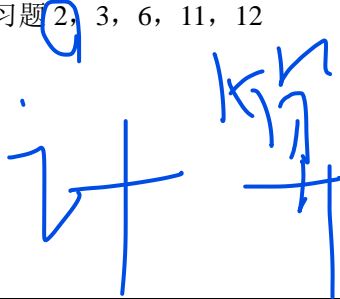


SE Summary 2022

各章主要知识点、图表、案例，结合实验、项目实践

Content	Points
第 1 章：软件工程概述 1.1 什么是软件工程 1.1.1 问题求解 1.1.2 软件工程师的角色是什么 1.2 软件工程取得了哪些进展 1.3 什么是好的软件 1.3.1 产品的质量 1.3.2 过程的质量 1.3.3 商业环境背景下的质量 1.4 软件工程涉及的人员 1.5 系统的方法 1.5.1 系统的要素 1.5.2 相互联系的系统 1.6 工程的方法 1.6.1 盖房子 1.6.2 构建系统 1.7 开发团队的成员 1.8 软件工程发生了多大的变化 1.8.1 变化的本质 1.8.2 软件工程的 Wasserman 规范 1.9 信息系统的例子	软件危机、计科与软工的关系 图 1-3、 Error、fault、failure 图 1-4; 高质量软件 (3 方面)、CMM、泛型 (paradigm)、 系统的要素、系统的定义 图 1-9 , 系统边界、 改变软件开发的关键因素 图 1-12、增量开发、 开发过程及团队成员的角色 图 1-11 Wasserman 规范、图 1-14 , 图 1-17 CASE 习题: 1.1, 1.3, 1.5, 1.8
第 2 章：过程和生命周期的建模 2.1 过程的含义 2.2 软件过程模型 2.2.1 瀑布模型 2.2.2 V 模型 2.2.3 原型化模型 2.2.4 可操作规格说明 2.2.5 可转换模型 2.2.6 阶段化开发：增量和迭代 2.2.7 螺旋模型 2.2.8 敏捷方法 2.3 过程建模工具和技术 2.3.1 静态建模：Lai 表示法 2.3.2 动态建模：系统动力学 2.4 实际的过程建模 2.4.1 Marvel 的案例研究 2.4.2 过程建模工具和技术应该具有的特性	过程、软件生命周期、软件开发通常包含的 9 个阶段。 软件生命周期模型：瀑布模型的缺点、阶段化开发：增量开发和迭代开发、螺旋模型 (图 2-9、2-10)、 敏捷方法: 4 原则、典型方法 (XP、Crystal、Scrum、ASD) XP: 4 特性, 12 个实践操作 (对编程); 静态模型、动态模型的区别 习题: 2.3, 2.8, 2.11
第 3 章 计划和管理项目	Project schedule、activity、milestone、precursor、activity graph、

<p>3.1 跟踪项目进展</p> <p>3.1.1 工作分解和活动图</p> <p>3.1.2 估算完成时间</p> <p>3.1.3 跟踪进展的工具</p> <p>3.2 项目人员</p> <p>3.2.1 人员角色和特性</p> <p>3.2.2 工作风格</p> <p>3.2.3 项目组织</p> <p>3.3 工作量估算</p> <p>3.3.1 专家判断</p> <p>3.3.2 算法方法</p> <p>3.3.3 机器学习方法</p> <p>3.3.4 找出适合具体情形的模型</p> <p>3.4 风险管理</p> <p>3.4.1 什么是风险</p> <p>3.4.2 风险管理活动</p> <p>3.5 项目计划</p> <p>3.6 过程模型和项目管理</p> <p>3.6.1 注册管理</p> <p>3.6.2 责任建模</p> <p>3.6.3 紧密结合里程碑</p>	<p>图 3-3 、CPM、表 3-4、图 3-4, Gantt chart 图 3-6、</p> <p>人员工作风格(4 种)、chief programmer team 图 3-11、egoless approach 、表 3-5</p> <p>从团队管理者和成员两个角度, 探讨团队最好的管理、工作方式。</p> <p>COCOMO II 基本模型、3 阶段、应用点 AP、FP、工作量估算、风险暴露 Risk Exposure、风险影响、风险概率 图 3-16</p> <p>降低风险的 3 种策略、risk leverage、CASE</p> <p>项目管理: 结合自己项目</p> <p>习题 2, 3, 6, 11, 12</p> 
<p>第 4 章 获取需求</p> <p>4.1 需求过程</p> <p>4.2 需求引发</p> <p>4.3 需求的类型</p> <p>4.3.1 解决冲突</p> <p>4.3.2 两种需求文档</p> <p>4.4 需求的特性</p> <p>4.5 建模表示法</p> <p>4.5.1 实体-联系图</p> <p>4.5.2 例子: UML 类图</p> <p>4.5.3 事件踪迹</p> <p>4.5.4 例子: 消息时序图</p> <p>4.5.5 状态机</p> <p>4.5.6 例子: UML 状态图</p> <p>4.5.7 例子: Petri 网</p> <p>4.5.8 数据流图</p> <p>4.5.9 例子: 用例</p> <p>4.5.10 函数和关系</p> <p>4.5.11 例子: 判定表</p> <p>4.5.12 例子: Parnas 表</p> <p>4.5.13 逻辑</p> <p>4.5.14 例子: 对象约束语言 (OCL)</p> <p>4.5.15 例子: Z</p> <p>4.5.16 代数规格说明</p>	<p>SRS、敏捷需求建模、风险承担者、需求源 图 4-2。</p> <p>需求的类型 表 4-2、需求定义与需求规格说明 图 4-3。</p> <p>需求建模 :E-R,UML; association 、 aggregation 、 composition、generalization</p> <p>MSC、State machine、Petri 网 图 4-12、数据流图 DFD、</p> <p>use-case diagram 用例图 图 4-15、decision table 判定表 图 4-16、Parnas 表、时态逻辑、OCL、SDL 3 个主要图 图 4-21。</p> <p>快速原型化的两种技术: throwaway, evolutionary</p> <p>建模与原型化在需求问题上的区别</p> <p>习题 1, 5, 6, 9</p>

<p>4.5.17 例子：SDL 数据</p> <p>4.6 需求和规格说明语言</p> <p>4.6.1 统一建模语言（UML）</p> <p>4.6.2 规格说明和描述语言（SDL）</p> <p>4.6.3 软件成本降低（SCR）</p> <p>4.6.4 需求表示法的其他特征</p> <p>4.7 原型化需求</p> <p>4.8 需求文档</p> <p>4.8.1 需求定义</p> <p>4.8.2 需求规格说明</p> <p>4.8.3 过程管理和需求的可跟踪性</p> <p>4.9 确认和验证</p> <p>4.9.1 需求确认</p> <p>4.9.2 验证</p> <p>4.10 测量需求</p> <p>4.11 选择规格说明技术</p>	
<p>第5章 设计体系结构</p> <p>5.1 设计过程</p> <p>5.1.1 设计是一种创造性过程</p> <p>5.1.2 设计过程模型</p> <p>5.2 体系结构建模</p> <p>5.3 分解和视图</p> <p>5.4 体系结构风格和策略</p> <p>5.4.1 管道和过滤器</p> <p>5.4.2 客户—服务器</p> <p>5.4.3 对等网络</p> <p>5.4.4 发布—订阅</p> <p>5.4.5 信息库</p> <p>5.4.6 分层</p> <p>5.4.7 组合体系结构风格</p> <p>5.5 满足质量属性</p> <p>5.5.1 可修改性</p> <p>5.5.2 性能</p> <p>5.5.3 安全性</p> <p>5.5.4 可靠性</p> <p>5.5.5 健壮性</p> <p>5.5.6 易使用性</p> <p>5.5.7 商业目标</p> <p>5.6 协作设计</p> <p>5.7 体系结构的评估和改进</p> <p>5.7.1 测量设计质量</p> <p>5.7.2 故障树分析</p> <p>5.7.3 安全性分析</p> <p>5.7.4 权衡分析</p> <p>5.7.5 成本效益分析</p>	<p>SAD、Architecture style（7种，分析对比）、</p> <p>7个满足质量属性的策略、fault-tree 故障树、cut-set-tree 割集树，图 5-12.</p> <p>KWIC 的不同设计方案对比。实验任务</p> <p>Active design review、Passive design review</p> <p>ROI, payback period</p> <p>Product line, core assert base、产品线的理念。</p> <p>习题 1, 3, 11, 14</p>

5.7.6 原型化 5.8 文档化软件体系结构 5.8.1 视图间的映射 5.8.2 文档化设计合理性 5.9 体系结构设计评审 5.9.1 确认 5.9.2 验证 5.10 软件产品线 5.10.1 战略范围 5.10.2 产品线体系结构的优势 5.10.3 产品线的演化	
第6章 设计模块 6.1 设计方法 6.2 设计原则 6.2.1 模块化 6.2.2 接口 6.2.3 信息隐藏 6.2.4 增量式开发 6.2.5 抽象 6.2.6 通用性 6.3 面向对象的设计 6.3.1 术语 6.3.2 继承与对象组合 6.3.3 可替换性 6.3.4 德米特法则 6.3.5 依赖倒置 6.4 在UML中体现面向对象设计 6.4.1 过程中的UML 6.4.2 UML类图 6.4.3 其他UML图 6.5 面向对象设计模式 6.5.1 模板方法模式 6.5.2 工厂方法模式 6.5.3 策略模式 6.5.4 装饰者模式 6.5.5 观察者模式 6.5.6 组合模式 6.5.7 访问者模式 6.6 设计中其他方面的考虑 6.6.1 数据管理 6.6.2 异常处理 6.6.3 用户界面设计 6.6.4 框架 6.7 面向对象度量 6.7.1 面向对象系统规模的度量	设计原则：6种 继承与组合 图6-13、 Liskov Substitutability Principle、 Law of Demeter 图6-14、 dependency inversion 依赖倒置 图6-15. 过程中的UML 图6-16 设计模式 Design pattern: (7种)。 模式与框架 Specialization Index ch6 习题 1, 22

6.7.2 面向对象系统设计质量的度量 6.7.3 在何处进行面向对象测量 6.8 设计文档	
第7章 编写程序 7.1 编程标准和过程 7.1.1 对单个开发人员的标准 7.1.2 对其他开发人员的标准 7.1.3 设计和实现的匹配 7.2 编程的指导原则 7.2.1 控制结构 7.2.2 算法 7.2.3 数据结构 7.2.4 通用性指导原则 7.3 文档 7.3.1 内部文档 7.3.2 外部文档 7.4 编程过程 7.4.1 将编程作为问题求解 7.4.2 极限编程 7.4.3 结对编程 7.4.4 编程向何处去	编程规范标准 指导原则：复用、通用性 文档：内部文档、外部文档 编程过程：XP、结对编程、融合、小组协同
第8章 测试程序 8.1 软件故障和失效 8.1.1 故障的类型 8.1.2 正交缺陷分类 8.2 测试的相关问题 8.2.1 测试的组织 8.2.2 对测试的态度 8.2.3 谁执行测试 8.2.4 测试对象的视图 8.3 单元测试 8.3.1 检查代码 8.3.2 证明代码正确性 8.3.3 测试程序构件 8.3.4 技术比较 8.4 集成测试 8.4.1 自底向上集成 8.4.2 自顶向下集成 8.4.3 一次性集成 8.4.4 三明治集成 8.4.5 集成策略的比较 8.5 测试面向对象系统 8.5.1 代码测试 8.5.2 面向对象测试和传统测试之间的区别 8.6 测试计划	故障的类型 测试步骤 图 8-3 黑盒测试、白盒测试 单元测试：测试点、测试用例 集成测试策略：6 种 修改后的自顶向下图 8-11、改进的三明治测试 图 8-14 Fault seeding 故障播种、两小组测试例、confidence 可信度 习题 10, 11

8.6.1 计划的目的是 8.6.2 计划的内容 8.7 自动测试工具 8.7.1 代码分析工具 8.7.2 测试执行工具 8.7.3 测试用例生成器 8.8 什么时候停止测试 8.8.1 故障播种 8.8.2 软件中的可信度 8.8.3 其他的停止测试的标准 8.8.4 识别易出故障的代码	
第9章 测试系统 9.1 系统测试的原则 9.1.1 软件故障根源 9.1.2 系统测试过程 9.1.3 配置管理 9.1.4 测试小组 9.2 功能测试 9.2.1 目的与职责 9.2.2 因果图 9.3 性能测试 9.3.1 目的和职责 9.3.2 性能测试的类型 9.4 可靠性、可用性以及可维护性 9.4.1 定义 9.4.2 失效数据 9.4.3 测量可靠性、可用性和可维护性 9.4.4 可靠性稳定性和可靠性增长 9.4.5 可靠性预测 9.4.6 操作环境的重要性 9.5 验收测试 9.5.1 目的和职责 9.5.2 验收测试的种类 9.5.3 验收测试的结果 9.6 安装测试 9.7 自动化系统测试 9.8 测试文档 9.8.1 测试计划 9.8.2 测试规格说明和评估 9.8.3 测试描述 9.8.4 测试分析报告 9.8.5 问题报告表 9.9 测试安全攸关的系统 9.9.1 设计多样性 9.9.2 软件安全性案例	软件故障根源; 测试过程 图 9-2 构建(集成)计划、spin 配置管理, regression test 回归测试 控制版本和发布的 3 种方式: delta、单独文件、条件编译 测试小组构成 因果图 图 9-5, 表 9-2 α 测试、β 测试、并行测试。 测试文档 习题 4

