第24章 状 态 图

本章内容：

- 对反应型对象建模
- 正向工程和逆向工程

状态图是UML中对系统的动态方面进行建模的五种图之一。一个状态图显示了一个状态机。活动图是状态图的一种特殊情况，其中所有或大多数状态是活动状态，而且所有或大多数转换都是由源状态中活动的完成所触发的。这样，活动图和状态图在对一个对象的生命期建模时都是有用的。然而，活动图显示的是从活动到活动的控制流，而状态图显示的是从状态到状态的控制流。【顺序图、协作图、活动图和用况图也对系统的动态特性建模；在第 18章中讨论顺序图和协作图；在第 19章中讨论活动图；在第17章中讨论用况图。】

状态图用于对系统的动态方面建模。大多数情况下，它包括对反应型对象的行为建模。一个反应型对象是这样一个对象，它的行为是通过对来自它的语境外部的事件作出反应来最佳刻画的。反应型对象具有清晰的生命期，它的当前行为受它的过去行为影响。为了可视化、详述、构造和文档化一个单独对象的动态特性，状态图可以被附加到类、用况或整个系统上。

状态图不仅在对一个系统的动态方面建模时有重要的意义，而且对于通过正向工程和逆向工程来构造可执行的系统也很重要。

24.1 入门

试考虑一个投资者，她为一座新摩天大楼的建造提供资金。她不可能对建造过程的细节感兴趣。材料的选择、贸易进度和许许多多关于工程细节的会议，对建造者来说是很重要的活动，但对提供项目资金的人来说却远没有那么重要。【在第1章中讨论对狗窝和对高大的建筑物建模之间的不同。】

投资者感兴趣的是对投资的良好回报，这也意味着保护投资免受风险。一个真正信任他人的投资者，会提供给建造者一笔钱，然后离开一段，仅当建造者准备递交大厦的钥匙时才返回。这样的投资者真正感兴趣的是这座大厦的最终状态。

较为务实的投资者仍是信任建造者的，但也想在交出钱之前证实该项目是对头的。因此，审慎的投资者不是只给建造者一大笔不在意的钱去随便花，而是将为这个项目设立清晰的里程碑，每一个里程碑对应某些活动的完成，并且仅当完成之后，下一个阶段的项目资金才会交到建造者手上。例如，在项目开始时，可能提供少量资金进行建筑设计工作。当建造事宜被批准时，可能为项目提供较多的资金。在这项工作做得使项目资金监管人员感到满意后，才可能拨

给更大量的资金，以便建造者破土动工。接下去，从破土动工到取得产权证书，这中间还会有其他的里程碑。

每个这样的里程碑都命名了该工程的一个稳定的状态：结构设计完成、工程设计完成、破土动工、基础设施完成、大厦经盖章批准使用等。对于投资者来说，跟踪大厦状态的变化比跟踪活动流要更加重要，而对建造者来说，通过波特图对工程的工作流建模要更重要。【在第19章中讨论甘特图和波特图。】

在对一个软件密集型系统建模时，你也将发现可视化、详述、构造和文档化某些对象的行为的最自然的方法是着眼于从状态到状态的控制流，而不是着眼于从活动到活动的控制流。你可以用一个流程图（并在UML中使用一个活动图）来描述后者。想象一下，对一个嵌入式家庭安全系统建模。这样的系统需要不间断地工作，并对来自外部的事件（如一个窗户被打破）作出反应。另外，事件的顺序会改变系统行为的方式。例如，如果系统是第一次报警，对窗户被打破的检测将只是触发一个警报。描述这样一个系统的行为，可通过对它的稳定状态（如Idle、Armed、Active和Checking等）建模、对触发从状态到状态变化的事件建模和对在每个状态改变时发生的动作进行建模来做最好的说明。【在第19章中讨论作为流程图的活动图；在第21章中讨论状态机。】

在UML中，状态图用于对一个对象的按事件排序的行为建模。如图 24-1所示，一个状态图是强调从状态到状态的控制流的状态机的简单表示。

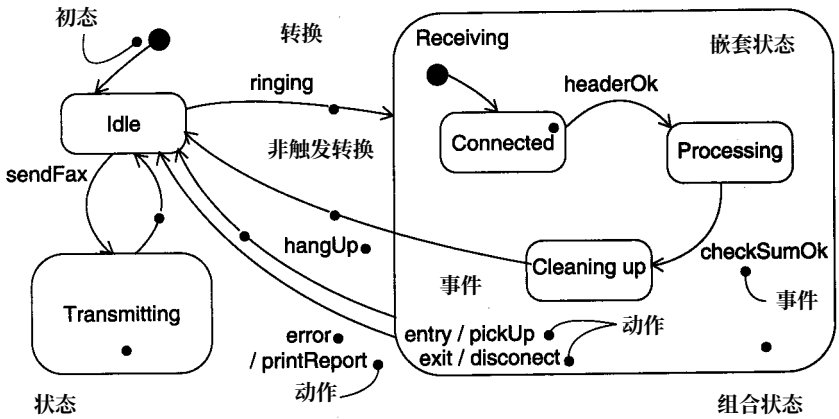

图24-1 状态图

24.2 术语和概念

一个状态图（*statechart diagram*）显示了一个状态机，它强调从状态到状态的控制流。一个状态机（*state machine*）是指一个行为，它说明对象在它的生命期中响应事件所经历的状态序列以及它们对那些事件的响应。一个状态（*state*）是指在对象的生命期中的一个条件或状况，在此期间对象将满足某些条件、执行某些活动或等待某些事件。一个事件（*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。转换（*transition*）是两个状态之间的关系，它指明对象在第一个状态中执行一定的动作，并当特定事件发生或特定条件满足时进入第二个状态。活动（*activity*）是状态机中进行的非原子执行。动作（*action*）是由引起模型状态改变或值的返回的可执行的原子计算组成的。在图形上，一个状态图是顶点和弧的集合。

1. 公共特性

状态图只是一种特殊种类的图形，它拥有像所有其他图一样的公共特性 —— 即名称和投影在一个模型上的图形内容。状态图与其他各种图的区别是它的内容。【在第7章中讨论图的一般特性。】

2. 内容

状态图通常包括：

- 简单状态和组合状态
- 转换，包括事件和动作

【在第21章中讨论简单状态、组合状态、转换、事件和动作；在第 19章中讨论活动图。】

注释 状态图基本上是一个状态机中的元素的一个投影。这意味着状态图可以包括分支、分叉、汇合、动作状态、活动状态、对象、初态、终态、历史状态等。事实上，状态图包括状态机的所有特征。活动图和状态图的区别在于，活动图基本上是活动图形上的元素的投影，它是状态机的一种特殊的情况，其中所有或大多数状态是活动状态，并且所有或大多数的转换都是由源态中的活动的完成所触发的。

与所有其他图一样，状态图也包括注解和约束。【在第6章中讨论节点和约束。】

3. 一般应用

你可以用状态图对系统的动态方面建模。这些动态方面可以包括出现在系统体系结构的任何视图中的任何一种对象的按事件排序的行为，这些对象包括类（含主动类）、接口、构件和节点。【在第2章中讨论体系结构的5个视图；在第13章中讨论实例；在第4章和第9章中讨论类；在第22章中讨论主动类；在第11章中讨论接口；在第25章中讨论构件；在第26章中讨论节点。】

当使用状态图对系统的某些动态方面建模时，你实际上是在任何建模元素的语境中做这件事情。然而，通常你将在整个系统、子系统或类的语境中使用状态图。你也可以把状态图附加到用况（对脚本建模）上。

当你对系统、类或用况的动态方面建模时，通常是按以下一种方式使用状态图。【在第16章中讨论用况；在第31章中讨论系统。】

- 对反应型对象建模

一个反应型 —— 或事件驱动的 —— 对象是这样一个对象，其行为通常是由对来自语境外部的事件作出反应来最佳刻画的。反应型对象在接收到一个事件之前通常处于空闲状态。当它接收到一个事件时，它的反应常常依赖于以前的事件。在这个对象对事件作出反应后，它就又变成空闲状态，等待下一个事件。对于这种对象，你将着眼于对象的稳定状态、能够触发从状态到状态的转换的事件、以及当每个状态改变时所发生的动作。

注释 形成对照的是，你将使用活动图对一个工作流或一个操作建模。活动图更适合对随时间变化的活动流建模，例如可在流程图中表示的那种活动流。

24.3 普通建模技术

24.3.1 对反应型对象建模

使用状态图的最常见的目的是对反应型对象、尤其是对类、用况、和整个系统的实例的行为建模。交互是对共同工作的对象群体的行为建模，而状态图是对一个单独的对象在它的生命期中的行为建模。活动图是对从活动到活动的控制流建模，而状态图是对从事件到事件的控制流建模。【在第15章中讨论交互；在第19章中讨论活动图。】

当你对反应型对象的行为建模时，基本上要说明三种事情：这个对象可能处于的稳定状态、触发从状态到状态的转换的事件、以及当每个状态改变时发生的动作。对反应型对象的行为建模还包括对一个对象的生命期建模，对象的生命期从这个对象的创建时刻开始，直到它被撤销而结束。强调在其中可能发现的这个对象的稳定状态。【在第21章中讨论对一个对象的生命周期建模。】

一个稳定状态表示一个条件，对象可能在该条件下存在一段可识别的时间。当一个事件发生时，这个对象可能发生从状态到状态的转换。这些事件也可能触发自身和内部的转换，在这种转换中转换的源和目的是同一个状态。在对事件或对状态的变化的反应中，对象是通过执行一个动作来作出响应的。【在第23章中讨论时间和空间。】

注释 当你对一个反应型对象的行为建模时，可以通过把动作联系到一个转换或一个状态的变化来说明该动作。用技术术语来说，其全部动作都附加到转换上的状态机叫做米利机（Mealy machine）；全部动作都附加到状态上的状态机叫做莫尔机（Moore machine）。从数学上讲，这两种方式具有同等的能力。实际上，通常你会使用米利机和莫尔机的组合来开发状态图。

对一个反应型对象建模，要遵循如下的策略：

- 选择状态机的语境，不管它是类、用况或是整个系统。
- 选择这个对象的初态和终态。为了指导模型的剩余部分，可能要分别地说明初态和终态的前置条件和后置条件。【在第10章中讨论前置和后置条件；在第11章中讨论接口。】
- 考虑对象可能在其中存在一段时间的条件，以决定该对象所在的稳定状态。从这个对象的高层状态开始，然后考虑它的可能的子状态。
- 在对象的整个生命期中，决定稳定状态的有意义的偏序。
- 决定可能触发从状态到状态的转换的事件。将这些事件建模为触发者，它触发从一个合法状态序列到另一个合法状态序列的转换。
- 把动作附加到这些转换（如同在米利机中）上，并且/或者附加到这些状态（如同在莫尔机中）上。

- 考虑通过使用子状态、分支、分叉、汇合和历史状态，来简化状态机。
- 核实所有的状态都是在事件的某种组合下可达的。
- 核实不存在死角状态，即不存在那种没有事件的组合能将这个对象转换出来的状态。
- 通过手工或者通过使用工具跟踪状态机，核对所期望的事件序列以及它们的响应。

例如，图24-2显示了一个状态图，用于分析一个简单的与语境无关的语言，正如在向 XML 输入或输出消息的系统中可能发现的那样。在这种情况下，该机器被设计得能分析与语法相匹配的一个字符流。

```
message : '<' string '>' string ';'
```

其中，第一个字符串表示一个标签，第二个字符串表示该消息体。给定一个字符流，只有遵从这个语法的形式良好的消息才可能被接受。

如图所示，这个状态机仅有三个稳定状态： `Waiting`、`GettingToken`和`GettingBody`。这个状态图被设计成有动作附加在转换上的一个米利机。事实上，在这个状态机中仅有一种事件，即对带有实际参数`c`（一个字符）的`put`的引用。在`Waiting`状态下，该机器丢弃任何不是开始符号（通过监护条件来说明）的字符。当接收到一个开始符号时，该对象的状态就改变为`GettingToken`。在这个状态中，机器保存任何不是结束符号（通过监护条件来说明）的字符。当接收到一个结束符号时，该对象的状态就改变为 `GettingBody`。在这个状态中，机器保存任何不是一个消息体结束标记（通过监护条件来说明）的字符。当接收到一个消息结束标记时，该对象的状态就改变为 `Waiting`，并返回一个表示该消息已被分析过的值（而机器等待接收另一个消息）。【在第20章中讨论事件。】

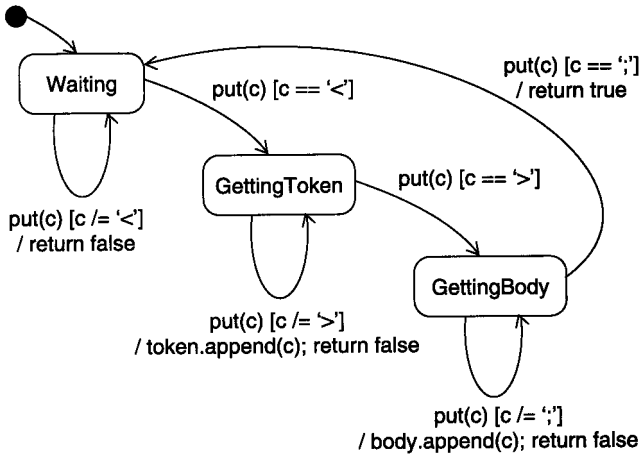

图24-2 对反应型对象建模

注意这个状态图描述了一个不间断运行的状态机；它没有终态。

24.3.2 正向工程和逆向工程

对状态图进行正向工程（*forward engineering*）（从一个模型产生代码）是可能的，如果这

个图的语境是一个类，尤其可能。例如，使用上一个状态图，一个正向工程工具能够为类MessageParser产生下面的Java代码。

```
class MessageParser {
public
  boolean put(char c) {
    switch (state) {
      case Waiting:
        if (c == '<') {
          state = GettingToken;
          token = new StringBuffer();
          body = new StringBuffer();
        }
        break;
      case GettingToken :
        if (c == '>')
          state = GettingBody;
        else
          token.append(c);
        break;
      case GettingBody :
        if (c == ';')
          state = Waiting;
        else
          body.append(c);
        return true;
    }
    return false;
  }
  StringBuffer getToken() {
    return token;
  }
  StringBuffer getBody() {
    return body;
  }
private
  final static int Waiting = 0;
  final static int GettingToken = 1;
  final static int GettingBody = 2;
  int state = Waiting;
  StringBuffer token, body;
}
```

这里需要一点小技巧。正向工程工具必须生成所需的私有属性和最终的静态常量。

逆向工程（*reverse engineering*）（从代码产生模型），从理论上说是可能的，但实际上并不很有用。选择什么构成一个有意义的状态是设计者的观点。逆向工程工具没有抽象的能力，所以不能自动产生有意义的状态图。比逆向工程更有趣的是关于实施系统执行的一个模型的动画。例如，给出上一个图，一个工具就像它们在运行系统中那样模仿图中的状态。类似地，也可以模仿转换的触发，显示事件的接收和执行动作的结果。在一个调试程序的控制下，你可以控制

执行的速度，设置断点，并在感兴趣的状态上停止动作，检测单个对象的属性值。

24.4 提示和技巧

当在UML中创建状态图时，要记住每个状态图只是系统的一个动态方面在同一模型上的一个投影。一个单独的状态图能捕捉单个反应型对象的语义，但没有任何一张状态图能捕获整个复杂系统的语义。

一个结构良好的状态图，应满足如下的要求：

- 关注于交流系统的动态的一个方面。
- 仅包含对于理解这个方面很重要的那些元素。
- 提供与它的抽象层次相一致的细节信息（只揭示对于理解很重要的那些特征）。
- 在米利机和莫尔机两种方式之间运用平衡。

当绘制一个状态图时，要遵循如下的策略：

- 给它取一个能表达它的目的的名称。
- 先对对象的稳定状态建模，然后对从状态到状态的合法转换建模。把分支、并发和对象流作为第二位的考虑，也可能把它们放在分离的图中。
- 摆放这些元素，尽量避免线段交叉。