**软件体系结构2018年1月期末考试资料**

|  |  |
| --- | --- |
|  |  |

**填空题**

1. 软件重用是指在两次或者多次不同的软件开发过程中重复使用相同或相近软件元素的过程。P4
2. 要提高软件开发效率，提高软件产品质量，必须采用工程化的开发方法与工业化的生产技术。技术上，采用基于重用的软件生产技术，在管理上采用多维的工程管理模式。P4
3. 构件是核心和基础，重用是必需的手段。
4. 构件是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统；结构上，它是语义描述、通信接口和实现代码的复合体。简单地说，构件是具有一定的功能，能够独立工作或能同其他构件装配起来协调工作的程序体，构件的使用同它的开发、生产无关。P4
5. 构件模型是对构件本质特征的抽象描述。有参考模型(3C模型)、描述模型(RESOLVE模型和REBOOT模型)、实现模型。P4

——流派：OMG(对象管理组织)的CORBA(通用对象请求代理结构)、Sun的EJB、Microsoft的DCOM(分布式构件对象模型)。P4

——构件分类方法：(1)关键字分类法(2)刻面分类法(3)超文本组织方法。P7

——构件组装技术：(1)基于功能的组装技术(2)基于数据的组装技术(3)面向对象的组装技术P11

——重用：面向对象的组装技术中：面向对象的软件开发方法以类库中的基类为父类采用构造法、子类法生成子类。

1. 体系结构的核心模型由5种组成元素构成：构件、连接件、配置、端口和角色；构件、连接件和配置是最基本的元素。P36
2. P51-P54

3.1.1管道过滤器

3.1.4层次系统

——层次系统最广泛的应用是分层通信协议

3.1.5仓库系统及知识库

1. C/S体系结构风格P56 3-4自然段
2. 软件体系结构的风格：

三层C/S体系结构将应用功能分成表示层、功能层和数据层三个部分。P59

——P59-61 3.3.1各层的功能

1. 三层C/S体系结构中，中间件是最重要的构件。P61
2. 浏览/服务器风格(B/S)，基本结构为：浏览器/Web服务器/数据库服务器 P66
3. CORBA技术规范主要内容包括接口定义语言(IDL)、接口池(IR)、动态调用接口(DII)、对象适配器(OA)。P68 3.5.1
4. 构建模型：为满足体系结构设计的需要，HMB风格的构建模型包括接口、静态结构和动态行为三部分。P78
5. P78 3.7.1层次消息总线（HMB）的构件模型

P80消息过滤

1. SOA是一种在计算环境中设计、开发、部署和管理离散逻辑单元(服务)模型的方法。是面向对象模型的一种代替。P197
2. SOA是一种粗粒度、松耦合的服务体系结构，服务之间通过简单、精确定义接口进行通信，不涉及底层编程接口和通信模型。

——SOA模型具有的特征：(1)松散耦合(2)粗粒度服务(3)标准化接口 P198

——SOA设计原则：(1)明确定义的接口(2)自包含和模块化(3)粗粒度(4)松耦合(5)互操作性、兼容和策略声明 P199

——SOA的三个主要抽象级别：操作、服务和业务流程 P199 8.2

1. 基础设计层、体系结构层、业务层 P200
2. SOA服务栈 ：发现服务层、描述服务层、消息格式层、编码格式层、传输协议层 P201
3. SOA的实现方法很多，其中主流方式包括：Web Service、企业服务总线和服务注册表 P202 8.4
4. SOA最重要的也是最成熟的三个协议：SOAP、WSDL、UDDI P205
5. Web Service模型中的操作包括发布、查找和绑定 P203
6. 9.1.2丰富的含义

RIA是Web开发和部署模式的一种演变。“富”互联网应用(RIA)体系结构中“富”的含义为：丰富的数据模型和丰富的用户界面 9.1.2 P234

1. RIA开发技术：Macromedia Flash/Flex、AJAX、Laszlo、Avalon、Java SWT、XUL、Bindows、Oracle Forms P237
2. AJAX是由几种蓬勃发展的技术以新的方式组合而成的，包含基于XHTML(可扩展超文本标识语言)和CSS标准的标识；使用DOM进行动态显示和交互；使用XMLHttpRequest与服务器进行异步通信；使用JavaScript绑定一切。 P239
3. Mashup技术 第二段 P240
4. Mashup从体系结构组成：API/内容提供者、Mashup站点、客户机的Web浏览器。 P240
5. MVC（模型-视图-控制器）

对于界面设计可变性的需求，MVC把交互系统的组成分解成模型、视图、控制三种构件…… P284

1. 一个模式有模式名称、问题、解决方案和效果四个基本成分。 P286
2. 设计模式，是一套被反复使用、多数人知晓、经过分类编目、代码设计经验的总结，使用设计模式的目的是为了提高代码的可重用性，让代码更容易被人理解，并保证代码的可靠性。

——设计模式的层次：面向对象模式、代码模式、框架应用模式、形式合约等 P289

——设计模式分类：

根据目的和用途分为创建型模式、结构型模式和行为型模式

根据处理范围不同，设计模式可分为类模式和对象模式。 P291

——23种设计模式，例如：适配器模式、建造者模式

1. 中间件采用自底向上的方式来划分，可分为底层中间件、通用型中间件和集成型中间件三大层次。底层中间件的主流技术有JVM(Java虚拟机)、CLR(公共语言运行库)、ACE(自适配通信环境);通用型中间件主流技术有RPC、ORB、MOM(面向消息的中间件)；集成型中间件主流技术有WorkFlow、EAI。 P301
2. RMI（远程方法调用）是Java的一组用于开发分布式应用程序的API。使用Java语言接口定义远程对象，集合了Java序列化和Java远程方法协议。目前使用Java远程消息交换协议(JRMP)进行通信。 P304
3. 基于体系结构的设计方法（ABSD方法）的三个基础：

（1）功能的分解

（2）通过选择体系结构风格来实现质量和业务需求

（3）软件模板的使用 P308

**名词解释：**

1. 软件重用：是指在两次或者多次不同的软件开发过程中重复使用相同或相近软件元素的过程。软件元素包括程序代码、测试用例、设计文档、设计过程、需求分析文档甚至领域知识。
2. 构件是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统；结构上，它是语义描述、通信接口和实现代码的复合体。简单地说，构件是具有一定的功能，能够独立工作或能同其他构件装配起来协调工作的程序体，构件的使用同它的开发、生产无关。
3. 消息总线：HMB风格的消息总线是系统的连接件，构件向消息总线登记感兴趣的消息，形成构件消息响应登记表。消息总线根据接收到的消息类型和构件-消息响应登记表的信息，定位并传递该消息给相应的响应者，并负责返回处理结果。必要时，消息总线还对特定的消息进行过滤和阻塞。
4. “内外有别”模型：在C/S与B/S混合软件体系结构的“内外有别”模型中，企业内部用户通过局域网直接访问数据库服务器，软件系统采用C/S体系结构；企业外部用户通过Internet访问Web服务器，通过Web服务器在访问数据库服务器，软件系统采用B/S体系结构。
5. “查改有别”模型：在C/S与B/S混合软件体系结构的“查改有别”模型中，不管用户是通过什么方式(局域网或Internet)连接到系统，凡是需执行维护和修改数据操作的，就使用C/S体系结构；如果只是执行一般的查询和浏览操作，则使用B/S体系结构。
6. SOAP是一个基于XML的，在松散分布式环境中交换结构化信息的轻量级协议，它为在一个松散的，分布式环境中使用XML交换结构化的和类型化的信息提供了一种简单的机制。
7. 设计模式：是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结，使用设计模式的目的是为了提高代码的可重用性，让代码更容易被他人理解，并保证代码可靠性。
8. 单例模式是创建型模型的一种，确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类成为单例类。它提供全局访问的方法。
9. 代理模式是一种对象结构型模式，可为某个对象提供一个代理，并由代理对象控制对原对象的引用。
10. 享元模式是一种对象结构型模式，通过运用共享技术，有效地支持大量细粒度的对象。
11. 观察者模式又称发布-订阅模式、模型-视图模式、源-监听器模式或从属者模式，是一种对象的行为型模式。它定义了对象之间的一种一对多的依赖关系，使得每当一个对象状态发生改变时，其相关依赖对象都得到通知并被自动更新。
12. 中间件是一种独立的[系统](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F" \t "_blank)软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源中间件位于客户机/ 服务器的操作系统之上，管理计算机资源和网络通讯，实现应用之间的互操作。

**简答题**

1. 管道-过滤器体系结构风格 具体介绍P51 3.1.1

在管道与过滤器风格的软件体系结构中，每一个构件都有一组输入与输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这里的过滤器必须是独立的实体，它不能与其他的过滤器共享数据，而且一个过滤器不知道它上游和下游的标识。

管道与过滤器风格的软件体系结构特点（优点）：

1. 使得软构件具有良好的隐蔽性和高内聚、低耦合的特点。
2. 允许设计师将整个系统的输入/输出行为看成是多个过滤器的行为的简单合成。
3. 支持软件重用。只要提供适合在两个过滤器之间传送的数据，任何两个过滤器都可连接起来。
4. 系统维护和增强系统性能简单。新的过滤器可以添加到现有的系统中来；旧的可以被改进的过滤器替换掉。
5. 允许对一些如吞吐量、死锁等属性的分析。
6. 支持并行执行。每个过滤器是作为一个单独的任务完成，因此可与其他任务并行执行。

管道与过滤器风格的软件体系结构的不利因素：

1. 通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但他们是独立的，所以设计师必须将每个过滤器看成一个完成的从输入到输出的转换。
2. 不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为重要。
3. 因为在数据传输上没有通用的标准，每个过滤器都增加了解析和合成数据的工作，这样导致了系统性能下降，并增加了编写过滤器的复杂性。
4. 基于事件的隐式调用系统 具体介绍P52 3.1.3
5. 风格思想：构件不直接调用一个过程，而是触发或广播一个或多个事件。
6. 从体系结构上说，这种风格的构件是一些模块，这些模块既可以是一些过程，又可以使一些事件的集合。过程可以用通用的方式调用，也可以在系统事件中注册一些过程，当发生这些事件时，过程被调用。
7. 基于事件的隐式调用风格主要特点：事件的触发者并不知道哪些构件会被这些事件影响。

隐式调用系统的优点：

1. 为软件重用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中。
2. 为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其他构件的接口。

隐式调用系统的缺点：

1. 构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其他构件是否会响应它。而且即使它知道事件注册了哪些构件的过程，它也不能保证这些过程被调用的顺序。
2. 数据交换的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠一个共享的仓库进行交互。在这些情况下，全局性能和资源管理便成了问题。
3. 既然过程的语义必须依赖于被触发事件的上下文约束，关于正确性的推理存在问题。
4. 层次系统组织 具体介绍P53 3.1.4

层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。

这种风格支持基于可增加抽象层的设计。这样允许将一个复杂问题分解成一个增量步骤序列的实现。支持功能增强，由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。

不足之处：

1. 并不是每个系统都可以很容易地划分为分层的模式，甚至即使一个系统的逻辑结构是层次化的，出于对系统性能的考虑，系统设计师不得不把一些低级或高级的功能综合起来。
2. 很难找到一个合适的、正确的层次抽象方法。
3. 三层C/S体系结构 具体介绍p58 3.3

C/S体系结构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受。

三层C/S体系结构将应用功能分成表示层、功能层和数据层。在三层C/S体系结构中，中间件是最重要的构件。

三层C/S体系结构的优点：

1. 允许合理划分三层结构功能，使之在逻辑上保持相对独立性，提供系统可维护性和可扩展性。
2. 允许更灵活有效地选用相应的平台和硬件系统
3. 并行开发，各层也可以选择各自最适合的开发语言。
4. 允许充分利用功能层有效隔离表示层和数据层
5. MVC设计模式 具体介绍P297 12.1.8

——常用的MVC设计模式中，包含哪三个抽象类及其含义？

MVC模式包括三个抽象类，View抽象类，Controller抽象类，Model抽象类

1. View抽象类，它从模型获得显示信息，并以特定的形式展示给用户，对于相同的信息可以有多个不同的显示形式或视图。
2. Controller抽象类，处理用户与软件的交互操作，其职责是决定软件的控制流程，确保用户界面与模型间的对应联系，它接收用户的输入，将输入反馈给模型，进而实现对模型的计算控制，再根据用户的需求，创建合适的视图返回到用户界面，它是使模型和视图协调工作的部件。
3. Model抽象类，它向视图和控制器提供业务逻辑服务。

三个抽象类之间的关系：

控制器把接收到的请求或者数据传送到模型去处理，再根据用户的要求，创建一个合适的视图，该视图从模型中读取处理后的结果，以特定的形式显示出来。

1. Web服务模型 具体介绍p202-203

在Web服务的解决方案中，一共有三种工作角色，其中服务提供者和服务请求者是必需的，服务注册中心是一个可选的角色。它们之间的交互和操作构成了SOA的一种可实现体系结构。

在Web服务模型中的操作包括发布、查找和绑定，这些操作可以单次或反复出现。

1. SCA服务构件的特点 具体介绍p198-199

服务构件体系结构（SCA）是基于SOA的思想描述服务之间组合和协作的规范，它描述用于使用SOA构建应用程序和系统的模型。

1. 提供了构建粗粒度构件的机制
2. 将传统中间件编程从业务逻辑分离出来，从而使程序员免受其复杂性困扰
3. 允许开发人员集中精神编写业务逻辑，而不必将大量的时间花费在更为底层的技术实现上。
4. Web Service作为SOA的实现技术时，应用系统的6个层次 具体介绍P203
5. 底层传输层(2)服务通信协议层

(3)服务描述层(4)服务层(5)业务流程层(6)服务注册层

**综合题**

重点：工厂方法模式、建造者模式、适配器模式、装饰模式

——给定应用场景，模式？-定义？-结构图？-原理？

创建型模式

（1）工厂方法（Factory method）模式：又称为“虚拟构造器”模式或多态模式，属于类的创建型模式。在工厂方法模式中，父类负责定义创建对象的公共接口，而子类则负责生成具体的对象，这样做的目的是将类的实例化操作延迟到子类中完成，即由子类来决定究竟应该实例化（创建）哪一个类。

（2）建造者（builder）模式：建造者模式强调将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。建造者模式是一步一步地创建一个复杂的对象，它允许用户只通过指定复杂对象的类型和内容就可以构建他们，用户不需要知道内部的具体构建细节。建造者模式属于对象创建型模式。

结构型模式

1. 适配器（Adapter）模式：适配器模式将一个接口转换成客户希望的另一个接口，从而使接口不兼容的那些类可以一起工作。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。在类适配器模式中，通过使用一个具体类将适配者适配到目标接口中；在对象适配器模式中，一个适配器可以将多个不同的适配者适配到同一个目标。
2. 装饰（decorator）模式：这是一种对象结构型模式，可动态地给一个对象增加一些额外的职责，就增加对象功能来说，装饰模式比生成子类更为灵活。