软件设计与体系结构

一、第1章 软件工程与软件设计

1.软件工程的概念

1. 软件工程是将系统的、规范的、可度量的的方法应用于软件的开发、运行和维护过程，以及对上述方法的研究
2. 软件工程是用工程、科学和数据的原则与方法，研制、维护计算机软件的有关技术及管理方法
3. 软件工程由方法、工具和过程三个要素组成。方法支撑过程和工具，而过程和工具促进方法学的研究。

2.软件开发过程模型

以软件需求完全确定为前提的瀑布模型

在软件开发的初始阶段只能提供基本需求的渐进式开发模型：原型模型、螺旋模型

以形式化开发方法为基础的变换模型

1. 瀑布模型：根据软件生存周期各阶段的任务从可行性研究开始，逐步进行阶段性变换，直至通过确认测试并得到用户确认的软件产品为止。瀑布模型在上一阶段的变换结果是下一阶段变换的输入，相邻两个阶段具有因果关系，紧密相连。
    - 特点：开始就**需求明确**，任务是阶段性逐步执行的，上个阶段的任务失误会蔓延到以后的各个阶段的任务；
    - 缺点：1.在软件开发的初始阶段指明软件系统的全部需求是困难的，有时甚至是不现实的。难以适应大规模复杂软件系统的开发。

      2.需求确定后，用户和软件项目负责人要等相当长的时间才能得到一份软件的最初版本，
2. 快速原型模型：快速建立可运行的程序，它完成的功能往往是最终产品功能的一个子集
    - 类比建筑项目，根据用户的要求先**快速开发一个初版**，再根据用户意见进行修改
3. 螺旋模型
    - **瀑布模型和原型模型结合**，并**增加了风险分析**，逐步进行软件开发，边开发边评审
    - 组成：需求定义，风险分析，工程实现，评审（不断循环执行）
    - 用户始终参与软件开发的评审，保证软件质量；**支持大型软件开发**
4. 统一软件开发过程
    - 组成：迭代时开发、需求管理、基于构件的软件体系结构、可视化规模、验证软件质量、控制软件变更
    - 4个阶段：**先启阶段，精化阶段，构建阶段，产品化阶段**
    - 工作流程：业务建模，需求，分析与设计，实施，测试，部署，配置与变更管理，项目管理，环境
    - **先启阶段：业务建模和需求； 构建阶段：分析与设计，开发，测试**

3.软件设计

1. 软件设计的重要性(必考)
    - 软件设计是对**软件需求的直接体现**
    - 软件设计为**软件实现提供直接依据**
    - 软件设计将**综合考虑软件系统的各种约束条件并给出相应方案**
    - 软件设计的质量将**决定最终软件系统的质量**
    - **及早发现**软件设计中存在的错误将**极大减少软件修复和维护所需的成本**
2. 软件设计的特征
    - 软件设计的**开端**是出现某些新的问题需要软件来解决，这些需要促使设计工作的开始，并成为整个设计工作最初的基础
    - 软件设计的**结果**是给出一个方案，它能够用来实现所需的、可以解决问题的软件，方案的描述可能是文字、图表，甚至是数学符号、公式等组成的文档或模型
    - 软件设计包含一**系列的转换过程。**
    - 产生**新的想法或思路**对软件设计非常重要
    - 软件设计的过程是**不断解决问题和实施决策的过程**
    - 软件设计是一个**满足各种约束的过程**
    - 大多数软件设计是一个**不断演化的过程**
3. 软件设计的要素
    - 目标描述
    - 设计约束
    - 产品描述
    - 设计原理
    - 开发规划
    - 使用描述

二、第2章 统一建模语言UML

1.UML的特点和用途

- 为使用者提供了**统一的，表达能力强大的可视化建模语言**，以描述应用问题的需求模型、设计模型和实现模型
- 提供对**核心概念的扩展机制**，用户可加入核心概念中没有的概念和符号，可为特定应用领域提出具体的概念、符号表示和约束
- **独立于实现语言和方法学，但支持所有的方法学**，覆盖了面向对象分析和设计的相关概念和方法学
- **独立于任何开发过程，但支持软件开发全过程**
- 提供对**建模语言进行理解的形式化基础**，用元模型描述基本语义，OCL描述良定义规则，自然语言描述动态语义
- 增强**面向对象工具之间的互操作性**，便于不同系统间的集成
- 支持**较高抽象层次开发所需的各种概念**，如协同、框架、模式和构件等，便于系统的重用

2. UML 2.0的建模机制

1. 结构建模
    - 类图：描述系统的静态逻辑结构，包括关联、聚集、继承和依赖关系
    - 包图：特殊类型的类图，描述类和接口如何进行逻辑划分
    - 对象图：类图的实例
    - 构件图：描述了系统实现中的结构和依赖关系
    - 组合结构图：描述较为复杂的系统元素以及元素之间的关系
    - 部署图：描述系统硬件的物理拓扑结构以及在此结构上运行的构件
2. 行为建模
    - 活动图：描述行为或行动的流程
    - 交互图：
        1. 顺序图：描述对象在其生存周期内的交互活动
        2. 通信图：描述特定行为中参与交互的对象及其连接关系
        3. 交互概览图：活动图的简化版本，强调执行活动所涉及的元素
        4. 时序图：强调消息的详细时序说明
    - 状态图：刻画一个元素内部的状态迁移
    - 用例图：通过用例来描述系统的功能性需求

3.面向对象开发方法

1. 基本概念:
    - 对象
    - 类
    - 继承
    - 聚集
    - 多态
    - 消息
2. 优势:
    - 简化软件开发过程
    - 支持软件复用
    - 改善软件结构

4.习题二第五题:

开发一个简单的网络购物平台，可以实现基本的用户登录、浏览商品、购买商品、生成订单、支付等功能。请画出该系统中存在的一些主要的类图。

三、第3章 软件设计基础

1.内聚和耦合

- 内聚:是前述信息隐藏和局部化概念的自然扩展，它标志着一个模块内部各成分彼此结合的紧密程度。

1. 低等级内聚:偶然性内聚、逻辑性内聚和时序内聚
2. 中等级内聚:过程性内聚和通信性内聚
3. 高等级内聚:顺序性内聚和功能性内聚

- 耦合：是对软件结构中模块间关联程度的一种度量

1. 低等级耦合:非直接耦合
2. 中等级耦合:数据耦合、特征耦合和控制耦合
3. 高等级耦合:内筒耦合

尽量使用数据耦合,减少控制耦合，限制外部耦合和公共耦合杜绝内容耦合

2.软件设计的基本概念:

软件设计主要针对需求分析过程中得到的软件需求规格说明，综合考虑各种制约因素，探求切实可行的软件解决方案并最终给出方案的逻辑表示，包括文档、模型等。

3.软件设计工作计划表

表 3-1　软件设计工作计划表

序号	活动	日程	负责人	参与者	资源要求	输入制品	输出制品

4.软件设计过程（问答题）

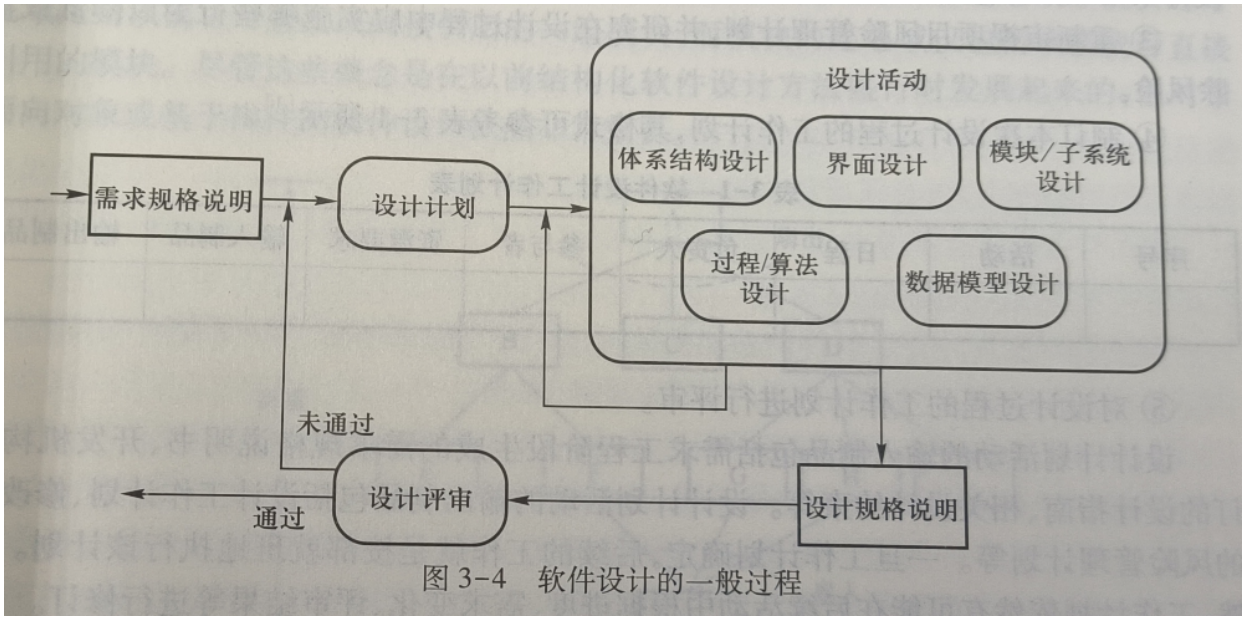

图 3-4　软件设计的一般过程

- 各个活动以需求阶段产生的需求规格说明为基础
- 首先对整个设计过程进行计划

- 然后实施具体的设计活动，这些设计活动本身可能是一个不断迭代和精化的过程
- 在设计活动完成后应形成规格说明
- 然后对设计过程和设计规格说明进行评审
    - 评审未通过再次修订设计计划并对设计进行改进
    - 评审通过则进入后续阶段

软件设计的主要活动

- 软件设计计划
- 体系结构设计
- 界面设计
- 模块/子系统设计
- 过程/算法设计
- 数据模型设计

5.软件设计的质量（问答题）

1. 对软件设计的质量进行综合评价
    - 结构良好
    - 充分性
    - 可行性
    - 简单性
    - 实用性
    - 灵活性
    - 健壮性
    - 可移植性
    - 可复用性
    - 标准化
2. 软件设计对最终软件产品的质量产生的影响有
    - 正确性
    - 可靠性
    - 运行效率
    - 可移植性
    - 可复用性
3. 软件设计对软件开发过程可能产生的影响包括
    - 开发效率
    - 交付时间
    - 风险管理

- 资源管理
- 成本
- 人员培训
- 合法性

四、第4章 面向对象的软件设计方法

1.设计精化（问答题）

1. 设计精化的任务
    - 精化软件架构
    - 调整软件构成类
    - 精化交互模型
    - 精化类之间的关系

作用：对设计模型再进行分析、细化和优化，以生成高质量的设计模型，为后续的实现阶段奠定基础

2.部署模型设计（问答题）

1. 部署模型设计需要考虑以下几点
    - 最终开发完成的软件包括那些制品形式
    - 软件运行环境存在哪些类型的物理节点
    - 不同节点之间的连接和通信形式是什么
    - 软件制品应该如何在物理机节点上进行部署，即他们的部署映射关系

3.案例分析，市场?

五、第5章 面向数据流的软件设计方法

1.事务流

六、第6章 用户界面设计

1.言之有理即可

七、第7章 软件体系结构风格与设计模式

1.设计模式

表 7-1　设计模式分类

		目标		
		创建型	结构型	行为型
范围	类	工厂方法(Factory Method)	适配器(Adapter)	解释器(Interpreter) 模板方法(Template Method)
	对象	抽象工厂(Abstract Factory) 构建者(Builder) 原型(Prototype) 单件(Singleton)	适配器(Adapter) 桥接(Bridge) 组合(Composite) 装饰器(Decorator) 外观(Facade) 享元(Flyweight) 代理(Proxy)	职责链(Chain of Responsibility) 命令(Command) 迭代器(Iterator) 中介者(Mediator) 备忘录(Memento) 观察者(Observer) 状态(State) 策略(Strategy) 访问者(Visitor)

- 创建型

1. Factory Method工厂方法：
    - 使得父类可集中描述公共行为，而将特别行为抽放于子类
    - 核心思想：在父类中，将创建对象的操作包装为一个虚函数，在描述公共行为的过程中调用该函数；在子类中重定义该虚函数来定制创建的对象，从而间接定制公共行为。利用虚函数的多态机制，Factory Method模式使得父类可集中描述公共行为，而将特别行为(不同对象的创建)抽放于子类
2. Abstract Factory抽象工厂：
    - 将公共的创建行为描述为一个抽象类，而将具体的创建方式用该抽象类的子类描述；创建产品时，可以将多个抽象类中的子类组合在一起，组合出不同类型的产品
    - 核心思想：为了提供灵活性，将需要创建的同一风格的一组小实体的一般特征提取出来，用一组抽象产品类来描述，同时将创建行为封装为一个抽象工厂类，提供通用的创建接口，而将各种具体的产品和具体的创建行为用抽象产品类和抽象工厂类的子类来描述，从而使得BigEntity和具体的产品特性和具体的创建行为隔离开来，既降低了耦合度，也使得灵活调整创建行为成为可能。
3. Singleton单件：
    - 一个类最多只有一个实例
    - 核心思想：通过将一个类的构造函数设置为protected或private,可有效阻止从外部直接创建该类的实例，同时设置一个静态成员函数,以负责创建唯一的实例并向外提供访问接口。在Smalltalk等语言中，还需要重定义new操作阻止从外部创建实例，在C++语言中则不需要。

- 结构型

1. Composite组合：
    - 有“递归组合”的特征（文本、图像组合起来A，A又和文本组合起来B，B又和文本、图像组合起来C）
    - 核心思想：为基本对象和组合对象提供一个公共的抽象父类，以表示所有对象，并建立起从该抽象父类到组合对象类的聚集关联，从而间接建立起“递归组合”特性。
2. Proxy代理：

- 构造一个有相同接口的代理对象，将操作请求转发给真实对象，目的是像客户隐藏“转发细节”，提供对真实对象的透明访问；typec转为USB
- 核心思想：构造一个具有相同接口的代理对象，然后将操作请求转发给真实对象，其目的是向客户隐藏“转发过程”的细节（如远程网络交互、智能决策、选择性转发等），提供对真实对象的透明访问。

- 行为型

1. Interator迭代器：
   - 遍历一个聚合对象中的各个元素
   - 核心思想：通过将与遍历有关的部分从聚合对象的描述中分离出来、单独成类，能够将遍历的状态信息用一个独立对象记录，从而可使有效处理多种遍历和并发遍历。另外，本模式可与Factory Method模式配合使用，支持从聚合对象直接创建相应的聚合器。
2. Observer观察者：
   - 一个对象有一对多的依赖关系，对象的数据改变——>所有依赖它的对象都得到通知并自动更新
   - 核心思想：对象是对数据和函数的封装，当一个类包含了太多的函数（或称操作）时，倾向于将其拆分为多个相互协作的类。每个协作类描述一部分行为，包含原来的一部分数据和函数，但这种拆分有一个副作用，因为各协作对象很可能会共享部分数据，所以需要维护相关对象在数据上的一致性。通过使用Obeserver模式，能够为相关对象制定一个交互协议，专门用作数据的一致性维护。

八、第8章 基于分布构件的体系结构（概念题）

九、第9章 软件体系结构评估（概念题）

1.软件体系结构评估概述

1. **目的**：在开发过程早期，通过分析系统的质量需求是否在软件体系结构中得到体现来识别软件体系结构设计中的潜在风险，预测系统质量属性，并辅助软件体系结构决策的制定
2. **包括**
   - **评估时机和参与人员**：早评估和晚评估；评估团队和利益相关人员
   - **评估结果和质量属性:**
     - 性能
     - 可靠性
     - 可用性
     - 安全性
     - 可变性
     - 可移植性
     - 功能性
     - 变化性
     - 可分解性
     - 概念完整性

- **评估的益处和代价：**
  - 把利益相关人员召集在一起
  - 强制特定质量目标的结合
  - 生成冲突目标的优先级
  - 对软件体系结构有一个清晰的说明
  - 提高软件体系结构文档的质量
  - 发现跨项目重用的机会
  - 都得到优化后的软件体系结构

2.软件体系结构评估方法

1. ATAM方法
   - 介绍
     - 介绍ATAM方法
     - 商业动机介绍
     - 软件体系结构介绍
   - 调查和分析
     - 确定软件体系结构方案
     - 产生质量属性效果树
     - 分析软件体系结构方案
   - 测试
     - 集体讨论并确定场景的优先级
     - 进一步分析软件体系结构方案
   - 报告
     - 展示结果
2. SAAM方法
   - SAAM方法的输入
   - SAAM方法的输出
     - 场景的形成
     - 描述软件体系结构
     - 场景的分类和优先级划分
     - 间接场景的单独评估
     - 评估场景交互
     - 形成总体评估
3. ARID方法
   - 排练
     - 确定评审人

        - 准备设计情况介绍
        - 准备种子场景
        - 准备材料
    - 评审
        - 介绍ARID方法
        - 介绍设计
        - 场景的集体讨论和优先级划分
        - 应用场景
        - 总结

问题：包括具体用了什么样的方法？？？不止用一个

十、去年习题

1.软件设计主要活动有哪些？如何做软件设计评审

- 软件设计主要活动
    1. 软件设计计划
    2. 体系结构设计
    3. 界面设计
    4. 模块/子系统系统设计
    5. 过程/算法设计
    6. 数据模型设计
- 如何做软件设计的评审
    - 目标：确保设计规格说明能够实现所有的软件需求，及早发现设计中的缺陷和错误，并确保设计模型已经精化到合格的软件工程师能够构造出符合软件设计者期望的目标软件系统
    - 重点关注的内容：
        - 设计模型是否能够充分地、无遗漏地支持所有软件需求的实现。
        - 设计模型是否已经精化至合理的程度,可以确保合格的软件实现工程师能够构造出符合软件设计者期望的目标软件系统。
        - 设计模型的质量属性,即设计模型是否已经经过充分的优化,以确保依照设计模型构造出来的目标软件产品能够表现出良好的软件质量属性。
    - 建议性原则：
        1. 对产品进行评审,而不是开发人员。
        2. 要有针对性,不要漫无目的。
        3. 进行有限的争辩。
        4. 阐明问题所在,但不要试图去解决问题。
        5. 要求事先准备,如果评审人没有准备好,则取消会议并重新安排时间。
        6. 为被评审的产品开发一个检查表。

7. 确定软件元素是否遵循其规格说明或标准,记录任何不一致的地方。
8. 列出发现的问题、给出的建议和解决该问题的负责人。
9. 坚持记录并进行文档化。

2.面向对象的方法和优势

1. 基本概念:
    - 对象
    - 类
    - 继承
    - 聚集
    - 多态
    - 消息
2. 优势:
    - 简化软件开发过程
    - 支持软件复用
    - 改善软件结构

3.描述软件设计体系结构步骤

1. 开发软件顶层结构
2. 搜索并选取可用设计资产
3. 设计技术支撑方案
4. 确认设计元素
5. 开发软件部署模型
6. 设计并发机制
7. 构建软件体系结构模型
8. 评审软件体系结构模型

4.软件设计规格说明书包括哪些内容

1. 系统概述
   1.1文档概览
   1.2术语定义
   1.3软件系统简述
   1.4软件设计目标
   1.5设计和实现约束
   1.6参考文献
2. 设计指南
   2.1体系结构设计指南

2.2界面设计指南
2.3模块/子系统设计指南
2.4过程/算法设计指南
2.5数据模型设计指南

3. 体系结构设计
4. 界面设计
5. 模块/子系统设计
6. 过程/算法设计
7. 数据模型设计
8. 需求-设计的可追踪性
9. 实施指南

5.工厂方法的核心思想

- 在父类中，将创建对象的操作包装为一个虚函数，在描述公共行为的过程中调用该函数；在子类中重定义该虚函数来定制创建的对象，从而间接定制公共行为。利用虚函数的多态机制，Factory Method模式使得父类可集中描述公共行为，而将特别行为(不同对象的创建)抽放于子类

6.面向对象的软件设计方法、面向数据流各自有什么特点

- 面向对象的软件设计

  面向对象开发方法的核心是利用面向对象的概念和方法对软件进行需求分析和设计，建立面向对象的软件分析和设计模型

- 面向数据流

  面向数据流的软件设计方法，即通常所说的结构化设计方法，其目标是为结构化软件的设计提供一个系统化的途径，使开发人员对软件有一个整体的认识

7.基于分布式构件的体系结构有什么特点？举个例子

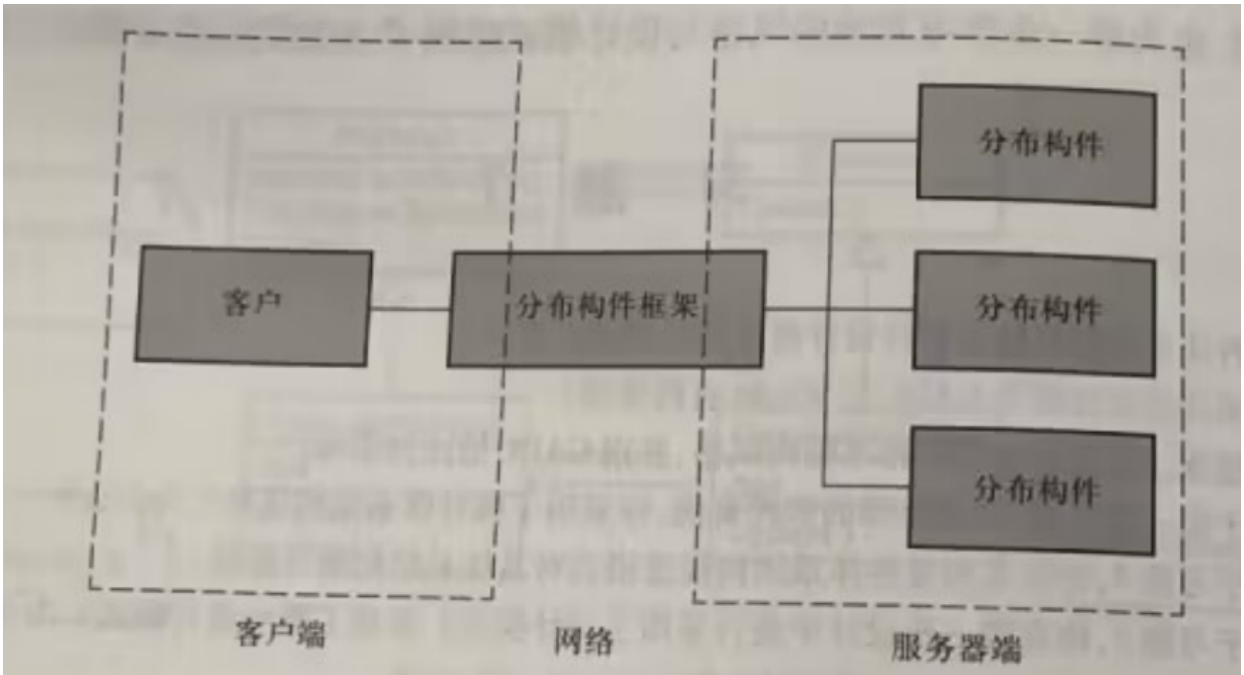

图 8-1　基于分布构件的系统体系结构

- 特点：在这个体系结构中,最关键的部分是分布构件框架,它封装了网络通信的细节,具有两部分功能:其一,向客户提供访问服务器上的分布构件的接口;其二,向服务器上的分布构件提供一个运行的环境(也称容器)。可见,在此方案下,客户和分布构件都不需要关心网络通信问题,只需要使用分布构件框架提供的访问接口即可。
- 例子:

8.用户界面设计考虑因素

- 用户熟悉程度
- 一致性
- 使惊讶最小化
- 可恢复性
- 用户帮助
- 用户多样性

9.设计模式

- 单例模式的核心思想：通过将一个类的构造函数设置为protected或private,可有效阻止从外部直接创建该类的实例，同时设置一个静态成员函数,以负责创建唯一的实例并向外提供访问接口。在Smalltalk等语言中，还需要重定义new操作阻止从外部创建实例，在C++语言中则不需要。
- 抽象工厂的核心思想：为了提供灵活性，将需要创建的同一风格的一组小实体的一般特征提取出来，用一组抽象产品类来描述，同时将创建行为封装为一个抽象工厂类，提供通用的创建接口，而将各种具体的产品和具体的创建行为用抽象产品类和抽象工厂类的子类来描述，从而使得BigEntity和具体的产品特性和具体的创建行为隔离开来，既降低了耦合度，也使得灵活调整创建行为成为可能。

10.软件重用在软件设计中的作用

- 软件重用，是指在两次或多次不同的软件开发过程中重复使用相同或相似软件元素的过程。
- 作用：
    - 减少风险：对关键软件完全进行重新开发且有很高的风险。新开发的软件系统未在实践中充分应用，可能在开发过程中存在的某些问题还未暴露出来。而新系统开发如果延迟完成,将会造成极大的损失。
    - 减少成本：根据以往的实践经验和统计,再工程的成本要明显比开发一个全新的软件低。

十一、总结

1.技巧：

对软件设计的理解，言之有理即可

2.题型

1. 5个简答题 50分
2. 1个设计题 30分
3. 1个开放性题目 20分