第一章

1、软件产品的特性是什么？
2、软件有那些分类标准？
3、软件危机有什么表现？
4、软件危机产生的原因是什么？
5、如何解决软件危机？
6、软件工程的基本原理是什么？
7、简述软件工程目标之间的关系。
8、你认为现在已经摆脱软件危机了吗？请说明理由。
9、对软件安全产生威胁的因素有哪些？
10、软件安全威胁是否可以完全消除？为什么？

第二章

1、在软件生命周期中软件开发过程由哪几个阶段组成？各阶段的任务是什么？
2、瀑布模型软件开发有哪些特点？又有哪些不足？
3、螺旋模型有哪些优缺点？
4、快速原型技术的基本思想是什么？
5、比较增量模型和螺旋模型的特点，有什么不同和相似的地方？
6、比较螺旋模型和组件集成模型的异同。
7、比较喷泉模型和敏捷开发模型的异同。
8、比较迭代模型和增量模型的异同。
9、敏捷开发的核心思想是什么？
10、安全开发生命周期的核心思想是什么？
11、公司计划开发一款新的手机应用软件，并希望尽快发布上市，该选择哪种软件开发过程模型，如何进行？
12、公司计划为某外贸公司开发一款采购和销售管理软件，目前对采购部完成了需求调研，销售部因为正在进行内部调整而调研进度缓慢，为了缩短软件开发周期，现在是否可以开始软件开发，如何进行？

第三章

1、可行性研究的任务是什么？
2、简述可行性研究的步骤。
3、说明系统流程图的作用。
4、数据流图中父图与子图的平衡指什么？
5、检查数据流图的原则是什么？
6、工作分解的作用是什么？
7、WBS 的分解方式有哪些？
8、对两个活动进行排序时，活动之间的依赖关系有几种？
9、软件项目进度图有几种形式，各有什么优缺点？
10、常见的对软件规模进行估算的方法有哪些？
11、软件项目成本估计技术有哪些？
12、软件项目效益分析的度量有哪些？
13、画出考试教务系统的系统流程图和数据流图。

14、画出公交车一卡通下车刷卡的分层数据流图。
15、画出公交车一卡通刷卡软件的工作分解结构。

第四章

1、常见的需求有哪些错误？
2、需求工程包括哪些活动？简要说明其内容。
3、什么是功能性需求？什么是非功能性需求？
4、对功能需求的要求是什么？
5、非功能需求存在什么问题？
6、需求获取的常用方法有哪些？
7、用户需求规格和软件需求规格的区别是什么？
8、需求分析的流程是什么？
9、如何给不同的功能确定优先级？
10、需求验证从哪几个方面进行？
11、需求变更的原因是什么？
12、需求变更管理的流程是什么？
13、需求跟踪的实现方法有哪些？
14、画出图书馆借书系统的实体-关系图。
15、画出网上购物网站中订单的状态转换图。

第五章

1、面向对象分析和设计解决了哪两个经典问题？
2、如何确定用例？
3、如何确定执行者？
4、用例之间的关系有哪些？
5、活动图的特点和作用是什么？
6、活动图包括哪些元素？
7、画出网上订票系统的用例模型。
8、画出网上订票的活动图。
9、分析打印机的状态并画出打印机的状态图。
10、完善手机打车软件用例图，补充到达目的地后相关用例。

第六章

1、什么是软件结构？有哪几种？
2、简述总体设计的步骤。
3、什么是模块的影响范围？什么是模块的控制范围？它们之间应该建立什么关系？
4、模块化要遵循哪些原则？
5、模块分解时，是否将系统分解得非常细，得到的功能模块越多越好呢？为什么？
6、模块分解时，是否将系统分解得非常细，得到的功能模块越多越好呢？为什么？
7、衡量模块独立性的两个标准是什么？请分别说明。
8、内聚包括哪几种？分别指什么？

9、模块的耦合性由低到高分为哪些？
10、简述结构化程序设计方法的基本要点。
11、简述软件结构设计优化准则。
12、变换分析设计与事务分析设计有什么区别？
13、画出手机打车软件的系统结构图。
14、画出公交一卡通刷卡软件的系统结构图。

第七章

1、简述详细设计的过程。
2、结构化程序设计应遵循的基本原则是什么？
3、详细设计有哪几种描述方法？
4、简述详细设计的图形描述工具，以及各自的概念和优缺点。
5、求数组 a[0]-a[10]中的次大值，请画出程序流程图。
6、请画出判断某一年是不是闰年的程序 N-S 图。
7、请画出公交一卡通刷卡扣款软件的 PAD。
8、请画出电梯根据某一层按键做出响应的 PDL 表示。
9、请以是否为头等舱、国内乘客、残疾乘客的顺序画出计算 7.3.4 节中超重行李费用的判定树。

第八章

1、类之间的关系有哪几种？
2、使用顺序图对系统进行建模时，要遵循什么策略？
3、画顺序图的一般步骤是什么？
4、在顺序图中消息有几种类型？返回消息是必须的吗？
5、顺序图和协作图的区别是什么？
6、画出客户在 ATM 上取款用例的类图。
7、画出在网上订火车票的对象图。
8、画出在网上购物的顺序图。
9、画出手机打车软件的构件图。

第九章

1、简述软件安全的原则。
2、简述威胁建模的步骤。
3、什么是 STRIDE 威胁模型？
4、什么是 DREAD 模型？
5、建立手机打车软件的 STRIDE 威胁模型。
6、在网上购物网站的登录系统中，存在恶意人员通过程序进行登录口令暴力破解的威胁，请分析该威胁的等级，并写出原因。

第十章

1、为什么要研究人机交互设计？
2、人机交互界面有哪些类型？
3、好的软件界面的特征有什么？
4、在软件交互设计中人的因素有哪些？

5、为什么要以用户为中心设计软件界面？
6、简述软件交互设计的要求。
7、软件界面设计最重要的是美观，这种观点对吗？为什么？
8、设计软件界面时需考虑的防错处理方式有哪些？
9、简述图形界面设计的一般原则。
10、填表输入界面设计的注意事项是什么？
11、任选 4 种手机应用导航方式，举出具体例子。

第十一章

1、在软件开发阶段的工作流程中，测试人员测试和代码审查的顺序是否可以互换？
2、软件编程语言经过了几个发展阶段？
3、开发软件时选择编程语言要考虑哪些因素？
4、为了提高软件的可维护性，在编码阶段应注意哪些问题？
5、什么是程序设计风格？
6、标识符命名的通用规则是什么？
7、为何要进行程序的注释？应该怎样进行程序的注释？
8、数据说明时的通用规则是什么？
9、有一种循环结构，其流程图如图 11-4 所示，这种控制结构不属于基本控制结构：它既不是先判断型循环，又不是后判断型循环。请修改此流程图，将它改为用基本控制结构表示的等效流程图。

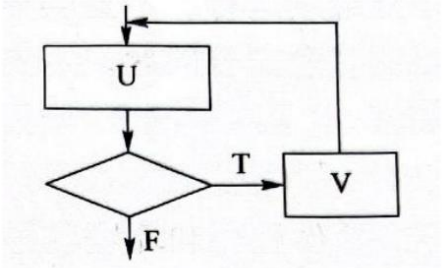


图 11-4　循环结构的流程图

第十二章

1、软件测试的概念和目的是什么？
2、测试文档有哪些？
3、软件测试的原则是什么？
4、非渐增式测试与渐增式测试有什么区别？渐增式测试如何组装模块？
5、一般驱动模块比桩模块容易设计，为什么？
6、简述白盒测试的概念及相关技术。
7、一个程序流程图如图 12-9 所示，请分别根据语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖设计测试用例。

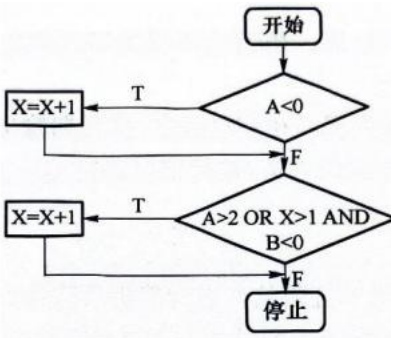


图 12-9　一个程序流程图

8、简述黑盒测试的概念及相关技术。

9、在某一 PASCAL 语言版本中规定“标识符是由字母开头的，后跟字母或数字的任意组合；有效字符数为 8 个，最大字符数为 80 个”，请写出测试数据等价类。

输入条件	有效等价类	无效等价类
标识符个数	1个（1），多个（2）	0个（3）
标识符字符数	1～8个（4）	0个（5），>8个（6），>80个（7）
标识符组成	字母（8），数字（9）	非字母数字字符（10），保留字（11）
第一个字符	字母（12）	非字母（13）
标识符使用	先说明后使用（14）	未说明已使用（15）

10、简述灰盒测试的概念及优缺点。

11、软件测试阶段如何划分？

12、测试停止的依据是什么？

13、模糊测试和渗透测试有什么区别？

第一章

1、软件产品的特性是什么?

①软件是一种逻辑实体，而不是具体的物理实体。具有抽象性，不可描述性

②在软件的运行和使用期间，没有硬件那样的机械磨损，老化问题。但存在退化问题。(用户体验)

③软件的生产与硬件不同，在它的开发过程中没有明显的制造过程，而是研制。是程序员脑力劳动的结果

④可变性，软件在生产过程中甚至在投入运行之后也还可以再改变。

⑤软件必须和运行软件的硬件保持一致

⑥软件的运行受计算机系统的影响

⑦软件一旦研制开发成功，其生产过程就变成复制过程

2、软件有那些分类标准?

按软件的功能进行划分：系统软件、支持软件、应用软件

基于软件规模进行划分：微型、小型、中型、大型、超大型、巨型

基于软件工作方式划分：实时处理软件、分时软件、交互式软件、批处理软件

按软件服务对象的范围划分：项目软件、产品软件

按使用的频度进行划分：一次使用、频繁使用

按软件失效的影响进行划分：高可靠性软件、一般可靠性软件

3、软件危机有什么表现?

①软件开发费用和进度失控。费用超支、进度拖延的情况屡屡发生。

②软件的可靠性差。尽管耗费了大量的人力物力，而系统的正确性却越来越难以保证，出错率大大增加，造成的损失惊人。

③生产出来的软件难以维护。很多程序缺乏相应的文档资料，程序中的错误难以定位，难以改正，有时改正了已有的错误又引入新的错误

④用户对“已完成”的系统不满意现象经常发生。

4、软件危机产生的原因是什么?

(1) 随着软件应用范围的增广，软件的规模愈来愈大，复杂性也急剧地增加。缺乏统一的、规范的方法论的指导，依靠个人经验。

(2) 忽视软件开发前期的需求分析。

编程越早，需求分析不充分，完成时间越长

（3） 缺乏软件开发的经验和有关软件开发数据的积累，使得开发工作的计划很难制定。

（4） 忽视软件文档，文档常常内容不一致，有时甚至没有文档，也是造成开发效率低下的原因。

（5） 忽视测试阶段的工作，或不负责任的测试员，提交用户的软件质量差。

（6） 轻视软件的维护，导致软件在交付后难以维护，程序错误难以定位和排除。

5、如何解决软件危机?

（1） 软件开发应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。

（2） 必须充分吸收和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法，特别是要吸取几十年来从事计算机硬件研究和开发的经验教训。

（3） 应该推广使用在实践中总结出来的开发软件的成功的技术和方法。

（4） 应该开发和使用更好的软件工具。

总之，为了解决软件危机，既要有技术措施（方法和工具），又要有必要的组织管理措施。

6、软件工程的基本原理是什么?

①用分阶段的生存周期计划严格管理开发过程。

②坚持进行阶段评审。

③实行严格的产品控制。

④采用现代程序设计技术。

⑤结果应能清楚地审查。

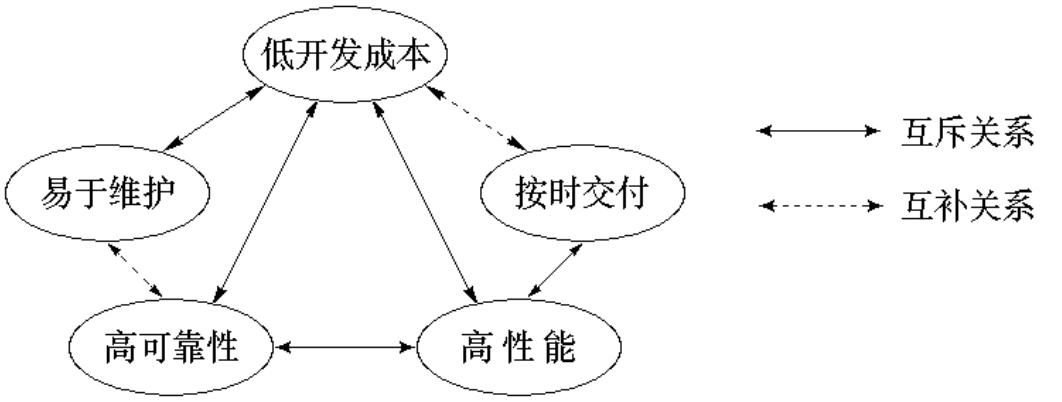
⑥开发小组的人员应少而精。

⑦承认不断改进软件工程实践的必要性。

7、简述软件工程目标之间的关系。

互斥关系：低开发成本与易维护之间、低开发成本与高可靠性之间、低开发成本与高性能之间、高可靠性与高性能之间、高性能与按时交付之间就存在冲突。

互补关系：易维护与高可靠性之间、低开发成本与按时交付之间。



8、你认为现在已经摆脱软件危机了吗？请说明理由。

我认为现在还未完全摆脱软件危机，因为随着经济发展，各行各业对软件人才的需求不断扩大，由于软件人才缺口依旧很大，造成了某些行业内软件开发进度难以预测，软件开发成本难以控制，用户对产品功能难以满足，软件产品质量无法保证，软件产品难以维护，软件缺少适当的文档资料等问题。

9、对软件安全产生威胁的因素有哪些?

（1）软件缺陷

从产品内部看，缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题；从产品外部看，缺陷是系统所需要实现的某种功能的失效或违背。

①软件开发安全意识淡薄，很少从“攻击者”的角度来思考软件安全问题。

②软件开发者缺乏安全知识，软件开发人员很少进行安全能力与意识的培训。

③软件趋向大型化，第三方扩展增多

（2）软件面临的外部威胁

恶意软件增多。计算机病毒、蠕虫、特洛伊木马、后门、僵尸、间谍软件等。

①破坏正常软件运行

②窃取重要数据

③监视用户行为

④控制目标系统

⑤加密文件索要赎金

10、软件安全威胁是否可以完全消除？为什么？

软件安全威胁不可以完全消除，原因如下:

①软件复杂性增加：随着软件系统的复杂性不断增加，新的安全漏洞和威胁也在不断出现。软件产品无完美者，这是开发过程中不可避免的现实。

②持续的新威胁：网络安全状况不断变化，新型威胁层出不穷，所有企业都应将安全视为头等大事。

③开源软件漏洞：国内企业软件项目开源软件使用率很高，开源软件漏洞指标仍处于高位，软件供应链的安全问题并没有得到根本性的改善。

④攻击手段不断进化：攻击者会寻找并追击公司供应链中的薄弱环节，保障基本的安全卫生仍是第一要务。

⑤隐晦式安全的风险：依赖“隐晦式安全”（security through obscurity）的系统永远不可能真正安全，面对复杂攻击者，任何保护都可能被突破。

⑥人为因素：人为错误和疏忽是导致安全威胁的一个重要因素，而完全消除人为错误是不现实的。

⑦技术限制：技术限制意味着我们无法总是预见和防御所有可能的安全威胁，尤其是那些未知的或尚未被发现的威胁。

⑧资源限制：在实际操作中，资源（如时间、金钱和专业知识）的限制意味着不可能完全消除所有的安全威胁。

因此，尽管我们可以采取多种措施来预防和减轻软件安全威胁，但完全消除它们是不现实的。我们需要持续地提高安全意识，加强安全措施，并准备应对新出现的威胁。

第二章

1、在软件生命周期中软件开发过程由哪几个阶段组成？各阶段的任务是什么？

软件生存期的六个活动，即问题定义和可行性研究、需求分析、设计、程序编码、测试及运行维护。

（1）问题定义和可行性研究

回答：系统要解决什么问题，是否可行？确定要开发软件系统的总目标及可行性。给出功能、性能、可靠性以及接口等方面的要求。估计可利用的资源 (硬件，软件，人力等)、成本、效益、开发进度。制定出完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查。

（2）需求分析和定义

对用户提出的要求进行分析并给出详细的定义。不是解决问题，而是确定软件系统必须做什么，及详细定义。编写软件需求说明书或系统功能说明书及初步的系统用户手册。提交管理机构评审。

（3）软件设计

把需求转变为软件结构，及模块的功能描述。概要设计 — 把各项需求转换成软件的体系结构。结构中每一组成部分都是意义明确的模块，每个模块都和某些需求相对应。详细设计 — 对每个模块要完成的工作进行具体的描述，为源程序编写打下基础。编写设计说明书，提交评审。

（4）程序编写

把软件设计转换成计算机可以接受的程序代码，即写成以某一种特定程序设计语言表示的“源程序清单”。要求：写出的程序应当是结构良好、清晰易读的，且与设计相一致的，并且是安全的。大部分计算机课程在这一层。

（5）软件测试

发现存在的问题。单元测试、集成测试、确认测试、系统测试几个阶段。

单元测试，查找各模块内部在功能和结构上存在的问题并加以纠正

集成测试，将已测试过的模块按一定顺序组装起来

确认测试，按规定的各项需求，逐项进行有效性测试，决定已开发的软件是否合格，能否交付用户使用。

（6）运行/维护

修改在运行中发现的软件错误，完善功能，或为了适应变化了的软件工作环境，需做适当变更。

2、瀑布模型软件开发有哪些特点？又有哪些不足？

（1）按照传统的瀑布模型来开发软件，有以下特点:

①阶段间有顺序性和依赖性;

②推迟实现的观点;

③质量保证的观点。

（2）瀑布模型缺点:

①项目开始阶段，开发人员和用户对需求的描述常常不全面。如果需求阶段没发现问题，则影响后面各阶段的工作

②各阶段所做的工作都是文档说明，一般用户不易理解文字所叙述的软件。用户的修改意见加大了修改难度

③改变会影响整个软件开发。事先选择的技术或需求迅速发生变化，需要返回到前面，对前面一系列内容进行修改。

3、螺旋模型有哪些优缺点？

优点:

风险驱动的，每个方案在实施前都要经过风险分析。如果风险过大，则项目应该停止，或改变方案。减少了过多的测试或测试不足所带来的风险。

缺陷:

1） 采用螺旋模型需要具有相当丰富的风险评估经验和专门知识，在风险较大的项目开发中，如果未能够及时标识风险，势必造成重大损失以及客户认可。

2） 过多的迭代次数会增加开发成本，延迟提交时间。

3） 如果执行风险分析会大大影响项目的利润，那么风险分析就没有意义了。只适合大规模软件开发。

4、快速原型技术的基本思想是什么？

原型模型的基本思想是：在与用户进行需求分析的同时，软件开发人员根据用户提出的软件基本需求，以比较小的代价快速开发一个原型，以便向用户展示软件系统应有的部分或全部功能；用户对原型提出改进意见，分析人员根据用户的意见，补充完善原型，然后再由用户评价，提出建议，如此往复，直到开发的原型系统满足了用户的需求为止。通

常原型系统是用户可以操作的系统，系统中已经包括了用户的需求，用户通过实际操作，比较容易发现漏掉的需求。

5、比较增量模型和螺旋模型的特点，有什么不同和相似的地方？

（1）相同点

①都是迭代式模型：增量模型和螺旋模型都是迭代式的开发模型，都可以允许系统在开发过程中不断完善和调整。

②都注重风险管理：两种模型都注重对项目风险的分析和管理，都可以在项目的开发过程中根据风险情况进行调整。

（2）不同点

①风险驱动程度不同：螺旋模型更加注重对项目风险的分析和管理，风险驱动程度更高；而增量模型更注重对需求变化和客户参与的灵活性。

②适用场景不同：增量模型更适合于那些需求变化较为频繁的项目，适用范围更广；而螺旋模型更适合于对风险管理要求更高的项目，尤其是复杂的大型系统开发项目。

③两者迭代层级不同：增量模型在活动级迭代；而螺旋模型在过程级迭代。

④两者需求分析的时间不同：增量模型常常是先做总体需求分析和设计，然后再编码和测试中逐个增量开发；而螺旋模型在开发周期内采用简化瀑布模型或快速模型。

⑤两者提交软件的方式不同：增量开发在上次增量的基础上提交新的一部分软件；而螺旋模型每次迭代都提交一个新的完整的软件版本。

⑥两者减少风险的方式不同：增量开发通过避免使用未成熟技术和经常的客户反馈等方法减少风险；而螺旋模型中直接增加了风险识别、风险分析、风险控制，计划性较强。

6、比较螺旋模型和组件集成模型的异同。

组件集成模型利用预先对封装好的软件构件来构造应用软件系统，它融合了螺旋模型的很多特征，支持软件开发的迭代方法。

7、比较喷泉模型和敏捷开发模型的异同。

（1）相同点：

①迭代开发：喷泉模型和敏捷开发模型都强调迭代开发，即通过多个小的迭代周期逐步构建和完善软件产品

②需求变化适应性：两者都能够适应需求的变化，允许在开发过程中根据用户反馈进行调整

③用户参与：喷泉模型和敏捷开发都鼓励用户参与到开发过程中，以确保最终产品能够满足用户的实际需求

（2）不同点：

①开发流程：

喷泉模型：是一种面向对象的软件开发模型，强调软件开发过程的迭代性和无间隙性。它允许各个阶段之间的反复和交叉，如分析、设计、编码和测试等活动不再是线性的、顺序的，而是相互重叠和迭代的

敏捷开发模型：更强调团队合作和灵活性，通过短暂的迭代周期快速交付部分功能，并根据实际反馈进行迭代和调整。敏捷开发模型适用于需求变化频繁、项目较灵活的情况

②项目管理：

喷泉模型：由于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，这可能导致项目管理的难度增加

敏捷开发模型：强调小团队的紧密合作，通常团队规模较小，更易于管理。敏捷方法更适用于较小的队伍，如 20、40 人或者更少

③文档和计划：

喷泉模型：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

敏捷开发模型：相对而言，敏捷开发过程中对文档的依赖较少，更侧重于面对面的沟通和快速响应

④适用场景：

喷泉模型：适用于面向对象的软件开发过程，适合需求变化频繁、项目较复杂的情况

敏捷开发模型：适用于需求变化频繁、团队协作紧密的项目，特别适合于那些需求不明确或者需要快速响应市场变化的场景

8、比较迭代模型和增量模型的异同。

他们的共同点是，通过若干个阶段的开发，完成整个软件，每阶段完成之后，都有一个新版本发布。这就有一个好处，相对于整个漫长的开发周期来说，每阶段都会见到亮光，有利于鼓舞团队的士气，降低项目的风险。

不同点，主要是阶段的划分上不太一样。增量模型是从功能量上来划分的，每阶段完成一定的功能。迭代模型是从深度或细化的程度来划分的，每阶段功能得到完善、增强。

9、敏捷开发的核心思想是什么?

敏捷就是“快。要快，就要更多地发挥个人的个性思维。敏捷开发是一种以人为核心、迭代、循序渐进的开发方法。敏捷方法的两大主要特征是对“适应性”的强调与对“人”的关注。强调对变化的快速响应能力，通过引入迭代式的开发手段。主要驱动核心是人，与流水线开发的思维相左。

10、安全开发生命周期的核心思想是什么?

安全开发生命周期的核心是将安全考虑集成在软件开发的每一个阶段，SDL 致力于减少软件中漏洞的数量和严重性。SDL 在开发过程的所有阶段中均引入了安全和隐私。

11、公司计划开发一款新的手机应用软件，并希望尽快发布上市，该选择哪种软件开发过程模型，如何进行?

选择敏捷开发模型。敏捷开发模型以其快速响应变化、迭代开发和以人为核心等特点，非常适合需要快速上市的软件开发项目。以下是如何进行敏捷开发的大致步骤:

第一步：找出完成产品需要做的事情——Product Backlog。

第二步：决定当前的冲刺（Sprint）需要解决的事情——Sprint Backlog。

第三步：冲刺。

第四步：得到软件的一个增量版本。发布给用户。然后在此基础上进一步计划增量的新功能和如何改进。

敏捷开发模型允许开发团队快速适应变化，及时响应市场需求，这对于希望尽快发布上市的手机应用软件来说是非常重要的。通过短周期的迭代，可以逐步完善产品功能，同时保持与市场的紧密联系，确保产品能够满足用户的实际需求。

12、公司计划为某外贸公司开发一款采购和销售管理软件，目前对采购部完成了需求调研，销售部因为正在进行内部调整而调研进度缓慢，为了缩短软件开发周期，现在是否可以开始软件开发，如何进行?

在这种情况下，为了缩短软件开发周期，可以考虑分阶段进行开发，即基于已经完成的采购部需求进行开发，暂时跳过销售部的功能开发，待销售部的需求调研完成后再进行后续开发。

具体步骤如下:

①确定最小可行产品（MVP）：基于采购部已完成的需求，确定最小可行产品，即采购管理模块可以独立使用，满足基本的业务需求。这样可以先着手开发，并在开发过程中灵活调整。

②分阶段开发：

第一阶段：完成采购部的管理功能，如采购订单管理、供应商管理、采购审批流程等。

第二阶段：在销售部调研完成后，开始销售相关模块的开发，例如销售订单管理、客户管理、销售预测等功能。

③并行调研与开发：在开发采购模块的同时，可以继续与销售部沟通，尽可能加快销售部的需求调研进度。这样一旦调研完成，您可以迅速进行销售模块的开发，减少等待时间。

④灵活调整开发计划：在开发过程中，根据销售部需求的变化及时调整开发内容。确保系统设计具有一定的灵活性，以便后期扩展和修改。

这种做法可以确保在调研不完全时仍能启动部分开发工作，同时避免因等待销售部调研完毕而造成的开发延误。

第三章

1、可行性研究的任务是什么?

可行性研究的目的是用最小的代价在尽可能短的时间内确定问题是否能够解决。也就是说可行性研究的目的不是解决问题，而是确定问题是否值得去解，研究在当前的具体条件下，开发新系统是否具备必要的资源和其他条件。可行性研究是压缩简化了的系统分析和设计过程，也就是说在较高层次上以较抽象的方式进行设计的过程。

在明确了问题定义之后，分析员应该给出系统的逻辑模型，然后从系统的逻辑模型出发，寻找可供选择的解法，研究每一种解法的可行性。一般来说，应从经济可行性、技术可行性、资源可行性、法律可行性和开发可行性等方面研究可行性。

2、简述可行性研究的步骤。

（1）复查系统规模和目标

访问关键人员、仔细阅读和分析有关材料，复查问题定义中的规模和目标，明确目标系统的一切限制和约束。

（2）研究目前正在使用的系统

现有系统是信息的重要来源。新系统必须能完成旧系统的基本功能，也必须解决旧系统的缺陷。了解相似的系统或软件。

（3）导出新系统的高层逻辑模型

从现有物理系统出发，导出现有系统的逻辑模型，设计新系统的逻辑模型，再导出物理模型。

（4）重新定义问题

复查问题定义，重新定义和修正问题，前四步骤是一个循环。

（5）导出和评价供选择的解法

从逻辑模型出发，导出几种解决方案 从技术可行性考虑排除不可行的方案，其次考虑实施可行性，估计开发成本和费用。最后，为每个技术、操作和经济都可行的系统制定实现进度表。

（6）推荐一个方案

根据可行性研究结果做出决定，是否可行，及给出可行的方案。

3、说明系统流程图的作用。

（1）制作系统流程图的过程是系统分析员全面了解系统业务处理情况的过程， 它是系统分析员作进一步分析的依据。

（2）系统流程图是系统分析员、管理人员、业务操作人员相互交流的工具。

（3）系统分析员可直接在系统流程图上拟出可以实现计算机处理的部分。

（4）可利用系统流程图来分析业务流程的合理性。

4、数据流图中父图与子图的平衡指什么?

平衡指的是子图的输入、输出数据流必须与父图中对应加工的输入、输出数据流相同。

5、检查数据流图的原则是什么?

① 数据流图上所有图形符号只限于前述四种基本图形元素，并且必须包括前述四种基本元素，缺一不可。

② 数据流图的主图上的数据流必须封闭在外部实体之间。

③ 每个加工至少有一个输入数据流和一个输出数据流。

④ 在数据流图中，需按层给加工框编号。编号表明该加工所处层次及上下层的亲子关系。

⑤ 规定任何一个数据流子图必须与它上一层的一个加工对应，两者的输入数据流和输出数据流必须一致。此即父图与子图的平衡。

⑥ 可以在数据流图中加入物质流，帮助用户理解数据流图。

⑦ 图上每个元素都必须有名字。

⑧ 数据流图中不可夹带控制流。

6、工作分解的作用是什么?

将一个项目分解为更多的工作细目，使项目变得更易管理、更易操作。

7、WBS 的分解方式有哪些?

按照产品的功能组成分解、按照项目各阶段活动分解。

8、对两个活动进行排序时，活动之间的依赖关系有几种?

强制性依赖关系：工作任务中固有的依赖关系，它是因为客观规律和物质条件的限制造成的，又称硬逻辑关系。

软逻辑关系：由项目管理人员确定的项目活动之间的关系，它是一种根据主观判断去调整和确定的关系，也称指定性相关、偏好相关或软相关。

外部依赖关系：项目活动对一些非项目活动和事件的依赖。

9、软件项目进度图有几种形式，各有什么优缺点?

三种。

（1）甘特图

优点：可以查看活动的工期、开始时间和结束时间以及资源的信息，可用于详细的时间管理。简单、直观、易于编制。

缺点：活动之间的依赖关系没有表示出来，难以进行定量的计算分析和计划的优化。

（2）网络图

优点：采用网络图进行进度控制，能够清晰地展现现在和将来完成的工作内容、各项工作单元间的关系。并且可以预先确定各任务的时差。了解关键作业或某一项进度的变化对后续工作和总工期的影响度，便于及时地采取措施或对进度计划进行调整。

缺点：不能系统地表达每项的起始时间与结束时间，不易于对单项任务的过程进行跟踪。

（3）里程碑图

优点：强调项目的关键节点、直观展示重要事件和进展。

缺点：不适合展示详细的任务进展；需要定期更新，以保持准确性。

10、常见的对软件规模进行估算的方法有哪些?

（1）经验法。根据管理人员以往的项目或领域的经验，对未来的工作量进行估计。

（2）类推法。将本项目的部分属性与高度类似的一个或几个完成的项目进行比对，适当调整后获得待估算项目的工作量、工期或成本估算值的方法。

（3）方程法。减小估算误差的方法：请多位专家分别估算程序的最小规模a，最可能的规模m，和最大规模b，再计算出这三种规模的平均值$\bar{a}$，$\bar{b}$和$\bar{m}$，然后使用下式计算出程序规模的估计值：

$$L=\frac{\bar{a}+4\bar{m}+\bar{b}}{6}$$

11、软件项目成本估计技术有哪些?

（1）代码行估计成本。根据代码行估计成本是比较简单的定量估算方法，它把开发每个软件功能的成本和实现这个功能需要用的源代码行数联系起来。通常根据经验和历史数据估计实现一个功能需要的源程序行数。

（2）任务分解估计成本。这种方法首先把软件开发工程分解为若干个相对独立的任务，再分别估计每个独立的开发任务的成本，最后加起来得出软件开发工程的总成本。

12、软件项目效益分析的度量有哪些?

软件项目的效益可以分为有形效益和无形效益。有形效益一般指经济效益，无形效益主要是从性质上和心理上进行衡量，很难进行量的比较。但是无形效益有特殊的潜在价值，且在某些情况下会转化成有形的效益。经济效益可用投资回收期、纯收入等指标进行度量。

13、画出考试教务系统的系统流程图和数据流图。

14、画出公交车一卡通下车刷卡的分层数据流图。

15、画出公交车一卡通刷卡软件的工作分解结构。

第四章

1、常见的需求有哪些错误?

需求不明确、隐含需求忽视、变更管理不规范、需求冲突和客户参与不足。

2、需求工程包括哪些活动？简要说明其内容。

需求工程活动主要围绕着需求的获取和分析、文档的编写和需求验证进行。需求工程包括需求开发和需求管理。需求开发可进一步分为需求获取、需求分析与建模、编写需求规格和需求验证 4 个阶段。需求管理包括需求基线、需求变更、需求跟踪。

（1）需求开发的任务

①需求获取

深人实际，在充分理解用户需求的基础上，积极与用户交流，捕捉、分析和修订用户对目标系统的需求，并提炼出符合解决领域问题的用户需求。

②需求分析与建模

对已获取的需求进行分析和提炼，并进行抽象描述，建立目标系统的概念模型，进一步对模型（原型）进行分析。

③编写需求规格

对需求模型进行精确的、形式化的描述，计算机系统的实现提供基础。在需求分析结束后，会形成 2 份文档：用户需求说明书和软件需求规格说明书。

④需求验证

以需求规格说明为基础输入，通过符号执行、模拟或快速原型等方法，分析和验证需求规格说明的正确性和可行性，确保需求说明准确、完整地表达系统的主要特性。

（2）需求管理的任务

需求管理贯穿整个需求工程，需求管理主要通过需求基线、需求变更、需求跟踪来支持需求演进。由于客户的需要总是不断变化的，因此进化需求是十分必要的。

①需求基线

需求基线是指开发团队承诺的在某一特定版本中实现的功能和非功能需求的一组集合。

②需求变更

实际上，在软件开发过程中，所有的需求都有可能变化，为了有效地控制和适应需求的变化，对需求变更的管理就成为需求工程的另外一类重要活动。

③需求跟踪

需求跟踪是指将单个需求和其他系统元素之间的依赖关系和逻辑联系建立跟踪，主要包括从用户需求到软件需求的跟踪、从软件需求到下游产品的跟踪。

3、什么是功能性需求？什么是非功能性需求？

功能性需求描述系统预期提供的功能或服务，包括系统应提供的服务，系统如何对输入做出反应，以及系统在特定条件下的行为。

非功能性需求常常指不与系统功能直接相关的一类需求，主要反映用户提出的对系统的约束，与系统的总体特性有关，如可靠性、反应时间、存储空间等。

4、对功能需求的要求是什么？

系统的功能需求描述应该完整、一致和准确。

完整性意味着应该对用户所需的所有服务全部给出描述。一致性意味着需求描述不能前后矛盾。准确性是指需求不能出现模糊和二义性的地方。

5、非功能需求存在什么问题？

一是信息传递的无效性，二是忽略了非功能需求的局部性。

6、需求获取的常用方法有哪些？

（1）面谈法：渐入佳境

（2）需求专题会：穷追猛打

（3）问卷调查：暗藏玄机

（4）现场观察：身临其境

7、用户需求规格和软件需求规格的区别是什么？

（1）编写者不同

用户需求是由用户编写（比如软件外包中甲方提供的技术协议或软件研制任务书）或者由用户阐述开发方的需求分析人员编写（大多数信息化系统的用户需求开发都是这种模式），再或者是由系统设计师编写（多数军软开发的模式）。

软件需求不像用户需求那么复杂，都是由开发方的软件需求分析人员编写。

（2）获取来源不同

用户自己编写的用户需求来源于用户的业务领域经验，客户阐述开发方的需求分析人员编写的用户需求来源于用户提供的各种资料以及开发方对用户的需求访谈记录，系统设计师编写的用户需求来源于系统方案。

软件需求是则是来源于用户需求以及其他利益相关方比如开发方管理部门所附加的需求。

（3）对应的测试级别不同

用户需求对应验收测试或确认测试，是编写验收测试或确认测试的用例的主要依据。

软件需求对应配置项测试，用以验证软件实现是否正确，是编写软件配置项测试的用例的主要依据。

（4）可裁剪性不同

用户需求文档不一定是必须的。比如，用户需求可以合并到产品的需求文档中。实际上，修订中的 GJB438C 就准备去掉软件研制任务书的要求。

而软件需求的文档（即软件需求规格说明）则是必须的。

（5）需求文档的主要内容不同

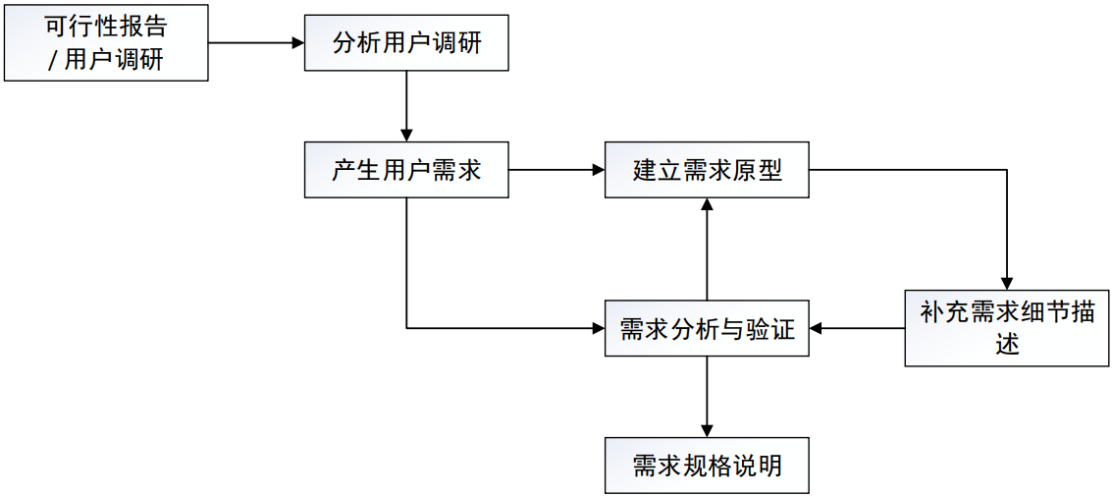
分析方法	用户需求内容	软件需求内容
结构化分析方法	系统的目标、系统的范围、系统的运行环境、系统的使用场景、系统结构图、业务流程图、功能性需求、非功能性需求、其他约束	功能性需求、非功能性需求、产品的分解结构、产品构建的需求、产品的外部结构需求、产品构建之间的接口、需求的优先级与分类、系统的数据视图、系统的处理流程、系统的设计约束
面向对象分析	系统的目标、系统的范围、系统的使用场景、业务用例、系统用例、非功能性需求、其他因素	系统的目标与范围、业务用例图、业务用例描述、系统用例图、系统用例描述、对用例的补充性说明、领域模型、系统的设计约束

（6）描述的详细程度不同

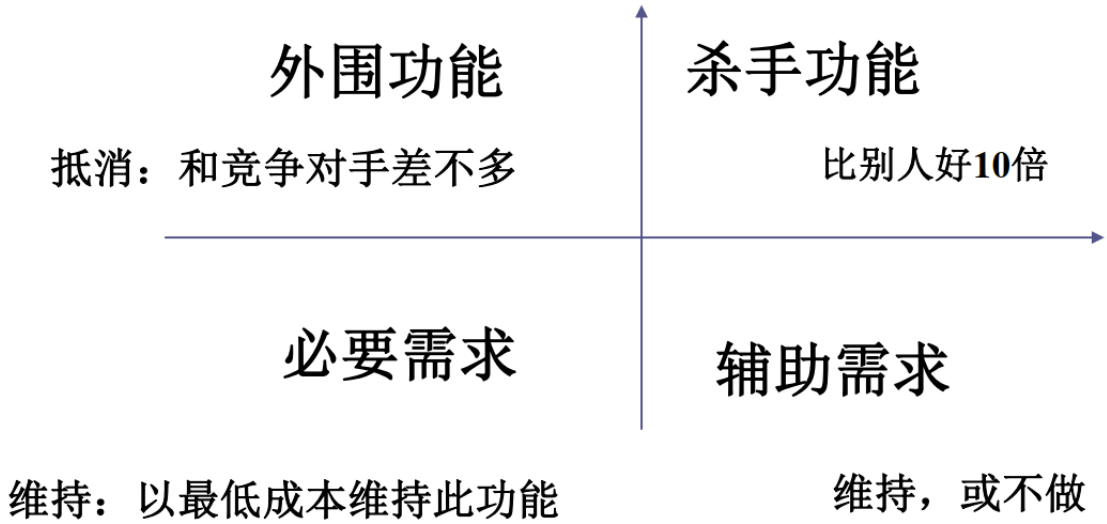
用户需求的描述中最重要的是要把软件系统的目标与范围描述清楚。

软件需求的描述则要更详细一些，软件的需求描述必须要可实现、可测试。

8、需求分析的流程是什么?



9、如何给不同的功能确定优先级?



10、需求验证从哪几个方面进行?

（1）有效性检查—指功能需求是否符合用户所提出的需求。每一项需求都必须准确地陈述其要开发的功能。

（2）一致性检查—与其他软件需求或高层需求不相矛盾，没有二义性。

（3）完整性检查—是否包含所有系统需求、用户的需求和约束。遗漏很难查出。注重用户的任务而不是系统的功能将有助于避免遗漏。

（4）可行性检查—需求是否实际可行。

（5）必要性检查—每一条需求描述都是用户需要的，可回溯

（6）可检验性检查—是否能设计出测试案例来验证需求。

11、需求变更的原因是什么?

（1） 合同签订马虎。签订合同时缺乏对客户需求认真对待。销售为使客户能够快速的签订合同，往往草率决定和片面同意客户提出的需求。

（2） 调研时没有深入理解客户需求，在上线前的需求调研分析阶段，项目组成员和客户的深入交流是减少频繁需求变更的关键阶段之一。但是由于双方的误解通常使需求交流难以进行。

（3） 开发过程周期长，用户的想法随时间改变了。

（4） 没有明确的需求变更管理流程，就会使需求变更变得泛滥。并不是所有的变更都要修改，也不是所有变更都要立刻修改，需求变更管理的目的是为了决定什么类型的变更需要修改和什么时候修改。

（5） 没有让客户知道需求变更的代价是需求变更泛滥的根本原因。变更都是有代价的，应该要评估变更的代价和对项目的影响，要让客户了解需求变更的后果。

12、需求变更管理的流程是什么?

（1）通过一个合适的渠道接受一项需求变更请求。

（2）变更控制委员会评估申请的变更 分析变更需求的理由是否充分，变更的技术可行性，估计实现变更的代价和可能产生的影响。

（3）分析需求变更可能影响的软件元素。数据库、系统设计元素、数据流程图、数据实体关系图、数据字典、处理说明、类和对象定义、类的方法、过程或函数。开发文档、用户文档、培训文档、联机文档和管理文档的更改，源程序，单元测试、集成测试、系统测试和验收测试方案等。

（4）制定需求变更影响分析报告表和需求变更工作量统计表，然后由需求变更控制委员会作出是否采纳变更的决策。

（5）一旦决定变更，应该及时通知所有相关的人员，最后，要按一定的程序来实施需求变更。

13、需求跟踪的实现方法有哪些?

（1） 正向跟踪。 检查《产品需求规格说明书》中的每个需求是否都能在后继工作成果中找到对应点。

（2） 逆向跟踪。 检查设计文档、代码、测试用例等工作成果是否都能在《产品需求规格说明书》中找到出处

14、画出图书馆借书系统的实体-关系图。

15、画出网上购物网站中订单的状态转换图。

第五章

1、面向对象分析和设计解决了哪两个经典问题?

问题一：传统的分析和设计方法将数据模型和处理模型分离。处理模型和数据模型实际上是描述解决一个问题的两个方面，但是建立两个模型使得系统分析人员很难检查它们的一致性和正确性。

问题二：如何从系统分析平滑地过渡到系统设计？传统的分析方法中分析模型到设计模型转变并不规范，容易引入错误。

2、如何确定用例?

①与系统实现有关的主要问题是什么?

②系统需要哪些输入/输出？这些输入/输出从何来？到哪里去?

③执行者需要系统提供哪些功能?

④执行者是否需要对系统的信息进行读、创建、修改、删除或存储?

3、如何确定执行者?

①谁使用系统的主要功能(主执行者)?

②谁需要从系统获得对日常工作的支持和服务?

③谁需要维护管理系统的日常运行(副执行者)?

④系统需要控制哪些硬件设备?

⑤系统需要与其他哪些系统交互?

⑥谁需要使用系统产生的结果?

4、用例之间的关系有哪些?

泛化关系、包含关系、扩展关系。

5、活动图的特点和作用是什么?

（1）特点：

①活动图可以有效地描述整个系统的流程，即活动图描述的是系统的全局的动态行为。

②系统任务中存在大量的并发活动，只有活动图是唯一能够描述并发活动的 UML 图。

③活动图还描述了系统中各种活动的执行的顺序，刻画一个方法中所要进行的各项活动的执行流程。

④活动图能够充分刻画不同参与者之间的交互行为。

（2）作用：

①描述用例的行为

活动图对用例描述尤其有用，它可建模用例的工作流，显示用例内部和用例之间的路径；它也可以向读者说明需要满足什么条件用例才会有效，以及用例完成后系统保留的条件或者状态。

②理解工作流程

活动图对理解业务处理过程十分有用。可以画出描述业务工作流的活动图与领域专家进行交流，明确业务处理操作是如何进行的将会有怎样的变化。

③描述复杂过程的算法

在这种情况下使用的活动图不过是 UML 版的程序流程图，常规的顺序、分支过程在活动图中都能得到充分的表现。

6、活动图包括哪些元素？

活动、操作、状态、转移、决策、分岔、联结、泳道

7、画出网上订票系统的用例模型。

8、画出网上订票的活动图。

9、分析打印机的状态并画出打印机的状态图。

10、完善手机打车软件用例图，补充到达目的地后相关用例。

第六章

1、什么是软件结构？有哪几种？

软件结构就是构成软件的模块的组织方式。模块之间可以有各种关系。一般可表示为层次结构和网状结构两种。

2、简述总体设计的步骤。

①设想供选择的方案（低、中、高）

②选取审查合理的方案（参照可行性与投资）

③功能分解和设计软件结构（使功能显而易懂）

④数据库设计

⑤制定测试计划（测试计划从需求阶段开始）

⑥编制设计文档

⑦审查和复查

3、什么是模块的影响范围？什么是模块的控制范围？它们之间应该建立什么关系？

一个模块的作用范围（或称影响范围）指受该模块内一个判定影响的所有模块的集合。一个模块的控制范围指模块本身以及其所有下属模块（直接或间接从属于它的模块）的集合。一个模块的作用范围应在其控制范围之内，且判定所在的模块应在其影响的模块在层次上尽量靠近。如果再设计过程中，发现模块作用范围不在其控制范围之内，可以用“上移判点”或“下移受判断影响的模块，将它下移到判断所在模块的控制范围内”的方法加以改进。

4、模块化要遵循哪些原则？

（1）改进软件结构提高模块独立性。

（2）在满足模块化要求的前提下尽量减少模块数量，在满足信息需求的前提下尽可能减少复杂的数据结构。

（3）模块规模应该适中。

（4）深度、宽度、扇出和扇入都应适当。

（5）模块的作用域应该在控制域之内。

（6）力争降低模块接口的复杂程度。

5、模块分解时，是否将系统分解得非常细，得到的功能模块越多越好呢？为什么？

不是。如果模块是相互独立的，当模块变得越小，每个模块花费的工作量越低；但当模块数增加时，模块间的联系也随之增加，把这些模块联接起来的工作量也随之增加。

6、模块的外部特性和内部特性分别包括哪些内容?

（1）模块的外部特性

模块的模块名、参数表、其中的输入参数和输出参数，以及给程序以至整个系统造成的影响。

（2）模块的内部特性

完成其功能的程序代码和仅供该模块内部使用的数据。

7、衡量模块独立性的两个标准是什么？请分别说明。

模块独立的度量标准：内聚和耦合。

耦合是模块之间的互相连接的紧密程度的度量。

内聚是模块功能强度（一个模块内部各个元素彼此结合的紧密程度）的度量。

8、内聚包括哪几种？分别指什么?

一般模块的内聚性分为七种类型。

（1）功能内聚

一个模块中各个部分都是为完成一项具体功能而协同工作，紧密联系，不可分割的。则称该模块为功能内聚模块。功能内聚模块是内聚性最强的模块。

（2）顺序内聚

指如果一个模块内处理元素和同一功能密切相关，而这些处理元素必须顺序执行，称为顺序内聚。通常一个处理元素的输出是另一个处理元素的输入。

（3）通讯内聚

指如果一个模块内各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，则称之为通信内聚模块。

（4）过程内聚

指在使用流程图作为工具设计程序时，常常通过流程图来确定模块划分。把流程图中的某一部分划出组成模块，就得到过程内聚模块。这类模块的内聚程度比时间内聚模块的内聚程度更强一些。

（5）时间内聚

又称经典内聚。这种模块大多为多功能模块，但要求模块中的各个功能必须在同一时间段内执行。如初始化模块和终止模块，处理故障模块。

（6）逻辑内聚

指模块把几种相关的功能组合在一起，每次被调用时，由传送模块的判定参数来确定该模块应执行哪一种功能。

（7）巧合/偶然内聚

指如果一个模块由完成若干毫无关系（或关系不大）的功能处理元素偶然组合在一起的，称为巧合内聚。巧合内聚是最差的一种内聚。

9、模块的耦合性由低到高分为哪些?

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。

10、简述结构化程序设计方法的基本要点。

首先研究、分析和审查数据流图。从软件的需求规格说明中弄清数据流加工的过程，检查有无遗漏或不合理之处，对于发现的问题及时解决并修改。

根据数据流图决定问题的类型。数据处理问题典型的类型有两种，分别是变换型和事务型，针对两种不同的类型分别进行分析处理。

由数据流图推导出系统的初始结构图。

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。

描述模块功能、接口及全局数据结构，修改和补充数据字典。

复查，如果出现错误，转入第二步修改完善，否则进入详细设计，同时制定测试计划。

11、简述软件结构设计优化准则。

（1） 输入部分优化。对每个物理输入设置专门模块，以体现系统的外部接口，其它输入模块并非真正输入，当它与转换数据的模块都很简单时， 可将它们合并成一个模块。

（2） 输出部分优化。为每个物理输出设置专门模块，同时注意把相同或类似的物理输出模块合并在一起，以降低耦合度，提高初始结构图的质量。

（3） 变换部分优化。根据设计准则，对模块进行合并或调整。

12、变换分析设计与事务分析设计有什么区别?

变换分析设计合用于具备明显变换特性数据流图，事务分析设计合用于具备明显事务特性数据流图。

13、画出手机打车软件的系统结构图。

14、画出公交一卡通刷卡软件的系统结构图。

第七章

1、简述详细设计的过程。

①确定每个模块的算法，选择适当工具表达算法执行过程；

②确定每一个模块的数据结构；

③为每一个模块设计一组测试用例；

④编写《详细设计说明书》；

⑤设计评审。

2、结构化程序设计应遵循的基本原则是什么?

①采用自顶向下、逐步求精的模块化方法设计程序

②尽量使用“基本结构”编程

③限制或不使用 goto 语句

3、详细设计有哪几种描述方法?

图形工具、表格工具、语言工具

4、简述详细设计的图形描述工具，以及各自的概念和优缺点。

（1）程序流程图

程序流程图独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。它以对控制流程的描绘直观、易于掌握而被设计人员青睐。

优点:

对控制流程的描绘很直观，便于初学者掌握。

缺点:

①本质上不具备逐步求精的特点，对于提高大型系统的可理解性作用甚微；

②程序流程图中用箭头代表控制流，因此程序员不受何约束，可以完全不顾结构化程序设计的精神，随意转移控制；

③不易表示数据结构。

（2）N-S 图

N-S 图是 Nassi 和 Shneiderman 提出来的，它体现了结构程序设计精神，是目前过程设计中广泛使用的一种图形工具。也叫盒图。

优点:

①功能域（一个特定控制结构的作用域）明确，可以从盒图上一眼看出来。

②没有箭头不允许随意转移控制，不可能任意转移控制。

③很容易确定局部和全程数据的作用域。

④很容易表现嵌套关系，也可以表示模块的层次结构。

缺点：

随着程序内嵌套的层数增多时，内层方框越来越小，会增加画图的难度，影响清晰度。

（3）PAD 图

用结构化程序设计思想表现程序逻辑结构的图形工具。PAD 图是一种由左往右展开的二维树型结构。PAD 图的控制流程为自上而下、从左到右地执行。

PAD 图的主要优点：

①程序结构清晰，结构化程度高。图中的竖线为程序的层次线，最左边竖线是程序的主线，其后一层一层展开，层次关系一目了然。

②支持自顶向下，逐步求精的设计方法。

③既可以表示程序逻辑，也可以描绘数据结构。

④用 PAD 图表现程序逻辑，易读易写，使用方便。

⑤容易转换成高级语言源程序，也可用软件工具实现自动转换。

5、求数组$a[0]$~$a[10]$中的次大值，请画出程序流程图。

6、请画出判断某一年是不是闰年的程序 N-S 图。

7、请画出公交一卡通刷卡扣款软件的 PAD。

8、请画出电梯根据某一层按键做出响应的 PDL 表示。

9、请以是否为头等舱、国内乘客、残疾乘客的顺序画出计算 7.3.4 节中超重行李费用的判定树。

第八章

1、类之间的关系有哪几种?

泛化、实现、关联、聚合、组合、依赖

2、使用顺序图对系统进行建模时，要遵循什么策略?

（1）设置交互语境，这些语境可以是系统、子系统、操作、类、用例和协作的一个脚本;

（2）通过识别对象在交互中扮演的角色，根据对象的重要性，将其从左向右的方向放置在顺序图中。

（3）设置每个对象的生命线。

3、画顺序图的一般步骤是什么?

1） 确定需要建模的工作流。

2） 从左到右布置对象。

3） 添加消息和条件以便创建每一个工作流。

4） 绘制总图以便连接各个分图。

4、在顺序图中消息有几种类型？返回消息是必须的吗?

总共有 4 种类型的消息。同步消息、异步消息、返回消息、简单消息。

返回消息不是必须的。

5、顺序图和协作图的区别是什么?

序列图和协作图都可以表示对象间的交互关系，但它们的侧重点不同。

序列图用消息的几何排列关系来表达对象间交互消息的先后时间顺序。

而协作图则建模对象（或角色）间的通信关系。一般适合大于 2 个参与者的情况。

6、画出客户在 ATM 上取款用例的类图。

7、画出在网上订火车票的对象图。

8、画出在网上购物的顺序图。

9、画出手机打车软件的构件图。

第九章

1、简述软件安全的原则。

原则 1：让最弱的环节变得安全

原则 2：实践纵深防御

原则 3：安全的错误退出

原则 4：最小权限原则

原则 5：权限分离原则

原则 6：保护机制的简单性

原则 7：最小共享机制

原则 8：勉强信任原则

原则 9：不要认为你的秘密是安全的

原则 10：完全仲裁（完全控制）

原则 11：心理接受能力

2、简述威胁建模的步骤。

1) 分解应用程序，软件建模；

2) 确定系统所面临的威胁，威胁建模；

3) 以风险递减的方式给威胁排序，威胁分级；

4) 选择应付威胁的方法，威胁消减。

3、什么是 STRIDE 威胁模型?

STRIDE 建模方法由 Microsoft 公司提出，该方法通过审查系统设计或架构来发现或纠正设计级的安全问题。

假冒 (Spoofing)，典型的例子是使用其他用户的认证信息进行非法访问。

篡改 (Tampering)，在未授权的情况下恶意地修改数据。这种修改可以是在数据库中保存的数据，也可能是在网络中传输的数据。

否认 (Repudiation)/可抵赖，用户从事一项非法操作，但该用户拒绝承认，且没有方法可以证明她是在抵赖。

信息泄漏(Information disclosure)，信息暴露给不允许对它访问的人。例如用户读到没有给她赋予访问权限的文件的内容，信息在网络中传递时内容被泄密。

拒绝服务(Denial of service)，拒绝对正当用户的服务。

权限提升(Elevation of privilege)，一个没有特权的用户获得访问特权，从而有足够的权限做出摧毁整个系统的事情。

4、什么是 DREAD 模型?

当对软件的各个组件进行威胁建模之后，需要对记录的威胁进行分析，确定各个威胁的等级，从而采取相应的措施减缓威胁。使用 DREAD 计算风险。

潜在破坏性 Damage potential

再现性 Reproducibility

可利用性 Exploitability

影响用户 Affected users

可发现性 Discoverability

5、建立手机打车软件的 STRIDE 威胁模型。

（1）用户端应用 (Mobile Client)

- **Spoofing (伪装)**:
  - 攻击者可能伪装成合法用户，使用伪造的身份信息登录或下单。
  - 可以通过分析 APP 的身份验证机制，确保防止不当的身份冒充。
- **Tampering (篡改)**:
  - 攻击者可能篡改本地应用数据，例如更改位置、修改价格、改变订单信息。
  - 需要确保数据传输和存储加密，防止篡改。
- **Repudiation (否认)**:
  - 用户可能否认曾经发起某个订单或支付。
  - 应该记录用户行为的审计日志，确保能追踪和验证所有操作。
- **Information Disclosure (信息泄露)**:
  - 应用可能泄露用户的私人信息，如位置、行程、支付信息等。
  - 需要加密用户数据，并限制应用获取的信息权限。
- **Denial of Service (拒绝服务)**:
  - 攻击者可能通过过载网络或服务器，导致应用无法使用或服务瘫痪。
  - 需要设计高可用性架构并采取流量监控和限制策略。

- Elevation of Privilege (权限提升):
    - 攻击者可能尝试通过漏洞提升应用权限，以获取更多的操作权限。
    - 需要严格的权限控制和代码审计。

（2）服务器端 (Backend Server)

- Spoofing (伪装):
    - 攻击者可能伪装成合法用户或管理员，伪造请求或数据。
    - 应确保使用强认证机制，防止伪造请求，且验证 API 请求来源。
- Tampering (篡改):
    - 攻击者可能篡改后端数据库中的数据，如篡改价格、订单状态等。
    - 数据库操作需要通过加密通道，并对数据进行有效的校验。
- Repudiation (否认):
    - 后端可能无法追溯某些重要的操作，导致用户或系统无法被追责。
    - 后端需要完整的日志记录和审计机制。
- Information Disclosure (信息泄露):
    - 服务器可能泄露敏感数据，如用户个人信息、支付信息、司机信息等。
    - 应加密传输中的所有敏感数据，并对服务器访问控制做出严格限制。
- Denial of Service (拒绝服务):
    - 服务器可能受到大规模的请求，导致系统瘫痪或无法响应正常请求。
    - 需要部署负载均衡、限流措施，并做好抗 DDoS 的防护。
- Elevation of Privilege (权限提升):
    - 内部员工或攻击者通过漏洞获得不该有的管理权限。
    - 必须实施最小权限原则，定期审计权限，确保没有不必要的访问权限。

（3）支付系统 (Payment Gateway)

- Spoofing (伪装):
    - 攻击者可能伪装成支付平台，发起伪造的支付请求。

- 应确保支付接口使用强认证和加密协议（如 OAuth、SSL/TLS）。

- **Tampering (篡改)**:
  - 支付请求可能被篡改，例如支付金额、支付账户等。
  - 支付请求和响应都必须进行加密，确保数据完整性。
- **Repudiation (否认)**:
  - 支付平台或用户可能否认进行过支付交易。
  - 应该记录所有支付操作，并确保交易日志的可验证性。
- **Information Disclosure (信息泄露)**:
  - 支付过程中涉及敏感信息，如信用卡号、银行卡信息等。
  - 必须加密所有支付数据，并使用安全的支付网关。
- **Denial of Service (拒绝服务)**:
  - 攻击者可能通过支付系统进行拒绝服务攻击，导致支付平台不可用。
  - 需要对支付接口进行负载均衡和 DDoS 防护。
- **Elevation of Privilege (权限提升)**:
  - 内部人员或攻击者通过漏洞提升权限，修改支付记录或资金流向。
  - 应定期审计支付系统的权限，确保没有过度权限。

（4）通信通道 (Communication Channel)

- **Spoofing (伪装)**:
  - 攻击者可能通过伪造的服务器或设备与用户端通信，进行中间人攻击。
  - 应确保客户端和服务器之间的通信使用 TLS 或其他加密协议进行保护。
- **Tampering (篡改)**:
  - 攻击者可以篡改用户和服务器之间的通信内容，如篡改位置数据、订单信息等。
  - 使用端到端加密协议，防止数据被篡改。
- **Repudiation (否认)**:
  - 双方可能无法确定通信双方的身份，导致无法追溯操作。

- 应使用数字签名验证通信双方的身份。

- **Information Disclosure (信息泄露)**:
  - 在传输过程中可能泄露敏感数据，如位置、行程、支付信息等。
  - 应使用加密技术保护通信内容，避免泄露敏感数据。

- **Denial of Service (拒绝服务)**:
  - 中间人攻击或流量攻击可能导致通信渠道无法使用。
  - 应部署抗 DDoS 机制，限制不良流量。

- **Elevation of Privilege (权限提升)**:
  - 攻击者可能通过通信漏洞获取更高权限的控制。
  - 定期进行安全性评估，确保通信协议的安全。

（5）数据库 (Database)

- **Spoofing (伪装)**:
  - 攻击者可能通过伪装身份访问数据库，篡改数据或进行未授权操作。
  - 必须使用强身份验证机制，并对所有访问进行日志记录。

- **Tampering (篡改)**:
  - 攻击者可以修改数据库中的订单、价格、用户数据等。
  - 应使用数据完整性校验，防止数据篡改。

- **Repudiation (否认)**:
  - 攻击者可能否认已访问或修改数据库中的数据。
  - 应该对所有数据库操作进行审计，并保留日志。

- **Information Disclosure (信息泄露)**:
  - 数据库可能包含敏感信息，攻击者可能通过漏洞获取。
  - 数据库中的敏感数据应进行加密处理，并采取访问控制策略。

- **Denial of Service (拒绝服务)**:
  - 攻击者可能对数据库进行拒绝服务攻击，导致数据库不可用。
  - 应使用数据库负载均衡、高可用性设置和备份方案。

- **Elevation of Privilege (权限提升)**:
  - 数据库内部可能存在权限漏洞，导致攻击者获得更高权限。

- 需要定期审查数据库的权限配置，并实施最小权限策略。

6、在网上购物网站的登录系统中，存在恶意人员通过程序进行登录口令暴力破解的威胁，请分析该威胁的等级，并写出原因。

针对网上购物网站登录系统中存在恶意人员通过程序进行口令暴力破解的威胁，我们可以使用 DREAD 模型来进行评估。以下是各个维度的分析：

（1）潜在破坏性 (Damage potential)

暴力破解登录口令可以导致恶意用户获取其他用户的账户和敏感信息，甚至进行未授权的交易或窃取个人数据。对于购物网站来说，一旦账户被盗，可能会造成金钱损失、声誉损害及数据泄露。评分：8（破坏性较大，但需要暴力破解完成）

（2）再现性 (Reproducibility)

暴力破解登录口令是一个可重复的过程。只要恶意人员获取了用户名，就可以通过不同的攻击方式（如字典攻击、穷举攻击）不断尝试不同的口令，直到成功为止。评分：9（几乎是 100%可再现的，只要没有采取防御措施）

（3）可利用性 (Exploitability)

暴力破解攻击的可利用性取决于网站是否采取了防护措施，如账户锁定、验证码、限制尝试次数等。如果这些措施没有被有效实施，攻击者就能相对容易地进行攻击。评分：8（如果没有防护措施，攻击比较容易，但如果有验证码或锁定机制，难度会增加）

（4）影响用户 (Affected users)

如果登录系统遭受暴力破解攻击，所有使用该系统的用户都可能受到影响，尤其是账户中有资金或敏感数据的用户。尤其是在购物网站上，用户个人信息和支付数据被盗的风险较高。评分：9（影响的用户广泛，尤其是存在较高安全需求的用户）

（5）可发现性 (Discoverability)

暴力破解攻击比较容易被发现，通常系统会检测到异常的登录尝试并记录相关日志。系统可能会触发警报，管理员可以发现这些异常行为并采取措施。评分：7（可以发现，但可能需要一定时间）

综合评分：根据 DREAD 评分模型，我们可以得出以下计算：

$$DREAD = (8 + 9 + 8 + 9 + 7)/5 = 8.2$$

这个风险评分为 8.2，属于一个较高的风险等级。意味着暴力破解登录口令是一个需要特别关注和尽快解决的安全威胁。

第十章

1、为什么要研究人机交互设计?

人机交互（Human-Computer Interaction）是研究人、计算机以及它们之间相互关系的技术，人机交互研究的目的是有效地完成人与机器之间的信息传递。这里的交互泛指一种沟通，即用户与计算机之间的信息识别过程。软件交互界面为用户提供直观的、感性的信息，支持用户运用知识、经验、感知和思维等过程获取和识别界面交互信息。

2、人机交互界面有哪些类型?

人机交互界面可广义地分为硬件交互界面和软件交互界面。硬件交互界面与软件交互界面都可以完成信息的输人与输出。硬件交互界面属于硬件设计范畴，即产品的硬件与用户身体直接接触部分的设计。软件交互界面属于软件设计范畴，通过软件图形界面使产品的功能价值得以实现，使用户对于产品所传递的信息易于理解和应用，如计算机软件视窗的设计。软件交互界面通过屏幕来获得用户输人的信息，又通过图形元素、文字元素、色彩元素以及对这些元素的合理编排等视觉化的表征将信息反馈给用户，这种方式也是当今社会人们获取信息的主要方式。

3. 好的软件界面的特征有什么?

（1）可使用性

可使用性是指软件使用起来简单。

（2）灵活性

灵活性是指软件能够按照用户的希望和需要，提供不同详细程度的系统响应信息。

（3）适度复杂性

软件界面中信息量的规模和组织的复杂程度就是界面的复杂性。在完成预定功能的前提下，应使软件界面越简单越好。

（4）可靠性

软件界面的可靠性是指无故障使用的间隔时间。软件界面应能保证用户正确、可靠地使用系统，保证程序和数据的安全性。

（5）界面美观性

优美的界面可以令人赏心悦目。界面美的内涵在于其有效表达思想的能力。

4、在软件交互设计中人的因素有哪些?

人的固有能力、个性差异、种族与文化因素、工作环境差异等等。

5、为什么要以用户为中心设计软件界面?

以用户为中心的软件交互界面设计可以帮助用户提高工作的有效性和效率，减少在交互使用过程中可能对用户心理、生理所产生的不良影响。

6、简述软件交互设计的要求。

软件界面应适合于展现功能，软件界面应容易被用户理解，最少步骤、最高效率，考虑用户的多样性，保持术语的一致性，允许熟练用户使用快捷键，提供及时且有价值的反馈，提供预防错误的机制，允许撤销操作，减轻记忆负担。

7、软件界面设计最重要的是美观，这种观点对吗？为什么?

不对。用户界面一定要适合于软件的功能，这是最基本的要求。如果用户无法通过这个界面来使用软件，“易用性” 根本就无从谈起。

8、设计软件界面时需考虑的防错处理方式有哪些?

（1） 对输入数据进行校验。

（2） 对于在某些情况下不应该使用的菜单项和命令按钮，应当将其“失效”（ 变成灰色，可见但不可操作）或者“隐藏”。

（3） 执行破坏性的操作之前，应当获得用户的确认。例如用户删除一个文件时，应当弹出确认对话框。

（4） 尽量提供 Undo 功能，用户可以撤销刚才的操作。

（5） 当检测到用户操作错误时，提供简明的错误处理手段。

9、简述图形界面设计的一般原则。

（1）合理的版式布局

（2）色彩的设计原则

（3）按钮的设计原则

（4）菜单界面的设计

（5）输入界面的设计

（6）合理使用图形隐喻

（7）联机帮助的设计原则

10、填表输入界面设计的注意事项是什么?

①有明确的提示，使用户可以不需要学习、训练，也不必记忆有关的语义、语法规则。

②填表输入界面充分地利用屏幕空间。

③在填表输入方式中，可以充分利用上下文信息，帮助用户完成输入 。

11、任选 4 种手机应用导航方式，举出具体例子。

（1）标签式导航。如，淘宝。

（2）列表式导航。如，学习强国。

（3）转盘式导航。如，抖音。

（4）隐喻式导航。如，我的汤姆猫。

第十一章

1、在软件开发阶段的工作流程中，测试人员测试和代码审查的顺序是否可以互换?

不可以。对经过测试的代码进行代码审查。

2、软件编程语言经过了几个发展阶段?

1 代：机器语言

2 代：汇编语言

3 代：高级程序设计语言，C, C++, Java

4 代：新一代编程语言，Python

3、开发软件时选择编程语言要考虑哪些因素?

（1）项目的应用领域：应尽量选取适合某个应用领域的语言

（2）算法和计算复杂性：要根据不同语言的特点，选取能够适应软件项目算法和计算复杂性的语言。

（3）软件的执行环境：要选取机器上能运行且具有相应支持软件的语言。

（4）性能因素：应结合工程具体性能来考虑,例如实时系统要求速度，就应选择汇编语言。

（5）数据结构的复杂性：要根据不同语言构造数据结构类型的能力选取合适的语言。

（6）软件开发人员的知识水平以及心理因素。知识水平包括: 专业知识，程序设计能力。心理因素：如对某种语言或工具的熟悉程度。受外界的影响（盲目追求高、新)。

4、为了提高软件的可维护性，在编码阶段应注意哪些问题?

使用标准的控制结构;

源程序文档化：标识符的命名、程序的注释、程序的视觉组织;

数据说明;

语句结构;

输入和输出。

5、什么是程序设计风格?

程序设计风格指一个人编制程序时所表现出来的特点，习惯逻辑思路等。

6、标识符命名的通用规则是什么?

（1）标识符的命名应当直观，可以望文知义。

（2）长度符合最小长度下的最大信息。

（3）变量名应当使用“名词”或“形容词+名词”

（4）函数名应当使用“动词”或者“动词+名词”的形式

（5）类和接口名首字母要大写

（6）常量名全大写，在单词间用单下划线分隔

（7）变量名和参数名第一个单词首字母小写，而后面的单词首字母大写

7、为何要进行程序的注释？应该怎样进行程序的注释？

夹在程序中的注释是程序员与日后的程序读者之间通信的重要手段。

序言性注释：通常置于每个程序模块的开头部分，它应当给出程序的整体说明，对于理解程序本身具有引导作用。

功能性注释：功能性注释嵌在源程序体中，用以描述其后的语句或程序段是在做什么工作，或是执行了下面的语句会怎么样。而不要解释下面怎么做。

8、数据说明时的通用规则是什么？

（1）数据说明的次序应当规范化

（2）说明语句中变量安排有序化

（3）使用注释说明复杂数据结构

9、有一种循环结构，其流程图如图 11-4 所示，这种控制结构不属于基本控制结构：它既不是先判断型循环，又不是后判断型循环。请修改此流程图，将它改为用基本控制结构表示的等效流程图。

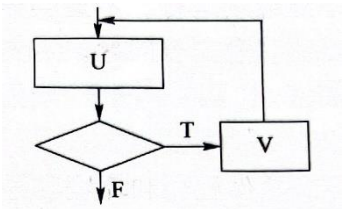
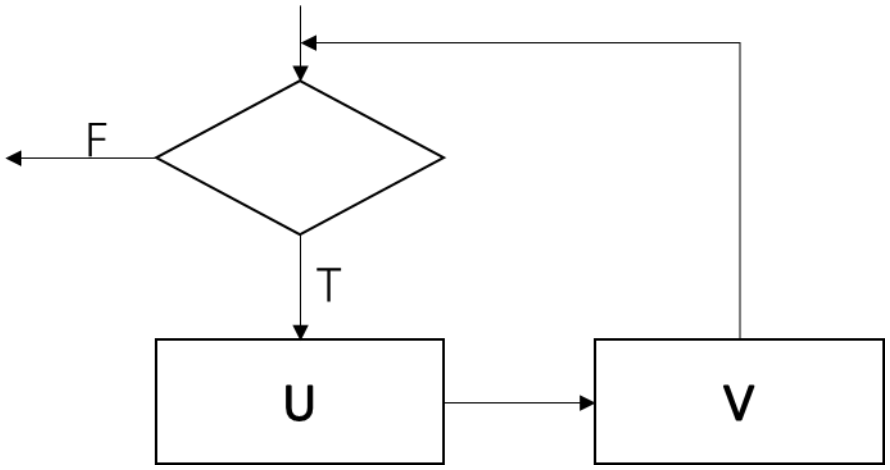


图 11-4　循环结构的流程图



第十二章

1、软件测试的概念和目的是什么?

测试的概念：测试是为了发现程序中的错误而执行程序的过程。

测试的目的：测试是为了发现程序中的错误而执行程序的过程；好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案；成功的测试是发现了至今为止尚未发现的错误的测试。

2、测试文档有哪些?

测试计划、测试规范、测试用例、测试报告、缺陷报告。

3、软件测试的原则是什么?

应制定测试计划并严格执行，排除随意性；

应尽早地开始软件测试；

避免程序员测试自己的程序；

注重设计测试用例，测试数据不仅要选择合理的输入数据，还要选择不合理的输入数据；

对发现错误较多的程序段，应进行更深入的测试；

测试不能证明软件无错；

软件缺陷的“免疫力”；

全面记录每一个测试结果；

长期保留测试用例等文档；

并非所有软件缺陷都修复。

4、非渐增式测试与渐增式测试有什么区别？渐增式测试如何组装模块?

（1）区别：

①非渐增式测试方法需要编写的测试用例较多，工作量较大；渐增式测试方法开销小。

②渐增式测试方法发现模块间接口错误早；而非渐增式测试方法晚。

③非渐增式测试方法发现错误，较难诊断；而使用渐增式测试方法，如果发生错误则往往和最近加进来的那个模块有关。

④渐增式测试方法测试更彻底。

⑤渐增式测试方法需要较多的机器时间。

⑥使用非渐增式测试方法，可以并行测试。

（2）渐增式测试就是逐个把未经测试的模块组装到已经过测试的模块上去进行集成测试，每加入一个新模块进行一次集成测试，重复此过程直到程序组装完毕。渐增式测试有两种不同的组装方法：自顶向下和自底向上结合。

5、一般驱动模块比桩模块容易设计，为什么?

因为驱动模块是模拟主程序或者调用模块的功能,处于被测试模块的上层，所以驱动模块只需要模拟向被测模块传递数据，接收、打印从被测模块返回的数据的功能，容易实现。而桩模块用于模拟那些由被测模块所调用的下属模块的功能，由于下属模块往往不只一个，也不只一层，由于模块接口的复杂性，桩模块很难模拟各下层模块之间的调用关系，同时为了模拟下层模块的不同功能，需要编写多个桩模块，而这些桩模块所模拟的功能是否正确，也很难进行验证。因此，驱动模块的设计显然比桩模块容易。

6、简述白盒测试的概念及相关技术。

白盒测试又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。盒子指的是被测试的软件，白盒指的是盒子是可视的，即清楚盒子内部的东西以及里面是如何运作的。白盒测试是在全面了解程序内部逻辑结构的情况下，对所有逻辑路径进行测试，是一种测试用例设计的方法。

逻辑覆盖是设计白盒测试方案的一种技术。是对一系列测试过程的总称。从覆盖源程序语句的详尽程度分为：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。

7、一个程序流程图如图 12-9 所示，请分别根据语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖设计测试用例。

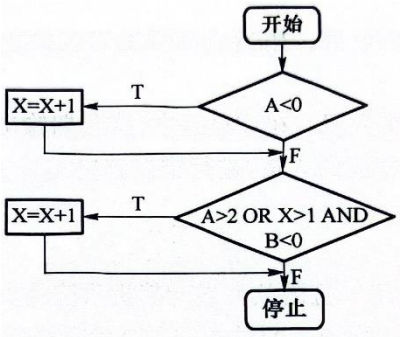


图 12-9　一个程序流程图

（1）语句覆盖（Statement Coverage）：

目标：执行程序中的每一条语句至少一次。

测试用例：

用例 1: A=-1, B=1, X=0（执行所有语句）

（2）判定覆盖（Decision Coverage）：

目标：每个判定的每个分支至少执行一次。

测试用例：

用例 1: A=-1, B=1, X=0（A<0 为真，进入第一个循环）

用例 2: A=3, B=1, X=0（A<0 为假，检查第二个判定）

（3）条件覆盖（Condition Coverage）：

目标：每个条件的每个可能结果至少执行一次。

测试用例：

用例 1: A=-1, B=1, X=0（A<0 为真）

用例 2: A=3, B=1, X=0（A<0 为假）

用例 3: A=3, B=-1, X=0（X>1 为假，B<0 为真）

用例 4: A=3, B=1, X=2（X>1 为真）

（4）判定/条件覆盖（Decision/Condition Coverage）：

目标：同时满足判定覆盖和条件覆盖的要求。

测试用例：

同判定覆盖和条件覆盖的测试用例组合。

（5）条件组合覆盖（Condition Combination Coverage）：

目标：测试所有条件的可能组合。

测试用例：

用例 1: A=-1, B=1, X=0（A<0 为真，X>1 为假，B<0 为假）

用例 2: A=3, B=1, X=0（A<0 为假，X>1 为假，B<0 为假）

用例 3: A=3, B=-1, X=0（A<0 为假，X>1 为假，B<0 为真）

用例 4: A=3, B=1, X=2（A<0 为假，X>1 为真，B<0 为假）

（6）路径覆盖（Path Coverage）：

目标：执行程序中所有可能的路径。

测试用例：

用例 1: A=-1, B=1, X=0（执行路径：开始 -> A<0 为真 -> X=X+1 -> A>2 OR X>1 AND B<0 为假 -> 停止）

用例 2: A=-1, B=-1, X=0（执行路径：开始 -> A<0 为真 -> X=X+1 -> A>2 OR X>1 AND B<0 为真 -> X=X+1 -> 停止）

用例 3: A=3, B=1, X=0（执行路径：开始 -> A<0 为假 -> A>2 OR X>1 AND B<0 为假 -> 停止）

用例 4: A=3, B=-1, X=2（执行路径：开始 -> A<0 为假 -> A>2 OR X>1 AND B<0 为真 -> X=X+1 -> 停止）

8、简述黑盒测试的概念及相关技术。

黑盒测试又称功能测试、数据驱动测试或基于规格说明的测试，是一种从用户观点出发的测试。黑盒测试被用来证实软件功能的正确性和可操作性。黑盒测试把待测程序当作一个黑盒，在不考虑程序内部结构和内部特性，只知道该程序输人和输出之间的关系或程序功能的情况下，仅依靠功能需求规格的说明书来确定测试用例，并推断功能是否正确。

相关技术：等价类划分、边界值分析、错误推测。

9、在某一 PASCAL 语言版本中规定“标识符是由字母开头的，后跟字母或数字的任意组合；有效字符数为 8 个，最大字符数为 80 个”，请写出测试数据等价类。

输入条件	有效等价类	无效等价类
标识符个数	1个（1），多个（2）	0个（3）
标识符字符数	1～8个（4）	0个（5），>8个（6），>80个（7）
标识符组成	字母（8），数字（9）	非字母数字字符（10），保留字（11）
第一个字符	字母（12）	非字母（13）
标识符使用	先说明后使用（14）	未说明已使用（15）

10、简述灰盒测试的概念及优缺点。

1999 年提出。单纯从名称上来看，灰盒测试是介于黑盒测试与白盒测试之间的一种测试方式。灰盒测试不像白盒那样详细、完整，但又比黑盒测试更关注程序的内部逻辑，常常是通过一些表征性的现象、事件、标志来判断内部的运行状态。灰盒测试多用于集成测试阶段，不仅关注输出、输入的正确性，同时也关注程序内部的情况。

优点:

①能够进行基于需求的覆盖测试和基于程序路径覆盖的测试；

②测试结果可以对应到程序内部路径，便于 bug 的定位、分析和解决；

③能够保证设计的黑盒测试用例的完整性，防止遗漏软件的一些不常用的功能或功能组合；

④能够降低需求或设计不详细或不完整对测试造成的影响。

缺点：

①投入的时间比黑盒测试大概多 20-40％的时间；

②对测试人员的要求比黑盒测试高，灰盒测试要求测试人员清楚系统内部由哪些模块构成，模块之间如何协作；

③不如白盒测试深入；

④不适用于简单的系统。所谓的简单系统，就是简单到总共只有一个模块。由于灰盒测试关注于系统内部模块之间的交互。如果某个系统简单到只有一个模块，那就没必要进行灰盒测试了。

11、软件测试阶段如何划分?

软件测试 V 模型从左到右描述了基本的开发过程和测试行为，明确了测试工程中存在的不同级别测试以及测试阶段和开发过程各阶段的对应关系。软件测试 V 模型指出，单元测试和集成测试是为了验证程序设计。单元测试对代码进行测试。集成测试是介于白盒测试与系统测试之间的一种测试，也叫灰盒测试。测试人员和用户进行软件的确认测试和验收测试，是依据需求说明书进行测试的，以确定实现的软件是否满足用户需求或合同要求。系统测试主要检测系统功能、性能的质量特性是否达到项目规划的指标。

软件测试 V 模型详细地描述了每个测试阶段所对应验证的对象，单元测试验收的对象是详细测试说明书，集成测试验证的对象是概要设计说明书，确认测试和系统测试验证的对象是需来说明书。

12、测试停止的依据是什么?

第一类标准：测试超过了预定时间，则停止测试。

第二类标准：基于测试用例的原则。在功能测试用例通过率达到 100%，非功能性测试用例达到 95%以上，允许正常结束测试。

第三类标准：随着测试的执行，软件已经满足了验收测试指定的功能和非功能性需求，则可以停止测试。

第四类标准：根据单位时间内查出故障的数量决定是否停止测试。直到发现的故障几乎为零或者单位时间内查出故障的数量小于设定的阈值，此时可以停止测试。

第五类标准：正面指出停止测试的具体要求，即停止测试的标准可定义为查出某一预订数目的故障。

13、模糊测试和渗透测试有什么区别?

模糊测试的技巧在于它是不符合逻辑的，是将尽可能多的把杂乱的数据输入软件中。

渗透测试指从一个攻击者的角度来检查和审核一个网络或软件安全性的过程。