第 1 章　绪论

1 软件工程概念的提出与发展

1.1 软件工程概念（简答题）

软件工程是应用计算机科学理论和技术以及工程管理原则和方法,按预算和进度实现满足用户要求的软件产品的工程,或以此为研究对象的学科。软件生产率、软件质量远远满足不了社会发展的需求 ,成为社会、经济发展的制约因素 ,把这一现象称为软件危机（填空题）;软件工程概念的提出是倡导以工程的原理、原则和方法进行软件开发（填空题）, 以期解决出现的软件危机。

1.2 软件工程概念的提出背景与发展历程

1.2.1 软件工程概念的提出背景（单选题）

软件工程这个术语首次出现是在 1968 年的 NATO 会议上。20 世纪 60 年代以来,随计算机的广泛应用,软件生产率、软件质量远远满足不了社会发展的需求,成为社会、经济发 的制约因素,这就是“软件危机”。而软件工程概念的提出就是为了倡导以工程的原理、原则方法进行软件开发,以期解决出现的“软件危机”。

1.2.2 软件工程的发展历程（单选题）

第一个时期：20 世纪 60 年代末到 80 年代初。

研究对象：主要围绕软件项目,开展了有关开发模型、开发方法和支持工具的研究

研究成果：

①提出了瀑布模型,试图为开发人员提供有关活动组织方面的指导。

②开发了过程式语言,如 Pascal 语言 C 语言、Ada 语言等。

③开发了一些方法,如 Jackson 方法、结构化方法等。

④开发了一些支持工具,如调试工具、测试工具等

第二个时期：20 世纪 80 年代以来

研究对象 :围绕对软件工程过程的支持,开展了一系列有关软件生产技术,特别 用技术和软件生产管理的研究和实践

（单选题）研究成果：

①提出了《软件生存周期过程》等一系列软件工程标准。

②计算机辅助软件工程（case）,软件复用，软件过程概念及能力成熟度模型。

③在工程技术方面,出现了面向对象语言,如 Smalltalk、C++、Eiffel 等

④提出了面向对象软件开发方法。

⑤在工程管理方面,开展了一系列过程改进项目。

2 软件开发的本质

2.1 软件的概念（填空题）

软件=程序+文档

（单选题）软件开发的本质概括为：不同抽象层术语之间，以及不同抽象层处理逻辑之间的映射。

2.2 模型概念

2.2.1 模型是在特定意图下所确定的角度和抽象层次上对物理系统的描述，对该系统边界的描述、对系统内各模型元素以及他们之间关系的语义描述。

2.3 软件系统模型分为：概念模型和软件模型。（单选题）

概念模型：描述软件是什么

软件模型：实现概念模型的软件解决方案。包括设计模型、实现模型和部署模型。

第 2 章 软件需求与软件需求规约

1 需求与需求获取

1.1 需求定义（单选题）

一个需求是有关一个“要予构造”的陈述,描述了待开发产品能上的能力、性能参数或其他性质。

1.2 需求特性（填空题）

(1)必要的,该需求是用户所要求的。(2)无歧义的,该需求只能用一种方式解释。(3)可测的,该需求是可进行测试的。(4)可跟踪的,该需求可从一个开发阶段跟踪到另一个阶段。 (5)可测量的,该需求是可测量的。

1.3 需求分类

功能需求：(1)关于该功能输入的所有假定,或为了验证该功能输入,有关检测的假定 (2)与外部有关的、功能内任一动作次序。(3)对异常条件的响应,包括所有内容或外部所产生的错误。 (4)需求的时序或优先程度。(5)功能之间的互斥规则。(6)系统内部状态的假定。(7)为了该功能的执行,所需要的输入和输出次序。(8)用于转换或内部计算所需要的公式等

非功能需求：(1)性能需求:规约了一个系统或系统构件在性能方面必须具有的一些特性。 (2)外部接口需求:规约了系统或系统构件必须与之交互的用户、硬件、软件或数据库元素,其中也可能规约交互格式、时间或其他因素等。接口需求可分为以下 7 类:用户接口、硬件 接口、软件接口、通信接口、内存约束、运行和地点需求。(3)设计约束:设计约束是一种需求,它限制了软件系统或软件系统构件的设计方案的范围。(4)质量属性:质量属性规约了软件产品所具有的一个性质(包括功能和其他需求)必须达到其质量方面一个所期望的水平。

1.5 需求发现技术（ 简答题）

自悟：需求人员把自己作为系统的最终用户，审视该系统并提出问题的初始发现需求技

术。（填空题）

交谈：需求人员通过提出问题／用户回答的方式，直接询问用户需要的初始发现需求技术。（填空题）

观察：通过观察用户执行其现行任务和过程，或通过观察他们如何操作与所期望的新系统，了解系统运行环境，特别是了解要建立的新系统与现存系统、过程以及工作方法之间必须进行的交互。

小组会：举行客户和开发人员的联席会议，与客户组织的一些代表共同开发需求。

提炼：通过复审有关需要的陈述，或功能和性能目标的陈述等技术文档，而获取相关需求的初始发现需求技术是提炼。（单选题）

2 需求规约

2.1 需求规约定义（简答题）

需求规约定义是一个软件项/产品/系统所有需求陈述的正式文档（单选题），它表示了一个软件产品/系统的概念模型。

2.2 需求规约特性（单选题、简答题）

(1)重要性和稳定性程度:按需求的重要性和稳定性,对需求进行分级,例如:基本需,可信选需求和期望需求。(2)可修改的:在不过多地影响其他需求的前提下,可以容易地修改一个单一需求。(3)完整的:没有被遗漏的需求。 (4)一致的:不存在互斥的需求。

2.3 规约需求的三种语言

2.3.1 非形式化的需求规约

以一种自然语言来表达需求规约,但是容易产生歧义、矛盾和不可测等问题（填空题）。一般适用于 规模比较小的,复杂程度不大高的小型软件项目,或在获取 SRS 时使用。

2.3.2 半形式化的需求规约

以半形式化符号体系(包括术语表、标准化的表达模式)来表达需求规约,半形式化的需求见约应遵循一个标准的表示模板。

(1)术语表明确地标识了一些词,可以基于某一种自然语言

(2)标准化的表达格式标识了一些元信息,支持以更清晰的方式系统化地来编制文档。

2.3.3 形式化的需求规约

以一种基于良构数学概念的符号体系来编制需求规约,一般伴有解释性注释的支持。(填空题)

(1)以数学概念来定义该符号体系的词法和语义。

(2)定义了一组支持逻辑推理的证明规则,并支持这一符号体系的定义和引用。

2.4 需求在软件开发中的作用(简答题)

(1)需求规约是软件开发组织和用户之间一份事实上的技术合同书,是产品 功能及其环境的体系。 (2)对于项目的其余大多数工作,需求规约是一个管理控制点(3)对于产品/系统的设计,需求规约是一个正式的、受控的起始点 (4)需求规约是创建产品验收测试计划和用户指南的基础,即基于需求规约一般还会产生 另外两个文档--初始测试计划和用户系统操作描述。

第 3 章 结构化方法

1 结构化需求分析

1.1 基本术语(单选题)

数据:客观事物的一种表示。

加工:数据变换单元

数据存储:数据的静态结构

数据源和数据潭:数据流的起点和归宿点。

1.2 数据流图（DFD 图）(综合应用题)

一种表示数据变换的图形化工具。 数据流程图的元素：数据源/数据潭，数据流，数据加工，数据存储

1.3 建模过程（简答题）

1、建立系统环境图，确定系统语境。 2、自顶向下，逐步求精，建立系统的层次数据流图。3、定义数据字典。4、描述加工。（结构化自然语言、判断树(DecisionTree)也称为决策树，是用来描述在一组不同的条件下，决策的行动是根据不同条件及其取值来选择的处理过程。业务规则的描述通常可以使用判断树这一过程描述工具。 判定表，描述加工的一种工具，通常用来描述一些不易用自然语言表达清楚或需要很大篇幅才能表达清楚。（单选题）

2 结构化设计

2.1 总体设计

总体设计引入两个基本概念：一是模块，二是模块调用。

2.2 总体设计步骤

变换型数据流程图是一个线性的顺序结构，由输入臂、输出臂和变换中心三部分组成。其中变换中心使系统数据发生本质的变化，输入臂将物理输入变换成逻辑输入，而输出臂则将逻辑输出变换成物理输出。

2.3 事务设计

2.2.1 如果模块为逻辑凝聚的模块，可以将它分解为一个检查业务类型的模块和一个调度模块，根据不同的业务类型，调度模块调用不同的下层模块。

2.4 模块化及其启发式规则

（1）模块：执行一个特殊任务的一个过程以及相关的数据结构。模块通常由两部分

组成：模块接口和模块体。

（2）模块化：把一个待开发的软件分解成若干个简单的、具有高内聚低耦合的模块，这一过程称为模块化。

（3）模块耦合：是对两个模块之间相互依赖程度的一种度量。模块间的依赖程度越大，则其耦合程度也就越大；反之，模块间的依赖程度越小，则其耦合程度也就越小。（单选题、填空题）

（4）模块间耦合类型（单选题）

内容耦合：一个模块直接修改或操作另一模块数据。

公共耦合：两个模块共同引用一个全局数据项。

控制耦合：一个模块向另一模块传递控制信号。

标记耦合：一个模块向两个模块传递一个公共参数。

数据耦合：模块之间通过参数来传递数据。

（5） 模块内聚：是指一个模块内部个成分之间相互关联程度的度量。也就是说，内聚是对模块内各处理动作组合强度的一种度量。很显然，一个模块的内聚越大越好。（单选题）

（6） 内聚的类型（单选题、填空题）

偶然内聚：模块的各成分没有任何关系 处理放在一起。

逻辑内聚：逻辑上相关的 。

时间内聚：模块内的功能在同一时间完成。

过程内聚：模块内的处理以特定的次序执行。

通信内聚：操作同一数据集。

顺序内聚：一个成分的输出作为另一成分的输入。

功能内聚：模块的所有成分完成单一的功能。

（7）启发式规则："高内聚、低耦合"

①改进软件结构，提高软件独立性。模块分解② 模块规模适中③力求深度、宽度、扇出、扇入适中。深度：表示其控制的层数。宽度：同一层次上模块总数的最大值。（填空题）扇出：一个模块直接控制的下级模块的数目。扇入：有多少个上级模块直接调用它。

④尽量使模块的作用域在其控制域内。

模块的控制域 这个模块本身以及所有直接或间接从属它的模块的集合。模块的作用域：受该模块内一个判断所影响的所有模块的集合。（单选题）

⑤尽力降低模块接口的复杂度

⑥力求模块功能可以预测

2.5 详细化设计工具

1.程序流程图：（1）顺序结构（2）选择结构（3）多分支结构（4）循环结构（填空题）

2.PAD 图 3.盒图。N - S（盒图）用三种符号表达 3 种控制结构，PAD 图用二维树形结构来表示程序的控制流。DFD（数据流图）用结构化分析方法表达了功能模型。（单选题）

第 4 章 面向对象方法—UML

1 UML 术语表

1．对象（object）：是系统中用来描述客观事物的一个实体。一个对象由一组属性和对这组属性进行操作的一组方法组成。

2． 类（Class）：是具有相同属性、操作、关系和语义的一组对象的集合，它为属于该类的全部对象提供了同一的抽象描述，其内部包括属性和服务两个主要部分。（单选题）

类的属性

①公有的,可以被其他类调用；②受保护的,只有其子类才能调用；

③私有的,只有本类的操作才能使用；④包内的,只有在同一包中声明的类才能使用

类的操作

①公有的,可以被其他类调用。②受保护的,只有其子类才能调用

③私有的,只有本类的操作才能使用。④包内的,只有在同一包中声明的类才能使用

3 .接口：是操作的一个集合，其中每个操作描述了类、构件或子系统的一个服务。

4. 协作：是一个交互，涉及交互的三要素：交互各方、交互方式以及交互内容。

5. 用况（usecase）/用例：对一组动作序列的描述，系统执行这些动作应产生对特定参与者有值的、可观察的结果。

6 .主动类:至少具有一个进程或线程的类。能够启动系统的控制活动，并且其对象的行为通常与其它元素行为并发的

7. 构件/组件:系统设计中的一种模块化部件，通过外部接口隐藏了它的内部实现,具有相同接口的构件可以相互替代，构件可以嵌套,构件用于表达解空间中可独立标识的成分。

8 .制品（Artifact）：系统中包含物理信息的、可替代的物理部件。部署制品：这类制品是构成一个可执行系统必要而充分的制品，例如：DLL、EXE 文件。工作产品制品：这类制品本质上是开发过程的产物，由源代码文件、数据文件等用来创建部署制品的事物构成。执行制品：这类制品是作为一个正在运行的系统的结果而被创建的。一般存在于内存之中

9 .节点是在运行时存在的物理元素，通常表示一种具有记忆能力和处理能力的计算机资源。

2 表达关系术语

1.关联反映了类和类之间的静态关系。关联在模型中，特别是在永久业务对象模型中是最基本的关系。关联是类目之间的一种结构关系，是对一组具有相同结构、相同链的描述。在 UML 中，用于描述关联的一定“内涵”的术语是关联名。（填空题）关联一端的类目对

另一端的类目的一种呈现，称为角色。（填空题）类中对象参与一个关联的数目称为该关联的多重性。（单选题）

2.泛化/继承：泛化是一般性类目（父类）和它的较为特殊性类目（子类）之间的一种关系，有时称为“is-a-kind-of”关系，UML 给出了以下 4 个约束：(1) 完整；(2) 不完整；(3) 互斥；(4) 重叠。3. 细化是类目之间的语义关系，其中一个类目规约了保证另一类目执行的契约。（简答题）

4.细化：类目之间都语义关系。

5.依赖是一种使用关系，用于描述一个类目，使用另一类目的信息和服务。用有向虚线段表示。依赖的分类：绑定、导出、允许、实例。关系、泛化和细化都是一类特定的依赖，可以保证 4 个术语能够表达客观世界中各种各样事物之间的关系。（单选题）

6. 如果在一个时间段内，整体类的实侧中至少包含一个部分类的实例，并且该整体类的实例负责创建和消除部分类的实例，特别是如果整体类的实例和部分类的实例具有相问的生存周期，那么这种关联关系称为组合。（填空题）

3. 面向对象的基本思想

获取需求,用各种文字说明、图形、表格等建立描述客观世界的抽象模型,识别与问题有关的类与类之间的联系,加上与实现环境有关的类(如界面等),逐步细化模型;经过设计的类与联系进行调整后,完成整个系统的描述,然后对类进行编码和测试,得到结果。

面向对象方法是一种根据客体之间的关系来建造系统模型的系统化方法，面向对象方法源于面向对象编程语言。（单选题）

3.1 UML 的模型表达式

在面向对象技术的发展中，一个重要的里程碑是 UML。（填空题）UML 的图形化工具分为两类：一类是结构图，用于表达系统或系统成分的静态结构模型，给出系统或系统成分

的一些行为信息；一类是行为图，用于表达系统或系统成分的动态模型，给出系统或系统成分的一些行为信息。（单选题）

四种建模工具：类图、use case 图、状态图和顺序图。

3.2 类图

类图是可视化地表达系统静态结构模型的工具，通常包含类、接口、关联、泛化和依赖关系等．有时为了体现高层设计思想，类图还可以包含包或子系统；有时为了凸显某个类的实例在模型中的作用，还可以包含这样的实例；有时为了增强模型的语义，还可在类图中给出与其所包含内容相关的约束.（单选题）

创建类图的四步

(1)模型化待建系统中的概念,形成类图中的基本元素。

(2)模型化待建系统中的各种关系,形成该系统的初始类图。

(3)模型化系统中的协作,给出该系统的最终类图

(4)模型化逻辑数据库模式。

3.3 用况图（简答题、综合应用题）

用况是对一个参与者（actor）使用系统的一项功能时所进行的交互过程的一个文字描述序列。 用况图是一种表达系统功能模型的图形化工具（单选题）。使用用况图可以实现业务建模和系统建模，业务模型和系统模型之间具有整体／部分关系，即业务模型是整体，系统模型是业务模型的一个组成部分。（单选题）

3.4 顺序图（简答题）

顺序图（sequence diagram）表示了对象之间传送消息的时间顺序，也就是对象之间的交互顺序。基本元素：活动者，指用况中的活动者。对象，指在用况中的内部对象。生命线：在顺序图中的一个对象下面的竖线，用以显示这个对象的生命期。消息，指场景内由

事件流定义的内部事件成为在对象和活动者或其他对象之间的消息。

3.5 状态图

状态图（state chart diagram）使用状态、事件和转换来记录对象在其生命周期中所历经的状态序列。

①对象的初始状态是图中任何事件都未对该对象起作用时的状态。

② 状态代表对象生命周期中的某一瞬间。

③ 转换表明作为对事件的响应结果，对象将从一种状态转换到另一种状态并执行某个动作。

④触发状态转换的事件在状态转换字符串中命名。双击一个状态转换 ,除事件签名以外，还可用字符串为其加注临界条件、动作表达式等标签。

第 5 章 面向对象方法——Rup

1 Rup 的特点（单选题、填空题）

以用况为驱动，以体系结构为中心，迭代、增量式开发。

2 核心工作流

2.1 需求获取（列出候选的需求、理解系统语境、捕获功能需求、捕获非功能需求）

2.2 需求分析

2.2.1 需求分析的目标：在系统用况模型的基础上，创建系统分析模型以及在该分析模型视角下的体系结构描述。

2.2.2 基本术语

分析类：是类的一种衍型，很少有操作和特征标记，而用责任来定义其行为，并且其属性和关系也是概念性的。（单选题）分为：实体类、边界类、控制类、用况细化（用况细化是一个协作。针对一个用况，其行为可以用多个分析类之间的相互作用细化，并记为用况细

化）、分析包。

2.2.3 分析模型的表达（简答题）

分析模型是由“分析系统”来定义的，分析系统包含一组具有层次结构的包，一个包可以包含一些分析类和用况细化。

2.3 分析的主要活动

体系结构分析、用况分析、类的分析、包的分析。

2.4 RUP 设计小结（简答题）

RUP 设计特点：（1）使用了一种公共的思想来思考设计，并使设计可视化。（2）给出了有关子系统、设计类和接口的需求。（3）支持对底线工作的分解，时之称为一些可以由不同开发组尽可能同时处理的、可管理的部分。

2.4.2 RUP 的设计方法

（1）给出表达设计模型的基本成分的术语。

（2）规约了设计模型的语法，指导模型的表达。

（3）给出了创建设计模型的过程以及相应的指导。

2.4.3 RUP 的设计模型（简答题、单选题）

（1）设计子系统和服务子系统，以及它们的依赖、接口和内容

（2）设计类，以及它们具有的操作、属性、关系及其实现需求

（3）用况细化

（4）设计模型视角下的体系结构描述

2.4.4 RUP 的部署模型

（1）节点，它们的特征以及连接

（2）主动类到节点的初始映射

2.4.5 设计模型与分析模型的比较（简答题）

分析模型	设计模型
概念模型	软件模型
可应用于不同的设计	特定于一个实现
使用了三个衍生类：边界、实体、控制	使用了多个衍生类，依赖实现的语言
几乎不是形式化的	是比较形式化的
开发费用少	开发费用多
结构层次少	结构层次多
动态的，但很少关注定序方面	动态的，但更多关注定序方面
概括给出了系统设计	表明了系统设计
整个生命周期内不能修改、增加等	整个生命周期内应该予以维护
为构建系统定义一个结构，是基本输入	构件系统时，尽可能保留分析模型所定义的结构

2.4.6 设计阶段的活动

（1）体系结构设计

（2）设计用况

（3）设计类

（4）设计子系统

2.4.7 RUP 设计对实现的影响

（1）设计子系统和服务子系统由实现子系统予以实现。

（2）设计类由文件化构件予以实现。

（3）在规划实现工作时，将要使用用况细化以产生一些“构造”。

（4）在节点上部署构件、形成分布系统时，将使用部署模型和网络配置。

2.5 RUP 的实现与测试

2.5.1 RUP 的实现目标（单选题）

（1）基于设计类和子系统生成构件

（2）对构成进行单元测试

（3）进行集成和连接

（4）把可执行的构件映射到部署模型

2.5.2 RUP 实现的主要活动

（1）实现体系结构（2）集成系统（3）实现子系统（4）实现类（5）完成单元测试

2.5.3 RUP 的测试

包括：内部测试、中间测试和最终测试

2.5.4 RUP 测试包括的主要活动

（1）计划测试（2）设计测试（3）实现测试（4）执行集成测试（5）执行系统测试

（6）评价测试

3.RUP 和 UML 之间的关系。（简答题）

（1）RUP 和 UML 构成了一种特定的软件开发方法学；

（2）UML 作为一种可视化建模语言，给出了表达事物和事物之间关系的基本术语，给出了多种模型的表达工具；

（3）RUP 利用这些术语定义了需求获取层、系统分析层、设计层、实现层，并给出了实现各层模型之间映射的基本活动以及相关的指导。

第 6 章 软件测试

1 软件测试目标

1.1 软件测试的对象（单选题，填空题）

软件 = 程序 + 文档

测试对象：各个阶段产生的源程序和文档。

1.2 软件测试定义（单选题）

按照特定规程发现软件错误的过程。在 IEEE 提出的软件工程标准术语中,对软件测试的定义是:使用人工或自动手段,运行或测定某个系统的过 程,其目的是检验它是否满足规定的需求,或清楚了解预期结果与实际结果之间的差异。

1.3 测试的**目的**

希望能以最少的人力和时间发现潜在的尽可能多的错误和缺陷。比如功能的错误，性能低下，易用性差。

1.4 “测试”和“调试”的区别(单选题)

测试证明“失败”，调试证明“正确”； 测试以已知条件开始， 测试时有计划的， 测试是一个发现错误、改正错误、重新测试的过程， 测试的执行是有规程的，测试由独立的测试小组完成，测试的执行和设计可由工具支持。

1.5 测试过程模型可分三类

环境模型:是对程序运行环境的抽象。程序运行环境包括支持其运行的硬件、固件和软件。 被测对象模型:该模型是从测试的角度对程序的抽象。错误模型:该模型是对程序中的错误及其分类的抽象。

2 软件测试技术

软件测试技术分为：白盒测试和黑盒测试。白盒测试又称为结构测试技术，典型的路径测试技术。黑盒测试技术又称为功能测试技术，包括事务处理流程技术、定义域测试技术和状态测试技术。白盒测试技术依据是程序的逻辑结构，而黑盒测试技术依据的是软件行为的

描述。（单选题，填空题）

2.1 测试覆盖及其它们之间的基本关系（综合应用题）

(1)路径覆盖:执行所有可能穿过程序控制流程的路径。(2)语句覆盖:至少执行程序中所有语句一次。 (3)分支覆盖:至少将程序中的每一个分支执行一次。(4)条件覆盖:指每个判定中的所有可能的条件取值,至少执行一次。(5)条件组合覆盖:是指设计足够的测试用例,使每个判定中所有可能的条件取值组合至少执行一 次。

几种测试覆盖存在以下基本关系:语句覆盖≤分支覆盖≤条件组合覆盖≤路径覆盖·

2.2 事务流程图与控制流程图之间的主要差异

(1)基本模型元素所表达的语义不同； (2)性能增加,可使事务数目和单个事务处理流程具有相当的复杂性。(3)一个事务不等同于路径测试中的一条路径,可能在中间某处就完成了某一用户工作，终结了一个事务。(4)事务流程图中的分支和节点可能是一个复杂的过程。(5)事务流程图表达的系统模型更接近现实。(6)事务测试技术最大的问题和最大的代价是获得事务流程图以及用例设计。在实际软 件测试工作中,好的用例设计是发现软件错误的关键额会事务流测试步骤。

2.3 事务流测试步骤

第一步:获得事务流程图；第二步:浏览、复审；第三步:用例设计；第四步:测试执行

2.4 运用等价类划分技术进行测试

2.4.1 测试步骤

软件测试是一个有程序的过程，包括测试设计、测试执行以及测试结果比较。（填空题）

第一步:建立等价类表。第二步:为有效等价类设计测试用例。第三步:为无效等价类至少设计一个测试用例。

2.4.2 划分等价类方法(单选题，简答题)

1、如果某个输入条件规定了输入数据的取值范围，则可以确立一个有效等价类和两个无效等价类。

2、如果某个输入条件规定了输入数据的个数，则可以确立一个有效等价类和两个无效等价类。

3、如果某个输入条件规定了输入数据的一组可能取的值，每一个输入值就是一个有效等价类，一个无效等价类。

4、如果某个输入条件是一个布尔值，则可以划分一个有效等价类和一个无效等价类。

5、如果某个输入条件规定了必须符合的条件，则可以划分一个有效等价类和一个无效等价类。

6、若在已划分的某个等价类中各元素在程序中的处理方式不同，则应将此等价类进一步划分为更小的等价类。

2.5 边界值分析法

是一种根据 I/O 边界等价类上或紧靠边界的条件，选择测试用例的更有效的方法。

2.6 因果图法（简答题）

（1）通过对软件规格说明书的分析，找出一个模块的原因和结果，并给每个原因和结果赋予一个标识符；（2）分析原因与结果之间以及原因与原因之间对应的关系，并画出因果图；(3)在因果图上标识出一些特定的约束或限制条件；(4)把因果图转换成判定表；(5)为判定表的每一列设计测试用例。

3 软件测试步骤（填空题）

(1)单元测试:单元测试主要检验软件设计的最小单元—模块。（单选题）该测试以详细设计文档 为指导,测试模块内的重要控制路径。在单元测试中，由于模块不是一个独立的程序，必须为每个模块单元测试开发驱动模块和承接模块，后者代替被测模块的下属模块，打

印人口检查信息，并将控制返回到它的上级模块。（填空题）单元测试通常考虑模块的 4 个特征：①模块接口；②局部数据结构；③重要的执行路径；④错误执行路径。（单选题）

(2)集成测:是软件组装的一个系统化技术,其目标是发现与接口有关的错误，将经过单元测试的模块构成一个满足设计要求的软件结构。集成测试是一种自顶向下递增组装软件的方法。以主控模块作为测试驱动模块。（单选题）

(3)有效性测试:有效性测试的目标是发现软件实现的功能与需求规格说明书不一致的错误。

(4)系统测试:系统测试验证将软件溶于更大系统中时整个系统的有效性。

3.5 采用事务流测试技术进行软件测试的步骤大体分为四步

第一步:获得事务流程图；第二步:浏览、复审。这一步主要是对事务分类,其中应关注“并生”、“丝分裂”、“吸收”和 “结合”等事务,选取一个基本事务集作为系统功能测试的基础,为测试用例设计奠定基础。第三步:用例设计。设计足够的测试用例,实现基本事务覆盖。第四步:测试执行。

第 7 章 软件生存周期过程与管理

1 软件生存周期过程概述

1.1 过程分类

《ISO/IEC 软件生存周期过程 I2207—1995》标准按过程主体把软件生存周期过程分为：基本过程、支持过程、组织过程。（单选题）

软件基本过程指那些与软件生产直接相关的活动集，可分为获取过程、供应过程、开发过程、运行过程、维护过程。（填空题）

2 过程描述

2.1 软件实现过程、活动和任务

意图：把已规约的行为、接口和实现约束转换为一些动作，创建称为“软件项”的软件产品和服务作为系统元素。活动和任务：软件实现策略。结果：定义了实现策略，标识了有关设计方面的实现技术约束，实现了一个软件，按提供协议，把该软件打包成一个软件并存储。

2.2 软件需求分析过程、活动和任务

意图：建立系统软件部分的需求。活动和任务：任务 1：建立软件需求和文档。任务 2：评估软件需求，并建立相应的评估结果文档。任务 3：按软件复审过程进行软件需求复审。结果：（1）需求已分配给系统的软件元素；（2）已分析软件需求的正确性和可测性；（3）已了解软件需求对运行环境的影响；（4）在软件需求和系统需求之间建立了一致性和可跟踪性；（5）已定义了实现软件需求的优先级别；（6）软件需求已得到批准并按需求进行了调整；（7）针对软件需求的更改对成本、进度和技术影响，已进行了相应的评估。（8）建立了软件需求的基线，并与有关部门进行了沟通。

2.3 软件体系结构过程、活动和任务

意图：为软件的实现和按需求进行验证提供设计方案。 活动和任务： 软件体系结构设计。结果：开发一种软件体系结构设计，描述实现该软件需求的软件项，设计每一软件项的内部接口和外部接口

2.4 软件验证过程、活动和任务（简答题）

意图：证明软件产品是否满足了所规约的需求。活动和任务：过程实现、验证。结果：（1）开发并实现了验证策略（2）标识了验证准则（3）执行了所需要的验证活动（4）标识并记录了缺点（5）给出了可用于客户和其它参与人员的验证活动结果。

2.5 软件确认过程、活动和任务（简答题）

意图：证实所期望使用的软件产品是否满足需求。活动和任务：过程实现、确认。结果：

开发并实现了确认策略，标识了所有需要的软件工作产品的确认准则，执行了所需要的确认活动，标识并记录了发现的问题，提供了证据证明：软件产品能够按照用户所期望的方式来使用，给出了可用于客户和其他参与人员的验证活动结果

3 应用说明

3.1 剪裁过程及应用

(1)围绕一个组织,其中该组织在一个协议中使用了这一标准。(2)影响一个项目,其中要求该项目满足一个引用该标准的协议。(3)反映一个组织的需要,其中该组织要供给产品或服务。

4 软件生存生命周期

4.1 瀑布模型

瀑布模型规定了各开发阶段的活动：系统需求、软件需求、需求分析、设计、编码、、测试和运行。（填空题）

4.1.1 瀑布模型的原理

自上而下具有相互衔接的固定顺序。 每一阶段的输入，即工作对象以及本阶段的工作成果，作为输出传送到下一阶段。

4.1.2 瀑布模型的贡献

在决定系统怎样做之前存在一个需求阶段，它鼓励对系统做什么有一个规约。在系统构造之前有一个设计阶段，它鼓励规划系统结构。每一阶段都有评审，允许获取方和用户的参与前一步作为下一步被认可的、文档化的基线。

4.1.3 瀑布模型存在的问题

要求客户能够完整、正确和清晰地表达他们的需求，并要求开发人员一开始就理解这一应用。由于需求的不确定性，使设计、编码和测试阶段都可能发生延期，并且当项目接近结束时，出现了大量的集成和测试工作。

在开始的阶段中，很难评估真正的进度状态，并且直到项目结束之前都不能演示系统的功能。在一个项目的早期阶段，过分地强调了基线和里程碑处的文档，并可能需要花费更多的时间用于建立一些用处不大的文档。

4.2 增量模型

原理：增量模型融合了瀑布模型的基本成分（重复应用）和原型实现的迭代特征，该模型采用随着日程时间的进展而交错的线性序列，每一个线性序列产生软件的一个可发布的“增量”。前提：需求可结构化。 适用于“技术驱动”的软件产品开发。优点：第一个可交付版本所需的成本和时间较少由于很快发布第一个版本，可以减少用户需求的变更。 允许增量投资，即开始时只对一个或两个增量投资。 缺点：如果没有对用户的变更要求进行规划，那么产生的初始增量可能会造成后来增量的不稳定。 如果需求不像早起思考的那样稳定和完整，那么一些增量就可能需要重新开发，从新发布。 由于进度和配置的复杂性，可能会增大管理成本，超出组织的能力。

4.3 演化模型

原理：演化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。 演化模型主要针对事先不能完整定义需求的软件开发。 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：任何功能一经开发就能进入测试以便验证是否符合产品需求。 帮助导引出高质量的产品要求。 减少软件开发活动的盲目性。缺点：很容易弱化需求分析阶段的工作。

4.4 螺旋模型

原理：螺旋模型是在“瀑布模型”和演化模型的基础上，加入两者都忽略的风险分析所建立的一种软件开发模型。 螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的

风险有所了解，继而做出应有的反应 因此特别适用于庞大、复杂并具有高风险的系统。优点：螺旋模型很大程度上是一种“风险驱动”的方法体系。 螺旋模型关注解决问题的基本步骤。（单选题）

4.4.3 工作过程

（1）制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件；（2）风险分析：分析评估所选方案，考虑如何识别和消除风险；（3）实施工程：实施软件开发和验证；（4）客户评估：评价开发工作，提出修正建议，制定下一步计划。

4.5 喷泉模型（单选题）

4.5.1 喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于采用对象技术的软件开发项目。

4.5.2 喷泉模型认为：软件开发过程自下而上周期的各阶段是相互迭代和无间隙的特性。

5 过程规划与管理

对于一个项目而言，过程计划管理是项目管理计划的主体，一般还可能存在一些对支持生存周期过程具有重要作用的其他计划，包括软件工程管理计划、软件配置管理计划、软件质量保证计划、软件验证和确认计划和软件度量计划等。(填空题)

5.1 四个环节

过程规划（P）、过程检测（C）、过程执行（D）、过程调整（A）

5.2 过程建立

1、选择软件生存周期模型 2、细化所选择的生存周期模型 3、为每一个活动或任务标识合适的实例数目 4、确定活动的时序关系，并检查信息流 5、建立过程计划的文档

5.3 过程监控

软件生存周期过程的监控，软件生存周期过程改变所产生的影响的评估。

第 8 章 集成化能力成熟度模型 CMMI

1 背景和原理

1.1 CMMI 的含义

CapabilityMaturityModelIntegrationforDevelopment，集成化能力成熟度模型是由美国国防部与卡内基-梅隆大学和美国国防工业协会共同开发和研制的。

1.2 CMMI 的目的

是帮助软件企业对软件工程过程进行管理和改进，增强开发与改进能力，从而能按时地、不超预算地开发出高质量的软件。

2 CMMI 的模型部件

针对开发的 CMMI 是一个有关产品和服务的过程改善的成熟度模型，集成了 3 个源模型：软件 CMM、产品集成开发 CMM 和系统工程 CMM。（填空题）

CMMI 是一种过程改善框架。过程改善：是指人为设计的一个活动程序，其目的是改进组织的过程性能和成熟度，并改进这一程序的结果。 CMMI 的模型部件：由一些过程域组成，过程域有自己的确定专用目标和公共目标。每个专用目标和公共目标的实现，分别依赖一些实践。每个专用实践有自己的子实践和确定的典型工作产品。每个过程域还有意图陈述、简介性注释以及相关过程域。

2.3 过程域

一个业务域中一束相关的实践，当它们一起得以实现时，就满足被认为对该过程域的改善具有重要作用的一组条件。

3 CMMI 的等级

CMMI 模型提供了两种过程改善路径，一是称为能力等级的过程改善路径，二是称为成熟度等级的过程改善路径。（填空题）

3.1 能力等级：是一种过程改善路径，该路径可使组织针对单一过程域不断改善该过程域。

过程能力：遵循一个过程可达到的预期结果的程度。表征组织对一个过程域的改善，是不断改善一个给定的过程域的一种手段。（填空题）

3.1.2 能力等级（单选题）

包含一个共性目标及其相关的共性实践，它们与一个过程域相关联，能够改进组织同那个过程域相关联的过程。

能力等级 0：未完成级：过程不完整。

能力等级 1：已执行级：实现了过程域的特定目标。

能力等级 2：已管理级：

能力等级 3 :已定义级 :按照组织的裁减指南从组织的标准过程中裁减出来的一个已管理(能力等级 2) 过程。

能力等级 4：已量化管理级：使用统计和其他定量技巧控制的一个已定义（能力等级 3）过程。

能力等级 5：已持续优化级：经过改进的一个量化管理过程。

3.2 成熟度等级：是一种过程改善路径，该路径可使组织针对一组过程域不断改善一组相关的过程域。

3.3 组织成熟度等级（单选题）

成熟度等级是指达到预先定义的一组过程域所有目标的一种过程改善等级。一个成熟度等级是由预先定义的一个过程域集及其相关的一些专用实践和共用实践组成的。

CMMI 的阶段式表示模型定义了 5 个成熟度等级，在持续的过程改进上，每一等级都是构成下一阶段基础的一个层次，这些等级用从 1 到 5 的数字表示。

成熟度等级 1：初始级过程是混乱的，应付式的。

成熟度等级 2：已管理能确保过程按照预定方针得到计划和执行

成熟度等级 3：已定义过程得到了很好地描述和理解，并应用标准、规程、工具及方法来表现。

成熟度等级 4：量化管理组织和项目为质量和过程绩效建立了量化目标并将其用作管理过程的标准。

成熟度等级 5：持续优化点关注通过渐进性和革新性过程改进和技术改进来持续地改进过程的绩效。

3.4 成熟度等级与能力等级的关系

（1）为了达到成熟度 2 级，2 级所包含的所有过程域必须达到能力等级 2 或更高级。

（2）为了达到成熟度 3 级，2 级、3 级所包含的所有过程域必须达到能力等级 3 或更高级。

（3）为了达到成熟度 4 级，2、3、4 级所包含的所有过程域必须达到能力等级 3 或更高级。

（4）为了达到成熟度 5 级，所有过程域必须达到能力等级 4 或更高级。

CMMI 模型基于过程途径思想，通过过程把软件质量的 3 个支撑点：受训的人员、规程和方法、工具和设备进行集成，以开发所期望的系统 / 产品。(填空题)