目录

[**1.1   基本元素** 2](#_Toc493163166)

[**1.2   核心元素** 2](#_Toc493163167)

[**1.3   扩展元素** 2](#_Toc493163168)

[二、BPMN基本元素 3](#_Toc493163169)

[2.1   基本元素 3](#_Toc493163170)

[三、BPMN核心元素 11](#_Toc493163171)

[3.1   核心元素 11](#_Toc493163172)

[四、BPMN扩展元素 21](#_Toc493163173)

[4.1   扩展元素 21](#_Toc493163174)

[一、BPMN的元素](http://blog.csdn.net/ronghao100/article/details/6703403)

我们使用BPMN元素进行建模，BPMN的元素分为五个类别：

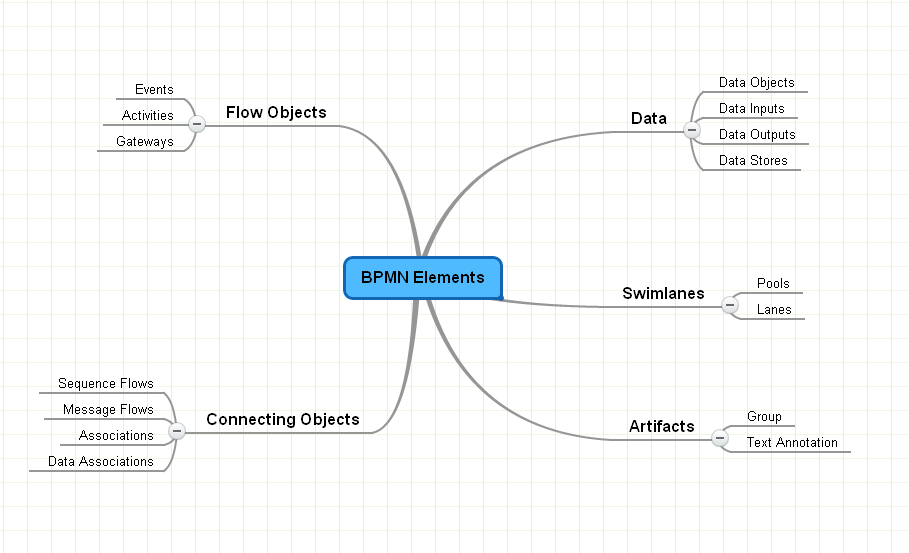


图10‑9BPMN的元素分类

1. 流对象（Flow Objects），流对象是定义业务流程的主要图形元素。它进一步细分为三个类别，分别是事件（Events）、活动（Activities）和网关（Gateways）；
2. 数据（Data），它分为四个类别：数据对象（Data Object）、数据输入（Data Inputs）、数据输出（Data Outputs）和数据存储（Data Stores）；
3. 连接对象（Connection Ojbects），用来把各个流对象或流对象与其他信息连接起来，它分为四种类别：顺序流（Sequence Flows）、消息流（Message Flows）、关联（Associations）和数据关联（Data Associations）；
4. 泳道（Swimlanes），用来区分不同部门或者不同参与者的功能和职责。Swimlanes包含两种类别：池（Pool）和道（Lane）；
5. 人工交付物（Artifacts），它用以给流程附加一些额外的信息，它分为两种类别：组（Group）和附注（Text Annotation）。

为了方便学习，我们将BPMN元素分为了三个层次，分别是基本元素、核心元素和扩展元素，通过基本元素的学习，我们能够开始进行简单的建模；通过核心元素的学习，能够满足我们平常大部分的建模需求；通过学习扩展元素，我们能够看到BPMN所能处理的复杂情况。在下面的各节里，我们将分别对这三个层次的元素进行讨论。

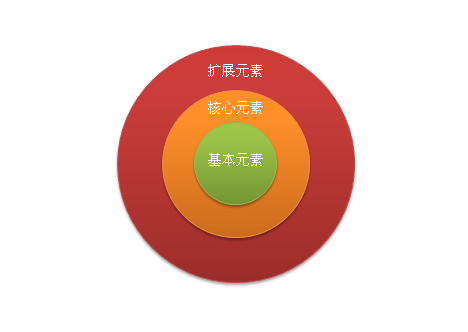


图10‑10BPMN元素的三个层次

**1.1   基本元素**

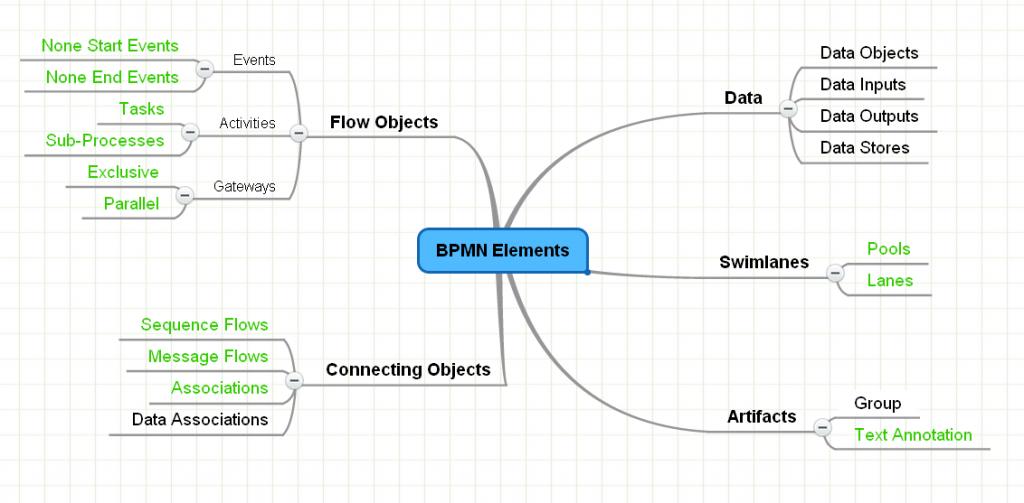


图10‑11BPMN的基本元素

**1.2   核心元素**

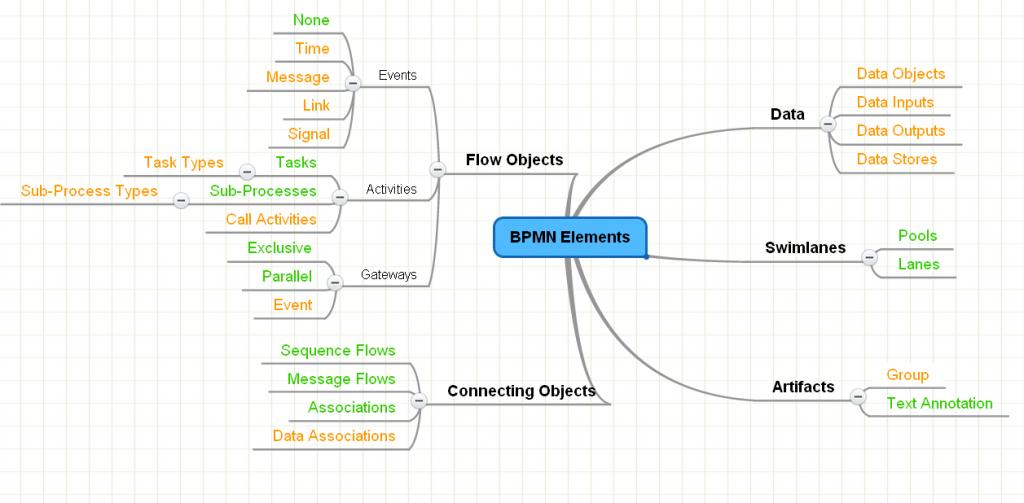


图10‑12BPMN的核心元素

**1.3   扩展元素**

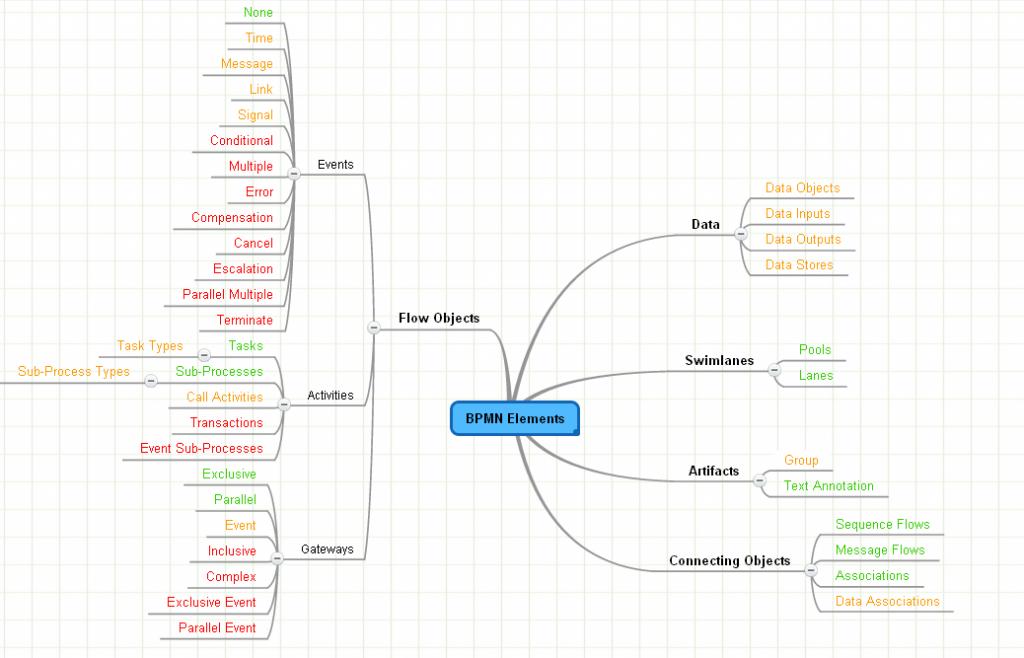


图10‑13BPMN的扩展元素

### [二、BPMN基本元素](http://blog.csdn.net/ronghao100/article/details/6707276)

### 2.1   基本元素

什么是BPMN基本元素？我们为什么要从BPMN元素中界定出一个基本元素的子集？BPMN基本元素是我们认为可以进行实际建模的BPMN元素的最小子集，使用BPMN基本元素我们可以创建非常简单的流程（编制）和协作模型，这些元素非常容易学习，对于没有技术背景的业务人员也很容易理解，适合草图和非正式的沟通。

BPMN基本元素包括了这些元素，如下图所示：

* 泳道：池和道；
* 三种连接对象：序列流、消息流和关联；
* 一种活动：任务（Task），其中包括了子流程任务（Sub-ProcessTask）；
* 两种事件：普通开始事件（NoneStart Event）和普通结束事件（None End Event）；
* 两种网关：排他网关（Exclusive）和并发网关（Parallel）；
* 一种人工制造物：附注。

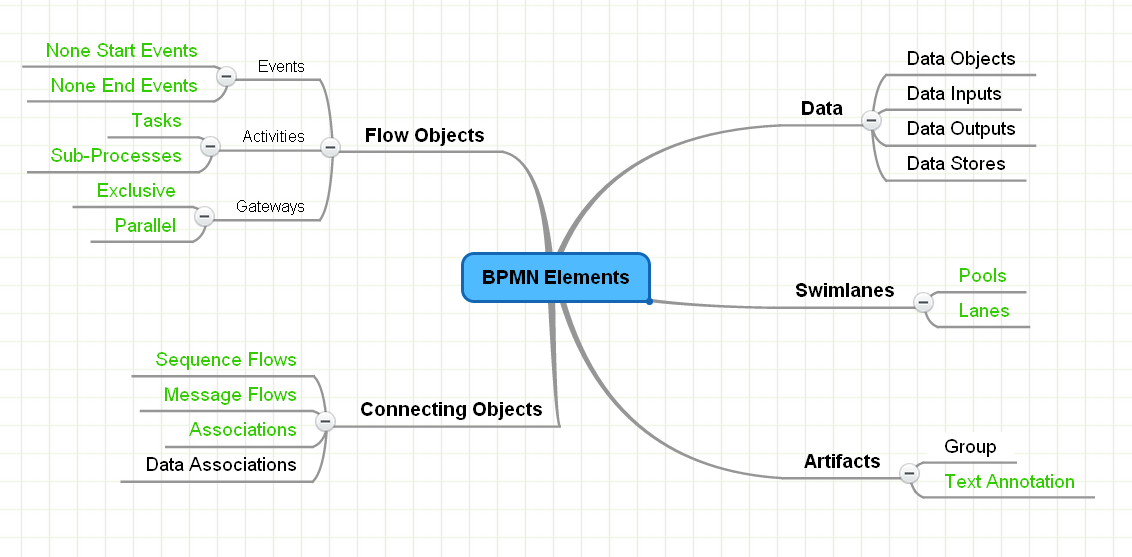


图10‑11BPMN的基本元素

         在下面对BPMN基本元素的介绍里，我们分为了两部分，第一部分我们从整体上了解BPMN元素几个大的分类：泳道、流对象、连接对象和人工制造物，其中也包括了对流程执行语义的说明；在第二部分里，我们则具体了解流对象的分类：事件、活动和网关，我们只介绍其中的基本元素。

#### 2.2   泳道、流对象、连接对象、人工制造物和流程执行语义

##### 泳道

我们使用泳道来区分不同部门或者不同参与者的功能和职责。泳道包含两种类别：池和道。我们使用池来表示流程的参与者，应用在公开流程和协作模型里（还记得我们BPMN的三种流程模型吗？如果没有印象了，它们就在10.2小节）。

池与池之间的交互只能使用消息流表示。

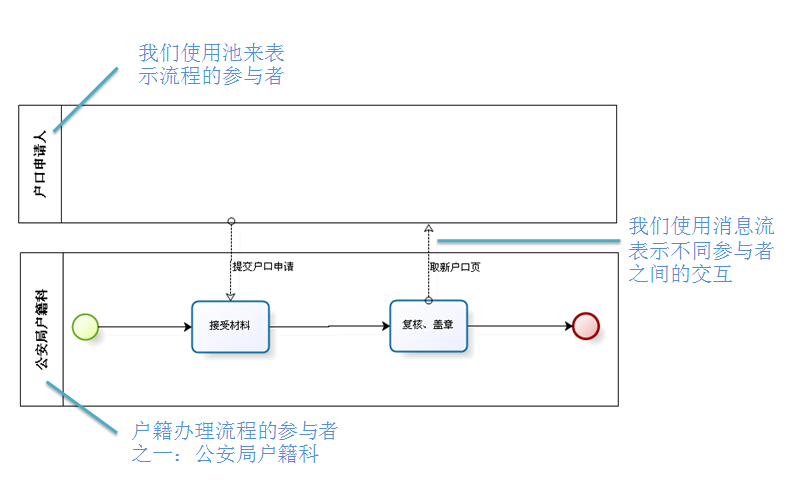


图10‑12池表示流程的参与者

         我们使用道对池内的活动进行分组，我们可以根据活动所属的部门或办理所需的职责（角色）来对活动进行分组。

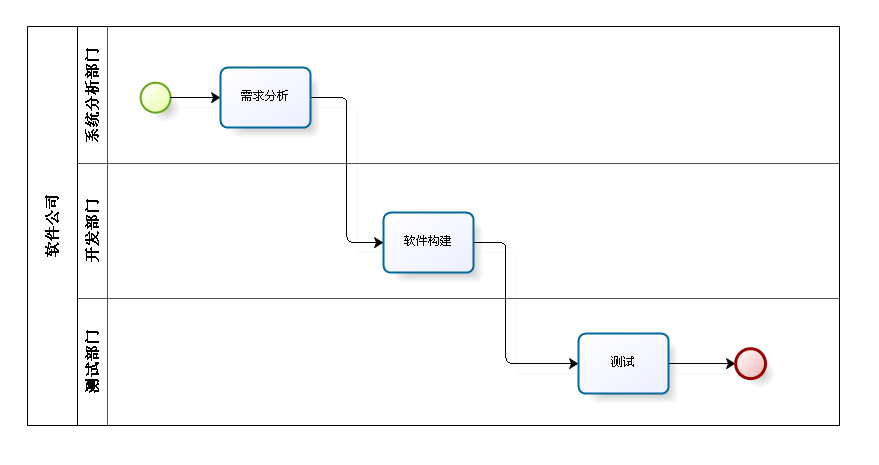


图10‑13使用道表示不同的部门

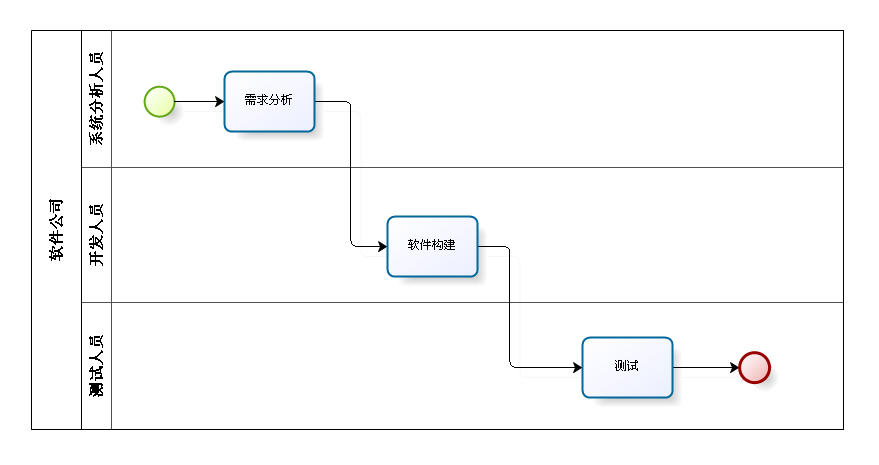


图10‑14使用道表示不同的角色

如果流程是私有流程并且活动没有分组的需要，那么通常我们会省略池。

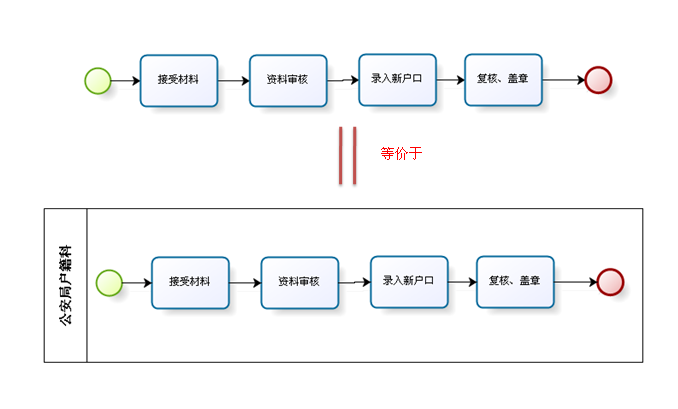


图10‑15在活动没有分组的私有流程里忽略池

##### 流对象

流对象是定义业务流程的主要图形元素。它进一步细分为三个大的类别，分别是事件、活动和网关。具体到事件、活动和网关里，流对象有很多的类别，BPMN元素的复杂性体现在流对象的众多类别上。

* 事件：发生在流程执行过程中的事情；
* 活动：在流程执行过程中执行的工作；
* 网关：控制流程的分支和聚合。

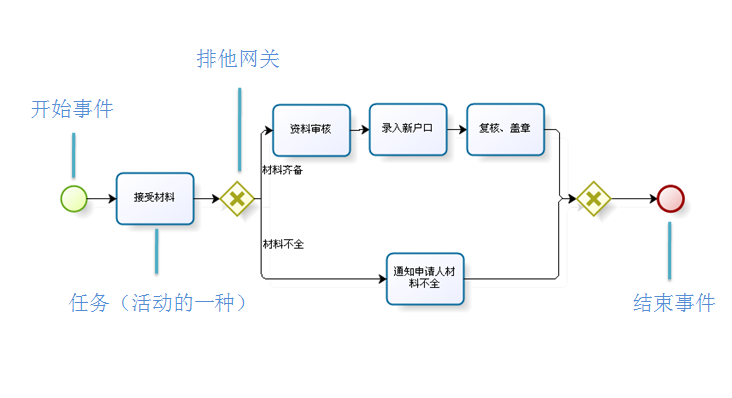


图10‑16流程中的流对象

##### 连接对象

连接对象用来把各个流对象或流对象与其他信息连接起来，它分为四种类别：顺序流、消息流、关联和数据关联。

* 顺序流：表示流对象的前后执行顺序；
* 消息流：表示公开流程或协作模型里参与者之间的消息、交互；
* 关联：为流程中的元素关联信息或数据；
* 数据关联：表示数据在流程、活动和数据对象之间的传递。

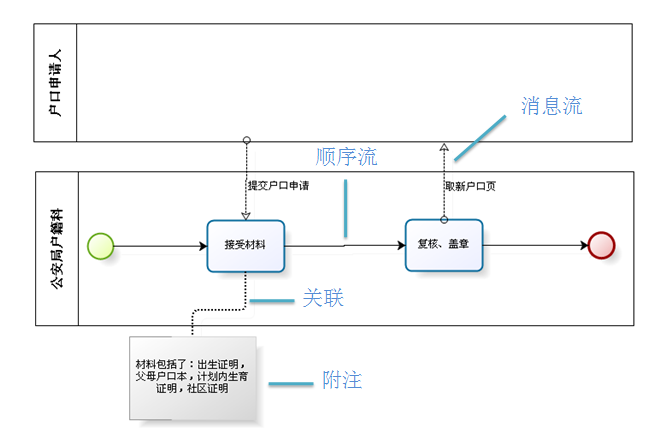


图10‑17流程中的连接对象

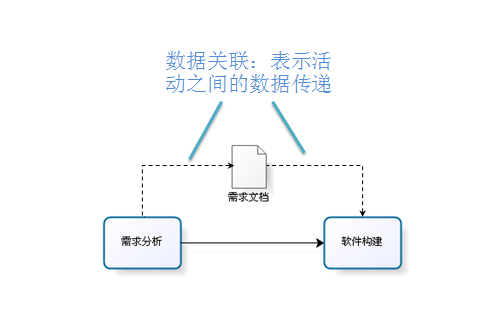


图10‑18数据关联

##### 人工制造物

人工制造物用以给流程附加一些额外的信息，不影响流程的流转，它分为两种类别：组和附注。

* 组：对元素进行分类；
* 附注：给元素附加信息，便于阅读。

##### 流程执行语义

我们可以将流程的执行看作是Token的流转，Token从一个源流对象经过顺序流流转到一个目标流对象，当一个流对象拿到一个Token时即开始执行，当执行完成后再把Token通过顺序流流转出去。类似于我们小时候玩的击鼓传花游戏，拿到花（Token）才能行动。

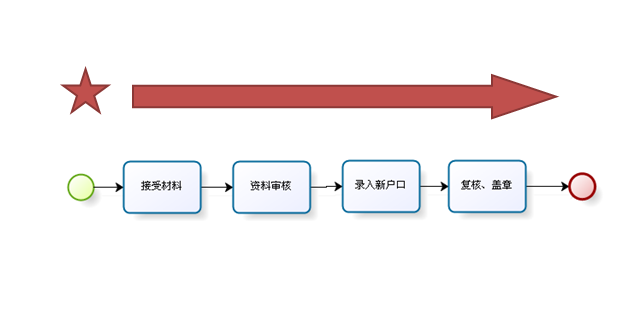


图10‑19把流程执行看成是Token的流转

每次流程接受到一个新的开始事件，就会启动一个新的案例（流程实例）开始执行。一个流程在同一时间可能有多个案例正在执行。

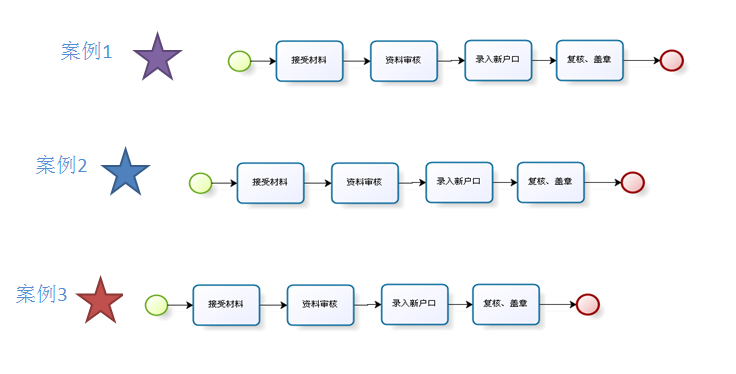


图10‑20案例

#### 2.3        事件、活动和网关

##### 事件

我们将发生在流程执行过程中的事情称为事件。事件有三种类型：

* 开始事件：触发流程的执行；
* 即时事件：在开始事件和结束事件之间发生的事件，会影响流程的流转，但不会启动或直接终止流程的执行；
* 结束事件：表明流程执行结束。

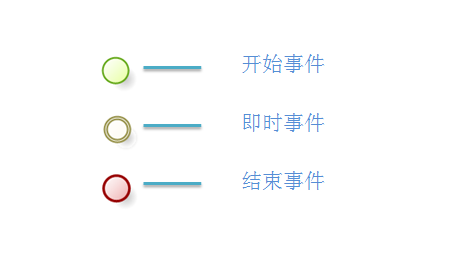


图10‑21事件的三种类型

对一个事件来说，通常会有产生该事件的原因（起因）和该事件所产生的影响（结果），例如一个消息的到来触发了一个事件并导致流程的启动，这个开始事件的起因是消息，它被称为消息开始事件，流程在结束时发送了一个消息，这个结束事件的结果是发送了消息，它被称为消息结束事件。我们使用不同的标识来区分事件的不同起因和结果。我们也使用事件不同的起因和结果来分类事件，完整的事件类型如下图所示：

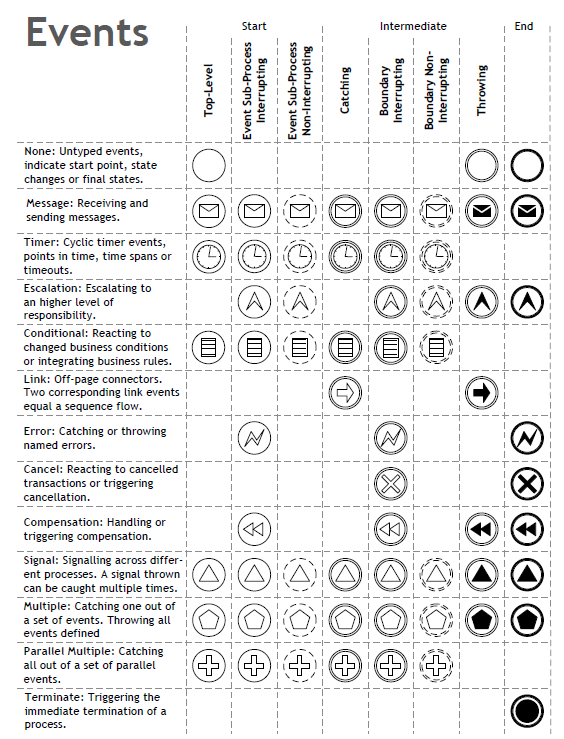


图10‑22完整的事件类型

在基本元素里我们只关注两种类型的事件：普通开始事件和普通结束事件。普通开始事件没有指定事件的起因，它只是产生Token，触发流程的执行；普通结束事件没有指定事件的结果，它消费掉Token，结束流程的执行。共有10种类型的开始事件和9种类型的结束事件，在后续的小节中我们会逐一讨论到。

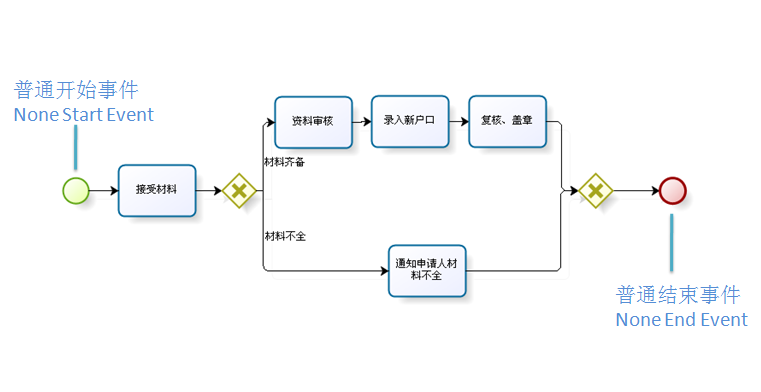


图10‑23普通开始事件和普通结束事件

作为一个其他开始事件的例子，我们接受一个邮件消息并开始在线的下订单任务，这个开始事件的起因是消息，所以称为消息开始事件。

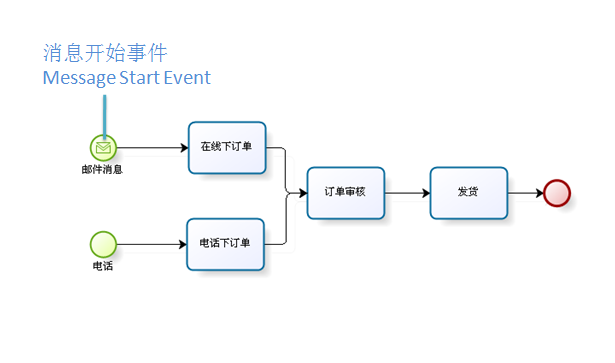


图10‑24消息开始事件

##### 活动

我们将在流程执行过程中执行的工作称为活动。活动可以是原子的，也可以是非原子的（复合的），活动分为两类：任务（原子的）和子流程（复合的）。任务是流程模型中最小工作单元，不能继续分解，子流程则能继续分解为一系列的子活动。在流程模型里，子流程可以展开也可以收起。

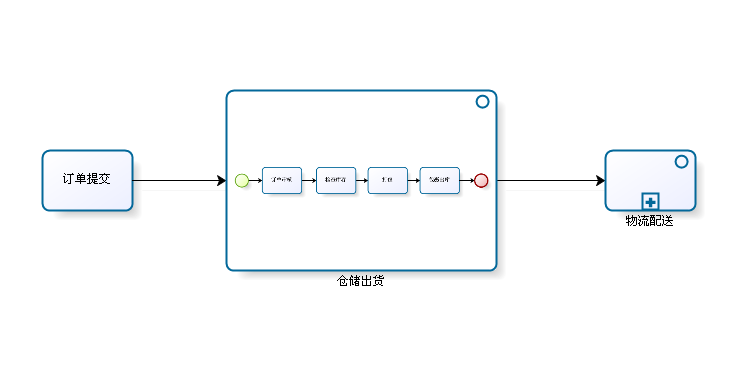


图10‑25任务和子流程

一个活动可能会有零到多个输入顺序流，也会有零到多个输出顺序流。需要记住的是，只要任何一个输入顺序流传入Token，活动都会立即执行，而执行完毕后如果有多个输出顺序流，它则会产生多个Token，每个输出顺序流传递一个（相当于并发）。

##### 网关

         网关控制流程的分支和聚合。有七种网关，各种类型网关的分裂和聚合行为由下面三种因素控制：

* 输入顺序流和输出顺序流上设定的条件；
* 输出顺序流上的事件；
* 网关自身上设定的条件。

在基本元素里我们关注两种类型的网关：排他网关和并发网关。

排他网关在分裂时，只会选择一个输出顺序流传递Token，排他网关在聚合时，任何一个输入顺序流传入Token，网关都会向后传递，不会对Token进行同步。我们使用“x”在网关内部标示出这是一个排他网关，默认情况下，我们也可以忽略标示。

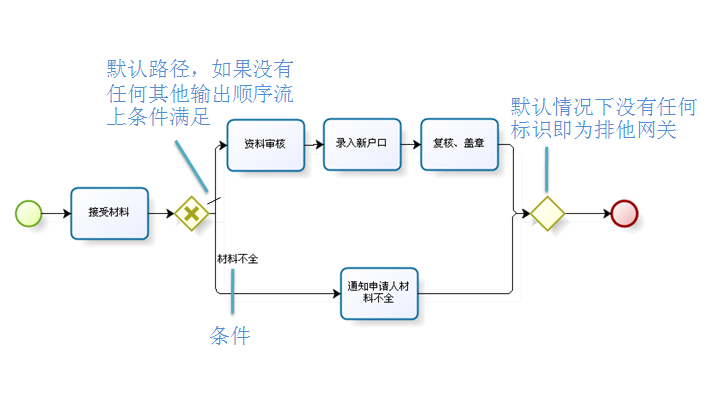


图10‑26排他网关

         并发网关在分裂时，会产生多个Token，每个输出顺序流传递一个（并发），并发网关在聚合时，会对所有输入顺序流的Token进行同步，只到所有输入顺序流的Token都到达后才会向后传递Token。我们使用“+”在网关内部标示出这是一个并发网关。

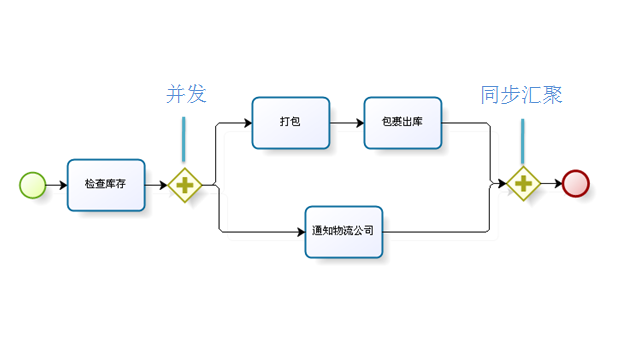


图10‑27并发网关

### [三、BPMN核心元素](http://blog.csdn.net/ronghao100/article/details/6737565)

### 3.1   核心元素

什么是BPMN核心元素？我们为什么要从BPMN元素中界定出一个核心元素的子集？BPMN核心元素是我们认为可以建模进行业务分析的BPMN元素的必要子集，使用BPMN基本元素适合草图和非正式的沟通，而当组织扩大或需要与组织外部人员进行沟通时，使用更加具体具有明确语义的建模元素变得重要，使用BPMN核心元素能够满足我们平常大部分的建模需要。

除了包括BPMN基本元素，核心元素增加了这些元素，如下图中的黄色部分所示：

* 更多的事件类型：计时（Time）、消息（Message）、链接（Link）和信号（Signal）；
* 更具体的活动类型：任务的7种类型、子流程的5种类型和调用活动（CallActivity）；
* 事件网关（Event-basedGateway）；
* 数据元素：数据对象（DataObject）、输入数据（Data Input）、输出数据（Data Output）和数据存储区（Data Store）；
* 数据关联。

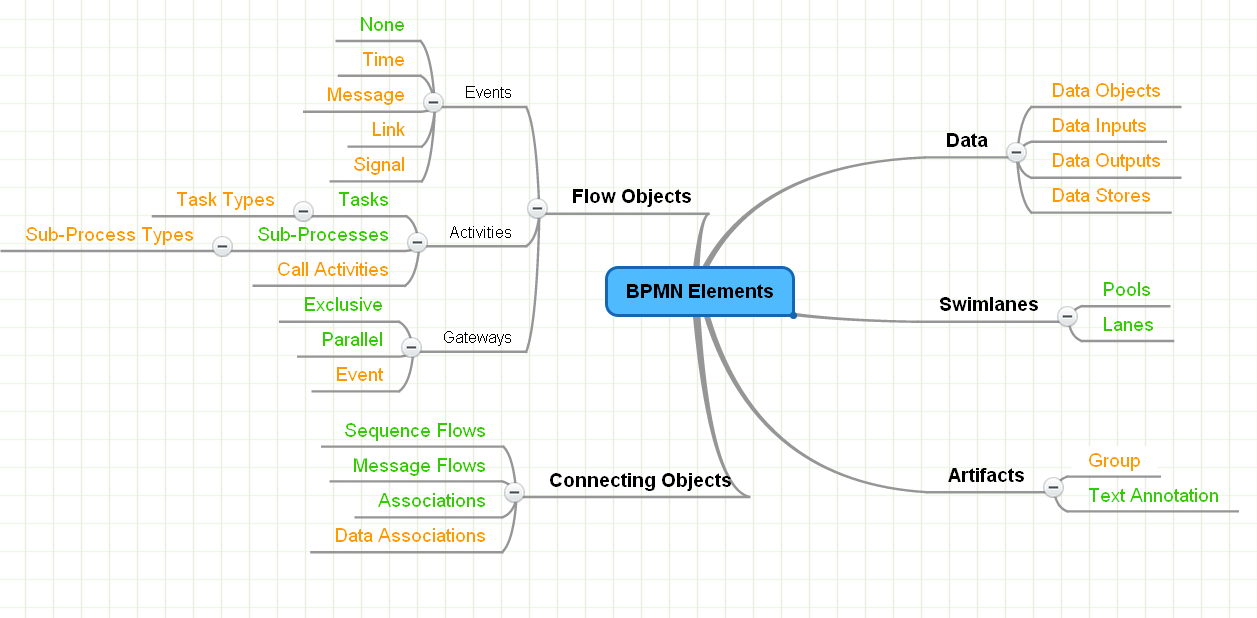


图10‑28BPMN的核心元素

#### 3.2        活动

在基本元素里，我们知道活动是在流程执行过程中执行的工作。活动分为两类：任务（原子的）和子流程（复合的）。任务是流程模型中最小工作单元，不能继续分解，子流程则能继续分解为一系列的子活动。在核心元素里，我们将讨论任务和子流程更具体的类型以及一种新的活动类型-调用活动，这样在我们建模时，模型会有更加具体的语义。

##### 任务

根据任务的执行语义，我们将任务分为8种类型：

* 普通任务（NoneTask）：默认的任务类型，未指定任务的执行语义；
* 发送任务（SendTask）：给外部参与者发送消息，消息发送完毕则任务执行完毕；
* 接受任务（ReceiveTask）：等待并接受从外部参与者发送过来的消息，消息接受完毕则任务执行完毕；
* 人工任务（UserTask）：在应用程序协助下完成的人工任务，例如一个从任务列表中拾取并执行的任务，比如办理户口中向计算机输入户口信息的任务；
* 手工任务（ManualTask）：没有任何流程引擎和应用程序协助的纯手工任务，例如对我提交户口资料的审核任务；
* 业务规则任务（BusinessRule Task）：该任务调用一个业务规则引擎进行数据的计算并获得计算结果；
* 服务任务（ServiceTask）：调用某种类型的服务，这种服务可以是WEB服务，也可以是自动化的应用程序；
* 脚本任务（ScriptTask）：定义一段执行脚本，由流程引擎执行。

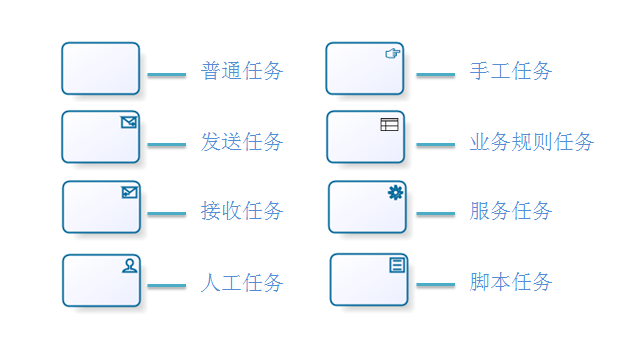


图10‑29任务的8种类型

##### 子流程

有5种不同类型的子流程：

* 嵌入的子流程（EmbeddedSub-Process）：子流程定义在父流程里，可以展开显示它所包含的模型细节，也可以收起隐藏细节（可参见图10-25），通常情况下，我们也直接称之为子流程（Sub-Process）；
* 重用的子流程（ReusableSub-Process）：我们在另外一个业务流程图定义了一个流程，然后在当前流程里调用这个流程，这个流程被我们复用，称之为重用的子流程，在父流程里，我们通过调用活动（Call Activity）调用其他流程：
* 事件子流程（Event Sub-Process）：子流程由事件触发，子流程与父流程之间没有顺序流连接，在扩展元素里，我们会具体讨论该类子流程；
* 事务性子流程（Transaction）：特殊的子流程类型，子流程内部的任务具有事务性，要么一起成功，要么一起失败。我们可以通过某种事务协议例如WS-Transaction控制它的行为，和事件子流程一样，它属于扩展元素；
* 即席子流程（Ad-Hoc Sub-Process）：子流程内部的任务不需要确定顺序关系，由任务执行者根据实际情况自己决定执行顺序。

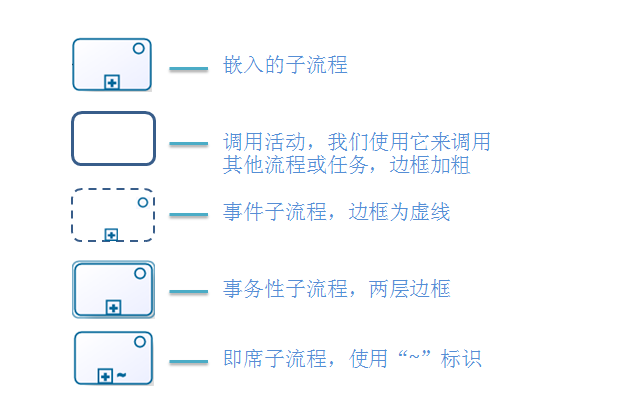


图10‑30子流程的5种类型

         我们看一个即席子流程的例子，当我去医院检查身体时，需要做各种常规检查、抽血和吃早饭，除了抽血一定要在吃早饭之前，其他项目则没有限制。那么可能的顺序有：常规检查->抽血->吃早饭（抽血处排队严重）、抽血->吃早饭->常规检查和抽血->常规检查->吃早饭。

         即席子流程赋予我们执行任务的灵活性，在很多情况下，任务之间并没有过度的依赖关系，在这种情况下，就可以根据案例的实际情况例如可用资源，重新灵活排定这些任务执行的顺序，避免等待和阻塞。即席子流程实现了工作流控制模式里的交叉并行路由（WCP\_17）模式，具体该模式的描述请参考本书的工作流模式附录部分。

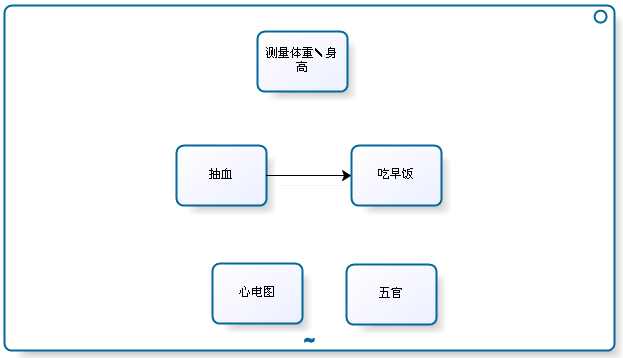


图10‑31即席子流程

##### 调用活动

我们使用调用活动（Call Activity）在当前流程中调用其他流程或全局任务（Global Task）。被调用的流程在当前流程中被称为重用的子流程。那么什么叫全局任务呢？能被其他流程通过调用活动调用的任务称为全局任务。

#### 3.3        事件

在基本元素一节中，我们知道，对一个事件来说，通常会有产生该事件的原因（触发器）和该事件所产生的影响（结果）。在本节中，我们将更加具体的讨论事件的行为，事件有两种行为：捕获触发器和抛出结果。

事件可以由触发器来指出产生该事件的原因，我们使用一个嵌入事件图形内部的图标来标识不同的触发器，存在不同种类的触发器，事件可以捕获触发器。

事件可以由结果来指出该事件所产生的影响，和触发器一样，我们使用一个嵌入事件图形内部的图标来标识不同的结果，事件可以抛出结果。

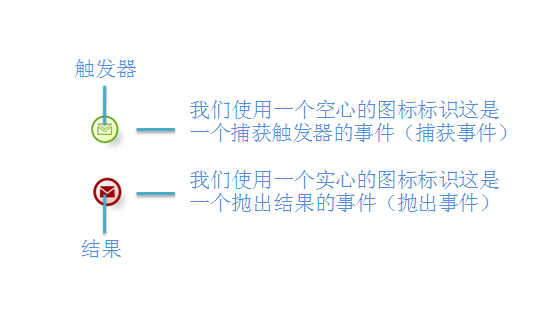


图10‑32捕获事件与抛出事件

         捕获事件等待它所指定的触发器并在捕获后产生Token；抛出事件等待Token并在捕获后产生一个结果。

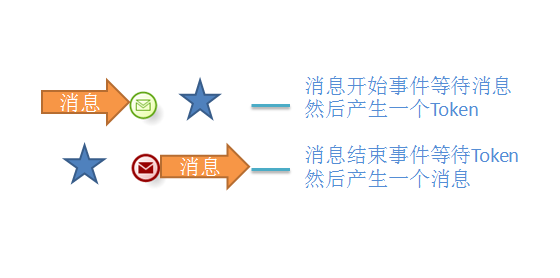


图10‑33消息开始事件与消息结束事件

##### 核心开始事件

         开始事件启动一个流程的新案例，它只有一个唯一的输出顺序流，没有输入顺序流。当它被触发后，会产生一个Token并顺着它的输出顺序流传出，有多种触发器可以触发开始事件，例如到一个预定的时间（定时触发器）、收到一个特定的消息（消息触发器）、收到一个特定的信号（信号触发器）等等。注意，开始事件只能捕获触发器不能抛出结果。

         有3种核心开始事件：

* 定时开始事件（TimerStart Event）：当满足时间条件时产生Token；
* 消息开始事件（MessageStart Event）：当接受到特定的消息后产生Token；
* 信号开始事件（SignalStart Event）：当接受到特定的信号后产生Token。



图10‑34核心开始事件

当满足设定的时间条件时，定时开始事件被触发。有多种设定时间条件的方式：指定一个特定的时间点，例如明天早上9点；指定一个时间间隔，例如两周后；指定一个周期性的时间，例如每天晚上12点开始运行所有的自动化[**测试**](http://lib.csdn.net/base/softwaretest)。

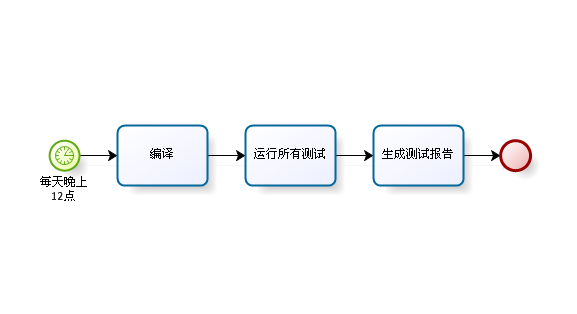


图10‑35定时开始事件

         消息开始事件和信号开始事件都是接受到特定的数据（消息/信号）后产生Token，它们的区别是什么呢？区别在于消息是一对一的，数据的发送和接受是点对点的形式；而信号是一对多的，数据的发送是广播的形式。

消息开始事件从流程外部参与者接受消息，并启动一个案例。

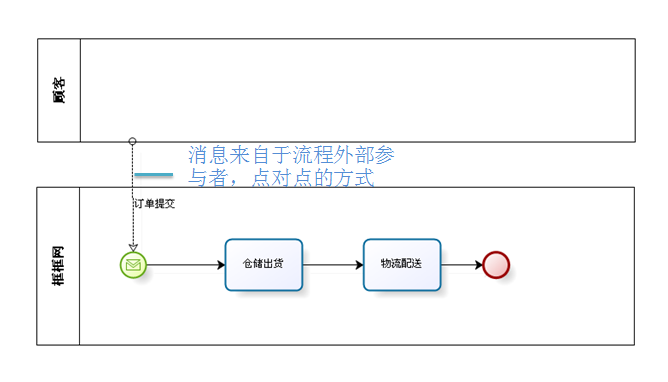


图10‑36消息开始事件

信号开始事件从流程外部参与者或外部流程订阅信号，接受并启动一个案例。

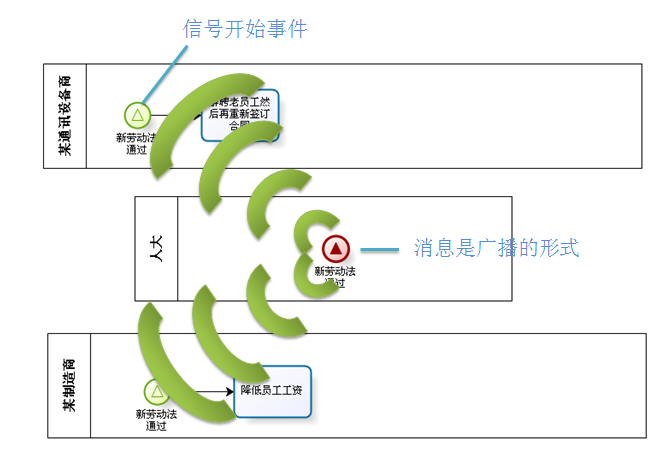


图10‑37信号开始事件

##### 核心结束事件

结束事件可以结束其所在分支的执行，也可以结束整个案例，同时，可以产生并抛出一个结果，例如发送一个消息或信号。它只有输入顺序流，没有输出顺序流。注意，结束事件只能抛出结果不能捕获触发器。

有3种核心结束事件：

* 消息结束事件（MessageEnd Event）：结束所在分支的执行并发送一个消息；
* 信号结束事件（SignalEnd Event）：结束所在分支的执行并广播一个信号；
* 终止结束事件（TerminateEnd Event）：结束整个案例的执行。

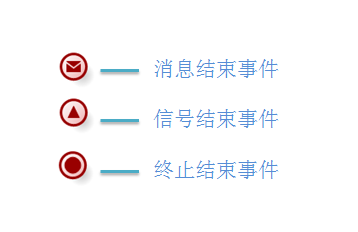


图10‑38核心结束事件

消息结束事件接受一个Token，结束所在分支的执行，给指定的外部流程或参与者发送一个消息。

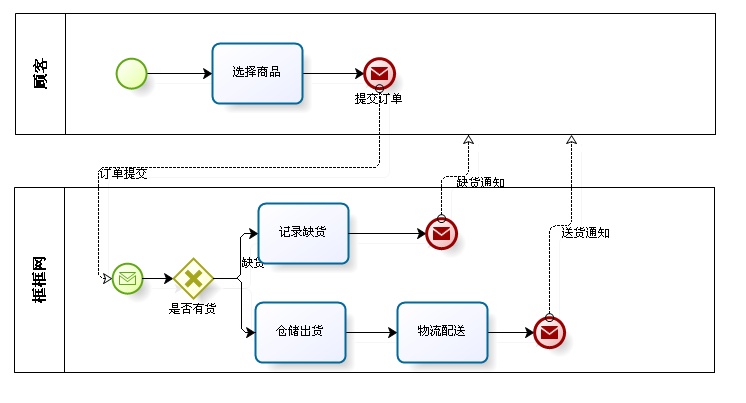


图10‑39消息结束事件

信号结束事件接受一个Token，结束所在分支的执行，广播一个消息，所有对该信号感兴趣的流程和参与者都可以订阅并采取各自的行动。例子请参考图10-37。

终止结束事件结束整个案例的执行。当案例到达某个状态即意味着案例执行结束了，案例中如果存在正在执行的任务，这些任务被取消，这个状态由终止结束事件代表。终止结束事件实现了工作流控制模式里的显式结束（WCP\_43）模式，具体该模式的描述请参考本书的工作流模式附录部分。

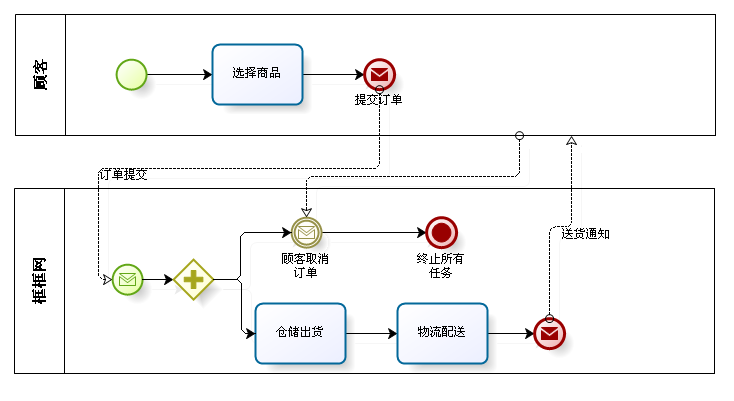


图10‑40终止结束事件

##### 核心即时事件

在开始事件和结束事件之间发生的事件称为即时事件，即时事件会影响流程的流转，但不会启动或直接终止流程的执行。即时事件即可以捕获触发器又可以抛出结果。

有两种使用即时事件的方式，第一种是将事件使用在流程顺序流中：

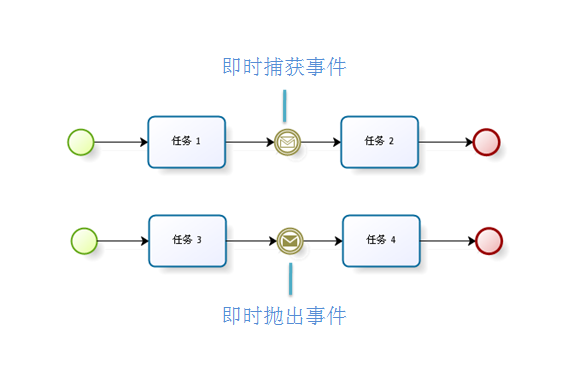


图10‑41使用在流程顺序流中的即时事件

         当事件使用在流程顺序流中时，事件既可以捕获触发器又可以抛出结果。当事件捕获触发器时，它首先等待一个Token，当Token到达后，它再等待一个触发器，当触发器被捕获后，它执行完毕，向后传递Token；当事件抛出结果时，它首先等待一个Token，当Token到达后，它抛出一个结果，然后执行完毕，如果存在输出顺序流的话就向后传递Token。

         第二种是将事件附加在任务或子流程的边界上，此时的事件只能捕获触发器，根据捕获触发器后的不同行为边界事件又分为两种：边界中断事件和边界非中断事件。

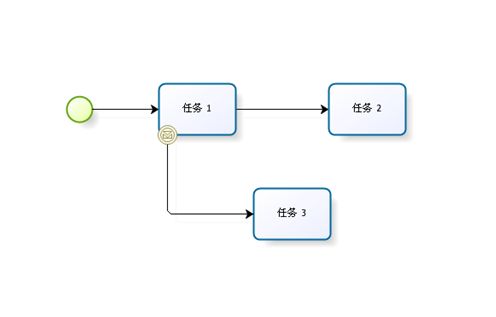


图10‑42边界中断事件

         当在任务1执行过程中，如果边界中断事件捕获了触发器，那么任务1将被停止执行，转为执行任务3；如果边界中断事件没有捕获触发器，那么任务1正常执行，然后执行任务2。

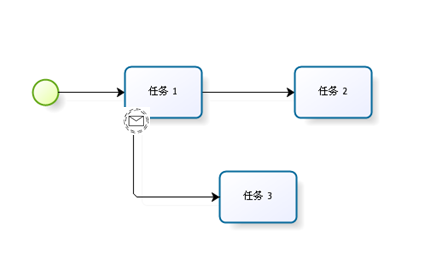


图10‑43边界非中断事件

         当在任务1执行过程中，如果边界非中断事件捕获了触发器，那么任务1不会被停止执行，只是我们同时开始执行任务3，任务1完成后正常触发任务2；如果边界中断事件没有捕获触发器，那么任务1正常执行，然后执行任务2，任务3不会触发。我们使用虚线来标识这是一个非中断事件。

         有4种核心即时事件：

* 普通即时事件（None IntermediateEvent）：只支持使用在流程顺序流中，什么都不做立刻触发；
* 定时即时事件（Timer IntermediateEvent）：支持使用在流程顺序流中捕获触发器，支持附加在任务/子流程边界上中断和非中断任务/子流程执行，满足时间条件时触发；
* 消息即时事件（MessageIntermediate Event）：支持使用在流程顺序流中捕获触发器和抛出结果，支持附加在任务/子流程边界上中断和非中断任务/子流程执行，发送消息或等待消息触发；
* 信号即时事件（SignalIntermediate Event）：支持使用在流程顺序流中捕获触发器和抛出结果，支持附加在任务/子流程边界上中断和非中断任务/子流程执行，广播信号或等待信号触发。

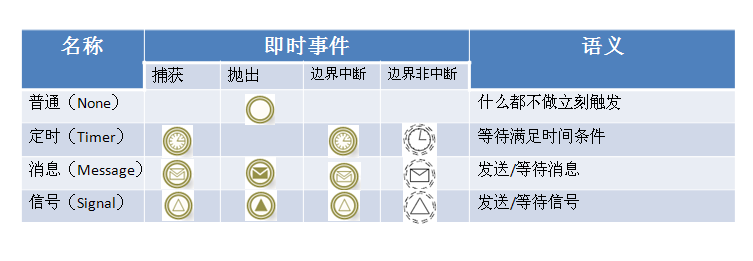


图10‑44核心即时事件

         普通即时事件收到Token后立刻触发，什么都不做，继续向后传递Token。我们使用普通即时事件来标识案例执行到一个特定的状态点或里程碑。

         我们使用定时即时事件有两种用途：为任务限定期限以及为有时间要求的任务进行时间延迟。

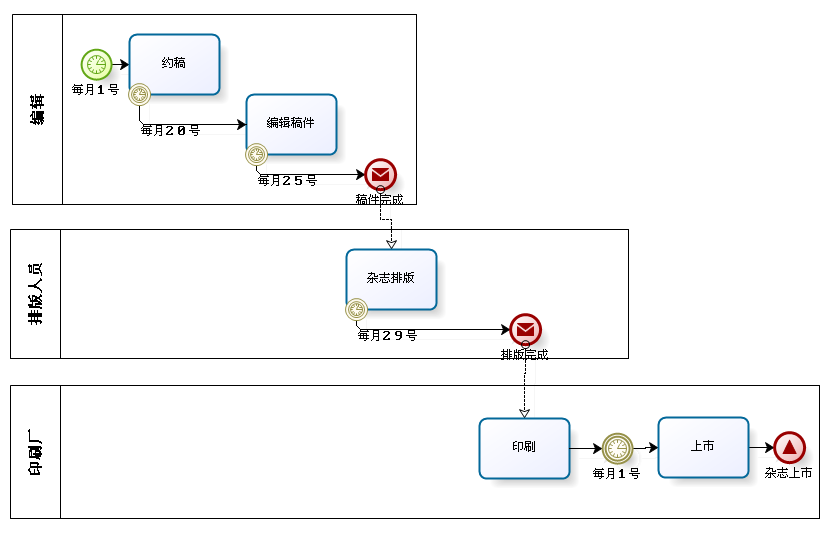


图10‑45定时即时事件

         我们使用消息和信号即时事件对任务执行过程中的消息/信号进行处理，我们也使用它们向外发送消息/信号。

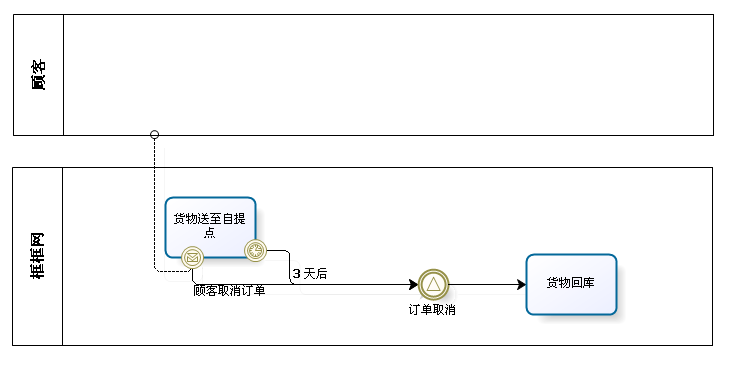


图10‑46消息/信号即时事件

#### 3.4        网关

网关控制流程的分支，在核心元素里，我们只关注一种网关：事件网关（Event-based Gateway）。

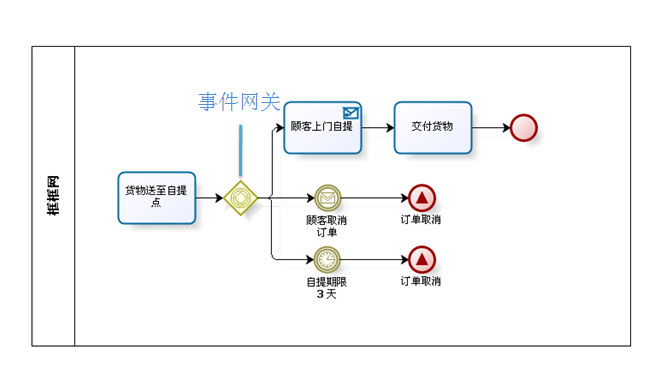


图10‑48事件网关

         和排他网关不同，事件网关并不在输出顺序流上设定的条件，也不在网关自身上设定条件，它的输出顺序流只支持连接即时捕获事件和接受任务。当事件网关接受到一个Token时，它会给后续每个输出顺序流都生成一个Token，后续即时捕获事件等待触发器，后续接受任务等待消息，第一个等待到触发器或消息的事件/任务被触发，继续向后传递Token，其他顺序流上的Token被取消不再向后传递。

事件网关被我们应用于以下场景：案例在某个点有多个分支可供选择，只能有一个分支被实际执行。与排他网关不同：选择并不是在后续分支被触发之前，相反，这个决定被尽可能的延后，每个分支都有可能被执行，具体哪个分支被实际执行取决于具体的案例运行环境。例如，分支都被触发，一旦有一个最新的消息抵达，一个分支开始实际执行，其他的分支将被取消。这一应用场景属于工作流控制模式的延迟决策模式（WCP\_16）。

#### 3.5        数据元素和数据关联

我们使用数据元素标识流程中的数据，这些数据可以是文档可以是邮件也可以是[**数据库**](http://lib.csdn.net/base/mysql)记录，通过数据元素，我们能够在流程图中标示出数据的流向与转换。

有4种数据元素：

* 数据对象（DataObject）：代表随着流程流转的信息，例如业务文档、邮件、业务数据；
* 数据输入（DataInput）:代表整个流程的外部数据输入；
* 数据输出（DataOutput）:代表整个流程的执行结果；
* 数据存储区（DataStore）:代表流程可以访问和写入的共享数据，例如数据库，共享数据的持久化管理并不由当前案例负责。

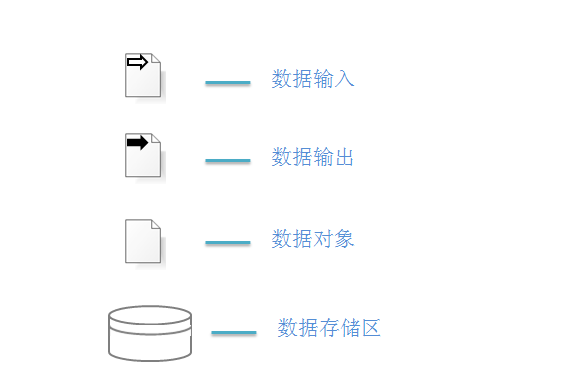


图10‑49数据元素

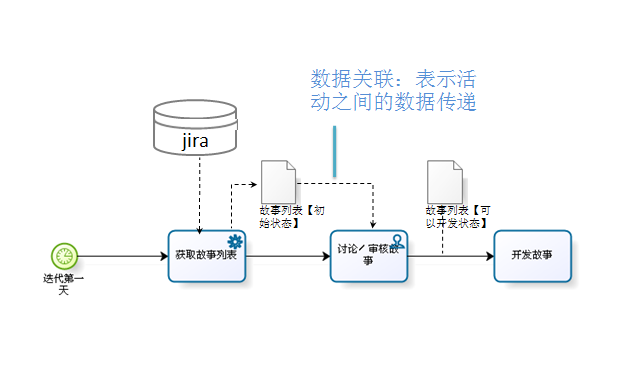


图10‑50使用数据元素标识出流程中的信息传递和信息状态

#### 3.6        组

我们使用组对流程中的元素进行分类。这个分类不会影响组内的顺序流。分类的目的在于更好的组织文档和方便我们进行流程分析。

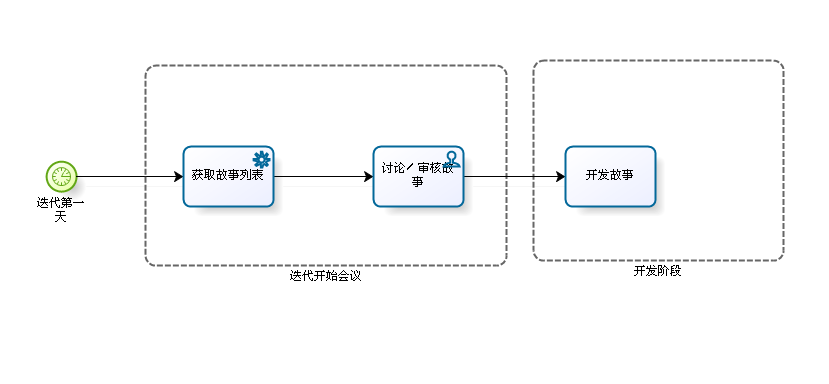


图10‑51组

### [四、BPMN扩展元素](http://blog.csdn.net/ronghao100/article/details/6755206)

### 4.1   扩展元素

什么是BPMN扩展元素？我们为什么要从BPMN元素中界定出一个扩展元素的子集？BPMN扩展元素是我们平时使用频率不高的BPMN元素，这些元素更多的面向开发人员而不是业务人员，它们强调流程执行的细节，例如对事件子流程和事务性子流程的定义和对更多具有具体执行语义事件类型的定义。BPMN扩展元素使BPMN变得复杂，它关注流程的执行层面。对于BPMN扩展元素，我们的观点是大概知道它们所表达的语义即可，需要时再查规范。

BPMN扩展元素包括了这些元素，如下图红色部分所示：

* 两种子流程：事件子流程（EventSub-Process）和事务性子流程（Transaction）；
* 八种事件：条件事件（ConditionalEvent）、链接事件（Link Event）、多重事件（Multiple Event）、并行多重事件（Parallel Multiple Event）、出错事件（Error Event）、补偿事件（CompensationEvent）、取消事件（Cancel Event）和升级事件（Escalation Event）；
* 四种网关：包容性网关（Inclusive）、复杂网关（Complex）、排他事件网关-实例化（ExclusiveEvent-based Gateway-instantiate）和并发事件网关-实例化（Parallel Event-based Gateway-instantiate）；

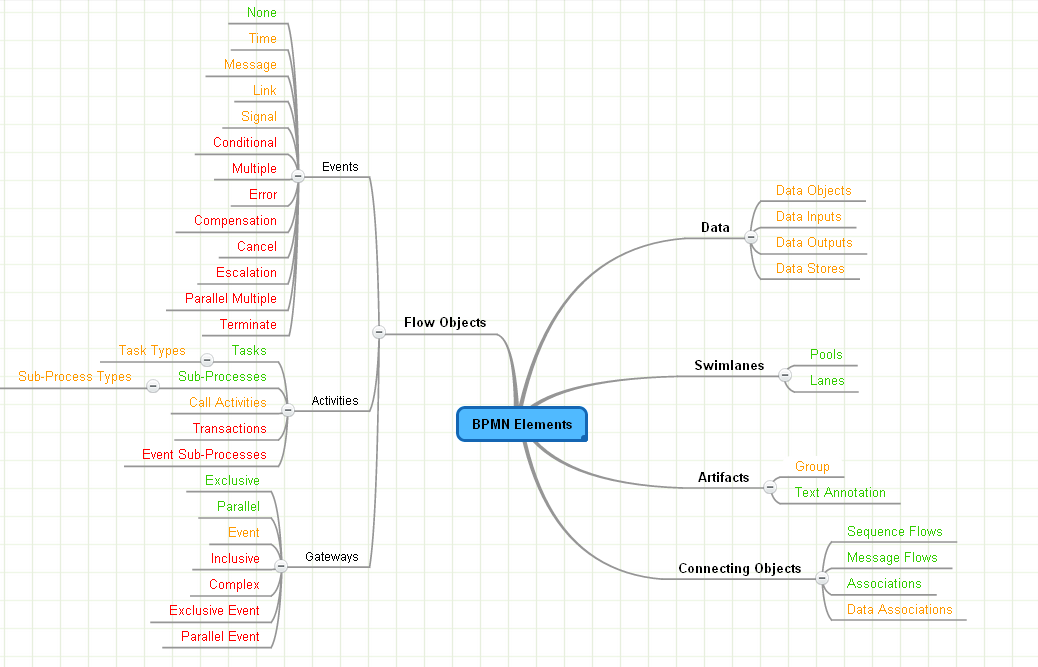


图10‑52BPMN的扩展元素

         BPMN扩展元素全部是流对象元素，在下面的小节中，我们将按照活动、事件和网关的顺序展开，同时，在活动里，我们会首先讨论活动的内部循环和多实例行为。

#### 4.2        活动

##### 内部循环和多实例行为

在实际生活中，我们经常需要重复执行某项任务直至满足一定的条件为止。例如，作为作者，我们需要不断的修改稿子直到编辑认可为止；作为顾客，我们与装修公司的合同里写到：装修不达到要求就不付款。

存在两种类型的循环：一种是类似于程序语言里的“while”，先判断是否满足循环的条件，如果满足才执行，然后再判断循环；一种是类似于程序语言里的“do while”，先执行，然后再判断是否满足循环的条件，如果满足就再循环。活动的内部循环行为实现了工作流控制模式里的结构化循环模式（WCP\_21）。

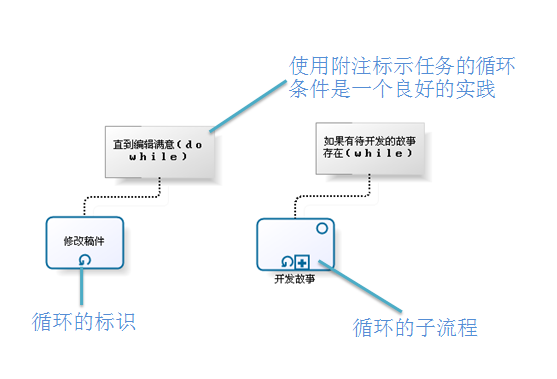


图10‑53活动的循环行为

         与活动的内部循环产生一个活动实例重复执行相比，活动的多实例行为会产生多个活动实例，这些活动实例可以并行执行也可以顺序执行。更加复杂的情况请参考工作流控制模式里的多实例模式。

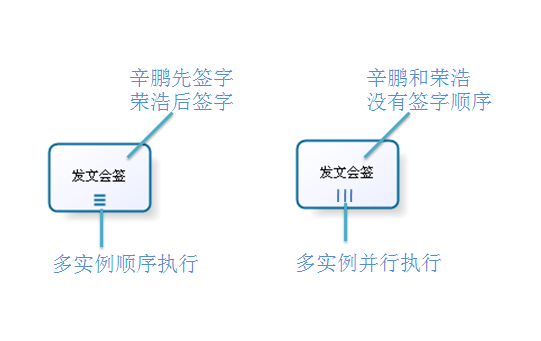


图10‑54活动的多实例行为

##### 子流程

         扩展元素里的子流程有两种：事件子流程和事务性子流程。

         事件子流程被嵌入到子流程里使用，处理子流程执行过程中发生的事件，我们使用虚线框标识事件子流程，它需要由一个事件触发器触发，根据不同的行为，事件子流程又分为中断和非中断两种，中断事件子流程中断父流程的执行，非中断事件子流程与父流程一同执行，我们使用两种不同的开始事件类型来区分这两种事件子流程的行为：中断事件子流程的开始事件（实线圆圈）与非中断事件子流程的开始事件（单虚线圆圈）。

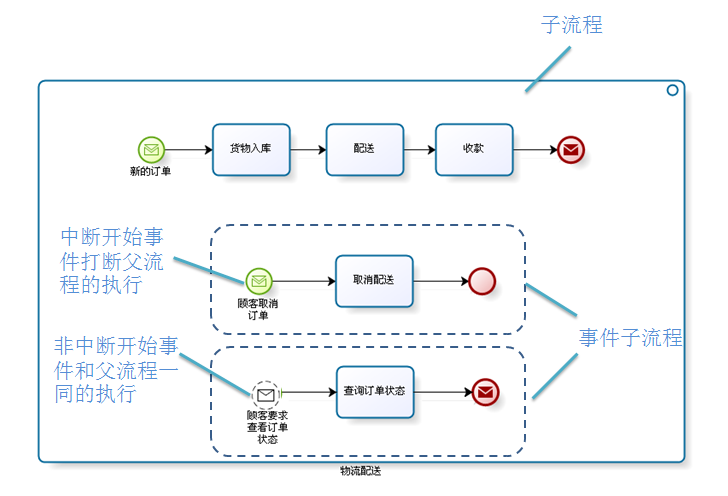


图10‑54事件子流程

事务性子流程具有ACID属性，它具有三个输出：

* 成功完成（SuccessfulCompletion）：事务成功完成，我们使用一个顺序流连接成功后的后续活动；
* 失败完成（Failed Completion）：事务执行不成功被取消，事务回滚，所有定义有补偿活动的活动都被补偿，我们使用一个取消捕获事件来连接事务取消后的后续活动；
* 严重异常（Hazard）：事务执行过程中系统出现严重的异常，事务无法回滚，活动无法补偿，我们使用一个异常捕获事件来处理这种严重的情况，通常我们会继续抛出异常或者通知上一级流程进行处理。

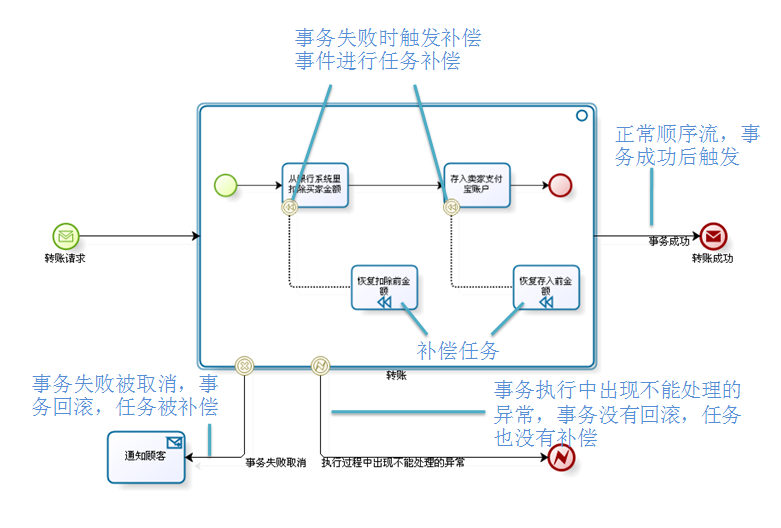


图10‑56事务性子流程