第一章

1.1The Nature of Software Development

Discussion 1

Define iterative incremental development. Give five examples of iterative incremental processes used in software development.

1.the spiral model.(螺旋模型)

2.the Rational Unit Process.(RUP)

3.Model Driven Architecture.(MDA)

4.the agile development process.(敏捷开发过程的)

5.Aspect-oriented software development.(面向方面的软件开发的)

Discussion 2

What is COBIT? How does it differ from ISO9000 and ITIL? What are the four domains COBIT groups IT-related efforts into?

COBIT（Control Objectives for Information and related Technology）是国际上通用的信息系统审计的标准，由信息系统审计与控制协会（ISACA）在1996年公布。这是一个在国际上公认的、权威的安全与信息技术管理和控制的标准，已经更新至COBIT2019版。它在商业风险、控制需要和技术问题之间架起了一座桥梁，以满足管理的多方面需要。该标准体系已在世界一百多个国家的重要组织与企业中运用，指导这些组织有效利用信息资源，有效地管理与信息相关的风险

Discussion 3

What is UML? What must a process do in order to adopt UML? Where is UML deficient? Name the three categories of models used in UML and describe what they do.

UML（统一建模语言，Unified Modeling Language） 又称标准建模语言，一种通用的可视化建模语言。 UML独立于任何软件开发过程，但采用UML的过程，必须支持一种面向对象的方法来生产软件。 UML在支持开发声明周期的详细设计阶段有些不足。 而模型可以分为以下三组： 1.状态模型：用于描述静态数据结构 2.行为模型：用于描述对象协作 3.状态转换模型：用于描述随着时间的推移系统所允许的状态

DIscussion 4

Which CMM level of maturity is needed for the organization to be able to respond successfully to a crisis situation? Explain.

第三级。 第三级成熟度的组织已经建立了标准的且不断得到改进的工作方式，软件开发的活动按照既定的标准、方法执行并且具有了一定的决策分析与解决问题的能力并能对风险进行一定程度上的管控。而并不具备第四级所拥有的能够进行风险预测的一些列能力，故仍然停留在第三级

Discussion 5

Based on your experiences with software products, how would you interpret the observation that the essence of software engineering is determined by inherent software complexity, conformity, changeability, and invisibility? How would you explain these four factors? How is software engineering different from traditional engineering, such as civil or mechanical engineering?

根据您对软件产品的经验，您如何解释软件工程的本质是由固有的软件复杂性、一致性、可变性和不可见性决定的观察结果？ 你如何解释这四个因素？ 软件工程与土木工程或机械工程等传统工程有何不同？

软件是一种创造性、开发行为的产品，一般情况下不是制造业重复行为的结果，但因软件本身存在的商品属性使得它的某些特点能够被归类。

复杂性是软件规模的函数以及组成软件产品的构件之间相互依赖关系的函数。

一致性是指应用软件需要与它所依赖的平台一致，并与现有信息系统保持一致（以便继承）。

可变性是软件产品描述的业务过程和需求经常发生变化。

不可见性是指输出程序的语句、二进制代码以及周边系统软件不可见。

相比与传统工程，软件工程对于客户需求的要求更精准、更全面，同时它对开发者的要求也更高，对开发工作的管理定位也更为明确，管理手段更为丰富，管理要求同样相对较高的水平。

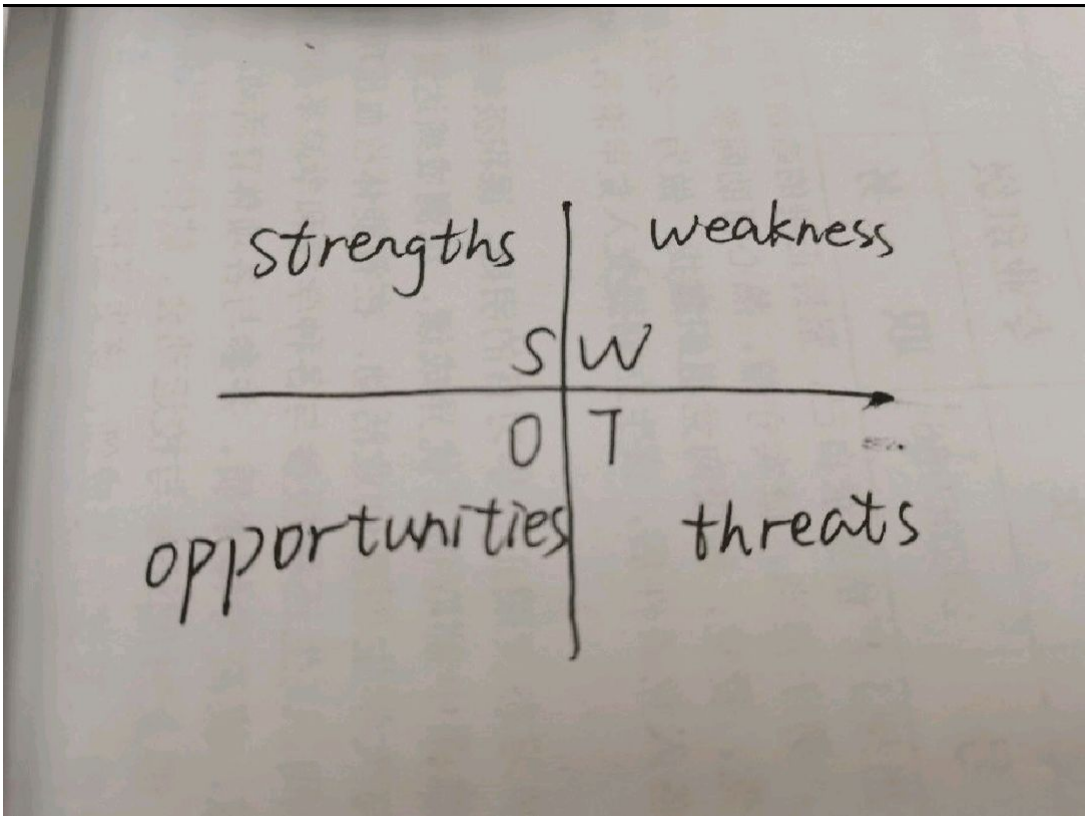
Discussion 6

Recall the definition of a stakeholder. Is a software vendor or a technical support person a stakeholder? Explain.

利益相关者是指在软件项目中存在利害关系的人。任何受到系统影响或者对系统开发产生影响的人都是利益相关者。无论软件供应商和技术支持人员都与软件项目所产生的利益密切相关，所以他们都是的利益相关者也就是stakeholder

1.2 System Planning

What does the acronym SWOT mean? Give a detailed overview of SWOT and how it would be applied in an organization. Use a diagram to explain your answer.



1.3 Systems for Three Management Levels

Discussion 1

What are the three management levels? Consider a banking application that monitors the usage patterns of a credit card by its holder in order to block the card automatically when the bank suspects a misuse (theft, fraud, etc.). Which management level is addressed by such an application? Give reasons.

三个管理层次是什么？ 考虑一个银行应用程序，它监视信用卡持有人的使用模式，以便在银行怀疑存在滥用（盗窃、欺诈等）时自动阻止该卡。 此类应用程序针对哪个管理级别？ 说明理由。

策略级、战术级、操作级；

策略级。

如果怀疑用户信用卡被盗用，涉及到查证， 要用到知识管理领域中的数据挖掘来发现:其能够发现知识和知识决策的关系以及模式。 其主要目的之一是关联和的的的的数据分析：发现一件事件能够导致另一事件的模式。

Discussion 2

Explain what a On-line Transaction Processing System is. What is a transaction in terms of OLTP systems? Why are transactions necessary in databases?

联机事务处理(OLTP)是指利用计算机网络，将分布于不同地理位置的业务处理计算机设备或网络与业务管理中心网络连接，以便于在任何一个网络节点上都可以进行统一、实时的业务处理的的活动或客户服务。 在联机事务处理系统中，事务（Transaction）:工作的一个逻辑单元，完成某一特定的业务的任务，并在业务任务完成后，保证数据库的完整性。 数据库技术的recovery操作可以以事务为单位进行。事务使系统能够更方便的进行故障恢复以及并发控制,从而保证数据库状态的一致性。事务使系统能够更方便的进行故障恢复以及并发控制,从而保证数据库状态的一致性的。

1.5 Development Models and Methods

Discussion

Briefly explain the following three agile development models/methods – eXtreme Programming (XP), Aspect Oriented Software Development, and Feature-Driven Development. Compare the second and the third model/method to XP from the generic viewpoint of emphasis placed on the phases of the analysis-design- implementation cycle. Search the Internet for information on these models/methods.

简要说明以下三种敏捷开发模型/方法——极限编程（XP）、面向方面的软件开发和功能驱动开发。 将第二个和第三个模型/方法与 XP 从重点放在分析-设计-实现周期的各个阶段的一般观点进行比较。 在 Internet 上搜索有关这些模型/方法的信息。

1.极限编程是一种软件开发方法，是统称为敏捷方法的一部分。XP建立在价值观，原则和实践的基础上，其目标是让中小型团队能够生产高质量的软件并适应不断变化的需求。

2..面向方面的软件开发 （AOSD） 是一种软件设计解决方案，可帮助解决其他软件方法（如程序性、结构化和对象化编程 （OOP） 未正确解决的模块化问题。AOSD 补充而不是替换这些其他类型的软件方法。

3.功能驱动开发 （FDD） 是一个敏捷的框架，顾名思义，它围绕功能的进展组织软件开发。但是，FDD上下文中的功能不一定是通常理解意义上的产品特征。相反，它们更类似于Scrum中的用户故事。换句话说，"完成登录过程"可能被视为功能驱动开发 （FDD） 方法中的一个功能。 比较:极限编程的主要目标在于降低因需求变更而带来的成本,一个应用了极限编程方法的系统开发项目在应对需求变更时将显得更为灵活。面向方面的软件开发（AOSD)能减少和系统的其他部分的耦合，是系统能更好的维护和重构，叫它切面，就是说可以在需要它的时候将其加入到系统中，且能很好的和原系统配合，当不需要的时候就可以去除，也不会重新修改代码。FDD是一个模型驱动的快速迭代开发过程，它强调的是简化、实用、 易于被开发团队接受，适用于需求经常变动的项目。简单地说，FDD"是一个以Architecture为中心的，采用短迭代期，目期驱动的开发过程。它首先对整个项目建立起一个整体的模型，然后通过两周一次'设计功能-实现功能'的迭代完成项目开发"。

第二章

2.1

Discussion 1

What is Process Hierarchy Modeling (PHM)? Is a Process Hierarchy Diagram part of Business Process Modeling Notation (BPMN)? Explain what a process is. Using diagrams explain the difference between composite and atomic processes.

过程层次建模 (PHM)，是为了填补业务过程设计和过程实现之间的空白，使用业务过程建模表示法(BPMN)来建模。 BPMN 专门用于对由活动定义的业务建模，这些活动能够产生对企业或其外部利益相关者有意义的价值的事务 过程的概念是分层的，过程可能包括其他过程和子过程，将过程中的原子活动成为任务， 过程层次图定义了业务过程模型中的静态结构 原子过程及任务不包含任何子过程 复合过程通过子过程来描述包的行为!

Discussion 2

Explain what Business Process Modeling Notation (BPMN) is. Are there any alternatives to BPMN? Using diagrams describe the four basic categories of modeling elements in BPMN.

BPMN定义了一个业务流程图（Business Process Diagram），该业务流程图基于一个流程图（flowcharting），该流程图被设计用于创建业务流程操作的图形化模型。而一个业务流程模型（Business Process Model），指一个由图形对象（graphical objects）组成的网状图，图形对象包括活动（activities)和用于定义这些活动执行顺序的流程控制器（flow controls）。 BPMN 提供了四种基本类型的建模元素: Flow objects(流对象),包括Events(事件),Activities(活动),Gateways(路由). Connecting objects(连接对象) Pools(Swimlanes)(泳池) Artifacts(人工制品)

Discussion 3

What are the three phases of solution envisioning? Describe each in detail.

阶段一：业务能力探索.此阶段确定业务能力，即为企业的IT解决方案提交具体成果的能力。此阶段描述能力案例，即提供了解决方案思路，为每个能力生成一个业务案例。的从IT的观点来看，每个能力案例可以被看作是为了达到业务目标的可复用的软件构件，当理解了业务问题的性质，并且理解了解决方案需要产生的具体成果时，第一个阶段才算完成。这些理解被记录在所提出的解决方案的初步构想中，可以传达给更广泛的利益相关者，的以便在构想研讨中做出评价和进一步决策。 阶段二:解决方案能力构想.此阶段的目的是将能力用例发展成解决方案概念，确保利益相关的者在这个方案的意见一致。解决方案概念将业务环境作为输入，产生的未来新工作方法作为输出。解决方案概念集中于最终的解决方案体系结构，并在解决方案构想研的讨中得到有效的发展。 阶段三:软件能力设计。此阶段开发软件能力体系结构、细化具有项目规划和风险分析的业务用例。此阶段是软件建模的一项活动，为构建解决方案开发高层模型并执行计划。建模计划包括功能性需求和非功能性需求以及那能力体系结构，显示高层软件构件之间的相互的作用。

2.2

Discussion

What is Requirements Elicitation? What does it involve? What artifacts does it produce? What is the difference between a functional and a non-functional requirement? Give an example of a functional and a non-functional requirement.

在需求工程中，需求获取是从用户、客户和其他利益相关者那里研究和发现系统需求的实践。这种实践有时也被称为“需求收集”。术语“启发”在书籍和研究中使用，以提出这样一个事实：良好的需求不能仅仅从客户那里收集，正如名称需求收集所表明的那样。需求获取非常重要，因为您永远无法通过询问系统应该做什么或不做什么（为了安全和可靠性）来确定您从用户和客户那里获得了所有需求。需求获取实践包括访谈、问卷调查、用户观察、研讨会、头脑风暴、用例、角色扮演和原型制作。在对需求进行分析、建模或指定之前，它们必须通过启发过程进行收集。需求获取是需求工程过程的一部分，通常紧随其后的是需求的分析和规范。常用的启发过程是利益相关者会议或访谈。

2.3

Discussion

Give four methods used in Requirements Elicitation. Describe each in detail.

面谈：发现事实和聚集信息的基本技术，大多数的面谈过程都是与客户一起进行的，与客户面谈大多数是用来导出用例需求。面谈有两种形式，结构化和非结构化 结构化的面谈需要提前准备，由一个明确的日程，并且许多问题都是提前确定的，有一些问题可以是无确定答案的，其他问题可以是有确定答案的。结构化需要非结构化进行补充，非结构化面谈更像是非正式化的会议，没有预计的问题和预计的目标，非结构化面谈的目的是鼓励客户讲出自己的想法，并且在这个过程中导出业务分析员没有想到的，没有能够提出的有关问题的需求，结构化和非结构化都必需有某个出发点和讨论的语境，他可以是一篇短的书面文档，或者面谈之前发给客户的面谈文件，解释面谈者的目标或提出一些问题。 有三种问题应该避免：1.固执己见的问题 2.带有偏见的问题 3.强加于人的问题

调查表：是向客户收集信息的有效的方法，他一般作为面谈的补充形式，而不是要替代他，一般而言调查表不如面谈有效，因为无法澄清问题和可能得到的响应 优点：回答者可以有时间考虑如何回答，并且可以是匿名的的 缺点：回答者不容易理解这些问题的含义 调查表应该设计得使问题回答得容易，特别的应该避免开发性的问题，大多数问题应该是封闭式的：多项选择，评价，排序

观察有三种形式：1.被动观察，观察业务活动但不干扰或直接干预他 2. 主动观察，参与到活动中，并且高效的成为团队中的一部分 3. 解释观察，在工作过程中，用户向观察者说明他的活动要使观察具有代表性，观察应该持续较长的一段时间，在不同的时间段和不同的工作负荷下进行 观察的主要困难在于，人们在被观察的情况下总想表现的不同的行为，他们总想按照形式化的规则和过程去做，这样会由于隐藏了正面或负面的方法而歪曲了实际情况。 观察法有着道德、隐私、法律方面问题的存在。

文档以及软件系统的研究是发现用例需求何领域知识需求的宝贵技术 要研究的系统报表格和报表包括：计算机屏幕和报告、相关的文档、操作系统手册、用户文档、技术文档、系统分析和设计模型

2.4

Discussion 1

Why is Requirements Negotiation and Validation needed?

因为请求：

重叠和冲突需求依赖矩阵

可能模棱两可或不切实际

可能仍未被发现

可能超出范围

有时在项目范围之外，但在信息系统范围内（请求太难实 现且应手动完成，可能优先级低，可能在硬件中实现

Discussion 2

Define requirement risk and priorities. Give five categories of risk and give an example of each.

风险是对项目计划的威胁， 风险决定项目的可行性

风险分析识别可能导致开发困难的需求，需要确定优先级，以便在项目面临延迟时能够容易地重新处理项目

风险类别：技术， 性能 ， 安全 ， 数据库完整性 ， 开发过程。

技术风险，需求在技术上难以实现。

性能风险，需求实现后，会延长长系统的响应时间。

安全风险，需求实现后，会破坏系统的安全性。

数据库完整性风险，需求不不容易易验证，并且可能导致数据不不一一致性。

开发过程风险，需求要求开发人人员使用用不不熟悉的非非常规开发方方法，如形式化规格说明方方法。

Discussion 3

Why is Change Management needed? Give the name of a tool used in Change Management.

问题1：一个需求可能会改变，被删除，或者可能会 添加一个新的需求， 下游变化成本，需要强有力的管理政策来记录更改请求， 评估变更的影响并实施更改。

问题2：软件配置管理工具。

第四章

4.1

Discussion 1

What is a UML Stereotype? Using the University Enrollment case study give an example of a stereotype.

在拓扑中，构造型是您应用于单元或功能（但不是需求）以指示单元或功能的用途的字符串。

构造型是用户生成的字段，用于描述有关单位或能力的用途、专用类型或其他信息。 通常，当您从拓扑外部的构造型资源创建单元时，构造型会自动添加到单元中。 但是，您可以添加自己的构造型来指示单元或能力的用途。 在拓扑中，构造型可以是任何字符串。

Discussion 2

Explain what a constraint is in the context of a UML class model. Give an example of a class constraint using the University Enrollment case Study.

对象约束语言简称OCL(Object Constraint Language)，它是一种用于施加在指定的模型元素上约束的语言。OCL表达式以附加在模型元素上的条件和限制来表现对该对象的约束，其中包括附加在模型元素上的不变量或约束的表达式，附加在操作和方法上的前置条件和后置条件等。

Discussion 3

What is a constrained association? Give an example.

限定关联是在二二元关联的一一端，有一一个属性框(限定词)，框中包含一一个或多个属性（二）。这些属性可以作为一一个索引码，用用于穿越从被限定的源类到关联另一一端的目目标类的关联。例例如，Flight和Passenger之间是多对多的（哈）。然而，当类Flight被属性seatNumber和departure限定时，关联的多重性就降为一对一。由限定词(flightNumber+seatNumber+departure)引入入的组合索引码能够被链接到只有零或一一个Passenger对象上。

Discussion 4

Describe tags and give an example of their use.

标签与约束相似，表示模型中的任意文本信息，写在大括号中。形如：tag = value。理解标签的概念需要区分标签定义和标签值。 标签定义是构造型的一个特性，显示为含有构造型声明的类矩形中的一个属性。 “标签值是一个名-值对，附属于一个模型元素，该模型元素使用了包含标签定义的构造型”

Discussion 5

What are the types of visibility for class attributes and methods?

类的属性和方法的可见性有public，protected，private和包可见性四种。

（1）public 除了类本身以外，属性和方法对其他模型元素也是可看见的。

（2）protected 属性和方法只对类本身，它的子类或友元（取决于具体语言二）是可看见的。

（3）private 属性和方法只对类本身和类的友元（取决于具体语言）是可视的。

（4）包可见性 如果对JAVA的属性或操作没有指定private、public或者protected关键字（或对整个类都没有指定public关键字哈），那么他们默认取得的可见性就是包可见性

Discussion 6

Should you ever use the friend relationship? Explain.

True friends help each other, not take advantage of each other unilaterally.

Discussion 7

Can a subclass access a protected data member or method of a parent class? Give an example.

子类可以“直接访问” public， protect的字段和方法，不能“直接访问” private的字段和方法。

Discussion 8

What is derived information? How can it be represented in a UML class diagram? Give an example.

导出信息是一种(最经常)应用于属性或关联的约束。导出信息从其他模型元素计算得到。严格地说,导出信息在模型中是冗余的--他可以在需要时计算出来。 导出信息的UML表示法就是在导出属性名或者关联名前加一条斜线。

Discussion 9

Describe qualified associations and association classes with examples.

在计算机面向对象的体系结构中，具有关联类。关联类既是类也是关联。它有着关联和类的特性。它将多个类连接起来同时又具有属性和操作

4.2

Discussion 1

Compare and contrast inheritance to encapsulation.

封装要求只能通过对象接口中的操作才能访问对象的状态（属性）。 继承允许子类直接访问保护属性，削弱了封装。

封装的定义：封装是实现面向对象程序设计的第一步（二），封装就是将数据或函数等集合在一个个的单元中（我们称之为类）。被封装的对象通常被称为抽象数据类型。

继承的特性：继承就是一种由已有的类创建新类的机制，是指在已有类的基础上扩展功能。继承中分为子类和父类，类可以有两种重要的成员：成员变量和方法。子类的成员中有一部分是子类自己声明定义的，另一部分是从它的父类继承的。其中子类继承父类中所有的属性和方法，但是对于private的属相和方法，由于这个是父类的隐私，所以子类虽然是继承了，但是没有可以访问这些属性和方法的引用。子类继承父类的成员变量作为自己的一个成员变量，就好象它们是在子类中直接声明一样，可以被子类中自己声明的任何实例方法操作（哈）。子类继承父类的方法作为子类中的一个方法，就象它们是在子类中直接声明一样，可以被子类中自己声明的任何实例方法调用。

继承的作用：继承主要实现重用代码，节省开发时间。

封装的作用：封装的意义在于保护或者防止代码（数据）被我们无意中破坏。在面向对象程序设计中数据被看作是一个中心的元素并且和使用它的函数结合的很密切，从而保护它不被其它的函数意外的修改。

Discussion 2

Explain how interface inheritance can be used to achieve the effect of multiple inheritance in languages such as Java.

在类名后面添加关键字implements，后面可跟多个接口名，接口名之间用逗号隔开。就可以达到多重继承的效果。

Discussion 3

Explain three problems that can occur with implementation inheritance.

脆弱的基类：对基类公共接口的改变使得子类中存在巨大的不可测风险。
重载和回调：向下回调和向下回调的结合引入了难以处理的循环依赖。

多重实现继承：多重继承引起复杂性的内在增长。

4.3

Discussion

Which kind of aggregation needs to be specified with the “frozen” constraint?

ExclusiveOwns聚合需要用{frozen}约束来指定。

4.4

Discussion

Explain the difference between synchronous and asynchronous messaging

同步与异步消息的区别

1、同步消息

同步消息传递涉及到等待服务器响应消息的客户端。消息可以双向地向两个方向流动。本质上，这意味着同步消息传递是双向通信。即发送方向接收方发送消息，接收方接收此消息并回复发送方。发送者在收到接收者的回复之前不会发送另一条消息。 同步执行的特征为：在两个通信应用系统之间必须要进行同步， 两个系统必须都在正常运行， 并且会中断客户端的执行流， 转而执行调用。发送程序和接收程序都必须一直做好相互通信的准备。发送程序首先向接收程序发起一个请求（发送消息）。发送程序紧接着就会堵塞它自身的进程， 直到收到接收程序的响应。发送程序在收到响应后会继续向下进行处理。

2.异步消息

异步消息传递涉及不等待来自服务器的消息的客户端。事件用于从服务器触发消息。因此，即使客户机被关闭，消息传递也将成功完成。异步消息传递意味着，它是单向通信的一种方式，而交流的流程是单向的。 当使用异步消息传送时， 调用者在发送消息以后可以不用等待响应， 可以接着处理其他任务。对于异步通信， 一个应用程序（请求者或发送者）将请求发送给另一个应用程序， 然后可以继续向下执行它自身的其他任务。发送程序无须等待接收程序的执行和返回结果， 而是可以继续处理其他请求。与同步方式不同， 异步方式中两个应用系统（发送程序和接收程序）无须同时都在运行， 也无须同时都在处理通信任务。

同步和异步消息传递的优点和缺点

异步消息传递有一些关键优势。它们能够提供灵活性并提供更高的可用性——系统对信息采取行动的压力较小，或者以某种方式立即做出响应。另外，一个系统被关闭不会影响另一个系统。例如，电子邮件——你可以发送数千封电子邮件给你的朋友，而不需要她回复你。 异步的缺点是它们缺乏直接性。没有直接的相互作用。考虑一下与你的朋友在即时通讯或电话上聊天——除非你的朋友及时回复你，否则

这不是聊天或谈话。 异步消息传递允许更多的并行性。由于进程不阻塞，所以它可以在消息传输时进行一些计算。

第五章

5.2 Multilayer Logical Architecture

Discussion 1

In a system model can can objects be allowed to communicate in an unrestricted fashion? Explain your answer.

在系统模型中，是否允许对象以不受限制的方式进行通信？解释你的答案.

当然不可以。如不受限制的话，随着新对象的增加，对象之间的通信路径数量呈指数增长。

对象数量较大时，复杂性极高。

Discussion 2

In system modeling what benefits do hierarchies have over networks? Explain your answer.

在系统建模中，层次结构在网络上有什么好处？解释你的答案。

1)以功能作为划分层次的基础。

2)第n层的实体在实现自身定义的功能时，只能使用第n-1层提供的服务。

3)第n层在向第n+1层提供的服务时，此服务不仅包含第n层本身的功能，还包含由下层服务提供的功能。

4)仅在相邻层间有接口，且所提供服务的具体实现细节对上一层完全屏蔽。 层次结构将复杂性从指数级降低到多项式级

Discussion 3

What does the term “complexity in the wires” mean?

“电线中的复杂性”一词是什么意思?

路径的数量，链接的数量。

Discussion 4

Explain Structural Complexity. Discuss the structural complexity of hierarchies compared to networks.

解释结构的复杂性。与网络相比，讨论层次结构的结构复杂性。

ediator helps in establishing loosely coupled communication between objects and helps in reducing the direct references to each other. This helps in minimizing the complexity of dependency management and communications among participating objects. Mediator helps to facilitate the interaction between objects in a manner in that objects are not aware of the existence of other objects. Objects depend only on a single mediator class instead of being coupled to dozens of other objects. During designing a solution to a problem, if you encounter a situation where multiple objects need to interact with each other to process the request, but direct communication may create a complex system, you can consider using mediator pattern. The pattern lets you extract all the relationships between classes into a separate class, isolating any changes to a specific component from the rest of the components.

EDITOR有助于在对象之间建立松散耦合的通信，并有助于减少对彼此的直接引用。这有助于最小化依赖关系管理和参与对象之间通信的复杂性。中介有助于以对象不知道其他对象存在的方式促进对象之间的交互。对象只依赖于一个中介类，而不是耦合到几十个其他对象。在设计一个问题的解决方案时，如果遇到多个对象需要相互交互来处理请求的情况，但是直接通信可能会创建一个复杂的系统，那么您可以考虑使用中介模式。该模式允许您将类之间的所有关系提取到一个单独的类中，从而将对特定组件的任何更改与其余组件隔离开来。

Discussion 5

What is a design pattern? Are design patterns always architectural patterns? Give an example of an architectural pattern. Give three examples of design patterns.

什么是设计模式？设计模式总是架构模式吗？给出一个架构模式的示例。给出三个设计模式的例子。

设计模式可以在某些情况帮助架构软件的静态结构。而架构的范围要大一些，更高层一些，考虑的更多的是非常重要的全局性的设计决策。一般好的（静态）架构可以尽量使变化发生在局部（模块内）而不影响整个系统。架构上的变化往往成本会非常高。而且设计模式只有一些是适用于架构的，还有一些只是用于具体的类设计的，剩下的一些则只是克服编程语言的限制而已。

分层模式：这种模式也称为多层体系架构模式。它可以用来构造可以分解为子任务组的程序，每个子任务都处于一个特定的抽象级别。每个层都为下一个提供更高层次服务。一般信息系统中最常见的是如下所列的4层。表示层(也称为UI层)、应用层(也称为服务层)、业务逻辑层(也称为领域层)、数据访问层(也称为持久化层)。

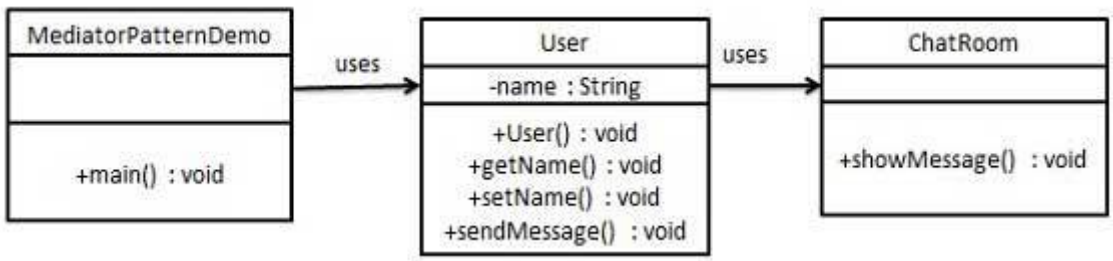
Discussion 6

What are the benefits to using the Mediator pattern? Use a diagram to explain your answer.

使用中介模式的好处是什么？用图表来解释你的答案。

中介者模式（Mediator Pattern）是用来降低多个对象和类之间的通信复杂性。这种模式提供了一个中介类，该类通常处理不同类之间的通信，并支持松耦合，使代码易于维护。中介者模式属于行为型模式。优点： 1、降低了类的复杂度，将一对多转化成了一对一。 2、各个类之间的解耦。 3、符合迪米特原则。缺点：中介者会庞大，变得复杂难以维护。意图：用一个中介对象来封装一系列的对象交互，中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。主要解决：对象与对象之间存在大量的关联关系，这样势必会导致系统的结构变得很复杂，同时若一个对象发生改变，我们也需要跟踪与之相关联的对象，同时做出相应的处理。何时使用：多个类相互耦合，形成了网状结构。

图表如下：



5.3 Architectural Modeling

Discussion

Can a components state change? Can a class be implemented by more than one component?

组件状态是否可以更改？一个类可以由多个组件实现吗？

问题1：可以，构建没有持久状态，不能与它的拷贝区分开来。

问题2：可以，一个类可以被多个构件实现，接口也可以。

5.4 Principles of Program Design and Reuse

Discussion 1

State the Law of Demeter and the Strong Law of Demeter.

陈述德米特定律和德米特定律。

迪米特法则(Law Of Demeter)的定义：如果两个软件实体无须直接通信，那么就不应当发生直接的相互调用，可以通过第三方转发该调用。

优点：

1、降低了类之间的耦合度，提高了模块的相对独立性。

2、由于亲合度降低，从而提高了类的可复用率和系统的扩展性。

Discussion 2

Why is mixed-instance cohesion necessary? Give an example using a diagram.

为什么混合实例内聚是必要的？用图表举例说明。

Code that is organized by functionality and does only one task is said to have high cohesion. Highly cohesive code is reusable, simple, and easy to understand. It also creates objects that are small and focused.

Code that is organized arbitrarily and has multiple tasks is said to have low cohesion. Such code is difficult to understand, maintain, and reuse, and is often complex. It also creates objects that are large and unfocused.

Having high cohesion is generally good, while having low cohesion is generally bad. When writing code, always strive to write highly cohesive code.

按功能组织的代码，只执行一项任务

具有高度的凝聚力。高度内聚的代码是可重用的、简单的、高效的

容易理解。它还可以创建小而集中的对象。

任意组织并具有多个任务的代码是

据说凝聚力很低。这样的代码很难理解，

维护、维护和重用，并且通常是复杂的。它还创建了

是大而不集中的。

具有高内聚性通常是好的，而具有低内聚性则是好的

内聚性通常很差。在编写代码时，始终努力编写

高度内聚的代码。

5.5 Collaboration Modeling

Discussion

In what collaboration model must the roles always be typed? Do collaboration models identify messages?

在什么协作模型中必须始终键入角色？协作模型是否识别消息？

问题1：复合结构模型。

问题2：协作模型不标识消息。

第七章

7.1 Business Objects and Persistence

Discussion

Explain the three levels of data models.

解释数据模型的三个层次。

数据模型所描述的内容包括三个部分：数据结构、数据操作、数据约束。

1、数据结构

主要描述数据的类型、内容、性质以及数据间的联系等，是目标类型的集合。目标类型是数据库的组成成分，一般可分为两类：数据类型、数据类型之间的联系。

数据类型如DBTG(数据库任务组)网状模型中的记录型、数据项，关系模型中的关系、域等。联系部分有DBTG网状模型中的系型等。数据结构是数据模型的基础，数据操作和约束都基本建立在数据结构上。不同的数据结构具有不同的操作和约束。

2、数据操作

数据模型中数据操作主要描述在相应的数据结构上的操作类型和操作方式。它是操作算符的集合，包括若干操作和推理规则，用以对目标类型的有效实例所组成的数据库进行操作。

3、数据约束

数据模型中的数据约束主要描述数据结构内数据间的语法、词义联系、他们之间的制约和依存关系，以及数据动态变化的规则，以保证数据的正确、有效和相容。它是完整性规则的集合，用以限定符合数据模型的数据库状态，以及状态的变化。

约束条件可以按不同的原则划分为数据值的约束和数据间联系的约束；静态约束和动态约束；实体约束和实体间的参照约束等。

7.2 Relational Database Model

Discussion 1

Explain columns, domains and rules. Using a diagram give an example to show how they are used

解释列、域和规则。使用图表举例说明如何使用它们

Explainnations：

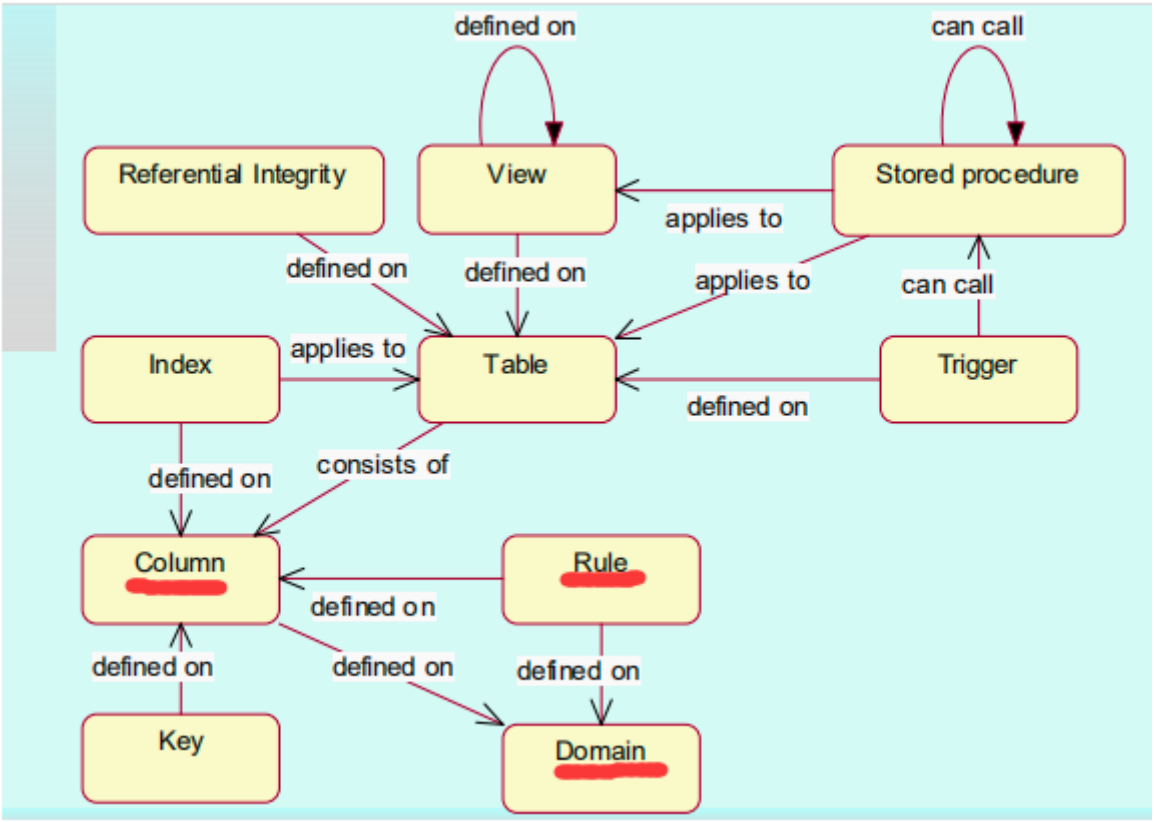
- 列、域和规则
  - 关系数据库采用由列和行组成的表来定义数据，行列的交叉处的数值必须是原子的；
  - 域定义了一个列可以取值的合法集
    - 域可以是匿名(gender char(1))，也可以被命名(gender Gender);

      > 域定义：create domain Gender char(1);

    - 一个命名域的使用可以在不同表的多个列定义中；
  - 可以为列和域设置业务规则来约束它们，业务规则中可以定义：
    - 默认值
    - 取值范围
    - 值的列表
    - 值的大小写
    - 值的格式(例如，“K”开头)

      > 只涉及单个列或域的较简单的业务规则可以在规则机制中定义，涉及多个表的更复杂的规则可以定义为引用完整性约束，定义业务规则的最佳机制是触发器。

Usage：



Discussion 2

What is referential integrity? How is it useful for the mapping from a UML class model? Explain the four declarative referential integrity constraints.

什么是引用完整性？它对从UML类模型映射有用吗？解释四个声明性引用完整性约束。

Definition：主码与外码之间的参照与被参照关系；

Four declarative referential integrity constraints：

Upd(R)、Del(R)：限制更新或删除操作

Upd(C)、Del(C)：级联操作

Upd(N)、Del(N)：设置为空值

Upd(D)、Del(D)：设置为缺省值

Discussion 3

What is a trigger? How is it related to referential integrity? Can a stored procedure call a trigger? Explain.

什么是触发器？它与引用完整性有什么关系？存储过程可以调用触发器吗？解释

Definition：触发器是一个小程序，用扩展的SQL语句编写，当定义了触发器的表发生修改操作时自动执行(触发)，修改操作可以是任何SQL修改语句——insert、update或delete；

Relation：触发器可以用来实施更复杂的引用完整性约束；

Reason：触发器是一种不能被调用的特殊存储过程

Discussion 4

What mathematical concept is a relational database model based on?

关系数据库模型基于什么数学概念?

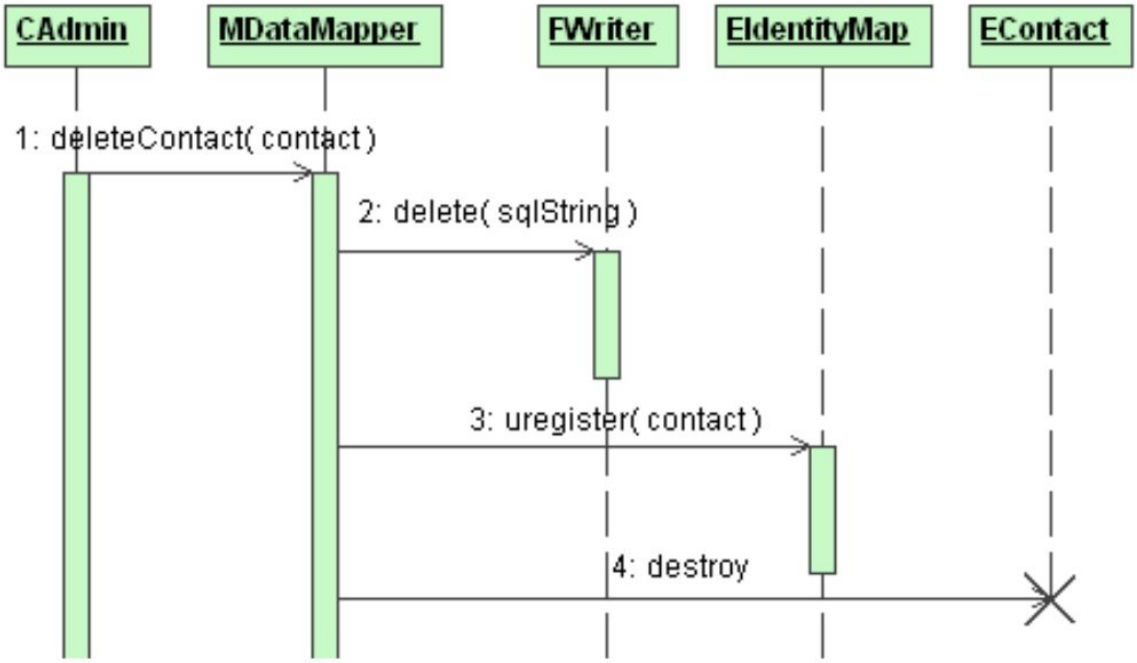
集合 Set

7.3 Object-Relational Mapping

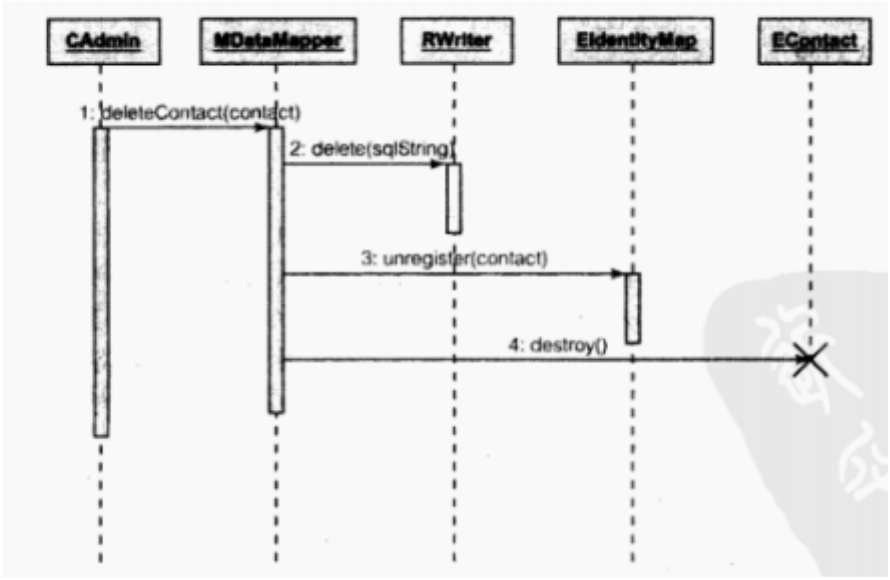
Discussion 1

Referring to the following figure, consider a situation in which an entity object needs to be unloaded to the database as a result of an update operation, instead of delete. How would the sequence diagram differ in such a case?

参考下面的图，考虑一个实体对象需要作为更新操作的结果卸载到数据库的情况，而不是删除。在这种情况下，序列图会有什么不同?



- 释放持久对象(Unloading persistent objects)
  - 释放(或称为检入)是装载的反操作;
  -



Discussion 2

Describe briefly the locks in pessimistic concurrency control.

简要描述悲观并发控制中的锁。

总是假设最坏的情况，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁， 这样别人想拿这个数据就会阻塞直到它拿到锁（共享资源每次只给一个线程使用，其它线程阻塞，用完后再把资源转让给其它线程）。 传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。 Java中synchronized和ReentrantLock等独占锁就是悲观锁思想的实现。

Discussion 3

What is a compensating transaction? How can it be used in program design?

什么是补偿交易？如何在程序设计中使用它?

补偿事务是当一个或多个执行步骤执行失败时，通过一系列步骤撤销操作来总体定义一个最终一致性的操作。 利用日志，事务完成标注成功，否则失败，失败后回滚。随着初始操作的执行，系统会记录每一个步骤的消息及其撤销方式。如果操作失败，则按照之前记录的步骤倒带回去，并执行撤销操作。如果多个步骤可以并行，可以对其撤销步骤进行并行处理。

7.4 Patterns for Managing Persistent Objects

Discussion 1

Describe the five levels of SQL programming interfaces.

描述SQL编程接口的五个级别。

第1层SQL用做数据定义语言(DDL)：DDL是定义数据库结构(数据库模式)的规格说明语言。数据库设计者和数据库管理员(DBA) 是第1层SQL的主要用户；

* 第2层SQL用做数据操纵语言(DML) 或查询语言；

* 第3层SQL被嵌入在常规的程序设计语言中，如C或COBOL。由于程序设计语言编译器不理解SQL，因此需要一个预编译器(预处理器)来将SQL语句翻译成对DBMS供应商

提供的DB库函数的调用。程序员可以选择直接使用DB库函数来编程，这样就可以不需要预编译器。

* 第4层SQL采用了与第3层同样的策略，将SQL嵌人到客户端程序中。但第4层SQL提供了更强的程序设计环境:应用生成器或第四代语言(4GL)。 4GL配置了屏幕界面和UI构建能力，由于IS应用系统要求复杂的GUI,所以4GL/SQL常常是构建这种应用的首选。

* 第5层SQL除了具有第3层和第4层的功能外，还可以将一些SQL语句从客户端程序移动到动态的(可编程的)服务器端数据库。SQL可以作为程序设计语言(例如Oracle中的

PL/SQL或Sybase与SQL Server中的事务SQL)来使用。就像8.2.5节介绍的那样，客户端程序可以调用服务器端程序。

Discussion 2

Can the amount of database recovery time be controlled by the designer/DBA? Explain.

数据库恢复时间是否可以由设计器/DBA控制？解释

可以，通过设置检查点出现的频率。

7.5 Designing Database Access Transactions

Discussion 1

Describe briefly the levels of transaction isolation.

简要描述事务隔离的级别。

未授权读取 也称为读未提交（Read Uncommitted）：允许脏读取，但不允许更新丢失。如果一个事务已经开始写数据，则另外一个事务则不允许同时进行写操作，但允许其他事务读此行数据。该隔离级别可以通过排他写锁实现。 授权读取 也称为读提交（Read Committed）：允许不可重复读取，但不允许脏读取。这可以通过瞬间共享读锁和排他写锁实现。读取数据的事务允许其他事务继续访问该行数据，

但是未提交的写事务将会禁止其他事务访问该行。 可重复读取（Repeatable Read） 可重复读取（Repeatable Read）：禁止不可重复读取和脏读取，但是有时可能出现幻读数据。这可以通过共享读锁和排他写锁实现。读取数据的事务将会禁止写事务（但允许读事务），写事务则禁止任何其他事务。 序列化（Serializable） 序列化（Serializable）：提供严格的事务隔离。它要求事务序列化执行，事务只能一个接着一个地执行，不能并发执行。仅仅通过行级锁是无法实现事务序列化的，必须通过其他机制保证新插入的数据不会被刚执行查询操作的事务访问到。

Discussion 2

Explain what a long transaction is and what they are used for.

解释什么是长事务以及它们的用途。

长事务，顾名思义就是一个事务执行了很长时间仍未结束。那么一个事务执行多长时间算是长事务？这在不同的数据库产品中有不同的定义，对于informix数据库来说占用逻辑日志个数的百分比达到长事务高水位线就被定义为一个长事务，而在oracle中一个事务执行时间超过6秒钟就被认为是长事务。一般来说在一个联机事务处理系统中事务执行时间应该控制在一定范围内，如果一个事务执行时间太长，会长时间锁定某些资源，不利于并发处理，有时还会因为某些资源耗尽导致数据库挂起。