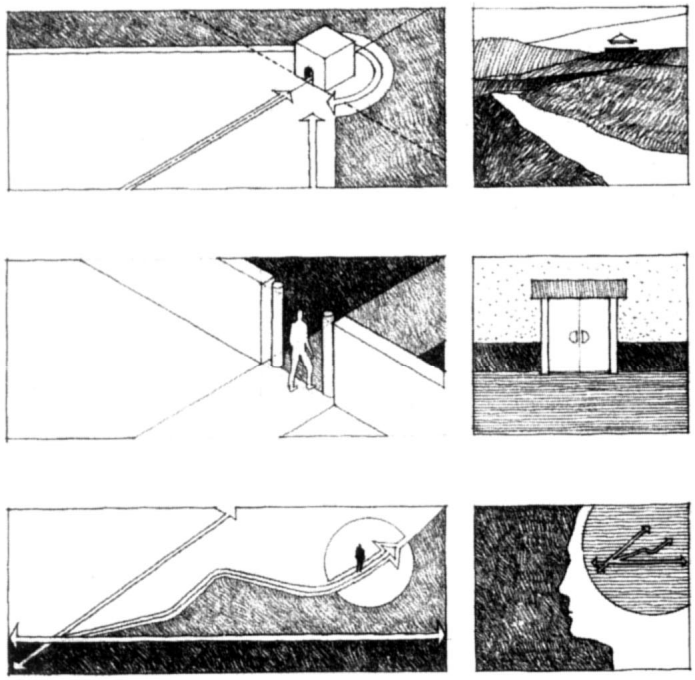
# 第五部分

## 对高级行为建模



# 第20章
# 事件和信号

**本章内容：**

- 信号事件、调用事件、时间事件和变化事件
- 对信号族建模
- 对异常建模
- 处理发生在主动对象或被动对象中的事件

在现实世界中，事情在发生。不仅事情在发生，而且许多事情都在同一时间发生，或发生在最意想不到的时间。“发生的事情”称作事件，它表示对一个在时间和空间上占据一定位置的有意义的事情的规格说明。

在状态机的语境中，使用事件来描述一个激发的产生，激发能够触发一个状态的转换。事件包括信号、调用、时间推移或状态改变。

事件可以是同步的，也可以是异步的，所以对事件的建模关系到对进程和线程的建模。

## 20.1 入门

一个完全静态的系统是极端无趣的，因为没有事情发生。所有真实的系统自身都含有某些动态特性，并且这些动态特性是由内部或外部发生的事情所触发的。在一个ATM机上，动作是由一个用户按下按钮引发而开始一个事务的。在一个自动机器人中，动作是由机器人碰上一个对象而引发的。在一个网络路由器中，动作是由检测消息缓冲区是否溢出而引发的。在一个化工厂中，动作是由一个化学反应所需的一个时间段用满而引发的。

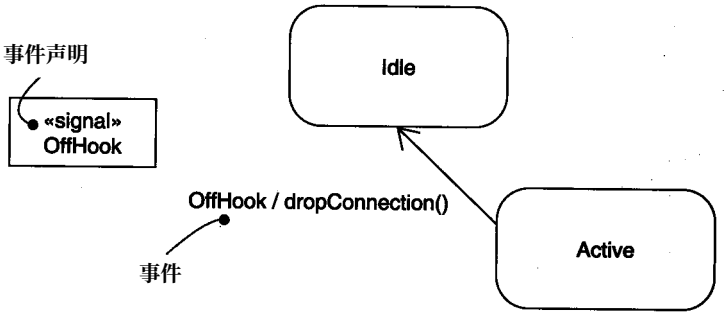



图20-1 事件

在UML中，每个发生的事情都被建模为一个事件。一个事件是对一个在时间和空间上占据一定位置的有意义的事情的规格说明。信号、时间推移或状态改变是异步事件，表示事件能在任何时间发生。调用一般是同步事件，表示对一个操作的调用。

UML提供了对事件的图形化表示，如图 20-1所示。这种表示法允许你将事件的声明（如信号offHook）以及用来触发一个状态转换的事件的使用（如信号 offHook，导致了一部电话从Active到Idle的状态转换）可视化。

## 20.2 术语和概念

事件（*event*）是对一个在时间和空间上占有一定位置的有意义的事情的规格说明。在状态机的语境中，一个事件是一次激发的产生，激发能够触发一个状态转换。信号（*signal*）是一种事件，表示一个在实例间进行通信的异步激发的规格说明。

1. 事件的种类

事件可以是内部的事件或外部的事件。外部的事件是在系统和它的参与者之间传送的事件。例如一个按钮的按下和一个碰撞检测器的中断都是外部事件。内部事件是在系统内部的对象之间传送的事件。溢出异常是一个内部事件的例子。【在第16章中讨论参与者；在第 31章中讨论系统。】

你可以用UML对4种事件进行建模：信号、调用、时间推移和状态的一次改变。【对象的创建和撤销也是一种信号，这在第 15章中讨论。】

2. 信号

一个信号表示由一个对象异步地发送、并由另一对象接收的一个已命名的对象。目前多数编程语言都支持异常情况的处理，异常情况是需要你去建模的最常见的一种内部信号。

信号和简单的类有许多共同之处。例如，信号可以有实例，尽管一般不需要对实例进行显式地建模。信号还可以包含在泛化关系中，以便对事件的层次结构建模，有些信号是一般的（如信号 NetworkFailure），有些信号是特殊的（如 NetworkFailure的一个特化WarehouseServerFailure）。像类一样，信号也可以有属性和操作。【在第4章和第9章中讨论类；在第5章和第10章讨论泛化。】

**注释** 一个信号的属性以它的参数形式出现。例如当你发送一个信号 Collision，你可以用参数的形式说明它的属性值，例如 Collision（5.3）。

一个信号可以作为状态机中一个状态转换的动作而被发送，或者作为交互中一个消息的发送而被发送。一个操作的执行也可以发送信号。事实上，当你为一个类或一个接口建模时，说明该元素行为的一个重要部分就是说明它的操作所发送的信号。在 UML中，可以用构造型为send的依赖关系对一个操作和它所发送的事件之间的关系进行建模。【在第21章中讨论状态机；在第15章中讨论交互；在第11章中讨论接口；在第5章中讨论依赖；在第6章中讨论构造型。】

在UML中，如图20-2所示，你可以将信号（和异常）建模为构造型化的类。你可以用一个

构造型为send的依赖来表示一个操作发送了一个特定的信号。

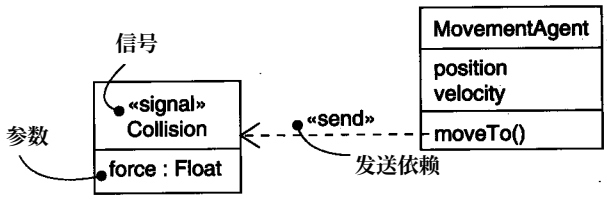



图20-2 信号

3. 调用事件

就像一个信号事件代表一个信号的发生一样，一个调用事件代表一个操作的调度。在这两种情况中，事件均可触发状态机中的一个状态转换。【在第21章中讨论状态机。】

信号是一个异步事件，而调用事件一般来说是同步的。也就是说，当一个对象调用另一个具有状态机的对象的一个操作时，控制就从发送者传送到接收者，该事件触发转换，完成操作后，接收者转换到一个新的状态，控制返还给发送者。

如图20-3所示，对调用事件的建模和对信号事件的建模没有区别。两种情况下都要显示带有参数的事件，并把它作为对状态转换的触发。

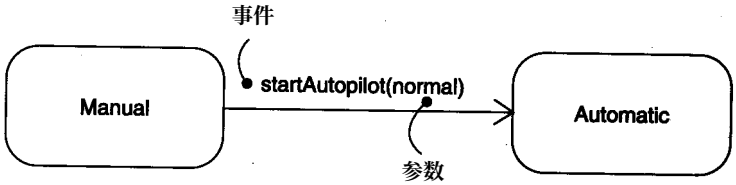



图20-3 调用事件

**注释** 尽管没有可视的线索能够区分信号事件和调用事件，但在模型的基架上它们的区别还是显而易见的。通过在操作列表中标明该操作，事件的接收者将知道其中的区别。通常，一个信号由它的状态机来处理，而一个调用事件则由一个方法来处理。你可以利用工具进行从事件到信号或到操作的导航。

4. 时间事件和变化事件

时间事件是表示一段时间推移的事件。如图 20-4所示，在UML中，用关键字after，后面跟着计算一段时间的表达式来对一个时间事件建模。表达式可以是简单的（如 after(2 seconds)），也可以是复杂的（如after(1 ms since exiting old)）。除非显式地说明，否则，这样一个表达式的开始时间是进入当前状态的时间。

变化事件是表示状态中的一个变化或某些条件的满足的事件。如图 20-4所示，在UML中，用关键字when，后面跟随布尔表达式来对一个变化事件建模。你可以用这样的表达式来标记一个绝对的时间（如 when(time = 11:00)），或对一个表达式作不间断地测试（如 when(altitude < 1000)）。

**注释** 尽管变化事件可以对一个要被不断测试的条件建模，但你还是可以典型地分析情

况，来发现何时在具体的点上及时地测试条件。

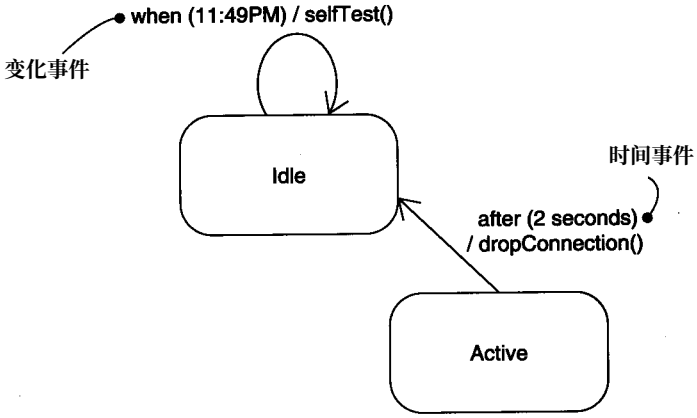



图20-4 时间事件和变化事件

5. 发送和接收事件

信号事件和调用事件至少涉及两个对象：一个是发送信号或调用操作的对象，另一个是事件指向的对象。因为信号是异步的，而且异步调用本身也是信号，所以事件的语义与主动对象和被动对象的语义是相互影响的。【在第22章中讨论进程和线程。】

任何类的任何实例都可以发送信号给一个接收对象，或调用接收对象的操作。当对象发送信号时，发送者调度信号，然后沿它的控制流继续进行，并不等待接收者的任何响应。例如，如果与一个ATM系统交互的参与者发送信号 pushButton，该参与者将沿着自己的路线继续进行，而不依赖于信号被发送到的系统。相反，当一个对象调用一个操作时，发送者调度这个操作，然后等待接收者的响应。例如，在一个商务系统中，类 Trader的一个实例可能调用在类Trade的某些实例上的操作confirmTransaction，因而影响对象Trade的状态。如果这是一个同步的调用，那么，对象Trader将一直等待直到该操作结束。【在第13章中讨论实例。】

> **注释** 在某些情况下，你可能想要显示一个对象向一组对象（多点播送）发送信号，或者向系统中列出的所有对象（广播）发送信号。对多点播送建模时，应显示一个对象向一组接收者集合上发送信号。对广播建模时，应显示一个对象，它发送信号到另一个代表整个系统的对象。

任何类的任何实例都能接收调用事件或信号。如果这是一个同步调用事件，那么发送者和接收者都处在该操作执行期间的一个汇集点上。也就是说，发送者的控制流中紧接着插入了接收者的控制流，直至该操作的活动完成。如果这是一个信号，那么发送者和接收者并不汇合：发送者调度信号后并不等待接收者的响应。在两种情况下，这个事件都可能失败（如果没有定义对事件的响应），它将触发接收者的状态机（如果有的话）或者唤醒一个普通的方法调用。【在第21章中讨论状态机；在第22章中讨论主动对象。】

在UML中，你可以将一个对象可能接收的调用事件建模为这个对象的类上的操作。在 UML中，你可以在类的附加栏中对信号命名，来对对象可能接收的已命名的信号进行建模，如图 20-5

所示。【在第4章中讨论操作；在第4章中讨论类的附加栏。】

注释 你也可以将已命名的信号以相同的方式附加到一个接口上。在这种情况下，你在这个附加栏列出的信号并不是一个信号的声明，而只是一个信号的使用。在类的正常栏中列出的是异步操作的信号。【在第11章中讨论接口；在第22章中讨论异步操作。】

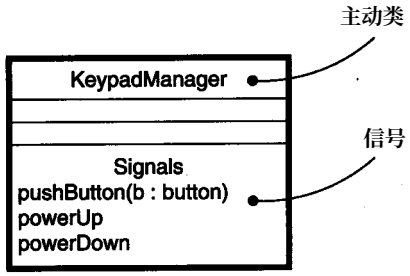



图20-5 信号和主动类

## 20.3 普通建模技术

### 20.3.1 对信号的族建模

在大多数事件驱动的系统中，信号事件是分层的。例如，一个自主机器人可以辨别外部信号（如Collision）和内部信号（如HardwareFault）。然而，内部信号和外部信号未必不相交。在这两种粗略的分类中，你甚至可以发现特化。例如，信号 HardwareFault可以被进一步特化为 BatteryFault和MovementFault。甚至这些信号还可以进一步被特化，如MoterStall是MovementFault的一种。【在第5章和第10章中讨论泛化。】

通过以这种方式对信号的层次建模，你可以说明多态的事件。例如，考虑一个状态机，它有一个仅当接收到Motorstall才能触发的转换。作为这个层次中的叶子信号，该转换只能被此信号触发，所以，它不是多态的。相反，假如状态机存在一个由 HardwareFault的接收所触发的转换，则这个转换是多态的，并能被一个 HardwareFault或它的任何一种特化信号（包括BatteryFault、MovementFault和MotorStall）触发。【在第21章中讨论状态机。】

对信号的族建模，要遵循如下的策略：

- 考虑一组给定的主动对象可能响应的所有不同种类的信号。
- 寻找信号的公共类，并使用继承将它们放在一般/特殊结构中。提升较为一般的信号，并降低较为特殊的信号。
- 在这些主动对象的状态机中寻找多态性，在发现多态性的地方，必要时通过引入中间的抽象信号来调整层次结构。

图20-6是对一个由自主机器人处理的信号的族建模。注意根信号（ RobotSignal）是抽象的，它没有任何具体的实例。这个信号有两个具体的直接特化信号（ Collision和

HardwareFault)，其中HardwareFault还可以进一步被特化。注意信号 Collision有一个参数。【在第5章中讨论抽象类。】

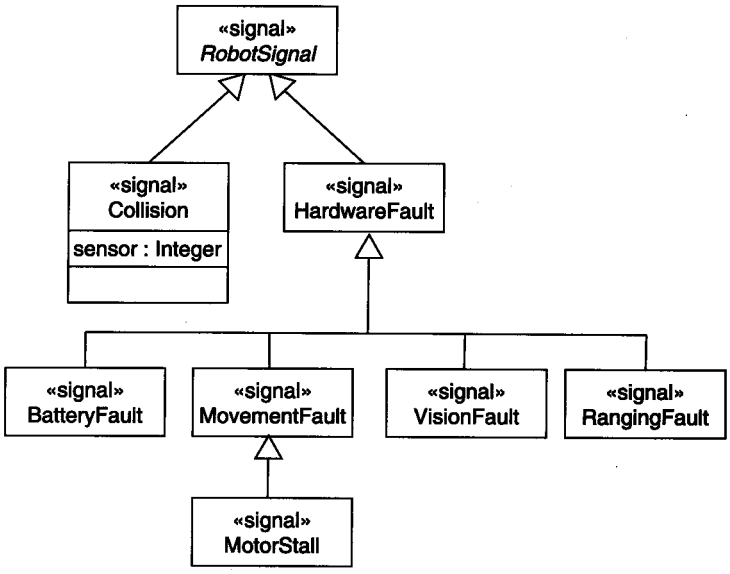



图20-6 对信号的族建模

## 20.3.2 对异常建模

可视化、详述和文档化类或接口的行为的一个重要部分是说明它的操作会产生的异常情况。如果交给你一个类或接口，你可以调用的操作是很清楚的，但每个操作所可能发生的异常则不清楚，除非你显式地对它们建模。【在第4章和第9章中讨论类；在第11章中讨论接口。】

在UML中，异常是一种信号，并被建模为构造型化的类。异常可以被附加到操作的说明中。对异常建模在某种程度上与对信号的一般族建模相反。对一个信号的族建模主要是说明一个主动对象接收的各种信号；而对异常建模主要是说明一个对象可能通过它的操作来发出的各种异常。【在第6章中讨论构造型。】

对异常建模，要遵循如下的策略：

- 对于每个类和接口以及这些元素的每个操作，考虑可能发生的异常条件。
- 按分层结构排列这些异常。提升一般的异常，降低特殊的异常，必要时引入中间异常。
- 对于每个操作，描述可能出现的异常。可以显式地表示（通过显示从一个操作到它的异常的Send依赖关系），也可以将它放在操作的说明中。

图20-7描述的是对异常的一个分层结构建模，这些异常是由一个包容器类（如模板类 Set）的标准库引发的。这个分层结构以抽象信号 Exception为根，它包括 3个特殊异常：Duplicate、Overflow和Underflow。如图20-7所示，操作add引发异常Duplicate和Overflow，操作remove仅引发异常Underflow。换一种做法，可以通过在每个操作的规格说明中命名异常，将这些依赖放置在后台中。不论哪种方法，通过了解每个操作可能发送的所

有异常，你就可以创建正确地使用类Set的客户端。【在第9章中讨论模板类。】

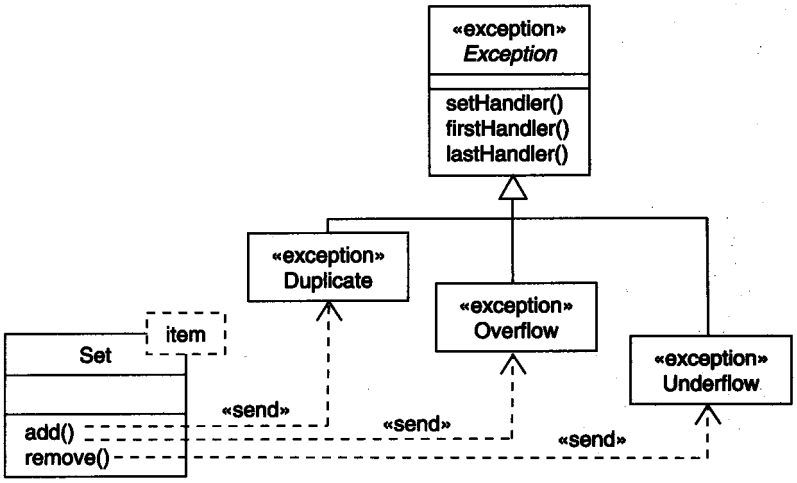



图20-7 对异常建模

## 20.4 提示和技巧

当你对事件建模时，要遵循如下策略：

- 建立信号的层次结构，以便发掘相关信号的公共特性。
- 不要使用发送信号，尤其不要使用发送异常来代替正常的控制流。
- 确保每个可能接收事件的元素背后都有一个适当的状态机。
- 确保不仅对那些接收事件的元素建模，而且还对那些发送事件的元素建模。

在UML中绘制一个事件时，要遵循如下的策略：

- 一般来说，要显式地对事件的分层结构建模，但要在每个类的基架或是在发送或接收这个事件的操作中，对它们的使用建模。