目录

[一：UML事物 3](#_Toc393113844)

[1.1：构建事物：描述概念或物理元素 3](#_Toc393113845)

[1.2：行为事物：描述跨越空间和时间的行为 3](#_Toc393113846)

[1.3：分组事物：描述事物的组织结构 3](#_Toc393113847)

[1.4：注释事物：用来对模型中的元素进行说明 3](#_Toc393113848)

[二：UML关系 4](#_Toc393113849)

[2.1：依赖 4](#_Toc393113850)

[2.2：关联 4](#_Toc393113851)

[2.3：泛化 4](#_Toc393113852)

[2.4：实现 4](#_Toc393113853)

[三：UML图种类及特征 4](#_Toc393113854)

[3.1：用例图( Use Case Diagram ) 4](#_Toc393113855)

[3.2：类图(Class Diagram) 5](#_Toc393113856)

[3.3：对象图( Object Diagram ) 5](#_Toc393113857)

[3.4：顺序图(Sequence Diagram) 6](#_Toc393113858)

[3.5：协作图(Collaboration Diagram) 6](#_Toc393113859)

[3.6: 状态图(State Chart Diagram) 6](#_Toc393113860)

[3.7: 活动图(Activity Diagram) 7](#_Toc393113861)

[3.8: 构件图(Component Diagram) 7](#_Toc393113862)

[3.9: 部署图(Deployment Diagram) 7](#_Toc393113863)

[四：用例图 8](#_Toc393113864)

[4.1：用例图的事物及解释 8](#_Toc393113865)

[4.2：用例图的关系及解释 8](#_Toc393113866)

[4.3：例子 9](#_Toc393113867)

[五：类图 10](#_Toc393113868)

[5.1：类图的事物及解释 10](#_Toc393113869)

[5.2：类图的关系和解释 11](#_Toc393113870)

[5.3:例子 13](#_Toc393113871)

[六：顺序图 14](#_Toc393113872)

[6.1：顺序图的事物及解释 14](#_Toc393113873)

[6.2：例子 15](#_Toc393113874)

[七：状态图 16](#_Toc393113875)

[7.1：状态图的事物及解释 16](#_Toc393113876)

[7.2：例子 17](#_Toc393113877)

[八：活动图 19](#_Toc393113878)

[8.1:活动图的事物及解释 20](#_Toc393113879)

[8.2：活动图的关系 21](#_Toc393113880)

[8.3：例子 21](#_Toc393113881)

[九：构件图 22](#_Toc393113882)

[9.1：构建图的事物及解释 22](#_Toc393113883)

[9.2：构件图的关系及解释 23](#_Toc393113884)

[9.3：例子 23](#_Toc393113885)

[十：部署图 25](#_Toc393113886)

[10.1：部署图中的事物及解释 25](#_Toc393113887)

[10.2：例子 26](#_Toc393113888)

# 一：UML事物

## 1.1：构建事物：描述概念或物理元素

类：具有相同属性相同操作 相同关系相同语义的对象的描述

接口：描述元素的外部可见行为，即服务集合的定义说明

协作：描述了一组事物间的相互作用的集合

用例：代表一个系统或系统的一部分行为，是一组动作序列的集合

构件：系统中物理存在，可替换的部件

节点：运行时存在的物理元素

## 1.2：行为事物：描述跨越空间和时间的行为

交互：实现某功能的一组构件事物之间的消息的集合，涉及消息、动作序列、链接

状态机：描述事物或交互在生命周期内响应事件所经历的状态序列

## 1.3：分组事物：描述事物的组织结构

包：把元素组织成组的机制

## 1.4：注释事物：用来对模型中的元素进行说明

注解：对元素进行约束或解释的简单符号

# 二：UML关系

## 2.1：依赖

依赖(dependency)是两个事物之间的语义关系，其中一个事物(独立事物)发生变化，会影响到另一个事物(依赖事物)的语义

## 2.2：关联

关联(association)是一种结构关系，它指明一个事物的对象与另一个事物的对象间的联系

## 2.3：泛化

泛化(generalization)是一种特殊/一般的关系。也可以看作是常说的继承关系

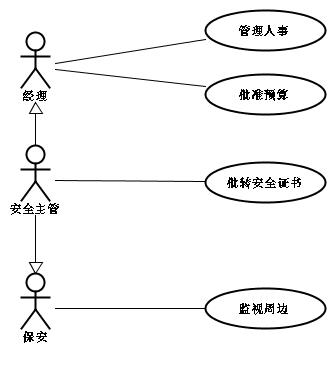
## 2.4：实现

实现(realization)是类元之间的语义关系，其中的一个类元指定了由另一个类元保证执行的契约

# 三：UML图种类及特征

## 3.1：用例图( Use Case Diagram )

用例图是从用户角度描述系统功能， 是用户所能观察到的系统功能的模型图，用例是系统中的一个功能单元



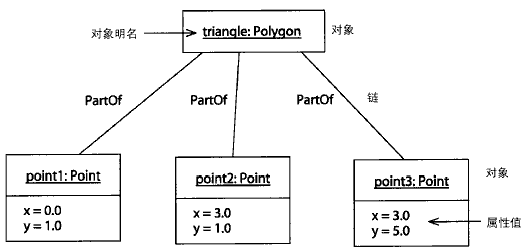
## 3.2：类图(Class Diagram)

※：类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联、依赖、聚合等，也包括类的内部结构(类的属性和操作)

※：类图是以类为中心来组织的，类图中的其他元素或属于某个类或与类相关联

## 3.3：对象图( Object Diagram )

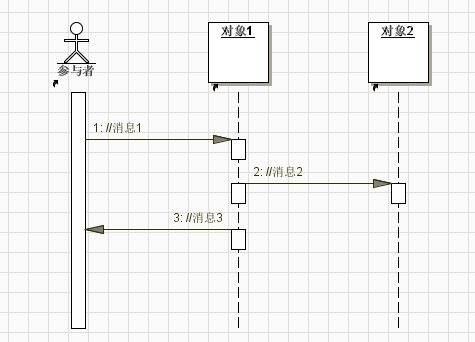
对象图是类图的实例，几乎使用与类图完全相同的标识。他们的不同点在于对象图显示类的多个对象实例，而不是实际的类



## 3.4：顺序图(Sequence Diagram)

※：顺序图显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互

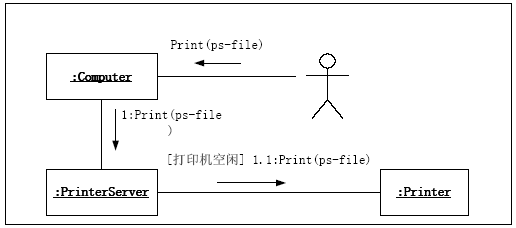
※：顺序图的一个用途是用来表示用例中的行为顺序。当执行一个用例行为时，顺序图中的每条消息对应了一个类操作或引起状态转换的触发事件



## 3.5：协作图(Collaboration Diagram)

※：协作图描述对象间的协作关系，协作图跟顺序图 相似，显示对象间的动态合作关系。除显示信息交换外，协作图还显示对象以及它们之间的关系.

※：协作图的一个用途是表示一个类操作的实现



## 3.6: 状态图(State Chart Diagram)

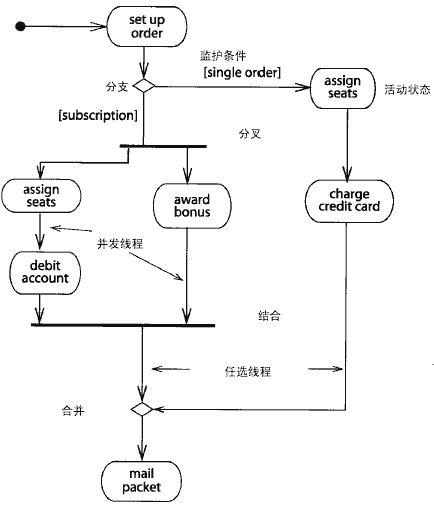
状态图是一个类对象所可能经历的所有历程的模型图。状态图由对象的各个状态和连接这些状态的转换组成



## 3.7: 活动图(Activity Diagram)

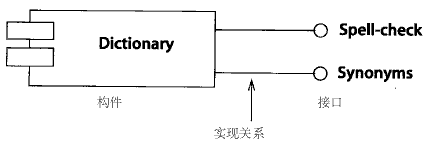
活动图是状态图的一个变体，用来描述执行算法的工作流程中涉及的活动

活动图描述了一组顺序的或并发的活动



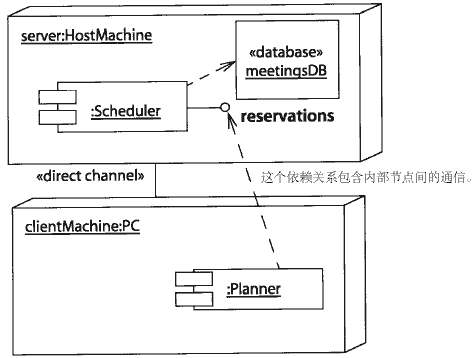
## 3.8: 构件图(Component Diagram)

构件图为系统的构件建模型—构件即构造应用的软件单元—还包括各构件之间的依赖关系，以便通过这些依赖关系来估计对系统构件的修改给系统可能带来的影响



## 3.9: 部署图(Deployment Diagram)

部署视图描述位于节点实例上的运行构件实例的安排。节点是一组运行资源，如计算机、设备或存储器。这个视图允许评估分配结果和资源分配



# 四：用例图

## 4.1：用例图的事物及解释

※：参与者(Actor)：在系统外部与系统直接交互的人或事物(如另一个计算机系统或一些可运行的进程)

※：用例(User Case)：系统外部可见的一个系统功能单元

## 4.2：用例图的关系及解释

※：参与者与用例之间的关系 🡪 关联：表示参与者与用例之间的交互，通信途径( → )

※：用例之间的关系

🡪包含：箭头出发的用例为基用例,箭头指向的用例为被包含的用例，称为包含用例；包含用例是必选的<<include>>

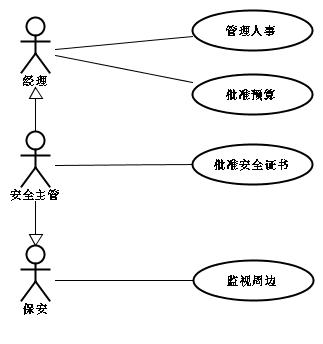
🡪扩展：箭头指向的用例为被扩展的用例，称为扩展用例，如果缺少扩展用例，不会影响到基用例的完整性，<<extend>>

※：参与者之间的关系 🡪 泛化：泛化关系是一般和特殊关系，发出箭头的一方代表特殊的一方，箭头指向的一方代表一般一方。特殊一方继承了一般方的特性并增加了新的特性。



## 4.3：例子

※：参与者之间的泛化关系

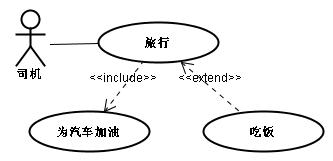


参与者：经理，安全主管，保安

用例：管理人事，批准预算，批准安全证书，监视周边

说明：在参与者之间不存在泛化关系的情况下，各个参与者参与 用例的情况分别是：经理参与用例管理人事和批准预算；安全主管参与用例批准安全证书；保安参与用例监视周边。由于安全主管与经理，安全主管与保安之间泛化关系的存在，意味着安全主管可以担任经理和保安的角色，就能够参与经理和保安参与的用例。这样，安全主管就可以参与全部4个用例。但经理或者保安却不能担任安全主管的角色，也就不能参与用例批准安全证书。

※：用例之间扩展和包含关系



用例的上下文是：短途旅行但汽车的油不足以应付全部路程。那么为汽车加油的动作在旅行的每个场景(事件流)中都会出现，不加油就不会完成旅行。吃饭则可以由司机决定是否进行，不吃饭不会影响旅行的完成。

※：航空售票的用例图



参与者(actor)：clerk，监督员，信用卡服务商，信息亭

用例(use case)： Buy tickets， Buy Subscription， Make charges， Survey sales

参与者Clerk参与(或称发起)Buy tickets和Buy Subscription 两个用例(关联关系)。这两个用例的事件流都包含Make charges用例(包含关系)。

系统由：Buy tickets， Buy Subscription， Make charges， Survey sales组成。

该系统主要包含：Buy tickets， Buy Subscription， Make charges，Survey sales这几个功能。

该系统主要面向的用户(参与者)：clerk，监督员，信用卡服务商，信息亭。

# 五：类图

## 5.1：类图的事物及解释

※：类

※：接口

一组操作的集合，只有操作的声明而没有实现

※：抽象类

不能被实例化的类，一般至少包含一个抽象操作

※：模板类

一种参数化的类，在编译时把模版参数绑定到不同的数据类型，从而产生不同的类

## 5.2：类图的关系和解释

※：聚合关系（特殊关联关系，指明一个聚集（整体）和组成部分之间的关系）

※：组合关系(语义更强的聚合，部分和整体具有相同的生命周期)



※：泛化关系(在面向对象中一般称为继承关系，存在于父类与子类、父接口与子接口之间)



※：实现关系



※：依赖关系

描述了一个类的变化对依赖于它的类产生影响的情况。有多种表现形式，例如绑定(bind)、友元(friend)等





## 5.3:例子

A: 图形编辑器

※：图形编辑器一般都具有一些基本图形，如直线、矩形等，用户可以直接使用基本图形画图，也可以把基本图形组合在一起创建复杂图形

※：如果区别对待基本图形和组合图形，会使代码变得复杂，而且多数情况下用户认为二者是一样的

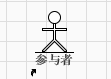
※：组合模式可以用相同的方式处理两种图形



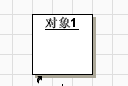
# 六：顺序图

## 6.1：顺序图的事物及解释

※：参与者



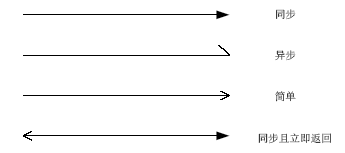
※：对象



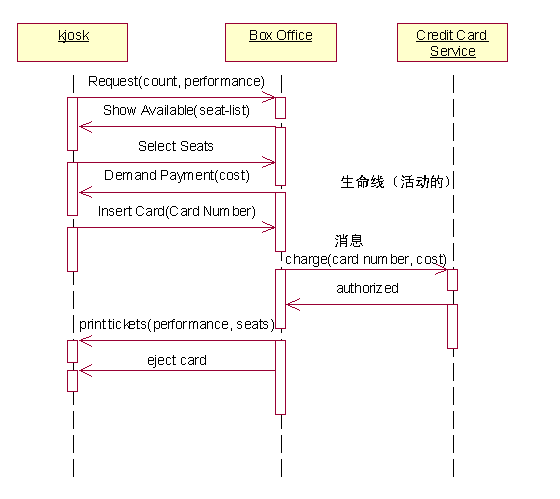
※：生命线

※：消息符号



## 6.2：例子



※：从这个例子中可以看出：

Kjosk类中的操作有

Show Available (seat-list)

Demand Payment (cost)

printtickets (performance, seats)

eject card

Box Office中的操作有

Request (count, performance)

Select Seats

Insert Card (Card Number)

authorized

Credit Card Service类中的操作有

charge(card number, cost)

※：此图中存在的事物有

对象(信息亭 Kjosk ，售票中心 Box Office， 信用卡服务 Credit Card Service )，生命线，消息符号。

信息亭发Request (count, performance)消息给售票中心，表示调用售票中心类的Request (count, performance)操作，来查询演出的信息。

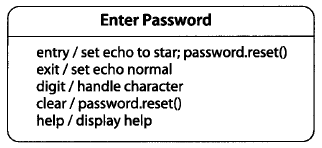
售票中心发Show Available(seat-list)消息给信息亭，表示调用信息亭类中的Show Available(seat-list)操作，给出可用的座位表。

# 七：状态图

## 7.1：状态图的事物及解释

※：状态

上格放置名称，下格说明处于该状态时，系统或对象要做的工作



※：转移

转移上标出触发转移的事件表达式。如果转移上未标明事件，则表示在源状态的内部活动执行完毕后自动触发转移

※：开始

初始状态(一个)

※：结束

终态(可以多个)

7.2：状态的可选活动表

※：入口动作

进入某一状态时执行的动作

entry/action

※：出口动作

离开某一状态时执行的动作

exit/action

※：外部转换

引起状态转换或自身转换，同时执行一个具体的动作，包括引起入口动作和出口动作被执行的转换

e(a:T)[exp]/action

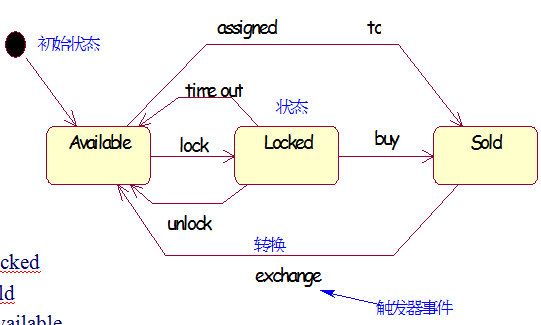
※：内部转换

引起一个动作的执行但不引起状态的改变或不引起入口动作或出口动作的执行

e(a:T)[exp]/action

## 7.2：例子

7.2.1:购票系统



※：图中包含以下状态

初始状态

Available状态

Locked状态

Sold状态

※：状态间的转移

初始状态Available状态

票被预订(lock)：AvailableLocked

预定后付款(buy)：LockedSold

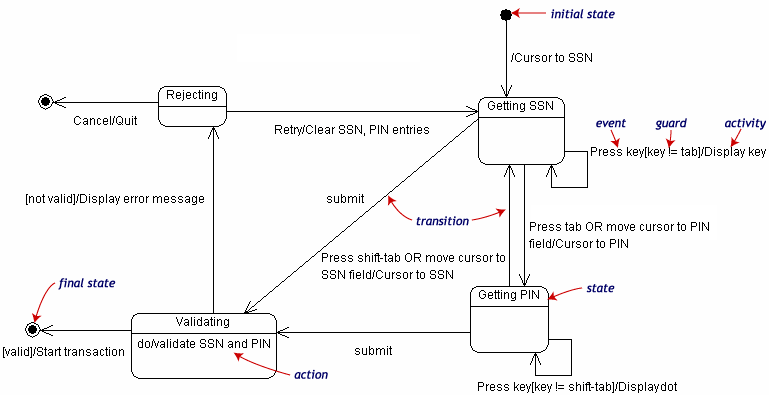
预定解除(unlock)：LockedAvailable

预定过期(time out)：LockedAvailable

直接购买(assigned to)：AvailableSold

换其它票(exchang) ，该票重有效：SoldAvailable

7.2.2: 网上银行登陆系统



登陆要求提交个人社会保险号(SSN)和密码(PIN)经验证有效后登陆成功。

登陆过程包括以下状态:

初态(Initial state)

获取社会保险号状态(Getting SSN)

获取密码状态(Getting PIN)

验证状态(Validating)

拒绝状态(Rejecting)

终态 (Final state)

|  |  |  |
| --- | --- | --- |
| **出发状态** | **动作** | **到达状态** |
| **Initial state** | 移动鼠标到 SSN | **Getting SSN** |
| **Getting SSN** | 键入非tab键，显示键入内容 | **Getting SSN** |
| 键入tab键，或移动鼠标到BIN | **Getting PIN** |
| 提交 | **Validating** |
| **Getting PIN** | 键入非shift-tab键，显示 “ \* ” | **Getting PIN** |
| 键入shift-tab键，或移动鼠标到SSN | **Getting SSN** |
| 提交 | **Validating** |
| **Validating** | 验证提交信息有效，状态转移 | **Final state** |
| 验证提交信息无效，显示错误信息 | **Rejecting** |
| **Rejecting** | 退出 | **Final state** |
| 重试，清除无效的SSN，PIN | **Getting SSN** |

# 八：活动图

## 8.1:活动图的事物及解释

※：活动 (ActionState)：

动作的执行



※：起点 (InitialState)

活动图的开始



※：终点(FinalState)

活动图的终点



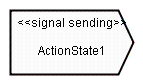
※：对象流(ObjectFlowState)

活动之间的交换的信息



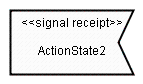
※：发送信号(signalSending)

活动过程中发送事件，触发另一活动流程



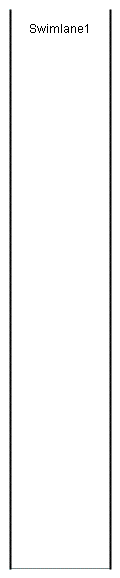
※：接收信号(SignalReceipt)

活动过程中接收事件，接收到信号的活动流程开始执行



※：泳道(SwimLane)

活动的负责者



## 8.2：活动图的关系

※：迁移(transition)

活动的完成与新活动的开始



※：分支(junction point)

根据条件，控制执行方向



※：分叉(fork)

以下的活动可并发执行



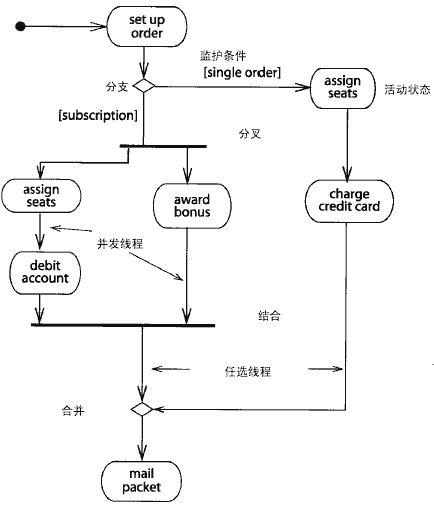
※：结合(join)

以上的并发活动再此结合



## 8.3：例子

8.3.1:订票



本活动图描述一个处理订单的用例执行过

(1)执行setup order

(2)根据order的类型是执行不同的分支：

single order：执行assign seat、charge credit card

subscription：同时执行assignseats、debit account或

award bonus

single order与subscription两步可同时进行

(3) 最后mail packet。

# 九：构件图

## 9.1：构建图的事物及解释

※：构件：

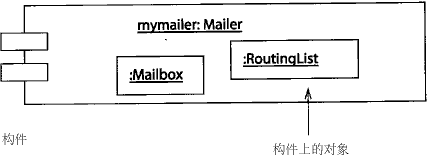
指系统中可替换的物理部分，构件名字(如图中的Dictionary)标在矩形中，提供了一组接口的实现。

※：接口：

外部可访问到的服务 (如图中的Spell-check)

※：构件实例：

节点实例上的构件的一个实例，冒号后是该构件实例的名字



## 9.2：构件图的关系及解释

※：实现关系

构件向外提供的服务

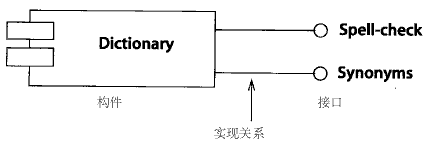
※：依赖关系

构件依赖外部提供的服务(由构件到接口)。

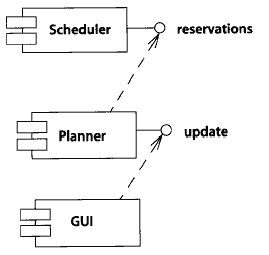
## 9.3：例子

9.3.1：Dictionary字典

该构件向外提供两个接口，即两个服务Spell-check拼写检查、Synonyms同义词。



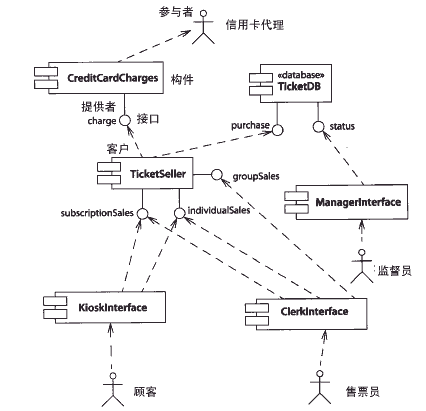
9.3.2：



图中“Planner计划者”构件向外提供一个“update更新”接口服务。

同时，该构件要求外部接口提供一个“Reservations预定”服务。

9.3.3：售票系统



※：图中依赖关系包括

顾客需要信息亭接口提供服务

售票员需要职员接口提供服务

信用卡付款需要信用卡代理提供服务

职员接口需要预订销售、个人销售和团体销售提供服务

管理接口需要数据库状态提供服务

售票处需要付款和购买提供服务

等等……

※：图中实现关系包括

信用卡付款提供付款服务

票数据库提供购买和状态查询服务

售票处提供预订购买、个人购买和团体购买服务

※：情景描述、

情景一：

购买个人票可以通过公用信息亭订购也可直接向售票员购买，但购买团体票只能通过售票员。

情景二：

买票的人可以根据任意选择预订销售或个人销售或团体销售中的一种方式，售票处为了方便销售，需要信用卡付款服务的支持，同时也必然需要票数据库处在有票可卖的状况中。

# 十：部署图

## 10.1：部署图中的事物及解释

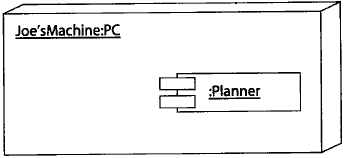
※：节点

节点用一长方体表示，长方体中左上角的文字是节点的名字 (如图中的Joe’sMachine:PC) 。

节点代表一个至少有存储空间和执行能力的计算资源。

节点包括计算设备和(至少商业模型中的)人力资源或者机械处理资源，可以用描述符或实例代表。

节点定义了运行时对象和构件实例(如图中的Planner构件实例)驻留的位置。



※：构件

系统中可替换的物理部分

※：接口

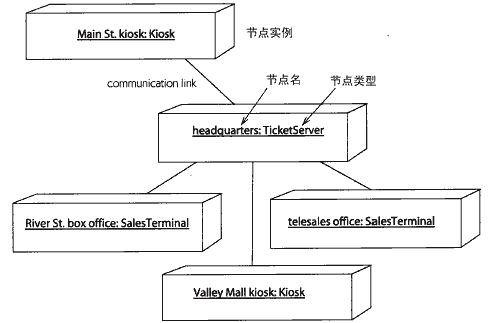
外部可访问的服务

※：构件实例

构件的一个实例

## 10.2：例子

10.2.1：实现层部署图

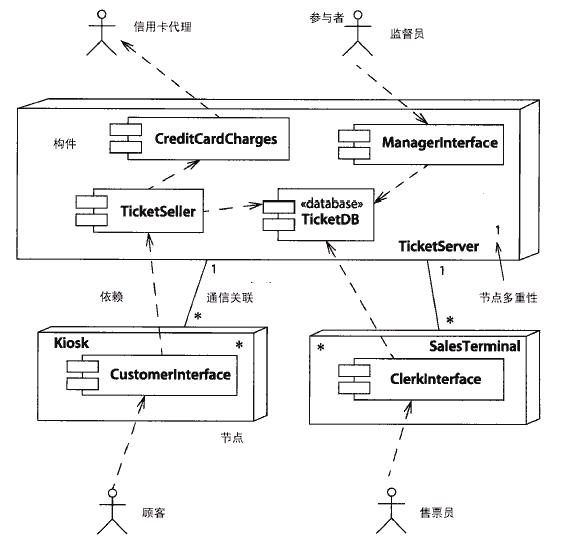


实例层部署图描述各节点和它们之间的连接。

本图中的信息与上张描述层部署图中的内容是相互对应的。

图中的关系是各个节点之间存在的通信关系。

10.2.2：描述层部署图



※：图中包括的各种关系如下：

通信链关系(不带箭头的直线)

TicketServe票服务器与Kiosk信息厅之间存在一对多的通信关联；与SalesTerminal售票终端也存在一对多的通信关联；

依赖关系(带箭头的虚线)

TicketSeller售票构件依赖CreditCardCharges信用卡付款构件和TicketDB票数据库构件提供的服务。

※：节点TicketServer(售票服务)上的构件

CreditCardCharges/ManagerInterface/ TicketSeller/TicketDB

节点Kiosk(信息亭)上的构件:

CustomerInterface

SalesTerminal(销售终端)上的构件:

ClerkInterface

※：图中顾客购票的情景如下：

顾客通过位于Kiosk节点的顾客接口控件进行购票的操作，该顾客接口构件的购票操作依赖于处于TicketServer节点上的售票构件提供的服务，售票构件要完成售票操作，又要依赖统一节点上信用卡付款构件提供的付款服务和票数据库构件