

LANZHOU UNIVERSITY OF TECHNOLOGY

结课作业

题 目 U M L 结 课 作 业

学生姓名 石成伟

学 号 1817700410

专业班级 软件四班

学 院 软件学院

# 目录

[摘要: 3](#_Toc42634579)

[1.简介 4](#_Toc42634580)

[2.UML架构 4](#_Toc42634581)

[3. UML对软件建模的意义 5](#_Toc42634582)

[4.需求分析的原则 6](#_Toc42634583)

[5.UML在需求分析中的应用实例 7](#_Toc42634584)

[三．类图 9](#_Toc42634585)

[1、类图的内容及注意事项： 9](#_Toc42634586)

[⑴ 9](#_Toc42634587)

[⑵ 9](#_Toc42634588)

[⑶ 9](#_Toc42634589)

[2、类图绘制的要点： 9](#_Toc42634590)

[⑴ 9](#_Toc42634591)

[⑵ 9](#_Toc42634592)

[⑶ 9](#_Toc42634593)

[⑷ 9](#_Toc42634594)

[3、实例： 10](#_Toc42634595)

[四、用例图 10](#_Toc42634596)

[1、简介： 10](#_Toc42634597)

[2.用例图绘制的注意事项： 11](#_Toc42634598)

[3.实例： 12](#_Toc42634599)

[五．状态图 12](#_Toc42634600)

[1.简介： 12](#_Toc42634601)

[2.状态图的注意事项： 12](#_Toc42634602)

[六．序列图 13](#_Toc42634603)

[1.简介： 13](#_Toc42634604)

[2.分类： 13](#_Toc42634605)

[3.序列图的注意事项： 13](#_Toc42634606)

[4.实例： 14](#_Toc42634607)

[七．数据流图 14](#_Toc42634608)

[1.简介： 14](#_Toc42634609)

[2.原理： 15](#_Toc42634610)

[3.数据流图的注意事项： 15](#_Toc42634611)

[4、实例： 15](#_Toc42634612)

# 摘要:

统一建模语言 UML( Unified Modeling Language )是面向对象软件开发的重要新技术，本文阐述了UML的特点、UML架构、软件需求分析原则及UML对软件建模的意义,并给出了UML技术在软件需求分析和设计阶段的成功应用。

软件规模的迅速增长已经使软件成为-一个新兴的产业。软件的稳定性、安全性至关重要。软件开发技术相对于软件产业发展的滞后导致了软件危机的出现，这就迫使人们考虑改进已应用了20多年的传统结构化设计方法,面向对象的软件工程( 00SE)正是为了适应软件产业的这种需求而出现的。

面向过程的技术着眼于功能分解,而面向对象的技术则着眼于对象建模。00SE技术要求开发方严格按照需求分析-设计-实施- -测试的步 骤来进行软件的开发。这几个阶段之间允许迭代,但缺-不可。在以前的软件开发过程中,需求分析往往被当作-一个可有可无的步骤而不受重视,甚至许多项目根本就不做需求分析,直接进人设计阶段。开发大规模软件时,这将会极大程度地影响软件的质量，降低软件开发的效率。需求分析是软件开发的基础,没有好的需求分析就不可能编写出优秀的软件,特别是大规模软件的需求非常复杂,而且用户的需求也会不断变化,好的需求分析是软件开发成败的关键。尽管大多数的开发都采用了典型场景的方法,但它极少用有效的形式归档。UML中所采用的用例( UesCase)就是将这些场景获取正式化,形式化的技术。

关键词: UML;统一建模语言;建模;需求分析;软件;面向对象

统一建模语言 (UML)

## 1.简介

统一建模语言是继80年代末和90年代初面向对象建模技术高潮后,应市场对统一建模语言的要求，由世界著名的面向对象技术专家Booch,Jacobson和Rum-baugh发起,在著名的Booch表示法，00SE方法和OMT方法的基础上发展而来的。UML1.1版本已于1997年11月被OMG(物体经营组)批准为标准，并且产生了UML 1.2,1.3,1.4和2.0版本。到目前为止, UML已获得了工业界、科技界和应用界的广泛支持。UML支持面向对象的分析设计和从需求分析开始的软件开发的全过程。

## 2.UML架构

⑴UML通过各种不同视图和这些视图之间的联系来描述软件系统的，需求分析和设计。视图中含若干个模型元素,如类、对象、用例结点(节点)、接口、包、注解(注意)、构件(成分)等。用于表示模型元素间相互连接的关系也是模型元素,如关联(协会)、泛化(概括)、依赖(依靠)、聚集(集合)等,这些关系的含义如下:

①关联连接模型元素 及链接链环实例；

②泛化表示 -般与特殊关系,即“- -般”元素是“特殊”元素的泛化,“特殊”元素是“一般”元素的特化(专业化)；

③依赖表示1 个元素以某种方式依赖于另1个元素；

④聚集表示整体与部分关系, 即“部分”元素是“整体”元素的一部分；

⑵UML制定了-整套完整的面向对象的标记和处理方法，主要体现在所规定的视图中。UML的概念及模型可以分成以下几类:

①静态结构UML的静态结构描述了系统中的结构成员及其相互关系。系统的结构成员,即类元，包括类、用例、构件和节点,为研究系统动态行为奠定了基础。表态视图包括类图、用例图、构件图和部署图。

②动态行为动态行 为描述了系统随时间变化的行为。动态行为又分为两种,一种是孤立对象的工作流程及状态的变化,对应的视图分别为活动图和状态机图,另一种是一系列相关对象之间交互作用时的通信,交互视图包括顺序图和协作图。

③模型组织管理模型组织管理说明了模型的分层组织结构。包是模型的基本组织管理单元,用于存储、访问控制、配置管理及结构包含可重用的模型单元库。模型的组织管理用类图来实现,但又跨越了其它视图并根据系统开发和配置组织这些视图。包之间的依赖关系是对包的组成部分之间的依赖关系的归纳,系统整个架构可以在包之间施加依赖关系。因此,包的内容必须符合包的依赖关系和有关的架构要求。

④扩展机制UML 还包括多种具有扩展能力的组件,这些组件的扩展能力有限,但很有用。这些组件包括约束、构造型和标记值,它们适用于所有的视图元素。

## 3. UML对软件建模的意义

UML对软件建模的意义随着计算机应用的 飞速发展,软件的复杂程序不断提高,源代码的规模越来越大,项目失败的可能性也相应增加。在长期的研究与实践中,人们越来越深刻地认识到,建立简明准确的表示模型是把握复杂系统的关键。模型是对事物的一种抽象,人们常常在正式建造实物之前,首先建立-一个简化的模型,以便更透彻地了解它的本质,抓住问题的要害。使用模型可使人们从全局上把握系统的全貌及其相关部件之间的关系,可以防止人们过早地陷入各个模块的细节。因此,建模是软件建设的功能需求分析的重要部分。对于软件的开发来讲，参与系统建设人员包括领域专家软件设计开发人员、管理人员和操作人员,他们之间的交流成为系统开发的最大难题,即如何将系统的功能需求分析以明确的、清晰的方式表达出来,使得系统在具体实现阶段能够顺利地、无歧义地进行。

UML作为-种建模语言,正是这样-种标准的表示,它通过统- ~语 义和符号表示来定义一些图和它们的意义,与使用的方法无关。所以,人们可以用各种方法使用UML,而不管方法如何变化，其基础都是UML的图,这就是UML的最终用途,即为不同领域的人们提供统- -的交流标准。

## 4.需求分析的原则

无论采用何种需求分析方法,传统的或是面向对象的,都应遵循--定的操作原则。可供参考的指导原则:

(1)在开始建立分析模型前要对问题域有很好的理解;

(2)利用开发原型,使得用户对人机交互界面有直观的印象;

(3)记录每个需求的起源及原因,这样为建立可回溯到客户的可追踪性提供了基础;

(4)使用多个需求视图,建立数据、功能和行为模型,多种视图的结合使用将减少忽略某些需求的可能性;

(5)为需求设定不同的优先级,先实现优先级高的需求,如将明确的需求优先级设为最高，其次为可变需求,最后为潜在需求;

(6)尽可能减少需求描述的模糊性,这-点对于用自然语言描述的需求尤为重要。

除了以上的基本原则,面向对象的需求分析方法在建立需求分析模型时,还要遵守如下4个原则:

①对问题域建模; ②描述模块功能; ③表示模型行为; ④分解模型以显示更多细节。

只要基于以上的原则进行需求分析,就会为软件开发打下坚实的基础,也就有了开发出优秀软件的可能性。

## 5.UML在需求分析中的应用实例

UML作为一各可视化图形建模语言,用于开发软件系统时,必须有可视化的建模工具和环境,即计算机助手软件工程工具,提供UML图形的编辑和美化，保证得到语法正确,语义完整的UML可视化模型,并提供文档管理和图形打印等辅助支持,才能真正提高系统开发的质量和效率。

目前,在基于UML的开发工具和环境方面, Rational Rose是一种应用比较广泛的建模工具。本文结合某物资订购系统开发的实例,利用Rose2002进行系统的UML可视化建模。

系统分析和设计的第1步也是最关键的1步,是用户需求的捕捉,引人用例的概念并用来开发需求,是面向对象技术进人第2代的标志。用例分析是从用户角度来描述业务过程的方法。用例图描述系统外部的执行者(Actor)与系统提供的用例(UseCase)之间的某种联系。通过系统分析,确定大致的设计目标:

⑴执行者系统的外部用户,可以是人或者其它的应用系统；

⑵用例执行者对 系统的应用方式,通过他/它(执行者)与系统的交互过程描述( Scenarios)来详细说明用户的真实需求。

用例图着重于从系统外部执行者的角度来描述系统需要提供哪些功能,并且指明了这些功能的执行者是谁。

通过与用户探讨,以识别出要处理的所有关键类及它们的相互关系。订购处理系统的域类主要有:订单、订单项、库存、客户等。图1是订购系统用例图。

顺序图由它表示对象之间传送消息的时间顺序。当执行1个用例行为时,顺序图中的每条消息对应1个类操作或状态机中引起转换的触发事件。图2是增加新订单的顺，序图。

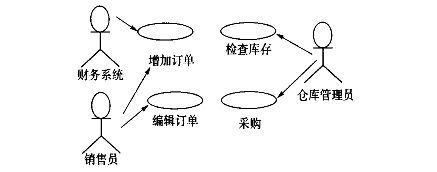


图1



图2

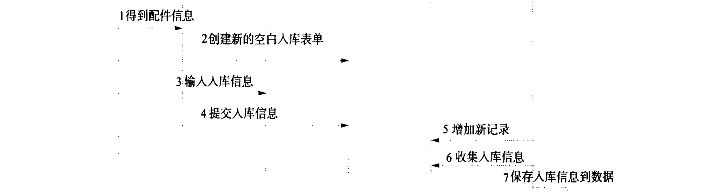


图3

协作图即用于建立在1次交互中有意义的对象和对象间的关系建模。对象和关系只有在交互时才有意义。类元角色描述了1个对象,关联角色描述了协作关系中的1个链。协作图的用途是表示1个类操作的实现。协作图可以说明类操作中用到的参数和局部变量以及操作中类之间的关联。当实现1个行为时,消息编号对应程序中的嵌套调用结构和信号传递过程。它侧重于描述对象之间是如何连接的,通常可以依靠支持工具从顺序图自动转换而来,这里不再给出图示符号。

状态机视图即1个类对象所可能经历的所有历程的模型图,它是对类所描述的对象的补充。状态机由对象的各个状态和连接这些状态的转换组成。每个状态对1个对象在其生命期中满足某种条件的1个时间段建模。当1个事件发生时,它会触发状态间的转换,导致对象从一种状态转化到另一新的状态。当转换相关的活动执行时，转换也同时发生。图3是采购过程的状态图。

# 三．类图

## 1、类图的内容及注意事项：

⑴从上到下，类图分为类名、属性和操作三个部分，类名是所有类图必须要有的，不可或缺；

⑵所选类如果有属性，则每一个属性都要有一个特定的名字；

⑶所选类如果有操作，则每一个操作都要有一个特定的名字。

## 2、类图绘制的要点：

⑴类的操作是针对自身的操作，而不是去操作其他人；

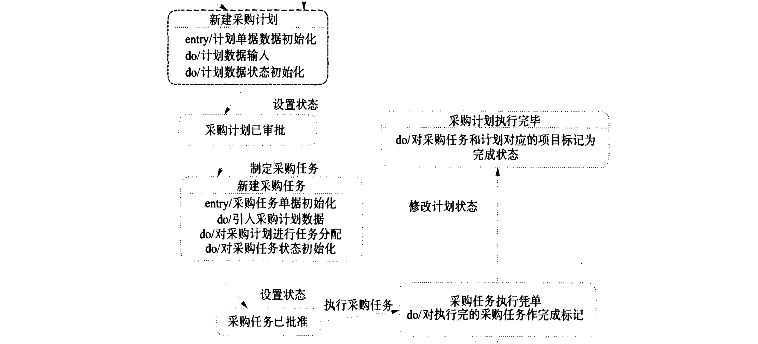
⑵两个类相关联时，需要在关联的类中加上被关联的ID，并且箭头指向被关联类；

⑶由于业务复杂性，-个显示中的实体可能会被分为多个类，这是很正常的,类不是越少越好。类的设计取决于怎样让后台程序的操作更加简单。

⑷类图中，不同的关系需要不同的箭头、可见性符号。

## 3、实例：

下图就是实例——公司财务系统的类图，员工、订单、库存三个类代表了三个抽象的实体。订单类则是员工和货物之间的一个抽象，连接了员工和货物订单，记录了销售这个状态中的一些信息，以及销售操作。采购同理。



# 四、用例图

## 1、简介：

用例图是最常见的一种图。它概括了用力角色和系统之间的关系，描述了系统功能需求，角色和系统的交互以及系统的反应。用例图有参与者、用例以及关系组成；参与者就是用户，用例是系统中的一个功能的概括，关系是参与用例图有参与者、用例、关系组成。参与者就是系统中的用户身份。用例是系统中的一个功能的概括。关系是参与者或者与用例的联系。其中关系可以分为关联、泛化、包含和扩展4种关系。关系的运用是用例图最难的部分。

⑴关联：

关联是参与者和用例之间的关系，用箭头表示;

⑵泛化：

泛化是参与者与用例或用例之间的关系，意思是特殊化。符号用空心的三角形。箭头为被指向。

⑶包含：

包含是用例之间的关系，意思是一个用例包含另一个子用例。子用例是必须存在的，没有子用例则功能不能完成。箭头为去指向。

⑷扩展：

扩展也是用例之间的关系，意思是一个用例可以扩展出一个子用例。与包含不同的是，这个子用例是不一定要存在的，没有也-样能完成功能。箭头为被指向。

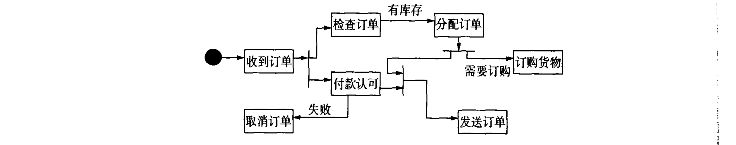
## 2.用例图绘制的注意事项：

⑴用例之间的关系，尤其是包含和泛化的区别，可以这样区分，试着把包含的用例让泛化的用例来包含后，看看是不是同样成立；

⑵箭头指向，泛化和扩展的指向和包含是反的；

⑶用例的表达是动词+名词的形式，只有一个动作或者名词的不是用例。比如所有书籍，是否可借等表述就不是用例。

## 3.实例：



# 五．状态图

## 1.简介：

状态图是UML中相对比较简单的一张图，用得也没前两种图多。状态图是描述一个实体基于事件反应的动态行为，显示了该实体所有可能的状态，以及事件发生时状态的转移条件。状态图的元素包括状态、转移和动作。

总而言之，状态图就是把类的状态的改变连成一张图。可以看做是类图的补充，因为状态本身就是类的状态。

## 2.状态图的注意事项：

⑴状态图不是流程图！连起来的是状态不是动作！

⑵状态图的状态是系统中类的状态，现实中发生但与系统无关的情况都不能被算作是状态。

# 六．序列图

## 1.简介：

序列图是用来描述对象之间消息发送的先后次序，阐明对象之间的交互过程以及系统执行过程某一具体时刻将会发生什么事件。抽象地概括，序列图就是把主体之间传递消息的操作以及消息本身按顺序排列出来。

## 2.分类：

序列图的消息主要分为3种，调用（同步），发送（异步），和返回。

⑴同步调用消息：

是指发送者把控制传递给接收者，然后停止活动，等待消息返回。它是一种即时的关系，返回消息需要直接放在这条消息之后。用实心的三角形表示。

⑵异步发送消息：

是指发送者把消息发送过去后，继续自己的活动，不需要等待消息返回来。返回消息可以在几个过程之后。用半个箭头表示。

⑶返回消息：

就是上述两种消息调用后所返回的消息，用虚线和普通的箭头显示， 当然，一般不需要返回的消息用普通箭头就可以。

## 3.序列图的注意事项：

⑴第一个对象是某操作者，第一步肯定是与系统进行交互：如果画详细点的话可以再加个界面，第一步与界面交互，再界面与系统交互。

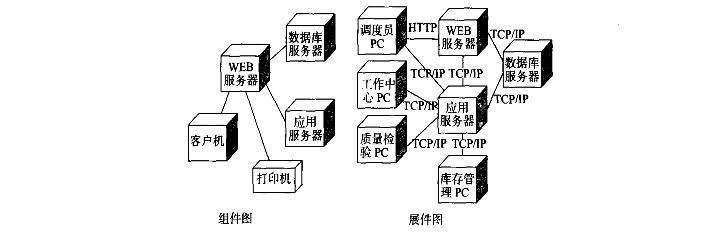
⑵确定哪些情况要同步或者异步的返回信息：返回信息必须是与发送消息的对象一致，方向相反。

⑶主体有对象和参与者两种情况：对象和参与者要区别表示。员工就是参与者，系统是对象。

⑷画图规范问题：在交互过程中要有连续的控制焦点，不能中断也不能没有。

## 4.实例：

借书系统的序列图如下所示。从员工发起库存检查申请到出库成功的信息。对象之间的交互包括借书申请、增加订单、编辑订单操作等。查找订单和采购操作需要直接从系统返回消息。



# 七．数据流图

## 1.简介：

数据流图虽然并不是UML图，然而同样很重要，而且画图的难度比较大。数据流图简称DFD，它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。简单的说，就是数据的流程图。数据流图的元素有外部实体、数据加工、数据存储和数据流，

## 2.原理：

就是把每个数据加工按照顺序用数据流连起来。系统外部输入和输出的数据则用外部实体来表达。每一步加工需要用到的数据存储，或者生成的数据存储，都用数据存储来表示。每个数据加工都需要按序号命名。由于系统数据的复杂性，不可能将所有数据操作画在一张数据流图上。需要进行分层操作，先画整体的数据流图即顶层图，再逐步细化，分为好几张图。

## 3.数据流图的注意事项：

⑴所有步骤都是数据的流动，不要把现实中实体的流动画进去。比如书，把书给员工，就不是系统中数据的流动，因此不能画进去。

⑵外部实体之间不能有数据传输，不能在员工和管理员之间直接画箭头。

⑶顶层系统只能有1个数据加工，然后一个个展开来，不能在顶层图画两个或以上的数据加工。

⑷顶层系统的输入、输出数据，应该和所有底层系统的输入输出数据一致，不能多出来或少。

⑸如果一层有多幅图，每幅图之间的数据要能够衔接起来。

## 4、实例：

由于系统比较简单，因此一共只有两层图。顶层是整个系统和外部的数据交换状态，底层是详细的数据流动。数据加工分为三块，采购申请检查，订单和销售。订单由员工提交，合格的申请发给管理员，分别描绘这三步加工的详细步骤。

