第9讲　UML概要设计建模

网络空间安全学院

芦效峰

活动图

一、定义

活动图是一种用于描述系统行为的模型视图，它可用来描述过程（业务过程、工作流、事件流等）中的活动及其迁移。简单地讲，活动图是“面向对象的流程图”。

活动图能够附加在如下建模元素中以描述该元素的行为。

- 用例
- 类
- 接口
- 组件
- 节点

二、活动图的主要应用

1．描述用例的行为

活动图对用例描述尤其有用，它可建模用例的工作流，显示用例内部和用例之间的路径；它也可以向读者说明需要满足什么条件用例才会有效，以及用例完成后系统保留的条件或者状态。 。

2．理解工作流程

活动图对理解业务处理过程十分有用。可以画出描述业务工作流的活动图与领域专家进行交流，明确业务处理操作是如何进行的，将会有怎样的变化。

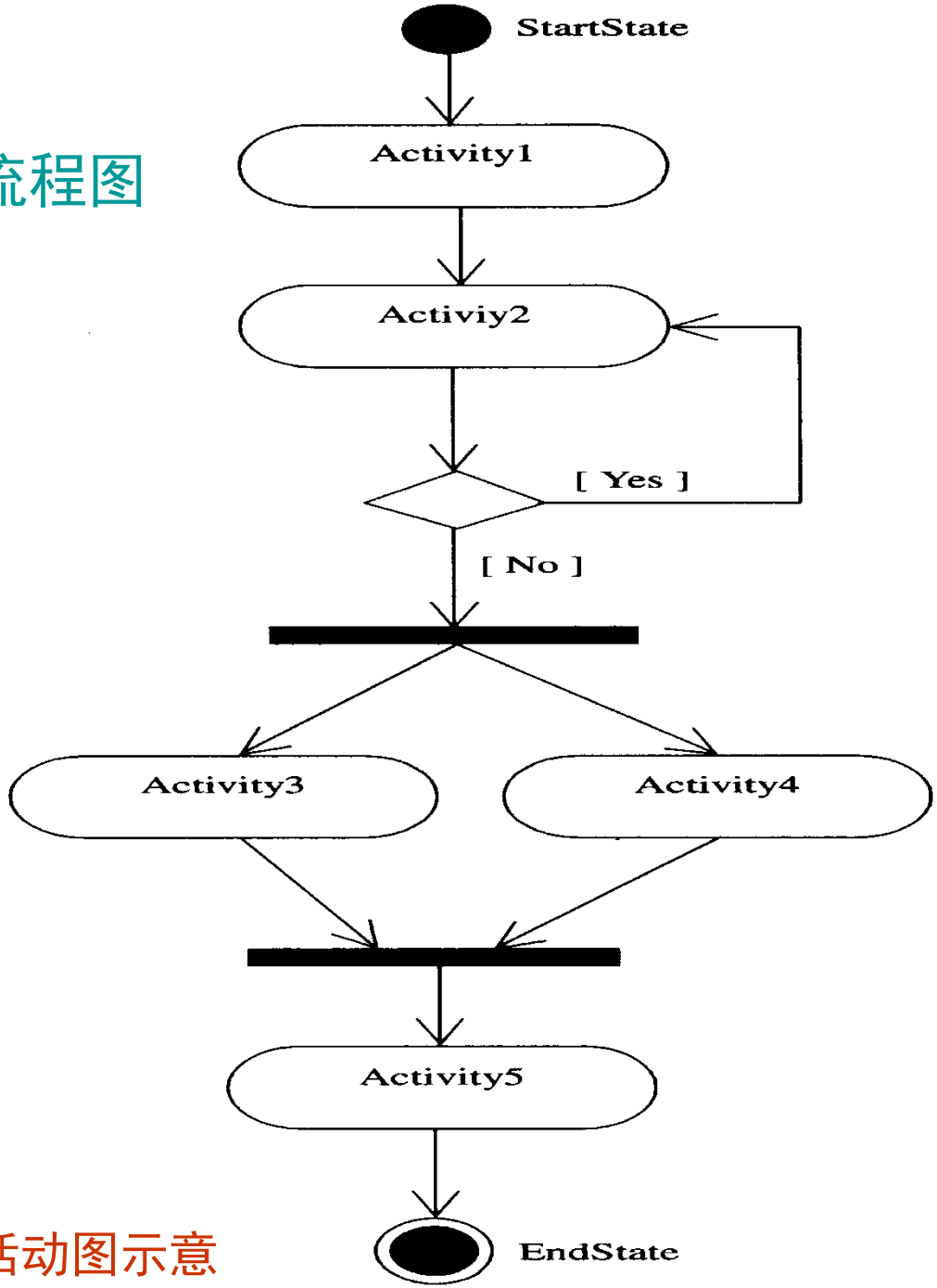
3．描述复杂过程的算法

在这种情况下使用的活动图不过是UML版的程序流程图，常规的顺序、分支过程在活动图中都能得到充分的表现。

三、活动图的基本元素

活动图保留了许多传统的流程图的特征，它包含如下元素。

- 活动
- 操作
- 状态
- 转移
- 对象流
- 分岔
- 决策
- 联结



活动图示意

1. 活动

活动在活动图中，用来指示要完成某项工作的动作或表示工作流的步骤。其UML标记符是一个带有圆角的矩形，如下图所示。

在确定活动名称时应该恰当地命名，选择准确描述所发生动作的几个词。例如，Save File或者 Create New Document就是比较恰当的活动名称，而Run或者Update 对读者而言是不完整的名称。

2. 操作

可以用操作在活动中增加更多详细的步骤。操作是活动中执行的小步骤。在下列情况下发生：

- 进入活动时发生的操作，标有entry字样。
- 活动进行时发生的操作，直到离开活动，标有do字样。
- 离开活动时发生的操作，标有exit字样。
- 特定事件发生时的操作，标有event字样和事件名。

操作是可选的，但提供的详细信息有助于后面完成系统设计。如果包括操作，则其在活动内显示，不管其属于上面哪一类。下面是带操作的活动例子。

显示可选航班

entry/寻找所选城市/日期的所有航班
entry/确定有空位的航班
do/显示有空位的航班清单
do/加亮显示机票最便宜的航班
event/用户请求机票价信息/显示机票价信息

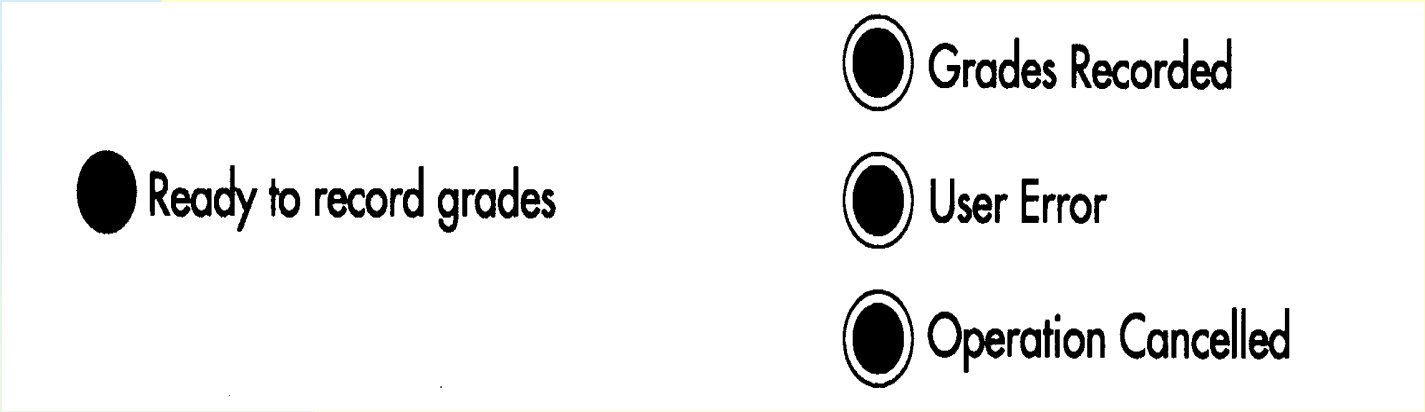
3. 状态

状态的标记符与活动类似，也是带圆角的矩形，但状态的圆角比较小，如下图所示。

状态通常使用一个指示系统当前状态的单词或者短语来标识。例如，Stopped是一个状态，而stop则是一个活动。

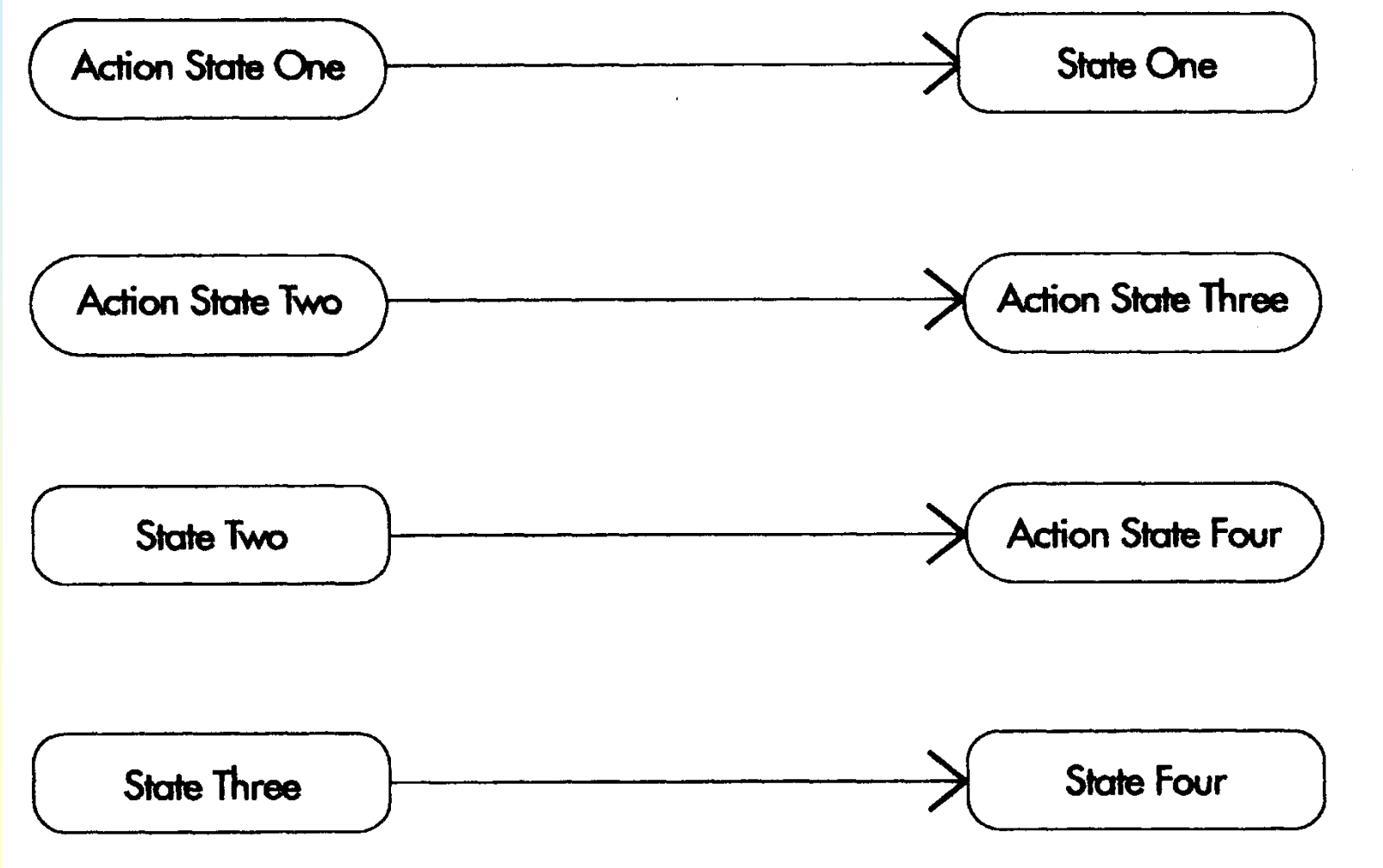
UML包含两个特殊状态，即开始状态和结束状态。开始状态以实心黑点表示，结束状态以带有圆圈的实心黑点表示。

注意：每一个活动图只能有一个开始状态，但是可以有无数个结束状态。如下图所示：



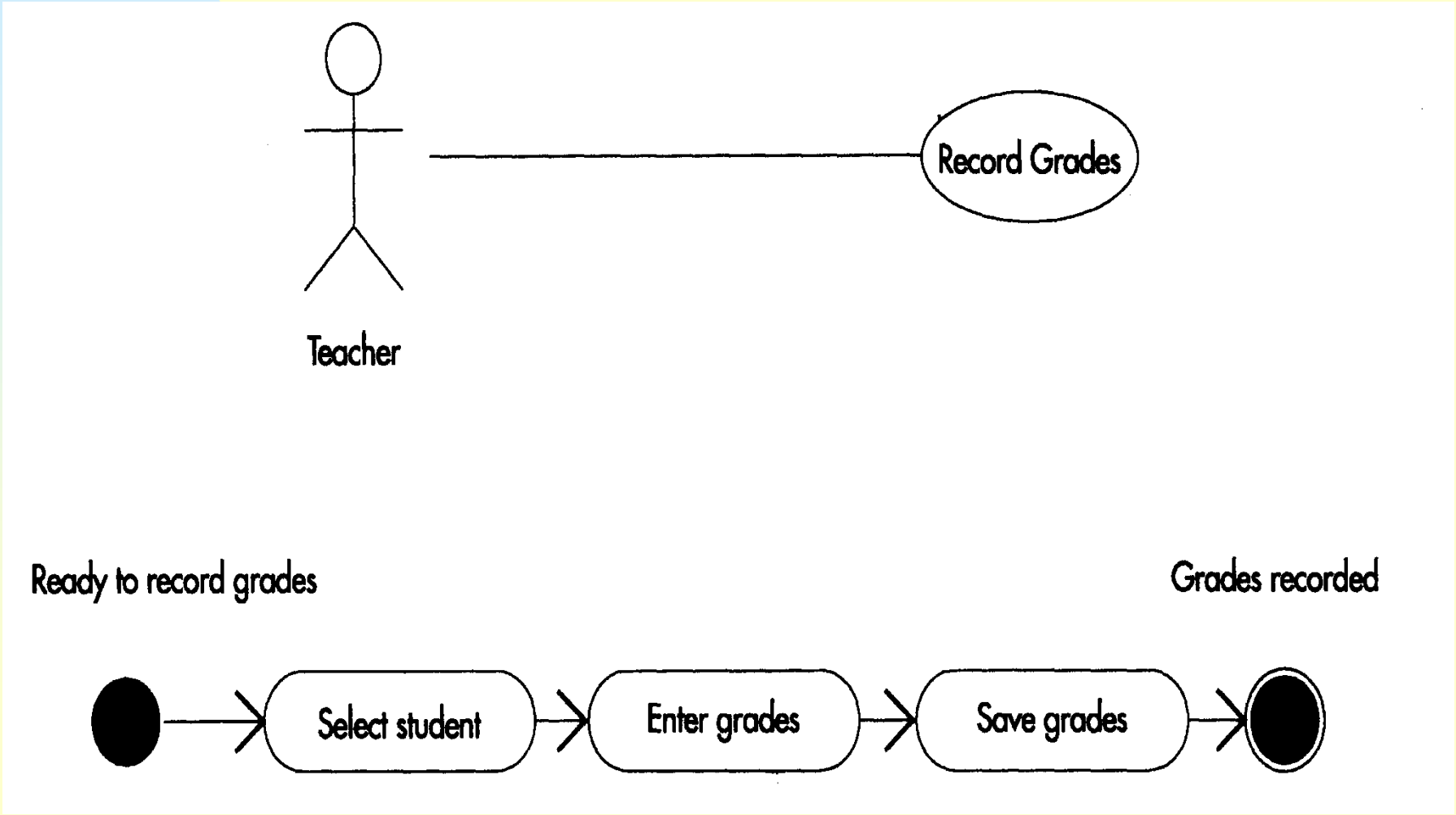
4. 转移

转移用来指示一种状态到另一个状态的控制流。它们可以显示活动之间或者状态之间的控制流。转移的标记符是带开放箭头的实线，如下图所示。



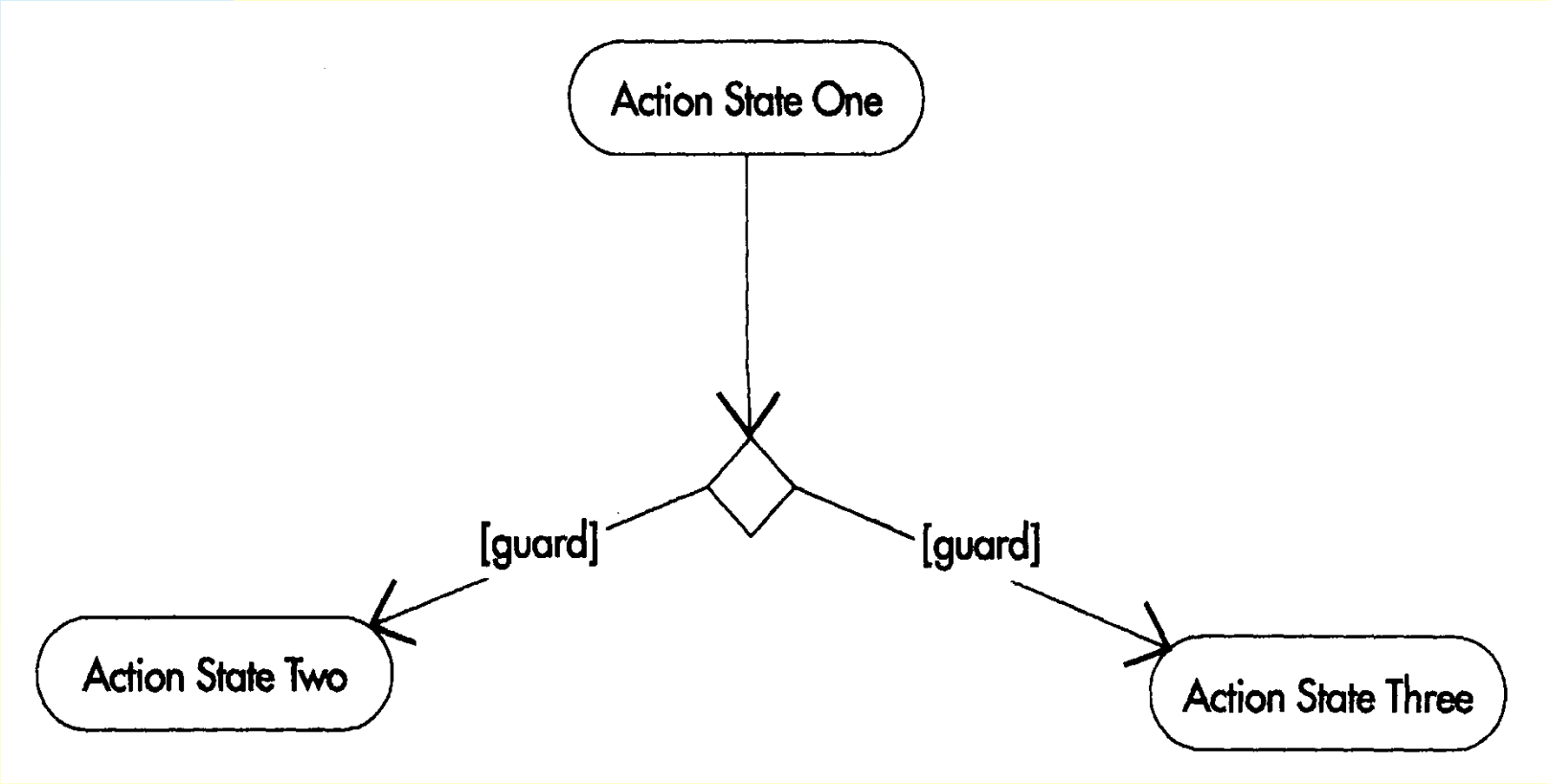
4. 综合运用

现在已经学习了活动图的4种主要标记符，下面综合运用这些标记来生成一个活动图，如下图所示。

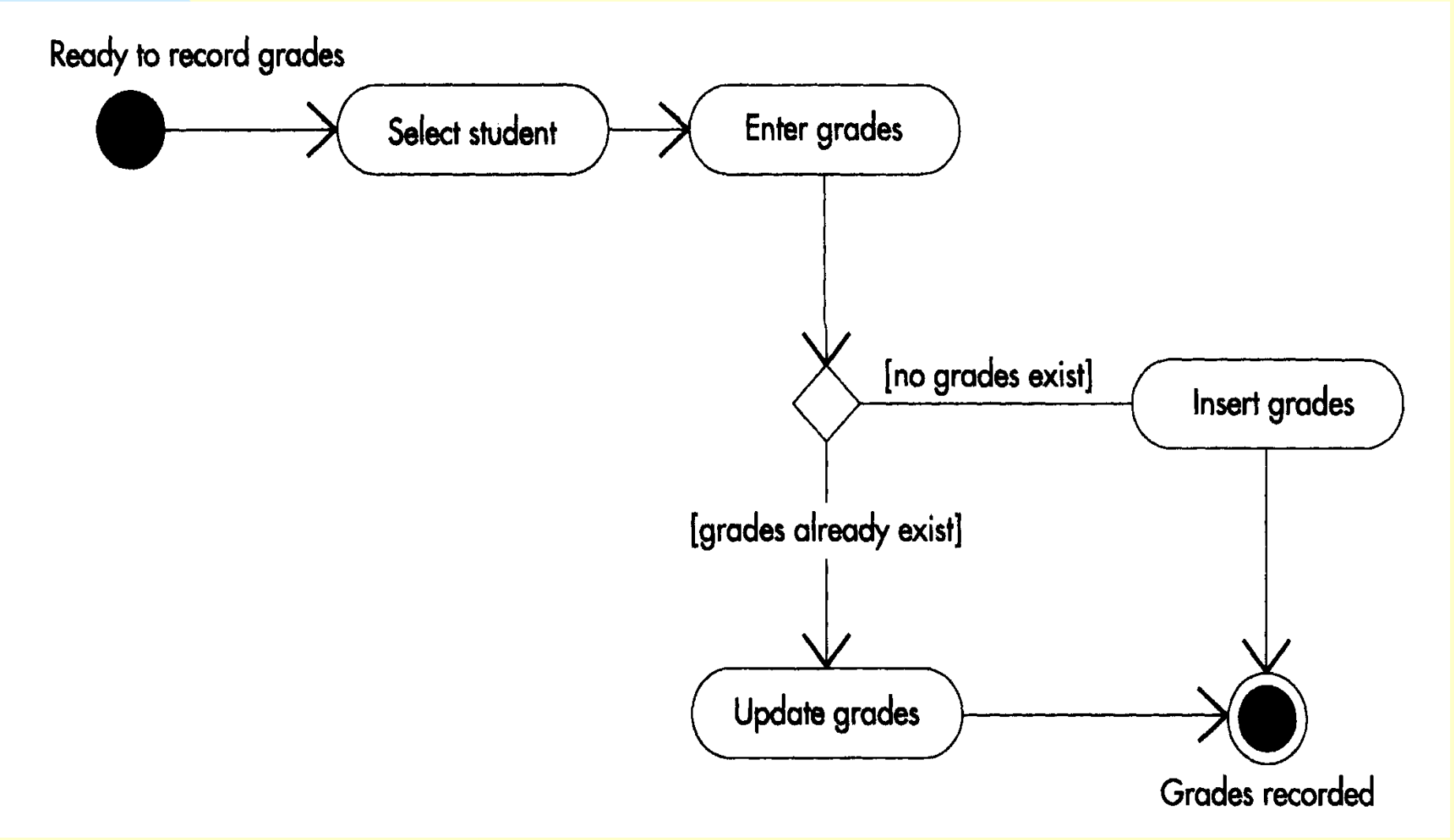


5、 决策

决策是基于判断条件选择控制流继续的方向。决策的UML符号是一个小菱形标记符，然后从这里再按条件控制分支转移到满足条件的活动，如下图所示。

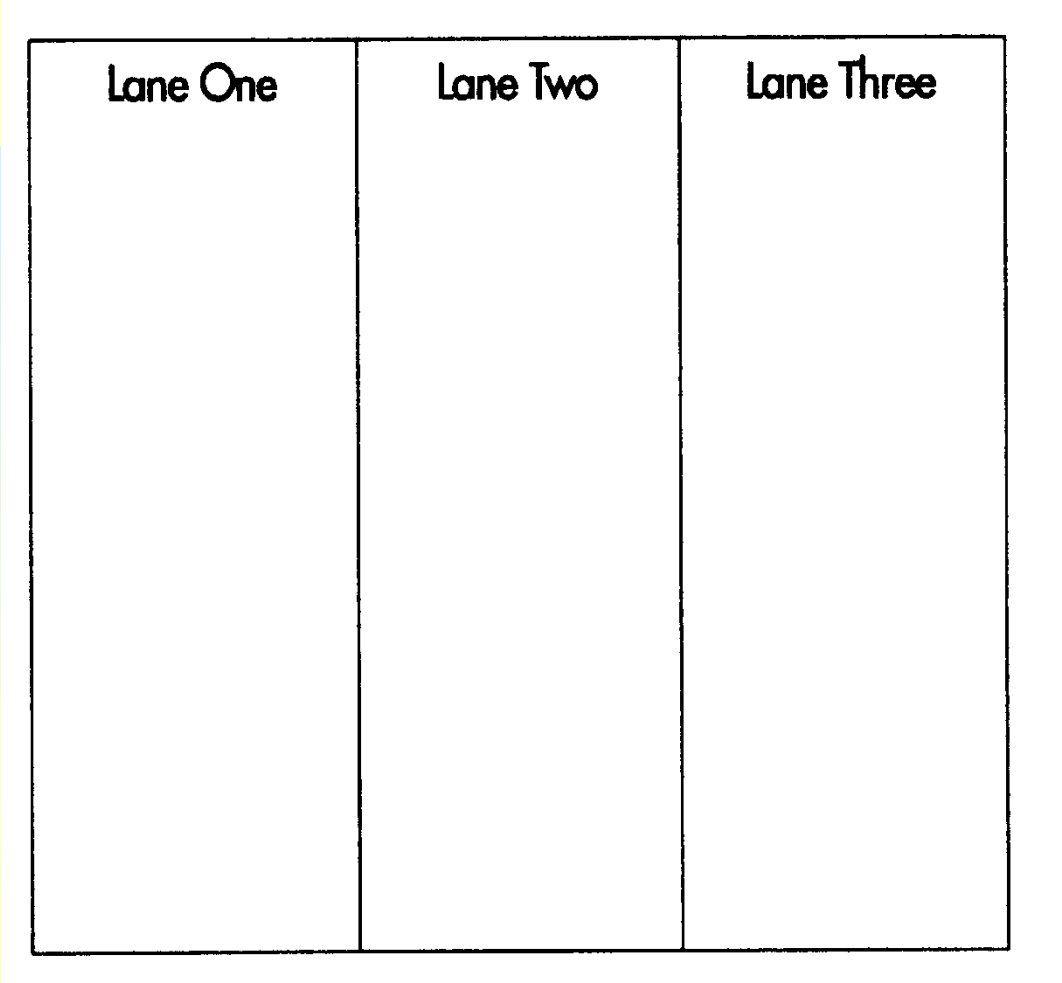


例 描述教师记录学生成绩用例的活动图

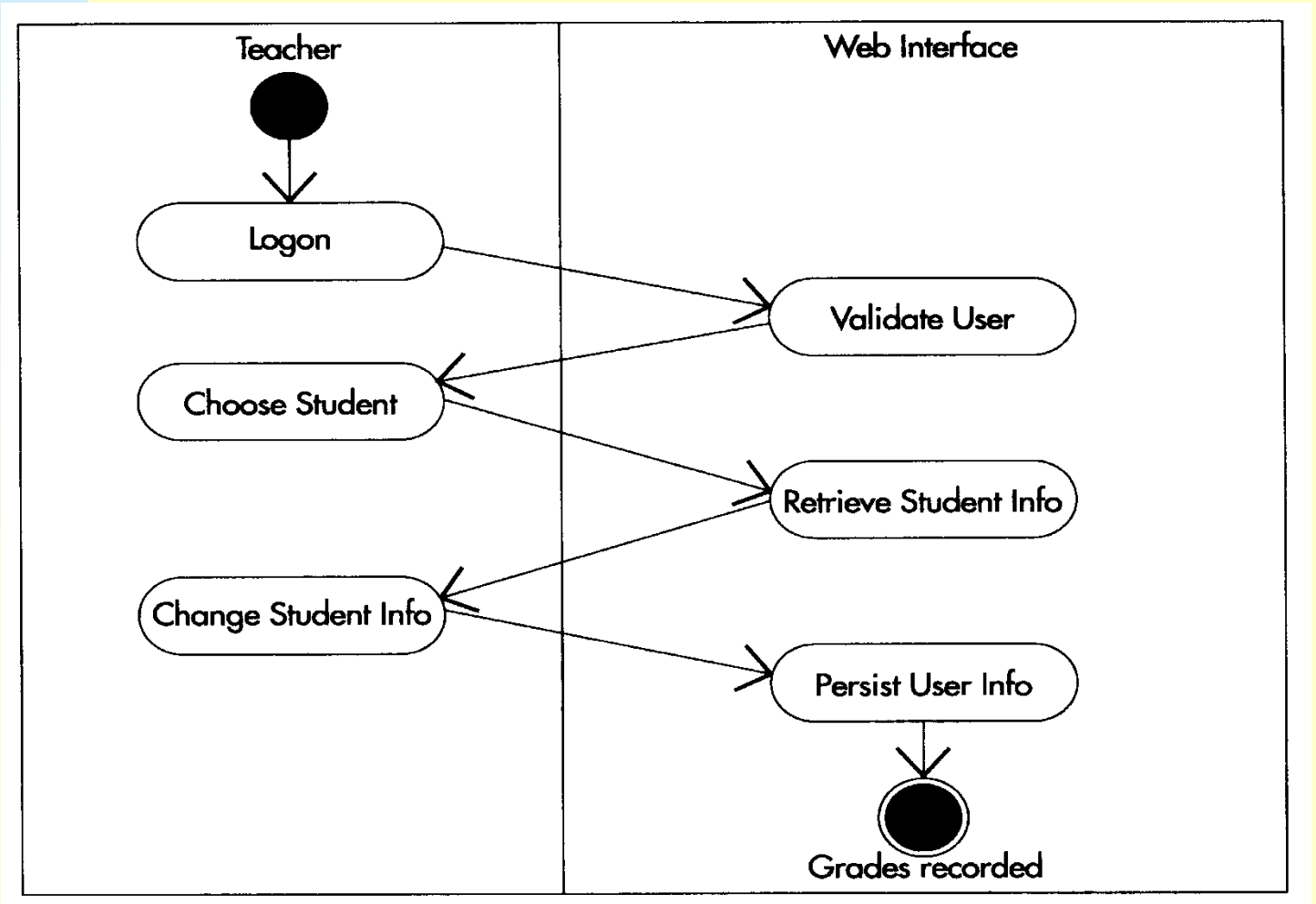


6. 泳道

泳道可以使活动图非常整洁，因为它们在很大程度上增强了活动图的可读性。泳道使用几个大矩形框表示，如下图所示。



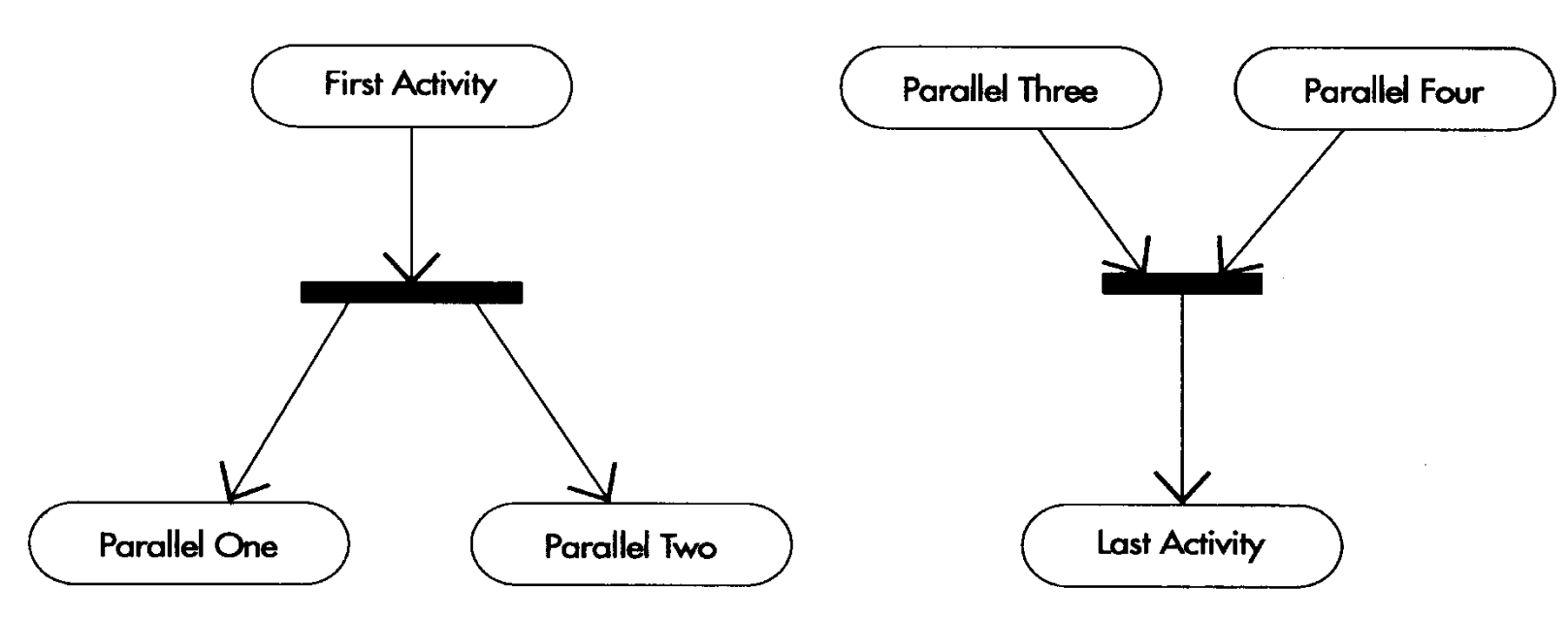
下面示例显示了一个在Teacher和Web Interface泳道之间交叉转移控制流的活动图。如果没有泳道，该活动图就无法说明Teacher 使用了Logon、Choose Student和Change Student Info活动，Web Interface使用Validate User、Retrieve Student Info和Persist User Info活动，如下图所示。

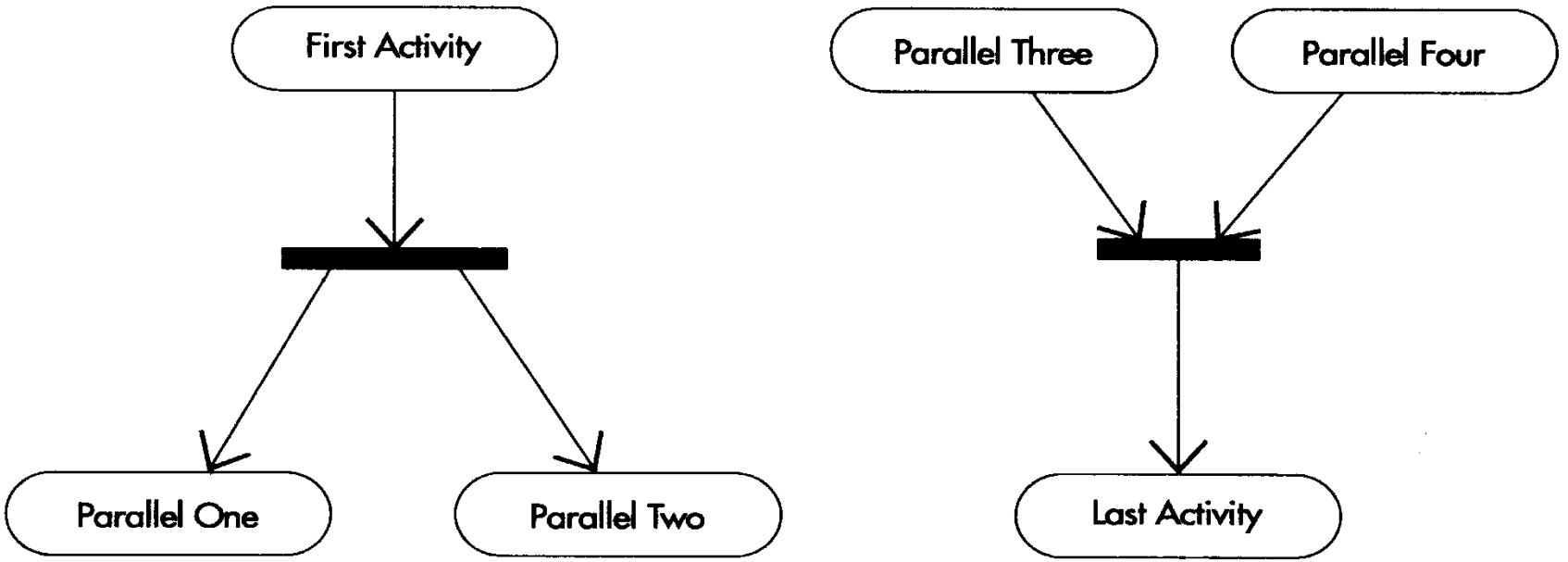


2．分岔和联结

分岔用来表示并行活动的分支处理，联结用来把并行活动的汇集到同步处理。

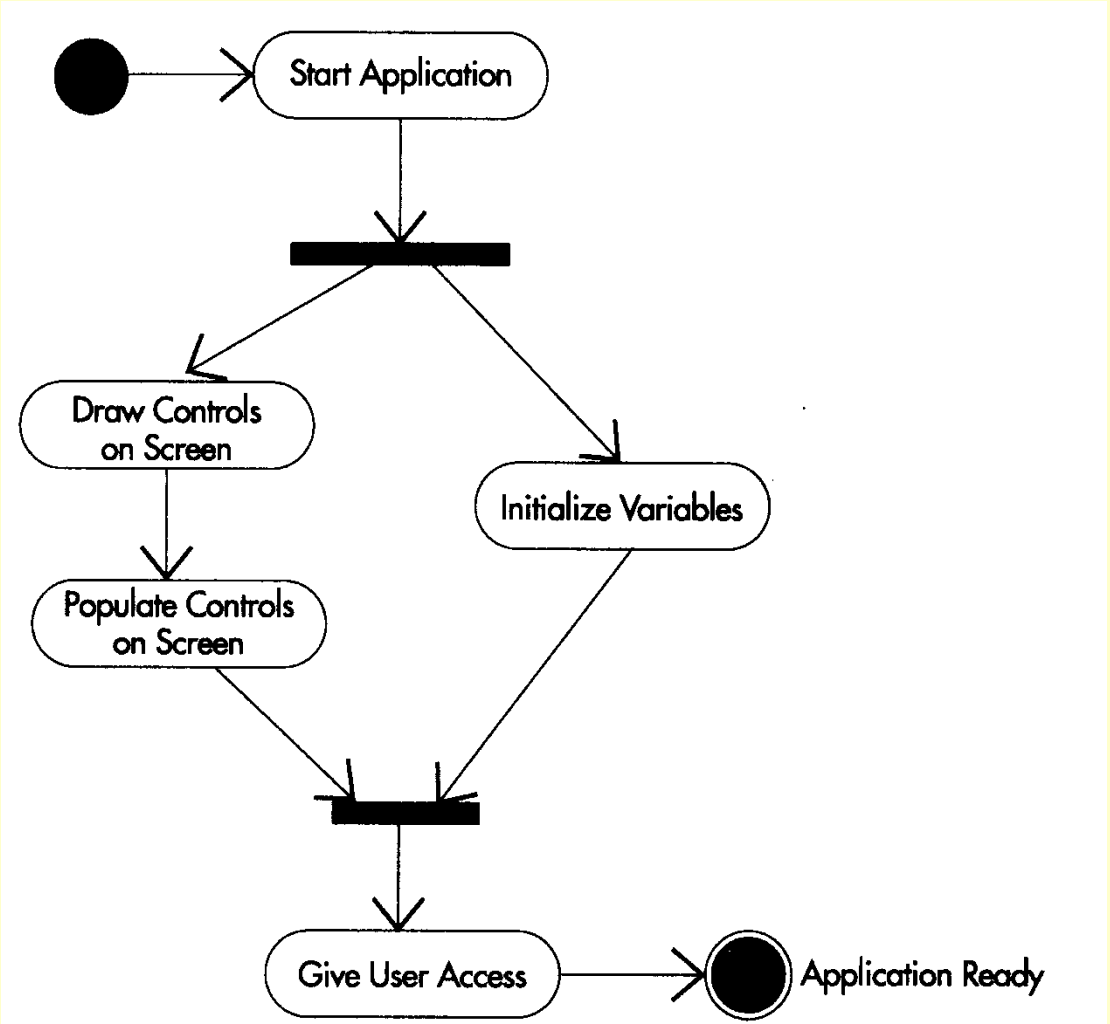
分岔和联结在UML中的表示方法相似，都用粗黑线表示。分岔具有一个转移人口，两个或者多个转移出口。分岔描述了单向处理控制流分成了多个控制流。联结与此相反，联结具有两个或者多个转移人口，只有一个出口。联结描述了不同的处理控制流合并到一起形成一个单向处理，如下图所示。





如果一个处理在其他处理之前到达了联结，它将会等待，直到所有的处理都准备好之后才会向联结传递控制权。

下例演示了分岔中的一个处理时间长于另一个的情况。当然，这完全是由每一个处理中的活动数假定的。由于我们不知道每一个活动有多长，因此不能保证哪一个首先完成。为此，我们在让用户访问应用程序之前插人了一个联结，以便确保两个独立的处理彼此连接在一起。



四、学习如何建模活动图

创建活动图共有5个任务：

1）标识需要活动图的用例。

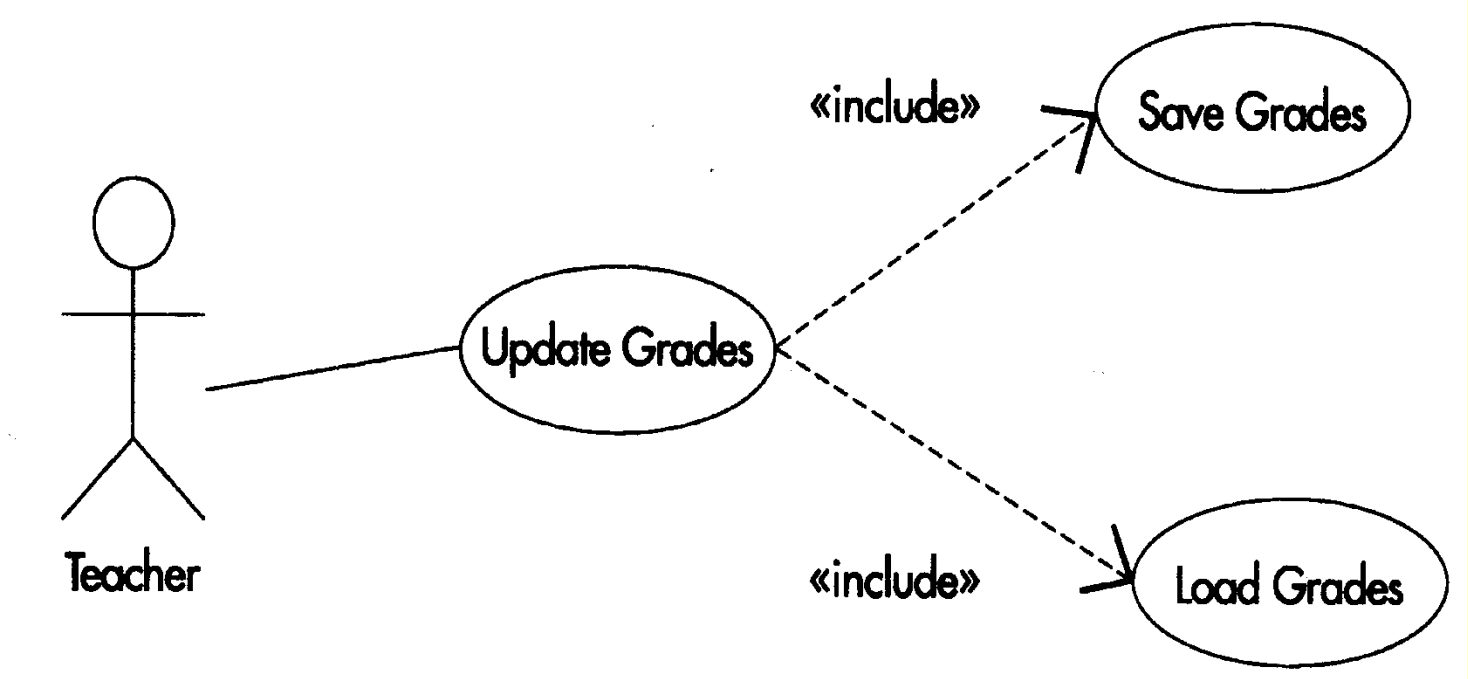
2）建模每一个用例的主路径。

3）建模每一个用例的从路径。

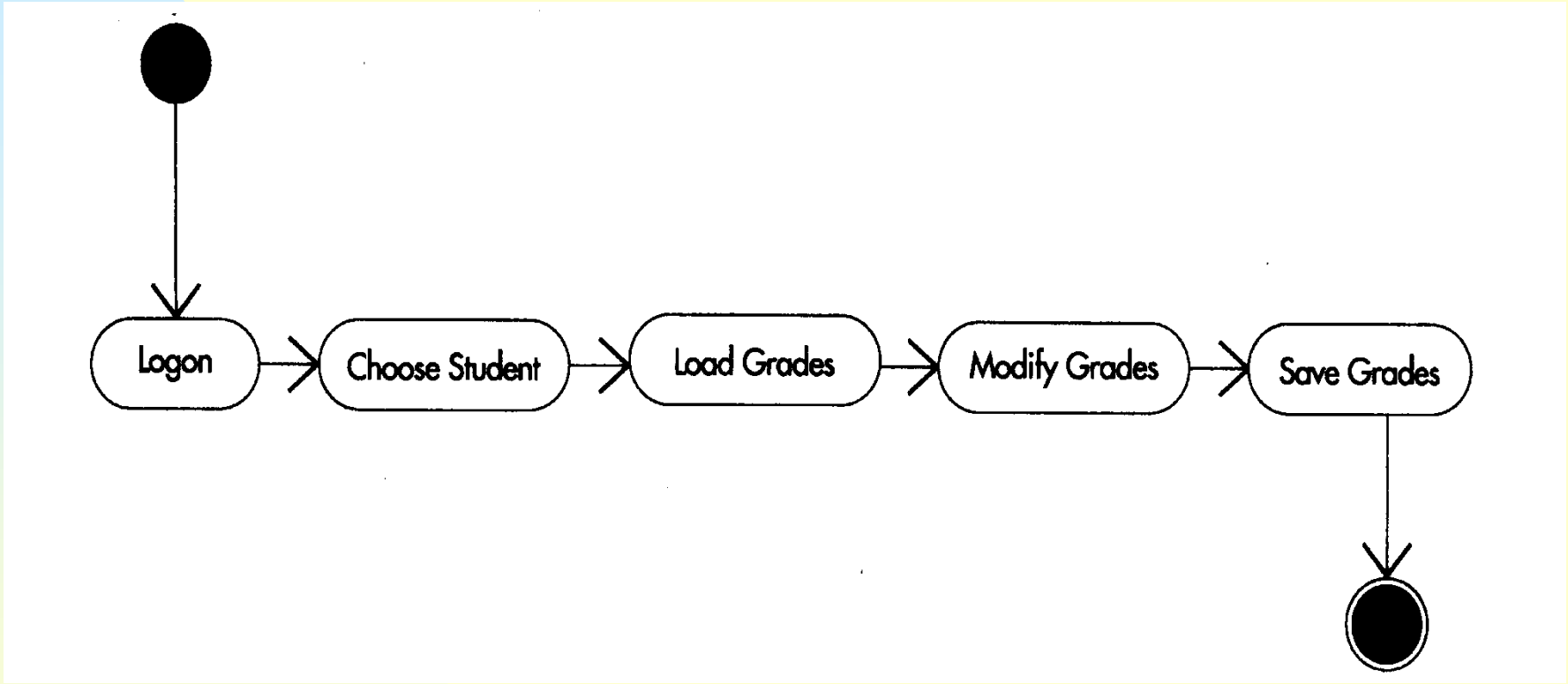
4）添加泳道来标识活动的事务分区。

5）改进高层活动并添加更多活动到图中。

在建模活动图之前，需要首先确定要建模什么。下面的教师更新分数用例是一组较大用例的一部分，我们就从它开始。如下图所示。这个用例实际上使用了3个用例。我们不仅有Update Grade用例，还有Save Grade和Load Grade用例。

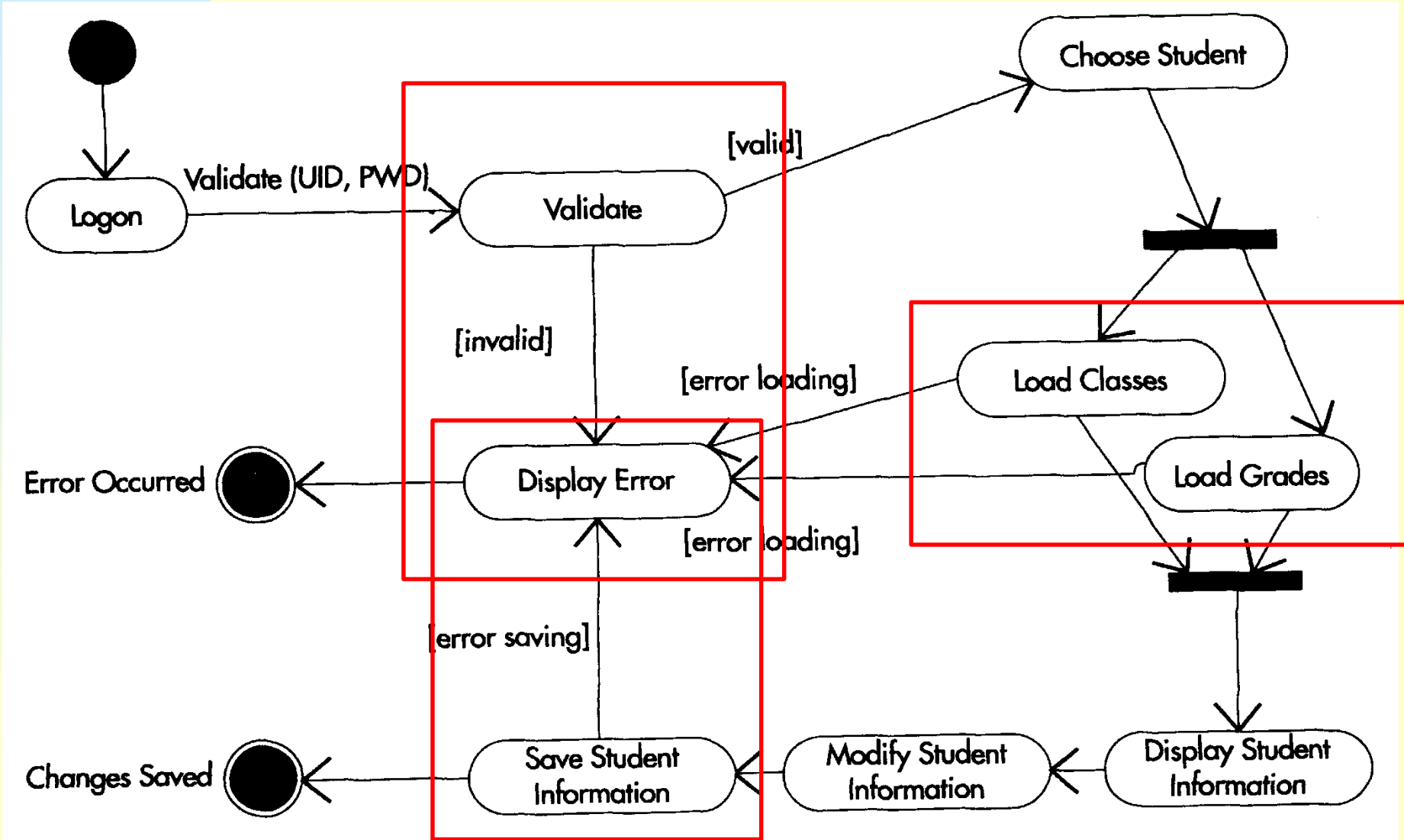


在开始创建用例的活动图时，往往先建立一条明显的路径执行工作流，然后从该路径进行扩展，如下图所示。



该路径仅考虑用例的正常活动路径（登录、选择学生、加载他们的分数，修改分数，保存修改结果等活动过程），没有考虑任何错误和判断的路径。

考虑用例其他可能的工作流情况。如处理错误，或许是执行其他活动。

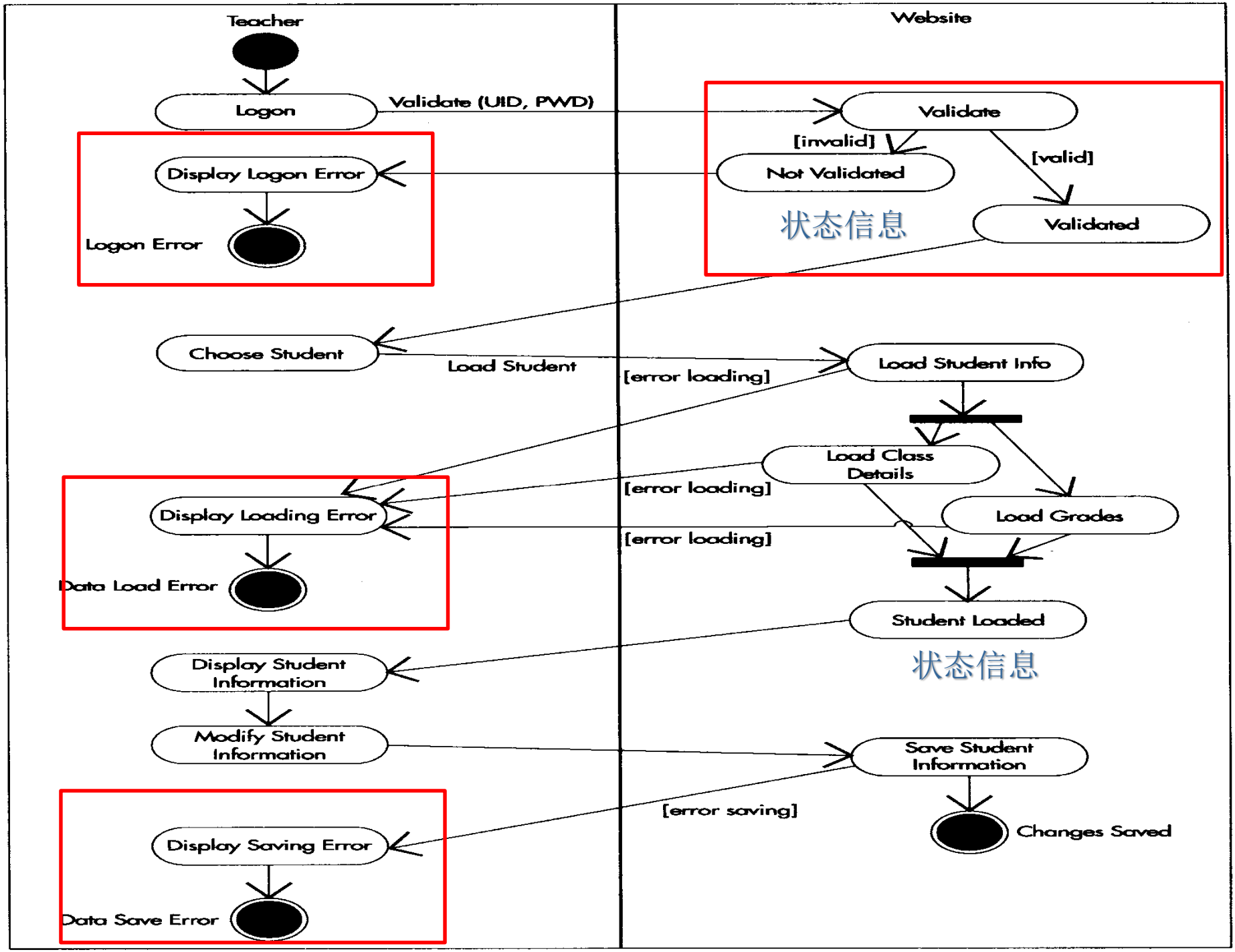


4．添加泳道

泳道对于提高活动图的可读性非常有益，在本例中也不例外。在活动图建模这一步中，可把活动图分成了两个游泳道，如下图所示。第一个游泳道是 Teacher，第二个是 Website。Teacher是用例的参与者，而 Website是提供后台功能的泛化组件。

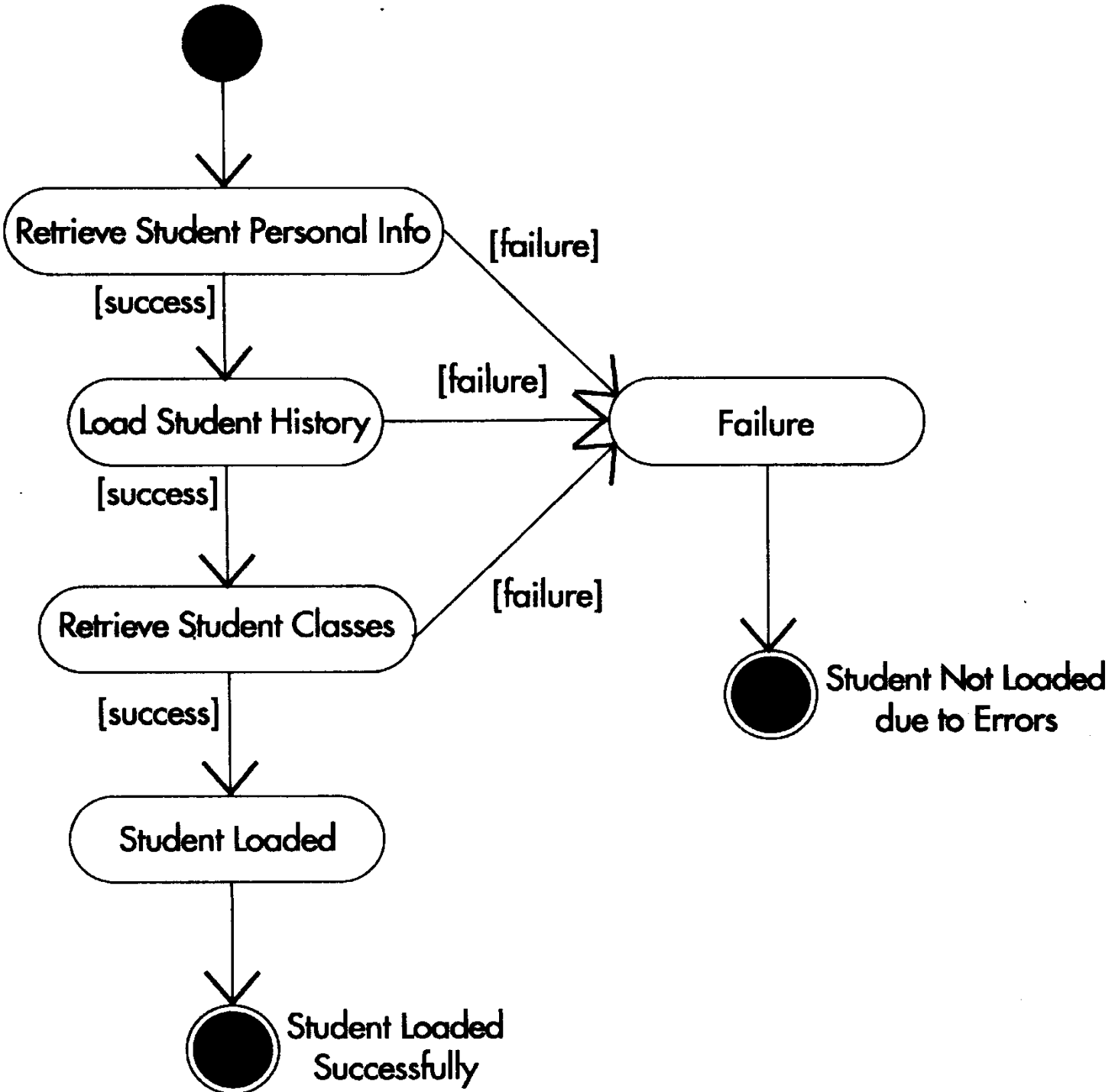
这里，我们将再一次反复向活动图添加更多的细节。在本例中，我们要添加状态以便指示现在处于哪一个转折点。在验证了教师的身份之后，把状态设置为Validated或者Not Validated，在加载学生信息之后，把状态设置为Student Loaded。

最后，我们决定让每一个出错消息都在活动图中的自己活动中，因此提供了Logon Error、Data Load Error和Data Save Error处理。


Teacher
Website
Logon
Validate (UID, PWD)
Validate
[invalid]
[valid]
Display Logon Error
Not Validated
Validated
Logon Error
状态信息
Choose Student
Load Student
Load Student Info
[error loading]
Load Class Details
[error loading]
Display Loading Error
Load Grades
[error loading]
Data Load Error
Student Loaded
状态信息
Display Student Information
Modify Student Information
Save Student Information
[error saving]
Changes Saved
Display Saving Error
Data Save Error

活动图建模的最后一步强调了反复建模的观点。在这一步中，添加更多的细节。

在教师记录学生分数用例的活动图中，我们可以看到 Load Student Info活动非常复杂，它实际上包含了许多功能，如下图所示。


Retrieve Student Personal Info
[failure]
[success]
Load Student History
[failure]
Failure
[success]
Retrieve Student Classes
[failure]
Student Not Loaded due to Errors
[success]
Student Loaded
Student Loaded Successfully

状态图

状态图的标记符与活动图的标记符非常相似，有时会让人混淆。其实，状态图用来表示单个对象的行为如何改变其状态。而活动图是用来建模不同区域的工作如何彼此交互。

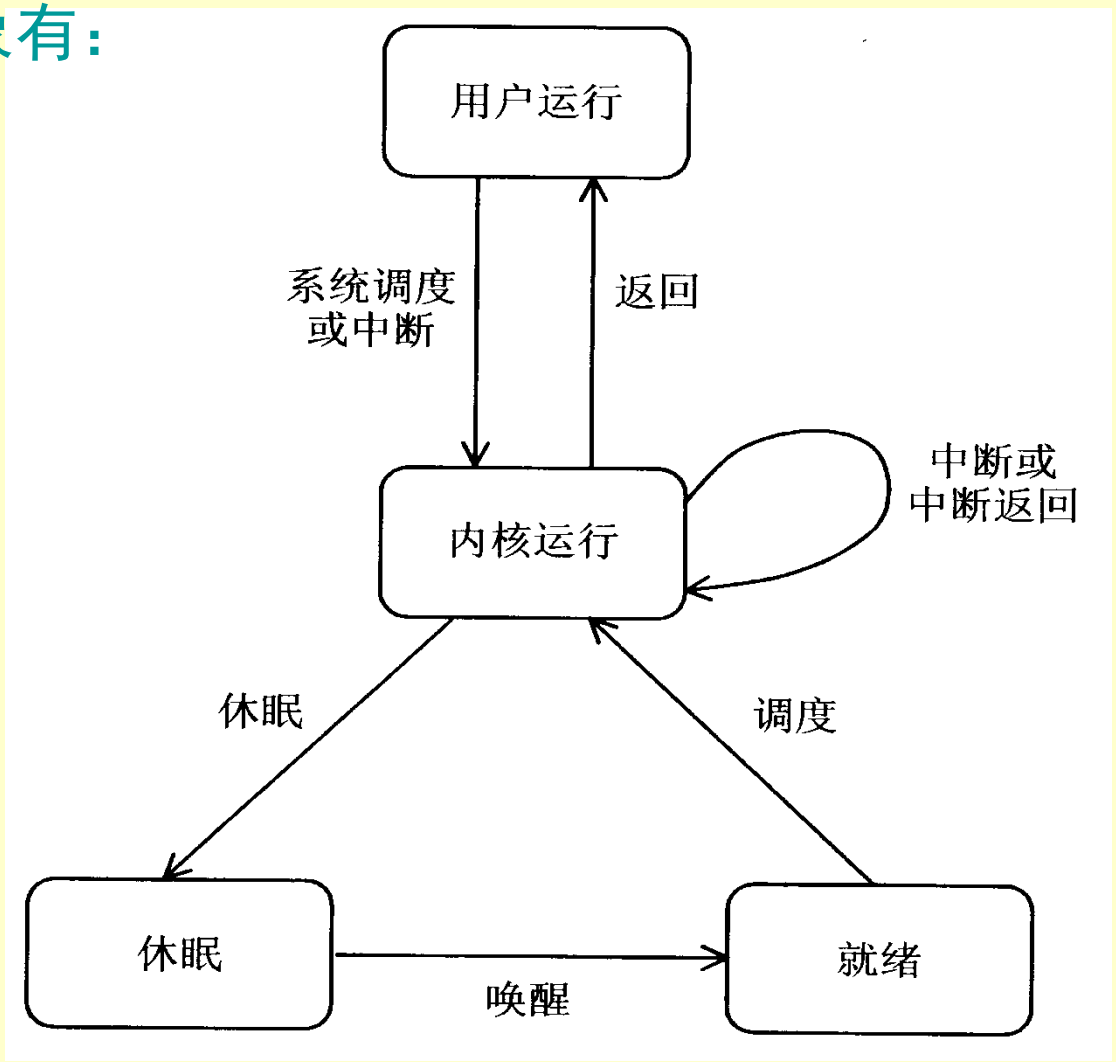
一、 定义状态图

状态图用来建模对象是如何改变其状态以响应事件，和展示对象从创建到删除的生命周期。状态定义为对象行为在某一个时刻的快照或者转折点。例如，计算机的状态可以定义为开机、启动、工作中、空闲、关机和离线等。状态图的任务就是用来描述一个对象所处的可能状态以及状态之间的转移，并给出状态变化序列的起点与终点。

状态图除了可以用于描述对象接收事件触发时的行为状态外，它还可以用于许多其他情况。例如，状态图可以用来说明基于用户输入的屏幕状态改变，也可以用来说明复杂用例的状态进展情况。

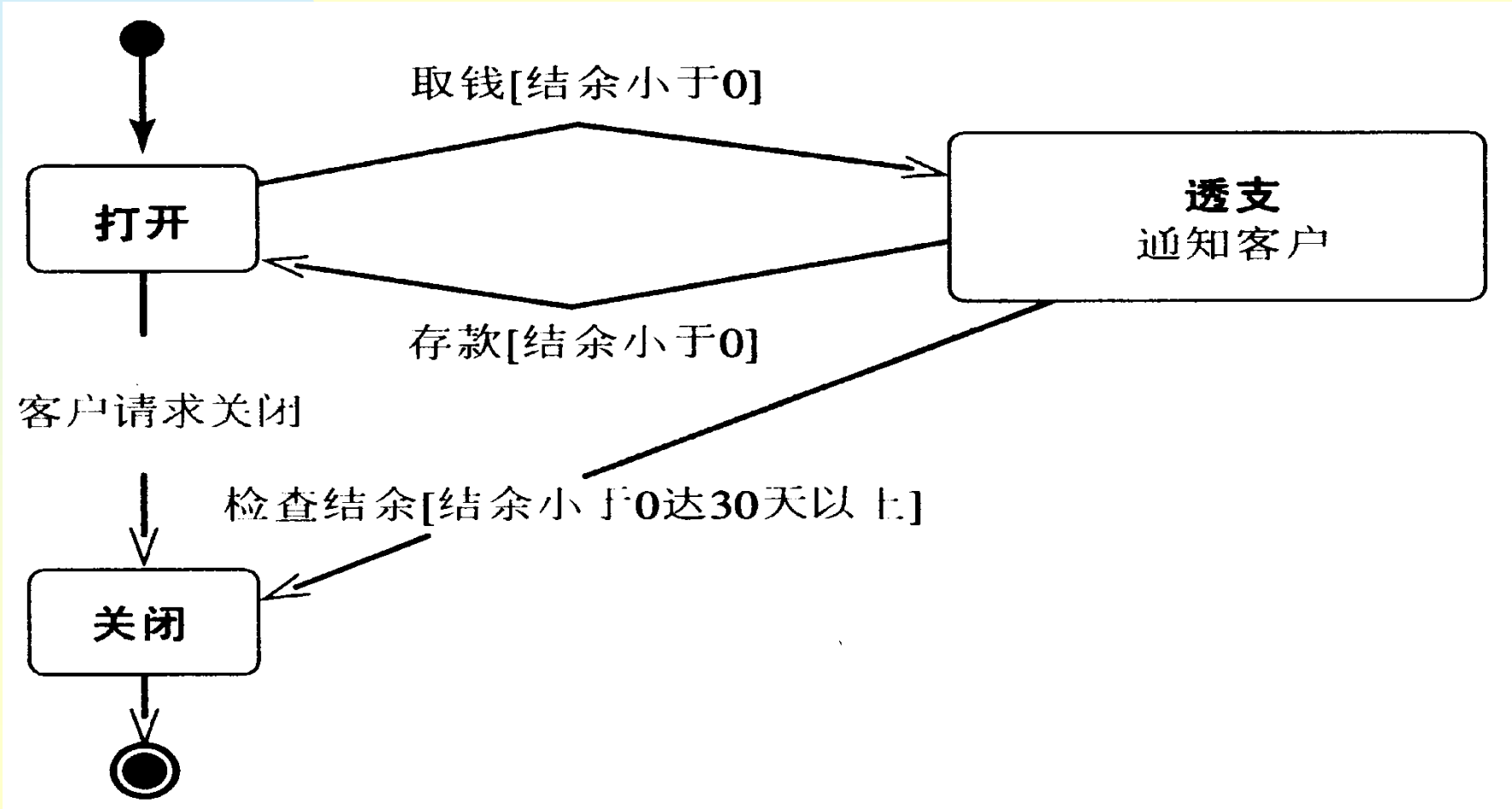
可建模状态图的对象有：

- 类
- 用例
- 子系统
- 整个系统



UNIX进程状态图

在一般系统中，不需对每个类创建状态图。当一个类实例（对象）有多种状态，每种状态中的行为表现又不相同，则可创建状态图。例如，银行帐户可以有几种不同的状态，可以打开、关闭或透支。在这些不同状态下，帐户的处理功能是不同的。

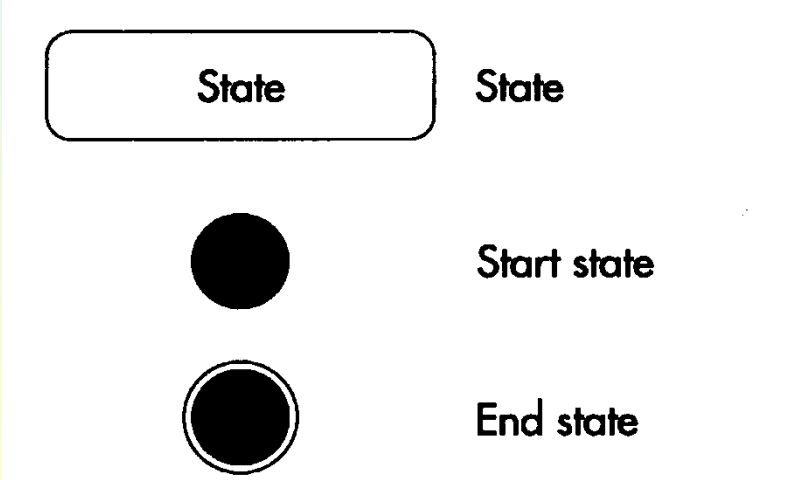


三、状态图的标记符

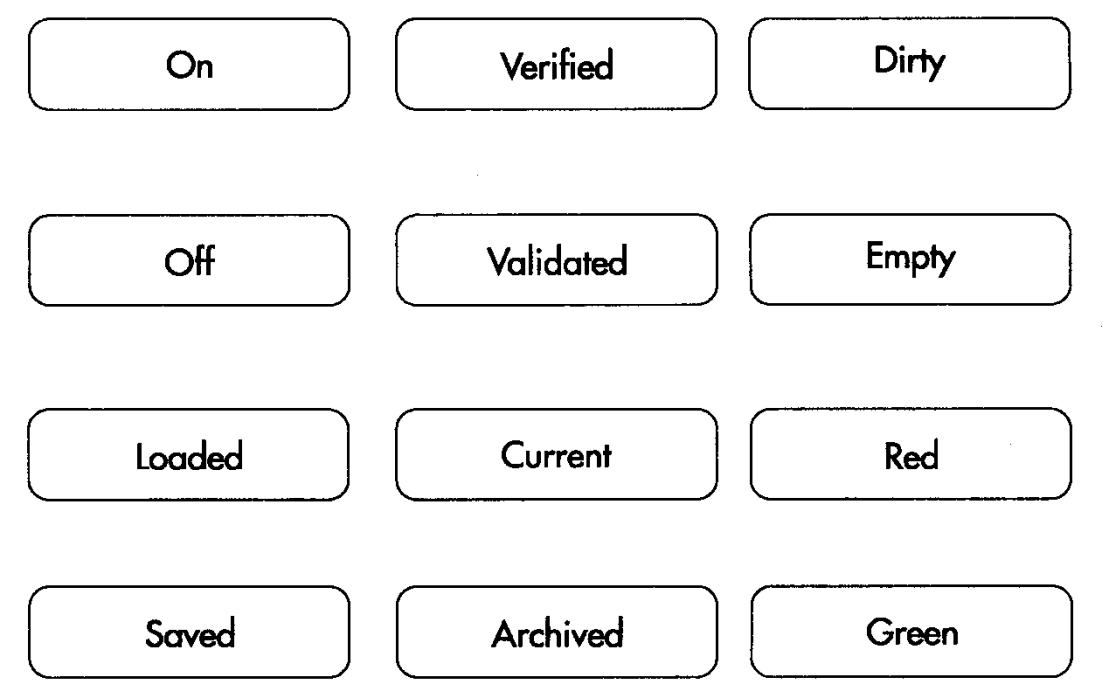
状态图由状态、转移和事件组成。联合使用状态和转移可以更好地建模它们，有时需要包含决策点和同步条来显示更高层次的细节信息。

1、 状态

状态图中共有3种独立的状态标记符，如下图所示

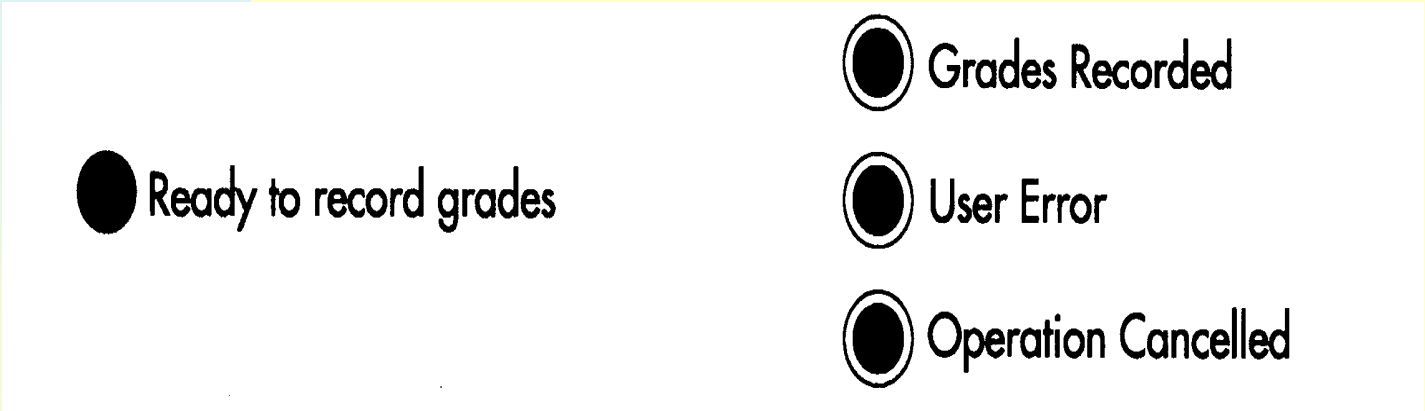


基本状态显示为圆角图形。状态的名称放在矩形中。这种标记符代表模型中满足条件的一个点。例如，诊断机的状态示例可能包含开、关、诊断和空闲等。标准状态的名称指示满足了什么条件，如下图所示。

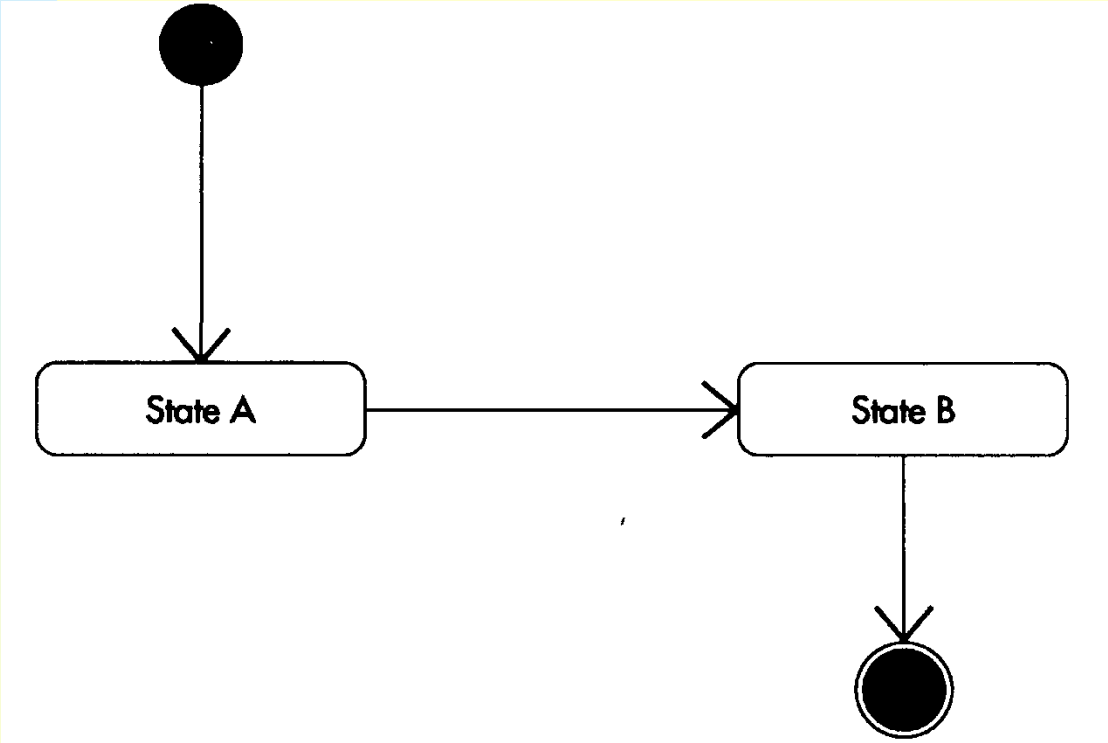


开始状态和结束状态标记符是指示模型的开始和结束状态的特殊标记符。模型中的开始状态是一个实心点。结束状态是带有圆圈的实心点。模型不必同时具有开始和结束状态，因为模型可以总是运行，从不停止。

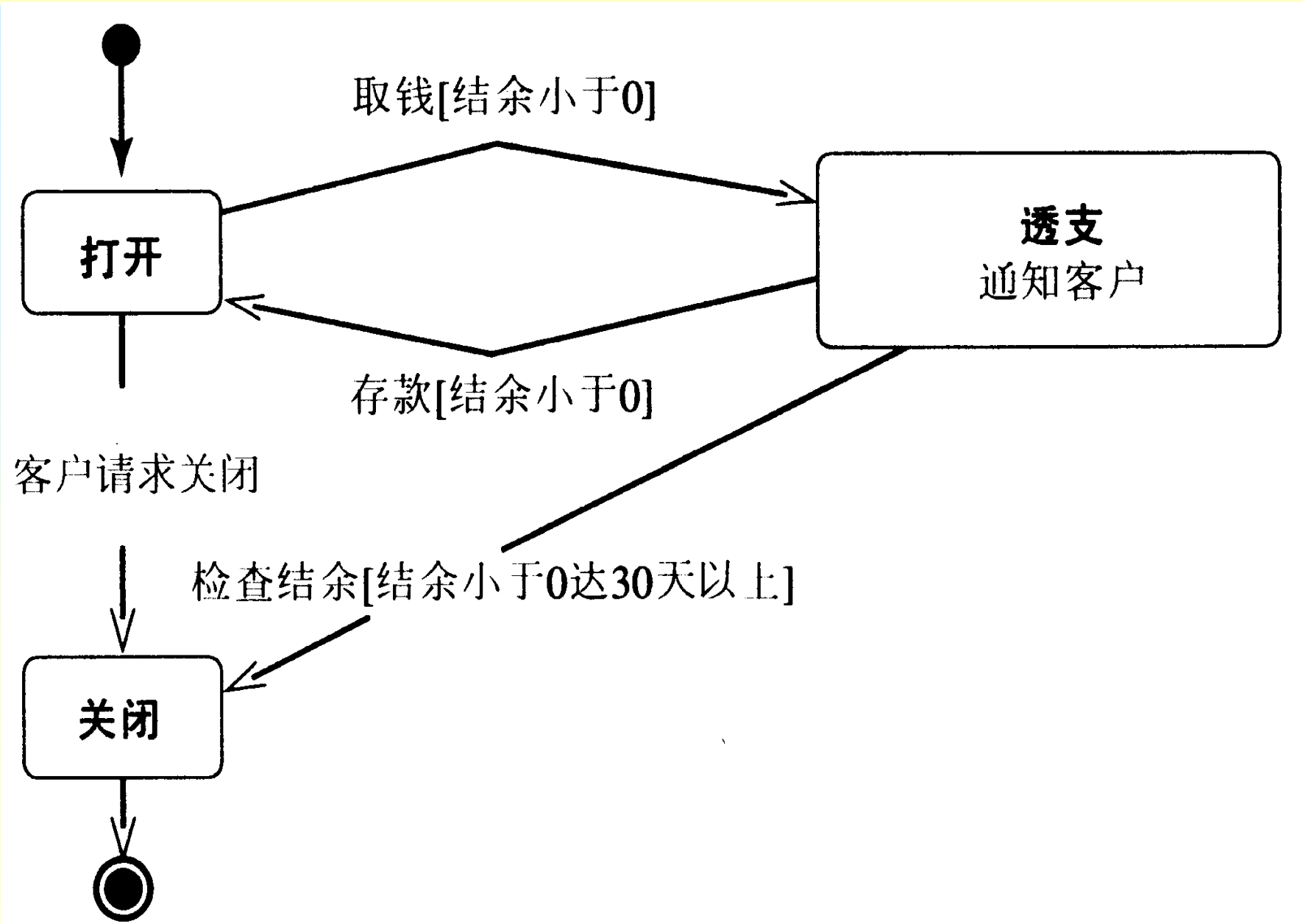
状态图中可以包含0到多个开始状态。状态图中也可以包含多个结束状态，每一个都表示一个模型能够终止的点。



转移用来显示从一个状态到另一个状态的处理流。转移使用从一个状态到另一个状态的开放箭头来标记，如下图所示。通常写上转移条件

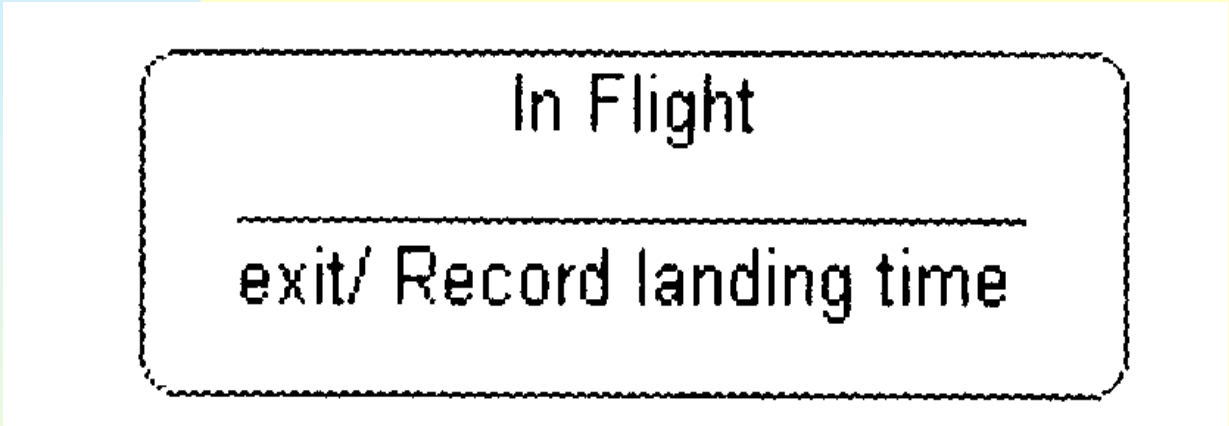


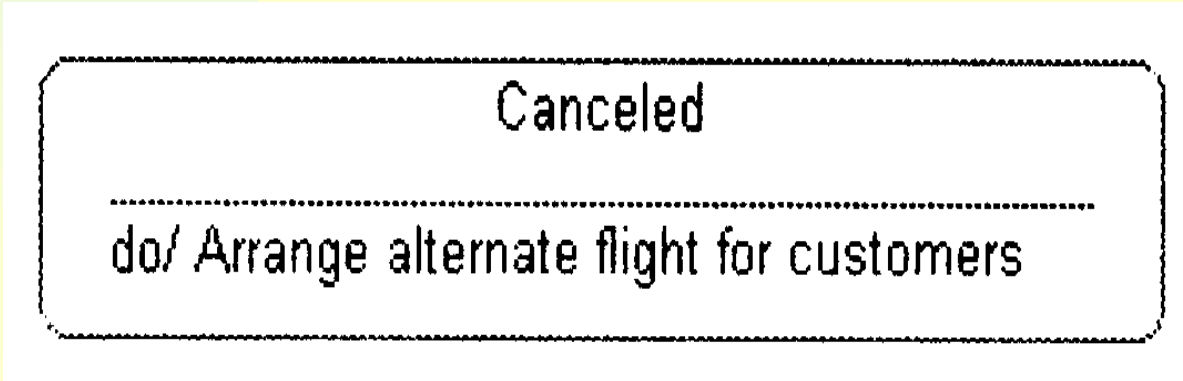
下面的示例演示了银行帐户中的转移及其有效状态，如下图所示。

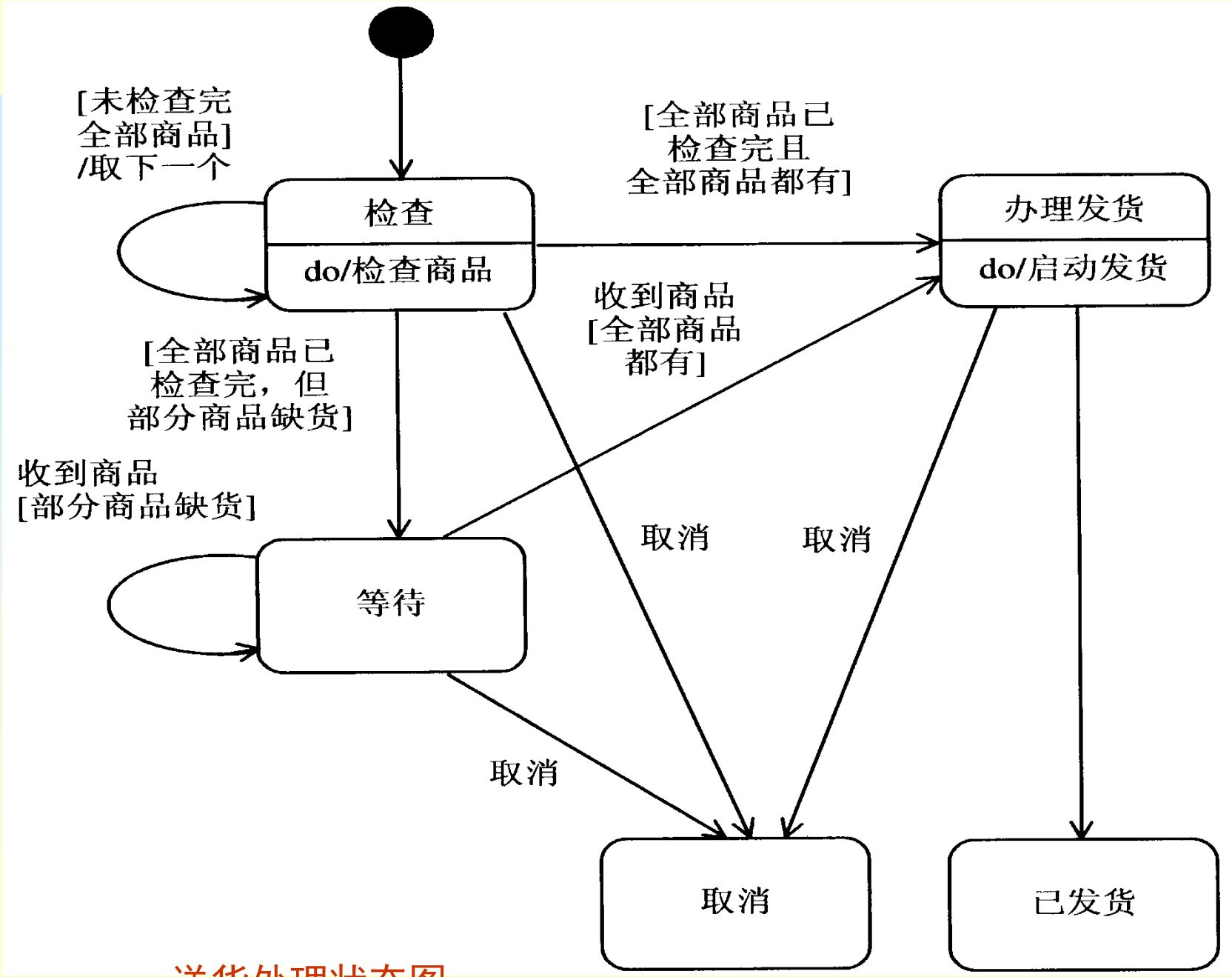


状态细节是指当对象处于特定状态时，可能要进行一些活动，例如生成报表、进行计算或向另一对象发送事件。

为了进一步描述对象在特定状态下的一些活动，可加入细节活动、进入、退出、事件和状态历史信息。

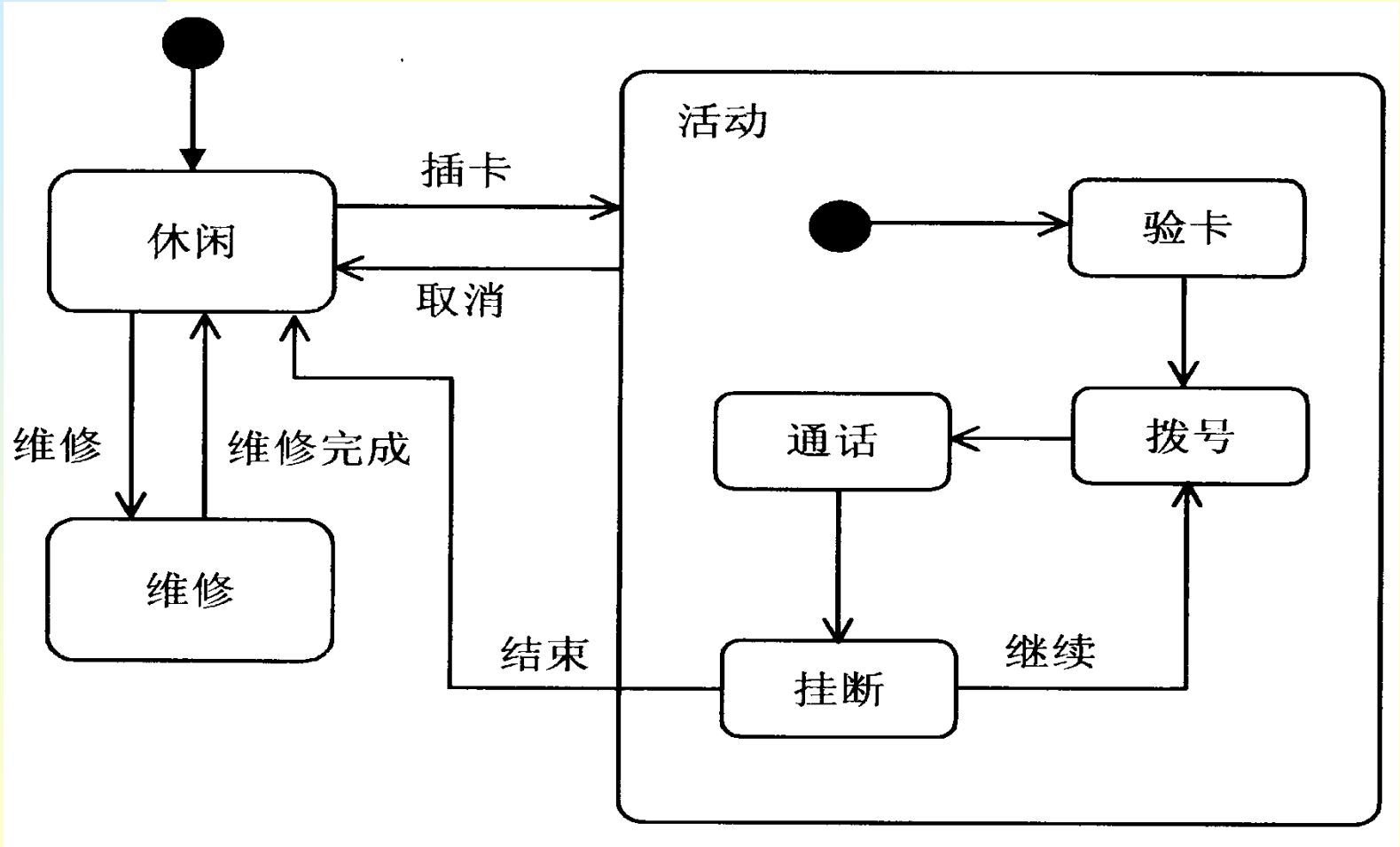





[未检查完
全部商品]
/取下一个
检查
do/检查商品
[全部商品已
检查完且
全部商品都有]
办理发货
do/启动发货
收到商品
[全部商品
都有]
[全部商品已
检查完，但
部分商品缺货]
收到商品
[部分商品缺货]
等待
取消
取消
取消
取消
已发货

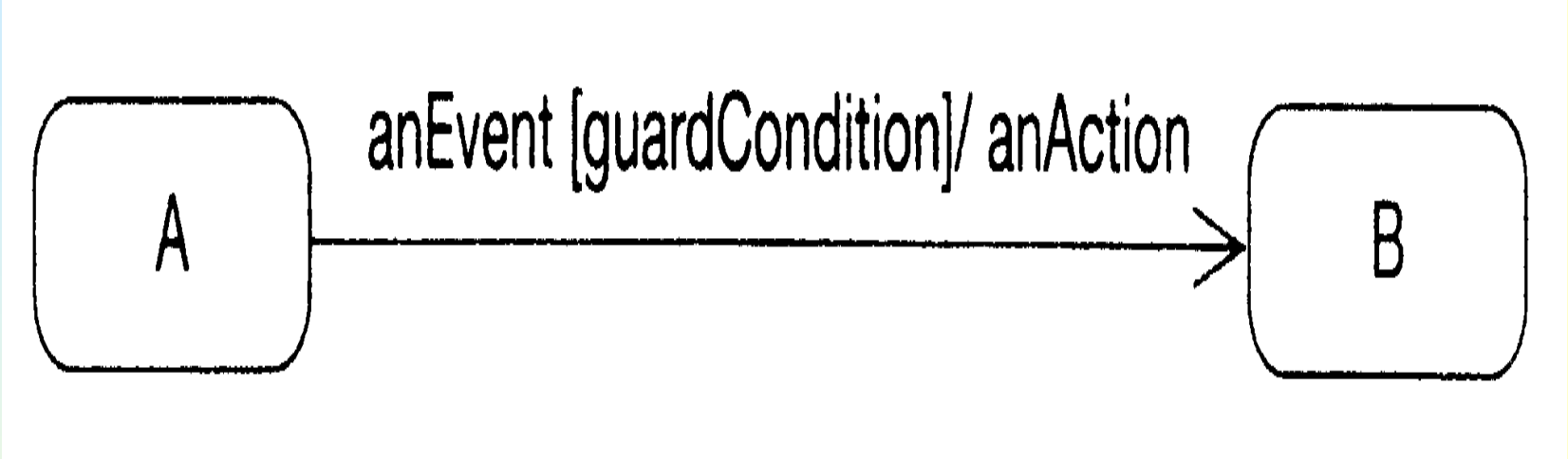
送货处理状态图

事件通常在从一个状态到另一个状态的转移路径上直接指定。事件用来指示是什么导致了模型中状态的改变。下图演示了事件的标记符。



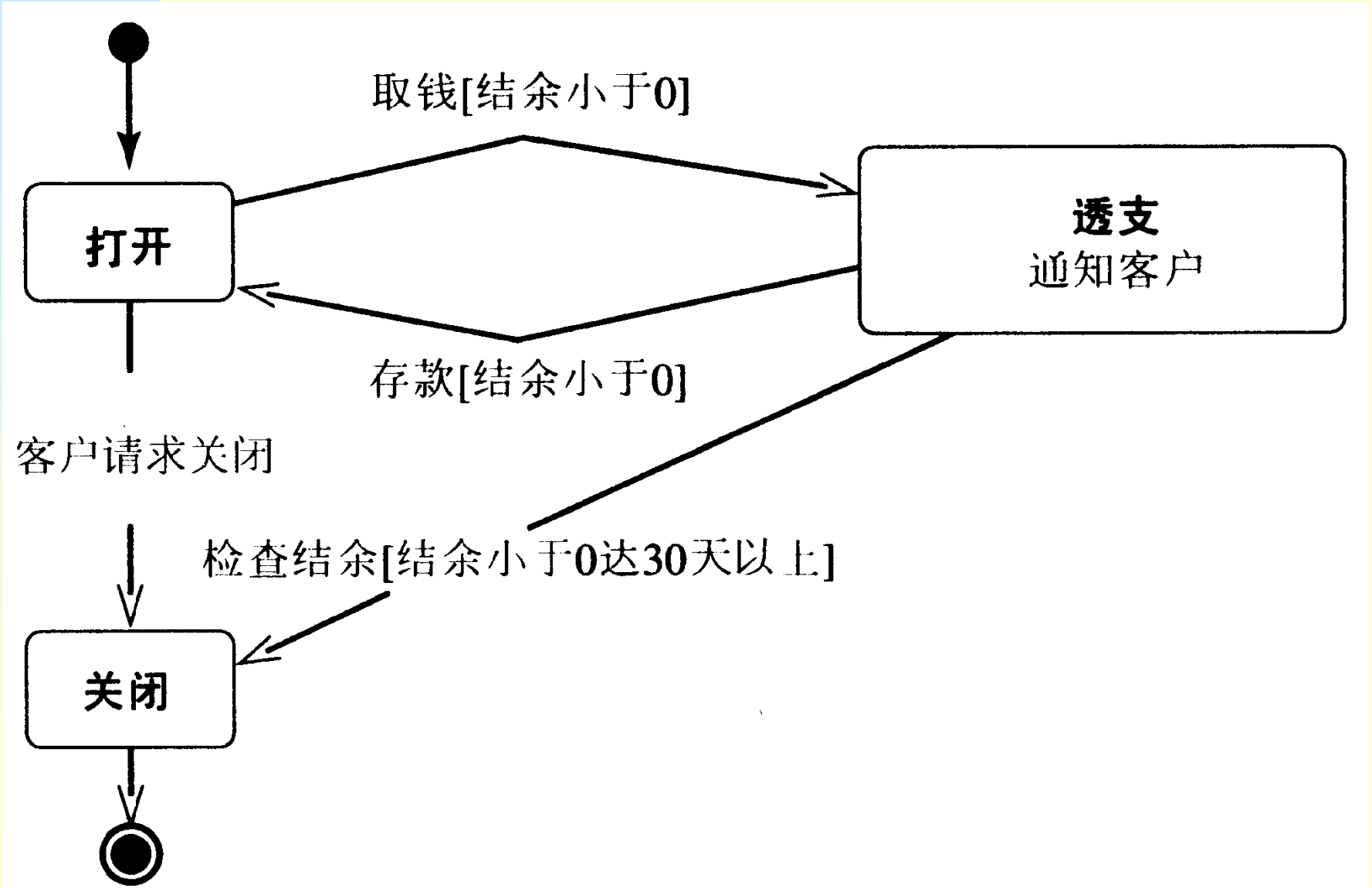
IC卡电话状态图

条件用来描述状态转移的前提。事件用来指示什么触发了转移，动作用来说明当转移发生时会产生什么情况。事件、条件和动作是转移的三个选项，其定义格式见下图所示。

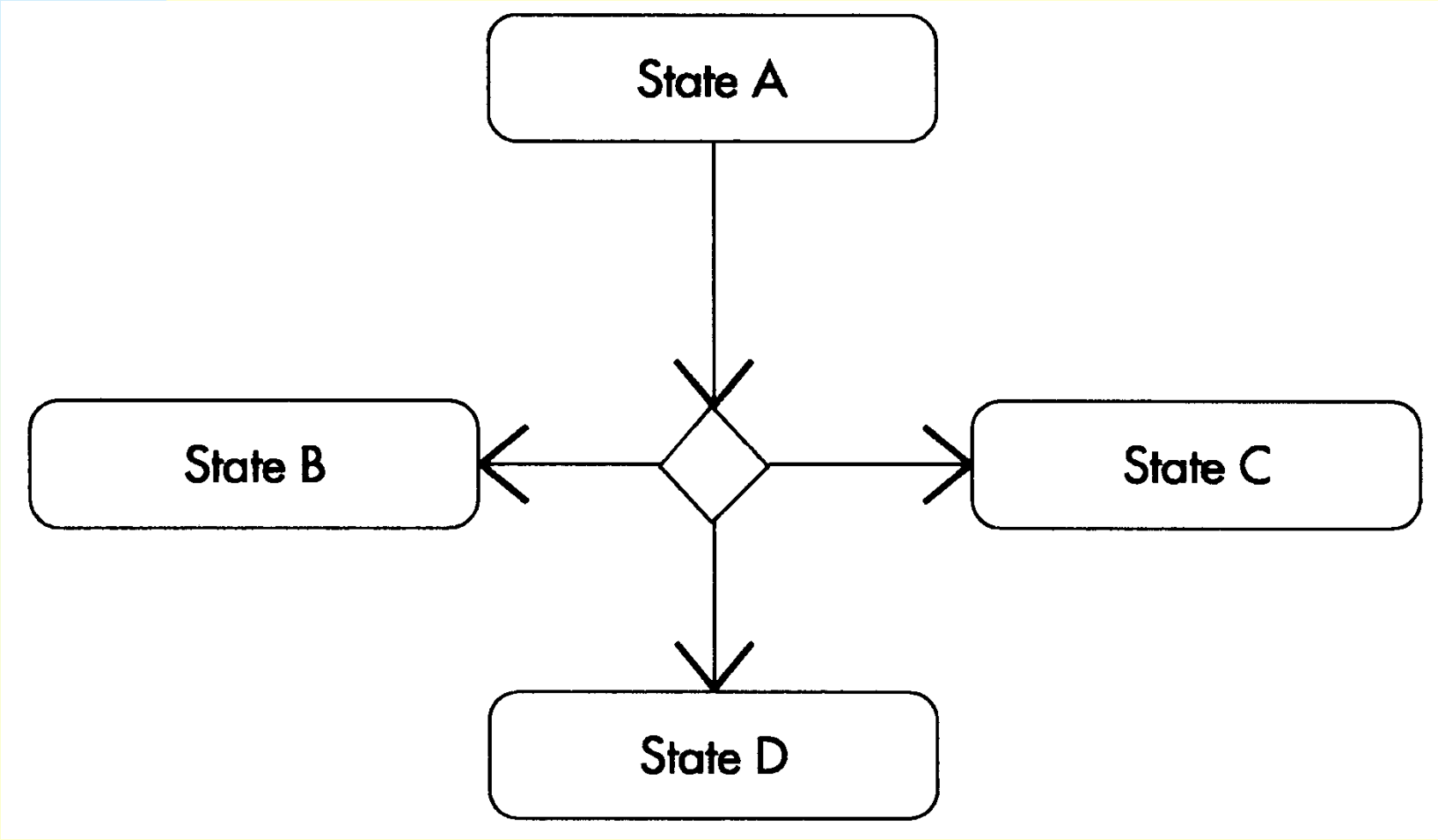


该图描述的信息是——“如果guardCondition为true，当anEvent发生时，将执行anAction，并立即进入状态B”

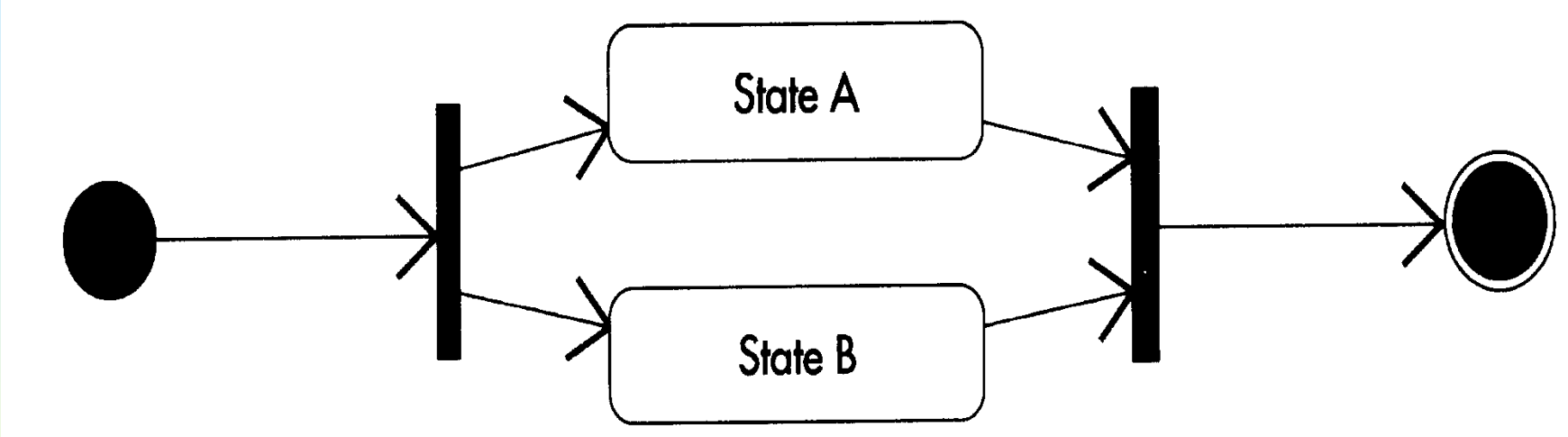
条件说明状态转移必须要满足的前提。条件一般为一个布尔表达式。如下图所示。



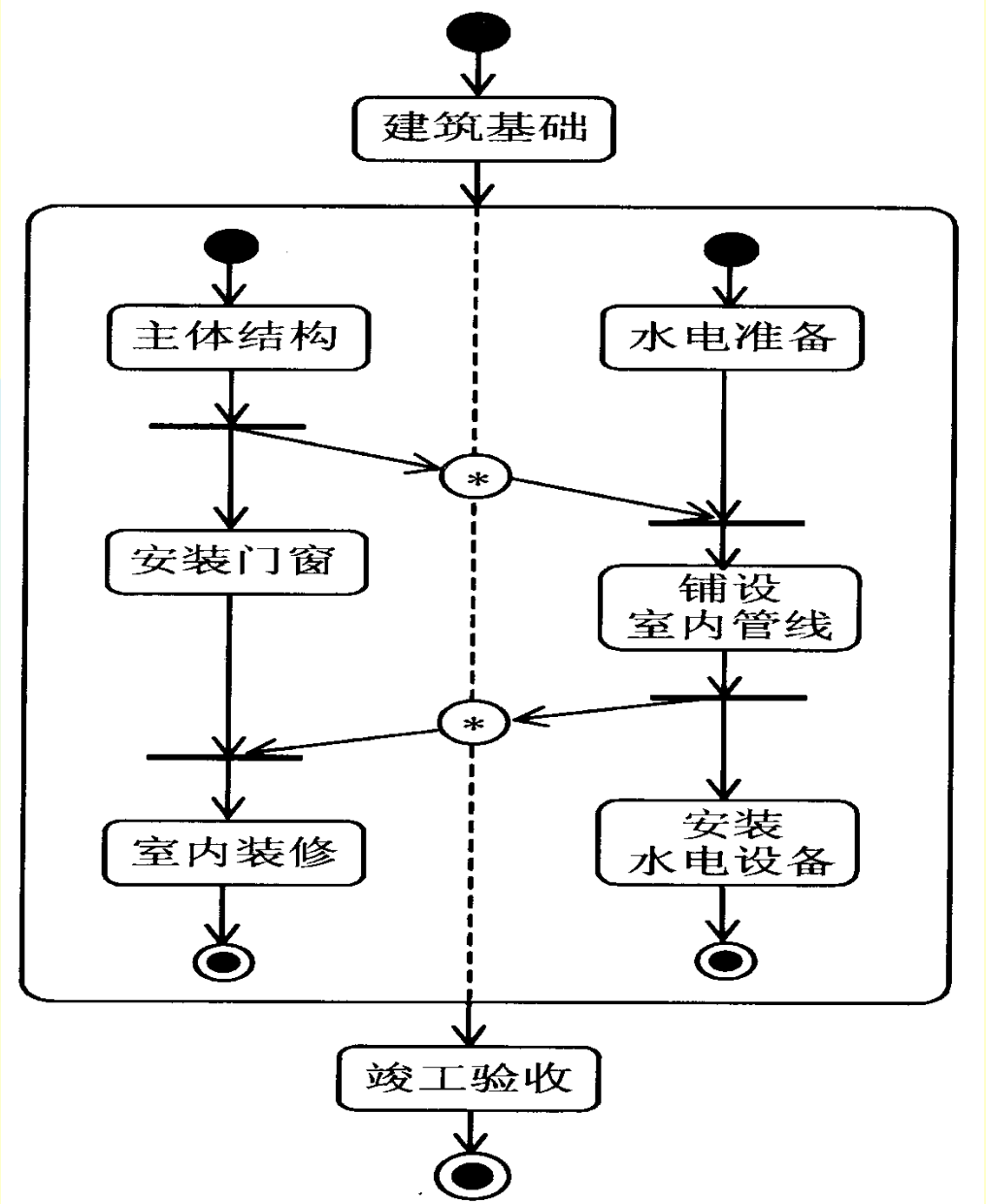
决策点在建模状态图时提供了方便，因为它通过在中心位置分组转移到各自的方向，从而提高了状态图的可视性，如下图所示。



状态图中使用同步条是为了说明并发工作流的分岔与联合。下图所示为同步条的标记符。



例 建筑住宅的状态图中，存在两个状态同步。

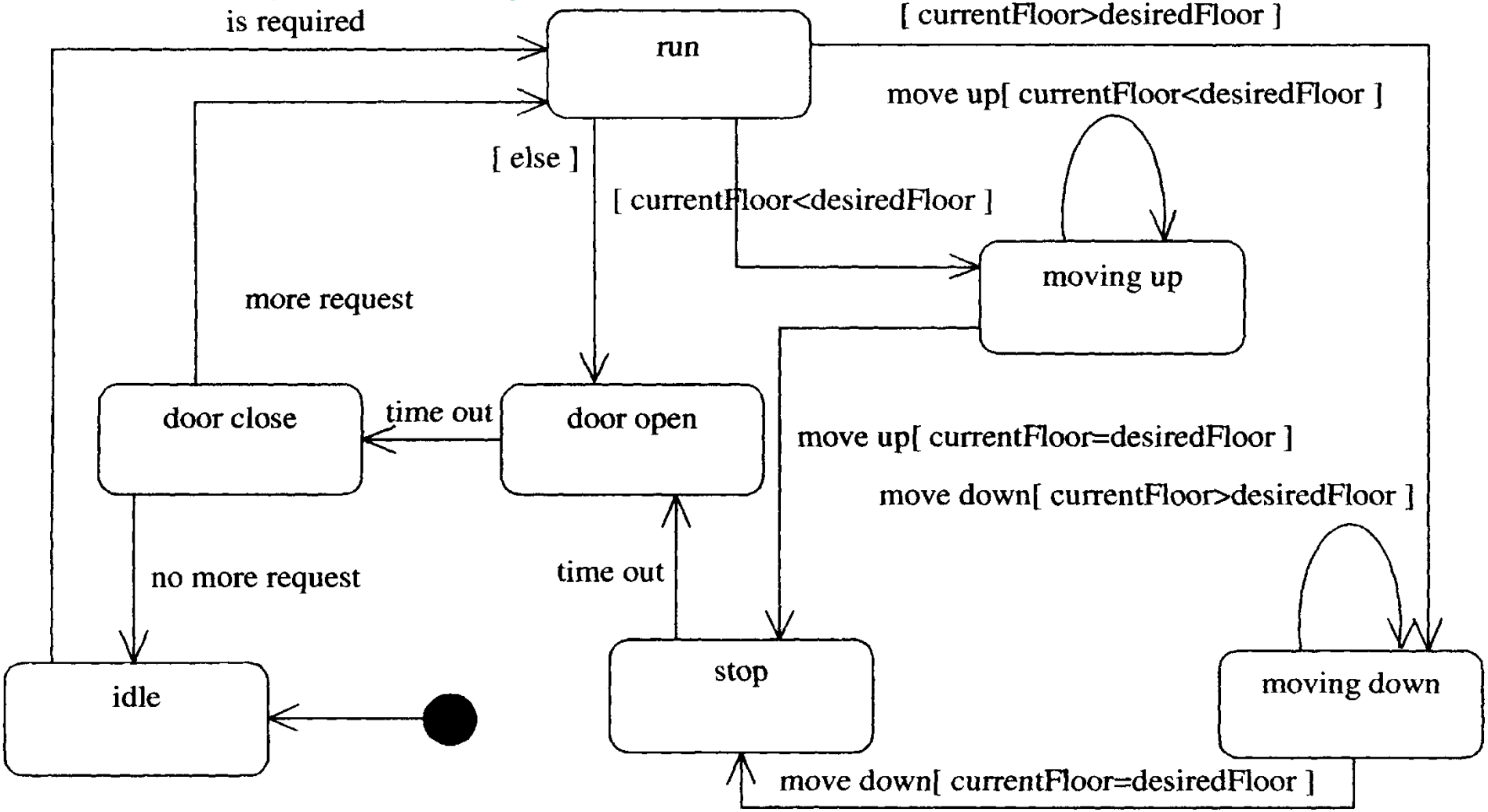


步骤:

1）标识出状态。

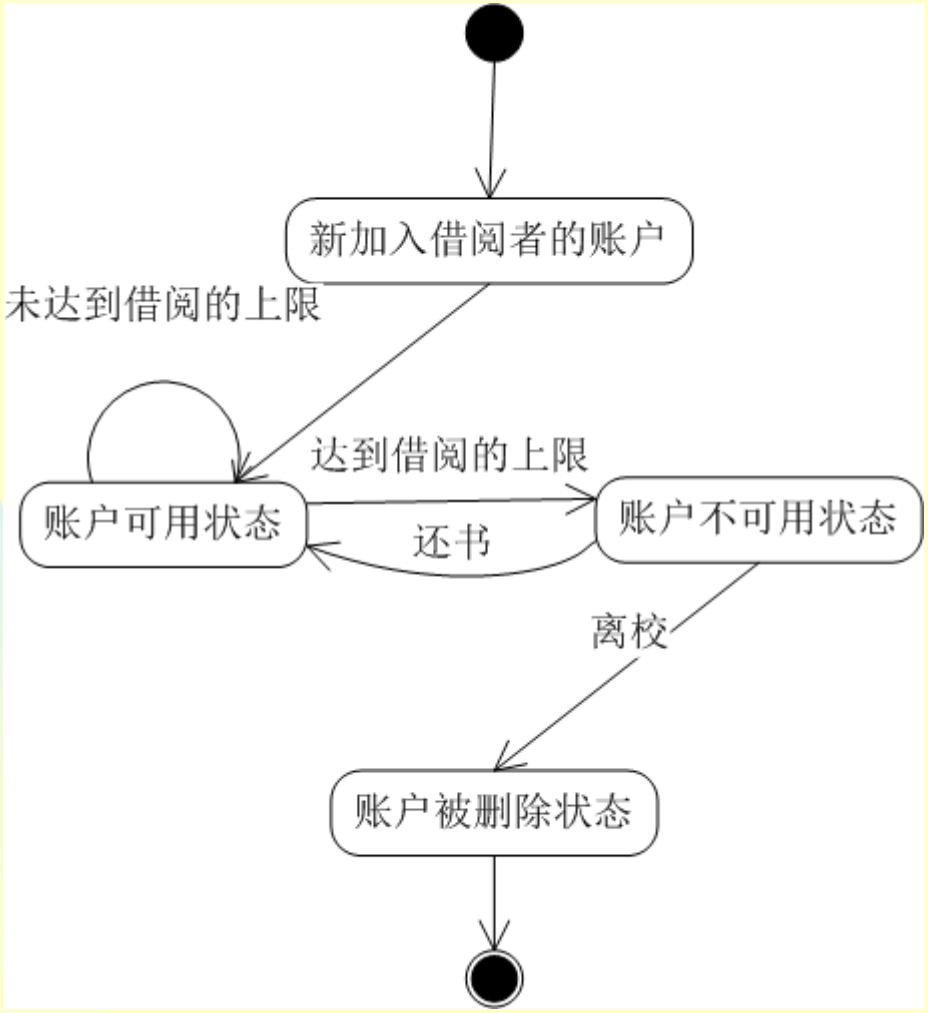
2）标识出转移。

3）标识出事件和动作。



练习

- 1. 图书馆读者的状态图


新加入借阅者的账户
未达到借阅的上限
达到借阅的上限
账户可用状态
还书
账户不可用状态
离校
账户被删除状态

概要设计文档

- 概要设计文档应说明：程序的总体结构、功能分配、模块划分、输入输出、接口设计、运行设计、数据结构设计和出错处理设计等，为详细设计提供基础
- 数据库E-R图
- 模块基本设计概念和处理流程：通过文字和图形的方式表示模块设计结构
- 设计用户的所有界面

第11讲　概要设计文档的主要内容

- 任务概述
- 总体设计
- 模块设计
- 接口定义
- 数据库设计
- 界面设计初步
- 出错处理设计

等等....

软件模块结构

- 模块功能的完善化

  一个完整的模块应当有以下几部分：

  ① 执行规定的功能的部分；

  ② 出错处理的部分。当模块不能完成规定的功能时，必须回送出错标志，出现例外情况的原因。

  ③ 如果需要返回数据给它的调用者，在完成数据加工或结束时，应当给它的调用者返回一个状态码。

模块设计的后处理

- 为每一个模块写一份处理说明
- 为每一个模块提供一份接口说明
- 确定全局数据结构和局部数据结构
- 指出所有的设计约束和限制
- 进行概要设计的评审
- 进行设计的优化(如果需要和可能的话)

接口设计

- 人机接口，人机接口是指计算机系统为完成人与机器之间互相传送信息而提供的功能的接口，包括硬件及程序。
- 软件-硬件接口，软件-硬件接口是指软件系统中软件与硬件之间的接口。例如软件与接口设备之间的接口。
- 软件接口，软件接口是软件系统中程序之间的接口。包括软件系统与其他系统或子系统之间的接口、程序模块之间的接口、程序单元之间的接口等。
- 通信接口，通信接口是指处理机和标准通信子系统之间的接口。包括为实现数据通信用来完成接口功能的部件、装置及有关软件。

接口设计基本内容

- 1、 接口的名称标识
- 2、 接口在该软件系统中的地位和作用
- 3、 接口在该软件系统中与其他程序模块和接口之间的关系
- 4、 接口的功能定义
- 5、 接口的规格和技术要求，包括它们各自适用的标准、协议或约定
- 6、 各个接口的数据特性
- 7、 各个接口的资源要求，包括硬件支持、存储资源分配等
- 8、 接口程序的数据处理要求
- 9、 接口的特殊设计要求
- 10、接口对程序编制的要求

数据库设计

- 数据库管理系统选型
- 数据库设计
- 设计E-R图
- 数据库规范化
- 建立数据库表结构

- 以BBS设计为例

设计数据库的步骤 1

- **收集信息：**

  与该系统有关人员进行交流、坐谈，充分理解数据库需要完成的任务

BBS论坛的基本功能：

- 用户注册和登录，后台数据库需要存放用户的注册信息和在线状态信息；
- 用户发贴，后台数据库需要存放贴子相关信息，如贴子内容、标题等；
- 论坛版块管理：后台数据库需要存放各个版块信息，如版主、版块名称、贴子数等；

设计数据库的步骤 2

- **标识对象（实体－Entity）**

  标识数据库要管理的关键对象或实体

实体一般是名词：

- 用户：论坛普通用户、各版块的版主。
- 用户发的主贴
- 用户发的跟贴（回贴）
- 版块：论坛的各个版块信息

设计数据库的步骤 3

- 标识每个实体的属性（Attribute）

论坛用户：
- 呢称
- 密码
- 电子邮件
- 生日
- 性别
- 用户的等级
- 备注信息
- 注册日期
- 状态
- 积分

主贴
- 发贴人
- 发贴表情
- 回复数量
- 标题
- 正文
- 发贴时间
- 点击数
- 状态：
- 最后回复时间

回贴
- 贴子编号
- 回贴人，
- 回贴表情
- 标题
- 正文
- 回贴时间
- 点击数

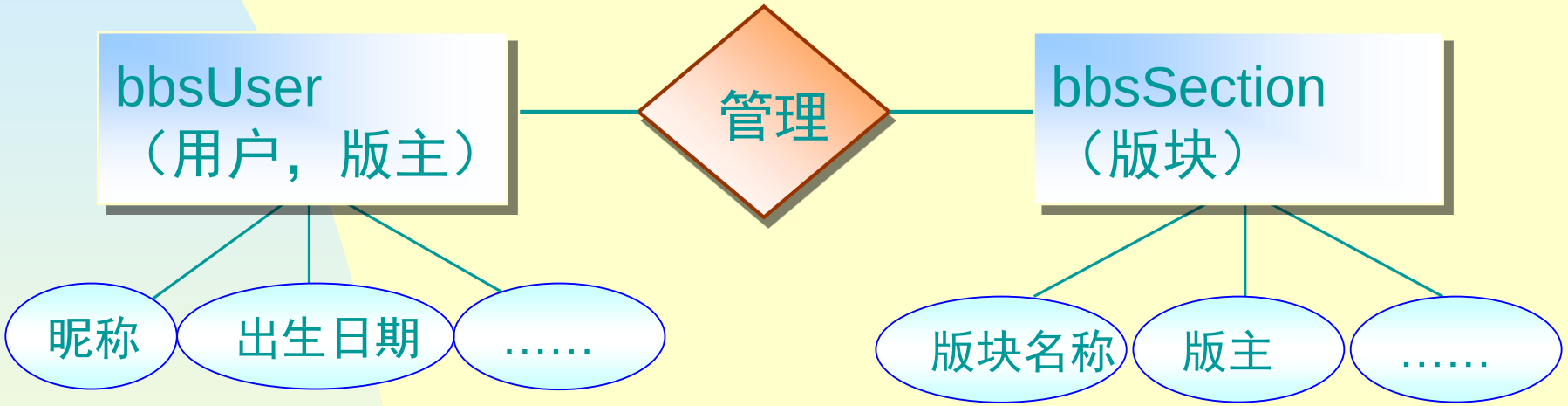
版块
- 版块名称
- 版主
- 本版格言
- 点击率
- 发贴数

设计数据库的步骤 4

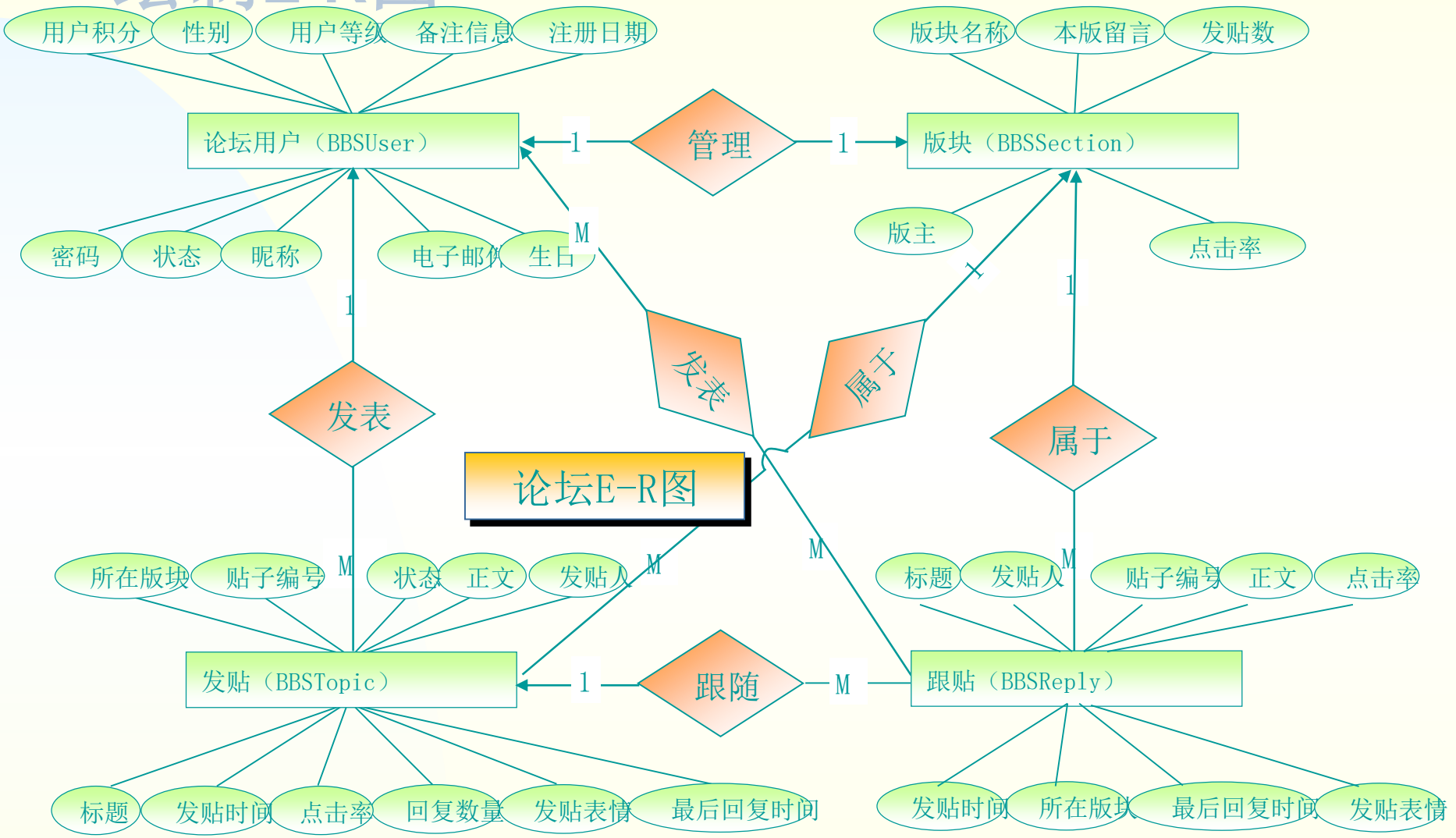
- **标识对象之间的关系（Relationship）**

  - 跟贴和主贴有主从关系：我们需要在跟贴对象中表明它是谁的跟贴；
  - 版块和用户有关系：从用户对象中可以根据版块对象查出对应的版主用户的情况；
  - 主贴和版块有主从关系：需要表明发贴是属于哪个版块的；
  - 跟贴和版块有主从关系：需要表明跟贴是属于哪个版块的；

绘制E-R图 1



绘制E-R图


用户积分
性别
用户等级
备注信息
注册日期
论坛用户（BBSUser）
1
管理
1
版块（BBSSection）
版块名称
本版留言
发贴数
密码
状态
昵称
电子邮件
生日
M
版主
点击率
1
发表
发表
论坛E-R图
属于
属于
1
M
M
M
M
所在版块
贴子编号
状态
正文
发贴人
标题
发贴人
贴子编号
正文
点击率
发贴（BBSTopic）
1
跟随
M
跟贴（BBSReply）
标题
发贴时间
点击率
回复数量
发贴表情
最后回复时间
发贴时间
所在版块
最后回复时间
发贴表情

如何将E-R图转换为表

- 实体->表
- 属性->字段（列）
- 关系->表之间的主外键关系
- 注意：没有主键的表添加ID编号列，它没有实际含义，用于做主键或外键，例如用户表中的“UID”列，版块表中添加“SID”列，发贴表和跟贴表中的“TID”列

如何将E-R图转换为表

UID主键

BBSUser（论坛用户）表

UID（用户编号，主键）
UName（用户昵称）
UPassword（密码）
UEmail（电子邮件）
UBirthday（生日）
USex（性别）
UClass（用户等级）
UStatement（用户备注）
URegDate（注册日期）
UState（用户状态）
UPoint（用户积分）

TID主键

BBSTopic（发贴）表

TID（标识主键列）
TNumber（贴子编号）
TSID（所在版块）
TUID（发贴人）
TReplyCount（回复数）
TEmotion（回复表情）
TTopic（主题）
TContents（正文）
TTime（回复时间）
TClickCount（点击数）
TFlag（状态）
TLastClickT（最后回复时间）

RID主键

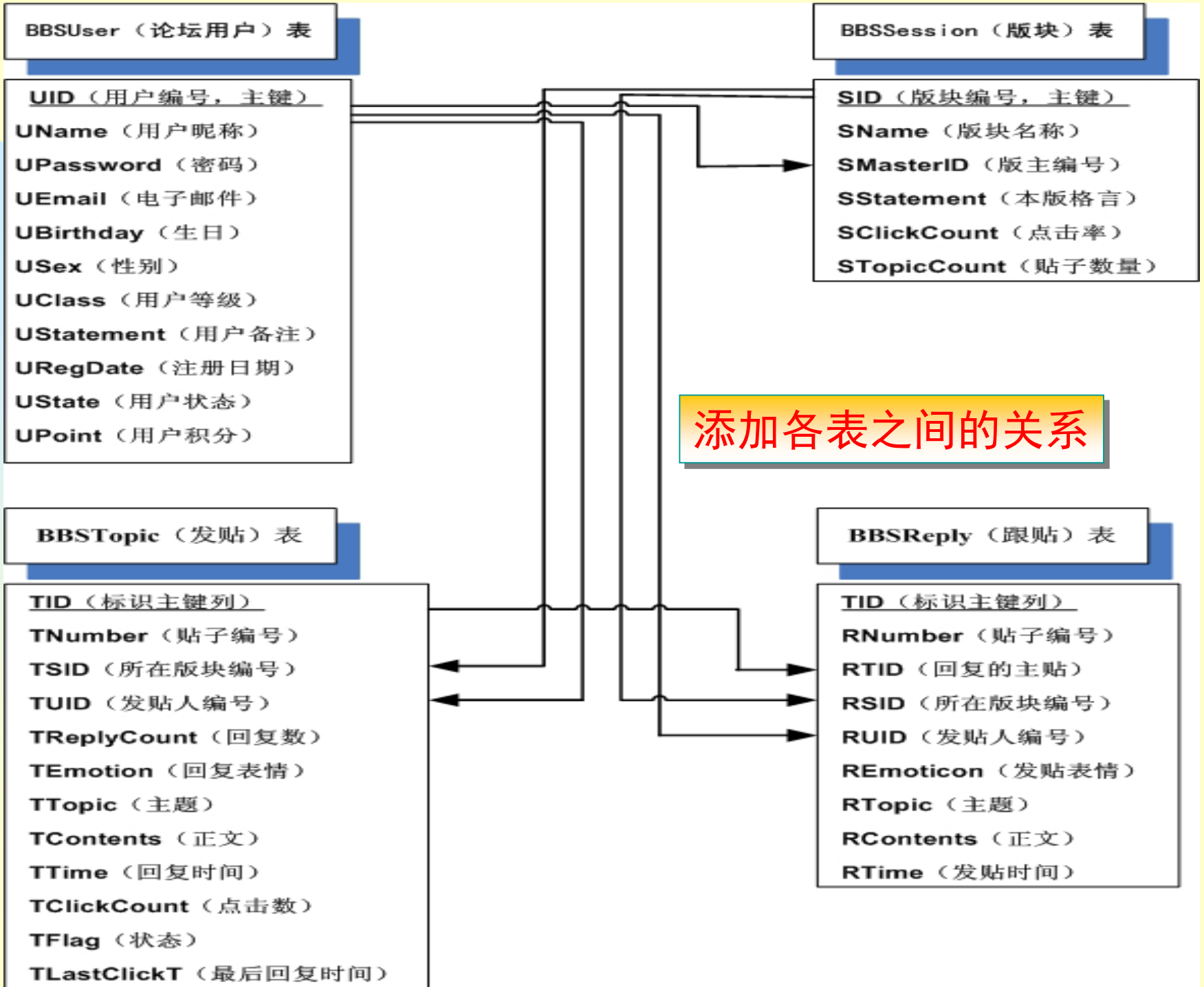
BBSReply（跟贴）表

RID（标识主键列）
RNumber（贴子编号）
RTID（回复的主贴）
RSID（所在版块编号）
RUID（发贴人编号）
REmoticon（发贴表情）
RTopic（主题）
RContents（正文）
RTime（发贴时间）

SID主键

BBSSection（版块）表

SID（版块编号，主键）
SName（版块名称）
SMasterID（版主编号）
SStatement（本版格言）
SClickCount（点击率）
STopicCount（贴子数量）

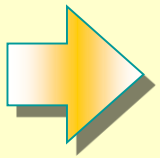

BBSUser（论坛用户）表
UID（用户编号，主键）
UName（用户昵称）
UPassword（密码）
UEmail（电子邮件）
UBirthday（生日）
USex（性别）
UClass（用户等级）
UStatement（用户备注）
URegDate（注册日期）
UState（用户状态）
UPoint（用户积分）
BBSSession（版块）表
SID（版块编号，主键）
SName（版块名称）
SMasterID（版主编号）
SStatement（本版格言）
SClickCount（点击率）
STopicCount（贴子数量）
添加各表之间的关系
BBSTopic（发贴）表
TID（标识主键列）
TNumber（贴子编号）
TSID（所在版块编号）
TUID（发贴人编号）
TReplyCount（回复数）
TEmotion（回复表情）
TTopic（主题）
TContents（正文）
TTime（回复时间）
TClickCount（点击数）
TFlag（状态）
TLastClickT（最后回复时间）
BBSReply（跟贴）表
TID（标识主键列）
RNumber（贴子编号）
RTID（回复的主贴）
RSID（所在版块编号）
RUID（发贴人编号）
REmoticon（发贴表情）
RTopic（主题）
RContents（正文）
RTime（发贴时间）

数据规范化

- 仅有好的RDBMS（关系型数据库管理系统）并不足以避免数据冗余，必须在数据库的设计中创建好的表结构
- Dr E.F.codd 最初定义了规范化的三个级别，范式是具有最小冗余的表结构。这些范式是：
  - 第一范式(1st NF －First Normal Fromate)
  - 第二范式(2nd NF－Second Normal Fromate)
  - 第三范式(3rd NF－ Third Normal Fromate)

第一范式 (1st NF)

BuyerID	Address
1	中国北京市
2	美国纽约市
3	英国利物浦
4	日本东京市
…	…



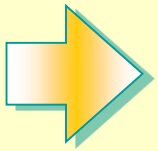
BuyerID	Country	City
1	中国	北京
1	中国	北京
4	日本	东京
2	美国	纽约
…	…	…

- **第一范式的目标是确保每列的原子性**
- **如果每列都是不可再分的最小数据单元（也称为最小的原子单元），则满足第一范式（1NF）**

第二范式 (2nd NF)

Orders

字　段	例　子
订单编号	001
产品编号	A001
订购日期	2000-2-3
价　格	$29.00
…	…



Orders

字　段	例　子
订单编号	001
订购日期	2000-2-3

Products

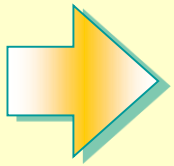
字　段	例　子
产品编号	A001
价　格	$29.00

- **如果一个关系满足1NF，并且除了主键以外的其他列，都依赖与该主键，则满足第二范式（2NF）**
- **第二范式要求每个表只描述一件事情**

第三范式 (3rd NF)

Orders

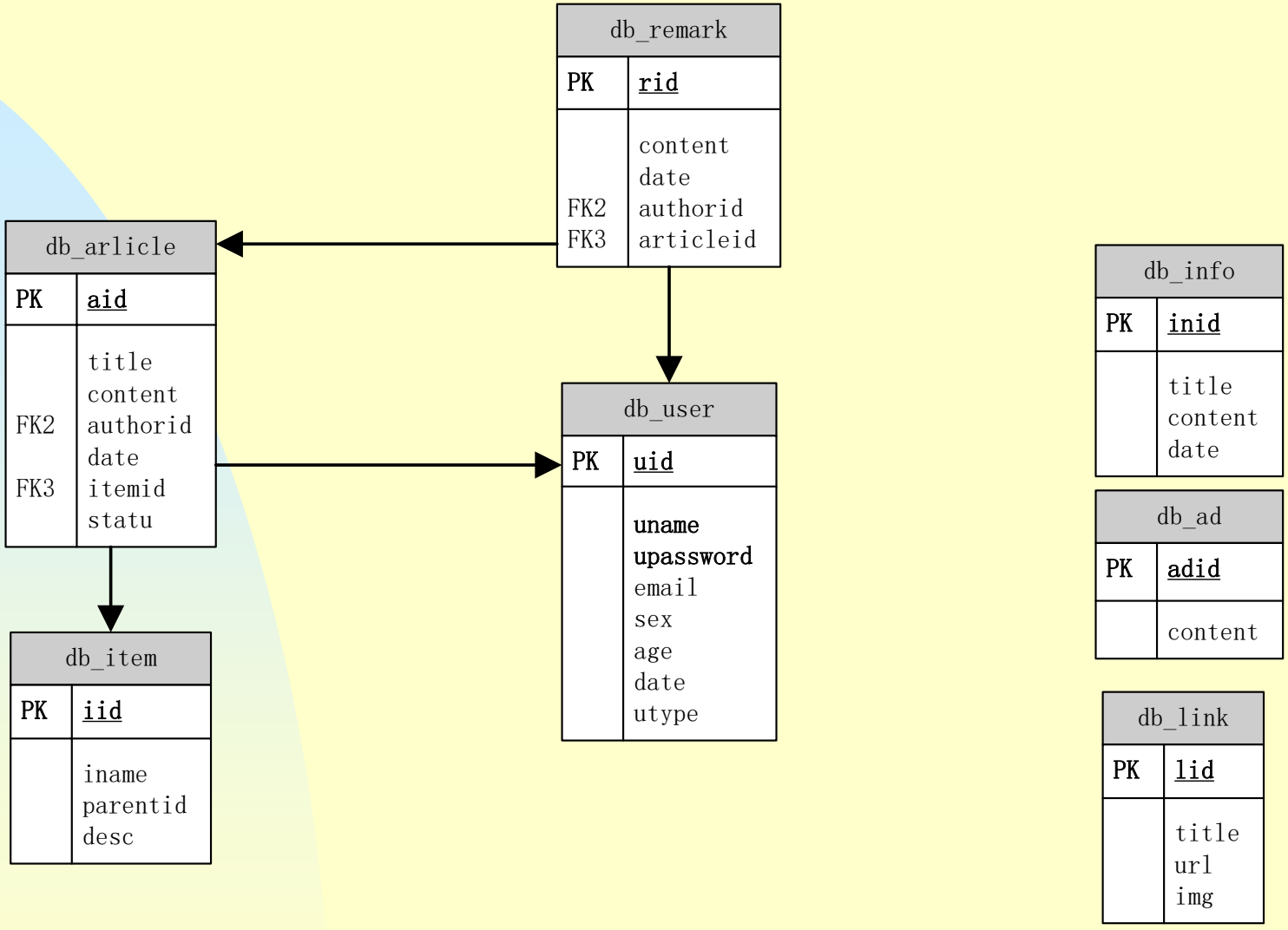
字　段	例　子
订单编号	001
订购日期	2000-2-3
顾客编号	AB001
顾客姓名	Tony
…	…



Orders

字　段	例　子
订单编号	001
订购日期	2000-2-3
顾客编号	AB001
…	…

- **如果一个关系满足2NF，并且除了主键以外的其他列都不传递依赖于主键列，则满足第三范式（3NF）**



最终生成：E-R实体关系图

- 常见完整性约束：
  - PRIMARY KEY　主码约束(主键)
  - UNIQUE　唯一性约束
  - NOT NULL　非空值约束
  - AUTO_INCREMENT 用于整数列默认自增1
  - UNSIGNED　无符号整数
  - DEFAULT default_value　默认值约束
  - DEFAULT current_timestamp 创建新记录时默认保存当前时间（仅适用timestamp数据列）
  - ON UPDATE current_timestamp 修改记录时默认保存当前时间（仅适用timestamp数据列）
  - CHARACTER SET name 指定字符集（仅适用字符串）

数据库编码命名规范

- 数据库：

  命名以字母“db”开头（小写），后面加数据库相关英文单词或缩写。如：db_CRM。

- 数据表：

  以字母“tb”开头（小写），后面加数据表相关英文单词或缩写。如：tb_User。

- 字段：

  一般采用英文单词或词组命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替。如：UserName。

数据库结构设计

数据表	功能
tb_content	内容数据表
tb_comment	用户评论表
tb_category	栏目信息表
tb_user	用户信息表

用户信息表

字段	类型	属性	描述
uid	unsigned int(10)	NOT NULL auto_increment PRIMARY KEY	用户id
username	varchar(20)	NOT NULL UNIQUE	用户名
password	varchar(50)	NOT NULL	用户密码
time	datatime	NOT NULL	注册时间
ip	varchar(16)	NOT NULL	登录IP
email	varchar(30)	NOT NULL UNIQUE	用户邮箱
level	tinyint(1)	NOT NULL default 1	用户级别 默认为普通权限 1

插入测试样本数据

- 最后，针对表结构特点，插入测试样本数据

用户界面设计原则

- （1）用户原则。人机界面设计首先要确立用户类型。划分类型可以从不同的角度，视实际情况而定。确定类型后要针对其特点预测他们对不同界面的反应。这就要从多方面设计分析。
- （2）信息最小量原则。人机界面设计要尽量减少用户记忆负担，采用有助于记忆的设计方案。
- （3）帮助和提示原则。要对用户的操作命令作出反应，帮助用户处理问题。系统要设计有恢复出错现场的能力，在系统内部处理工作要有提示，尽量把主动权让给用户。
- （4）媒体最佳组合原则。多媒体界面的成功并不在于仅向用户提供丰富的媒体，而应在相关理论指导下，注意处理好各种媒体间的关系,恰当选用。