**系统架构设计师考试论文解析**

**2013——2018年**

**目 录**

[2013年论文 1](#_Toc32257)

[论软件架构建模技术与应用 1](#_Toc30429)

[论企业应用系统的分层架构风格 1](#_Toc3758)

[论软件可靠性设计技术的应用 2](#_Toc3775)

[论分布式存储系统架构设计 2](#_Toc3056)

[2014年论文 3](#_Toc24673)

[论软件需求管理 3](#_Toc29292)

[论非功能性需求对企业应用架构设计的影响 4](#_Toc29667)

[论软件的可靠性设计 5](#_Toc22441)

[论网络安全体系设计 5](#_Toc1798)

[2015年论文 6](#_Toc28497)

[论应用服务器基础软件 6](#_Toc32759)

[论软件系统架构风格 6](#_Toc11109)

[论面向服务架构及应用 7](#_Toc9004)

[论企业集成平台的技术与应用 8](#_Toc22284)

[2016年论文 9](#_Toc3453)

[论软件架构评估（质量属性） 9](#_Toc18666)

[论软件设计模式 10](#_Toc15256)

[论数据访问层设计技术及其应用 11](#_Toc28959)

[论微服务架构及其应用 11](#_Toc29313)

[2017年论文 11](#_Toc26727)

[论软件系统建模方法及应用 11](#_Toc14535)

[论软件架构风格 12](#_Toc4323)

[论无服务器架构及其应用 13](#_Toc20103)

[论软件质量保证 14](#_Toc7839)

[2018年论文 15](#_Toc17025)

[论软件开发过程RUP及其应用 15](#_Toc19661)

[论软件体系结构演化 15](#_Toc29890)

[论面向服务架构设计及应用 16](#_Toc28233)

[论NoSQL数据库技术及其应用 17](#_Toc7089)

###### 

###### 2013年论文

论软件架构建模技术与应用

一、简要叙述所参与管理和开发的软件项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、简要叙述“4+1”视图模型的主要内容。

1、“4+1”视图模型从5个不同的视角来描述软件架构，每个视图只关心系统的1个侧面，5个视图结合在一起才能反映系统的软件结构的全部内容。这5个不同的视角包括逻辑视图、开发视图、进程视图、物理视图和场景。

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。 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OO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。

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。

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。进程视图强调并发性、分布性、系统集成性和容错能力，以及逻辑视图中的功能抽象如何适应进程结构等，它也定义了逻辑视图中的各个类的操作具体是在哪一个线程中被执行。 进程视图可以描述成多层抽象，每个级别分别关注不同的方面。

物理视图。物理视图在UML中被称为部署视图，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。

场景。场景可以看作是那些重要系统活动的抽象，它使4个视图有机联系起来。场景对应UML中的用例视图。

2、结合实际项目，详细说明项目软件架构的内容。这部分内容应包括：在设计软件架构时，分别使用了 “4+1”视图中的哪些视图，每个视图中包含的模型有哪些等。

三、说明该项目软件架构的实施效果，分析其是否满足了项目的需求并说明原因。

论企业应用系统的分层架构风格

一、简要描述所参与管理和开发的企业应用系统建设项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、需要结合项目实际情况指出所开发的应用系统的总体架构，特别是架构的层次关系。分层架构设计是一种常见的架构设计方法，能够有效简化设计，使设计的系统结构清晰，便于提高复用能力和产品维护能力。一般来说，企业应用系统的架构可以分为 表现层、中间层和持久层三个层次。

表现层。表现层主要负责接收用户的请求，对用户的输入、输出进行检查与控 制，处理客户端的一些动作，包括控制页面跳转等，并向用户呈现最终的结果信息。表现层主要采用MVC结构来实现。控制器负责接收用户的请求，并决定应该调用哪个模型来处理；然后，模型根据用户请求调用中间层进行相应的业务逻辑处理，并返回数据；最后，控制器调用相应的视图来格式化模型返回的数据，并通过视图呈现给用户。

中间层。中间层主要包括业务逻辑层组件、业务逻辑层工作流、业务逻辑层实体和业务逻辑层框架四个方面。业务逻辑层组件分为接口和实现类两个部分，接口用于定义业务逻辑组件，定义业务逻辑组件必须实现的方法。通常按模块来设计业务逻辑组件，每个模块设计为一个业务逻辑组件，并且每个业务逻辑组件以多个DAO组件作为基础，从而实现对外提供系统的业务逻辑服务。业务逻辑层工作流能够实现在多个参与者之间按照某种预定义的规则传递文档、信息或任务的过程自动进行，从而实现某个预期的业务目标，或者促进此目标的实现。业务逻辑层实体提供对业务数据及相关功能的状态编程访问，业务逻辑层实体数据可以使用具有复杂架构的数据来构建，这种数据通常来自数据库中的多个相关表，业务逻辑层实体数据可以作为业务过程的部分I/O参数传递，业务逻辑层的实体是可序列化的，以保持它们的当前状态。业务逻辑层是实现系统功能的核心组件，采用容器的形式，便于系统功能的开发、代码重用和管理。

持久层。持久层主要负责数据的持久化存储，主要负责将业务数据存储在文件、 数据库等持久化存储介质中。持久层的主要功能是为业务逻辑提供透明的数据访问、持久化、加载等能力。

三、考生需要结合项目实际情况，举例说明在设计表现层、中间层和持久层时需要考虑的主要问题，例如：在持久层设计时需要考虑MVC模型中的模型、视图和控制器分别对应哪些组件：在中间层设计时需要考虑框架与业务组件之间的关系；在持久层设计时需要考虑如何支持对多种类型数据的透明访问。

论软件可靠性设计技术的应用

一、概要论述你参与管理和开发的信息系统项目以及你在其中所承担的主要工作。

二、结合项目实际，论述你在进行软件可靠性设计时遵循的基本原则，你所采用的具体可靠性设计技术的基本内容。

可靠性设计需要遵循的原则有：

1、软件可靠性设计是软件设计的一部分，必须在软件的总体设计框架中使用，并且不能与其他设计原则相冲突。

2、软件可靠性设计在满足提高软件质量要求的前提下，以提高和保障软件可靠性为最终目标。

3、软件可靠性设计应确定软件的可靠性目标，不能无限扩大，并且排在功能、用户需求、开发费用之后考虑。

常见的可靠性设计技术有容错设计、检错设计、降低复杂度设计等技术。

容错设计技术：对于软件失效后果特别严重的场合，如飞机的飞行控制系统、空中交通管制系统等，采用容错设计技术。常见的容错设计技术有三种：恢复块设计、N版本程序设计和冗余设计。

恢复块设计：选择一组软件操作作为容错设计单元，把普通的程序块变成恢复块。一个恢复块包含有若千个功能相同、设计差异的程序块文本，一个运行文本，多个备份文本，构成“动态冗余”，一旦运行文本出现故障，则用备份文本替换。软件容错的恢复块方法就是使软件包含有一系列恢复块。

N版本程序设计：N版本程序的核心是通过设计出多个模块或不同版本，对于相同初始条件和相同输入的操作结果，实现多数表决，防止其中某一软件模块/版本的故障提供错误的服务，以实现软件容错。

冗余设计：在一套完整的软件系统之外，设计一种不同路径、不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。缺点是费用和资源的消耗会有所增加。

检错技术：在软件系统中，无需在线容错的地方，或不能采用冗余设计技术的部分，如果对可靠性要求较高，故障有可能导致严重的后果时，一般采用检错技术，在软件出现故障后能及时发现并报警，其缺点是不能自动解决故障。

降低复杂度设计：软件复杂性与软件可靠性有着密切的关系，是产生软件缺陷的重要根源。在设计时考虑降低软件的复杂性，是提高软件可靠性的有效方法。降低复杂度设计的思想是在保证实现软件功能的基础上，简化软件结构，缩短程序代码，优化软件数据流向，降低软件复杂度，从而提高软件可靠性。

（结合实际工作，具体解释遵循的原则和采用的一种或多种可靠性设计技术）

三、阐述你在具体的可靠性设计工作中，为了分析影响软件可靠性的主要因素，所采用的可靠性分析方法。

在软件可靠性设计之前和软件可靠性设计过程中，都需要采用软件可靠性分析和预测方法，来确定当前系统中的主要可靠性因素和目标。常见的软件可靠性分析方法包括故障树分析方法、失效模式与效应分析方法等。

故障树分析方法：一种自顶向下的软件可靠性分析方法，即从软件系统不希望发生的事件（顶事件)，特别是对人员和设备的安全及可靠性产生重大影响的事件开始，向下逐步追查导致顶事件发生的原因，直至基本事件（底事件)，从而确定软件故障原因的各种可能组合方式和（或）发生概率。基本的步骤是软件故障树的建立、定性分析和定量分析。

失效模式与效应分析方法：在软件开发阶段的早期，通过识别软件失效模式，分析造成的后果，研究分析各种失效模式产生的原因，寻找消除和减少其有害后果的方法，以便尽早发现潜在的问题，并采取相应的措施,从而提髙软件的可靠性和安全性。SFMEA的分析对象可以是开发早期阶段的高层次的子系统、部件，也可以是详细设计阶段的单元、模块。对于不同的分析对象，其软件失效模式是不同的，采用的SFMEA分析方法也不同，前者采用系统级分析方法（systemFMEA)，后者为详细级分析方法（detailedFMEA)。其基本的步骤是系统定义、软件失效模式分析、软件失效原因分析、软件失效影响分析、改进措施分析。

（结合实际工作，具体阐述自己所采用的一种或多种可靠性分析方法）

论分布式存储系统架构设计

一、简要描述所参与分析和开发的分布式存储系统项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、说明在分布式存储系统架构设计中所使用的四种主要分布式存储技术，并根据考生所参与的实际项目，详细叙述所选用的一种分布式存储技术，并说明选择该技术的原因和实施效果。

在分布式存储系统架构设计中所使用的分布式存储技术主要包括四类：

1、集群存储技术。集群存储系统是指架构在一个可扩充服务器集群中的文件系统，用户不需要考虑文件是存储在集群中什么位置，仅仅需要使用统一的界面就可以访问文件资源。当负载增加时，只需在服务器集群中增加新的服务器就可以提高文件系统的性能。集群存储系统能够保留传统的文件存储系统的语义，增加了集群存储系统必须的机制，可以向用户提供高可靠性、高性能、可扩充的文件存储服务。

2、分布式文件系统。分布式文件系统是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连。分布式文件系统的设计基于客户机/服务器模式。一个典型的网络可能包括多个供多用户访问的服务器。另外，对等特性允许一些系统扮演客户机和服务器的双重角色。分布式文件系统以透明方式链接文件服务器和共享文件夹，然后将其映射到单个层次结构，以便可以从一个位置对其进行访问，而实际上数据却分布在不同的位置。用户不必再转至网络上的多个位置以査找所需的信息。

3、网络存储技术。网络存储系统就是将“存储”和“网络”结合起来，通过网络连接各存储设备，实现存储设备之间、存储设备和服务器之间的数据在网络上的高性能传输。为了充分利用资源，减少投资，存储作为构成计算机系统的主要架构之一，就不再仅仅担负附加设备的角色，逐步成为独立的系统。利用网络将此独立的系统和传统的用户设备连接，使其以髙速、稳定的数据存储单元存在。用户可以方便地使用浏览器等客户端进行访问和管理。

4、P2P网络存储技术。P2P网络存储技术的应用使得内容不是存在几个主要的服务器上，而是存在所有用户的个人电脑上。这就为网络存储提供了可能性，可以将网络中的剩余存储空间利用起来，实现网络存储。人们对存储容量的需求是无止境的，提高存储能力的方法有更换能力更强的存储器，另外就是把多个存储器用某种方式连接在一起，实现网络并行存储。相对于现有的网络存储系统而言，应用P2P技术将会有更大的优势。P2P技术的主体就是网络中Peer，也就是各个客户机，数量是很大的，这些客户机的空闲存储空间是很多的，把这些空间利用起来实现网络存储。

三、冗余是提高分布式存储系统可靠性的主要方法，冗余的存储结构可以保证部分服务器失效时，数据服务仍可正常访问。常用的冗余技术包括：数据备份，数据分割，门限方案，纠错编码和纠删编码等。考生根据所参与的实际项目指出采用了何种冗余技术，并说明其原因和实施效果。

###### 2014年论文

论软件需求管理

本文第一部分应花400-600字的篇幅进行项目简介，涉及项目背景、规模、人员、作者的角色，开发的系统有什么样的一些功能，大体的设计。

接下来的主体部分中，着重描述的，应是问题3，对于问题2只需要花200-400字的篇幅大致介绍概念层次的内容。

在对问题3进行论述时，要注意选问题2中的一些活动来论述，其中2个主题是比较好展开的，分别为：变更控制与需求跟踪。

1、变更控制：

变更控制的工作程序依次为：提出与接受变更申请、对变更初审、变更方案论证、项目变更控制委员会审查、发出变更通知并开始实施、变更实施的监控、变更效果的评估、判断发生变更后的项目是否已纳入正常轨道。

（1）提出与接受变更申请。提出变更申请应当及时以正式方式进行，并留下书面记录。变更的提出可以是各种形式，但在评估前应以书面形式提出。

（3）对变更的初审。变更初审的目的是为了对变更提出方施加影响，确认变更的必要性，确保变更是有价值的；进行格式校验，完整性较验，确保评估所需信息准备充分；在干系人间就提出供评估的变更信息达成共识。

（3）变更方案论证。变更方案的主要作用，首先是对变更请求是否可实现进行论证，如果可能实现，则将变更请求由技术要求转化为资源需求，以供CCB决策。常见的方案内容包括技术评估和经济评估，前者评估需求如何转化为成果，后者评估价值和风险。

（4）项目变更控制委员会审查。审查过程由项目所有者据变更申请及评估方案，决定是否批准变更。审查通常是文档会签形式，重大的变更审查可以包括正式会议形式。审查过程应注意分工，项目投资人虽有最终的决策权，但通常在专业技术上并非强项。所以应当在评审过程中将专业评审、经济评审分开，对涉及项目目标和交付成果的变更，客户的意见应放在核心位置。

（5）发出变更通知并开始实施。评审通过，意味着项目基准的调整，同时确保变更方案中的资源需求及时到位。项目基准的调整，包括项目目标的确认，最终成果、工作内容和资源、进度计划的调整。需要强调的是，变更通知后，不只是包括实施项目基准的调整，更要明确项目的交付日期、成果对相关干系人的影响。如变更造成交付期的调整，应在变更确认时发布，而非在交付前公布。

（6）变更实施的监控。要监控的，除了调整过的项目基准中所涉及变更的内容外，还应当对项目的整体基准是否反映项目实施情况负责。通过监控行动，确保项目的整体实施工作是受控的。通常由项目经理负责项目基准的监控，管理委员会监控变更明确的主要成果、进度里程碑等，可以委托监理单位承担监控职责。

（7）变更效果的评估。变更评估首要的评估依据，是项目基准，可需结合变更的初衷来看要达到的目的是否已达成，以及评估变更方案中的技术论证、经济论证内容与实施过程的差距并推进解决。

（8）判断发生变更后的项目是否已纳入正常轨道。项目基准调整后，需要确认的是相应的资源配置和人员是否及时到位，更需多加关注。之后对项目的整体监控应按新的项目基准进行，当确认新的项目基准已经生效则按正常的项目实施流程进行。

2、需求跟踪

根据国家标准GB/T 8567-2006，SRS中的每个软件配置项的需求到其涉及的系统（或子系统）需求都要具有双向可追踪性。所谓双向跟踪，包括正向跟踪和反向跟踪，正向跟踪是指检查SRS中的每个需求是否都能在后继工作成果中找到对应点；反向跟踪也称为逆向跟踪，是指检查设计文档、代码、测试用例等工作成果是否都能在SRS中找到出处。

论非功能性需求对企业应用架构设计的影响

本文第一部分应花400-600字的篇幅进行项目简介，涉及项目背景、规模、人员、作者的角色，开发的系统有什么样的一些功能，大体的设计。

接下来的内容是比较好组织的，因为非功能性需求的范围非常之广，只要作者在论述之前，表明这是非功能需求，然后写关于如何应对这种需求即可。这种需求可以是以下方面的内容：

1、性能

性能（performance）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，或者在某段时间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。性能测试经常要使用基准测试程序（用以测量性能指标的特定事务集或工作量环境）。

2、可靠性

可靠性（reliability）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。可靠性通常用平均失效等待时间（Mean Time To Failure，简称MTTF）和平均失效间隔时间（Mean Time Between Failure，简称MTBF）来衡量。在失效率为常数和修复时间很短的情况下，MTTF和MTBF几乎相等。

3、可用性

可用性（availability）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（security）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。

5、可修改性

可修改性（modifiability）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。

6、功能性

功能性（functionality）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（changeability）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、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（interoperation），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。

论软件的可靠性设计

可靠性设计是架构考试中反复考查的一个知识点。

本文第一部分应花400-600字的篇幅进行项目简介，涉及项目背景、规模、人员、作者的角色，开发的系统有什么样的一些功能，大体的设计。

接下来介绍主流的软件可靠性设计技术，常见的可靠性设计技术有容错设计、检错设计、降低复杂度设计等技术。

容错设计技术：对于软件失效后果特别严重的场合，如飞机的飞行控制系统、空中交通管制系统等，采用容错设计技术。常见的容错设计技术有三种：恢复块设计、N版本程序设计和冗余设计。

恢复块设计：选择一组软件操作作为容错设计单元，把普通的程序块变成恢复块。一个恢复块包含有若千个功能相同、设计差异的程序块文本，一个运行文本，多个备份文本，构成“动态冗余”，一旦运行文本出现故障，则用备份文本替换。软件容错的恢复块方法就是使软件包含有一系列恢复块。

N版本程序设计：N版本程序的核心是通过设计出多个模块或不同版本，对于相同初始条件和相同输入的操作结果，实现多数表决，防止其中某一软件模块/版本的故障提供错误的服务，以实现软件容错。

冗余设计：在一套完整的软件系统之外，设计一种不同路径、不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。缺点是费用和资源的消耗会有所增加。

检错技术：在软件系统中，无需在线容错的地方，或不能采用冗余设计技术的部分，如果对可靠性要求较高，故障有可能导致严重的后果时，一般采用检错技术，在软件出现故障后能及时发现并报警，其缺点是不能自动解决故障。

降低复杂度设计：软件复杂性与软件可靠性有着密切的关系，是产生软件缺陷的重要根源。在设计时考虑降低软件的复杂性，是提高软件可靠性的有效方法。降低复杂度设计的思想是在保证实现软件功能的基础上，简化软件结构，缩短程序代码，优化软件数据流向，降低软件复杂度，从而提高软件可靠性。

注意在结合项目进行论述时，只要论述其中的2-3个方面即可。

论网络安全体系设计

本文第一部分应花400-600字的篇幅进行项目简介，涉及项目背景、规模、人员、作者的角色，开发的系统有什么样的一些功能，大体的设计。

完成本论文，需要了解标准中定义的五类安全服务。

五类安全服务：

1、鉴别服务：鉴别参与通信的对等实体和数据源的合法性。对等实体鉴别和数据源鉴别：由第N层实体提供，可向第N+1层实体证实。安全服务由第N层实体提供，可向第N+1层实体证实数据源。

2、访问控制服务：能够阻止未经授权而利用通过OSI模型的可访问资源。

3、数据保密性服务对数据提供保护，防止数据未经授权而被泄漏，防止在系统之间交换数据时被截取。它还内含四项服务：连接保密性、无连接保密性、选择字段保密性、通信业务流保密性。

4、数据完整性服务：防止系统之间交换数据，非法修改数据或丢失数据。数据完整性可分四类：实体完整性、域完整性、参照完整性、用户定义的完整性。

禁止否认服务：阻止通信双方否认发送和接收数据的行为。

###### 2015年论文

论应用服务器基础软件

论述并分析应用服务器在软件设计、开发、部署、运行和管理阶段，应该提供哪些核心功能？

写作要点：

一、简要描述所参与分析和开发的软件系统开发项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、论述和分析应用服务器应该具备的核心功能。

应用服务器是应用设计、开发、部署、运行、管理、维护的平台。应用服务器既是应用开发的平台，包括表示层、应用层和数据层的设计模式和编程环境；同时又是多层结构应用的部署、运行平台，对多层结构应用进行配置、启动、监控、调整，并在开发的不同阶段提供不同的功能。

1. 设计阶段，应用服务器完成底层通信、服务，并屏蔽掉复杂的底层技术细节，向用户提供结构简单、功能完善的编程接口，让用户可以专心于商务逻辑的设计。

2. 开发阶段，应用服务器提供了完全开放的编程语言和应用接口，同时也提供快速开发的工具和手段，帮助用户提高开发效率。

3. 部署阶段，应用服务器提供了对多种网络环境的支持，帮助用户在复杂的网络环境中配置系统参数，发挥系统最大性能。

4. 运行阶段，应用服务器基于开发技术标准，提供了系统的运行环境，提供了系统的名字解析、路由选择、负载平衡、事务控制等服务，并提供系统容错、修复、迁移、升级扩展等功能。

5. 管理阶段，应用服务器提供图形化界面来管理整个系统的资源，而且系统在运行期间也能动态监控和管理。

三、针对作者实际参与的软件系统开发项目，说明所采用的应用服务器，并描述该应用服务器在开发、部署和运行阶段的实际应用效果。

论软件系统架构风格

写作要点：

一、简要叙述所参与管理和开发的软件项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、常见的架构风格5大类，至少选2-3个类进行说明。

Garlan和Shaw将软件架构风格分为五大类，数据流风格、调用/返回风格、独立构件风格、虚拟机风格和仓库风格。其中：

（1）数据流风格包括批处理序列架构风格和管道/过滤器架构风格；

（2）调用/返回风格包括主程序/子程序架构风格、数据抽象和面向对象架构风格和层次结构架构风格；

（3）独立构件风格包括进程通信架构风格和事件驱动的架构风格；

（4）虚拟机风格包括解释器架构风格和基于规则的系统；

（5）仓库风格包括数据库架构风格和黑板架构风格。

其他的还有特定领域软件架构、状态转移等以及分布式处理等。其中分布式架构风格中有客户机/服务器风格、浏览器/服务器风格、CORBA、DCOM、EJB。

每一种具体的软件结构风格的模型如下：

1．数据流风格包括批处理序列和管道/过滤器架构风格。

（1）批处理序列架构风格。组件为一系列固定顺序的计算单元，组件间只通过数据传递交互。每个处理步骤是一个独立的程序，每一步必须在前一步结束后才能开始，数据必须是完整的，以整体的方式传递。

（2）管道/过滤器架构风格。每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流，经过处理，产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，包括通过计算和增加信息丰富数据，通过浓缩和删除精炼数据，通过改变记录方式转化数据，递增地转化数据等。在输入被完全消费之前，输出便产生了。这里构件被称为过滤器，连接件就是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。

2．调用/返回风格包括主程序/子程序架构风格、数据抽象和面向对象架构风格以及层次结构架构风格

（1）主程序/子程序架构风格。单线程控制，把问题划分为若干处理步骤，构件即为主程序和子程序。子程序通常可合成为模块。过程调用作为交互机制，即充当连接件。调用关系具有层次性，其语义逻辑表现为子程序的正确性取决于它调用的子程序的正确性。

（2）数据抽象和面向对象架构风格。这种风格的构件是对象。对象是抽象数据类型的实例。在抽象数据类型中，数据的表示和它们的相应操作被封装起来。对象的行为体现在其接受和请求的动作。连接件即是对象间交互的方式，对象是通过函数和过程的调用来交互的。对象具有封装性，一个对象的改变不会影响其他对象。对象拥有状态和操作，也有责任维护状态。这种结构风格中包含有封装、交互、多态、集成、重用等特征。

（3）层次结构架构风格。层次系统组织成一个层次结构。构件在一些层实现了虚拟机。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。这个风格的特点是每层为上一层提供服务，使用下一层的服务，只能见到与自己邻接的层。大的问题分解为若干个渐进的小问题，逐步解决，隐藏了很多复杂度。修改一层，最多影响两层，而通常只能影响上层。上层必须知道下层的身份，不能调整层次之间的顺序。

3．独立构件风格包括进程通信架构风格和事件驱动的架构风格

（1）进程通信架构风格。构件是独立的过程，连接件是消息传递。这种风格的特点是构件通常是命名过程，消息传递的方式可以是点对点、异步和同步方式、以及远过程调用等

（2）事件驱动的架构风格。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。系统中的其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程。一个事件的触发就导致了另一个模块中的过程的调用。这种风格中的构件是非命名的过程，它们之间交互的连接件往往是以过程之间的隐式调用（Implicit Invocation）来实现的。基于事件的隐式调用风格的主要优点是为软件重用提供了强大的支持，为构件的维护和演化带来了方便，其缺点是构件放弃了对系统计算的控制。

4．虚拟机风格包括解释器架构风格和基于规则的系统

（1）解释器架构风格。一个解释器通常包括完成解释工作的解释引擎，一个包含将被解释的代码的存储区，一个记录解释引擎当前工作状态的数据结构，以及一个记录源代码被解释执行的进度的数据结构。具有解释器风格的软件中含有一个虚拟机，可以仿真硬件的执行过程和一些关键应用。其缺点是执行效率较低。

（2）基于规则的系统。基于规则的系统包括规则集、规则解释器、规则/数据选择器以及工作内存。

5．仓库风格包括数据库架构风格和黑板架构风格

（1）数据库架构风格。数据库架构是库风格最常见的形式。构件主要有两大类，一个是中央共享数据源，保存当前系统的数据状态，另一个是多个独立处理元素，处理元素对数据元素进行操作。

（2）黑板架构风格。黑板架构包括知识源、黑板、控制三部分。知识源包括若干独立计算的不同单元，提供解决问题的知识，知识源响应黑板上的变化，也只修改黑板。黑板是一个全局数据库，包含解域的全部状态，是知识源互相作用的唯一媒介。知识源响应是通过黑板状态的变化来控制。黑板通常应用在对于解决问题没有确定性算法的系统中，例如信号处理、问题规划、编译器优化等软件系统的设计中。

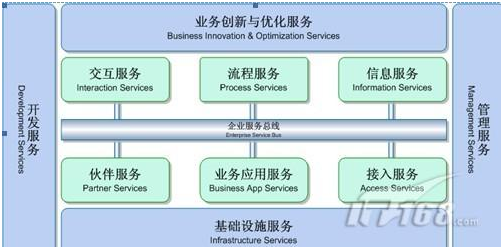
三、结合项目的实际状况，指出在架构设计时选择使用软件架构风格的情况，包括选择的依据、如何做的，要给出实际的效果及分析。

论面向服务架构及应用

写作要点：

一、按题目要求介绍作者参与的项目基本信息。

二、SOA技术参考架构中包含的服务类别包括：



1、开发服务(Development Services)用于实现新开发的组件以及重用基础架构的能力。

　　2、业务创新优化服务(Business Innovation &amp; Optimization Services)用于从IT和业务两个层面来监控和管理运行情况。

　　3、管理服务(Management Services)包括对服务、应用和资源的管理和保护能力，如通过负载均衡来有效的分配系统计算资源。

　　SOA解决方案中的很多服务都是由已有应用系统提供的，接入服务(Access Services)提供访问已有应用或遗留系统的能力，同时提供已有应用、打包应用程序与ESB之间的桥接能力，将已有系统中的功能和信息转化为服务。

　　4、业务应用服务(Business App Services)指那些通过新的计算平台JavaEE来实现的新应用，它们所实现的功能和信息也都转化为服务提供出来。

　　在业务流程需要与外部的合作伙伴、供应商交互的情况下，伙伴服务(Partner Services)提供文档、协议以及伙伴管理的能力，比如说，可以提供企业边界处不同安全级别差异的转换。

　　5、信息服务(Information Services)是那些跟信息(而不是活动)有关系的服务，比如将多个系统中异构的数据，聚合、转换为业务需要的统一整齐的业务数据对象来访问。信息服务通过联合、复制和转换来解决基于不同实现方式的不同数据源之间的数据共享难题。

　　6、流程服务(Process Services)是指把多个服务聚合成为一个服务流程对应业务过程的服务，这种复合服务通常是长时间运行的过程。流程服务提供服务控制能力，将多个服务串起来实现一个业务流程。

7、交互服务(Interaction Service)一方面将人的活动，通过人机交互以服务的方式出现在整个业务过程中，作为流程服务)中的一部分;另一方面将IT的功能和数据传递给最终用户，并满足用户特定的使用习惯。

三、第3个问题是题目要重点描述的内容，要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。同时文章收尾要对效果进行评价。

论企业集成平台的技术与应用

一、按题目要求介绍作者参与的项目基本信息。

二、企业信息集成是解决“孤岛”问题的需要，技术发展的同时也推动了集成架构等相关的研究。企业集成平台的核心是企业集成架构，包括信息、过程、应用集成的架构。

企业集成的关键应用技术可从两个大的方面来选择技术进行论述，即：数据交换格式和分布式应用集成基础框架。

1、数据交换格式

（1）EDI

EDI（Electronic Data Interchange，电子数据交换）是一种利用计算机进行商务处理的方法，它将贸易、运输、保险、银行和海关等行业的信息，用一种国际公认的标准格式，通过计算机通信网络，供有关部门、公司与企业之间进行数据交换与处理，并完成以贸易为中心的全部业务过程。

EDI格式处理的目的是将在功效上与纸介质文件等同的电子表单用统一的（或标准的）格式进行表示，以保证各个独立开发的计算机应用间能够实现表单数据共享与集成。用于描述电子表单格式的标准称为EDI格式标准或EDI标准，目前广泛使用的EDI格式标准主要有两个UN/EDIFACT和ANSIX12，它们分别由联合国欧洲经济委员会（UN/ECE）和美国国家标准化协会（ANSI）制定。

（2）XML

XML（Extensible Markup Language，可扩展标记语言），它是国际组织W3C制定的一个面向各类信息的数据存储工具和可配置载体的开放式标准。提出XML的目的是为了更好地适应Web应用的需求，解决HTML在表达能力、可扩展性和交互性等方面的缺陷。XML是通过对SGML标准进行简化而形成的元标记语言，具有语法清晰简单和结构无歧义等优点。它利用一套定义标记的规则将文件的内容和外观进行分离，实现了XML文档的可延伸性及自我描述特性，从而使各种业务信息可以在全球信息网或企业间的应用系统中传递、处理及储存。这里需要指出的是，虽然XML称为可扩展标记语言，但它本身并不是一种标记语言，而是一种创建、设计和使用标记语言的根规则集，是一种创建标记语言（如HTML）的元语言。

（3）STEP

STEP标准（Standard for the Exchange of Product Model Data）是一个描述如何表达和交换数字化产品信息的ISO标准（ISO10303），其目的是提供一种不依赖于具体系统的中性模型和机制，并将其用来描述整个生命周期内的产品数据。

2、分布式应用集成基础框架

（1）CORBA

CORBA的全称是公共对象请求代理体系结构（Common Object Request Broker Architecture），它是对象管理组织（OMG）为解决分布式处理环境中硬件和软件系统的互连而提出的一种标准的面向对象应用程序体系规范。

（2）COM +

COM +是Microsoft公司基于Windows平台的一个分布式企业应用模型，它与Windows操作系统紧密结合，是沿着DDE-OLE-OLE2-COM-DOOM-COM＋的路线发展而来。目前COM、DCOM和COM +应用比较广泛。

（3）Web Service

Web Service（Web服务）是指服务提供者将应用作为服务部署在Web上，通过使用Web服务描述语言（WSDL）来描述特定Web服务提供的功能。服务请求者在需要一种Web服务时，可以通过Internet，在Web服务的注册机构中查找分布在Web站点上的Web服务，并自动实现与服务的绑定，完成数据交换，在这个过程中无需人工干预。由于Web服务的系统架构和实现技术基本上基于已有的技术，因此，Web服务可以看成是现有应用面向Internet的一个延伸。

三、第3个问题要根据项目的实际情况来写自己是怎么做的，遇到什么样的问题，如何解决的。同时文章收尾要对效果进行评价。

###### 2016年论文

论软件架构评估（质量属性）

本题内容按模拟题中的“论基于场景的软件体系结构评估方法”组织内容即可，因为目前常用的架构评估方法，均为基于场景的评估方法。

一、首先用400-600字的篇幅简要叙述作者参与开发的软件系统的概要和所担任的工作。

二、架构所关注的质量属性主要包括：性能、可用性、可修改性、安全性。

1、性能

性能（performance）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，或者在某段时间内系统所能处理的事件的个数。

2、可用性

可用性（availability）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。

3、安全性

安全性（security）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。

4、可修改性

可修改性（modifiability）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。

三、架构评估方法主要从SAAM与ATAM中选择。

1、SAAM评估方法

SAAM的分析和评估目的、评估参与者、评估活动或过程以及评估结果说明如下。

（1）评估目的

SAAM (Scenario-based Architecture Analysis Method)目的是验证基本的体系结构假设和原则，评估体系结构固有的风险。SAAM 指导对体系结构的检查，使其主要关注潜在的问题点，如需求冲突。SAAM不仅能够评估体系结构对于特定系统需求的使用能力，也能被用来比较不同的体系结构。

（2）评估参与者

风险承担者、记录人员、软件体系结构设计师

（3）评估活动或过程

SAAM分析评估体系结构的过程包括六个步骤，即形成场景、描述体系结构、场景的分类和优先级确定、间接场景的单个评估、场景相互作用的评估、总体评估。

（4）评估结果

SAAM评估的主要有形输出包括：

1）把代表了未来可能做的更改的场景与构架对应起来，显现出构架中未来可能会表现出较高复杂性的地方，并对每个这样的更改的预期工作量做出评估。

2）理解系统的功能，对多个构架所支持的功能和数量进行比较。

如果所评估的是一个框架，SAAM评估将指明框架中未能满足其修改性需求的地方，有时还会指出一种效果更好的设计。SAAM评估也能对两个或者三个备选构架进行比较，明确其中那一个能够较好地满足质量属性需求，而且做的更改较少、不会在未来导致太多的复杂的问题。

2、ATAM评估方法

ATAM的分析和评估目的、评估参与者、评估活动或过程以及评估结果说明如下。

（1）评估目的

ATAM（Architecture Tradeoff Analysis Method ），即构架权衡分析方法的评估目的是依据系统质量属性和商业需求评估设计决策的结果。ATAM希望揭示出构架满足特定质量目标的情况，使我们更清楚地认识到质量目标之间的联系，即如何权衡多个质量目标。

（2）评估参与者

1）评估小组。该小组是所评估构架项目外部的小组，通常由3~5人组成。该小组的每个成员都要扮演大量的特定角色。他们可能是开发组织内部的，也可能是外部的。

2）项目决策者，对开发项目具有发言权，并有权要求进行某些改变，他们包括项目管理人员，重要的客户代表，构架设计师等。

3）构架涉众（stakeholders）。包括关键模块开发人员、测试人员、用户等。

（3）评估活动或过程

整个ATAM评估过程包括九个步骤，按其编号顺序分别是描述ATAM方法、描述商业动机、描述体系结构、确定体系结构方法、生成质量属性效用树、分析体系结构方法、讨论和分级场景、描述评估结果。

论软件设计模式

本题为模拟题原题，具体写作要求为：

一、首先用400-600字的篇幅简要叙述作者参与开发的软件系统的概要和所担任的工作。

二、设计模式的基本分类：

· 创建型模式。创建型模式抽象了实例化过程，它们帮助一个系统独立于创建、组合和表示它的那些对象。创建型模式包括工厂方法、抽象工厂、生成器、原型、单例模式等。

· 结构型模式。结构型模式涉及到如何组合类和对象以获得更大的结构。结构型模式包括适配器、桥接、组成、装饰、外观、享元、代理等。

· 行为模式。行为模式涉及到算法和对象间职责的分配。行为模式不仅描述对象或类的模式，还描述了它们之间的通信模式。常用的行为模式有观察者、策略等。

1. 你在项目中运用了何种设计模式以及如何用此模式进行分析与设计。要紧密结合主题项目，选择1-2种设计模式进行讨论就可以了。

论数据访问层设计技术及其应用

一、首先用400-600字的篇幅简要叙述作者参与开发的软件系统的概要和所担任的工作。

二、数据访问层的技术主要在于数据映射的问题如写Hibernate或iBATIS的应用。

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

iBATIS一词来源于“internet”和“abatis”的组合，是一个由Clinton Begin在2002年发起的开放源代码项目。于2010年6月16号被谷歌托管，改名为MyBatis。是一个基于SQL映射支持Java和·NET的持久层框架。

三、详细论述你在项目中运用相关技术进行开发的，此时无非就是如何用好这些技术。

Hibernate的调优方案：

制定合理的缓存策略；

尽量使用延迟加载特性；

采用合理的Session管理机制；

使用批量抓取，设定合理的批处理参数（batch\_size）;

进行合理的O/R映射设计。

Mybatis调优方案：

MyBatis在Session方面和Hibernate的Session生命周期是一致的，同样需要合理的Session管理机制。MyBatis同样具有二级缓存机制。 MyBatis可以进行详细的SQL优化设计。

论微服务架构及其应用

一、首先用400-600字的篇幅简要叙述作者参与开发的软件系统的概要和所担任的工作。

二、微服务的特点包括：

微服务的基本思想在于考虑围绕着业务领域组件来创建应用，这些就应用可独立地进行开发、管理和加速。在分散的组件中使用微服务云架构和平台使部署、管理和服务功能交付变得更加简单。

微服务是利用组织的服务投资组合，然后基于业务领域功能分解它们，在看到服务投资组合之前，它还是一个业务领域。

微服务这一概念出现于2012年，是因软件作者Martin Fowler而流行，他承认这并没有精确地定义出这一架构形式，虽然围绕业务能力、自动化部署、终端智能以及语言和数据的分散控制有一些常见的特性。

开源工作流平台 “Imixs-Workflow“发布了一款新的微服务架构，作为工作流来管理解决方案。Imixs的微服务（ Imixs-Microservice）提供了一个工作流封装成微服务架构。这一服务可以独立于其背后的技术，绑定到任何业务应用中去。这允许业务应用改变业务逻辑的时，不用更改任何代码。这业务目标可以通过工作流模型控制。

Imixs的微服务是基于Imixs的工作流引擎（ Imixs-Workflow Engine）的复杂功能构建的，它可以以多种不同的方法来控制业务数据。Imixs的微服务可以发送电子邮件推送消息、日志业务交换，还可以确保所有类型业务数据的安全。

Imixs的工作流模型可以给业务处理模型（Imixs-Workflow Modeller）中的每种状态单独的设计一个ACL。这许可了高度复杂的业务应用程序，并在每个流程实例周围驻起了安全层。[4]

三、详细论述在项目中如何应用微服务架构进行开发的。

###### 2017年论文

论软件系统建模方法及应用

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、需要较为详细地说明目前各种常见的信息系统建模方法的核心思想，并对每种方法所创建的模型进行简要描述。

（1）结构化建模方法。

结构化建模方法是以过程为中心的技术，可用于分析一个现有的系统以及定义新系统的业务需求。结构化建模方法所绘制的模型称为数据流图（DFD）。对于流程较为稳定的系统可考虑结构化建模方法。

（2）信息工程建模方法（或数据库建模方法）。

信息工程建模方法是一种以数据为中心，但过程敏感的技术，它强调在分析和研究过程需求之前，首先研究和分析数据需求。信息工程建模方法所创建的模型被称为实体联系图（ERD）。主要用于数据建模。

（3）面向对象建模方法。

面向对象建模方法将“数据”和“过程”集成到被称为“对象”的结构中，消除了数据和过程的人为分离现象。面向对象建模方法所创建的模型被称为对象模型。随着面向对象技术的不断发展和应用，形成了面向对象的建模标准，即UML（统一建模语言）。UML定义了几种不同类型的模型图，这些模型图以对象的形式共建一个信息系统或应用系统。目前比较常用的建模方法。

三、论文中需要结合项目实际工作，详细论述在项目中是如何使用所选定的信息系统建模方法创建系统的逻辑模型和物理模型，并详细说明实施效果。

论软件架构风格

一、结合自己所参与的软件项目，概要介绍该项目的背景及主要内容，并明确指出在其中所承担的主要任务和开展的主要工作。

二、常见的架构风格5大类，至少选2-3个类进行说明。（注意本部分内容虽然题目要求是详细论述，但实际上不是文章的重心，问题3才是结合项目的详细论述部分）

Garlan和Shaw将软件架构风格分为五大类，数据流风格、调用/返回风格、独立构件风格、虚拟机风格和仓库风格。其中：

（1）数据流风格包括批处理序列架构风格和管道/过滤器架构风格；

（2）调用/返回风格包括主程序/子程序架构风格、数据抽象和面向对象架构风格和层次结构架构风格；

（3）独立构件风格包括进程通信架构风格和事件驱动的架构风格；

（4）虚拟机风格包括解释器架构风格和基于规则的系统；

（5）仓库风格包括数据库架构风格和黑板架构风格。

其他的还有特定领域软件架构、状态转移等以及分布式处理等。其中分布式架构风格中有客户机/服务器风格、浏览器/服务器风格、CORBA、DCOM、EJB。

每一种具体的软件结构风格的模型如下：

1．数据流风格包括批处理序列和管道/过滤器架构风格。

（1）批处理序列架构风格。组件为一系列固定顺序的计算单元，组件间只通过数据传递交互。每个处理步骤是一个独立的程序，每一步必须在前一步结束后才能开始，数据必须是完整的，以整体的方式传递。

（2）管道/过滤器架构风格。每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流，经过处理，产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，包括通过计算和增加信息丰富数据，通过浓缩和删除精炼数据，通过改变记录方式转化数据，递增地转化数据等。在输入被完全消费之前，输出便产生了。这里构件被称为过滤器，连接件就是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。

2．调用/返回风格包括主程序/子程序架构风格、数据抽象和面向对象架构风格以及层次结构架构风格

（1）主程序/子程序架构风格。单线程控制，把问题划分为若干处理步骤，构件即为主程序和子程序。子程序通常可合成为模块。过程调用作为交互机制，即充当连接件。调用关系具有层次性，其语义逻辑表现为子程序的正确性取决于它调用的子程序的正确性。

（2）数据抽象和面向对象架构风格。这种风格的构件是对象。对象是抽象数据类型的实例。在抽象数据类型中，数据的表示和它们的相应操作被封装起来。对象的行为体现在其接受和请求的动作。连接件即是对象间交互的方式，对象是通过函数和过程的调用来交互的。对象具有封装性，一个对象的改变不会影响其他对象。对象拥有状态和操作，也有责任维护状态。这种结构风格中包含有封装、交互、多态、集成、重用等特征。

（3）层次结构架构风格。层次系统组织成一个层次结构。构件在一些层实现了虚拟机。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。这个风格的特点是每层为上一层提供服务，使用下一层的服务，只能见到与自己邻接的层。大的问题分解为若干个渐进的小问题，逐步解决，隐藏了很多复杂度。修改一层，最多影响两层，而通常只能影响上层。上层必须知道下层的身份，不能调整层次之间的顺序。

3．独立构件风格包括进程通信架构风格和事件驱动的架构风格

（1）进程通信架构风格。构件是独立的过程，连接件是消息传递。这种风格的特点是构件通常是命名过程，消息传递的方式可以是点对点、异步和同步方式、以及远过程调用等

（2）事件驱动的架构风格。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。系统中的其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程。一个事件的触发就导致了另一个模块中的过程的调用。这种风格中的构件是非命名的过程，它们之间交互的连接件往往是以过程之间的隐式调用（Implicit Invocation）来实现的。基于事件的隐式调用风格的主要优点是为软件重用提供了强大的支持，为构件的维护和演化带来了方便，其缺点是构件放弃了对系统计算的控制。

4．虚拟机风格包括解释器架构风格和基于规则的系统

（1）解释器架构风格。一个解释器通常包括完成解释工作的解释引擎，一个包含将被解释的代码的存储区，一个记录解释引擎当前工作状态的数据结构，以及一个记录源代码被解释执行的进度的数据结构。具有解释器风格的软件中含有一个虚拟机，可以仿真硬件的执行过程和一些关键应用。其缺点是执行效率较低。

（2）基于规则的系统。基于规则的系统包括规则集、规则解释器、规则/数据选择器以及工作内存。

5．仓库风格包括数据库架构风格和黑板架构风格

（1）数据库架构风格。数据库架构是库风格最常见的形式。构件主要有两大类，一个是中央共享数据源，保存当前系统的数据状态，另一个是多个独立处理元素，处理元素对数据元素进行操作。

（2）黑板架构风格。黑板架构包括知识源、黑板、控制三部分。知识源包括若干独立计算的不同单元，提供解决问题的知识，知识源响应黑板上的变化，也只修改黑板。黑板是一个全局数据库，包含解域的全部状态，是知识源互相作用的唯一媒介。知识源响应是通过黑板状态的变化来控制。黑板通常应用在对于解决问题没有确定性算法的系统中，例如信号处理、问题规划、编译器优化等软件系统的设计中。

三、问题中明确要求回答选择架构的原因，其实是要求考生在组织论文时，说明作者选择架构的依据是什么，而各种架构应用在作者担任的项目中，有何优势与劣势。当这些内容分析清楚之后，合适的架构自然浮出水面来了。然后附带性的讲一讲架构的内容，架构具体内容与设计都已不是重心。最后谈一谈整体效果收尾。

论无服务器架构及其应用

一、结合自己所参与的软件项目，概要介绍该项目的背景及主要内容，并明确指出在其中所承担的主要任务和开展的主要工作。

二、问题2要求回答一些知识类型的问题，需要的知识内容如下：

首先值得说明的是无服务器架构并不是不再需要服务器，只是开发人员不再需要担心基础设施，因为一切都由云提供商负责。使用这种方法，开发人员只需部署适当的代码，其他一切由云提供商自动管理。

在传统的Web应用程序架构中，你必须管理基础架构，并确保其满足可扩展性和安全性需求。例如，客户端在一边，服务器在另一边。客户端发送一个“请求”，服务器回复“响应”。但是，如果无法满足应用程序需求，则很快就要扩展服务器端了。

无服务器模型提供了完全不同的方法。与传统架构不同，无服务架构在无状态计算容器中运行，这些容器是事件触发的，短暂的(只能持续一次调用)，并由第三方完全管理。就像一个“黑盒子”，这个服务你只需上传代码并实时自动处理。当一个请求进来时，就会运行你的Lambda功能的容器。

在成本方面，使用无服务器模型，通常仅支付服务请求和运行代码所需的计算时间。计费以100毫秒为单位进行计量，使其具有成本效益，并且易于自动从每天几个请求到每秒数千次都可以。

无服务器架构的优点包括：

（1）降低运营成本：无服务架构本质上是一个外包解决方案。基础设施不会消失。然而，与常规云服务相比，事实上，只需要根据流量规模和形式支付需要的计算量，这可能会大大节省运营成本，特别是对于具有不同变化的早期和动态应用负载要求。

（2）可扩展性强：可扩展性强在云服务领域并不新鲜，但无服务架构将其提升到一个全新的水平。无服务架构的缩放功能不仅可以降低计算成本，还可以减少运行管理，因为缩放是自动的。使用无服务器，无需明确添加和删除实例到服务器阵列，并让供应商为你扩展应用程序。由于云计算提供商根据每个请求执行扩展，所以甚至不需要考虑在内存不足之前可以处理多少并发请求的问题。

（3）分离问题：无服务器几乎迫使你实施关注模型的分离，通过该分离将应用程序分成不同的部分，以使每个部分都解决一个单独的问题。

（4）隔离进程：在无服务器环境中，每个Lambda函数都完全隔离。如果其中一个功能关闭，它不影响其他功能，它不会导致服务器崩溃。

无服务器架构的缺点包括：

（1）缺乏控制权：通过任何外包策略，你都可以将某些系统的控制权给第三方供应商。由于系统停机，意外的限制，成本的变化，功能的丧失，强制的API升级等，这种缺乏控制可能会显现出来。此外，如果需要专门的服务器进行专门的流程，那么必须自己运行这个专门的服务器。一个无服务器架构，在大多数情况下，提供商业化的基础设施，将以广义的方式运行你的流程。

（2）长时间运行流程的高成本：如果你的进程持续运行很长时间，则可能会需要运行自己的服务器。因为这不仅涉及到成本，还涉及到拥有的技能或者想要投入运行自己的服务器的专注;在评估这些解决方案时，请考虑所有这些方面。

（3）供应商锁定将基础架构管理完全外包给无服务器提供商，无疑将自己锁定到该供应商。每个供应商都有自己的标准和编程框架，不容易改变。在几乎每一种情况下，无论从供应商使用的无服务器功能，将由另一个供应商进行不同的实现。如果要切换供应商，几乎肯定需要更新操作工具(部署，监控等)，可能还需要更改代码。

三、结合项目经验详细论述无服务器架构的应用，并分析效果如何，哪些方面做得好，哪些方面做得不好。

论软件质量保证

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、问题2涉及到一些知识性内容，需要的素材如下：

质量保证是指定期评估项目总体绩效，建立项目能达到相关质量标准的信心。质量保证对项目的最终结果负责，而且还要对整个项目过程承担质量责任；质量控制是指监测项目的总体结果，判断它们是否符合相关质量标准，并找出如何消除不合格绩效的方法。

软件质量保证（Software Quality Assurance，SQA）是指为保证软件系统或软件产品充分满足用户要求的质量而进行的有计划、有组织的活动，这些活动贯穿于软件生产的各个阶段即整个生命周期。SQA由各项任务构成，这些任务的参与者有两种人，分别是软件开发人员和质量保证人员。前者负责技术工作，后者负责质量保证的计划、监督、记录、分析及报告工作。软件开发人员通过采用可靠的技术、方法和措施，进行正式的技术评审，执行软件测试来保证软件产品的质量。质量保证人员则辅助软件开发人员得到高质量的最终产品。

美国卡耐基·梅隆大学软件工程研究所推荐了一组有关质量保证的计划、监督、记录、分析及报告的SQA活动，这些活动由一个独立的SQA小组执行。

（1）制订SQA计划。SQA计划在制订项目计划时制订，由相关部门审定。它规定了软件开发小组和质量保证小组需要执行的质量保证活动。有关该计划的详细内容，请阅读《系统分析师教程》20.7.2节。

（2）参与开发该软件项目的软件过程描述。软件开发小组为将要开展的工作选择软件过程，SQA小组则要评审过程说明，以保证该过程与企业政策、内部的软件标准、外界所制订的标准以及项目开发计划的其他部分相符。

（3）评审。评审各项软件工程活动，核实其是否符合已定义的软件过程。SQA小组识别、记录和跟踪所有偏离过程的偏差，核实其是否已经改正。

（4）审计。审计指定的软件工作产品，核实其是否符合已定义的软件过程中的相应部分。SQA小组对选出的产品进行评审，识别、记录和跟踪出现的偏差，核实其是否已经改正，定期向项目负责人报告工作结果。

（5）记录并处理偏差。确保软件工作及工作产品中的偏差已被记录在案，并根据预定规程进行处理。偏差可能出现在项目计划、过程描述、采用的标准或技术工作产品中。

（6）报告。记录所有不符合部分，并向上级管理部门报告。跟踪不符合的部分直到问题得到解决。

除了进行上述活动外，SQA小组还需要协调变更的控制与管理，并帮助收集和分析软件度量的信息。

三、结合具体参与的项目，从上面写到的几个方面来展开论述说明自己是如何进行质量保证工作的。最后总结效果，并指出不足，给出改进意见。

###### 2018年论文

论软件开发过程RUP及其应用

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、本文内容的组织可以将问题2与问题3结合起来论述。先说明RUP的四个阶段及RUP的特征，然后再论述每个阶段，作者开展了哪些工作。

RUP中的软件过程在时间上被分解为四个顺序的阶段，分别是初始阶段、细化阶段、构建阶段和交付阶段。每个阶段结束时都要安排一次技术评审，以确定这个阶段的目标是否已经满足。如果评审结果令人满意，就可以允许项目进入下一个阶段。

四个阶段的核心任务分别为：

（1）初始阶段

· 明确地说明项目规模。这涉及了解环境及最重要的需求和约束，以便于可以得出最终产品的验收标准。

· 计划和准备商业理由。评估风险管理、人员配备、项目计划和成本/进度/收益率折中的备选方案。

· 综合考虑备选构架，评估设计和自制/外购/复用方面的折中，从而估算出成本、进度和资源。此处的目标在于通过对一些概念的证实来证明可行性。该证明可采用可模拟需求的模型形式或用于探索被认为高风险区域的初始原型。初始阶段的原型设计工作应该限制在确信解决方案可行就可以了。该解决方案在细化和构建阶段实现。

· 准备项目的环境，评估项目和组织，选择工具，决定流程中要改进的部分。

（2）细化阶段

· 快速确定构架，确认构架并为构架建立基线。

· 根据此阶段获得的新信息改进前景，对推动构架和计划决策的最关键用例建立可靠的了解。

·为构建阶段创建详细的迭代计划并为其建立基线。

· 改进开发案例，定位开发环境，包括流程和支持构建团队所需的工具和自动化支持。

· 改进构架并选择构件。评估潜在构件，充分了解自制/外购/复用决策，以便有把握地确定构建阶段的成本和进度。集成了所选构架构件，并按主要场景进行了评估。通过这些活动得到的经验有可能导致重新设计构架、考虑替代设计或重新考虑需求。

（3）构建阶段

· 资源管理、控制和流程优化。

·完成构件开发并根据已定义的评估标准进行测试。

· 根据前景的验收标准对产品发布版进行评估。

（4）产品化阶段（提交阶段）

· 执行部署计划。

·对最终用户支持材料定稿。

·在开发现场测试可交付产品。

·制作产品发布版。

·获得用户反馈。

·基于反馈调整产品。

·使最终用户可以使用产品。

RUP最核心的3个特征是：用例驱动、以架构为中心的、迭代和增量。

制品（Artifact）——what的问题：制品是活动生成、创建或修改的一段信息。也可译为产品、工件等，和制品的意思差不多。

工作流（Workflow）——when 的问题：工作流描述了一个有意义的连续的活动序列，每个工作流产生一些有价值的产品，并显示了角色之间的关系。

论软件体系结构演化

软件体系结构的演化是使用系统演化步骤去修改系统，以满足新的需求。简要论述系统演化的6个步骤。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、首先需要弄清楚的是此处的“软件体系结构演化”实际上指的是ABSD方法中的最后一个阶段。体系结构演化是使用系统演化步骤去修改应用，以满足新的需求。主要包括以下六个步骤。

1、需求变动归类

首先必须对用户需求的变化进行归类，使变化的需求与已有构件对应。对找不到对应构件的变动，也要做好标记，在后续工作中，将创建新的构件，以对应这部分变化的需求。

2、制订体系结构演化计划

在改变原有结构之前，开发组织必须制订一个周密的体系结构演化计划，作为后续演化开发工作的指南。

3、修改、增加或删除构件

在演化计划的基础上，开发人员可根据在第1步得到的需求变动的归类情况，决定是否修改或删除存在的构件、增加新构件。最后，对修改和增加的构件进行功能性测试。

4、更新构件的相互作用

随着构件的增加、删除和修改，构件之间的控制流必须得到更新。

5、构件组装与测试

通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成，形成新的体系结构。然后对组装后的系统整体功能和性能进行测试。

6、技术评审

对以上步骤进行确认，进行技术评审。评审组装后的体系结构是否反映需求变动，符合用户需求。如果不符合，则需要在第2到第6步之间进行迭代。

三、论文中需要结合项目实际工作，阐述6个步骤的具体应用，此时可以重点讲述其中的2-3个方面，不必面面俱到的论述，最后说明实施效果。

论面向服务架构设计及应用

说明面向服务架构的主要技术和标准，详细阐述每种技术和标准的具体内容。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、面向服务架构的主要技术有Web服务、ESB。涉及到的标准有：

1、UDDI协议

UDDI（统一描述、发现和集成协议）计划是一个广泛的、开放的行业计划，它使得商业实体能够（1）彼此发现，（2）定义他们怎样在Internet上互相作用，并在一个全球的注册体系架构中共享信息。UDDI是这样一种基础的系统构筑模块，他使商业实体能够快速、方便地使用他们自身的企业应用软件来发现合适的商业对等实体，并与其实施电子化的商业贸易。

UDDI同时也是Web服务集成的一个体系框架。它包含了服务描述与发现的标准规范。UDDI规范利用了W3C和Internet工程任务组织（IETF）的很多标准作为其实现基础，比如扩展标注语言（XML）、HTTP和域名服务（DNS）等协议。另外，在跨平台的设计特性中，UDDI主要采用了已经被提议给W3C的SOAP（Simple Object Access Protocol，简单对象访问协议）规范的早期版本。

2、WSDL规范

WSDL是Web Services Description Language（Web服务描述语言）的缩写，是一个用来描述Web服务和说明如何与Web服务通信的XML语言。它是Web服务的接口定义语言，由Ariba、Intel、IBM、MS等共同提出，通过WSDL，可描述Web服务的三个基本属性：

1、服务做些什么——服务所提供的操作（方法）；

2、如何访问服务——和服务交互的数据格式以及必要协议；

3、服务位于何处——协议相关的地址，如URL。

WSDL文档以端口集合的形式来描述Web服务，WSDL服务描述包含对一组操作和消息的一个抽象定义，绑定到这些操作和消息的一个具体协议，和这个绑定的一个网络端点规范。

3、SOAP协议

SOAP（Simple Object Access Protocol）简单对象访问协议是在分散或分布式的环境中交换信息的简单的协议，是一个基于XML的协议。它包括四个部分：SOAP封装（Envelop），封装定义了一个描述消息中的内容是什么，是谁发送的，谁应当接受并处理它，以及如何处理它们的框架；SOAP编码规则（Encoding Rules），用于表示应用程序需要使用的数据类型的实例；SOAP RPC表示（RPC Representation），表示远程过程调用和应答的协定；SOAP绑定（Binding），使用底层协议交换信息。

三、论文中需要结合项目实际工作，论述构建SOA架构时遇到的问题以及如何解决的，效果如何。注意本主题才是文章的重心所在。

论NoSQL数据库技术及其应用

详细论述常见的NoSQL数据库技术及其所包含的主要内容，并说明NoSQL数据库的主要适用场景。

一、应结合自己参与的信息系统项目，说明在其中所承担的工作。

二、

NoSQL的主要优势：

（1）避免不必要的复杂性

（2）高吞吐量

（3）高水平扩展能力和低端硬件集群

（4）避免了昂贵的对象-关系映射

NoSQL的缺点：

（1）数据模型和查询语言没有经过数学验证

（2）不支持ACID特性

（3）功能简单

（4）没有统一的查询模型

NoSQL数据库的四大分类：

1、键值(Key-Value)存储数据库

这一类数据库主要会使用到一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT系统来说的优势在于简单、易部署。但是如果DBA只对部分值进行查询或更新的时候，Key/value就显得效率低下了。例如：Tokyo Cabinet/Tyrant, Redis, Voldemort, Oracle BDB.

2、列存储数据库。

这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。如：Cassandra, HBase, Riak。

HBase：HBase是一个分布式的、面向列的开源数据库，该技术来源于 Fay Chang 所撰写的Google论文“Bigtable：一个结构化数据的分布式存储系统”。就像Bigtable利用了Google文件系统（File System）所提供的分布式数据存储一样，HBase在Hadoop之上提供了类似于Bigtable的能力。HBase是Apache的Hadoop项目的子项目。HBase不同于一般的关系数据库，它是一个适合于非结构化数据存储的数据库。另一个不同的是HBase基于列的而不是基于行的模式。

3、文档型数据库

文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可 以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。如：CouchDB, MongoDb. 国内也有文档型数据库SequoiaDB，已经开源。

Mongo DB：Mongo DB 是目前在IT行业非常流行的一种非关系型数据库(NoSql),其灵活的数据存储方式备受当前IT从业人员的青睐。Mongo DB很好的实现了面向对象的思想(OO思想),在Mongo DB中 每一条记录都是一个Document对象。Mongo DB最大的优势在于所有的数据持久操作都无需开发人员手动编写SQL语句,直接调用方法就可以轻松的实现CRUD操作。

Sequoia DB：SequoiaDB是一款分布式非关系型文档数据库，可以被用来存取海量非关系型的数据，其底层主要基于分布式，高可用，高性能与动态数据类型设计SequoiaDB可以独立作为一款高性能可扩展的NoSQL数据库使用，也可与当前主流分布式计算框架Hadoop紧密集成。

4、图形(Graph)数据库

图形结构的数据库同其他行列以及刚性结构的SQL数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL数据库没有标准的查询语言(SQL)，因此进行数据库查询需要制定数据模型。许多NoSQL数据库都有REST式的数据接口或者查询API。如：Neo4J, InfoGrid, Infinite Graph.

三、论文中需要结合项目实际工作，详细论述在项目中所采用的noSQL数据库，并详细说明实施效果。