实验九 UML，逻辑， 软件体系结构设计（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

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

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

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

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。

# 1.UML

1.UML介绍：

UML（Unified Modeling Language，统一建模语言）是一套用于软件系统分析与设计的标准视觉建模语言。它提供了一组图形表示方法，这些方法能够在软件的开发过程中帮助定义、可视化、构造和记录软件系统的各个组件。UML虽然最常用于软件工程领域，但其实也适用于其他如工业流程和数据流程管理的非软件系统。下面详细介绍几种常用的UML图表：

用例图 ：

用例图描述了系统的功能和用户（包括人与其他系统）如何与这些功能交互。

用例图通常用于捕获系统的功能性需求，帮助开发者和客户达成对需求的共识。

通过表示系统、用户和用户的动作之间的关系，用例图有助于揭示系统的边界和交互接口。

类图：

类图描述了系统中的类以及它们之间的关系，包括继承、关联、依赖和聚合。

类图是用来详细描述软件系统的数据视图和静态视图的，非常适合对象导向架构。

序列图 ：

序列图显示对象之间如何交互，以及这些交互是如何随时间变化的。

序列图特别侧重于消息传递的时间序列，能够帮助开发者理解系统运行时的行为。

状态图：

状态图显示了系统或对象的状态，以及引起从一个状态转换到另一个状态的事件。

状态图对于描述对象在其生命周期内的各种状态非常有效。

活动图：

活动图是用来描述业务流程或系统功能流的，展示了活动的流程和决策的分支。

它们强调功能流程中的操作和操作之间的顺序。

组件图 ：

组件图展示了软件系统中组件的组织和依赖关系。

组件图用于了解系统的高级组件如何协同工作，这些组件通常对应软件的物理部分，如库、包、文件等。

部署图：

部署图描述了物理硬件上软件组件的物理安置。

它揭示了系统运行的物理基础设施，如服务器、计算机和网络设备以及软件组件之间的关系。

2.应用UML对系统进行建模的步骤：

理解和阐明需求：首先，需要了解系统需要做什么。这将涉及与利益相关者进行交谈，并可能出现在系统概述或需求文档中。UML可用于创建用例图，帮助理解和阐明系统的功能要求。

系统设计：设计系统的主要部分和它们如何一起工作。UML中的类图和对象图可以帮助在这个阶段创建系统的结构图。

详细设计：在这个阶段，我们将决定系统的具体实现。例如，要确定系统中每个类的属性和方法，以及如何存储数据。在这个阶段，我们会使用类图和序列图来细化系统的设计。

编码：这一阶段将设计转化为实际的代码。

测试：测试阶段确保你的系统作为一个整体，以及其各个部分，都满足其需求并能在各种情况下如预期运行。UML的状态图和活动图在此阶段尤其有用，其中状态图可以帮助你理解系统或其部分在其生命周期中会经历什么状态，而活动图则能帮助你理解测试用例的各个步骤。

部署：将完成的系统部署到其最终运行环境中。在这个阶段，部署图可以帮助理解系统在硬件环境中的布局，以及各个硬件和软件元素之间的交互。

# 2.计算机学科常用逻辑和应用

在计算机学科中，常用的逻辑有命题逻辑（Propositional Logic）、谓词逻辑（Predicate Logic）、时间逻辑（Temporal Logic）和模态逻辑（Modal Logic）。

1. 命题逻辑：命题逻辑是研究命题之间的逻辑关系的形式系统。它用于描述和推理命题之间的真值关系，并在计算机科学中用于程序的推理和验证。例如，程序中的条件语句（如if语句）可以使用命题逻辑来判断条件的真假。另外，命题逻辑也常用于逻辑门电路的设计和分析。

2. 谓词逻辑：谓词逻辑是一种扩展了命题逻辑的逻辑系统，用于描述对象和关系之间的逻辑关系。在计算机科学中，谓词逻辑常用于描述形式化语义和符号推理。例如，在形式化验证中，谓词逻辑可以用于描述系统的状态和转换关系，从而进行推理和验证系统的正确性。

3. 时间逻辑：时间逻辑在计算机科学中用于建模和验证并发系统。它可以描述系统中的时间顺序和时间约束，以及并发操作之间的因果关系。时间逻辑在形式验证、实时系统和并发算法等领域都有应用。例如，时序逻辑（如线性时态逻辑）可以用于描述和验证硬件电路的时序性质。

4. 模态逻辑：模态逻辑是一种用于描述可能性、必然性和知识等概念的逻辑系统。模态逻辑在计算机科学中应用广泛，特别是在人工智能和多智能体系统中。它可以用于描述和推理关于世界的知识、信念和行动的可能性和必然性。例如，模态逻辑可以用于描述一个智能代理的知识和信念，以及根据不同的情景和目标来推演最佳的行动。

在计算机学科中，这些逻辑可以用于形式化系统的建模、推理和验证。它们提供了一种严谨的方法来描述和分析计算机系统的行为和特性，从而确保系统的正确性和可靠性。

# SAD

见文档。

# 4.研究案例：

**1.B/S 体系结构**

B/S结构是三层C/S体系结构的一种实现方式，主要包括浏览器，Web服务器和数据库服务器。B/S结构

主要利用不断成熟的WWW技术，结合浏览器的多脚本语言，采用通用浏览器来实现原来需要复杂的专

用软件才能实现的强大功能，节约了开发成本。

B/S体系结构的核心是Web服务器，可以将应用程序以网页的形式存放在Web服务器上。

当用户运行某个应用程序时，只需要在可以断的浏览器中输入响应的 URL，向 Web 服务器提出 HTTP

请求。

当Web 服务器接收 HTTP 请求之后，会调用相关的应用程序（Servlets），同时向数据库服务器发送数

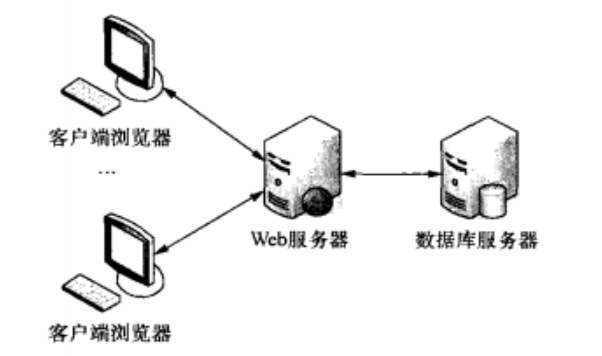
据操作请求。

数据库服务器对数据操作请求进行响应，将结果返回给Web服务器的应用程序。

Web服务器应用程序执行业务处理逻辑，利用 HTML 来封装操作结果，通过浏览器呈现给用户。在B/S

结构中，数据请求、王爷生成、数据库访问和应用程序执行全部由Web 服务器来完成。

原理图：



优点：

客户端只需要安装浏览器，操作简单。

运用HTTP标准协议和统一客户端软件，能够实现跨平台通信。

开发成本比较低，只需要维护Web服务器程序和中心数据库。

缺点：

个性化程度比较低，所有客户端程序的功能都是一样的。

客户端数据处理能力比较差。

在B/S结构的系统中，数据提交一般以页面为单位，动态交互性不强，不利于在线事务处理。

B/S体系结构的可扩展性比较差，系统安全性难以保障。

B/S结构的应用系统查询中心数据库，其速度要远低于C/S体系结构。

1. **公共对象请求代理（CORBA）体系结构**

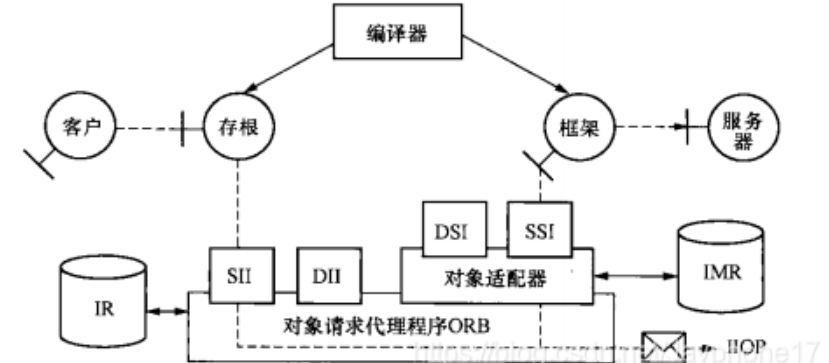
公共对象请求代理（Common Object Request Broker Architecture，CORBA）是由对象管理组织

（Object Management Group，OMG）提出来的，是一套完整的对象技术规范，其核心包括标准语

言、接口和协议。

在异构分布式环境下，可以利用CORBA来实现应用程序之间的交互操作，同时，CORBA也提供了独立于

开发平台的编程语言的对象重用方法。



优点：

1. 实现了客户端程序与服务器程序的分析。

2. 将分布式计算模式与面向对象技术结合起来，提高了软件复用率。

3. 提供了软件总线机制，软件总线是指一组定义的完整的接口规范。

4. CORBA能够支持不同的编程语言和操作系统，在更大的范围内，开发人员能够相互利用已有的开发

成果。

1. **正交 体系结构**

正交体系结构是一种以 垂直线索构件族 为基础的层次化结构，包括组织层和线索。

在每一个组织层中，都包含具有相同抽象级别的构件。

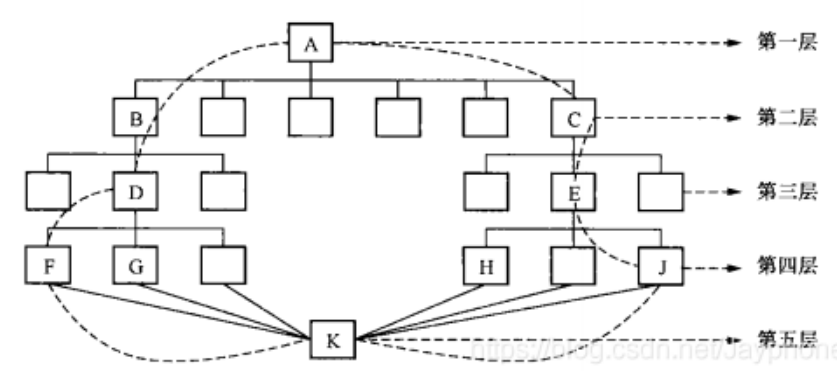
线索是子系统的实例，是由完成不同层次功能的构件通过相互调用而形成的，每一条线索完成系统的一

部分相对独立的功能。

在正交体系结构中，每条线索的实现与其他线索的实现无关或关联很少。在同一层次中，构件之间不存

在相互调用关系。

原理图：



优点：

1. 结构清晰。

2. 便于修改和维护。

3. 易于重用。