《高级软件体系结构》习题集

第一章

1、根据自己的经验，谈谈对软件危机的看法。

答：软件危机的表现主要分为四个方面。软件成本日益增长，开发进度难以控制，软件质量差，软件维护困难。

**开发进度难以控制**

由于软件是逻辑、智力产品，软件的开发需建立庞大的逻辑体系，这是与其他产品的生产不一样的。

在软件开发过程中，用户需求变化等各种意想不到的情况层出不穷，令软件开发过程很难保证按预定的计划实现，给项目计划和论证工作带来了很大的困难。

盲目增加软件开发人员并不能成比例地提高软件开发能力。相反，随着人员数量的增加，人员的组织、协调、通信、培训和管理等方面的问题将更为严重。

**软件质量差**

软件项目即使能按预定日期完成，结果却不尽人意。

在“软件作坊”里，由于缺乏工程化思想的指导，程序员几乎总是习惯性地以自己的想法去代替用户对软件的需求，软件设计带有随意性，很多功能只是程序员的“一厢情愿”而已，这是造成软件不能令人满意的重要因素。

**软件维护困难**

由于在软件设计和开发过程中，没有严格遵循软件开发标准，各种随意性很大，没有完整的真实反映系统状况的记录文档，给软件维护造成了巨大的困难。

在软件使用过程中，原来的开发人员可能因各种原因已经离开原来的开发组织，使得软件几乎不可维护。

工业界为维护软件支付的费用占全部硬件和软件费用的40%-75%。

造成软件危机的原因主要包括：

**缺乏正确的理论指导**

缺乏有力的方法学和工具方面的支持。由于软件不同于大多数其他工业产品，其开发过程是复杂的逻辑思维过程，其产品极大程度地依赖于开发人员高度的智力投入。由于过分地依靠程序设计人员在软件开发过程中的技巧和创造性，加剧软件产品的个性化，也是发生软件危机的一个重要原因。

**软件规模越来越大**

随着软件应用范围的增广，软件规模愈来愈大。大型软件项目需要组织一定的人力共同完成，而多数管理人员缺乏开发大型软件系统的经验，而多数软件开发人员又缺乏管理方面的经验。各类人员的信息交流不及时、不准确、有时还会产生误解。

软件项目开发人员不能有效地、独立自主地处理大型软件的全部关系和各个分支，因此容易产生疏漏和错误。

**软件复杂度越来越高**

软件不仅仅是在规模上快速地发展扩大，而且其复杂性也急剧地增加。软件产品的特殊性和人类智力的局限性，导致人们无力处理“复杂问题”。

所谓“复杂问题”的概念是相对的，一旦人们采用先进的组织形式、开发方法和工具提高了软件开发效率和能力，新的、更大的、更复杂的问题又摆在人们的面前。

克服软件危机的方法主要包括：

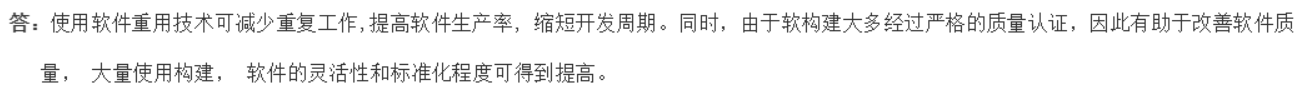
人们面临的不光是技术问题，更重要的是管理问题。管理不善必然导致失败 。

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。

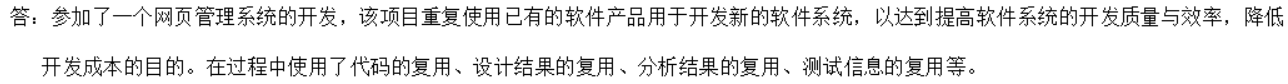
在技术上，应该采用基于重用的软件生产技术；在管理上，应该采用多维的工程管理模式。

2、就项目管理方面而言，软件重用项目与非重用项目有哪些不同之处。

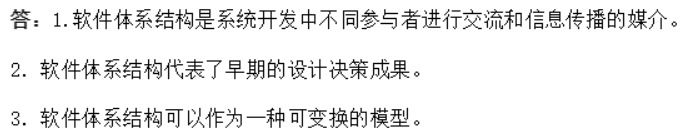
答：软件重用项目需要考虑生产可重用软部件的机制。同时，软件重用项目可以减少开发活动中大量的重复性工作，能提高软件生产率，降低开发成本，缩短开发周期。同时由于大量使用软构件，软件的灵活性和标准化程度也可望得到提高。因此，软件重用项目在管理是比非重用项目更节约管理资源，同时管理起来更方便。



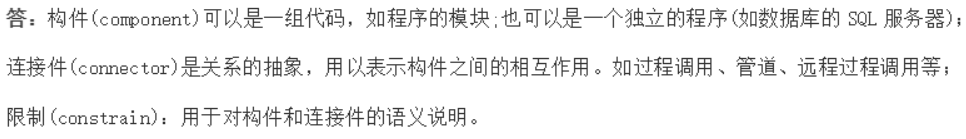
3、实际参与/组织一个软件重用项目的开发，然后总结你是如何组织该项目的开发的。



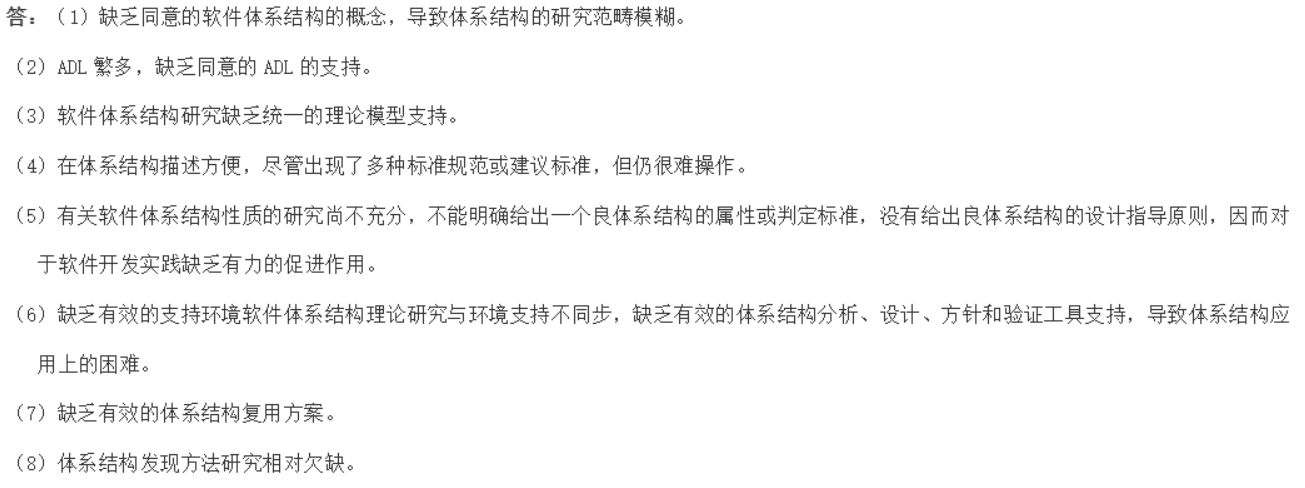
4、为什么要研究软件体系结构？



5、根据软件体系结构的定义，你认为软件体系结构的模型应该由哪些部分组成？



6、在软件体系结构的研究和应用中，你认为还有哪些不足之处？



7. 详细了解什么面向服务体系结构？

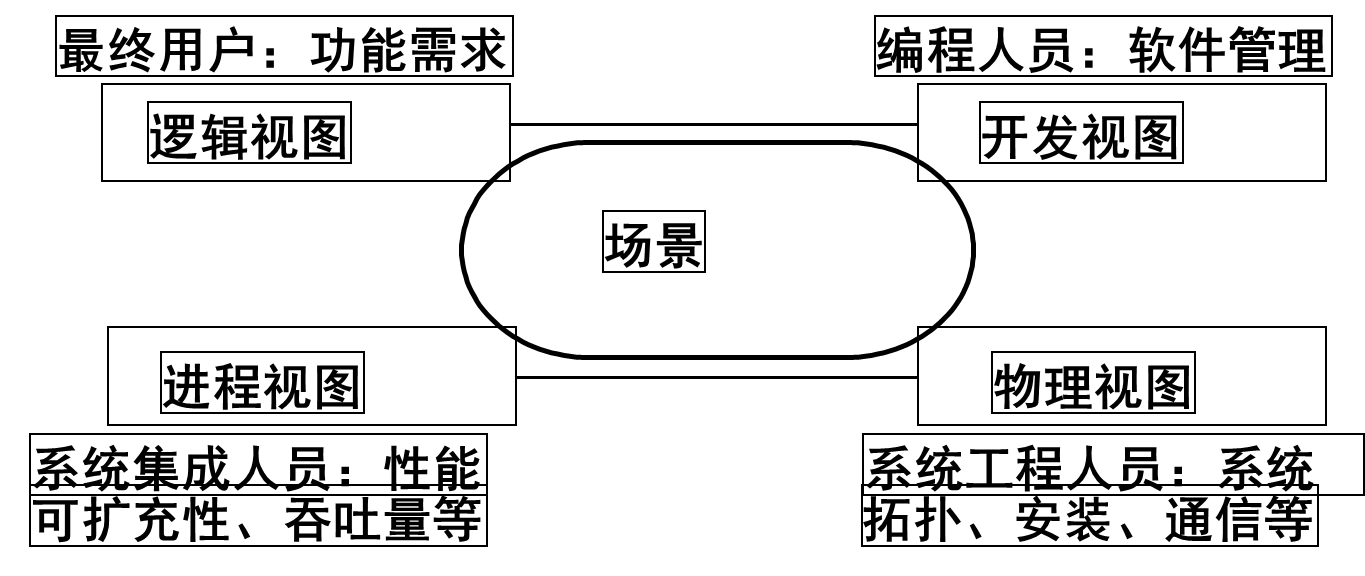
面向服务的体系结构（SOA）是基于用于同步和异步应用程序的请求/答复设计范例的分布式计算的演变。应用程序的业务逻辑或各个功能被模块化，并作为针对消费者/客户端应用程序的服务呈现。这些服务的关键是它们的松散耦合性质。即服务接口独立于实现。应用程序开发人员或系统集成商可以通过组合一项或多项服务来构建应用程序，而无需了解服务的基础实现。例如，可以在.Net或J2EE中实现服务，而使用该服务的应用程序可以在其他平台或语言上。

为什么选择SOA？IT企业基础结构在操作系统，应用程序，系统软件和应用程序基础结构之间是异构的。 一些现有的应用程序用于运行当前的业务流程，因此不能从头开始构建新的基础架构。 企业应迅速敏捷地响应业务变化； 利用对应用程序和应用程序基础结构的现有投资来满足新的业务需求； 支持与客户，合作伙伴和供应商互动的新渠道； 并具有支持有机业务的架构。 SOA具有松散耦合的性质，允许企业以粒度方式插入新服务或升级现有服务，以满足新的业务需求，提供使服务跨不同渠道可使用的选项，并将现有企业和遗留应用程序公开为服务 ，从而保护现有的IT基础架构投资。

第二章

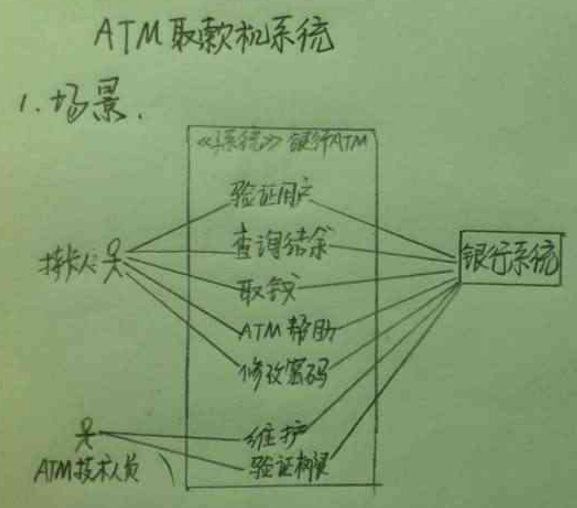
1. 选择一个规模合适的系统，为其建立“4+1”模型。

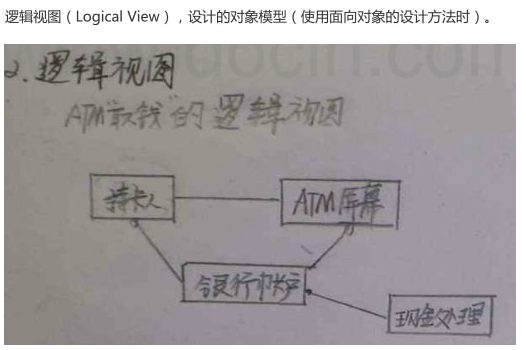
答：模型概述（具体见PPT）



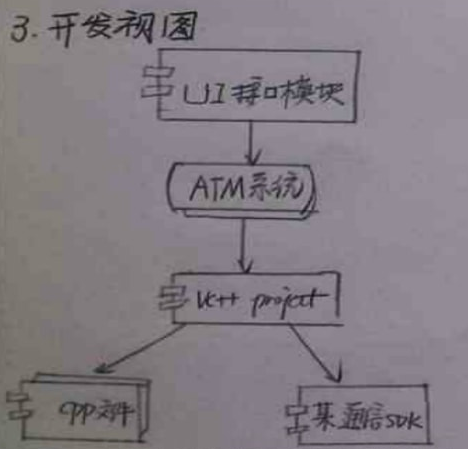
这里设计一个ATM系统的4+1模型。

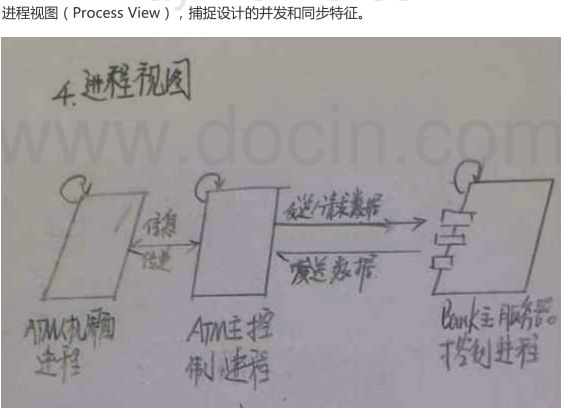
场景如下：



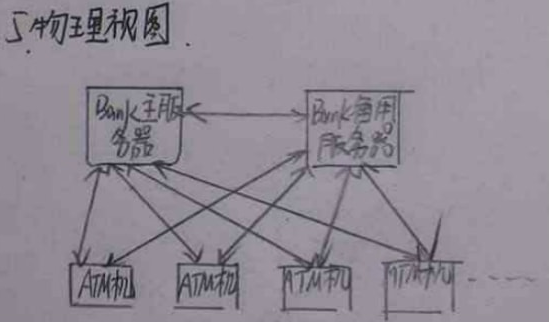






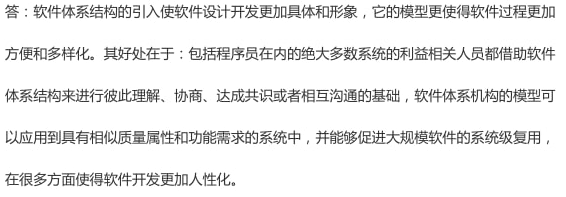




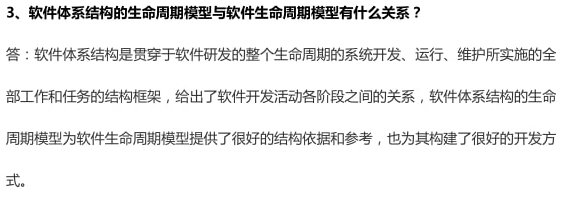




2、引入了软件体系结构以后，传统软件过程发生了哪些变化？这种变化有什么好处？



3、软件体系结构的生命周期模型与软件生命周期模型有什么关系？

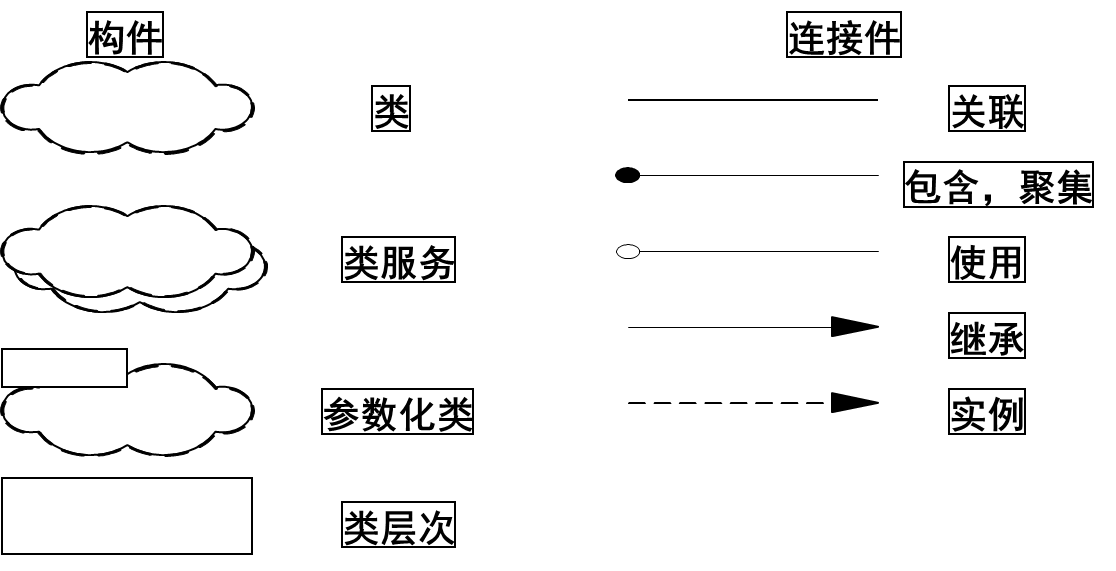


4、详细了解4+1视图。

答：（架构图已在题目1中回答）

**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。在逻辑视图中，系统分解成一系列的功能抽象，这些抽象主要来自问题领域。在面向对象技术中，通过抽象、封装和继承，可以用对象模型来代表逻辑视图，用类图来描述逻辑视图。

可以从Booch标记法中导出逻辑视图的标记法，只是从体系结构级的范畴来考虑这些符号，用Rational Rose进行体系结构设计。



逻辑视图中使用的风格为面向对象的风格，逻辑视图设计中要注意的主要问题是要保持一个单一的、内聚的**对象模型贯穿整个系统。**



对于规模更大的系统来说，体系结构级中包含数十甚至数百个类。



**开发视图**也称模块视图，主要侧重于软件模块的组织和管理。开发架构关注软件开发环境下实际模块的组织。软件打包成小的程序块（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织成分层结构，每个层为上一层提供良好定义的接口。系统的开发架构用模块和子系统图来表达，显示了“输出”和“输入”关系。与逻辑视图一样，可以使用Booch标记法中某些符号来表示开发视图。



在开发视图中，最好采用4-6层子系统，而且每个子系统仅仅能与同层或更低层的子系统通讯，这样可以使每个层次的接口既完备又精练，避免了各个模块之间很复杂的依赖关系。

设计时要充分考虑，对于各个层次，层次越低，通用性越强，这样，可以保证应用程序的需求发生改变时，所做的改动最小。开发视图所用的风格通常是层次结构风格。



**进程视图：**考虑一些非功能性的需求，如性能和可用性。它解决并发性、分布性、系统完整性、容错性的问题，以及逻辑视图的主要抽象如何与进程结构相配合在一起，即定义逻辑视图中的各个类的具体操作是在哪一个线程（Thread）中被执行。过程视图侧重系统的运行特性。服务于系统集成人员。通过扩展Booch对Ada任务的表示法，来表示进程视图



**物理视图**主要考虑如何把软件映射到硬件上，它通常要考虑到系统性能、规模、可靠性等。解决系统拓扑结构、系统安装、通讯等问题。

大型系统的物理视图可能会变得十分混乱，因此可以与进程视图的映射一起以多种形式出现，也可单独出现。



具有进程分配的大型ACS系统的物理视图（下列图）



**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。在开发体系结构时，它可以帮助设计者找到体系结构的构件和它们之间的作用关系。同时，也可以用场景来分析一个特定的视图，或描述不同视图构件间是如何相互作用的。

场景可以用文本表示，也可以用图形表示。下列图表示的是本地呼叫场景的一个原型。



第三章

1. 试分析和比较B/S，二层C/S和三层C/S，指出各自的优点和缺点。

答：二层C/S体系结构将应用一分为二，服务器负责数据管理，客户机完成与用户的交互任务。优点（1）C/S体系结构具有强大的数据操作的事务处理能力，模型思想简单，易于人们理解和接受。（2）对软硬件的变化有极大的适应性和灵活性，易于对系统进行扩充和缩小。（3）系统中的功能构建充分隔离，节约大量费用。缺点：（1）开发成本较高。（2）客户端程序设计复杂（3）信息内容和形式单一（4）用户界面风格不一，使用繁杂不易推广。（5）软件移植困难（6）软件维护和升级困难（7）新技术不能轻易应用。三层CS在上面的基础上进行了改造，并增加了一个服务器，其优点：（1）允许合理的划分三层结构的功能，能提高系统和软件的可维护性和可扩展性。（2）具有良好的可升级性和开放性。（3）应用的各层可以并行开发，可以选择各自最适合的开发语言。（4）为严格的安全管理奠定了坚实的基础。

B/S风格就是上述三层应用结构的一种实现方式，其具体结构为：浏览器/Web服务器/数据库服务器。优点（1）基于B/S体系结构的软件，系统安装，修改和维护全在服务器端解决。（2）提供了异种机，异种网，异种应用服务的联机，联网，同意服务的最现实的开放性基础。缺点（1）缺乏对动态页面的支持能力，没有集成有效的数据库处理能力。（2）在数据查询等响应速度上，要远远低于C/S体系结构。（3）数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理应用。

2. 组织或参与一个采用B/S和C/S混合体系结构的软件项目的开发，总结开发经验。

答：首先，开发者根据一定的原则，将系统的所有子功能分类，决定哪些子功能适合采用C/S，哪些适合采用B/S。适合采用C/S的子功能应具备以下特点：1安全性要求高；2要求具有较强的交互性；3使用范围小，地点固定；4要求处理大量数据。例如，仓库管理系统中的入库单、领料单的输入功能，财务系统中的凭证输入功能等等。而适合采用B/S的子功能应具备以下特点：1使用范围广，地点灵活；2功能变动频繁；3安全性、交互性要求不同。例如：企业内部信息发布功能，意见箱输入功能，公司财务分析表的查询功能，总裁决策支持系统中的查询功能等等。

相对于单独采用C/S或B/S，这种方案的优点在于：1保证敏感数据的安全性，特别是对数据库的修改和新增记录加强了控制；2经济有效地利用企业内部计算机的资源，简化了一部分可以简化的客户端；3既保证了复杂功能的交互性，又保证了一般功能的易用与统一；4系统维护简便，布局合理；5网络效率最高。

如果系统开发者在系统设计阶段决定采用这种C/S与B/S相结合的模式，那么在系统开发生命周期的如下各个阶段相对这种新模式都应有所响应。

在系统设计阶段主要考虑的是MIS系统平台选择问题。在详细设计阶段，系统开发者需要根据企业自身的业务特点，以及一定的选择原则，来决定各个子功能采用哪一种模式并在系统说明书上分别注明。在编码设计阶段，系统开发者需要针对采用不同模式的子功能，选用不同的编码方式(例如：C/S可以采用VB编程环境，而B/S采用ASP方法)，然后编译生成不同的客户应用及Web服务程序。在安装调试阶段，其特点主要体现在系统的物理结构上，即特定的客户应用程序将被安装在特定的使用者的客户端上，Web服务程序需要被安装在Web服务器上，而每个客户端上都将被安装上浏览器，同时，客户应用的使用者必须接受一定的培训。在软件维护阶段，针对不同模式的子功能应采取不同维护方式。

3. 组织或参与一个采用三层体系结构的软件项目的开发，总结开发经验。

三层体系结构包括：用户界面表示层(USL) 、业务逻辑层(BLL) 、数据访问层(DAL) 。各层的作用 ：1：数据访问层:主要是对原始数据(数据库或者文本文件等存放数据的形式)的操作层，而不是指原始数据，也就是说，是对数据的操作，而不是数据库,具体为业务逻辑层或表示层提供数据服务。2：业务逻辑层:主要是针对具体的问题的操作，也可以理解成对数据层的操作,对数据业务逻辑处理，如果说数据层是积木，那逻辑层就是对这些积木的搭建。3：表示层:主要表示WEB方式,也可以表示成WINFORM方式,WEB方式也可以表现成:aspx, 如果逻辑层相当强大和完善,无论表现层如何定义和更改,逻辑层都能完善地提供服务。

三层是指逻辑上的三层，即使这三个层放置到一台机器上。 三层体系的应用程序将业务规则、数据访问、合法性校验等工作放到了中间层进行处理。在保证客户端功能的前提下，为用户提供一个简洁的界面。这意味着如果需要修改应用程序代码，只需要对中间层应用服务器进行修改，而不用修改成千上万的客户端应用程序。“中间业务层”的用途有很多，例如：验证用户输入数据、缓存从数据库中读取的数据等等....但是，“中间业务层”的实际目的是将“数据访问层”的最基础的存储逻辑组合起来，形成一种业务规则。要保证“数据访问层”的中的函数功能的原子性！即最小性和不可再分。“数据访问层”只管负责存储或读取数据就可以了。

我们用三层结构主要是使项目结构更清楚，分工更明确，有利于后期的维护和升级。它未必会提升性能，因为当子程序模块未执行结束时，主程序模块只能处于等待状态。这说明将应用程序划分层次，会带来其执行速度上的一些损失。但从团队开发效率角度上来讲却可以感受到大不相同的效果。需要说明一下，三层结构不是.NET的专利，也不是专门用在数据库上的技术。它是一种更加普适的架构设计理念。此种架构要在数据库设计上注意表之间的关系，尽力满足主与子的关系。在功能上对用户要有一定的限制，不要表现在对于子表的删除操作一定要慎重，以免造成主表与子表的数据在逻辑上出现的主表的外键在子表中没有相对应的值。

“三层结构”开发模式，入门难度够高，难于理解和学习。这是对于初学程序设计的人来说的。以这种模式开发出来的软件，代码量通常要稍稍多一些。这往往会令初学者淹没在茫茫的代码之中。望之生畏，对其产生反感，也是可以理解的。

4. 在软件开发中，采用异构结构有什么好处，其负面影响有哪些？

答：1）.结构有不同的处理能力的强项和弱点，一个系统的体系结构应该根据实际需要进行选择，以解决实际问题。

2）.软件包，框架，通信以及其他一些体系机构上的问题，目前存在者多中标准。即使再某一段时间内某一标准占据着统治地位，但变动最终是绝对的。

3）.工作中，我们总会遇到一些遗留下的代码，它们仍有效用，但是却与新系统有某种程度上的不协调。然而在很多场合，将技术与经济综合进行考虑时，总是决定不重写它们。

4）.在某一单位中，规定了共享共同的软件包或相互关系的一些标准，仍会存在解释或表示习惯上的不同。

负面影响：大多数应用程序只使用10%的代码实现系统的公开功能，剩下90%的代码完成系统管理功能：输入和输出，用户界面，文本编辑，基本图表，标准对话框，通信，数据确认和旁听追踪，特定领域的基本定义等。

5. 至少详细了解一种体系结构风格.

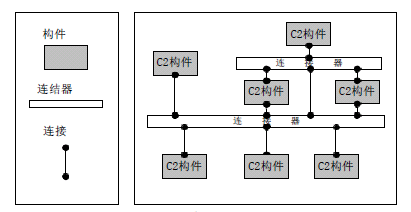
答：**C2风格**是一种基于构件和消息的架构风格。它支持大粒度的复用和灵活的系统组成，强调构件间的松耦合。通过连接件绑定在一起的按照一组规则运作的并行构件网络。C2风格中的系统组织规则如下：

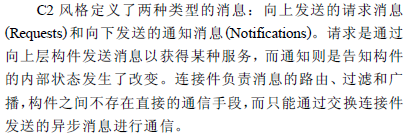
◎ 系统中的构件和连接件都有一个顶部和一个底部；

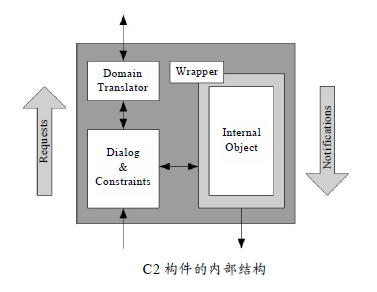
◎ 构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直接连接是不允许的；

◎ 一个连接件可以和任意数目的其它构件和连接件连接；

◎ 当两个连接件进行直接连接时，必须由其中一个的底部到另一个的顶部。







C2风格具有下列特点：

◎ 系统中的构件可实现应用需求，并能将任意复杂度的功能封装在一起；

◎ 所有构件之间的通讯是通过以连接件为中介的异步消息交换机制来实现的；

◎ 构件相对独立，构件之间依赖性较少。

第四章

**1、体系结构描述有哪些方法？有哪些标准和规范？**

文本

描述已自动生成

Rational建议标准：与IEEE P1471相比，Rational建议标准的体系结构描述方案涉及面比较窄，所注重的层次比较低，因而更具体。由于将体系结构的描述限于UML和RUP，具有一定的局限性，但该建议标准结合了业界已经广泛采用的建模语言和开发过程，因而易于推广，可以有效实现在跨组织之间重用体系结构描述结果

**2、体系结构描述语言与程序设计语言有什么区别？**

文本, 信件

描述已自动生成

上的多种推理分析

典型元素含义的比较：

表格

描述已自动生成

**3、选择一个规模适中的系统，使用UML为其建模。**

为一个网上购物系统利用UML为其建模，如下：

统一开发过程RUP把整个软件开发过程分为初始、细化、构造、交付四个阶段，具有用例驱动、以架构为中心、迭代和增量的特点。同其它软件开发方法相比较，RUP具有自身独特的优势，为软件开发提供了重要的方法论指导根据对网上购物系统的体系结构及建模分析，采用UML作为建模语言，结合RUP的基本开发过程，提出适合网上购物系统开发的建模过程。该过程遵循了RUP四个阶段的理论，主要是对初始和细化两个阶段进行了详细的分析。整个过程包括业务建模、需求建模、对象建模、数据库建模和物理建模等五个步骤，每个步骤都会生成一定的系统模型，并用相应的UML图来描述这些模型。在建模过程中，采用了RUP中迭代增量式的开发思想，把系统的建模进一步分解为迭代，一个迭代是一个从系统的业务建模到物理建模的完整过程，每一个迭代都会产生一个模型版本，是最终模型的一个子集，它增量式地发展，从一个迭代过程到另一个迭代过程，直到成为系统的最终模型。

1 业务建模

业务建模用于对网上购物系统环境的业务过程进行建模。系统分析人员通过网上购物系统的业务建模能够了解系统所处的环境和业务过程，业务建模能够将这些信息进行体现，并表现环境中存在或可以觉察到的过程，从而详细说明网上购物系统所要支持的业务过程。业务建模既确定了业务过程涉及到的业

务对象和领域对象，又确定了每个业务过程所需要的资源和能力，包括人员、每个人员的职责和执行的操作、过程的执行方式和协作等。这些信息对于下一步的需求建模是非常重要和有用的。

业务建模一般通过业务过程图来进行描述。业务过程图是对事件逻辑的归类，这些事件被认为是业务的基本元素。其目的是将整个业务领域作为一个过程集进行描述，而不关心过程的次序或单个过程之间的交互作用。业务过程图不必严格精密，它应该全面而不是精确。重要的是，通过查看业务过程图，系统分析人员、设计人员、开发人员和用户能够迅速获得关于业务范围和活动的总体印象。

2 需求建模

需求建模的主要工作是获得系统的需求，建立待开发系统的模型。而用例则有助于更好地了解系统需求并以规范化的格式进行描述。需求建模就是要以用例的方式来描述系统的功能，其主要工作成果是用例模型。采用用例模型来描述进行需求建模的主要过程如下：（1）确定所要开发系统的参与者，参与者可以是人，也可以是与系统交互的外部系统。网上购物系统的参与者主要有员、工作人员、顾客、支付系统等。（2）从执行者的角度出发，分析他和系统需要进行的交互作用，并从这些交互过程中抽象出用例。从顾客的角度出发，网上购物系统一般有以下用例：用户登录、用户注册、浏览商品、搜索商品、购买商品、下订单、支付等；从系统员的角度出发，网上购物系统一般有以下用例：用户登录、用户、商品、订单等。（3）对每一用例确定其主要的业务过程。例如“用户登录”用例的业务过程为顾客、系统员、工作人员等通过用户登录可以获得相应的服务；“支付”用例的业务过程为顾客为所够买物品选择付款方

式进行付款。（4）以信息流为中心逐步形成完整的用例模型。网上购物系统的完整用例模型包括很多的用例图，其中既包括系统的顶层用例图，也包括各种细化的用例图。

3 对象建模

确定了系统的需求分析、得出系统的用例模型以后，需要进行的主要任务就是对系统进行对象建模。对象建模的主要工作是把需求建模阶段产生的用例模型转化为系统的静态结构模型和动态行为模型，使建立的系统在特定的环境下完成需求分析中的任务和功能，有利于系统的实现和迭代。这其中主要包括静态结构建模和动态对象建模两部分。

4 数据库建模

数据库建模是从计算机系统的角度对系统所要处理的数据进行建模。数据库系统是整个网上购物系统的基础，数据库建模的好坏直接影响到整个系统的结构、实现的复杂程度、性能、安全性和可维护性等。传统的逻辑数据库建模工具“实体一联系(E．R)\图只针对数据建模，不能对行为建模。而UML的类图能够更好的用于数据库建模。UML的类图不但对数据建模，而且能对行为建模，这些行为在物理数据库中被设计成触发器和存储过程。即使是关系数据库，也可以在类图设计落实后，再采取标准方法把类图映射到具体的关系模型。从类图到关系模型的转换，按照一个类映射为一个关系的原则进行，而类的属性即为关系的属性，标识的标识符即为关系的主键。

5 物理建模

物理建模用于网上购物系统建模过程的最后阶段，是对网上购物系统的物理方面进行建模。它使用UML中的组件图描述网上购物系统中代码组件的物理

结构及各个组件之间的依赖关系，使用配置图定义网上购物系统的软硬件结构及通信机制，表示软硬件系统之间的协作关系。

以上五个步骤是根据RUP的四个阶段细化的结果，分别对应了RUP中相应的核心工作流。其中业务建模对应RUP的业务建模工作流，需求建模对应RUP的需求建模工作流，对象建模和数据库建模对应RUP的分析和设计工作流，物理建模对应RUP的实施工作流，对开发完成的系统进行测试、部署和分别对应RUP的测试工作流、部署工作流等。

**4、详细了解一种设计模式**

在简单工厂模式中创建实例的方法通常为静态（static）方法，因此简单工厂模式（Simple Factory Pattern）又叫作静态工厂方法模式（Static Factory Method Pattern）。  
  
简单来说，简单工厂模式有一个具体的工厂类，可以生成多个不同的产品，属于创建型设计模式。简单工厂模式不在 GoF 23 种设计模式之列。  
  
简单工厂模式每增加一个产品就要增加一个具体产品类和一个对应的具体工厂类，这增加了系统的复杂度，违背了“开闭原则”。

“工厂方法模式”是对简单工厂模式的进一步抽象化，其好处是可以使系统在不修改原来代码的情况下引进新的产品，即满足开闭原则。

**优点和缺点**

**优点：**

1. 工厂类包含必要的逻辑判断，可以决定在什么时候创建哪一个产品的实例。客户端可以免除直接创建产品对象的职责，很方便的创建出相应的产品。工厂和产品的职责区分明确。
2. 客户端无需知道所创建具体产品的类名，只需知道参数即可。
3. 也可以引入配置文件，在不修改客户端代码的情况下更换和添加新的具体产品类。

**缺点：**

1. 简单工厂模式的工厂类单一，负责所有产品的创建，职责过重，一旦异常，整个系统将受影响。且工厂类代码会非常臃肿，违背高聚合原则。
2. 使用简单工厂模式会增加系统中类的个数（引入新的工厂类），增加系统的复杂度和理解难度
3. 系统扩展困难，一旦增加新产品不得不修改工厂逻辑，在产品类型较多时，可能造成逻辑过于复杂
4. 简单工厂模式使用了 static 工厂方法，造成工厂角色无法形成基于继承的等级结构。

**应用场景**

对于产品种类相对较少的情况，考虑使用简单工厂模式。使用简单工厂模式的客户端只需要传入工厂类的参数，不需要关心如何创建对象的逻辑，可以很方便地创建所需产品。

**模式的结构与实现**

简单工厂模式的主要角色如下：

* 简单工厂（SimpleFactory）：是简单工厂模式的核心，负责实现创建所有实例的内部逻辑。工厂类的创建产品类的方法可以被外界直接调用，创建所需的产品对象。
* 抽象产品（Product）：是简单工厂创建的所有对象的父类，负责描述所有实例共有的公共接口。
* 具体产品（ConcreteProduct）：是简单工厂模式的创建目标。

其结构图如下图所示。

图示

描述已自动生成

第五章

**1、什么是动态软件体系结构？动态软件体系结构与静态软件体系结构有什么区别**

****

动态软件体系结构的动态性包括：

* 交互性动态性
* 结构化动态性
* 体系结构动态性

由于系统需求，技术，环境，分布等因素的变化而最终造成软件体系结构的变动，称之为软件体系结构演化。软件系统在运行时刻的体系结构变化称之为软件体系结构的动态性

**2、基于构件的动态软件体系结构模型的层次结构是什么？**

答：基于构件的动态系统结构模型支持运行系统的动态更新，该模型分为三类，分别是应用层，中间层和体系结构层。

（1）应用层：处于最底层，包括构件链接，构件接口和执行

（2）中间层：包括连接件配置，构件配置，构件描述及执行

（3）体系结构层：位于最上层，控制和管理整个体系结构，包括体系结构配置，体系结构描述和执行。

**3、如何使用πADL进行动态体系结构建模？能使用一种动态描述语言对一个简单系统的体系结构进行描述。**

图示

描述已自动生成

第六章

**1、什么是Web服务体系结构？与传统的结构相比，使用Web服务有**哪些好处？

Web服务体系结构：

◎ Web服务作为一种新兴的Web应用模式，是一种崭新的分布式计算模型，是Web上数据和信息集成的有效机制。

◎ Web服务就像Web上的构件编程，开发人员通过调用Web应用编程接口，将Web服务集成进他们的应用程序，就像调用本地服务一样。

优势：

◎ 高度的通用性和易用性

◎ 完全的平台、语言独立性

◎ 高度的集成性

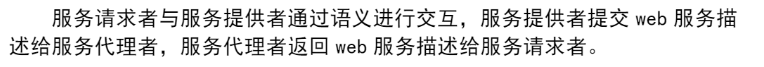
◎ 容易部署和发布

**2、在Web服务中，如何实现其松散耦合的特点？**

文本

描述已自动生成

**3、试分析服务提供者、服务请求者和服务代理三者的作用，以及它们之间的工作流程。**



**4、试解释Web服务栈的层次结构。**

信件

低可信度描述已自动生成

**5、Web服务有哪些核心技术，这些技术是如何在Web服务中发挥作用的。**

* 作为Web服务基础的XML
* 简单对象访问协议
  + SOAP信封
  + SOAP编码规则
  + SOAP RPC表示
  + SOAP绑定
* Web服务描述语言
* 统一描述、发现和集成协议

**6、从管理的角度看，SOA有什么优点？**

集成现有系统，不必另起炉灶

面向服务的体系结构可以基于现有的系统投资来发展，而不需要彻底重新创建系统。通过使用适当的SOA框架并使其用于整个企业，可以将业务服务构造成现有组件的集合。使用这种新的服务只需要知道它的接口和名称。服务的内部细节以及在组成服务的组件之间传送的数据的复杂性都对外界隐藏了。这种组件的匿名性使组织能够利用现有的投资，从而可以通过合并构建在不同的机器上、运行在不同的操作系统中、用不同的编程语言开发的组件来创建服务。遗留系统可以通过Web 服务接口来封装和访问

服务设计松耦合，带来多方面优点

服务是位置透明的，服务不必与特定的系统和特定的网络相连接。服务是协议独立的，服务间的通信框架使得服务重用成为可能。对于业务需求变化，SOA能够方便组合松耦合的服务，以提供更为优质和快速的响应，允许服务使用者自动发现和连接可用的服务。松耦合系统架构使得服务更容易被应用所集成，或组成其他服务，同时提供了良好的应用开发、运行时服务部属和服务管理能力。提供对服务使用者的验证（authentication）授权（authorization），来加强安全性保障，这一点也优于其他紧耦合架构。

统一了业务架构，可扩展性强

在所有不同的企业应用程序之间，基础架构的开发和部署将变得更加一致。现有的组件、新开发的组件和从厂商购买的组件可以合并在一个定义良好的SOA框架内。这样的组件集合将被作为服务部署在现有的基础构架中，从而使得可以更多地将基础架构作为一种商品化元素来加以考虑，增强了可扩展性。又由于面向服务的敏捷设计，在应对业务变更时，有了更强的“容变性”

加快了开发速度，减少了开发成本

SOA允许清晰地表示流程流，这些流程流通过在特定业务服务中使用的组件的顺序来标识。这给商业用户提供了监视业务操作的理想环境。业务建模反映在业务服务中。流程操纵是以一定的模式重组部件（构成业务服务的组件）来实现的。这将进一步允许更改流程流，而同时监视产生的结果，因此促进了持续改进。重用现有的组件降低了在增强或创建新的业务服务过程中带来的风险，也减少了维护和管理支持服务基础架构的风险。

**7、在实际开发中，如何实现Web服务和SOA结构？**

几乎所有的SOA应用场合都是和Web Service绑定的，且Web Service是现在最适合实现SOA的技术，目前，我们基本认同Web Service技术在几方面体现了SOA的需要：

基于标准访问的独立功能实体满足了松耦合要求 --- 在Web Service中所有的访问都通过SOAP进行，用WSDL定义的接口封装，通过UDDI进行目录查找，可以动态改变一个服务的提供方而无需影响客户端的配置，外界客户端是根本不关心服务器端的实现的。

适合大数据量、低频率访问，符合服务大颗粒度功能 --- 基于性能和效率平衡的要求，SOA的服务提供的是大颗粒度的应用功能，而且跨系统边界的访问频率也不会像程序见函数调用那么频繁。 通过WSDL和基于文本的SOAP请求，可以实现能一次性接受处理大量数据。

基于标准的文本消息传递为异构系统提供通讯机制：web service所有的通讯都是通过SOAP进行的，而SOAP是基于XML的，XML是结构化的文本消息。

即SOA不是Web Servcie，而Web Service是目前最适合实现SOA的技术。

　　 即Web Service是一套标准，可以使SOA思想用于实践。

第七章

1、请把基于体系结构的软件开发模型与其他软件开发模型进行比较。

答:一、基于体系结构为基础的基于构件组装的软件开发模型，如基于构件的开发模型和基于体系结构的开发模型等。基于体系结构的开发模型是以软件体系结构为核心，以基于构件的开发方法为基础。然后采用迭代增量方式进行分析和设计，将功能设计空间映射到结构设计空间，再由结构设计空间映射到系统设计空间的过程。该开发模型把软件生命周期分为软件定义、需求分析和定义、体系结构设计、软件系统设计和软件实现5个阶段.

特点∶是利用需求分析结果设计出软件的总体结构，通过基于构件的组装方法来构造软件系统。

优点∶基于构件的开发方法使得软件开发不再一切从头开发，开发的过程就是构件组装的过程，维护的过程就是构件升级、替换和扩充的过程。构件组装模型导致了软件的复用，提高了软件开发的效率。软件体系结构的出现使得软件的结构框架更清晰，有利于系统的设计、开发和维护。并且软件复用从代码级的复用提升到构件和体系结构级的复用。

缺点∶由于采用自定义的组装结构标准，缺乏通用的组装结构标准，因而引入了较大的风险。可重用性和软件高效性不易协调，需要精干的有经验的分析和开发人员，一般开发人员插不上手。客户的满意度低，并且由于过分依赖于构件，所以构件库的质量影响着产品质量。

二、以软件需求完全确定为前提软件开发模型，如瀑布模型等。

特点︰软件需求在开发阶段已经被完全确定，将生命周期的各项活动依顺序固定，强调开发的阶段性;

优点∶开发流程简单。

缺点∶是开发后期要改正早期存在的问题需要付出很高的代价，用户需要等待较长时间才能够看到软件产品，增加了风险系数。并且如果在开发过程存在阻塞问题，则影响开发效率。

三、在开始阶段只能提供基本需求的渐进式开发模型，如螺旋模型和原型实现软件开发模型等。

特点∶软件开发开始阶段只有基本的需求，软件开发过程的各个活动是迭代的。通过迭代过程实现软件的逐步演化，最终得到软件产品。在此引入了风险管理，采取早期预防措施，增加项目成功几率，提高软件质量;

优点∶强调了其他模型均忽略了的风险分析。逐步调整原型使其满足客户的要求，而同时也使开发者对将要做的事情有更好的理解。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。

缺点:由于需求的不完全性，从而为软件的总体设计带来了困难和削弱了产品设计的完整性，并要求对风险技能管理水平的高要求。

2、请把基于体系结构的软件设计方法与其他软件设计方法进行比较。

答∶基于体系结构的软件设计方法为产生软件系统的概念体系结构提供构造，概念体系结构是由Hofimeister、Nord和Soni提出的四种不同的体系结构中的一种，它描述了系统的主要设计元素及其关系。概念体系结构代表了在开发过程中作出的第一个选择，相应地，它是达到系统质量和商业目标的关键，为达到预定功能提供了一个基础。

体系结构驱动，是指构成体系结构的商业、质量和功能需求的组合。使用基于体系结构的软件设计方法，设计活动可以在体系结构驱动一决定就开始，需求抽取和分析活动与设计活动并行。特别是在不可能预先决定所有需求时，例如产品线系统或长期运行的系统，快速开始设计是至关重要的。

基于体系结构的开发方法从系统的总体结构入手，将一个系统的体系结构显示化，以在高抽象层次处理诸如全局组织和控制结构、功能到计算元素的分配、计算元素间的高层交互等设计问题。

基于体系结构的开发方法相比其他软件设计方法，出现了基于构件、基于框架、基于模板的开发技术，项目利用基于体系结构开发的思想，从需求到高层框架设计、架构设计、平台实现，充分体现了基于体系结构开发中从高层设计入手，复用体系结构框架、设计和实现可复用的构件的原则。

此外，与其他软件设计方法的区别还有︰

1、在基于软件体系结构的开发方法中，单个用户界面和业务逻辑的改动不

会影响到系统整体，可根据新的需求选择或设计新的构件进行组装。2、在基于软件体系结构的开发方法中，开发人员的组织用户界面和业务逻

辑的定制可由领域人员来完成，不必专业计算机软件人员参与;专业计算机人员则集中精力做构件库的部分，完成业务逻辑的抽象，制作不同粒度的构件，完成模型的开发。

3、在基于软件体系结构的开发方法中，不同业务对象有相似的业务逻辑，可

直接复用其构件。不同的业务逻辑可以有相似业务流程、业务规则、和原子数据操作，这些都直接可复用构件库，这种构件级的重用将大大节省了开发成本

4、在基于软件体系结构的开发方法中，在针对新的需求的开发，可以复用原

有系统的设计和实现，在系统维护过程更新某一模块，不会影响系统的其它部分。

3、如何才能提高软件系统的可演化性。

答:在生物学里面演化又称进化，是指生物在不同世代之间具有差异的现象，以及解释这些现象的各种理论。但是在计算机软件领域里，

“演化”这

个术语一般是指在性质和特性方面的递增式的改变。在某种意义上来说，这

个在某个方面或某些方面属性的改变的过程，导致了新特性的出现或进行改进。一般来说，这种改变都是诸如改变类的成员以适应环境的改变。改变会使得它们更有用或更有意义，而且在某种程度上会增加它们的价值。与此同时，演化也会去掉一些不合适的特性。也可以认为，软件系统的演化指的是在软件系统的生命周期内软件维护和软件更新的行为和过程。在软件系统的生命周期中，演化是一项贯穿始终的活动。

在如何提高软件系统的可演化性上，Lehman提出了软件演化的八条规律︰

(1)必须频繁地变化以适应要求。

(2)软件的复杂度不断地增长。

(3)通过自我调节以符合产品需求和过程特性。

(4)在软件的生命周期中保持一定的组织稳定性。

(5)不同的版本之间保持一定的连贯性。

(6)功能持续地增加。

(7)在没有严格的维护和适应性修改的情况下会出现质量衰退。

(8)是一个反馈系统。

第八章

1、什么是软件体系结构的可靠性？为什么要研究软件体系结构的可靠性？

软件可靠性 (software reliability )是软件产品在规定的条件下和规定的时间区间完成规定功能的能力。规定的条件是指直接与软件运行相关的使用该软件的计算机系统的状态和软件的输入条件，或统称为软件运行时的外部输入条件；规定的时间区间是指软件的实际运行时间区间；规定功能是指为提供给定的服务，软件产品所必须具备的功能。软件可靠性不但与软件存在的缺陷和（或）差错有关，而且与系统输入和系统使用有关。软件可靠性的概率度量称软件可靠度。

2、如何模型化系统的可靠性？

◎ 通过系统的详细说明书，确定系统所采用的体系结构风格。

◎ 把每一种体系结构风格转换成状态视图，并计算状态视图中每一个状态的可靠性及其相应的迁移概率。

◎ 通过整个系统的体系结构视图，把所有的状态视图集成为一个整体状态视图。

◎ 通过整体状态视图构造系统的迁移矩阵，并计算系统的可靠性。

3、软件体系结构风险分析有哪些基本步骤？

◎ 采用体系结构描述语言ADL对体系结构进行建模

◎ 通过模拟方法执行复杂性分析

◎ 通过FMEA和模拟运行执行严重性分析

◎ 为构件和连接件开发其启发式风险因子

◎ 建立用于风险评估的CDG

◎ 通过图论中的算法执行风险评估和分析

4、软件体系结构测试是什么？熟悉使用化学抽象机进行测试的方法。

软件体系结构测试与程序测试不同，它是检查软件设计的适用性，这种测试不考 虑软件的实现代码。与传统的软件测试一样，基于体系的软件测试也需要研究测试内容、测试准则、测试用例、测试充分性及测试方法。在传统测试方法中，测试准则是基于实现和规约得到的，基于实现的测试准则是结构化的，它是利用软件的内部结构来定义测试数据以覆盖系统。实现完整测试的典型方法是利用测试准则定义测试需求，进而生成测试用例。

第九章

1、为什么要评估软件体系结构？

答:对一个系统的体系结构进行评估，是为了在系统被构建之前预测它的质量，并不需要精确的评估结果，通过分析SA体系结构对于系统质量的主要影响，进而提出改进。此外，软件体系结构的设计是整个软件开发过程中关键的一步。

而对于当今世界上庞大而复杂的系统来说，若没有一个合适的体系结构，却想要要有一个成功的软件设计几乎是不可想象的。

不同类型的系统需要不同的软件体系结构，甚至一个系统的不同子系统也需要不同的体系结构。因此，体系结构的选择往往会成为一个系统设计成败的关键。

2、从哪些方面评估软件体系结构？

1、性能

性能是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段事件内系统所能处理的事件的个数。

经常用单位时间内所处理事务的数量或系统完成某个事务处理所需的时间来对性能进行定量的表示。

2、可靠性：可靠性是软件系统在应用或系统错误面前，在意外或错误使用的情况下维持软件系统的

功能特性的基本能力。

可靠性可以分为两个方面:

( 1)容错。在错误发生时确保系统正确的行为，并进行内部“修复”

。例如在一个分布式软

件系统中失去了一个与远程构件的连接，接下来恢复了连接。在修复这样的错误之后，软件

系统可以重新或重复执行进程间的操作直到错误再次发生。

( 2)健壮性。这里说的是保护应用程序不受错误使用和错误输入的影响，在遇到意外错误事件时确保应用系统处于已经定义好的状态。值得注意的是，和容错相比，健壮性并不是说在错误发生时软件可以继续运行，它只能保证软件按照某种已经定义好的方式终止执行。

3、可用性：可用性是系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。

4、安全性：安全性是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝

服务的能力。安全性是根据系统可能受到的安全威胁的类型来分类的。安全性又可划分为机密性、完整性、不可否认性及可控性等特性。其中，机密性保证信息不

泄露给未授权的用户、实体或过程;完整性保证信息的完整和准确，防止信息被非法修改;可控性保证对信息的传播及内容具有控制的能力，防止为非法者所用。

5、可修改性

可修改性是指能够快速地以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。可修改性包含四个方面:

( 1)可维护性。这主要体现在问题的修复上︰在错误发生后“修复”软件系统。

( 2)可扩展性。这一点关注的是使用新特性来扩展软件系统，以及使用改进版本来替换构件并删除不需要或不必要的特性和构件。

( 3 )结构重组。这一点处理的是重新组织软件系统的构件及构件间的关系，

(4)可移植性。可移植性使软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器。6、功能性

功能性是系统所能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。

7、可变性：可变性是指体系结构经扩充或变更而成为新体系结构的能力。这种新体系结构应该符合预先定义的规则，在某些具体方面不同于原有的体系结构。当要将某个体系结构作为一系列相关产品（例如，软件产品线)的基础时，可变性是很重要的。

8、可集成性

可集成性是指系统能与其他系统协作的程度。

9、互操作性

作为系统组成部分的软件不是独立存在的，经常与其他系统或自身环境相互作用。为了支持互操作性，软件体系结构必须为外部可视的功能特性和数据结构提供精心设计的软件入口。程序和用其他编程语言编写的软件系统的交互作用就是互操作性的问题，

这种互操作性也影响应用的软件体系结构。

10、概念完整性

能够统一所有层面系统设计概念的能力

11、可分解性

支持生产系统某个子集的能力

3、ATAM评估方法的基本步骤是什么？

步骤一：描述ATAM的方法，步骤二：描述业务动机 ，步骤三：描述体系结构 ，步骤四：描述体系结构方法，步骤五：生成质量属性效果树步骤，六：分析体系结构方法 ，步骤八：描述评估结果

4、详细了解一种体系结构风险分析方法。

SAAM、ATAM（上一题有）