软件工程知识点（全）

（一）软件工程概述

软件危机及其表现

定义

软件危机指的是计算机软件开发维护过程中所遇到的一系列严重问题。概述来说，软件危机包含两方面的问题：一是如何开发软件，以满足不断增长，日趋复杂的需求，二是如何维护数量不断膨胀的软件产品

表现

1. 对软件的开发成本与进度估算不准确，开发成本超出预算，项目经常延期，无法按时完成任务
2. 开发的软件不能满足用户需求
3. 软件产品的质量低
4. 开发软件的可维护性差
5. 软件通常没有适当的文档资料
6. 软件的成本不断提高
7. 软件开发的生产率的提高赶不上硬件的发展和人们需求的增长

软件工程的概念

软件工程师采用工程的概念、原理、技术和方法来指导软件开发和维护的工程的学科，以工程化的原理和方法来解决软件问题

软件工程的三要素

方法，工具，过程

（二）软件的生命周期

软件生命周期的3个阶段

软件定义，软件开发 ，运行维护

软件生命周期的8个小阶段

软件定义：问题定义，可行性研究，需求分析

软件开发：总体设计，详细设计，编码，单元测试，综合测试

运行维护（持续满足用户需求）

软件工程的7种模型

（1）瀑布模型

适用对象：适用于用户需求明确，完整、无重大变化的软件项目

特点：时间上具有顺序性与依赖性；推迟实现的观点；质量保证的观点（1.审核机制，2.文档齐全）

优点：适用于用户需求明确，完整、无重大变化的软件项目的开发，成功的原因很大的程度上是因为它基本上是一种文档驱动的模型

缺点：实际项目很少按照模型给出的顺序进行，用户常常难以清楚的给出所有需求，用户必须有耐心，等到系统开发完成

（2）原型模型—快速原型模型

适用对象：用户**不能给出完整，准确的需求**，或开发者不能确定算法的有效性，操作系统的适应性或人机交互的形式

特点：可根据用户的一组基本需求，快速建立一个原型（可运行的软件），然后进行评估，进一步细化，调整原型，满足用户需求，也使开发者对将要做的事情有更好的理解

缺点：没有考虑软件的整体质量与长期的可维护性；为了演示，可能采用不合适的操作系统、编程语言、低效率的算法，这些不理想的选择成了系统的组成部分；开发过程不便于管理

（3）增量模型

适用对象：软件体系结构必须是开放的

特点：早期的版本实现用户的基本需求，并提供给用户评估的平台；反复的应用瀑布模型的基本成分和原型模型的迭代特征，每一个线型过程产生一个"增量"的发布或提交，该增量均是一个可运行的产品

优点：在较短时间内向用户提交可完成部分工作的产品，并分批、逐步地向用户提交产品 ；整个软件产品被分解成许多个增量构件，开发人员可以一个构件一个构件地逐步开发 ；逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品；采用增量模型比采用瀑布模型和快速原型模型需要更精心的设计，但在设计阶段多付出的劳动将在维护阶段获得回报

缺点：在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。向现有产品中加入新构件的过程必须简单、方便，也就是说，软件体系结构必须是开放的； 开发人员既要把软件系统看作整体。又要看成可独立的构件，相互矛盾;多个构件并行开发，具有无法集成的风险

（4）螺旋模型

适用对象：复杂的大型软件，开发一个原型往往达不到要求

特点：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；主要适用于内部开发的大规模项目，随着过程的进展演化，开发者和用户能够更好的识别对待每一个演化级别上的风险；随着迭代次数的增加，工作量加大，软件开发成本增加

优点：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；减少了过多测试或测试不足；维护和开发之间并没有本质区别

（5）喷泉模型

适用对象：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

特点：

1.开发过程有分析，系统设计，软件设计，实现四个阶段

2.各设计阶段相互重叠，它反映了软件过程并行性的特点

3.以分析为基础，资源消耗成塔型

4.反映了软件过程迭代性的自然特征，从高层返回底层无资源消耗

5.强调增量开发，整个过程是一个迭代逐步提炼的过程

（6）RUP—统一软件开发过程

每个周期的4阶段：初始，细化，构造，移交

每个阶段围绕的5个的核心工作流：需求，分析，设计，实现，测试

4阶段及其对应的任务：

1. 初始阶段：进行问题定义，确定目标，评估其可行性，降低关键风险
2. 细化阶段：制定项目计划，配置各类资源，建立系统架构（包括各类视图）、
3. 构造阶段：开发整个产品，并确保产品可移交给用户
4. 移交阶段：产品发布，安装，用户培训

（7）敏捷开发与极限编程

特点：具有以**人为核心、循环迭代、响应变化**的特点，着眼于高质量的快速交付令客户满意的工作软件 ；**开发过程以代码为核心**，而不是以文档为核心，代码编写，测试，发布，重构，然后进入第二次迭代，经过多次小型迭代开发过程逐步逼近实际需求；以人为本。程序员在软件开发中不再是单纯被管理的对象，而是开发的主体

代表模型：scrum模型，XP极限编程

（三）UML设计

全称：**UML（Unified Modeling Lauguage）统一建模语言**

UML的5类图又细分为9种图：

1. 用例图
2. 静态图（类图，对象图，包图）
3. 行为图（状态图 活动图）
4. 交互图（时序图 协作图）
5. 实现图（组件图 配置图）

用例图

用例的定义：从用户的观点对系统行为的一个描述

用来从用户的观察角度收集系统需求

用例图表达系统的**外部事物与系统的交互**，它表达了系统的功能，即系统所提供的服务

整个软件项目开发可以用Use Case 驱动的方式进行

类图

建立类图的步骤

1. 研究分析问题领域
2. 发现对象与类，明确它们的含义和责任，确定属性。
3. 发现类之间的关系。把类之间的关系用关联、泛化、聚集、组合、依赖等关系表达出来。
4. 设计类与关系。调整和细化已得到的类和类之间的关系，解决诸如命名冲突、功能重复等问题。
5. 绘制类图并编制相应的说明。

类图的三个抽象层次

概念层，说明层，实现层

活动图与状态图的区别

状态图：用来描述对象，子系统，系统的生命周期。通过状态图可以了解一个对象所能达到的所有状态，以及对象收到的事件对对象状态的影响。

活动图：显示动作及其结果。着重描述操作（方法）实现中所完成的工作以及用例实例或对象中的活动，它是状态图的一个变种。

状态图与活动图的区别：**活动图主要描述动作及对象状态改变的结果。状态图主要描述的是事件对对象状态的影响**

时序图与协作图的区别

协作图和时序图都是表示对象间的交互作用，只是它们侧重点有所不同。

时序图描述了交互过程中的时间顺序，但没有明确的表达对象间的关系，协作图描述了对象间的关系，但时间顺序必须从序列号获得。

（四）需求分析

如何获取用户需求及常用方法

1. 访谈
2. 情景分析
3. 联合分析小组
4. 快速建立软件原型

需求分析的任务

1. 需求获取
2. 分析建模
3. 文档编写
4. 需求验证

面向对象需求分析的三种模型

1. 对象模型（类图）
2. 动态模型（活动图）
3. 功能模型（用例图）

面向对象设计的原则

抽象，信息隐蔽，模块独立，高内聚，低耦合

需要掌握如何判断低耦合与高内聚

（五）软件测试与维护

软件测试

软件测试，是为了**发现错误**而执行程序的过程

测试只能找出程序中的错误，但在未发现错误时，并不能证明程序中没有错误

软件测试的目标

发现错误并不是软件测试的最终目标

测试阶段的根本目标是尽可能多的发现软件中潜藏的错误，最终把一个高质量的软件系统交给用户使用

黑盒测试常用方法

边界法 等价类 决策表 正交表 场景测试

白盒测试常用方法

静态代码分析 语句覆盖 条件覆盖 判定覆盖 判定/条件覆盖 修正判定/条件覆盖 路径覆盖

软件测试各阶段

测试阶段	主要依据	测试人员及方法	主要测试内容
单元测试	系统设计文档	由程序员执行白盒测试	接口测试，路径测试
集成测试	系统设计文档与软件需求	由程序员执行黑盒测试与白盒测试	接口测试、路径测试、功能测试、性能测试
系统测试	软件需求	由独立测试小组执行黑盒测试	功能测试、健壮性测试、性能测试、UI测试、安全性测试、压力测试、可靠性测试、安装/反安装测试等
验收测试	软件需求	由用户执行黑盒测试	同上

调试与测试的区别

调试（也称为纠错）作为成功测试的后果出现，**也就是说，调试是在测试发现错误之后排除错误的过程（测试是发现错误的过程）**。调试就是把症状和原因联系起来的尚未被人深入认识的智力过程。

软件4种维护及其定义

定义

就是在软件已经交付使用之后，为保证软件在相当长的时期能够正常运作所进行的软件活动

类型

改正性维护 ：为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，所进行的诊断和改正错误的过程就叫做改正性维护

适应性维护 ：使软件适应外部环境与数据环境发生变化的维护

扩充与完善性维护：满足用户对软件提出的新的功能与性能，提高软件可维护性的要求的维护

预防性维护：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

决定软件可维护性的基本要素

可理解、可测试、可修改、可移植和可重用性

文档是影响软件可维护性的决定因素