

活动图

活动图的概述

- 用例图显示系统应该做什么，活动图则指明了系统将如何实现它的目标。活动图可以理解为用例图的细化。
- 活动图本质上是一种流程图，它描述从活动到活动的控制流。
- 用来建模工作流，活动图可以显示用例内部和用例之间的路径。

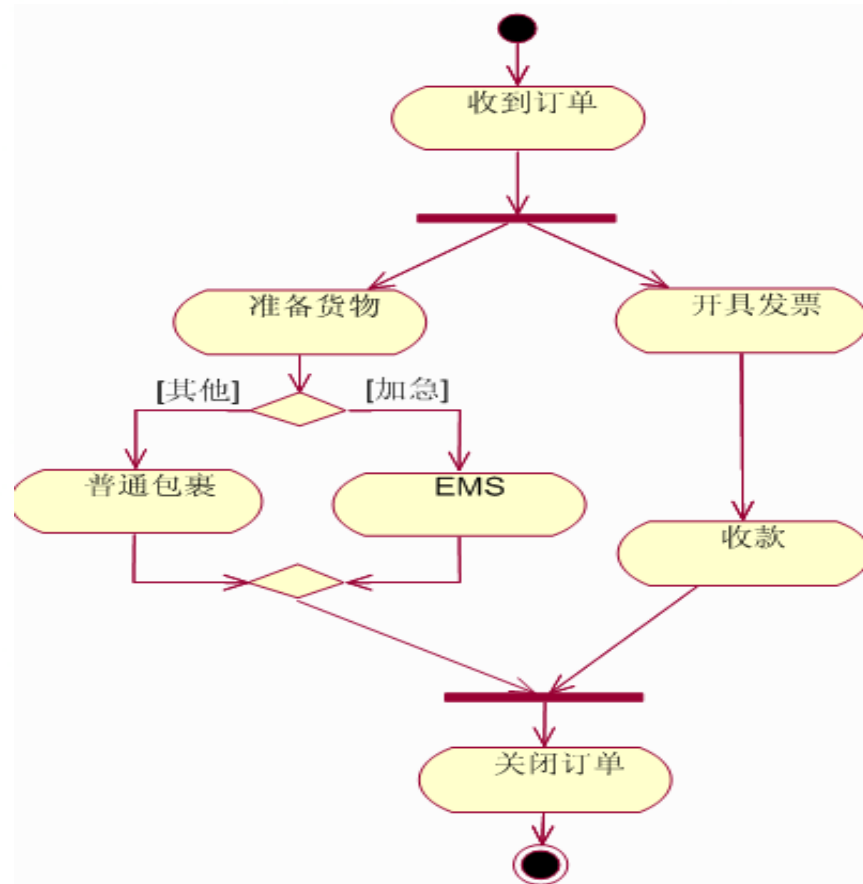
活动图的概述

- **活动**：指某件事情正在进行的状态。
- 活动图是一种**描述系统行为的图**，它用于**展现参与行为的类所进行的各种活动的顺序关系**。
- 活动图描述**系统中发生的操作流程**，用来在**面向对象系统的不同组件之间建模工作流和并行过程行为**。
 - 例如，可以使用活动图描述某个用例的基本操作流程。

活动图的概述

活动图定义：

活动图是由**活动节点**和**转换流程**构成的图。它描述系统或业务的一系列活动构成的控制流，描述系统从一种活动转换到另一种活动的整个过程，**即**用来描述事物或对象的活动变化流程。活动图用于对系统的**计算流程**和**工作流程**建模。



某公司销售过程的活动图

活动图与流程图的区别

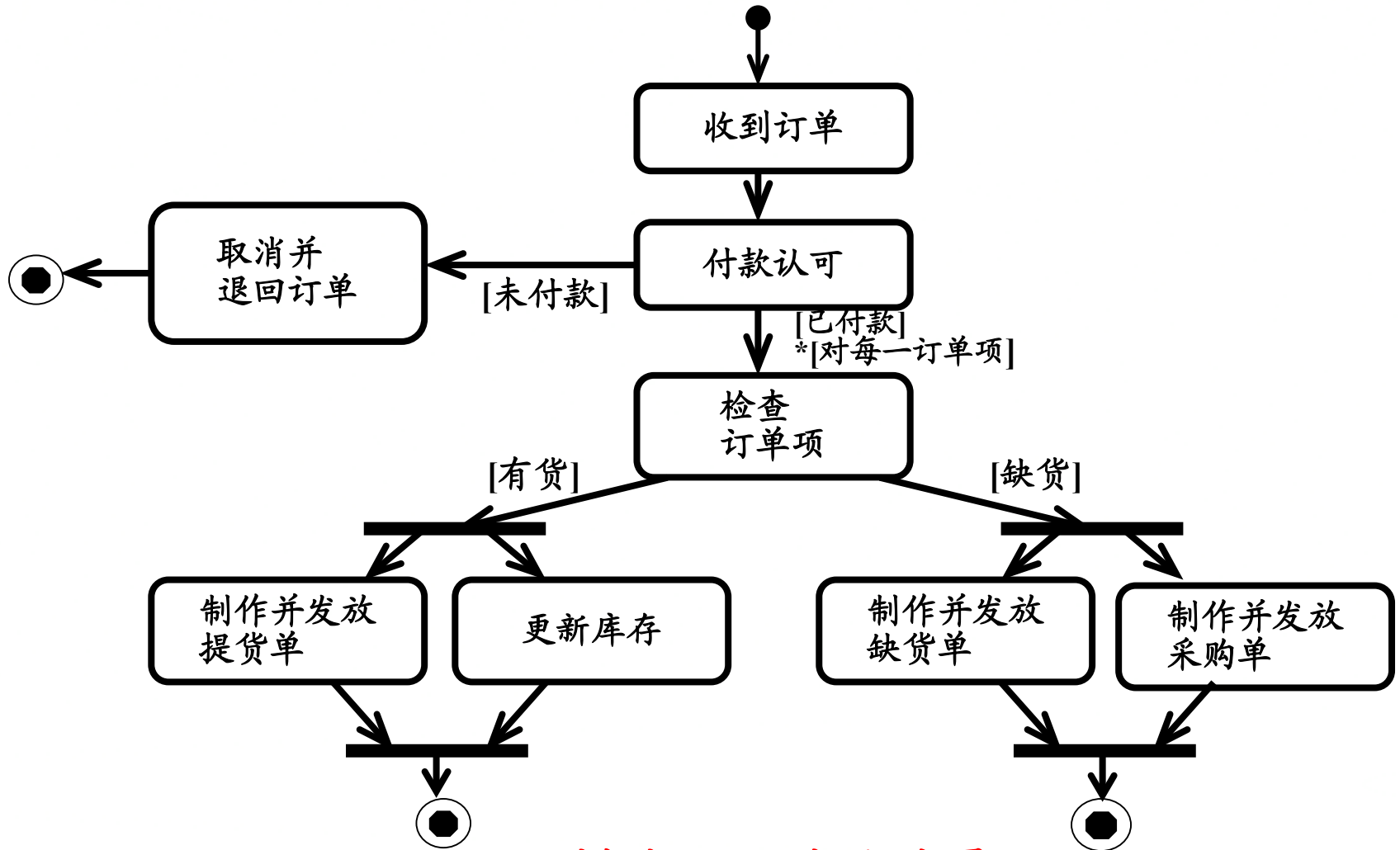
- **流程图**着重描述处理过程，它的主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系；而**活动图**描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统的行为，而非系统的处理过程。
- **活动图**能够表示并发活动的情形，而流程图不能。
- 活动图是**面向对象**，流程图是**面向过程**的。

活动图的概述

活动图的作用：

活动图常用来描述业务或软件系统的活动轨迹，描述了系统的活动控制流程。我们常用活动图对**业务过程**、**工作流**和**用例实现**进行建模。

活动图的概述



描述用例的活动图

活动起点

活动

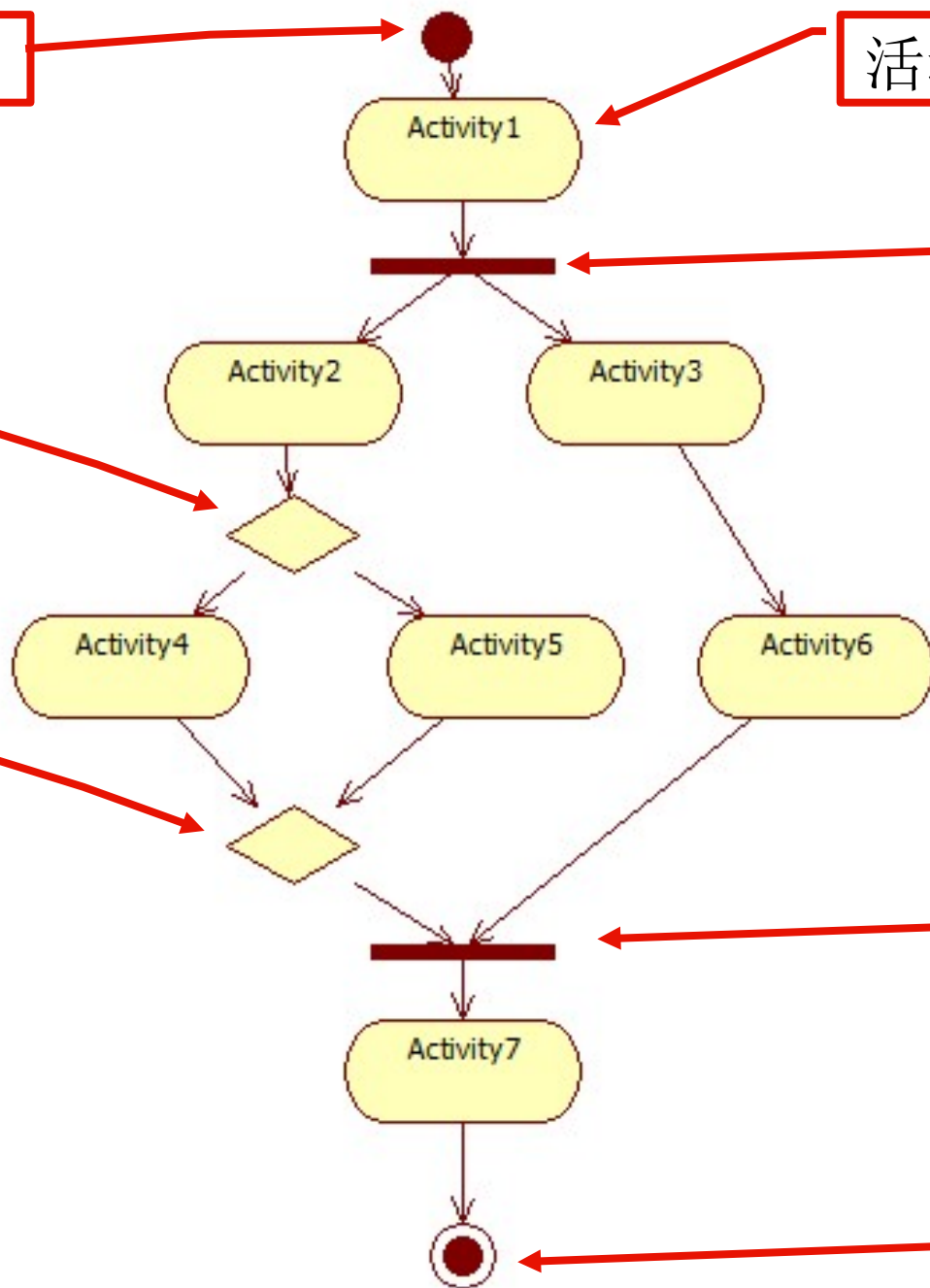
分叉

分支

合并

汇合

活动终点



活动图的表示

- 活动图的组成元素
 - 活动图的元素包括初始节点、终点、活动节点、转换、分支、分叉与汇合。其中，转换、分支、分叉与汇合把多个活动节点连接在一起。
- 活动图和交互图是UML中对系统动态方面建模的两种主要形式，交互图强调对象与对象之间的交互消息，而活动图则强调的是从活动到活动的控制流程。

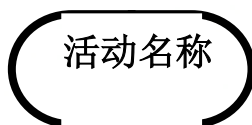
活动图的表示



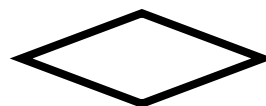
起点



终点



活动



判断条件



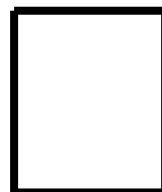
同步条



接收信号



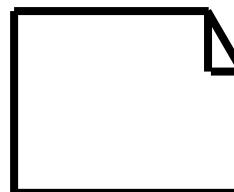
发送信号



泳道



转移



注释体



注释连接

活动图的表示

1. 初始节点和终点

初始节点表示活动的起点；终点表示活动的终结点。
用一个**实心圆**表示初始节点，用一个**圆圈内加一个实心圆**来表示活动终点。在活动图中，可能包含多个活动终点，但**有且仅有一个**起始点。



初始节点和终点



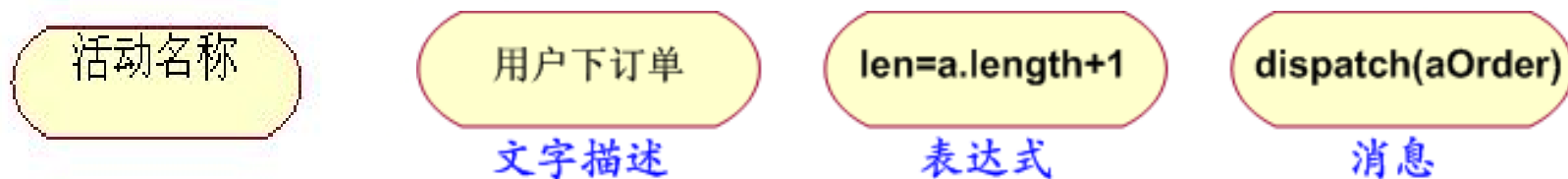
- 在一个活动图中只能有一个开始状态，但可以有多多个结束状态。下图演示了开始状态和结束状态一对多的关系。
- 从图中我们可以看出，该活动图仅包含一个开始状态，但是对应了3个结束状态。从开始状态进入到“口渴了”状态之后无论转移到哪个活动都将结束控制流。



活动图的表示

2. 活动节点

活动节点是活动图中最主要的元素之一，它用来表示一个活动，一个活动表示多个动作的集合（**步骤**）。活动节点用一个**圆角矩形**表示。活动的名称写在圆角矩形内部。



活动节点的表示

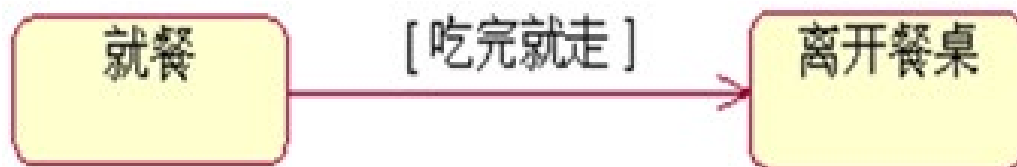
活动图的表示

3. 转换

当一个活动结束时，活动控制流就会马上传递给下一个活动节点，在活动图中称之为“转换”，用一条带箭头的直线来表示转换。



转换的表示

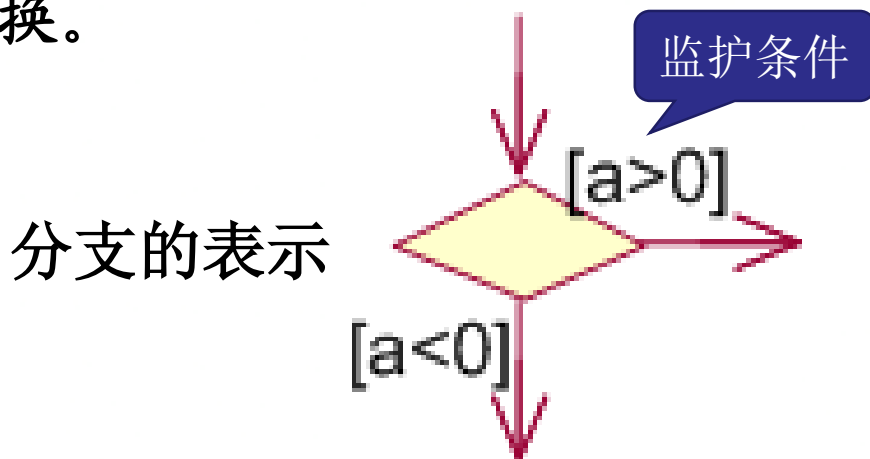


活动图的表示

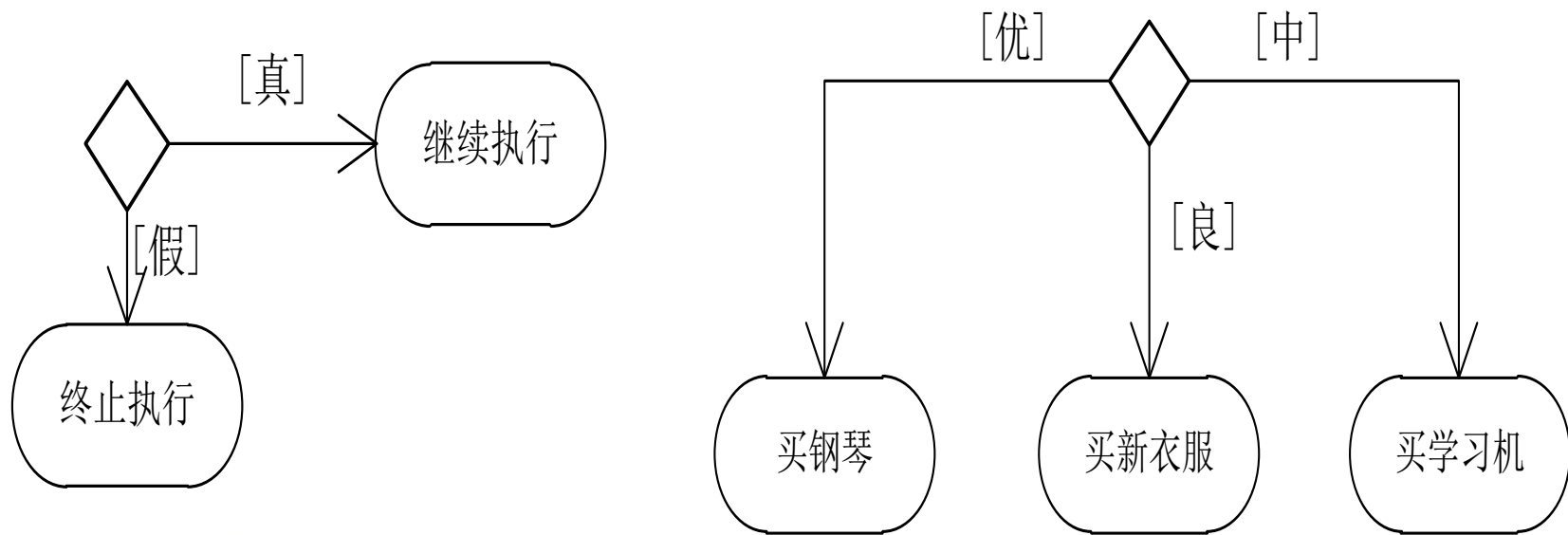
4. 分支与监护条件（判断）

在实际应用中，有三种活动控制流，分别是**顺序结构**、**分支结构**、**循环结构**。当从一个活动节点到另一个活动节点的转换需要条件时，常用**分支与监护条件**来表示活动的分支结构。

分支是用菱形表示的，它有一个**进入转换**（箭头从外指向分支符号），一个或多个**离开转换**（箭头从分支符号指向外）。而每个离开转换上都会有一个监护条件，用来表示满足某种条件时才执行该转换。



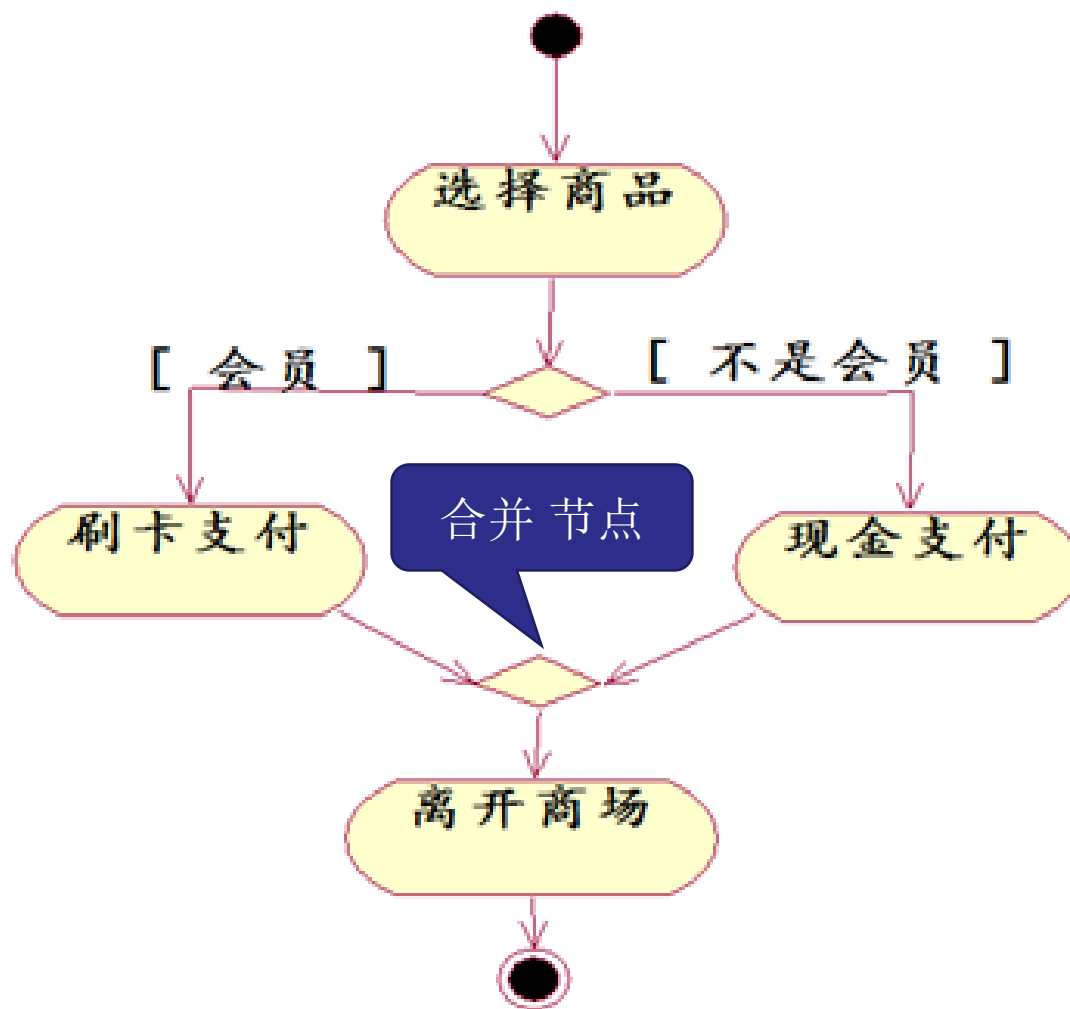
活动图的表示





- **合并将两条路径连接到一起，合并成一条路径。前面使用菱形用作判断，并根据条件转向不同的活动或状态。这里菱形被用作合并节点，用于合并不同的路径。我们可以将合并节点当成一个省力的工作，它将两条路径重合部分建模为同一步骤序列。**
- **实际应用中，菱形标记符不管是用作判断还是作为合并控制流，在活动图中都使用得十分广泛，几乎每个活动图中都会用到。**





练习

- 活动图中用于将判断节点的多个控制流合并的元素是（ ）。

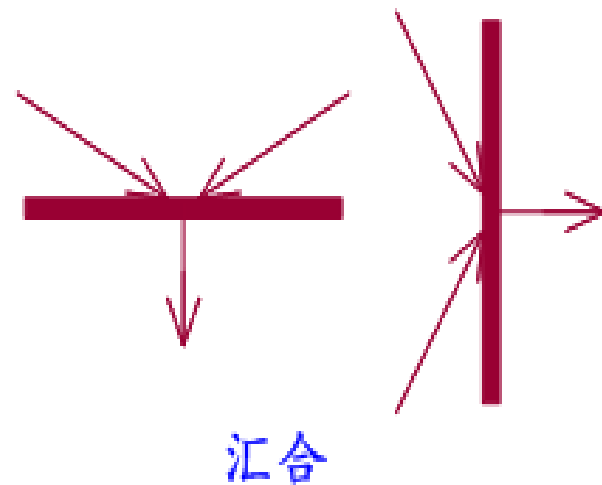
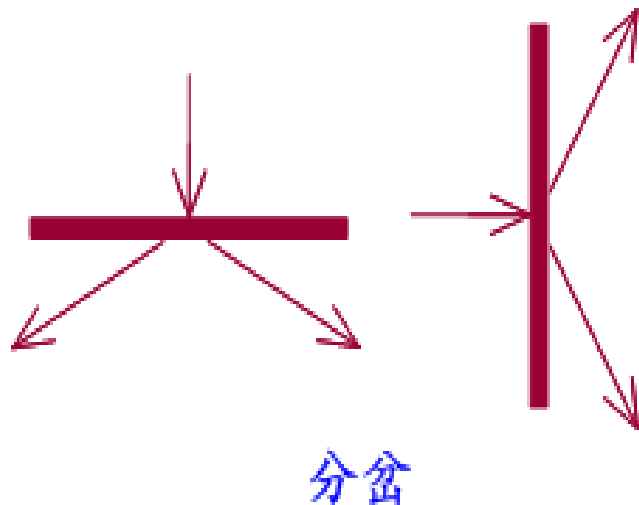
- A. 汇合节点
- B. 判断节点
- C. 合并节点
- D. 分叉节点

答案：C

活动图的表示

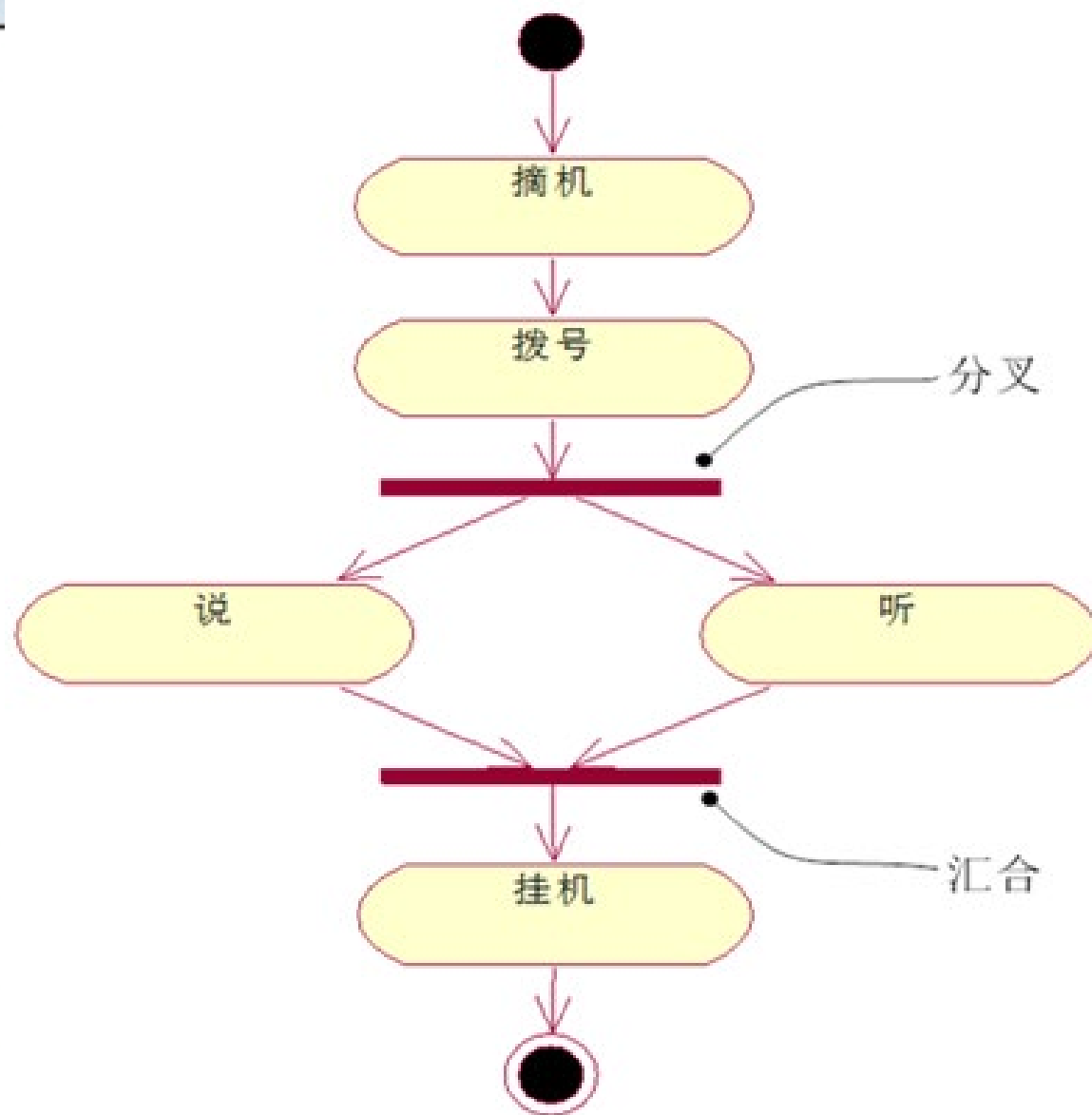
• 5. 分叉与汇合


- 在实际应用中，如果活动的转换是有条件的，我们就用分支与监护条件来表示转换，如果一些活动是并发执行的，我们就用分叉和汇合来表示并发活动。分叉线和汇合线都使用加粗的水平线或垂直线段表示。



活动图的表示

- **分叉**：每个分叉可以有一个输入转换和两个或多个输出转换，每个转换都可以是独立的控制流。
- **汇合**：当两个或多个并发控制流都达到汇合点后，活动流程才能进入下一个活动节点。
- **分叉**用来表示两个或者多个并发活动的分支；而**汇合**则用于同步这些并发活动的分支，当且仅当所有的并发分支(活动)都到达汇合点后，活动流程才能进入下一个活动节点。





例如下图中，“获得订单”活动之后的分叉表示活动“安排付款”和“调货”可以并发进行，两个活动之后的汇合表示，需要等到两个活动全部完成之后才可以继续进行下一个活动“交货”。



下图中用了一个分叉和一个汇合描述进入火车站候车厅前的活动图。首先到达火车站，此时要求分别安检随身携带的行李和检查乘车车票，这两项检查是并发进行的，当两个活动都完成时同时到达下一个状态后，才能进入候车厅动作。



练习

- 活动图中（ ）用于将两个或多个并发控制流合并到一起，仅当所有控制流都到达时，才形成一个单向的输出控制流。

- A. 分支节点
- B. 汇合节点
- C. 分叉节点
- D. 合并节点

答案： B

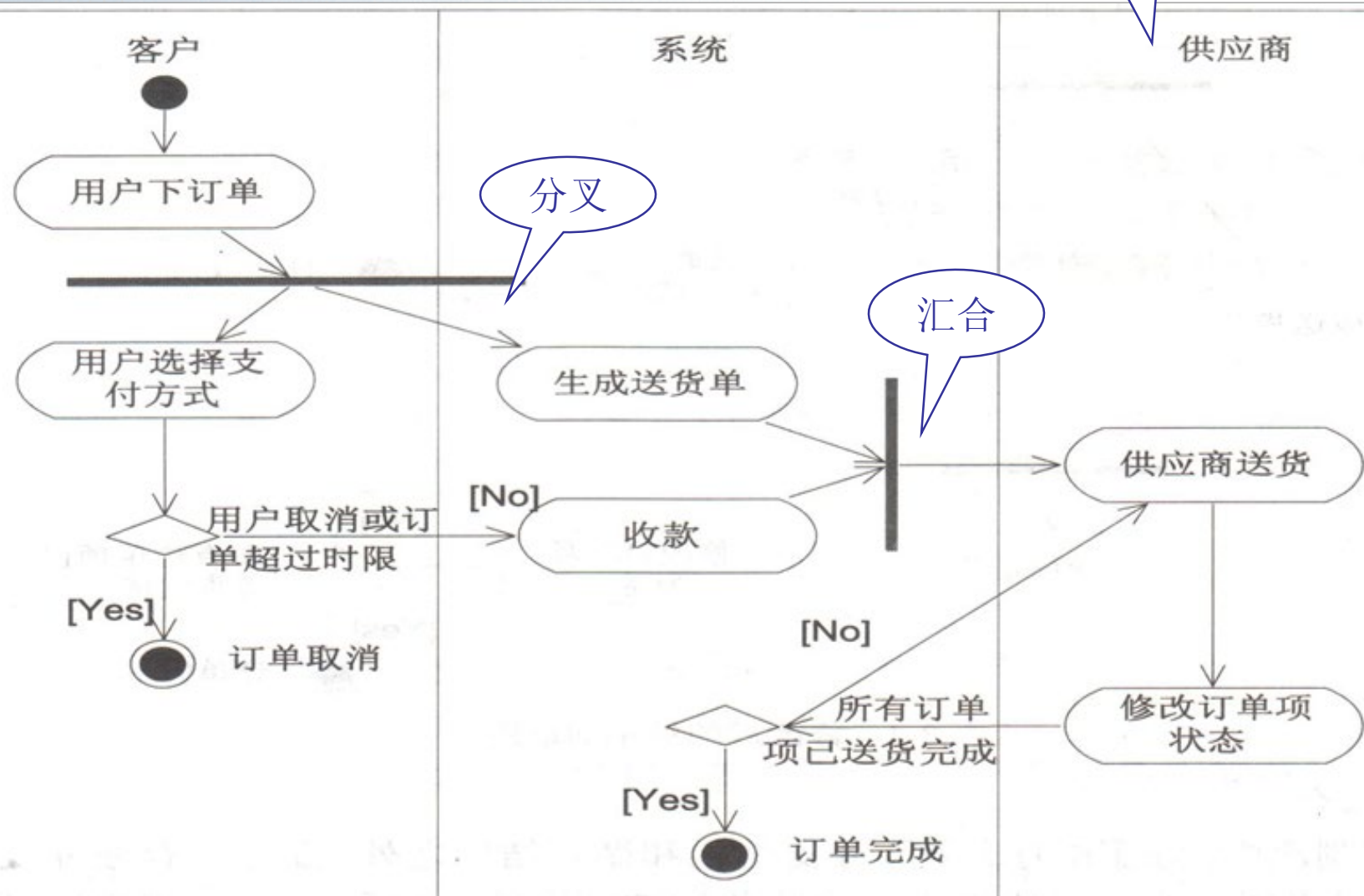
活动图的表示

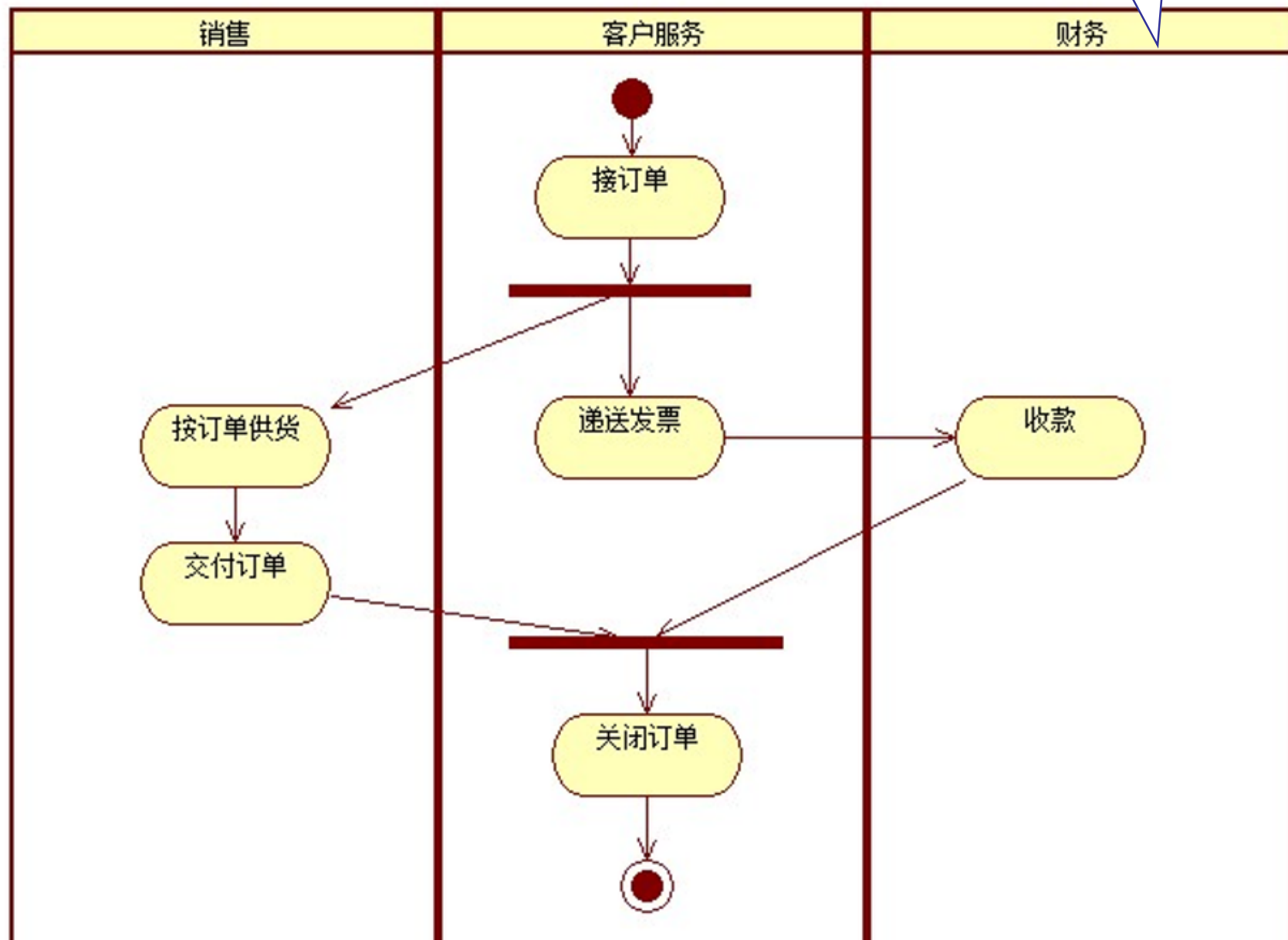
6、泳道：就像一个游泳运动员只能在一个泳道里进行比赛一样，一个对象也只能在一个业务流程中担任一个（或一类）职责。

- 为了有效地表示各个活动由谁负责的信息，可以通过**泳道** (Swim Lane)来实现。
- 每个泳道用一条垂直的线将它们分开，并且每个泳道都必须有一个唯一的名称，每个活动节点、分支必须只属于一个泳道，而**转换**，**分叉**与**汇合**是可以跨泳道的。通过泳道，不仅体现了整个活动控制流，还体现出了每个活动的**实施者**。
- 泳道最主要的用途是在**分析用例场景时用来获取角色职责**。



泳道





练习

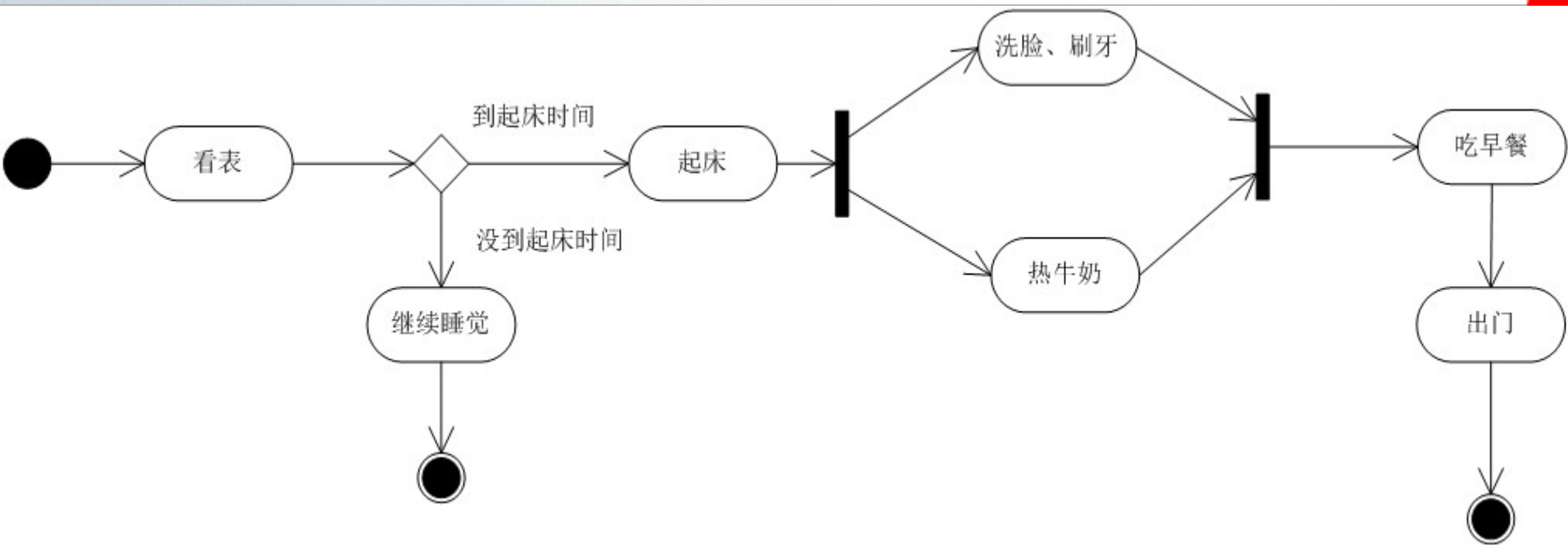
• 活动图中用于活动分组的元素是（ ）。

- A. 泳道
- B. 控制流
- C. 判断节点
- D. 包

答案： A

练习

- 活动描述：小张每天醒来后先看表是否到起床时间，如果没到继续睡觉；如果到了，抓紧时间起床，洗漱完毕后吃早餐，饭后出门。
- 小张早晨起床上班的活动，用活动图描述

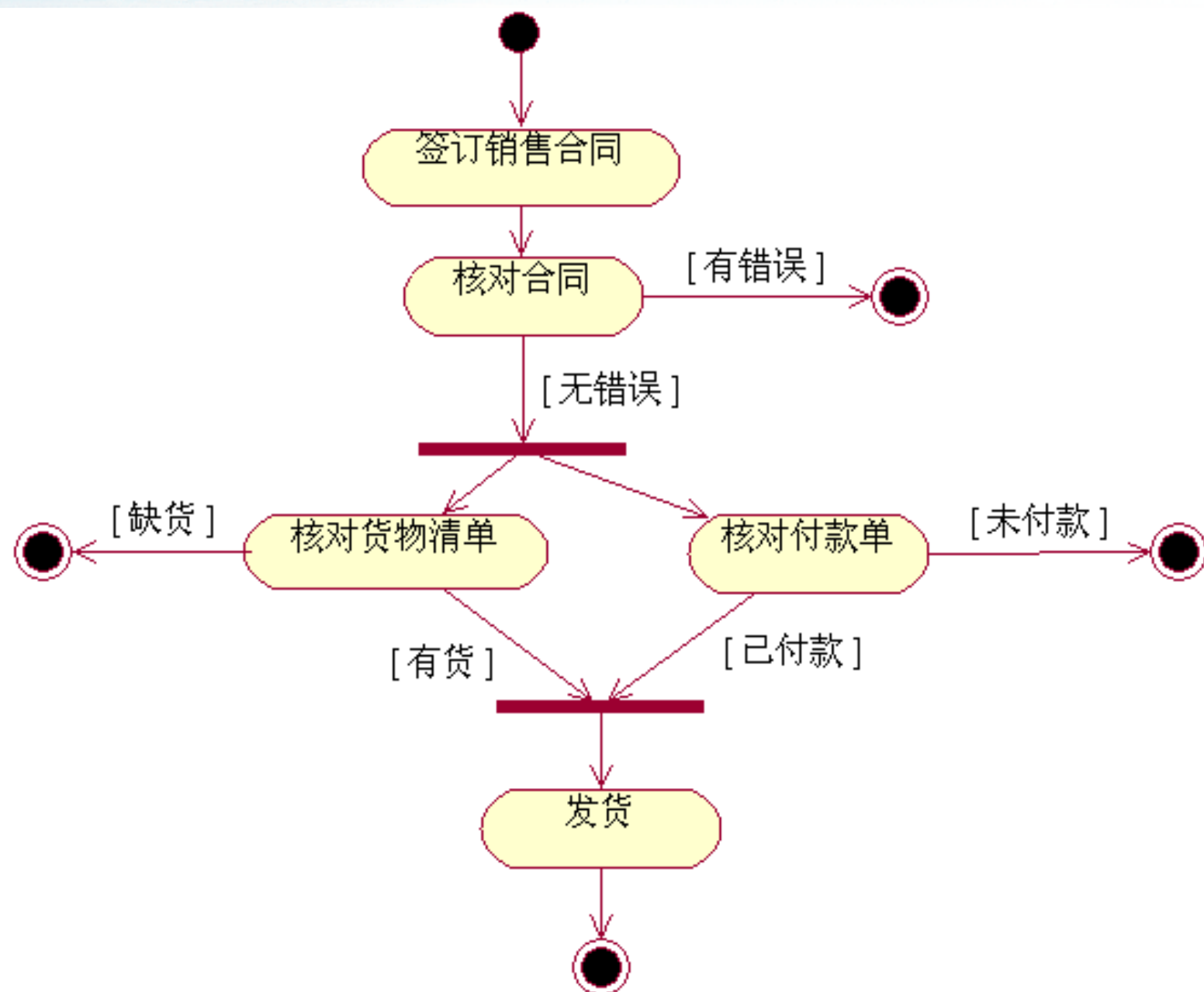


上图绘制的就是小张早晨日常生活的活动图，通过这种图形化模型可以把动作的流程性表达的更加清楚，动作的内容、流程、判断、交互、并发都能很好的表达。

练习

- 销售合同从签订到履约的过程

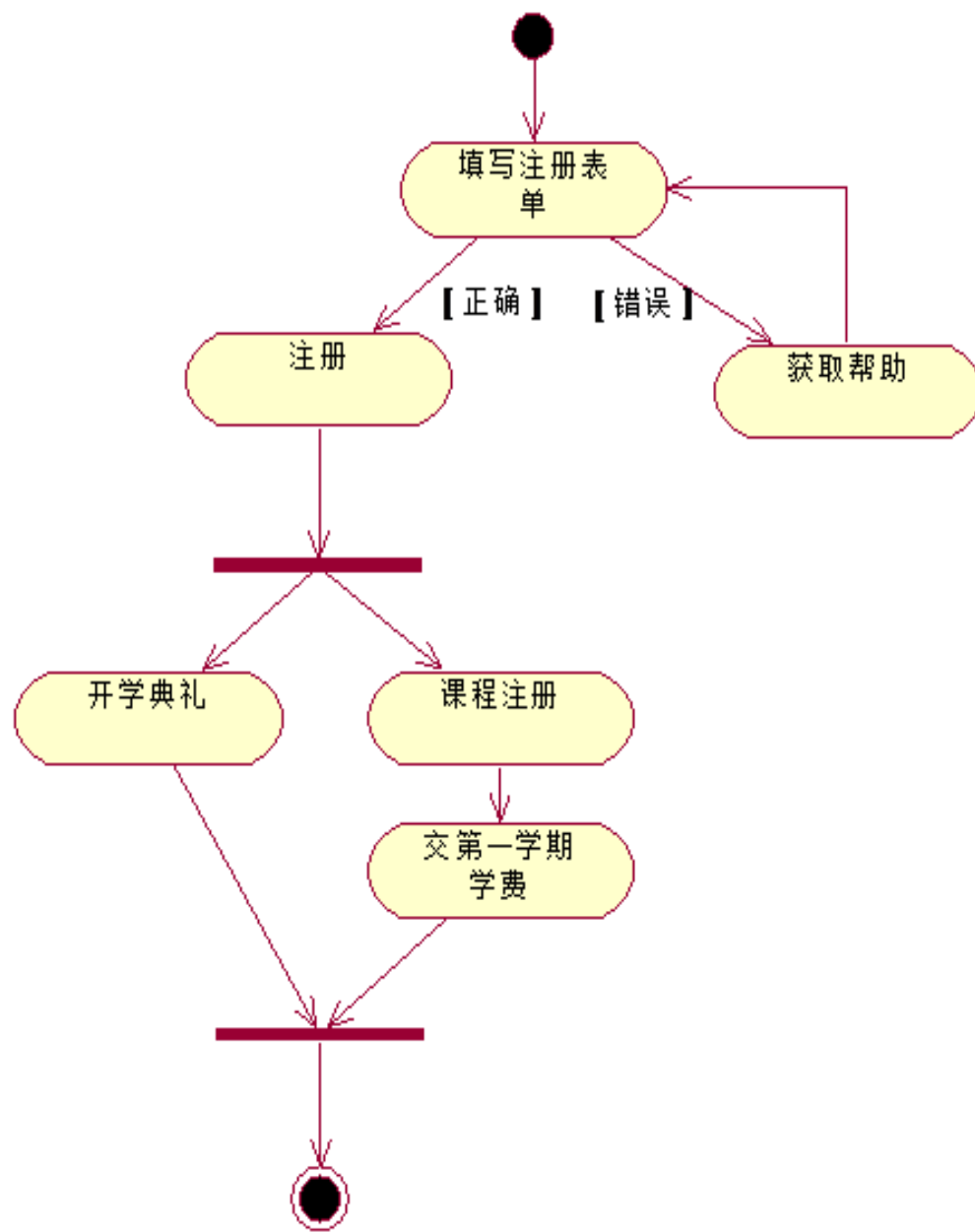
- 销售合同签订后，要进行核对。如果发现错误，则终止履约；如果没有错误，则要核对货物清单确定是否有货，还要核对付款单确定对方是否已经付款，只有这两项都完成，才可以发货。如果无货或对方尚未付款，则终止履约。请绘制销售合同从签订到履约的过程活动图。



练习

下面的文字描述了某大学新生报到的过程：

新生首先要填一张新生注册表单。如果填写不正确，则在别人的帮助下重新填写，直至填写正确，然后进行注册。注册成功之后，要进行开学典礼，同时在新生选课系统中注册，然后交齐第一个学期的学费。试使用活动图描述上述过程。



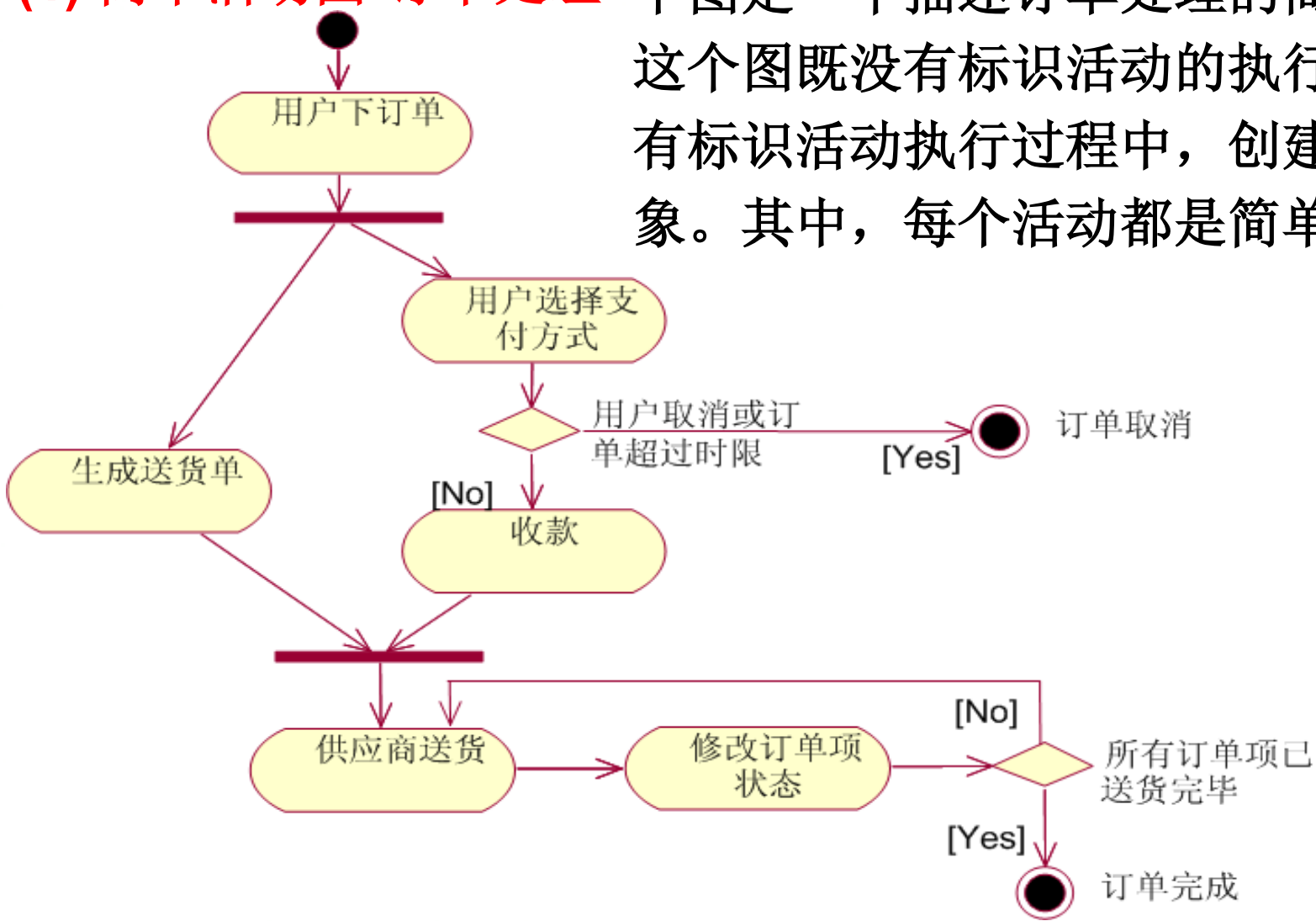
活动图分类

- 按照活动图表示的信息不同，将活动图分为：
 - (1)简单活动图
 - (2)标识泳道的活动图
 - (3) 标识对象流的活动图
 - (4)复合活动图



(1) 简单活动图-订单处理

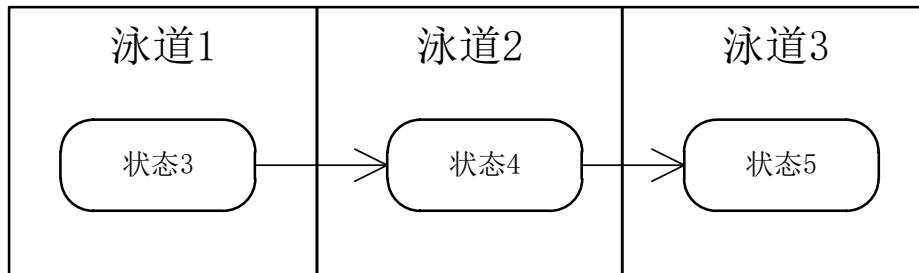
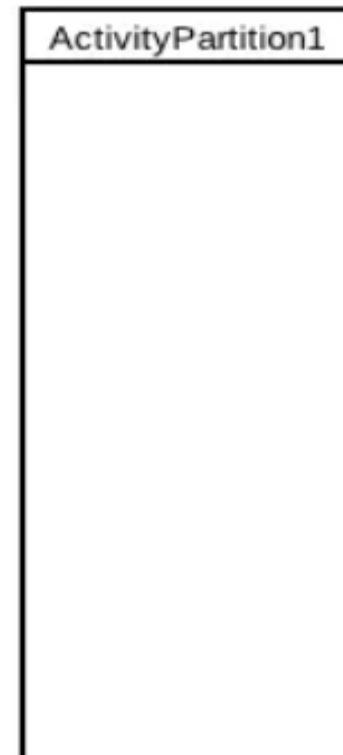
下图是一个描述订单处理的简单活动图，这个图既没有标识活动的执行者，也没有标识活动执行过程中，创建了哪些对象。其中，每个活动都是简单的活动。




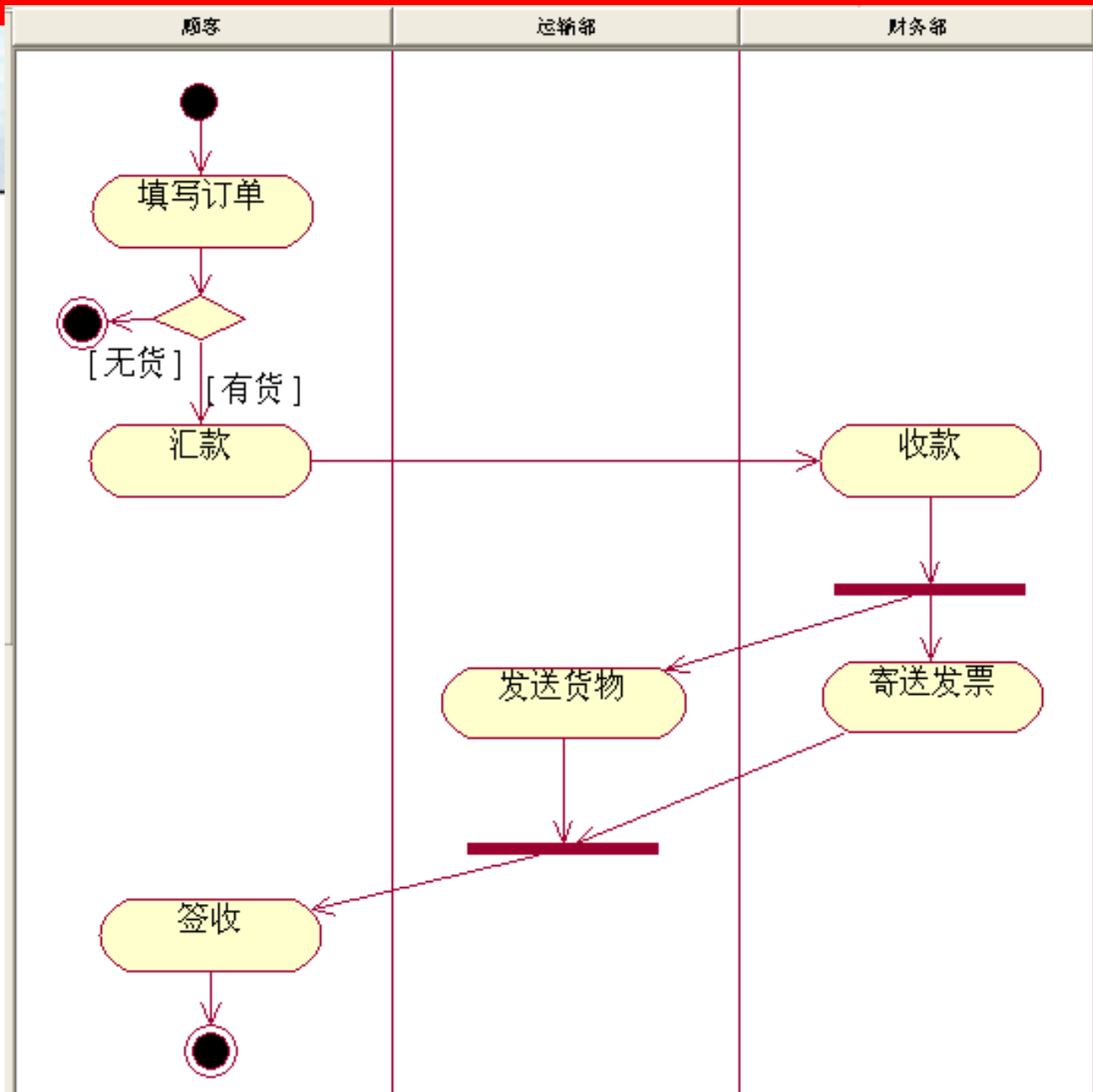
活动图分类

(2)标识泳道的活动图

- 为了有效地表示各个活动由谁负责的信息，可以通过泳道 (Swim Lane) 来实现。

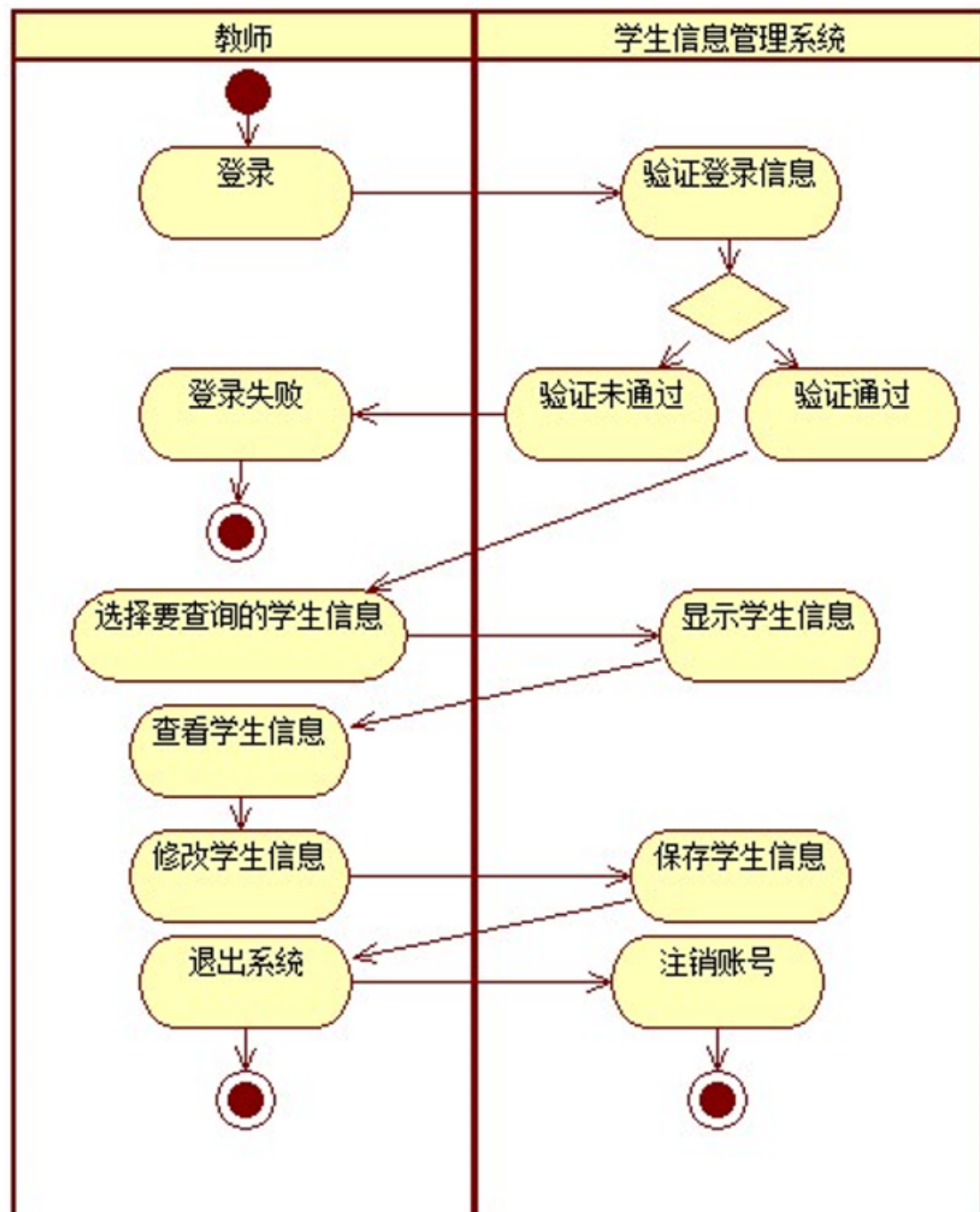


- 
- 每个泳道用一条垂直的线将它们分开，并且每个泳道都必须有一个唯一的名称。每个活动节点，**分支必须只属于一个泳道**，而**转换，分叉与汇合是可以跨泳道的**。通过泳道，不仅体现了整个活动控制流，还体现出了每个活动的**实施者**。
 - 泳道最主要的用途是在**分析用例场景时用来获取角色职责**。



练习

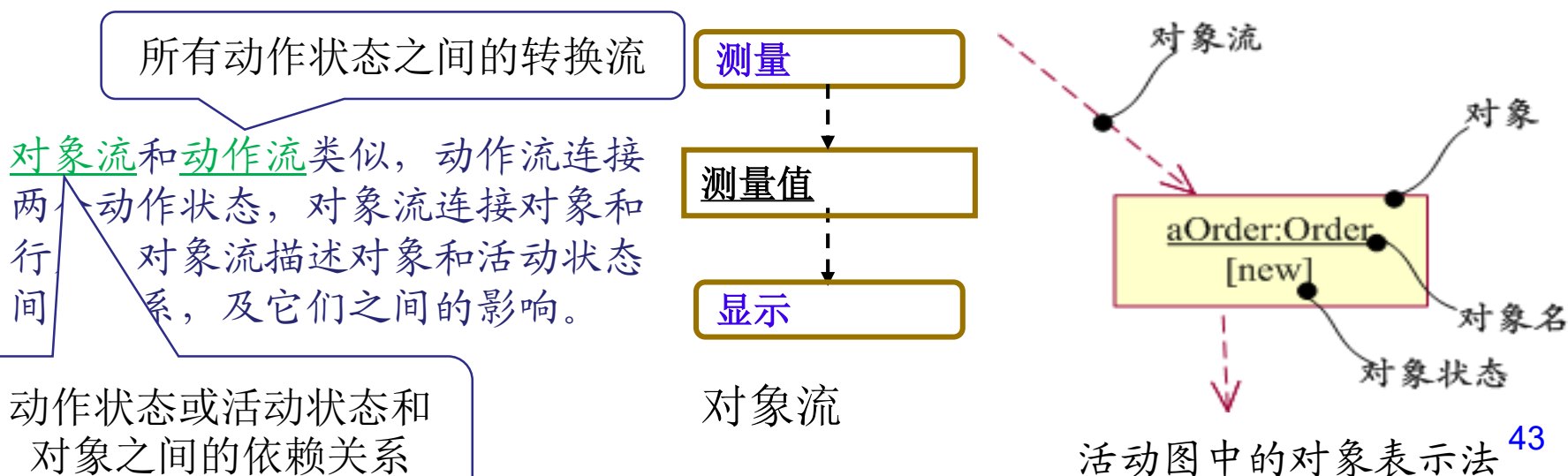
建立“教师查看、修改学生信息”用例的活动图。教师在登录时，系统会验证教师输入的账号、密码、动态码等登录信息，如果验证未通过，则登录失败。如果验证通过，教师登录成功，并选择需要查询的学生，系统会显示教师选中的学生信息。教师查看信息后，修改学生信息，修改完成后保存学生信息，这时系统会将修改后的信息保存到数据库。之后教师退出系统，系统注销教师账号。



活动图分类

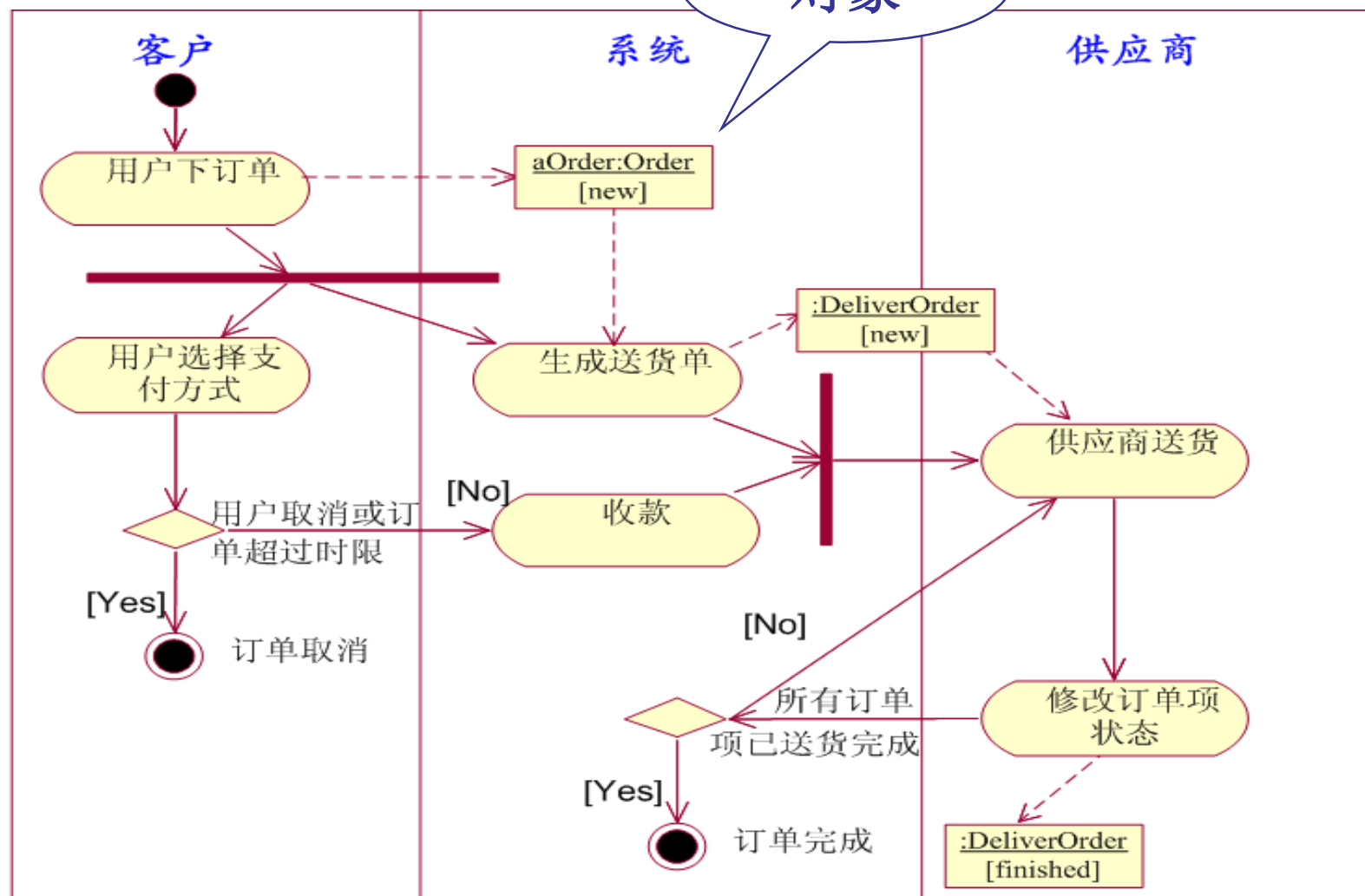
(3) 标识对象流活动图

- 在活动图中，存在这样一些现象：一种情况是，可能存在一些对象进入一个活动节点，经过活动处理，修改了对象的状态；另一种情况是，活动节点创建或删除了一些对象；还有一些情况是，输出一些对象。在这些活动中，对象与节点活动是紧密相关的，我们可以在活动图中把相关的对象标识出来。
- UML中，可以在活动图中标识一个对象的**角色**、**状态**和**属性值**的变化。



活动图分类

对象

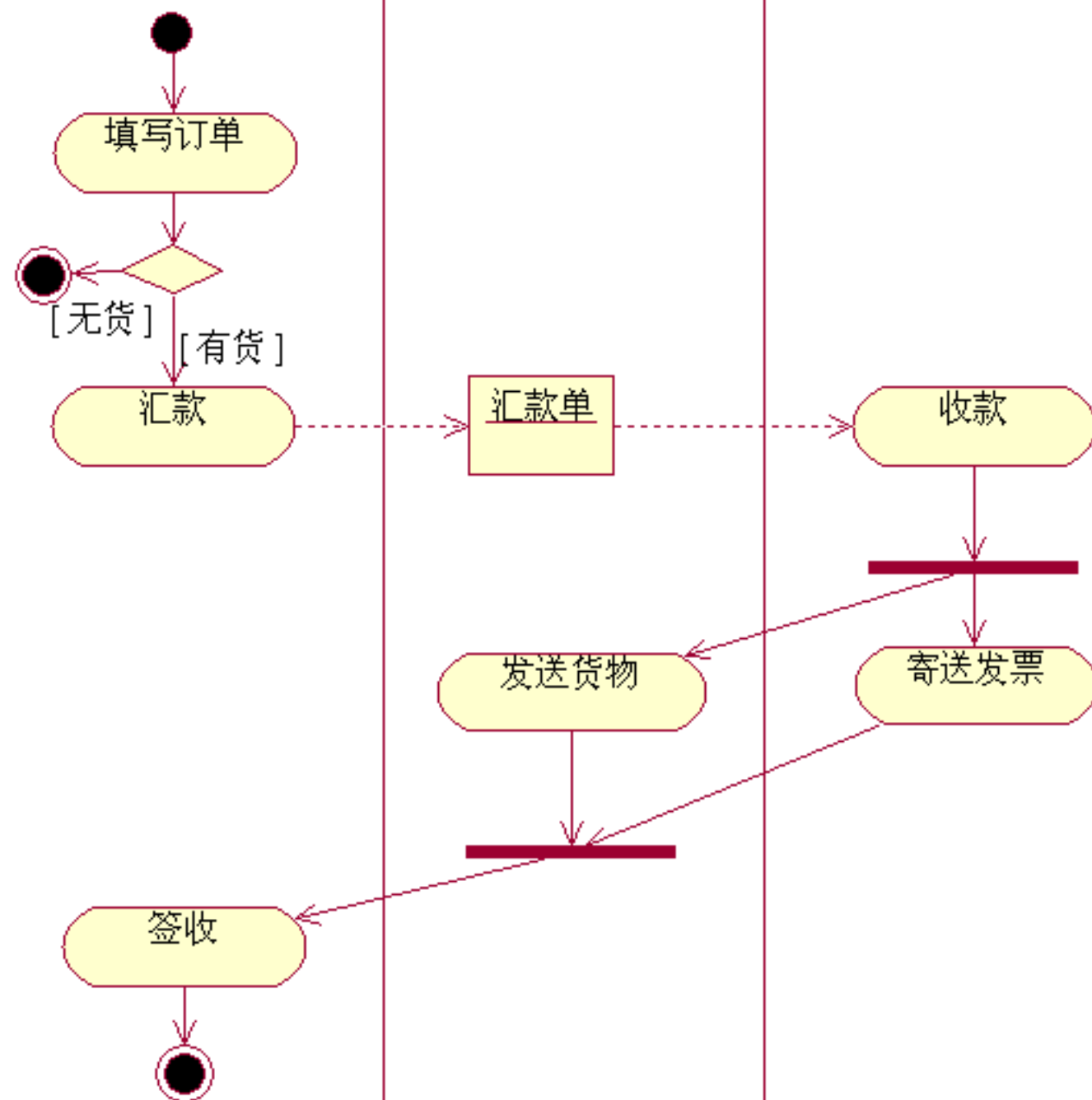


活动图分类

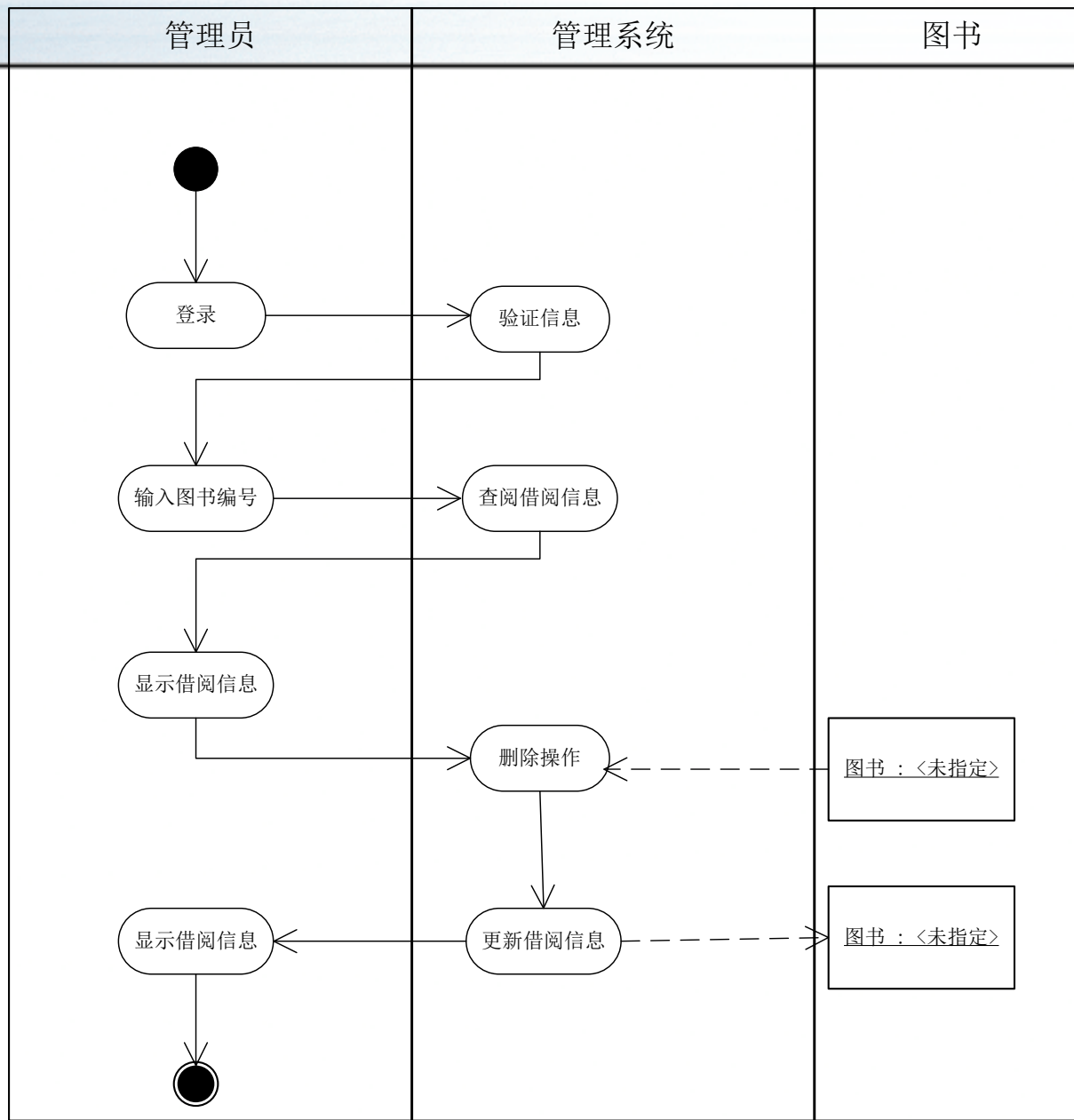
- 在图中，标识了一些关键的对象流，对象的状态也在图中作了标识：
 - 当“用户下订单”时，将创建一个**Order**类的实例，用来存放订单的信息，该**Order**类还包含着相应的**OrderItem**（针对每个产品一条）
 - 当“生成送货单”时，将根据**Order**类的实例创建多个**DeliverOrder**（送货单）的实例。
 - 当“修改订单项状态”之后，**DeliverOrder**对象的状态将变成**finished**。

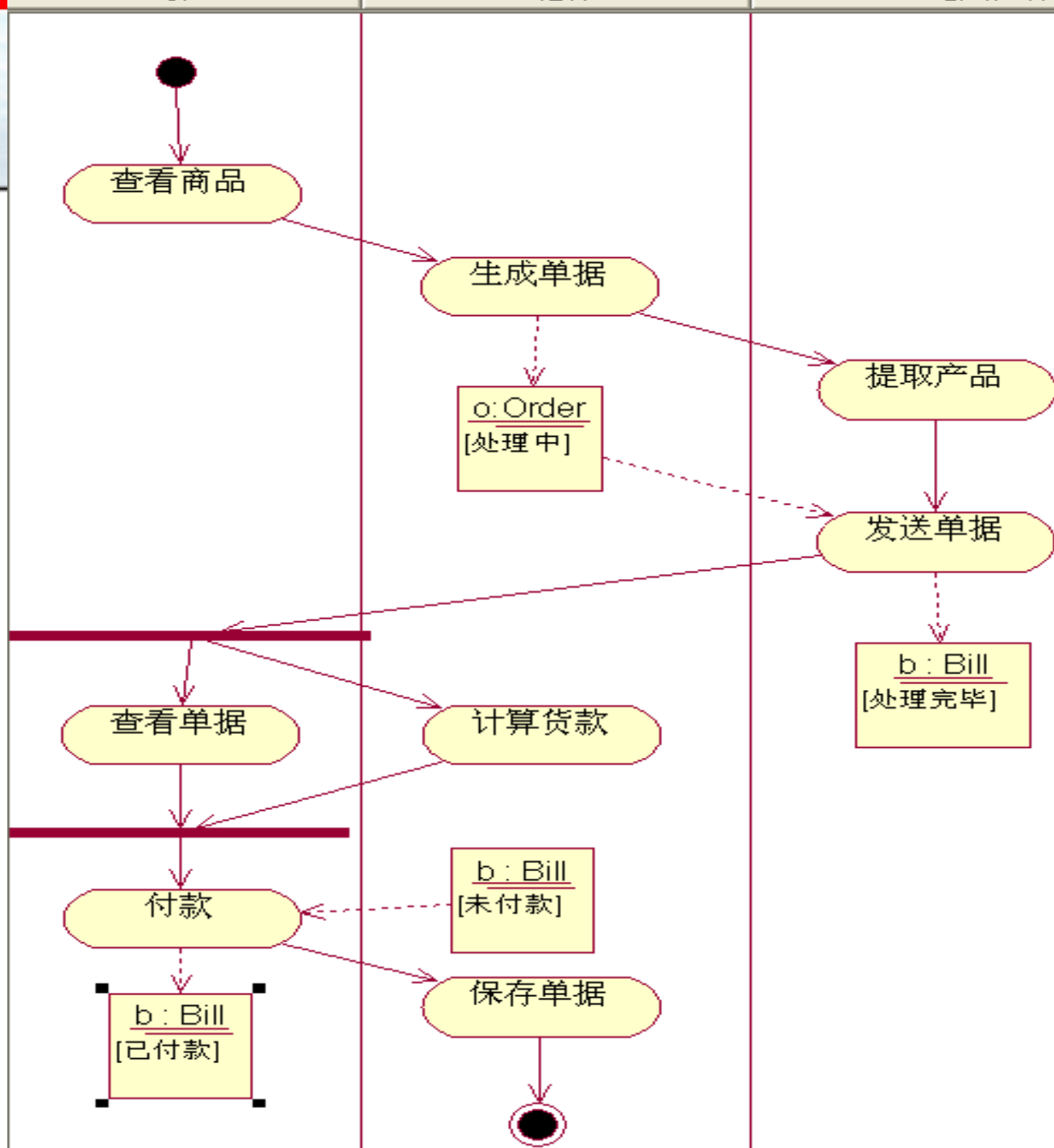
活动图分类

- 当然，在这张活动图中实际上还蕴藏着许多对象流，例如：
 - 当“收款”后，**Order**类的实例的状态就变成了“已付款”。
 - 当“修改订单项状态”后，**Order**类中部分订单项的状态就变成了“已送货”。
 - 当用户取消或订单超过时限时，**Order**类的状态就将成为**Cancel**。
- 在实际应用中，绘制活动图时并不一定需要将所有的对象流都标识出来，这样会使活动图变得复杂、混乱。在实际建模中，只对重要的对象进行描述。



活动图分类

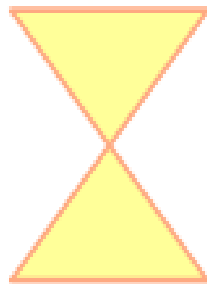




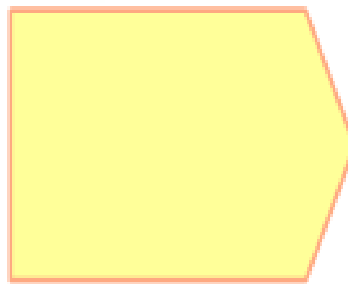
活动图分类

- (4) 标识信号的活动图

- 在交互图中，利用“信号”可以增加活动图的可读性。信号是表示两个对象之间进行异步通讯的方式，当一个对象接收到一个信号时，将触发信号事件。
- 信号在活动图中，有三种信号元素，它们是：发送信号，接收信号和时间信号。



时间信号



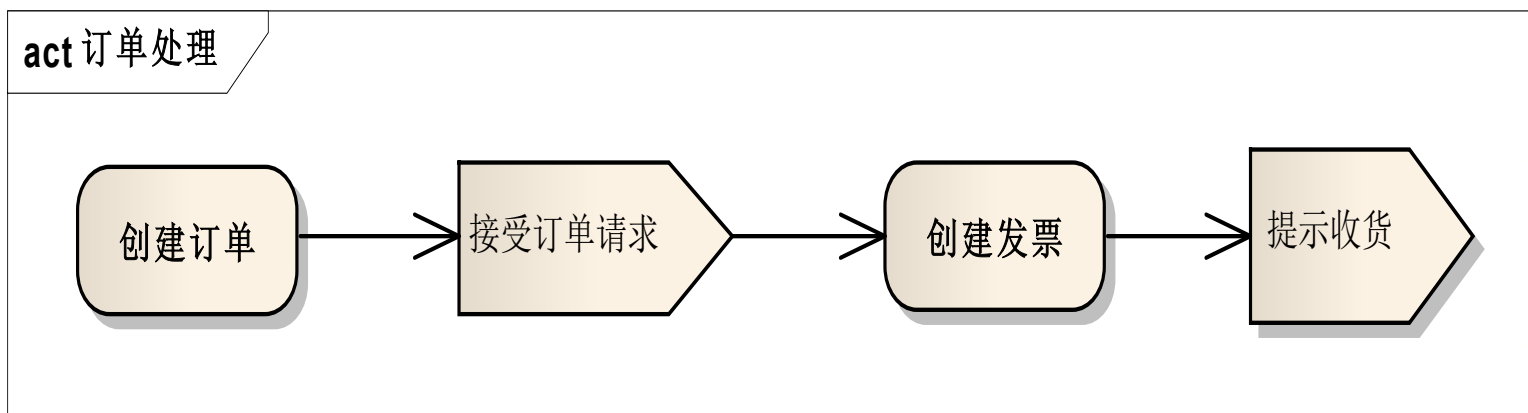
发送信号



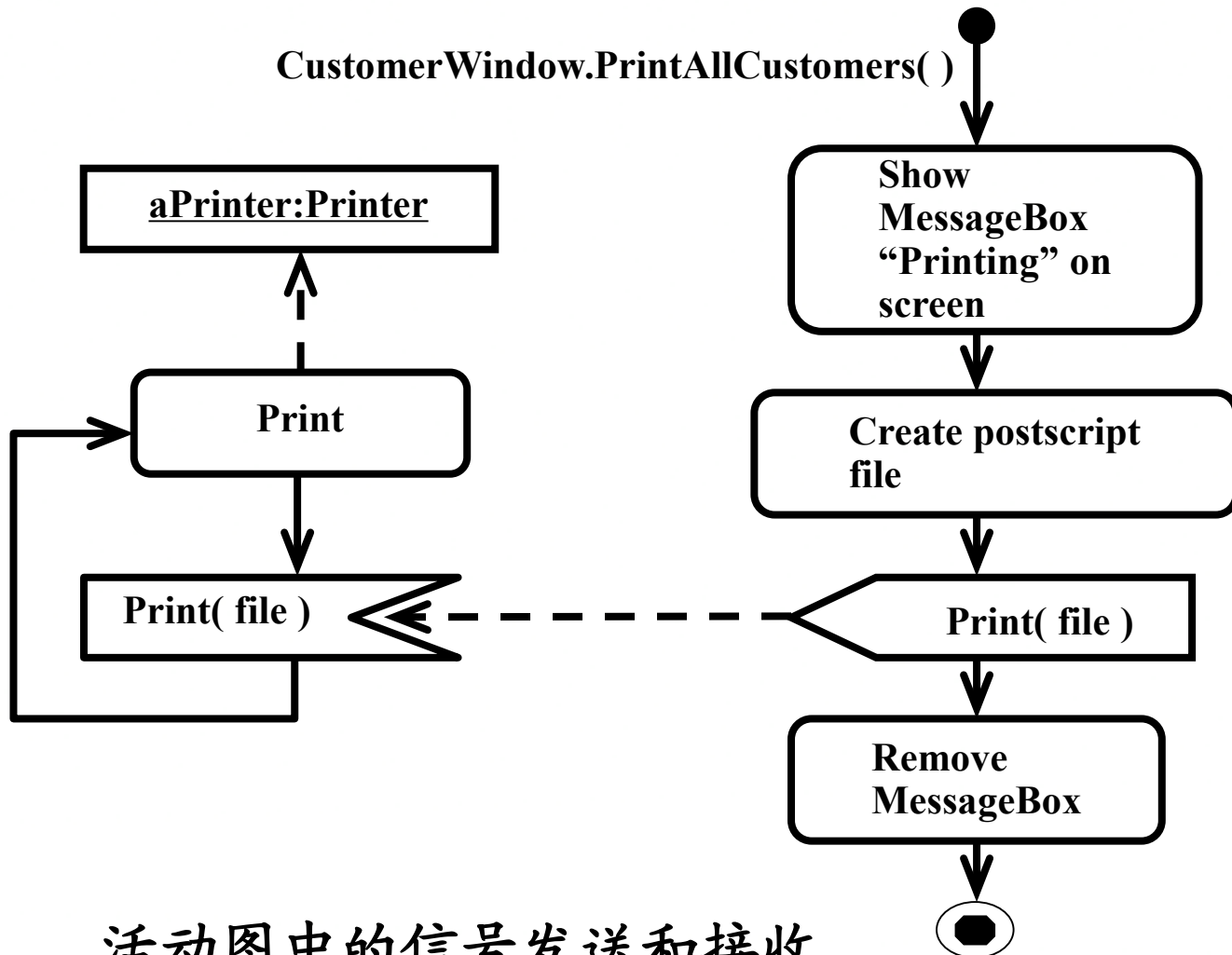
接收信号

活动图分类

- **时间信号**：时间信号是用来表示随着时间的流逝而自动发出的信号，时间信号表示，当时间到达某个特定的时刻时，就会触发时间事件，例如每天**10**点时，闹钟开始响铃，**10**点钟发出响铃的信号就是时间信号。
- **发送信号**：也就是发出一个异步消息，对于发送者而言，就是发送信号；对于接收到这种消息的目标而言，就是“接收信号”。
- **接收信号**：就是接收者收到的一个外部信号。



活动图分类



活动图中的信号发送和接收

活动图分类

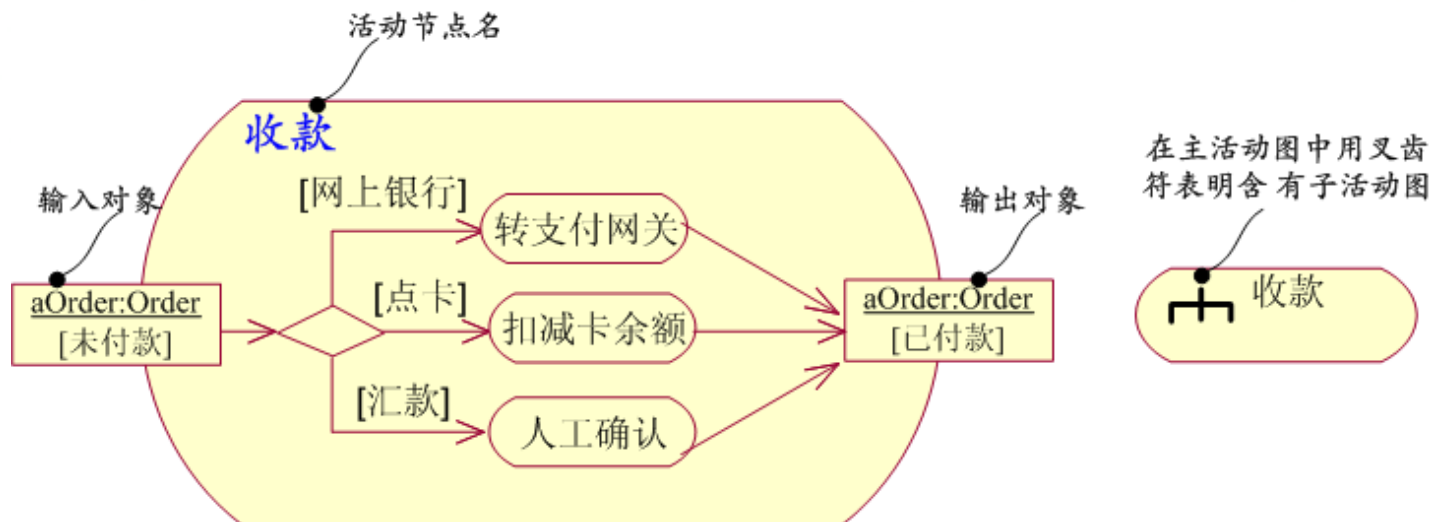
- 活动图中标识时间信号

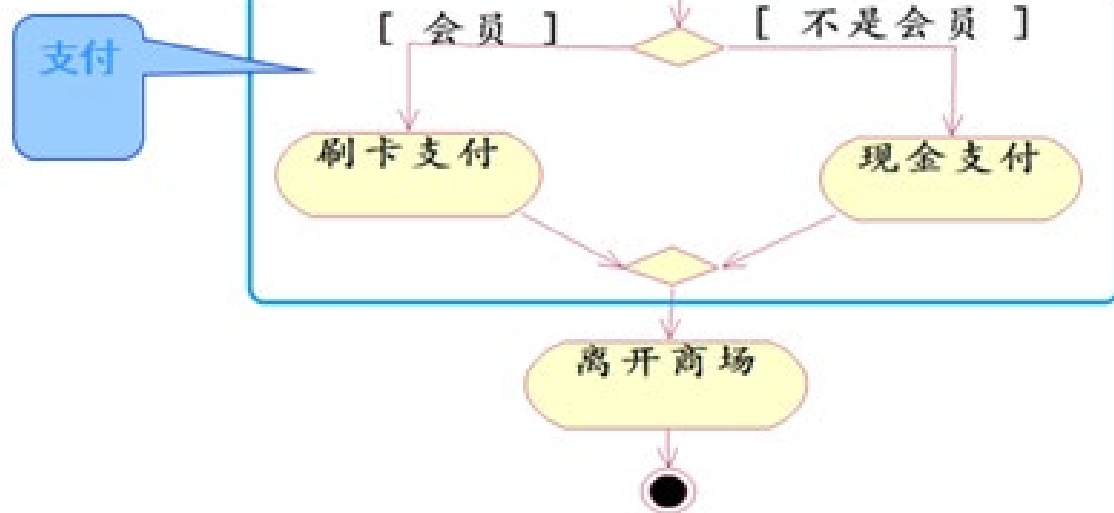
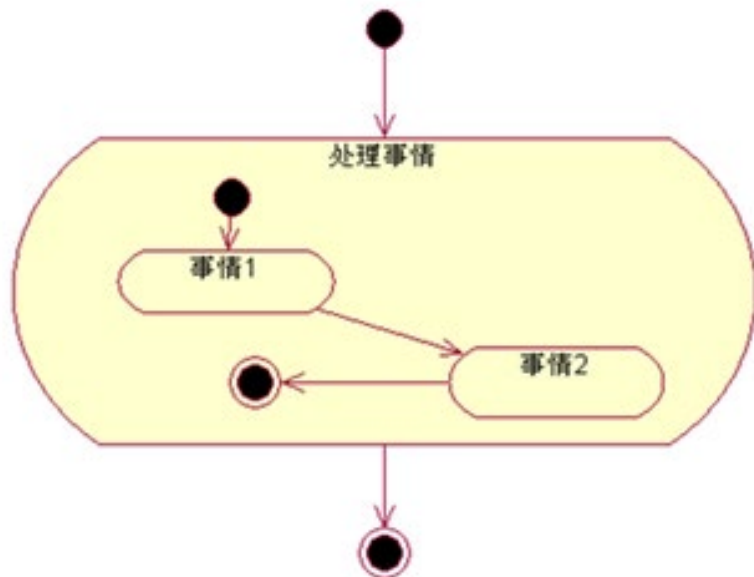


活动图分类

• (5) 嵌套活动图

- 如果一个活动图又包含了子活动图，则称这种图为**嵌套活动图**(也称为主活动图)。当一个活动图很复杂，我们可以把其中的一组相关活动看作一个子活动图，这时，在嵌套活动图中，用子活动图的简图代替子活动图。我们可以将子图单独放在一个图中详细说明它的活动，然后，在嵌套活动图引用子活动图。下图是一个嵌套活动图，其中的收款(活动)又是一个子活动图，子活动图的详细说明如下图所示。





构建活动图

- 画活动图的建议：
 - 首要思考本活动图要表达什么内容，表达的重点是什么，根据这一点来确定合适的活动粒度。
 - 可先用比较大粒度的活动，目的是先弄清楚流程总体框架。
 - 流程大体确定后，逐步细化。
 - 需要重点说明的部分，活动粒度应该足够细。

构建活动图

绘制活动图几个关键步骤:

(1) 如希望在活动图中标识出活动的**实施者**, 我们就应该采用标识**泳道**的活动图, 并在绘制活动图前, 先找出活动的执行者, 然后找出每个执行者参与的活动。

(2) 在描述活动节点关系时, **最大限度的**采用**分支**, **分叉**和**汇合**等基本的建模元素来描述活动控制流程。

(3) 如果希望标识出活动节点执行前后对象的创建、销毁情况, 以及对象的状态变化情况, 那么, 在绘制活动图时, 应该标识**对象流**, 以及**对象的状态变化**。

(4) 如果希望标识活动图中更详细的信息, 就应该在活动图中, 利用一些高级的建模元素 (如**顺序活动图**、**并发活动图**、在活动图中标识**发送信号**与**接收信号**、用**扩展区**来标识活动的循环执行等等)。

构建活动图



1、对**工作流程**建模（**业务分析阶段**）

用活动图对业务流程建模时，活动图中，每一条泳道表示一个职责单位（可以是个人，也可以是一个部门），每个泳道的执行者（或职责人）体现了职能部门的工作职责、业务范围、部门之间的交互关系。

构建活动图

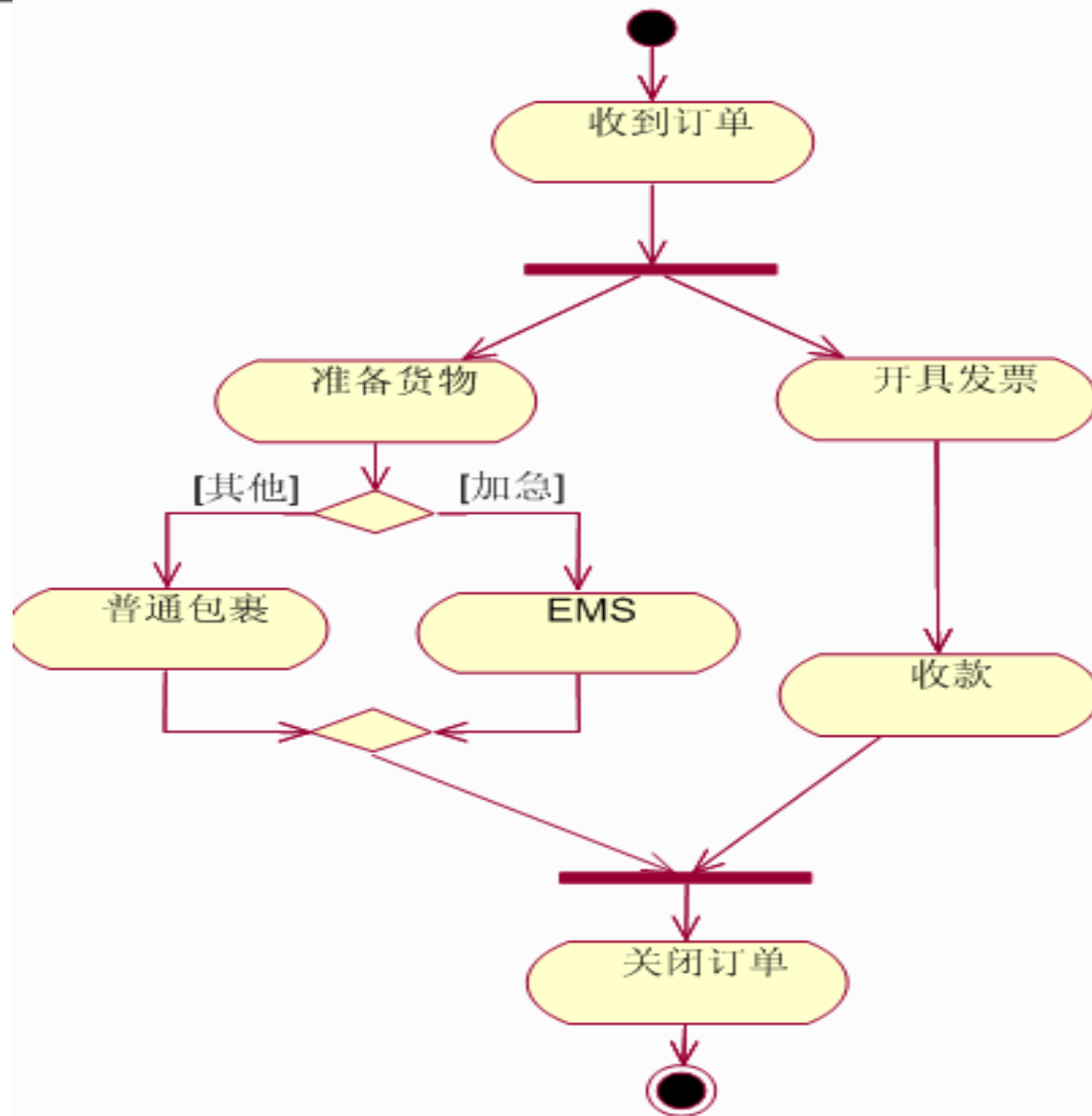


2 对**操作流程**建模（**系统分析和设计阶段**）

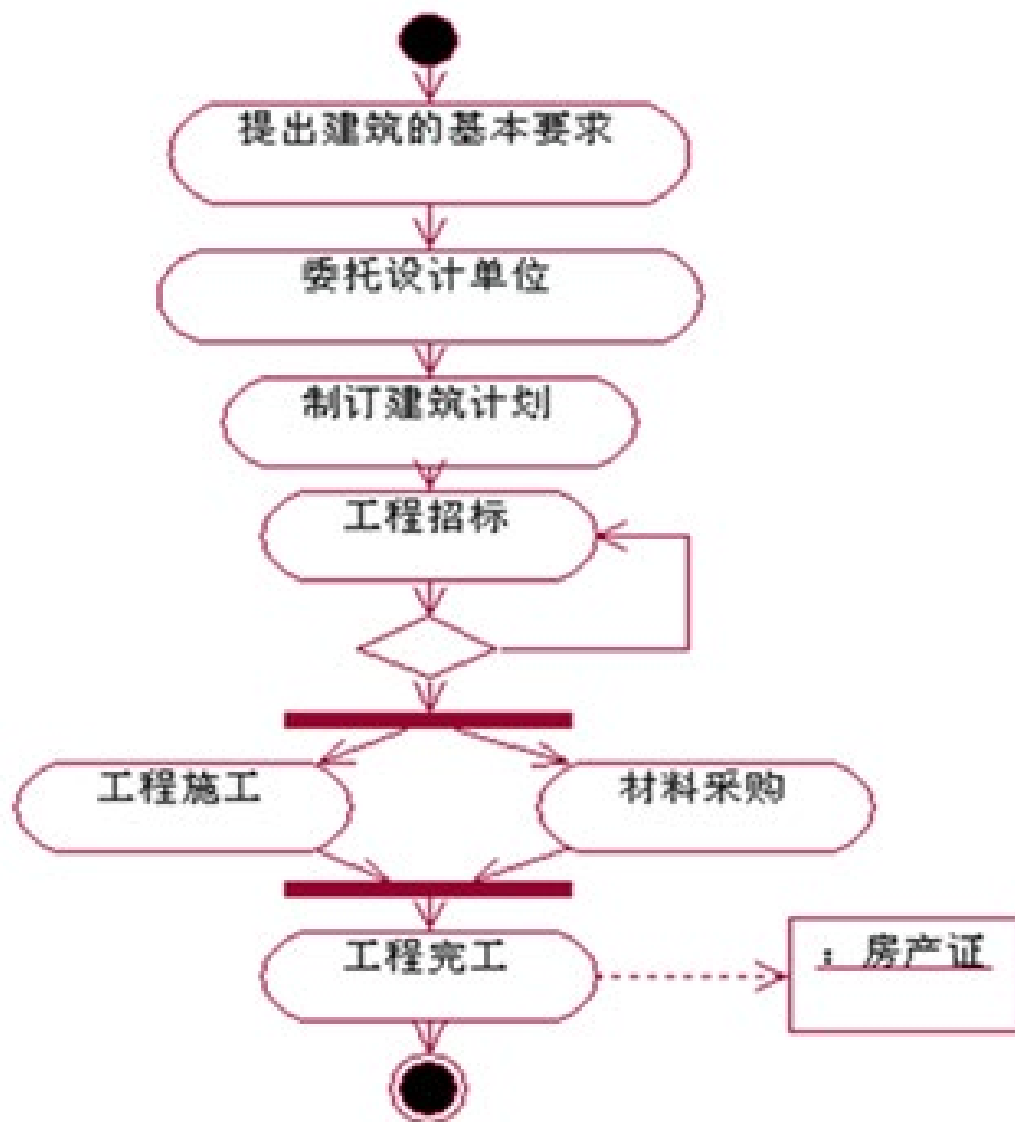
在系统设计期间，我们用活动图对对象的职责进行建模，**每一个对象占据一个泳道，而活动是该对象的成员方法。**

在系统分析设计阶段，采用带泳道的活动图的情况较少，因为顺序图会更好体现对象间的交互关系。活动图更适合于对其流程进行概述，最常用的场景是通过活动图对**用例描述**中的事件流进行建模。当用例的事件流较复杂，分支较多时，一张清晰明了的活动图能够帮助开发人员更好地理解程序的逻辑。

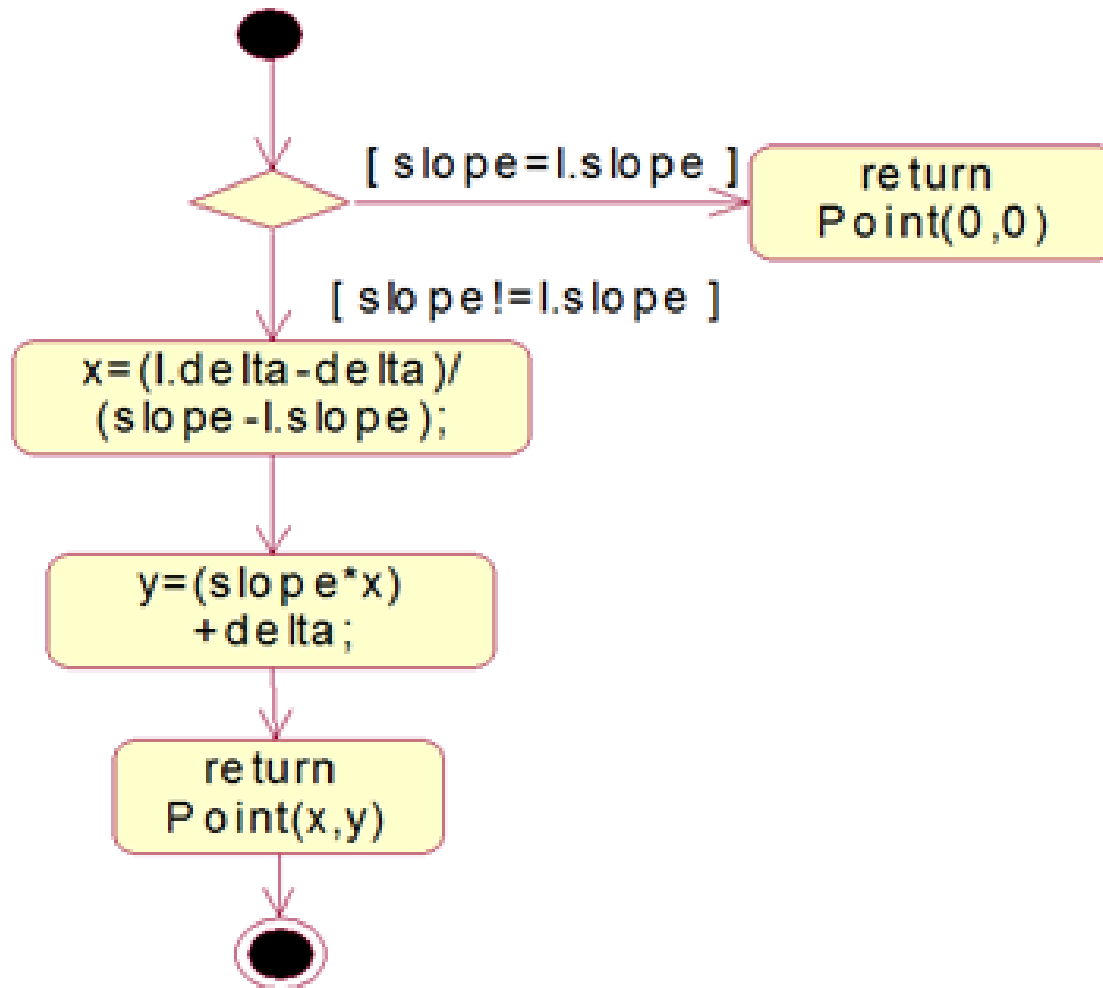
构建活动图-对系统 workflows 的建模



构建活动图-对工程组织过程建模



构建活动图-对算法流程建模

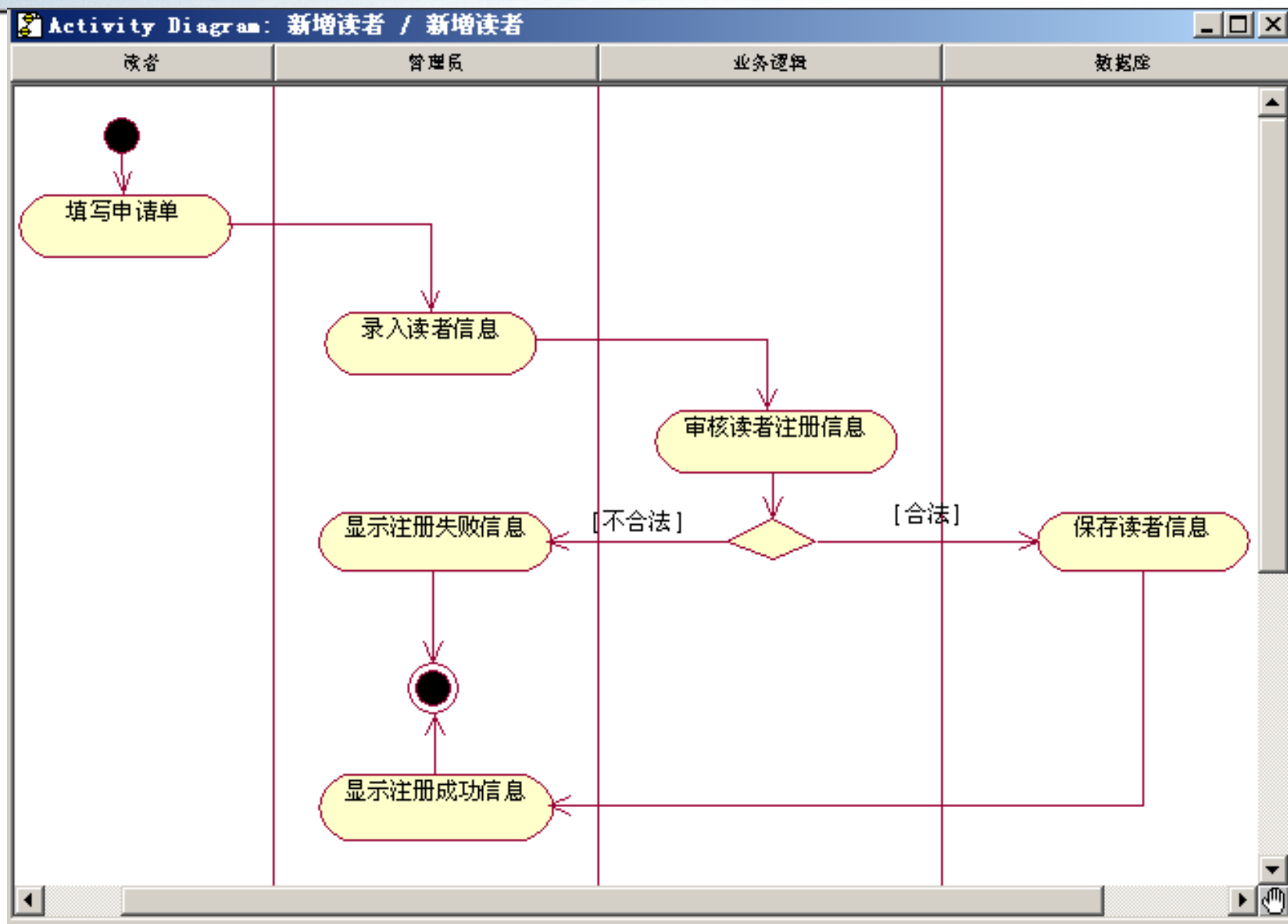


构建活动图——案例1



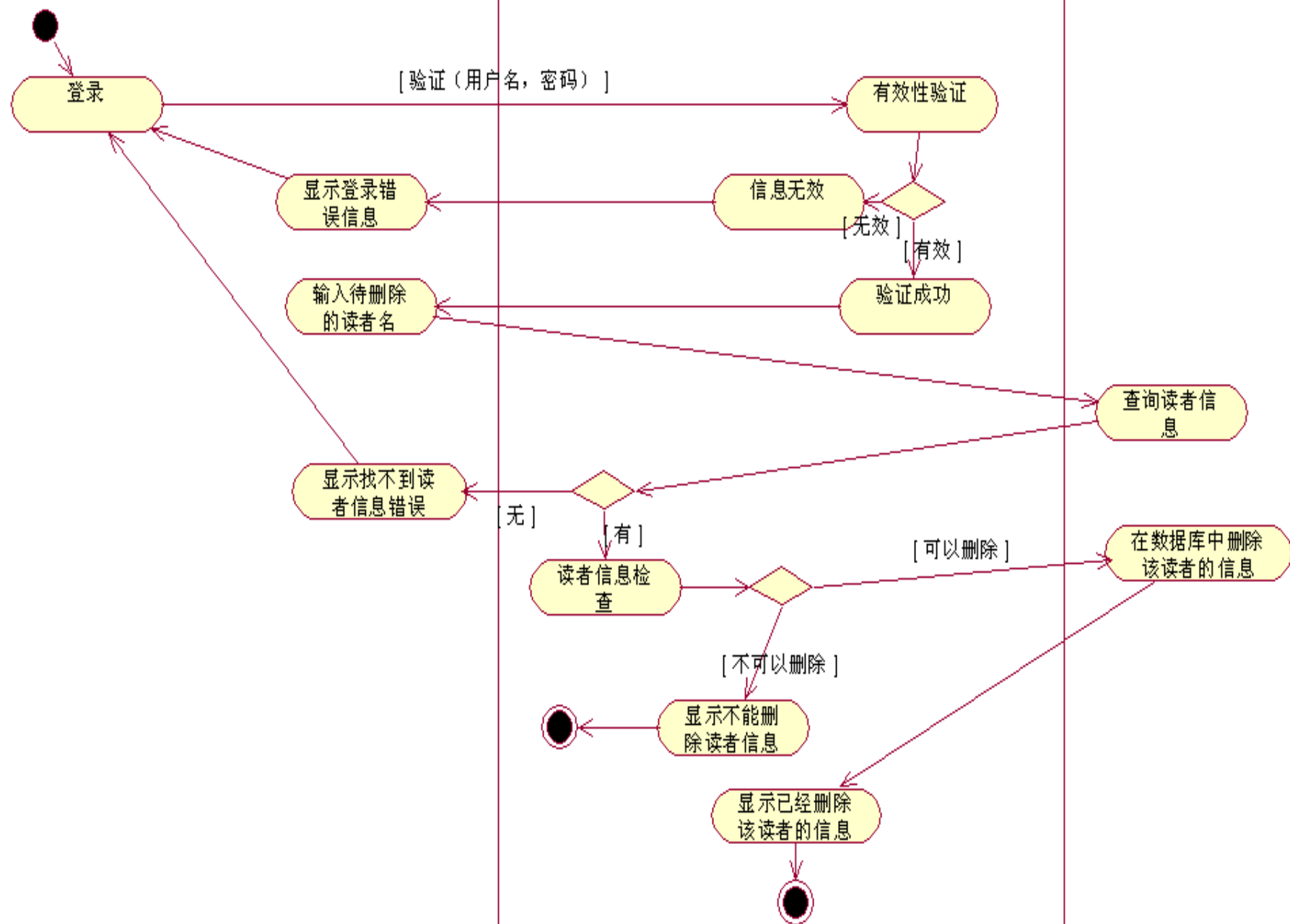
- "新增读者"用例属于读者信息管理中的一个功能，主要用于在系统中增加新的读者信息，其具体的办理流程是：
 - (1) "读者"填写申请表，并交给"图书管理员";
 - (2) "图书管理员"将申请表中的信息通过录入界面，输入到图书管理系统;
 - (3) 系统中的"业务逻辑"组件将判断输入的信息是否合法;
 - (4) 如果不合法则转入步骤(5)，否则转入步骤(6);
 - (5) 显示“显示注册失败信息”，转到(8);
 - (6) 在“数据库”保存用户信息;
 - (7) 显示“显示注册成功信息”;
 - (8) 结束。

构建活动图——案例1



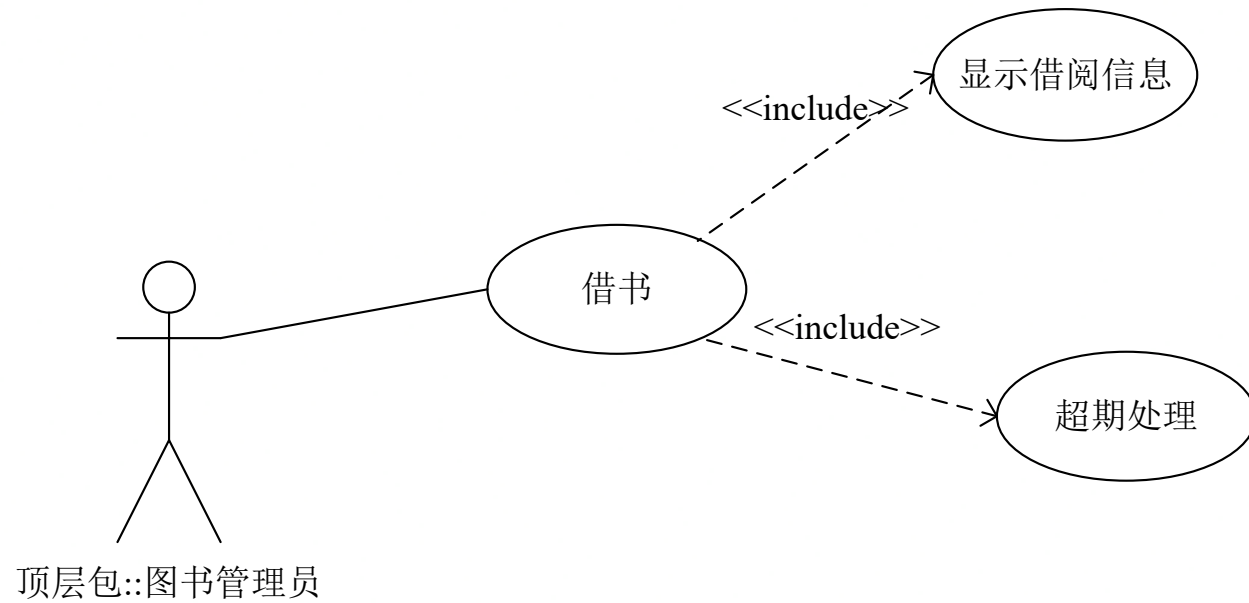
构建活动图——案例1

- 绘制“删除读者信息”用例的活动图。删除读者信息按照以下步骤进行：
 - (1) 管理员在录入界面，输入待删除的读者名；
 - (2) “业务逻辑”组件在数据库中，查找待删除的读者名；
 - (3) 如果不存在，则显示出错信息，返回步骤(1)，如果存在则继续；
 - (4) “业务逻辑”组件判断“待删除的读者”是否可以删除；
 - (5) 如果不可以，则显示出错信息，返回步骤(8)，如果可以则继续；
 - (6) 在数据库中，删除相关信息；
 - (7) 显示删除成功信息；
 - (8) 结束。

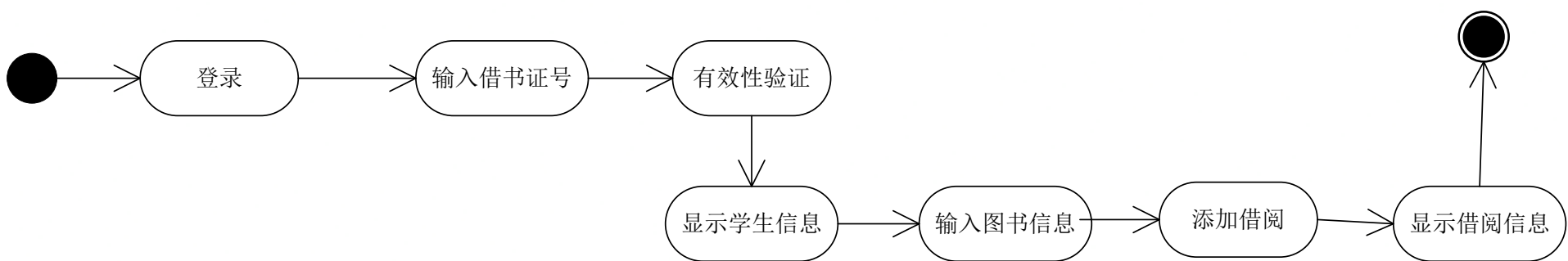


借书操作活动图

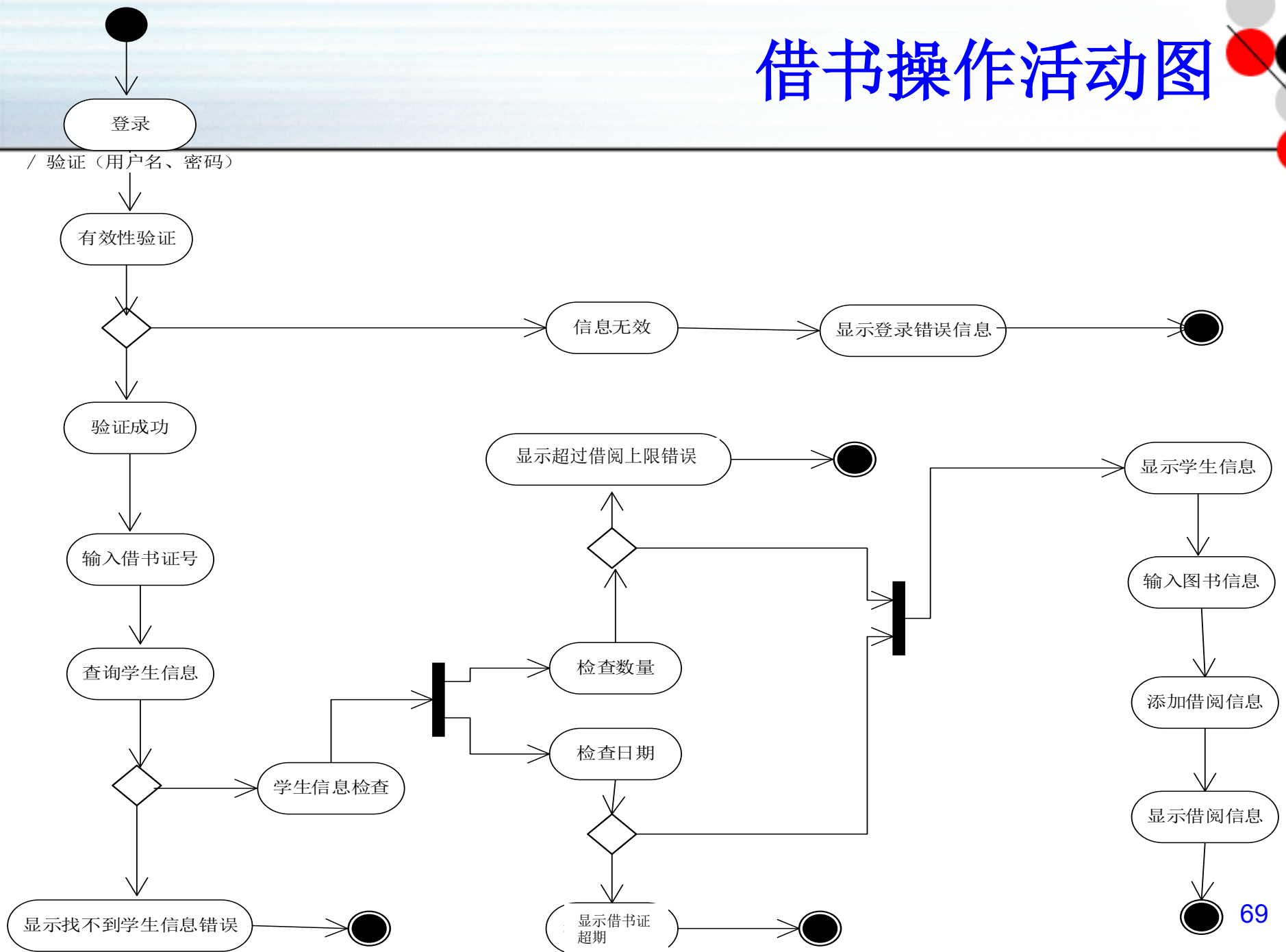
- 借书用例



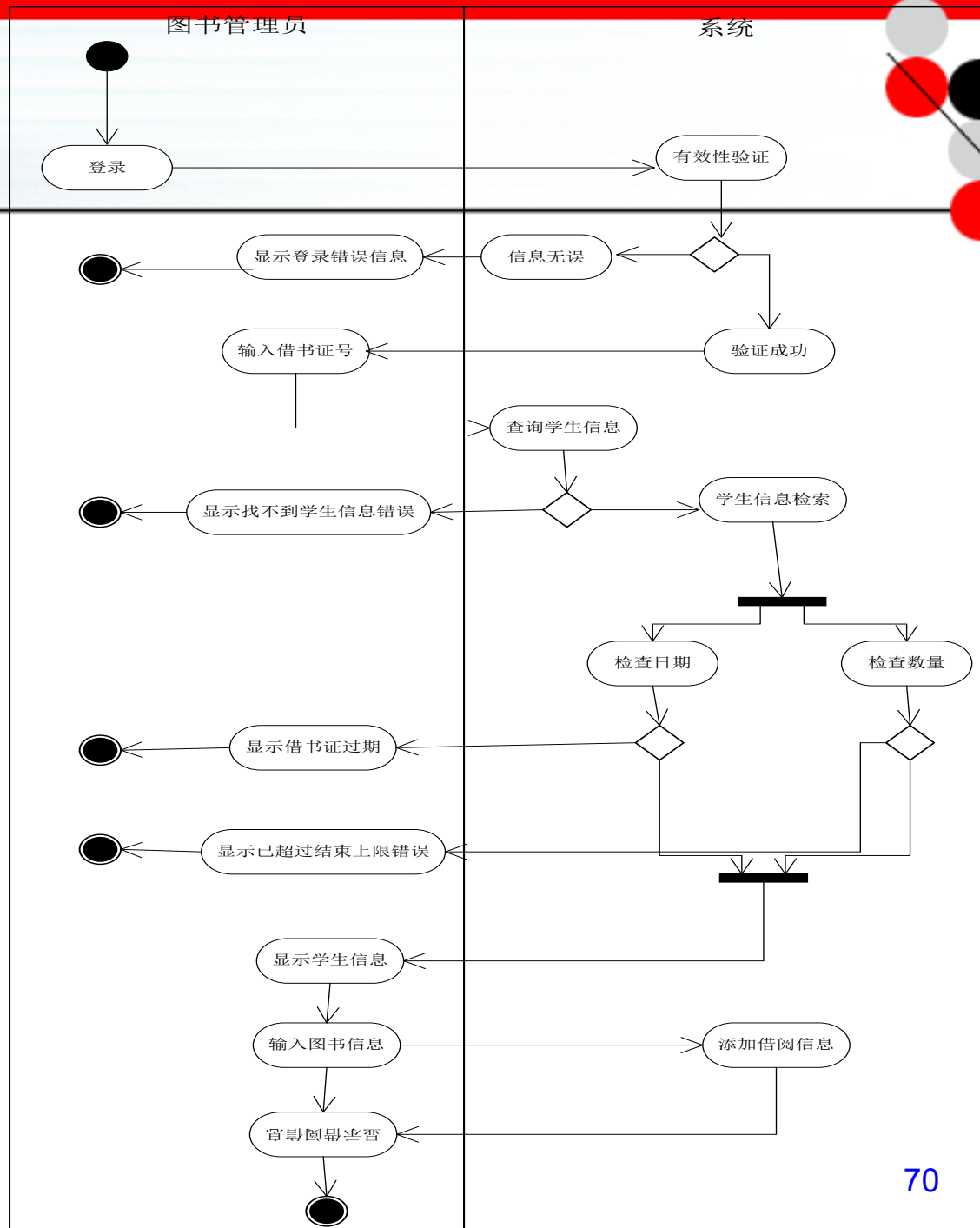
借书操作活动图



借书操作活动图

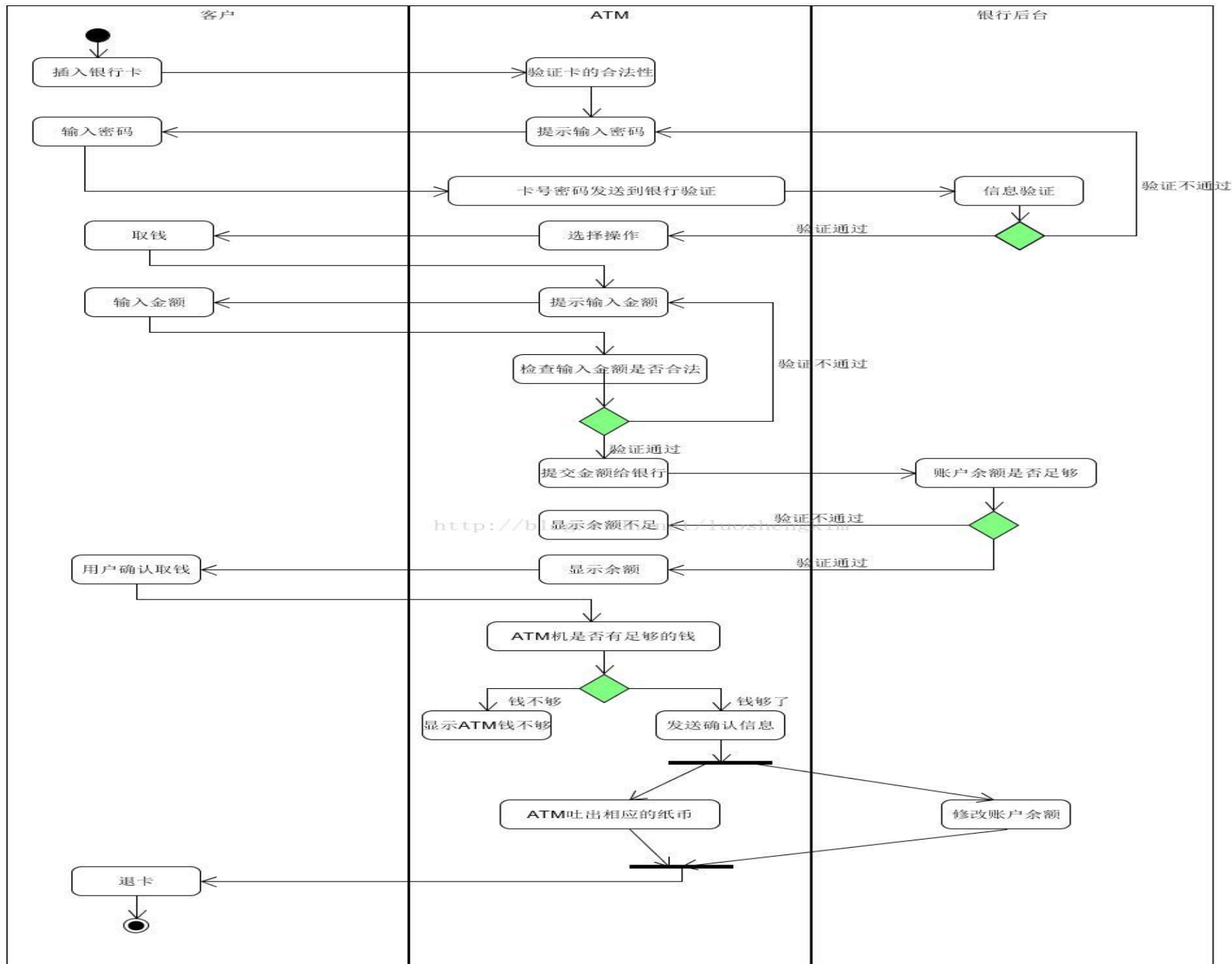


借书操作活动图



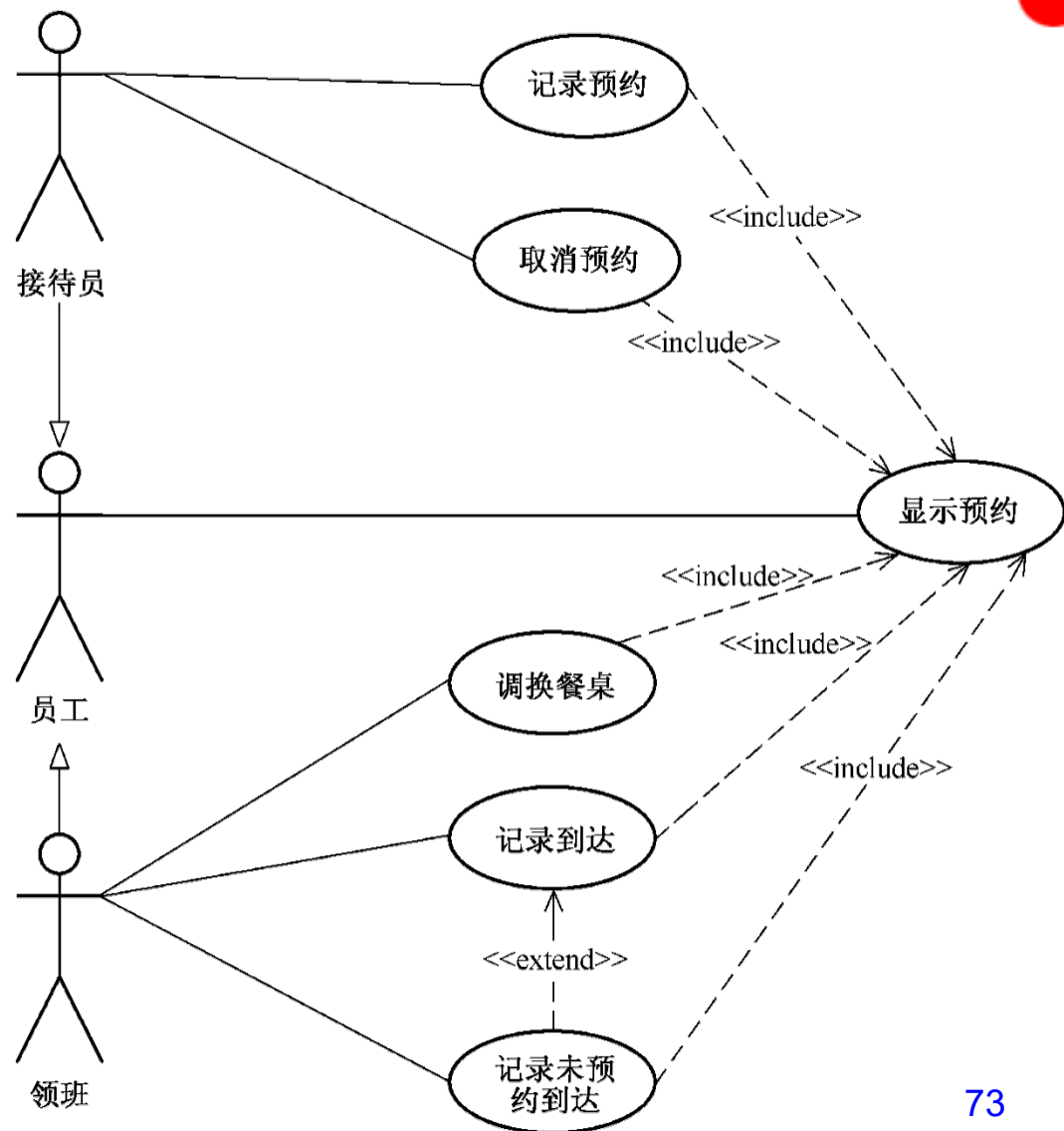
构建活动图——案例2

- 选择你熟悉银行的**ATM**机器，仔细研究观察“人”“机器”对话交互，实现取钱功能的过程。（需要考虑**ATM**钱箱缺币、账号余额不足等意外情况）
- 用活动图画出用户取钱的过程。



构建活动图——案例3

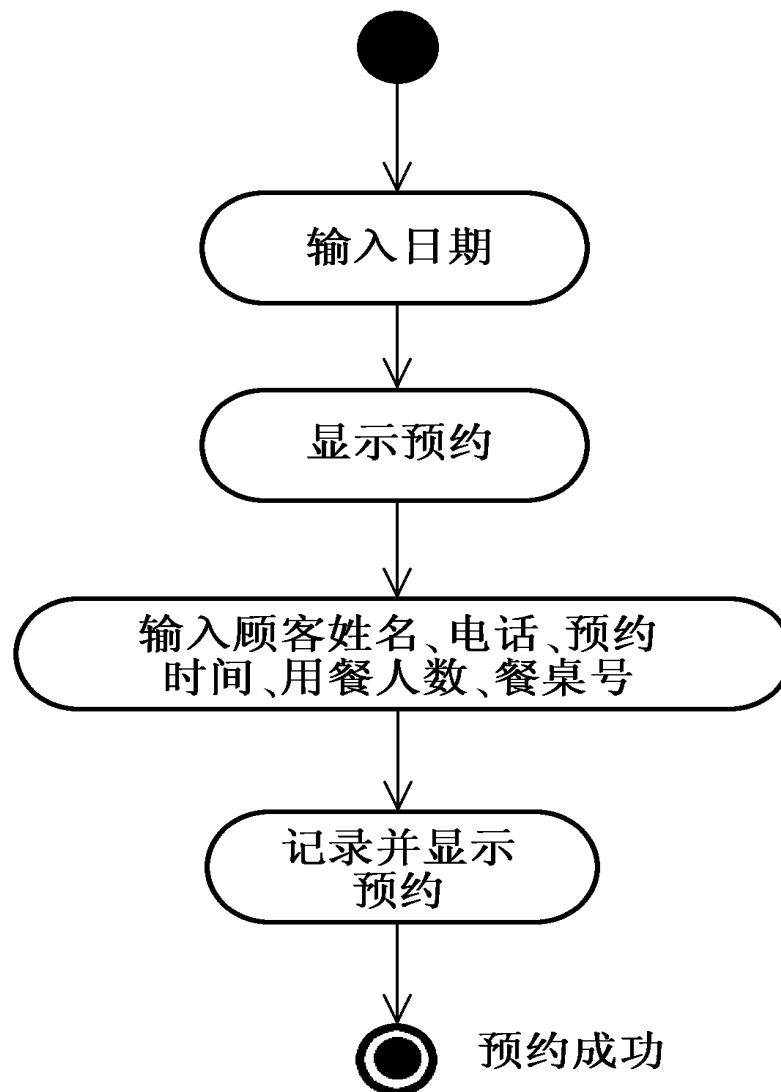
“餐馆订餐”系统的用例图



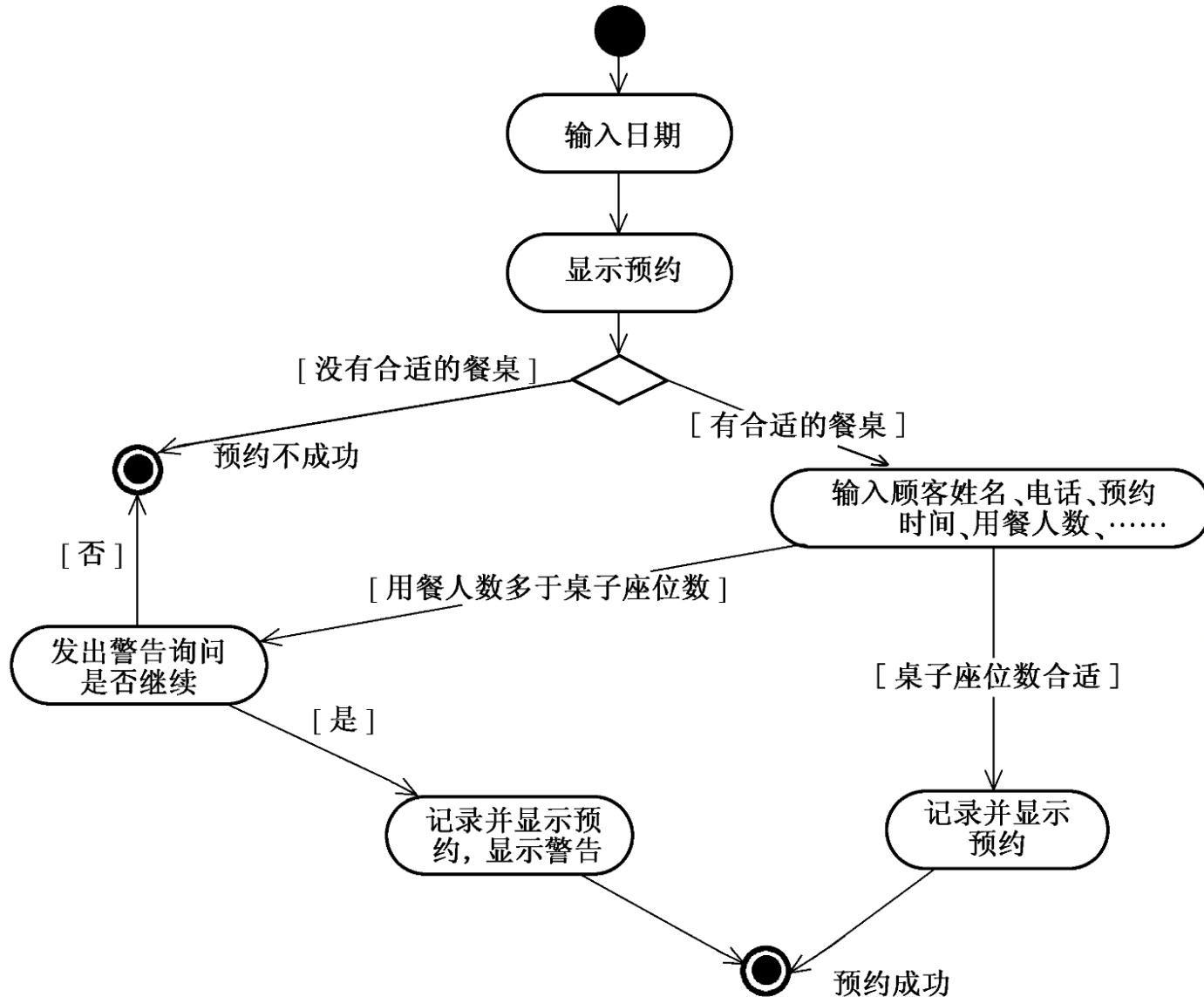
“记录预约”用例的事件路径如下：

1. 接待员输入要预约的日期
2. 系统显示该日的预约
3. 有一张合适的餐桌可以使用，接待员输入顾客的姓名和电话号码、预约的时间、用餐人数和餐桌号
 - 3a 没有合适的餐桌可以使用
 - 3a1 用例终止
4. 系统记录并显示该预约
 - 4a 输入的预约人数多于餐桌能容纳的人数
 - 4a1 系统发出一个警告信息，询问用户是否要继续预约
 - 4a1a 如回答“否”，用例将不进行预约而终止
 - 4a1b 如回答“是”，预约将被输入，并附有一个警告标志

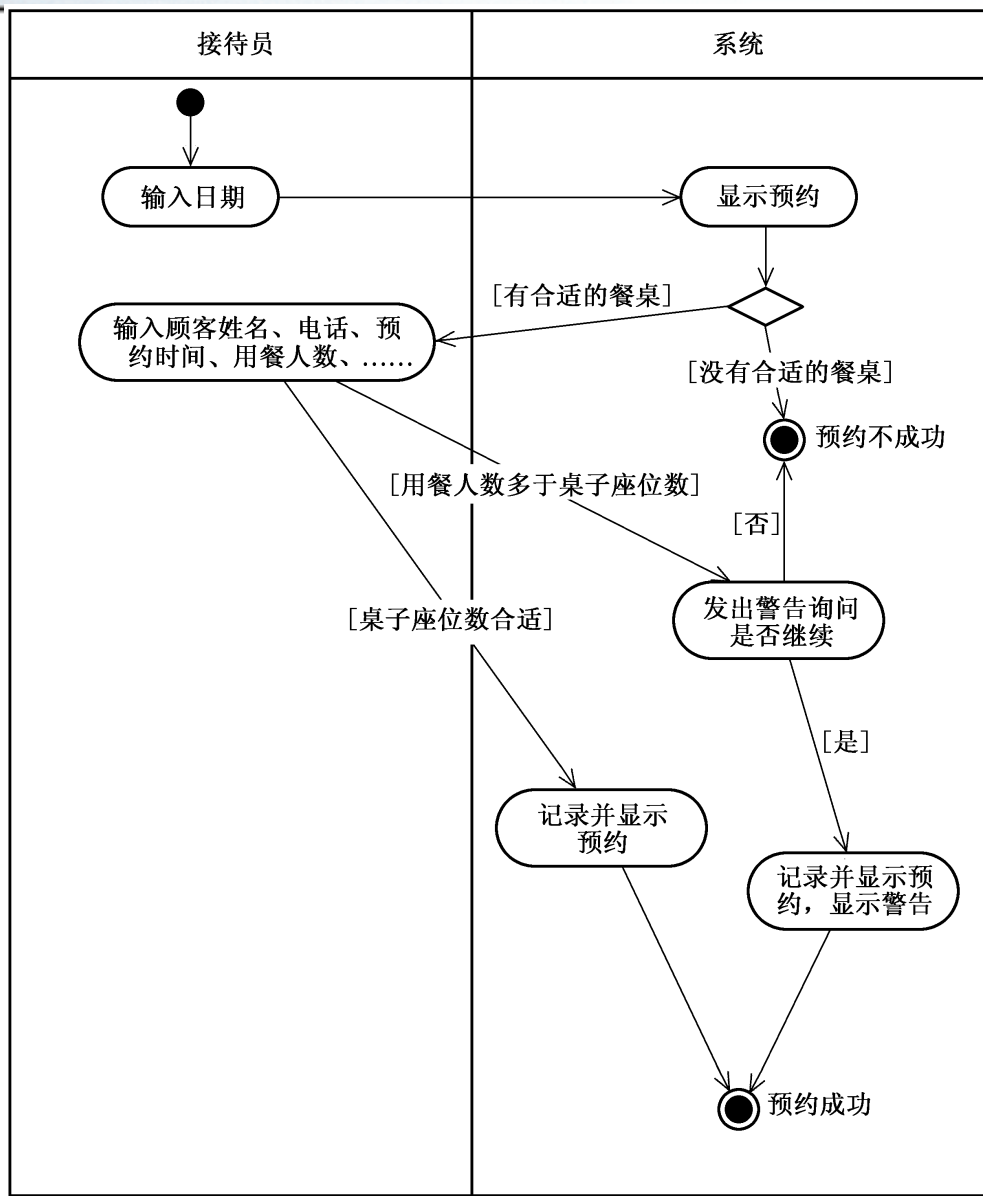
建模主事件流



建模扩展事件流



划分游泳道后的活动图



活动图的优缺点

活动图的**优点**:

最适合支持并行行为，而且也是支持多线程编程的有力工具。

活动图的**缺点**:

很难清晰地描述动作与对象之间的关系。