SE1－习题

1. 软件工程是如何克服软件危机的？

答：用工程化的方法来开发软件，强调管理的作用、强调项目、强调协调、强调控制

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。

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，这就是软件工程。

1. 软件的缺陷为什么在软件开发和维护过程中会扩大？抽象的概念，会在后续的过程中不断放大（逻辑上的问题）

答：

* 需求不清晰、团队工作、技术问题、项目管理
* 缺乏质量文化，不重视质量计划，对质量、资源、任务、成本等的平衡性把握不好，容易挤掉需求分析、评审、测试、等时间，遗留的缺陷会比较多
* 系统分析时对客户的需求不是十分清楚，或者和用户的沟通存在一些困难
* 开发流程不够完善，存在太多的随机性和缺乏严谨的内审或评审机制，容易产生问题。
* 文档不完善，风险估计不足等。

1. 原型开发的目的、特点和包括那几类？

答：主要优点是：及早地提供一个可运行的版本；

　　特点：① 加快需求的确定；② 简化项目管理；③ 加强用户参与与决策。

原型开发主要有三种类型，各自的目的分别是

1. 丢弃型（探索型）：这种原型的目的是弄清用户对目标系统的要求，确定希望的特性，探讨方案的可行性，主要针对开发目标模糊，用户对开发对项目缺乏经验

2. 样品型（实验性）：用于考核方案是否合适，规模说明是否可靠

3. 渐增式演化型（进化型）：这种目的不在于改进规格说明，而是将系统建造得易于变化，逐步进化最终系统，满足需求的变动

1. 简述模型在软件开发中的作用。

答：模型的意义：

1. 软件开发模型是软件开发全部过程、活动和任务的结构框架。

2. 软件开发模型能清晰、直观地表达软件开发全过程，明确规定了要完成的主要活动和任务，用来作为软件项目开发的基础。

有一套理论、方法、工具帮助开发团队与用户，或团队成员间更好地相互沟通，在客户的需求与实际的建筑技术之间找好一个契合点；帮助开发小组更好地规划系统设计，更快的开发

1. 论述过程、方法和工具在软件工程实践中的关系。

答：软件工程3要素：方法、工具、过程

方法：为软件开发提供“如何做”的技术。

工具：为软件工程方法提供自动或半自动的软件支撑环境。

过程：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。

核心关注“质量”

1. 分析比较各种软件开发模型的特征和优、缺点？

答：软件生存期模型也称为过程模型，是从软件项目需求定义直至软件运行维护为止，跨越整个生命周期的系统开发、运行和维护所实施的全部过程、活动和任务的结构框架。

典型的生存期模型有：

瀑布模型：特点，(1) 阶段间具有顺序性和依赖性；(2) 推迟实现的观点（减少返工，节省成本）；(3) 质量保证的观点（避免错误的放大）；

缺点：不适应需求快速、频繁变更的情况；

演化模型：主要针对事先不能完整定义需求的软件项目，此模型可以减少风险；为了探索可行性和弄清需求，作第一次试验开发，以取得有效的反馈信息，来支持软件的最终设计和实现；（第一次试验开发出来的软件称为原型）

螺旋模型：是瀑布模型与演化模型结合，并增加两者所忽略的风险分析；通常用来指导大型软件项目开发，它将开发划分为：制定计划、风险分析、实施开发、客户评价

喷泉模型：主要用于采用对象技术的软件开发项目， “喷泉”这个词体现了面向对象软件开发过程迭代和无缝的特性

RAD（快速）模型：使用基于构件的建造方法，强调极短的开发周期。主要阶段：业务建模、数据建模、处理建模、应用生成、测试及反复；采用第四代技术

RAD的缺陷：对于大型项目需要大量的RAD人员 、要求开发者与用户在很短时间内完成一个系统，任何一方没有完成约定，都会导致RAD项目失败。

RAD不适用于：难以适当模块化的项目、技术风险高的项目、高性能指标且该指标通过接口调整适用的系统。

V模型：瀑布模型的变形，将Coding编码阶段作为分界，分析和设计在其左侧，测试和维护在右侧，价值在于厘清各阶段的作用

增量模型：强调能拆分成相对独立的任务；能够相对独立地开发。

优点：整个产品被分解成若干个构件逐步交付地，用户可以不断地看到所开发软件的可支行中间版本。

　　　将早期增量作为原型有且于明确后期增量的需求。

　　　降低开发风险。

　　　重要功能被首先交付，从而使其得到最多的测试。

缺点：需要软件具备开放式的体系结构。

　　　容易退化为边做边改方式，使软件过程的控制失去整体性。

　　　需求难以在增量实现之前详细定义，因此增量与需求的准确映射以及所有增量的有效集成可能会比较困难。

形式化方法模型：是采用形式化的数学方法将系统描述转换成可执行的程序。

适合于对安全性、可靠性和保密性要求极高的软件系统、这些系统需要在投入支行前进行验证。

优点：由于数学方法具有严密性和准确性，形式化方法开发过程所交付的软件系统具有较少的缺陷和较高的安全性。

缺点：开发人员需要具备一定技能并经过特殊训练。

基于组件的开发模型：

组件开发技术的两个重要因素：基于组件的软件体系结构；基于组件的开发过程。

优点：充分体现软件复用的思想；实现快速交付软件。

缺点：商业组件的修改受到限制，影响系统的演化。

智能模型：也称为基于知识的软件开发模型，它是知识工程与软件工程在开发模型上结合的产物。

智能模型与其它模型不同点是它的维护不在程序一级上进行，而是在功能规约一级上进行，从而可以把精力更加集中于具体描述的表达上，这样把问题的复杂性大为降低。

统一过程RUP：初始阶段 -> 细化阶段 -> 构造阶段 -> 发布阶段

*RUP的4+1视图*：

逻辑视图Logical View：设计的对象模型

进程视图Process View：捕捉设计的并发和同步特征

物理视图Physical View：描述了软件到硬件的映射，反映了分布式特性

开发视图Development View：描述了开发环境中软件的静态组织结构

（用例视图）use-case View：架构的描述，即所做的决定，可以围绕着这四个视图来组织，然后由一些用例(use cases)或场景(scenarios)来说明

敏捷开发（agile development）

敏捷开发方法的宗旨是：沟通、简化、反馈、激励。

强调人的作用，构建起具有合作精神的、自组织的、有凝聚力量的团队。

敏捷开发是一种*开发过程*，而*不是*软件生命周期模型。

SE2－习题

项目是在既定的资源和要求的限制下，为实现某种目标而相互联系的一次性的工作任务。

项目的基本特征：明确的目标、项目的独特性、项目的时限性、项目的不确定性、结果的不可逆转性

软件项目的特点：1) 目标的渐进性；2) 项目的阶段性---强调阶段性的审核，来有效控制质量；化繁为简，各个击破，即分解目标分期实现；3) 智力密集型；4) 不确定性（不清晰的客户需求、设计不完全可以预测、不断变化的需求、不断变化的技术）

管理就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。

软件管理的主要方面是：软件项目的工作范围、可能风险、需要资源（人、硬件/软件）、要实现的任务、经历的里程碑、花费工作量（成本）、进度安排等。

1. 软件项目管理的主要方面是什么？

答：软件项目管理就是为了使软件项目能够按照预定的成本、进度、质量顺利完成，而对人员People、产品Product、过程Process和项目Project进行分析和管理的活动。其主要方面是：

1）软件项目计划与组织

2）软件项目成本管理

3）软件项目进度控制

4）软件质量保证

5）软件配置管理

6）生成软件项目管理文档

1. 软件生存期与软件项目的生命期有什么区别？

答：软件的生存周期是指从构思开始至软件不可再用结束时的时间周期。

软件项目的生存周期与软件的生存期是不同的。软件项目的生命周期是指从项目批准到交付使用的过程。而立项前的调研、系统维护、升级等可以不是项目生命周期的阶段，但是可以作为软件的生存期的一部分。

1. 简要解释软件项目计划中每一部分的目的。

答：项目范围：明确项目目标，主要功能，性能限制，系统接口，特殊要求（非功能需求），开发概述。

　　资源：根据不同活动的特点计划分配资源（人力资源、硬件资源、软件资源、知识资源、工具包）。

　　进度安排：定义全部项目任务，识别出关键任务，跟踪关键任务的进展状况，保证能及时发现拖延进度的情况。

　　估算：确定时间、人时及项目成本，对活动、工作成功及各种角色（包括用户）简历适当的预期，确定项目里程碑、估算增加需求、改变资源、用项目监控因素来影响产品的质量的影响。

　　度量：获得对过程、产品、资源、环境的理解，确定以后预测的基线和模型；通过理解过程、产品各要素之间的关系建立模型，由已知的要素推算、估计其他要素，以合理分配资源、合理制定计划；根据得到的量化信息，可以帮助识别障碍物、查找问题根源，以及能提高产品质量和过程效率的其他方法。以以前的量化信息进行比较，可以证实这些方法是否有效。

1. 简述应对项目风险的主要方法及应注意的问题。

答：风险避免：如果采用主动的策略来处理风险，则避免风险总是最好的策略；可以通过建立风险环节计划来达到。

　　风险监控：随着项目的进展，开始进行风险监控活动；项目管理者监控某些能指出风险概率正在变高还是变低的因素。

　　风险管理及意外事件计划：假设换件风险的努力失败了，风险变成了现实，就需要风险意外计划。

1. 成本估算方法有哪几种？

答：估算方法的分类：基于经验（由熟悉该领域的专家组成评估小组，拆分WBS，然后进行估算）、基于模型（算法驱动式模型FP/COCOMO<构造性成本模型>、数据驱动式模型、复合式模型< Bayesian方法、类比方法、人工神经网络等>）

1. 影响软件成本的主要因素有哪些具体方面？

答：软件的规模、复杂度、开发工具、开发人员的业务素质、软件工程管理水平及开发工作环境等。

1. 软件质量的特性有哪些，怎样度量？

答：软件质量的6个特性：功能性，可靠行，易使用性，效率，可维护性，可移植性。通过产品质量、过程中质量和软件维护质量度量等三种方法来度量。

度量：根据一定的规则，将数字或符号赋予系统、构件、过程等实体的特定属性，从而能清晰地理解该实体及其属性。

1. 软件技术审查和管理复审的作用是什么？可以起到发现错误和缺陷，进而消除它们的作用

答：1、由一组评审者按照规范的步骤对软件需求、设计、代码或其他技术文档进行仔细地检查，以找出和消除其中的缺陷。

作用：

（1）发现软件在功能、逻辑、实现上的错误；

（2）验证软件符合它的需求规格；

（3）确认软件符合预先定义的开发规范和标准；

（4）保证软件在统一的模式下进行开发；

（5）便于项目管理。

2、管理复审是指系统地评价鉴定全部管理工作的一种控制方法，它侧重于管理决策方面的审核，包括对组织中关于计划、工程技术、生产、营销、人事、会计以及财务方面的审核。

管理复审的主要任务是，在软件生命周期的每个阶段结束时，对工程项目的成本、实际花费的经费、投资回收的前景、项目的进度等经济因素从管理角度进行审查。从而找出在成本、进度、质量上存在的问题，为后续改进提供问题基础。

1. 论述软件项目特点和管理要点。

答：软件项目的特点包括：发展快，经验失效快，复杂度高，主要依靠人的脑力劳动，过程的可见性差，结果难于测试，市场模式不完善。

管理要点：软件管理坚持平衡原则，高效原则，分解原则，实施控制原则，分类管理原则，分类管理原则。简单有效原则，规模控制原则。

1. 论述软件估算的困难和解决方法。

答：软件估算的困难：软件是不可见的，每个软件项目都是独一无二的，现实的估计必须考虑项目的约束、可用的资源、人缘的技能等因素，进一步增加的估计的复杂性。因此估算没有历史数据可以参照。估算目标的度量选择有难度。偏见是人们更倾向于有低估所需的时间，估值知识实际值的1/2或更低。估算准确与需求不定带来矛盾。另外估算中存在误区。

解决办法：采用合理的估算技术；使用独立的估算者；依靠实际数据反馈，做调整。不确定因素避免方法：利用历史数据修正；多重估算，并进行估算结果的归纳；估算结果的平均值，加权平均数，采用修正——迭代式估算等

1. 影响软件质量的因素分哪两大类？直接可度量的和间接可度量的

答：1）可以直接测度的因素（例如，每个功能点的错误）；2）能间接测度的因素（例如，可用性和可维护性）

1. 项目整体管理和项目范围管理各是什么，它们与软件生命周期管理有什么区别？

答：项目整体管理是唯一贯穿启动到收尾所有过程组的知识体系，从项目启动到项目收尾项目整体管理都得管。

　　项目范围管理是对项目应该包括什么和不应该包括什么进行相应的定义和控制。

　　软件生命周期管理贯穿软件产品构思开始至软件不可用结束的时间周期。

SE3－习题

1. CMMI与ISO9000有什么异同？

答：联系：两者都共同着眼于质量和过程管理，目前2000版的ISO更多的和CMMI有直接对应的关系，甚至是大量的CMMI4和CMMI5级的要求。

　　区别：CMMI是专门针对软件产品开发和服务，而ISO9000涉及的范围则相当宽。

　　　　　CMMI强调软件开发过程的成熟度，即过程的不断改进和提高；而ISO9000则强调可接收的质量体系的最低标准。

相同点：

　　1）ISO 9000和CMMI都关注软件产品质量和过程改进。

　　2）CMMI和ISO9000需要具体的软件管理规范支持，两者都定义了要做什么，但都没有定义如何做，都需要公司有自己的软件工程管理支持，都可用作为软件企业的过程改善框架。

不同点：

　　1）CMM和ISO9001在抽象程度上不一样。CMMI更具体些，ISO9000更抽象些。CMM侧重技术管理的过程改进，ISO9000覆盖面广，涉及公司各个职能部门。ISO9001重在整体，CMMI则强调企业内部素质。CMMI是专门针对软件工业的，而ISO9000则面向所有工业。

　　2）CMMI和ISO9000在质量要素条款组织和描述方式上不一样。ISO9000是确保每一个产品要素和相关服务的质量可重复地被保证，针对合同环境下设计、开发、生产、服务等环节，给出了所需要的最基本质量要素。ISO9000根据一个企业的质量体系中是否覆盖了所有要求的质量要素（以文档化的形式），且这些要素是否有效地按定义方式实施来判断该企业是否符合ISO9000要求。CMMI的结构是层次化的结构，ISO9000结构是简单的线性结构，包含20个质量要素，除“管理职责”和“质量体系”两个质量要素外，其余18个均为过程要素。

1. 简述RUP（Rational Unified Process）的基本内容。

答：初始阶段：建立一个业务案例或理由，确定核心需求，初步风险评估，原型。

　　细化阶段：建立构架，建立综合计划，交付的是系统行为模型（系统环境、场景、领域模型），以及基于领域模型的基线产品构想、开发、计划、评估标准产品发布描述，其他还有用户手册、测试计划等。

　　构造阶段：编程，产品包括可执行代码、系统和用户手册、部署计划、质量保证结果等工作产品。

　　发布阶段：已完成的所有制品。

1. 什么是CMMI过程域？举例说明其含义。CMMI – 软件能力成熟度模型

答：CMMI过程域是互相关联若干软件实践活动和有关基础设施的集合。

成熟度等级2已管理级（7个）：需求管理、项目计划、项目监督和控制、供应合同管理、度量和分析、过程和产品质量管理、配置管理

成熟度等级3已定义级（11个）：需求开发、技术解决方案、产品集成、验证、确认、组织级过程焦点、组织级过程定义、组织级培训、集成化项目管理、风险管理、决策分析和解决方案

成熟度等级4量化管理级（2个）：组织级过程性能、项目定量管理

成熟度等级5优化级（2个）：组织级改革和实施、因果分析和解决方案

1. CMMI的连续式（continuous representation）模型和阶段式（staged representation）模型各有什么优缺点？

答：连续式表示法涉及选择一个特定的过程域进行改进，以及确定对这个过程域预期达到的能 力等级。在这种情况下，过程已得到执行或仍不完整，这一点是重要的。因此，采用名称 “不完整级”作为连续式表示法的起点。

　　连续式分组：过程管理、项目管理、工程、支持

阶段式表示法涉及选择某一成熟度级别下的多个过程域进行改进；单个过程已得到执行或 仍不完整就不是首要关注点。因此，采用名称“初始级”作为阶段式表示法的起点。

阶段式分组：初始级、已管理级、已定义级、量化管理级、优化级

1. 在什么情况下产品质量可能决定于开发团队的质量？举例说明什么类型的软件产品特别依赖于个人的天赋和能力。

答：软件需求不明确、需求变化快、使用全新的技术、面对新兴的行业等情况下，软件产品的质量可能决定于开发团队的质量。

　　互联网软件、游戏软件等软件产品特别依赖于个人的天赋和能力。

1. 过程评估在软件过程改进中有什么作用？

答：通过评估可以帮助组织对当前的软件过程环境与基础有一个正确的认识。

　　实际上，评估通常扮演了一个改进活动计划的催化剂的角色。

　　正式的评估可以为组织赢得声誉。

1. 软件技术审查和管理复审的作用是什么？同第14题
2. 软件工程的文档分哪几大类，主要作用是什么？管理类档、开发类文档、用户类文档

答：分为开发文档和产品文档两大类

开发文档包括：《功能要求》、《投标方案》、《需求分析》、《技术分析》、《系统分析》、《数据库文档》、《功能函数文档》、《界面文档》、《编译手册》、《QA文档》、《项目总结》等。

产品文档包括：《产品简介》、《产品演示》、《疑问解答》、《功能介绍》、 《技术白皮书》、《评测报告》、《安装手册》、《使用手册》、《维护手册》、 《用户报告》、《销售培训》等。

主要作用：

* 提高软件开发过程的能见度。把开发过程中发生的事件以某种可阅读的形式记录在文档中。
* 管理人员可以吧这些记载下来的材料作为检查软件开发进度和开发质量的依据，实现对软件开发的工程管理。
* 提高开发效率。软件文档的编制，使得开发人员对各个阶段的工作都进行周密思考、全盘权衡、减少返工。并且在开发早期发现错误和不一致性，便于及时加以纠正。
* 作为开发人员在一定阶段的工作成果和结束标志。
* 记录开发过程中有关信息，便于协调以后的软件开发、使用和维护。
* 提供对软件的运行、维护和培训的有关信息，便于管理人员、开发人员、操作人员、用户之间的协作、交流和了解。使得软件开发活动更科学，更有成效。
* 便于潜在用户了解软件的功能、性能等各项指标，为他们选购符合自己需要的软件提供依据。

SE4－习题

1. 简述软件分析员在系统分析中的任务和作用。

答：系统分析员（system analyst）是对大型、复杂的信息系统建设任务中，承担分析、设计和领导实施的领军人物。 要做好与客户之间的关系，同时对客户的需求要正确的理解，要选择合适的开发技术，同时做好与客户间沟通交流，学会说服对方。

工作内容：

　　1、参与用户需求调研、负责系统体系结构、功能、性能的分析和总体设计工作；

　　2、负责项目的开发流程管理，进行项目的计划、管理、跟进工作；

　　3、参与并指导开发工程师完成系统详细设计和开发工作，解决相应业务、技术难题；

　　4、参与编制用户手册、协助客户的系统软件、硬件平台的安装实施工作；

　　5、制定项目文档格式，编写项目管理规范要求的相关文档

作用：系统分析工作是解决一个问题的工作，目标是将一个对计算机应用系统的需求转化成实际的物理实现。

1. 简述总体设计的一般过程？

答：总体设计又称为概要设计或初步设计。总体设计过程通常由系统设计和结构设计两个主要阶段组成。系统设计：确定系统的具体实现方案。结构设计：确定软件结构。

9个步骤：设想供选择的方案；选取合理的方案；推荐最佳方案；功能分解；设计软件结构；设计数据库；制定测试计划；书写文档；审查和复审

1. 好的软件体系结构设计应遵循哪些原则？

答：软件系统的体系结构设计的原则是满足合适性、结构稳定性、可扩展性、可复用性

1. 详细设计的任务是什么？

答：详细设计是软件工程中软件开发的一个步骤，就是对概要设计的一个细化，就是详细设计每个模块实现算法，所需的局部结构。

详细设计包括业务对象设计、功能逻辑设计、数据库设计和界面设计等工作。详细设计是系统实现的依据，需要考虑所有的设计细节。

详细设计的主要任务是设计每个模块的实现算法、所需的局部数据结构。详细设计的目标有两个：实现模块功能的算法要逻辑上正确和算法描述要简明易懂。

主要任务：① 确定软件各个组成部分内算法以及部分的内部数据组织。② 选定某种过程表达形式来描述各种算法。③ 进行详细设计评审（课件PPT）

1. 模块化的三个重要特征是什么，阐明各自的作用。

答：三个重要特征是：

1. 相对独立性，可以对模块单独进行设计、制造、调试、修改和存储，这便于由不同的专业化企业分别进行生产；
2. 互换性，模块接口部位的结构、尺寸和参数标准化，容易实现模块间的互换，从而使模块满足更大数量的不同产品的需要；
3. 通用性，有利于实现横系列、纵系列产品间的模块的通用，实现跨系列产品间的模块的通用。

模块化一个重要原则是信息尽量隐蔽在模块中

1. 举例说明各种耦合情况和各种内聚情况。

答：

耦合：指模块间相互关联的程度。它取决于各个模块之间接口的复杂程序、调用模块的方式以及哪些信息通过接口。

耦合性由低到高分别是（即独立性由强到弱）：非直接耦合、数据耦合、标记耦合、控制耦合、外部耦合、公共耦合、内容耦合

尽量使用数据耦合，少用控制耦合，限制公共环境耦合的范围，完全不用内容耦合。

内聚：指一个模块内部各个元素之间关系的紧密程度。

内聚性由高到低分别是（同模块独立性）：功能内聚、信息内聚、通信内聚、过程内聚、时间内聚、逻辑内聚、巧合内聚

度量模块独立性的2个定性的标准： 耦合和内聚；模块独立性是模块抽象和信息隐藏的直接结果。模块独立性是软件质量好坏的关键。

1. 简述层次方框图与软件结构图的异同点。

答：相同：形式类似。

　　不同：

1. 层次图描绘软件的层次结构，层次方框图描绘的是数据结构。
2. 层次图的方框表示模块或子模块，层次方框图的方框表示数据结构整体或其子集。
3. 层次图的连线表示调用关系，层次方框图表示组成关系。
4. 面向数据流的设计方法包含那些步骤？需要画图\*\*\*

答：面向数据流的设计方法把信息流映射成软件结构，信息流的类型决定了映射的方法，根据系统的数据流进行的设计，又称结构化设计。

数据流包括：变换型、事务型

变换型数据流图呈一种线性状态，信息沿输入通路进入系统，同时由外部形式变换成内部形式，进入系统的信息通过变换中心，经加工处理以后再沿输出通路变换成外部形式离开软件系统。（要先找到变换中心，然后在变换中心上增加一个主模块，在变换中心左侧的均为输入，右侧的为输出）

某个加工将它的输入分离成一串发散的数据流，形成许多活动路径，并根据输入的值选择其中一条路径。具有这样特征的数据流图称为事务型数据流图。

通常把完成选择分派任务的部分叫做事务处理中心。

1. 什么是结构化分析，有什么特点？

答：结构化**分析**方法（SA，Structure Analysis）是面向数据流进行需求分析的方法，适合于数据处理类型软件的需求分析。

特点：抽象概念、面向数据流、自顶向下逐层分析

结构化**设计**方法（Structure Design，SD）是基于模块化、自顶向下逐层细化、结构化程序设计等程序设计技术基础上发展起来的。

结构化设计方法的主要内容：自顶向下，逐步细化；软件结构；程序结构；数据结构；软件过程；模块化；抽象化；信息隐蔽

主要特点是快速、自然和方便。成功率较高，发展较为成熟；简单、易掌握，适应于瀑布模型；特别适合于数据处理领域中的应用，对规模大的项目，特别复杂的应用不太适应。难于解决软件重用问题，难于适应需求的变化。

1. 解释体系结构中深度、宽度、扇出、扇入对软件的影响。

答：深度表示软件结构中控制的层数。

　　宽度是软件结构中同一个层次上的模块总数的最大值。

　　一个模块的扇入是指直接调用该模块的上级模块的个数。

　　一个模块的扇出是指该模块直接调用的下级模块的个数。

设计原则：低扇出高扇入，扇入大表示模块的重用性高，利用率高。扇出大表示模块的复杂度高，所以要高扇入低扇出。

1. 事务型软件结构图有什么特点，原因是什么？

答：事务型结构图的特点：数据流是以事务为中心的

1. 论述需求工程过程，说明各阶段关系。

答：需求工程过程包括需求开发和需求管理两个部分，

需求开发的一般过程分为需求获取、需求分析、编写需求规格说明书（SRS）、需求验证四个阶段；

需求管理则主要包括需求基线的建立、需求变更控制以及需求跟踪等活动。

各阶段关系：

*需求获取*是通过与用户的交流，对现有系统的观察及对任务进行分析，从而开发、捕获和修订用户的需求；

*需求分析*是通过之前需求获取所得到的信息，为最终用户所看到的系统建立一个概念模型，作为对需求的抽象描述；由需求模型构件生成精确的形式化的描述，即需求规格说明书，它将作为用户和开发者之间的一致协议；

需求分析的结果应该通过评审、测试等手段验证它的正确性、完整性和一致性，这就是*需求验证*。

贯穿于整个过程中，*需求管理*中最基本的任务则是明确需求，并使所有相关人员达成共识；建立需求跟踪能力联系链，确保所有用户需求被正确地应用，并且在需求发生变更时，能够完全地控制其影响范围，始终保持产品与需求的一致性。

1. 什么是软件性能？论述软件开发中如何提高软件的性能。

答：软件的性能是软件的一种非功能特性，它关注的不是软件是否能够完成特定的功能，而是在完成该功能时展示出来的及时性。

由于感受软件性能的主体是人，不同的人对于同样的软件能有不同的主观感受，而且不同的人对于软件性能关心的视角也不同。包括系统响应时间和应用延迟时间、吞吐量、并发用户数、资源利用率。

1. 程序的编码风格主要体现在哪几个方面？

答：程序的编码风格即为程序设计风格或编程风格，其主要作用是使无论是程序本人还是其他人，都能比较容易的阅读、理解及修改源代码；

　　主要表现在四个方面：源程序文档化、数据说明方法，表达式和语句结构及输入/输出方法；

SE5－习题1

1. 什么是对象，对象有哪几种形式？

答：对象是指一组属性以及这组属性上的专用操作的封装。属性通常是一些数据，有时它也可以是另一个对象。

　　对象可以表现有形的实体，作用，事件，性能说明

1. 什么是类，类与对象间是什么关系？

答：类是一种复杂的数据类型，它是将不同类型的数据和与这些数据相关的操作封装在一起的集合体。类是对某一类对象的抽象，而对象是某一种类的实例，对象与类实例是同义词。

1. 面向对象方法特征有哪些？

答：封装性、继承性、多态性

1. 面向对象设计涉及那几个主要活动？

答：分析活动、设计活动、编码活动

1. OOD如何体现抽象信息、隐藏和模块化这三个概念的？

答：通过类的引入、封装，把数据结构和操作这些数据的方法紧密地结合在一起所构成的模块，模型是现实的抽象，对象的封装性实现隐藏。

1. 举例说明类的整体部分结构（”is a”）和类的组装结构（”has a”）。

答：一般-特殊（generalization-specialization）：“is a ”的关系；机动车-卡车，是一般与特殊的关系。

　　整体-部分（whole-part）：“has a ”的关系；车子-轮子，是整体与部分的关系。

1. 领域分析的目标是什么，依据是什么？

答：领域分析指对特定问题空间内的知识进行提取、整理和建模，它为以后涉及该问题空间的系统开发提供复用基础。

领域分析的目标是为领域产品族中相似应用系统建立核心资产的过程，用于应用工程的复用。

其依据是领域知识源：已有文件、已有应用、客户考察、专家建议、目前/未来的需求。

1. 类的开发有几个途径，如何进行类的开发？

答：一是新建基类，二是通过类间的继承来进行类的开发

1. 什么是对象关系模型和对象行为模型，有什么不同？

答：对象关系模型是描述对象与对象之间的关系的模型。

　　对象行为模型指的是对象与对象之间相互关系的行为模型。

　　关系模型和对象模型存在概念上的阻抗不匹配，但是在关系数据库的存储模型上是一致的，无论你从关系模型的三大范式理论出发，还是从对象模型的ORM理论出发，最终一定会得到一致的数据库表设计。

1. 什么是面向对象开发过程，讨论各阶段任务和要点。

答：面向对象开发过程就是指利用面向对象方法进行的软件开发的完整流程，主要分为四个阶段：分析、设计、编程、测试。

面向对象分析包括需求分析和需求模型化两个部分。其主要作用是明确用户的需求，并用标准化的面向对象模型来规范地表达这一需求，最后形成面向对象的分析模型。它是系统设计的依据。

面向对象设计的任务是把分析阶段得到的问题域需求转变成符合要求的、抽象的系统实现方案，主要分为两个阶段：系统设计和对象设计。要点设计针对问题域子系统、人-机交互子系统、任务管理子系统、数据管理子系统、类中的服务和关联。

编程和测试则分别是由开发人员编写程序具体代码并在系统代码开发完成后对系统整体性能和功能进行测试。

1. 用覆盖的观点讨论面向对象的软件测试策略。

答：语句覆盖（基本路径覆盖）：保证所有语句至少执行一次；

判定（分支）覆盖：保证所有的判定结果（真假结果）至少执行一次；（不用组合叠加）

条件覆盖：保证所有的判定中的条件至少执行一次；

判定（分支）/条件覆盖：保证所有的判定结果，以及判定的条件至少执行一次；（结果的真假至少要有一次，条件的真假也至少有一次）

条件组合覆盖：保证所有判定的条件的结果组合，至少要执行一次；（条件的真假，组合情况至少要有一次）

路径覆盖：

还有其他的覆盖方式则是将这前三种覆盖进行组合，使得测试覆盖更加全面，可以覆盖到程序中所有可能的路径。由此可以看出只有对程序内部十分了解才能进行适度有效的覆盖测试。但是贯穿在程序内部的逻辑存在着不确定性和无穷性，尤其对于大规模复杂软件。因此我们不能穷举所有的逻辑路径，即使穷举也未必会带来好运（穷举不能查出程序逻辑规则错误，不能查出数据相关错误，不能查出程序遗漏的路径）。所以正确使用覆盖（白盒）测试，就要先从代码分析入手，根据不同的代码逻辑规则、语句执行情况，选用适合的覆盖方法。任何一个高效的测试用例，都是针对具体测试场景的。逻辑测试不是片面的测试正确的结果或是测试错误的结果，而是尽可能全面地覆盖每一个逻辑路径。这样才能使得测试的工作量与准确性达到一个合理的平衡。

1. 结合软件工程要素，论述面向对象方法的思想。

答：软件工程的方法（开发方法）、工具（支持方法的工具）、过程（管理过程）称为软件工程的三要素。

方法支撑过程和工具；过程和工具促进方法学的研究。

面向对象方法OOM是一种运用对象、类、消息传递、继承、封装、聚合、多态性等概念来构造软件系统的软件开发方法。

它的优点是：所开发的程序是面向对象程序，直接描述客观世界的对象及其相互关系；具有模块性、继承性和类比性；可广泛支持对象的重用；十分有利于开发过程的质量控制；极大地减小了系统维护的复杂程度。

缺点是：面对大型系统时，若缺乏整体系统设计划分，易造成系统结构不合理、各部分关系失调等问题；只能在现有业务基础上进行分类整理，较难从科学管理角度进行理顺和优化；需要一定的软件支持环境。

因此，通过三要素的关系可以看出，面向对象方法决定了它需要一定的开发工具和严格合理的管理过程来支持，同时，在这两个条件得到满足的情况下，面向对象的方法将能够极大的方便、优化工程的开发以及管理维护。

SE5－习题2

1. 什么是OOA建模语言，它应该包括哪些方面？建模语言UML

答：Object-Oriented Analysis（面向对象分析方法）是确定需求或者业务的角度，按照面向对象的思想来分析业务。

分析模型

1、对象模型：对用例模型进行分析，把系统分解成互相协作的分析类，通过类图/对象图描述对象/对象的属性/对象间的关系，是系统的静态模型

2、动态模型：描述系统的动态行为，通过时序图/协作图描述对象的交互，以揭示对象间如何协作来完成每个具体的用例，单个对象的状态变化/动态行为可以通过状态图来表达

3、功能模型，即用例模型à作为输入。

1. 什么是Use-Cases模型，主要作用是什么？

答：User Cases Mode是系统既定功能及系统环境的模型。

　　用例模型最重要的作用是将系统行为传达给客户或最终用户。

1. UML中有哪些动态建模的工具，各表达什么内容？

答：时序图／顺序图：顺序图用二维表来表示交互，纵向是时间轴，横向是参与的

角色以及它们交换的消息。（重点在消息的时间顺序）

　　协作图：与时序图一样，协作图表示对象间的动态协作关系。它除了说明消息的交互外，还显示对象及其间的关系。（重点在交换消息与对象间的关系）

　　状态图：是对单个类的对象的生命周期进行建模，描述了对象时间上的动态行为，每个对象被认为是事件驱动的孤立实体。

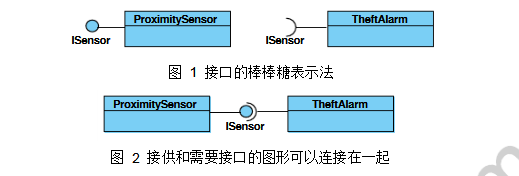
活动图：活动图是用状态机对工作流进行建模的特殊形式，它和流程图很类似，不过它支持并发控制。

1. UML中的接口类有什么特点，举例说明。

答：接口描述了一组实现这一接口的分类器所必须共同遵守的合同，这种合同包括结构上的也包括行为上的，此外，接口是不能实例化的。接口实现则是一种依赖关系，其描述了接口与分类器之间的一种特殊的实现关系。

图1示例了接口的棒棒糖表示方法，其中“棒棒糖”表示与之相连的类提相应的接口，也就是说相关联的类实现了这一接口。

从图中我们可以看出，ProximitySensor类实现了ISensor接口，而TheftAlarm类则需要一个ISensor接口。由于两个类一个提供，而另一个需要，因此我们可以将其连接在一起，如图2所示。



SE6－习题

1. 什么时候测试结束，为什么？（课堂讨论：测试是不可能穷尽所有情况的，当满足某些指标条件满足时就可以停止测试了。测试通常在软件交付时或是测试预算用完时就停止了。）

答：通过在测试过程中进行风险、测试、覆盖率、缺陷和信心这5个方面的评估，并和测试计划中的要求进行比较，若能满足要求，就结束测试。

在软件消亡之前，如果没有测试的结束点，那么软件测试就永无休止，永远不可能结束。软件测试的结束点，要依据自己公司具体情况来制定，不能一概而论。个人认为测试结束点由以下几个条件决定：

1) 基于“测试阶段”的原则

2) 基于“测试用例”的原则

3) 基于“缺陷收敛趋势”的原则

4) 基于“缺陷修复率”的原则

5) 基于“验收测试”的原则

6) 基于“覆盖率”的原则

7) 基于“项目计划”的原则

8) 基于“缺陷度量”的原则

9) 基于“质量成本”的原则

10) 基于“测试行业经验”的原则

1. 如何估算程序中的错误？（SE6 PPT page 8）

答：一般采用传统的估算方式，1000行代码含有6个bug，Linux Kernel约有15000个bugs，XP的bug数至少两倍于这个数。

1. 软件的缺陷为什么在软件开发和维护过程中会扩大？同第2题
2. 什么叫测试用例，如何设计测试用例（SE6 PPT page 27）

答：**测试用例**是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。

编写测试用例主要分为白盒测试和黑盒测试。

**白盒测试**，又称结构测试，它把测试对象看做一个透明的盒子（glass box），它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。**白盒测试方法有**：1) 逻辑覆盖测试是通过对程序逻辑结构的遍历实现程序的覆盖。从覆盖源代码的不同程度可以分为以下**六个标准**：语句覆盖、判定覆盖（又称为分支覆盖）、条件覆盖、判定-条件覆盖（又称为分支-条件覆盖）、条件组合覆盖和路径覆盖。2) 基本路径测试用例设计：程序的控制流图、程序环路复杂性、图矩阵；3) 条件测试、数据流测试、循环测试等

**黑盒测试**，注重测试软件功能性需求。黑盒测试用于发现：功能不对或遗漏、界面错误、数据结构或外部数据库访问错误、性能错误、初始化和终止错误。**黑盒测试方法**：等价划分（Equivalence Classes）、边界值分析（Boundary Value Analysis）、错误推测法（error guessing methods）、因果图（cause effect graphing）、决策表技术（decision table techniques）、基于图的测试（系统模型）

1. 简述回溯测试和作用。

答：**回归测试**是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。

自动回归测试将大幅降低系统测试、维护升级等阶段的成本。

回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。因此，通过选择正确的回归测试策略来改进回归测试的效率和有效性是非常有意义的。

1. 纠错技术中的归纳法和演绎法的中心思想是什么？

答：**归纳法**：归纳法或归纳推理，有时叫做归纳逻辑，是根据对某类事务中具有代表性的部分对象及其属性之间必然联系的认识，得出一般性结论的方法。

归纳法是从特殊到一般，**优点**是能体现众多事物的根本规律，且能体现事物的共性。**缺点**是容易犯不完全归纳的毛病。

**演绎法**：与归纳法相反，是从既有的普遍性结论或一般性事理，推导出特殊性结论的一种方法。由较大范围，逐步缩小到所需的特定范围。

演绎法是从一般到特殊，**优点**是由定义根本规律等出发一步步递推，逻辑严密结论可靠，且能体现事物的特性。**缺点**是缩小了范围，使根本规律的作用得不到充分的展现。

1. 什么是边界值分析，与等价类划分有什么不同？（SE6 PPT page 70）

答：实践表明，输入域的边界比中间更容易发生错误，作为等价类划分法的补充边界值分析（boundary value analysis， BVA）可作为一种测试方法。

边界值分析法与等价类划分法区别：

(1) 边界值分析不是从某等价类中随便挑一个作为代表，而是使这个等价类的每个边界都要作为测试条件。

(2) 边界值分析不仅考虑输入条件，还要考虑输出空间产生的测试情况。

1. 什么是软件测试中逻辑覆盖，一般软件测试至少应保证哪些覆盖？（SE6 PPT page 33）

答：**逻辑覆盖测试**是通过对程序逻辑结构的遍历实现程序的覆盖。从覆盖源代码的不同程度可以分为以下六个标准：语句覆盖、判定覆盖（又称为分支覆盖）、条件覆盖、判定-条件覆盖（又称为分支-条件覆盖）、条件组合覆盖和路径覆盖。

一般软件测试至少应保证语句覆盖

1. 比较在集成测试中，自顶向下与自底向上、深度优先与宽度优先的优缺点。（SE6 PPT page 109）

答：

**自顶向下的集成**是从主控模块（主程序，即根结点）开始，按照系统程序结构，沿着控制层次从上而下，逐渐将各模块组装起来。在从上向下的集成测试过程中，需对那些未经集成的模块开发桩模块。在集成过程中，可以采用宽度优先或深度优先的策略向下推进。

**自底向上的集成**是从最底层模块（即叶子结点）开始，按照调用图的结构，从下而上，逐层将各模块组装起来。在从下而上的集成测试环境中，需对那些未经集成测试的模块开发驱动模块。

自顶向下：可在测试早期实现并验证系统主要功能不需驱动模块 ，但需要桩模块。

自底向上：设计测试用例容易，但要到最后程序才能作为一个整体。

1. 简述黑盒子测试和白盒子测试及其适用性。

答：**白盒测试**，又称结构测试，它把测试对象看做一个透明的盒子，它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。主要适用于测试的前期，在单元测试阶段。

**黑盒测试**，注重测试软件功能性需求。黑盒测试用于发现：功能不对或遗漏、界面错误、数据结构或外部数据库访问错误、性能错误、初始化和终止错误。适用于测试的后期，它不考虑控制结构，而是注重信息域。

1. 简述软件测试中桩模块和驱动模块的作用。（课堂讨论 & SE6 PPT page 99）

答：软件测试中桩模块和驱动模块者属于辅助模块。在进行单元测试中，进行测试的模块（部分），它会一些模块被调用，也会需要调用其他模块，则在测试中需要设计调用模块和被调用模块，使得测试通顺利进行，则称调用的模块为驱动模块，被调用的模块为桩模块。

**桩模块**的使命除了使得程序能够编译通过之外，还需要模拟返回被代替的模块的各种可能返回值（什么时候返回什么值需要根据测试用例的情况来决定）。

**驱动模块**的使命就是根据测试用例的设计去调用被测试模块，并且判断被测试模块的返回值是否与测试用例的预期结果相符。

1. 简述渐增式测试方法与非渐增式测试方法的优劣。（SE6 PPT page 101）

答：**非渐增式测试**：一次就把所有通过了单元测试的模块组合在一起进行全程序的测试。缺点：发现错误难以诊断定位，又称“莽撞测试”。

**渐增式测试**：从一个模块开始，测一次添加一个模块，边组装边测试，以发现与接口相联系的问题。

(1) 非渐增式方法把单元测试和集成测试分成两个不同的阶段，前一阶段完成模块的单元测试，后一阶段完成集成测试。而渐增式测试往往把单元测试和集成测试合在一起，同时完成。

(2) 非渐增式需要更多的工作量，因为每个模块都需要驱动模块和桩模块，而渐增式利用已测试过的模块作为驱动模块或桩模块，因此工作量少。

(3) 渐增式可以较早地发现接口之间的错误，非渐增式最后组装时才发现。

(4) 渐增式有利于排错，发生错误往往和最近新加入的模块有关，而非渐增式发现接口错误推迟到最后，很难判断是哪一部分接口出错。

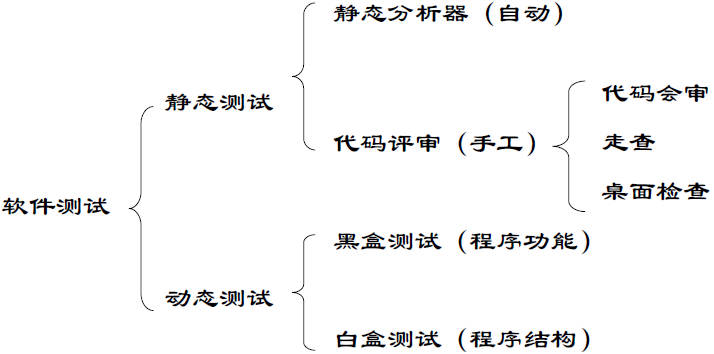
(5) 渐增式比较彻底，已测试的模块和新的模块再测试。

(6) 渐增式点用时间较多，但非渐增式所需更多的驱动模块和桩模块也占用一些时间。

(7) 非渐增式开始可并行测试所有模块，能充分利用人力，对测试大型软件很有意义。

1. 软件测试分几种类型，主要解决什么问题？（SE6 PPT page 18）

答：**软件测试类型**：



1. 什么是软件可测试性？讨论软件工程与软件可测试性。（SE6 PPT page 12）

答：**软件的可测试性**是指软件发现故障并隔离、定位其故障的能力特性，以及在一定的时间和成本前提下，进行测试设计、测试执行的能力。（软件可测试性就是一个计算机程序能够被测试的容易程度。）

·可操作性－运行地越好，被测试的效率越高；

·可观察性－所看见的，就是所测试的；

·可控制性－对软件的控制越好，测试越能够被自动执行与优化；

·可分解性－通过控制测试范围，能够更好地分解问题，执行更灵巧的再测试；

·简单性－需要测试的内容越少，测试的速度越快；

·稳定性－改变越少，对测试的破坏越小；

·易理解性－得到的信息越多，进行的测试越灵巧；

1. 论述软件测试的基本任务，方法和策略及其在现代软件工程的地位。

答：**测试**是使用人工和自动手段来运行或检测某个系统的过程，其目的在于检验系统是否满足规定的需求或弄清预期结果与实际结果之间的差别。（SE6 PPT page 9）

软件测试的**基本任务**：是为了发现软件产品所存在的软件缺陷（bug），从而纠正（fix）这些软件缺陷，使软件系统更好地满足用户的需求（SE6 PPT page 5）

**方法**：白盒测试与黑盒测试、自顶向下与自底向上、深度优先与广度优先等

**策略（一般原则）**：（SE6 PPT page 11）

* 尽早和不断地进行软件测试
* 测试用例合理
* 程序员避免检查自己的程序
* 用例应包括不合理的输入条件
* 注意错误的群集现象（80%错误来自20%的模块）
* 严格执行测试计划
* 对测试结果作全面检查
* 妥善保存测试计划和用例

**软件测试的阶段**：

单元测试：主要测试的是各个方法实现的功能和方法的限制条件等是否正确。

集成测试：主要测试接口，看集成时链接是否正确

系统测试：就是将所有的开发代码组合完整测试软件是否能满足用户的要求

确认测试：是为了验证软件是符合用户的需求和为满足用户需求更改方法后的验证测试

（回归测试：软件修改后，再次验证问题的测试）

1. 论述软件测试与软件质量的关系。

答：**测试**是使用人工和自动手段来运行或检测某个系统的过程，其目的在于检验系统是否满足规定的需求或弄清预期结果与实际结果之间的差别。（SE6 PPT page 9）

软件测试的**基本任务**：是为了发现软件产品所存在的软件缺陷（bug），从而纠正（fix）这些软件缺陷，使软件系统更好地满足用户的需求（SE6 PPT page 5）

因此，软件测试是保证软件质量的一个重要环节。

软件测试能够找出软件缺陷，确保软件产品满足需求。测试可以查找错误并进行修改，从而提高软件产品的质量。但是测试不是质量保证。

软件质量保证则是避免错误以求高质量，并且还有其他方面的措施以保证质量问题。

SE7－习题

1. 如何理解敏捷的理念？

答：敏捷团队依靠变化来获取活力。

团队几乎不预先设计，因此，不需要一个成熟的初始设计。他们更愿意保持设计尽可能的干净、简单，并使用许多单元测试和验收测试作为支援。

设计灵活和易于理解，利于持续地改进设计，以便每次迭代生成的系统都最适合于那次迭代的需求。

采取一些OO的设计原则来避免软件“腐化”。

1. 敏捷的定义。

答：敏捷开发(Agile Development)是一种以人为核心、迭代、循序渐进的开发方法。

怎么理解呢？首先，我们要理解它不是一门技术，它是一种开发方法，也就是一种软件开发的流程，它会指导我们用规定的环节去一步一步完成项目的开发；而这种开发方式的主要驱动核心是人；它采用的是迭代式开发；

1. 敏捷方法与传统方法各自的优缺点。

答：优势：敏捷开发的高适应性，以人为本的特性，和轻量型的开发方法即以测试为驱动取代了以文档为驱动，这三个主要的特点，也就是敏捷开发相对与传统开发方式的主要有点。因为他更加的灵活并且更加充分的利用了每个开发者的优势，调动了每个人的工作热情。

劣势：与传统开发方式相比，敏捷开发的最主要的劣势在于敏捷开发欢迎新的需求，而在每次新的需求产生时都可能引起整个系统的大幅度的修改。因为开发者在开发上一个版本的时候，完全没有考虑以后的优化将要如何进行。这样的开发方式实际的软件开发过程中，并不一定总是有效的。

而另一个方面，敏捷开发因为缺乏很多在敏捷开发中被认为“不重要”的文档，这样在一个大型项目比如一个操作系统开发的时候，由于其项目周期很长，所以很难保证开发的人员不更换，而没有文档就会造成在交接的过程中出现很大的困难。

1. 敏捷方法开发中可能出现的问题？如何应对这些问题？

答：敏捷方法明显地降低了文档的数量，甚至声明代码本身就是文档。这就需要开发人员为代码添加更多的注释，但是对于不习惯敏捷开发或团队新成员则很难做到，他们必须不断询问有经验的开发人员，这样会导致延迟迭代交付时间，甚至增加开发费用。而传统开发则强调文档对团队成员的指导作用，开发人员可以在不知道项目细节的情况下完成开发。

　　敏捷开发中强调交互和客户的参与。每次迭代前，团队和客户都将召开一个会议，团队成员将介绍在这次迭代中所作的工作，而客户则根据成员的介绍给出新的功能要求。实际中大部分情况，这种例会是非常乏味和沉闷的，因为团队成员必须重复地向其他

　　成员和客户展示自己负责的模块，接受给出各种对需求的更改，而且每次迭代都是如此，通常为迭代分配的时间都是以周为单位的，开发人员经常感到时间不够用，尤其是自己负责的模块中包含一些复杂的算法时，时间就变得越紧，经常使得迭代延迟。而传统开发中客户不会参与开发过程，实现过程中开发人员只是根据文档编写代码，然后交付最终产品。这样开发人员不必关心那些频繁的迭代会议，而且时间上更加宽泛，有利于开发出更好的产品。

　　敏捷开发对开发人员的个人技能要求更高。敏捷开发强调交互和客户的参与，这就意味着每个团队成员都要具备一定的个人能力和社交技能。每次迭代都必须给客户提供完整的功能模块，并且需要让客户明白当前的开发进程以及开发中遇到的一些问题，开发人员不但需要将自己的工作描述清楚，还得正确理解客户提出的新需求，这需要具备较好的沟通技能。而实际中，不一定每一个开发人员都能具备这样的技能，一旦某一个人不能正确理解一些重要信息，将可能导致下一次迭代不能准确交付，更糟糕的是，如果理解有误，会使得开发的模块中包含多余的功能，结果是对模块进行修改，从而增加开发费用。因此，开发人员社交技能的提升将增加开发过程的稳定性。

　　敏捷开发允许增加需求也导致两个设计中的问题：系统过于僵硬和机动性强。僵硬是指系统中一旦有更改的地方容易引起其他模块的级联改动。机动性是指可能由于需求改变而压缩一些可重复使用的组件，这意味着大量的工作量和对系统整体稳健性带来风险。如果系统中存在这些问题，会违反面向对象设计中的接口隔离原则，从而导致部署过程中的很多问题。

1. 针对敏捷方法的十二条核心实践，结合软件工程问题论述其有效性。

答：1. On-Site Customer（现场客户）：客户是 Team 成员，在开发现场和开发人员一起工作。

　　2. 计划项目（Planning Game）

　　3. 频繁地小规模发布软件（Small Releases）

　　4. 简单设计（Simple Design）

　　5. 测试驱动开发（Test Driven Development）

　　6. 持续集成（Continuous Integration）

　　7. 集体拥有代码（Collective Code Ownership）

　　8. 编程规范（Coding Standards）

　　9. 重构（Refactoring）

　　10. System Metaphor（系统隐喻）

　　11. Pair Programming（结对编程）

　　12. 平稳的工作效率（Sustainable Pace）

“敏捷原则” ，它们是敏捷实践区别于重型过程的特征所在

1. 我们最优先要做的是通过尽早、持续地交付有价值的软件来使客户满意。
2. 在项目的整个开发期间，业务人员和开发人员必须天天在一起工作。
3. 即使到了开发后期，也欢迎需求变化。
4. 经常性地交付可以工作的软件。
5. 可以工作的软件是主要的进度度量标准。
6. 围绕被激励起的个体来构建项目。为他们提供所需的环境和支持，并信任他们能胜任工作。（但对个人的能力要求更高）
7. 最好的架构、需求和设计来自于自组织的团队。
8. 在团队内部，最有效果和最有效率的传递信息的方法是面对面地交流。
9. 敏捷过程提倡可持续的开发速度。
10. 不断地关注最优秀的技术和良好的设计能增强敏捷能力。
11. 简单是根本的。（够用原则Just Enough）
12. 开发团队每隔一定时间，都会对如何能有效地工作进行反省，然后相应地对自己的行为进行调整。
13. 针对软件危机，论述传统软件工程方法和敏捷方法的解决思路。

答：软件危机出现于 20 世纪 60 年代末，软件危机是指在计算机软件的开发和维护过程中遇到的一系列严重问题。

比如：

1）软件开发的成本和进度难以准确估计，延迟交付甚至取消项目的现象屡见不鲜

2）软件存在着错误多、性能低、不可靠、不安全等质量问题

3）软件维护极其困难，而且很难适应不断变化的用户需求

为了解决“软件危机”，许多计算机和软件科学家参照技术过程的一般模式提出了软件 生存期概念（Life cycle）及其瀑布模型（Waterfall Model） 。

计算机软件的生存周期描述了软 件孕育、诞生、成长、成熟、衰亡的生存过程： 制定计划、需求分析、软件设计、程序编制、软件测试、运行维护。 但是， 传统软件工程方法推行多年以来， 人们对于软件危机的 “恐惧” 仍没有丝毫减弱， 相反随着软件系统的急速膨胀而增强，表现为：对软件开发成本和进度的估计常常不准确， 开发成本超出预算，实际进度比预定计划一再拖延的现象并不罕见；用户对“已完成”系统 不满意的现象也经常发生； 软件产品的质量往往靠不住， Bug 一大堆， 补丁一个接一个等等。 于是，无论是产业界还是理论界，都开始对传统软件工程学思想产生怀疑，甚至背叛，于是 以“人”为核心的敏捷开发方法就是在这样的背景下产生的，它背叛了传统软件工程学中 以“过程”为核心，把设计和开发尽可能分开，尽量弱化“人”在整个工程中地位的思想。 敏捷开发实际上包括了许多优秀的软件开发习惯。 首先， 这种方法改变了软件测试的流 程，在编写代码前进行测试，减少了开发风险；此外，通过这种方法，可以对软件进行持续 集成，即每个小时都在集成，任何部件间的冲突都可以随时解决；这种方法的“动态规划” 和“重构”做法，意味着开发者可以对软件架构进行持续改进，可以根据用户的需求随时进 行改进， 而利用传统的软件开发方法则很难对软件的架构进行调整， 同时也会造成成本的大 幅增加。

1. 如何将现有的软件开发向敏捷开发方法转换？期间会遇到哪些困难，如何解决？

答：1、控制开发时间：保持每两周上线一个功能。

这个东西很重要，这是避免产品人员无法克制自己欲望加大量需求的一个好工具。但时间限定，人数限定，天马行空的产品狗只能做最关键的需求。

2、看板

提供看板，是团队沟通透明，减少沟通成本的工具。团队成员能够从看板当中知道项目进度和对接人是谁。

3、每天早上5-10分钟短会

这个短会是每天上午围绕看板沟通的会议，看看每个人的进度如何，遇到什么问题和风险，及时进行沟通协作。

4、工具

工具够用就行，之前用过tower，tower挺好用。不过我团队就5个人，都坐在一起，后面就没用tower。为了项目经理管控进度，就用了公司统一配的JIRA。工具没有强制性必备的，够用合适就行。

5、不开长会，有问题快速沟通。

6、总结、复盘。

最后一点，我们每一期迭代都会总结团队遇到的问题，复盘解决方案。我们自己做过数据分析，我们的开发的效率在不加班的情况下，比以前至少提速了30%。

---------------------------------------------------------------------------------------------------------------------------------

　　敏捷方法明显地降低了文档的数量，甚至声明代码本身就是文档。这就需要开发人员为代码添加更多的注释，但是对于不习惯敏捷开发或团队新成员则很难做到，他们必须不断询问有经验的开发人员，这样会导致延迟迭代交付时间，甚至增加开发费用。而传统开发则强调文档对团队成员的指导作用，开发人员可以在不知道项目细节的情况下完成开发。

　　敏捷开发中强调交互和客户的参与。每次迭代前，团队和客户都将召开一个会议，团队成员将介绍在这次迭代中所作的工作，而客户则根据成员的介绍给出新的功能要求。实际中大部分情况，这种例会是非常乏味和沉闷的，因为团队成员必须重复地向其他成员和客户展示自己负责的模块，接受给出各种对需求的更改，而且每次迭代都是如此，通常为迭代分配的时间都是以周为单位的，开发人员经常感到时间不够用，尤其是自己负责的模块中包含一些复杂的算法时，时间就变得越紧，经常使得迭代延迟。而传统开发中客户不会参与开发过程，实现过程中开发人员只是根据文档编写代码，然后交付最终产品。这样开发人员不必关心那些频繁的迭代会议，而且时间上更加宽泛，有利于开发出更好的产品。

　　敏捷开发对开发人员的个人技能要求更高。敏捷开发强调交互和客户的参与，这就意味着每个团队成员都要具备一定的个人能力和社交技能。每次迭代都必须给客户提供完整的功能模块，并且需要让客户明白当前的开发进程以及开发中遇到的一些问题，开发人员不但需要将自己的工作描述清楚，还得正确理解客户提出的新需求，这需要具备较好的沟通技能。而实际中，不一定每一个开发人员都能具备这样的技能，一旦某一个人不能正确理解一些重要信息，将可能导致下一次迭代不能准确交付，更糟糕的是，如果理解有误，会使得开发的模块中包含多余的功能，结果是对模块进行修改，从而增加开发费用。因此，开发人员社交技能的提升将增加开发过程的稳定性。

　　敏捷开发允许增加需求也导致两个设计中的问题：系统过于僵硬和机动性强。僵硬是指系统中一旦有更改的地方容易引起其他模块的级联改动。机动性是指可能由于需求改变而压缩一些可重复使用的组件，这意味着大量的工作量和对系统整体稳健性带来风险。如果系统中存在这些问题，会违反面向对象设计中的接口隔离原则，从而导致部署过程中的很多问题。

1. 比较不同敏捷过程的特点。

答：

1.　极限编程XP：沟通，简单，反馈，胆识，为四项基本准则，快速反馈，假设简单，递增更改，优质工作，为5条法则，几乎无文档。在所有的敏捷方法中，XP对日期产生的兴趣最多，并且在对良好不定的问题领域的特殊实践方面最为具体。

2.　SCRUM：特别强调开发队伍和管理层的交流协作。每天，开发队伍都会向管理层汇报进度，如果有问题，也会向管理层要求帮助解决。

3.　自适应软件开发ASD：ASD强调开发方法的适应性，这一思想来源于复杂系统的混沌理论。ASD不像其他方法那样有很多具体的实践做法，它更侧重为ASD的重要性提供最根本的基础，并从更高的组织和管理层次来阐述开发方法为什么要具备适应性。

4.　水晶系列方法：相对于其它敏捷方法，水晶系列方法强调软件开发流程的纪律性，所以它比其它敏捷方法易于使用，但它的生产率不如XP等其它敏捷方法。水晶系列与XP一样，都有以人为中心的理念，但在实践上有所不同。人们一般很难严格遵循一个纪律约束很强的过程，因此，与XP的高度纪律性不同，水晶系列方法试图用最少纪律约束而仍能成功的方法，从而在产出效率与易于运作上达到一种平衡。

5.　动态系统开发方法：坚持功能在项目的全过程中都可以改变，当功能被允许改变时，通过使用时间框控制的目的就能达到。重视为项目营造一个正确的文化氛围，如手册中描述了项目有不同的侧重点，并指出对于那些缺陷在传统方法中转变起来是多么的困难。也非常重视协作价值和原理，以及文档。

6.　功能驱动开发方法，短时间的迭代阶段和可见可用的功能，适合于那些不确定或常变更的需求的系统。它抓住了软件开发的核心问题领域，即正确和及时地构造软件。

SE8－习题

1. 什么是软件的逆向工程和再工程？

答：**逆向工程**是复原软件的描述和设计目标的软件分析过程。例如通过反汇编、反编译和动态跟踪等方法，分析出其效果实现过程，推倒出设计并且文档化，帮助工程师在重组系统的结构之前理解程序。软件源代码可以作为逆向工程过程的输入，没有源代码时只能用可执行代码开始。

**再工程**是指对既存软件系统进行调查，将旧系统作为新系统的描述，并重新开发的过程。其目的是重新审视现有的系统，以便进一步利用新技术来改善系统或促进现存系统的再利用。包括：对系统重新建立文档，组织系统改造，用更先进的语言转换系统，修改或更新系统的体系结构，修改或更新系统的数据。

1. 什么是软件配置管理，有什么作用？

答：软件配置管理是一个管理学科，它对配置管理项的开发和支持生存周期给予技术上和管理上的指导。配置管理的应用取决于项目的规模、复杂程序和风险大小。

作用：

　软件配置管理负责软件的标识变更、控制变更和发布变更；

　建立和维护软件产品在整个软件生存周期中的完整性、一致性和可追溯性；

　使各有关人员所见所用的都是有效版本

1. 基线在配置管理中有什么作用？

答：基线（base line）是一个配置项在生存周期内某一特定时间正式标明、固定并经正式批准的一个版本。作用是为了有效地控制变动。

1. 假定你是某个小项目的管理者，你将为项目定义什么基线？以及如何控制它们？

答：作为项目管理者，一般会设立三种基线：分配基线、功能基线和产品基线。

在软件需求分析阶段结束，经过正式评审和批准的软件需求规格说明后会设立分配基线用于最初批准的分配配置标识。然后在系统分析和软件定义结束，获得经过正式评审和批准系统设计规格说明书后会设立功能基线，用于最初批准的功能配置标识。最后在软件组装与系统刚测试阶段结束时，经过正式评审和批准的有关软件产品的配置项的规格说明书后会设立产品基线标识最初批准的产品配置。

1. 软件维护的基本内容是什么？

答：在软件运行/维护阶段对软件产品所进行的修改就是维护（maintain）。

改正性维护

　　是指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。这方面的维护工作量要占整个维护工作量的17%～21%。所发现的错误有的不太重要，不影响系统的正常运行，其维护工作可随时进行：而有的错误非常重要，甚至影响整个系统的正常运行，其维护工作必须制定计划，进行修改，并且要进行复查和控制。

适应性维护

　　是指使用软件适应信息技术变化和管理需求变化而进行的修改。这方面的维护工作量占整个维护工作量的18%～25%。由于目前计算机硬件价格的不断下降．各类系统软件屡出不穷，人们常常为改善系统硬件环境和运行环境而产生系统更新换代的需求；企业的外部市场环境和管理需求的不断变化也使得各级管理人员不断提出新的信息需求。这些因素都将导致适应性维护工作的产生。进行这方面的维护工作也要像系统开发一样，有计划、有步骤地进行。

完善性维护

　　这是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。这些功能对完善系统功能是非常必要的。另外，还包括对处理效率和编写程序的改进，这方面的维护占整个维护工作的50%～60%，比重较大．也是关系到系统开发质量的重要方面。这方面的维护除了要有计划、有步骤地完成外．还要注意将相关的文档资料加入到前面相应的文档中去。

预防性维护

　　为了改进应用软件的可靠性和可维护性，为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使应用系统适应各类变化而不被淘汰。例如将专用报表功能改成通用报表生成功能，以适应将来报表格式的变化。这方面的维护工作量占整个维护工作量的4%左右。

1. 软件维护有哪些特点，分几类，每类的内容是什么？

答：软件维护主要有以下三大特点。

　　1）结构化维护与非结构化维护差别巨大。

　　2）维护的代价高昂。维护费用是最明显的代价，其他一些代价如可用的资源供维护任务使用，以致耽误开发良机，看似合理的有关改错的要求不能及时满足引起用户不满，引入了新的潜伏错误等。

　　3）维护的问题很多。如难以很好的理解别人写的程序，需要维护的软件没有说明文档，软件设计时没有考虑将来的修改等。

软件维护活动类型大概有四类：纠错性维护；适应性维护；完善性维护或增强；预防性维护或再工程。

纠错性维护：识别和纠正软件错误，改正性能上的缺陷，排除实施中的误使用而进行的诊断和改正错误的活动。

适应性维护：使软件适应处理环境或数据要求的变化而修改软件的活动。

完善性维护或增强：修改或在开发软件，以扩充软件功能，增强软件性能等。

预防性维护或再工程：采用先进的软件工程方法对需要维护的软件或软件的某一部分进行设计、编码和测试。

1. 软件的可维护性与哪些因素有关，如何提高软件的可维护性？

答：软件维护就是在软件交付使用后进行的修改，修改之前必须理解待修改的对象，修改之后应该进行必要的测试，以保证所做的修改是正确的。如果是改正性维护，还必须预先进行调试以确定错误的具体位臵。因此，决定软件可维护性的因素主要有下述5个：

1）可理解性

软件可理解性表现为外来读者理解软件的结构、功能、接口和内部处理过程的难易程度。模块化（模块结构良好，高内聚，松耦合）、详细的设计文档、结构化设计、程序内部的文档和良好的高级程序设计语言等等，都对提高软件的可理解性有重要贡献。

2）可测试性

诊断和测试的容易程度取决于软件容易理解的程度。良好的文档对诊断和测试是至关重要的，此外，软件结构、可用的测试工具和调试工具，以及以前设计的测试过程也都是非常重要的

3）可修改性

软件容易修改的程度和之前讲过的设计原理和启发规则直接有关（回顾）。耦合、内聚、信息隐藏、局部化、控制域与作用域的关系等等，都影响软件的可修改性。

4）可移植性

软件可移植性指的是，把程序从一种计算环境（硬件配臵和操作系统）转移到另一种计算环境的难易程度。把与硬件、操作系统以及其他外部设备有关的程序代码集中放到特定的程序模块中，可以把因环境变化而必须修改的程序局限在少数程序模块中，从而降低修改的难度。

5）可重用性

重用是指同一事物不做修改或稍加改动就在不同环境中多次重复使用。大量使用可重用的软件构件来开发软件。

可通过如下几点来提高软件的可维护性：

1）建立明确的软件质量目标。应该在一定程度上满足可维护的各个特性，但各个特性的重要性又是随着程序的用途或计算机环境的不同而改变的。对编译程序来说，效率和可移植性是主要的；对信息管理系统来说，可使用性和可修改性可能是主要的。

2）使用先进的软件开发技术和工具。利用先进的软件开发技术和工具是软件开发过程中提高软件质量、降低成本的有效方法之一，也是提高可维护性的有效技术。

3）进行明确的质量保证审查。在软件开发每个阶段结束前的技术审查和管理复审中，应该着重对可维护性进行审查。如在需求分析阶段的审查中，应该对将来要改进的部分和可能会修改的部分加以注意并指明；如在设计阶段的审查中，应从容易修改、模块化和功能独立的目标出发来评价软件的结构和过程；在代码审查中应强调编码风格和内部说明文档对软件可维护性的影响；在测试阶段的审查中，应对软件的配置复审进行重视；在维护阶段完成每项维护工作后，都应对维护工作本身进行仔细认真的复审，如源代码的修改与文档的修改是否一致。

4）选择可维护的程序设计语言。程序设计语言的选择对维护的影响比较大。低级语言一般很难理解，很难掌握，因而很难维护。

5）改进程序文档。文档是影响软件可维护性的重要因素，文档有时比程序代码更重要。程序文档是对程序的总目标、程序各组成部分之间的关系、程序设计策略、程序实现过程的历史数据等的说明和补充。程序文档对提高程序的可理解性有着重要作用。好的文档是建立可维护性的基本条件。另外，在软件维护阶段，利用历史文档可以大大简化维护工作。

1. 什么是软件维护的副作用，如何防止软件维护的副作用？

答：软件维护的副作用，就是指由于修改程序而导致的错误或其它不需要的活动。三类主要副作用，即：修改代码的副作用、修改数据的副作用和修改文档资料的副作用。

1）修改代码的副作用

例如，删除或修改一个子程序、改变逻辑运算符、改进程序的执行效率以及把设计上的改变翻译成代码的改变等等，都容易引入错误。修改代码所带来的副作用，一般可以在回归测试过程中发现。

2）修改数据的副作用

例如，重新定义局部或全局的常量、重新定义记录或文件的格式、重新排列输入/输出或子程序的参数时，等等。在修改数据时，有可能造成软件设计与数据结构不匹配，因而导致错误。修改数据的副作用可以通过详细的设计文档来加以限制。这种文档描述了交叉引用，把数据元素、记录、文件和其他结构联系起来。

3）修改文档资料的副作用

例如，对交互输入的程序或格式所进行的修改，如果没有正确地记入文档中，就会引起重大问题。维护应该着眼于整个软件配置。每当对数据流、软件结构、模块过程或任何其它有关特征进行修改时，都必须对相应的文档资料进行更新。不能正确反映软件当前状态的文档资料可能比完全没有文档资料更坏。因此，必须在软件再次交付之前对整个软件配置进行评审，以减少文档的副作用。

为了控制因修改而引起的副作用，在修改时应做到：

1.按模块把修改分组；

2.自顶向下地安排所修改模块的顺序；

3.每次修改一个模块；

4.对于每个修改了的模块，在安排修改下一个模块之前，要确定这个修改的副作用。可以使用交叉引用表、存储映象表、执行流程跟踪等。

1. 在现实环境中使用的软件系统必须进行变更，否则就会逐渐失去其效用。这是为什么？试解释之。

答：（主观题。参考了需求变更原因，供参考。一方面是用户：他们是项目需求的提出者。一个十分常见的现象是用户提出需求以后，在软件开发过程中用户改变了需求，这只能迫使开发工作返工，丢弃一些无法修正的部分。无疑这会造成一定的损失，但又无法完全避免。要求用户一次性把需求讲清楚，并且不允许此后需求有任何变更，这是不现实的。只能尽量减少需求变更，降低它所造成的影响。二是系统因素：在系统内部，如计算机硬件、系统软件或数据的变更要求与其相适应。三是外部环境因素：与软件运行相关的工作制度或法规、政策的变更，或是业务要求变更导致的需求变更。四是需求分析阶段工作缺陷：需求调研、分析、定义和评审工作不够充分，致使需求规格说明中隐含着问题，在开发过程中才有所发现。或者需求开发中开发人员与用户沟通不够充分，如未能如实获得用户的潜在需求等。软件需求一旦出现变更，它可能要涉及到一些相关的代码和文档的修改，为此要把这一变更通知到所有相关人员。提出需求变更有可能在开发的任何阶段，并且随着项目的进展，越晚的需求变更引起的损失越大。）