第一章

1、 根据自己的经验，谈谈对软件危机的看法。

答：软件危机是指软件生产方式无法满足迅速增长的计算机需求，开发和维护过程出现的一系列问题。

以下几个原因导致：（1）软件自身特点

（2）开发人员的弱点

（3）用户需求不明

（4）缺乏正确理论指导

（5）开发规模越来越大

（6）开发复杂度越来越高

可以通过软件生命周期的模型和软件工具的使用来缓解危机，通过程序自动化和软件工业化生产的方法实现软件标准化的目标，进一步缓解软件危机带来的影响。

软件危机有利有弊，除了带来许多麻烦，也给我们带来许多挑战，克服危机的过程，我们在技术上和创新上都有了一个提升，也算是间接为软件产业的发展做了贡献。

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：使用软件重用技术可减少重复工作,提高软件生产率, 缩短开发周期。同时，由于软构建大多经过严格的质量认证，因此有助于改善软件质量， 大量使用构建， 软件的灵活性和标准化程度可得到提高。

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

答：参加了一个网页管理系统的开发，该项目重复使用已有的软件产品用于开发新的软件系统，以达到提高软件系统的开发质量与效率，降低开发成本的目的。在过程中使用了代码的复用、设计结果的复用、分析结果的复用、测试信息的复用等。

4、为什么要研究软件体系结构？

答：

（1）软件体系结构是系统开发中不同参与者进行交流和信息传播的媒介。

（2）软件体系结构代表了早期的设计决策成果。

（3）软件体系结构可以作为一种可变换的模型。

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？

答：

（1）构件(component)可以是一组代码，如程序的模块;也可以是一个独立的程序(如数据库的 SQL 服务器)；

（2）连接件(connector)是关系的抽象，用以表示构件之间的相互作用。如过程调用、管道、远程过程调用等；

（3）限制(constrain)：用于对构件和连接件的语义说明。

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？

答：

（1）缺乏同意的软件体系结构的概念，导致体系结构的研究范畴模糊。

（2）ADL 繁多，缺乏同意的 ADL 的支持。

（3）软件体系结构研究缺乏统一的理论模型支持。
（4）在体系结构描述方便，尽管出现了多种标准规范或建议标准，但仍很难操作。
（5）有关软件体系结构性质的研究尚不充分，不能明确给出一个良体系结构的属性或判定标准，没有给出良体系结构的设计指导原则，因而对于软件开发实践缺乏有力的促进作用。
（6）缺乏有效的支持环境软件体系结构理论研究与环境支持不同步，缺乏有效的体系结构分析、设计、方针和验证工具支持，导致体系结构应用上的困难。
（7）缺乏有效的体系结构复用方案。
（8）体系结构发现方法研究相对欠缺。

7、 详细了解什么是面向服务体系结构？

第二章

1、选择一个规模合适的系统，为其建立“4+1”模型。

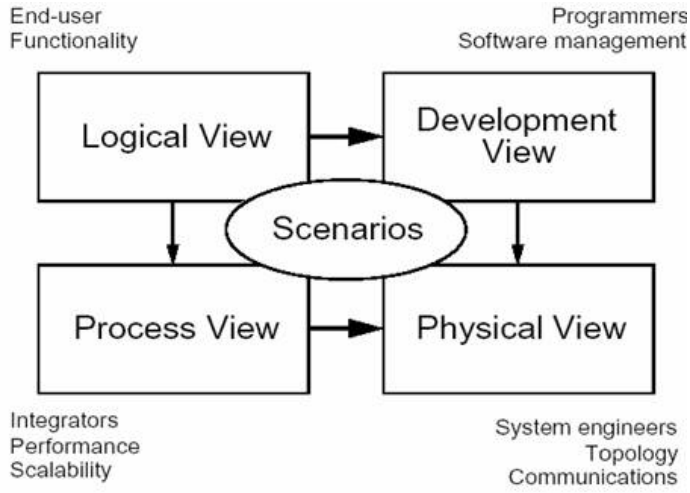
逻辑视图（Logical View），设计的对象模型（使用面向对象的设计方法时）。

过程视图（Process View），捕捉设计的并发和同步特征。

物理视图（Physical View），描述了软件到硬件的映射，反映了分布式特性。

开发视图（Development View），描述了在开发环境中软件的静态组织结构。

架构的描述，即所做的各种决定，可以围绕着这四个视图来组织，然后由一些用例 （use cases）或场景(scenarios)来说明，从而形成了第五个视图。



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？

答：软件体系结构的引入使软件设计开发更加具体和形象，它的模型更使得软件过程更加方便和多样化。其好处在于：包括程序员在内的绝大多数系统的利益相关人员都借助软件体系结构来进行彼此理解、协商、达成共识或者相互沟通的基础，软件体系机构的模型可以应用到具有相似质量属性和功能需求的系统中，并能够促进大规模软件的系统级复用，在很多方面使得软件开发更加人性化。

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？

答：软件体系结构是贯穿于软件研发的整个生命周期的系统开发、运行、维护所实施的全部工作和任务的结构框架，给出了软件开发活动各阶段之间的关系，软件体系结构的生命周期模型为软件生命周期模型提供了很好的结构依据和参考，也为其构建了很好的开发方式。

4、详细了解 4+1 视图。

（1）逻辑视图（Logical View），设计的对象模型（使用面向对象的设计方法时）。

（2）过程视图（Process View），捕捉设计的并发和同步特征。

（3）物理视图（Physical View），描述了软件到硬件的映射，反映了分布式特性。

（4）开发视图（Development View），描述了在开发环境中软件的静态组织结构。

（5）架构的描述，即所做的各种决定，可以围绕着这四个视图来组织，然后由一些用例（use cases）或场景(scenarios)来说明，从而形成了第五个视图。

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。

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.

第四章

1、体系结构描述有哪些方法？有哪些标准和规范？

传统软件体系结构描述方法：

1) 图形表达工具

对于软件体系结构的描述和表达，一种简洁易懂且使用广泛的方法是采用由矩形框和有向线段组合而成的图形表达工具。在这种方法中，矩形框代表抽象构件，框内标注的文字为抽象构件的名称，有向线段代表辅助各构件进行通讯、控制或关联的连接件。

目前，这种图形表达工具在软件设计中占据着主导地位。尽管由于在术语和表达语义上存在着一些不规范和不精确，而使得以矩形框与线段为基础的传统图形表达方法在不同系统和不同文档之间有着许多不一致甚至矛盾，但该方法仍然以其简洁易用的特点在实际的设计和开发工作中被广泛使用，并为工作人员传递了大量重要的体系结构思想。

为了克服传统图形表达方法中所缺乏的语义特征，有关研究人员试图通过增加含有语义的图元素的方式来开发图文法理论。

2) 模块内连接语言

软件体系结构的第二种描述和表达方法是采用将一种或几种传统程序设计语言的模块连接起来的模块内连接语言 MIL（Module Interconnection Language）。由于程序设计语言和模块内连接语言具有严格的语义基础，因此他们能支持对较大的软件单元进行描述，诸如定义/使用和扇入/扇出等操作。

MIL 方式对模块化的程序设计和分段编译等程序设计与开发技术确实发挥了很大的作用。但是由于这些语言处理和描述的软件设计开发层次过于依赖程序设计语言，因此限制了他们处理和描述比程序设计语言元素更为抽象的高层次软件体系结构元素的能力。

2、体系结构描述语言与程序设计语言有什么区别？

ADL 与其他的语言比较具有以下能力：

1) 构造能力：ADL 能够使用较小的独立体系结构元素来建造大型软件系统；
2) 抽象能力：ADL 使得软件体系结构中的构件和连接件描述可以只关注他们的抽象特性，而不管其具体的实现细节
3) 重用能力：ADL 使得组成软件系统的构件，连接件甚至是软件体系结构都成为软件系统开发和设计的可重用部件
4) 组合能力：ADL 使得其描述的每一系统元素都有其自己的布局结构，这种描述布局结构的特点使得 ADL 支持软件系统的动态变化组合
5) 异构能力：ADl 允许多个不同的体系结构描述关联存在；(6)分析和推理能力：ADL 允许对其描述的体系结构进行多种不同的性能和功能上的多种推理分析。ADL 与需求语言的区别在于后者描述的是问题空间，而前者描述的是接空间。ADL 与建模语言的区别在于后者对整体行为的关注要大于部分的关注，而 ADl 集中在构建的表示上。

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，论文采用了 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管理员、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；从系统管理员的角度出发，网上购物系统一般有以下用例：用户登录、用户管理、商品管理、订单管理等。（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、系统管理员、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、部署和管理分别对应 RUP 的测试工作流、部署工作流等。

4、详细了解一种设计模式

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？

答：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，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。

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π ADL 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。

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？

答：

（1）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。

（2）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。

2、在 Web 服务中，如何实现其松散耦合的特点？

答：C/S 结构是松散耦合系统，它们通过消息传递机制进行通话，由客户端发出请求给服务器，服务器进行相应处理后经传递机制送回客户端。

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。

答：服务请求者与服务提供者通过语义进行交互，服务提供者提交 web 服务描述给服务代理者，服务代理者返回 web 服务描述给服务请求者。

4、试解释 Web 服务栈的层次结构。

答：XML(可扩展标记语言)、SOAP(简单对象访问协议)、WSDL(web 服务定义语言)、UDDI(统一描述发现和集成)。

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。

答：

（1）Web 服务技术核心基于可扩展标记语言 XML 的标准，包括简单对象访问协议，Web 服务描述语言和统一描述，发现和集成协议。

（2）SOAP 定义了三部分：定义了描述消息和如何处理消息的框架的封装，表达应用程序定义的数据类型实例的编码规则以及描述远程调用和应答的协议和 SOAP 编订。

（3）WSDL 为服务者提供以 XML 格式描述的 WEB 服务请求的标准格式，经网络服务描述为能够进行消息交换的通信端点集合，以表达一个 Web 服务能做什么，他的位置在哪里，以及如何调用等信息。

（4）UDDL 规范描述了 Web 的概念，同时也定义了一种编程接口。通过 UDDL 提供的标准接口，企业可以发布自己的 Web 服务供其他企业调用和查询，业可以查询特地服务的描述信息，并动态的绑定到该服务上，通过 UDDL，Web 服务可以真正实现信息的“一次注册到处访问”。

6、从管理的角度看，SOA 有什么优点？

答：

（1）更易于维护；服务提供者和服务和服务使用者的松散耦合关系及对开放标准的采用确保了该特性的实现。

（2）更高的可用性；该特性在服务提供者和服务使用者的松散耦合关系上得以体现。使用者无需了解提供者的实现细节。

（3）更好的伸缩性；依靠服务设计、开发和部署所采用的架构模型实现伸缩性。服务提供者可以彼此独立调整，以满足服务需求。

7、在实际开发中，如何实现 Web 服务和 SOA 结构？

答：

（1）声明技术：J2EE 编程模型就是使用声明技术提供应用程序逻辑和中间件配置分离的一个例子。

（2）抽象：在某些情况下，SOA 基础结构中可以提供 API，以用于特定的用途。例如，SOA 基础结构可以提供错误报告和审核机制。在设计此类 API 时应非常小心，要注意

其易用性。我们应优先考虑声明技术，而不是对这些机制进行编程配置。同样，在标准 API 可用时，我们应通过这些标准 API 公开 SOA 基础结构功能，而不是采用自己开发编写的方式。

（3）代码生成：在无法避免代码复杂性的地方，可以使用代码生成技术。例如，Web 服务描述语言(Web Services Definition Language,WSDL)就可以为开发人员隐藏 SOAP、HTTP 和 JMS 的复杂细节。这是通过组合用 WSDL 表示的可由计算机处理的接口定义和可从 WSDL 生成相关调用代码的语言特定实现的工具来实现的。

（4）工具：在不可避免 SOA 基础结构的细节进入开发人员代码的情况下，我们可以通过使用合适的工具扩展开发环境来减少开发人员工作的复杂性。IBM Rational® Software Development Platform 产品所提供的基于 Eclipse 的环境可使用自定义插件、代码片段和用户指南轻松地进行扩展。

（5）模型驱动的开发：模型驱动的开发技术可以被视为前面两种方法的特定复杂组合，同时利用了工具和代码生成功能来简化开发体验。开发人员生成统一建模语言(Unified Modeling Language，UML）模型，此类模型可转换为相应的代码，其中包含利用 SOA 基础结构所必需的代码。

（6）总之，在定义面向服务的体系结构及其基础结构时，我们必须特别注意开发人员的需求。当为开发人员提供指南，以告知他们应如何开发或使用服务时，我们应该寻找可促进这些指导方针遵循的机制。SOA 内的控制对其成功甚为关键。

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：软件开发模型有演化模型、螺旋模型、喷泉模型、智能模型等。传统软件开发模型存在开发效率不高,不能很好地支持软件重用等缺点。在多个大中型软件项目的实践基础上,提出了基于体系结构的软件开发模(ABSD)。ABSD 模型把整个基于体系结构的软件过程划分为体系结构需求、设计、文档化、复审、实现、演化等 6 个子过程,讨论了各个子过程所要完成的工作,给出了 ABSD 模型在劳动和社会保险领域的一个应用实例。实践表明,采用 ABSD 模型进行软件项目开发,具有结构清晰、易于理解、可移植性强、重用粒度大等优点。

2、如何才能提高软件系统的可演化性。

答：构造性和演化性是软件的两个基本特性。软件进行渐变并达到所希望的形态就是软件演化，软件演化是由一系列复杂的变化活动组成。对软件变化的控制是软件开发者历来追求的目标。引起软件变化的原因是多方面的，如基本设施的改变，功能需求的增加，高性能算法的发现，技术环境因素的变化等。所以对软件变化甚至演化进行理解和控制显得比较复杂和困难。

3、如何才能提高软件系统的可演化性。

答：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 8 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）软件系统演化是一个反馈系统。

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？

答：

（1）通过系统的详细说明书，确定系统所采用的体系结构风格。

（2）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。

（3）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。

（4）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。

2、如何模型化系统的可靠性？

答：

（1）采用体系结构描述语言 ADL 对体系结构进行建模

（2）通过模拟方法执行复杂性分析

（3）通过 FMEA 和模拟运行执行严重性分析

（4）为构件和连接件开发其启发式风险因子

（5）建立用于风险评估的 CDG

3、软件体系结构风险分析有哪些基本步骤？

答：体系结构文档化，体系结构复审，体系结构实现，体系结构演化。

4、软件体系结构测试是什么？熟悉使用抽象化学机进行测试的方法。

答：

（1）软件体系结构测试要研究的对象是软件体系结构设计，并以此为基础产生高层次的测试用例集，以指导代码层的测试活动。他的目的是找出体系结构设计的错误和缺陷，产生指导代码测试的测试计划和测试用例。

（2）内容：构件端口行为与连接件约束是否一致、兼容，单元件的消息是否一致、可达，相关端口是否可连接，体系结构风格是否可满足。

（3）准则：测试覆盖所有的构件及各个构件的结构，包括：各个连接件的结构、构件之间的直接连接、构件之间的间接连接。

第九章

1 为什么要评估软件体系结构？

答：所谓软件体系结构的分析评估，就是事先通过代价低廉的评估活动来识别软件结构中存在的潜在风险，找出软件体系结构中影响系统质量的主要因素及改进措施，并在此基础上检验软件的质量需求是否在具体设计中得到实现，并预见未来软件质量。

软件体系结构在软件开发和管理中扮演者越来越重要的角色，软件体系结构设计对软件质量有着至关重要的影响，对此最终确保系统的质量有重要的意义。软件体系结构评估，是对系统的某些值的关心的属性进行评估和判断。评估的结果可用于确认潜在的风险，并检查设计阶段系统需求的质量，在系统被实际构造之前，预测其属性质量。

2、从哪些方面评估软件体系结构?

答：（1）性能是指系统的影响能力，即要经过多长时间才能对某个事件作出响应，或者在某段事件内系统所能处理的事件的个数。（2）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。（3）可用性是系统能够正常运行的时间比例。经常用两次故障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。安全性又可划分为机密性、完整性、不可否认性及可控性等特性。（5）可修改性是指能够快速的以较高的性能代价比对系统进行变更的能力。（6）功能性是系统所能完成所期望的工作的能力。（7）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。（8）可集成性是指系统能与其他系统协作的程度(9)互操作性是指与其他环境或者系统本身相互作用的能力。

3、ATAM 评估方法的基本步骤是什么？

答：https://max.book118.com/html/2017/0706/120735283.shtm

（1）ATAM（Architecture Tradeoff Analysis Method：架构权衡分析方法），是评价软件构架的一种综合全面的方法。这种方法不仅可以揭示出架构满足特定质量目标的情况，而且可以使我们更清楚地认识到质量目标之间的联系——即如何权衡诸多质量目标。

（2）ATAM 要求一下 3 个小组的参与和合作

① 评估小组：所要评价的架构的项目外部的小组，3-5 人组成；每个成员扮演大量特定角色：评估小组负责人、评估负责人、场景书记员、进展书记员、计时员、过程观察员、过程监督者、提问者。

② 项目决策者：对项目开发具有发言权，并有权要求进行某些改变，如项目管理人员、设计师、承担开发费用的客户。

③ 架构涉众：在架构中有既得利益的人，能够清晰阐述架构应该满足的具体质量属性目标，由 12-15 人组成。

（3）ATAM 评估分为 9 个步骤，即：

① ATAM 方法的描述：评估负责人向参加会议的相关人员介绍 ATAM 方法；在这一步，要对每个人解释参与的过程，并留出解答疑问的时间。

② 商业动机的陈述：项目决策者从商业角度，向相关人员介绍系统概况和主要商业动机，强调系统最重要的功能，阐述技术、管理、经济、政治方面的任何相关限制，介绍主要的相关人员，介绍体系结构的驱动因素（即促使形成该架构的主要质量属性目标）。

③ 架构表述：设计师在适合的层次上对架构进行详略适当的介绍；表述应该传达架构的本质，避开不太重要的方面；塑造软件架构的需求，与这些需求相关的可度量的量，用

以满足这些需求的任何现有标准/模型/方法；需要介绍的重要的架构信息包括：系统存在的预警，模块和分层视图，构件-连接件视图，与部署视图；介绍构架方法、模式和采用的战术，如它们实现了什么质量属性以及是如何实现的。

④ 对架构方法进行分类：评估小组已经充分了解了设计师在设计系统时所使用的的模式和战术，并确保为使用的每一个模式和方法进行了明确的命名；评估小组还应能够发现没有提及的方法和模式，对所用的模式进行记录和分类，作为后续分析的基础。

⑤ 生成质量属性效用树：通过效用树对数量属性目标进行详细清晰的阐述，效用树的根节点代表系统总体的“适宜性”，质量属性构成效用树的 2 级节点；继续对质量属性进行细化，得到第 3、4、…级节点；划分场景的优先级。

⑥ 分析架构方法：针对划分了优先级的质量需求，评估它们的匹配情况；设计师解释架构如何支持每个场景，小组成员考察设计师用来实现场景的架构方法；评估小组将相关架构决定编成文档，确定其有风险决策、无风险决策、敏感点、权衡点；确信所采用的的方法实例化知乎能够满足所要达到的质量属性要求。

⑦ 头脑风暴并确定场景优先级。

⑧ 分析架构方法。

// ⑤ 确定的场景主要是从架构设计师的角度看待系统的质量属性需求；⑦⑧ 是从相关人员的角度讨论场景，以确定设计师所想的与涉众所要的是否吻合。

⑨ 陈述结果：需要对在 ATAM 分析中所得到的各种信息进行归纳总结，并呈现给相关人员；在这一陈述中，评估负责人概要介绍 ATAM 评估的各个步骤和得到的各种信息，包括商业环境、塑造该架构的主要需求、约束条件等。最终得出的最重要的结果如下：文档化架构方法，若干场景及其优先级，效用树，有风险决策、无风险决策、敏感点、权衡点，与已编档的无风险决策。

4、详细了解一种体系结构风险分析方法。

【实例研究】

A 公司是一家相对较大的软件和硬件企业，专业从事网络设备的开发。从单一的产品开始现在，已经延伸到包括摄相服务器、扫描服务器、光盘服务器以及其他的存储服务器在内的产品。公司原来的产品都是一个一个地开发，每个软件组织一个项目组。为了适应快速变化的市场，降低开发成本，公司想引入产品线方法。然而，软件产品线开发涉及了一个软件开发组织的多个产品，选择了软件产品线意味着要承担由此带来的许多风险。所以，公司的 CTO 王总决定在弄清三个问题之后再做决定，首先就是本公司的业务范围是否适合使用产品线方法，其次是如何在原有产品的基础上建立产品线，最后是成功实施产品线的主要因素是什么？

[问题 1]

请用 100 字以内文字说明 A 公司是否适合采用产品线方法？为什么？

答：A 公司是适合采用产品线方法的。软件产品线体系结构是指一个软件开发组织为相关应用或产品建立的公共体系结构，此外同领域模型一样，软件产品线体系结构中可以分为共性部分和个性部分；产品线体系结构是产品线核心资源早期和主要部分，在产品线的生命周期里，产品线体系结构应该保持相对小和缓慢的变化以便在生命周期中尽量保持一致。

[问题 2]

请用 200 字以内文字说明如何在原有产品的基础上建立产品线？

答　　：

	演化方式	革命方式
基于现有产品集	基于现有产品体系结构开发产品线的体系结构 经演化现有构件的文件一次开发一个产品线构件	产品线核心资源的开发基于现有产品集的需求和可预测的、将来需求的超集
全新产品线	产品线核心资源随产品新成员的需求而演化	开发满足所有预期产品线成员的需求的产品线核心资源

[问题 3]

请用 150 字以内文字说明成功实施产品线的主要因素是什么？

答：首先每个产品都由来自公共资产库中的组件组成，然后按照预先定义的变化机制，如参数化或继承，对这些组件进行必要的裁剪，添加任何必须的新组件，根据一个产品线范围内的公共架构来组装这些组件。于是，构建一个产品（系统）主要工作是组装和繁衍，而不是创造；主要的活动是集成而不是编程。每条软件产品线都有一个预先定义的指南或计划，用来定义确切的产品构建方法。