

5 关系

除了第4章中概述的通用元素外，ArchiMate 语言还定义了一组核心的通用关系，每个关系都可以连接一组预定义的源和目标概念（在大多数情况下是元素，但在少数情况下也是其他关系）。许多这些关系是“超载”的；即，它们的确切含义根据它们连接的源和目标概念而有所不同。

关系分类如下（见图 21）：

- 结构关系，模拟相同或不同类型概念的静态构造或组合
- 依赖关系，它模拟元素如何用于支持其他元素
- 动态关系，用于对元素之间的行为依赖性建模
- 不属于上述类别之一的其他关系

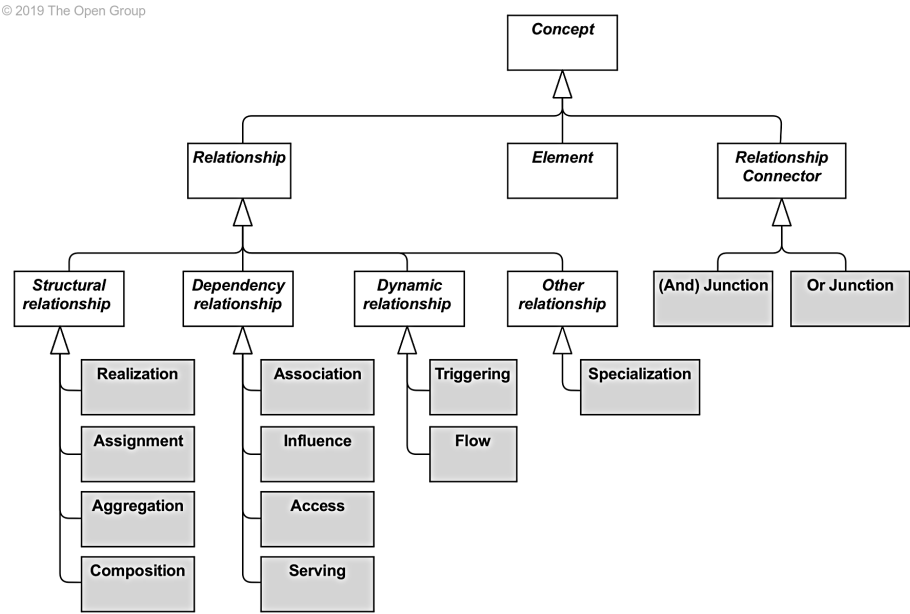


图21：关系概览

每个关系都有一个“从”和一个“到”概念（元素、关系或关系连接器）作为端点。以下限制适用：

- 两个关系之间不允许有任何关系
- 与关系连接器连接的所有关系必须属于同一类型
- 连接两个元素并依次通过关系连接器连接的相同类型的关系链仅在这两个元素之间的相同类型的直接关系有效时才有效
- 将一个元素与第二个关系连接起来的关系只能是聚合、组合或关联；聚合或组合仅从复合元素到第二个关系有效

明确命名或标记任何否则会模棱两可或被误解的关系是一种很好的做法。

为了可读性，本文档中的元模型图并未显示语言中所有可能的关系。第5.7节描述了一组推导规则，用于推导模型中元素之间的间接关系。同一类型的两个元素之间总是允许聚合、组合和专业化关系，并且任何两个元素之间以及任何元素和关系之间总是允许关联。允许关系的确切规范在附录B中给出。

5.1 结构关系

结构关系 表示架构中的“静态”连贯性。联合（组合、聚合、分配或实现）概念（关系的“从”端）始终是一个元素；对于分配和实现，它可以是元素或关系连接器。联合（组合、聚合、分配或实现）概念（关系的“至”方）在某些情况下也可能是另一个关系或关系连接器。

作为本节中提出的图形符号的替代方案，结构关系也可以通过将联合概念嵌套在联合元素中来表达。但是请注意，如果这些元素之间允许存在多个结构关系，这可能会导致视图不明确（尽管在模型中是明确的）。

5.1.1 组成关系

组合关系 表示一个元素由一个或多个其他概念组成。

组合关系的灵感来自 UML 类图中的组合关系。组合是表示存在依赖性的整体/部分关系：如果删除组合，则（通常）也删除其部分。当您对现实世界的元素建模时——例如，表示为业务参与者的部门和团队的组织结构——这种依赖性适用于这些元素本身。当您对范例或类别建模时——这在企业架构中很常见——这种依赖性可能被解释为适用于它们的真实世界实例。例如，可以将一种特定的服务器建模为由设备和系统软件组成的节点；

相同元素类型的两个实例之间始终允许存在组合关系。

除此之外，元模型还明确定义了可以通过组合关系连接的其他源和目标元素。

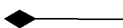
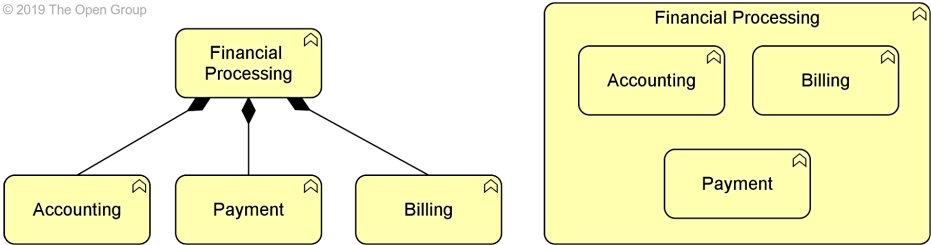


图22：组成符号

组合关系的解释是源元素的*整体或部分*由目标元素的*整体*组成。另见第5.1.5节。

例子

示例 2显示了“财务处理”业务功能由三个子功能组成的两种表达方式。



示例2：组合

5.1.2 聚合关系

聚合关系 表示一个元素结合了一个或多个其他概念。

聚合关系的灵感来自 UML 类图中的聚合关系。与组合不同，聚合并不意味着聚合和聚合概念之间存在依赖关系。

同一元素类型的两个实例之间始终允许存在聚合关系。

除此之外，元模型还明确定义了可以通过聚合关系连接的其他源和目标元素。

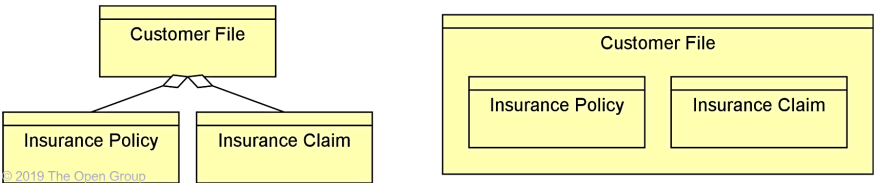


图23：聚合符号

聚合关系的解释是源元素的*全部或部分*聚合了目标概念的*整体*。另见第5.1.5节。

例子

示例 3显示了两种表示“客户文件”聚合“保险单”和“保险索赔”的方式。



示例3：聚合

5.1.3 赋值关系

分配关系 表示责任的分配、行为的表现、存储或执行。

分配关系将活动结构元素与它们执行的行为单元、业务参与者与由它们履行的业务角色以及节点与技术对象联系起来。例如，它可以将内部活动结构元素与内部行为元素相关联，将接口与服务相关联，或者将节点、设备和系统软件与工件相关联。附录 B中列出了完整的允许关系集。

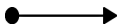


图24：分配符号

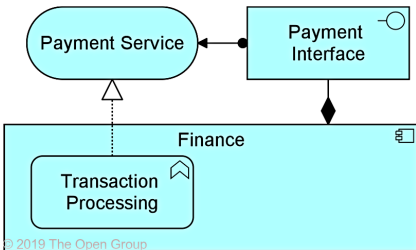
在3.4节描述的ArchiMate框架中，总是从主动结构指向行为，从行为指向被动结构，从主动结构指向被动结构。ArchiMate 2.1规范及之前版本中的非定向符号（在关系的两端显示黑球）仍然被允许但已弃用。

与所有结构关系一样，分配关系也可以通过嵌套模型元素来表示。上面说的方向也是嵌套的方向；例如，执行该角色的业务参与者中的业务角色，执行该功能的应用程序组件中的应用程序功能，或存储它的节点内的工件。

赋值关系的解释是将整个或部分源元素赋值给整个目标元素（另见第5.1.5节）。这意味着，例如，如果将两个活动结构元素分配给同一行为元素，则它们中的任何一个都可以执行完整的行为。如果需要两个活动结构元素来执行行为，则可以使用分组元素或连接（参见第 5.5 节），如果这些元素的组合具有更实质性和独立性，则协作将是正确的方法表达这个。

例子

例4包括两种表达赋值关系的方式。“Finance”应用组件被分配给“Transaction Processing”应用功能，“Payment Interface”被分配给“Payment Service”。



示例4：赋值

5.1.4 实现关系

实现关系 表示一个实体在更抽象的实体的创建、成就、维持或操作中起着关键作用。

实现关系表示更抽象的实体（“什么”或“逻辑”）是通过更具体的实体（“如何”或“物理”）来实现的。实现关系用于模拟运行时实现；例如，一个业务流程实现一个业务服务，一个数据对象实现一个业务对象，一个工件实现一个应用程序组件，或者一个核心元素实现一个激励元素。

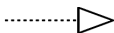
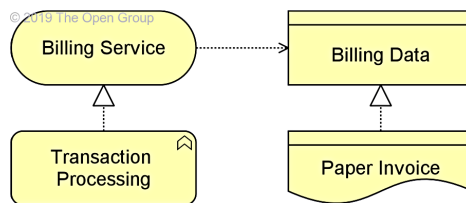


图25：实现符号

实现关系的解释是源元素的全部或部分实现了 目标元素的全部（另见第5.1.5节）。这意味着，例如，如果两个内部行为元素与同一服务有实现关系，则它们中的任何一个都可以实现完整的服务。如果两个内部行为元素都需要实现，则可以使用分组元素或and连接（参见第5.5.1节）。对于对实现激励元素的积极、中性或消极贡献的较弱类型，应使用影响关系（参见第5.2.3节）。

例子

示例 5说明了使用实现关系的两种方法。一个“交易处理”业务功能实现了一个“计费服务”；“账单数据”业务对象通过表示“纸质发票”实现。



示例5：实现

5.1.5 结构关系的语义

结构关系 描述源端的元素包含、分组、执行或实现关系的目标端的概念。结构关系可以传递地应用于源元素的（可能未建模的）部分。以下是这些语义如何工作的一些示例：

- 部分的组成和聚合关系也适用于整体

例如，如果 A 的一部分聚合了 B，则 A 本身也被认为聚合了 B。反之，如果 A 聚合了 B，则可以解释为 A 的某些部分聚合了 B。

- 行为元素的分配关系也适用于活动结构元素

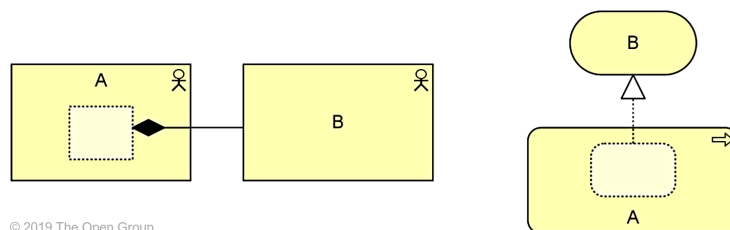
例如，如果将业务角色 A 分配给业务流程 B，则 A 的某些部分可能执行 B。反之，如果将 A 的一部分分配给 B，则 A 本身也被认为分配给了 B。

- 与外部行为元素的实现关系也适用于内部行为元素

例如，如果服务 B 由进程 A 实现，则 B 可能由 A 的某个部分实现。反之，如果 A 的一部分实现了 B，则 A 本身也被认为实现了 B。

例子

在示例 6 的左侧，整个业务参与者 B（可能是一个部门）通过 A 中的一些未建模元素在业务参与者 A（可能是一个部门）中组成。在右侧的示例中，业务流程 A 完全通过 A 中的一些未建模元素实现业务服务 B。



示例6：结构关系的语义

5.2 依赖关系

依赖关系 描述元素如何支持或被其他元素使用。区分四种类型的依赖关系：

- 服务关系 表示控制依赖，用实线表示
- 访问关系 表示数据依赖关系，用虚线表示
- 影响关系 表示影响依赖性，用虚线表示
- 关联关系 表示不被任何其他关系涵盖的依赖关系

请注意，尽管这些关系的表示法类似于 UML 中依赖关系的表示法，但这些关系在 ArchiMate 表示法中具有不同的含义并且（通常）指向相反的方向。这样做的一个优点是它产生了具有方向性的模型，其中表示支持、影响、服务或实现依赖关系的大多数箭头都“向上”指向客户/用户/业务，正如您在分层视点示例中看到的那样在 C.1.5 节中。这个方向的另一个原因，特别是服务关系，是它从“调用者”或“发起者”中抽象出来，因为服务可以主动或被动地交付。交付方向始终相同，但交互的起点可以在任一端。UML 的依赖性通常用来表示后者，表明调用者依赖于被调用的某个操作。然而，为了对这种类型的主动性进行建模，ArchiMate 语言提供了触发关系（第 5.3.1 节），它可以解释为动态（即时间）依赖性。类似地，流关系用于模拟某物（通常是信息）如何从一个元素传输到另一个元素，这也是一种动态的依赖关系。

5.2.1 服务关系

服务关系 表示一个元素向另一个元素提供其功能。

服务关系描述了行为或活动结构元素提供的服务或接口如何为其环境中的实体服务。这种关系适用于行为方面和活动结构方面。

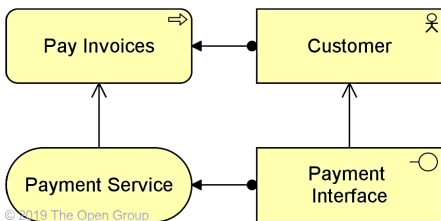
与本标准的早期版本相比，这种关系的名称已从“used by”更改为“serving”，以更好地反映其方向并使用主动动词：a service serves a user。关系的意义没有改变。仍然允许使用“由……使用”，但已弃用，并将在标准的未来版本中删除。



图26：服务符号

例子

示例 7说明了服务关系。“支付接口”服务于“客户”，而“支付服务”服务于该客户的“支付发票”业务流程。



示例7：服务

5.2.2 访问关系

访问关系 表示行为和主动结构元素观察或作用于被动结构元素的能力。

访问关系表示流程、功能、交互、服务或事件使用被动结构元素“做某事”；例如，创建一个新对象，从对象中读取数据，写入或修改对象数据，或删除对象。关系也可以用来表示对象只是与行为相关联；例如，它对事件附带的信息或作为服务的一部分提供的信息进行建模。箭头（如果存在）表示被动结构元素的创建、更改或使用。访问关系不应与使用类似符号的 UML 依赖关系相混淆。

请注意，在元模型级别，关系的方向始终是从主动结构元素或行为元素到被动结构元素，尽管符号可能指向另一个方向以表示“读取”访问，并且在两个方向上来表示读写访问。对派生关系使用访问时必须小心，因为关系上的箭头与其方向无关。

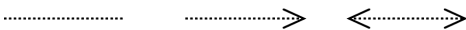
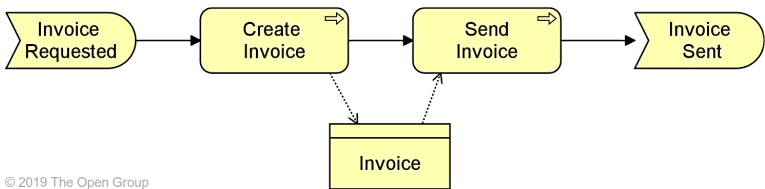


图27：访问符号

或者，可以通过将被动结构元素嵌套在访问它的行为或主动结构元素中来表达访问关系；例如，在应用程序组件中嵌套数据对象。

例子

示例 8说明了访问关系。“创建发票”子流程写入/创建“发票”业务对象；“发送发票”子流程读取该对象。



示例8：访问

5.2.3 影响关系

影响关系 表示一个元素影响某个激励元素的实现或实现。

影响关系用于描述某些架构元素影响激励元素（例如目标或原则）的实现或实施。通常，激励因素在一定程度上得到实现。例如，始终如一地满足“随时随地为客户服务”的原则，将有助于实现“提高市场份额”的目标。换句话说，原则有助于实现目标。反过来，要实施“随时随地为客户提供服务”的原则，对某些面向客户的应用程序组件施加“24x7 Web 可用性”的要求可能会很有用。这可以建模为对该原则有影响的需求，以及反过来影响该需求的应用程序组件。使用影响关系对这些依赖关系进行一致建模会产生可追溯的激励路径，该路径解释了为什么在此示例中某个应用程序组件有助于实现企业“增加市场份额”的目标。

除了这种从核心元素向上到需求和目标的贡献的“垂直”使用之外，该关系还可以用于模拟激励元素之间的“水平”贡献。这种情况下的影响关系描述了一些动机因素可能影响（实现或实施）另一个动机因素。通常，在一定程度上实现了激励因素。其他因素的影响可能会影响这个程度，这取决于其他因素本身得到满足的程度。例如，提高客户满意度的目标的实现程度可以用参与市场访谈的满意客户的百分比来表示。例如，该百分比可能会受到提高公司声誉的目标的影响；IE，更高层次的改进会导致更高的客户满意度。另一方面，裁员的目标可能对公司声誉产生负面影响；即，更多的裁员可能导致公司声誉的增加减少（甚至减少）。因此（间接地），提高客户满意度的目标也可能受到负面影响。

实现关系应该用来表示对目标对象的存在或实现至关重要的关系，而影响关系应该用来表示对目标对象的存在或实现不重要的关系。例如，代表建筑团队的业务参与者可能会实现建造建筑物的目标，而在已经足够的团队中增加额外的熟练建筑工人的要求可能会影响建造建筑物的目标，但也会实现开放的额外目标特定日期之前的建筑物。此外，影响关系可用于建模：

- 一个要素对某些激励要素的实现或实施有积极贡献的事实，或
- 一个因素负面影响——即阻止或抵消——这种成就的事实

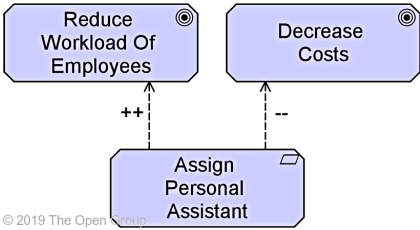
属性可用于指示影响的符号和/或强度。可能的属性值的选择留给建模者；例如，{++, +, 0, -, --} 或 [0..10]。默认情况下，影响关系对具有未指定符号和强度的贡献进行建模。

-- +/- -->

图28：影响符号

例子

示例 9 说明了使用影响关系来模拟同一需求“分配个人助理”的不同效果。这对“减少员工的工作量”有很强的积极影响，但对“降低成本”有很强的消极影响。



示例9：影响

5.2.4 关联关系

关联关系 表示未指定的关系，或者未由另一个 ArchiMate 关系表示的关系。

关联关系始终允许存在于两个元素之间，或者关系与元素之间。

关联关系可以在绘制第一个高级模型时使用，其中关系最初以通用方式表示，然后细化以显示更具体的关系类型。在元模型图片中，明确显示了关联关系的一些特定用途。默认情况下，关联是无向的，但可以有向的。另见第5.2.5节。

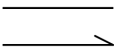
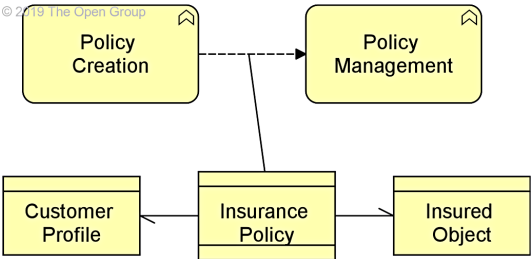


图29：关联符号

例子

示例 10 说明了合同和该合同所引用的两个业务对象之间的两个定向关联关系。它还显示了流关系与此契约之间的关联，以指示在两个功能之间传递的信息类型。



示例10：关联

5.2.5 依赖关系的语义

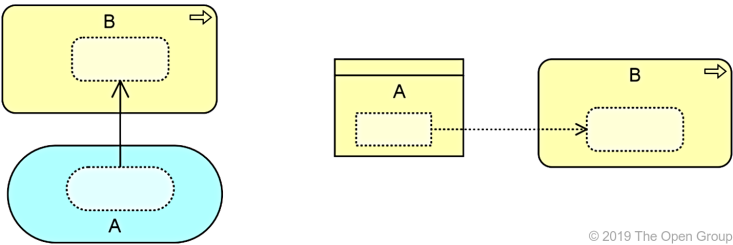
依赖关系 描述了目标元素的一部分依赖于源元素的一部分。尽管两个元素之间存在依赖关系，但这并不一定意味着这适用于由任何结构关系定义的元素的所有部分。

这种语义允许您在高层次上对依赖关系建模（删除细节），而不是在更详细的层次上暗示特定的依赖关系。这意味着，例如：

- 在服务关系中，内部行为元素的某些部分由外部行为元素的某些部分服务；例如，如果业务服务 A 服务于业务流程 B，则 A 的某些未建模的子服务可能服务于 B 的未建模的子流程
- 在访问关系中，行为元素的某些部分访问被动结构元素的某些部分；例如，如果应用程序功能 A 访问数据对象 B，则 A 的某些未建模的子功能可能会访问 B 的未建模部分
- 在影响关系中，核心要素的某些部分影响激励要素的某些部分；例如，如果应用程序组件 A 影响需求 B，则 A 的某些未建模部分可能会影响 B 的某些未建模部分
- 在关联关系中，一个元素的某些部分与另一个元素的某些部分相关；如果 它是有方向的，它只能用于那个方向的推导（见第 5.7 节）

例子

在示例 11 的左侧，业务流程 B 的一部分由应用程序服务 A 的一部分提供服务。在右侧示例中，业务流程 B 的一部分访问（读取）业务对象 A 的一部分。



示例11：依赖关系的语义

5.3 动态关系

动态关系 描述了架构中元素之间的时间依赖性。区分两种类型的动态关系：触发和流。

5.3.1 触发关系

触发关系 表示元素之间的时间或因果关系。

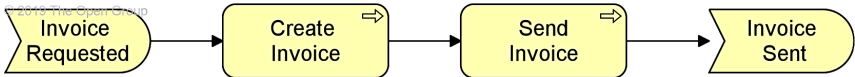
触发关系用于对过程中行为元素的时间或因果优先级建模。触发关系的解释是源元素的某些部分应该在目标元素开始之前完成（另见第5.3.3节）。请注意，这并不一定代表一个行为元素主动启动另一个行为元素；交通灯变绿也会触发汽车通过十字路口。



图30：触发符号

例子

示例 12 说明了触发关系用于对（子）过程和/或事件之间的因果依赖性建模。



示例12：触发

5.3.2 流量关系

流动关系 表示从一个元素到另一个元素的转移。

流关系用于对行为元素之间的信息流、商品流或金钱流进行建模。流量关系并不意味着因果关系。

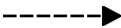
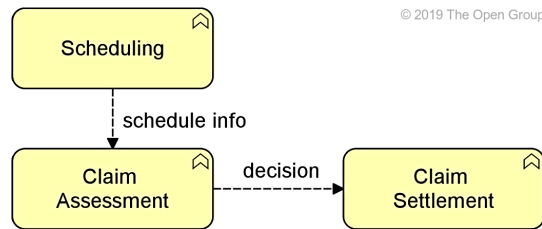


图31：流程符号

例子

示例 13显示了一个“Claim Assessment”业务功能，它将关于索赔的决策转发给“Claim Settlement”业务功能。为了确定索赔评估的顺序，“索赔评估”使用从“调度”业务功能接收的调度信息。



示例13：流程

5.3.3 动态关系的语义

触发和流关系的语义不同。触发关系遵循与结构关系相同的语义（第5.1.5节）。A到B的触发关系表示B中的所有步骤都在A的一部分之前。例如，当A和B是业务流程时，表示业务流程B中的所有步骤都在A的一部分发生之后执行，但是A中的步骤可以在B中的某些或所有步骤发生之后发生。希望这样做的建模小组可以对 ArchiMate 模型施加更强的触发解释（B中的所有内容都在A中的所有内容之前）。

流关系遵循与依赖关系相同的语义（参见第5.2.5节）。从A到B的流关系表示A的整体或部分将某些东西（例如信息）传递给B的整体或部分。

5.4 其他关系

5.4.1 专业化关系

特化关系 表示一个元素是另一个元素的特定种类。

专业化关系受到 UML 类图中泛化关系的启发，但适用于专业化更广泛的概念。

同一元素类型的两个实例之间始终允许存在专门化关系。

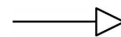
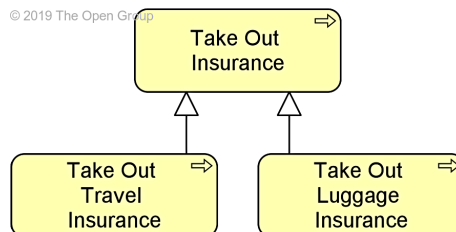


图32：专业化符号

或者，可以通过将专用元素嵌套在通用元素中来表达专用化关系。

例子

示例 14说明了进程的专业化关系的使用。在这种情况下，“Take Out Travel Insurance”和“Take Out Luggage Insurance”业务流程是更通用的“Take Out Insurance”业务流程的专业化。



示例14：专业化

5.4.2 其他关系的语义

专门化关系的语义是整个通用元素由专门化元素专门化。

5.5 关系连接器

5.5.1 结

联结不是与本章中描述的其他关系相同意义上的实际关系，而是关系连接器。

联结 用于连接相同类型的关系。

在许多情况下使用联结来连接相同类型的关系。如果还允许这些概念之间存在该类型的直接关系，则仅允许在两个概念之间使用具有连接此类关系的结点的路径。简而言之，您不能使用联结来创建概念之间的关系，否则是不允许的。

一个junction可能有多个传入关系和一个传出关系，一个传入关系和多个传出关系，或者多个传入和传出关系（后者可以被认为两个连续的junctions的简写）。

可以和junction结合使用的关系都是动态依赖关系，还有赋值和实现。联结用于明确表示所有元素一起必须参与关系（和联结）或至少一个元素参与关系（或联结）。or联结可用于表示包含和排除 or 条件，建模者可以通过命名联结以反映其类型来指示这些条件。允许省略指向结点的关系的箭头。

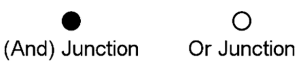
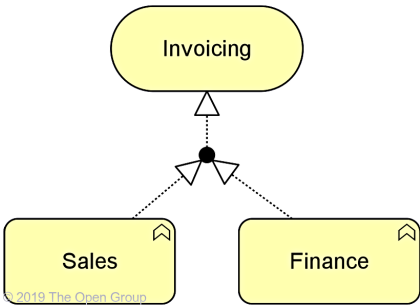


图33：结符号

用于触发关系的连接类似于 BPMN 中的网关以及 UML 活动图中的分叉和连接（没有关联的语义）。它们可用于模拟高级流程。可以将标签添加到结点的传出触发关系，以指示适用于该关系的选择、条件或保护。这样的标签只是一种非正式的指示。没有为这些关系定义正式的操作语义，因为 BPMN 和 UML 等实现级语言的执行语义不同，而 ArchiMate 语言不想过度限制到这些语言的映射。

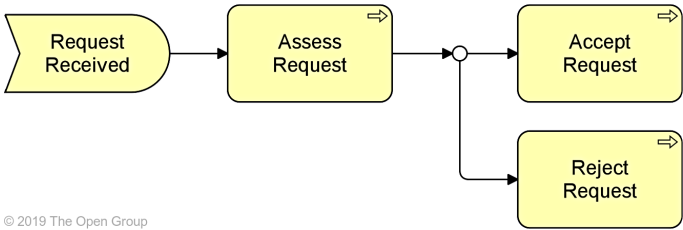
例子

在 示例15中，模型中的and连接点用于表示“销售”和“财务”业务功能共同实现“进销存”业务服务。



示例15：（和）连接

在 示例 16中，或 连接点用于表示选择：业务流程“评估请求”触发“接受请求”或“拒绝请求”。（两个单独的触发关系的通常解释，一个从“评估请求”到“接受请求”，一个从“评估请求”到“拒绝请求”，是“评估请求”触发了其他两个业务流程。）




示例16：或连接

5.6 关系概要

表 3概述了 ArchiMate 关系及其定义。

表3：关系

结构关系	符号	角色名称
------	----	------

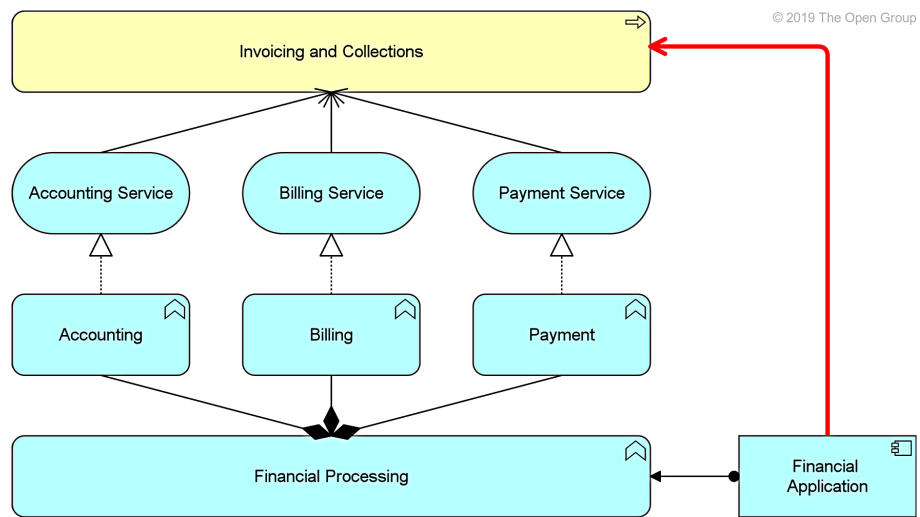
作品	表示一个元素由一个或多个其他概念组成。		← 由 → 组成
聚合	表示一个元素结合了一个或多个其他概念。		← 聚集 → 聚集于
任务	表示责任的分配、行为的表现、存储或执行。		← 分配给 → 已分配
实现	表示一个实体在更抽象的实体的创建、成就、维持或操作中起着关键作用。		← 实现 → 实现
依赖关系		符号	角色名称
服务	表示一个元素向另一个元素提供其功能。		← 服务 → 服务于
使用权	表示行为和主动结构元素观察或作用于被动结构元素的能力。		← 访问 → 访问者
影响	表示一个元素影响某些动机元素的实现或实现。		← 影响 → 影响
协会	表示未指定的关系，或未由另一个 ArchiMate 关系表示的关系。		关联 ← 关联到 → 关联自
动态关系		符号	角色名称
触发	表示元素之间的时间或因果关系。		← 触发 → 触发
流动	表示从一个元素转移到另一个元素。		← 流向 → 流自
其他关系		符号	角色名称
专业化	表示一个元素是另一个元素的特定种类。		← 专业化 → 专业化
关系连接器		符号	角色名称
交界处	用于连接相同类型的关系。	 (And) Junction  Or Junction	

5.7 关系推导

在 ArchiMate 语言中，您可以根据建模关系导出模型中元素之间的间接关系。这使得从与在特定模型或架构视图中显示不相关的中间元素中抽象出来并支持影响分析成为可能。进行此类推导的精确规则在附录B中指定。

例子

在 示例 17中，假设目标是从模型中的应用程序功能、子功能和服务中抽象出来。在这种情况下，可以从“财务应用”到“发票和收款”业务流程（从链分配-组合-实现-服务）衍生出间接服务关系（右侧粗红色箭头）。



示例17：从关系链中推导

关系的推导旨在作为创建详细模型摘要的一种方式。这是一种在模型中删除（抽象）细节的方法，同时仍然可以做出有效的“陈述”。因此，推导总是意味着从更多细节到更少细节。与其他建模语言相比，这种机制是 ArchiMate 语言的独特属性之一。

该语言允许建模者直接创建关系，这些关系必须是有效的派生关系，而派生的成分在模型中不可用。这些关系（例如，应用程序组件和应用程序服务之间的实现关系）假设所需的成分（例如，存在派生关系所需的应用程序功能；然而，这些缺失的元素不需要明确建模，派生关系可以像没有派生一样使用。因此，建模者可以完全自由地选择所需的详细程度。

因为推导的本质是做简化或归纳，不能用来推断更多的细节。例如，可以对从应用程序组件到应用程序服务的实现关系进行建模，但无法从中得出有关此派生的确切来源的结论（例如，哪些功能实现了哪些服务）。

这是建模者在设计过程中应该添加的信息：更高层次、更抽象的模型可以通过详细说明派生关系来细化（在前面的示例中，通过添加实现应用程序服务的应用程序功能，以及分配应用程序组件）。

重要的是要注意所有这些派生关系在 ArchiMate 语言中也有效。它们没有显示在标准中包含的元模型图中，因为这会降低它们的易读性。但是，附录B中的表格显示了语言中两个元素之间所有允许的关系。