第一章 软件与软件工程

1、软件失效曲线：

由于变更的副作用导致失效率提高。不断的变更是软件退化的根本原因。

·遗留软件（1-26）

2、当代软件工程的目标：

是“修改在进化论理论上建立的方法论”，即“软件系统不断经历变更，新的软件系统从旧系统中建立起来，并且所有新旧系统都必须具有互操作性和协作性。”

3、**软件工程**定义：

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。

4、**软件工程层次图（P7）/层次化技术（2-5）**：质量关注点、过程、方法、工具。

·软件过程、活动、动作、任务（P7）

·**过程**不是对如何构建计算机软件的严格规定，而是一种具有可适应性的调整方案。

5、通用的软件工程过程框架通常包含以下：（2-20，2-29）

（1）**5个框架活动**：

沟通、策划、建模（需求分析设计）、构建（编码测试）、部署（交付维护）。

（2）典型的**普适性活动**：

软件项目跟踪和控制、风险管理、软件质量保证、技术评审、测量、软件配置管理、可复用性管理、工作产品的准备和生产。

6、软件工程实践的精髓，解决问题的精髓：理解问题（沟通和分析）、策划解决方案（建模和软件设计）、实施计划（代码生成）、检查结果（测试和质量保证）。

7、关注软件工程整体实践的原则：存在价值、保持简洁、保持愿景、关注使用者、面向未来、提前计划复用、认真思考。

第二章 过程模型

**1、软件过程框架描述：**

每个框架活动由一系列软件工程动作构成；

每个软件工程动作由若干个任务集来定义；

这个任务集明确了将要完成的工作任务、将要产生的工作产品、所需要的质量保证点、以及用于表明过程状态的项目里程碑。（3-8）

·举例小、大项目中，某个框架活动的某一动作的**任务集**（P19）

·过程流（P17）

**2、传统过程模型（4-8）**

（1）瀑布模型：图、优缺点（P20）

（2）V字模型：图（4-9）

（3）原型开发过程模型：开始于沟通，建模快速设计（P21）

（4）增量过程模型：构造一系列可执行的中间版本

（5）演化过程模型：降低风险，采用最广泛，比如螺旋模型、原型模型、并行模型

（6）**统一过程UP**

一种“用例驱动，以构架为中心的迭代和增量”软件过程和统一建模语言（UML）紧密结合。

（7）**统一软件过程RUP**

一种“风险驱动”的基于UML和构件式架构的迭代、递增型开发过程。

四个阶段：初始、精化、构建、转移（4-28）

（8）**统一建模语言UML**

作为需求模型和设计模型的表示方式，提供了必要的技术支持和面向对象的软件工程实践，但它并不提供过程框架来指导项目团队。UML包含了大量用于面向对象系统建模和开发的符号，已经成为面向对象软件开发的行业标准。

3、个人软件过程PSP和团队软件过程TSP

第三章 敏捷和敏捷过程

1、**敏捷开发**：软件工程精简版，它保留了基本的框架活动（沟通策划建模构件部署），但将其缩减到一个推动项目组朝着构建和交付发展的最小任务集。（P30）

2、**敏捷软件开发宣言**：“个人和他们之间的交流胜过开发过程和工具；可运行的软件胜过宽泛的文档；客户合作胜过合同谈判；对变更的良好响应胜过按部就班的遵循计划。”

3、**敏捷**（5-4）可以应用于任何软件过程。为了实现这一目标，过程的设计应使项目团队适应于任务，并且使任务流水线化，在了解敏捷开发方法的流动性的前提下进行计划的制定，保留最重要的工作产品并使其保持简洁，强调这样一个增量交付策略：根据具体的产品类型和运行环境，尽快将可工作的软件交付给客户。

4、敏捷软件过程具有增量的可适应性。使用增量式开发策略，在很短的时间间隔内交付软件增量（可执行原型或部分实现的可运行系统）来适应（不可预测的）变更的步伐。

**5、两类敏捷过程模型：**

（1）**Scrum**

·框架活动包括：需求、分析、设计、演化、交付；

·开发活动由工作单元packets组成

·**冲刺**sprint：每个框架活动中工作任务在相对较短的时间盒的期限内完成称为一个冲刺；冲刺是一个个的增量原型开发周期，每个周期为2-4周。

·过程流图（P33），特点（P35）

·Scrum团队：一个自组织的跨领域团队，由产品负责人（产品待定项）、Scrum master（Scrum管理）、和一个小型开发团队（设置冲刺）组成。

·Scrum开发的主要制品：产品待定项、冲刺待定项backlog、代码增量，燃尽图（5-31）。

（2）**XP框架（极限编程）**

·框架活动包括：策划、设计、编码、测试；

·**用户故事**描述待开发软件所需的输出、特征以及功能。

·**项目速度**是第一个发布版本中实现的客户故事个数。

·**CRC卡**（类-职责-协作者）确定和组织与当前软件增量相关的面向对象的类。

·**重构**：不改变软件外部行为的修改/优化代码的方式，意味着在构建系统时持续进行设计。

·现场客户：·结对编程：·测试驱动：·持续集成：见末尾

**·XP与RUP的共性与区别（P36，5-22）**

第六章 理解需求

1、**需求工程**：

是指致力于不断理解需求的大量任务和技术。他是一个软件工程动作，开始于沟通并持续到建模活动，为设计和构建奠定了坚实的基础。

（1）需求工程包括七项任务：起始、获取、细化、协商、规格说明、确认和管理。（P66）

**（2）规格说明**：可以是一份写好的文档、一套图形化的模型、一个形式化的数学模型、一组使用场景、一个原型或上述各项的任意组合。

·**需求规格说明书SRS**（模板8-46）

（3）软件需求（8-7）

·特性：**FURPS+**——功能性、可用性、可靠性、性能、可支持性 + 设计约束、实现需求、接口需求、物理需求

·软件需求的三个层次（8-11）

2、建立根基

（1）确认**利益相关者**（stakeholder）：直接或间接从正在开发的系统中获益的人。如业务运行管理人员、产品管理人员、市场营销人员、内部和外部客户、最终用户、顾问、产品工程师、软件工程师、支持和维护工程师以及其他人员。（P280）

识别**项目干系人**：是积极参与项目，或其利益因项目的实施或完成受到积极或消极影响的个人和组织，他们还会对项目的目标和结果施加影响。

（2）识别多重观点；（3）协同合作；

（4）**首次提问：**元问题-关注沟通活动本身的效率（P69，8-21）

问题，影响到的人，具体影响，一个成功的解决方案

（5）非功能需求NFR；（6）可追溯性（P70）

3、获取需求的方法（8-30）

（1）协作收集需求；

·**小规格说明**：在很多情况下列表描述的对象或服务需要更多的解释，为了完成这一任务，利益相关者为列表中的条目编写小规格说明。

（2）使用场景；（3）获取工作产品

4、开发用例

（1）**用例**：从“最终用户的角度”描述系统或软件，讲述有固定风格的故事“最终用户如何在特定环境下与系统交互”。

5、构建分析模型（有基于场景的、基于类的、行为元素）

6、协商需求；7、需求监控；8、确认需求；9、需求验证；（P81，8-50）

10、需求评审（审查、小组评审、结对编程、走查、同级调查、轮查、临时评审）

11、需求管理（建立需求基线、需求跟踪、需求变更控制）

第七章 需求建模

1、各类模型

（1）**基于场景的建模**：用例图与用例说明，（活动图、泳道图）

·用例“**异常**”一般是代表对用例的异常处理，它描述了这样一种场景（可能是失败条件或参与者选择了替代方案），该场景导致系统展示出某些不同的行为。用例应该注明异常处理：即如果软件能检测出异常所发生的条件，就应该马上处理这个条件。在某些情况下，异常处理可能拖累其他用例处理条件的开发。（9-19）

（2）**基于类的建模**：类图（关系9uml-30）、状态图，UML建模，CRC建模（P96）

·识别分析类的表现（外部实体、事物、角色等）和特征（P92）

·描述属性

·定义操作

（3）功能建模：活动图、顺序图

·交互图：顺序图、协作图

（4）**行为模型**：状态图、活动图/泳道图、顺序图

（4）数据模型：域分析模型，描述问题信息域的模型（9-7）

（5）**面向流的模型**：数据流图；表示系统的功能元素并且描述当功能元素再系统中运行时怎样进行数据交换。

第八章 设计概念

1、坚固性、适用性（程序符合开发目标）、愉悦性

**2、从需求模型到设计模型的转换图（P107 重要）**

3、设计过程

（1）软件**质量指导原则**和属性（P109）

（2）**通用设计任务集**列表（P111）

4、设计概念

（1）数据抽象：是描述数据对象的命名数据集合

（2）软件**体系结构**：软件的整体结构和这种结构为系统提供概念完整性的方式；是程序构件（模块）的结构或组织、这些构件交互的方式以及这些构件所用数据的结构。（12-15）

（3）**模式**；（4）**关注点分离**；（5）**模块**化（模块数-成本的图P114）

（6）信息**隐蔽**原则；（12-23）

（7）功能独立性：**内聚性和耦合性**；

（8）**逐步求精**：一种自顶而下的设计策略；

（9）**重构**：是使用这样一种方式改变软件系统的过程：不改变代码设计的外部行为而是改进其内部结构，来使之得到简化。

（10）**设计类**：分析类细化为设计类（实体类、边界类、控制类），具有完整性与充分性、原始性、高内聚性、低耦合性。

5、设计模型

**（1）分析模型与设计模型的维度图（P118，12-6 重要）**

（2）包含的**设计元素**（P120，12-5、34之后）：数据/类、体系结构、接口、构件级（构件图）、部署级（部署图）

·接口设计的三个元素：用户界面；与其他系统、设备、网络、信息生成者或使用者的外部接口；各种设计构建之间的内部接口。

第九章 体系结构设计

0、是什么和重要性（13-3）

**1、体系结构风格的简单分类（图 P130）**

（1）**以数据为中心的体系结构**（黑板模式，13-17）

（2）**数据流体系结构**（管道与过滤器）

（3）调用和返回体系结构（主程序/子程序，远程过程调用）

（4）面向对象的体系结构

（5）**层次体系结构**（MVC体系结构）

2、体系结构模式（并发性、持久性、分布性）；体系结构考虑要素（P134）

3、体系结构设计

（1）体系结构上下文图

（2）定义体系结构原型

（3）将体系结构细化为构件

（4）描述系统实例（体系结构描述语言ADL，13-36）

第十章 构件级设计

1、**构件**：系统中模块化的、可部署的、可替换的部件，该部件封装了实现并对外提供一组接口。

2、架构设计的目的/好处：模块与层次之间的关系/联系更加清晰、具有逻辑性

3、**设计基于类的构建**

**（1）基本设计原则**（14-8）

·**开闭原则（OCP）**：模块（构件）应该对外延具有开放性、对修改具有封闭性。

·Liskov替换原则（LSP）：子类可以替换它们的基类

·依赖倒置原则（DIP）：依赖抽象，而非具体实现

·接口分离原则（ISP）：多个客户专用接口比一个通用接口要好

·发布/复用等价性原则（REP）：复用的粒度就是发布的粒度

·共同封装原则（CCP）：一同变更的类应该合在一起

·共同复用原则（CRP）：不能一起复用的类不能被分到一组

（2）构件级设计指导方针

·构件、接口、依赖与继承（P151）

（3）**内聚**性/专一性，越高越好

·功能内聚、分层内聚、通信内聚、偶然/逻辑/时间/过程内聚

（4）**耦合**性，越低越好

·内容/公共/控制/标记/数据/非直接

4、实施构件级设计的步骤（P154）

5、**基于构件的软件工程CBSE**和软件设计（P156，14-27）

·对象请求代理ORB（14-34），构建对象模型COM（14-36）

第十二章 质量概念

1、质量的概念：各类观点（P185）

2、设计质量，符合质量，用户满意度公式

3、**软件质量**定义：在一定程度上应用有效的软件过程，创造有用的产品，为生产者和使用者提供明显的价值。

4、质量模型：使用质量模型、产品质量模型（P186，质量维度19-10）

5、软件质量困境：通过足够的努力来产生可接受的质量而不埋没项目

6、“**足够好的软件**”：提供用户期望的高质量功能和特性，但同时也提供了其他更多的包含已知错误的理解的或特殊的功能和特性。软件供应商希望广大的最终用户忽视错误，因为他们对其他的应用功能是如此满意。

7、质量的成本（P189）

（1）预防成本

（2）失效成本：内部失效（软件发布之前）、外部失效

**8、实现软件质量的四大管理和时间活动（P193，19-21）**

（1）软件工程方法；（2）项目管理技术；（3）**质量控制QC**；（4）**质量保证QA**

测试和评审

第十三章 评审

1、软件评审是软件工程工作流中的“过滤器”；

2、从质控角度出发，技术评审**TR**最有效

隐错bug，故障fault，错误error，缺陷defect

3、缺陷放大，缺陷传播

4、评审度量相关公式：工作量、错误数、错误密度（P197）

5、非正式评审IPR，走查WT，审查IN，循环评审RRR

6、**正式技术评审FTR**：是一种由软件工程师以及其他人进行的软件质量控制活动。（P200）

7、评审会议：FTR关注的范围越小，发现错误的可能性越大。

·样本驱动评审SDRs（20-26）

8、评审指导原则（P202）；评审指标（20-27）

第十四章 软件质量保证

1、包含6个方面：（P207）

2、覆盖的内容、活动要素：（P2）

3、SQA工作组的职责/任务/目标/度量：（P209，21-5）

第十五-十六章 软件测试

1、测试原则，通用测试标准（22-19），好的“测试”（23-17）

2、测试生存周期：测试计划、测试用例设计、开发、执行、结果数据的收集评估

3、缺陷及缺陷等级：（23-12）

4、终止测试的标准：严格（23-15）

**5、验证与确认（V&V）**

·验证：确保软件正确的实现某一特定功能的一系列活动；

·确认：确保开发的软件可追溯到客户需求的一系列活动；

**6、测试策略图（P218，22-7）满足V字模型**

---------------------单元测试----------------------------

8、测试用例设计（23-20）

（1）**测试用例**定义：

（2）包含的方法：（23-31）

9、黑盒测试，白盒测试

**（1）白盒测试（P255）内部视角**（23-24）

·局部数据结构、独立路径

·所有独立路径至少被执行一次

·基本路径测试：

流图、计算环复杂度、导出独立路径、逐个设计测试用例（输入+预计输出）

**（2）黑盒测试（P22）外部视角**（23-37）

·边界条件、错误处理路径

·接口测试

·等价类划分

对有效等价类设计一个测试用例即可，对所有无效等价类都需要一一设计用例

·边界值分析BVA

10、其他测试

·面向对象的测试OO（24-...）

面向对象的类测试等同于传统软件的单元测试

行为测试

·高级测试：

α测试——开发者视角

β测试——用户视角，测试更充分

---------------------集成测试22-13----------------------------

11、自顶而下集成测试

12、自底而上集成测试

**13、持续集成SCI（P237）**

冒烟测试ST：对整个系统进行彻底的测试

·bug太多，要先进行代码评审，再次测试并补充测试用例

14、回归测试RT（22-17）

人工智能应用，测自身代码及相关代码，重复测试（P238）

第十七章 软件配置管理

1、**软件配置管理SCM**定义：是在整个软件过程中应用的一种普适性活动（P245）

2、一些概念（29-1/18）

（1）**基线**：“已经通过正式评审和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制规程才能修改它。”它能帮助我们在不严重阻碍合理变更的条件下控制变更。是软件开发中的里程碑。其标志是在正式技术评审中已经获得批准的一个或多个软件配置项的交付。

（2）软件配置（对象）项：（P249）

（3）依赖性和变更管理

**3、SCM过程的层次图（P253）**

**4、变更控制过程图（P253）**

补充

1、项目估算（33--）

项目估算、软件范围、估算技术、估算准确性

2、项目进度安排（34--）

延误原因，进度安排原则，挣值分析与计算

问答举例：

**0、在需求活动的协商情景中，“双赢“意味着什么？**

“双赢”是在协商情景中能够取得的最好的结果。

“双赢”中的“双”指的是利益相关者和软件团队。利益相关者的“赢”在于获得能满足客户大多数需要的系统或产品；而作为软件团队的一员，“赢”在于按照实际情况、在可实现的预算和时间期限内完成工作。

在一个理想的需求工程情境中，需求工程任务（起始、获取和细化）能确保得到足够详细的客户需求，以开展后续的软件工程活动。但是实际上，一个或多个利益相关者，是必须要进入协商过程的，在多数情况下要让利益相关者以成本和产品投放市场的时间为背景，权衡功能、性能和其他的产品或系统特性。因此，协商过程的目的就是：保证所开发的项目计划在满足利益相关者要求的同时，也能反映软件团队所处真实世界的限制（如时间、人员、预算等）。

对于需求活动的协商情景，可以不再采用传统的需求规格说明书方式，而是采用“握手”的双向沟通过程——“握手”是一种实现“双赢”的方式。在握手过程中，软件团队提出需求解决方案、描述它们的影响、与客户代表沟通他们的意图，客户代表审核提议的解决方案、关注丢失的特性并寻求新需求的清晰性。如果客户接受提议的解决方案，则说明需求是足够好的。握手方法有助于多样需求的识别、分析和多样选择，并可促进以双赢为目标的协商。

**1、对以下四个方面各设计5个问题**

A 设计者对于用户要问的问题

1）对产品的基本需求有哪些？

2）对产品开发过程所使用的开发工具（语言、框架等）有无特殊要求？

3）对产品的预期发布时间有无限制？（确定大致的需求期限）

4）对这一版的产品设计是否满意？

5）哪些设计/功能需要修改？还有什么新的要求？

B 用户对设计者应该问的问题

1）是否明白了我们的需求以及产品待实现功能的范围？

2）是否具备设计开发此产品的能力（工具、人员、技术设备等）？

3）对于各项功能模块的实现，预期的完成时间是多少？

4）关于软件长期的运行与维护，应该如何解决？

5）如果我们对测试结果不满意，或者中途有新需求想要加入，应该如何解决？

C 用户对将要构建的软件的自问

1）这个软件产品需要有哪些功能，包含哪些方面？（需求）

2）这个软件产品的设计开发成本、代价大约是多少？（成本）

3）我们大致什么时间需要这个软件产品投入使用？（时间）

4）能否找到擅长这类产品实现的设计开发人员？（专业性与可靠性）

5）这个软件产品有哪些功能需要特别关注（比如权限、用户信息安全等）？

D 设计人员对于软件产品和建造该产品采取的软件过程的自问

1）这个软件产品的功能需求、范围、目的以及时间期限具体是什么？

2）建造该产品需要什么样的工具（语言、框架、设备等）？

3）应该基于什么样的过程模型来设计实现比较合适？

4）该产品在设计开发过程中可能存在的风险以及规避风险的预案？

5）该产品在长期运维中可能存在哪些问题/有哪些注意事项？

**2、为沟通活动设计一系列动作，选定一个动作为其设计一个任务集**

1）设计沟通活动中的动作（假设是小型项目）：

在技术工作开始前和客户（及其他利益相关者）沟通，理解利益相关者项目目标，收集需求，定义软件特性和功能。

2）选定其中的“需求收集”动作，为其设计一个任务集：

制定项目的利益相关者列表；

邀请所有的利益相关者参加一个非正式会议；

征询每个人对于软件特征和功能的需求；

讨论需求，并确定最终的需求列表；

划定需求优先级；

标出不确定域。

**3、详细描述3个适合用于瀑布模型的软件项目**

1）小型的校园社团展示系统（web网页设计）：稳定的低风险项目，规模小，实现简单，易受控制，并且开发者对目标、环境非常熟悉。

2）航空航天软件项目（有固定功能的精密型项目）：需求在规划和设计阶段就已经确定，且项目开发周期内需求没有或极少变化，对需求变更进行严格控制。

3）企业外包的仓库管理系统（双方签署合同/说明保证）：有合同式的合作方式，开发者严格按照说明执行，客户需求明确且不参与软件实现过程。

**4、统一过程和UML是同一概念吗？**

不是同一概念。

统一过程可作为模型，提供一种“用例驱动、以体系结构为核心、迭代及增量”的软件过程框架供项目团队使用，它由UML方法和工具支持。

统一过程尝试从传统的软件过程中挖掘最好的特征和性质，但是以敏捷软件开发中许多最好的原则来实现；统一过程认识到与客户沟通以及从用户的角度描述系统（即用例）并保持该描述的一致性的重要性；它强调软件体系结构的重要作用，并帮助架构师专注于正确的目标，例如可理解性、对未来变更的可适应性以及复用；它建立了迭代的、增量的过程流，提供了演进的特征。

而UML统一建模语言，作为需求模型和设计模型的表示方式，提供了必要的技术支持和面向对象的软件工程实践，但它并不提供过程框架来指导项目团队。

UML包含了大量用于面向对象系统建模和开发的符号，已经成为面向对象软件开发的行业标准。

**5、为什么迭代过程更容易管理变更，是不是每一个敏捷过程都是迭代的？只用一次迭代就能完成项目的敏捷过程是否存在？**

1）迭代过程更容易管理变更，可以从两个方面理解：

第一，因为每个迭代周期都会根据软件开发计划，针对迭代目标，推进核心开发流程（需求、设计、编码、测试、部署），如果项目在需求、目标、范围上有变更，可以在下一次迭代开始时就采取措施，更利于变更的管理；第二，一次迭代的执行结果，往往成为调整项目估计和策划下次迭代的经验参照和依据，使得下次迭代计划更为精细与切实可行。

2）每个敏捷过程都是迭代的：

由于敏捷过程需要使用增量式开发策略，在很短的时间间隔内交付软件增量来适应变更的步伐，因此每一个敏捷过程都是迭代的。这种迭代方法允许客户周期性地评价软件增量，向软件项目组提出必要的反馈，影响为适应反馈而对过程进行的适应性修改。

3）存在只用一次迭代就能完成项目的敏捷过程：

比如只包含一次冲刺的Scrum模型，即只用一个增量原型开发周期就可以完成整个项目的待办事务，也就是说对于某个小型项目，设定这个冲刺的冲刺待定项恰好包含该产品团队选择的所有产品待定项，团队可以在一个冲刺周期中完成整个项目。

**6、描述XP中3个以上的重要概念。**

1）**现场客户**：是指始终在开发团队中的一位客户，他的主要工作是回答问题、编写验收测试、运行验收测试、指导迭代和接收版本；

2）**结对编程**：是指让两个人共同设计和开发代码的实践。结对者是全职合作者，轮流执行键入和监视，这样可以保证持续的设计和代码评审工作，有利于提高设计质量、更快地解决问题；

3）**重构**：是指在不改变代码外在行为的前提下对代码做出修改，以改进代码的内部结构。它是一种有纪律的、经过训练的、有条不紊的代码整理方法，可以将整理过程中不小心引入错误的可能性降到最低。从本质上看重构就是在代码写好后改进它的设计；

4）**测试驱动**：是指在编写某个具体功能的代码之前先编写测试代码，然后只编写使测试通过的功能代码，通过测试来推动整个开发的进行，有助于明确用户需求、编写高质量代码、并加速开发过程；

5）**持续集成**：是指在任何时候只要有一项任务完成，就集成新代码、构造系统并测试，是每日构建/每晚构建的一种极限形式。它的目标之一是让开发人员在第一时间了解到软件的错误，并迅速排除错误。

**7、是否每一个敏捷过程都可以用第3章所提及的通用框架活动来描述？建一张表，将通用活动和每个敏捷过程所定义的活动对应起来。**

软件工程的通用过程框架定义了5种基本的框架活动——沟通、策划、建模、构建以及部署。每个敏捷过程所定义的框架活动对应如下表所示：

|  |  |
| --- | --- |
| **敏捷过程** | **框架活动** |
| Scrum | 需求、分析、设计、演化、交付 |
| XP | 策划、设计、编码、测试 |
| 看板法 | 策划（待定/选定）、分析、开发、测试 |
| DevOps | 策划、编码、构造、测试、  集成、部署、运维、监控 |