**1.什么是软件？**

**软件是计算机及手机等终端设备运行的程序、数据、文档和服务的集合。**

**2.什么是软件危机？**

**软件危机是计算机软件在研发、运行、维护和管理过程中，出现的一系列严重问题的现象。软件危机直接导致软件工程的产生，概念提出时间：1968年10月。**

**3.软件危机产生的原因以及解决措施(技术方法、开发工具、组织管理)**

**原因：**

**(1)软件开发规模、复杂度和需求量不断增加；**

**(2)软件需求分析与设计不完善有欠缺，致使软件开发、维护和管理或文档出现问题；**

**(3)没有按照工程化方式运作，开发过程无统一的标准和准则、规范方法；**

**(4)研发人员与用户或研发人员之间互相的交流沟通不够或文档资料不完备；**

**(5)软件测试调试不规范不细致，提交的软件质量不达标；**

**(6)忽视软件运行过程中的正常维护和管理。**

**解决措施：**

**(1) 技术方法。运用软件工程的技术、方法和标准规范。**

**(2) 开发工具。选用先进高效的软件工具，同时采取切实可行的实施策略。**

**(3) 组织管理。研发机构需要组织高效、管理制度和标准严格规范、职责明确、质量保证、团结互助、齐心协力，注重文档及服务。**

**4.软件工程的概念（重点）**

**软件工程是软件开发、运行、维护和引退的系统方法。**

**软件工程 = 工程原理 + 技术方法 + 管理技术**

**5.软件工程的4个发展阶段**

**传统软件工程、对象工程、过程工程、构件工程**

**6.软件工程的知识体系**

**实践知识域、教育基础知识域、相关学科知识域**

**7.软件工程三要素**

**方法、工具、过程与管理**

**8.软件生命周期的概念以及分为几个阶段,并各阶段的任务**

**软件生存周期是从开始研发软件到软件停止使用的整个过程。是指软件产品从提出开发需求开始，经过开发、使用和维护，直到最后淘汰的整个周期。**

**阶段：开发策划、需求分析、概要设计、设计、编码、测试、运行与维护。**

|  |  |
| --- | --- |
| **开发策划** | **主要完成问题定义、可行性论证、制定开发计划和项目申报立项,明确“要解决什么问题”。** |
| **需求分析** | **需求分析和定义阶段任务不是**  **具体地解决问题，而是确定软件须具备的具体功能、**  **性能等，即“必须做什么”及其他指标要求。** |
| **概要设计** | **主要设计软件的总体(外部)结构,结构组成模块,模块层次结构、调用关系及功能。并设计总体数据结构等。** |
| **详细设计** | **对模块功能、性能、可靠性等进行具体技术描述，并转化为过程描述。** |
| **编写程序** | **又称编码（具体实现），将模块的控制结构转换成程序代码。** |
| **测试** | **为了保证软件需求和质量，在设计测试用例基础上对软件进行检测。** |
| **运行与维护** | **对交付并投入使用的软件进行各种维护，并记录保存文档。** |

**9.传统的软件生命周期模型及各自特点 ：瀑布模型、原型模型、螺旋模型**

|  |  |
| --- | --- |
| **瀑布模型** | |
| **3个阶段** | **计划时期、开发时期、运行时期** |
| **3个特点** | 1. **开发过程的顺序性。**   **（2）统筹兼顾不过早编程。**  **（3）严格要求保证质量。** |
| **缺陷** | **将充满回溯且相互重叠的软件开发，过程硬性地分为多个阶段，随着开发软件规模的增加，造成的危害增大。** |
| **适用范围** | **需求确定、变化小的项目，升级的项目以及以前有类似开发经验的项目。** |
| **原型模型** | |
| **概述** | **原型是指模拟某种产品的原始模型，软件开发中的原型是软件的一个早期可运行的版本，它反映了最终系统的重要特性。** |
| **特点** | **实际可行具有最终系统的基本特征构造方便、快速，造价低** |
| **适用范围** | **用户需求开始定义不清、决策方法、结构化程度不高的项目** |
| **增量模型** | |
| **概述** | **增量模型灵活性很强，利用增量模型开发的软件被作为一系列的增量构件进行设计、实现、集成和测试，每个构件具有一定功能，并最终能组合成一个具有完整功能软件的模块。** |
| **适用范围** | **软件需求不明确、设计方案有一定风险的软件项目** |
| **缺陷** | **（1）需要软件具备开放式的体系结构。**  **（2）软件过程的控制易失去整体性。** |
| **增量模型与**  **瀑布模型之间的本质区别** | **瀑布模型属于整体开发模型，**  **增量模型属于非整体开发模型。** |
| **螺旋模型** | |
| **概述** | **螺旋模型将瀑布模型和快速原型模型结合，强调了其他模型所忽视的风险分析，适合于大型复杂系统，吸收了“演化”的概念，可使开发人员和客户对每个演化层的风险有所了解，继而做出应有反应。** |
| **开发过程** | **制定计划、风险分析、实施工程和客户评估** |
| **特点** | 1. **是一个演化软件过程模型，将原型实现的迭代特征与瀑布模型的线性顺序模型中控制的和系统化的方面结合起来** 2. **开发过程具有周期性重复的螺旋线状。强调了风险分析。** 3. **采用循环的方式逐步深入** 4. **确定系列里程碑，确保客户满意** |
| **适用范围** | **庞大而复杂、高风险的系统** |

**几种模型的比较**

|  |  |  |
| --- | --- | --- |
| **模型** | **优点** | **缺点** |
| **瀑布模型** | **阶段评审、文档驱动** | **系统可能不满足客户的需求** |
| **原型模型** | **关注满足客户需求** | **可能导致系统设计差、效率低，难于维护。** |
| **螺旋模型** | **风险分析** | **风险分析人员需要有经验且经过充分训练。** |

**1.0.面向对象的生命周期模型有哪些及各自特点**

|  |  |
| --- | --- |
| **喷泉模型** | |
| **概述** | **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](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1/2262089" \t "https://baike.baidu.com/item/%E5%96%B7%E6%B3%89%E6%A8%A1%E5%9E%8B/_blank)的软件开发过程。该模型认为[软件开发过程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E8%BF%87%E7%A8%8B/3758917" \t "https://baike.baidu.com/item/%E5%96%B7%E6%B3%89%E6%A8%A1%E5%9E%8B/_blank)自下而上周期的各阶段是相互迭代和无间隙的特性。** |
| **特点** | 1. **软件开发过程有4个阶段：需求分析、总体设计、详细设计和实现** 2. **各阶段相互重叠，反映了软件过程并行性的特点。** 3. **以分析为基础，资源消耗成塔形，在分析阶段消耗的资源最多。** 4. **反映了软件过程迭代性的自然特性，从高层返回低层无资源消耗。** 5. **强调增量开发，依据分析一点设计一点的原则，并不要求一个阶段的彻底完成，整个过程是一个迭代的逐步提炼的过程。** 6. **是对象驱动过程，对象是活动作用的实体，也是项目管理的基本内容。** 7. **实现中由于活动不同，可分为系统实现和对象实现，这既反映了全系统的开发过程，也反映了对象族的开发和重用过程。** |
| **适用范围** | **利用面向对象技术的软件开发项目** |
| **RUP（Rational Unified Process），[统一软件开发过程](https://baike.baidu.com/item/%E7%BB%9F%E4%B8%80%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E8%BF%87%E7%A8%8B/2088774" \t "https://baike.baidu.com/item/RUP/_blank)** | |
| **概述** | **在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发活动为开发成员提供了必要的准则、模板和工具指导,并确保共享相同的知识基础。** |
| **开发阶段** | **初始阶段、细化阶段、构造阶段、交付阶段。** |
| **适用范围** | **需求不确定，事先不能完整定义产品所有需求、计划多变的开发的软件** |

* **个体和交互 胜过 过程和工具**
* **可以工作的软件 胜过 面面俱到的文档**
* **客户合作 胜过 合同谈到**
* **响应变化 胜过 意计划**

**1.可行性分析**

|  |  |
| --- | --- |
| **概念** | **可行性分析，是对拟研发软件项目分析论证可行性和必要性的过程。** |
| **特点** | **预见性、公正性、可靠性、科学性等** |
| **意义** | **是软件项目开发前非常重要的一个关键环节，决定整个软件项目的开发成败，具有非常重要的经济意义和现实意义。** |
| **任务** | **决定软件项目“做还是不做”、及完成对可行项目的“初步方案”。** |
| **5个方面** | **技术可行性分析、经济可行性分析、社会可行性分析、开发方案可行性分析、运行可行性分析** |

**2.可行性分析的三个重要方面**

**技术可行性分析、经济可行性分析、社会可行性分析**

**3.成本效益分析(公式、应用)**

|  |
| --- |
| **公式** |
| **（1）投入产出比R。是指软件项目全部投资与产出增加值总和之比。**  **R=K/IN (K为投资总额，IN为软件生存期内各年增加值的总和)**  **（2）货币的时间价值。由于利率的变化等因素，货币的时间价值能较准确地估算。假设年利率为i，若项目开发所需经费即投资为P元,则n年后可得资金数为F元：**  **F = P﹒(1+ⅰ) n**  **反之，若n年后可得效益为F元，则这些资金现在的价值为：**  **P = F ∕ (1+ⅰ) n** |
| **应用** |
| **假设开发一套企业应用系统需要投资20万元,5年内每年可产生直接经济效益9.6万元，设年利率为5 %,试计算投入产出比。**  **考虑到货币的时间价值，5年的总体收入应当逐年按照上式估算，其每年的收人折算到当前的数据如表2-3所示。** |
| **新软件项目的投入产出比(效益成本比)为：R= 41.5630/20 = 2.0782。**   1. **投资回收期。指使累计的经济效益等于最初的投资费用所需的时间。投资回收期越短，利润获得越大越快，项目越值得开发。**   **比如：两年后收人17.8513万元，尚缺2. 15万元没有收回成本，还需要时间：2. 15 / 8.2928=0.259（年），即投资回收期（时间）为2.259年。** |
| **例题：货币的时间价值F = P﹒(1+ⅰ) n（P为本金、i为利率、n为年数、F为效益或利润）**  **某软件公司项目的利润分析如下表所示。设贴现率为10%，则第二年结束时的利润总额净现值为（）元。**    **[解析] 投资回收分析、投资回报率和净现值是三种常用的用于评估经济可行性的技术。其中，现值的计算公式为：PVn=1/(1+i)n，其中，PVn是从现在起到第n年1元的现值，i为贴现率。所以，第二年的利润现值为：**  **11000/(1+0.1)=10000 12100/(1+0.1)2=10000**  **累加为20000** |

**4. 系统流程图的意义及其绘制(各部件的业务处理与数据流向) P46**

**1.需求分析**

**概念：主要是搞清软件应用用户的实际具体需求，包括功能需求、性能需求、数据需求、安全及可靠性要求、运行环境和将来可能的业务变化及拓展要求等，并建立系统的逻辑模型，写出“软件需求规格说明(SRS)”等文档。**

**任务：需求分析的基本任务是通过软件开发人员与用户的交流和讨论，准确地分析理解原系统，定义新系统的基本功能、性能、开发时间、投资情况人员安排等，并获得需求规格说明书。**

**2. 软件需求包含哪些类别(业务需求、用户需求、功能需求、非功能需求)**

|  |  |
| --- | --- |
| **业务需求** | **业务需求反映组织机构或客户对系统和产品高层次的目标要求，它们在项目视图与范围文档中予以说明。** |
| **用户需求** | **用户需求是从用户角度描述系统所完成的任务或者是用户期望有的产品属性。用户需求文档用于描述用户使用软件产品要完成的任务。** |
| **功能需求** | **功能需求描述系统所提供的功能或服务。即定义系统的主要功能、系统的输入/输出信息、系统的约束等。** |
| **非功能需求** | **安全性、可靠性、响应时间、可移植性、可重用性等** |

**3. 结构化分析**

**概念：结构化分析方法是面向数据流的需求分析方法，SA方法根据软件内部的数据传递、变换关系，自顶向下、逐层分解，绘出满足功能要求的模型。**

**指导思想：自顶向下、逐步求精**

**基本原则：抽象与分解**

**工具：E-R图、数据流图、状态转换图、数据字典（核心）；**

**E-R图描述数据模型、数据流图描述功能模型、状态转换图描述系统的各种行为模型、数据字典用来描述软件使用或产生的所有数据对象。**

**3.软件设计的阶段(总体:详细)**

1. **总体设计（概要设计）：主要确定总体架构、总体设计文档和方案。**
2. **详细设计：即过程设计，是具体细化，确定组成模块及联系、处理过程、数据库及网络、界面设计、软件设计文档（含具体方案）等**

**4.总体设计阶段的主要任务是什么？**

1. **软件系统总体结构设计和模块结构设计。**
2. **软件处理流程设计。**
3. **确定软件的功能及调用关系。**
4. **数据库及数据结构总体设计。**
5. **网络及接口概要设计。**
6. **确定软件系统的实现方案。**
7. **出错问题概要设计。**
8. **性能、可靠性及安全性概要设计。**
9. **文档及维护概要设计。**

**1.模块化、信息隐蔽、分解、耦合、内聚 （软件设计的基本原则）**

**模块化：是将复杂的软件划分为功能相对独立且易于处理的模块的过程，软件的层次结构正是模块化的具体体现。模块化是软件的一个重要属性，其特性提供了处理复杂问题的一种方法，同时也有利于软件的有效实现和管理。**

**信息隐蔽：主要是指模块所包含的“过程及数据”信息，对于其他模块需要隐蔽。**

**分解：**

**耦合：是指在软件模块之间（外部）互相依赖的方式和程度，是影响软件复杂程度及性能的重要因素。**

**内聚：内聚表示一个模块内部各个组成元素之间互相结合得紧密程度，是信息隐蔽和局部化概念的自然扩展。偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚和功能内聚。**

**2.数据流的分类(中心变换型、事务处理型)**

**中心变换类型：这类数据流图可看成是对输入数据进行转换而得到输出数据的处理过程。这类图的特点是DFD图可以明显分为“输入-处理-输出”3部分，DFD图的转换采用变换分析技术。**

**事务处理型：事务是完成作业处理的最小单元，输入流在经过某个“事务中心”时接收数据并分析确定其类型，然后根据所确定的类型为数据选择其中的一条加工路径。事务处理设计方法的步骤：确定DFD中的事务中心和加工路径；进行以及分析，设计上层模块；进行二级分析，设计中下层模块。**

**3.详细设计过程的任务是什么**

**1模块的算法设计。 2模块内的数据结构设计。 3模块接口设计。**

**4其他相关设计。 5模块测试用例设计。 6编写详细设计文档。**

**7详细设计评审及实现的方案。**

**1.对象三要素：对象标识、属性和服务**

**2.面向对象的定义**

**面向对象=对象+类+继承+消息通信**

**3.对象、类、实例、封装、继承、多态**

**对象：是系统中的基本运行实体（如函数、控件），是代码和数据的集合，即现实世界中的具体事物，是构成软件系统的基本单位，面向对象系统是数据抽象与过程抽象的综合。面向对象方法以对象分解代替传统方法的功能分解。面向对象的系统由对象组成，复杂对象由简单的对象组合而成。**

**类：也称为对象类，是对具有相同属性和服务的一组对象的抽象定义。**

**实例：类定义了各个实例所共有的结构，类的每个实例都可使用类中定义的操作。实例的当前状态由实例所执行的操作定义。**

**封装：是指将软件内部具体实现进行隐藏，将数据和操作数据的源代码进行有机结合，形成“类”，类的成员包括数据和函数。**

**继承：是父类和子类之间共享数据结构和方法的一种机制，是以现存的定义的内容为基础，建立新定义内容的技术，是类之间的一种关系。单重继承：子类只继承一个父类的数据结构和方法；多重继承：子类继承了多个父类的数据结构和方法。**

**多态：是指多种类型的对象在相同的操作或函数、过程中取得不同结果的特性。**

**4.对象模型的五个层次以及OOA的五项建模活动**

**对象模型：类与对象层、属性层、服务层、结构层和主题层**

**对应OOA过程中建立对象模型的5项主要活动：发现对象、定义类、定义属性、定义服务、划分结构**

**5.面向对象的建模**

**1确定对象类。通过分析确定所有的对象类。**

**2定义数据词典。主要用于描述类、属性和关系**

**3组织并简化对象类。通过继承进行组织和简化对象类。**

**4测试访问路径。测试所有的访问路径。**

**5对象分组建立模块。有对象之间的关系和对象的功能将对象进行分组，并建立模块。**

**6.面向对象的设计准则(抽象、信息隐蔽、高内聚、低合、可重用)**

**抽象：抽象是指强调实体的本质内在的属性，忽略一些无关紧要的属性。**

**信息隐蔽：在OOM中即为“封装性”，是保证软件部件具有优良的模块性的基础。**

**高内聚：指子系统内部是由一些关系密切的类构成，处理少数的“通信类”外，子系统中的类应只与该子系统中的其他类协作，从而构成具有强内聚性的子系统。**

**低耦合：按照抽象与封装性，使子系统之间的联系尽量少，子系统应具有良好的接口，子系统通过接口与系统的其他部分联系。**

**可重用：软件重用是提高开发效率和质量的重要途径。**

**7.面向对象分析与设计说明书**

用例图

系统行为

类图

系统中的词汇

交互图

系统中

状态图

事物之间的

活动图

相互协作行为

用户的身份分析

描述

说明

**1.软件实现的主要任务**

**软件实现是通过软件编程或利用复用与生成技术集成、测试和调试，具体完成软件产品（项目/工程）的过程。**

**主要任务是将软件详细设计的结果转换为目标软件。**

**2.软件实现的过程包括编程和单元测试**

**在微观上，软件实现是指通过编程、调试、单元与集成测试、系统测试等创建软件产品的过程。软件实现是在软件设计的基础上进行的，其本身也涉及设计和测试工作。**

**3.软件实现的输入和输出**

**软件实现的输入是：软件详细设计文档，输出是：源程序、目标程序和用户指南。**

**4.面向对象实现的准则**

**高可重用性：软件重用有多个层次，在编程阶段主要考虑代码复用。**

**高可扩充性：提高软件可重用性，也可提高软件可扩充性。此外，还有封装实现策略、不应以同一方法完成全部关联、尽量少使用多分支语句、精心确定够用方法。**

**高可靠性及健壮性：可靠性指软件的安全性及在规定的条件和时间内完成规定的功能和性能；健壮性是指软件对于不符合规范要求的输入处理能力。**

**5.软件版本管理**

**中间版本：指软件未完成前且未进行正式测试的版本。**

**α版本：α版本是指软件未完成但可以提交进行严格测试的软件版本。**

**β版本：β版本是指可以稳定运行的软件系统，可以交付给终端用户进行测试。**

**发布版本：是指正式向社会发布或用户提交使用的软件产品。**

**维护版本：是对发布版本进行错误纠正，以及进行功能或性能改进的软件版本。**

**6.软件编程语言的发展与分类**

**发展：**

**第一代语言：机器语言和汇编语言。从第一台电子计算机开始**

**第二代语言：主要应用于各种计算，20世纪50年代末和60年代初**

**第三代语言：具有直接支持结构化构建，并具有很强的过程能力和数据结构能力特点的结构化与现代程序语言，通用高级语言；面向对象的语言；专用语言。**

**第四代语言：超高级编程语言，强大的数据管理能力；提供一组高效的、非过程化的命 令；可以满足多功能、一体化的要求。**

**分类：**

**语言层次：面向机器语言；面向问题语言**

**语言适用性：通用语言；专用语言**

**语言面向方面：面向过程语言；面向对象语言**

**应用领域：科学计算语言；数据处理语言；实时处理语言**

**语言级别：低级语言；高级语言**

**7.软件实现阶段有哪些文档**

**软件产品规格说明（SPS）**

**计算机编程手册（CPM）：主要说明目标软件的编程环境和编程信息**

**软件用户手册（SUM）：描述手工操作该软件的用户应如何安装和使用一个计算机软件系统或子系统**

**1.软件测试的定义**

**在规定的条件下对软件程序进行检测运行操作，以发现程序错误，衡量软件标准，并对其是否能满足设计要求进行评估的过程。**

**2.软件测试的最主要任务**

**功能测试**

**3.软件测试的步骤有哪些?每个步骤的主要内容和测试方法是什么？**

**单元测试：单元模块内和模块之间的功能测试、容错测试、边界测试、约束测试、界面测试、重要的执行路径测试，单元内的业务流程和数据流程等。**

**集成测试：包括系统集成后的功能测试、业务流程测试、界面测试、重要的执行路径测试、容错测试、边界测试、约束测试及接口测试等。一次性集成及测试；增殖式集成与测试（自顶向下增殖测试、自底向上增殖测试、混合增殖测试）**

**有效性（确认）测试：包括系统性初始化测试、功能测试、用户需求确认、业务处理或数据处理测试、性能测试、安全性测试、安装性测试、压力测试、恢复测试等。**

**系统测试：恢复测试、安全测试、强度测试、性能测试。多任务测试、临界测试、中断测试和划分等价类**

**4.软件测试方法主要分为哪几类?每类方法又分别有哪些测试方法?**

**黑盒测试法：等价类分析法、等价类划分法、决策表测试**

**白盒测试法：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、路径覆盖、循环覆盖、模块接口测试**

**5.黑盒测试的分类(等价类、边界值等)**

**6.等价类和边界值测试用例的设计**

**等价分类法：划分等价类（有效等价类和无效等价类）、设计测试用例**

**边界值分析法：测试数据（最大值、最小值、最大值+1、最小值-1）**

**7.白盒测试的语句覆盖、分支覆盖、条件覆盖等**

**语句覆盖：每个语句执行一次。**

**判定覆盖：每个判定的真假分支都至少遍历一遍。**

**条件覆盖：每个判断的每个条件的可能值至少执行一次。**

**8.软件调试的任务是什么?有哪两部分组成**

**软件调试：是指使用调试工具修改或去除各种软件错误的过程，也是重现软件故障并定位其根源，并最终解决软件问题的过程。定位+纠错**

**9.软件调试的方法**

**强行排错（内存排错、特定语句排错、自动调试工具）：经常使用、方法简单、效率低**

**回溯法排错：在小程序中常用的一种有效排错方法，对较大程序回溯路径多、难度大，可 排查缩小范围后在使用**

**归纳排错法：以一些错误征兆为线索着手排查，通过分析各种关系归纳问题总结规律查找错误。搜集相关数据、组织数据并发现规律、提出假设获得进展、确认假设。**

**演绎法排错：是从一般原理或前提出发，经过排查精化过程推导出错误。列举假设、排除不正确假设、进一步排查定位、证明假设**

**10.软件维护的概念和分类(完善、适应、纠错、预防)**

**软件维护是在软件运行使用阶段对软件产品进行的调试和完善。**

**完善性维护：为满足用户新的功能或性能要求，扩充软件功能、增强性能、改善效率、提高可维护性，占维护工作量大50%。**

**适应性维护：指随着业务和技术的发展，外部环境或数据环境等更新变化，为了适应其变化而去修改软件的过程。25%**

**纠错性维护：对特定环境下暴露的一些问题或隐患，即是采取措施进行诊断识别和修改的过程。21%**

**预防性维护：提高软件功能及性能等。4%**

**1.项目管理的五个过程、九大知识领域,重要的几个领域的含义**

**项目管理的五个过程：启动，计划，执行，控制，收尾**

**九大知识领域：集成管理+（进城吃饭，疯狗踩人），时间管理（进程管理），成本管理，质量管理，范围管理，风险管理，沟通管理，采购管理，人员管理**

**2.进度估算的方法(定额估算法、经验公式法、CPM等**)

**定额估算法：T=Q/(S\*R) T是时间，Q是工作量，R是人力或设备的数量，S是生产效率**

**经验公式法：D=a\*E^b D是月进度，E是人月工作量，a为2-4的参数，b为1/3左右的参数**

**PERT与CPM：关键路径法是一种网络图方法。**

**3.进度管理的图形工具(甘特、网络、里程碑、资源等)**

**甘特图：可以清晰的表达每个任务活动的进展及并发情况。**

**网络图：网络分析方法编制的进度计划图。**

**里程碑：经过评审并确认后的可交付物才能成为里程碑。**

**4.关键路径的分析**

**路径与关键路径：在网络图中，从始点开始，按照每个活动的顺序，连续不断地到达终点的一条通路称为路径，时间最长的为关键路径。**

**5.质量管理、风险管理的概念**

**质量管理：质量计划+质量保证+质量控制**

**风险管理：指对软件项目可能出现的风险，进行识别、评估、预防、监控的过程。**