UML14种图概述

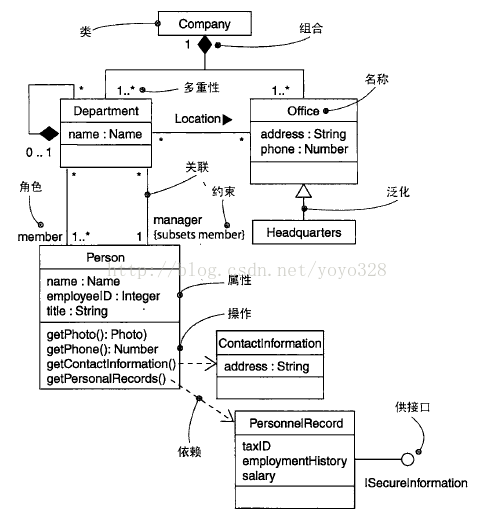
# 1 结构图(Structure Diagram)

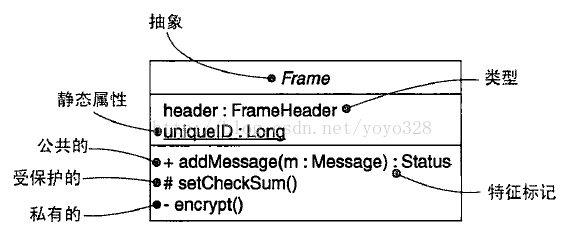
结构图强调的是系统式的建模。

**1.1 静态图(Static Diagram)**

**1.1.1 类图(Class Diagram)**

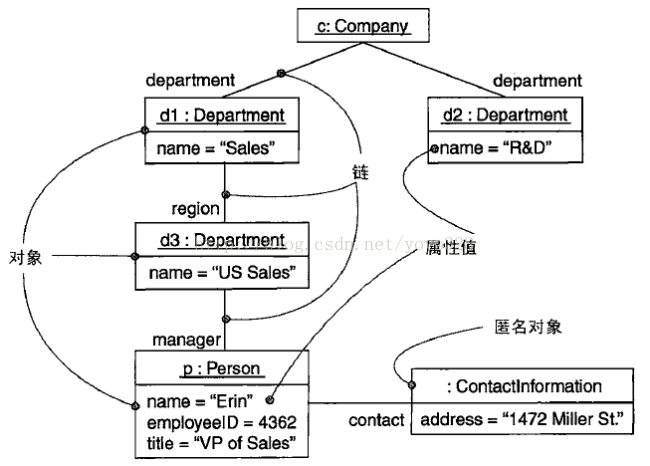
类图是显示一组类、接口、协作以及它们之间关系的图。





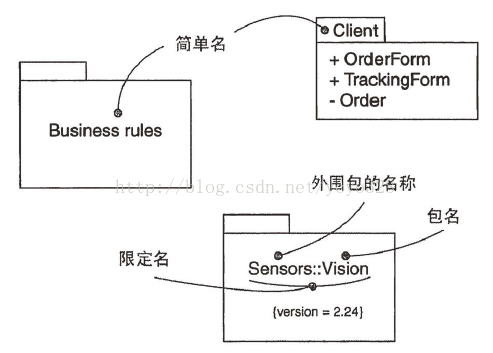
**1.1.2 对象图(Object Diagram)**

对象图是表示在某一时间点上一组对象以及它们之间的关系的图。



**1.1.3 包图(Package Diagram)**

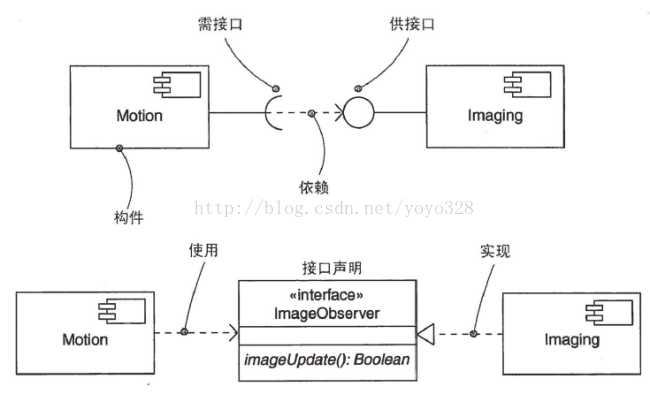
包图是包和包之间的关系构成，是维护和控制系统总体结构的重要建模工具，用于描述系统的分层结构。



**1.2 实现图(Implementation Diagram)**

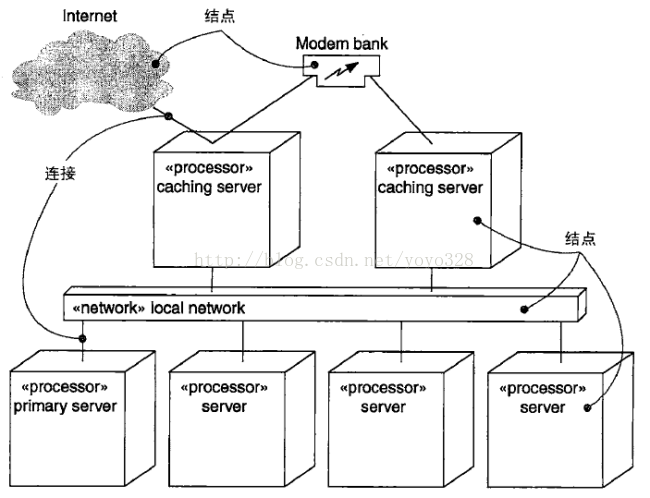
**1.2.1 组件图(Component Diagram)**

组件图描述的是在软件系统中遵从并实现一组接口的物理的、可替换的软件模块。



**1.2.2 部署图(Deployment Diagram)**

部署图是一种展示运行时进行处理的结点和在结点上生存的制品的配置的图。

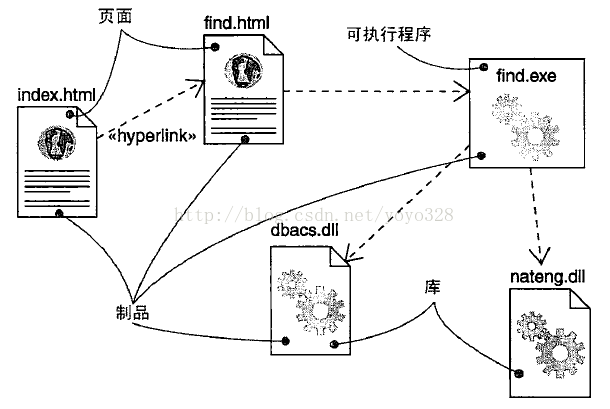


**1.3 复合结构图(Composite Structure Diagram)**

复合结构图显示类元内部结构，包括它与系统其他部分的交互点。也显示各部分的配置与关系，这些部分一起执行类元的行为。

**1.4 制品图(Artifact Diagram)**

制品图展示了一组制品及它们之间的关系。

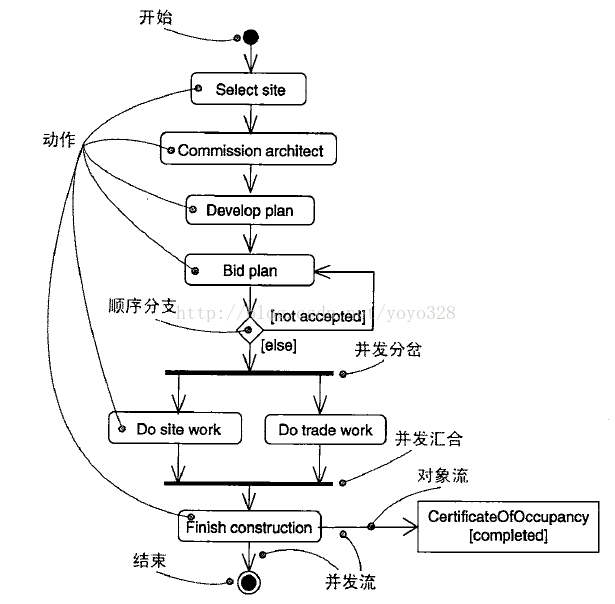


# 2 行为图(Behaviour Diagram)

行为图强调系统模型中触发的事件。

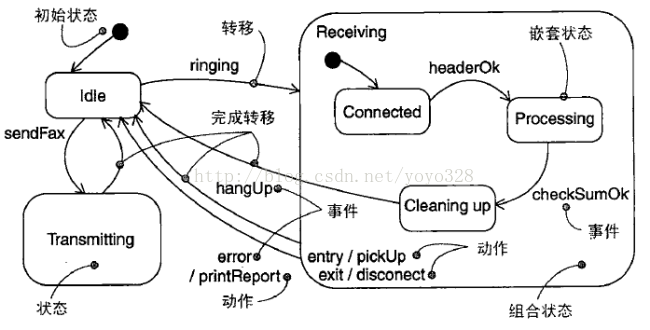
**2.1 活动图(Activity Diagram)**

活动图显示从活动到活动的流。



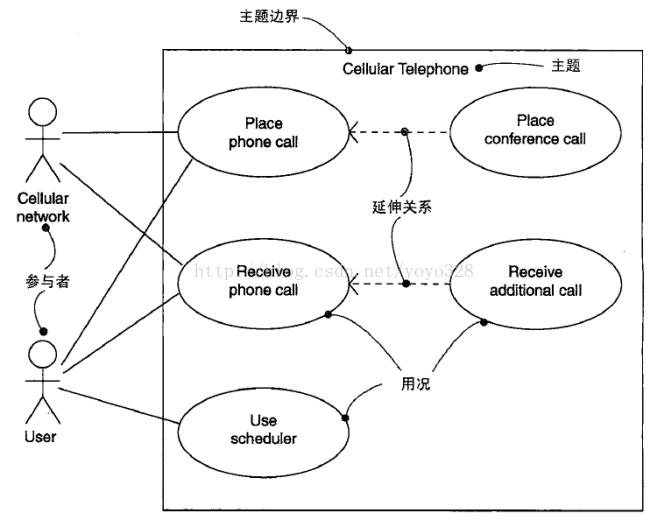
**2.2 状态图(State Machine Diagram)**

状态图显示了一个状态机，它强调从状态到状态的控制流。



**2.3 用例图(User Case Diagram)**

用例图是表现一组用例、参与者以及它们之间关系的图。

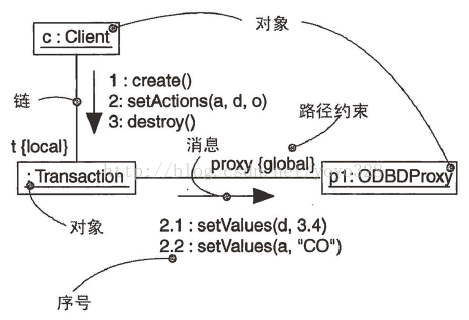


**2.4 交互图(Interaction Diagram)**

交互图属于行为图形的子集合，强调系统模型中的资料流程。

**2.4.1 通信图(Communlcation Diagram)协作图**

通信图是强调发送和接收消息的对象的结构组织的交互图。

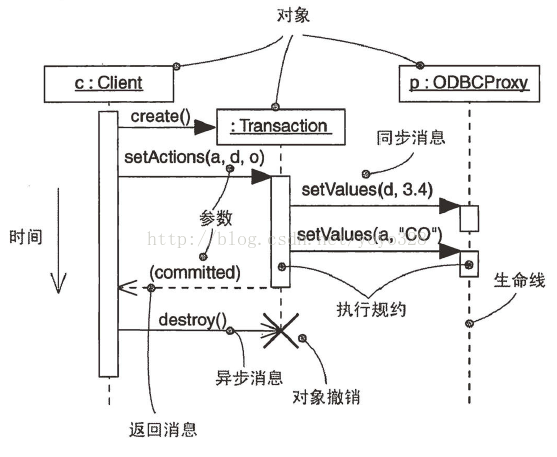


**2.4.2 交互概述图(Interaction Overview Diagram)**

交互概述图是将活动图和顺序图嫁接在一起的图，可以看作活动图的变体，它将活动节点进行细化，用一些小的顺序图来表示活动节点内部的对象控制流，也可以看作顺序图的变体，它用活动图来补充顺序图。

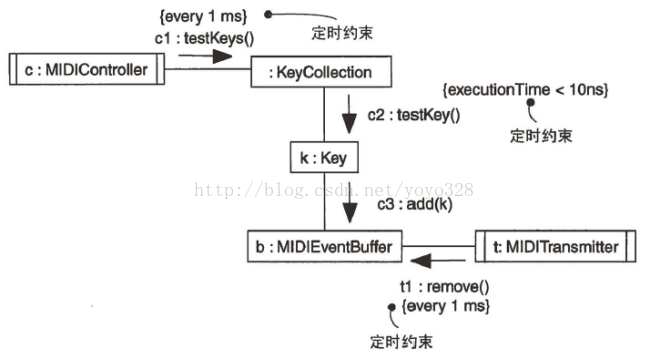
**2.4.3 时序图(Sequence Diagram)**

时序图显示对象之间的关系，强调对象之间消息的时间顺序，显示对象之间的交互。



**2.4.4 时间图(Timing Diagram)**

时间图被用来显示随时间变化，一个或多个元素的值或状态的更改。也显示时控事件之间的交互和管理它们的时间和期限约束。



uml各类图--完整全面实例

## 一、UML是什么？UML有什么用？

**UML是什么？**

Unified Modeling Language（统一建模语言）是对象管理组织（OMG）制定的一个通用的、可视化的建模语言标准，可以用来可视化（visualize）、描述（specify）、构造（construct）和文档化（document）软件密集型系统的各种工件（artifacts，又译制品）

UML是一种标准的图形化建模语言，是面向对象分析与设计的**标准表示**，它：

星星不是一种可视化的程序设计语言，而是一种可视化的**建模语言**(用于分析设计)

星星不是工具或知识库的规格说明，而是一种建模语言规格说明，是一种**表示的标准**

星星不是过程，也不是方法，但允许任何一种过程和方法**使用**它

**UML有什么用？**

软件开发系统规模比较复杂时，需要用图形抽象地表达复杂概念，增强设计的灵活性、可读性和可理解性，以便暴露深层次的设计问题，降低开发风险。有必要采用一套通用的图形语言和符号体系描述组织的业务流程和软件需求，促进业务人员、开发人员之间一致、高效地交流。

## 二、UML的历史

**UML发展背景：**

P. Coad和E.Yourdon提出OOA和OOD

G. Booch提出面向对象开发方法

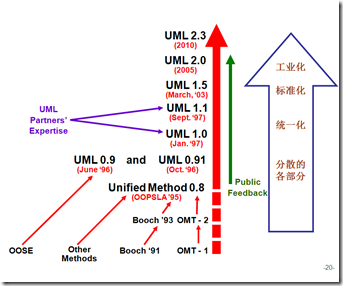
Jacobson提出OOSE

Rumbaugh提出的OMT

……

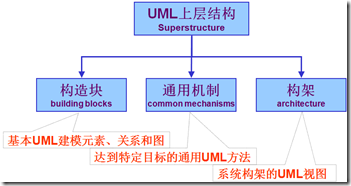
UML的出现结束了这场方法学战争

**UML发展历程：**

[](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111346356863.png)

## 三、UML的上层结构（Superstructure）

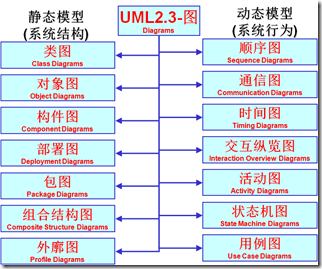
至于UML底层的基础结构（Infrastructure），软件开发工程师们没必要了解，只需要懂得上层结构就行了。

[](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111347453736.png)

## 四、UML建模工具

比较流行的有Rational Rose ，Microsoft Visio、Enterprise Architect 、Visual UML等。我现在使用的UML建模工具是Enterprise Architect 8.0，推荐使用这款，比较好用。

五、UML的图（重点）

[](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111348491377.png)

1、用例图（use case diagram）

|  |  |
| --- | --- |
| 灯泡用例图（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  灯泡核心概念      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  灯泡推荐使用场合      业务建模、需求获取、定义  某图书馆管理系统：  是一个基于Web的计算机应用系统；  读者可以查询图书信息以及借阅信息；  读者可以通过系统预约所需的图书；  图书馆工作人员利用该系统完成读者的借书、还书业务；  图书馆工作人员可以对图书信息、读者信息等进行维护；  对于到期的图书，系统会自动向读者发送催还信息；  管理员会定期进行系统维护；  …… | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111349143429.png)  UC01：“借书”用例文档  用例名称：借书  用例标识：UC01  涉及的参与者：工作人员  涉及的用例：无  描述：工作人员利用该用例为读者完成借书过程  前置条件：工作人员必须登录到当前系统  涉众利益：       读者：能够方便的找到并借出所需的图书       工作人员：能够快速并准确的完成借书工作  基本事件流：工作人员帮助读者借阅图书     1.用例起始于读者带着所要借的图书来到借阅前台；     2.工作人员录入读者信息；     3.工作人员逐一录入所有的图书信息：     \* 3.1 工作人员录入一本图书信息；     \* 3.2 系统确认该读者可以借阅当前图书；     4.工作人员确认本次借阅信息；     5.系统记录本次借阅情况。  后置条件：系统将读者借阅信息正确地记录到数据库中  备选事件流     2a. 读者身份不合法     2b. 读者存在欠费信息，不允许借书     3.2a. 该读者不允许借阅当前图书  字段列表：     5. 借阅信息主要包括：读者图书证号、图书编号、借阅日期（默认为当天日期）、借阅天数以及归还日期。  业务规则     3.2 系统根据当前读者的借阅规则来判断是否可以借阅图书；而借阅规则取决于读者的类型(如本科生、研究生、老师等)和图书的类型(如科技类、文学类、新书等)，并可动态配置  非功能需求：无  设计约束：无  部署约束：无  未解决的问题     2b. 读者存在多少欠费记录时，才不允许借书？     3.2 借阅规则的具体配置情况需和用户进一步讨论？ |

2、活动图（activity diagram）

|  |  |
| --- | --- |
| 灯泡活动图（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  灯泡核心概念  状态、活动、组合活动、对象  转移、分支  并发、同步  泳道  灯泡推荐使用场合  业务建模、需求、类设计 | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111351592292.png) |

3、静态结构图

|  |  |  |
| --- | --- | --- |
| 类图(Class Diagram) | 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： | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352039756.png) |
| 对象图(Object Diagram) | 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  对象图展示我当前借书情况： | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352088856.png) |
| 包图(Package Diagram) | 展现有模型本身分解而成的组织单元(包)以及它们的依赖关系。  核心概念：包(、框架、层、子系统) 、依赖  包图展示系统分层结构： | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352122665.png) |
| 组合结构图(Composite Structure Diagram) | 描述系统中某一部分(组合结构)的内部结构，包括该部分与系统其它部分的交互点。  核心概念：组合结构、部件、端口、协议  组合结构图展示借书内部结构： | [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352171175.png) |

 4、顺序图（Sequence Diagram）

|  |  |
| --- | --- |
| 灯泡顺序图（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  灯泡核心概念  对象、生命线、激活、交互、消息  交互帧(Interaction Frame)  灯泡推荐使用场合  用例分析、用例设计 | “借书”用例实现的顺序图[image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352223099.png) |

5、交互纵览图(Interaction Overview Diagram)

|  |  |
| --- | --- |
| 灯泡交互纵览图(Interaction Overview Diagram)  活动图和顺序图的混合物  直观地表达一组相关顺序图之间的流转逻辑  灯泡核心概念  交互帧  分支、转移  灯泡推荐使用场合  用例分析、用例设计 | 交互纵览图组织多个顺序图[image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352253495.png) |

6、通信图(Communication Diagram)

|  |  |
| --- | --- |
| 灯泡通信图(Communication Diagram)  UML 1.x中称为协作图(Collaboration Diagram)  表示一组对象间关系以及交互活动  灯泡核心概念  对象、协作角色  协作、交互、消息  灯泡推荐使用场合  用例分析、用例设计 | “借书”用例实现的通信图  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352307579.png) |

7、时间图(Timing Diagram)

|  |  |
| --- | --- |
| 灯泡时间图(Timing Diagram)  一种交互图，展现消息跨越不同对象或角色的实际时间信息；  具体描述单个或多个对象状态变化的时间点以及维持特定状态的时间段；  顺序图是表示交互的主要手段，可以在顺序图中增加时间约束来表明对象状态变化的时间点以及维持特定状态的时间段。  灯泡核心概念  时间约束、持续时间约束、生命线  状态、条件、事件 | “打电话”顺序图的时间约束  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/20120611135238356.png) 利用时间图描述时间约束  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352458607.png) |

8、状态机图(State Machine Diagram)

|  |  |
| --- | --- |
| 灯泡状态机图(State Machine Diagram)  UML1.x为状态图(Statechart Diagram)  利用状态和事件描述对象本身的行为  灯泡主要概念  状态、初态、终态、复合状态  事件、转移、动作  并发  灯泡推荐使用场合  类设计 | “图书”类的状态机图  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352502483.png) |

9、构件图(Component Diagram)

|  |  |
| --- | --- |
| 灯泡构件图(Component Diagram)  封装类为构件  描述在系统实现环境中的软件构件和之间的关系  灯泡主要概念  构件、工件、接口(所供接口、所需接口)  依赖、实现  灯泡推荐使用场合  系统设计、实现、部署 | 构件图描述类的实现环境  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352554681.png) |

10、部署图（Deployment Diagram）

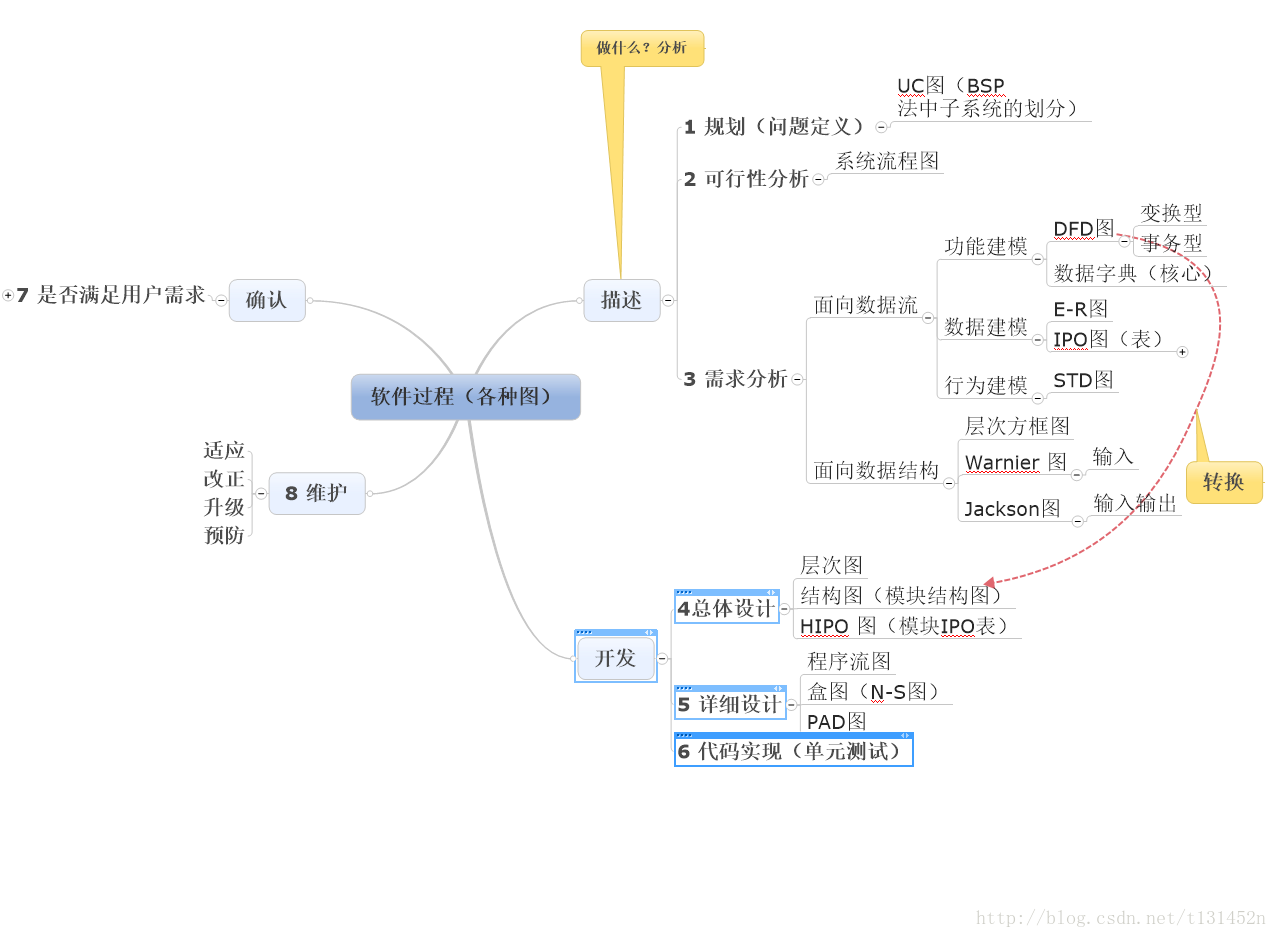
|  |  |
| --- | --- |
| 灯泡部署图（Deployment Diagram）  描述系统所需的硬件构件的物理部署  灯泡主要概念  节点、构件、位置  连接、依赖  灯泡推荐使用场合  系统设计、实施、部署 | 部署图描述系统部署情况  [image](http://images.cnblogs.com/cnblogs_com/way-peng/201206/201206111352594553.png) |

软件工程——软件开发过程中用到的各种图

软件的开发过程，也可以说是软件生命周期。主要包括四个基本活动：描述、开发、确认、演进。四个基本活动分八个阶段进行，各个阶段都用到了各色图形工具。而且在我们写文档的过程中，这些图都是宝贝啊！

（前言：这些图，大多来自网络，经个人整理而成。给自己，也算是给大家以后写文档画图做个参考。）

一、宏观导图

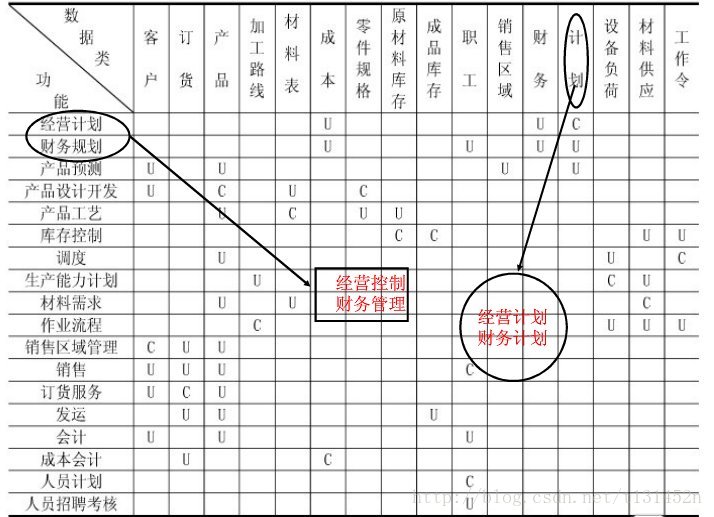


 导图说明：我们的软件开发中用到的各种图型工具都是为了辅助我们更好的理解开发的阶段或者过程。上图是根据软件过程中各个阶段所需要用到的各种图的一个小结。下面是各种图的简介和示例。

二、谈细节：

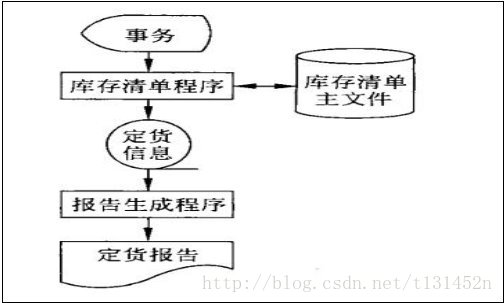
## 1、问题定义（规划阶段）：

  UC图:（Use Creat图）它是BSP（business system planning）法中常用的子系统划分工具。



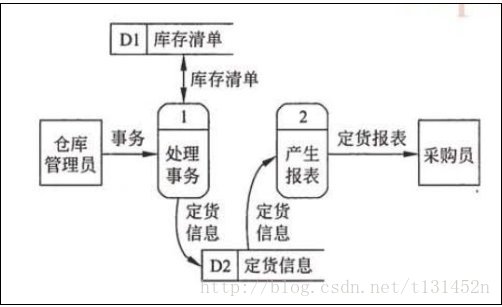
## 2、可行性分析

  2.1系统流程图：是描述系统物理模型的一种传统工具。它是表达数据在系统各部件之间流动的情况，而不是对数据加工处理的控制过程，它是物理数据流图而不是程序流程图。系统流程图形象的呈现了软件的功能，即使不懂软件的人也可以轻松的看懂，可以说它是软件设计师与用户之间沟通、交流的有效工具。

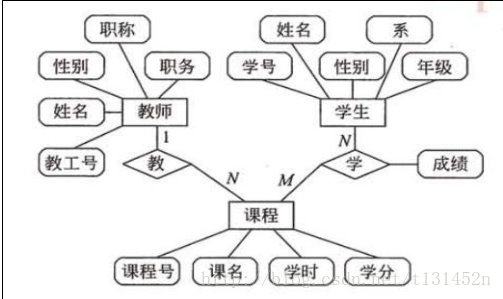


## 3、需求分析：

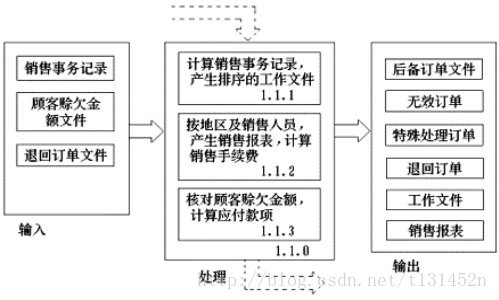
  3.1 DFD图（Data Flow Diagram):从数据传递和加工角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程.建立系统的功能模型。



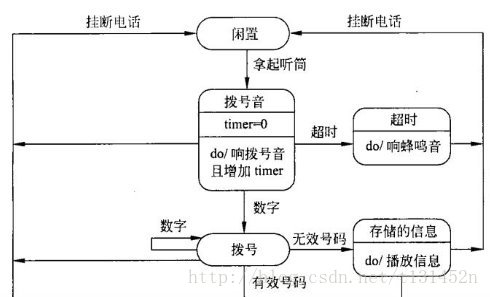
  3.2 ERD（Entity-Relationship Diagram)图：当数据量很大并且数据间关系复杂时对于数据的分析就得用到它来刻画系统数据模型



  3.3 IPO（input process output）图描述了输入数据、处理数据、输出数据之间的关系。

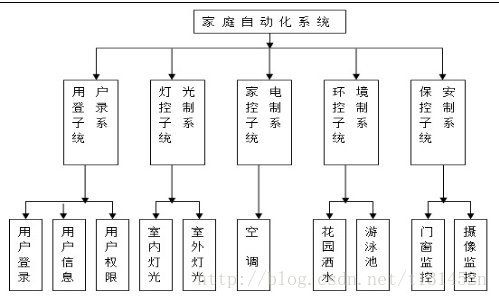


 3.4 STD（State Transition Diagram）图：刻画系统响应外部事件的过程。为系统的行为建模。

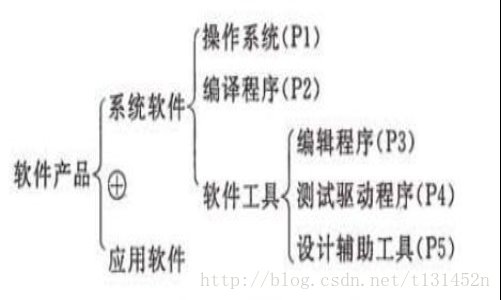


  面向数据结构的几个图形工具：

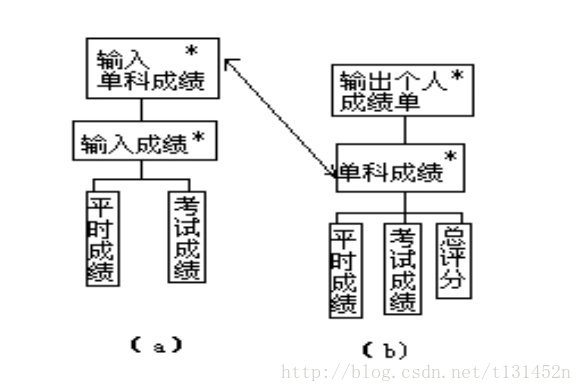
  3.5 层次方框图：用来展示数据的层次结构



  3.6 warnier图：和层次方框图一个意思，不过她能描述的手段比层次图更加丰富。

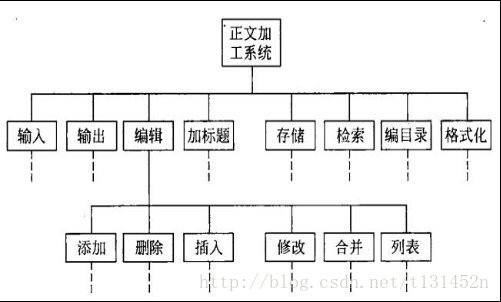


  3.7 Jackson图

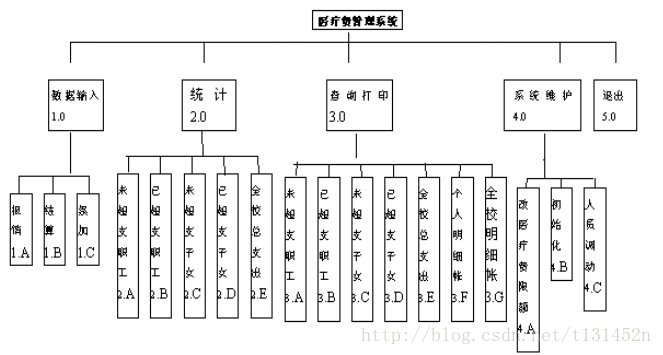


## 4、概要设计（总体设计）：

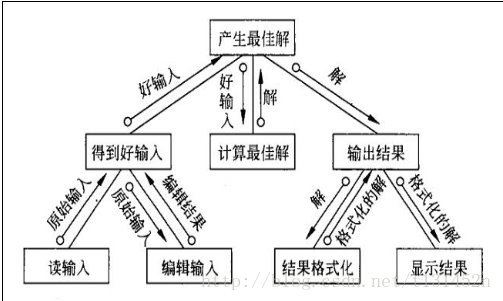
 4.1层次图：描述层次结构



 4.2 HIPO图=层次图+IPO图



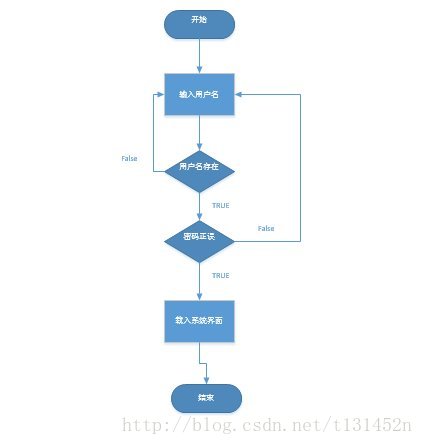
  4.3 （模块）结构图：这是结构化开发中最常用的描述一个系统体系结构的工具图之一。



## 5、详细设计：

按模块进行细分

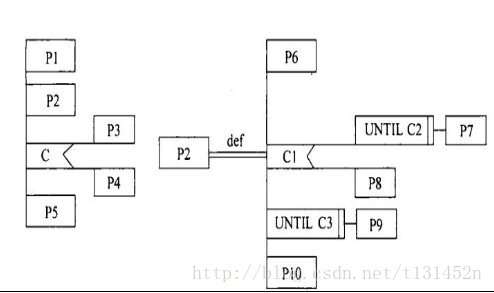
  5.1程序流程图：



  5.2 N-S图（盒图）



  5.3 PAD（Problem Analysis diagram)图



判定表

判定树

## 6、实现

界面，成功数据示例及其对应主要示例代码（无关紧要的代码放在附录），错误调试，问题排除，复杂度评估，代码质量提升

## 7、测试

按测试的方式来进行，测试用例

## 8、维护

三、总结：这篇博客，算是一个整理工作，对于软件工程过程中各种图有了一个宏观上的了解，还有很多不会画，存在不理解的图，大多数是从网上找的图。但是这都不能影响继续前进的脚步。这是学习的第一个阶段，积累。后续的文档中将要一一的尝试一下这些图。将理论付诸实践！