**Chapter 1 软件工程概述**

1、软件的概念和特点。

**软件的概念：**

软件=程序（按事先设计的功能和性能需求执行的指令序列）

+数据（是程序能正常操作信息的数据结构）

+文档（与程序开发、维护和使用有关的图文材料）

**软件的特点：**

* 软件是开发的或者是工程化的，并不是制造的
* 软件生产是简单的拷贝
* 软件会多次修改
* 软件开发环境对产品影响较大
* 软件开发时间和工作量难以估计
* 软件的开发进度几乎没有客观的衡量标准
* 软件测试困难
* 软件不会磨损和老化（会退化）
* 软件维护易产生新的问题

2、软件危机、现状和产生的原因。

**软件危机：**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。（项目超出预算、项目超过计划完成时间、软件运行效率很低、软件质量差、软件通常不符合要求、项目难以管理并且代码难以维护、软件不能交付）

**现状：**软件成本日益增加、软件技术进步小于需求增长。

**原因：**

客观：软件本身特点（逻辑部件、规模庞大）。

主观：不正确的开发方法（忽视需求分析、错误认为软件开发=程序编写、轻视软件维护）。

3、软件工程的定义、三要素和发展过程。

**定义：**（IEEE）应用系统化的、学科化的、定量的方法，来开发、运行和维护软件，即将工程应用到软件。

**三要素：**方法、工具、过程。

**发展过程：**传统的软件工程→对象工程→过程工程→构件工程。（无作坊式阶段）

**Chapter 2 软件过程模型**

1、软件的生命周期、软件过程定义、能力成熟度模型CMM的概念。

**生命周期：**软件产品或软件系统从设计、投入使用到被淘汰的全过程。

**软件过程：**是在工作产品构建过程中，所需完成的工作活动、动作和任务的集合。

**CMM（Capability Maturity Model）：**能力成熟度模型，为评估软件组织的生产能力提供了标准、为提高软件组织的生产过程指明了方向。

初始级（有能力的人和个人英雄主义）→可重复级（基本项目管理）→已定义级（过程标准化）→量化管理级（量化管理）→优化级（持续的过程改进）。

**2、**常见的几种软件过程模型，比较各自优缺点。（含敏捷）

**瀑布模型：**第一个软件过程模型、经典生命周期模型、线性模型、推迟实现、以文档驱动、无迭代。

**优点：**每个阶段都有里程碑和可交付产品（指对应的文档，可执行程序在后期才能产生）、每个阶段结束前完成文档审查、及早改正错误。

**缺点：**增加工作量、早期错误发现晚、开发风险大、不适应需求变化。

**适用于：**系统需求明确且稳定、技术成熟、管理严格的场合。

**原型模型（是演化模型）：**一个部分开发的产品、抛弃原型or把原型发展成最终产品、快。

**优点：**减少需求不明确带来的风险。

**缺点：**采用的技术和工具不一定主流、连续修改导致质量低下、设计者在质量和原型中折中，客户意识不到一些质量问题。

**适用于：**客户只有总体目标但不清楚具体输入输出、开发者不确定算法效率等（开发者经验少）。

**增量模型（原型模型+迭代）：**满足用户需求的一个小功能（增量是一次发布增加一个功能，迭代是一次性发布全部功能，每次发布进行改进）、非整体开发、每个增量开发采用瀑布or快速原型。

**优点：**不需完整的需求、软件能尽早进入市场、人力资源不需要太多、可以看到中间产品、降低开发风险。

**缺点：**增量大小难确定、软件必须具备开放式结构、易退化成边做边改。

**适用于：**需求可能发生变化、风险大、希望尽早进入市场的项目。

**螺旋模型（原型模型+瀑布模型+演化模型）：**开发活动和风险管理结合起来控制风险、分为若干次迭代。

**优点：**原型的进化、及时调整决策、降低风险。

**缺点：**迭代效率不高导致推迟交付时间、要求开发人员经验丰富。

**适用于：**需求不明确or变化大的复杂系统。

**喷泉模型：**面向对象、早期定义对象、开发步骤多次迭代。

**优点：**各阶段无明显界限、同步开发、节省时间。

**缺点：**开发人员数量多、不利于管理、文档管理严格、审核难度大。

**基于构件的开发模型：**考虑核心是集成。

**优点：**软件复用、降低成本和风险、提高质量。

**缺点：**模型复杂、不能完全满足需求、无法控制系统演化。

**适用于：**系统之间有共性。

**统一过程模型：**面向对象、UML、动态视角、静态视角、实践视角。

**适用于：**大团队、大项目。

**敏捷开发：**极限编程等、基本原理（客户满意度、小而有激情、非正式）+开发准则（积极交流）

**优点：**快速响应变化和不确定、可持续开发速度、适应商业竞争。

**缺点：**通过测试但不是用户期望、重构困难。

**适用于：**商业竞争场合。

**选择软件过程模型：**

* 前期需求明确的情况下，尽量采用瀑布模型
* 用户无系统使用经验，需求分析人员技能不足的情况下，尽量借助原型模型
* 不确定因素很多，很多东西无法提前计划的情况下，尽量采用增量模型或螺旋模型
* 需求不稳定的情况下，尽量采用增量模型
* 资金和成本无法一次到位的情况下，可采用增量模型
* 对于完成多个独立功能开发的情况，可在需求分析阶段就进行功能并行，每个功能内部都尽量遵循瀑布模型
* 全新系统的开发必须在总体设计完成后再开始增量或并行
* 编码人员经验较少的情况下，尽量不要采用敏捷或迭代模型
* 增量、迭代和原型可以综合使用，但每一次增量或迭代都必须有明确的交付和出口原则

**Chapter 3 需求分析**

1、需求分析的概念和过程（4步）。

**概念：**确定系统必须具有的功能和性能，系统要求的运行环境，并且预测系统发展的前景。

**过程：**需求获取、需求提炼（建立分析模型）、需求描述（形成需求规格说明书）、需求验证（有效性、一致性、完备性、现实性检查）。

2、面向过程结构化分析方法（自顶向下分解），数据流图。

**数据模型：**ER图、数据字典（数据对象描述、加工规约、状态规约）；

**功能模型：**DFD；

**行为模型：**状态变迁图。

**数据流图：**顶层仅包含一个加工、数据流封闭在外部实体之间、不要标控制流与激发条件、每个加工至少一个输入和输出、主图必须包括四种图示（数据加工、数据源或终点、数据流、数据存储文件）

3、面向对象的分析方法（Booch、Yourdon、OMT、OOSE、UML），用例图。

**数据模型：**类图；

**功能模型：**用例图；

**行为模型：**活动图、时序图、状态图。

**用例图：**从用户的角度描述系统场景、关联（连接参与者和用例）、用例关系（include指向分解出的功能、extend指向基础、inheritance指向父亲、association指向消息接收方）。

4、面向过程、面向对象需求分析建模三类模型过程中（功能模型、数据模型、行为模型）各包含哪些内容。

（见上一题）

**Chapter 4 软件设计**

1、软件设计过程、软件设计的概念和原则。

**过程：**分析、设计、编码、测试、维护。

**概念：**（IEEE）软件系统或组件的架构、构件、接口和其他特性的定义过程及该过程的结果。

**原则：**

* 设计应该是一种架构
* 设计应该是模块化的
* 设计应该包含数据、体系结构、接口和组件各个方面
* 设计由软件需求分析过程中获得信息驱动，采用可重复使用的方法导出
* 设计应该采用正确清楚的表示法

2、与设计相关的8个概念：抽象、体系结构、设计模式、模块化、信息隐藏、功能独立、细化、重构。

**抽象：**忽略具体的信息将不同事物看成相同事物的过程。

**体系结构：**软件的整体结构和这种结构为系统提供概念上完整性的方式。

**设计模式**：在给定上下文环境中一类共同问题的共同解决方案。

**模块化：**软件被划分为命名和功能相对独立的多个组件（通常称为模块），通过这些组件的集成来满足问题的需求。

**信息隐藏：**模块定义和设计时应当保证模块内的信息（过程和数据）不可以被不需要这些信息的其他模块访问。

**功能独立：**每个模块只负责需求中特定的子功能，并且从程序结构的其他部分看，该模块具有简单的接口。

**细化：**逐步求精的过程。

**重构：**不改变组件功能和行为条件下，简化组件设计（或代码）的一种重组技术。

3、系统设计从体系结构、数据、接口和组件四方面进行设计。面向过程和面向对象的系统设计，各自包含哪些设计内容？

**面向过程：**系统结构图（变换、事务）、组件设计（图形（流程图、盒图）、表格（决策表）、语言（PDL））

**面向对象：**部署图、子系统、类图。

强内聚、低耦合

4、传统的系统设计方法，程序流程图。

5、面向对象的系统设计方法，顺序图。

对象、生命线、消息、激活。

**Chapter 5 程序实现**

1、个别编程规范。

2、程序三个结构是哪些？

顺序结构、条件/选择结构和循环结构。

**Chapter 6 软件质量保证**

1、质量保证相关概念（定义）。

**质量控制：**审查产品相关的各个方面质量的过程。

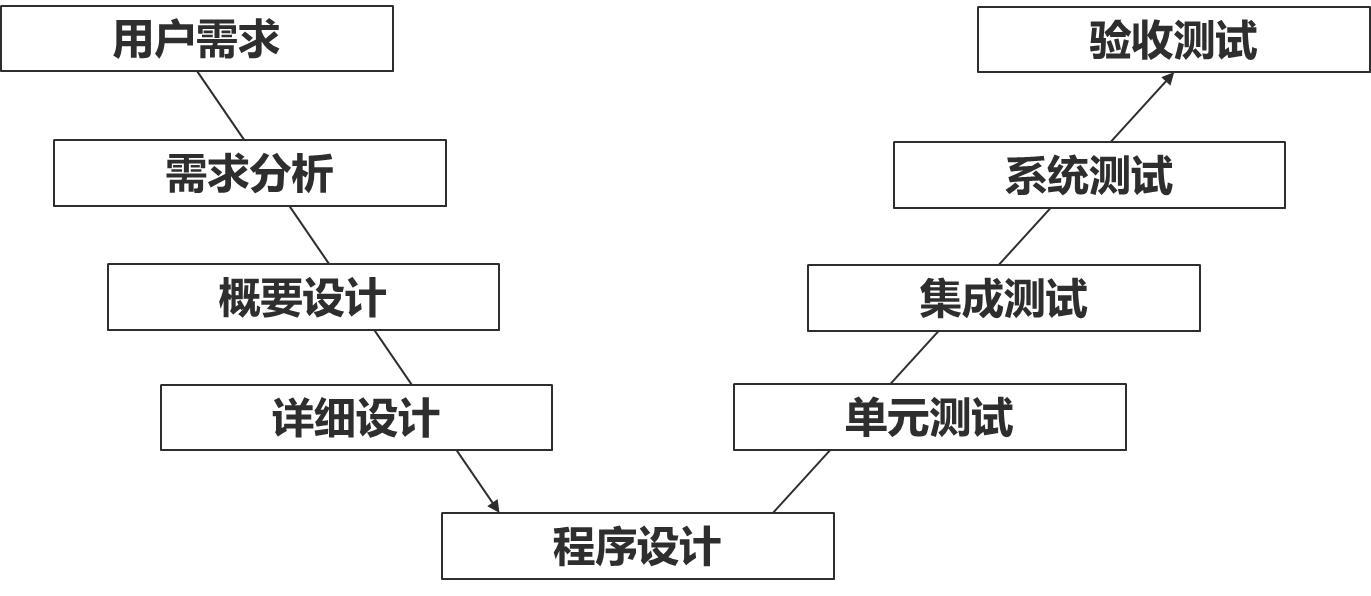
**质量保证**：系统监测和评估工程的各个方面，最大限度提高质量最低标准。

**质量成本：**追求质量过程或在履行质量有关活动中引起的费用以及质量不佳引起的下游费用等所有费用。

**软件质量：**明确表示是否符合功能和性能要求，明确地记载开发标准和所有专业开发软件的期望的隐性特点。

**软件质量保证：**遵照一定的软件生产标准、过程和步骤对软件质量进行评估的活动。

2、软件测试的策略（V模型）。



**V模型**非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程期间各阶段对应关系。

3、单元测试的内容、集成测试的分类、系统测试的分类、验收测试的分类。

**单元测试：**主要目的是验证软件模块是否按详细设计的规格说明正确运行。

方法：从程序的内部结构出发设计测试用例（一般白盒）。

**集成测试：**主要目的是检查多个模块间是否按概要设计说明的方式协同工作。

方法：自顶向下（桩模块）的集成方法、自底向上（驱动））的集成方法、SMOKE方法。

**系统测试：**主要目的是验证整个系统是否满足需求规格说明。

方法：黑盒测试。

**验收测试：**从用户的角度检查系统是否满足合同中定义的需求，以及以确认产品是否能符合业务上的需要。

方法：*α*测试（一个用户）、*β*测试（多个用户）。

4、回归测试的概念。

**回归测试：**指有选择地重新测试系统或其组件，以验证对软件的修改没有导致不希望出现的影响，以及系统或组件仍然符合其指定的需求。（尽量采用自动化测试）

5、测试技术常见术语的概念：软件缺陷、验证和确认、测试与质量保证、质量与可靠性、调试与测试、测试用例。

**软件缺陷：**（多实现、少实现都是缺陷，未按时完成不是缺陷）

* 软件未实现产品说明书要求的功能。
* 软件出现了产品说明书指明不能出现的错误。
* 软件实现了产品说明书未提到的功能。
* 软件未实现产品说明书虽未明确提及但应该实现的目标。
* 软件难以理解、不易使用、运行缓慢或者——从测试员的角度看——最终用户会认为不好。

**验证：**保证软件特定开发阶段的输出已经正确完整地实现了规格说明。

**确认：**对于每个测试级别，都要检查开发活动的输出是否满足具体的需求或与这些特定级别相关的需求。

**软件测试：**找出软件缺陷，并确保修复。

**软件质量保证：**创建、执行改进过程并防止缺陷的标准和方法。

**质量与可靠性：**可靠性、可维护性、可用性、效率、可移植性、功能性。

软件测试目标是发现缺陷；软件调试目标是定位与修复缺陷。

**测试用例**：是测试输入、执行条件、以及预期结果的集合，是为特定的目的开发的，例如执行特定的程序路径或验证与指定的需求相符合。

6、白盒测试、黑盒测试、静态分析各有哪些方法？

**白盒：（动态）**

**语句覆盖：**设计若干个测试用例，运行被测程序，使得每一可执行语句至少执行一次。

**分支（判定）覆盖：**设计若干个测试用例，运行被测程序，使得程序中每个判断的取真分支和取假分支至少经历一次。分支覆盖又称为判定覆盖。

**条件覆盖：**设计若干个测试用例，运行被测程序，使得程序中每个判断的每个条件的可能取值至少执行一次。

**条件组合覆盖：**设计足够的测试用例，运行被测程序，使得每个判断的所有可能的条件取值**组合**至少执行一次。

7、掌握逻辑覆盖与等价类划分测试方法。

**Chapter 7 软件维护**

1、软件维护的基本概念。

**定义：**（IEEE）软件维护是指由于软件产品出现问题或需要改进而对代码及相关文档的修改，其目的是对现有软件产品进行修改的同时保持其完整性。

2、理解软件维护的四个基本类型的含义。哪种占比重最大？哪种最小？

**纠错性维护：**识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误用。

**适应性维护：**使软件适应外部环境、数据环境的变化。

**完善性维护（占比最大）：**满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。

**预防性维护（占比最小）：**采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编制和测试。

3、可维护性的决定因素。

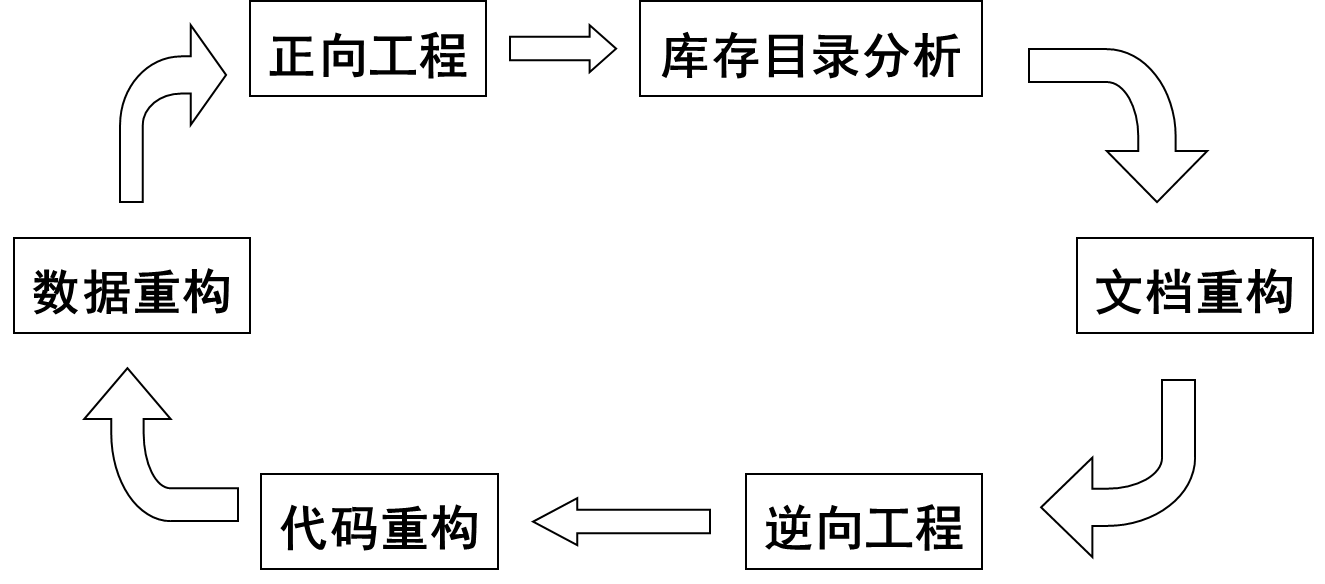
**决定因素：**软件可理解性、软件可测试性、软件可修改性、软件可移植性、软件可重用性。

**环境因素：**软件维护文档、软件运行环境、软件维护组织、软件维护质量。

4、软件维护过程模型、软件再工程、逆向工程的概念。

**过程模型：**分类与鉴别阶段→分析阶段→设计阶段→实现阶段→系统测试阶段→验收测试阶段→交付阶段。（IEEE定义的七个阶段）

**软件再工程：**对现有软件进行仔细审查和改造，对其进行重新构造，使之成为一个新的形式，同时包括随之产生的对新形式的实现。



**逆向工程：**分析程序以便在比源程序更高的抽象层次上创建出程序的某种描述的过程，也就是说，逆向工程是一个恢复设计结果的过程。

**主要内容：**数据的逆向工程、处理的逆向工程、用户界面的逆向工程、逆向工程的工具。（无需求分析逆向工程）

**Chapter 8 软件项目管理**

1、项目管理四要素（4P）：人员、产品、项目、过程（概念）

**人员：**招聘、选拔、绩效管理、培训、薪酬、职业发展、组织和工作设计、团队/文化的发展。

**产品：**策划一个项目以前，应当建立产品的目标和范围，考虑可选的解决方案。

**项目：**理解成功项目管理的关键因素，掌握项目计划、监控和控制的一般方法。

**过程：**软件过程提供一个框架，在此框架下可以制定项目开发的综合计划。

2、软件度量有哪些方法：生产率估计（基于规模（KLOC）、基于功能点（FP））、工作量度量（算法成本模型、COCOMO模型）。

**基于规模（千行代码KLOC）：**这些代码指的是源代码，通过源代码的行数来直观度量一个软件程序有多大规模。

* 生产率（PM）：PM = L/E，L表示代码总量（单位：KLOC），E表示软件工作量（单位：人月）。
* 每千行代码的平均成本（CKL）：CKL = S/L，S为软件项目总开销，L表示代码总量（单位：KLOC）。
* 代码出错率（EQRl）：EQRl = Ne/L，Ne表示代码出错的行数，L表示代码总量（单位：KLOC）。
* 文档与代码比（Dl）：Dl = Pd/L，Pd表示文档页数，L表示代码总量（单位：KLOC）。

**优点：**简单易行、自然直观。

**缺点：**依赖于程序设计语言的表达能力和功能、软件开发初期很难估算出最终软件的代码行数、对精巧的软件项目不合适。

**基于功能点（FP）：**FP = UFC×TCF = UFC × (0.65 + 0.01×ΣFi)

* UFC (Unadjusted Function Component)：未调整功能点计数，5个信息量的“加权和”
* TCF (Technical Complexity Factor)：技术复杂度因子
* Fi：14个因素的“复杂性调节值” (i =1..14)
* 0.65，0.01都是经验常数，现在由国际组织根据大量项目跟踪分析获得。
* 五个组件：ILF（软件内部的数据存储）、EIF（调用软件外部的接口）、EI（软件输入）、EO（软件输出）、EQ（输入输出综合查询）

**优点：**与程序设计语言无关, 在开发前就可以估算出软件项目的规模。

**缺点：**没有直接涉及算法的复杂度，不适合算法比较复杂的软件系统；

功能点计算主要靠经验公式，主观因素比较多。

**算法成本模型：**E=A+B×(ev)C

其中A、B、C是经验常数，E是工作量（人月），ev是估算变量（LOC或功能点）

E=5.2×(KLOC)0.91 Walston-Felix模型

E=5.5+0.73×(KLOC)1.16 Bailey-Basili模型

E=3.2×(KLOC)1.05 Boehm简单模型

E=5.288×(KLOC)1.047 Doty模型，用于KLOC>9的情况

E=-91.4+0.355FPAlbrecht和Gaffney模型

E=-37+0.96FP Kemerer模型

E=-12.88+0.405FP 小型项目回归模型

**COCOMO模型：**构造性成本模型（不是动态多变量！），Boehm于1981年提出，用于对软件开发项目的规模、成本、进度等方面进行估算；COCOMO模型是一个综合经验模型，模型中的参数取值来自于经验值，并且综合了诸多的因素、比较全面的估算模型。

基本模型：

基本COCOMO模型

E = a×(KLOC)b ; E是工作量(人月) ，a和b是经验常数

D = c×Ed ;D是开发时间(月) ，c和d是经验常数