（终点）
- ActivityInitial（初始）
- 活动参数
- ActivityPartition (分区)
- 活动区域 (区域)
- 参与者
- 工件
- 关联元素（关联）
- （用于使用边界）
- 节点(中央缓冲区节点)
- 更改
- 选择
- 类
- 协作
- CollaborationOccurrence (协作应用)
- 注解(注记)
- 部件
- 约束
- 数据存储
- 决策
- 部署规范（部署部署规范）
- 设备
- 图表（图例）
- 图表（图表注记）
- DocumentArtifact（文档工件或文档）
- 实体（资料）
- 实体对象（实体）
- 入口点（入口）

- 枚举
- 异常处理程序（异常）
- 执行环境（执行环境）
- 扩展区域
- 出口点（出口）
- 特征
- FinalState（终点）
- FlowFinalNode（流终点）
- ForkJoinH（分叉/汇合-水平）
- ForkJoinV (分叉/汇合- Vertical)
- 门（图表门）
- GUIElement (用户界面控件)
- HistoryState（历史）
- 超链接
- InformationItem（信息项）
- 初始状态（初始）
- 交互
- 交互片段（片段）
- InteractionState（状态/延续）
- 接口
- 可中断活动区域
- 问题
- 连接点
- 生命线
- 合并节点（合并）
- MessageEndPoint（端点或信息端点）
- MessageLabel (信息标签)
- 元类
- 节点
- 物件
- 对象边界（边界）
- 对象控件（控件）
- 对象实体（实体）
- 包
- 包装组件
- 部件
- 端口
- 原始
- 原始类型
- 进程
- 配置文件
- 提供接口（曝露接口）

- 接收事件（接收）
- 需求
- 边界（存在）
- 控件(控件)
- 健壮实体（实体）
- 屏幕
- 发送事件（发送）
- 序列边界（边界）
- 序列控制（控件）
- SequenceEntity (实体)
- 信号
- 状态
- 状态机（状态机）
- 状态生命线(状态生命线)
- 构造型
- 结构化活动（结构活动）
- 同步状态（同步）
- 库表
- 终止
- 测试用例（测试用例）
- 文本
- 用例（用例）
- 边界（存在）
- 价值时间线（价值生命线）

注记

- 您还可以识别标准或扩展的UML连接器以添加到工具箱页面定义中
- 当元素项部署在MDG工具箱页面中时，您还可以同步从它们创建的所有元素的工具标记值和约束

工具箱页面中使用的连接器

当您为您的MDG 技术创建工具箱页面时，您可以合并标准UML连接器和通过扩展UML连接器创建的新连接器。您可以在工具箱配置文件中定义要使用的连接器。工具箱页面定义表的工具箱连接器名称列出了您用于标识的名称：

- 包含在工具箱页面中的标准连接器或
- 您正在扩展的标准连接器以定义要包含在工具箱页面中的新连接器

您在 "工具箱页面构造型元素"中列出的每个名称前面都有命名空间UML ::。括号中的文本表示默认工具箱页面中显示的标签名称，这与UML :: 语句文本有任何不同。

工具箱页面定义的连接器名称

- 抽象
- 聚合（聚合）
- 组装
- 关联(Associate)
- 关联类（关联类）
- 调用(调用)
- CommunicationPath (通讯路径)
- 组合（复合）
- 连接器
- 控制流（控件流）
- 代表
- 依赖
- 部署
- 扩展
- 概括（继承概括
- 信息流（信息流）
- InterruptFlow（中断流）
- 调用
- 显现
- 信息
- 嵌套
- NoteLink（注记连接）
- ObjectFlow (物件流)
- 发生
- PackageImport（包导入）
- PackageMerge（包合并）
- 先于
- ProfileApplication（应用程序）
- 实现（实现或实现）
- 递归

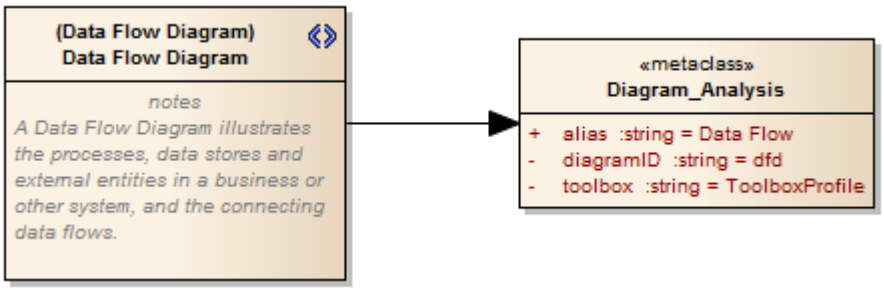
- 重新定义
- 表示
- 代表
- 捆绑(角色捆绑)
- SelfMessage（自我信息）
- 替代
- TagValAssociation（标记值）
- TemplateBinding (模板捆绑)
- TraceLink (跟踪)
- 转移
- UCExtend（扩展）
- UCInclude（包括）
- 用途
- 用例链接（使用）

注记

- 您还可以识别标准或扩展的UML元素以添加到工具箱页面定义中

创建自定义图表Profiles

当您开发MDG 技术时，可以创建扩展图表类型并将它们作为自定义图表Profiles在您的MDG 技术中。例如，您可以创建一个图表图形配置文件分析，将 DFD 图定义为内置图的扩展，如下所示：



创建扩展图表类型

节	行动
1	创建一个配置文件，与要包含的MDG 技术同名；例如，SysML。 此配置文件自动包含一个子类图。根据您打算创建多少新图表类型，您可以定义： • 一个子图上的一种图类型 • 一个图表上有多种图表类型，或 • 在几个图表上分组的几种图表类型 在第三种情况下，创建您需要的任何其他子类图。图表名称不必反映技术名称。
2	打开子类图并创建构造型元素，为其命名自定义图类型；例如，块定义。 同样在构造型元素的 属性”对话框中，在 注记”字段中，键入图表用途的简要说明。 当部署技术并创建此自定义类型的图表时，此描述将显示在 新建图表”对话框的右下角。
3	创建一个元类元素并为其指定所选内置图表类型的名称，并带有前缀 Diagram_。 例如 Diagram_Logical 自定义类图类型，或 Diagram_Use Case 自定义用例图类型。
4	将扩展连接器从构造型元素拖到元类元素。
5	单击 Diagram_xxxx 元类元素，按 F9 并创建以下部分或全部属性，以设置自定义图表类型的属性： • 别名： string =类型（其中类型将出现在图表标题栏上的 图表”一词之前；例如， 块图表”） • diagramID： string = abc（其中 abc 是将出现在图表框架标签中的图表类型） • 工具箱： string = ToolboxName（其中 ToolboxName 是每次打开此类型的图表时自动打开的工具箱工具箱配置文件的限定名称，格式为 TechID::ToolboxName”） • toolboxPage： string = 形式为 PageName=1;”的状态值列表（其中 PageName 是扩展 ToolboxPage 的构造型元素的名称；如果此string不为空，则所有值为 1 ”的工具箱页面将展开，所有其他工具箱页面将折叠） • frameString: string = FrameFormatString （其中 FrameFormatString 是一个string ，包含用于定义框架标题的替换宏，可以带有或不带有附加分隔符（如 ()）；可以使用的宏有： -#DGMALIAS# -#DGMID# -#DGMNAME# -#DGMNAMEFULL#

	- #DGMOWNERNAME# - #DGMOWNERNAMEFULL# - #DGMOWNERTYPE# - #DGMSTEREO# -#DGM类型# • 泳道： string = Lanes=2;Orientation=Horizontal;Lane1=Title1;Lane2=Title2; (其中 Lanes 可以是任意值，但 Lane<n> 值的数量必须等于 Lanes 的值；Orientation 可以省略，在这种情况下泳道默认为垂直) • styleex： string = 一个或多个值范围 • pdata： string = 一个或多个值 • showForeign: string = 1
6	根据您在步骤1中采用的配置文件包组织，以及您是否需要任何其他 Stereotype-Metaclass元素对，在此图或另一个子图上重复步骤 2-5。
7	使用最适合您设置的配置文件包组织的方法将图表保存为图表配置文件。
8	将图形配置文件添加到MDG 技术中使用的图表文件中。

内置图表类型

在自定义Enterprise Architect以更好地满足您的需求时，您可以创建一个配置文件：

- 重新定义在新组合元素下创建的内置子图的类型
- 定义快速链接器菜单提供一种连接器类型的内置图表的类型，或
- 扩展内置图表类型以创建自定义图表类型

在每种情况下，您都需要提供您正在使用的每种内置图表类型的准确名称；这些名称是：

- 活动
- 分析
- 协作
- 部件
- 复合结构
- 风俗
- 部署
- 交互概览
- 逻辑（用于类图）
- 物件
- 包
- 序列
- 状态图
- 定时
- 用例（注记两个词之间的空格）

属性值 - styleex & pdata

创建图表配置文件时，您可以使用 pdata 和 styleex 属性定义使用该配置文件创建的图表的一系列特征。如果其中一个属性同时定义多个特征，则将这些值放在一个用分号分隔的string中；例如：

HideQuals=0;AdvanceElementProps=1;ShowNotes=1;

访问

选择元类元素，然后显示 "属性"对话框并定义或更新属性 "styleex"或 "pdata"。

将属性类型指定为 "string"，然后在 "初始值"字段中指定您需要的图表特征。

使用这些方法中的任何一种来显示 "属性"对话框。

功能区	设计>元素>编辑器>属性
上下文菜单	在浏览器窗口或图表中|右键单击元类元素|特征|属性
键盘快捷键	F9

样式： string =

- AdvancedConnectorProps=1;（显示连接器属性字符串）
- 高级元素属性=1；（显示元素属性string ）
- AdvancedFeatureProps=1;（显示特征属性string ）
- AttPkg=1；（显示包可见类成员）
- DefaultLang=语言；（设置图表的默认语言；语言可以是 C++ 或Java等内置语言之一，也可以是自定义语言）
- 排除RTF=1；（从生成的报告中排除图表图像）
- 手绘=1；（应用手绘模式）
- HideConnStereotype=1;（隐藏连接器构造型标签）
- 隐藏质量=0；（显示限定符和可见性指标）
- NoFullScope=1；（隐藏完全范围的元素名称，例如 "ParentClass::ChildClass"将显示为 "ChildClass"）
- SeqTopMargin=50；（设置顺序图上序列的高度）
- 显示列表=1；（使图表直接在图表List 中打开）
- 显示列表=2；（使图表直接在甘图表图视图中打开）
- 显示列表=3；（使图表直接打开到规范管理器中）
- 显示列表=4；（使图表直接打开到关系矩阵）
- 显示维护=1；（显示元素维护区）
- ShowNotes=1；（显示元素注记隔间）
- ShowOpRetType=1;（显示操作返回类型）
- 显示测试=1；（显示元素测试区）
- SuppConnectorLabels=1;（抑制所有连接器标签）

- 抑制括号=1；（抑制无参数操作的括号）
- TConnectorNotation=选项；（其中 Option 是UML 2. 1 、IDEF1X 或信息工程之一）
- TExplicitNavigability=1;（显示不可导航的连接器末端）
- 可见属性详细信息=1；（在图表上显示属性详细信息）
- 白板=1；（应用白板模式）

数据： **string =**

- 隐藏属性=0；（显示元素隔间）
- 隐藏Estereo=0；（在图中显示元素的刻板印象）
- 隐藏操作=0；（显示元素操作区）
- 隐藏父母=0；（以显示图表中元素的其他父级）
- 隐藏道具=0；（显示属性方法）
- 隐藏关系=0；（表示关系）
- 隐藏立体声=0；（显示属性和操作原型）
- 操作参数 =3；（显示操作参数）
- ShowCons=1；（显示元素约束）
- 显示图标=1；（使用刻板印象图标）
- 显示请求=1；（显示元素需求隔间）
- 显示SN=1；（显示序列注记）
- 显示标签=1；（显示元素标记值）
- 补充CN=0；（显示合作编号）
- 使用别名=1；（使用图表中的别名或元素，如果可用）

设置技术元素图片

当您定义可在您的技术中使用的元素时，您可能希望使用图形图像来表示这些元素，这些图像将显示在用户通过该技术创建的图表上，当它被部署在用户的模型中时。

捕获图像以表示MDG 技术元素

节	行动
1	显示图像管理器，并使用加新按钮，将合适的图像从其源位置导入到MDG 技术开发模型中。
2	设计并创建一个构造型(UML)配置文件，其中包含（如果合适的话）技术所拥有的每个元素或连接器的构造型定义。 这些原型定义可以包含形状脚本，这些脚本又包含导入的图像。
3	设计并创建一个带有原型元素的工具箱配置文件，其中包含每个元素或连接器的属性，这些元素或连接器可以从工具箱中拖放到图表上。 这些属性标识技术元素或连接器的名称、任何修改构造型（可能包含所需的图像）以及技术object所基于的UML或扩展元素或连接器。 例如： SysML::块(UML ::类) • SysML是技术配置文件 • UML ::类是用作基础的UML元素，并且 • 块是修改类以将其转换为 SysML块元素的构造型
4	设计并创建一个图表工具箱配置配置文件的图形配置文件。 打开工具箱配置文件中定义的类型的图表时，它会依次打开图表配置文件配置文件的一组工具箱页面。
5	根据需要创建或更新技术，将UML配置文件、图表配置配置文件、工具箱配置文件和图像文件添加到开发模型中的技术中。
6	酌情部署该技术。 当用户将技术应用到他们自己的模型中，并在该技术下创建图表时，他们在图表上创建的元素应该由您在创建技术时分配给这些元素的图像来表示。

注记

- 建议如果您创建包含MDG 技术图像的形状脚本（步骤 2），则应使用完全限定的图像名称，以避免与其他技术中使用的图像冲突
- 您可能会在这些步骤中多次来回工作，在确定对它们的要求时添加对象

定义验证配置

使用 模型验证配置”对话框，您可以选择在用户执行验证时执行和不执行哪些验证规则集。

无需手动执行此配置，并且每次启动Enterprise Architect并且已将不同的技术设置为活动时都可能必须为您的技术更改设置，您可以在您的技术的MDG 技术选择 (MTS) 文件中定义配置设置.

访问

在您在工作中使用的任何文件浏览器中找到并打开 .MTS 文件。您按照这两个库表中的指示编辑文件，然后保存文件。

白名单

要将一组规则指定为白名单（即，任何添加到此列表的内容都打开），请在文本编辑器中打开您的 MTS 文件，然后将此 <ModelValidation> 块复制并粘贴到 < MDG内的顶层.Selections> 块：

<模型验证>

<RuleSet name="BPMNRules"/> <!-- Project.DefineRuleCategory 调用中定义的规则集 ID -->

<RuleSet name="MVR7F0001"/> <!-- 注意你也可以开启/关闭系统规则！ -->

</模型验证>

确保规则集 ID 不包含任何空格。

黑名单

要将一组规则指定为黑名单（即，任何添加到此列表的内容都已关闭），请在文本编辑器中打开您的 MTS 文件，然后将此 <ModelValidation> 块复制并粘贴到 < MDG的顶层.Selections> 块：

<ModelValidation isBlackList="true">

<RuleSet name="BPMNRules"/>

<RuleSet name="MVR7F0001"/>

</模型验证>

在此示例中，“BPMNRules”是在 Project.DefineRuleCategory 调用中定义的规则集 ID - 请参阅项目类了解详细信息。 “MVR7F0001”是一个内置的规则集。当您激活适当的技术时，将应用这些验证选项。全局（默认）技术已打开所有规则。

合并模型Builder模板

当用户在其项目中创建模型时，他们可以从 模型生成器”对话框中显示的一系列系统提供的模型模板中选择要开发的模型类型。您还可以开发自定义模型模板并通过MDG 技术将其添加到此列表中。

访问

您可以直接编辑 .mts 文件，使用您使用的任何文件浏览器来定位和打开该文件。

将模型模板添加到MDG 技术

节	行动
1	创建一个包，其中包含您要在模型模板中提供的所有子包、图表、元素、注记和信息链接。 请参阅 EAExample.eap模型以了解您可能包含的内容，或从标准模板创建模型并查看生成的内容。 作为模型模板，包需要是自包含的，并且不包含任何依赖项或其他指向包外元素的链接。
2	将您的包导出到 XML。 如果您希望模板的支持文档显示在模型生成器对话框的右侧面板中，请在与 XML 文件相同的目录位置创建一个包含此文档的 .rtf 文件。.rtf 文件的文件名也必须与 XML 文件相同。建议您在模型中的文档工件元素中创建 .rtf 文件，然后将文件（ 文档-编辑 >文件> 另存为 (导出到文件)”功能区选项）导出到模式XML 文件的位置。这样可以将文档保留在您的开发模型中。
3个	要允许每个技术有多个自定义类别：在文本编辑器中打开您的 .mts 文件并向 <Technology>元素添加两个附加属性： • 类别列表，其中 包含以逗号分隔的自定义类别名称列表，或单个内置类别的名称（例如 业务”） • 类别映射， 其中包含形式为 组名称1=类别名称A ;组名称2=类别名称B ;”的选项对列表，依此类推；类别名称必须全部在 ategoryList”中
4个	在 .mts 文件中创建对 XML 文件的引用；在文本编辑器中打开您的 .mts 文件，然后将此 <ModelTemplates>元素复制并粘贴到 < MDG .Selections> 块内的顶层： <模型模板> <模型name="模板名称" 位置="MyTemplatePackage.xml" 默认= 是” 图标 = “34” isFramework=" false "/> </模型模板> 您可以在 .mts 文件中的 <ModelTemplates>元素中包含任意数量的 <Model> 元素，每个模型模板一行。 <Model>元素内的属性具有以下含义：

	• name: 在模型构建器对话框中显示的模型模板的名称，当您选择模型蓝图或执行 模型构建器（模式库）”菜单选项时显示该名称 • 位置：包含模型模板包导出的 XML 文件的路径，相对于Enterprise Architect安装路径中模型模式目录的位置： - 如果 XML 文件直接位于模型模式目录中，则 路径仅包含文件名（例如，MyPattern1.xml） - XML 可以与MDG 技术XML 文件位于同一文件夹中，RTF 文件位于同一文件夹中 - 如果你将所有文件放在模型模式的子目录中， 路径包含目录名称（例如，MyTechnology\MyPattern2.xml） - 您也可以指定固定路径（例如，C:\Program Files\MyTechnology\MyPattern3.xml） • 图标：包含Enterprise Architect基本图标列表的索引；要显示适当的视图图标，请使用以下值之一： - 29 =用例 - 30 = 动态 - 31 =类 -32=部件 - 33 = 部署 - 34 = 简单 • isFramework：定义模型模式的可能用途；有三个可能的值： - isFramework="true" - 永远不会剥离 GUID；该模式旨在 作为任何模型的可重复使用包 - isFramework= 可选”- 剥离 GUID 的提示；模式是 旨在作为可重复使用的包，但用户可以选择 - isFramework=" false " - 总是删除 GUID（默认，如果没有 所述）；该模式可以在 一个模型 • groupName：如果指定了多个自定义类别，此属性用于引用此模式属于哪个类别。
5个	使用编辑的 MTS 文件重新生成MDG 技术。

添加导入/导出脚本

在Enterprise Architect中，可以从一系列 XMI 和 XML 格式的外部文件导入包和导出（或发布）包。您还可以将此功能合并到您的MDG 技术中，添加一个包含您自己的可扩展样式表语言 (变换) 的脚本以在文件格式之间进行转换。

合并一个导出（发布）脚本

节	描述
1	在您首选的编辑器中，创建一个 XSLT 以将源格式（如 "发布模型包"对话框中列出的）转换为您正在生成的目标格式。
2	在Enterprise Architect中，打开 Scripter 窗口并在您首选的脚本引擎下创建一个脚本作为普通脚本。 将 XSLT 剪切并粘贴到脚本编辑器中。
3	在MDG 技术创建向导中将脚本添加到您的MDG 技术中。
4	对您需要的技术 .mts 文件进行任何添加，然后再次使用MDG 技术创建向导来完全生成技术 .xml 文件。 在文本编辑器中打开技术 .xml 文件（不是 .mts 文件）并找到 <脚本部分。
5	编辑 <脚本行以设置适当的名称、类型和语言： • 名称是要在 "发布>模型交换"功能区面板中显示的技术选项文本 • *type*是单词 "Publish-"，后跟要导出的文件格式的名称，如 "发布模型包"对话框中所列 • 语言是 XSLT 例如： <脚本 名称= "你的技术" type="发布-UML 1 (XMI 1)" 语言="XSLT"> <内容 xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="bin.base64"> </内容> </脚本>
6	保存MDG 技术.xml 文件，并将其部署到您的系统上。

合并导入脚本

节	描述
1	在您首选的编辑器中，创建一个 XSLT 以将源格式转换为目标 XMI 格式。

2	在Enterprise Architect中，打开 Scripter 窗口并在您首选的脚本引擎下创建一个脚本作为普通脚本。 将 XSLT 剪切并粘贴到脚本编辑器中。
3	在MDG 技术创建向导中将脚本添加到您的MDG 技术中。
4	对您需要的技术 .mts 文件进行任何添加，然后再次使用MDG 技术创建向导来完全生成技术 .xml 文件。 在文本编辑器中打开技术 .xml 文件（不是 .mts 文件）并找到 <脚本部分。
5	编辑 <脚本行以设置适当的名称、类型和语言： • 名称是要在Enterprise Architect 发布>模型交换>导出”功能区选项中显示的技术选项文本 • *type*是单词 Ímport-”，后跟要生成的 XMI 文件格式的名称，如 发布模型包”对话框中所列 • 语言是 XSLT 例如： <脚本 名称= 你的技术” type="导入-UML 1 (XMI 1)" 语言="XSLT"> <内容 xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="bin.base64"> </内容> </脚本>
6	保存MDG 技术.xml 文件，并将其部署到您的系统上。

注记

- 在 XSLT 1 .0 中创建脚本的内容

部署MDG 技术

MDG 技术可以通过以下两种方式之一部署：作为 .xml 文件或从插件

.

从 .xml文件部署

要将您的技术部署为文件，您有多种选择：

- 将技术 .xml 文件导入 %APPDATA%\ Sparx Systems \EA\导入文件夹（供您个人使用）
- 将技术导入文件导入浏览器窗口的 资源”选项卡（供所有项目用户访问）
- 将文件复制到Enterprise Architect安装目录下的 MDGTechnologies 文件夹（默认为C:\Program Files\Sparx Systems\EA ）；当您重新启动Enterprise Architect时，您的MDG 技术已部署
- 将文件复制到文件系统中的任何文件夹，包括网络驱动器 - 使用 特定>技术>管理技术”功能区选项，单击高级按钮并将文件夹添加到 技术”路径；这种部署方法使您能够快速轻松地将技术部署到 LAN 上的所有Enterprise Architect用户
- 将文件上传到 Internet 或 Intranet 位置：使用 特定>技术>管理技术”功能区选项，单击高级按钮并将 URL 添加到 技术”路径；这种部署方法使您能够快速轻松地将技术部署到更广泛的Enterprise Architect用户组

从插件
部署插件

从插件
部署您的技术插件
，您必须编写一个 EA_OnInitializeTechnologies函数。这个例子是用 VB.Net 编写的：
公共函数EA_OnInitializeTechnologies(ByVal存储库As EA.Repository) 作为物件
EA_OnInitializeTechnologies = 我的.资源.MyTechnology
结束函数

形状脚本

您最初在建模中使用的元素和连接器在形状、颜色和标签方面符合标准UML表示法。但是，您可以扩展标准对象以创建新对象，并使用形状脚本自定义这些新对象的外观，以定义您想要施加到默认或主形状上的确切特征。您使用专用脚本语言创建形状脚本脚本，以定义元素或连接器的新形状、方向、颜色和标签。每个脚本都与一个原型相关联，每个具有该原型的元素或连接器都将采用由形状脚本定义的形状脚本。

如果要标准化外观以应用于许多元素，可以将形状脚本脚本附加到MDG 技术构造型配置文件中的构造型元素的属性。

如果您已将形状脚本应用于某些元素和/或连接器，但不想在特定图表上显示这些形状脚本，您可以使用图表的“属性”对话框关闭该图表上的形状脚本的显示。

开始形状脚本

由于形状脚本与构造型相关联，因此您可以通过 "UML类型"对话框的 构造型"选项卡来定义它们；每个构造型都可以有一个形状脚本。设置形状脚本的过程非常简单但非常灵活。

访问

功能区	设置 >参考> UML类型 >构造型

形状脚本进程

节	行动
1	从对话框右侧的列表中选择要将形状脚本附加到的构造型。 您选择一个现有的构造型，但如果没有合适的构造型，您可以创建一个新的构造型，一旦保存，它就会显示在列表中并且可以选择。
2	在 覆盖外观"面板中，选择 形状脚本"单选按钮，然后单击 分配"按钮。 显示形状编辑器。
3	类型或将脚本复制到编辑窗口中。 要在 预览"面板中审阅形状，请单击 刷新"按钮。
4	如果您定义了一个复合形状脚本形状（带有装饰和标签的主要形状，或单独的部分，例如具有源端和目标端形状的连接器），请单击 下一个形状"按钮以翻阅形状的组件，在 预览"面板。
5	写完形状脚本后，单击确定按钮返回 构造型"选项卡。 然后单击保存按钮以保存形状脚本及其对构造型的分配。
6	将适当的标准UML元素或连接器拖放到您的图表中。该object将是您选择作为构造型的 基类"的类型。 右键单击object并选择 属性"选项。 在 属性"对话框中，单击 构造型"下拉箭头，选择您创建的构造型并单击 确定"按钮。 object的形状现在反映了形状脚本给构造型的形状。

注记

- 使用元素形状脚本来修改元素的外观会使一些正常的 外观"上下文菜单选项变得多余，因此它们将被禁用
- 无法修改或覆盖MDG 技术中定义的类型的形状脚本
- 形状脚本不支持字体选择，因为最佳用户体验是通过允许用户自己设置字体来实现的
- UML定义了通过Profiles扩展UML语法的标准机制；出于这个原因，形状脚本不能独立于原型应用于任何元

素

- 形状脚本不能用于使用贝塞尔线样式的连接器
- 形状脚本目前不支持：
  - 循环结构
  -字符串操作
  - 算术操作
  - 变量声明

形状编辑器

当您通过 "UML类型"对话框的 构造型"选项卡创建形状脚本脚本时，您可以使用形状编辑器编写脚本。这提供了功能代码编辑器功能，包括智能感知形状脚本公共属性功能。

访问

功能区	设置>参考> UML类型>构造型：（选择或指定构造型）：形状脚本+分配

编辑器选项

选项	行动
格式	点击下拉箭头，选择形状脚本版本（目前只有形状脚本1 .0可用）。
导入	单击此按钮可从文本文件 (.txt) 中导入形状脚本。将显示A文件浏览器，您可以通过它找到要导入的文件。 找到并选择文件后，单击 打开"按钮将脚本导入编辑面板。
导出	单击此按钮可将形状脚本导出到文本文件。将显示A文件浏览器，您可以通过该浏览器指定要导出到的文件。 确定文件后，单击 保存"按钮以完成导出并返回到形状编辑器。
<编辑面板>	类型此面板中的脚本命令。
确定	单击此按钮可退出形状编辑器。 要保存您的形状脚本，请单击 构造型"选项卡上的保存按钮。
下一个形状	如果您有一个由不同组件组成的形状，请单击此按钮以在 预览"面板中旋转多个形状定义。
刷新	单击此按钮可解析您的脚本并在预览窗口中显示结果。

编写脚本

要为元素或连接器创建替代表示，您可以编写一个定义表示的大小、形状、方向和颜色的形状形状脚本。 A形状脚本包含许多用于定义形状不同方面的部分；对于一个元素，这些包括：

- 主要object
- 标签
- 装饰（例如，一个 Document元素可能包含一个描述文档的图标）

对于连接器，这些部分包括：

- 主要object
- 形状源
- 形状目标
- 标签

形状脚本在使用默认（UML）表示的基础上运行，除非脚本包含替代定义。那是：

- 如果你有一个只包含一个装饰的形状脚本，这个装饰会被添加到正常绘制的object之上
- 如果你有一个空的形状例程，它会覆盖默认值；因此，空白的 形状标签”会阻止为具有它们的元素创建正常的浮动文本标签

你也可以使用 C 风格的注释来注释你的脚本；例如：

// C 风格单行注释

/* 多行

支持评论 */

脚本不区分大小写： 形状” 形状”相同。

脚本结构

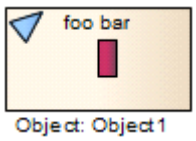
Layout	Description

示例元素布局脚本	主要形状
	{
	// 绘制object
	}
	形状标签
	{
	// 绘制浮动文本标签
	}
	装饰 <标识符>
	{
	// 在object内部绘制一个 16x16 的装饰
	}
	
	< 标识符 > string是一个字母数字单词。

连接器示例布局脚本	主要形状 {

	// 划清界线
	}
	形状目标
	{
	// 在目标端绘制形状
	}
	形状源
	{
	// 在源端绘制形状
	}
	标签 <位置标签>
	{
	// 定义标签的文本
	}
	
	<positionLabel> string可以是以下任何一种：
	● 左上标签
	● 左下标签
	● 中顶标签
	● 中间底部标签
	● 右上标签
	● 右下标签

子形状	形状可以A子形状，子形状必须在主形状脚本之后声明，但从方法命令中调用。
	这是声明排序的示例：
	
	主要形状
	{
	// 初始化属性 - 这些必须在绘制命令之前
	noshadow = "真";
	h_align = "中心";
	
	//绘图命令（方法）
	矩形（0,0,100,100）；
	println ("foo bar");
	
	// 调用子形状
	addsubshape ("红色", 20, 70);
	
	// 子形状的定义
	形状红色
	{
	setfillcolor (200,50,100);
	矩形（50,50,100,100）；

	}
	}
	
	//标签的定义
	形状标签
	{
	setOrigin ("SW",0,0);
	println("物件：#NAME#");
	}
	
	//装饰的定义
	装饰三角形
	{
	//划一个三角形装饰
	开始路径（ ）；
	移动到（0,30）；
	线托（50,100）；
	lineto (100,0);
	结束路径（ ）；
	设置填充颜色（153,204,255）；
	fillandstrokepath();
	}
	
	此脚本产生的形状是：
	 

申报顺序	形状可以由属性声明、方法/命令调用和子形状定义组成，它们必须按该顺序出现；也就是说，属性声明必须出现在所有方法调用之前，子形状定义必须出现在最后。

形状属性

使用形状脚本定义形状时，使用属性定义该属性的属性。属性包括：

- 形状相对于图表和其他元素的位置
- 形状组件相对于形状边框的位置
- 形状是否有用户可编辑的区域
- 形状是否可以调整大小、缩放、旋转或停靠

属性语法

属性 =“值 ”;”

示例

主要形状
{
//初始化属性——必须在绘制命令之前
noshadow = "真";
h_align = "中心";
//绘图命令
矩形（0,0,100,100）；
println ("foo bar");
}

属性

Attribute Name	Description
大胆的	string 描述：如果您希望当前形状或子形状中的所有打印命令都以粗体显示，请设置为True 。 有效值： True或False （默认 = False ）
斜体	string 描述：如果您希望当前形状或子形状中的所有打印命令都以斜体显示，请设置为True 。 有效值： True或False （默认 = False ）
bottomAnchorOffset	（ int ， int ） 当创建一个形状脚本描述嵌入元素（例如一个嵌入的端口）时，使用这个属性从它的父元素的底部边缘偏移形状。 例如：

	bottomAnchorOffset= (0,-10); 将嵌入的元素从底部边缘向上移动 10 个像素。
可停靠	string 描述：使形状默认为可停靠，以便它可以与图表上的其他元素（其他形状脚本和标准元素）对齐并连接。您无法使用 外观"菜单选项反转可停靠状态；要更改状态，您必须编辑形状脚本。 有效值：标准或关闭
可编辑字段	string 描述：将形状定义为元素的可编辑区域。 此字段仅影响元素形状，不支持线条字形。 有效值：别名、姓名、注记、刻板印象
端点Y，端点X	整数 描述：仅用于连接器的保留目标和源形状；此点确定主连接线连接到末端形状的位置。 默认值：0 和 0
固定纵横比	string 描述：设置为True以固定纵横比。如果您不想固定纵横比，请不要使用此选项。
h_Align	string 描述：根据 layoutType 属性影响打印文本和子形状的水平放置。 有效值：左、中或右
布局类型	string 描述：确定子形状的大小和位置。 有效值：leftright、topdown、border
leftAnchorOffset	（ int ， int ） 当创建一个形状脚本描述嵌入元素时（例如一个端口元素的属性描述），使用这个属性从它的父元素的左边缘偏移形状。 例如： leftAnchorOffset= (10,0); 将嵌入的元素从左边缘向右移动 10 个像素
无影	string 描述：设置为True以禁止渲染形状的阴影。 有效值： True或False （默认 = False ）
方向	string 描述：仅适用于装饰形状，以确定装饰在包含元素字形中的位置。 有效值：NW、 N 、NE、 E 、SE、 S 、SW、W
首选身高	描述：由边界布局类型使用 - 北和南。 用于绘制连接器的源和目标形状以确定线的宽度。

首选宽度	描述：由边界布局类型使用 - 东和西。 由 leftright layoutType 形状使用，其中可扩展性为false ，以确定它们为布局目的占用多少空间。
右锚偏移	（ int ， int ） 当创建一个形状脚本描述嵌入元素（例如一个嵌入的端口）时，使用这个属性从它的父元素的右边缘偏移形状。 例如： rightAnchorOffset= (- 10,0); 将嵌入的元素从右边缘向左移动 10 个像素。
可旋转	string 描述：设置为False以防止形状旋转。此属性仅适用于线条字形的源形状和目标形状。 有效值： True或False （默认 = True ）
可扩展的	string 描述：设置为False以阻止形状相对于关联的图表字形调整大小。 有效值： True或False （默认 = True ）
topAnchorOffset	（ int ， int ） 当创建一个形状脚本描述嵌入元素（例如一个嵌入的端口）时，使用这个属性从它的父元素的顶部边缘偏移形状。 例如： topAnchorOffset= (0,10); 将嵌入的元素从顶部边缘向下移动 10 个像素。
v_Align	string 描述：根据 layoutType 属性影响打印文本和子形状的垂直放置。 有效值：顶部、中心或底部

绘图方法

使用形状脚本创建形状时，可以使用方法定义形状的值。这些值包括以下内容：

- 形状是什么 - 一个矩形，一条线，一个球体
- 形状的大小
- 形状和边框的颜色
- 形状具有的隔间和隔间文本
- 显示在形状中和形状周围的文本和标签
- 形状是否包含或包含捕获的图像

您可以通过按 Ctrl+Space 列出脚本中任何点的有效方法（命令）。

方法语法

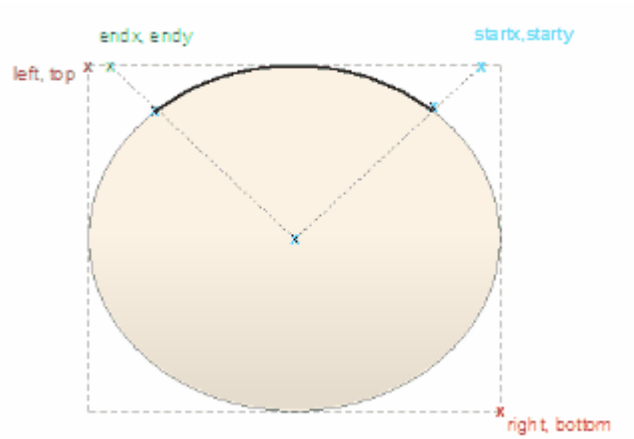
<方法名> "(" <参数列表> ")";";

示例

主要形状
{
// 初始化属性 - 这些必须在绘制命令之前
noshadow = "真";
h_align = "中心";
//绘图命令（方法）
矩形（0,0,100,100）；
println ("foo bar");
}

方法

Method Name	Description
addsubshape（ string shapename（ int宽度， int高度））	添加一个必须在当前形状定义中定义的名为“shapename”的子形状。
附加隔间文本（ string ）	将额外的字符串附加到隔间的文本中。 添加文本的隔间取决于在使用“appendcompartmenttext”之前使用“setcompartmentname”设置的隔间名称。 必须调用此方法才能显示隔间。
弧（ int左， int顶部， int右， int底部， int起点x， int起点， int终点x，	绘制一个椭圆逆时针圆弧，椭圆的范围在左、上、右和下。 圆弧的起点由椭圆与椭圆中心到该点的连线（startingpointx，startingpointy）

int终点）	的交点定义。 弧的终点类似地由椭圆与从椭圆中心到该点的线（endpointx，endpointy）的交点定义。 例如： 弧（0, 0, 100, 100, 95, 0, 5, 0）； 
arcto（ int左， int顶部， int右， int底部， int起点 x， int起点， int终点 x， int终点）	至于圆弧方法，只不过是从当前位置画一条线到圆弧的起点，然后将当前位置更新到圆弧的终点。
bezierto（ int controlpoint1x， int controlpoint1y， int controlpoint2x， int controlpoint2y， int endpointx， int endpointy）	绘制贝塞尔曲线并更新画笔位置。

defSize（ int宽度， int高度）	设置元素的默认大小。
	这可能出现在 IF 和 ELSE 子句中，每个子句具有不同的值，并导致元素在每次值更改时自动调整大小。
	if(HasTag("水平","true"))
	{
	defSize(100,20);
	矩形（0,0,100,100）；
	}
	else
	{
	defSize(20,100);
	矩形（0,0,100,100）；
	}
	本示例在每次更改标签值'标记值'时将形状设置为指定的默认大小。
	设置后，Alt+Z 还会将形状调整为定义的尺寸。
	int width 和int height 的最小值都是 10。

绘制形状（ ）	以通常的非形状脚本表示法渲染形状；随后的绘图命令叠加在本机符号上。

	仅元素形状脚本支持此方法；不支持线形状脚本。
绘制父图形（ ）	在扩展非 UML物件类型时使用。 呈现从父原型定义的形状。如果没有继承的形状脚本可用，则行为与形状脚本() 相同。
椭圆（ int left， int top，int right， int底部 ）	绘制一个椭圆，其范围由左、上、右和下定义。
结束路径（ ）	结束定义路径的绘图命令的序列。
填充行程路径（ ）	用当前的填充颜色填充之前定义的路径，然后用当前的笔绘制它的轮廓。
填充路径（ ）	用当前填充颜色填充先前定义的路径。
获取默认填充颜色（ ）	获取元素的默认填充颜色。这可以是所有元素的标准填充颜色，或者，如果在 首选项"对话框的 图表>外观"页面上选择了 无素使用组样式"选项，则为元素**类型**定义的默认填充颜色。
获取默认线颜色（ ）	获取元素的默认线条颜色。这可以是所有元素的标准线条颜色，或者，如果在 首选项"对话框的 图表>外观"页面上选择了 无素使用组样式"选项，则为元素**类型**定义的默认线条颜色。

隐藏标签（ string标签名称 ）	隐藏由 labelname 指定的标签，其中 labelname 是以下值之一：
	• 中间标签
	• 中底标签
	• 左上标签
	• 左下标签
	• 右上标签
	• 右下标签
	注记：通过将指定的标签设置为隐藏来实现此功能。对脚本的任何后续更改都不会再次显示标签。
	抑制标签的推荐方法是覆盖该形状。例如，抑制默认构造型标签：
	形状中间底部标签
	{
	打印（""）;
	}

图像（ string imageId，int左， int顶部， int右，int底部 ）	在图像管理器中绘制名称为 imageId 的图像。 图像必须存在于使用原型的模型中；如果模型中不存在，则必须将其作为参考数据导入或从技术文件中选择。 如果图像在技术文件中，则它的文件名格式应为 <technology ID>::<imagename>.<extension>。
lineto（ int x， int y ）	从当前光标位置到 x 和 y 指定的点绘制一条线，然后将笔光标更新到该位置。（另见注记部分。）
移动（ int x， int y ）	将笔光标移动到 x 和 y 指定的点。
多边形（ int centerx， int centery， int	绘制一个正多边形，中心在点 (centerx, centery)，边数为 numberofsides。

numberofsides， int radius，float rotation ）	
打印（ string文本 ）	打印指定的文本string 。 您不能更改此文本的字体大小或类型。
printifdefined(string propertyname, string truepart(, string falsepart))	如果给定属性存在并且具有非空值，则打印 “truepart”，否则打印可选的 “falsepart”。 您不能更改此文本的字体大小或类型。
println(string文本)	将一行文本附加到形状和换行符。 您不能更改此文本的字体大小或类型。
printwrapped（ string文本 ）	打印指定的文本string ，如果文本比其包含的形状宽，则将其包裹在多行中。 您不能更改此文本的字体大小或类型。
矩形（ int左， int顶部， int右， int底部 ）	绘制一个矩形，其范围为左、上、右、下。值是百分比。
圆形矩形（ int左， int顶部， int右， int底部， int abs_cornerwidth， int abs_cornerheight ）	绘制一个圆角矩形，其范围由左、上、右和下定义。 角的大小由 abs_cornerwidth 和 abs_cornerheight 定义；这些值不随形状缩放。
设置附件模式()	定义连接器如何连接到元素形状，在元素边上的任意点（参数 “normal”）或在每个边的中心点（参数 “diamond”），具体取决于 “矩形表示法”选项的设置。请参阅示例脚本帮助主题。
设置隔间名称（ string ）	将隔间名称设置为提供的string 。 此方法必须在调用 appendcompartmenttext 之前使用；在调用 appendcompartmenttext 之后调用它会清除已添加到隔离专区的任何文本。
设置默认颜色（ ）	将画笔和画笔颜色恢复为默认设置，或恢复为用户定义的颜色（如果可用 ）。
setfillcolor（ int red， int green， int blue ）或 setfillcolor（颜色新颜色 ）	设置填充颜色。 您可以通过定义 RGB 值或使用任何颜色查询返回的颜色值来指定所需的颜色，例如： GetUserFillColor() 或 获取用户边框颜色（ ） 在所有情况下，setfillcolor 优先于适用于元素的任何颜色定义。
setfixedregion（ int xStart， int yStart， int xEnd， int yEnd ）	修复连接器中可以绘制子形状的区域，以便子形状不会随着连接器线的长度或方向重新缩放。 有关示例，请参阅示例脚本主题中的 “旋转方向”脚本。
setfontcolor（ int red， int green， int blue ）或 setfontcolor（颜色新颜色 ）	设置文本string的字体颜色。 您可以通过定义 RGB 值或使用任何颜色查询返回的颜色值来指定所需的颜色，例如： GetUserFontColor() 或 获取用户填充颜色（ ）

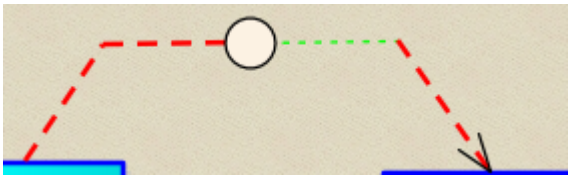
	您可以将此命令与任何文本打印命令一起使用。
setlinestyle（ string线型）	修改使用笔的命令的笔划模式。 string linestyle: 具有以下有效样式： ● 坚硬的 ● 短跑 ● 点 ● 破折号 ● 点划线 ● 双倍的 （另见注记部分。）
setorigin（ string relativeTo， int xOffset，int yOffset）	相对于主形状定位浮动文本标签。 ● relativeTo 是N 、NE、 E 、SE、 S 、SW、W、NW、CENTER 之一 ● xOffset 和 yOffset 以像素为单位，不是百分比值，可以为负数
setpen(int red, int green, int blue, int penwidth) 或 setpen(Color newcolor, int penwidth)	将画笔设置为定义的颜色并设置画笔宽度。 此方法仅适用于画线命令。它不影响任何文本打印命令。
setpencolor(int red, int green, int blue) 或 setpencolor(Color newColor)	设置画笔颜色。 您可以通过定义 RGB 值或使用任何颜色查询返回的颜色值来指定所需的颜色，例如： 获取用户填充颜色（ ） 此方法仅适用于画线命令。它不影响任何文本打印命令。
setpenwidth(int penwidth)	设置画笔的宽度。笔宽应在1到 5 之间。 此方法仅适用于画线命令。它不影响任何文本打印命令。
显示标签（ string标签名称）	显示由 labelname 指定的隐藏标签，其中 labelname 是以下值之一： ● 中顶标签 ● 中间底部标签 ● 左上标签 ● 左下标签 ● 右上标签 ● 右下标签
startcloudpath（ puffWidth、puffHeight、噪声）	与 startpath 类似，不同之处在于它使用云状曲线段（粉扑）绘制路径。 参数： ● float puffWidth（默认 = 30），粉扑之间的水平距离 ● float puffHeight（默认 = 15），烟团之间的垂直距离 ● 浮动噪声（默认 = 1 .0），烟团位置的随机化
开始路径（ ）	启动定义路径的绘图命令的序列。
行程路径（ ）	用当前画笔绘制先前定义的路径的轮廓。

注记

- 如果为一条由多条线段组成的线绘制形状脚本，并为线段定义不同的线型，则除中心线段以外的所有线段都使用第一个定义的线型；中心线段使用定义的第二种线型，如图所示：

```
形状 主要的
{
    无影=真；
    // 此笔样式将被忽略，因为没有绘制任何内容。
    setpen ( 0 , 0 , 0 , 1 );
    SetLineStyle （ 实心” ）；

    // 这种笔样式将用于非中心线段，因为它是
    // 第一个用于绘图。
    设置笔( 255 , 0 , 0 , 2 );
    SetLineStyle （ 破折号” ）；
    移动到( 0 , 0 );
    lineto ( 50 , 0 );

    // 这个线型用在中间段，没有其他的，因为它
    // 不是第一个绘制的。
    setpen ( 0 , 255 , 0 , 1 );
    SetLineStyle ( "点" );
    lineto ( 100 , 0 );

    // 此线型仅用于中心段的注释。
    setpen ( 0 , 0 , 0 , 1 );
    SetLineStyle （ 实心” ）；
    setfixedregion ( 40 ,- 10 , 60 , 10 );
    椭圆( 40 ,- 10 , 60 , 10 );
}
```

具有此形状脚本A依赖连接器可能类似于以下内容：



颜色查询

在定义形状时，您可能希望保留已为基本形状定义的填充、边框和字体颜色。您可以使用颜色查询来设置颜色定义，以检索 SetPenColor 和 SetFillColor 命令的参数。这些查询可以用来代替参数。

- getUserFillColor() - 返回当前元素的用户选择的填充颜色
- getUserBorderColor() - 返回当前元素的用户选择的边框/线条颜色
- getUserFontColor() - 返回当前元素的用户选择的文本字体颜色
- getUserPenSize() - 返回当前元素的用户选择的线条粗细
- getDefaultFillColor() - 返回当前元素的默认填充颜色，而不使用应用于此元素的颜色
- getDefaultLineColor() - 返回当前元素的默认线条颜色，而不使用应用于此元素的颜色
- getStatusColor() - 返回当前元素的状态颜色；如果没有为此状态定义颜色，或者没有针对此类型显示状态颜色，则此查询将返回与状态相同的结果

例如：

主要形状

{

setfillcolor(getuserfillcolor());

setpencolor(getuserbordercolor());

矩形（0,0,100,100）；

}

注记

- 用户颜色是在基础object没有被形状脚本修改的情况下设置的颜色；它们将被定义使用 - 按优先级递减的顺序 - 格式工具栏选项， 外观"选项（F4）或 首选项"对话框（ 开始>外观>首选项>首选项"）
- 因为用户颜色是为元素定义的颜色，随后应用了构造型和形状脚本，它们不能在形状编辑器的 预览"面板中描述

条件分支

您可以使用 IfElse"语句或计算结果为True或False的查询方法将条件分支合并到您的形状脚本中。

当您使用这些条件分支语句时，您可以使用 return 命令在满足分支条件时终止脚本的执行。示例脚本主题提供了这方面的几个示例，例如 返回语句形状"脚本。

查询方法

当您在形状脚本中使用形状脚本语句时，条件通常是object具有某个标记或属性，并且可能该标记或属性具有特定值。您可以使用此处描述的两种查询方法之一设置条件语句来检查属性和值。

查询

Method	Description
boolean HasTag(string标记名， （ string标记值 ） ）	如果 'tagname' 存在且其值为非空，则 HasTag(tagname) 评估为 ' True '；否则它评估为 ″false ″。 如果 'tagname' 存在且其值为 'tagvalue'，则 HasTag(tagname,tagvalue) 计算结果为 ' True '。 如果 'tagname' 不存在且 'tagvalue' 为空，则 HasTag(tagname,上下文) 也将评估为 ' True '，此时将 'empty' 和 'missing' 视为具有相同的含义。

boolean HasProperty(string属性名，
（ string属性值 ） ）
	如果关联的元素具有名称为 propertyname 的属性，则计算结果为True 。
	如果提供了第二个参数 propertyvalue，则该属性必须存在，并且该属性的值必须等于属性才能使方法评估为True 。
	propertyvalue 参数可以有多个值，以逗号分隔；例如：
	if(HasProperty("类型","类,行动,活动,接口"))
	{
	SetFillColor(255,0,0);
	绘制原生形状（ ）；
	}
	这个形状脚本将对 if(HasProperty()) 语句中指定的四种元素之一以外的任何类型的元素使用标准元素填充颜色；这四种类型中的任何一种元素都将以红色填充显示。

HasProperty 和用户选择的设置

功能() 方法的A特殊应用是检查属性设置，其中您为用户提供了属性功能，以便为使用构造型元素的特定实例设置该属性。因此，用户可以将元素形状脚本图表上，并通过元素上下文菜单，设置一个或多个属性在渲染图表object时所对应的属性。因此，元素可能在一个图表上具有一种外观，但在另一个图表上具有不同外观，因为它在两个图表上具有不同的属性设置。

要在您的配置文件中指定用户可选择的属性，请创建适当的构造型元素，并且 - 对于正在定义的每个属性- 将具有构造型 «diagram property» 的属性添加到此元素。对于属性，键入将显示在上下文元素的时间名称菜单上的选项文本；例如， 是红色的″。还要给属性一个别名，这将是属性的名称，因为它被存储并且属性() 方法将评估它。如果您将属性的初始值设置为1 ，则上下文会设置时间菜单选项；如果没有初始值，属性选项将默认为不设置。

还要定义一个 _image 属性，该属性具有应用 HasProperty() 方法的形状脚本脚本。在本例中，形状脚本定义了两个类属性（ Is Red 和 Is Triangle ），用于形状脚本() 方法，用于检查是否设置了选项。

主要形状

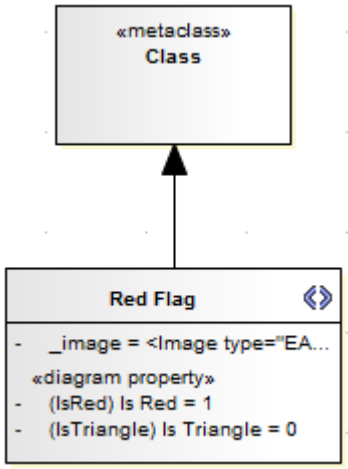
{

if (HasProperty("IsRed"," 1 "))

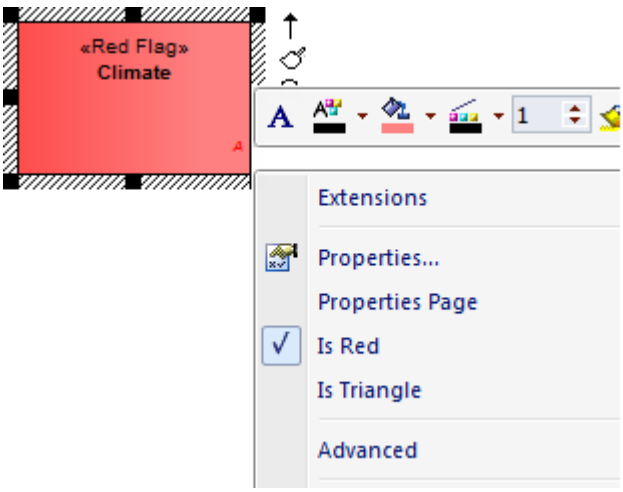
{

设置填充颜色（255,128,128）；
}
if (HasProperty("IsTriangle","0"))
{
多边形（50,50,3,50,0）；
}
else
{
绘制原生形状（）；
}
}

当定义扩展元素类型的构造型时，它将类似于：



MDG 技术创建并发布给用户后，当他们从工具箱中拖动原型元素时，它将根据定义的当前设置属性，用户可以通过上下文菜单访问和重新设置，如显示：



显示元素/连接器属性

自定义形状A常见组件是文本string ，它可以包括元素或连接器的属性之一的名称和值。要显示文本，请使用以下命令之一：

- 打印
- 打印和
- 打印包装

这些都采用一个string参数来表示要显示的文本。元素或连接器属性可以使用替换宏 #<propertyname># 添加到文本中；例如：

println("姓名：#NAME#");

您可以通过多次发出命令来显示多个属性，每个属性一次。您可以显示的元素和连接器属性在此处列出。此外，您还可以通过在标签名称前加上标记值来显示标签，如下所示：

print("#TAG:条件#");

您还可以像测试系统命名的属性一样测试和显示元素的自定义属性；例如：

if(hasproperty("名称","值"))

...

和：

print("#Name#");

Properties for Element Shape Scripts

- actualname - same as 'name' except that it does not react to the 'Use Alias if Available' setting
- addin - returns a value from an invoked Add-In function; syntax:
  addin:<addin_name>, <function_name>, <parameter> [, <parameter> ...]
  Note that in the hasproperty() argument, Enterprise Architect requires the hash characters for addin values:
  if(hasproperty("#ADDIN:MyAddin,MyValue#", "TheValue")) {
- alias
- author
- bookmark
- bookmarkvalue
- cardinality
- classifier
- classifier.actualname - same as 'classifier.name' except that it does not react to the 'Use Alias if Available' setting
- classifier.alias
- classifier.metatype
- classifier.name
- classifier.stereotype
- classifier.type
- complexity
- concurrency
- datecreated
- datemodified
- diagram.author

- diagram.handdrawn
- diagram.mdgtype
- diagram.mdgview
- diagram.name
- diagram.stereotype
- diagram.type
- diagram.version
- ES (adds the End Stereotype character(s) as determined by the "Use extended << and >> characters" option)
- haslinkeddocument
- hiddenparents
- incomingedge (returns "none", "left", "right", "top", "bottom", or "multiple")
- isabstract
- isactive
- iscomposite
- isdrawcompositelinkicon
- isembedded
- isinparent
- isleaf
- islocked
- isroot
- isspec
- istagged
- isvisible
- keywords
- language
- metatype
- multiplicity
- name
- notes
- notesvisible
- outgoingedge (returns "none", "left", "right", "top", "bottom", or "multiple")
- packagename
- packagepath
- package.stereotype
- parentedge ("right", "left", "top", "bottom")
- parent.metatype
- partition (returns "vertical" or "horizontal")
- persistence
- phase
- priority
- propertytype
- propertytype.alias

- propertytype.metatype
- propertytype.name
- propertytype.stereotype
- qualifiedname
- rectanglenotation
- scope
- showcomposeddiagram (returns "True" or "False")
- SS (adds the Start Stereotype character(s) as determined by the "Use extended << and >> characters" option)
- status
- stereotype
- stereotypehidden
- subtype
- type
- version
- visibility

用于连接器的属性形状

- 实际名称 - 与 "名称"相同，只是它不响应 "使用别名如果可用"设置
- 插件
  - 从调用的插件返回一个值插件
  函数；句法：
  插件：<addin_name>, <function_name>, <parameter> [, <parameter> ...]
  注记在 hasproperty() 参数中， Enterprise Architect需要哈希字符作为附加值：
  if(hasproperty("#ADDIN:MyAddin,MyValue#", "TheValue")) {
- 别名
- 图。作者
- diagram.connectornotation
- 图.手绘
- 图.mdgtype
- 图.mdgview
- 图名
- 图.刻板印象
- 图表类型
- 图.版本
- 方向
- 影响
- ES - 添加由 "使用扩展的 << 和 >> 字符"选项确定的结束构造型字符
- 警卫
- 根
- 小岛
- 姓名
- 旋转方向（ "上"、 "下"、 "左"、 "右"）

- 源.actualname - 与 '源.name' 相同，只是它不响应 '使用别名if Available' 设置
- 源.聚合
- 源.别名
- 源.可变的
- 源.约束
- 源.元素名
- 源.元素.刻板印象
- 源.metatype 这四个源.metatype属性的详细信息，请参见
- 源帮助.一般*Define Metamodel*约束帮助topic
- 源.metatype.specific
- 源.metatype.both
- 源.多样性
- 源.multiplicityisordered
- 源.name
- 源.qualifiers
- 源.RectangleNotation
- 源.刻板印象
- 源.targetscope
- SS - 添加由 使用扩展的 << 和 >> 字符”选项确定的开始构造型字符
- 刻板印象
- target.actualname - 与 ẗarget.name”相同，只是它不响应 使用别名如果可用”设置
- 目标.聚合
- 目标别名
- 目标可变
- 目标约束
- 元素
- 目标.元素.刻板印象
- 目标.元类型
- 目标.多重性
- target.multiplicityisordered
- 目标名称
- 目标限定符
- 目标.矩形符号
- 目标定型
- 目标.targetscope
- 触发器
- 类型
- 重量

子形状

当您使用形状脚本定义元素或连接器形状时，您可以从单独的组件构建形状，定义为子形状。使用子形状，您可以创建更类似于它们所代表的对象的复杂形状。

子形状布局

要设置布局类型，请使用布局属性，该属性必须在脚本的初始化属性部分中设置；换句话说，在调用任何方法之前。此属性的有效值为：

- LeftRight - 具有此布局的形状与子形状并排，第一个添加在左侧，随后的子形状添加到右侧
- 自上而下 - 将子形状垂直排列，第一个子形状添加到顶部，随后的子形状添加到下方
- 边框 - 这需要 addsubshape 方法的附加参数来指定子形状要占据的包含形状的哪个区域： N 、 E 、 S 、 W 或 CENTER；每个区域只能被一个子形状占据
  分配给E或 W 区域A子形状必须在其声明中指定其首选宽度属性，类似地，添加到N或S的子形状必须设置其首选高度属性；在这种情况下，这些属性的值被视为静态长度并且不缩放字形

示例

主要形状

{

布局类型= 自上而下”；

setfillcolor(0,0,255);

矩形（0,0,100,100）；

addsubshape("sub",50,100,20,0);

addsubshape("sub",50,100,30,-100);

addsubshape("sub",50,100,40,-200);

addsubshape("sub",50,100,50,-300);

形子

{

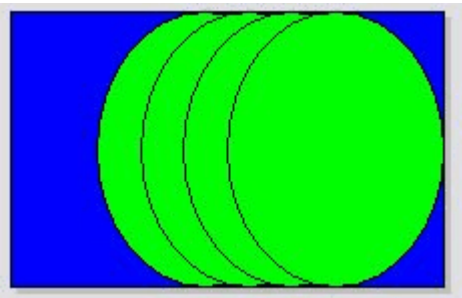
setfillcolor(0,255,0);

椭圆（0,0,100,100）；

}

}

脚本定义了这个形状：



为元素添加自定义分区

当您以正常的矩形格式在图表上显示元素时，可以在该框架内显示许多隔间以显示附加的特征，例如属性、操作和注记，使用图表 属性"和 元素"隔间可见性' 对话框。如果要显示其他添加的特征，例如相关元素或端口和部件，可以使用形状脚本将自定义隔间添加到元素的图表显示中。您通常会将此形状脚本脚本添加到配置文件中的构造型元素中。

创建自定义隔间后，您可以将链接的注记添加到元素以显示隔间的内容，就像您可以为元素的其他特征一样。

访问

在配置文件中定义构造型元素，并使用特殊属性 形状脚本"来指定添加自定义隔间的形状脚本。

功能区	设计>元素>特征>属性：[创建一个名为 _"image"的属性]>单击 初始值"字段中的...图标 设置>参考> UML类型>构造型：（选择或指定构造型）：形状脚本>分配
上下文菜单	在图表中，右键单击元素|特征|属性：[创建一个名为 '_image' 的属性] |单击 初始值"字段中的...
键盘快捷键	F9：[创建一个名为 ..."的属性] > 单击 初始值"字段中的浏览图标图标

向元素添加自定义隔间

此表提供有关创建定义自定义注记的形状脚本的注释，以及各种示例。

Process	Description
开发脚本	对于选定的构造型，打开形状编辑器。 在脚本中，将*shape main*替换为： • 形状 *ChildElement*或 • 形状相关元素 如果您愿意，可以保持*shape main* ，以调整主元素的某些属性（例如颜色）；但是，主要形状需要调用 DrawNativeShape()"才能正常工作。 此时，您可以使用 'HasProperty' 查询方法来搜索子元素或相关元素，以查找要在隔间中显示的特定属性（例如构造型）。形状脚本属性确定通过连接器链接到当前元素的元素A属性。 由形状脚本定义的每个单独的自定义隔间的可见性是使用 隔间可见性"对话框控制的。 ChildElement 隔间默认可见，可以使用隔间可见性选项隐藏，而 RelatedElement 隔间是隐藏的 默认情况下，必须使用隔离专区可见性选项显式启用。 还要注意，当子元素与父元素在图表上或不在图表上时（如示例1和 3 中所示），子元素可以显示在自定义隔间中。如果相关元素不在同一图表上（如示例 4 和 5 中所示），则它们只能在隔间中列出。
附上链接注记	您可以使用以下两种方法之一来创建链接的注记以显示自定义隔间内容： • 方法1（元素当前显示自定义隔间）- 突出显示自定义隔间中的相关或子

	元素名称，然后右键单击它并选择 创建链接注记"选项；自定义隔间会自动关闭，链接的注记会添加到列出该隔间中所有元素名称的图表中 • 方法 2（元素不一定显示自定义隔间）- 从元素的 公共"页面图表一个注记工具箱，并使用注释链接连接器将其链接到包含自定义隔间的元素 右键单击连接器并选择 将此注记链接到元素特征"选项，以显示 将注记链接到元素特征"对话框；单击 特征类型"字段中的下拉箭头，然后单击自定义隔间的名称，例如 属性"，然后单击确定按钮 该隔间的内容显示在注记中 在方法 2 中，如果显示隔间，则该方法不会隐藏该隔间。如果隔间已隐藏，建议您使用此方法。 您对隔间中的元素列表或其名称所做的任何更改都会立即反映在注记中，以保持显示信息的准确性。

脚本示例1：添加隔间而不调整父元素	//为子元素添加隔间。
	形状子元素
	{
	//选择子元素是否具有属性stereotype，如果有则设置
	//属性的隔间名称。
	if(HasProperty("stereotype", "属性"))
	{
	SetCompartmentName("属性");
	}
	//选择子元素是否具有公共范围，如果有，则添加+
	//子隔间的符号。
	如果（HasProperty（ 范围"， 公共"））
	{
	AppendCompartmentText("+");
	}
	//将子元素名称添加到子隔间。
	AppendCompartmentText("#NAME#");
	}
	
	属性属性检查所有子元素以查看它们是否具有 <<property>> 的形状脚本型。如果找到此构造型，则 SetCompartmentName"函数设置一个名为 属性"的隔间。
	然后脚本检查子元素是否具有 公共"范围，如果有，则附加 +"符号。
	最后，"AppendCompartmentText"函数将子元素的名称添加到隔间。
	如果 SetCompartmentName"已经声 了一个隔间，则属于同一隔间的任何其他子元素都会自动添加到该隔间，而无需声明新的隔间名称（即，所有具有构造型 <<property>> 的子元素最终进入 属性"隔间）。

脚本示例2: 调整父元素的颜色并添加子隔间	//形状主要影响父级
	主要形状
	{
	//设置父元素的颜色为红色
	setfillcolor(255,0,0);
	//绘制父母的原生形状
	绘制形状（ ）；

	}
	
	//形状将子分隔添加到父元素。
	形状子元素
	{
	if(HasProperty("stereotype", "part"))
	{
	SetCompartmentName("零件");
	}
	else if(HasProperty("stereotype", "mystereotype"))
	{
	SetCompartmentName("我的构造型");
	}
	
	AppendCompartmentText("#NAME#");
	}
	
	'shape main' 部分将主元素的颜色设置为红色，并根据原型子元素添加子隔间。
	该脚本检查子元素是否应用了原型值 "part"或 "mystereotype"。如果有多个子元素，具有 "part"和 "mystereotype"原型的组合，则会创建两个隔间，称为 "Parts"和 "My构造型"。
	为了显示隔间，必须调用 "AppendCompartmentText"将内容插入隔间。
	传递给 "SetCompartmentName"和 "AppendCompartmentText"的值不能包含换行符。

脚本示例3: 如果子元素在图上不可见，则仅列出隔间中的子元素	形状子元素
	{
	//选择子元素是否在图表上。
	if(hasproperty("IsVisible", " False "))
	{
	//为零件创建一个隔间。
	如果（有属性（ 类型"， 部分"））
	{
	SetCompartmentName("零件");
	}
	//为端口创建一个隔间。
	else if(hasproperty("type", "port"))
	{
	SetCompartmentName("端口");
	}
	//将子元素名称添加到隔间。
	AppendCompartmentText("#NAME#");
	}
	}

	此脚本为属于当前元素但在当前图表上不可见的端口和部件元素添加自定义隔间。 如果子元素属性True元素则返回False 。 如果子元素已经在图表上可见，这可以用来防止子元素被列在自定义隔间中，从而避免显示冗余信息。

脚本示例4：从拥有形状脚本的元素中显示作为依赖连接器目标的元素	形状相关元素
	{
	//选择我们正在处理的当前连接器是否有
	//依赖类型。
	if(HasProperty("连接器.类型", "依赖"))
	{
	//选择如果我们当前正在检查的元素是
	//当前连接器的目标。
	if(HasProperty("元素.IsTarget"))
	{
	//设置车厢名称
	SetCompartmentName("dependsOn");
	if(HasProperty("元素.构造型", ""))
	{
	}
	else
	{
	AppendCompartmentText("«#Element.Stereotype#»");
	}
	AppendCompartmentText("#Element.Name#");
	}
	}
	}
	
	使用此脚本，如果 Class1 具有带有 "RelatedElement"形状脚本脚本的构造型，并且 Class1 是目标 Class2 的 Dependency 连接器的源，则名称 Class2 将显示在类的隔间1 ，称为 "dependsOn"。

脚本示例5: 在一个元素的一个隔间内显示一个已实现接口的列表	形状相关元素
	{
	//选择当前正在处理的连接器是否为Realization
	if(HasProperty("连接器.类型", "实现"))
	{
	//只有相关元素we才显示这个隔间
	//正在检查的是有这个的连接器的目标
	//形状脚本元素作为源
	if(HasProperty("元素.IsTarget"))
	{
	//如果元素是接口，则显示在
	//'realizedInterfaces' 隔间

	if(HasProperty("元素.类型", "接口"))
	{
	SetCompartmentName("realizedInterfaces");
	AppendCompartmentText("#Element.Name#");
	}
	}
	}
	}
	
	如果元素类1具有此形状脚本，并且是元素接口1的实现连接器的源，则名称'接口1 ' 将显示在类1的 'realizedInterfaces' 隔间中。

注记

- 如果您在分隔名称中使用标点符号，则会在保存脚本时将其删除；例如： 端口、部件和属性”变为 端口部件和属性”
- 'RelatedElement'形状脚本具有检查连接器和连接器另一端的元素的扩展功能；它们仅适用于元素并且仅用于检索要在该元素的隔间内显示的信息

显示复合图表

您可以将元素定义为正在复合（使用 新图表|组合复合结构图表”上下文菜单选项），在这种情况下，元素有一个子复合图来描述元素的子结构。您还可以使用上下文菜单选项在元素复合元素元素通常，重新定义复合元素外观的形状脚本脚本会有效规避这些选项的效果，但您可以编辑脚本以响应 在分区中显示复合图表”选项并在中间部分显示子复合图元素。

为了显示复合图，脚本需要 边框”的布局类型，在绘制时将复合图添加到主形状的中心子形状。因此，定义形状脚本的语句是：

```
主要形状
{
布局类型="边框";
if(HasProperty("ShowComposedDiagram", "true"))
{
addsubshape("ComposedDiagram", "CENTER");
}
形状组合图
{
绘制组合图（）；
}
}
```

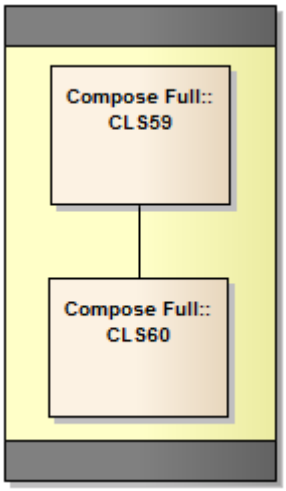
例子

包含组合图的形状脚本的示例是：

```
形状主要
{
//设置边框类型
布局类型="边框";
//设置奶油填充颜色
setfillcolor(255, 255, 200);
//为object划一个基本矩形。
矩形（0, 0, 100, 100）；
//在形状的顶部添加一些填充
addsubshape("填充", " N ");
//选择上下文菜单选项的设置
if(HasProperty("ShowComposedDiagram", "true"))
{
//将组合图添加到object的中心
addsubshape("ComposedDiagram", "CENTER");
}
//在形状的底部添加一些填充。
```

addsubshape("填充", " S ");
形状填充
{
//设置这个元素的高度
首选高度 = 20;
//设置填充颜色为灰色
setfillcolor(128, 128, 128);
//划一个占据object宽度的矩形
// 高度为 20 像素。
矩形（0, 0, 100, 100）；
}
形状组合图
{
//划组合图。
绘制组合图（ ）；
}
}

此脚本生成形状：



目前仅支持组合图作为主形状的中心子形状。将图表添加到任何其他位置将导致组合图表无法正确绘制或根本不绘制。该图可以是子形状的子形状，但前提是父形状和子形状都具有“中心”方向。例如：

//这个shapescript很好，因为shape E是shape C的中心，也就是shape D的中心；也就是说，所有指向
//DrawComposedDiagram 的形状都是“CENTER”。

主要形状
{
layouttype = "边框";
矩形 (0, 0, 100, 100);
添加子形状（“D”，“CENTER”）；
形状D

```
{
布局类型= 边框″；
添加子形状（“C”，“CENTER”）；
形状 C
{
布局类型= 边框″；
添加子形状（“ E ”，“CENTER”）；
addsubshape（ 填充″，“ N ”）；
addsubshape（ 填充″，“ S ”）；
形状E
{
绘制组合图（ ）；
}
形状填充
{
首选高度 = 20;
setfillcolor (10, 30, 80);
矩形 (0, 0, 100, 100);
}
}
}
}

//这个shapescript不好 - 形状E是“CENTER”，形状C是“S ”，形状D是“CENTER”；因为形状 C 的方向是“S ”
//图表不会绘制。

主要形状
{
layouttype = "边框";
矩形 (0, 0, 100, 100);
添加子形状（“ D ”，“CENTER”）；
形状D
{
布局类型= 边框″；
添加子形状（“C”，“ S ”）； //<- 这很糟糕，DrawComposedDiagram 调用的所有父子形状都必须是
// 面向“中心”
形状 C
{
布局类型= 边框″；
添加子形状（“ E ”，“CENTER”）；
addsubshape（ 填充″，“ N ”）；
addsubshape（ 填充″，“ S ”）；
```

```
形状E
{
绘制组合图（ ）；
}
形状填充
{
首选高度 = 20;
setfillcolor (10, 30, 80);
矩形 (0, 0, 100, 100);
}
}
}
}
```

注记

- 为了显示复合图，'New图表|应在图表中元素的 "上下文"菜单上选择 "显示复合图表"选项
- 组合图以自然大小显示，因此无法将父元素调整为小于组合图的大小

保留名称

当你写一个形状脚本脚本时，有一些术语是保留的，因为它们在脚本中具有特殊的含义；将它们用于特定目的。

元素

元素（例如类、状态或事件）具有这些为形状部分保留的名称。

Name	Description
主要形状	主要形状是整体形状。
形状标签	形状标签给形状一个分离的标签。
装饰 <标识符>	装饰为形状提供由 <identifier> 中的名称定义的装饰。
形状子元素	允许基于属于当前元素的子元素添加自定义隔间。
形状相关元素	允许根据属于当前元素的相关元素添加自定义隔间。

连接器

连接器（如关联、依赖或概括）具有形状的这些保留名称。

Name	Description
主要形状	主要形状是整体形状。
形状源	源形状是连接器源端的一个额外形状。
形状目标	目标形状是连接器目标端的额外形状。
形状 LeftTopLabel	Shapes 为左上角的连接器定义了一个分离的标签。
形状 MiddleTopLabel	Shapes 为中间顶部的连接器定义了一个分离的标签。
形状 RightTopLabel	Shapes 为右上角的连接器定义了一个分离的标签。
形状 LeftBottomLabel	Shapes 为左下角的连接器定义了一个分离的标签。
形状 MiddleBottomLabel	Shapes 为中间底部的连接器定义了一个分离的标签。
形状 RightBottomLabel	Shapes 为右下角的连接器定义了一个分离的标签。

语法文法

形状脚本A一部分可能非常复杂，包含许多命令和参数。此表提供了形状脚本结构的细分，说明了命令和参数是如何构造的。第一个条目是顶层声明，随后的条目显示了依次更详细的组件的组成。

语法符号

- * = 零个或多个
- + = 一个或多个
- | = 或
- ; = 终结者

Symbol	Description
形状脚本 ::=	<形状>*;
形状::=	<ShapeDeclaration> <ShapeBody>;
形状声明 ::=	<形状类型> <形状名称>;
形状类型 ::=	形状”	装修”	标签”;
形状名称 ::=	<保留形状名称>	<字符串文字>;
保留形状名称 ::=	有关完整的保留形状列表，请参阅保留名称。
形状体 ::=	"{" <InitialisationAttributeAssignment>* <DrawingStatement>* <SubShape>* "}";
初始化属性分配 ::=	<属性> "=" <值> ";";
属性 ::=	有关属性名称的完整列表，请参阅形状属性。
绘图声明 ::=	<IfElseSection>	<方法>;
IfElseSection ::=	"if" "("<QueryExpression>")" <TrueSection> (<ElseSection>);
查询表达式 ::=	<QueryName> "(" <ParameterList> ")"; 有关查询及其参数的描述，请参见方法。
查询名称 ::=	有关可能的查询名称，请参阅查询。
TrueSection ::=	"{" <绘图说明>* "}"
其他部分 ::=	" else " "{" <DrawingStatement>* "}"
方法 ::=	<方法名> "(" <参数列表> ")" ";";
方法名 ::=	有关方法名称的完整列表，请参阅绘图方法。

示例脚本

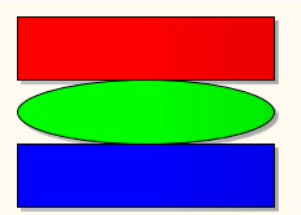
您可以使用形状脚本创建各种形状、效果和文本语句，以增强您创建的元素和连接器的外观和信息价值。此处提供了此类脚本的一些示例。

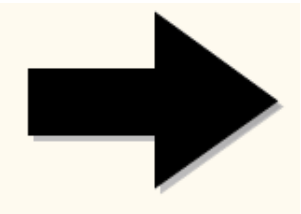
访问

功能区	设置 >参考> UML类型 >构造型（指定构造型）：形状脚本+ 赋值，或 设置>参考> UML类型>构造型（指定构造型）：形状脚本+编辑

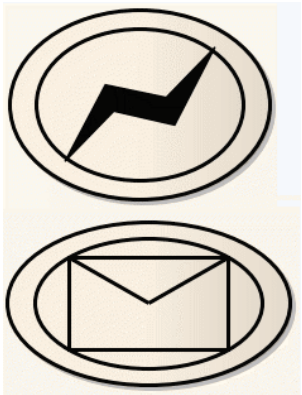
例子

Shape	Script

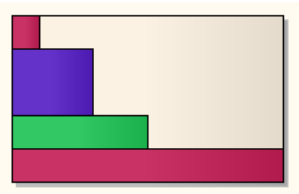
	// 基本形状
	主要形状
	{
	setfillcolor(255, 0, 0); // (R ,G, B)
	矩形（0, 0, 90, 30）； // (x1, y1 ,x2,y2)
	
	setfillcolor(0, 255, 0); // (R ,G, B)
	椭圆（0, 30, 90, 60）； // (x1, y1 ,x2,y2)
	
	setfillcolor(0, 0, 255); // (R ,G, B)
	矩形（0、60、90、90）； // (x1, y1 ,x2,y2)
	}

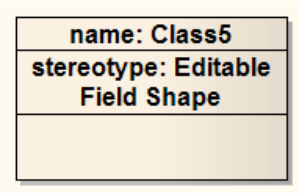
	// 单一条件形状
	主要形状
	{
	if (触发器("", "Link"))
	{
	// 只有当object有标记值时才绘制Trigger=Link
	// 设置路径的填充颜色
	setfillcolor(0, 0, 0);
	开始路径（）； //开始追踪路径
	移动到（23、40）；
	lineto(23, 60);
	lineto(50, 60);
	lineto(50, 76);
	lineto(76, 50);

	lineto(50, 23);
	lineto(50, 40);
	结束路径（ ）； // 结束追踪路径
	// 用填充颜色填充跟踪的路径
	填充行程路径（ ）；
	返回;
	}
	}

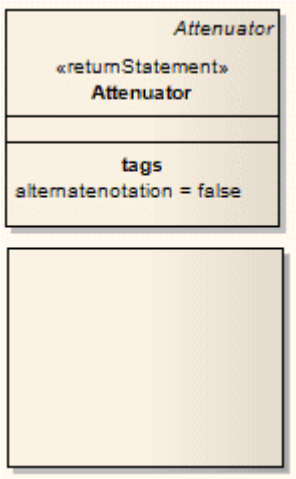
	// 多条件形状
	主要形状
	{
	开始路径（ ）；
	椭圆（0, 0, 100, 100）；
	结束路径（ ）；
	填充行程路径（ ）；
	椭圆（3, 3, 97,97）；
	
	if (触发器("", "None"))
	{
	返回;
	}
	
	if (触发器("", "Error"))
	{
	setfillcolor(0, 0, 0);
	开始路径（ ）；
	移动（23, 77）；
	lineto(37, 40);
	lineto(60, 47);
	线托（77、23）；
	lineto(63, 60);
	lineto(40, 53);
	线（23, 77）；
	结束路径（ ）；
	填充行程路径（ ）；
	返回;
	}
	if (触发器("", "信息"))
	{
	矩形（22、22、78、78）；
	移动到（22、22）；
	lineto(50, 50);
	线（78, 22）；
	返回;

	} }

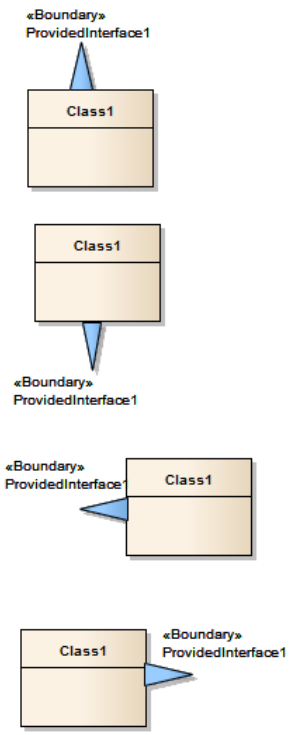
	// 子形状
	主要形状
	{
	矩形（0, 0, 100, 100）；
	
	addsubshape("红色", 10, 20);
	addsubshape("蓝色", 30, 40);
	addsubshape("绿色", 50, 20);
	addsubshape("红色", 100, 20);
	
	形状红色
	{
	setfillcolor(200, 50, 100);
	矩形（0, 0, 100, 100）；
	}
	
	形状 蓝色
	{
	setfillcolor(100, 50, 200);
	矩形（0, 0, 100, 100）；
	}
	
	形状绿色
	{
	setfillcolor(50, 200, 100);
	矩形（0, 0, 100, 100）；
	}
	}

	// 可编辑的字段形状
	主要形状
	{
	矩形（0, 0, 100, 100）；
	addsubshape("namecompartment", 100, 20);
	addsubshape("stereotypecompartment", 100, 40);
	
	形状名称隔间
	{
	h_align = "中心";
	可编辑字段= 名称”；
	矩形（0, 0, 100, 100）；
	println("姓名：#姓名#");

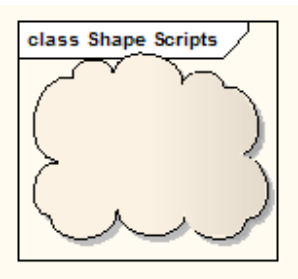
	}
	
	形状定型隔间
	{
	h_align = "中心";
	editablefield = "刻板印象";
	矩形（0, 0, 100, 100）；
	println("刻板印象：#stereotype#");
	}
	}

	// 返回语句形状
	主要形状
	{
	if (hasTag("alternatenotation", " false "))
	{
	//绘制ea的内置字形
	绘制形状（ ）；
	//用return语句退出脚本
	返回;
	}
	else
	{
	//替代符号命令
	//...
	矩形（0, 0, 100, 100）；
	}
	}

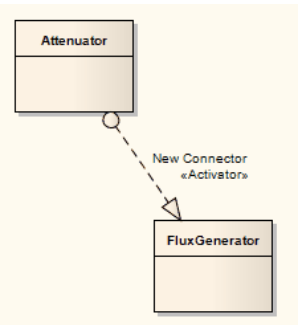
	//嵌入元素形状在父边缘的位置
	主要形状
	{
	定义大小（60,60）；
	开始路径（ ）；
	if(hasproperty("parentedge","top"))
	{
	移动到（0,100）；
	线（50,0）；
	线到（100,100）；
	}
	if(hasproperty("parentedge","bottom"))
	{
	移动到（0,0）；
	线（50,100）；
	lineto(100,0);




```
}
if(hasproperty("parentedge","left"))
{
移动到（100,0）；
线（0,50）；
线到（100,100）；
}
if(hasproperty("parentedge","right"))
{
移动到（0,0）；
线（100,50）；
线（0,100）；
}
结束路径（）；
设置填充颜色（153,204,255）；
填充行程路径（）；
}
```

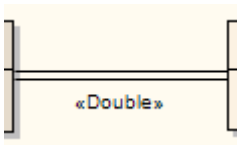


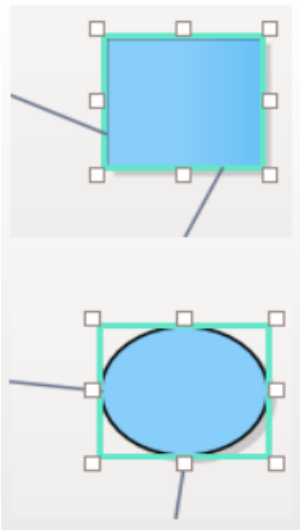

```
// 云路径示例形状
主要形状
{
开始云路径（）；
矩形（0, 0, 100, 100）；
结束路径（）；
FillAndStrokePath();
}
```



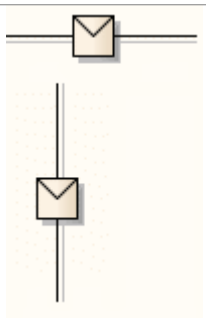

```
// 连接器形状
主要形状
{
// 画一条虚线
无影=真；
setlinestyle("DASH");
移动到（0,0）；
lineto(100,0);
}

形状源
{
// 在源端画一个圆
可旋转=真；
开始路径（）；
椭圆（0,6,12,-6）；
结束路径（）；
```

	填充行程路径（）；
	}
	
	形状目标
	{
	// 在目标端画一个箭头
	可旋转=真；
	开始路径（）；
	移动到（0,0）；
	线（16,6）；
	线托（16，-6）；
	结束路径（）；
	填充行程路径（）；
	}

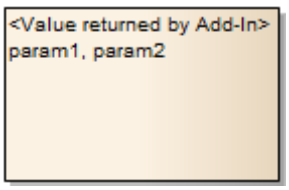
	// 双线
	主要形状
	{
	setlinestyle("DOUBLE");
	移动到（0,0）；
	lineto(100,0);
	}

	// 设置附件模式
	形状 主要的
	{
	if (hasproperty ("rectanglenotation" , " 1 "))
	{
	SetAttachmentMode ("正常");
	矩形(0 , 0 , 100 , 100);
	}
	else
	{
	SetAttachmentMode ("钻石");
	椭圆(0 , 0 , 100 , 100);
	}
	}
	
	在此示例中，如果元素启用了 矩形表示法”，则 SetAttachmentMode("normal") 命令将允许连接器连接到元素（第一个形状）每个边缘的任意点。如果元素关闭了 Rectangle Notation”，SetAttachmentMode("diamond") 命令将只允许连接器连接到元素每条边的中心点；也就是说，呈菱形（第二种形状）。您不能将附着点移动到该边的其他位置。

	// 旋转方向 主要形状

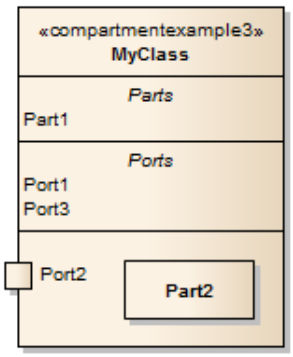


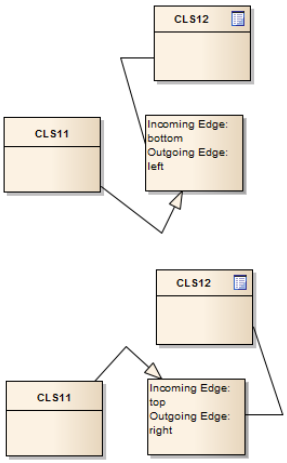
```
{
移动到（0,0）；
lineto(100,0);
固定区域（40，-10,60,10）；
矩形（40，-10,60,10）；
if(hasproperty("旋转方向","向上"))
{
移动到（60，-10）；
线（50,0）；
线托（60,10）；
}
if(hasproperty("旋转方向","向下"))
{
移动到（40，-10）；
线（50,0）；
线（40,10）；
}
if(hasproperty("旋转方向","左"))
{
移动到（40，-10）；
线（50,0）；
线（60，-10）；
}
if(hasproperty("旋转方向","右"))
{
移动（40,10）；
线（50,0）；
线托（60,10）；
}
}
```

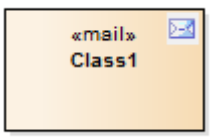



```
// 获取加载项返回A值
主要形状
{
//划一个简单的矩形
矩形（0,0,100,100）；

//打印从插件
返回的string值插件
 我的插件"，
//函数 "MyExample"有两个string参数
Print("#ADDIN:MyAddin, MyExample, param1, param2#");
}

// 附加功能的方法签名：
```

	// Public函数MyExample(存储库As EA.Repository , // eaGuid As字符串, args As Variant) As Variant

	// 添加基于子元素的自定义分区
	// 或相关元素
	
	（请参阅向元素帮助添加自定义分区帮助）

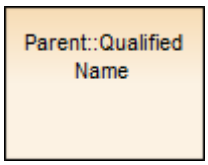
	// 返回连接器的输入和输出边缘
	// 进入和离开一个对象
	主要形状
	{
	//划一个简单的矩形
	矩形（0,0,100,100）；
	
	//打印元素上的传入边
	Print("传入边缘：#incomingedge#\n");
	
	//在元素上打印出边
	Print("输出边缘：#outgoingedge#\n");
	}

	// 在默认的顶部画A装饰图标
	// 元素形状
	装饰邮件
	{
	方向=“NE”；
	image ("图标图像", 0, 0, 100, 100);
	// "icon image" 是加载到图像管理器中的 16x16 图像的名称
	}

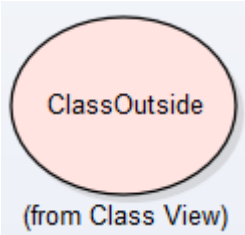
	// A文件中绘制图像和可编辑的名称字段
	主要形状
	{
	addsubshape ("theimage", 100, 100);
	addsubshape ("namecompartment", 100, 100);
	
	塑造形象
	{
	image ("元素图像", 0, 0, 100, 100);
	// "元素" 是加载到图像管理器中的图像的名称

	}
	形状名称隔间
	{
	h_align = "中心";
	可编辑字段= 名称”；
	println("#name#");
	}
	}

	// 检查是否A复合元素图标
	// 如果是这样，画一个
	装修补偿
	{
	方向=“SE”；
	if(hasproperty("IsDrawCompositeLinkIcon","true"))
	{
	开始路径（ ）；
	椭圆（-80,29,-10,71）；
	椭圆（10,29,80,71）；
	移动（-10,50）；
	线托（10,50）；
	结束路径（ ）；
	行程路径（ ）；
	}
	}

	// 允许A形状脚本显示完整的对象
	// 拥有元素的名称，包括拥有元素
	// 并且拥有包，当图表属性时
	// '禁用完全范围的对象名称' 选项是
	// 取消选择，就像没有A元素一样
	// 形状脚本。
	主要形状
	{
	布局类型= 边框”；
	矩形 (0, 0, 100, 100);
	addsubshape（ 填充”，“N”）；
	addsubshape（ 名称”， 中心”）；
	形状填充
	{
	首选高度=8；
	}
	形状名称
	{
	v_align="顶部";

	h_align= 中心”； printwrapped ("#qualifiedname#"); } }

	// 元素时显示所属包的名称
	// 用于不在该软件包A图表上，并且
	// 选择了图表属性 显示命名空间”选项。
	主要形状
	{
	布局类型= 边框”；
	v_align = "中心";
	h_align = "中心";
	椭圆 (0, 0, 100, 100);
	printwrapped ("#name#");
	addsubshape （ 路径”，“ S ”）；
	形状路径
	{
	v_align="顶部";
	h_align= 中心”；
	if (hasproperty ("packagepath", ""))
	{
	}
	else
	{
	printwrapped ("(来自#packagepath#)");
	}
	}
	}

	// 在右上角显示元素概括的列表。
	//注记：只列出不在当前图上的元素。
	形状 主要的
	{
	layouttype = "边框" ;
	矩形(0 , 0 , 100 , 100);
	添加子形状 （ 名称” ， 中心” ）;
	addsubshape （ 父母” ，“ N ”）;
	
	形状 姓名
	{
	v_align = "居中" ;
	h_align = "居中" ;
	粗体=真 ;
	打印 （ “#name#” ）；

```
        }

        形状 父母
        {
                v_align = "顶部" ;
                h_align = "右" ;
                斜体=真 ;
                打印 ( “#hiddenparents#” ) ;
        }
}
```

标记值类型

在使用标记值时，您可以根据预定义的、系统提供的标记值类型创建自己的、自定义的标记值值。有了这些，您可以创建：

- 复杂且基于预定义类型的标记值，有或没有标签过滤器
- 结构化标记值是复合的，包含其他标记值
- 从各种参考数据库库表返回标记值
- 将用户提供的数据插入到文本string中的屏蔽标记值，例如提示行或字段名称

通过将任何类型的标记值添加到配置文件中的构造型元素，您可以为标记元素在技术中的出现和行为方式定义附加元信息。标记值由构造型元素的属性标识。

注记

- 您可以使用 设置>模型>传输>导出参考”和 导入数据”功能区选项在模型之间传输标记值类型定义；标记参考标记值类型导出为属性类型

从预定义类型标记值类型

当您在使用标记值时，您可能需要使用结构化的标记值；也就是说，标记值以特定格式捕获和呈现更复杂的信息。可以为您的模型轻松创建此类标签的基本类型（您在窗口的 标签”页面创建标签时调用的类型），因为您可以将自定义的结构化属性标记值标记值基于预定义的标记值类型和过滤器范围。

访问

功能区	设置 >参考> UML类型 >标记值类型

创建自定义结构化标记值类型

字段/按钮	描述
标签名称	类型为您的新标记值类型提供的适当名称。
描述	（可选）键入标记值类型的简短说明或用途。
细节	复制和粘贴或键入作为新的标记值类型基础的预定义结构化标记值类型的语法。
节省	点击此按钮保存新的结构化标记值类型。 标记值类型显示在定义的标签类型列表中。
新的	或者，单击此按钮以清除字段，以便您可以输入另一个新的标记值类型的信息。

预定义的结构化类型

标记值定义了一个模型元素具有的范围广泛的属性和特性，而这些属性中有些是复杂的值。例如，您可能希望您的用户在上限和下限之间选择一个值（使用 旋转"箭头）、设置日期、从调色板中选择一种颜色，或者通过检查清单进行操作。

您可以从许多预定义的标记值类型和过滤器中的任何一个创建这些复杂的标记值，其中一些可能是您自己创建的（ 设置 >参考> UML类型 >标记值类型"）。

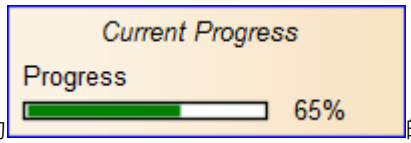
标记值类型格式

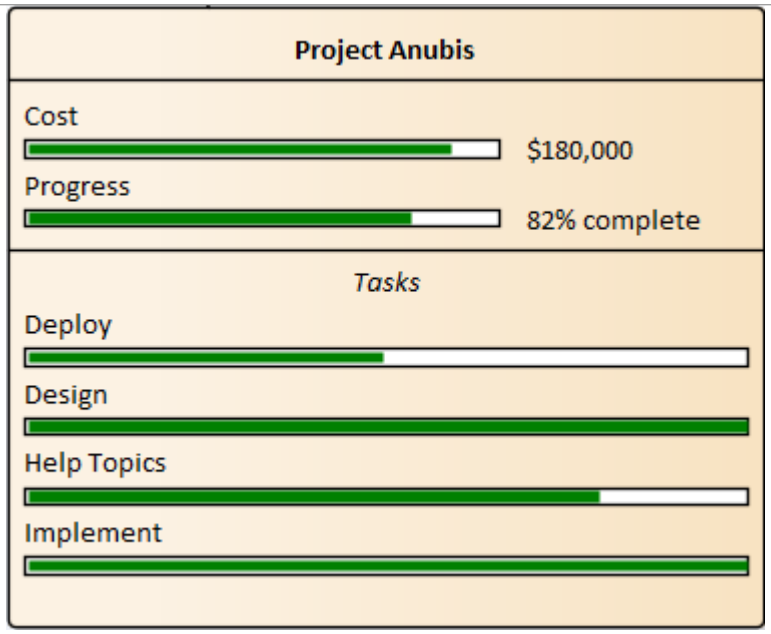
对于每个标记值类型，描述包括创建用于标记值的初始值的语法。名称和格式区分大小写。

标记值类型	格式
Addin广播	类型=AddinBroadcast； 值=您的插件名称； 用于：允许插件 要响应编辑此标记值的尝试，可以显示一个对话框，在该对话框中可以编辑值和注记。
布尔值	类型=布尔值； 默认值=Val； 用于：提供True或False的输入，其中任何一个都可以是默认值。
清单	类型=检查清单； 值=Val1，Val2，Val3； 用于：在批准或执行某项操作之前创建要完成或满足的事项清单。 Val1、Val2、Val3 等指定清单项，每一项都通过属性窗口的 标签"选项卡呈现，并带有一个复选框；在选中每个复选框之前，该标签的值为 不完整"，此时该值为 完全"。 例如： 类型=检查清单； 值=更改是否解决了给定的任务\问题，代码是否有足够的错误处理，代码是否有意义，代码是否符合编码约定； 虽然元素标记值隔间和 标签"选项卡窗口字段显示值 完全"或 不完整"，但文档和 Web 报告将显示清单项目列表和每个项目的状态（ True表示选中，False表示未选中）。
分类器	类型=分类器； 值 = 类型 1，类型 2； 刻板印象=刻板印象1； 用于：已弃用- 使用**RefGUID**和**RefGUIDList**
颜色	类型=颜色； 默认值=Val； 用于：从颜色选择器菜单输入颜色值，其中该值是颜色的 Hex RGB 值。 例如，蓝色的 Hex RGB 为 0000FF，而绿色的 Hex RGB 为 00FF00。

常量	类型=常量； 默认值=Val； 用于：创建只读常量值。
风俗	类型=自定义； 用于：使用掩码值为预定义类型创建自己的模板。
日期	类型=日期； 用于：从日历菜单中输入标记值的日期。
约会时间	类型=日期时间； 用于：已弃用-使用**日期** 从日历菜单中输入标记值日期。
图表参考	类型=图表参考 用于：参考模型中的图表。
目录	类型=目录； 默认值=Val； 用于：从浏览器输入目录路径。 您可以将默认目录路径设置为string值。
枚举	类型=枚举； 值=Val1，Val2，Val3； 默认=Val2； 用于：定义一个逗号分隔的列表，其中 Val1、Val2 和 Val3 表示列表中的值，Default 表示列表的默认值。
文件	类型=文件； 默认值=Val； 用于：从文件浏览器对话框输入文件名。命名文件可以在其默认应用程序中启动。 您可以将默认文件设置为包含文件路径和文件名的string 。
浮点数、十进制、双精度	类型=浮动； 类型=十进制； 类型=双； 默认值=Val； 用于：输入浮点、十进制或双精度值。这些类型都映射到相同类型的数据。 您可以为任何或所有这些设置默认值。
图像参考	类型=ImageRef； 用于：提供指向图像管理器中保存的图像文件的链接。
整数	类型=整数； 默认值=Val； 用于：输入一个整数和一个默认值

备忘录	类型=备忘录； 用于：为标签输入大而复杂的值。

进度条	类型=进度条；
	隔间=<名称>； - 设置要在其中显示进度条的隔间的名称；多个标记值可以在一个隔间添加进度条
	文本=<文本>; - 在进度条的右侧显示 <text>；要使用文本显示标记的值，请使用 #VALUE#，例如 $#VALUE# 或 #VALUE#%
	最小值=n; - 设置可以在进度条中显示的最小值（必须是整数）
	最大值=n; - 设置进度条可以显示的最大值（必须是整数）
	用于：当元素显示在图表上并且在图表 属性"对话框的 元素"页面上启用了标签隔间时，在元素的隔间中显示进度条。标签名称显示在进度条上方，作为其标签。
	• 如果 MinVal 或 MaxVal 均未设置，则进度条的默认值为 0 和 100
	• 如果设置了 MinVal 但未设置 MaxVal，则最大值默认为 MinVal+100
	• 如果设置了 MaxVal 但未设置 MinVal，则最小值默认为 0
	• 如果同时设置了 MinVal 和 MaxVal，则 MinVal 必须低于 MaxVal
	例子：
	隔间=当前进度；
	类型=进度条；
	文字=#VALUE#%;
	 
	
	当在名为 的标记中使用 ProgressBarTag1，其值设置为 65。
	
	类型=进度条；
	最小值=1000；
	最大值=100000；
	文本=$ #VALUE#;
	 
	
	在名为 Progress 的标签中使用时，其值设置为 4530。
	
	具有多个进度条的元素。

	
RefGUID	类型=RefGUID； 值 = 类型 1，类型 2； 刻板印象=刻板印象1； 或者 类型=RefGUID； 元类型=类型； 用于：通过指定元素的GUID来参考模型中的元素，其中： • Type1 和 Type2 指定一个或多个允许的图表对象（例如类、部件、属性或操作） • Stereotype1 表示允许的刻板印象 元类型可用于引用分类器或属性类型： • 元类型=分类器；呈现所有企业架构师定义的分类器类型以供选择 • 元类型=属性；呈现所有端口、部件和属性以供选择 您可以通过在属性窗口中单击标记值的 ... 按钮来设置该类型的标记值的分类器、属性或操作。 您也可以右键单击属性窗口中的RefGUID标记值名称，选择 在项目中查找浏览器"选项，在浏览器窗口中定位被引用object 。 打印RefGUID标记值元素值时，形状脚本将打印引用的名称。
RefGUIDList	类型=RefGUIDList； 值 = 类型 1，类型 2； 刻板印象=刻板印象1； 或者 类型=RefGUIDList； 元类型=类型； 用于：通过指定每个元素的GUID来参考模型中的元素列表，其中： • Type1 和 Type2 指定一个或多个允许的图表对象（例如类或部件） • Stereotype1 表示允许的刻板印象 元类型可用于引用分类器或属性类型：

	• 元类型=分类器；呈现所有企业架构师定义的分类器类型以供选择 • 元类型=属性；呈现所有端口、部件和属性以供选择 您可以通过单击窗口 标签"选项卡中属性标签标记值的  按钮来设置此类标签标记值的分类器、属性或操作。
旋转	类型=旋转； 下界=x; 上界=x; 默认值=Val； 用于：创建一个以 LowerBound 为最小值、UpperBound 为最大值的旋转控件。 您还可以在该范围内设置默认值。
字符串	类型=字符串； 默认值=Val； 用于：输入一个string值，最长为 255 个字符，以及一个默认文本string 。 对于较长的文本，使用 Type=Memo。
时间	类型=时间； 用于：输入标记值的时间。
时间戳	类型=时间戳； 用于：从日历菜单中输入标记值的日期和时间。
网址	类型=网址； 默认值=Val； 用于：输入 Web URL。 URL 应以： • 'http://' • ' https ://' 或 • '万维网。' 您可以将默认 URL 设置为string值。

标签过滤器

您可以使用过滤器来限制可以应用标记值的位置。

过滤器	格式
适用于	适用于 = 类型 1，类型 2； 描述：限制此标签可以应用到的元素类型，其中 Type1 和 Type2 是有效类型。 可能的值为： • 所有元素类型 • 所有连接器类型 • 属性

	• 操作和 • 操作参数
基本刻板印象	BaseStereotype=S1,S2; 描述：限制此标签所属的构造型，其中 S1 和 S2 是允许的构造型。

注记

当使用标记值为构造型定义 属性"时，您可以通过使用过滤器 BaseStereotype"并指定不存在的构造型来防止该标记值（再次）添加到元素。例如，"BaseStereotype=NotAvailable；"。

这样就可以定义标记值的类型，但是在给任何元素添加新的标记值时，该标记值不会出现在下拉列表中。

创建自定义屏蔽标记值类型

如果您正在创建自定义的预定义标记值类型，则可以通过定义将数据格式化为模板的掩码，在设计模型以接受数据条目时获得极大的灵活性。

访问

功能区	设置 >参考> UML类型 >标记值类型

创建掩蔽标记值类型

字段	行动
标签名称	类型掩码标记值类型的适当名称。
描述	（可选）键入标记值类型的描述或用途。
细节	类型或复制粘贴标记值结构： 类型=自定义； 掩码=<掩码值>; 模板=<模板文本>; 掩码值在下一个表中进行解释，并通过示例演示如何使用模板。 模板文本定义了在每次使用此自定义标记值时要显示的信息，例如数据的字段名称和提示。
节省	单击此按钮可保存新的屏蔽标记值类型。 标记值类型显示在定义的标签类型列表中。
新的	或者，单击此按钮以清除字段，以便您可以输入另一个新的标记值类型的信息。

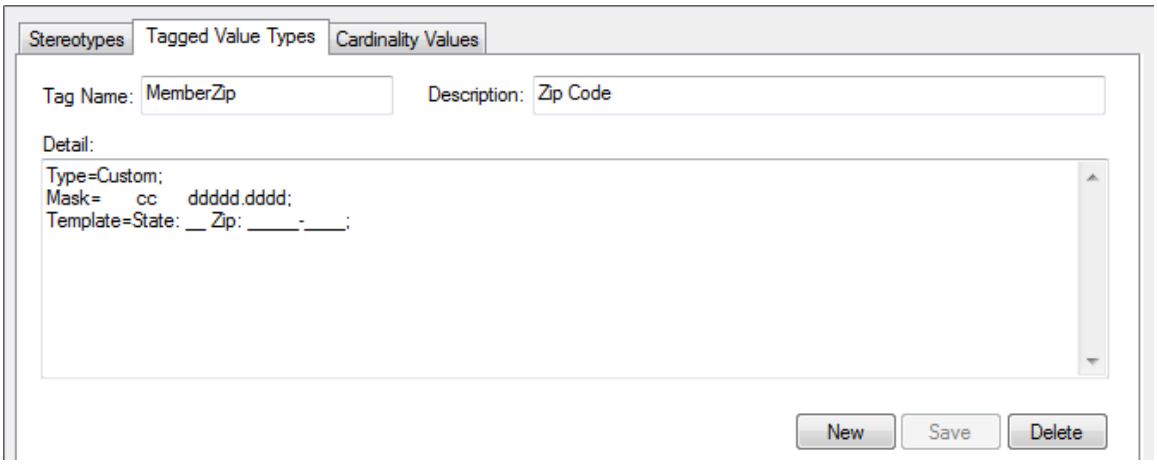
掩码值

定义掩码标记值类型中的掩码格式时，请使用以下字符：

面具	行动
D	仅在此字符空间中显示一个数字。
d	仅在此字符空间中显示数字或空格。
+	在此字符空间中显示 +、- 或空格。

C	仅在此字符空间中显示一个字母。
C	仅在此字符空间中显示字母或空格。
A	在此字符空间中显示任何字母数字字符。
一个	在此字符空间中显示任何字母数字字符或空格。
.或 <空格>	留下一个字符空间，由 Template 参数中的文本填充。使用点可能更容易查看您设置了多少空格。

示例



在图中，Mask 参数首先定义了七个空格，这些空格被 Template 参数定义的字符占据。

Mask 中的前两个可见字符均由小写 c 表示，表示用户可以输入字母字符或空格的信息。

接下来的六个空格再次表示模板定义的字符，后面是五个分别由ad表示的字符，表示用户可以输入数字或空格形式的数据。点标记一个空格，用模板中的连字符填充，后跟四个 ds (数字或空格) 。

Template 语法为 Mask 参数定义了模板，填充了 Mask 中的空白。文本是每次使用该标记值时要打印的信息；下划线值表示用户输入的数据将占用的字符空间，如 掩码”选项中定义的那样。

创建参考标记值

使用标记值时，您可能希望使用参考数据标记值，它用于返回Enterprise Architect参考表中保存的值。此类标记值的基本类型（您在属性窗口的标记页面中创建标记时调用的类型）可以很容易地专门为您的模型创建，因为您可以将自定义的参考标记值类型建立在一个范围内预定义的标记值类型和过滤器。

访问

功能区	设置 >参考> UML类型 >标记值类型

创建自定义参考标记值类型

字段/按钮	描述
标签名称	类型新标记值类型的适当名称。
描述	（可选）键入标记值类型的描述或用途。
细节	复制并粘贴或键入作为新标记值类型基础的预定义参考标记值类型的语法。
节省	点击此按钮保存新的参考标记值类型。 标记值类型显示在定义的标签类型列表中。
新的	或者，单击此按钮以清除字段，以便您可以输入另一个新的标记值类型的信息。

注记

- 如果在创建标记值类型后参考数据中的值发生变化，您必须重新加载系统以反映标记值类型的变化

预定义参考数据类型

如果您想创建自己的、自定义的参考标记值，您可以基于一系列预定义的参考标记值类型。每个预定义的参考标记值类型都返回特定参考数据表中保存的值。

标记值类型

每个描述都包括创建用于标记值的初始值的语法。标记值类型和格式条目区分大小写。

标记值类型	格式
作者	类型=枚举； 列表=作者； 返回为模型定义的数据的下拉列表：作者。
基数	类型=枚举； 列表=基数； 返回为模型定义的数据的下拉列表：基数类型。
客户	类型=枚举； 列表=客户； 返回为模型定义的数据的下拉列表：客户。
复杂性类型	类型=枚举； 列表=复杂性类型； 返回为模型定义的数据的下拉列表：复杂性类型。 虽然复杂性类型可以作为项目参考数据导出和导入，但它们无法更新，因此在所有项目中都是有效的标准。
约束类型	类型=枚举； 列表=约束类型； 返回的为模型定义的数据的下拉列表：约束类型。
努力类型	类型=枚举； 列表=努力类型； 返回为模型定义的数据的下拉列表：工作量类型。
维护类型	类型=枚举； 列表=维护类型； 返回为模型定义的数据的下拉列表：维护类型。
对象类型	类型=枚举； 列表=对象类型； 返回为模型定义的数据的下拉列表：物件类型。
阶段	类型=枚举； 列表=阶段；

	返回为模型定义的数据的下拉列表：阶段。
问题类型	类型=枚举； 列表=问题类型； 返回为模型定义的数据的下拉列表：问题类型。
角色类型	类型=枚举； 列表=角色类型； 返回为模型定义的数据的下拉列表：角色类型。
需求类型	类型=枚举； 列表=需求类型； 返回为模型定义的数据的下拉列表：需求类型。
资源	类型=枚举； 列表=资源； 返回为模型定义的数据的下拉列表：资源。
风险类型	类型=枚举； 列表=风险类型； 返回为模型定义的数据的下拉列表：风险类型。
RTFT模板	类型=枚举； 列表=RTF模板； 返回为模型定义的数据的下拉列表：文档报告模板。
场景类型	类型=枚举； 列表=场景类型； 返回为模型定义的数据的下拉列表：场景类型。
测试类型	类型=枚举； 列表=测试类型； 返回为模型定义的数据的下拉列表：测试类型。