第一章

**1、软件的概念、特征**

软件定义：能够完成预定的功能和性能，能够对相应数据进行加工的程序，以及描述程序，操作的文档。

软件特征：

1. 软件是逻辑产品，是设计开发的，不是生产制造的。
2. 软件受开发环境较大影响。
3. 软件开发事件和成本难以估计。
4. 软件生产是简单拷贝，但大部分是定制。
5. 软件完全测试非常困难。
6. 软件不磨损，老化，但会多次修改。
7. 软件维护易产生新问题。

**2、软件危机、原因（成本增加、技术落后于需求）**

软件危机定义：计算机软件在开发和维护工程中遇到的一系列问题。

产生原因：

* + - 1. 成本增加。
      2. 技术进步落后于需求增长。

**3、软件工程的定义、发展过程（四个阶段）**

软件工程定义：

1. 应用系统化的，科学化的，定量的方法，来开发，运行和维护软件，即，将工程应用到软件。
2. 对1中各种方法的研究。

发展过程：

1. 第一代软件工程——传统的软件工程。
2. 第二代软件工程——对象工程。
3. 第三代——过程工程。
4. 第四代­——构件工程。

第二章：软件过程

**1、软件的生命周期、软件过程概念**

软件生命周期：软件产品或系统从设计。投入使用到被淘汰的全过程。

软件过程概念：为建造高质量软件所需要的活动，动作和任务的框架。

**2、常见的几种软件过程模型（瀑布、增量、演化、喷泉、构件、协同开发（并行开发）的基本思想、优缺点）。敏捷开发的基本思想（敏捷过程）**

瀑布模型

基本思想：规定了各项软件工程活动，以及他们自上而下相互衔接的固定持续，如同瀑布流水，逐级而下。

优点：

1. 阶段间具有顺序性和依赖性。
2. 结构简单，清晰。
3. 文档驱动保障质量，每个阶段必须完成规定的文档；每个阶段结束前完成文档审查，及早更改错误。

缺点：

1. 线性过程太理想化。
2. 各阶段划分固定，产生大量文档，极大增加了工作量。
3. 用户等到开发后期才见到开发结果，增加开发风险。
4. 早期错误可能后期才发现，造成严重后果。

使用范围：瀑布模型适用于系统需求明确，技术成熟，工程管理较严格的场合。

增量模型

基本思想：从一组给定的需求开始，通过构造一系列可执行中间版本来实施开发活动。第一个版本纳入一部分需求，下一个版本纳入更多的需求，依此类推，直到系统完成。

优点：

1. 开发顺序灵活，可灵活调整优先级，先开发需求稳定的模块。
2. 初始阶段不需要太多人力资源。后期根据需要调整。
3. 一个开发周期的问题不会影响整体，有效管理技术风险。

缺点：

1. 难以给定合适的增量。
2. 增量之间相交部分不好处理，需做全盘系统分析。
3. 由于需求变化，易退化为边做边改，失去整体控制。

适用范围：软件可分批交付，易模块化，需求不完整，难一次开发；项目管理人员把握全局的水平高。

演化模型

基本思想：首先实现系统最核心的，最重要的功能。

使用范围：会随着事件的推移而演化的软件。

演化过程模型：

1. 原型模型。
2. 螺旋模型。

原型模型

基本思想：通过总体目标或已确认需求，实现一个可实际运行的系统初步模型，供开发人员和进行交流和评审，以便较准确获得用户的需求，然后采用逐步求精的方法使原型逐步完善。

优点：

1. 开发效率高。
2. 成本低，开发时间短。
3. 能与用户直观交流，澄清模糊需求，并且调动用户积极性，及早发现系统潜在问题。

缺点：

1. 限制开发人员创新。
2. 没有考虑软件的整体质量以及长期可维护性。
3. 容易造成质量差问题。

螺旋模型

结合了原型模型的迭代性质和瀑布模型的系统、可控性特点。

螺旋模型强调风险管理，因此该模型适用于大型系统和软件的开发。

基本思想：过程沿着螺线旋转，在笛卡尔坐标的四个象限上分别表达了四个方面的活动：

1. 制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件。
2. 风险分析：分析所选方案，考虑如何识别和消除风险。
3. 设计实施：设计、编码、测试、优化。
4. 用户反馈：评价开发工作，提出修正建议。

优点：

1. 支持用户需求的动态变化。
2. 原型看作可执行需求说明，易于用户和开发人员理解，为用户参与决策提供方便。
3. 强调可扩充性和可修改性，进化贯穿软件生存周期，利于软件适应能力。
4. 引入严格的风险识别，风险分析和风险控制较完善。
5. 方便决策，降低开发风险。

缺点：

1. 若迭代次数多或者效率低，将增加成本和开发时间。
2. 要求队伍有丰富的风险评估经验和知识。

适用范围：需求不明确的软件，特别是大型软件系统。

喷泉模型

基本思想：喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程。

优点：

1. 各阶段无明显界限，可同步进行。
2. 软件开发效率高，时间短。

缺点：

1. 各个开发阶段重叠，需大量开发人员，不利于管理。
2. 需要严格管理文档，使审核难度加大。

适用范围：面向对象且基于网络的软件开发。

协同开发模型

基本思想：定义过程网络，网络上多个工程活动、动作和行为同时存在，提供精确的项目当前状态图，而一个并发过程模型是由用户要求、管理决策和结果复审驱动的。

优点：

1. 可用于所有类型软件开发，特别是客户/服务器结构更加有效，能够提供精确的当前项目状态图。
2. 各阶段子活动层次划分封闭，利于解决复杂问题。

缺点：

1. 要求较强的团队合作能力。
2. 大量并发的项目进度控制困难。

适用范围：所有类型的开发。

基于构件的模型

优点：降低成本和风险，节约时间。

缺点：

1. 缺乏通用的组装结构标准。
2. 可重用性和高效性不易协调，要求开发人员有丰富检验。
3. 影响产品质量，客户满意度低。

适用范围：适用于现有构件库充足、接口定义良好的面向对象软件开发。

敏捷开发过程(Agile Development)

基本思想：在开发的同时响应变化。变化就是软件开发本身，软件构建、团队成员、使用新技术都会带来变化。

敏捷开发过程：

1. 基本原理和开发准则的结合。
2. 基本原理强调客户满意度和较早的软件增量交付；小但有激情的团队；非正式的方法；最小的软件工程产品；简化整体开发。
3. 开发准则强调分析和设计的交付，以及开发者和客户之间积极持续的交流。
4. 目前的敏捷过程模型主要包括极限编程（XP），自适应软件开发（ASD），动态系统开发方法（DSDM）等。

适用范围：小型软件。

第三章：需求分析

**1、需求分析的定义、类型、过程（4个阶段，理解其主要工作）**

需求分析定义：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。

类型：

1. 功能性需求：描述系统应该做什么，即为用户和其他系统完成的功能，提供的服务。
2. 非功能需求：必须遵循的标准，外部界面的细节，技术指标，实现的约束条件，质量属性等。

需求分析过程：

1. 需求获取
2. 需求提炼 （需求分析）
3. 需求描述（撰写需求规格说明书）
4. 需求验证

**2、结构化分析方法，重点讲解模型：数据流图（会画、会分析、能查错）**

**3、面向对象的分析方法，重点讲解模型：用例图（会画、会分析、能查错）**

**4、需求规格文档编制（基本结构，主要用于实验）**

第四章：系统设计

**1、软件设计的定义、过程（架构设计、详细设计的概念）、软件设计的概念和原则**

**2、传统的系统设计方法，重点讲解模型：程序流程图（会画、会分析、能查错）**

**3、面向对象的系统设计方法，重点讲解模型：顺序图（时序图） （会画、会分析、能差错）（放在第三章讲的）**

第六章：质量保证

**1、质量保证相关概念：软件质量、质量控制（3个内容）、质量保证（原则）、质量成本（3种成本）、软件质量保证（3类活动）、软件评审（3种形式）、软件可靠性（2类评估指标）**

**2、软件测试的策略：V模型及各阶段主要内容、方法**

**3、软件测试的技术：白盒测试（逻辑覆盖）、黑盒测试（等价类划分和边界值分析）（会设计测试用例、会查错）**

第七章：软件维护

**1、软件维护的定义、分类（4类）**

**2、软件维护应注意的问题（3个方面基本内容）**

**3、软件维护过程模型**

**4、软件维护技术（程序理解的任务、再工程的定义和模型、逆向工程的定义和主要内容）**

第八章：项目管理

**1、软件项目管理相关概念：项目管理、四要素（了解）**

**2、软件度量：软件度量定义、决定要素，基于规模测量的基本思想，基于代码行数、功能点度量方法（给定数据计算）、三点期望值法、基本的COCOMO模型（给定数据计算）**

**3、项目计划：关键路径计算**