软件工程导论

chapter1 软件工程介绍

**软件：**

1. **指令的集合（计算机程序），通过执行这些指令可以满足预期的特征、功能和性能需求**
2. **数据结构，使程序可以充分利用信息**
3. **描述程序操作和使用的文档**

软件分类（7类，后面+软件）：

系统，应用，工程/科学,嵌入式，产品线，Web应用，人工智能

软件特性：

1. 软件是设计的，而非制造的
2. 软件不会“磨损”，但会退化，软件工程方法的目的是降低软件失效曲线突变的幅度和斜率
3. 虽然工业向着基于构件的构造模式发展，但大多数软件仍根据用户需求定制

软件退化：

在完整的生存周期内，软件的每次变更都可能引入新的错误，使得失效率增加

不断的变更是软件退化的根本原因

软件危机：

落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

表现：

（1）软件成本日益增长

（2）开发进度难以控制

（3）软件质量差

（4）软件维护困难

原因：

（1）用户需求不明确

（2）缺乏正确的理论指导

（3）软件规模越来越大

（4）软件复杂度越来越高

解决方法：

用现代工程的概念，原理，技术和方法进行计算机软件的开发，管理和维护，由此诞生了软件工程。

软件工程是用工程，科学和数学的原则与方法来研制，维护计算机软件的有关技术及管理方法。

chapter2 过程综述

软件工程（定义）：

1. 将系统化的、规范的、可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件
2. 在（1）中所述方法的研究

（是一种层次化的技术）

软件工程 基础是过程 根基是质量关注点

软件过程（定义）：

软件过程是一个为建造高质量软件所需完成的任务的框架，即形成软件产品的一系列步骤。包括中间产品、资源、角色及过程中采取的方法、工具等范畴。

软件过程>过程框架>普适性活动>框架活动>软件工程动作>任务集

通用过程框架：沟通、策划、建模、构建、部署

惯例过程模型：提高软件质量、项目的可管理性以及对于交付时间和项目费用的可预测性

敏捷过程模型：强调可操作性和可适应性

**软件生命周期(Software Life Cycle,SLC)是软件的产生直到报废或停止使用的生命周期。**

**问题定义->可行性研究->需求分析->软件设计->程序编码->程序测试->运行维护**

**瀑布模型：工作线性，需求准确定义，相对稳定**

**增量过程模型：迭代运用瀑布模型，规避技术风险**

**原型开发模型：为定义需求服务，提出基本功能，无详细定义输入处理输出等情况**

**螺旋模型：将瀑布模型和**[**快速原型模型**](https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E5%8E%9F%E5%9E%8B%E6%A8%A1%E5%9E%8B)**结合起来，强调了其他模型所忽视的风险分析，采用循环的方式逐步开发完善软件，特别适合于大型复杂的系统。**[**螺旋模型**](https://baike.baidu.com/item/%E8%9E%BA%E6%97%8B%E6%A8%A1%E5%9E%8B)**由风险驱动，强调可选方案和约束条件从而支持软件的重用，有助于将**[**软件质量**](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F)**作为特殊目标融入产品开发之中。**

**喷泉模型：以用户需求为动力，以对象为驱动的模型，主要用于描述**[**面向对象**](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1/2262089)**的软件开发过程。该模型认为**[**软件开发过程**](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E8%BF%87%E7%A8%8B/3758917)**自下而上周期的各阶段是相互迭代和无间隙的特性。**

**统一过程：用例驱动，以架构为核心，迭代并且增量。**

**起始 细化 构建 转化 生产**

**1.任何功能开发后就进入测试过程，及早进行验证**

**2.早期风险识别，采取预防措施**

chapter 4 敏捷

**敏捷：**

**适应变更，交通通畅，客户参与，有效控制**

**强调可运行软件（增量）的快速交付**

**利用变更为客户创造竞争优势**

**极限编程XP：**

**策划，设计，编码，测试**

**设计简洁 鼓励重构**

**编码关键概念 结对编程 详细代码-接口**

**测试：在编码开始前进行单元测试**

chapter6 系统工程

**系统：组织在一起通过处理信息来实现预定目标的要素集合或排列**

**系统要素：软件，硬件，人员，数据库，文档，规程**

系统建模：假设 简化 限制 约束 偏好

系统模型分类：HP建模 UML建模

chapter 7 需求工程

**需求工程任务：**

**起始、导出、精化、协商、规格说明、确认和管理**

需求工程工作产品：

1. 必要性和可行性陈述
2. 系统或产品的范围说明
3. 参与需求导出的客户、用户和其它共同利益者的列表
4. 系统技术环境的说明
5. 需求列表及每个需求适用的领域限制
6. 一系列使用场景
7. 能够更好定义需求的原型

**需求开发的方法:参考需求工程任务**

chapter 8 构建分析模型

分析模型的作用：

使用文字和图表形式综合描绘需求的数据、功能和行为，可以更直接地评审它们的正确性、完整性和一致性

从多个维度表现需求，减少出错遗漏的几率

分析模型的目标：

1. 描述客户需要什么
2. 为软件设计奠定基础
3. 定义在软件完成后可以被确认的一组需求

分析模型的构建原则：

关注可见需求 抽象级别要求相对高

分析模型的每个元素都应能增加对软件需求的整体理解

对于基础结构和其他非功能性的模型应推延到设计阶段再考虑

最小化整个系统的关联

确认分析模型为所有共利益者都带来价值

尽可能保持模型简洁

chapter 9

书上概念

chapter 10

体系结构风格：

以数据为中心

数据流（管道）体系结构（传统编译器）

调用和返回体系结构

面向对象体系结构

层次体系结构

chapter 11 构件级建模

构件：计算机软件中模块化的构造快

系统中某一定型化、可配置和可替换的部件，封装实现并暴露了一系列接口

构件设计原则:

**开关原则：用无需对构建内部做修改就可以进行扩展的方式说明构件（分离缓冲区）**

**替换原则：子类可以替换基类（替换后使用基类的构件仍能正常工作）**

**依赖倒置原则：依赖于抽象而非具体实现**

**接口分离原则：多个用户专用接口比一个通用接口好**

chapter 13、14

**单元测试：**

**侧重于构件中的内部处理逻辑和数据结构，可以对多个构件并行执行**

**测试接口、数据结构、边界条件、路径**

**内聚性：**

**构件或者类只封装那些相互关联密切的构件或类自身有密切关系的属性和操作**

**耦合性：**

**类之间彼此联系程度的一种定性度量**

**尽可能保持低耦合**

**集成测试：**

**是构造软件体系结构的系统化技术**

**自顶向下，自底向上**

**目标是利用已通过单元测试的构建建立设计中描述的程序结构**

**可进行增量测试**

确认测试：用户可见

系统测试：恢复 安全 压力 性能

测试用例：

具有可测试性

操作 观察 控制 分解 简单 稳定 易理解

好的测试特征：

具有较高的发现错误的可能性

不冗余

不简单也不复杂

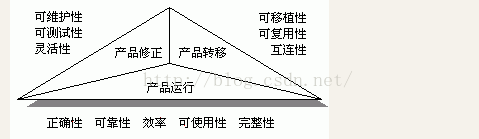
**黑盒：测试产品所完成的指定功能（功能需求），接口处执行测试，检查系统的基本方面**

**白盒：测试产品的内部运行情况（程序控制结构），基于过程细节的封闭检查，测试贯穿软件的逻辑路径和构件间的协作**

**手工测试、自动化测试**

chapter 15.22

**McCall的质量因素**

****

**measure测度 产品特性量化展示**

**metrics 度量一个xx具有给定属性的量化测量标准**

**Indicators指标 一个度量或多个度量的组合**

**度量的作用**

**为创建有效的分析模型、设计模型、可靠的代码和完全的测试提供必要的理解**

chapter 21 项目管理

**4p：人员people 产品product 过程procedure 项目project**

**W5HH：Why What When Who Where How Howmuch**

chapter 23

chapter 24

**里程碑**

**分析 设计 程序设计 测试**

**（分各个阶段）**

chapter 25风险

**RMMM**

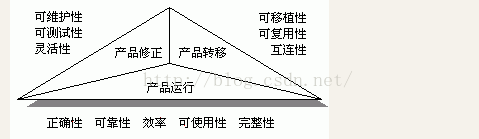
**风险缓解监测和管理计划**

**计划将所有风险分析工作文档化**

**评估风险 保证缓解 收集信息**

chapter 26

**McCall软件质量模型：**



软件质量保证活动：

质量的量化、标准

正式技术评审：FTR方式

质量保证成本：预防、鉴定、失效（成本）

chapter 27

**软件配置管理SCM**

**标识 变更控制 版本控制 配置审核 报告**

**提高适应变更的能力 减少所需要花费的工作量**

**Verification (验证) : You do it right.**

**侧重在软件系统是否正确地反映了需求.**

**Validation (确认) : You do the right thing.**

**侧重在是否按照客户的意图来进行系统开发**