软件工程：是指导计算机软件开发和维护的一门工程学科。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。

软件的生存周期的六个阶段及基本任务：a.计算机系统工程：确定待开发软件的总体设计要求和范围，以及该软件与其他计算机系统元素之间的关系，进行成本估算，做出进度安排，并进行可行性分析，并在若干个可行的解决方案中做出选择。b.需求分析：确定软件的功能、性能、数据、界面等要求，生成软件需求规约。c.设计：确定软件的系统设计和详细设计。d.编码：用某种语言，将设计的结果转换成为可执行的代码。e.测试：发现并纠正软件中的错误和缺陷。f.运行和维护：当发现软件中潜藏的错误或需要增加新的功能或使软件适应外界环境的变化等情况出现时，对软件进行修改。

软件过程成熟度等级：1.初始级：软件构成是无需的，甚至很混乱的。2.可重复级：建立了基本的项目管理来跟踪成本，进度和功能特性。3.已定义级：已将管理和工程活动两方面的软件过程文档化、标准化，并综合成有组织标准的软件过程。4.已管理级：收集对软件过程和产品质量的详细度量值对软件过程和产品都有定量的组件和控制。5.优化级：过程的变化反馈和先进的新思想、新技术促使过程不断改进。

程序设计语言：用于书写计算机程序的语言，它是一种实现性的语言。

软件过程模型：1.瀑布模型2.演化模型3.增量模型4.原型模型5.螺旋模型6.喷泉模型

系统工程的任务：识别用户需求；系统建模和模拟；成本估算及进度安排；可行性分析；生成系统规格说明。

可行性分析：经济可行性；技术可行性；法律可行性。

软件设计的原则：抽象与逐步求精，模块化，信息隐藏。

模块的独立性：模块完成独立的功能并且与其他模块的接口简单，模块间关联和依赖程度尽可能小。

模块化：把程序划分为独立命名并且可独立访问的模块。每个模块完成一个子功能，把这些模块集成起来构成一个政体，可以完成指定的功能满足用户的需求。

逐步求精：为了能集中精力解决主要问题而尽量推迟对问题细节的考虑。

信息隐藏：每个模块的实现细节对于其他模块来说应该是隐蔽的，模块中所包含的信息不允许其他不需要这些信息的模块使用，隐藏的不是模块的一切信息，而是模块的实现细节。

模块独立：是模块化、抽象、信息隐藏和局部化概念的直接结果。由两个定性标准衡量；耦合衡量不同模块彼此间互相依赖的紧密程度；内聚衡量一个模块内部各个元素彼此结合的紧密程度。

耦合种类：（由低到高）非直接耦合、数据耦合（数据）、标记耦合、控制耦合、外部耦合、公共耦合、内容耦合；设计原则为尽量使用数据耦合，少用控制耦合和特征耦合，限制公共环境耦合的范围，完成不用内容耦合。

内聚种类：（由弱到强）低：偶然内聚、逻辑内聚、时间内聚。中：过程内聚、通信内聚。高：顺序内聚、功能内聚。

结构图的深度：结构图中控制的层数。结构图的宽度：结构图中同一层次上模块总数的最大值。

结构图的扇出：指该模块直接调用的模块数目。结构图的扇出：指能直接调用该模块的模块数目。

启发式设计策略：改造程序结构图，降低耦合度，提高内聚度；避免高扇出，并随着深度的增加，力求高扇入；模块的影响范围应限制在该模块的控制范围内；降低模块接口的复杂程度和冗余程度，提高一致性；模块的功能应是可预测的，避免对模块施加过多的限制；尽可能设计单入口和单出口的模块。

对象：具有相同状态的一组操作的集合。

类：对具有相同属性和行为的一个或者多个对象的描述。

封装：把对象的属性和服务结合在一起形成一个独立的系统单位并尽可能隐蔽对象的内部细节。

继承：特殊类的对象拥有其一般类的全部属性和服务，称作特殊类对一般类的继承。

消息：用来请求对象执行某个处理或回答某些信息的要求。

方法：对象所能执行的操作，即类中所定义的服务。

多态性：在程序中，同一符号或名字在不同的情况下，具有不同解释的现象。

构件：是某系统中有价值的、几乎独立的并可替换的一个部分，它在良好定义的体系结构语境内满足某清晰的功能。一个独立发布的功能部分，可以通过其接口访问它的服务。软件系统中具有相对独立功能，可以明确标识，接口由规约指定，与语境有明显依赖关系，可独立部署，且多由第三方提供的可组装软件实体；软件构件需承载有用的功能，并遵循某种构件模型。

构件的要素：规格说明；一个或多个实现；受约束的构建标准；包装方法；部署方法。

构件的分类描述：枚举分类；刻面分类；属性-值分类

构件库管理系统的功能：构件的分类存储；构件的检索；构件库浏览；删除构件；构件使用情况评价

构件检索方法：规约匹配；型构匹配；术语轮廓匹配；行为采样

敏捷软件开发：致力于降低变化的成本；强调价值；强调人的作用；使用增量和迭代的开发方法

人机界面设计时经常遇到的问题：系统响应时间；用户求助设施；错误信息处理；命令标记

黄金原则：让用户拥有控制权；减少用户的记忆负担；保持界面一致

程序设计风格：编写程序时，应该力图从编码原则的角度提高程序的可读性，改善程序质量。包括源程序的文档化、数据说明、数据结构和输入输出。

软件测试的目的：发现软件中的错误和缺陷，并加以纠正。

软件测试的基本原则：所有的测试都应可追溯到客户需求；应在测试工作真正开始前的较长时间就进行测试计划；Pareto原则可应用于软件测试；测试应从小规模开始，逐步转向大规模；穷举测试不可能；为了达到最有效的测试，应有对立第三方来承担测试；应包括合理的输入条件和不合理的输入条件；严格执行测试计划，排除测试的随意性；应对每个测试结果做全方面检查；妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便；检查程序是否做了应该做的事仅是成功的一般，另一半是检查程序是否做了不该做的事；规划测试时不要设想程序中不会查出错误。

软件维护：在交付以后，修改软件系统或部件以排除故障、改进性能或其他属性或适应变更了的环境的过程。

软件可维护性：指理解、改正、调整和改进软件的难易程度。影响的主要因素有：可理解性、可测试性

可修改性、可移植性。

白盒测试：又称为结构测试，这种方法把测试对象看做一个透明的盒子，测试人员根据程序内部的逻辑结构及有关信息设计测试用例，检查程序中所有的路径是否都按部就班按预定的要求正确地工作。

黑盒测试：又称为行为测试，这种方法把测试对象看做一个黑盒子，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需求。

白盒测试方法：逻辑覆盖测试、基本路径测试、数据流测试、循环测试

逻辑覆盖测试：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖

黑盒测试方法：等价类划分、边界值分析、比较测试、错误猜测、因果图方法

测试策略：单元测试、集成测试、确认测试、系统测试

单元测试内容：接口、局部数据结构、边界条件、独立路径、错误处理路径

集成测试分类：非增量集成测试、增量集成测试（自顶向下集成测试、自底向上集成测试）

系统测试：恢复测试、安全保密性测试、压力测试、性能测试

软件调试的目的：确定错误的原因和准确位置，并加以纠正

调试方法：蛮力法、回溯法、原因排除法

软件项目管理：指软件生存周期中软件管理者所进行的一系列活动，其目的是在一定的时间和预设范围内，有效的利用人力、资源、技术和工具，使软件系统或软件产品按原定计划和质量要求如期完成。

项目管理：通过项目经理和项目组织的努力，运用系统理论方法对项目及其资源进行计划、组织、协调、控制，旨在实现项目的特定目标的管理方法体系。

软件度量：指计算机软件范围内的测量

风险管理：实际上是就是一系列管理项目风险的步骤，它标识软件项目中的风险，预测风险发生的概率以及风险造成的影响，并对所有可能出现的风险进行评估，找出那些可能导致项目失败的风险，然后才去相应的措施来缓解风险。风险管理的活动主要包括风险标识、风险预测、风险评估、风险管理和控制。

风险标识：风险管理的第一步对潜在的危险进行识别。

风险预测：评价每种风险发生的可能性或概率以及当该风险发生时所导致的后果。

软件演化：指软件在发布以后，进行的一些列活动的总称。

软件测试：设计合适的测试用例，用尽可能少的测试用例来反馈尽可能多的软件错误。

可移植性：软件迁移到另一种环境的能力。

结构化设计的基本要点：将结构化分析得到的数据流图映射成软件体系结构的一种设计方法。强调模块化、自顶向下、逐步求精、信息隐蔽、高内聚低耦合等设计抓则；分为概要设计和详细设计两大步骤。

软件需求规约：分析人物的最终产物，通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。

逆向工程：在软件生存周期中，将软件的某种形式描述转换成更抽象的活动，在软件开发时，先进行需求分析，然后进行软件体系结构设计，在进行部件级设计，继而进行编码的过程的正向工程的逆过程。

重构：在同一抽象级别上转换系统的描述形式。

设计恢复：指借助工具从已有程序中抽象出有关数据结构设计、体系结构设计和过程设计的信息。

数据字典：=定义 +与 ，或 |或 {}重复 （）可选 “”基本元素

软件需求工程：需求获取、需求分析与协商、系统建模、需求规约、需求验证以及需求管理

软件体系结构设计：