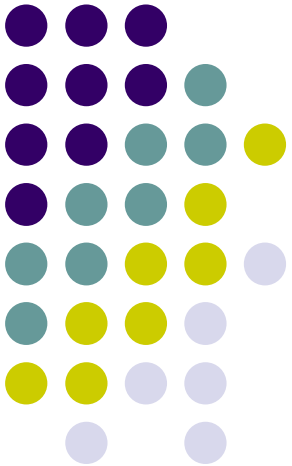
系统分析与设计

Lecture 07 Dynamic modeling (2)

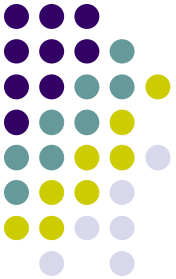


by Dr Y.Yang
Associate Professor
School of Computer Science & Technology
Soochow University
yyang@suda.edu.cn

Review

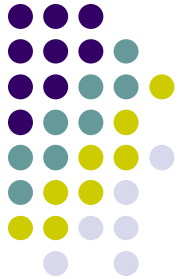
- 系统中传递的消息类型和表示方法
- **UML**中时序图的描述方法
- 同步消息和异步消息的定义和描述方法
- **UML**中协作图的描述方法
- 时序图和协作图的区别

Today's Topics



- 了解系统中工作流程和对象状态变化的描述方式
- 了解引起对象状态迁移的事件的描述方法
- 掌握UML绘制状态图的方法和步骤
- 掌握并发状态图的描述方法
- 掌握UML绘制活动图的方法和步骤
- 掌握时序图、协作图、状态图和活动图建立动态行为模型的方法和步骤

introduction



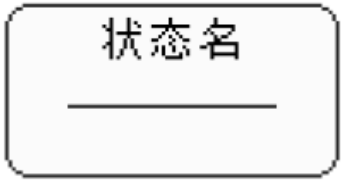
- **UML**中，采用时序图、协作图、状态图和活动图来建立动态行为模型。
- 状态图可以表现一个对象在生存期的行为、所经历的状态序列、引起状态转移的事件以及因状态转移而引起的活动；
- 活动图则用于描述系统中一个活动到另一个活动的控制流、活动的序列、工作的流程和并发的处理行为

状态图(State Diagram)

- 状态图(State Diagram)用来描述一个特定对象的所有可能状态及其引起状态转移的事件（如收到消息、时间已到、报错、条件为真等）。

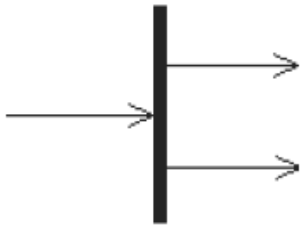
- 大多数面向对象技术都用状态图表示单个对象在其生命周期中的行为。一个状态图包括一系列的状态以及状态之间的转移。

状态图中包含的各种元素

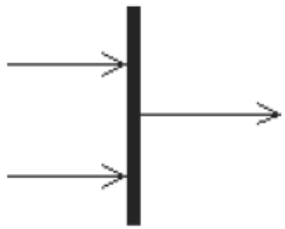


状态

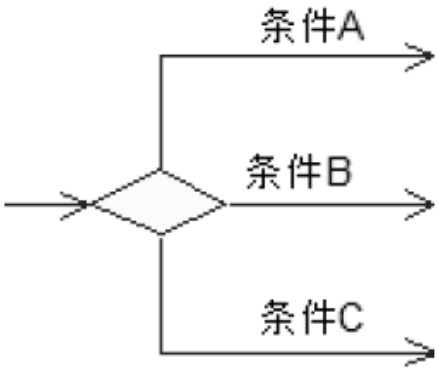
转移



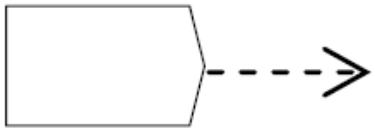
并发分裂



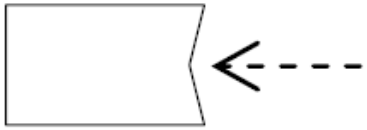
并发接合



条件判定



发出信号

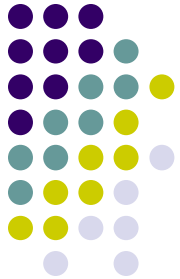


接收信号

起始状态

终止状态

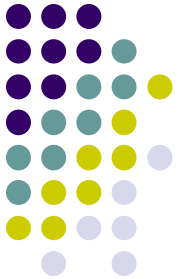
对象的状态图符



- **状态**　所有对象都具有状态，状态是对象执行了一系列活动的结果。当某个事件发生后，对象的状态将发生变化。

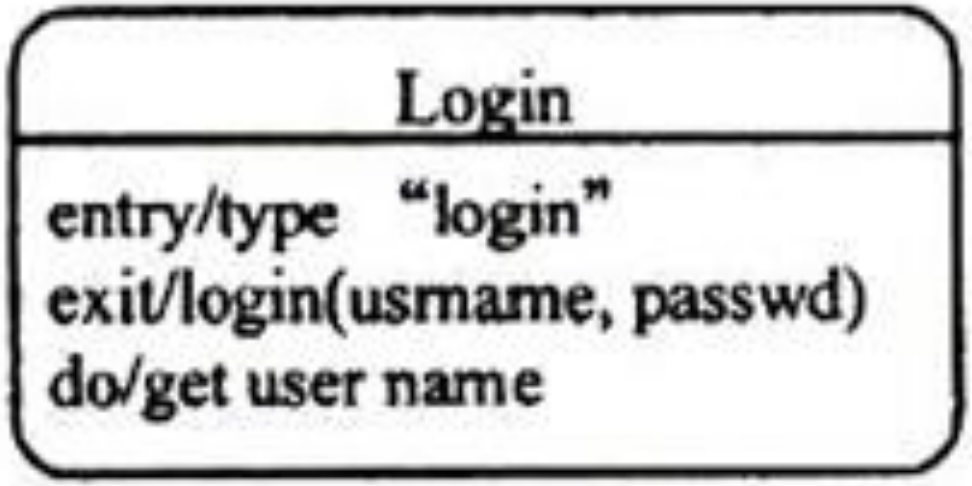
- 状态图中定义的状态有：**初态、终态、中间状态、复合状态**。其中，初态是状态图的起始点，而终态则是状态图的终点。一个状态图只能有一个初态，而终态则可以有多个。

Continued…

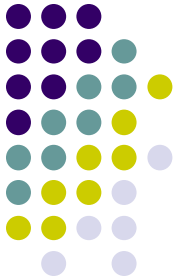


- 一个状态的图符由状态名、状态变量和内部活动3个部分组成。
  - 状态名：一个状态图中，状态名唯一；可以由匿名状态名
  - 状态变量：是状态图所描绘的类的属性（任选项）
  - 活动：列出了该状态时要执行的事件和动作（任选项），活动有3个标准事件：
    - **entry**：指明进入状态时的特定动作
    - **exit**：指明退出状态时的特定动作
    - **Do**：指明在该状态中执行的动作

Example：带活动域的状态

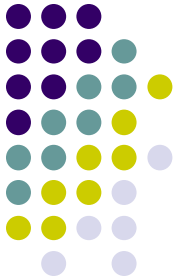


状态的迁移（转移）



- 一个对象从一个状态改变成另一个状态称为状态迁移
- 状态的迁移用连接这两个状态的实箭线表示。在状态的迁移箭线上写上引起该迁移的事件、条件和动作。
- 当事件发生时，动作发生，执行从一个状态到另一个状态的迁移，称为迁移点火或状态触发。

引起状态迁移的原因



1. 出现某一事件
2. 自动触发

迁移的完整组成

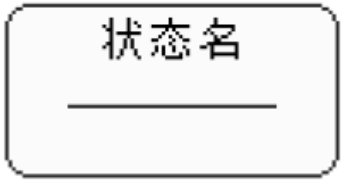
- 源状态（**source state**）：受转移影响的状态。
- 事件触发器（**event trigger**）：能够引起状态转移的事件。
- 监护条件（**guard condition**）：一个布尔表达式，当因事件触发器的接收而触发转移时，对这个布尔表达式求值，
- 效应（**effect**）：效应即执行的动作，是在转移激活时所执行的。
- 目标状态（**target state**）：即在转移完成后的活动状态。

状态图的描述

描述一个状态图的图符元素除了状态图符和迁移图符外，还有起始状态、结束状态、条件判定、发出信号、接收信号和并发等各种图符元素。

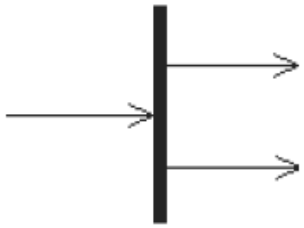
- 起始状态：代表状态图的起点，本身无状态；用实心圆表示；
- 结束状态：代表状态图的最后状态，本身无状态。用一个圆中套一个实心圆表示；
- 条件判定：是一个转折点，用空心菱形表示；
- 并发状态： 分为分劈和接合两种图符。用一个粗短实线表示，称为并发杆；
- 信号图符

状态图中包含的各种元素

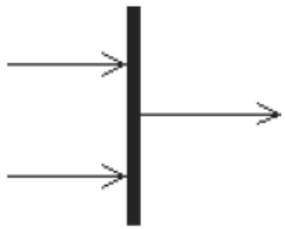


状态

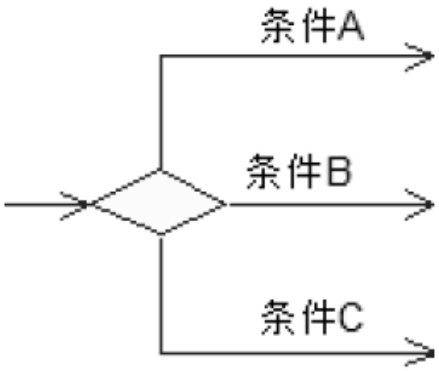
转移



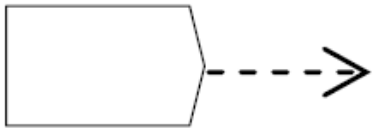
并发分裂



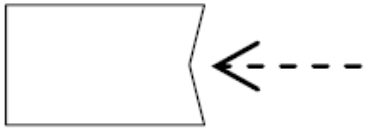
并发接合



条件判定



发出信号

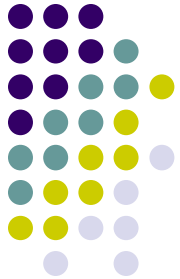


接收信号

起始状态

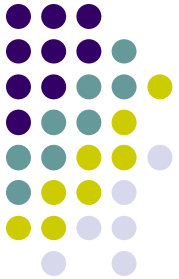
终止状态

如何寻找和定义对象的状态



- 广义上讲，对象属性的任何不同值的组合就是对象的一个状态，全部状态的集合描述了一个对象的状态空间。
- 对于确定对象的状态有重要意义的属性就是状态属性。状态属性的特征是：
  - 一般具有少量的可选值
  - 属性值的转换有一定的限制
- 在建立状态模型时，要正确地找到一个对象的全部状态属性，并根据它们的值划分对象的状态

Continued…



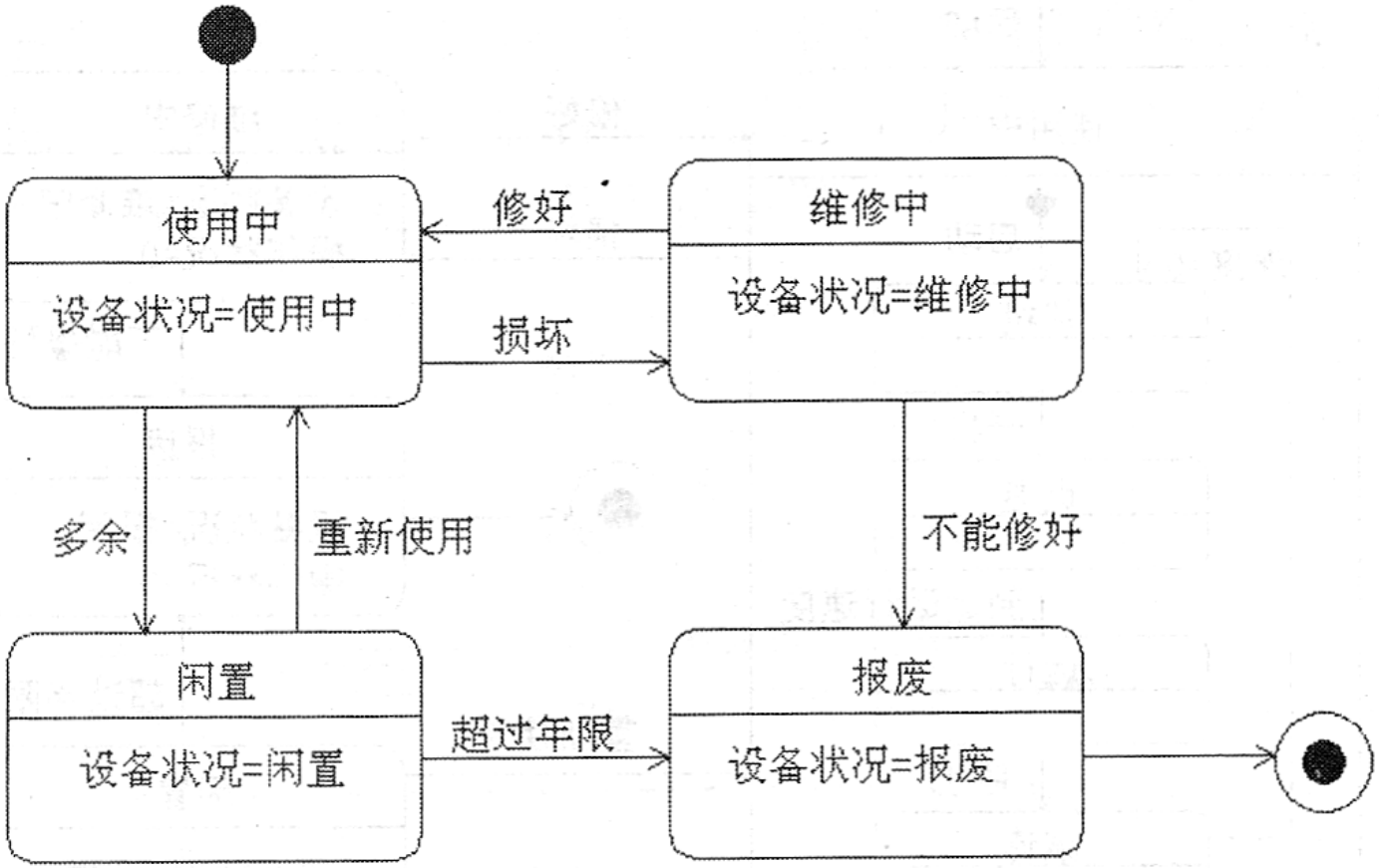
为一个对象的行为建模时应该注意：

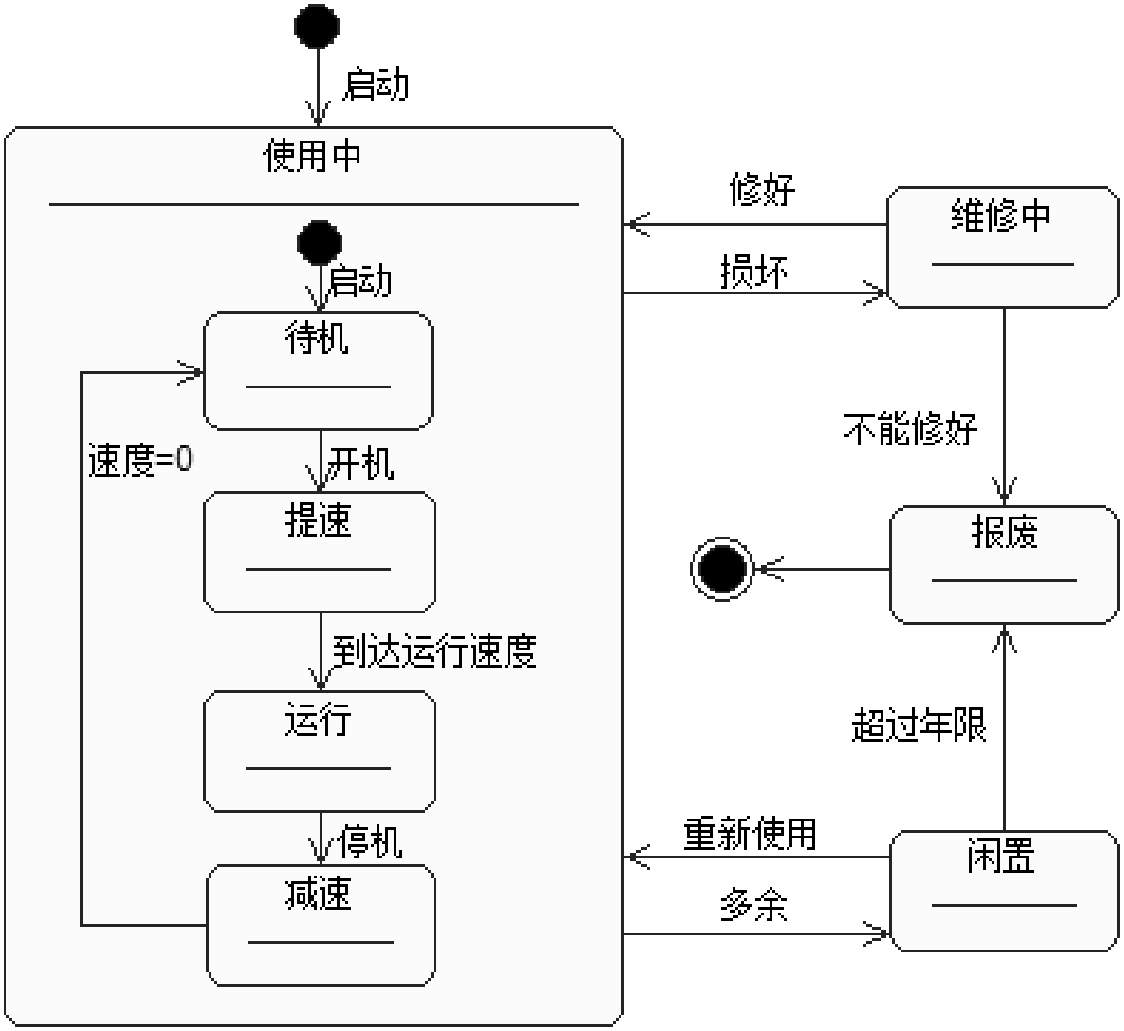
- 对象生存期中状态的数量有限
- 每个状态持续时间也有限
- 以下情况可以触发状态迁移
  - 发生某个事件
  - 完成某个活动
  - 某个活动执行

example

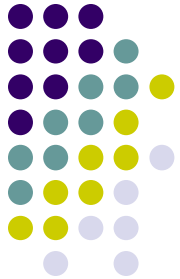
一个设备使用情况的状态图（嵌套状态）

- *一个正常使用的设备如果损坏，将送去维修，修好后可重新投入使用，否则就将报废。*
- *正常使用的设备如果是多余的，将被闲置，闲置的设备随时可重新使用，但闲置超过一定年限后将被报废。*
- *正常使用的设备在启动后将从待机然后提速至正常运行，减速后回到待机。*


使用中
设备状况=使用中
修好
维修中
设备状况=维修中
损坏
多余
重新使用
不能修好
闲置
设备状况=闲置
超过年限
报废
设备状况=报废

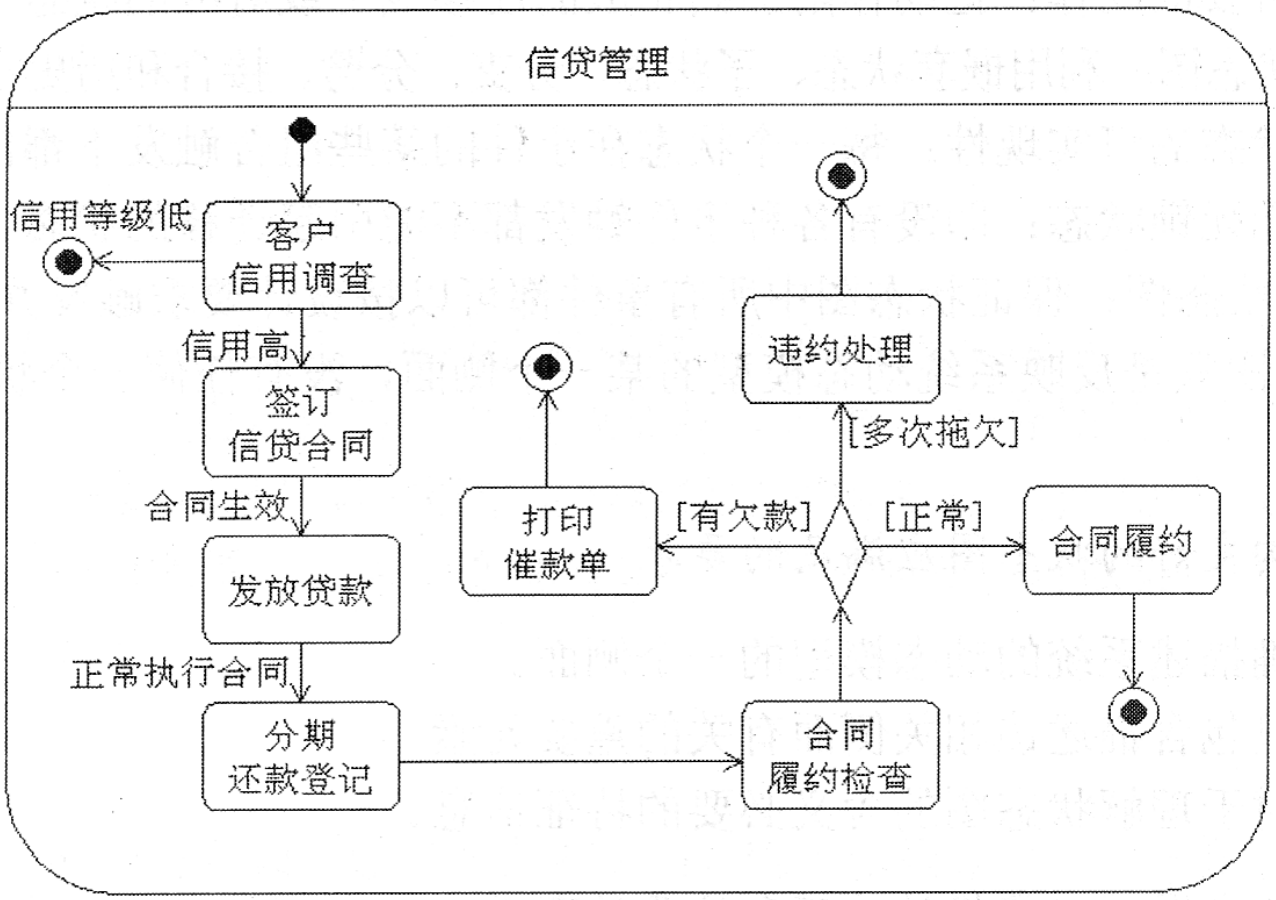

启动
使用中
修好
维修中
损坏
启动
待机
速度=0
开机
提速
不能修好
报废
到达运行速度
运行
超过年限
停机
重新使用
闲置
减速
多余

嵌套状态和简单状态

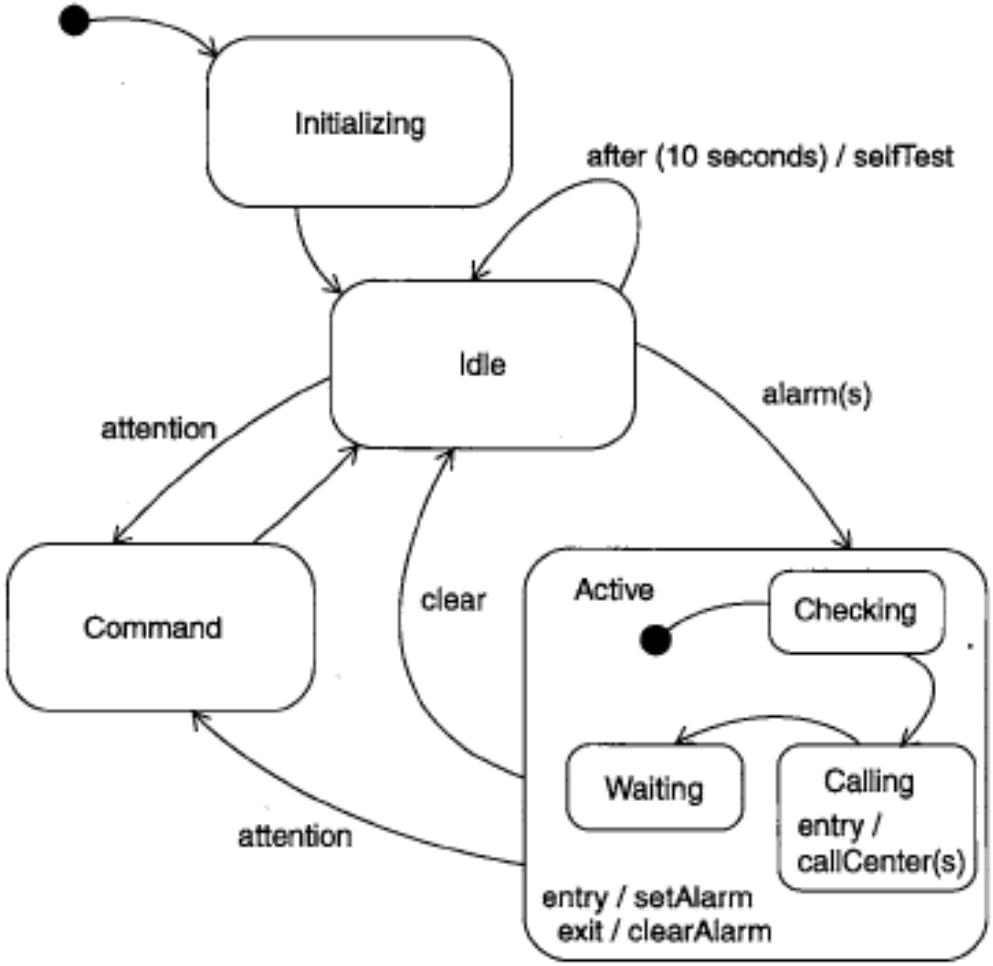


- 在一个状态图符的活动区还有一个或多个状态图称为嵌套状态，被嵌套的状态称为子状态。一个子状态还可以有嵌套状态
- 一个不含内嵌套的状态，称为简单状态。简单状态对应一个动作。
- 嵌套状态中每个被嵌套的状态图都对应于该嵌套状态内正在进行的一个活动。

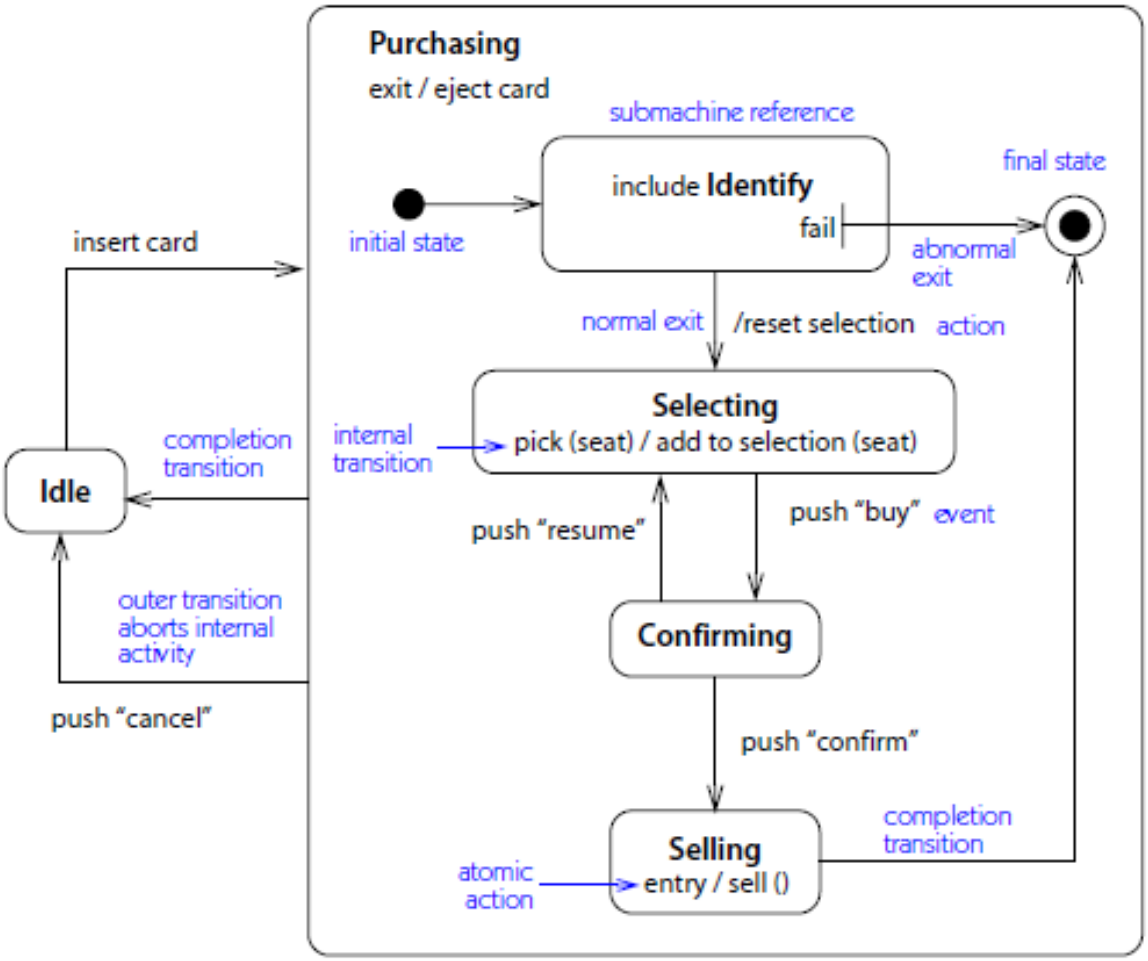
“信贷管理”子系统用例图的状态图



家庭安全系统中控制器的状态图

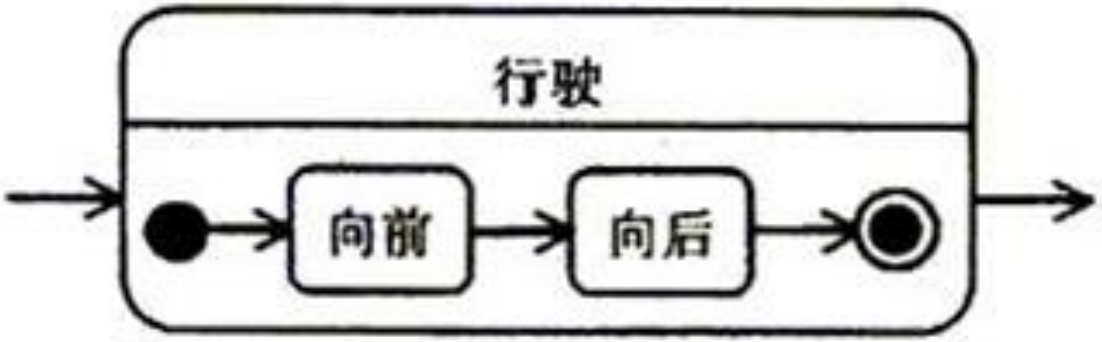


More example：
the control for a ticket-selling machine



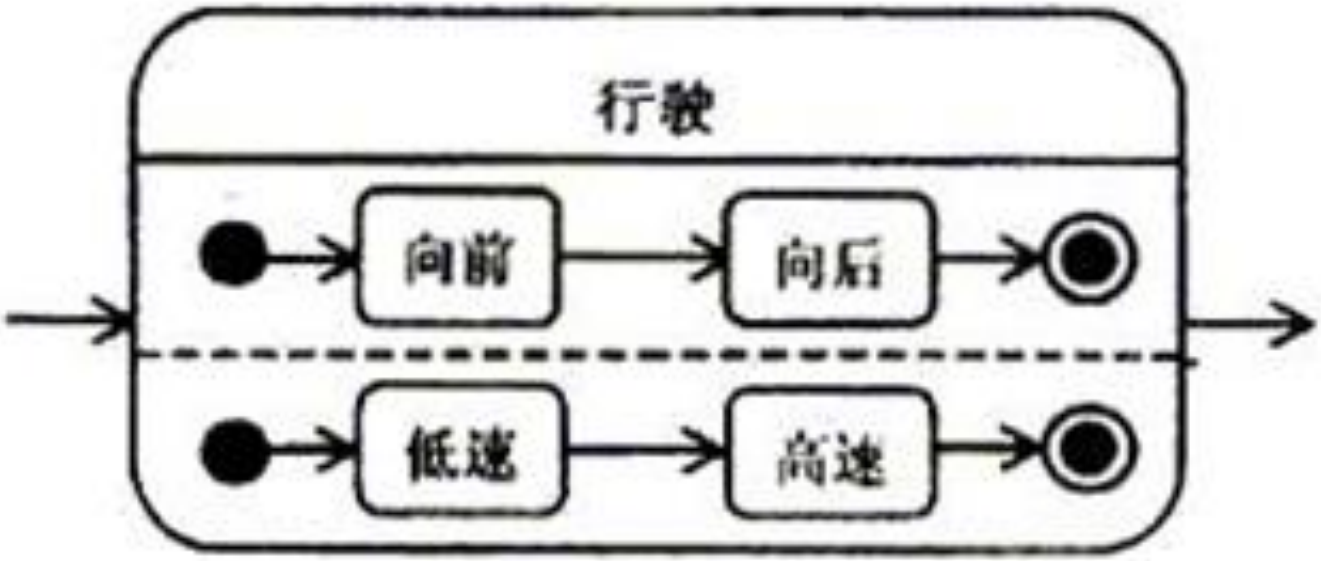
子状态的关系

- 子状态之间有"或关系"和"与关系"两种关系。或关系说明在某一时刻仅可到达一个子状态。例如，一个处于行驶状态的汽车，在"行驶"这个复合状态中有向前和向后两个不同的子状态，在某一时刻汽车要么向前，要么向后。

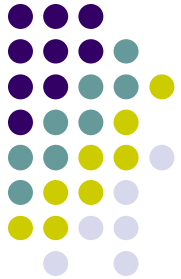


Continued…

- 与关系说明复合状态中在某一时刻可同时到达多个子状态(称为并发子状态)。具有并发子状态的状态图称为并发状态图。

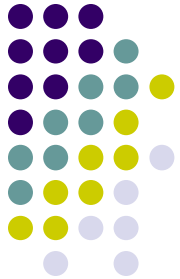


顺序状态



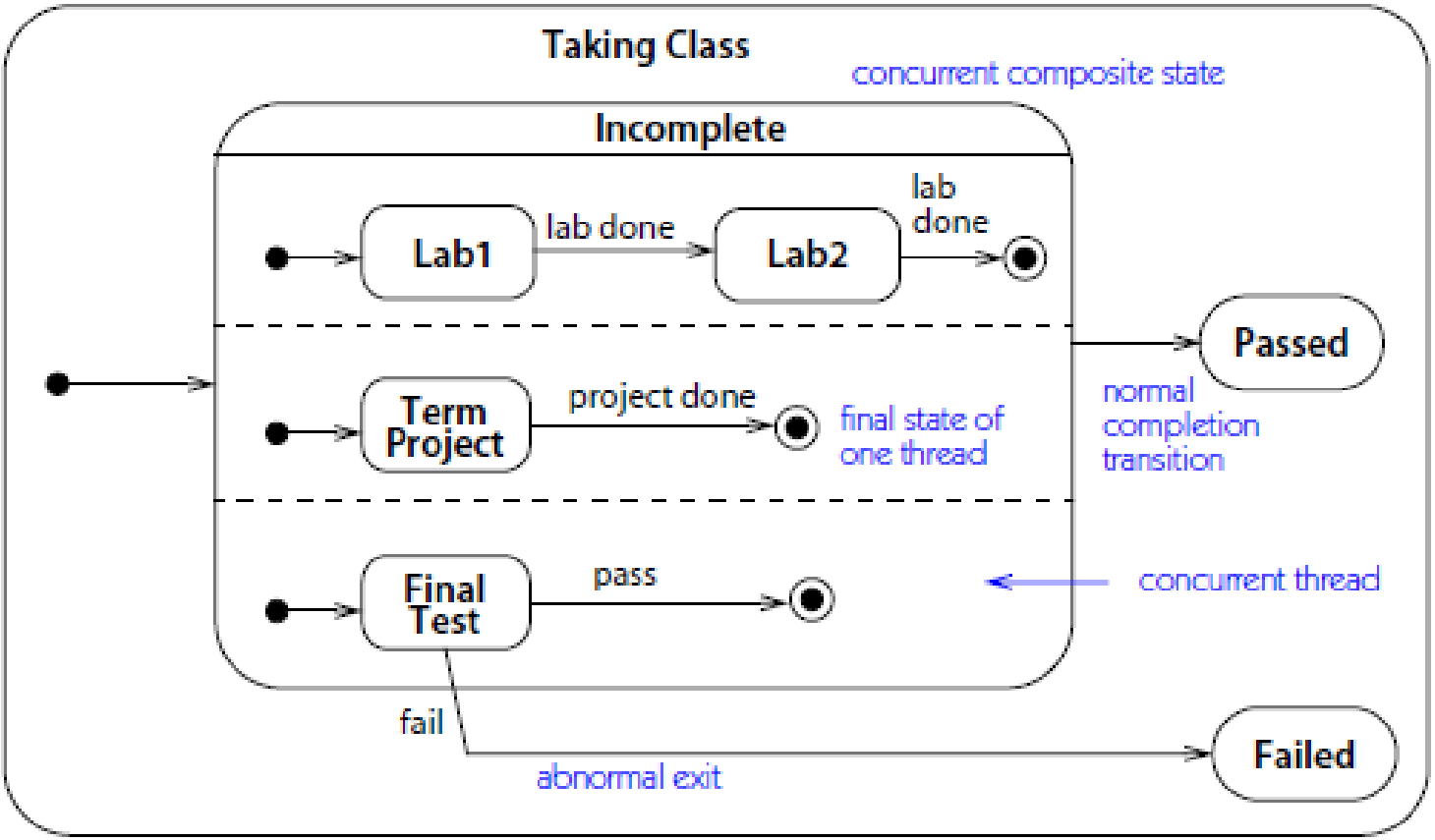
- 顺序状态也称为不相交状态，表明状态图中的状态并没有并发迁移现象，状态之间的迁移是串行的，即一个接一个地顺序迁移。

并发状态与同步

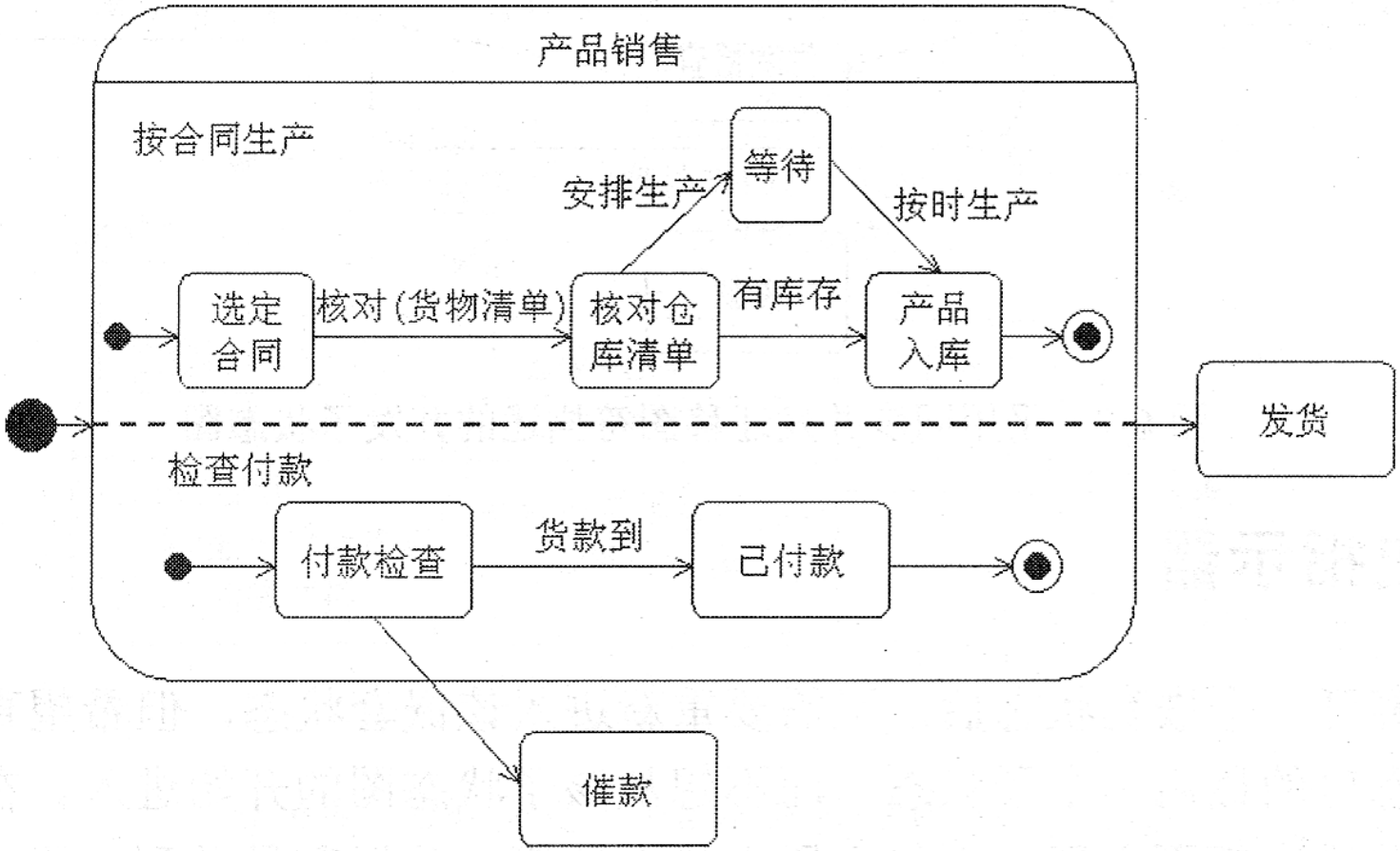


- 一个状态也可以有多个并发的子状态，并发子状态之间可以用虚线分隔，用虚线分隔的每一个区域表示一个并发的子状态，它有一个名字，并有一个内部的状态图。

example: the concurrent decomposition of taking a university class



Example：
一个按销售合同生产和销售产品的并发子状态图

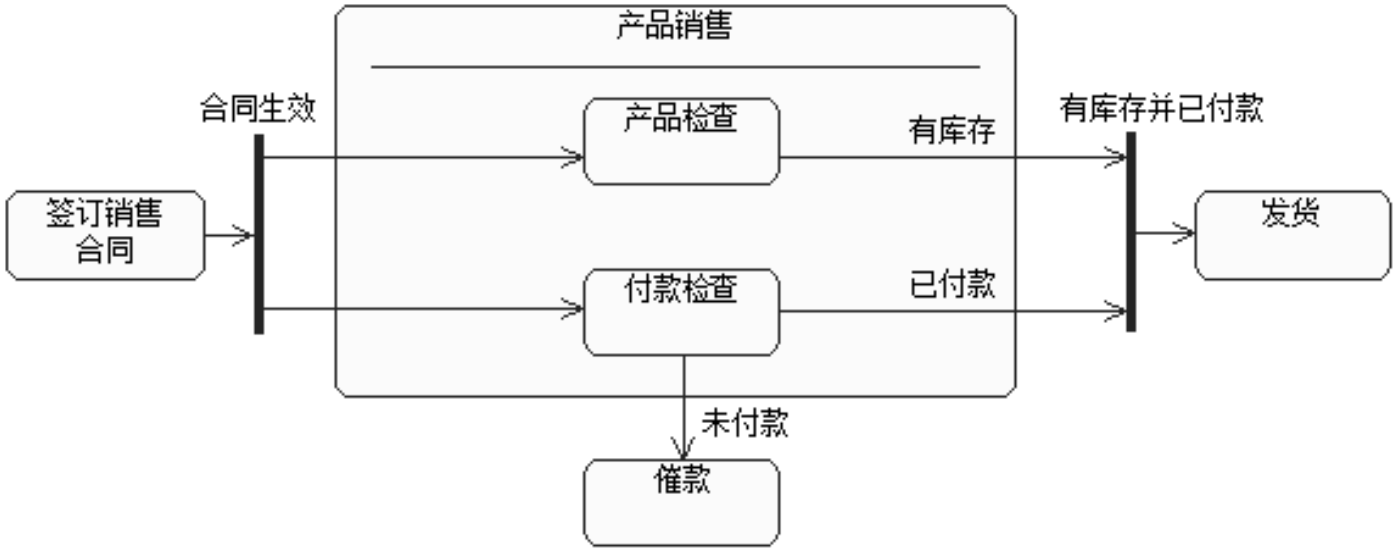


explanation

- 在按销售合同生产和销售产品的企业中，有两个过程是同时并发执行的：一个过程是检查仓库是否有合同要求的产品，另一个过程是检查客户是否已付款。只有两个检查都成功，才能发货，缺一不可。
- 在第一个过程中，首先检查合同，如果仓库中有合同清单所要求的货物名单和相应的数量，产品已经备齐，等待发货；
- 如果仓库中没有合同清单所要求的货物名单或相应的数量，则等待一段时间，组织安排生产，生产完毕产品入库，等待发货。
- 另一个过程是检查付款单，如果客户按合同约定汇来货款，说明该合同已付款，可以发货。
- 如果超过合同期限没有汇款，应向客户发催款通知，不能发货。

同步并发杆

- 还可以用复合迁移的同步并发迁移图符来表示并发子状态
- 使用同步杆在并发活动中起到同步作用
- 同步杆旁边标有迁移发生的条件



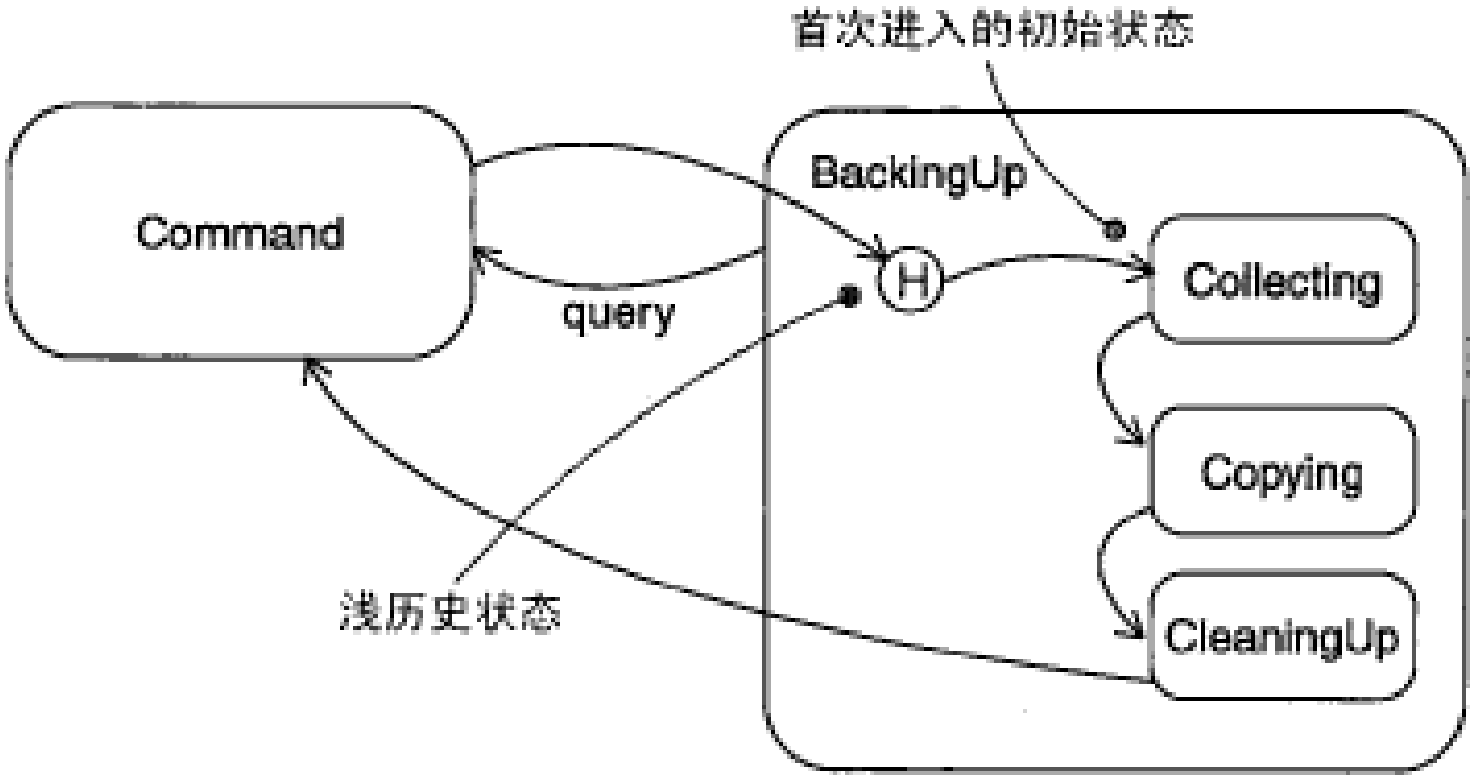
状态图中的历史指示器

- 有时，当离开一个嵌套状态后，又需要重新进入该状态，但希望直接进入到上次离开该嵌套状态时的最后一个子状态，而不想从该子状态图的开始进入，可以用历史指示器来表示这种情况。

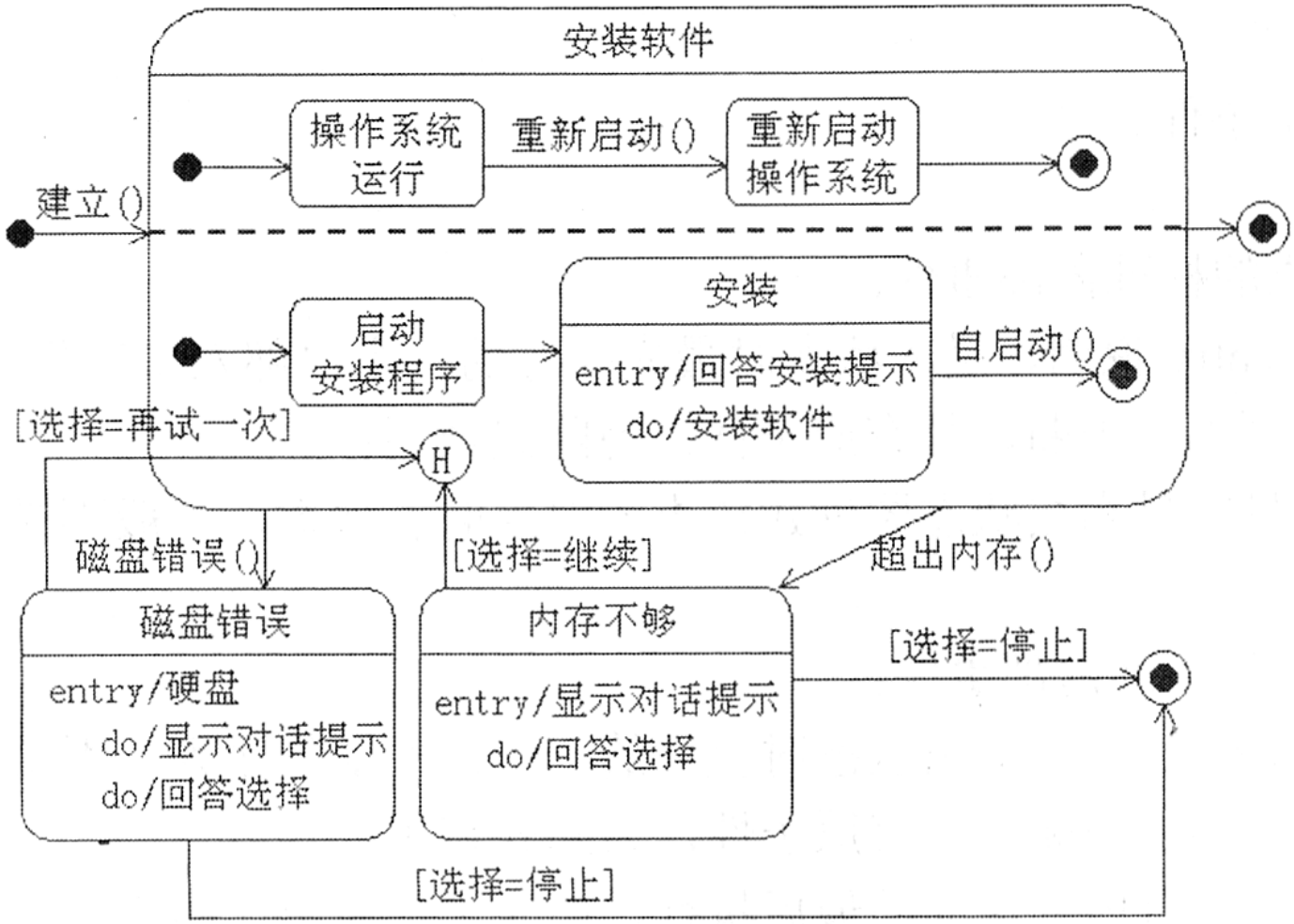
- 历史指示器用来记录状态图内部的历史状态，用里面标有**H (history)**的圆圈表示。

- 历史计数器是一个伪状态。

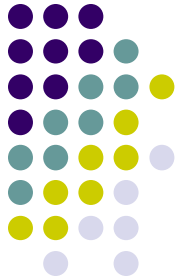
example


首次进入的初始状态
Command
BackingUp
H
query
Collecting
Copying
CleaningUp
浅历史状态

带有历史指示器的软件安装过程状态图



explanation



- 状态图描述了一个软件的安装过程。一个“建立（）”迁移触发使系统进入安装软件状态。在软件安装状态中嵌套有两个并发的子状态图：一个是正在运行的操作系统，另一个是运行安装软件的程序。只有在操作系统正在运行的情况下，才能启动安装软件的程序，因此它们是并发执行。

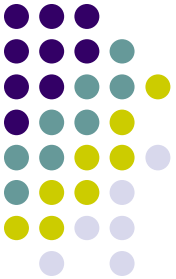
- 当软件安装完毕时，操作系统和安装软件的程序都要重新启动，安装的软件才能开始工作。操作系统运行子状态图的功能很清楚，这里重点讨论安装程序运行子状态图的状态迁移过程。

Continued…

- 在安装程序运行子状态图中有一个内有字母 **H** 的圆圈，表明该子状态图处于历史状态指示器的作用之下，当有迁移触发历史指示器时，应恢复到离开该状态图迁移时的前一个状态。

- 在图所示安装程序运行子状态图中，首先进入“启动安装程序”夕状态，再自动进入“安装”软件状态。在“安装”软件状态中，操作人员按照安装程序的提示进行软件安装。如果在安装过程中出现“磁盘错误”或“超出内存”错误时，安装程序暂时停止安装进程，在屏幕上提示出错信息并要求操作者进行选择。

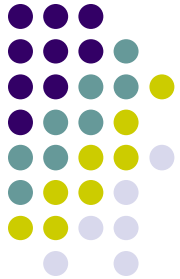
Continued…



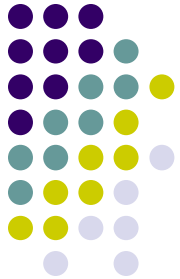
- 出现“磁盘错误”时，操作者可以选择“再试一次”或“终止安装”迁移；出现“超出内存”错误时，操作者可以选择“继续安装”或“终止安装”迁移。如果选择“终止安装”迁移，则安装程序停止运行，退出安装程序，回到操作系统。

- 如果选择“再试一次”或“继续安装”迁移时，该迁移触发历史指示器。此时，并不是回到安装程序运行子状态图中的起始状态，而是返回到安装程序运行子状态图中发生迁移时的前一个状态（历史状态），即“安装”软件状态，继续进行软件安装。

状态迁移的进一步讨论



- 状态迁移表示当一个特定的事件发生或某些条件满足时，一个源状态下的对象将完成一些特定的动作，成为状态触发，使对象从源状态迁移到一个新的目标状态。
  - 活动状态：迁移发生时，该迁移进入的状态为活动状态，它将执行相应的动作。
  - 非活动状态：迁移发生时，该迁移离开的状态为非活动状态



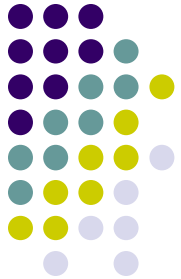
描述状态迁移的形式化语法格式：

事件[条件]/动作表达式 发送子句

说明：

- 事件：指已发生并可能引发某种活动的一件事
- [条件]：关系或逻辑表达式
- 动作表达式：一个触发状态迁移时可执行的过程表达式
- 发送子句：动作的一个特例，说明调用的事件名（操作）是哪个对象的

事件



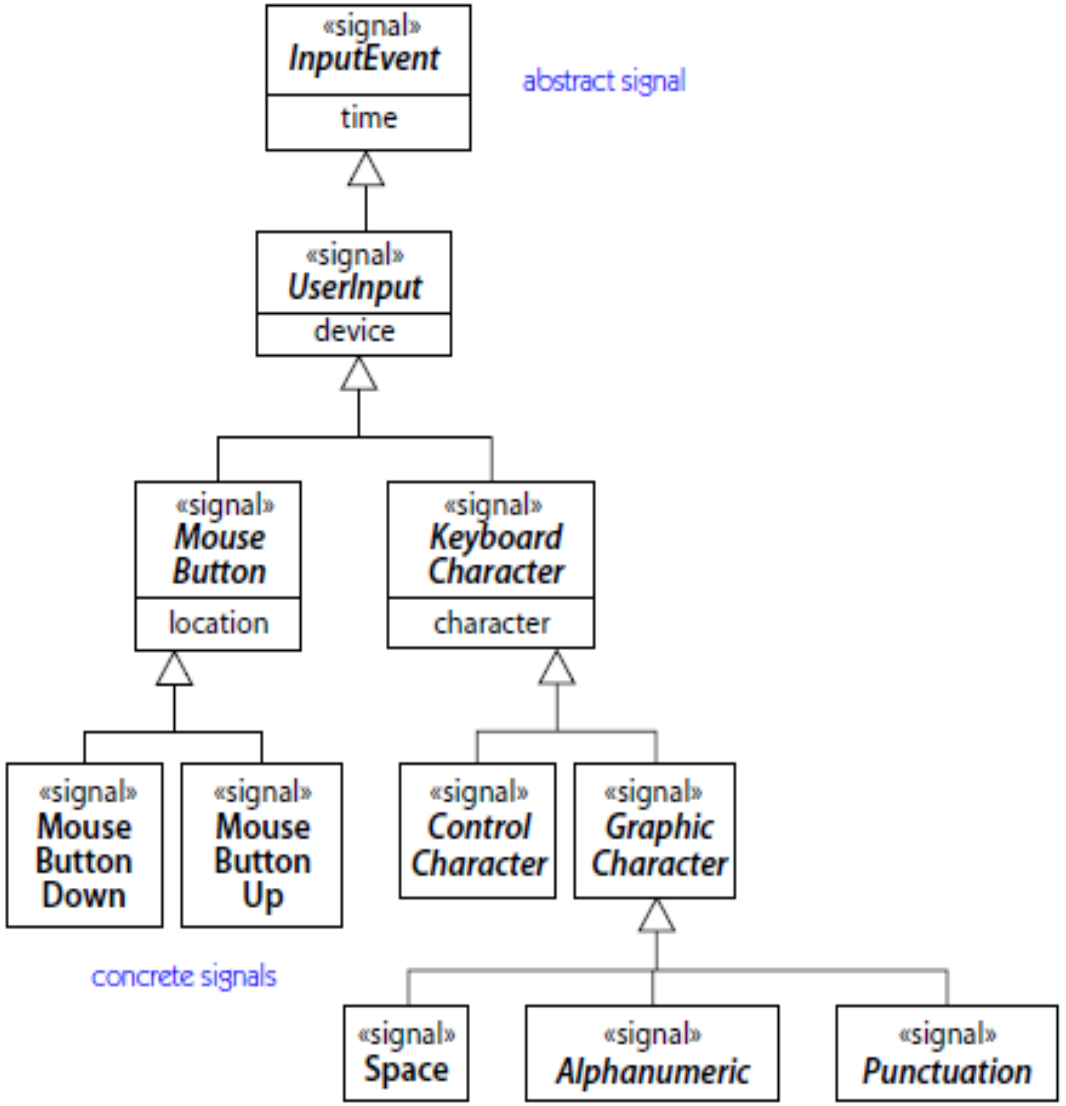
- 事件指已发生并可能引发某种活动的一件事
- 事件名（参数表）
- 事件的种类：

Event Type	*Description*	*Syntax*
call event	Receipt of an explicit synchronous request among objects that waits for a response	op (a:T)
change event	A change in value of a Boolean expression	**when** (exp)
signal event	Receipt of an explicit, named, asynchronous communication among objects	sname (a:T)
time event	The arrival of an absolute time or the passage of a relative amount of time	**after** (time)

1. 状态内部事件
   - 入口事件entry
   - 出口事件exit
   - do事件
   - include事件
   - 自定义内部事件

2. 消息
   - 调用事件
   - 信号事件

3. 时间事件
   - After事件
   - Defer事件
   - When事件



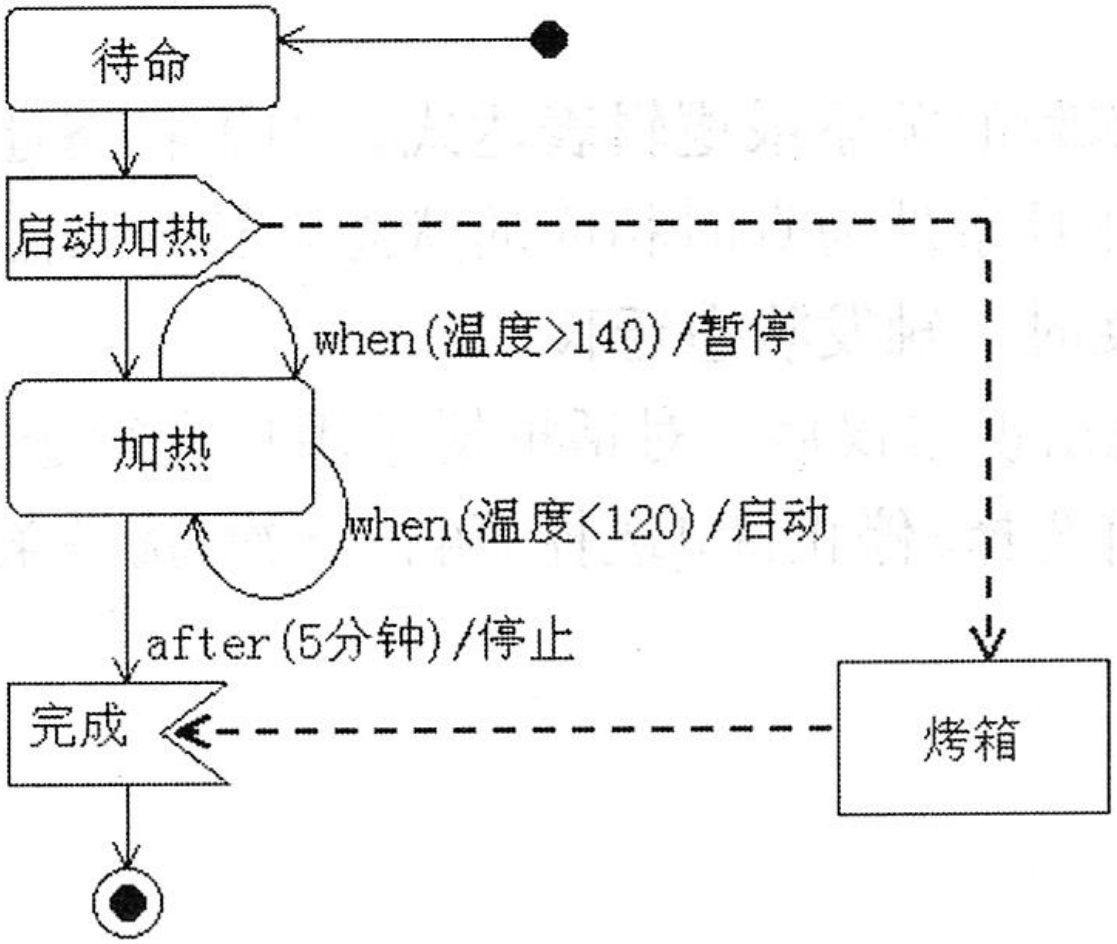
还有出错情况<<error>>等事件

example

- 以下描述一个电烤箱控制系统的状态图

系统接受到启动加热的信号后进入加热状态，如果温度高于140度，暂停加热；如果温度低于120度，重新启动加热。5分钟后加热停止，完成一次烘烤过程。

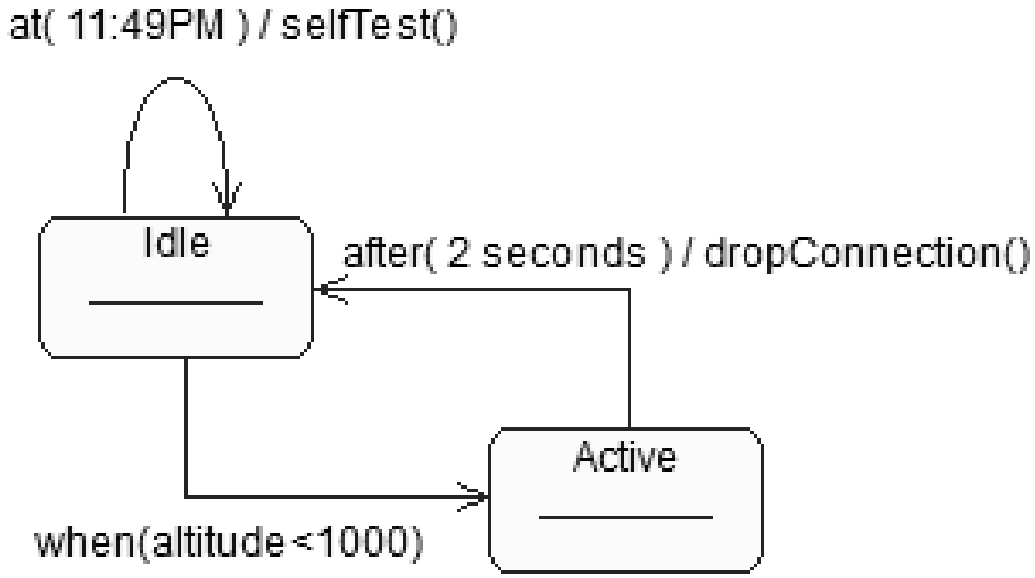
电烤箱控制系统的状态图



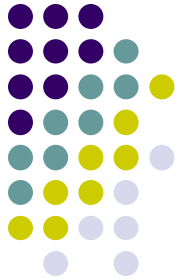
More example

- 如图所示有两个状态：
  “Idle”空闲状态和
  “Active”激活状态。

- 当11:49PM的时候，
  “Idle”状态发生自转移
  进行自检，当“altitude”
  小于1000时状态变为
  “Active”激活状态，
  2s后激活状态断开连接
  回到“Idle”空闲状态。



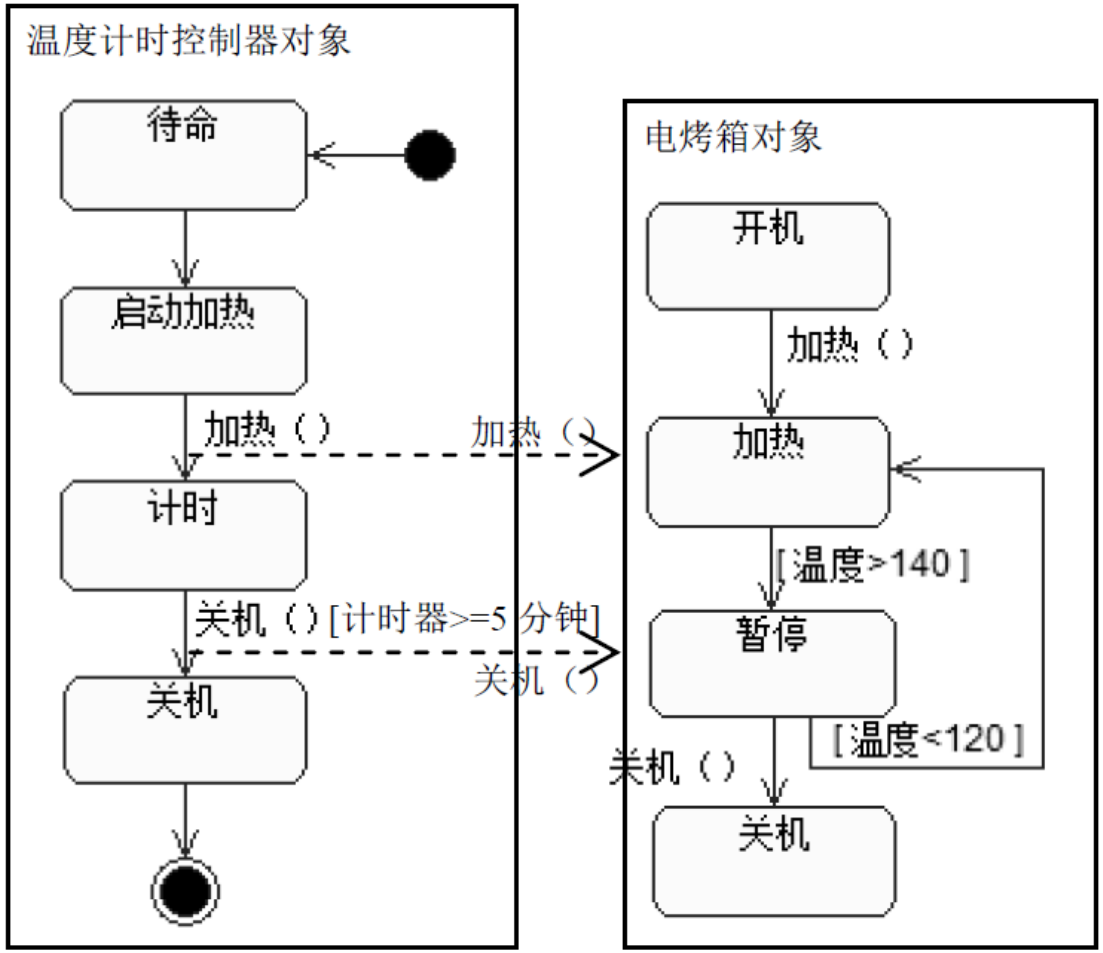
小结：状态迁移的种类



- 自迁移：源状态和目标状态为同一状态的迁移。
- 内部迁移：该迁移在状态内部进行，不引起状态改变。内部迁移由标准内部事件或用户自定义的事件引发。如**do**、**entry**、**exit**、**include**等。
- 自动迁移：在迁移箭线上没有条件和触发事件，当一个状态完成后，自动触发迁移，进入下一个状态
- 复合迁移：由条件判定、并发分劈和一些简单迁移组合而成。

状态图之间发送消息

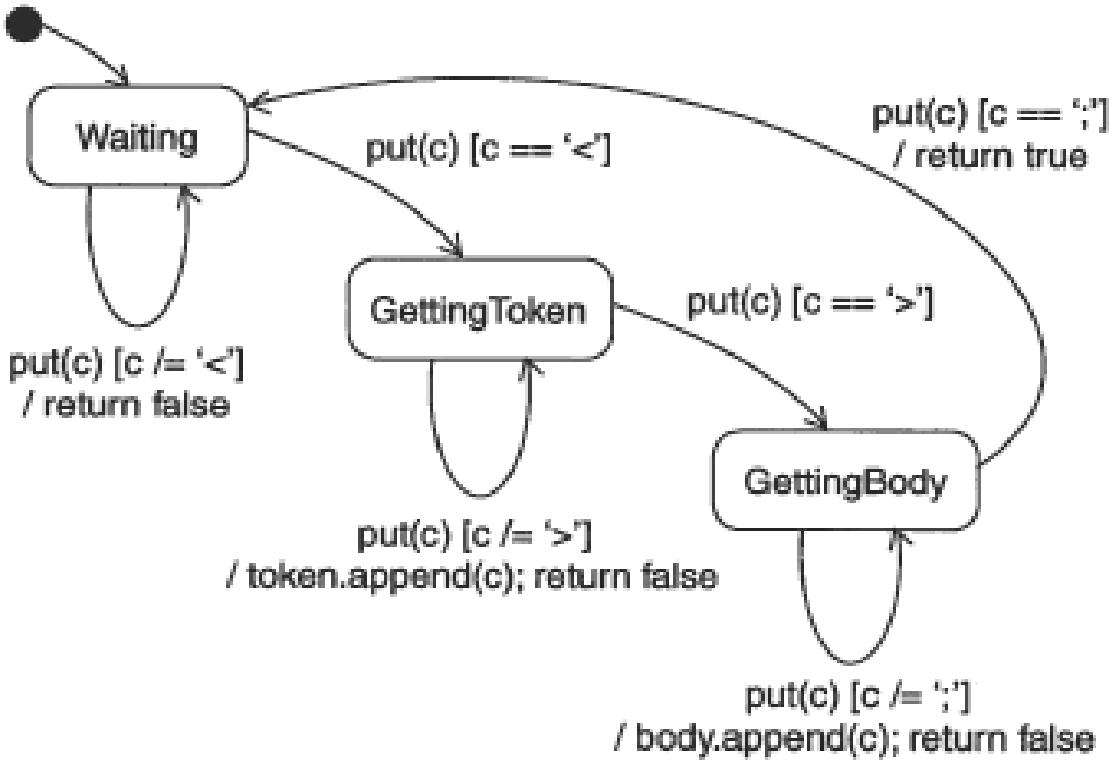
- 状态图可以向其他状态图发送消息。
- 可以用动作（如在发送字句中指明接收者）或状态图之间的虚线箭头表示。



由状态图产生代码（正向工程）

分析与语法相匹配的字符流：

Message: ‘<’ string ‘>’ string ‘;’




```
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;
```

课后思考1

- 如何对一个手机系统的控制器建立一个基本的状态图（仅考虑通话功能）？

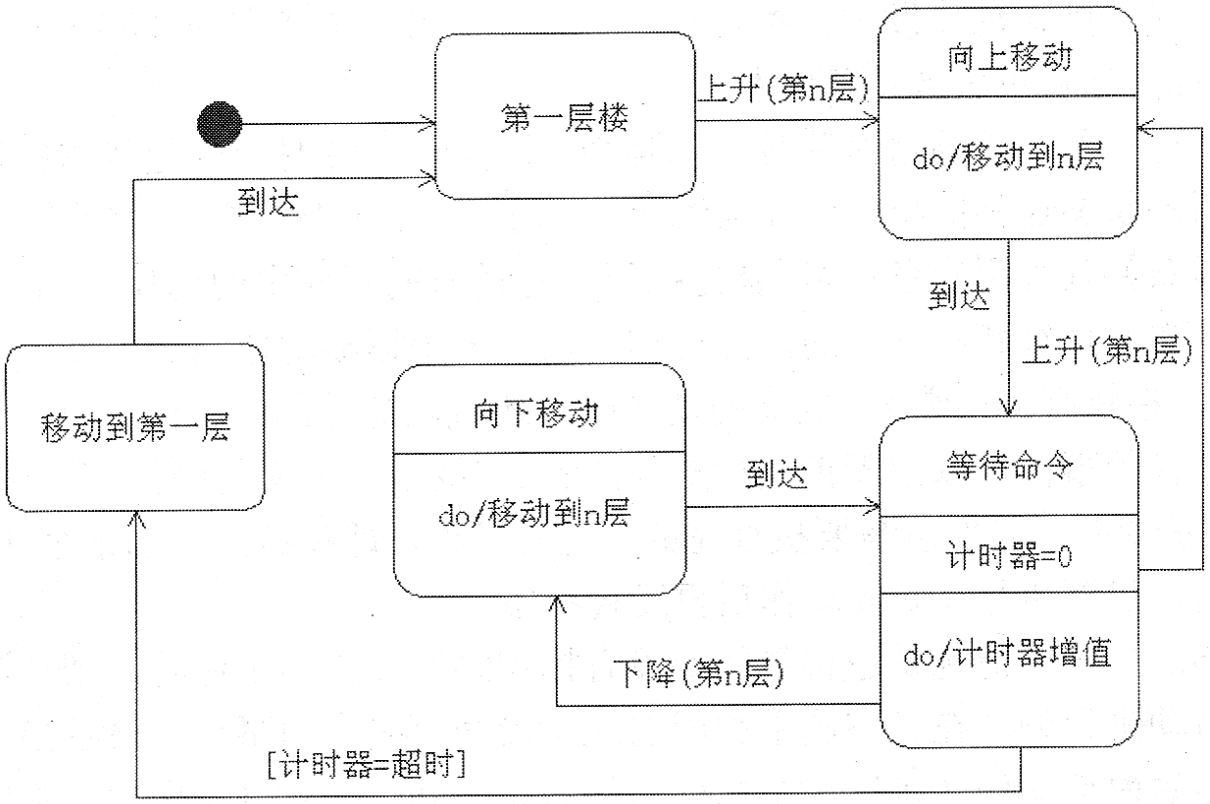
课后思考2

通常情况下，一个无人值守电梯的轿箱通常停放在大楼的第一层。当某楼层有乘客按下按钮，电梯轿箱便会按照指令上升到该楼层接乘客，然后按照乘客的指令升降到指定楼层，到达后的乘客走下电梯。电梯轿箱停在该楼层，等待下一个乘客的按钮指令。

但是，系统对于等待的时间有一定的限制，在时间限制之内又有乘客按下按钮，电梯则重复前面的动作，电梯轿箱仍按照指令上升或下降到指定楼层，到达后，电梯轿箱继续等待下一个乘客的按钮指令。

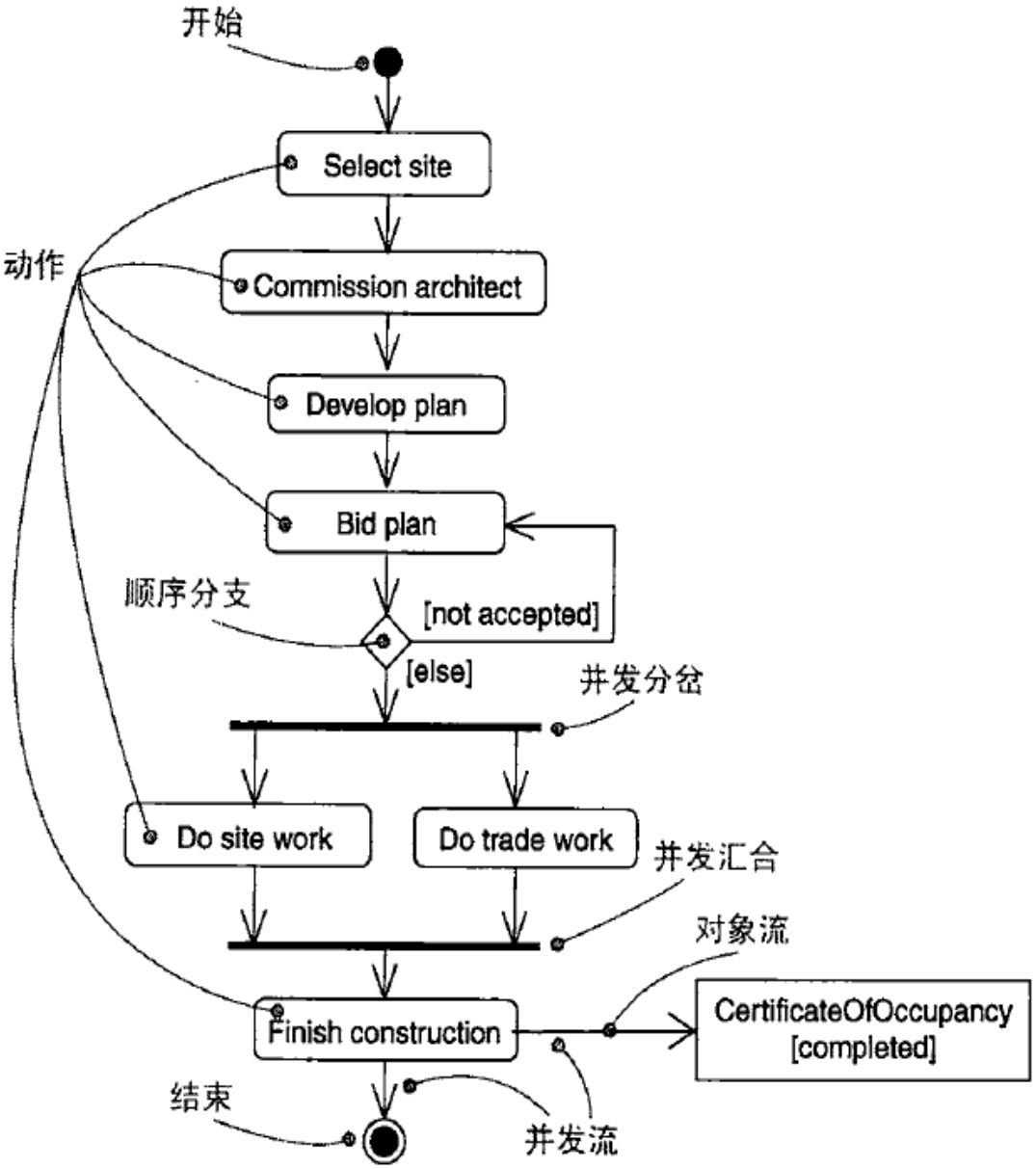
在每次的等待中，如果等待时间超过限制，电梯轿箱会自动返回到大楼的第一层，在那里继续等待乘客。

请建立该电梯轿厢的状态图。

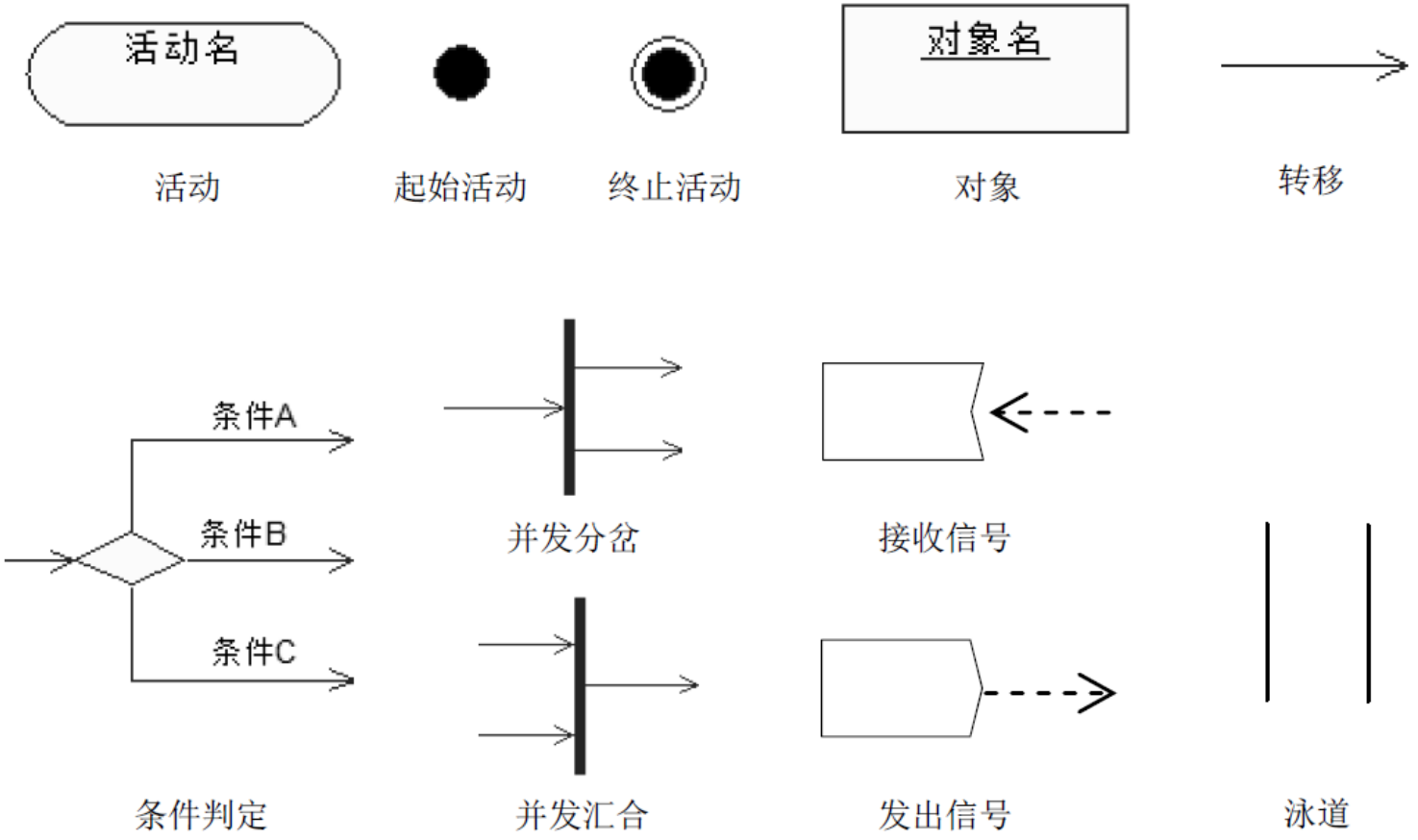

第一层楼
上升(第n层)
向上移动
do/移动到n层
到达
到达
上升(第n层)
移动到第一层
向下移动
do/移动到n层
到达
等待命令
计时器=0
do/计时器增值
下降(第n层)
[计时器=超时]

活动图

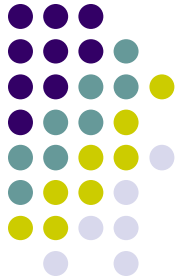
- **UML**中，活动图是系统动态行为建模的图形工具之一，用来表示完成一个操作所需要的活动，或者是一个用例实例（场景）的活动。
- 活动图实际上也是一种流程图，描述的是活动的序列。
- 活动图特别适合于描述动作流和并发处理行为。



活动图包含的元素


活动名
对象名
活动
起始活动
终止活动
对象
转移
条件A
条件B
条件C
并发分岔
接收信号
条件判定
并发汇合
发出信号
泳道

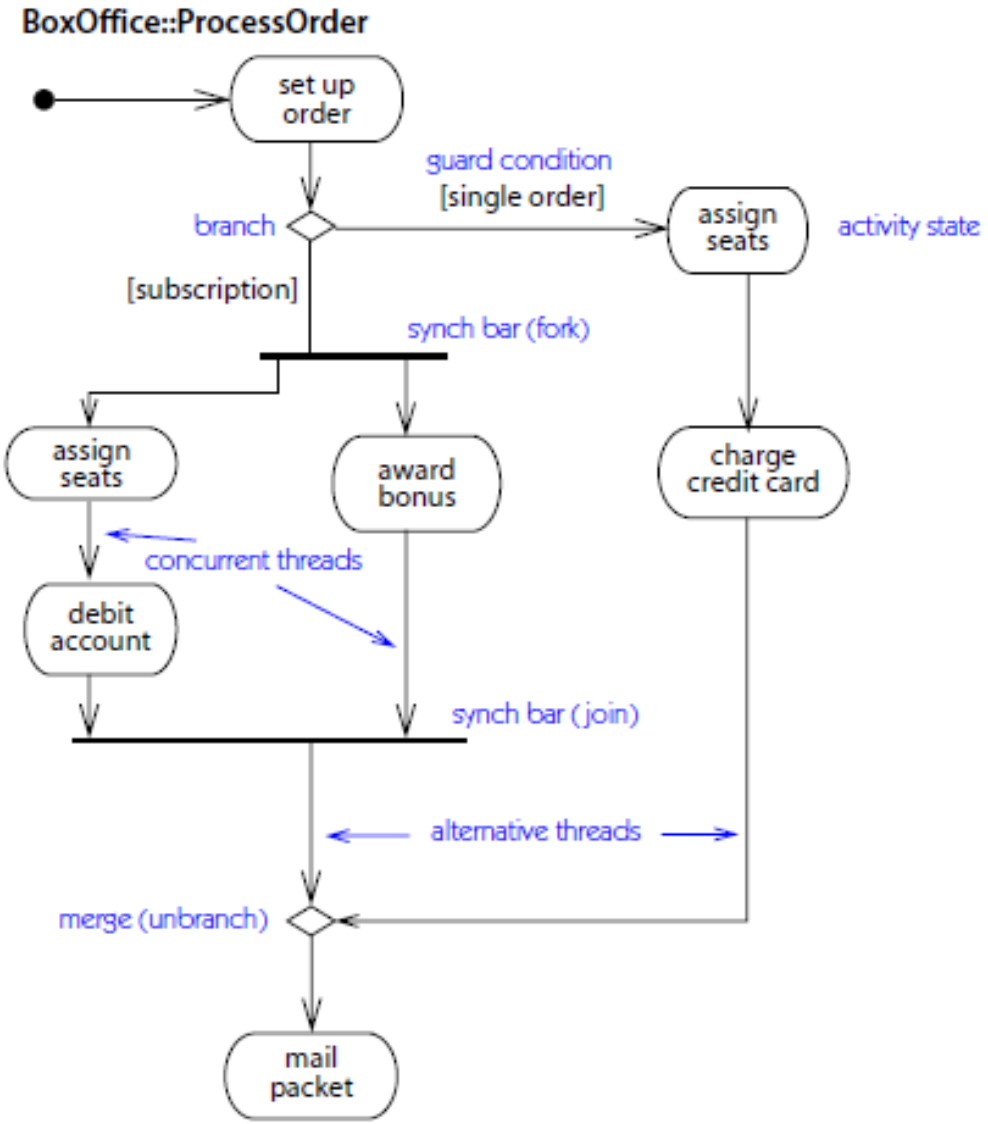
活动图包含的元素



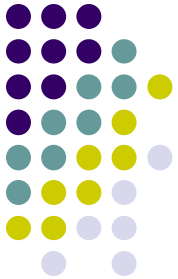
- 活动图展示从活动到活动的流。一个活动是一个状态机中进行的非原子的执行单元。
- 活动的执行最终延申为一些独立动作的执行。
- 活动图一般包括：
  - 动作和活动结点
  - 流
  - （对象值）
  - ……

  以及必要的约束和注解。

example


BoxOffice::ProcessOrder
set up order
guard condition
[single order]
branch
assign seats
activity state
[subscription]
synch bar (fork)
assign seats
award bonus
charge credit card
concurrent threads
debit account
synch bar (join)
alternative threads
merge (unbranch)
mail packet

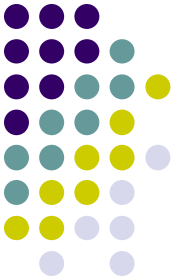
活动图与状态图的区别和联系



- 活动图是由状态图变化而来的,它们各自用于不同的目的。

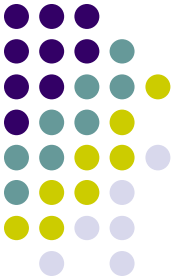
- 活动图与状态图的本质区别：
  - 活动图状态迁移不需要事件触发，活动执行完毕可以直接进入下一个活动状态；
  - 活动置于责任区（泳道）内，责任区将活动按责任目标和组织归属的原则分类。

活动图与状态图相似处



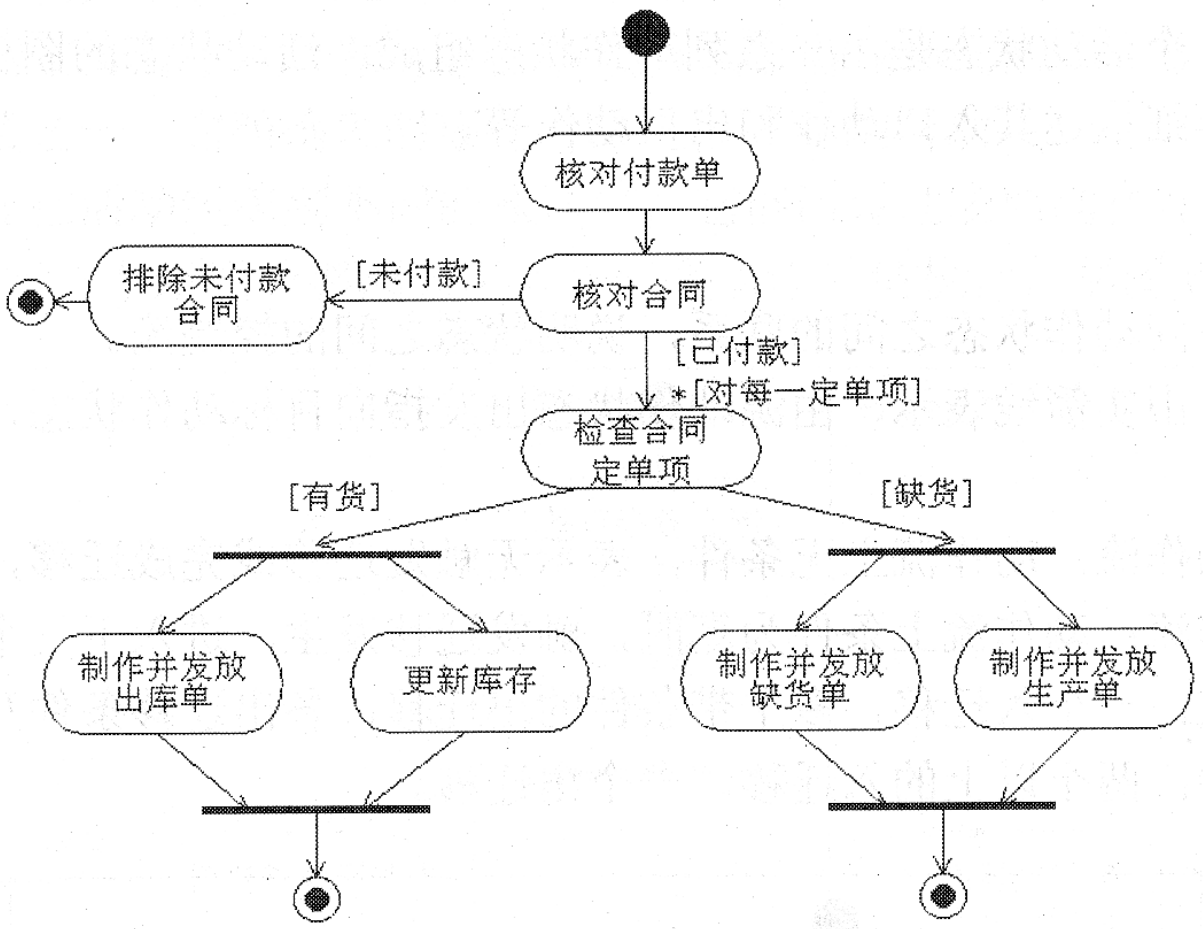
- 都有相同的开始点和结束点
- 活动图中，也可以画条件判定符号（菱形）。
- 活动图中的状态为动作状态，用上下两条平行线段和两侧圆弧构成的图框表示。框内标识动作状态名和描述动作的语句，动作状态之间的迁移用箭头表示，迁移上可以附加条件、发送字句和动作表达式。
- 活动图是状态图的变形，它根据对象状态的变化捕获动作（所完成的工作和活动）和它们的结果，表示各个动作及其之间的关系。

活动图和状态图的不同点



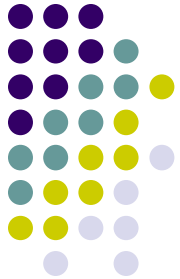
- 与状态图不同的是，活动图中动作状态之间的迁移不是靠事件触发。当动作状态中的活动完成时迁移就被触发。
- 活动图中，事件只能附加到开始点到第一个动作之间的迁移。
- 活动图中，有“泳道”的概念。

例：一个“检查合同”、“核对付款单”并“发放出库单”的活动图



说明

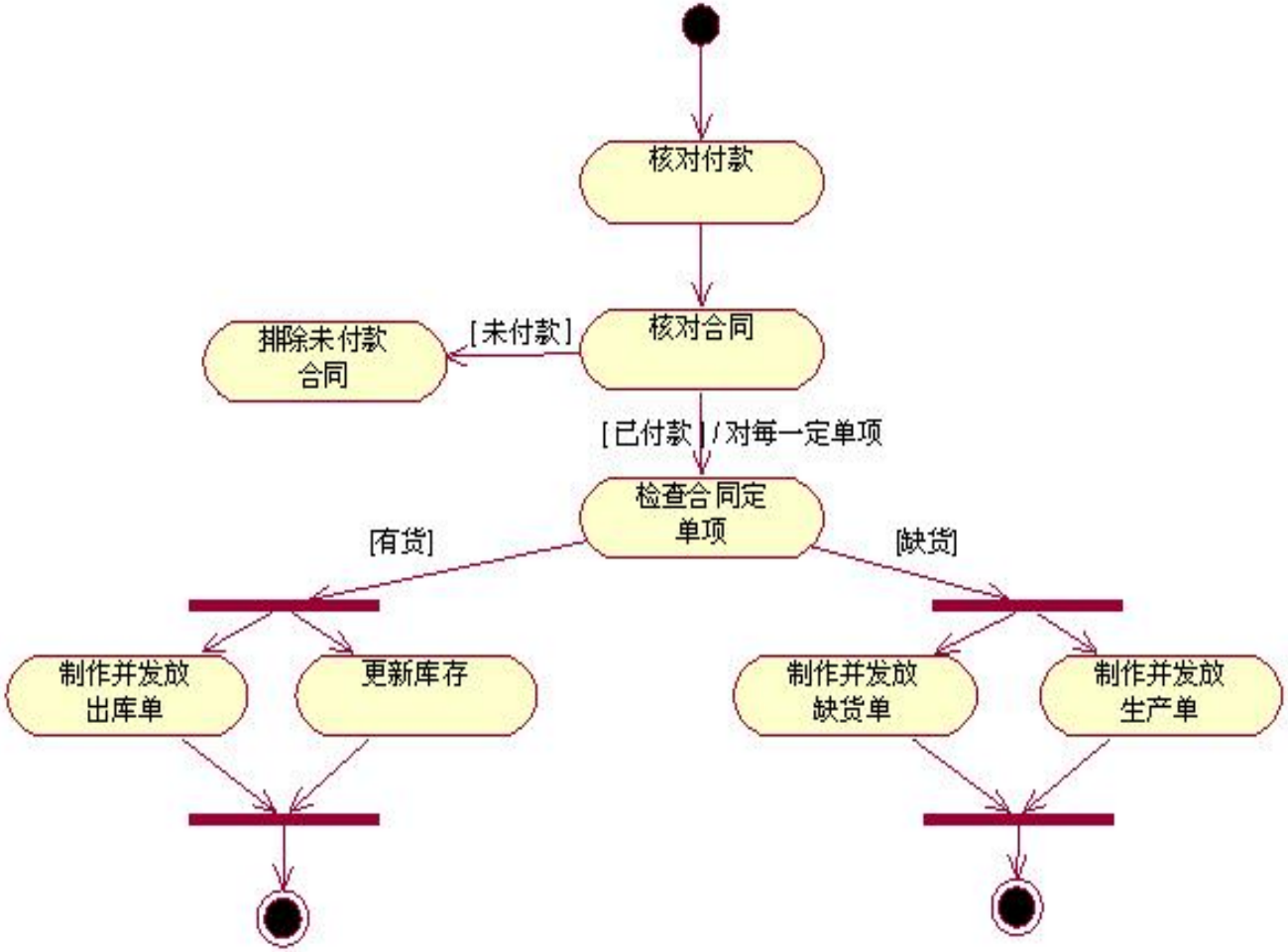
- 在如图所示活动图中，首先对财务系统传送来的付款单与销售合同进行核对检查，排除未付款的合同。对已付款的合同，将合同定单与库存清单逐项进行检查核对。检查核对有两种结果：有货或缺货。如果仓库有合同所需货物，制作并发放出库单，同时更新库存、发送货物并在相关合同上标注合同履约标志；如果仓库没有合同所需货物，则制作并发放缺货单，同时制作并向生产调度部门发放生产单。



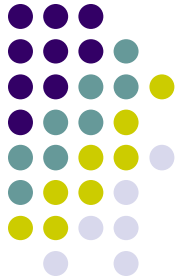
- 通过以上图例可以看出，活动图和状态图非常类似。它有与状态图相同的开始点和结束点，活动图中的状态称为动作状态，由上下两条平行线段和两侧圆弧构成的图框表示。框内标示动作状态名和描述动作的语句，动作状态之间的迁移用箭头表示，迁移上可以附加条件、发送子句和动作表达式。活动图是状态图的变形，它根据对象状态的变化捕获动作（所完成的工作和活动）和它们的结果，表示各个动作及其之间的关系。

- 与状态图不同的是，活动图中动作状态之间的迁移不是靠事件触发，当动作状态中的活动完成时迁移就被触发。在活动图中，事件只能附加到开始点到第一个动作之间的迁移。在活动图中，还可画条件判定符号（菱形符号）。条件判定符号可以有两个或两个以上携带条件的输出迁移，当其中的某个条件为真时，该迁移被触发。此外，活动图中还使用了“泳道”的概念。

在Rose中建立活动图



讨论：动作和活动

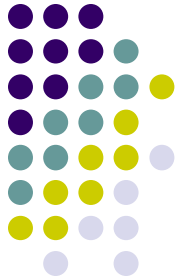


- UML中动作和活动的含义是不同的：

- 动作： 一组可执行的语句
  - 迁移性：与状态迁移有关
  - 原子性：这组语句不可中断
  - 连续性：一组语句必须连续执行，直到完毕

- 活动：一组可执行的动作
  - 有限性：完整的活动有一定的期限
  - 非原子性：这组动作可因某一事件发生而中断

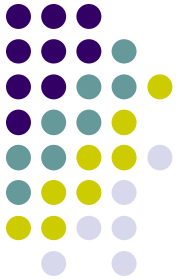
深入理解活动图中的概念



1. 动作状态

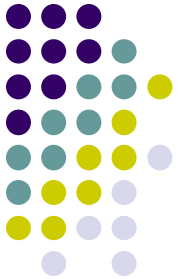
动作状态表达不可中断的动作或操作的执行，用于对实例中原子性（不可分割）动作或算法的执行步骤建立模型

动作状态的特点



- 状态的内部动作不可中断：不能有入口动作、出口动作和内部迁移；
- 可以有一个入迁移和至少一个出迁移；
- 动作状态必须指定在单泳道内，指明负责该泳道的对象运行该状态中的动作；
- 除了出现在动作状态的出迁移的事件中外，其他无关事件不出现在出迁移的事件中；
- 在一张活动图中，同一动作状态可多次出现。

2. 活动状态（活动结点）

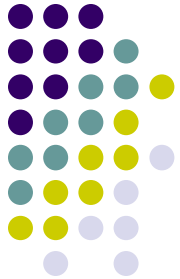


活动状态拥有一组不可中断的动作或操作，表达一个非原子的运行。

活动状态有如下特点：

- 活动本身可以中断，且通常需要持续一个时间段才能完成；
- 活动状态中可以有入口动作、出口动作和内部迁移；
- 活动状态可以分解，也可以用另一个活动图来表达；
- 活动图中，活动状态图标也用动作状态图标表示。

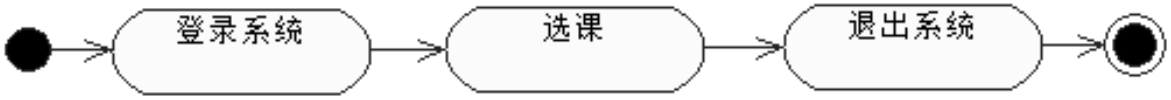
一个活动状态是由一系列动作状态组成。



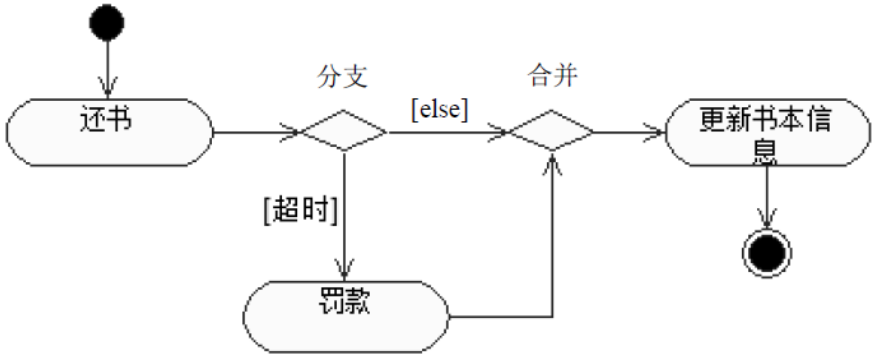
3. 动作流

一个实例的不同动作状态之间的联系，说明状态之间的控制流。

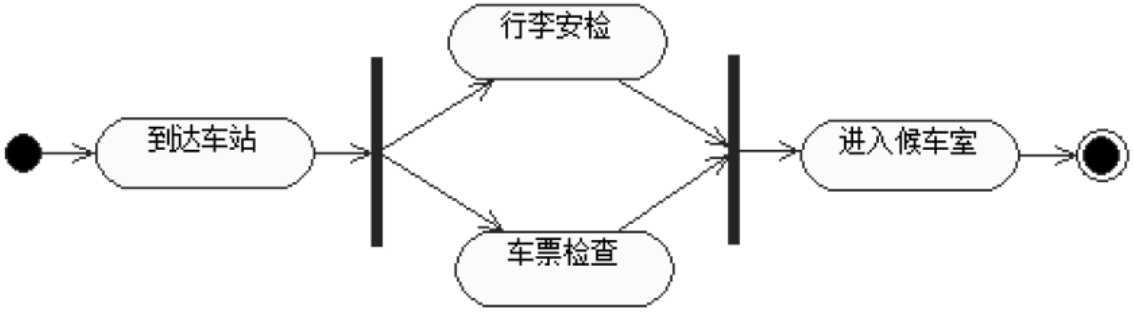
- 动作流：由实箭线表示，相当于状态图中的迁移；
- 无条件动作流：
- 条件动作流：
- 条件分支：
- 条件合并：



- 顺序的动作流（控制流）

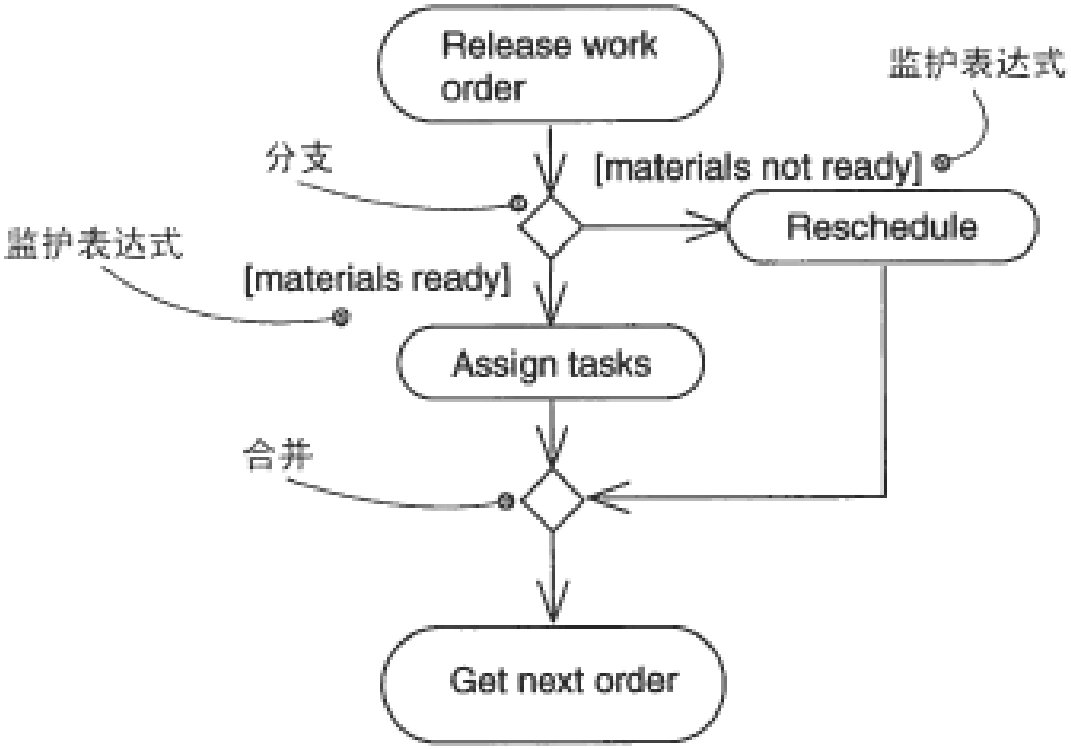


- 包含分支的动作流（控制流）

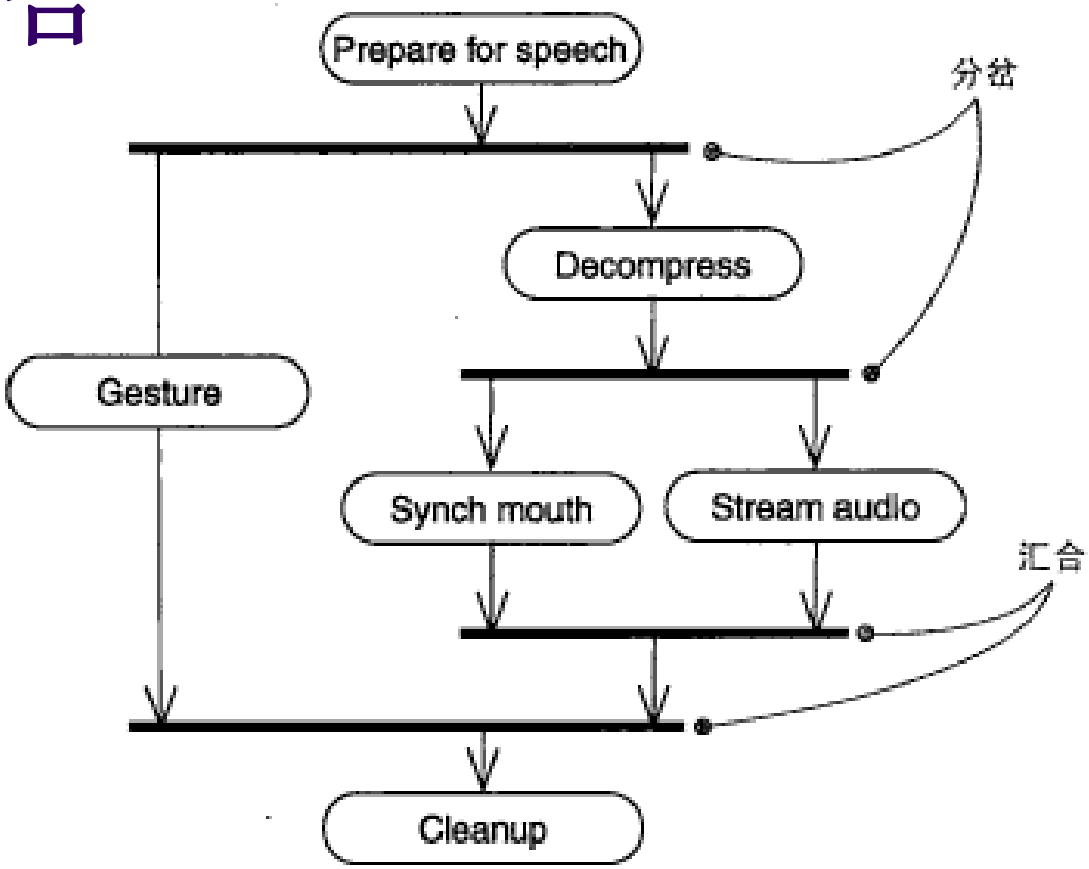


- 并发动作流（控制流）

分支


Release work order
监护表达式
分支
[materials not ready]
Reschedule
监护表达式
[materials ready]
Assign tasks
合并
Get next order

分岔和汇合



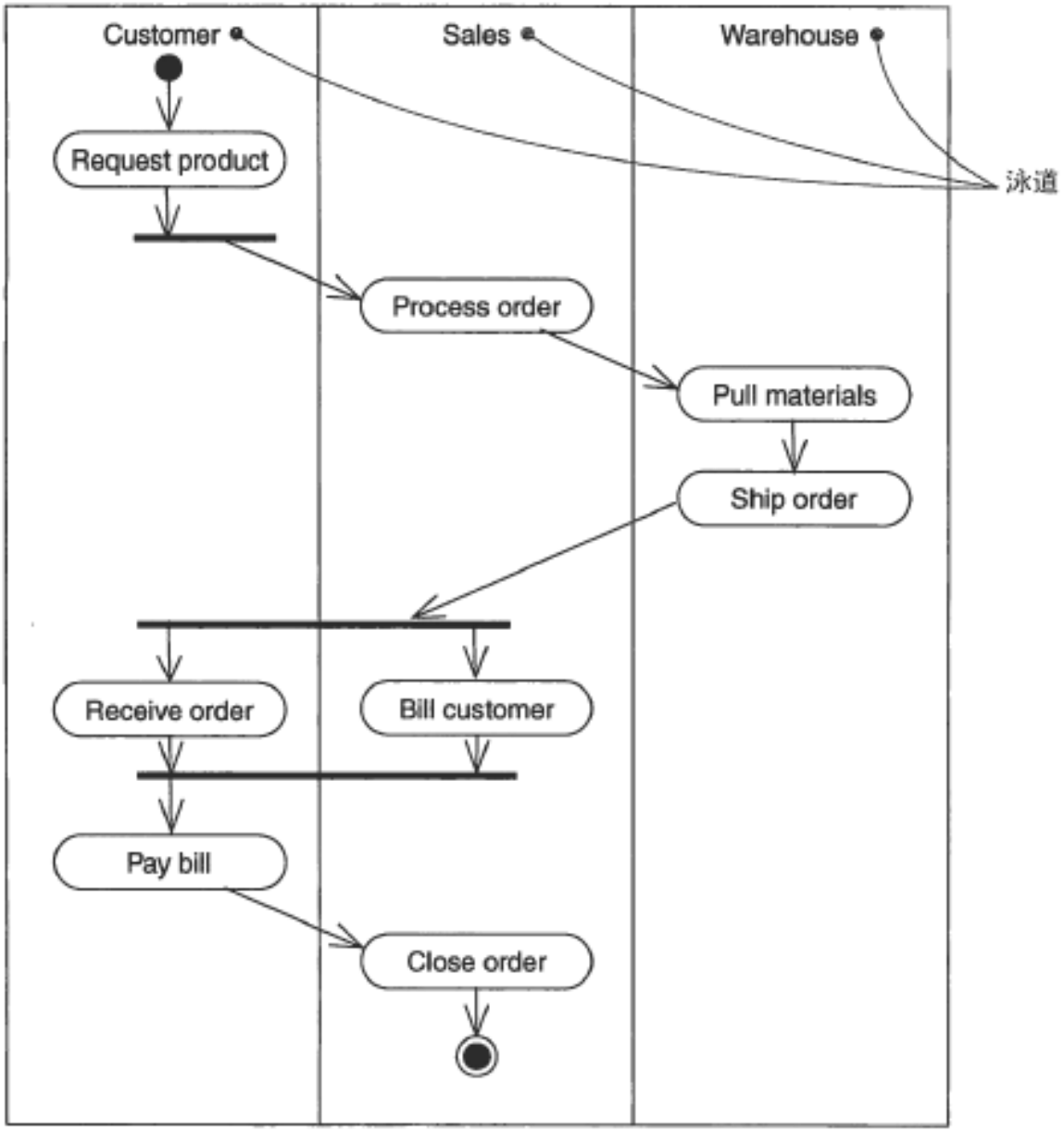
模拟人讲话和做手势的声音和动作同步的仿真系统内的并发控制流

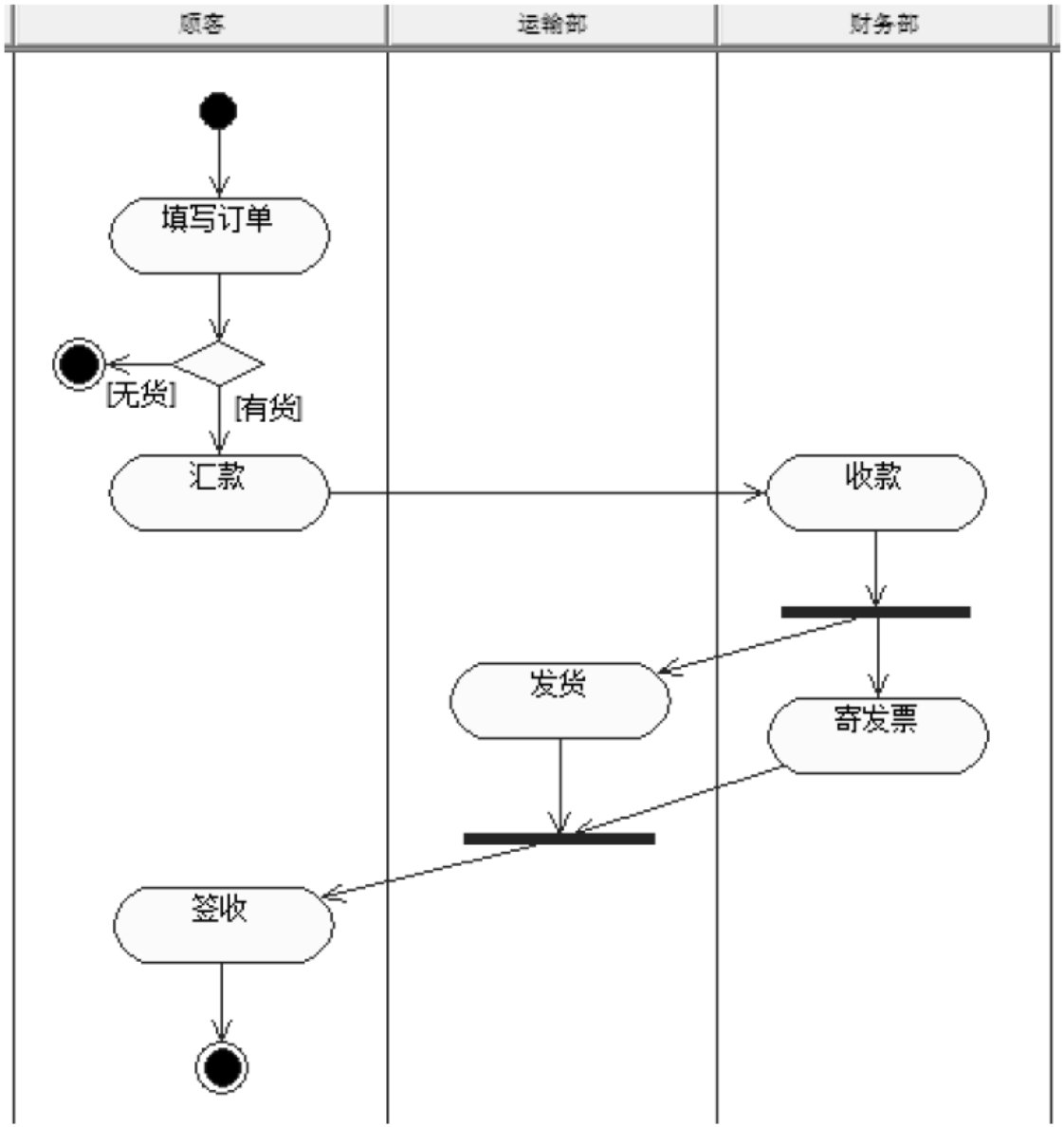
4. 泳道

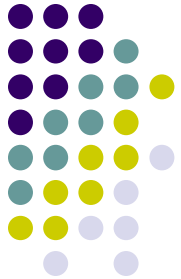
- 活动图告诉你发生了什么,但没有告诉你该项活动由谁来完成。在程序设计中,这意味着活动图没有描述出各个活动由哪个对象来完成。

- 泳道解决了这一问题。它将活动图的逻辑描述与顺序图、合作图的责任描述结合起来。

- 泳道用矩形框来表示,属于某个泳道的活动放在该矩形框内,将对象名放在矩形框的顶部,表示泳道中的活动由该对象负责。


Customer
Sales
Warehouse
泳道
Request product
Process order
Pull materials
Ship order
Receive order
Bill customer
Pay bill
Close order

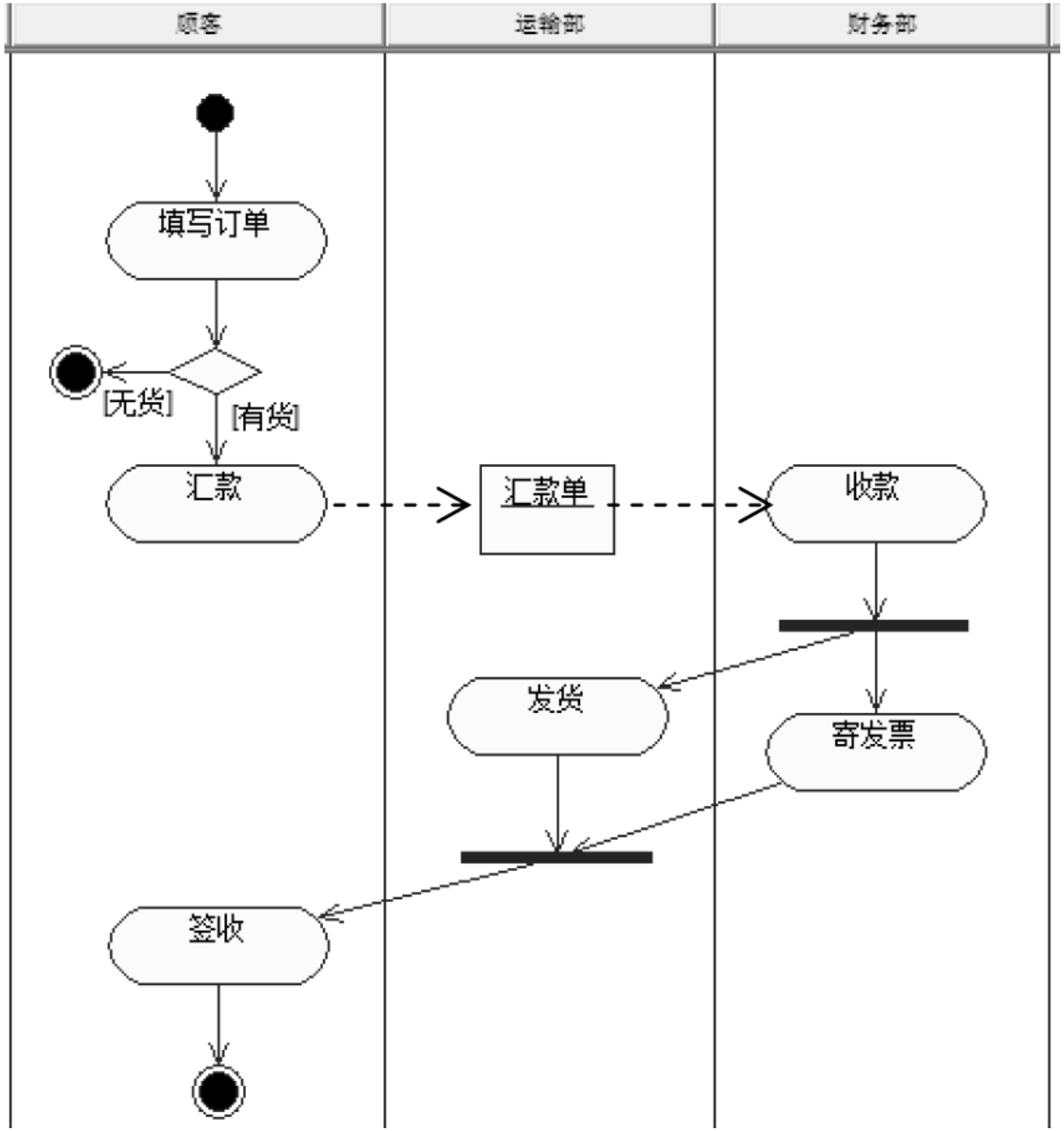

顾客
运输部
财务部
填写订单
[无货]
[有货]
汇款
收款
发货
寄发票
签收

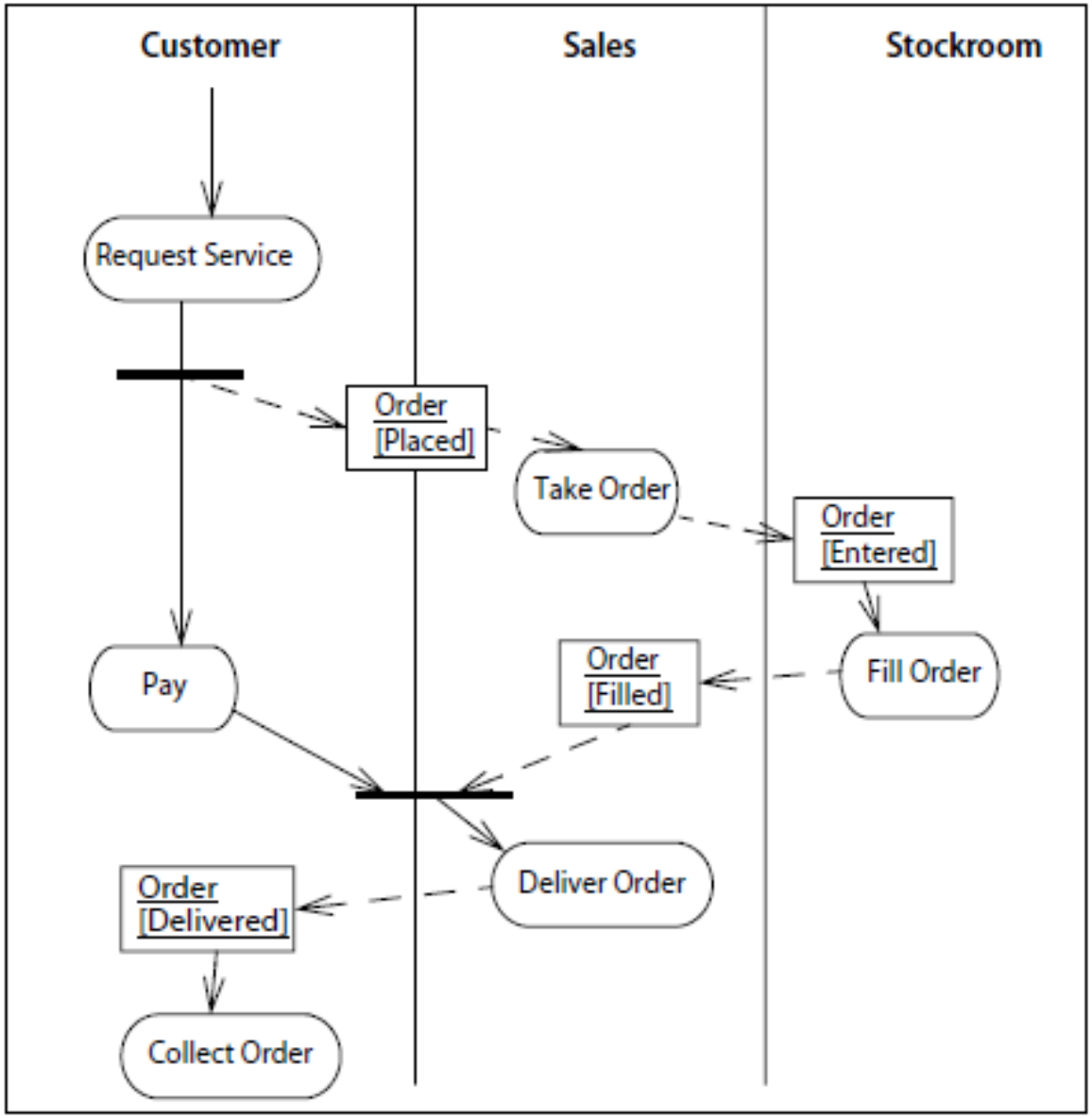


5. 活动图中的（其他）对象

- 在活动图中可以出现对象。对象可以作为活动的输入或输出,对象与活动间的输入/输出关系由虚线箭头来表示。

- 如果仅表示对象受到某一活动的影响,则可用不带箭头的虚线来连接对象与活动 。


顾客
运输部
财务部
填写订单
[无货]
[有货]
汇款
汇款单
收款
发货
寄发票
签收


Customer
Sales
Stockroom
Request Service
Order
[Placed]
Take Order
Order
[Entered]
Pay
Order
[Filled]
Fill Order
Deliver Order
Order
[Delivered]
Collect Order

6. 信号

在活动图中可以表示信号的发送与接收,分别用发送和接收标志来表示。发送和接收标志也可与对象相连,用于表示消息的发送者和接收者。

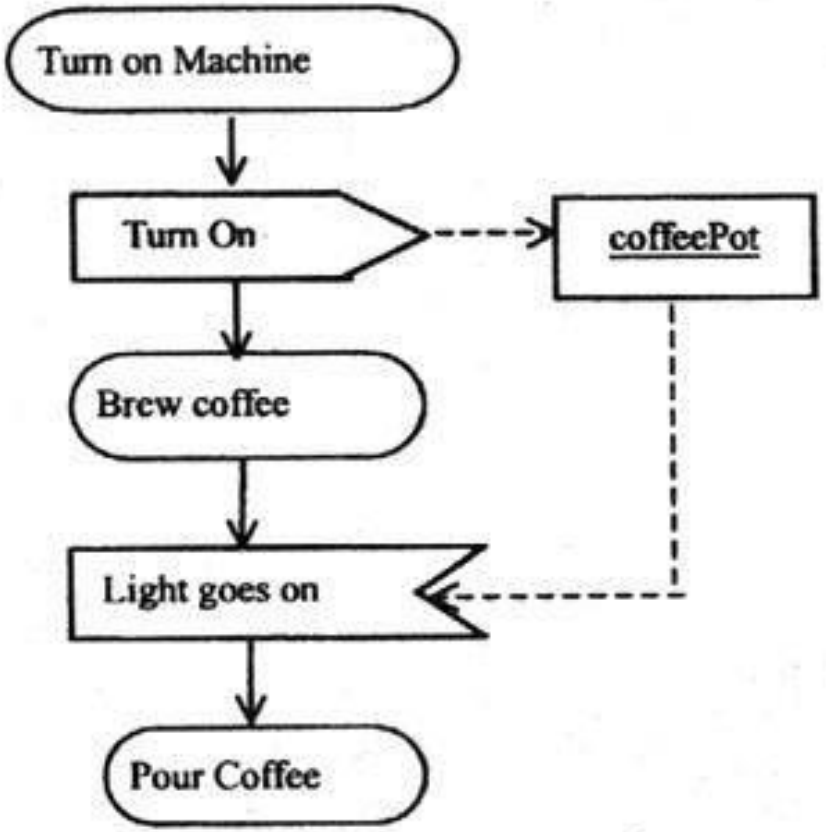
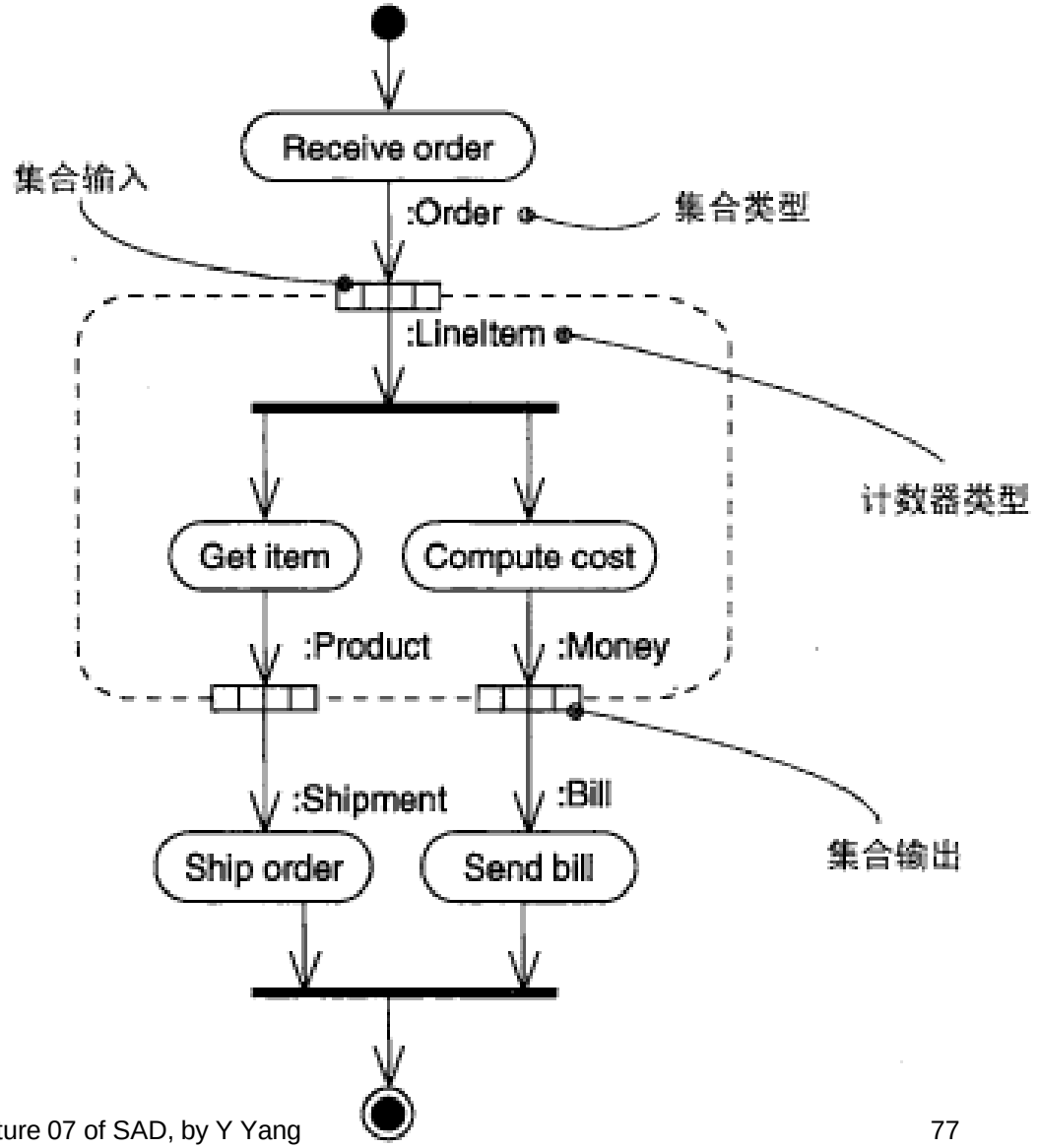


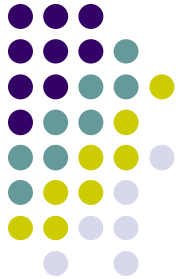
图 3 带有消息接收和发送标志的活动图

7. 扩展区域（expansion region）

同一个操作经常要在一组元素上执行。不用循环，而直接用扩展区域来建模。

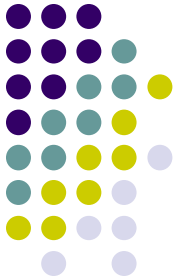


活动图的层次关系



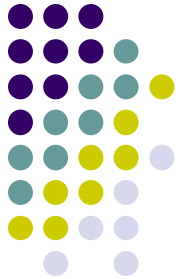
- 对于一个复杂的面向对象系统来说，需要很多个活动图对其进行描述。
- 活动按其构成可以分为简单活动和组合活动。
- 一个不含有内部嵌套活动或动作的活动，称为简单活动。
- 一个内部嵌套了若干活动或动作的活动，称为组合活动，其中被嵌套的活动称为子活动。

讨论：动态建模中四种图的运用



- 不要对系统中的每个类（对象）都画状态图。尽管这样做很完美,但太浪费精力,其实你可能只关心某些类的行为。

- 正确的做法是:为帮助理解类而画它的状态图。状态图描述跨越多个用例的单个对象的行为,而不适合描述多个对象间的行为合作。为此,常将状态图与其它技术(如顺序图、合作图和活动图)组合使用。



- 顺序图和合作图适合描述单个用例中几个对象的行为。其中顺序图突出对象间交互的顺序,而合作图的布局方法能更清楚地表示出对象之间静态的连接关系。当行为较为简单时,顺序图和合作图是最好的选择。但当行为比变复杂时,这两个图将失去其清晰度。

- 因此,如果想显示跨越多用例或多线程的复杂行为,可考虑使用活动图。另外,顺序图和合作图仅适合描述对象之间的合作关系,而不适合对行为进行精确定义,如果想描述跨越多个用例的单个对象的行为,应当使用状态图。