**实验九 UML，逻辑， 软件体系结构设计（一）**

**一、实验目的**

**1. 深入理解UML**

**2. 了解计算机学科中的逻辑**

**3. 学习对比软件体系结构设计GB和IEEE最新SAD (Software Architecture Document)的标准**

**4. 研究经典软件体系结构案例**

**3. 完成自己项目的SRS**

**二、实验内容**

**1. 阅读“The Unified Modeling Language Reference Manual”，进一步学习UML知识，理解如何应用UML对系统进行建模**

UML（Unified Modeling Language，统一建模语言）是一种用于软件工程的标准建模语言。UML 提供了一系列图形表示和符号，用来描述软件系统的结构和行为。UML 是一种通用的、面向对象的设计语言，它不依赖于任何特定的编程语言。

UML 主要包括以下几种类型的图：用例图（Use Case Diagrams）：用来描述系统功能和用户（角色）之间的交互。类图（Class Diagrams）：用来描述系统中的类及其属性、方法和类之间的关系。序列图（Sequence Diagrams）：用来描述对象之间的交互，特别是交互发生的顺序。状态图（State Diagrams）：用来描述一个对象的生命周期或一个操作的执行过程。活动图（Activity Diagrams）：用来描述业务流程或算法的步骤。组件图（Component Diagrams）：用来描述系统的物理组成部分和这些部分之间的关系。部署图（Deployment Diagrams）：用来描述系统的硬件配置和软件组件的分布。

UML 主要的作用是提供了一种标准的方式来可视化一个系统的架构。它帮助工程师们理解复杂系统的设计，并能在设计阶段发现潜在的问题。通过 UML 图，开发人员能在开始编码之前清晰地理解系统的关键方面，从而提高生产效率，减少错误，并促进团队之间的沟通和理解。

UML 包括一组基本的建模元素，这些元素可以组合在一起，形成 UML 图。主要的 UML 元素包括：类（Class）：表示系统中的实体或对象类型。接口（Interface）：描述了类或组件提供的服务或可执行的操作。

属性（Attribute）：描述了类或接口的特性或状态。操作（Operation）：描述了类或接口可以执行的行为或方法。

关系（Relationship）：描述了元素之间的交互或依赖。

UML 的主要图表类型包括：用例图、类图、序列图、状态图、活动图、组件图和部署图。

UML类图绘制首先要明确系统中的类，并找到每一个类对应的属性。例如一个图书管理系统，可能会有Book、User、Library这三个类，其中Book类的属性应该包括 title（书名），author（作者），ISBN（国际标准书号），available（是否可借）等。方法可能包括 borrowBook()（借书），returnBook()（还书）等。其次要明确类与类的关系，常见的关系有一对一、一对多、多对一、多对对等。例如User 和 Book 之间，一个用户（User）可以借阅多本书（Book），同时一本书可以被多个用户借阅。在 UML 中，这是多对多的关系。

UML用例图绘制首先要分析系统需求，其次要确定参与主体即参与者和用例，最后要确定参与者与用例之间的关系，即确定参与者与用例之间的交互方式。通常，这些关系包括参与者执行用例、参与者与用例之间的信息传递等。在图书管理系统中，参与者是User（用户）和Admin（管理员）。用例包括用例：Borrow Book（借书）、Return Book（还书）、Search Book（搜索书籍）等。

UML顺序图描述系统如何执行以及用户如何交互。UML顺序图的主要建模元素包括角色（Actor）、对象（Object）、生命线（Lifeline）、控制焦点（Focus of control）、消息（Message）。绘制UML顺序图首先要确认参与者和对象，然后在顺序图的顶部放置参与者，然后在下方按照交互顺序放置对象，同时为每个对象绘制一条垂直的生命线，表示它们在交互过程中的存在。然后根据系统交互的逻辑，在对象之间添加消息。通过消息在对象生命线上的位置来表示时间顺序。最后，对于复杂的交互或关键部分，可以添加注释和说明来解释其功能和用途。

**2. 浏览“LOGIC IN COMPUTER SCIENCE--Modelling and Reasoning about Systems”，了解常用逻辑及其在计算机学科中的应用**

命题逻辑：这是逻辑系统中最基本的一种逻辑，它关注命题之间的真假关系。在计算机科学中，命题逻辑在电路设计等领域得到了应用。通过命题逻辑，我们可以将复杂的系统行为简化为一系列的命题和它们之间的关系，从而更容易地进行分析和推理。

谓词逻辑：谓词逻辑是一种更为复杂的逻辑系统，它允许我们描述和推理关于对象及其属性的关系。在关系数据库中，谓词逻辑扮演着重要的角色。数据库管理系统使用谓词逻辑来定义查询和操作数据，使得用户能够灵活地检索和处理数据。

**3. 分工协作，参考国标“13 - 软件(结构)设计说明(SDD)”等资料，对比参考SAD最新标准IEEE-42010.pdf，针对自己的项目设计SAD初稿。**

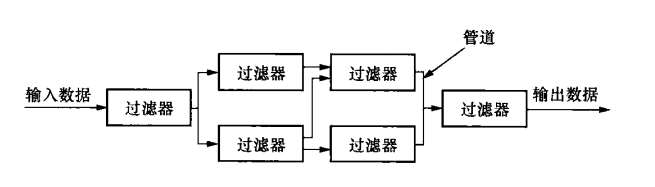
**4. 分工协作，学习、检索研究经典软件体系结构案例。**

**On-the-Criteria-To-Be-Used-in-Decomposing-Systems-into-Modules.pdf**

[**http://www.cs.cmu.edu/~ModProb/index.html**](http://www.cs.cmu.edu/~ModProb/index.html)

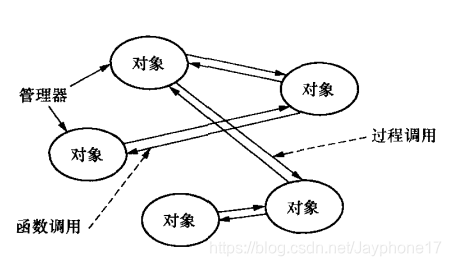
1. 管道/过滤器体系结构

主要包括过滤器和管道两种元素。在这种结构中，构件被称为过滤器，负责对数据进行加工处理。每个过滤器都有一组输入端口和输出端口，从输入端口接收数据，经过内部加工处理之后，传送到输出端口上。数据通过相邻过滤器之间的连接件进行传输，连接件可以看作输入数据流和输出数据流之间的通路，这就是管道。



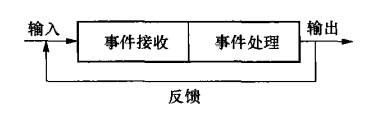
1. 面向对象体系结构

在面向对象体系结构中，软件工程的模块化、信息隐藏、抽象和重用原则得到了充分的体现。在这种体系结构中，数据表示和相关原语操作都被封装在抽象数据类型中。在这种风格中，对象是构件，也成为抽象数据类型的实例。对象与对象之间，通过函数调用和过程调用来进行交互。



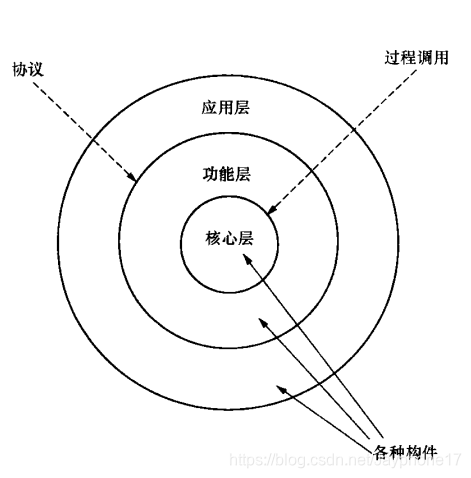
1. 事件驱动体系结构

事件驱动就是在当前系统的基础之上，根据事件声明和发展状况来驱动整个应用程序运行。事件驱动体系结构的基本思想是：系统对外部的行为表现可以通过它对事件的处理来实现。在这种体系结构中，构件不再直接调用过程，而是声明事件。系统其他构件的过程可以在这些事件中进行注册。当触发一个事件的时候，系统会自动调用这个事件中注册的所有过程。因此，触发一个事件会引起其他构件的过程调用。



1. 分层体系结构

在分层风格中，系统将划分为一个层次结构。每一层都具有高度的内聚性，包含抽象程度一致的各种构件，支持信息隐藏。分层有助于将复杂系统划分为独立的模块，从而简化程序的设计和实现。通过分解，可以将系统功能划分为一些具有明确定义的层，较高层是面对特定问题，较低层具有一般性。每层都为上层提供服务，同时又利用了下层的逻辑功能。在分层体系结构中，每一层只对相邻层可见。层次之间的连接件是协议和过程调用。用以实现各层之间的交互。



**5. 完成软件需求规格说明SRS**

**下周五（含）前将软件需求规格说明提交给相应的助教**

**项目跟踪，建立能反映项目及小组每个人工作的进度、里程碑、工作量的跟踪图或表，将其保存到每个小组选定的协作开发平台上，每周更新。**