**第1章**

软件工程: 是应用计算机科学、数学及管理科学等原理，开发软件的工程。

软件分类：系统软件（居于计算机系统中最靠近硬件的一层，其他软件一般都通过系统软件发挥作用）； 支撑软件（是支撑软件的开发、维护和运行的软件）； 应用软件（是特定应用领域专用的软件）。

软件生存周期的6个阶段及其任务

指软件产品或软件系统从产生、投入使用到被淘汰的全过程。

1. 计算机系统工程。选择出可行的方案

2、需求分析。解决待开发软件要“做什么”的问题。确定软件的功能、性能、数据、界面等要求，生成软件需求规约。

3、设计。解决待开发软件“怎么做”的问题。软件设计通常可分为系统设计和详细设计。

4、编码。将设计的结果转化为可执行的程序代码。

5、测试。发现并纠正软件中的错误和缺陷。测试主要包括单元测试、集成测试、确认测试和系统测试。

6、运行和维护。发现错误时修改，或者增添新功能。

5.软件过程模型：瀑布模型、演化模型、增量模型、原型模型、螺旋模型、喷泉模型、基于构件的开发模型、形式化方法模型。

瀑布模型：它给出了软件生存周期活动的固定顺序，上以阶段的活动完成后想下一阶段的活动过渡，最终得到开发产品。

优缺点：瀑布模型、快速原型、螺旋模型(比较三者的优缺点)

瀑布模型：优点是可强迫开发人员采用规范的方法，严格地规定了每个阶段必须提交的文档，要求每个阶段交出的所有产品都必须经过质量保证小组的仔细检查；缺点是由文档驱动。

快速原型：优点是有助于保证用户的真实需要得到满足。

螺旋模型：优点是对可选方案和约束条件的强调有利于已有软件的重用，减少了过多测试或测试不足所带来的风险，维护只是模型的另一个周期，在维护和开发之间并没有本质的区别；缺点是它靠风险驱动。

演化：对软件需求缺乏准确的认识时；

增量：需求经常发生变化

原型：能快速低成本的构建原型

喷泉：支持面向对象的开发

6.能力成熟度模型

CMM模型： 初始级（有纪律的过程）可重复级（标准、一致的过程）已定义级（可预测的过程）已管理级（持续改进的过程）优化级

（1）初始级：软件过程的特点是无秩序的，甚至是混乱的。  
（2)可重复级：建立了基本的项目管理过程来跟踪成本、进度和功能特性。

（3）已定义级：已将管理和工程活动两方面的软件过程文档化、标准化，并综合成该组织的标准软件过程。  
（4）已管理级：收集对软住过程和产品质量的详细度量值，对软件过程和产品都有定量的理解和控制。  
（5）优化级：过程的量化反馈和先进的新思想、新技术促使过程不断改进。

**第2章 系统工程**

1.系统工程的任务（照片）

1、识别用户的要求

2、系统建模和模拟

3、成本估算及进度安排

4、可行性分析

5、生成系统规格说明

2.可行性分析三方面

从经济、技术、法律等分析所给的解决方案是否可行，能否在规定的资源和时间的约束下完成。

1、技术可行性。只要根据系统的功能、性能、约束条件等，分析在现有资源和技术条件下系统能否实现。

2、经济可行性。经济可行性主要进行成本效益分析，从经济角度，确定系统是否值得开发。

3、法律可行性。主要研究系统开发过程中可能涉及到的合同、侵权、责任以及各种与法律相抵触的问题。

4. 方案的选择跟折衷：要在多个可行的实现方案中作出选择。

**第3章 需求工程**

1.需求分析的任务

1、确定对系统的综合要求：功能需求，性能需求，可靠性和可用性需求，出错处理需求，接口需求，约束，逆向需求，将来可能提出的要求；

2、分析系统的数据要求；

3、导出系统的逻辑模型；

4、修正系统开发计划。

**第4章 设计工程**

1.软件设计的任务

1、数据/类设计

数据/类设计将分析类模型变换成类的实现和软件实现所需要的数据结构。

2、体系结构设计

体系结构设计定义了软件的整体结构，由软件部分、外部可见的属性和他们之间的关系组成。

1. 接口设计

描述了软件内部、软件和协作系统之间以及软包同人之间的通信方式。

4、软件级设计

部件级设计将软件体系结构的结构性元素变换为对软件部件的过程性描述。

1. 软件设计原则

抽象与逐步求精；模块化；信息隐藏；功能独立：内聚、耦合、扇入扇出

1、抽象：是在软件设计的规模逐渐增大的情况下，控制复杂性的基本策略。抽象的过程是从特殊到一般的过程，上层概念是下层概念的抽象，下层概念是上层概念的精化和细化。

2、模块化：即把软件按照规定原则，划分为一个个较小的，相互独立的但又相互关联的部件。模块化实际上是系统分解和抽象的过程。

3.内聚：是一个模块内部各个元素彼此结合的紧密程度的度量。

4.耦合：是模块之间的相对独立性（相互连接的紧密程度）的度量。

软件体系结构的风格：数据为中心、数据流风格、调用和返回风格、面向对象风格、层次式风格的体系结构

**第5章 结构化分析与设计**

1.结构化分析的过程

①理解当前的现实环境，获得当前系统的具体模型（物理模型）。  
②从当前系统的具体模型抽象出当前系统的逻辑模型  
③分析目标系统与当前系统逻辑上的差别，建立目标系统的逻辑模型。  
④为目标系统的逻辑模型作补充。

2.结构图的几个概念

1、深度：深度是指程序结构图中控制的层数

2、宽度：宽度是指程序结构图中同一层次上模块总数的最大值

3、扇出：一个模块的扇出是指该模块直接调用的模块数目

4、扇入：一个模块的扇入是指能直接调用该模块的模块数目。

3.启发式设计策略

1、改造程序结构图，降低耦合度，提高内聚度。

2、避免高扇出，并随着深度的增加，力求高扇入。

3、模块的影响范围应限制在该模块的控制范围内

4、降低模块接口的复杂程度和冗余程度，提高一致性

5、模块的功能应是可预测的，避免对模块施加过多的限制。

6、尽可能设计单入口和单出口的模块。

**第7章 面向对象方法基础**

1.对象：是指一组属性以及这组属性上的专用操作的封装体。

2.类：是一组具有相同属性和相同操作的对象的集合。也就是说，对象是客观世界中的实体，而类是同一类实体的抽象描述。

3.继承：是类间的一种基本关系，是基于层次关系的不同类共享属性和操作的一性机制。

4.多态性：是指同一个操作作用于不同的对象上可以有不同的解释，并产生不同的执行结果，相同操作的消息发送给不同的对象时，每个对象将根据自己所属类中定义的这个操作去执行，从面产生不同的结果。

1. 消息：对象间传递的手段

**第9章 基于构件的软件开发**

1.构件的定义：某系统中有价值，几乎独立并可替换的一部分，在良好定义的体系结构语境内满足某种清晰的功能。

2.构件的要素：规格说明、一个或多个实现、受约束的构件标准、包装方法、部署方法

3.构件描述模型：3C模型、REBOOT模型

4.常用的构件标准：CORBA、COM/DCOM、EJB

5.构件分类模式：枚举分类、刻面分类、属性-值分类

6.构件库管理系统的功能

1、构件的分类存储：根据构件库的分类模型将入库的构件储存在构件库中，并保存构件的描述信息。  
2、构件检索：从构件库中检索出满足用户要求或接近用户要求的构件。  
3、构件库浏览：浏览库中的全部或部分构件。  
4、删除构件：将不再使用的构件从构件库中删去。  
5、构件使用情况评价：根据用户使用和检索构件的反馈意见，对构件作出评价，为进一步的改进提供依据

7.构件检索方法

(1)规约匹配  
基于有序的谓词逻辑的匹配，通过谓词演算公式进行精确匹配，通过逻辑连接符和逻辑量词进行部分精确匹配。  
(2）型构(signature)匹配  
通过接口的定义进行匹配，适用于函数之类的构件。

（3）术语轮廓匹配  
基于构件编目描述语言的匹配，将每一个构件的编目描述作为该构件的一个特征矢量，通过测算矢量的距离进行匹配。  
（4）行为采样  
基于构件测试的匹配，根据测试结果相同的概率进行匹配

查准率，查全率，效率

**第10章 敏捷软件开发**

1.敏捷软件开发的含义

1致力于降低变化的成本；2强调价值；3强调人的作用；4使用增量和迭代的开发方法

**第11章 人机界面设计**

1.设计人员在进行人机界面设计时经常遇到的问题

1、系统响应时间：用户执行某个控制动作到软件做出响应的时间

2、用户求助设施

3、错误信息处理

4、命令标记

2.黄金原则1、让用户拥有控制权2、减少用户的记忆负担3、保持界面一致

**第12章 程序设计语言和编码**

1.程序设计风格

1、源程序文档化

（1）标识符的命名（2）程序的注释（3）视觉组织

2、数据说明

（1）数据说明次序规范化（2）说明语句中变量安排有序化（3）使用注释说明复杂的数据结构

3、语句结构

（1）一行内只写一条语句（2）首先考虑清晰性（3）直截了当说明程序员的用意（4）其他常用规则

4、输入和输出

**第13章 软件测试**

软件测试的目的：发现软件中的错误和缺陷，并加以改正

白盒测试：按照程序内部的结构测试程序，检验程序中的每条路径是否都能按预定要求正确工作。逻辑覆盖测试，基本路径测试，数据流测试，循环测试

黑盒测试：按照程序的功能测试程序，检验与程序功能有关的输入、输出与程序执行是否正确。有四种方法既等价分类法、边界值分析法、错误猜测法和因果图法

测试策略：V模型；单元测试；集成测试；确认测试；系统测试

1.系统测试定义：是将已经确认的软件、[计算机硬件](http://baike.baidu.com/view/1024731.htm)、[外设](http://baike.baidu.com/view/1162088.htm)、网络等其他元素结合在一起，进行信息系统的各种[组装测试](http://baike.baidu.com/view/2303221.htm)和[确认测试](http://baike.baidu.com/view/106760.htm)，系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。

分类：1、回复测试2、安全保密性测试3、压力测试4、性能测试

2.调试、测试的区别

测试的目的是发现错误，当测试发现错误后需要进行调试，调试的目的是确定错误的原因和准确位置，并加以纠正。

**第15章 软件维护与再工程**

1. 软件维护的概念：软件维护是指软件系统交付使用之后，为了改正错误或满足新的需要而修改软件的过程。
2. 软件维护分为：纠错性维护、适应性维护、改善型维护、预防性维护

3.软件可维护性：是指理解、改正、调整和改进软件的难易程度。

对软件可维护性影响的主要因素有；可理解性、可测试性、可修改性和可移植性。

4．逆向工程：指在软件生存周期中，将软件的某种形式描述转换成更抽象形式的活动。

5.再工程：运用逆向工程、重构等技术，在充分理解原有软件的基础上，进行分解、综合、并重新构建软件，用于提高软件的可理解性、可维护性可复用性或演化性。

**第16章 软件项目管理**

1.软件项目管理：是指软件生存周期中软件管理者所进行的一系列活动，其目的是在一定的时间和预设范围

内，有效地利用人力、资源、技术和工具，使软件系统或软件产品按原定计划和质量要求如期完成。

2.软件度量：是指计算机软件范围内的测量，主要是为产品开发的软件过程和产品本身定义相关的测量方法和标度，对软件开发过程度量的目的是为了对过程进行改进，对产品进行度量的且的是为了提高产品的质量。度量的作用是为了有效地采用定量的方式来进行管理。

3.风险管理就是识别和评估风险，建立、选择、管理解决风险的可选方案和组织方案。风险管理的活动主要包括风险标识、风险预测、风险评估、风险管理和控制。

6.可维护性：软件可被修改的能力。修改可能包括纠正、改进或软件对环境、需求和功能规格说明变化的适应。

7.可移植性：软件产品从一种环境迁移到另外一种环境的能力。

9.风险标识分类：项目风险、技术风险、商业风险

8.什么是风险预测：风险预测也称风险估算，风险预测评价每种风险发生的可能性或概率以及当该风险发生时所导致的后果。

风险预测活动包括如下内容：  
1、建立一个尺度，以反映风险发生的可能性。  
2、描述风险的后果。。  
3、估算风险对项目及产品的影响。

4、标注风险预测的整体精确度，以免产生误解。

填空：

5.构件分类模式：枚举分类、刻面分类、属性-值分类

1、自顶向下结合的渐增式测试法，在组合模块时有两种组合策略：深度优先策略和广度优先策略。

2、按数据流的类型，结构化设计方法有两种设计策略，它们是变换型设计和事物型设计。

3、风险管理包括：风险标识、风险预测、风险评估、风险管理与监控

4、数据流图和数据字典共同构成了系统的逻辑模型，是需求规格说明书的主要组成部分

5、一般而言，单独测试某一模块时，要为它设计驱动模块和 桩 模块

6、为了便于对照检查，测试用例应由输入数据和预期的输出结果两部分组成。

7、衡量模块独立性的两个定性标准是什么1.耦合性 2.内聚性

8、用于描述基本加工的小说名的三种描述工具是结构化语言、判定表和判定树。

9、G．J．Myers提出：软件测试是为了发现错误而执行程序的过程。

10、再画分层的DFD时,父图与子图的输入输出数据流要相同。

11.汽车跟发动机之间是什么关系：整体与部分