软工

**1.2/1.3软件危机 软件工程的目标**

**软件危机的表现A**

1.成本和进度预估不准。

2.用户对软件系统不满意。

3.软件产品的质量往往靠不住。

4.软件常常是不可维护的。

5.软件通常没有适当的文档资料。

6.软件成本在计算机系统总成本中所占的比例逐年上升。

7.软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。

**原因**

1.缺乏经验和开发数据的积累，使得开发工作的计划很难制定。

2.交流存在障碍，需求获取不充分或有误 。

3.软件开发过程不规范。

4.软件规模的增大，复杂性提高。

5.需要分工协作，不仅涉及技术问题，更重要的是必须有科学严格的管理。

6.缺少有效的评测手段，质量不能完全保证。

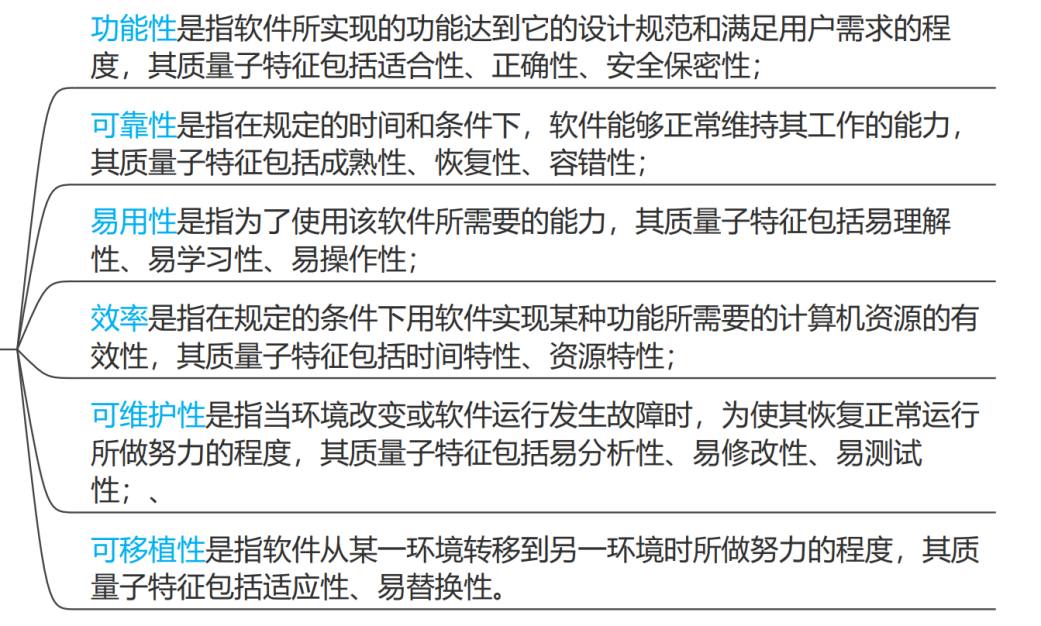
**软件工程的概念**

软件工程是指导**计算机软件开发和维护**的工程学科。采用**工程的概念、原理、技术和方法**来**开发与维护软件**，把经过时间考验而证明正确的**管理技术**和当前能够得到的最好的**技术方法**结合起来，以经济地开发出高质量的软件并有效地维护它。

**软件工程的目标A**

运用先进的**软件开发技术**和**管理方法**来提高软件的质量和生产率，也就是要以较短的周期、较低的成本生产出高质量的软件产品，并最终实现软件的工业化生产。

**质量**



**1.4与第2章 软件生存期**

**三个时期的任务**

软件定义：问题定义、可行性研究、需求分析和定义

软件开发：概要设计、详细设计、编码和单元测试、集成测试和系统测试

软件运行维护：改正性维护、适应性维护、完善性维护、预防性维护

**产物**

1.软件需求规格说明书：描述将要开发的软件做什么。

2.项目计划：描述将要完成的任务及其顺序，并估计所需要的时间及工作量。

3.软件测试计划：描述如何测试软件，使之确保软件应实现规定的功能，并达到预期的性能。

4.软件设计说明书：描述软件的结构，包括概要设计及详细设计。

5.用户手册：描述如何使用软件。

**生命周期模型的特点与优缺点**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 瀑布模型 | 快速原型模型 | 增量模型 | 螺旋模型 | 喷泉模型 |
| 特点 | 1.阶段间具有顺序性和依赖性2.推迟物理实现3.各阶段须提交文档并审核 | 快速建立起来的可运行程序，用于获知真正需求 | 把软件产品作为一系列增量构建来设计、编码、集成和测试，完成特定功能 | 将瀑布模型与快速原型模型结合，加入风险分析，四个过程 | 迭代和无间隙特性、活动存在交叠 |
| 优点 | 使用规范化方法、要求严格、产品经过评审 | 满足真实需求、线性顺序进行、规格文档正确描述需求、减少犯错可能、速度快 | 短时间提供可用产品、可逐步添加功能、失败风险较低、 | 有利于软件重用、有助于把质量作为目标、减少了测试次数带来的风险 |  |
| 缺点 | 产品与需求不匹配、需求不可修改、完全依赖于书面的规格说明 | 客户与开发者对原型理解不同、准确设计比较困难 | （注意事项）软件结构必须开放，加入新构建必须简单方便 | 要求开发者具有丰富风险评估经验和知识 |  |
| 场合 | 用户的需求非常清楚全面，且在开发过程中没有或很少变化、用户的使用环境非常稳定 | 对所开发的领域比较熟悉且有快速的原型开发工具、进行产品移植或升级 | 进行已有产品升级或新版本开发、完成期限严格要求的产品 | 螺旋模型只适合于大规模的软件项目 |  |

**统一过程的核心过程工作流**

① **业务建模**工作流。用商业用例为商业过程建立文档。

② **需求**工作流。目标是描述系统应该做什么，确保开发人员构建正确的系统。为此，需明确系统的功能需求和非功能需求（约束）。

③ **分析和设计**工作流。其目标是说明如何做。结果是分析模型和设计模型。

④ **实现**工作流。用分层的方式组织代码的结构，用构件的形式来实现类，对构件进行单元测试，将构件集成到可执行的系统中。

⑤ **测试**工作流。验证对象之间的交互、是否所有的构件都集成了、是否正确实现了所有需求、查错并改正。

⑥ **部署**工作流。制作软件的外部版本、软件打包、分发、为用户提供帮助和支持。

**统一过程的阶段**

① **初始阶段**。初始阶段主要关注**项目计划和风险评估**，其目的是确定是否值得开发目标信息系统。

② **细化阶段**。细化阶段关心定义系统的**总体框架**，其目标是：细化**初始需求**（用况）、细化**体系结构**、监控**风险**并细化它们的优先级、细化**业务案例**以及制订项目管理计划。

③ **构造阶段**。构造阶段是**建立系统**，构造信息系统的第1个具有操作质量的版本，以能够交付给客户进行测试的版本结束，有时称为**测试版本**。

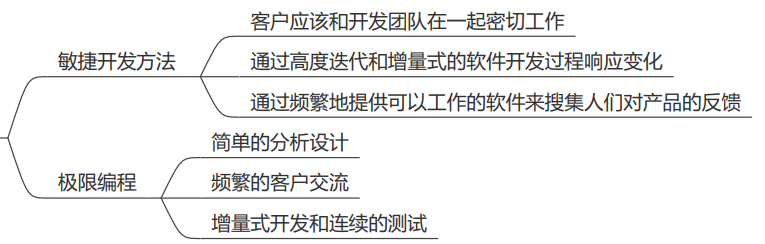
④ **移交阶段**。移交阶段包含测试时期，以**发布完整的系统**而终止，其目标是确保信息系统真正满足客户的需求。

**瀑布模型如何指导开发过程 线性化 有文档**

？

**敏捷开发与极限编程 原则 特点**

1.尽早持续地交付软件 2.欢迎需求变更 3.经常交付软件 4.天天一起工作 5.围绕有积极性的个体构建项目 6.面对面交谈 7.可运行软件是进度首要度量标准 8.稳定开发速度 9关注优秀的技能和好的设计 10.简单 11.自组织团队 12.反省与调整



**1.5软件工程方法学** 方法+工具+过程

**结构化方法学**

1.强调**自顶向下顺序**地完成软件开发的各阶段任务;

2.结构化方法要么面向行为，要么面向数据，**缺乏使两者有机结合**的机制。

**面向对象方法学**

1.是**将数据和对数据的操作**紧密地结合起来的方法。

2.软件开发过程是**多次反复迭代**的演化过程。

3.面向对象方法在概念和表示方法上的一致性，保证了各项**开发活动之间的平滑过渡**。

4.对于**大型、复杂及交互性比较强**的系统，使用面向对象方法学更有优势。

**3.1**

**需求获取任务**

1.发现和分析问题，并分析问题的原因/结果关系。

2.与用户进行各种方式的交流，并使用调查研究方法收集信息。

3.按照三个成分观察问题的不同侧面：即数据、过程和接口。

4.将获取的需求文档化，形式有用例、决策表、需求表等。

**需求获取原则**

**深入浅出的原则**。就是说，需求获取要尽可能**全面、细致**。获取的需求是个全集，目标系统真正实现的是个子集。

**以流程为主线的原则**。在与用户交流的过程中，应该**用流程将所有的内容串起来**。如信息、组织结构、处理规则等。这样便于交流沟通。流程的描述既有宏观描述，也有微观描述。

**需求分析任务A**

获取需求：通过启发和引导，获取用户或客户的**原始需求**

分析需求：完整性（全部必要信息）、正确性（准确无误无歧义）、合理性（协调一致不冲突）、可行性（技术经济社会可行）、充分性（是否全面周到），对获取的需求进行必要的增删

定义需求：清晰、全面、系统、准确地**描述成为正式的文档**，产生需求规格说明书

验证需求：确保需求**确认无误**，保证用户**接受理解**，**严格评审**

**4.1软件设计概念和原则**

**不同设计阶段的任务**

1.概要设计（设计程序的体系结构，也就是确定程序由哪些模块组成以及模块间的关系）：

（1）体系结构设计：体系结构设计定义软件的主要结构元素及其之间的关系。

（2）接口设计：接口设计描述用户界面，软件和其他硬件设备、其他软件系统及使用人员的**外部接口**，以及各种构件之间的**内部接口**。

（3）数据设计：传统方法主要根据需求阶段所建立的**实体—关系图（ER图）**来确定软件涉及的**文件系统的结构及数据库的表结构**。

2.详细设计（详细地设计每个模块，确定实现模块功能所需要的算法和数据结构）

过程设计：过程设计的主要工作是确定软件各个组成部分内的算法及内部数据结构，并选定某种过程的表达形式来描述各种算法。

**设计原则 A**

1.分而治之：当模块数目增加时，每个模块的规模将减小，开发单个模块的成本确实减少了；但是，随着模块数目增加，模块之间关系的复杂程度也会增加，设计模块间接口所需要的工作量也将增加

2.模块独立性：尽量使用数据耦合 少用控制耦合 限制使用公共耦合（除非传递大量数据） 完全不用内容耦合

3.提高抽象层次：设计开始时应尽量提高软件的抽象层次，按抽象级别从高到低进行软件设计

4.复用性设计：在构造新的软件系统时不必从零做起，可以直接使用已有的软构件即可组装（或加以合理修改）成新的系统

5.灵活性设计：任何代码、逻缉、概念在这个系统中都应该是唯一的（尽量）

结构化设计方法的实施要点

(1) 研究、分析和审查数据流图。

(2) 根据数据流图决定问题的类型：变换型和事务型。针对两种不同的类型分别进行分析处理。

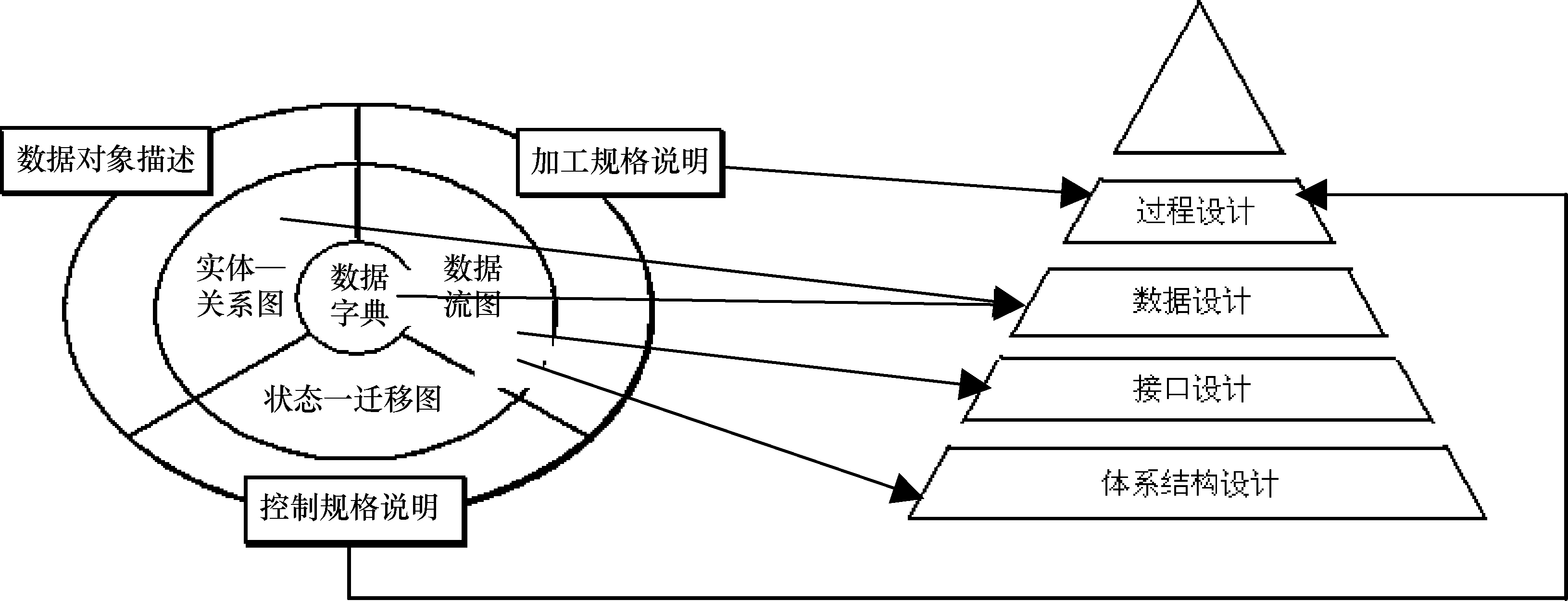
(3) 由数据流图推导出系统的初始结构图。

(4) 利用一些启发式原则来改进系统的初始结构图，直到得到符合要求的结构图为止。

(5) 根据分析模型中的实体关系图和数据字典进行数据设计，包括数据库设计或数据文件的设计。

(6) 在上面设计的基础上，并依据分析模型中的加工规格说明、状态转换图进行过程设计。

(7) 制定测试计划。



**5.1uml图 图的类别 类的关系**

**7.1体系结构的风格与特点**

体系结构的重要作用体现在以下三个方面 ：

（1）体系结构的表示有助于风险承担者（项目干系人）进行交流。

（2）体系结构突出了早期设计决策。

（3）软件体系结构是可传递和可复用的模型。

**7.5设计模式的特征**

**10.1各种测试的任务**

**机器测试：在设定的测试数据上执行被测程序的过程。又称动态测试 A**

1.黑盒：测试人员完全**不考虑**程序内部的逻辑结构和内部特性，只依据程序的**需求规格说明书**，检查程序的功能是否符合它的功能说明

2.白盒：允许测试人员**利用**程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序**所有逻辑路径**进行测试。通过在**不同点检查程序的状态**，确定实际的状态是否与预期的状态一致。

**人工测试：检查程序静态结构，发现编译无法发现的错误**

1.代码审查：以小组会的形式，发现程序在**结构、功能、编码风格**等方面存在的问题。可查出30%~70%的错误

2.走查：以小组会的形式进行，把测试数据“输入”到被测程序，并在纸上**跟踪监视程序的执行情况**，让人代替机器沿着程序的逻辑走一遍。

3.桌前检查：设计模块时，程序员**自己检查**。

**软件测试的步骤A**

单元测试：发现各**模块内部可能存在的各种差错**。单元测试需要从**程序的内部结构**出发设计测试用例。

组装测试：也叫做集成测试或联合测试。通常，在单元测试的基础上，需要将**所有模块按照设计要求组装成为系统**。

确认测试：又称有效性测试。它的任务是**验证软件的有效性**，即验证软件的功能和性能及其他特性是否**与用户的要求一致**。

系统测试：是将通过确认测试的软件，**作为整个计算机系统的一个元素**，与计算机硬件、外设、某些支持软件、数据、人员等其他系统元素**结合在一起**，在**实际运行（使用）环境**下，对计算机系统进行一系列的组装测试和确认测试。

**10.2白盒测试逻辑覆盖法A**

1.语句覆盖：设计若干个测试用例，运行被测程序，使得每一**可执行语句**至少执行一次

2.判定（分支）覆盖：设计若干个测试用例，运行被测程序，使得程序中**每个判断**的取真分支和取假分支至少经历一次

3.条件覆盖：设计若干个测试用例，运行被测程序，使得程序中**每个判断的每个条件**的可能取值至少执行一次

4.判定-条件覆盖：设计足够的测试用例，使得判断中**每个条件的所有可能取值**至少执行一次，同时**每个判断本身**的所有可能判断结果至少执行一次

5.条件组合覆盖：设计足够的测试用例，运行被测程序，使得每个判断的**所有可能的条件**取值组合至少执行一次

6.路径测试是设计足够的测试用例，覆盖程序中**所有可能的路径**

**10.3独立路径法**

基本路径测试是在程序控制流图的基础上，通过**分析控制构造的环路复杂性**，导出基本可执行路径集合，从而设计测试用例的方法

环路复杂性=判断节点数+1

确定基本路径集

设计测试用例

图形矩阵

**10.4黑盒测试 等价类划分 边界值方法A**

**有效等价类：**是指对于**程序的规格说明**来说，是合理的、有意义的输入数据构成的集合。利用它，可以检验程序是否**实现了规格说明预先规定的功能和性能**。

**无效等价类：**是指对于程序的规格说明来说，是不合理的、无意义的输入数据构成的集合。程序员主要利用这一类测试用例检查程序中**功能和性能的实现**是否有**不符合规格说明要求**的地方。

具体示例见PPT

**11.1四种维护**

**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**，应进行的**诊断和改正错误**的过程。

**适应性维护：**随着信息技术的飞速发展，软件运行的**外部环境**（新的硬、软件配置）或**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，为了使软件**适应这种变化**，而修改软件的过程叫做适应性维护。

**完善性维护（最多）：**为了满足**新的功能与性能要求**，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。

**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、编码和测试。

**11.3-11.4再工程和软件重构 逆向工程**

**逆向工程：**通过分析**目标系统**，标识**系统的部件及其交互关系**，然后使用**其他形式或者更高层的抽象**来**创建系统表现**的过程。

**重构：**在不改变代码外在行为的前提下，修改代码来改进**程序的内部结构**，提高**程序的可理解性和可维护性**等，进而帮助尽早的发现缺陷，提高编程速度。

**软件再工程：**通过对目标系统的检查和改造，其中包括设计恢复(库存目录分析)、再文档、逆向工程、程序和数据重构以及正向工程等一系列活动，旨在将逆向工程、重构和正向工程组合起来，将现存系统重新构造为新的形式，以开发出质量更高、维护性更好的软件。

**12.4软件能力成熟度模型 五个**

初始级（ML1）

已管理级（ML2）

已定义级（ML3）

已量化管理级（ML4）

优化级（ML5）

**13.6配置管理的作用**

目的：为某个过程或某个项目的软件项建立和保持完整性，以便相关方便于使用

1.制订软件配置管理计划。包括：

配置标识规则；

如何建立配置数据库，并将配置项置于配置管理之下；

配置管理人员的职责及配置管理活动；

所采用的配置管理工具、技术和方法。

2.实施变更管理，防止项目进行中因变更导致的混乱。

3.实施版本管理和发布管理。