软件工程期末复习

第一章、软件工程概述

软件工程方法学包含三个元素：**方法，工具和过程。**方法是软件完成软件开发的各项任务的技术方法，工具是为运用方法而提供的自动的或半自动的软件工程支撑环境；过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

目前最广泛使用的软件工程方法学是，**传统方法学和面向对象方法学。**

传统方法学：
优点：降低了软件产品的复杂性；提高了软件的可理解性；简化了软件的开发和维护工作； 促进了软件重用。
缺点：不能适应事务变化的要求；开发周期长；当软件规模庞大时，使用传统方法学开发往往不成功，此外使用传统方法学开发出的软件难以维护。

面向对象方法学：
优点：
1、易维护
采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是在局部模块，所以维护起来是非常方便和较低成本的。
2、质量高
在设计时，可重用现有的，在以前的项目的领域中已被测试过的类使系统满足业务需求并具有较高的质量。
3、效率高
在软件开发时，根据设计的需要对现实世界的事物进行抽象，产生类。使用这样的方法解决问题，接近于日常生活和自然的思考方式，势必提高软件开发的效率和质量。
4、易扩展
由于继承、封装、多态的特性，自然设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

软件生命周期：

> **软件定义时期：问题定义，可行性研究和需求分析。**
> 问题定义：要解决的问题是什么
> 可行性研究：对于上一个阶段所确定的问题有行得通的解决办法吗？
> 需求分析：确定目标系统必须具备哪些功能

开发时期：总体设计，详细设计，编码和单元测试，综合测试

> 总体设计：又称为概要设计，从各种设计方案中选出最佳方案，制定出最佳方案的详细设计计划，设计软件的体系结构，确定程序模块间的关系。
> 详细设计：总体设计以比较抽象的方式提出了解决问题的方法，详细设计就是把方法具体化。
> 编码和单元测试：写出正确的，容易理解，容易维护的程序模块，并测试每一个模块
> 综合测试：通过各种类型的测试（以及相应的调试）使软件达到预定的要求

维护时期：维护

> 维护通常有四类：
> 改正性维护，诊断和改正软件使用过程中发现的错误。
> 适应性维护，修改软件以适应环境的变化。
> 完善性维护，根据用户需求扩充软件，使软件更完善。
> 预防性维护，修改软件为将来的维护活动预先准备

软件过程：

瀑布模型（文档驱动）

1.阶段间具有顺序性和依赖性
2.推迟实现的观点（不过早编程）
3.质量保证的观点
瀑布模型早期不可逆（考试中一般认定不可逆）

快速原型模型（用户需求驱动）

是用户和设计变换最频繁的方法
快速建立一个能反应用户需求的原型系统，让用户试用，通过实践了解系统概貌。（第四代技术4GL）

螺旋模型（风险驱动）

主要用于内部开发的大规模软件项目，每个阶段都要进行风险评估

敏捷过程与极限编程

敏捷过程（适用于中小型项目）

1.个体和交互胜过过程和工具
2.可以工作的软件胜过面面俱到的文本
3.客户合作胜过合同谈判
4.响应变化胜过遵循计划

极限编程

客户作为开发团队的成员
短期交付

结对编程：一人编码一人审查和测试，两人角色可交换
简单设计，及时调整计划。

第二章、可行性研究

用最小的代价在尽可能短的时间内确定问题是否可以解决
技术可行性，经济可行性，操作可行性
重点还有数据流图，自己看书

第三章、需求分析

软件定义时期的最后一个阶段，基本目的：回答“系统必须怎么做”

第五章、总体设计

总体设计基本目的：回答“概述的说系统应该怎么实现”，总体设计另一任务：设计软件的结构，确定程序由哪些模块组成，以及各模块的关系。

设计过程：
总体设计通常由两个阶段组成：系统设计阶段，确定系统的实现方案；结构设计阶段，确定软件结构。

典型的总体设计过程：
1.设想提供方案
2.选取合理方案
3.推荐最佳方案
4.功能分解
5.设计软件结构
6.设计数据库
7.制定测试计划
8.书写文档
9.审查和复查

信息隐藏和局部化：
模块独立：高内聚低耦合
耦合（以下耦合出现顺序耦合度依次递增）

> 耦合衡量不同模块相互依赖的紧密程度，内聚衡量模块内各元素结合的紧密程度
> 两个模块间通过参数交换信息，交换的信息仅仅是数据，数据耦合（低耦合）。若交换信息中由控

> 制信息，控制耦合（中等耦合）。
> 把整个数据结构作参数传递，被调用模块只需要使用一部分元素就出现了特征耦合。
> 当两个或多个模块通过一个公共数据环境相互作用的时候，公共耦合
> 内容耦合（最不希望）：一个模块访问另一个模块内部数据；一个模块不通过正常入口而转到另一个模块内部；两个模块有一部分程序代码重叠；一个模块有多个入口

内聚（一下内聚出现顺序内聚程度依次递增）

> 低内聚：模块间完成一组任务，这些任务彼此有关系但关系松散，偶然内聚。
> 一个模块完成的任务在逻辑上属于相同或相似的一类，逻辑内聚。
> 一个模块包含的任务必须在同一时间段执行（如模块初始化），时间内聚
> 中内聚：模块中处理元素相关并依特定次序执行，过程内聚
> 模块中所有元素都用同一个输入数据并产生同一个输出数据，通信内聚
> 高内聚：模块中处理元素和统一功能密切相关，而且这些处理必须依次执行，顺序内聚。
> 模块中所有元素属于一个整体并且处理同一个功能，功能内聚

第六章、详细设计

McCabe方法：
根据程序控制流的复杂程度度量程序的复杂程度，环形复杂度
重点：计算画图（P103）

第七章、实现

白盒测试

> 覆盖率低-->高
> 语句覆盖
> 选择足够多的测试数据，使程序中每个语句至少执行一次
> 判定覆盖
> 语句覆盖的基础上每个判定的每个可能都执行一次
> 条件覆盖
> 不仅每个语句执行一次，并且使判定表达式中每个条件都取到各种可能的结果

黑盒测试（根据需求说明）

> 等价划分法
> 每类中的典型值在测试中的作用与这一类中所有其他值作用相同，可以从每个等价类中取一组数据作为测试数据

> 有效等价类+无效等价类=等价类
> 边界值法
> 若合法数据是0到100
> 4值法：-1，0，100，101
> 6值法：-1，0，1，99，100，101

调试
调试（也叫纠错）作为成功测试的后果出现（在测试发生错误时出现）

第八章、维护

软件维护的特点

> 回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他 代码 产生错误。

非结构化维护

> 若软件配置只有程序代码，程序内部文档不足，使评价更加困难，对软件结构，全程数据结构，系统接口，性能和设计约束等经常会产生误解。因为没有测试文档所以不能进行回归测试

结构化维护

> 有完整的软件配置，维护从文档开始，确定软件结构，性能，接口。然后修改设计并且对所做的修改进行复查，编写响应代码。可以回归测试，把修改后的文件再次交付。

第九章、面向对象方法学

面向对象开发方法将面向对象的思想应用于软件开发过程中，指导开发活动，是建立在对象"概念基础上的方法学

面向对象方法学优点：

> 1.与人类思维方式一致
> 2.稳定性好
> 3.可重用性高
> 4.可维护性高

面向对象建模
对象模型（类图），动态模型（时序图），功能模型（用例图）

对象模型
类与类之间：关联，泛化（继承），依赖和细化

第十章、面向对象分析

面向对象分析抽取和整理用户需求并建立问题域精确模型。
大型系统对象模型五个层次：主题层、类与对象层、结构层、属性层、服务层

建立动态模型
1）编写交互脚本，虽然脚本不包括所有偶然事件，但是至少保证不遗漏常见交互行为
2）从脚本中提取事件，确定事件发生的动作对象以及接受时间的目标对象
3）排列事件发生次序，确定每个对象的状态及状态间转换关系，用状态图描绘

第十三章、软件项目管理
软件项目管理使为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对人员（People）、产品（Product）、过程（Process）和项目 (Project)进行分析和管理的活动。
进度计划：Gantt图
人员组织

> 民主制程序员组：小组成员完全平等，享有充分民主，通过协商做出技术决策。
> 主程序员组：用能力强的程序员作为著程序员
> 采用主程序员组的考虑：
> 1）软件开发人员多数缺乏经验
> 2）程序设计过程中有许多事务性工作，如大量信息的存储和更新
> 3）多渠道通信很费时间，将降低程序员生产力
> 主程序员组特性：
> 1）专业化：该组每名成员仅完成各自受过训练的工作部分
> 2）层次化：主刀大夫指挥每名组员工作，并对项目全权负责

现代程序员组：见教材