第一章

1.什么是软件？

是一系列按照特定顺序组织的计算机数据和指令的集合，包括程序、数据和文档。

2.什么是软件危机，其内容主要是指什么？

软件危机是计算机软件在它的开发和维护过程中所遇到的一系列严重问题。主要包含两方面的问题：如何开发软件，怎样满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。软件发展第二阶段的末期，由于计算机硬件技术的进步。一些复杂的、大型的软件开发项目提出来了，但软件开发技术的进步一直未能满足发展的要求。在软件开发中遇到的问题找不到解决的办法，使问题积累起来，形成了尖锐的矛盾，因而导致了软件危机。

软件危机产生的原因是由于软件产品本身的特点以及开发软件的方式、方法、技术和人员引起的。

3.什么是软件工程？

软件工程主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本、改进软件产品质量、提高软件生产率水平的目标。

4.软件工程的目标及其组成部分。

软件工程的目标是：在给定成本、进度的前提下，开发出具有适用性、有效性、可修改性、可靠性、可理解性、可维护性、可重用性、可移植性、可追踪性、可互操作性和满足用户需求的软件产品。

软件工程 3 个要素：方法、工具和过程。

5.软件开发方法的定义。

软件开发方法是一种使用早已定义好的技术集及符号表示习惯组织软件生产过程的方法。主要有

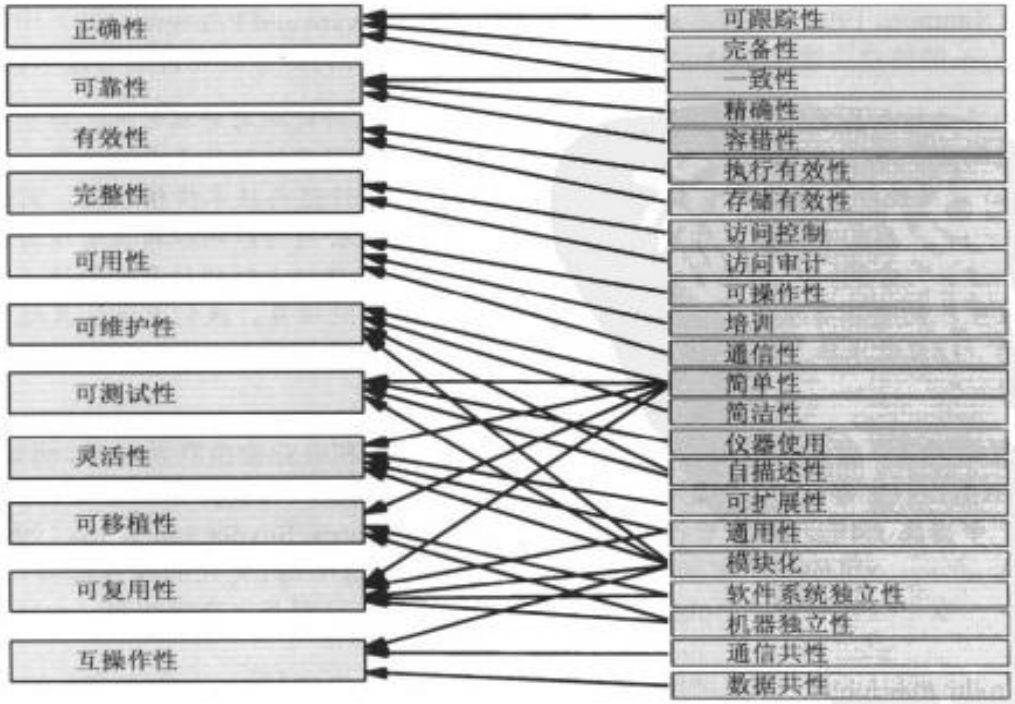
1）结构化方法

2) 面向数据结构的软件开发方法

3)面向对象软件开发方法

6.好的软件的一些主要衡量指标。例如 McCall 的质量模型。

McCall 的质量模型



正确性	在预定环境下，软件满足设计规格说明及用户预期目标的程度。它要求软件本身没有错误
可靠性	软件按照设计要求，在规定时间和条件下不出故障，持续运行的程度
效率	为了完成预定功能，软件系统所需的计算机资源的多少
完整性	为某一目的而保护数据，避免它受到偶然的或有意的破坏、改动或遗失的能力
可使用性	对于一个软件系统，用户学习、使用软件及为程序准备输入和解释输出所需工作量的大小
可维护性	为满足用户新的要求，或当环境发生了变化，或运行中发现了新的错误时，对一个已投入运行的软件进行相应诊断和修改所需工作量的大小
可测试性	测试软件以确保其能够执行预定功能所需工作量的大小
灵活性	修改或改进一个已投入运行的软件所需工作量的大小
可移植性	将一个软件系统从一个计算机系统或环境移植到另一个计算机系统或环境中运行时所需工作量的大小
可复用性	一个软件（或软件的部件）能再次用于其他应用（该应用的功能与此软件或软件部件的所完成的功能有关）的程度
互连性	又称相互操作性。连接一个软件和其它系统所需工作量的大小。如果这个软件要联网或与其它系统通信或要把其它系统纳入到自己的控制之下，必须有系统间的接口，使之可以联结

第二章过程和生命周期的建模

1.什么是软件生命周期？主要分为哪些阶段？各个阶段的主要任务及产生的主要制品？

软件有一个孕育、诞生、成长、成熟、衰亡的生存过程。这个过程即为计算机软件的生存期。当过程是在开发软件产品时，把这种软件开发过程称为软件生命周期。

可行性研究：

任务：了解用户要求和现实环境，从技术、经济、市场等方面研究并论证开发该软件系统的可行性

阶段性产品：可行性论证报告、初步的项目开发计划

需求分析：

任务：确定用户对待开发软件系统的需求包括：功能、性能、运行环境约束

阶段性产品：软件需求规格说明书 SRS(功能，性能和运行环境约束)

概要设计：

任务：根据 SRS 建立目标软件系统总体结构、设计全局数据库和数据结构，规定设计约束，制定集成测试计划等等。

阶段性产品：概要设计规格说明书、数据库或数据结构设计说明书、集成测试计划。

详细设计：

任务：细化概要设计所生成的各个模块，并详细描述程序模块的内部细节(算法，数据结构等)，形成可编程的程序模块，制订单元测试计划

阶段性产品：细化概要设计所生成的各个模块，并详细描述程序模块的内部细节(算法，数据结构等)，形成可编程的程序模块，制订单元测试计划

实现：

任务：根据详细设计规格说明书编写源程序，并对程序进行调试和单元测试，验证程序与详细设计文档一致性

阶段性产品：源程序代码

集成测试：

任务：根据概要设计规格说明书，将经过单元测试的模块逐步进行集成和测试

阶段性产品：生成满足概要设计要求、可运行的系统源程序和系统集成测试报告

确定测试：

任务：根据软件需求规格说明书，测试软件系统是否满足用户的需求

阶段性产品：可供用户使用的软件产品(文档，源程序)

维护：

任务：对使用后的软件进行维护例如:

- ✓ 修正使用过程中发现的错误－纠错性维护
- ✓ 增加新的功能－完善性维护
- ✓ 从一个环境搬迁到另一个环境－适应性维护

阶段性产品：版本更新的软件产品

2.需求分析的定义。

需求分析是指开发人员要准确地理解用户的要求,进行细致的调查分析,将用户非形式化的需求陈述转化为完整的需求定义,再由需求定义转化为相应的软件需求规格说明书（即需求分析的结果）的过程。

3.典型的软件开发过程模型的特点（优缺点）及要求，特别是原型法、瀑布模型、增量和迭代等。

① 瀑布模型
② V 模型
③ 原型化模型
④ 可操作规格说明
⑤ 可转换模型
⑥ 阶段化开发：增量和迭代
⑦ 螺旋模型
⑧ 敏捷方法

原型法：
优点：

1. 允许需求或设计反复调查
2. 减少开发中的风险和不确定性

缺点：

1. 为了使原型尽快的工作，没有考虑软件的总体质量和长期的可维护性。
2. 为了演示，可能采用不合适的操作系统、编程语言、效率低的算法，这些不理想的选择成了系统的组成部分。
3. 开发过程不便于管理。

瀑布模型：
特点：

1. 提供了软件过程模型的基本框架（模板）。采用结构化方法开发,包括结构化分析、结构化设计、结构化程序设计和结构化测试方法。
2. 阶段间具有顺序性和依赖性，强调每一阶段活动的严格顺序。
3. 推迟实现的观点。
4. 每个阶段必须完成规定的文档，以经过评审确认了的阶段工作产品（文档）驱动下一阶段的工作
5. 每个阶段结束前完成文档审查，及早改正错误。

优点：
采用规范的方法；严格规定每个阶段提交的文档；要求每个阶段交出的产品必须经过验证。
缺点：

1. “瀑布模型是由文档驱动的”
   a) 实际项目很少按照该模型给出的顺序进行；

b） 用户常常难以清楚地给出所有需求；
c） 用户必须有耐心，等到系统开发完成。

2. 对如何处理开发中产品和活动的变化没有提供相关的指导
3. 将软件开发视为制造而不是创造
4. 创造一个产品没有迭代的活动
5. 需要等待很长时间

螺旋模型：
优点：

1. 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标
2. 减少了过多测试或测试不足
3. 维护和开发之间并没有本质区别

缺点：

1. 风险驱动，需要相当丰富的风险评估经验和专门知识，否则风险更大
2. 主要适用于内部开发的大规模软件项目，随着过程的进展演化，开发者和用户能够更好的识别和对待每一个演化级别上的风险
3. 随着迭代次数的增加，工作量加大，软件开发成本增加

喷泉模型：
特点：主要用于支持面向对象开发过程体现了软件创建所固有的迭代和无间隙的特征

4. 原型法的特点以及分类：探索型原型、实验型原型和演化型。

5.极限编程的特点。
强调敏捷方法的四个特征

- 交流：保持客户和开发者的交换看法
- 简单性：选择简单设计和实现
- 勇气：尽早并经常性交付功能(敢于承诺并信守诺言)
- 反馈:开发过程中各种活动循环

第三章
1.软件可行性研究的内容。
可行性研究的内容：
1） 技术可行性
技术可行性要分析各种技术因素，例如：
使用现有的技术能否实现这个系统？
是否有胜任开发该项目的熟练技术人员？
能否按期得到开发该项目所需的软件、硬件资源？
2）经济可行性
对经济合理性进行评价，所要考虑的问题是：
这个系统的经济效益能否超过它的开发成本？
这就需要对项目进行价格/利益分析，即“投入/产出”分析。
由于利益分析取决于软件系统的特点，因此在软件开发之前，很难对新系统产生的效益作出

精确的定量描述，所以往往采用一些估算方法。

3）操作可行性

操作可行性评价系统运行后会引起的各方面变化，如：对组织机构管理模式、用户工作环境等产生的影响。

4）社会可行性

社会可行性主要讨论法律方面和使用方面的可行性。例如，被开发软件的权利归属问题、软件所使用的技术是否会造成侵权等问题。

2. 估算工作量的主要方法：代码行、任务分解技术、自动估算成本技术。（了解即可）

1）代码行技术

- 软件成本 = 每行代码的平均成本×估计的源代码总行数
- 代码行技术是比较简单的定量估算软件规模的方法。
- 依据以往开发类似产品的经验和历史数据，估计实现一个功能所需要的源程序行数。

当有以往开发类似产品的历史数据可供参考时，估计出的数值还是比较准确的。把实现每个功能所需要的源程序行数累加起来，就可得到实现整个软件所需要的源程序行数

2）任务分解技术

- 软件开发项目分解为若干个相对独立的任务，分别估计每个单独任务的成本：
- 单独任务成本 = 任务所需人力估计值×每人每月平均工资；
- 软件开发项目总成本估计 = 各个单独任务成本估计值之和。

3）自动估计成本技术

- 采用自动估计成本的软件工具估计。

第四章

2.需求的类型：功能需求、非功能需求或质量需求、设计约束、过程约束。

功能需求: 根据要求的活动描述需求行为

质量需求或非功能需求: 描述软件必须拥有的质量特征

设计约束: 已经做出的设计决策或对问题解决方案集的限制的设计决策

过程约束: 对用于构建系统的技术和资源的限制

3.两种需求文档：需求定义文档和需求规格说明书。

4.需求规格说明书的主要内容。（感觉有点怪）

需求规格说明: 文档化过程步骤，将需求重新陈述为关于要构建的系统将如何运转的规格说明

- 详细描述输入和输出 ，包括
  - 输入的源
  - 输出的目的地,
  - 有效范围
  - 输入输出的数据格式
  - 数据协议
  - 窗口格式和组织
  - 计时约束
- 根据接口的输入输出重新陈述要求的功能
- 对用户的质量需求，设计适配标准

需求定义可以处于环境域的任何地方，可能包括系统的接口
规格说明书仅仅限制在环境域和系统域的交集之中

5. 常用的需求建模表示方法：ER 图、事件跟踪、状态机、Petri 网、数据流图、用例图和原型法。

第五章
1.概念设计和技术设计的内容。

- 概念设计：告诉客户系统将做什么
  - 数据来自哪里?
  - 系统中数据会发生什么情况?
  - 对用户来说，系统将会是什么?
  - 向用户提供的选择是什么?
  - 事件的计时是什么?
  - 报表和屏幕是什么样的?
- 优秀的概念设计的特性
  - 客户语言
  - 不包含技术行话
  - 描述系统功能
  - 与实现无关
  - 与需求文档链接起来
- 技术设计：告诉变成这系统将做什么
  - 对主要硬件部分及其功能的描述
  - 软件构件的层次和功能
  - 数据结构
  --- 数据流

2. 好的设计的衡量：内聚和耦合。

3. 常用的内聚和耦合度类型。

2. 耦合性和内聚性有几种类型？其耦合度、内聚强度的顺序如何？

答：低：非直接耦合→ 数据耦合→标记耦合→ 控制耦合→外部耦合→ 公共耦合→内容耦合 ：高

强：功能内聚→ 信息内聚→ 通信内聚→ 过程内聚→ 时间内聚→ 逻辑内聚→ 巧合内聚：弱

第六章
1. OOM 中的典型特征，其中特别是封装、继承和多态。
标识
抽象
分类
封装
继承

多态
持久性

2. 类和对象的概念及其构成。
类: 用来描述一组具有相同属性和行为的对象
类的层次是根据类间的相同性或差异性组织的
一个类定义成子类
子类可以继承父类的结构、行为以及属性
抽象类用于简化层次
抽象类不能定义对象，除非作为一个子类的实例

对象具有：
1.属性 (例如颜色，大小，位置)
2.行为或操作 (例如起飞，降落，维修)
每个对象都是某个类的实例

第七章
1. 注意编程程序过程中应遵循一定的标准和过程。
- 对单个开发人员的标准
  - 编写代码文档的方法
- 对其他开发人员的标准
  - 集成人员, 维护人员, 测试人员
  - 文档序言
  - 对代码分析的自动化工具
- 设计和实现的匹配
  - 低耦合, 高内聚, 定义明确的接口
  -

2.注意实现容错技术的主要手段是冗余，冗余通常分为四类:(1)结构冗余。 (2)信息冗余 (3)时间冗余和（4）冗余附加技术。

3.软件中的注释主要分：序言性注释和功能性注释两种。

第八章&第九章
1. 测试的目标和衡量标准。
- 测试目标: 发现错误
- 只有当发现了错误时，测试才被认为是成功的
  - 故障识别是确定由哪一个故障或哪些故障引起失效的过程
  - 故障改正是修改系统使得故障得以去除过程

2.测试的分类（或组织）。各种类型的测试的主要任务及所依赖的文档。
- 模块测试、构件测试、单元测试
- 集成测试
- 功能测试

- 性能测试
- 验收测试
- 安装测试
- Alpha 测试
- Beta 测试

（以下是由书上 P292 文字和配图总结的，”---”后为所依赖的文档）

1) 模块测试、构件测试、单元测试
   验证针对设计预期的输入类型，构建能否适当地运行。
   ---构件代码
2) 集成测试
   验证系统构件是否能够按照系统和程序设计规格说明中描述的那样共同工作的过程。
   ---设计规格说明
3) 功能测试
   对系统进行评估，以确定集成的系统是否确实执行了需求规格说明中描述的功能。其结果是一个可运转的系统。
   ---系统功能需求
4) 性能测试
   将系统与这些软件和硬件需求的剩余部分进行比较。当测试在客户的实际工作环境中成功地执行时，它会产生一个确认的系统。
   ---其他软件需求
5) 验收测试
   与客户交换意见，以确定系统是按照客户的期望运转的，根据客户的需求描述对系统进行检查。
   ---客户需求规格说明
6) 安装测试
   确保系统将按照它应该的方式来运行
   ---用户环境
7) Alpha 测试
   α 测试是指软件开发公司组织内部人员模拟各类用户对即将面市软件产品（称为α版本）进行测试，试图发现错误并修正。
8) Beta 测试
   β 测试是由软件的多个用户在实际使用环境下进行的测试，这些用户返回有关错误信息给开发者。

3.黑盒测试和白盒测试的思想，了解白盒测试中的基本路径测试等方法。

黑盒测试（又称行为测试）把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能需求。

白盒测试（又称为结构测试）把测试对象看作一个透明的盒子，测试人员根据程序内部的逻辑结构及有关信息设计测试用例，检查程序中所有逻辑路径是否都按预定的要求正确地工作。

白盒测试的测试方法有代码检查法、静态结构分析法、静态质量度量法、逻辑覆盖法、基本路径测试法、域测试、符号测试、Z 路径覆盖、程序变异。

4.单元测试的主要内容。

主要内容：边界测试、错误处理测试、路径测试、局部数据结构测试、模块接口测试。

5. 集成测试的类型及主要的测试策略。（测试策略没有找到）

- 自底向上的测试
- 自顶向下测试
- 一次性测试
- 三明治测试
- 改进的自顶向下测试
- 改进的三明治测试

6. 确认测试的内容。

测试内容：

- 安装测试
- 功能测试
- 可靠性测试
- 安全性测试
- 时间及空间性能测试
- 易用性测试
- 可移植性测试
- 可维护性测试
- 文档测试

7.测试系统中的测试过程：功能测试、性能测试、验收（或确认）测试、安装测试，及它们的内容。（我是摘 PPT 的，求大神补充）

功能测试

- 比较系统的实际功能与需求
- 根据需求文档开发测试用例

性能测试

- 用于检查
  - 响应速度
  - 结果的精确性
  - 数据的可访问性
- 由小组进行设计和执行

验收测试

- 让客户和用户能够确定我们构建的系统满足了他们的期望
- 编写、执行和评估都是由客户来进行

安装测试

- 测试之前
  - 配置系统
  - 将正确的数量和种类的设备连接到主处理器上
  - 与其他系统建立通信
- 测试
  - 回归测试: 验证系统被正确地安装

第十章

1. 维护活动的类型：

- **改正性**: 维护对日常的系统功能的控制
- **适应性**: 维护对系统修改的控制
- **完善性**: 完善现有系统
- **预防性**: 防止系统性能下降到不可接受的程度

2. 各种维护活动的主要内容和目标。（找不到）

3. 软件再生：文档重构、重组、逆向工程、再工程，以及它们各自的内容和含义。

文档重构：

对源代码进行静态分析，给出更多的信息

- 第一步是将代码提交给一个分析工具
- 输出可能包括:
  - 构建调用关系
  - 类层次
  - 数据接口表
  - 数据字典信息
  - 数据流表或数据流图
  - 控制流表或控制流图
  - 伪代码
  - 测试路径
  - 构件和变量的交叉引用

重组

改变代码结构

- 解释源代码以及用内部形式表示源代码
- 利用转换规则来简化内部表示
- 重新生成结构化的代码

逆向工程：

根据代码重新创建设计和规格说明信息

- 尽量基于软件规格说明和设计方法恢复工程性信息
- 逆向工程被广泛使用，仍然存在一些主要的障碍
  - 实时系统的问题

- 极端复杂系统

再工程：
对现有工程进行逆向工程，接着再改变规格说明和设计以完成逻辑模型 ；然后，根据修改的规格说明和设计生成新的系统

- 是扩展的逆向工程
  - 在不改变整个系统功能的前提下，生产新的软件源代码
- 再工程步骤
  - 系统进行逆向工程
  - 修改并完成软件系统模型
  - 生成新系统