1. **软件工程概述**

**1、软件：程序、数据、相关文档的完整集合**

程序：按事先设计的功能和性能要求编写的指令序列

数据：使程序能正常操纵信息的数据结构

文档：与程序开发、维护和使用有关的图文材料

**2、软件特点：**

1. 软件是逻辑产品，更多地带有个人智慧因素
2. 软件不会磨损
3. 成本高
4. 维护困难
5. 对硬件依赖性强
6. 对运行环境变化敏感
7. **软件发展历程：**

程序设计——程序系统——软件工程——软件产业

1. **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的一系列严重问题的现象

1. **软件工程：**

采用工程的概念、原理、技术和方法来开发、维护软件，把管理技术和开发技术有效结合，需要计算机科学和数学、工程科学、管理科学三方面支撑

1. **软件工程基本原则**
2. 严格管理
3. 阶段评估
4. 产品控制
5. 现代程序设计技术
6. 审查
7. 合理安排人员
8. 不断改进软件工程实践（技术、经验、工具）
9. **软件工程技术原则**

抽象，信息隐蔽，模块化，局部化，一致性，完备性，可验证性

1. **软件工程学：**

软件工程理论、软件工程方法学、软件工程管理

1. **软件工程方法学三要素：**

方法、工具、过程

1. **软件过程和模型**
2. **软件生存周期：从提出软件产品开始，到该产品被淘汰的全过程**

分为三个阶段：定义阶段（软件计划、需求分析），开发阶段（设计、编码、测试），维护阶段

1. **软件过程模型**

瀑布模型：线性模型，计划--需求分析--设计--编码--测试--运行维护 P15

原型模型：圆形，基本需求--开发初始系统--完善需求--修改完善系统 P16

螺旋模型：瀑布+原型，制订计划--风险分析--实施工程--用户评估 P18

增量模型：分段的线性模型 P20

敏捷模型：适用于需求很多，变化很大的情况

喷泉模型：主要用于支持面向对象开发过程

1. **统一过程模型RUP**

初始阶段（生命周期目标里程碑）——细化阶段（生命周期结构里程碑）——构造阶段（初始功能里程碑）——交付阶段（产品发布里程碑）

1. **RUP核心工作流**
2. 商业建模工作流
3. 需求工作流
4. 分析和设计工作流
5. 实现工作流
6. 测试工作流
7. 部署工作流
8. 配置和变更管理工作流
9. 项目管理平衡各种可能产生冲突的目标
10. 环境工作流
11. **UML基本构造块：**

事务、关系、图

事务：UML模型中的静态元素

1. 结构事务：类、接口、协作、用例、主动类、构件、节点
2. 行为事务：交互、状态机
3. 分组事务：包
4. 注释事务：注释

关系：反应类与类之间的联系的方法和性质，分为关联、依赖、泛化、实现、聚合

图：一组元素的表示，包含事务及其关系的组合

1. 静态结构：类图、对象图、用例图
2. 动态行为：时序图、协作图、状态图、活动图
3. 物理架构：构件图、部署图
4. **需求分析与建模**
5. **需求分析的任务：**

需求获取、需求建模、需求规格说明、需求评审（分析当前系统的物理模型，导出符合用户需求的目标系统的逻辑模型）

分为功能需求（系统必须提供的服务）和非功能需求（性能需求，系统必须满足的定时约束或容量约束）

1. **可行性分析：**

技术可行性、经济可行性、法律可行性、操作可行性

1. **逻辑模型：**

包括数据模型、功能模型、行为模型

1. **需求分析方法**
2. 结构化分析方法（SA）
3. 面向对象分析方法（OOA）
4. **需求分析主要过程 P41**
5. 获取需求以建立当前系统物理模型
6. 抽象出当前系统概念模型
7. 建立目标系统的逻辑模型
8. 对目标系统逻辑模型进行补充
9. 编写和评审需求分析文档
10. **结构化分析方法**

特点：自顶向下 逐层分解

1. **数据流图（DFD图）：**

画图步骤： P47

1. 确定源点、终点、输入输出数据流
2. 分解顶层的处理
3. 确定所用的数据存储
4. 用数据流把各个部分连接起来，形成连通数据流

分层DFD图画图原则： P48

先全局后局部，先整体后细节，先抽象后具体（顶层--中间层--底层）

1. **数据字典： P49**

数据流、数据元素、数据存储、变换处理、源点及终点

1. **数据建模：**

概念模型包含三种相互关联信息：实体、对象属性、对象间的关系

1. **实体—联系图（E-R图）**

详见 P59

1. **扩充E-R图（\*） P61**
2. **数据建模步骤：**
3. 标识和定义在建模问题范围内的实体
4. 标识实体和定义实体间基本联系
5. 主要包括开发属性池、定义属性、建立属性所有权、改善模型等
6. 定义主键和外键，以标识唯一的实体
7. **行为建模（\*） P62**
8. **状态转换图（STD图）**

主要符号：状态、变迁、事件 P64

1. **需求规格说明书（SRS）**
2. 引言：编写目的、项目背景、定义、参考资料
3. 项目概述：目标、运行环境、条件与限制
4. 数据描述：数据字典、数据采集
5. 功能需求：功能划分、功能描述
6. 性能需求：数据精度、时间特性、适用性
7. 运行需求：用户界面、硬件接口、软件接口、故障处理
8. 其他需求
9. **软件设计与编码**
10. **软件设计：把软件需求转换为软件表达的过程**

从技术观点看，分为：数据设计、体系结构设计、接口设计、过程设计

从软件工角度，分为：总体设计（概要设计）和详细设计

**2、软件设计主要方法：**

1. 模块化：按自顶向下把软件系统划分成若干可独立命名且独立访问模块，每个模块完成一个子功能

模块基本属性：接口、功能、状态、逻辑

1. 抽象：抽取出物质本质的共同特性而暂不考虑细节
2. 逐步求精：通过程序细节连续细化开发程序体系
3. 信息隐蔽：设计决定所刻画的模块特性对其他模块不可见
4. 体系结构

（6）设计模式

（7）重构

（8）功能独立性：高内聚低耦合，独立性较强

**3、耦合： P75**

不同模块间彼此联系程度的度量

（低——高）非直接耦合、数据耦合、标记耦合、控制耦合、外部耦合、公用耦合、内容耦合

**4、内聚： P75**

模块内各个元素彼此结合紧密程度

（低——高）偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚、功能内聚

**5、概要设计：将分析模型转换为设计模型**

体系结构设计、数据结构设计、数据库设计、接口设计、过程设计

**6、概要设计原则：**

1. 改进软件结构提高模块独立性
2. 模块规模应该适中
3. 深度、宽度、扇出、扇入都应适当（扇出大说明缺乏中间层次，扇入大好）
4. 模块作用域应在控制域内
5. 力争降低模块接口复杂程度
6. 设计单入口单出口模块
7. 模块功能应该可以预测

......

**7、概要设计常用图形工具**

1. 层次图 P88
2. HIPO图（层次图+输入/输出/处理图） P88-89
3. 结构图 传入模块、传出模块、变换模块、协调模块 P91

**8、结构化设计方法：**

适用于变换型结构和事务型结构的目标系统

**9、变换流：**

输入——变换中心——输出

1. 确定数据流图中变换中心、逻辑输入、逻辑输出
2. 设计软件结构顶层、第一层——变换结构
3. 设计中下层结构

**10、事务流：**

接受事务——事务分析——动作1、动作2......

1. 确定事务中心和加工路径
2. 设计顶层和第一层
3. 中下层模块的设计﹑优化工作与变换结构相同

**11、结构化设计过程及原则（\*） P95**

**12、详细设计：细化概要设计有关结果，做出软件的详细规格说明**

1. 数据结构设计
2. 数据库逻辑结构设计和物理设计
3. 算法过程设计
4. 信息编码设计
5. 测试用例设计
6. 其他设计
7. 编写“详细设计说明书”

**13、程序流程图（程序框图） P97**

顺序、分支、循环三种结构

**14、盒图（N-S图） P98**

顺序型、选择型、WHILE重复型、UNTIL重复型、多分支选择型

**15、问题分析图（PAD图） P99**

把程序过程控制结构表示成二维树，允许递归使用

1. **判定表、判定树 P100**

**17、过程设计语言（PDL）**

也称伪码

**18、人机界面设计原则（\*）**

用户掌控系统、减轻用户记忆负担、保持界面一致

**19、人机界面五种交互方式（\*）**

菜单选择、表单填写、命令语言、直接操作、自然语言

**20、面向数据结构设计方法（\*）**

Jackson方法 P107-112

Wamier方法 P113-114

**21、软件编码准则：**

保持简洁、模块化、简单化、结构化、文档化、格式化

**第五章 面向对象的分析与建模**

1. **面向对象方法学：**

优先考虑实体（问题论域的对象），认为对象是实体属性和实体行为的综合，整合了数据和处理，以数据或对象为主线

面向对象是使用对象、类、继承和多态等机制，而对象之间仅能通过传递消息实现相互通信

1. **面向对象方法学基本特点：**
2. 符合人们习惯的思维方式
3. 稳定性好
4. 可重用性好
5. 容易开发大型软件产品
6. 可维护性好
7. 易于测试和调试
8. **面向对象分析模式：**

功能模型、对象模型、动态模型

功能模型描述系统所能实现的功能，用用例图表示

对象模型描述系统静态结构，用UML类图表示

动态模型描述系统行为和操作，用UML交互图（时序图、状态图、活动图等）表示

1. **用例分析建立功能模型：**
2. 确定系统边界和参与者
3. 建立场景
4. 捕获用例
5. 定义用例间的关系
6. 绘制用例图 P135
7. **建立对象模型：**
8. 确定类和对象
9. 定义属性与服务
10. 定义关系和建立类图 （P142--143）
11. 完善对象模型
12. **建立动态模型：**
13. 细化用例场景
14. 建立序列图
15. 建立状态图
16. 完善动态模型

动态行为图包括用例图（P135）、顺序图/时序图（P146）、协作图（P147）、状态图（P148--150）、活动图（P151）

**第六章 面向对象的软件设计与实现**

1. **面向对象设计：**

利用面向对象技术，将用面向对象的分析所创建的分析模型转变为软件的设计模型

1. **面向对象设计模型：**

问题域部分、人机交互部分、任务管理部分、数据管理部分

每一部分设计都在对象层、属性层、服务层、关系层、包层五个层次进行，因此，每部分设计实际上都包括识别类和对象、定义属性、定义服务、识别关系、识别包五个活动

1. **面对对象的软件设计过程分为：**

系统设计（体系结构设计）和详细设计（子系统设计）

1. **系统设计的主要内容（\*） P156--157**
2. **系统顶层架构选择：**
3. 流程处理模式
4. 客户/服务器模式
5. 模型——视图——控制器模式（MVC）
6. 分层模式

采用UML包图设计顶层架构

1. **系统设计的分层模式：**

将整个软件系统分为若干层次，最顶层直接面向用户提供软件系统的操作界面，其余各层为紧邻其上的层次提供服务。

1. **详细设计的主要内容：**

对问题域部分、人机交互部分、任务管理部分、数据管理部分等四个部分进行具体的设计，主要是把系统设计中概念性的分析模型通过细化转换为可实现的具体模型。

1. **问题域部分设计步骤（\*） P162--163**
2. **任务的选择和调整策略（\*） P166**

常见任务有事件驱动型任务、时钟驱动型任务、优先任务、关键任务、协调任务

1. 识别任务类型
2. 审查和定义每个任务
3. **数据管理子系统设计**
4. 选择数据存储管理模式
5. 设计数据管理子系统
6. 对象模型向数据库关系模型的变换
7. **面向对象程序设计语言的优点（\*）**
8. 一致的表示方法
9. 可重用性
10. 可维护性
11. **面向对象程序设计语言的特点（\*）**
12. 支持类的定义、对象的静态声明或动态创建、属性和操作的定义、继承、聚合和管理的表示等
13. 提供类机制、封装机制和继承机制
14. 支持多态、多继承的表示和支持机制等高级特性
15. **UML类图中的关系模式**

泛化、实现、关联、聚合、依赖等

就耦合强度而言，依赖关系最弱，关联、聚合、组合更强，泛化最强（通过类的继承机制实现）

1. **软件测试技术**
2. **一些常用术语：**

错误（error）：编写代码时出现过错

缺陷（fault）：缺陷是错误的表现

失效（failure）：缺陷执行时发生失效

事故（incident）：出现失效时，可能会也可能不会呈现给用户

测试（test）：处理错误、缺陷、失效、事故

测试用例（test case）：有一个标识，与程序行为有关。**有一组输入和一组预期输出。**

1. **软件测试的原则： P186-187**
2. 软件测试伴随软件生命周期的全过程，而不是一个独立的阶段
3. 软件测试只能证明软件存在错误，而不能证明软件没有错误
4. “穷尽测试”是不可能的，必须在满足适当的标准的情况下终止测试

......

1. **软件测试方法和步骤**

分为静态测试和动态测试

静态测试通常不要求在计算机上实际执行所测程序，主要以一些人工模拟技术对软件进行分析和测试。静态测试包括代码审查和静态结构分析。

动态测试是通过一组预先按照一定测试准则构造的实例数据来动态运行程序，从而达到发现程序错误的过程。动态测试包括黑盒测试和白盒测试。

1. **静态测试**

最常见的静态测试是找出源代码语法错误，这类错误可由编译器自动完成

人工检测方法主要有代码检查法、静态结构分析法

1. 代码检查法：主要是桌面检查、代码走查和走查方式
2. 静态结构分析法：通过生成各种图表，来帮助对程序的静态分析；静态错误分析主要用于确定在源程序中是否有某类错误与或“危险”结构
3. **白盒测试**

又称结构测试或逻辑驱动测试，是以程序的内部逻辑结构为基础的一种测试技术。测试者完全知道程序的内部逻辑结构和处理过程，这种方法按照程序的内部结构测试程序，检测程序中的主要执行通路是否能按预定的要求正确工作。

白盒测试的动态测试技术主要包括程序插桩、逻辑覆盖、基本路径测试。

1. **逻辑覆盖：**

语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖等

1. **基本路径测试：**
2. 根据源代码导出程序的控制流图G
3. 计算程序控制流图G的环形复杂度V（G）

V（G）=E-N+2，E为边数，N为节点数；V（G）=P+1，P为判定节点数

1. 确定线性独立路径的基本集合
2. 准备测试用例，强制执行基本路径集合中的每条路径
3. **黑盒测试**

又称功能测试或数据驱动测试，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口处进行测试，只通过输入数据、进行操作、观察输出结果，检查软件系统是否按照需求规格说明书的规定正常运行，软件能否适当地接收输入数据并产生正确的输出信息，并且保持外部信息（如数据库和文件）的完整性

黑盒测试一般分为功能测试和非功能测试，功能测试主要包括：等价类划分、边界值分析、因果图、错误推测法、判定表驱动法等。

1. **等价类划分法**

包括有效等价类和无效等价类两种，划分等价类的方法有：按区间、按数值、按数值集合、按限制条件、按限制规则、按处理方式等

构造测试用例：

1. 设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类，重复这一步，直到所有有效等价类均被测试用例覆盖
2. 设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步使所有无效等价类被覆盖
3. **边界值分析法：**

可作为等价类划分法的一种有益补充，使等价类的每个边界都作为测试条件。

1. **因果图法： P204**

当需要测试的程序输入数据间存在相互制约关系，或存在输入情况的不同组合时

1. 分析程序规格说明，找出哪些是原因哪些是结果
2. 根据程序规格说明，找到原因和结果之间的关联，并用因果图符号描绘
3. 在因果图上标出原因与原因、结果与结果之间需要满足的约束条件
4. 把因果图转换成判定表
5. 根据判定表设计测试用例
6. **错误推测法：**

列举出程序中所有可能存在的错误和容易发生错误的特殊情况，根据它们选择测试用例

1. **软件测试过程：**

要经历单元测试、集成测试、验收测试（确认测试）、系统测试四个阶段，前三个是用来测试软件内部的，系统测试是测试软件与外部环境的接口与通信。

1. **单元测试**

在软件编码完成后，对编写的程序模块进行的测试，又称为“模块测试”。目的在于检查每个程序单元模块是否能正确实现详细设计中说明的模块功能、性能、接口和设计约束要求，发现各模块内部可能存在的错误

单元测试主要使用黑盒测试技术，必要时辅以白盒测试

1. **单元测试的内容：**
2. 模块接口测试
3. 局部数据结构测试
4. 基本路径测试
5. 输入/输出测试
6. 出错处理测试
7. 边界条件测试
8. **单元测试的环境**

动态测试过程中，需要为每个单元测试开发一个驱动模块和一个或多个桩模块。

1. 驱动模块，用来模拟所测模块的上一级模块，相当于一个接受测试数据，并把数据传送给所测模块，然后打印相关结果的“主程序”
2. 桩模块，又称“存根程序”，用来代替被所测模块所调用的子模块
3. **集成测试**

在单元测试的基础上，将模块按照总体设计时的要求组装成子系统或整个系统进行测试，因此集成测试又称组装测试。

模块与模块集成构成子系统，子系统与子系统集成构成整个软件系统，一般由专门的测试小组来进行，通常采用黑盒测试。

1. 非增量式集成，又称一次性集成，首先将各模块作为单个实体进行测试，然后将所有已测试好的模块一次性组合到被测系统中，组成最终系统进行测试。
2. 增量式集成，按不同顺序依次向被测系统加入新的模块，进行集成测试，新模块可以用来取代被测系统在之前的测试中使用的相应的驱动模块或桩模块。

增量式集成测试三种不同测量：自底向上集成、自顶向下集成、混合式集成

1. **系统测试**

将已经集成好的软件系统，作为计算机系统的一个元素，与计算机系统的其他元素结合在一起，在真实运行环境下进行测试。

系统测试包括：功能测试、强度测试、兼容性测试、可用性测试、安全性测试、性能测试、恢复性测试等。

1. **验收测试（确认测试）**

在整个系统集成测试完毕后，通过检验和提供客观证据证实软件是否满足预定的预期用途的需求而进行的测试。其目的是验证与证实软件是否满足软件需求规格说明书中规定的要求。

验收测试一般包括有效性测试和软件配置复查，验收测试一般由独立的第三方测试机构进行。

1. **α测试与β测试**
2. α测试是由一个用户在开发环境下进行的测试，也可以是开发机构内部的用户在模拟实际操作的环节下进行的测试
3. β测试是由软件的多个用户在一个或多个用户的实际使用环境下进行的测试。测试时，开发者通常不在测试的线程，因此β测试是在开发者无法控制的环节下进行的软件现场应用。

**21、面向对象的软件测试（\*） P221**

1. **软件配置与软件维护**
2. **软件配置活动**

软件配置是一个软件在其生命周期内产生的各种版本必备的文档、程序、数据、标准和规约等信息的总称。

1. **主要软件配置管理活动：**

（1）标识配置项

（2）管理变更请求

（3）管理基线和发布活动

1. **基线：**

为了控制软件产品的变化，使其保持一定程度的稳定，并成为继续发展的一个固定基础。将已经正式通过审核和批准的规约或产品，称之为基线，作为进一步开发的基础。

1. **版本管理要解决的问题（\*） P239**
2. **现代版本管理内容：（\*）**

并发开发支持管理、并发变更管理、现代的工作空间管理、现代的构建和发布管理。

1. **建立基线三大原因：**

重现性、可追踪性、报告

五大优点（\*） P243

1. **软件维护方法：**

改正性维护（纠错性维护）、适应性维护、完善性维护、预防性维护

1. **软件维护成本：**

M=P+K\*exp（c-d）

M：维护中消耗的总工作量

P：生产性工作量

K：经验常数

c：复杂程度

d：维护人员对软件的熟悉程度

1. **软件的可维护性**

维护人员理解、修改、改动或改进软件的难易程度

1. **影响可维护性的因素：**
2. 可理解性：读者理解软件结构、功能、接口和内部处理过程的难易程度
3. 可测试性：验证程序正确性的度量程度
4. 可修改性：软件可修改的难易程度
5. 可移植性：系统对硬件和软件环境的兼容程度
6. 可重用性

软件重用有三种：知识重用、方法和标准重用、软件成分重用。

1. **软件可维护性的定量度量（\*） P259**
2. **提高可维护性的方法：（\*）**
3. 建立明确的软件质量目标和优先级
4. 使用提高软件质量的技术和工具
5. 选择可维护性好的程序设计语言
6. 改进程序文档
7. 保证质量审查
8. **质量保证与软件工程标准化**
9. **软件质量：**

软件产品或服务特性的整体

软件质量特性是用以描述和评价软件产品质量的一组属性

1. **软件质量特征指标**
2. 功能特征
3. 可靠性特征
4. 易用性特征
5. 效率特征
6. 可维护性特征
7. 可移植性特征
8. **ISO软件质量评价模型**

按照ISO/TC97/SC7/WG3/1985-1-30/N382，软件度量模型分为3层

1. 高层，软件质量需求评价准则（SQRC），8个元素
2. 中层，软件质量设计评价准则（SQDC），21个元素
3. 低层，软件质量度量评价准则（SQMC），具体各单位自行制定
4. **McCall质量评价模型**

基于11个特性之上，正确性、可靠性、效率、完整性、可使用性、可维护性、可测试性、灵活性、可移植性、可复用性、互联性

Mij代表第i个打分员为第j个准则加的权

设wij=Mij/(Mi1+Mi2+...+MiL) （wij为第i个打分员第j个准则除以所有准则的和）

Wj=1/n\*(w1j+w2j+...+wLj) （Wj为第j个准则所有打分员的和/n）

L个准则对子系统得分分别为a1,a2,...,aL，则子系统得分为

Fq=a1\*W1+a2\*W2+...+aL\*WL

1. **Boehm质量评价模型**

给出60个质量度量公式，任务主要从三方面评价：可使用性、可维护性、可移植性

1. **质量控制（\*）**

SQC，保证每一件工作产品都满足对它的需求而应用于整个开发周期中的一系列审查、评审和测试

1. **质量保证（\*）**

SQA，保证产品和服务充分满足用户要求的质量而进行的有计划、有组织、有系统的活动

1. **质量保证体系图（\*） P275**

纵向开发阶段（全过程），横向组织机构（全员）

1. **软件可靠性：**

软件系统在规定的时间间隔内，按照规定的条件，完成规定功能而不发生故障的概率。在这个定义中包含的随机变量是“时间间隔”。

显然随着运行时间的增加，运行时遇到程序故障的概率也将增加，即可靠性随着时间间隔的加大而减小。

1. **可靠性评价指标**
2. 可用性
3. 初期故障率
4. 偶然故障率
5. 平均失效前时间（MTTF）
6. 平均失效间隔时间（MTBF）
7. 缺陷密度（FD）
8. 平均失效恢复时间（MTTR）
9. 平均不工作时间（MTBD）
10. 平均操作错误时间（MTBHE）
11. 软件系统不工作时间均值（MDT）
12. 初始错误个数（NC）
13. 剩余错误个数（ND）
14. **系统稳态可用性计算**

稳态可用性A=MTTF/（MTTF+MTTR）\*100%

MTBF=MTTF+MTTR，MTBF越大，无故障工作时间越长，越稳定可靠

1. **平均无故障运行时间估算**

Et：测试前程序中缺陷总数

It：机器指令总数衡量的程序长度

τ：测试（调试）时间

Ed（τ）：在0~τ时间内发现的错误总数

Ec（τ）：在0~τ时间内改正的错误总数

Eτ：在0~τ时间内剩余的缺陷数

∴ 假设发现的缺陷都及时改正 Ed（τ）=Ec（τ）

剩余的缺陷数 Eτ=Et-Ec（τ）

单位长度程序中剩余缺陷数 Et/It-Ec（τ）/It

∴ 软件的平均无故障时间和单位长度中剩余的故障数成反比

MTTF=1/K（Et/It-Ec（τ）/It）

1. **植入故障法估计原有总故障Et**

人为植入Ns个故障，一段时间测试后发现ns个植入的故障，n个原有的故障

∴ Et≈N=（n/ns）\*Ns

1. **分别测试法估计原有总故障Et**

两个测试人员同时测试一个软件程序的两个副部，T表示测试时间

T=0时，故障总数B0

T=T1时，测试员甲发现故障数B1

T=T2时，测试员乙发现故障数B2

T=T1时，甲乙发现的相同故障数为Bc；

∴ B0=B2\*B1/Bc 每隔一段时间并行测试，取B0平均值为Et估算值

1. **软件复杂性：**

指程序的结构性、模块性、简明性、简洁性和可理解性的程度。

程序复杂性度量的方法主要有三种：代码行度量法、McCabe度量法、Halstead的软件科学。

1. **代码行度量法：**

用程序代码行的多少来衡量程序的复杂性。

基本考虑是统计一个程序模块的源代码行数目，并以源代码行数做为程序复杂性的度量。

1. **McCabe度量法**

又称环路复杂性度量，是一种基于程序控制流的复杂性度量方法。

计算环路复杂性的方法：在一个有向图G（程序图）中，环路的个数V(G)=m-n+2。

其中，V(G)是有向图G中环路个数，m是图G中弧数，n是图G中结点数。

1. 环路复杂度取决于程序控制结构的复杂度。
2. 环路复杂度是可加的。
3. 对于复杂度超过10的程序，应分成几个小程序，以减少程序中的错误 。
4. 错误与程序的判定加上例行子程序的调用数目成正比。
5. **Halstead的软件科学**

根据程序中可执行代码行的操作符和操作数的数量来计算程序的复杂性

H：程序长度

V：程序容量，即程序所需信息（位）的容量

B：程序潜在错误数量

N1：程序中不同运算符的个数

N2：程序中不同运算对象的个数

预测的Halstead长度 H＝N1·log2N1 +N2·log2N2

程序容量 V＝H·log2(N1+N2)

程序潜在错误数量 B=H·log2（N1+N2）/3000=V/3000

1. **提高软件可靠性的方法：**

提高和保证软件可靠性应贯穿整个软件生命周期，落实到软件生命周期中各阶段的每个环节。

（1）建立以可靠性为核心的质量管理体系；

（2）选择合适的软件开发方法；

（3）软件重用；

（4）使用开发管理工具；

（5）加强测试；

（6）引入模型化技术和容错设计技术。

1. **软件工程标准化**

包括程序设计语言标准化和软件过程标准化

1. **软件工程标准化的层次**

分为五个级别，即国际标准（ISO）、国家标准（GB）、行业标准（IEEE）、企业(机构)标准及项目（课题）标准。

1. **我国软件工程标准化工作（\*）**

（1）基础标准：GB/T 11457-89 软件工程术语

GB 1526-891（ISO 5807-1985）信息处理--数据流程图、程序流程图、系统结构图、程序网络图和系统资源图的文件编制符号及约定等；

（2）开发标准：GB 8566-88 软件开发规范等；

（3）文档标准：GB 8567-88 计算机软件产品开发文件编制指南等；

（4）管理标准：GB 12504-90 计算机软件质量保证计划规范等。

1. **ISO 9000标准（\*）**

ISO 9000族标准源自英国标准BS5750。国际标准化组织(ISO)于1987年产生了首版IS09000族标准，即ISO 9000：1987系列标准。

它包括: ISO8402《质量——术语》、ISO9000《质量管理和质量保证标准——选择和使用指南》、ISO9001《质量体系——设计开发、生产、安装和服务的质量保证模式》、ISO9002《质量体系——生产和安装的质量保证模式》、ISO9003《质量体系——最终检验和试验的质量保证模式》、ISO9004《质量管理和质量体系要素——指南》等6项国际标准，通称为ISO9000系列标准。

ISO 9000系列标准的主体部分可以分为两组：

1) “需方对供方要求质量保证”的标准——9001～9003

2) “供方建立质量保证体系”的标准——9004

1. **质量认证（\*）**

质量体系认证则是通过第三方机构，依据规定程序对提供产品，服务单位的质量管理出具书面保证(ISO质量管理体系认证合格证书)，证明其符合ISO 9000标准规定要求所做出的评价。

1. **质量管理体系认证的好处（\*） P286**
2. **推行ISO 9000的过程（\*） P286-287**

知识准备——立法——宣贯——执行——监督、改进

1. **申请质量认证应具备的条件（\*） P287**

**28、软件能力成熟度模型CMM（\*）**

是一种用于评价软件承包能力并帮助其改善软件质量的方法，也就是评估软件能力与成熟度的一套标准，它侧重于软件开发过程的管理及工程能力的提高与评估。

CMM把软件过程从无序到有序的进化过程分成5个阶段，并把这些阶段排序，形成5个逐层提高的等级。

初始级——可重复级——已定义级——已管理级——优化级

1. **软件项目管理**
2. **软件项目管理：**

为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对人员（People）、产品（Product）、过程（Process）进行分析和管理的活动。

1. **软件项目管理内容：**

主要任务包括计划、组织、指导、控制

1. 风险管理

（2）人员的组织与管理

（3）软件度量

（4）软件项目计划管理

（5）质量保证

（6）软件过程能力评估

（7）软件配置管理

1. **项目组织结构**

参与软件项目的人员可以分为五类。

1. 高级管理者
2. 项目(技术)管理者
3. 开发人员
4. 用户代表
5. 辅助人员
6. **人员配置（\*） P307**
7. **软件项目估算**
8. 对软件规模进行估算；
9. 估算软件项目所需的工作量；
10. 估算项目的时间（进度）；
11. 估算项目成本。
12. **代码行技术（LOC）**

以KLOC（千代码行）为计量单位

1. 多人分别估计最小行数a、最大行数b、最有可能行数m
2. 计算平均值aa、bb、mm
3. 求出估计的代码行数L=（aa+4mm+bb）/6
4. 估算工作量=代码总估算长度/估算生产率
5. 估算总成本=月平均薪水X估算工作量
6. 生产率=KLOC/PM（人月）
7. 成本=元/LOC
8. 质量=错误数/KLOC
9. 文档=文档页数/KLOC
10. **功能点技术**

用系统的功能数量来测量其规模，它以一个标准的单位来度量软件产品的功能，与实现产品所使用的语言和技术没有关系，通常在软件系统设计初期进行估算。

外部输入EI，外部输出EO，用户查询EQ，内部逻辑文件ILF，外部接口文件EIF。

1. 未调整功能点数UFP=a1xEI+a2xEO+a3xEQ+a4xILF+a5xEIF
2. 计算技术复杂因子TCF=0.65+0.01xDI （14项因子之和DI=F1+F2+...+F14）
3. 计算功能点数FP=UFPxTCF
4. 估算工作量=项目估算FP/估算生产率（由经验得到）
5. 估算总成本=平均月工资x估算工作量
6. **对象点技术（\*） P311**
7. **IBM模型**

工作量E=5.2\*L0.91 L为源代码行数，以KLOC计，E是工作量（以PM计）

项目持续时间D=4.1\*L0.36 D以月计

人员需要量S=0.54\*E0.6 S以人计

文档数量DOC=49\*L1.01 DOC以页计

1. **Putnam模型**

L——源代码行数（以LOC计）

K——整个开发过程所花费的工作量（以人/年计）

td——开发持续时间（以年计）

Ck——技术状态常数，反映“妨碍开发进展的限制”

L=Ck\*K1/3\*td4/3

K=L3/(Ck3\*td4)

td=[L3/(Ck3\*K)]1/4

1. **COCOMO模型**

静态、单变量，是一种精确、易于使用的成本估算方法

1. 基本COCOMO模型

MM--------表示开发工作量，度量单位为人月。

TDEV--------表示开发进度，度量单位为月。

DSI--------源指令条数，即代码或源程序行数。若一行有两个语句，则算做一条指令。不包括注释语句。

KDSI＝1000DSI。

工作量 MM=a x (KDSI)b

进度 TDEV=c x (MM)d

1. 中间COCOMO模型

工作量 MM=a x (KDSI)b x EAF (工作量调节因子EAF为15个要素乘积EAF=F1\*F2\*...\*F15)

1. 详细COCOMO模型

用于估算独立的软部件，它包含中间模型的所有特性，并详细对工作量调节因子在软件过程中每个步骤的影响做出评估。

1. **软件文档（\*）**

软件文档(document)也称文件，通常指的是一些记录的数据和数据媒体，它具有固定不变的形式，可被人和计算机阅读。它和计算机程序共同构成了能完成特定功能的计算机软件。

1. **编制软件文档目的：（\*）**
2. 作为开发人员在一定阶段工作中的结束标志；
3. 向管理人员提供开发过程的进展状态；
4. 详细记录开发过程的技术信息，为以后的测试、调试、维护等工作提供依据；
5. 提供软件运行、维护、培训的信息，便于管理人员、开发人员和用户之间的信息沟通。
6. **文档编制内容（\*）**

分为用户文档、技术文档、维护文档

14种基本文档 P330

1. **提高文档编制的质量：（\*）**
2. 针对性，2）精确性 ，3）清晰性，4）完整性，5）灵活性，6）可追溯性
3. **文档的管理和维护：（\*） P330-331**
4. **工作分解结构（WBS）技术：**

WBS是一个分级的树型结构，是一个对项目工作由粗到细的分解过程。主要是将一个项目分解成易于管理的几个部分或几个细目，以便确保找出完成项目工作范围所需的所有工作要素。在工作分解的基础上通过对各项任务进行编码，把项目的所有要素在一个共同的基础（WBS）上建立关联，在此基础上建立各管理过程的所有信息沟通。

WBS 图 P319

1. **进度计划（进度安排）**
2. 横道图（甘特图）：只列出一些关键任务，分析并行情况，绘制成时间表 P320
3. 里程碑图：标识项目特殊事件或关键点，是项目工作中的主要检查控制点，是完成阶段性工作的标志 P320
4. 关键路径网络计划法：

利用网络图表示计划任务的进度安排及其中各项工作之间的相互关系，在此基础上进行网络分析，计算网络时间，确定关键活动和关键路径，并利用时差，不断改善CPM网络计划，求得工期、资源与成本的优化方案。

网络图由箭线、节点和由节点与矢线连成的通路组成。

网络图绘制7项原则 P321

工序时间T=（a+4m+b）/6 最有利时间a，正常时间m，最不利时间b

节点最早开始时间TE

节点最迟结束时间TL

节点时差Si=TL-TE

工序最早开始时间TES TES(i,j)=TEi

工序最早结束时间TEF TEF(i,j)=TES(i,j)+tij

工序最迟结束时间TLF TLF(i,j)=TLj

工序最迟开始时间TLS TLS(i,j)=TLF(i,j)-tij

时差：在不影响整个工程项目完工时间条件下，某项工序的最迟开始时间与最早开始时间之差，或最迟结束时间与最早结束时间之差

总时差：工序的最迟开始时间和最早开始时间之差

关键路径：总时差为0的工序，项目工期=关键路径长度

通路时差：关键路径和非关键路径时间之差，等于线路上各工序单时差之和 P323

1. 计划评审技术（\*） P325