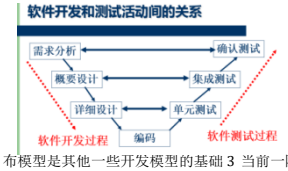
【第一章】**软件：**与计算机系统操作有关的程序、数据以及相关文档的完整集合。**软件包含：**程序、配置文件、系统文档因此，**软件不等于计算机程序**。**软件特点** 1 逻辑实体、智力产品，制造即拷贝 2 无磨损和老化，不遵循“浴盆曲线”，但存在退化问题 3 尚未摆脱手工方式，软件移植的需要，复杂（问题复杂性/ 程序结构复杂性），软件开发的性质如成本、进度、质量等难以估计控制，维护困难，可复用性**软件分类：**1 按功能：系统软件/支撑软件/应用软件 2 按工作方式：实时处理/分时/交互/批处理 3 按服务对象：项目 / 产品（定制 / 通用）4 按失效影响 ：关键软件 / 非关键软件 5 规模：微型、小型、中型、大型、甚大型、极大型。**软件危机的表现：** 1 软件开发成本和进度失控，维护代价高 2 用户不满意 3 软件质量不可靠 4 软件不可维护 5 无文档资料 6 计算机系统中软件成本比重加大 7 软件开发生产率提高不能满足要求。**软件危机的原因：***1* 软件的规模和复杂性 2 人类智力的局限性 3 协同工作的困难性 4 缺乏方法学和工具 5 用户描述不精确、二义、遗漏，双方理解有偏差。**软件危机缓解途径：** 1 组织管理、协同配合的工程 2 软件工程的理论模型、技术方法 3 软件工具。**为什么会有软件工程这个概念**？因为软件危机。**软件工程的三要素** 1 过程：管理部分 2 方法：技术手段 3 工具：自动或半自动地支持软件的开发和管理。**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。它包括了多方面的任务，如项目计划与估算、软件系统需求分析等。软件**工具**为软件工程方法提供了自动的或半自动的软件支撑环境。目前，已经推出了许多软件工具，这些软件工具集成起来，建立起称之为计算机辅助软件工程(CASE)的软件开发支撑系统。软件工程的**过程（例如喷泉模型）**则是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。过程定义了（见下面红字部分）软件工程是一种层次化的技术。任何工程方法（包括软件工程）必须以有组织的质量保证为基础。全面的质量管理和类似的理念刺激了不断的过程改进，正是这种改进导致了更加成熟的软件工程方法的不断出现。支持软件工程的根基就在于对质量的关注。**软件工程过程：**定义了 1 方法使用的顺序 2 要求交付的文档资料 3 为保证质量和适应变化所需要的管理 4 软件开发各个阶段完成的里程碑。**软件生命周期：**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)。【图】**软件测试分哪几个阶段？和软件开发过程的关系？软件开发和测试活动之间的关系：**软件开发和软件测试都是软件生命周期中的重要组成部分，软件测试是保证软件开发产物质量的重要手段。测试是贯穿于整个开发流程了，而不是只是在编码完成以后才开始。**瀑布模型**是将软件生存周期的各项活动规定为按固定顺序而连接的若干阶段工作，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。最终得到软件产品。**优点** 1 瀑布模型是使用时间最长、应用面比较广泛的开发模型 2 瀑布模型是其他一些开发模型的基础 3 当前一阶段完成后，只需要去关注后续阶段**缺点** 1 不能适应用户需求的变化 2　到最后阶段才能得到可运行的软件版本。**适用场合：**对于规模较小,软件需求较为稳定的项目或子系统,采用瀑布模型能够显著提高软件开发的质量和效率。**演化模型（原型模型）**是一种全局的软件(或产品)　生存周期模型。属于迭代开发方法。该模型可以表示为：第一次迭代(需求->设计->实现->测试->集成)->反馈->第二次迭代(需求->设计->实现->测试->集成)->反馈->……**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** 1 需要为系统的每个新版本交付文档，不划算 2　新需求的不断增加，使系统结构退化，变更成本上升 3　不支持风险分析**螺旋模型** 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**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**建设周期长。**适用场合**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 。**喷泉模型：**软件复用与生命周期中多项开发活动集成，主要支持面向对象的开发方法。**优点** 1 软件系统可维护性较好 2 各阶段相互重叠，表明了面向对象开发方法各阶段间的交叉和无缝过渡 3 整个模型是一个迭代的过程，包括一个阶段内部的迭代和跨阶段的迭代 4 模型具有增量开发特性，即能做到“分析一点、设计一点、实现一点，测试一点”，使相关功能随之加入到演化的系统中 5 模型由对象驱动，对象是各阶段活动的主体，也是项目管理的基本内容 6 该模型很自然地支持软部件的重用。**缺点**由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，因此不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。



什么是模块化：将一个软件划分为一组具有相对独立功能的部件，每个部件称为一个模块；当把所有的模块组装在一起时，便可获得满足用户需求的软件系统。

面向对象方法比结构化方法好在哪，传统的方法不足在哪里？（实际上是先有了面向对象语言然后再有的方法，开发中利用面向对象的方法思想开发）用 c++来举例子是如何支持这样的概念的,对象、类、继承、封装、聚合、关联、消息、多态。**面向对象的基本思想（OO 方法和结构化方法相比）：**1 为了克服传统开发方法的不足,面向对象方法解决问题的思路是从现实世界中的客观对象（如人和事物）入手,尽量运用人类的自然思维方式来构造软件系统，这与传统的结构化方法从功能入手和信息工程化方法从信息入手是不一样的。2 在面向对象方法中，把一切都看成是对象。**相似之处：**　遵循软件工程的一般原理**什么是面向对象？**从程序设计方法的角度看，面向对象是一种新的程序设计范型(paradigm)，其基本思想是使用对象、类、继承、封装、聚合、关联、消息、多态性等基本概念来进行程序设计。从软件方法学的角度看，面向对象方法是一种运用对象、类、继承、封装、聚合、关联、消息、多态性等概念来构造系统的软件开发方法。

OO 基本思想：(对象)：从现实世界中客观存在的事物出发来建立软件系统，强调直接以问题域（现实世界）中的事物为中心来思考问题、认识问题，并根据这些事物的本质特征，把它们抽象地表示为系统中的对象，作为系统的基本构成单位。这可以使系统直接映射问题域，保持问题域中事物及其相互关系的本来面貌。**(属性与操作)**用对象的属性表示事物的状态特征；用对象的操作表示事物的动态特征。**(封装)**对象的属性与操作结合为一体，成为一个独立的、不可分的实体，对外屏蔽其内部细节。**封装概念**：把对象的属性和操作结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。只是向外部提供接口，降低了对象间的耦合度。**(分类)**对事物进行分类。把具有相同属性和相同操作的对象归为一类，类是这些对象的抽象描述，每个对象是它的类的一个实例。**(聚集)**复杂的对象可以用简单的对象作为其构成部分（有小的聚集成大的）。**(继承)**通过在不同程度上运用抽象的原则,可以得到较一般的类和较特殊的类。特殊类继承一般类的属性与操作，从而简化系统的构造过程及其文档。**(类的封闭性)**类具有封闭性，把内部的属性和服务隐藏起来，只有公共的服务对外是可见的。**(消息)**对象之间通过消息进行通讯，以实现对象之间的动态联系。**(关联)**通过关联表达类(一组对象)之间的静态关系。聚集和继承集中体现了 OO 的重要的特性：**复用**。(**复用的特性是怎么体现的？**体现在聚集、继承两个）**用 OO 的方法（结构化的方法举例软件工程精髓）来描述“自顶向下，逐步求精”**（含义：从顶层开始逐层向下分解，直至系统的所有模块都小到易于掌握为止　，具体实现：把一个大的问题，抽象成一个一个的类去解决。举例子，例如在画 DFD 图：　DFD 图分为三级 顶级、0 级、1 级，每一层的都比前一层更加的精化体现着自顶向下、怎么去设计软件体系结构、最后得出一个个类（函数过程）。抽象、问题分解、逐步求精这三步骤）

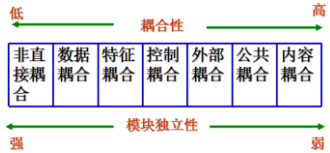
抽象：在不同的高度看待或解决问题。从事物中舍弃个别的非本质的特征，而抽取共同的、本质特征的做法叫抽象；1 过程抽象：任何一个完成确定功能的操作序列，其使用者都可把它看作一个单一的实体，尽管实际上它可能是由一系列更低级的操作完成的。2 数据抽象:根据施加于数据之上的操作来定义数据类型，并限定数据的值只能由这些操作来修改和观察。两种抽象举例子下面　**OO 方法好在哪？**因为他**封装**了。独立性就好了，可维护性就高了。**为什么 OO 方法开发出来的可维护性高？模块独立性高？**（回抽象封装信息隐藏，回答封装的意义）**封装的意义：**封装的重要意义：使对象能够集中完整地描述并对应一个具体事物。体现了事物的相对独立性，使对象外部不能随意存取对象的内部数据，避免了外部错误对它的“交插感染”。对象的内部的修改对外部的影响很小，减少了修改引起的“波动效应”。公开静态的、不变的操作，而把动态的、易变的服务隐藏起来。**联系起来封装+信息隐藏+模块独立性+可维护性　信息隐蔽：**　定义什么操作（函数）可被其他对象访问。每一个对象将愿意提供给所有对象的公共操作公开化。　它也提供仅局限于特定对象的其它的操作（受保护的和私有的）。实际上，其他的对象对被请　求的对象怎样提供操作没有感知.

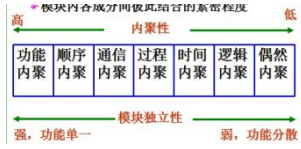
【第三章】需求分析 DFD 图 数据流图

【第四章】功能模型 用例图 活动图

【第五章】类图【第六章】不考

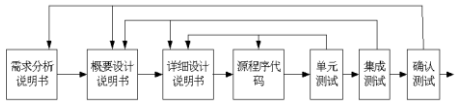
【第七章】**自顶向下，逐步求精（top-down design）”** ：从顶层开始逐层向下分解，直至系统的所有模块都小到易于掌握为止　**什么是模块：**1.结构化：过程、函数 2. OO: 对象类。**什么是模块间的关系**结构化：调用关系 OO:　类之间的结构关系(继承和聚集)，消息传递关系。**什么是模块化** 1 将一个软件划分为一组具有相对独立功能的部件，每个部件称为一个模块；当把所有的模块组装在一起时，便可获得满足用户需求的软件系统 2　模块化体现了“分而治之”的问题分析和解决方法。**为什么要进行模块化（局部化）：**　模块化体现了“分而治之”的问题分析和解决方法。**模块化的目的**①进行功能分解，把复杂的大的功能划分成简单的小的子功能，尽量降低每个模块的成本。②尽量使每个模块间的接口不能太多，太多会使接口成本增加。兼顾二者可取得最佳的划分状态，确保软件总成本最低。**怎么进行模块化（模块设计原则）**1 信息隐藏 2 高内聚度（强）3 低耦合度（松）**抽象：** 1 过程抽象（计算）将完成一个特定功能的动作序列抽象为一个函数名和参数表（模块）eg:　比较字符串: 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 。**什么是信息隐藏**（1）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（2）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。**信息隐藏和局部化的优点**（1）支持模块的并行开发（设计和编码）（2）模块的独立性更好（3）便于系统功能的扩充（4）便于测试和维护，减少修改影响向外传播的范围**模块化、信息隐藏，局部化是什么关系**局部化与信息隐藏是一对密切相关的概念。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。因此，局部化意味着减少模块之间的联系，有助于实现模块之间的信息隐藏。在软件测试和维护期间经常需要修改一些模块的内容。信息隐藏和局部化降低了模块之间的联系，使得在修改一个模块时对其他模块的影响降到最低。“隐藏”的意思是，有效的模块化通过定义一组相互独立的模块来实现，这些独立的模块彼此之间仅仅交换那些为了完成系统功能所必需的信息，而将那些自身的实现细节与数据“隐藏”起来。**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。**模块化独立性两个衡量标准：** 1 耦合（coupling）模块间的关联紧密程度 2 内聚（cohesion）模块内各成分间彼此结合的紧密程度。关联紧密程度【图 1】　模块内各成分间彼此结合的紧密程度【图 2】



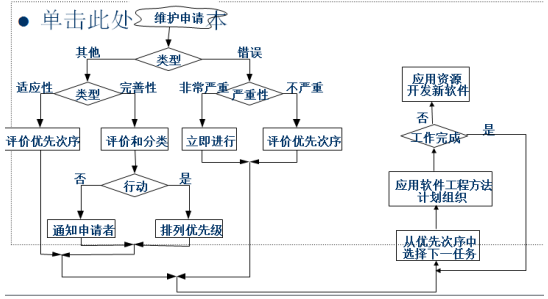


软件设计的原则 如何提高模块独立性：　信息隐藏+局部化+高内聚低耦合+简化模块的接口的复杂程度。*4.1 提高模块独立性：*消除重复功能，改善软件结构 ① 完全相似：在结构上完全相似，可能只是在数据类型上不一致。此时可以采取完全合并的方法。② 局部相似：找出其相同部分，分离出去，重新定义成一个独立的下一层模块。还可以与它的上级模块合并。*4.5 简化模块接口的复杂程度：*1 模块接口复杂是软件发生错误的一个主要原因。2 接口复杂或不一致（即看起来传递的数据之间没有联系），是紧耦合或低内聚的征兆。**接口复杂\模块接口的复杂性包括三个因素：** 1 传送信息的数量：参数的个数，最好 1 个（通常 2—4 个）2 耦合方式：call 方式 vs “直接引用”（在参数类型上，尽量少用指针、过程等类型的参数）3 传送信息的结构：尽量用数据型、少用控制型。

【第 8 章】软件设计方法 PDL，语法在里面。



【第 9 章】**软件设计原则**:1 提高模块独立性 2 调整模块的大小 3 适当的模块深度、宽度、扇出和扇入 4 模块的作用范围应在控制范围之内 5 简化模块的接口的复杂程度。**界面(HCI)特征怎么做(OFFICE WINDOWS 举例子)**1 可使用性:使用简单；用户界面术语的标准化，一致化；用户 Help 帮助功能；能快速的系统响应；有容错能力 2 灵活性:算法的可隐可现性，能满足不同层次用户；　按照用户需要，提供不同详细程度的响应信息；用户可以根据需要定制、修改菜单的方式 3 可靠性：在完成预定功能的前提下，尽可能简单，但不是安排成线性序列；用户界面的可靠性是指无故障使用的间隔时间；　用户界面应能保证用户正确的、可靠的使用系统，保证有关程序和数据的安全性.**界面设计考虑** 1 系统响应时间：计算机应对用户操作做出反应，用户根据反馈做出操作结果如何的判断。如果执行时间较长，应该给出相应的提示信息　2 用户求助机制：系统要提供帮助，帮助用户学习使用系统，还应该在用户出现操作困难时随时提供帮助。3　错误信息处理：系统要能检测和处理可能出现的错误，出错信息包含出错位置、出错原因和修改建议等方面，出错信息应该清楚、易理解 4 命令方式：应该让输入和输出具有一致性，在应用程序的不同部分，甚至是不同应用程序之间，要具有相似的外观和布局，具有相似的交互方式和相似的信息显示格式**程序设计风格是指什么** 1 源程序文档化：程序应加注释，用自然语言或伪代码描述，它说明了程序的功能，对理解程序提供了明确指导 2　数据说明：一个语句说明多个变量时，各变量名应按字典序排列。对于复杂的数据结构，要加注释 3 语句结构：程序设计应该简单易懂，语句构造应该简单直接 4 输入 / 输出方法：输入前应该有提示信息，例如输入时提供可选择的边界值，输出数据表格化，图形化　【第 11 章】18 分题目　概念初步的软件系统存在错误，如何：1 发现错误?2 纠正错误？软件测试是软件质量保证活动中关键步骤：1 对 SRS、设计规格说明书以及编码的最后复审 2 其工作量往往占软件开发总工作量的 40%以上 3 软件测试是确保软件质量的一种有效（可操作)手段。软件测试有其特殊性和规律　– 因为软件是逻辑产品。**软件测试原则** 1 尽早地、不断地进行软件测试 2 程序员应避免检查自己的程序 3 在设计测试用例时应包括合理的输入条件和不合理的输入条件 4 充分注意测试中的群集现象(发现错误越多，说明残存错误越多）5 严格执行测试计划，注意排除测试中的随意性 6 应当对测试结果做全面检查 7 妥善保管测试计划、测试用例、出错统计、和最终分析报告，为维护提供方便**软件测试和开发的关系【上图】**测试：单元测试，集成测试，确认测试对应开发：详细设计，概要设计，需求分析。**白盒测试逻辑覆盖准则强弱**语句覆盖　<判定覆盖（通常<）条件覆盖<=条件/判定覆盖<条件组合覆盖【**强弱关系，例外情况】**。**软件维护的分类(经过测试的软件为什么要维护）**1 纠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，应当进行的诊断和改正错误的过程就叫做改正性维护 2　适应性维护　　使用过程中。外部环境（新　　的硬、软件配置），数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）可能发生变化。　为使软件适应这种变化，而去修改软件的过程就叫做适应性维护。3 完善性维护　　在软件的使用过程中，　用户往往会对软件提出新的功能与性能要求。为了满足这些要求，需要修改或再开发软件，以扩充软件　　功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动叫做完善性维　护。4 预防性维护　预防性维护是为了提高软件的可维护性、可靠性等，为以后进一步改进软件打下良　好基础**影响维护的因素**　1　系统大小：系统越大，理解掌握起来越困难。系统越大，所执行功能越复杂。



因而需要更多的维护工作量 2 程序设计语言：语言的功能越强，程序的模块化和结构化程度越高，指令数就越少，程序的可读性越好 3 系统年龄：老系统随着不断的修改，结构越来越乱；维护人员经常更换，程序又变得越来越难于理解。许多老系统在当初并未按照软件工程的要求进行开发，因而没有文档，或文档太少。　在长期的维护过程中文档在许多地方与程序实现变得不一致，在维护时就会遇到很大困难 4 其它：数据库技术的应用；软件开发技术的先进程度；　应用的类型；数学模型；任务的难度；开关与标记、IF 嵌套深度、索引或下标数等。**维护活动的流程图【下图】**

【第 12 章】**什么是软件的可维护性**目前广泛使用的是用如的七个特性来衡量程序的可维护性可理解性、可使用性、可测试性、可移植性、可修改性、效率、可靠性**影响软件可维护性的因素** 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、OO4 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……

什么是软件维护的副作用，有哪几个方面什么是：由于维护或者在维护过程中其他一些不恰当的行为而引入新的错误 1 修改代码的副作用：在修改源代码时，　都可能引入错误。例如，删除或修改一个子程序 2 修改数据的副作用：　在修改数据结构时，有可能造成软件设计与数据结构不匹配，因而导致软件出错 3 修改文档的副作用：　对数据流、软件结构、　模块逻辑或任何其它有关特性进行修改时，必须对相关技术文档进行相应修改。否则会导致文档与程序功能不匹配，　缺省条件改变，　新错误信息不正确等错误。使得软件文档不能反映软件的当前状态。

【第 13 章】一个概念项目的估算**估算**是对软件产品、过程、资源（人力、成本等）进行预测。用于签订合同、立项、制定工作计划。

【第 14 章】

配置管理是指通过执行版本控制、变更控制等规程，以及使用合适的配置管理软件，来保证所有配置项的完整性和可跟踪性，减少系统混乱，提高软件生产效率**为什么要进行配置管理**主要目标是标识变更；控制变更；确保变更正确地实现；报告有关变更**什么是有效的配置管理**任何项目成员都要对其工作成果进行配置管理，但不必付出过多的精力，最低要求是保证重要工作成果不发生混乱。 配置管理规范应当清晰明了，便于执行，没有太多细节要求，不给项目人员添加过多的负担。选择配置管理工具应当综合考虑价格、易用性和功能因素。令人满意的工具通常是价格低廉、简便易用、功能恰好够用。

基线是软件生存期中各开发阶段的一个特定点，它的作用是把开发各阶段工作的划分更加明确化，使本来连续的工作在这些点上断开，以便于检查与肯定阶段成果**基线的修改** SCI 一旦成为基线，就被放入项目数据库中。若要修改基线，首先要将它拷贝至私有工作区并在项目数据库中锁定，不允许他人使用。在私有工作区完成修改控制过程并复审通过后，再将修改后的 SCI 推出并送至项目数据库，同时解锁**对象是类的一个实例。软件测试 VS 调试 目的不同　软件测试为了发现错误，调试为了定位错误和纠正错误。软件测试可由独立的测试组进行，调试由开发人员完成。常用的基线【下图】**

逐步求精、分层结构与抽象三者之间的相互关系如何软件设计的精髓：自顶向下，逐步求精。即将软件的体系结构按自顶向下方式，对各个层次的过程细节和数据细节逐层细化。这样的结构实际就是一个模块的分层结构，即分层的过程。在实施时，采用抽象的方法，自顶而下，给出不同的抽象层次。在最高的抽象层次上，可以使用问题所处环境的语言概括的描述该问题的解法。而在较低的抽象层次上，则采用过程化的方法。在描述问题的解法时，我们可以配合使用面向问题的术语和面向现实的术语。但在最低的抽象层次上，应使用能够直接实现的方式来描述这个解法。**PDL** 数据说明用以指明数据名的类型以及作用域,其形式: declareas 子程序结构 procedure　interface　return　end 分程序结构 begin end 分支控制结构 条件分支:ifthen else end　多分支:case Of,||;…[default]:end case 循环控制结构当型循环 loop while end loop 直到型循环:loop until end if

当条件为真时退出循环体 输入输出结构 print read　display　input output

CMM 初始级 可重复级 已定义级 已管理级 优化级（成熟度提高 风险降低）

【第1章补充】软件工程概念:用工程的、科学的概念、原理、技术和方法，进行软件的开发、管理和维护 **软件危机概念**是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。软件危机泛指在计算机软件的开发和维护过程中所遇到的一系列严重问题。无法解决只能缓解。　**三要素的关系(两两)**软件工程方法为软件开发提供了 “ 如何做 ” 的技　术。它包括了多方面任务，如项目计划与估算、软件系统需求分析等。软件工具为软件工程方法提供　了自动的或半自动的软件支撑环境。目前已经推出了许多软件工具，这些软件工具集成起来，建立起　称之为计算机辅助软件工程(CASE)的软件开发支撑系统。软件工程的过程则是将软件工程的方法和工　具综合起来以达到合理、及时地进行计算机软件开发的目的。过程定义了（见上红字）软件工程是一　种层次化的技术。任何工程方法（包括软件工程）必须以有组织的质量保证为基础。全面的质量管理　和类似的理念刺激了不断的过程改进，正是这种改进导致了更加成熟的软件工程方法的不断出现。三 要素之间相互关联和支持。**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？优缺点　相同：同样遵循抽象原则和模块化原则。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，难以控制抽象层次.**OO的优点**1）把软件系统看成是各种对象的集合，这更接近人的思维，使得使用者和维护人员容易理解2）软件需求的变动往往是功能的变动，而功能的执行者即对象一般不会有太大的变化。3）对象包括属性和方法。对象把属性和方法的具体实现方式一起封装起来，这使得方法与之相关的属性不再分离，提高每个子系统的相对独立性，从而提高软件的可维护性。4）支持封装，继承，多态，和抽象，提高了软件的可重用性，可扩展性，封装->模块独立性好->软件可维护性高（封装意义），同时由于它的稳定性好对软件局部进行修改时，不会引起整体变化，而且对局部的修改易实现，易于测试。4）OO直接从问题入手来进行系统的建模，降低了使用成本，让用户在使用新的软件能够在最短的时间里了解使用方法，能正确、清楚地使用软件。5）把大的问题分解成相互独立的小的问题处理，降低了开发技术难度，开发工作的管理也变得容易了，开发大型软件变得容易，成本也较低。**软件开发和软件测试关系(两张图+第一页中部+补充)**测试过程：1)单元测试以详细设计说明书和源程序代码为依据，对源程序中每一个程序单元测试程序中每个模块是否有错误。（白盒）2)集成测试以概要说明书和详细设计说明书为依据，是为了测试软件总体结构是否有错误（黑盒+部分白盒），也就是检查概要设计是否合理有效。3)确认测试一需求分析说明书和概要设计说明书为依据，主要是检查已实现的软件是否满足需求规格说明书中确定了的各种需求。（黑盒）4)系统测试是把将软件系统与其他部分（数据库、硬件）集成后测试，以确定软件是否可以支付使用。

【第2章补充】OO特征封装性、继承性、多态性、抽象　多态：同一消息为不同的对象接受时可产生完全不同的行动，这种现象称为多态性。多态是指同一个命名可具有不同的语义。OO方法中，常指在一般类中定义的属性或操作被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。对于子类，可用不同的方法替代实现父类的方法（C++语言支持编译时的静态多态性和运行时的动态多态性，分别是通过函数重载和虚函数来实现的）**为什么OO高复用性，体现？**1继承2聚集（拿来就用）3封装+信息隐藏+模块独立性+可维护性　5面向对象程序设计的重用性体现在类库上，用对象描述问题域中的事务，并用类作为同种对象的抽象表示。类库中的类不仅可以直接使用，而且可以在类库中类的基础上派生新类,对基类进行扩充和修改以满足应用需求，极大的提高了程序的重用性。 类是系统的基本构成单位，它符合可复用构件所应具备的那些特性:完整性、独立性、可标识性、一般性、抽象、封装：尤其是OO　方法的对象封装性，为软件复用技术提供了良好的条件。而继承机制本质上就是一种复用机制，因此面向对象方法为软件复用提供了一个良好的环境。

【第7章补充】软件工程的精髓：自顶向下逐步求精；具体表现：结构化：　分析阶段：分层DFD，由顶级数据流图到一一级数流图;设计阶段:从需求分析到概要设计再到详细设计　OO:分析阶段：从用例的分割到细化活动图对应1:1:n的过程，从需求分析报告到系统分析报告再到系统设计报告，设计阶段:从初始类图到细化类图再到三层精化类图　**模块独立性是指**软件系统中每个模块只涉及软件要求的具体的子功能，　而和软件系统中其它的模块的接口是简单的。是指模块内部各部分及模块间的关系的一种衡量标准，由内聚和耦合来度量。

【第11章补充】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;

【13】度量:对软件产品、软件开发过程或资源的某一属性的定量描述（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。

测量:度量的过程,为产品或过程的某个属性的范围、数量维度容量或大小提供量化的指示,涉及到测量的方法过程工具和数值结果。**评估:**对系统、部件或过程的某一属性所具有的程度进行的主观分析和评定，得到量化结果。**评价（评审）:**为了确定软件模块、包或产品的接受或分发，对特定软件模块、包或产品应用特定的文档化的评估标准的活动。　**软件质量度量:**McCall的三层次软件质量度量模型(软件质量因素:产品修改、产品转移、产品运行)**可靠性:**表明了一个程序按照用户的要求和设计的目标，执行其功能的正确程度。软件可靠性是指在规定的运行环境中和规定的时间内无失效运行的概率，记作Ps(t) ；发生故障的概率，记作Pf(t) **软件可靠性度量:**用于测量软件可靠性的度量**Gantt图是**安排软件进度计划的又一种工具。图左边列出项目的开发活动,上方列出日历时间。在每一活动的开始和结束时间各画一个三角形，当活动的开始和结束时候，把三角形涂实。

优点:简单易用,容易修改,比PERT图更加直观方便。**缺点:**不能显示各项活动之间的依赖关系。

代码行数LOC估算法优点:直观、简单易行;**缺点:**依赖于特定程序设计语言；可能对一些设计精巧的软件项目不利;项目开发前期估算困难；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;

FP优点: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；度量可以用于软件开发的初期

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

SQA的原则在开始制定项目计划时，充分考虑用户（或客户）的要求，确定实用的质量特征。详细制定质量计划。在对开发过程进行质量评估的基础上，检查质量测试的结果以发现与质量计划的偏差，确定适当的修正方案对阶段性成果进行各种质量检查、校对等评审活动。不断地优化和完善质量计划。尽早发现并及时改正错误和缺陷。由独立的测试小组进行质量测试。通信代价：通信需花费时间和代价，会引起软件错误增加，降低软件生产率。**经验估算模型:**CoCoMo模型：构造性成本模型(人月)

【14】配置管理贯穿于整个软件生命周期。软件配置项是软件工程中产生的工作成果,它是配置管理的基本单位。一个SC中最早的SCI是系统规格说明书，随后是软件项目规划和软件需求规格说明书，随着开发的深入，SCI迅速增加。报告的形成就是基线为了提高配置管理的效率和安全性，项目应当设有配置管理员这个角色。流程(下面第四张图)

为什么需要软件配置管理在软件开发时，变更是不可避免的，而变更时由于没有进行变更控制，可能加剧项目中的混乱。为协调软件开发使得混乱减到最小，使用配置管理技术，使变更所产生的错误达到最小并最有效地提高生产率。

变更控制：软件工程过程中某一阶段的变更,均要引起软件配置的变更，这种变更必须严格加以控制和管理，保持修改信息，并把精确、清晰的信息传递到软件工程过程的下一步骤。变更控制包括建立控制点和建立报告与审查制度检出和检入处理实现了两个重要的变更控制要素，即存取控制和同步控制。存取控制管理各个用户存取和修改一个特定软件配置对象的权限。同步控制可用来确保由不同用户所执行的并发变更。（锁）

程序设计风格win10：源程序文档化:程序文本中，注释行的数量占到整个源程序的1/3到1/2，甚至更多。数据说明：数据说明的次序规范化，说明语句中变量安排有序化，使用注释说明复杂数据结构语句结构：清晰性，可读性高；输入／输出方法:图形输出，DOS程序独占整个显示屏幕，其他程序在后台等待而Windows程序的所有输出都是图形，Windows下的输出是设备无关的，Windows下的应用程序使用图形设备接口（GDI）来进行图形输出。输入消息：包括键盘和鼠标的输入。

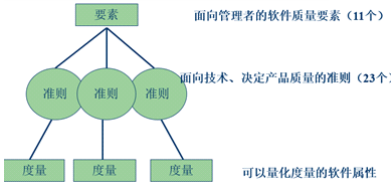
界面设计win10体现：1 可使用性：使用简单：使用简单：Windows 7 有自带的欢迎中心，其中有最基础教学。Windows 10则通过Contana作为系统初始化时的新手引导。术语的标准化，一致化：微软有对于Windows桌面应用程序的一系列术语表。帮助：在任何界面按下F1，就会有离线或在线的帮助程序打开。另外在网络等部分，有特殊的疑难解答可以半自动地解决一定故障。能快速响应：在硬件不成为瓶颈的情况下，由于从Windows 95开始引入了抢占式多任务处理，一般来说当系统资源没有被全部占用的情况下，可以做到随点随开。有容错能力：由于是抢占式多任务处理，一个程序的崩溃不会带动系统的崩溃。在Windows 7/10中，即使是驱动程序出现故障，一般也不至于系统宕机。 2 灵活性　能满足不同层次用户：从Windows 10开始，自带应用商店，使得基础用户可以不借助浏览器，下载到官方安全的应用，免受全家桶和恶意软件的侵害。Windows 7/10都有家长模式、访客模式、一般账户模式，满足各个方面用户的需要。管理员用户则可以通过Windows的特性，完全控制电脑。提供不同详细程度的响应信息：用户可以从UI的变化和基础的任务管理器上直观感受到软件和操作系统的响应。高级用户可以通过系统提供的各种工具查看系统响应的内容。定制、修改菜单的方式：系统和一些软件提供有限的开始菜单和右键菜单的定制功能，使得用户在不至于搞乱系统的情况下个性化菜单或者切换到易于使用的模式。3可靠性：无故障使用的间隔时间：在使用了NT6内核之后，Windows 7/10已经很少由系统本身的bug甚至驱动引发蓝屏了。而使用同样内核的服务器版本则可以达到7x24的使用标准。

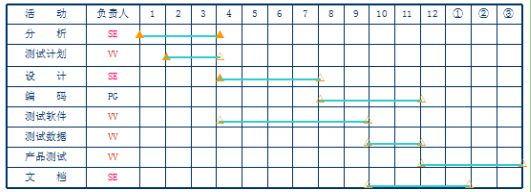
HCI风格：命令行：命令提示符/Powershell　简单菜单式：在Windows 7/10的自身程序内已经没有这种交互了。老版本的Windows有fdisk.exe属于这种交互。 点选界面：几乎所有的界面都是基于点选的。超文本、多任务：Windows的桌面提供多任务的切换途径；Windows的帮助文档是基于超文本的。

内部属性（面向技术人员软件人员）软件产品、过程和资源本身的属性，如软件产品的结构、模块化程度、复杂性、程序长度等

【12】提高可维护性的方法1确定质量管理目标　（可理解、修改、测试）2规范化的程序设计风格　（模块化、结构化）3选择可维护性高的语言　（高级语言）4完善程序文档 5保证软件质量审查方法

什么是软件的可维护性:纠正软件系统出现的错误和缺陷，以及为满足新的要求进行修改、扩充或压缩的容易程度。什么是软件维护：软件在交付给用户使用后，由于自身问题以及应用需求和环境的变化，对它进行维护不可避免。软件维护主要是指根据需求变化或硬件环境的变化对应用程序进行部分或全部的修改，修改时应充分利用源程序. 修改后要填写程序改登记表，并在程序变更通知书上写明新旧程序的不同之处。





注△：活动开始和结束；▲：活动已开始和结束；SE 系统工程师；W 质量保证人员；PG 程序员