【第 1 章 软件与软件工程】

**软件**=计算机系统操作有关的程序、数据、文档的完整集合

**文档的概念和作用：**是与程序开发、维护和使用有关的图文材料记录软件开发活动和阶段性成果,为理解软件所必需的阐述性资料;促进对软件的开发、管理和维护;便于各种人员(用户,开发人员等)的交流,记录软件的配置及变更。**举例:**需求报告对所开发软件的功能、性能、用户界面及运行环境等作出详细的说明，它是用户与开发人员双方对软件需求取得共同理解基础上达成的协议，也是实施开发工作的基础。设计说明书是概要设计阶段的工作成果，它应说明功能分配、模块划分、程序的总体结构、输入输出以及接口设计、运行设计、数据结构设计和出错处理设计等，为详细设计奠定基础。详细设计说明书着重描述每一模块是怎样实现的，包括实现算法、逻辑流程等。

**软件的特点:**逻辑实体、智力产品;制造即拷贝;无磨损和老化,不遵循“浴盆曲线”,但存在退化问题。尚未摆脱手工方式;软件移植的需要;复杂（问题复杂性 / 程序结构复杂性）软件开发的性质如成本、进度、质量等难以估计控制,维护困难，有可复用性。

**软件工程的三要素:**过程:管理部分;方法:完成软件工程项目的技术手段;工具：自动或半自动地支持软件的开发、维护和管理。

**软件工程过程**：定义了方法使用的顺序,要求交付的文档资料,为保证质量和适应变化所需要的管理,软件开发各个阶段完成的里程碑.

**用开发过程说明三要素的关系:**（开发过程走了什么过程中用了瀑布模型、喷泉模型，在过程中用了方法面向对象、结构化方法，根据这些方法用什么工具在每个阶段做了什么，最后得出：要素之间相互关联和支持）软件工程方法（结构化、面向对象）为软件开发提供了“如何做”的技术。

▲**软件工程的目标** 1.总体目标:在规定的成本和期限内,完成满足用户需求的软件产品。2.具体目标:正确性,可靠性,有效性,可维护性,可重用性,可追踪性,可移植性,可互操作性。

▲**软件工程的原则**

1.**抽象**（过程抽象+数据抽象）（含义：抽取事物的最基本特性与行为，忽略非基本细节。**举例：**数理方程/软件的分层次。好处：可以控制软件开发过程的复杂性、有利于软件的可理解性和开发过程的管理抽象可以分多级）.(**过程抽象：**任何一个完成确定功能的操作序列，其使用者都可把它看作一个单一的实体， 尽管实际上它可能是由一系列更低级的操作完成.**数据抽象**:根据施加于数据之上的操作来定义数据 类型，并限定数据的值只能由这些操作来修改和观察。)

**抽象的原则：**自顶向下经历抽象级别从高到低的设计阶段，后阶段在前一阶段的基础上更接近软件的实现元素，又称为“逐步求精”。

**逐步求精**（全屏幕编辑器）：体系结构级,关心接口和职责（构件图，接口类图）->构件级,职责与协作(类图、交互图)->详细设计级,属性和操作（详细的类图.交互图）。

1. **信息隐藏**（含义：将模块中软件设计决策封装起来的技术。**举例：**功能模块的接口）3.**模块化**（含义：模块是程序中逻辑上相对独立的成分。**举例**：C语言的函数；OO语言的包/类/对象。好处：有助于信息隐藏和抽象）4.**局部化**（含义：在一个物理模块内集中逻辑上相互关联的计算资源0。**举例：**使用局部变量。好处：缩小软件元素的作用范围）5.**一致性**（含义：整个软件系统的概念/符号/术语一致；程序内部接口一致；软硬件接口一致；代码风格一致等等。）6.**完全性**（含义：软件系统完全实现所需功能，不遗漏的程度；当出错时，保持正常的能力。）7.**可验证性**（含义：软件系统应易于检查、测试和评审）。

**好处：**抽象、信息隐藏、模块化、局部化支持软件工程的可理解性、可修改性和可靠性，有助于提高软件产品的质量和开发效率。

使用一致性、完全性、可验证性的原则可以帮助人们实现一个正确的系统。

**举例：（1抽象 2模块化 3信息隐藏 4局部化）**

结构化：1.结构体.函数 2.函数.子程序，文件系统 3.局部变量 4.局部变量

OO：1.类2.包.类3封装，类的定义将其说明与实现显式的分开，对象与对象之间通过消息机制得以实现4.对象作为独立存在的实体,将自由数据和操作封闭在一起,使自身的状态、行为局部化，类以及类中的成员属性

**局部化和模块化的关系：**局部化间接实现模块化。模块化需要局部化

**简述以下概念,并阐述其重要性:可靠性,可理解性,可重用性,可修改性及其价值**

**可靠性:**具有能够防止因概念、设计和结构等方面的不完善而造成的系统失效，具有挽回因操作不当而造成软件系统失效的能力;在某个给定时间间隔内，程序按照规约成功运行的概率。**价值**：其为度量软件质量的重要内容、某些重要的计算机应用系统上软件可靠性关乎整个系统的成败。**可重用性:**一个软构件可以在多个场合应用的程度称为构件的可重用性。提高开发效率、降低软件的开发和维护费用。**可修改性**：软件交付后，能对其修改，使之适应新需求、新环境的能力。可修改性越好，维护人员理解成本就越低，维护费用就越低。

**生命周期:1**可行性研究，2需求分析,3概要设计,4详细设计,5实现,6集成测试,7确认测试,8使用与维护,9退役(软件定义:1-2软件开发3-7 维护8-9)。**对应关系:**需求分析对应确认测试，概要设计对应集成测试，详细设计对应单元测试。

**简述各生命周期的主要任务（后面作业题有剩余的）:**

**1.**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 **成果**可行性论证报告 初步的项目开发计划**7**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 **成果**可供用户使用的软件产确认测试报告 **8**对使用后的软件进行**成果**维护版本更新的软件产品维护记录

**各种模型的含义、优缺点、场景:**

**瀑布模型:**假设软件需求在初期几乎可完全确定;**主要思想:**1.软件开发过程与软件生命周期是一致的;2.相邻二阶段之间存在因果关系3.每一阶段的结束点作为里程碑;4.需对阶段性产品进行评审，如果评审不合格,需返工;**优点:**1.使用时间最长、应用面比较广泛;2.是其他一些开发模型的基础;3.思路简洁、明确，阶段间因果明确，紧密联系;4.其可行性研究、需求、设计、编码、测试分离,有利于软件的体系结构设计,规范了软件开发活动,有利于开发人员的组织管理;5.缓解软件危机**缺点:**1.必须确定软件需求后才能进行后续开发工作,但多数场合给出全部需求是困难的;2.到最后阶段才能得到可 运行的软件版本 不能适应用户需求的变 化 ;3.上游阶段的“过失”会为软件制品带来“缺陷”并潜伏在制品中,会误导下游的开发活动,若未被发现运行时会造成“故障”,寻找修复故障会比较困难**场景:**对于规模较小、软件需求比较稳定的项目或子系统,能够显著提高软件开发的质量和效率;**原型模型:概念:**软件开发人员根据客户提出的软件(部分或全部)定义,快速地开发一个原型。原型向客户展示了待开发软件系统的全部或部分功能和性能,在征求客户对原型意见的过程中,进一步修改、完善、确认软件系统的需求并达到一致的理解;**优点**：1.支持需求的动态变化;2.有助于获取用户需求,便于用户对需求的理解;3.尽早发现软件中的错误;**缺点:**不支持风险分析.**适用场景**：开发团队对所开发领域比较熟悉而且有快速的原型开发工具**螺旋模型:** **基本思想:**1.将瀑布模型与原型模型进行有机结合;2.增加风险分析步骤。**优点:**1.支持需求的动态变化;2.有助于获取用户需求,便于用户对需求的理解;3.尽早发现软件中的错误;4.支持风险分析,可降低或者尽早消除软件开发风险;5.适合于需求动态变化、开发风险较大的系统;**缺点:**由于确定的不确定性,软件开发初期无法进行软件体系结构设计,多次迭代会导致软件体系结构变坏,为软件理解和维护带来困难;**场景:**在需求不明确的情况下,适合用螺旋模型进行开发,便于风险控制和需求变更;特别适合于大型复杂的系统。**原型模型和螺旋模型既是迭代模型,又是进化模型。**

**工具：** **项目管理工具**AMS Realtime,Microsoft Project, Viewpoint,  Project

Control Panel  ,Ittoolkit   
**软件开发工具：业务系统建模工具：**Rational Rose,Argo UML ,Visio ,Real-Time Studo(以上也可以是对应的**分析和设计工具**）**测试工具：**Code Medic(为标准UNIX调试器gdb提供图形界面，支持C/C++、Java、汇编、FORTRAN、Modua-2测试)   Bug Collector Pro（实现多用户数据库，支持软件团队追踪已报告的bug、管理调试工作流）   Jprobe Thread Analyzer（支持线程测试问题：死锁、延迟）   C++ Test（C／C++单元测试）**原型建造工具（专用性）用户界面工具:**Macromedia

Authorware,PowerDesigner/PowerBuilder,Motif Common Desktop Environment

**【第 2 章 UML 与 RUP 统一过程】**

▲**面向对象的软件开发方法**

面向对象的软件开发方法通过提供对象、对象间消息传递等语言机制让分析人员在解空间中直接模拟问题空间中的对象及其行为，从而削减了语义断层，拉近了问题空间与解空间的距离，从而简化了软件工程师在二者之间架设“桥梁”的工作，并为软件开发活动提供了直观、自然的语言支持和方法学指导。

**面向对象 = 对象+类+继承+聚合+多态+消息**

▲**面向对象方法的优势** 1.简化软件开发过程。面向对象方法不仅可以用来进行需求分析，还可以支持软件 的设计、实现和测试，构成了覆盖软件开发主要阶段的广谱软件开发方法学。 2.支持软件复用。在源代码级复用方面，面向对象方法通过继承机制和代理方法使 得复用者不需要直接修改被复用的类；在设计级复用方面，近年来迅速发展的设计 技术在软件界大显身手，贡献良多。 3.改善软件结构。面向对象方法通过对属性和操作的封装实现了软件工程倡导的信息隐藏原则。

▲**UML：用例视图：**用例图，从外部用户的角度描述系统功能，并指出参与者

**结构视图**（分别从不同层面表示系统的静态结构）：包图（描述系统的分解结构，表示包与包之间的关系）、类图（描述系统静态结构）、对象图（类图的一个实例，描述在某个状态下，或者某个时间段系统中活跃的对象及关系）

**行为视图**（从不同的侧面刻画系统的动态行为）：交互图［描述对象之间通过消息传递进行的交互和协作，分为顺序图（对象之间消息发送的时间序）、通信图（对象间的动态协作关系，也可通过消息序号表示消息传递的时间序，不如顺序图直观）］、状态图（描述类的对象的动态行为）、活动图［控制流（一个操作完成后对其后续操作的触发）、信息流（刻画操作间的信息交换）］

**构件视图：**构件图(描述软件系统中各组成构件,构件的内部结构以及构件间依赖关系)

**部署视图**：部署图（描述软件系统中各类工件在物理运行环境中的分布情况）

**▲用例图中的关系**

**1.关联：**参与者---执行用例。**2.包含(include)**:一个用例A使用了另外一个用例B。A- ->B。边上include。**3.扩展(extend)**:A扩展自B。A- ->B。边上extend。**4.继承**：例如制定优异生选课计划继承自制定选课计划（一般不用）

**▲类图中的关系**

①**泛化与特化**：表示类的继承关系。子类父类。

②**关联**：类之间有关联（对象组合）A类用到了B类的信息A->B。一对多的时候，边上：1—\*。双向关联没有箭头，单向关联有箭头，我们一般用单向。

③**聚合**：强关联，整体与部分的关系。EngineCar。一对多：1-\*。

④**组合**：强聚合，整体的对象负责代表部分的对象的生命周期。LimbMonkeyKing

⑤**依赖**：ClassA的定义依赖于ClassB的定义。例子：A的构造函数的参数中有ClassB。A中有个函数以B作为参数。A - ->B。

**【第6章 软件设计基本原则】**

·**模块的设计原则**：高内聚低耦合，信息隐藏。

▲**内聚：**偶然性内聚、逻辑性内聚、时间性内聚（经典内聚）->过程性内聚、通信性内聚->信息内聚、功能性内聚。

▲**耦合**：1非直接耦合->2数据耦合与控制耦合->3外部耦合->4公共耦合->5内容耦合。尽量使用 1/2，限制3/4，杜绝5。

**启发式设计策略：**改造软件结构，降低耦合度，提高内聚度；减少扇出，追求高扇入；

使任一模块的作用域在其控制域内(**作用域**是指受模块内部判定影响的所有模块,**控制域**是指其所有的下属模块);降低模块接口复杂度和冗余度,提高协调性。

**变换流：**信息通常以“外部世界”所具有的形式进入系统，经过处理后，又以这种形式离开系统，呈现在结果界面上。**事务流**数据沿输入通道到达一个转换，该转换根据输入流的类型和特征在若干动作路径中选择一条来执行。

变换分析：复审基本系统模型；理解和精化数据流图；确定数据流图的类型；划分输入流、输出流的边界；执行二级分解；精化、改良软件结构。

**选三种内聚度、三种耦合度各举一个例子说明。**

**功能内聚：**一个模块中各个部分都是完成某一具体功能必不可少的组成部分，或该模块中所有部分都是为了完成一项具体功能而协同工作，紧密联系，不可分割。（不建议举此例）**信息内聚：**完成多个功能各个功能都在同一数据结构上操作，每一项功能有一个唯一的入口点。如课程管理系统上的查找课程、删除课程、修改课程都会使用“课程”这个数据结构。**通信内聚：**一个模块各个功能部分都使用了相同的输入数据，或产生了相同的输出数据。个人信息管理系统中：从文件中读取信息，打印功能和统计功能都会使用文件中的信息。**逻辑内聚：**这种模块把几种相关的功能组合在一起，每次被调用时，由传送给模块的判定参数来确定该模块应执行哪一种功能。

**非直接耦合：**两个模块之间没有直接关系，它们之间的练习完全时通过主模块的控制和调用来实现的。main()函数调用A B C函数，其中A用到B的返回值。**控制耦合：**通过传送开关、标志、名字等控制信息，明显的控制选择另一模块的功能。如A的返回值为0时调用C，A的返回值为1时调用B。**公共耦合：**一组模块都访问同一个公共数据环境。如A\B都调用以局部变量time,并且会对其修改。

**如何降低耦合度**：根据问题的特点，选择适当的耦合类型；降低模块接口的复杂性(传送信息的数量,耦合方式,传送信息的结构)；把模块的通信信息放在缓冲区中;

**软件设计的原则：**提高模块独立性；调整模块的大小；适当的模块深度、宽度、扇出和扇入模块的作用范围应在控制范围之内简化模块的接口的复杂程度。

**【第八章、人机交互设计】**

**·重点题型：针对特殊人群，设计有针对性的GUI。**

老人：图片字体大且清晰，描述简单，目录层级少，打开步骤少，操作简单，实用性好，功能不复杂、只需要基本功能

小孩：色彩丰富。避免繁杂，大量图片和文字作为提示，风格轻松愉快，宽松简洁的布局，高明度色彩，带拼音，可爱的卡通形象和字体设计，趣味和有故事性的背景。

·**用户界面设计的基本原则**①易理解性；②易操作性；③灵敏性；④一致性；⑤容错性；⑥人性化。

·**用户界面设计模型的表示**：①静态元素（无变化的文本、图标、图形、图像等）；②动态元素（因运行状态而异自动呈现的不允许修改的文本、表格、图标、图形、图像等）；③用户输入元素（可编辑文本、单选按钮、多选框、选择列表、可编辑的表格）；④用户命令元素（按钮、菜单、超链接等）。

·**用户界面设计过程的主要活动:** 1**.用户分析、任务分析及建模**。首先必须分析用户的特征，分析用户需要通过目标软件系统完成哪些工作任务，为完成这些任务人机之间需要进行哪些信息交互。2**.概念设计**。主要目标是确定屏幕中应该包含的主要内容，以及用户基于该屏幕可施行的主要操作。3.**界面流设计**。主要目标是确定屏幕之间的跳转关系，即一幅屏幕在何种情况下,或者在响应何种用户操作命令后将跳转至另一屏幕。两种原因：①单屏幕空间容量有限，不足以表现所有必要界面元素；②用户的界面操作可能导出新的屏幕，以便在新的屏幕上进行面向特定业务功能的界面交互。**表示方式主要是UML交互图（顺序图）和类图**。4.界面精化。主要任务是基于概念设计和界面流设计，给出目标软件产品界面的完整的、详细的设计。

**【第九章、软件详细设计】**

·详细设计模型的质量要求1.正确性。模型中若干设计元素通过模型指定的协作方式能够实现所有的软件需求。2.优化性。模型以充分优化的方式实现所有的软件需求。3.设计充分性。模型的细化和精确程度足以作为软件编程人员的全部工作基础，没有含混、笼统和歧义之处。

·详细设计过程的主要活动①用例设计；②子系统设计；③构件设计；④类设计；⑤数据模型设计；⑥设计整合与验证。

**【第十章、软件实现】**

**·程序设计语言的演化**：第一代：汇编；第二代：Fortran/Cobol/Basic，有成熟的函数库；第三代：结构化语言C/Pascal/Ada；面向对象语言C++/Java；专用领域语言Lisp/Prolog；第四代：数据库查询语言Sql。

·第三代语言的特点：支持结构化程序设计，具有较强的过程能力和数据结构能力。第四代特点：更高的抽象层次；只需要告诉计算机“做什么”，而不必告知“怎么做”。

·编程的规则：1.节俭化（代码尽量简洁）；2.模块化；3.简单化（命名一致性，数据结构尽量简单）4.结构化（缩进统一/函数代替重复出现的代码/代码单入口单出口）5.文档化（文档自说明）6.格式化

**·编程风格如何影响软件质量？**

编程风格在很大程度上影响着程序的可读性、可测试性、可维护性。

结构化要做到按标准化的次序说明数据，按字母顺序说明对象名，使用容易理解的结构化程序部件，根据背景排列程序各部分。

·调试策略：1.原始法（单步执行，printf）2.回溯法（沿着代码控制流往回找）3.排除法。

**【第十二章、软件测试】**

**·考判断题，含解释说明。**

**·区分概念：语句覆盖/判定/条件/判定+条件/修正的判定+条件/路径**

**·比较级：语句<判定/条件/非原子性路径<判定条件组合/原子性路径（判定与条件不可比）**

**·流程图与流图的转换；**

**·V(G)=区域数=谓词节点数+1=基本路径数=E-N+2**

**·测试的级别：单元、集成、系统、确认**

**·**软测定义：测试是使用人工的或自动的手段来运行或检测某个系统的过程,其目的在于检验它是否满足约定的需求或是比较预期结果与实际结果之间的差别。

·测试的目的：①发现软件中的错误，提高软件质量；②好的测试方案是尽可能发现迄今为止尚未发现的错误的测试方案；③成功的测试是发现至今为止尚未发现的错误的测试。④软件测试是发现软件错误的过程，测试只能查找出程序中的错误，而不能证明程序没有错误。

**【第十三章、软件维护】**

**·维护的分类(**按原因划分):1.纠错性维护。为诊断和改正软件系统中潜藏的缺陷而进行的活动.2.适应性维护.为适应软件运行环境变化(如操作系统变更等)而修改软件的活动.3.完善性维护.是根据用户在软件使用过程中提出的一些新需求而实施的维护活动.4.预防性维护。是优化软件系统结构和可理解性，改善可维护性和可靠性。

·**可维护性**：软件的可维护性是指理解.改正.调整和改进软件的难易程度。是软件质量的重要属性，是指导软件工程活动的一条基本原则，也是软件工程追求的一个目标。

·**影响可维护性的因素**：开发方法有关：设计、编码和测试不规范，软件配置不全开发环境有关：①训练有素的软件团队；②维护团队是否熟悉经过多次维护的系统；③软件的可理解性，包括软件结构、描述软件制品的语言、文档及标准化程度等；④操作系统的标准化程度；⑤维护工具和环境；⑥生成测试用例的能力；⑦对于嵌入式系统维护应有专门的调试工具。

·**维护的技术手段：**软件重构。包括：文档重构、重组（简化与结构化源代码）、逆向工程、再工程。

**【第十五章、软件度量与估算】**

**·**测量（Measure）是按照统一的规则为现实世界的实体属性定值。软件工程的测量是按照测量标准直接、客观地采集软件制品、过程或资源的特征、属性，并获得数据。例如：测量程序的代码行数、操作符的种类、个数，程序中缺陷的个数等等。测量涉及测量对象、选用的量纲、方法、工具、过程和数据结果。

·软件估算（Estimation）是根据经验、历史资料或模型，结合项目实际对软件 制品、过程、资源进行预测。估算一般用于签订合同、制定工作计划、进行项目预算等。这里涉及软件过程工作量的估算。

·软件度量（Metrics）软件产品、软件开发过程或资源简单属性的定量描述，（如程序规模、操作数个数、占用内存）能用以解释软件所具有的一个给定属性对软件质量影响的程度。

**·软件度量：规模度量、复杂性度量（McCabe基于图论；Halstead基于操作符与操作数）、质量度量、可靠性度量。**

**·内部属性与外部属性：软件工程的软件制品、过程、资源都具有外部属性和内部属性。外部属性体现了软件制品、过程、资源与环境的关系。内部属性指软件制品、过程和资源本身的技术属性。**

**二者关系：外部属性在软件开发过程中很难测量和控制，但它是由 软件的内部属性决定的。**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 软件制品 | 过程 | 资源 |
| 内部 | 程序语言；代码风格；模块化；复用性；耦合度与内聚度 | 项目管理过程；业务建模过程；需求过程；设计过程； 构造过程；测试过程；部署过程；配置管理过程；工具与环境支持过程 | 人；软硬件环境；经验 |
| 外部 | 软件系统的可靠性；健壮性；效率；可用性；可维护性；可移植性 | 资源保障；可控制性；可观 察性；稳定性 | 成本；时间；合作机制 |

**▲软件规模的度量**

**（一）代码行度量**（直接度量，依赖于程序设计语言）

·含义：通过项目总代码行数去度量项目的规模属性。（书本原话：用代码行数（LOC）表示软件项目的规模是自然和直观的。可以用人工或工具直接测 量。几乎所有软件开发组织都保存软件项目的代码行数记录，开发初期可以度量软 件规模，度量生产率、每行代码的平均人力成本、文档和代码的比例关系、每千行代 码存在的软件缺陷个数等。）  
·优点：简单直观，容易操作。  
·缺点：1.依赖于特定程序设计语言；2.可能对一些设计精巧的软件项目不利；3.项目开发前期估算困难；4.适用于过程式程序设计语言，对非过程程序设计语言不太适用；

**（二）基于功能点度量**（间接度量，不依赖于语言）

**·**含义：根据事务信息处理程序的基本功能定义的，采用5个测量参数，涉及多种因素的间接度量方式。（因而可以估算出软件项目的规模）

·优点：1.与程序设计语言无关，适用于过程式和非过程式程序设计语言； 2.度量可以用于软件开发的初期。（因为初期即能确定系统的输入/输出）

·缺点：1.涉及主观因素较多，如加权值；2.信息领域数据有时不容易采集；3.FP(功能点)的值没有直观物理意义。

**两者联系：抽象度越高的语言，同样的功能点对应的码会越少。**

**▲软件质量度量**

**·软件质量的定义**：软件制品满足规约和隐含需求特征及特征总和的程度。特征包括：功能性，可靠性，有效性，可使用性，可维护性，可移植性。·2011：增加了安全性和兼容性的两个特征，扩展了子特征的概念。 软件特征的变化反映了软件与时俱进的特点，为软件质量度量和通过技术手段提高 软件制品质量指明了方向。

·**McCall三层次度量模型**：质量要素指标（Factor）、评价准则（Criteria）、属性度量(Metric)。Factor包括：运行性、修正性、适应性。评价准则21个。

**所有的质量度量模型（McCall、ISO）是层次性模型，不同在于分的哪些层次,每一层有哪些方面。**

**【第十六章、软件项目管理与过程改进】**

·目的：软件项目管理的目的是使项目能够按照预定的成本、 进度、质量顺利完成。

·一般经验估算模型：基于代码行的模型 E=a(KLOC)b+c；基于功能点的模型 E=a+b\*FP，不同模型采用不同参数

**▲软件项目度量与估算**

**（一）采用代码行/功能点度量的工作量估算**

·E=(a+4m+b)/6: 采用上述估算方法可估算出LOC/FP的乐观值a，悲观值b，一般值m。**（表格16.5的几列数据要会算）**

·一般经验估算模型：基于代码行用E=a(KLOC)^b+c,abc查表；E是人月工作量；基于功能点用E=a+b\*FP。ab查表。

**（二）COCOMO模型（Constructive Cost Model,构造性成本模型）**

·COCOMO模型分为基本、中间、详细三个层次。

·通信开销=u\*n\*(n-1)/2

·基本COCOMO模型：用于系统开发初期，估算系统开发工作量和所需要的时间。E=a(L)^b;D=c(E)^d;abcd查表；E是人月工作量；D是开发时间。

N=E/D是建议参加项目的人数。

·中间COCOMO模型：用于详细设计阶段，估算各个子系统的工作量和开发时间。E=a(L)^b\*EAF;L是代码行估算值，单位kloc；ab查表。EAF是工作量调节因子，EAF=（15个Fi连乘）

·详细COCOMO模型：用于估算独立的软构件，如子系统内部的各个模块，软件的集成和测试。

（三）COCOMO Ⅱ

**·**E=a(L)^b\*EAF+ER。ER表示复用构件、代码自动生成需要的工作量，单位人月。

**（四）Putnam模型**

**·时间推前或延迟k%对工作量的影响？**

·L=C\*E^(1/3)\*t^(4/3).L是代码行数；E是工作量，单位人年，包括维护；t是开发时间；C表示软件开发环境的常数，查表。

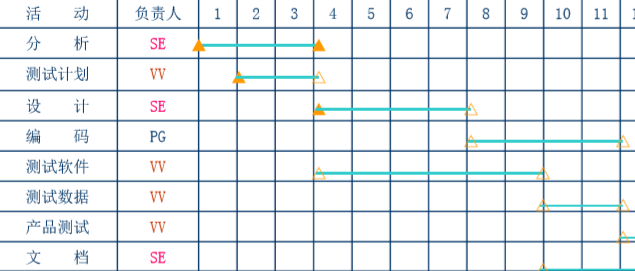
**·E=L^3/(C^3\*t^4)。比如提前10%的时间（t=0.9t）,算出工作量E=1.524E。说明要增加52%的工作量。**

·Putnam模型虽然揭示了软件项目的工作量、软件开发时间和程序代码长度三者之间的关系，但它没有反映软件制品、软件项目、软件开发人员和计算机软硬件资源等属性。因此用该模型进行软件项目的成本估算是十分粗糙的。

**▲软件项目计划----工程进度安排**

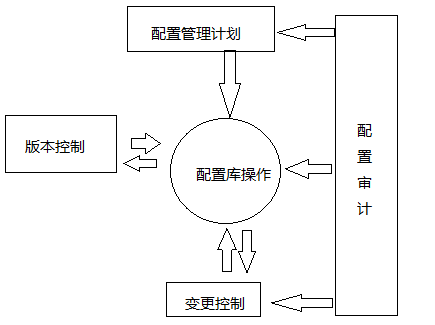
**·两个重要的图：网络图（找关键路径，非关键路径的点可调控）；甘特图（网络图=>甘特图）**

**Gantt图优点：**简单易用，容易修改，比PERT图更加直观方便。

**Gantt图缺点：**不能显示各项活动之间的依赖关系

**▲软件配置管理**

·配置管理的目的：目的是为了减少混乱，提高软件生产率。是对软件修改进行标识、组织和控制的技术，用来协调和控制整个系统的过程。

**·基线技术：**标志软件开发过程的各个里程碑，通过复审的**软件配置项**(SCI, Software Configuration Item, SCI是配置管理的基本单位)是构成基线的重要内容，它标志开发过程中一个阶段的结束。

**·基线的使用**：1.某SCI成为基线，就会被存入项目数据库；2.要想改动SCI，就要复制到私有工作区并在项目数据库中锁住。3.在私有工作区完成修改并通过复审后，新SCI释放并送回项目数据库，并解锁。

**·常用基线：**1.系统工程--系统规约；2.需求分析—软件需求规约；3.软件设计—设计规约；4.编码—源代码；5.测试—测试计划、过程、数据；6.集成—可运行系统。

·**常用工具**：版本控制git；DSEE。

·**DSEE**：历史管理、配置管理、任务管理、监控管理。

·**SCI**：（1）系统规约；（2）软件项目规划；（3）需求分析结果；a.软件需求规约；b.可执行的或“纸样”原型；（ 4）初步用户手册；（5）设计规约；a.数据设计描述；b.总体 结构设计描述；c.模块设计描述；d.界面设计描述；e.对象描述；（6）源代码清单；（7）测试规约；a.测试计划和过程；b.测试用例和实验结果；（8）操作和安装手册；（9）可执行程序；a.每个模块的可执行代码；b.链接到一起的代码；（10）数据库描述；a.数据模型和文件结构；b.初始化映像；（11）联机用户手册；（12）维护文档； a.软件问题报告；b.维护申请；c.预计变更的顺序；（13）软件工程的标准和过程。

**配置管理的任务**：标识配置项；版本管理。

**配置项的状态有三种**：“草稿”（Draft）、“正式发布”（Released）和“正在修改”(Changing)。**变迁**：配置项刚建立时其状态为“草稿”。配置项通过评审（或审批)后，其状态变为“正式发布”。此后若更改配置项，必须依照“变更控制规程”执行，其状态变为“正在修改”。当配置项修改完毕并重新通过评审（或审批）时，其状态又变为“正式发布”，如此循环.

**▲软件过程改进**

**·什么是C**MM**?** Capability Maturity Model for Software，软件成熟度模型。CMM是一个概念模型，模型框架和表示是刚性的，不能随意改变。但模型的解释和实现有一定弹性。CMM 源于大型软件开发实践，反映了 软件过程评估和软件过程改进的需要，是一个有效的大型软件开发、维护过程模型。 CMM 的应用进一步规范、指导软件开发组织的自身建设，使软件开发组织从混乱的、低效的不成熟状态，向有纪律的、高效的成熟状态转变。

·**关键过程域KPA**：描述软件的过程属性，通过完成一组相互关联的活动，实现**建立过程能力**至关重要的一组目标。（软件开发组织的软件过程能力是组织能够承接软件项目的重要依据）

·CMM的能力成熟度共分5级：L1初始级；L2可重复级；L3已定义级；L4 已管理级；L5优化级。从2级开始，每级包括若干个KPA。

·CMM评估阶梯：L1初始级；L2（软件项目计划+软件项目跟踪与监督）；L3（综合软件管理）；L4（定量的过程管理）；L5（过程变更管理）。

**·什么是C**MM**I？**CMMI是若干过程模型的综合和改进，是支持多个工程学科和领域的、系统的、一致的过程改进框架，能够适应现代工程的特点和需要，能够提高过程的质量和工作效率。

·**CMMI等级：**L0不完全级；L1已执行级；L2已管理级；L3已定义级；L4定量管理级；L5优化级。

【简答题】

**OO方法与结构化方法有何不同？这两种方法有哪些方面是相同的？**

**相同：**都是软件系统的开发方法、运用分解和抽象原则上的要求一致,抽象通过分解体现、代码与数据的分隔化;**不同：**结构化方法采用自顶向下、分层解决的分解原则，OO 则是封装也称为信息隐藏。面向对象程序的基本元素不再是函数，而是对象；程序的组成不再是函数的集合，而是对象的集合及其交互。结构化方法编写程序思路清晰、条理严谨便于阅读理解。OO注重功能的用户使用方法，操作简单，界面清楚，软件使用简单。结构化方法对复杂问题帮助有限。OO面对信息隐藏的和封装封闭的内部细节，很难得到高度可复用的软件，总是试图抽象出更公用的类，难以控制抽象

**传统的结构化开发方法已经非常成熟，为什么还会有OO开发方法？**

传统的结构化开发只有借助相当复杂的方法操纵解空间中的对象才能得到问题的软件解，这就是所谓的语义断层。面向对象的软件开发方法通过提供对象、对象间消息传递等语言机制让分析人员在解空间中直接模拟问题空间中的对象及其行为，从而削减了语义断层，拉近了问题空间与解空间的距离，从而简化了软件工程师在二者之间架设“桥梁”的工作，并为软件开发活动提供了直观、自然的语言支持和方法学指导。此外还有以下优势：1.简化软件开发过程 2.支持软件复用 3.改善软件结构

**面向对象（OO）方法体现了经典软件工程中抽象、模块化、重用等重要思想，请说明。抽象：**OO中的类体现了软件工程中的抽象原则。类的定义中指出类是一组具有内部状态和运动规律对象的抽象。**模块化：**OO的封装特性体现了软件工程中的模块化原则。类是封装的基本单位，类的定义将其说明与实现显式的分开。对象与对象之间通过消息机制得以实现，从而实现模块化**信息隐藏：**OO中的模块接口和类的私有成员体现了软件工程中的信息隐藏原则。模块接口使得外部只能通过接口调用模块。类通过私有成员的方式来隐藏对象的属性和实现细节，从而实现信息隐藏。**重用：**OO中通过继承和多态来实现重用。继承可以使子类沿用父类的某些特征，多态可以使某一相同接口具有不同功能。**局部化：**定义局部变量之类的。

**结构化的软件开发方法如何在开发的各阶段体现“自顶向下，逐步求精”的思想**

分为结构化分析和结构化设计;结构化分析是一种着眼于数据流，自顶向下，逐层分解，建立系统的处理流程，以数据流图和数据字典为主要工 具，建立系统的逻辑模型。结构化设计细化数据流图，确定数据流图的类型。之后把数据流图映射到软件的模块结构，分解出高层模块，中下层模块。最后对模块结构进行优化，得到更为合理的软件结构，并且设计模块的接口。设计第额段分为

**模块独立性与信息隐蔽分别是什么含义，二者有何关系？**

独立性：指模块内部各部分及模块间的关系的一种衡量标准。耦合：模块间的关联紧密程度，内聚：模块内各成分间彼此结合的紧密程度;信息隐蔽：将模块中的软件设计决策封装起来的技术。模块应该设计得使其所含的信息（过程和数据）对那些不需要这些信息的模块不可访问，模块之间仅仅交换那些为完成系统功能所必须交换的信息。信息隐蔽会导致模块的高内聚和低耦合，因此就会使模块的独立性很强。

**什么是抽象.抽象可分为过程抽象、数据抽象请举例说明这两种抽象，并说明其好处。**

int Compare (CString, CString) 计算字符串的长度: int GetLength(CString )（2）数据抽象（表示）将诸多数据对象的定义（描述）抽象为一个数据类型名，以后可通过该数据类型名来定义多个具有相同性质的数据对象Eg: 1, 2, 3,—>Integer 软件工程书，人工智能书， —>书 类 。

**逐步求精、分层结构与抽象三者之间的相互关系如何？**

逐步求精是一种设计策略，将软件的体系结构自顶向下，对各个层次的过程细节和数据逐步细化，从而确定最后的 体系结构。流体系结构就是一种分层结构，在实施时，采用的是抽象化的方法，自顶向下，给出不同的抽象层次。

**局部化是基本的软件工程原理，以高级程序设计编程为例，说明是如何遵从这一原理的。**

局部化就是指将一些使用上密切相关的元素尽可能放在一起。对一个模块来说，局部化是期望模块所使用的数据尽可能是在模块内部定义的。

举例子：在编程实现时尽量每个模块独立性,每个模块的参数不进行交换,模块与模块之间只涉及到应该传递的参数传递;(随便写一段代码，涉及局部变量，说明一下即可)

**人机界面设计对软件系统有何影响？如何设计好的人机界面？**

方便软件系统的操作，简化使用难度，有助于发现软件需求的新来源，有助于基于用户特征做出正确的需求工程决策，甚至可帮助需求工程师推导出新的软件需求项。

易理解性；易操作性；灵敏性；一致性；容错性；人性化。

**软件测试要经过哪些步骤？这些测试与软件开发各阶段之间有什么关系?**

单元测试，集成测试，确认测试，系统测试;

前三个对应开发：详细设计，概要设计，需求分析

**黑盒测试中的等价分类法和边界值分析法有什么区别和联系？**

等价类：把程序的输入数据集合按输入条件划分为若干个等价类，每一等价类的相对于输入条件表示为一组有效或无效的输入，然后为每一等价类设计一个测试用例，这样既可大大减小测试的次数又不丢失发现缺陷的机会;

边界值：是对等价分类技术的补充，即在一个等价类中不是任选一个元素作为此等价类的代表进行测试，而是选择此等价类边界上的值。此外，采用BVA技术导出测试用例时，不仅要考虑输入条件，还要考虑输出的状态。

**如何保证软件的可维护性？**

1 训练有素的软件团队 2 维护团队是否熟悉经过多次维护的系统 3 软件的可理解性 4 操作系统的标 准化程度 5 维 护工具和环境 6 生成测试用例的能力

1软件和文档的可理解性 2文档是否齐全 3软件开发方法学----结构化、 OO 4软件的结构是否合理（结构的可扩展 性） 5 开发人员素质 6 是否使用标准的 OS，程序设计语言，文档结构

**据统计软件维护工作量高达软件整个生存周期工作量的70%，请说明有哪几类维护，分别对应哪些方面的修改？**

1. 纠错性维护。为诊断和改正软件系统中潜藏的缺陷而进行的活动。 2.适应性维护。为适应软件运行环境变化（如 操作系统变更等）而修改软件的活动。 3.完善性维护。 是根据用户在软件使用过程中提出的一些新需求而实施的 维护活动。 4.预防性维护。 是优化软件系统结构和可理解性，改善可维护性和可靠性。

**面向规模的度量与面向功能的度量有何不同？试比较各自的优缺点。**

代码行：优点：简单直观易行。缺点：依赖于程序设计语言，它们对设计精巧和复杂的软件项目会产生不利的评判，不适合于非过程语言，在估算时需要一些可能难以得到的信息。功能点：优点：和程序设计语言无关；基于项目开发初期就可能得到的数据。缺点：计算是基于主观的而非客观的数据；信息域（及其它维）的计算可能难以搜集事后信息；FP值没有直观的物理意义

**简述CMM（软件能力成熟度模型）每个等级的含义。**

五个等级:(1)初始级(initial)。工作无序,项目进行过程中常放弃当初的计划。管理无章法,缺乏健全的管理制度。开发项目成效不稳定,项目成功主要依靠项目负责人的经验和能力,他一但离去,工作秩序面目全非。(2)可重复级(Repeatable)。需求管理软件项目计划项目跟踪和监督分包合同管理软件质量保证软件配置管理(3)已定义级(Deﬁned)。开发过程,包括技术工作和管理工作,均已实现标准化、文档化。建立了完善的培训制度和专家评审制度,全部技术活动和管理活动均可控制,对项目进行中的过程、岗位和职责均有共同的理解。(4)已管理级(Managed)。产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势,如预测偏差,实现及时纠正。(5)优化级(Optimizing)。可集中精力改进过程,采用新技术、新方法。拥有防止出现缺陷、识别薄弱环节以及加以改进的手段。可取得过程有效性的统计数据,并可据进行分析,从而得出最佳方法。

**请以一次软件开发为例，阐述一下在开发的各个阶段过程、方法和工具是如何相互支撑的。**

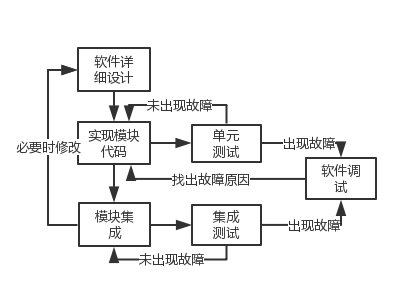
可行性研究与计划阶段：制定初步的项目开发计划和可行性论报告;需求分析阶段：使用DFD图来描述需求分析，CASE：建模(数据流图, 数据字典, 小说明) 、模型的存储, 显示和检索;设计阶段：使用uml图进行概要设计。根据概要设计结果进行算法和数据结构设计。模块化、局部化、信息隐藏、 单入口单出口原则 。CASE：RationalRose;编程阶段：面向过程/对象的程序设计 使用C、C++、JAVA等工具 根据详细设计规格说明书编写程序;测试阶段：黑盒测试，白盒测试(基路径覆盖法)，CASE：代码审查器、断言处理器、测试用例生成器

**软件工程从根本上解决软件危机了吗？试说明理由。**

开发软件所需的高成本同产品的低质量之间有着尖锐的矛盾，这种现象就是所谓的“软件危机”。软件工程是用工程、科学和数学的原则与方法研制、维护计算机软件的有关技术及管理方法。软件工程的诞生只能大大减少软件开发成本并提高软件质量，不能从根本上消除软件危机。

【作业题】

**简述软件工程全生命周期中各阶段的主要任务、完成途径、成果。**

**需求任务：任务**：通过对应用问题及其环境的理解和分析准确、一致和完全地刻画用户需求，形成软件需求规格说明书。包括：需求获取、建模、验证、管理。**需求完成途径：**获取：调查、研讨 建模实现：OO:UCD+Act D+CD+SD Structured:DFD、DD。**需求成果：**需求规格说明书、需求分析说明书、相关工作分析、验证结果说明。**需求补充：**需求分为功能性和非功能性。**设计任务：**定义：针对需求给出的需求模型，考虑各种制约因素，探索切实可行的软件解决方案并最终给出方案的逻辑表示。**任务：**以SRS为基础，设计出满足用户需求的软件(软件结构，算法，数据结构)，给出软件实现的解决方案，生成软件设计的规格说明书。包括体系结构设计、UI设计、数据库设计、设计验证**。概要设计任务：**设计软件的整体实现框架，定义模块划分 以及模块之间的关系**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)**设计途径：**利用抽象、局部求精、模块化、信息隐藏等原则。**体系结构设计**：包图、构件图。**UI：**交互图（主要是顺序图）、类图（表示静态逻辑关系）**概要设计：**层次图、结构图、HIPO（IO,processing）图 **详细设计：**用例图、类图、状态图、活动图。**总结OO：**Act D,CD,SD Structured: Flow D、 StructuredD (from DFD)、PDL(过程设计语言)来表示设计。**设计成果：**概要设计说明书、详细设计说明书（包含界面、功能逻辑、数据三个方面的设计）。**实现任务：**设计和编写子程序、类等模块，选择控制结构并组织语句块，找出缺陷并调试修正，对代码细节设计进行评审，配置管理，集成测试等。

**实现完成途径**:见图**实现成果**：完整的、可执行的软件代码。

**测试任务：**运行程序或模拟系统的执行，发现程序缺陷（与预期不符或不可接受的结果）的过程，分为代码走查、单元测试、集成测试、系统确认测试。测试途径：主要有白盒测试、黑盒测试。测试成果：《总体测试方案》《测试用例》《测试报告》《缺陷分析报告》《度量分析报告》。

**程序的文档化有哪些具体要求？**

1.有效、适当地使用注释，保证注释有意义说明性强。2.使用含义鲜明的变量名。3.协调使用程序段注释和程序行注释。4.始终坚持编制文档。

**一个程序段的文档应包含什么内容呢？**

1.代码的功能2.代码的完成者3.代码在整个软件系统中的位置4.代码编制、复审的时间5.保留代码的原因6.代码中如何使用数据结构和算法。

**阐述文档在提高软件可维护性上的作用**

软件可维护性是指理解、改正、调整和改进软件的难易程度。可维护性是软件质量的重要属性，设计软件开发方法、工具、过程，软件制品、文档，软件管理、资金收入、维护计划和维护团队等。1.概要设计文档中提供了初始类模型、细化的类以及动态交互模型，详细设计文档中提供了三层精化类图、事务处理逻辑、数据库设计，故此两种文档可在很大程度上帮助维护人员理解代码和整体框架。2.软件维护时如更改代码则可能需要进行回归测试，此时会依赖于软件测试计划相关的文档。3.维护人员留下的系统维护日志、错误记载能帮助维护人员快速定位变更位置，从而提高维护效率。

**UML各图分别对应OO方法的哪个阶段，起什么作用？**

**用例图：**用于需求分析阶段，表示从软件系统的外部使用者的角度看到的各项系统功能，并清晰的说明软件系统的边界，即用例的集合构成了软件系统所应该有的功能。详细设计阶段进行精化设计。**类图：**需求获取阶段表达领域概念模型即业务领域中的概念及概念之间的关系。需求分析阶段：表示软件需求模型的静态结构部分。设计（UI+详细设计）和实现阶段表示软件的结构和详细设计。**活动图：**需求阶段描述实体为完成某项功能而执行的操作序列，精确描述单个用例中的处理流程，也可描述多个用例联合处理流程。在详细设计和实现阶段精确地描述线程之间的并发。但在描绘对象之间的交互协作方面不如交互图，在描述对象的行为方面不如状态图。**交互图**（UI界面流设计+详细设计）=顺序图（强调时间序）+通信图（强调对象之间的交互、协作关系）。经常用于描述单个用例功能的实现方式，或软件系统在某个给定的使用场景下对象之间的相互协作流程，或某个复杂操作的逻辑实现模型。需求分析阶段：分析、表述业务处理过程或与软件相关的处理流程；设计和实现阶段：根据交互图确定对象的操作以及操作实现方法；测试阶段：通过交互图了解软件处理过程并据此设计测试用例。**状态图**：在需求分析与详细设计阶段：描述一个类的典型对象，一般对行为比较复杂，行为的状态特征比较明显的类建立状态图。**包图：**体系结构设计阶段，刻画包之间的构成了依赖关系。**构件图**（体系结构设计+详细设计）：描述构建之间接口定义以及构建之间的依赖关系。描述软件系统中某个局部的构件化设计。描述软件系统中某个版本的配置情况，以展示其内部构造。**部署图**：体系结构设计阶段，表示软件系统的可执行工件在运行环境中的分布情况，如DLL, jar。是软件安装、维护工程师了解系统部署状况的主要信息源，也是设计、实现人员理解软件运行环境、确定软构件划分的重要参考。**对象图**：软件系统中的某些对象在运行过程中的瞬间快照。

PROCEDUEF average

INTERFACE RETURNS average,total.input,total.valid;

INTERFACE ACCEPTS value,minimum,maximum;

TYPE value[1,100] IS SCALAR ARRAY;

TYPE average,total,input,total.valid,minimum,maximum,sum IS SCALAR;

TYPE i IS INTEGER;

i=1;

total.input-total.valid=0;

sum=0;

DO WHILE value[i]< > -999 and total.input<100

total.input加l;

IF value[i]> = minimum and value[i]< = maximum

THEN

total.valid加l

sum=sum+value[i]

ELSE skip

END IF;

i加l;

END DO;

IF total.valid>0

THEN average=sum/total.valid

ELSE average=-999;

END IF

END average

·数据字典：用来描述系统中所涉及的每一个数据，是一个数据描述的集合（即对数据的进一步描述）。加工说明：对转换动作的进一步描述。

**数据字典DD模板：**A=B+C+D+…;

DD中的表示：=：定义为；+：加；[ ]:任选其一；{ }:可重复使用；（）可选符号，可有可无；\* \*：注释。比如：省=[江苏|浙江| …] 借书单=ID+（姓名）+日期+｛书｝

**加工说明例如**

判断读者是否存在

判断读者信息的合法性和有效性

查询读者信息库，判断该读者是否已经注册

