一、作为一种建模语言，UML的定义包括UML语义和UML表示法两个部分。

* UML语义：描述基于UML的精确元模型定义。
* UML表示法：定义UML符号的表示法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。这些图形符号和文字所表达的是应用级的模型，在语义上它是UML元模型的实例。

二、标准建模语言UML可以由下列5类图来定义。

* 用例图：从用户角度描述系统功能，并指出各功能的操作者。
* **静态图：包括类图和对象图。**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，也包括类的属性和操作，类图描述的是一种静态关系，在系统的整个生命周期都是有效的。对象图是类图的实例，几乎使用与类图完全相同的标识。一个对象图是类图的一个实例。由于对象存在生命周期，因此对象图只能在系统某一时间段存在。
* 行为图：描述系统的动态模型和组成对象间的交互关系，**包括状态图和活动图**。状态图描述类的对象所有可能的状态以及事件发生时状态的转移条件，状态图是对类图的补充，活动图描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并进行活动。
* 交互图：描述对象间的交互关系，包括**时序图和协作图**。时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；协作图描述对象间的协作关系，协作图跟时序图相似，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。如果强调时间和顺序，则使用时序图；如果强调上下级关系，则选择协作图。
* 实现图：包括**组件图和部署图**。组件图描述代码部件的物理结构及各部件之间的依赖关系，组件图有助于分析和理解部件之间的相互影响程度；部署图定义系统中软硬件的物理体系结构。

采用UML来设计系统时，第一步是描述需求；第二步根据需求建立系统的静态模型，以构造系统的结构；第三步是描述系统的行为。其中在第一步与第二步中所建立的模型都是静态的，包括用例图、类图、对象图、组件图和部署图等5种图形，是标准建模语言UML的静态建模机制。其中第三步中所建立的模型或者可以执行，或者表示执行时的时序状态或交互关系。它包括状态图、活动图、时序图和协作图等4种图形，是标准建模语言UML的动态建模机制。

首先对UML中的各个图的功用做一个简单介绍：

**1、用例图**

描述角色以及角色与用例之间的连接关系。说明的是谁要使用系统，以及他们使用该系统可以做些什么。一个用例图包含了多个模型元素，如系统、参与者和用例，并且显示了这些元素之间的各种关系，如泛化、关联和依赖。

 用例图（Use Case Diagram）是被称为参与者(Actor)的外部用户所能观察到的系统功能的模型图

* 列出系统中的用例和参与者
* 显示哪个参与者参与了哪个用例的执行

 核心概念

用例：系统中的一个功能单元，可以被描述为参与者与系统之间的一次交互作用

* 参与者、参与者泛化
* 用例与参与者之间的关系：关联
* 用例之间关系：扩展、包括、泛化

 推荐使用场合

业务建模、需求获取、定义

作者：小胡子杰克  
链接：https://www.jianshu.com/p/4970d5e2b2bf  
来源：简书  
简书著作权归作者所有，任何形式的转载都请联系作者获得授权并注明出处。

**2、类图**

类图是描述系统中的类，以及各个类之间的关系的静态视图。能够让我们在正确编写代码以前对系统有一个全面的认识。类图是一种模型类型，确切的说，是一种静态模型类型。

是软件的蓝图，详细描述了系统内各个对象的相关的类，以及这些类之间的静态关系。

核心概念：类、接口、依赖、关联、泛化、实现

类图展示实体类的静态关系：

**3、对象图**

与类图极为相似，它是类图的实例，对象图显示类的多个对象实例，而不是实际的类。它描述的不是类之间的关系，而是对象之间的关系。

表示在某一时刻类的对象静态结构和行为。

核心概念：对象、链接、多重性

**4、活动图**

描述用例要求所要进行的活动，以及活动间的约束关系，有利于识别并行活动。能够演示出系统中哪些地方存在功能，以及这些功能和系统中其他组件的功能如何共同满足前面使用用例图建模的商务需求。

* 活动图（Activity Diagram）
  + 通过动作来组织，主要用于描述某一方法、机制或用例的内部行为
* 核心概念
  + 状态、活动、组合活动、对象
  + 转移、分支
  + 并发、同步
  + 泳道
* 推荐使用场合
  + 业务建模、需求、类设计

**5、状态图**

    描述类的对象所有可能的状态，以及事件发生时状态的转移条件。可以捕获对象、子系统和系统的生命周期。他们可以告知一个对象可以拥有的状态，并且事件(如消息的接收、时间的流逝、错误、条件变为真等)会怎么随着时间的推移来影响这些状态。一个状态图应该连接到所有具有清晰的可标识状态和复杂行为的类；该图可以确定类的行为，以及该行为如何根据当前的状态变化，也可以展示哪些事件将会改变类的对象的状态。状态图是对类图的补充。

**6、序列图**（顺序图）

序列图是用来显示你的参与者如何以一系列顺序的步骤与系统的对象交互的模型。顺序图可以用来展示对象之间是如何进行交互的。顺序图将显示的重点放在消息序列上，即强调消息是如何在对象之间被发送和接收的。

* 顺序图（Sequence Diagram）
  + 用于显示对象间的交互活动
  + 关注对象之间消息传送的时间顺序
* 核心概念
  + 对象、生命线、激活、交互、消息
  + 交互帧(Interaction Frame)
* 推荐使用场合
  + 用例分析、用例设计

**7、协作图**

    和序列图相似，显示对象间的动态合作关系。可以看成是类图和顺序图的交集，协作图建模对象或者角色，以及它们彼此之间是如何通信的。如果强调时间和顺序，则使用序列图；如果强调上下级关系，则选择协作图；这两种图合称为交互图。

**8、构件图** （组件图）

    描述代码构件的物理结构以及各种构建之间的依赖关系。用来建模软件的组件及其相互之间的关系，这些图由构件标记符和构件之间的关系构成。在组件图中，构件时软件单个组成部分，它可以是一个文件，产品、可执行文件和脚本等。

**9、部署图** （配置图）

    是用来建模系统的物理部署。例如计算机和设备，以及它们之间是如何连接的。部署图的使用者是开发人员、系统集成人员和测试人员。

# 几种图的区别：

一：这九种模型图各有侧重，

1:用例图侧重描述用户需求，

2:类图侧重描述系统具体实现；

二：描述的方面都不相同，

1:类图描述的是系统的结构，

2:序列图描述的是系统的行为；

三：抽象的层次也不同，

1：构件图描述系统的模块结构，抽象层次较高，

2：类图是描述具体模块的结构，抽象层次一般，

3：对象图描述了具体的模块实现，抽象层次较低。

在有的文献书籍中，将这九种模型图分为三大类：

结构分类、动态行为和模型管理：

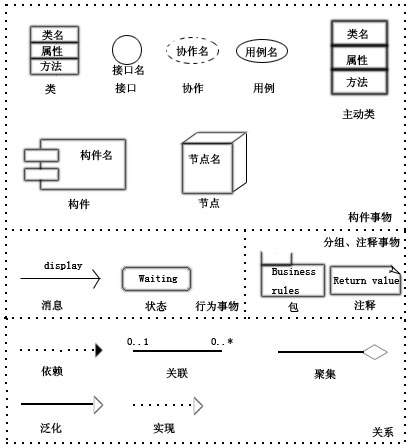
**1：结构分类包括用例图、类图、对象图、构件图和部署图，**

**2：动态行为包括状态图、活动图、顺序图和协作图，**

**3：模型管理则包含类图。**

**画图说明**

UML(统一建模语言)：是面向对象的可视化建模的一种语言。是数据库设计过程中，在E-R图（实体-联系图）的设计后的进一步建模。  
UML中有3种构造块：事物、关系和图，事物是对模型中最具有代表性的成分的抽象；关系是把事物结合在一起；图聚集了相关的的事物。具体关系图标如下：

  
说明：  
构件事物是名词，是模型的静态部分。  
行为事物是动态部分，表示行为。  
分组事物是组织部分。  
注释事物是解释部分。

依赖：一个事物变化会引起另一个事物变化。  
聚集：特殊的关联，描述整体与部分的组合关系。  
泛化：是一种特殊与一般的关系，如子元素（特殊）与父元素（一般），箭头指向父元素。  
实现：类元之间的关系，其中一个类元指定了由另一个类元保证执行的契约。一般用在接口和实现他们的类之间或用例和实现它们的协作之间。

UML提供9种视图：类图、对象图，用例图，序列图、协作图，状态图、活动图，构件图和部署图。

在UML系统开发中有三个主要的模型：

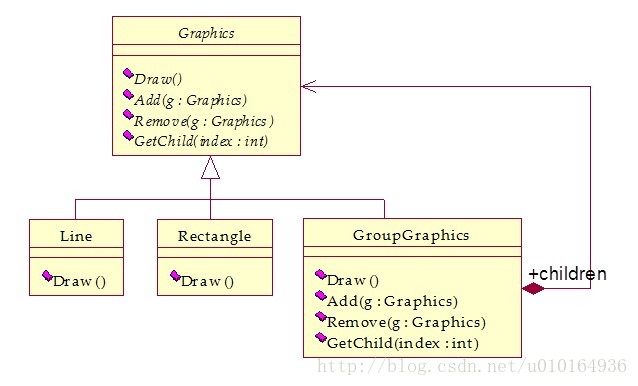
功能模型: 从用户的角度展示系统的功能，包括用例图。  
对象模型: 采用对象，属性，操作，关联等概念展示系统的结构和基础，包括类图。  
动态模型: 展现系统的内部行为。 包括序列图，活动图，状态图。

下面具体说明：

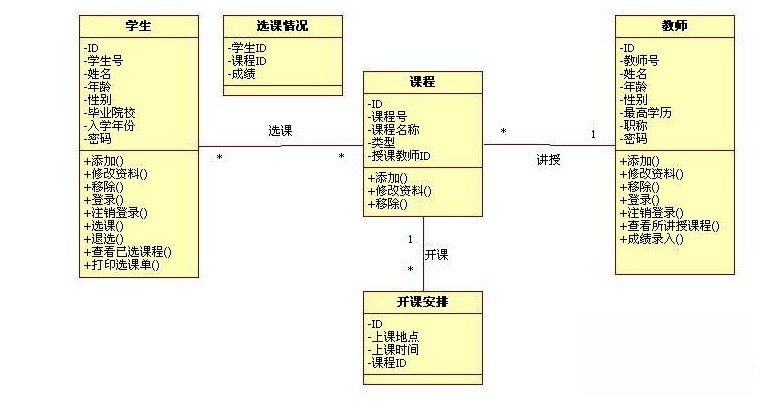
## 1.类图：

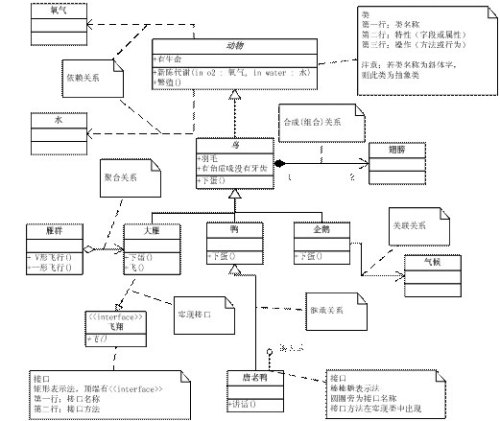
**类图描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联、依赖、聚合等，也包括类的内部结构(类的属性和操作)**

**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**

****

描述一组对象、接口、协作等事物之间的关系。如下图(摘自网络)：

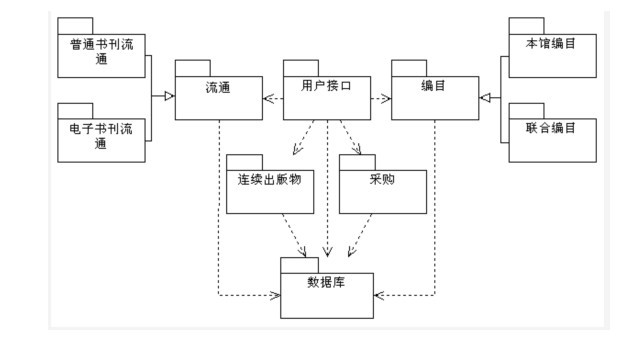


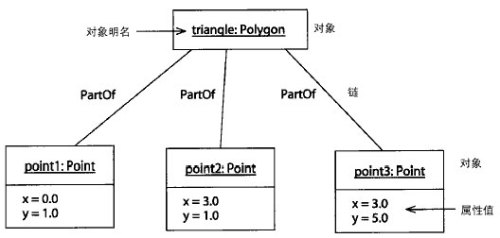


注：#表示protected，+表示Public，-表示private

## 2.对象图：

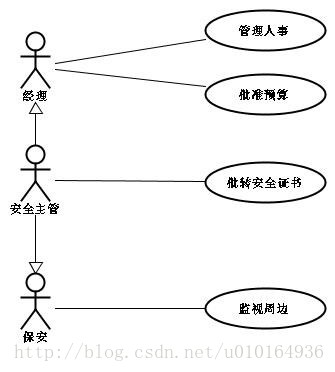
描述一组对象之间的关系，是具有具体属性值和行为的一个具体事物，其是类图中所建事物实例的静态快照，其与类图的主要区别是一个是抽象的,而对象图是具体的。如下图(摘自网络)：



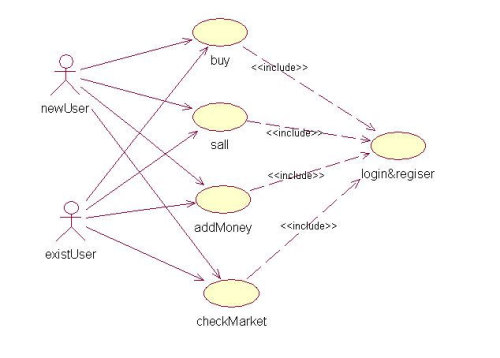
  
**对象图是类图的实例，几乎使用与类图完全相同的标识。他们的不同点在于对象图显示类的多个对象实例，而不是实际的类**

## 3.用例图：

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



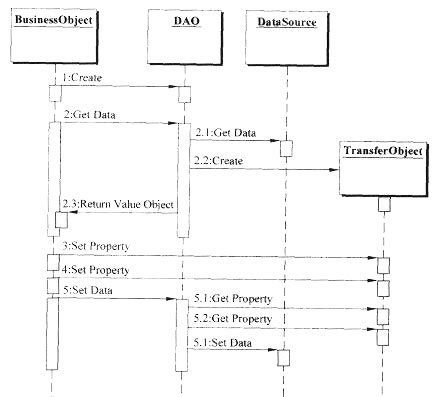
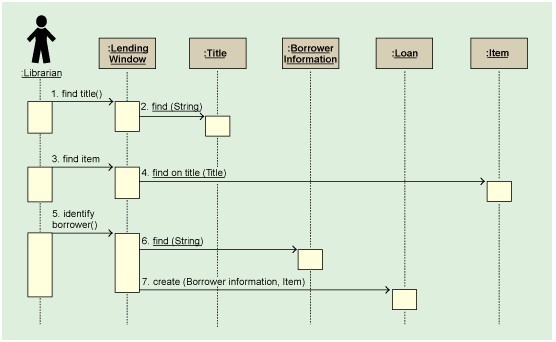
描述一组用例、参与者以及它们之间的关系，其展示的是该系统在它的外面环境中所提供的外部可见服务。如下图(摘自网络)：



## 4.交互图：

包括序列图（顺序图）和协作图，两者对应，顺序图是强调消息时间顺序，有对象生命线和控制焦点。协作图是强调接收和发送消息的对象的结构组织，有路径和顺序号。如下图(摘自网络)：

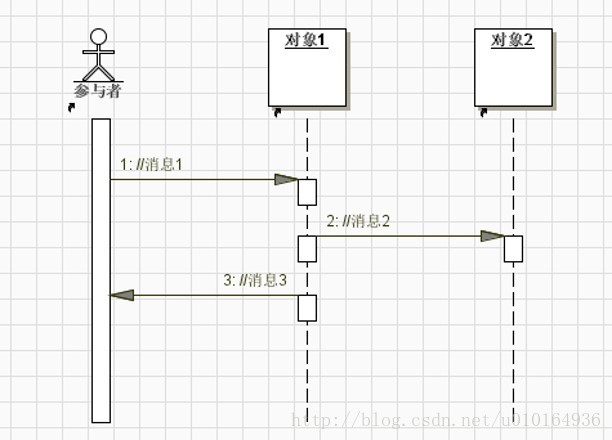
### 序列图：



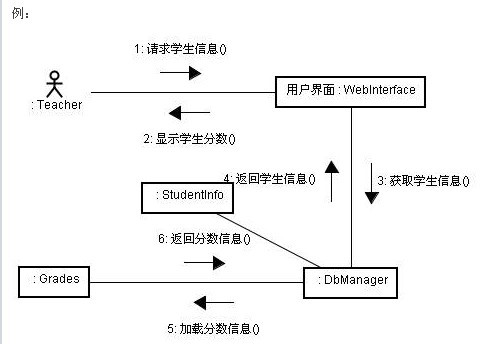
### **顺序图**

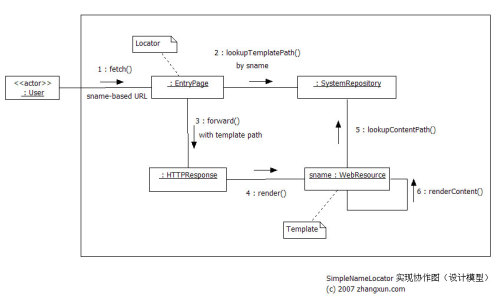
**显示对象之间的动态合作关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互**

**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**

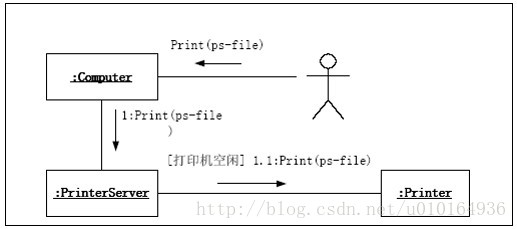
****

## 5.协作图：

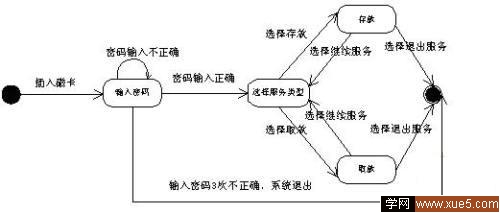


  
**协作图描述对象间的协作关系，协作图跟顺序图 相似，显示对象间的动态合作关系。除显示信息交换外，协作图还显示对象以及它们之间的关系.**

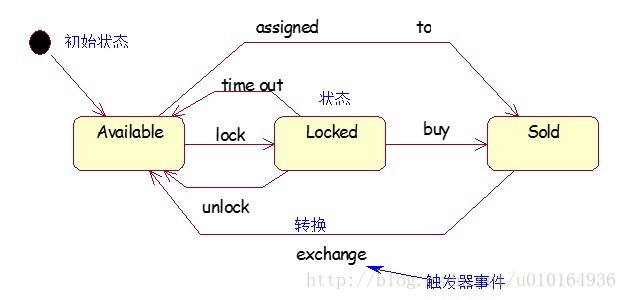
**协作图的一个用途是表示一个类操作的实现**

****

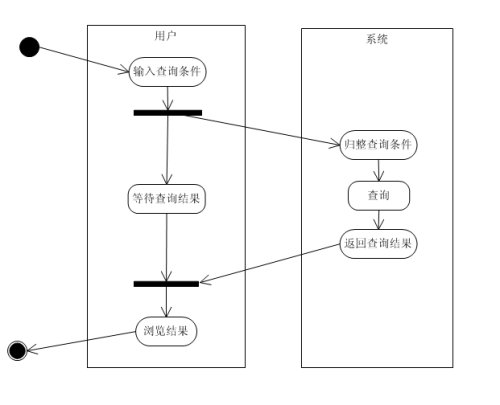
## 6.状态图：

展示了一个状态机，由状态、转换、事件和活动组成。强调事件行为的顺序。如下图(摘自网络)：  


**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**

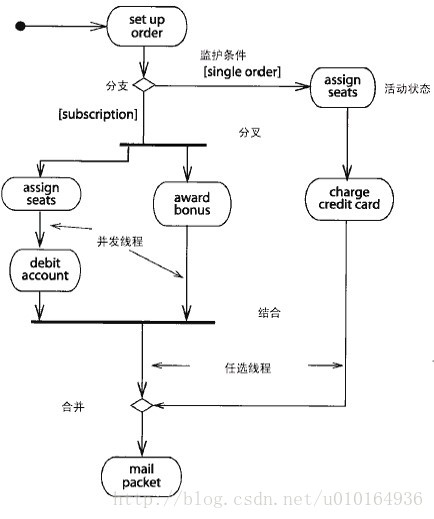
****

## 7.活动图：

是一种特殊的状态图，实现一个活动到另一个活动的流程。如下图(摘自网络)：  


**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**

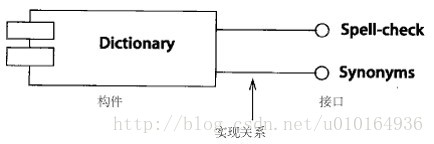
**活动图描述了一组顺序的或并发的活动**

****

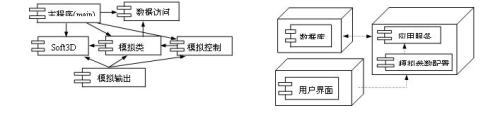
## 8.构件图

构件图展示一组构件之间的组织和依赖关系，并以全局的模型展示出来。

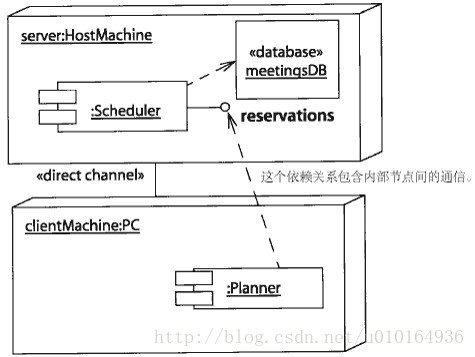
**构件图为系统的构件建模型—构件即构造应用的软件单元—还包括各构件之间的依赖关系，以便通过这些依赖关系来估计对系统构件的修改给系统可能带来的影响**

****

## 9.部署图：

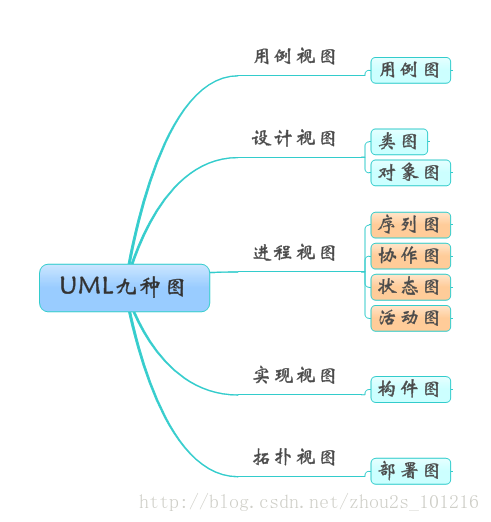
部署图是构件的配置及描述系统如何在硬件上部署。如下图(摘自网络)：  


**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**

****

# 一、基本概念

　　如下图所示，UML图分为用例视图、设计视图、进程视图、实现视图和拓扑视图，又可以静动分为静态视图和动态视图。静态图分为：用例图，类图，对象图，包图，构件图，部署图。动态图分为：状态图，活动图，协作图，序列图。



## **1、用例图（UseCase Diagrams）：**

  　　用例图主要回答了两个问题：1、是谁用软件。2、软件的功能。从用户的角度描述了系统的功能，并指出各个功能的执行者，强调用户的使用者，系统为执行者完成哪些功能。



## **2、类图（Class Diagrams）：**

　用户根据用例图抽象成类，描述类的内部结构和类与类之间的关系，是一种静态结构图。 在UML类图中，常见的有以下几种关系: 泛化（Generalization）,  实现（Realization），关联（Association)，聚合（Aggregation），组合(Composition)，依赖(Dependency)。

  　　　各种关系的强弱顺序： 泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖

### 2.1.泛化

　　　　【泛化关系】：是一种继承关系，表示一般与特殊的关系，它指定了子类如何继承父类的所有特征和行为。例如：老虎是动物的一种，即有老虎的特性也有动物的共性。



### 2.2.实现

　　　　【实现关系】：是一种类与接口的关系，表示类是接口所有特征和行为的实现。



### 2.3.关联

 　　　　【关联关系】：是一种拥有的关系，它使一个类知道另一个类的属性和方法；如：老师与学生，丈夫与妻子关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。

        　 【代码体现】：成员变量



### 2.4.聚合

　　　　【聚合关系】：是整体与部分的关系，**且部分可以离开整体而单独存在**。如车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。

**聚合关系是关联关系的一种**，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。

        　 【代码体现】：成员变量



### 2.5.组合

　　　　【组合关系】：是**整体与部分的关系，但部分不能离开整体而单独存在**。如公司和部门是整体和部分的关系，没有公司就不存在部门。

**组合关系是关联关系的一种，是比聚合关系还要强的关系**，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

　　　　【代码体现】：成员变量

　　　　【箭头及指向】：带实心菱形的实线，菱形指向整体



### 2.6.依赖

   　　【依赖关系】：是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖.

        【代码表现】：**局部变量、方法的参数或者对静态方法的调用**

        【箭头及指向】：带箭头的虚线，指向被使用者



### 2.7 各种类图关系



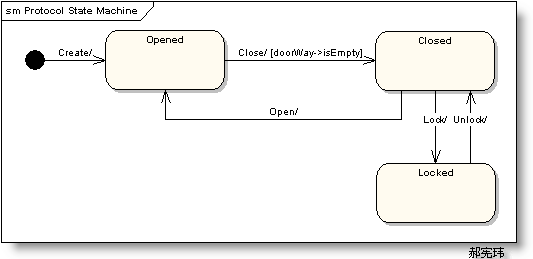
## **3、对象图（Object Diagrams）:**

　描述的是参与**交互的各个对象在交互过程中某一时刻的状态**。对象图可以被看作是类图在某一时刻的实例。



## ****4、状态图（Statechart Diagrams）：****

是一种由状态、变迁、事件和活动组成的状态机，用来描述类的对象所有可能的状态以及时间发生时状态的转移条件。



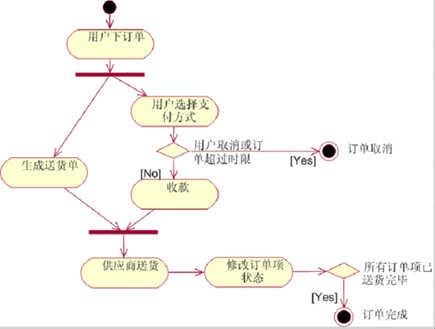
## **5、活动图（Activity Diagrams）：**

是状态图的一种特殊情况，这些状态大都处于活动状态。本质是一种流程图，它描述了活动到活动的控制流。

交互图强调的是对象到对象的控制流，而活动图则强调的是从活动到活动的控制流。

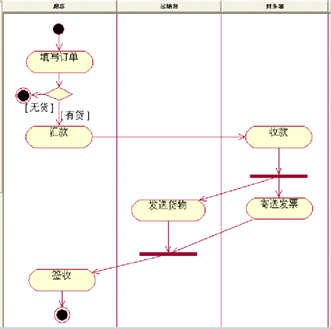
活动图是一种表述过程基理、业务过程以及工作流的技术。

它可以用来对业务过程、工作流建模，也可以对用例实现甚至是程序实现来建模。



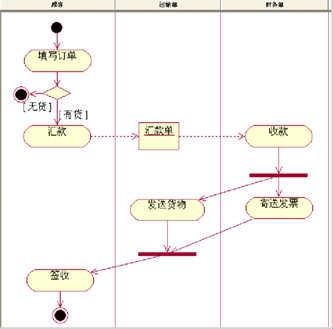
### 5.1 带泳道的活动图

　　　　　　泳道表明每个活动是由哪些人或哪些部门负责完成。



### 5.2 带对象流的活动图

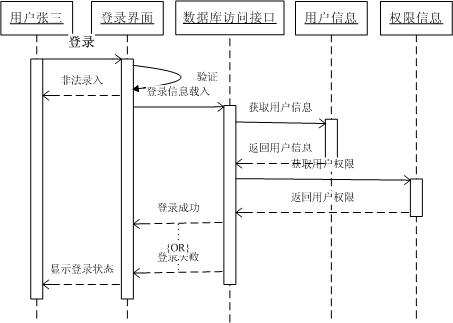
　　　　用活动图描述某个对象时，可以把涉及到的对象放置在活动图中，并用一个依赖将其连接到进行创建、修改和撤销的动作状态或者活动状态上，对象的这种使用方法就构成了对象流。对象流用带有箭头的虚线表示。



## **6、序列图-时序图（Sequence Diagrams）：**

交互图的一种，描述了对象之间消息发送的先后顺序，强调时间顺序。

　　　　序列图的主要用途是把用例表达的需求，转化为进一步、更加正式层次的精细表达。用例常常被细化为一个或者更多的序列图。同时序列图更有效地描述如何分配各个类的职责以及各类具有相应职责的原因。

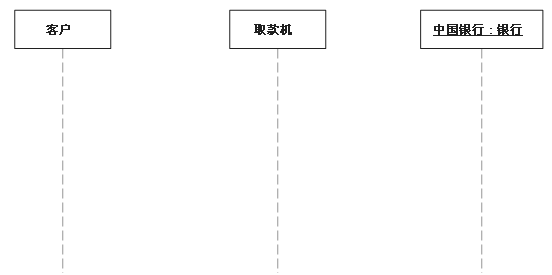


　　消息用从一个对象的生命线到另一个对象生命线的箭头表示。箭头以时间顺序在图中从上到下排列。

　　序列图中涉及的元素：

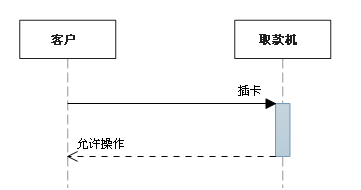
### 6.1 生命线

　　　　生命线名称可带下划线。当使用下划线时，意味着序列图中的生命线代表一个类的特定实例。



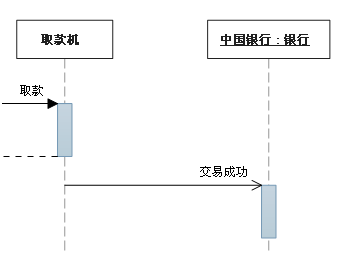
### 6.2 同步消息

　　　　同步等待消息

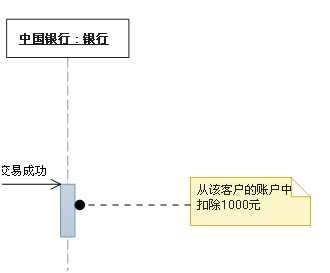


### 6.3 异步消息

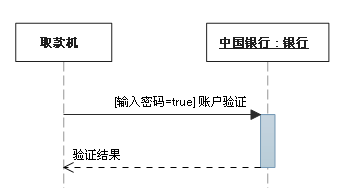
　　　　异步发送消息，不需等待



### 6.4 注释



### 6.5 约束

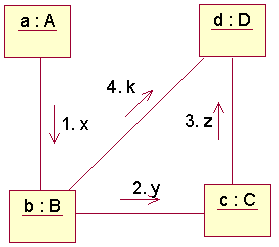


### 6.6 组合

**组合片段用来解决交互执行的条件及方式**。它允许在序列图中直接表示逻辑组件，用于通过指定条件或子进程的应用区域，为任何生命线的任何部分定义特殊条件和子进程。常用的组合片段有：抉择、选项、循环、并行。

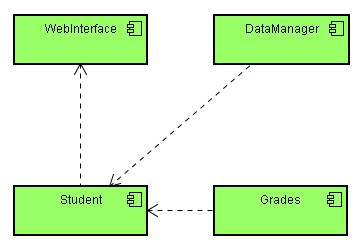
## **7、协作图（Collaboration Diagrams）：**

交互图的一种，描述了收发消息的对象的组织关系，强调对象之间的合作关系。时序图按照时间顺序布图，而写作图按照空间结构布图



## ****8、构件图（Component Diagrams）：****

构件图是用来表示系统中构件与构件之间，类或接口与构件之间的关系图。其中，构建图之间的关系表现为依赖关系，定义的类或接口与类之间的关系表现为依赖关系或实现关系。

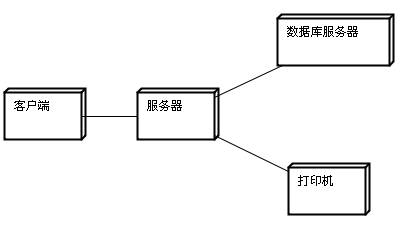


## **9、部署图（Deployment Diagrams）：**

描述了系统运行时进行处理的结点以及在结点上活动的构件的配置。强调了物理设备以及之间的连接关系。

部署模型的目的：

描述一个具体应用的主要部署结构，通过对各种硬件，在硬件中的软件以及各种连接协议的显示，可以很好的描述系统是如何部署的；平衡系统运行时的计算资源分布；可以通过连接描述组织的硬件网络结构或者是嵌入式系统等具有多种硬件和软件相关的系统运行模型。



# 二、图的差异比较

　　1.序列图(时序图)VS协作图

       序列图和协作图都是交互图。二者在语义上等价，可以相互转化。但是侧重点不同：序列图侧重时间顺序，协作图侧重对象间的关系。

共同点：时序图与协作图均显示了对象间的交互。

不同点：时序图强调交互的时间次序。

           协作图强调交互的空间结构。

 　  2.状态图VS活动图

      状态图和活动图都是行为图。状态图侧重从行为的结果来描述，活动图侧重从行为的动作来描述。状态图描述了一个具体对象的可能状态以及他们之间的转换。在实际的项目中，活动图并不是必须的，需要满足以下条件：1、出现并行过程&行为；2、描述算法；3、跨越多个用例的活动图。

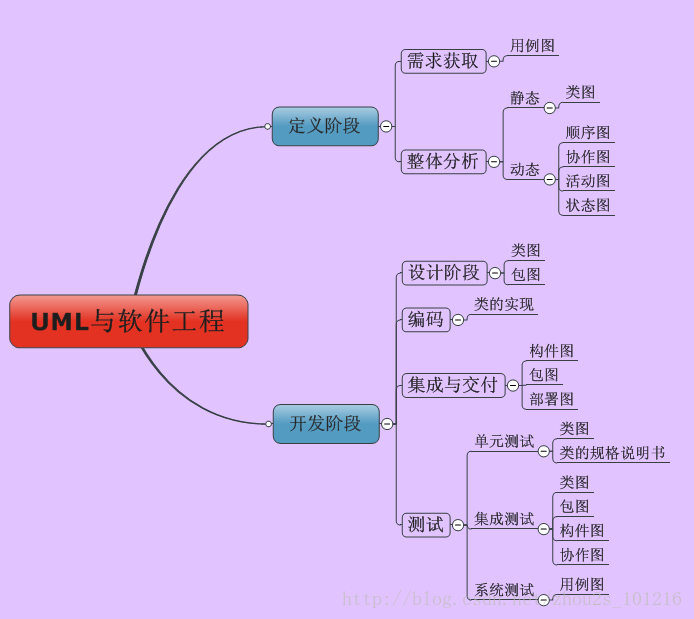
 　  3.活动图VS交互图

　　二者都涉及到对象和他们之间传递的关系。区别在于交互图观察的是传送消息的对象，而活动图观察的是对象之间传递的消息。看似语义相同，但是他们是从不同的角度来观察整个系统的。

# 三、UML与软件工程

　　UML图是软件工程的组成部分，软件工程从宏观的角度保证了软件开发的各个过程的质量。而UML作为一种建模语言，更加有效的实现了软件工程的要求。

　　如下图，在软件的各个开发阶段需要的UML图。



　　下表是UML使用人员图示：

