1. What is object technology? What do you perceive as object technology’s strength? It’s weakness?

(1) object technology是指导软件构造的一组原则（抽象、封装、多态），以及支持这些原则的语言、数据库和其他工具。

(2)优点：Reflects a single paradigm.

Facilitates architectural and code reuse.

Reflects real world models more closely.

Encourages stability.

Is adaptive to change

1. What is UML? List at least three benefits of developing with UML.

(1)UML是一种统一建模语言，用于可视化、指定、构造和记录软件密集型系统的构件。

(2)可以直接与各种编程语言连接。可构建精确完整的模型。对于系统架构、需求、测试、项目规划和发布需求有明确的文档描述。

1. What process characteristic best fit the UML? Describe each characteristic.

用例驱动、以架构为中心、迭代增量。

1. What is a use-case driven process? What is use-case? What are the benefits of use case?

(1)用例定义为整个开发进程打好了基础。

(2) 在不展现一个系统或子系统内部结构的情况下，对系统或子系统的某个连贯的功能单元的定义和描述。

(3)简洁方便、易于理解。

1. What is system’s architecture? What is an architecture-centric Process?

一个系统的架构被用作一个主要的工件，用于对正在开发的系统进行概念化、构造、管理和演化。

1. What is iteration? What are the benefits of Iterative Development?

（1）把开发组织成一系列固定的短期的小项目。

（2）在进行大规模投资之前，关键风险得到解决。

初始迭代允许早期用户反馈。

测试和集成是连续的。

目标里程碑着眼于短期。

进度是通过评估实现来衡量的。

可以部署部分实现。

1. What are the basic principles of OO technology? Describe each in detail.

(1) 抽象、封装、模块化、层次结构

(2) 抽象：不是一种具体的表现形式，表示某种事物的理想本质。

封装：对客户端隐藏实现。

模块化：把复杂的事情分解成可管理的部分。

层次结构：帮助人们理解复杂的系统。

1. What is use case model? Which artifacts can be included in a use case model?

(1) 根据用例描述系统功能需求的模型。

(2) use case、actors.

1. List three types of relationships existed between different use cases and give examples.

用例泛化、包含用例、扩展用例。

1. Explain the following diagram and their elements with examples.

1) Use case diagram

用例为参与者和系统之间的对话建模。

一个用例由一个参与者发起，以调用系统中的某个功能。

2) Activity diagram

用例模型中的活动图可用于捕获用例中的活动。本质上是一个流程图，显示了从一个活动到另一个活动的控制流程。

3) Sequence diagram

强调消息时间顺序的交互图。

4) Collaboration diagram

强调参与交互的对象的组织的图。

5) Class diagram

系统的静态视图。包括系统词汇表、协作、逻辑数据库模式。

6) state chart diagram

显示状态机。

7) Deployment diagram

显示运行时处理节点的配置、节点之间的通信链路、组件实例和驻留在它们上的对象。

1. Describe the similarities and differences between the sequence diagram and collaboration diagram.

两者可互相转换而不丢失信息。

协作关系图除了互动之外，还要展示人际关系。能够更好地可视化协作模式。更适合于可视化给定对象上的所有效果。更容易用于头脑风暴会议。

序列图显示消息的显式序列、控制焦点。更好地可视化整体流。更适合实时规范和复杂场景。

1. Define the different relationships in class diagram: dependency, association, aggregation, composition, generalization.

Dependency：一个类的改变可能影响或提供信息给其他类。

Association：两个类在概念上有连接关系时，类之间的连接称为关联；提供了不同类的对象可以相互作用的连接。

Aggregation：一种特殊的关联形式，在聚合（整体）及其部分之间建立一个整体-部分关系的模型，体现了一种层次结构，整体类位于部分类的上层，多个部分类处于并列的层次。

Composition：是整体与部分的关系，组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

Generalization：其中一个类共享一个或多个类的结构和/或行为。是“is a kind of”关系

1. What is a node in deployment diagram? List two different types of nodes.

(1) 运行时存在的表示计算资源的物理元素。

(2) Processor Node，Device Node

1. Describe the mechanisms of extensibility of UML.

构造型stereotype（表示新的建模元素）

标记值tagged value （表示新的建模属性）

约束constraint （表示新的建模语义）

1. What is the function of Stereotypes？Give two examples of stereotypes.

stereotype是UML词汇的扩充，允许你创建新的UML建筑块，这些新的建筑块和原有的类似，但特别针对你自己的问题。

1. Explain the six best practices of software engineering.

迭代的开发软件Develop Iteratively

需求管理Manage Requirements

使用基于构件的体系结构Use Component Architectures

可视化软件建模Model Visually (UML)

验证软件质量Continuously Verify Quality

控制软件变更Manage Change

1. What is RUP? How many phases are there in RUP? Describe each phase’s purpose and milestone.

RUP试图总结现代软件开发过程中所有好的实践经验，形成一种有很强适应性的软件开发过程。

4。

初始阶段: Inception目标是为系统建立商业案例和确定项目的边界。

细化阶段：Elaboration目标是分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中最高风险的元素。

构建阶段：Construction所有剩余的构件和应用程序功能被开发并集成为产品,所有的功能被详尽的测试。

交付阶段：Transition目的是将软件产品交付给用户群体。

1. Name and briefly describe the “4+1”views of architecture.

Use-case view，Logical view，Implementation view，Process view，Deployment view

1. What is the difference between analysis and design?

（1）分析注重于理解解析问题，设计注重于问题的解决。

（2）分析时的设计不会很详细，设计时会把操作和属性都定好。

（3）分析有关系统架构，设计接近真正的代码。

（4）分析是分析功能性的需要，设计是设计对象的生命周期。

1. Please describe the whole process of OO analysis and design with UML.

（1）Key Concepts

Define the High-Level Organization of Subsystems

Identify Key Abstractions

Create Use-Case Realizations

Checkpoints

（2）

Identify classes and subsystems

Identify subsystem interfaces

Update the organization of the Design Model

Checkpoints

1. What is a layered architecture? Give examples of typical layers.

Application subsystems->Bussiness Specific->Middleware->System Software

1. What are analysis mechanisms? What are design mechanisms? Give examples.

分析主要的产出是序列图和状态机图，表达后台对象（类的实例）如何交互和协作完成用例所规定的功能；

设计则更贴近实现，与实现所基于的硬件和软件环境有关，需要解决具体的界面、输入、输出和对象持久化的问题，应该是基于一个框架或实现模式来进行，需要确定每个类和消息的细节。

1. What is an analysis class? Name and describe the three analysis stereotypes. Give examples.

(1)分析类直接与应用逻辑相关，而不关注技术实现的问题。

(2) Boundary Class: Intermediates between the interface and something outside the system.

Entity Class: Key abstractions of the system.

Control Class: Use-case behavior coordinator.

1. What is Use-case realization? What‘s your understandings about the benefit of the use-case realization structure.

(1) Use case realization是Use case 的实现, 通过描述这些抽象元素的协作关系来分析实现方式以及进一步细化。

(2) 目的为了把需求和实现分离。一个用例实现可以实现几个用例，一个用例也可以由多个用例实现来实现。这样具体的实现方案就可以不过分依赖于需求阶段的用例划分。

1. Describe the steps occurred in the use-case analysis.

补充用例描述

对于每个用例实现：从用例行为中查找类，将用例行为分发到类

对于每个结果分析类：描述职责、描述属性和关联、限定分析机制

统一分析类

检查点

1. What’s the package, and why we need package?

(1) 包是将元素组织成组的通用机制。

(2)易于管理，可用于组织正在开发的模型。

1. What is a subsystem? What is an interface? How does a subsystem differ from a package?

(1) 子系统是一种模型元素，它具有包（其中可包含其他模型元素）和类（其具有行为）的语义。子系统的行为由它所包含的类或其他子系统提供。子系统实现一个或多个接口，这些接口定义子系统可以执行的行为。

(2) 接口指实体把自己提供给外界的一种抽象化物（可以为另一实体），用以由内部操作分离出外部沟通方法，使其能被内部修改而不影响外界其他实体与其交互的方式。

(3) Is a “cross between”a package and a class

Realizes one or more interfaces that define

1. What is the purpose of describing the run-time architecture? How to model the process view?

(1) 分析并发需求。识别进程和线程。确定流程生命周期。将流程映射到实现上。在流程之间分配模型元素。

(2) 流程可以使用类图、交互图和组件建模。

1. What is the purpose of describing the distribution? How to model the deployment view?

(1) 减少处理器负载、特殊加工要求、规模化关系到的经济问题、对系统的分布式访问。

(2)

1. Describe the 3 typical distribution patterns, C/S , B/S and P2P.

C/S：客户端/服务器模式

B/S：浏览器/服务器模式

P2P：点对点模式

1. What is the agile process model? What is Scrum？What is XP？What are the differences among Scrum, XP and RUP?

(1) 敏捷过程模型：强调项目的灵活性，并在一些基本原则的指导下，采用非正式的方式（支持者认为这并不会降低过程的有效性）执行软件过程。

(2) Scrum是迭代式增量软件开发过程，通常用于敏捷软件开发。Scrum包括了一系列实践和预定义角色的过程骨架。

(3)XP是一门针对业务和软件开发的规则，它的作用在于将两者的力量集中在共同的、可以达到的目标上。

(4) XP希望通过充分的交流和沟通，使产品的设计尽可能简单明了，同时通过客户经常性的反馈，生产出符合客户要求的软件产品，并且有勇气迎接需求的改变。RUP试图总结现代软件开发过程中所有好的实践经验，形成一种有很强适应性的软件开发过程。Scrum拥有明确的最高目标——发布产品的重要性高于一切，团队高度自治，成员们熟悉开发过程中涉及的各种技术，紧密合作，确保每个迭代都朝着最高目标推进。

1. What is the object oriented design approach of GRASP? Please give 9 GRASP patterns with examples and definitions.

（1）创建者(Creator)

问题：应该由谁来创建某类的实例

解决方案：满足以下条件，应该由A类实例来创建B类的实例。

- A包含或者聚集B

- A记录B

- A直接使用B

- A具有B的初始化数据。

（2）信息专家(Information Expert)

问题：给对象分配职责的基本原则是什么？

解决方案：把职责分配给具有完成职责所需信息的类。

（3）低耦合(Low coupling)

问题：如何减少变化带来的影响

解决方案：分配职责时，使耦合性尽可能低，用该原则评估方案。

（4）控制器(Controller)

问题：在UI层之上首先接受和协调系统操作的对象是什么？

解决方案：把职责分配给能代表以下选择之一的对象：

代表整个“系统”，“根对象”，运行软件的设备或主要子系统。

代表发生系统操作的用例场景。

（5）高内聚(High Cohesion)

问题：怎么保持对象保持有内聚、可理解和可管理，并且能够支持低耦合？

解决方案：分配职责可保持较高的内聚性。利用这一点来评估方案。

（6）多态性(Polymorphism)

问题：如何处理基于类型的选择？如何创建可插拔的软件构件？

解决方案：当相关选择或行为随类型有所不同时，使用多态操作为变化的行为类型分配职责。

（7）纯虚构(Pure Fabrication)

问题：当不想违背高内聚和低耦合或者其他目标，但是基于专家模式锁提供的方案又不合适时，哪些对象应该承担这一职责？

解决方案：人为的制造一个类分配一组高内聚的职责，该类并不代表问题领域里面的概念，是需要的类，用以支持高内聚和低耦合。

（8）间接性(Indirection)

问题：为了避免两个或者多个事物之间直接耦合，应该如何分配职责？如何使对象解耦，以支持低耦合并提高复用性潜力？

解决方案：将职责分配给中介对象，是其作为其他构件或服务之间的媒介，以避免它们之间的直接耦合，中介实现了其他构件之间的间接性。

（9）防止变异(Protected Variations)

问题：如何设计对象，子系统和系统，使其内部的变化或不稳定性不会对其他元素产生不良影响？

解决方案：识别变化或不稳定的地方，分配职责用以在变化地方创建稳定的接口。

1. What is the object oriented design approach of GoF? Please give 7 GoF patterns with examples and definitions.

(1) Adapter 适配器模式

解决不相容的接口问题，为具有不同接口的类似构件提供稳定的接口。

通过中介适配器对象(intermediate adapter object),将构建的原有插口转换为其他接口。

(2) Factory工厂模式

解决“当有特殊考虑(例如存在复杂创建逻辑、为了改良内聚而分离创建职责等)时，应该由谁来创建？”的问题。创建称为工厂的纯虚构对象来处理这些创建职责。

(3) Singleton单实例类模式

解决“谁来创建工厂自身，如何访问工厂？”的问题。需要支持单一实例(即单实例类)的全局可见性或单点访问。对类定义静态方法用以返回单实例。

如：Windows 是多进程多线程的，在操作一个文件的时候，就不可避免地出现多个进程或线程同时操作一个文件的现象，所以所有文件的处理必须通过唯一的实例来进行。

(4) Strategy策略模式

在单独的类中分别定义每种算法/策略/政策，并且使其拥有共同接口。

如：旅行的出游方式，每一种旅行方式都是一个策略

(5) Composition组合模式

定义组合和原子对象的类，使它们实现相同的接口。

(6) 外观模式

外部与一个子系统的通信必须通过一个统一的门面对象进行。门面模式提供一个高层次的接口，使得子系统更易于使用。

(7) 观察者模式

观察者模式定义了一种一队多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态上发生变化时，会通知所有观察者对象，使他们能够自动更新自己。

1. What is the relationships between the patterns of GoF and GRASP?

一些GRASP原则是对设计模式的归纳，GRASP从整体出发，GoF从细节出发。