软件工程

复习

@wjl

2023 年 2 月 12 日

目录

1 软件工程概述

1. 软件的概念：软件是产品，由满足一定要求的，具有可用性、可靠性的程序系统和与之相匹配的文档资料所组成

2. 软件的特点：复杂性、一致性、可变性、不可见性

3. 软件危机：软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题

4. 软件工程定义：将系统的、规范的、可度量的方法应用于软件的开发、运行和维护的过程

2 软件开发过程

2.1 软件开发过程概念

软件开发中的一系列有序的活动以及它们之间的相互关系

2.2 软件过程模型

1. 瀑布模型

- 特点：阶段间具有顺序性和依赖性，推迟实现

- 适用的项目类型：需求已准确定义和相对稳定的项目。系统软件、生产控制软件、图形处理软件、科学与工程计算

2. 快速原型模型

- 特点：快速、反复循环，直到客户确认为止

- 适用的项目类型：需求模糊的系统，已有产品或产品原型，只需客户化的工程项目

3. 螺旋模型

- 特点：每经历一个螺旋就会有可交付的成果，风险驱动

- 适用的项目类型：内部开发的大规模项目

4. XP-极限编程

- 特点：高度动态的过程，非常短的迭代周期

- 适用的项目类型：需求不确定/变化快，项目历时不超过半年、开发人数少、在同一地点工作的中小型团队

5. RUP（统一流程）

- 4 个阶段：初始阶段、细化阶段、构造阶段、交付阶段

- 6 个核心过程工作流：业务建模、需求、分析与设计、实现、测试、部署

- 3 个核心支持工作流：配置与变更管理、项目管理、环境

- 6 个最佳实践：迭代式开发、管理需求、使用基于构件的体系结构、可视化建模、验证软件质量、控制软件变更

2.3 敏捷软件开发

1. 敏捷软件开发宣言：以人为核心、迭代、增量式的开发方法

2. 敏捷优秀实践：技术实践、管理实践、敏捷团队

3. 极限编程的最佳实践：迭代计划会议、每日站立会议、可视化管理、迭代验收

3 需求分析

3.1 需求的两个层次

1. 用户需求：一般性需求，从用户角度描述需求，描述系统外部行为
2. 系统需求：详细需求，从开发者角度描述需求

3.2 需求的两种类型

1. 功能需求：指定软件必须做什么：系统应提供的服务、系统应如何对特定输入做出反应，以及系统在特定情况下的行为

2. 非功能需求：描述系统必须具备的特性和约束，包括性能、可靠性、安全性、可用性等，与其他系统的接口、软硬件环境等。通常比功能需求更关键，不能满足性能需求的系统将无法使用

3.3 需求获取技术

访谈、场景、用例、原型

3.4 需求验证

一致性、完整性、现实性、有效性

3.5 结构化分析

功能模型（数据流图（DFD 图））

3.6 面向对象分析

1. 类与类之间的关系
2. 用例图、顺序图、类图、状态图、活动图

4 软件设计

4.1 OOD 的 4 个部分

1. 数据/类设计：将类模型转化为设计类的实现和必要的数据结构
2. 体系结构设计：定义软件系统各主要成分之间的关系
3. 接口设计：软件内部各成分之间、软件与其他协同系统之间及软件与用户之间的交互机制
4. 构件设计：把软件系统各主要成分转化成软件构件的过程性描述

4.2 软件设计的 5 个基本原理

1. 模块化：把系统划分成独立命名且可独立访问的模块，每个模块完成一个功能，然后逐一实现这些模块，最后把这些模块集成起来构成一个整体
2. 抽象：抽出事物的本质而忽略细节

3. 逐步求精：把一个时期内必须解决的种种问题按优先级排序的技术细化过程

4. 信息隐藏：一个模块的数据和模块的实现细节对不需要这些信息的模块来说是不能访问的。独立的模块间仅仅交换为完成系统功能而必须交换的信息

5. 模块独立：使每个模块完成一个相对独立的特定子功能，并且和其他模块之间的关系很简单

4.3 面向对象设计的 3 个准则

1. 弱耦合

2. 强内聚

3. 复用：利用某些已开发的、对建立新系统有用的软件元素来生成新的软件系统，提高生产效率和软件质量

4.4 体系结构风格

数据为中心的体系结构、数据流体系结构、分层体系结构、C/S 体系结构

4.5 基本设计原则

开闭原则、类的单一职责原则、Liskov 替换原则、接口隔离原则、迪米特法则

4.6 构件详细设计

程序流程图、盒图、PAD 图、判定表、判定树、PDL（伪码）

5 软件测试

5.1 软件测试的目标

1. 测试是为了发现程序中的错误而执行程序的过程

2. 好的测试方案是极可能发现尚未发现的错误的测试

3. 成功的测试是发现了尚未发现的错误的测试

5.2 软件测试方法

黑盒测试、白盒测试

5.3 软件测试步骤

单元测试、集成测试、确认测试、系统测试

5.4 单元测试

1. 五个基本特性

- 接口：确保数据能够正确地进出构件

- 局部数据结构：确保程序执行过程中临时存储的数据的完整性

- 独立路径：执行控制结构中的所有独立路径，以确保构件中的所有语句至少执行一次
- 错误处理路径：设置适当的处理错误的通路，并认真测试这些通路
- 边界条件：处理数组 n 的第 n 个元素，i 次循环中的第 i 次迭代时的检测，检测最大值或最小值的边界

2. 驱动程序：接收测试数据，传送给被测试的模块，并且打印出有关的结果

3. 桩程序：代替被测模块所调用的模块，它使用被它代替的模块的接口，可能做最少量的数据操作，并把控制归还给调用它的模块

5.5 集成测试

1. 非渐增式集成、渐增式集成
2. 自顶向下集成、自底向上集成、三明治集成

5.6 回归测试

回归测试是指重新执行已经做过的测试的某个子集，以保证变更没有传播不期望的副作用

5.7 确认测试

1. α 测试：在开发者的场所由用户进行，在开发者关注和控制的环境下进行
2. β 测试：最终用户在自己的场所进行，开发者通常不在场

5.8 系统测试

恢复测试、安全测试、压力测试、性能测试

5.9 调试

1. 调试过程
2. 蛮干法、回溯法、原因排除法

5.10 白盒测试

1. 逻辑覆盖
- 语句覆盖：被测程序中的每个语句至少执行一次
- 判定覆盖：每个判定的取真分支和取假分支至少执行一次
- 条件覆盖：每个判定的每个条件的可能的取值至少执行一次
- 判定/条件覆盖：每个判定的每个条件都取到各种可能的值，且每个判定也都取到各种可能的结果
- 条件组合覆盖：每个判定的所有可能的条件取值组合至少执行一次
- 路径覆盖：每条可能的路径至少执行一次

2. 基本路径测试
- 画出流图
- 计算环形复杂度
- 以该复杂度为指南确定独立路径的基本集合

- 从该基本集合导出测试用例

5.11 黑盒测试

1. 等价类划分：把软件的输入数据集合按输入条件划分为若干个等价类，从中生成测试用例

2. 边界值分析：特别考虑等价类的边界值

3. 错误推断：靠经验和直觉推测程序中可能存在错误，从而有针对性的编写检查这些错误的用例

6 软件项目管理

1. 项目管理对象

2. 软件质量的度量

3. 软件项目估算：

- 功能点估算

4. 软件项目进度安排