第 8 章系统分析与设计方法

8.2 需求分析与软件设计

8.2.1 需求分析的任务与过程

1. 【2009 年题 20 解析】

需求管理是一个对系统需求变更、了解和控制的过程。需求管理过程与需求开发过程相互关联，当初始需求导出的同时就启动了需求管理计划，一旦形成了需求文档的初稿，需求管理活动就开始了。

关于需求管理过程域内的原则和策略，可以参考：

① 需求管理的关键过程领域不涉及收集和分析项目需求，而是假定已收集了软件需求，或者已由更高一级的系统给定了需求。

② 开发人员在向客户以及有关部门承诺某些需求之前，应该确认需求和约束条件、风险、偶然因素、假定条件等。

③ 关键处理领域同样建议通过版本控制和变更控制来管理需求文档。

2. 【2011 年题 19 解析】

需求定义方法包括严格定义方法和原型方法两种。严格定义方法适用于需求已全面获取，需求较为明确的情况。如果达不到这个要求，则适宜用原型方法。

【答案】C。

3. 【2011 年题 20 解析】

需求管理是一种用于查找、记录、组织和跟踪系统需求变更的系统化方法。而非对需求开发的管理。需求开发包括：需求获取、需求分析、需求定义和需求验证，而非需求管理。需求的跟踪属于需求管理的范畴。

C 选项是程序实现过程。

【答案】D。

4. 【2012 年题 24 解析】

在初步的业务需求描述已经形成的前提下，基于 UML 的需求分析过程大致可分为以下步骤：

(1)、**利用用例及用例图表示需求**。从业务需求描述出发获取执行者和场景；对场景进行汇总、分类、抽象，形成用例；确定执行者与用例、用例与用例图之间的关系，生成用例图。

(2)、**利用包图和类图表示目标软件系统的总体框架结构**。根据领域知识、业务需求描述和既往经验设计目标软件系统的顶层架构；从业务需求描述中提取“关键概念”，形成领域概念模型；从概念模型和用例出发，研究系统中主要的类之间的关系，生成类图。

5. 【2014 年题 16 解析】

作为一份正式文档，系统建议方案至少应该包含以下内容：

①前置部分。包括标题、目录和摘要。摘要部分以 1～2 页的篇幅总结整个系统建议方案报告，提供系统方案中的重要事件、地点、任务和原因，以及系统方案是如何实现的等信息。

②系统概述。包括系统建议方案报告的目的、对**问题的陈述**、**项目范围**和报告内容的叙述性解释。

③系统研究方法。简要地解释系统建议方案报告中包含的信息是如何得到的，研究工作是如何进行的。

④**候选系统方案及其可行性分析**。系统阐述每个候选系统方案，并对每个方案进行可行

性评价。

⑤建议方案。在对各个候选系统方案进行可行性评价之后，通常会推荐一个解决方案，并且要给出推荐该解决方案的理由。

⑥结论。简要地描述摘要的内容，再次指出系统开发的目标和所建议的系统方案。同时，需要再次强调项目的必要性和可行性，以及系统建议方案报告的价值。

⑦附录。系统分析师认为阅读者可能会感兴趣的所有信息，但这些信息对于理解系统建议方案报告的内容来说不是必要的。

【答案】D。

8.2.2 如何进行系统设计

1. 【2018 年题 9 解析】

软件设计层面的功耗控制主要可以从以下方面展开：

① 软硬件协同设计，即软件的设计要与硬件的匹配，考虑硬件因素。
② 编译优化，采用低功耗优化的编译技术。
③ 减少系统的持续运行时间，可从算法角度进行优化。
④ 用“中断”代替“查询”
⑤ 进行电源的有效管理

答案 D。

2. 【2018 年题 23 解析】

答案：C。

3. 【2018 年题 24 解析】

无论采用哪种设计技术，高质量的**数据设计**将改善程序结构和模块划分，降低过程复杂性。

软件结构设计的主要目标是开发一个模块化的程序结构,并表示出模块间的控制关系。

人机界面设计描述了软件与用户之间的交互关系。

答案：BAD。

8.3 结构化分析与设计

8.3.2 结构化设计

1. 【2012 年题 22 解析】

程序流程图(Program How Diagram，PFD)，N-S 图与 PFD 类似，IPO 图是由 IBM 公司发起并逐步完善的一种流程描述工具。

用于描述系统中每个模块的输入，输出和数据加工的图是 IPO 图，而非程序流程图。

答案：A。

4. 【2015 年题 24 解析】

结构化程序设计的三种基本控制结构就是：顺序、分支和循环。

8.4 面向对象的分析与设计

8.4.1 面向对象的基本概念

1. 【2011 年题 22 解析】

面向对象技术中的类分为三种：实体类、边界类、控制类。

实体类是用于对必须存储的信息和相关行为建模的类。实体对象(实体类的实例)用于保

存和更新一些现象的有关信息，例如：事件、人员或者一些现实生活中的对象。实体类通常都是永久性的，它们所具有的属性和关系是长期需要的，有时甚至在系统的整个生存期都需要。

边界类是一种用于对系统外部环境与其内部运作之间的交互进行建模的类。这种交互包括转换事件，并记录系统表示方式(例如接口)中的变更。

常见的边界类有窗口、通信协议、打印机接口、传感器和终端。如果您在使用 GUI 生成器，您就不必将按钮之类的常规接口部件作为单独的边界类来建模。通常，整个窗口就是最精制的边界类对象。边界类还有助于获取那些可能不面向任何对象的 API(例如遗留代码)的接口。

控制类用于对一个或几个用例所特有的控制行为进行建模。控制对象(控制类的实例)通常控制其他对象，因此它们的行为具有协调性质。控制类将用例的特有行为进行封装。

2. 【2013 年题 22 解析】

【答案】B、C。

实体类是用于对必须存储的信息和相关行为建模的类。实体对象(实体类的实例)用于保存和更新一些现象的有关信息，例如：事件、人员或者一些现实生活中的对象。实体类通常都是永久性的，它们所具有的属性和关系是长期需要的，有时甚至在系统的整个生存期都需要。

边界类是一种用于对系统外部环境与其内部运作之间的交互进行建模的类。这种交互包括转换事件，并记录系统表示方式(例如接口)中的变更。

常见的边界类有窗口、通信协议、打印机接口、传感器和终端。如果您在使用 GUI 生成器，您就不必将按钮之类的常规接口部件作为单独的边界类来建模。通常，整个窗口就是最精制的边界类对象。边界类还有助于获取那些可能不面向任何对象的 API(例如遗留代码)的接口。

控制类用于对一个或几个用例所特有的控制行为进行建模。控制对象(控制类的实例)通常控制其他对象，因此它们的行为具有协调性质。控制类将用例的特有行为进行封装。

【答案】C、B。

3. 【2017 年题 25 解析】

面向对象的分析模型主要由**顶层架构图、用例与用例图、领域概念模型**构成；设计模型则包含以包图表示的软件体系结构图、以交互图表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程化处理过程的活动图等。

8.4.3 统一建模语言

1. 【2009 年题 25 解析】

用例是在系统中执行的一系列动作，这些动作将生成特定参与者可见的价值结果。它确定了一个和系统参与者进行交互，并可由系统执行的动作序列。用例模型描述的是外部执行者(Actor)所理解的系统功能。用例模型用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结果，表明了开发者和用户对需求规格达成的共识。

两个用例之间的关系主要有两种情况：一种是用于重用的包含关系，用构造型 include 表示；另一种是用于分离出不同行为的扩展，用构造型 extend 表示。

包含关系：当可以从两个或两个以上的原始用例中提取公共行为，或者发现能够使用一个构件来实现某一个用例的部分功能是很重要的事时，应该使用包含关系来表示它们。

扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种事情，可以断定将这个用例分为一个主用例和一个或多个辅用例描述可能更加清晰。

【答案】：A。

2. 【2009 年题 26 解析】

面向对象的设计模型包含以包图表示的软件体系结构图，以交互图表示的用例实现图，完整精确的类图，针对复杂对象的状态图和用以描述流程化处理的活动图等。

3. 【2010 年题 24 解析】

在 RUP 中采用“4+1”视图模型来描述软件系统的体系结构。“4+1”视图包括逻辑视图、实现视图、进程视图、部署视图和用例视图。

分析人员和测试人员关心的是系统的行为，因此会侧重于用例视图；

最终用户关心的是系统的功能，因此会侧重于逻辑视图；

程序员关心的是系统的配置、装配等问题，因此会侧重于实现视图；

系统集成人员关心的是系统的性能、可伸缩性、吞吐率等问题，因此会侧重于进程视图；

系统工程师关心的是系统的发布、安装、拓扑结构等问题，因此会侧重于部署视图。

4. 【2014 年题 24 解析】

UML 对系统架构的定义是系统的组织结构，包括系统分解的组成部分，以及它们的关联性、交互机制和指导原则等提供系统设计的信息。具体来说，就是指以下 5 个系统视图：

(1)逻辑视图(设计视图)。逻辑视图也称为设计视图，它表示了设计模型中在架构方面具有重要意义的部分，即类、子系统、包和用例实现的子集。

(2)进程视图。进程视图是可执行线程和进程作为活动类的建模，它是逻辑视图的一次执行实例，描述了并发与同步结构。

(3)实现视图。实现视图对组成基于系统的物理代码的文件和构件进行建模。

(4)部署视图。部署视图把构件部署到一组物理节点上，表示软件到硬件的映射和分布结构。

(5)用例视图。用例视图是最基本的需求分析模型。

5. 【2014 年题 31 解析】

“4+1”视图模型从五个不同的视角来描述软件架构，每个视图只关心系统的一个侧面，五个视图结合在一起才能反映软件架构的全部内容。

(1)逻辑视图。逻辑视图主要支持系统的功能需求，即系统提供给最终用户的服务。

(2)开发视图。开发视图也称为模块视图，在 UML 中被称为实现视图，它主要侧重于软件模块的组织和管理。开发视图要考虑软件内部的需求。

(3)进程视图。进程视图侧重于系统的运行特性，主要关注一些非功能性需求，例如，系统的性能和可用性等。进程视图强调并发性、分布性、系统集成性和容错能力。

(4)物理视图。物理视图在 UML 中被称为部署视图，它主要考虑如何把软件映射到硬件上，它通常要考虑到解决系统拓扑结构、系统安装和通信等问题。

(5)场景。场景可以看作是那些重要系统活动的抽象，它使四个视图有机联系起来，从某种意义上说场景是最重要的需求抽象。场景视图对应 UML 中的用例视图。

下面是题目选项中几种 UML 图的解释，从中可以了解题目所描述的，是哪一种 UML 图。

(1)对象图(object diagram)。对象图描述一组对象及它们之间的关系。对象图描述了在类图中所建立的事物实例的静态快照。和类图一样，这些图给出系统的静态设计视图或静态进程视图，但它们是从真实案例或原型案例的角度建立的。

(2)活动图(activity diagram)。活动图将进程或其他计算结构展示为计算内部一步步的控制流和数据流。活动图专注于系统的动态视图。它对系统的功能建模和业务流程建模特别重要，并强调对象间的控制流程。

(3)状态图(state diagram)。状态图描述一个状态机，它由状态、转移、事件和活动组成。状态图给出了对象的动态视图。它对于接口、类或协作的行为建模尤为重要，而且它强调事件导致的对象行为，这非常有助于对反应式系统建模。

(4)类图(class diagram)。类图描述一组类、接口、协作和它们之间的关系。在 OO 系统的建模中，最常见的图就是类图。类图给出了系统的静态设计视图，活动类的类图给出了系

统的静态进程视图。

4+1”视图

【答案】A、D、B。

6. 【2015 年题 23 解析】

包含：当可以从两个或两个以上的用例中提取公共行为时，应该使用包含的关系来表示它们。

扩展：如果一个用例明显地混合了两种或者两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样可能会使描述更加清晰。这种情况下才是扩展关系。比如导出数据模块，有导出 excel，导出 word 等，这些导出与模块之间是扩展。

泛化：当多个用例共同拥有一种类似的结构和行为时，可以将他们的共性抽象成为父用例泛化关系是从另一个角度来看的继承关系，也就是说，当两个用例之间可能存在父子关系时，可判定为泛化关系。

在本题中，“电话注册”与“邮件注册”都属于“会员注册”，他们是“会员注册”的具体形式，所以存在父子关系，可判定为泛化关系。

【答案】：C。

7. 【2018 年题 31 解析】

4+1 视图即：逻辑视图、开发视图、物理视图（部署视图）、进程视图、场景。答案 A。

8.4.4 其他

1. 【2011 年题 23 解析】

面向对象设计原则包括：

单一职责原则：设计目的单一的类

开放-封闭原则：对扩展开放，对修改封闭

李氏(Liskov)替换原则：子类可以替换父类

依赖倒置原则：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程

接口隔离原则：使用多个专门的接口比使用单一的总接口要好

组合重用原则：要尽量使用组合，而不是继承关系达到重用目的

迪米特(Demeter)原则(最少知识法则)：一个对象应当对其他对象有尽可能少的了解

迪米特法则的应用准则：

1) 在类的划分上，应当创建有弱耦合的类。类之间的耦合越弱，就越有利于复用。

2) 在类的结构设计上，每一个类都应当尽量降低成员的访问权限。一个类不应当 public 自己的属性，而应当提供取值和赋值的方法让外界间接访问自己的属性。

3) 在类的设计上，只要有可能，一个类应当设计成不变类。

4) 在对其它对象的引用上，一个类对其它对象的引用应该降到最低。

其中迪米特原则的主要理念是让一个对象尽可能少的了解其它对象，这样，就能尽可能少的产生违规操作，让设计出来的系统更稳定。在本题中，C 选项提到“尽可能提高对其属性和方法的访问权限”违背了迪米特原则。

【答案】C。

2. 【2012 年题 20 解析】

里氏替换原则是面向对象设计原则之一，由 Barbara liskov 提出，其基本思想是，一个软件实体如果使用的是一个基类对象，那么一定适用于其子类对象，而且觉察不出基类对象和子类对象的区别，即把**基类都替换成它的子类**，程序的行为没有变化。反过来则不一定成立，如果一个软件实体使用的是一个子类对象，那么它不一定适用于基类对象。在运用里氏替换原则时，**尽量将一些需要扩展的类或者存在变化的类设计为抽象类或者接口，并将其作为基类，在程序中尽量使用基类对象进行编程**。由于子类继承基类并实现其中的方法，程序

运行时，子类对象可以替换基类对象，如果需要对类的行为进行修改，可以扩展基类，增加新的于类，而无需修改调用该基类对象的代码。

【答案】A。

3. 【2015 年题 26 解析】

单一职责原则：设计目的单一的类。
开放-封闭原则：对扩展开放，对修改封闭。
李氏(Liskov)替换原则：子类可以替换父类。
依赖倒置原则：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程。
接口隔离原则：使用多个专门的接口比使用单一的总接口要好。
组合重用原则：要尽量使用组合，而不是继承关系达到重用目的。
迪米特(Demeter)原则(最少知识法则)：一个对象应当对其他对象有尽可能少的了解。

4. 【2016 年题 21 解析】

本题考查的是教程“4.4.2 面向对象的分析设计”的内容。

面向对象的分析模型主要由顶层架构图、用例与用例图、领域概念模型构成。

设计模型则包含以包图表示的软件体系结构图、以交互图表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程化处理过程的活动图等。

8.5 用户界面设计

8.5.1 用户界面设计的原则

1. 【2009 年题 32 解析】

本题考查应用系统输入设计的基本知识。

人的因素在系统输入设计中扮演了很重要的角色。输入应该尽可能地简单，以降低错误发生的可能性，如对于范围可控的数据，**使用选择的方式替代用户输入**；**只输入变化的数据等**。输入应该尽可能使用已有含义明确的设计，**需要采用模仿的方式而非创新**。为了避免用户理解的二义性，**应该对表格中输入的数据给出提示信息**。

【答案】B。

2. 【2014 年题 20 解析】

用户界面设计的 3 条黄金规则为：
让用户拥有控制权；
减少用户的记忆负担；
保持界面一致。

【答案】A。

8.5.3 其他

1. 【2010 年题 35 解析】

系统输入设计中，通常通过内部控制的方式验证输入数据的有效性。数据类型检查确保输入了正确的数据类型；自检位用于对主关键字进行基于校验位的检查；**域检查用于验证数据是否位于合法的取值范围**；格式检查按照已知的数据格式对照检查输入数据的格式。

8.10 其他

1. 【2014 年题 17 解析】

JRP **是一种相对来说成本较高的需求获取方法**(而非需求分析与验证的方法)，**但也是十分有效的一种**。它通过联合各个关键用户代表、系统分析师、开发团队代表一起，通过有组织的会议来讨论需求。通常该会议的参与人数为 6～18 人，召开时间为 1～5 小时。

JRP 的主要意图是收集需求，而不是对需求进行分析和验证。实施 JRP 时应把握以下主要原则：

(1)在 JRP 实施之前，**应制订详细的议程，并严格遵照议程进行**。

(2)按照既定的时间安排进行。

(3)尽量完整地记录会议期间的内容。

(4)**在讨论期间尽量避免使用专业术语**。

(5)充分运用解决冲突的技能。

(6)会议期间应设置充分的间歇时间。

(7)鼓励团队取得一致意见。