**目录**

[Module 1: Requirements Overview 2](#_Toc26905038)

[Module 2: Analysis and Design Overview 3](#_Toc26905039)

[Module 3: Architectural Analysis 4](#_Toc26905040)

[Module 4: Use-Case Analysis 5](#_Toc26905041)

[Module 5: Identify Design Elements 6](#_Toc26905042)

[Module 6: Describe the Run-time Architecture 7](#_Toc26905043)

[Module 7: Describe Distribution 9](#_Toc26905044)

[Module 8: Use-Case Design 10](#_Toc26905045)

[Module 9: Subsystem Design 11](#_Toc26905046)

[Module 10: Class Design 11](#_Toc26905047)

[问答 13](#_Toc26905048)

[复习题 26](#_Toc26905049)

# Module 1: Requirements Overview

1. What is a Use-Case Model?

A model that describes a system’s functional requirements in terms of use cases;

根据用例描述系统功能需求的模型

A model of the system’s intended functionality(use cases) and its environment (actors).

一个系统的模型包含了功能（用例）和它的环境（角色）

1. What is an actor?

An actor represents anything that interacts with the system.

一个角色代表所有与系统交互的物体。

1. What is a use case? List examples of use case properties.

A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor.

用例是执行一系列的动作系统并产生一个可观察的结果值到一个特定的角色

1. What is the difference between a use case and a scenario?

A scenario is an instance of a use case.

一个脚本是用例的实例

1. What is a Supplementary Specification and what does it include?

Functionality、Usability、Reliability、Performance、Supportability、Design constraints、Supplementary、Specification

功能、可用性、可靠性、性能、可支持性、设计约束、可扩展性、规格

# Module 2: Analysis and Design Overview

1. What is the purpose of the Analysis and Design Discipline?

The purposes of Analysis and Design are to:

Transform the requirements into a design of the system-to-be.

把需求转换成系统设计

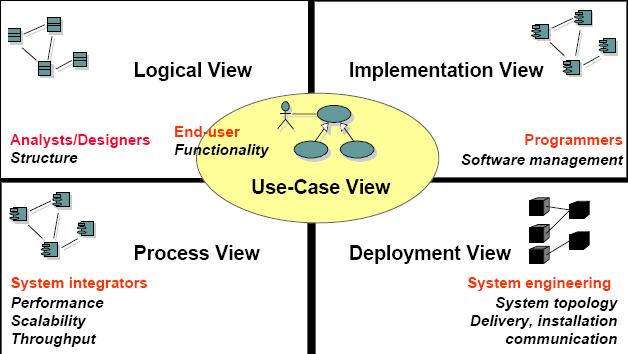
Evolve a robust(强健的) architecture for the system.

为系统开发一个强健的架构

Adapt the design to match the implementation environment, designing it for performance.

修改设计去与实现环境匹配并改善性能。

1. Name and briefly describe the 4+1 Views of Architecture.



逻辑视图：

实现视图

过程视图

部署视图

用例视图

1. What is the difference between Analysis and Design?

|  |  |
| --- | --- |
| Analysis | Design |
| * Focus on understanding the problem   关注如何理解问题   * Idealized design   理想化设计   * Behavior   行为   * System structure   系统架构   * Functional   功能   * Requirements   需求   * A small model   一个小模型 | * Focus on understanding the solution   关注如何理解解决方法   * Operations and attributes   操作与属性   * Performance   性能   * Close to real code   接近源码   * Object lifecycles   对象生命周期   * Nonfunctional requirements   无功能的需求   * A large model   一个大的模型 |

1. What is architecture?

Software architecture encompasses a set of significant decisions about the organization of a software system.

软件架构包含一系列软件系统组织的重要决策

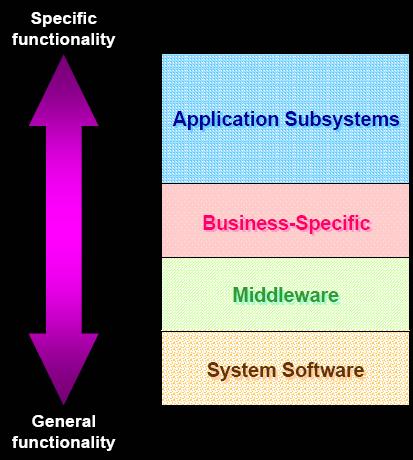
# Module 3: Architectural Analysis

1. What is a package?

A package is a general-purpose mechanism for organizing elements into groups. It is a model element that can contain other model elements.

一个包是一种组织元素进组群的通用机制

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



分层体系结构把系统分成若干个处理层次，例如：逻辑层，业务层，数据层的分层

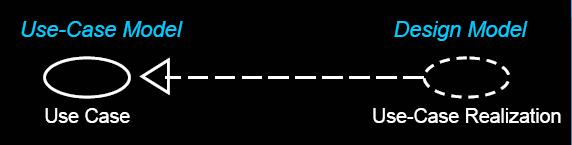
# Module 4: Use-Case Analysis

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

Boundary Class、Entity Class、Control Class.

边界类，实体类，控制类

1. What is a Use-Case Realization(实现)?



Use-Case Realization provides traceability from Analysis and Design back to Requirements.

用例的实现提供了从分析设计到需求的回溯。

1. How many Interaction diagrams should be produced during Use-Case Analysis?

Two.Collaboration Diagrams and Sequence Diagrams

2个。

时序图和协作图

# Module 5: Identify Design Elements

1. What is an interface?

Purpose:

To identify the interfaces of the subsystems based on their responsibilities

接口是一种类，他表示对一组紧凑的公共特征和职责的声明。

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

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

子系统与包在语义上具有差异：子系统是一种通过一个或多个它所实现的接口来提供行为的包。包并不提供行为；它们只不过是用来容纳提供行为的对象的容器。之所以要使用子系统而不使用包，是因为子系统完全封装自己的内容，只通过自己的接口提供行为。其好处在于，与包不同，只要子系统的接口保持不变，就可以完全自由地更改子系统的内容和内部行为。另外，子系统还提供了一种“可替换的设计”元素：任何两个实现相同接口的子系统（或类，就此而论）都可以互换

1. What is a subsystem used for, and how do you identify them?

Subsystems can be used to partition the system into parts that can be independently

子系统用于把系统分成几个独立的部分

如果某个协作中的各个类只是在相互之间进行交互，并且可生成一组定义明确的结果，就应将该协作和它的类封装在一个子系统中

1. What are some layering and partitioning considerations?

|  |  |
| --- | --- |
| Layering Considerations: | Partitioning Consid·erations |
| * Visibility   可见的   * Volatility   易变的   * Generality   概述的   * Number of layers | * Coupling and cohesion   耦合与内聚   * User organization   用户组织   * Competency and/or skill areas   能力或技能范围   * System distribution   系统分布   * Secrecy   保密性   * Variability   可变的 |

# Module 6: Describe the Run-time Architecture

1. What is a process? What is a thread?

|  |  |
| --- | --- |
| Process | Thread |
| * Provides heavyweight flow of control   提供强控制流   * Is stand-alone   独立的   * Can be divided into individual threads   可被独立线程划分 | * Provides lightweight flow of control   提供轻度控制流   * Runs in the context of an enclosing process   运行在封闭的进程上下文 |

1. Describe the two strategies for mapping classes and subsystems to processes.

Two Strategies (used simultaneously)：

1. Inside-Out

由内到外

Group elements that closely cooperate and must execute in the same thread of control

同组的关系密切的元素必须在同一个线程控制

Separate elements that do not interact

相互独立的元素不发生交互

Repeat until you reach the minimum number of processes that still provide the required distribution and effective resource utilization

重复直到进程的最少值仍然提供所需的分布和有效资源

1. Outside-In

由外到内

Define a separate thread of control for each external stimuli

为每一个外部刺激定义一个独立的线程控制

Define a separate server thread of control for each service

为定义每一个服务一个独立的服务线程控制

Reduce number of threads to what can be supported

减少线程数量去

1. How do you model the Process View? What modeling elements and diagrams are used?

Processes can be modeled using:

Active classes (Class Diagrams)类图 and Objects(Interaction Diagrams)交互图

Components (Component Diagrams)构件图

# Module 7: Describe Distribution

1. What is a node? Describe the two different “types” of nodes.
2. Node:

Physical run-time computational resource

Processor node处理器节点

Executes system software

执行系统软件

Device node设备节点

Support device

支持设备

Typically controlled by a processor

典型被处理器控制

1. Connection：连接

Communication mechanism

通信机制

Physical medium

物理媒介

Software protocol

软件协议

1. Describe some of the considerations when mapping processes to nodes.

Process-to-Node Allocation Considerations

1. Distribution patterns

分布式模式

1. Response time and system throughput

响应时间与系统生产量

1. Minimization of cross-network traffic

网络流量最小化

1. Node capacity

节点容量

1. Communication medium bandwidth

通信媒介带宽

1. Availability of hardware and communication links

硬件与通信线路的可用性

1. Rerouting requirements

重编路由需求

1. How do you model the Deployment View? What modeling elements and diagrams are used?

Deployment modeling elements contains node and connection. Reference to Question 1.

部署建模元素包含节点与连接。

# Module 8: Use-Case Design

1. What is meant by encapsulating(封装) subsystem interactions? Why is it a good thing to do?
2. Interactions can be described at several levels.

Subsystem interactions can be described in their own interaction diagrams.

子系统交互可以被他们的交互图描述。

1. Advantages of Encapsulating Subsystem Interactions:

Use-case realizations:

Are less cluttered

更少混论

Can be created before the internal designs of subsystems are created (parallel development)

在子系统开发前就能被创建出来（并行开发）

Are more generic and easier to change(Subsystems can be substituted.)

更简单作出改变（子系统可以被替换）

# Module 9: Subsystem Design

1. How many interaction diagrams should be produced during Subsystem Design?

One or more interaction diagrams per interface operation

# Module 10: Class Design

1. Are statecharts created for every class?

Statechart is a directed graph of states (nodes) connected by transitions (directed arcs). Statechart for one of the classes that exhibits state-controlled behavior.

状态图状态图是一个由弧线连接的有向图。状态图表现出状态控制的转变

1. What are the major components of a statechart? Provide a brief description of each.

Significant, dynamic attributes;

重要的。具有多种属性

Existence and non-existence of certain links;

存在与非存在的特定环节

Events

事件

Transitions

转移

Activities and Actions

活动与行为

1. What is the difference between an association and a dependency?
2. Dependency is a relationship between two objects.

从属是两个对象之间的关系

1. Associations are structural relationships

关联是结构化的关系

1. Dependencies are nonstructural relationships

从属不是结构化的关系

1. In order for objects to “know each other” they must be visible

为了让对象互相知道，他们必须是可见的

1. An instance of an association is a link
   * All links become associations unless they have global, local, or parameter visibility

所有成为关联的连接具有整体，局部，或参数的可见性

* + Relationships are context-dependent

关系是基于上下文的

1. Dependencies are transient links with:
   * A limited duration

有限的持续时间

* + A context-independent relationship

不基于上下文的关系

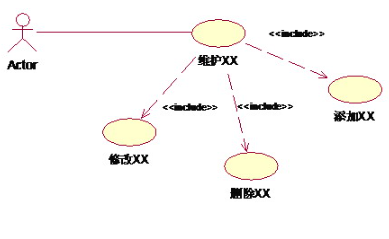
* + A summary relationship

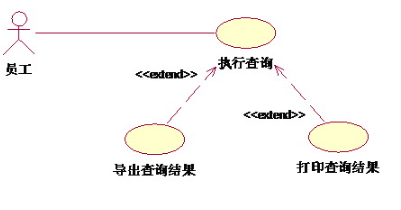
对关系的总结

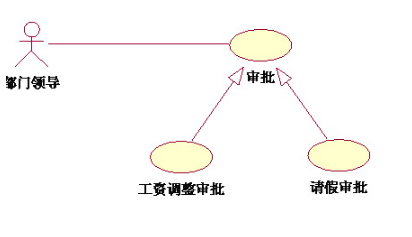
# 各种图

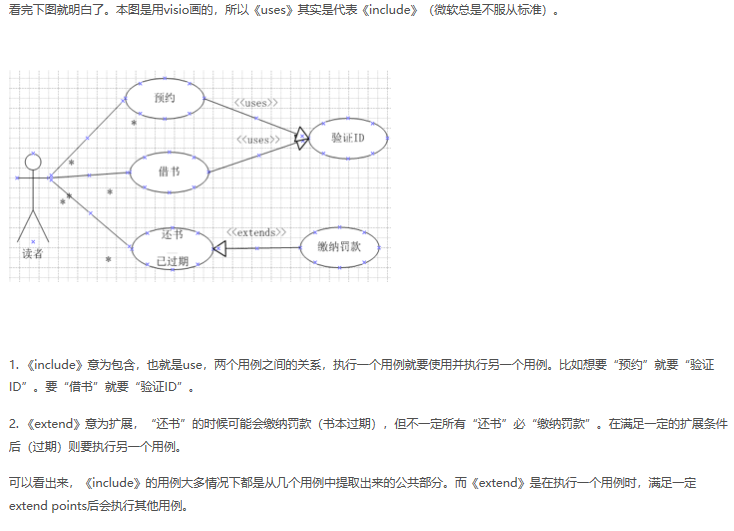
1. 用例图

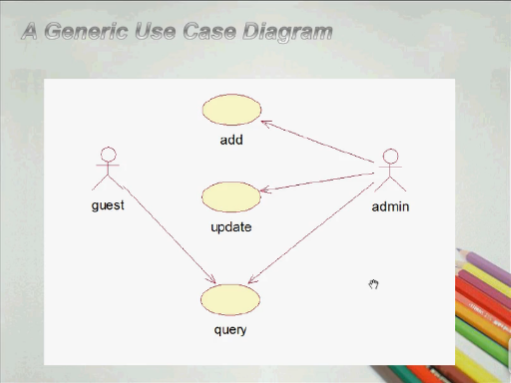
用例之间可以抽象出包含(include)、扩展(extend)和泛化(generalization)几种关系



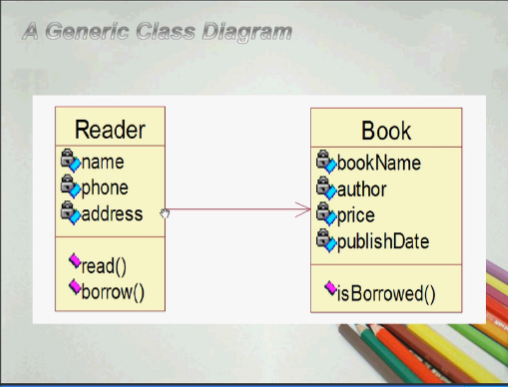


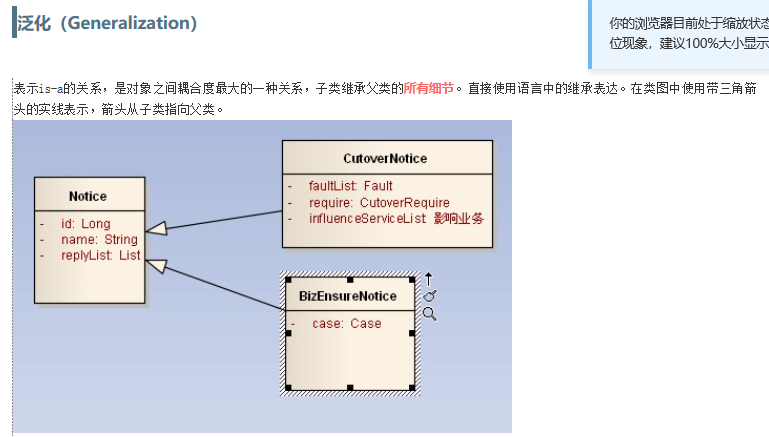


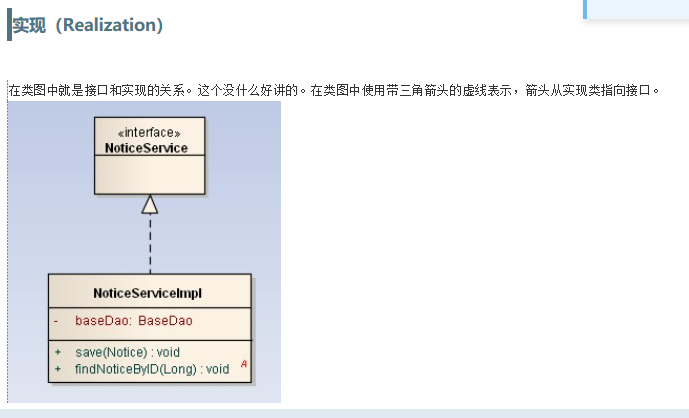


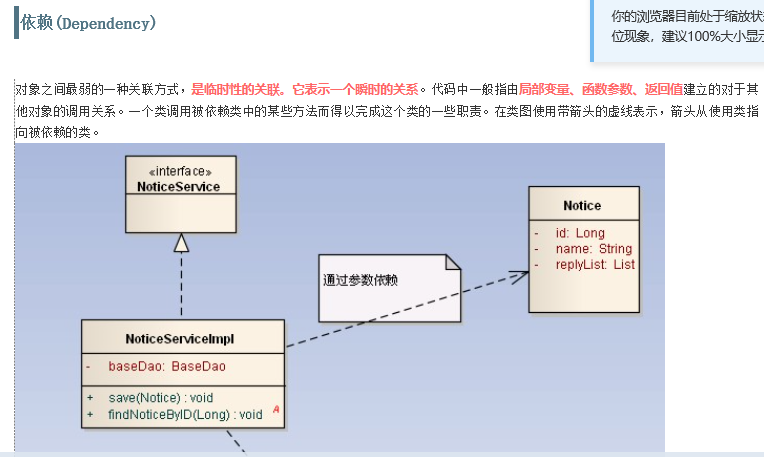


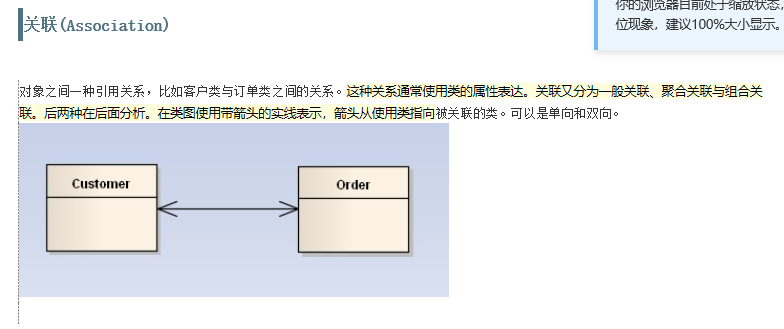
1. 类图

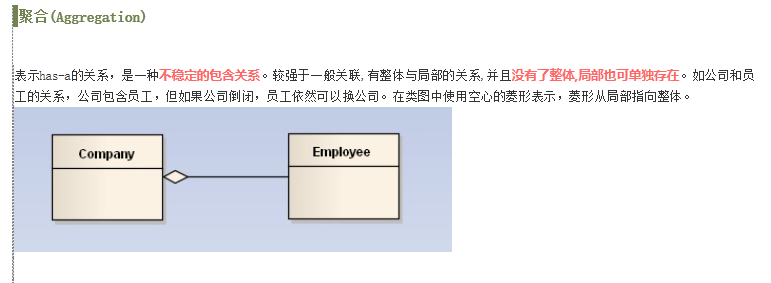


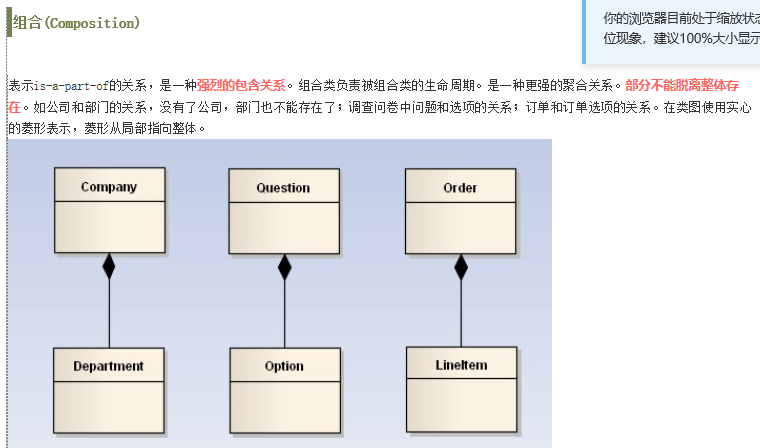


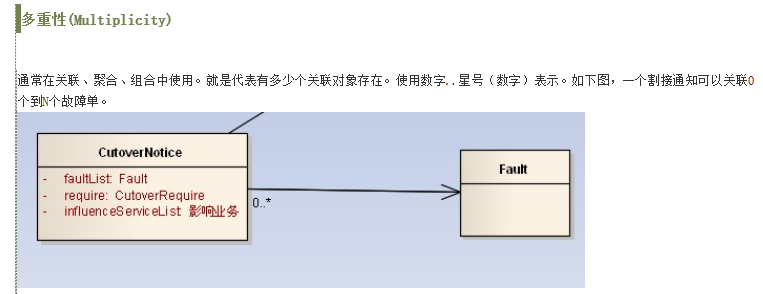




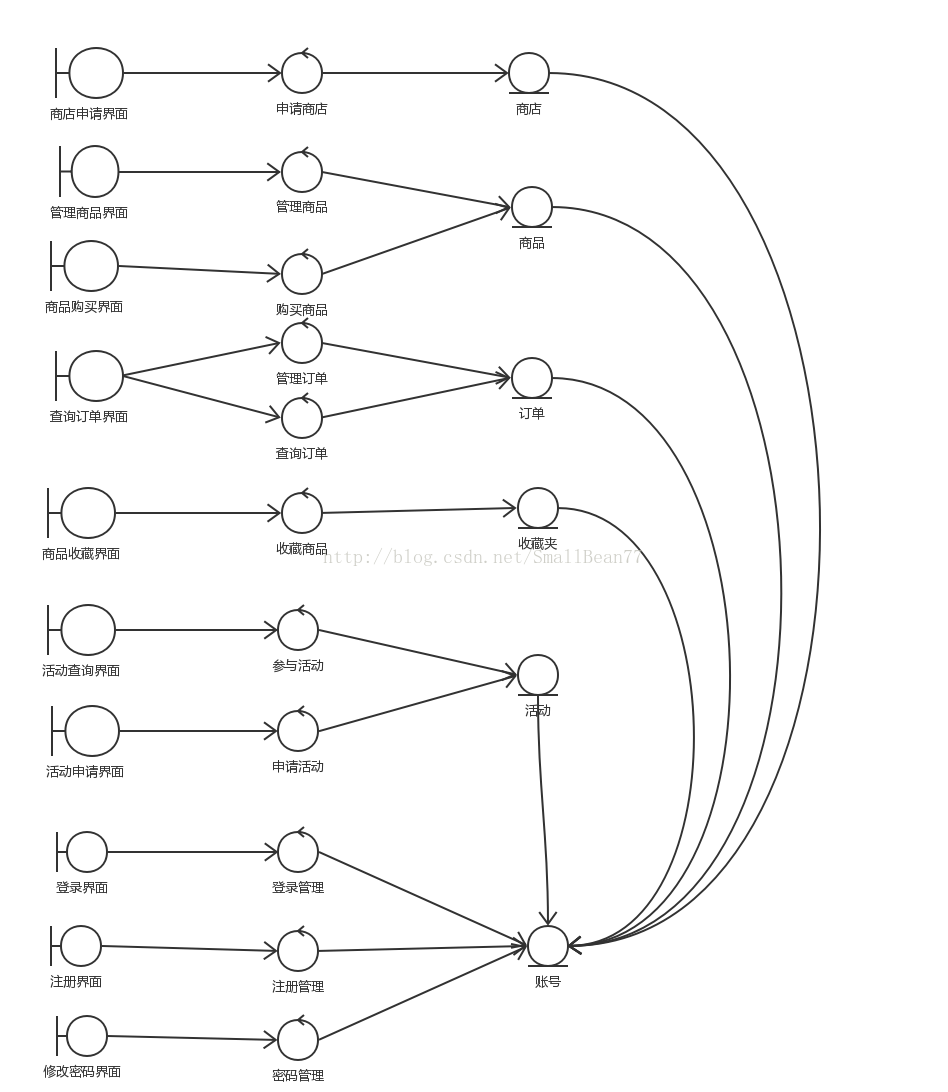




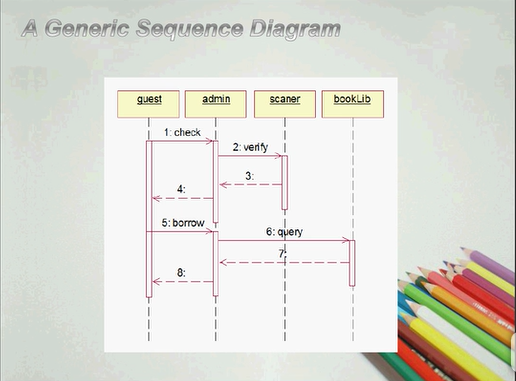


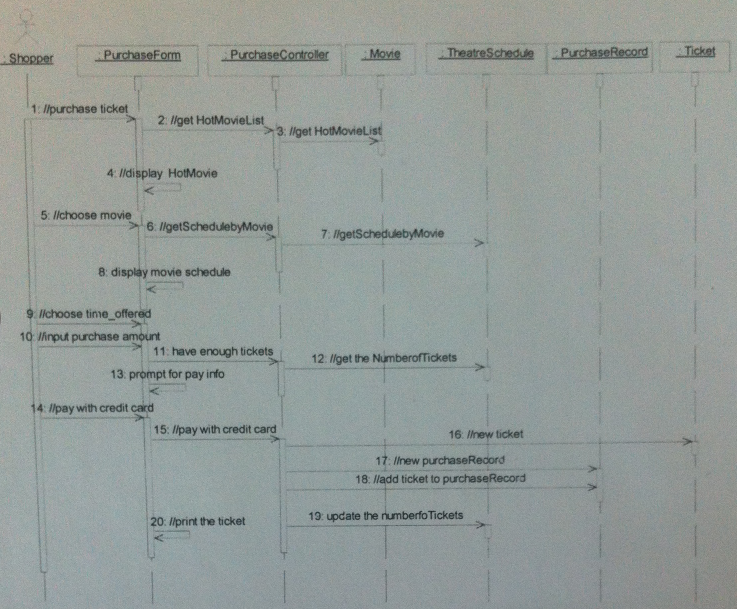


分析类图

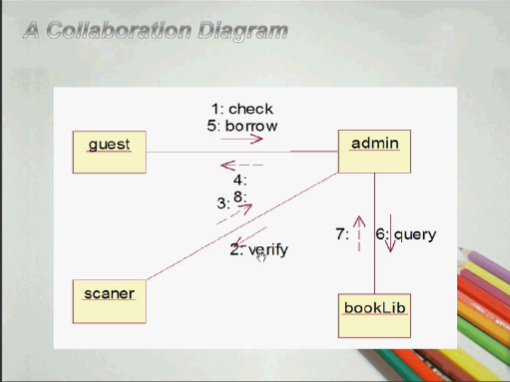


1. 时序图

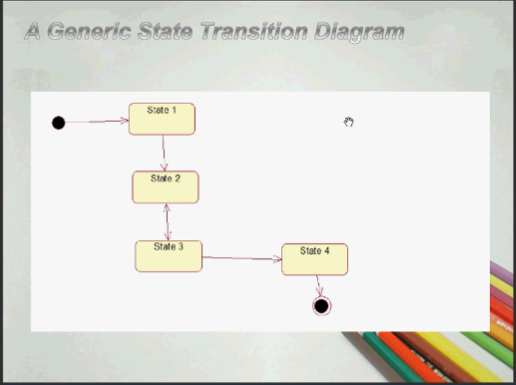




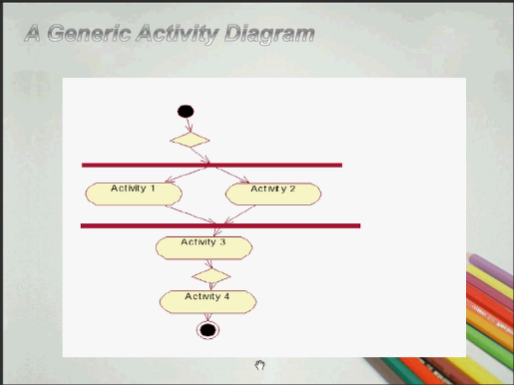
1. 协作图



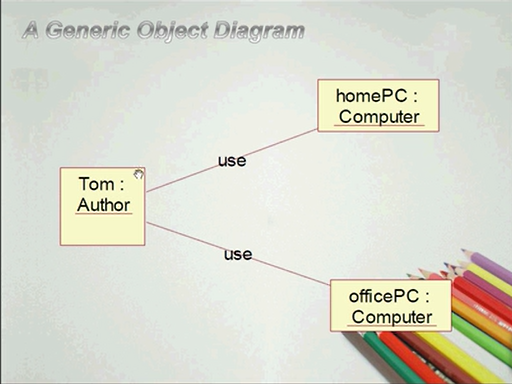
1. 状态图



1. 活动图



1. 对象图



# 问答

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

Object

【A set of principles (abstraction, encapsulation(封装), polymorphism(多态性)) guiding software construction, together with languages, databases, and other tools that support those principles.】

面向对象技术是一系列支持软件开发的原则（抽象，封装，多态性），以及支持这些原则的程序设计语言，数据库和其它工具。

【Reflects a single paradigm.

Facilitates architectural and code reuse.

Reflects real world models more closely.

Encourages stability.

Is adaptive to change】

反映一个特定实例。

有利于构件和代码重用。

更加真实地反映现实世界模型。

具有更好的稳定性。

能适应需求的变化。

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

【UML is Unified Modeling Language, it is a language for Visualizing, Specifying, Constructing, Documenting the artifacts of a software-intensive system. 】

UML是统一建模语言，是一门用于对面向对象开发的产品进行可视化建模，说明，架构和文档编制的标准语言。

【The UML builds models that are precise, unambiguous, and complete.

UML models can be directly connected to a variety of programming languages.

The UML addresses documentation of system architecture, requirements, tests, project planning, and release requirements. 】

UML帮助建立精确、完整、不含糊的模型。

UML模型可以和多种程序设计语言建立直接连接。

UML指导文档编制，包括系统架构文档，需求文档，测试文档，项目计划，版本说明等。

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

【Use-case driven + Architecture-centric + Iterative and incremental.】

用例驱动

以架构为中心

迭代和增量开发

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

【Use cases defined for a system are the basis for the entire development process.

为系统定义的用例作为整个开发过程的基础。

用例是描述系统所执行的一系列顺序事件流，并且某个角色可以看到执行的结果。

【Benefits of use cases:

Concise, simple, and understandable by a wide range of stakeholders.

Help synchronize the content of different models. 】

简洁明了，可被各个项目相关人理解。

帮助实现不同模型间的同步。

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

【A system’s architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development. 】

系统的架构是开发过程中最重要的一项产出，它定义了系统的概念、结构，是管理开发过程、展开系统的重要指导。

以架构为中心的开发过程：

统一开发过程强调架构是项目小组给系统定形的中心，因为一个模型难以反映系统的各个方面，统一开发过程支持多个模型和视图。

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

迭代：迭代是在既定计划和评价标准之下执行的一系列软件开发活动，每次迭代是一次集成的软件开发过程包括测试，并产生一个可执行的软件版本。

好处：

迭代可以在大投资前解决可预见的风险。

早期迭代可以获得用户反馈。

连续地测试和集成的开发过程。

客观的里程碑集中在短期。

通过对执行过程的评估来衡量开发进度。

部分可执行部件可被配置。

【Critical risks are resolved before making large investments.

Initial iterations enable early user feedback.

Testing and integration are continuous.

Objective milestones focus on the short term.

Progress is measured by assessing implementations.

Partial implementations can be deployed. 】

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

【Abstraction+Encapsulation+Modularity+Hierarchy】

抽象，封装，模块化，层次

【The essential characteristics of an entity that distinguishes it from all other kinds of entities.

Defines a boundary relative to the perspective of the viewer..

Is not a concrete manifestation, denotes the ideal essence of something. 】

抽象：提取出一个实体区分其它类型实体的本质特征，定义外界所能观察到的边界，并不具体表示某个实体，而是表示出其基本特征。

【Hides implementation from clients. 】

封装：对用户隐藏执行过程。

【Breaks up something complex into manageable pieces.

Helps people understand complex systems. 】

模块化：将复杂系统分成几个可控制的模块，帮助人们理解复杂系统。

【……】

层次：是一种从高到低有确定次序的结构，同一层的元素具有相同的抽象程度。

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

【A model that describes a system’s functional requirements in terms of use cases.

A model of the system’s intended functions (use cases) and its environment (actors).

用例模型：根据用例描述系统的功能需求。

用户，角色和communicate-association。

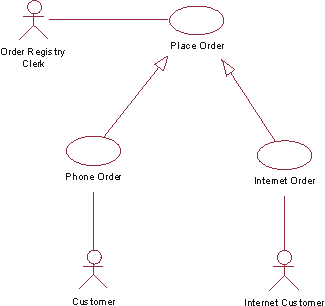
1. List three types of relationships existed between different usecases and give examples.

【用例泛化、包含用例、扩展用例】

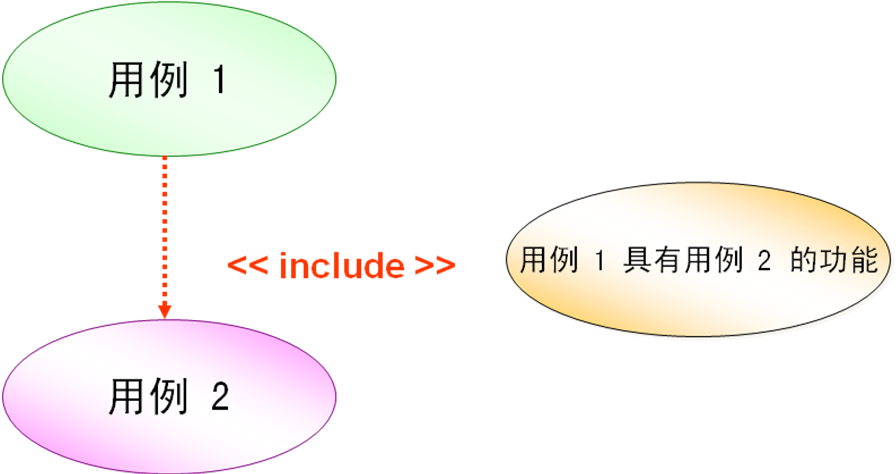
A. 泛化关系

用例的泛化

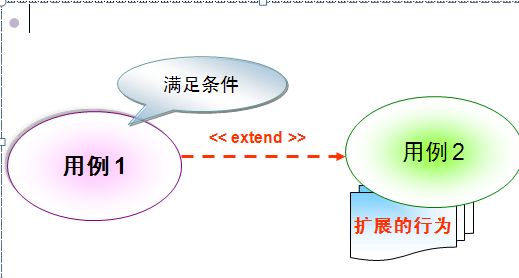
子用例可以依赖父用例的结构，可以在父用例的基础上增加额外的行为。

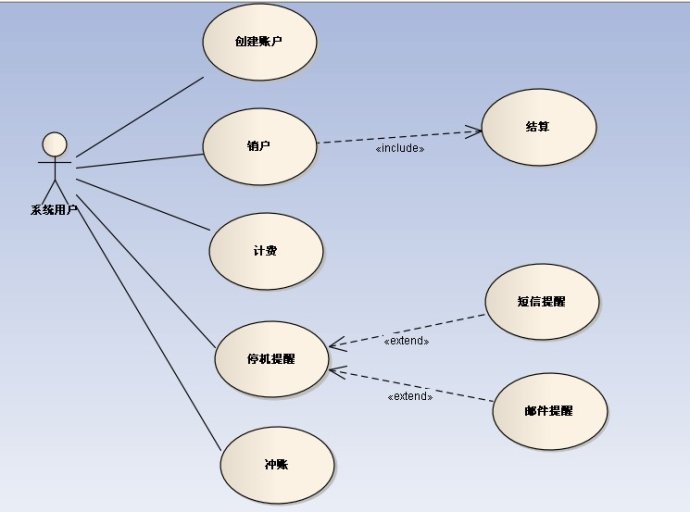


（2）包含关系(include)：位于两个用例之间的包含关系意味着基用例显式地在其指定位置将另一个用例包含进来, 使其成为自己的行为的一部分。在具有包含关系的两个用例中，被包含的那个用例不能单独存在，它只能以实例的形式存在于包含它的用例之中。 用例2包含用例1



扩展关系：两个用例之间的扩展关系，代表基用例可以隐式地包含另一个用例作为其行为的一部分，包含的位置间接地由另一个用例（扩展用例）确定。用例2是用例1的扩展行为



我们再用另外一个场景的用例说明一下include和extend,因为就这两个玩意比较容易搞混。  
销户：因为销户必需先进行账户结算，所以这里用include  
停机提醒：有两个可选项，短信提醒和邮件提醒，所以用extend.  


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

1) usecase diagram

【A use case models a dialog between actors and the system.

A use case is initiated by an actor to invoke a certain functionality in the system. 】

[用例图](http://doc.qiuhao.com/phrase/200603071221195.html)（[use case diagram](http://doc.qiuhao.com/phrase/200604231323515.html)）就是由角色、[用例](http://doc.qiuhao.com/phrase/200604240937105.html)以及它们之间的关系构成的图。

user，charicter和communicate-association

2)Activity diagram

【An activity diagram in the use-case model can be used to capture the activities in a use case.

It is essentially a flow chart, showing flow of control from activity to activity. 】

活动图是一种行为图（behavior diagram），通常用来表达业务流程、工作流或系统流程中一连串的动作。

3) sequence diagram

【A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. 】

顺序图是强调消息传递的时间顺序的一种交互图。

4) Collaboration diagram

【A collaboration diagram emphasizes the organization of the objects that participate in an interaction. 】

协作图强调对象在参与活动中的组织。

5) class diagram

【Static view of a system. Include The vocabulary of a system, Collaborations, A logical database schema.】

[类图](http://doc.qiuhao.com/phrase/200603071659325.html)([Class diagram](http://doc.qiuhao.com/phrase/200604241343565.html))是显示了模型的静态结构，特别是模型中存在的[类](http://doc.qiuhao.com/phrase/200603090857555.html)、类的内部结构以及它们与其他类的关系等。

6) statechart diagram

【A statechart diagram shows a state machine. 】

[状态图](http://doc.qiuhao.com/phrase/200604161451045.html)([Statechart Diagram](http://doc.qiuhao.com/phrase/200604241340495.html))是描述一个实体基于事件反应的动态行为，显示了该实体如何根据当前所处的状态对不同的时间做出反应的。

7) deployment diagram

【The deployment diagram shows:

Configuration(配置) of processing nodes at run-time.

Communication links between these nodes.

Component instances and objects that reside on them. 】

[部署图](http://doc.qiuhao.com/phrase/200604161812215.html)([deployment diagram](http://doc.qiuhao.com/phrase/200604241346265.html)，[配置图](http://doc.qiuhao.com/phrase/200604241346445.html))是用来显示系统中[软件](http://doc.qiuhao.com/phrase/200604232134205.html)和硬件的物理[架构](http://doc.qiuhao.com/phrase/200604241328115.html)。

从部署图中，可以了解到软件和硬件[组件](http://doc.qiuhao.com/phrase/200603302222545.html)之间的物理关系以及处理节点的组件分布情况。

使用部署图可以显示运行时系统的结构，同时还传达构成应用[程序](http://doc.qiuhao.com/phrase/200604232224305.html)的硬件和软件元素的配置和部署方式。

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

【Semantically equivalent---Can convert one diagram to the other without losing any information.

Model the dynamic aspects of a system.

Model a use-case scenario.】

相同点：可以将一种图转换成另一种图而不丢失任何信息。对系统的动态行为进行建模，对用例的情节进行建模。

【Collaboration diagrams: Sequence diagrams:

-Show relationships in addition -Show the explicit sequence of messages.

to interactions.

-Better for visualizing patterns -Show focus of control.

of collaboration.

-Better for visualizing all of the -Better for visualizing overall flow.

effects on a given object. -Better for real-time specifications

-Easier to use for brainstorming sessions. and for complex scenarios. 】

不同点：

交互图 顺序图

根据交互行为显示对象间的关系。 显示外部消息的顺序。

更好的观察协作模型。 显示控制焦点。

更好的观察一个对象所受到的各种影响。 更好的观察全部的事件流。

更适于运用于头脑风暴会议。 更适于实时描述和描述复杂情景。

1. Define the different relationships in class diagram: dependency(依赖), association(关联),aggregation(聚合), composition(组合即强聚合), generalization(泛化).

【Dependency:一个类的改变可能影响或提供信息给其他类。

* 两个类之间的依赖关系，表明其中的一个类（客户类）依赖于另一个类（供应类）所提供的某些服务*。*

Association: The semantic relationship between two or more classifiers that specifies connections among their instances.类之间的连接

* 关联：给出两个或更多的类之间的语义联系，说明了它们实体之间的关系。

Aggregation: A special form of association that models a whole-part relationship between the aggregate (the whole) and its parts.

聚合表示组成和整体的所有关系。Is an “is a part of “ relationship.

composition: 组合即强聚合，表示整体对组成的包容关系；

Generalization: A relationship among classes where one class shares the structure and/or behavior of one or more classes. Is an “is a kind of”relationship. 】

表示一个类共享其它类的结构或者行为的一种类与类之间的关系。Is an “is a kind of”relationship.

1. What is a node in deployment diagram(部署图) ? List two diffent types of nodes.

【A physical element that exists at run-time and represents a computational resource.

Processor Node+Device Node. 】

结点是存在于运行时系统中的物理元素，代表了一种可计算资源。

处理机节点：运行软件

设备节点：由处理机控制的设备

1. Describe the extensibility mechanisms of UML .

【扩展机制extensibility mechanisms.

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.
2. Explain the six best practices of software engineering.

【迭代的开发软件Develop Iteratively

1. 边做边改模型（Build-and-Fix Model）
2. 瀑布模型（Waterfall Model）
3. 快速原型模型（Rapid Prototype Model）
4. 增量模型（Incremental Model）
5. 螺旋模型（Spiral Model）
6. 演化模型(evolution model)
7. 喷泉模型(fountain model)
8. 智能模型(四代技术（4GL）)
9. 混合模型（hybrid model）

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 is in RUP ? Describe each phase’s purpose and milestone.

【Rational Unified Process.

RUP（Rational Unified Process，统一软件开发过程，统一软件过程)是一个面向对象且基于网络的程序开发方法论。

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】

用例视图，逻辑视图，实现视图，过程视图+部署视图

1. What is the difference between analysis and design?

【Analysis: Design:

-Focus on undrestanding the problem. -Focus on understanding the solution.

-Idealized design. -Operations and attributes.

-Behavior. -Performance.

-System structure. -Close to real code.

-Functional requirements. -Object lifecylces.

-A small model. -Nonfunctional requirements.A large model. 】

分析： 设计：

集中在理解问题。 集中在理解解决方案。

是理想化设计。 设计相关操作和属性。

行为。 性能。

系统架构。 接近真实代码。

功能需求。 对象生命周期。

是一个小模型。 非功能需求，是一个大模型。

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

【Key Concepts

Define the High-Level Organization of Subsystems

Identify Key Abstractions

Create Use-Case Realizations

Checkpoints】

关键概念：

定义高层组织和子系统。

识别关键的抽象。

创建用例实现。

设置检查点。

【Identify classes and subsystems

Identify subsystem interfaces

Update the organization of the Design Model

Checkpoints】

识别各个类和子系统。

识别子系统的接口。

校正设计模型的组织结构。

设置检查点。

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

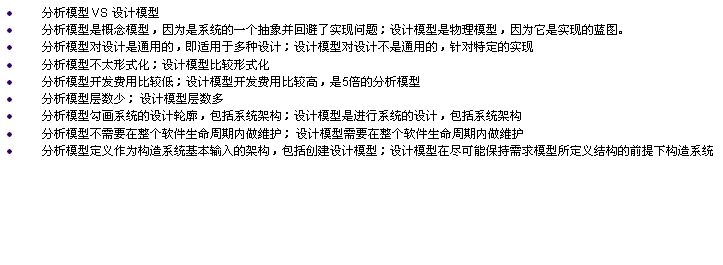
【Application subsystems->Bussiness Specific->Middleware->System Software】

层次体系结构就是利用分层的处理方式来处理复杂的的功能，层次系统要求上层子系统使用下层子系统的功能，而下层子系统不能够使用上层子系统的功能。

C/S（两层）体系结构）   
客户机/服务器结构简称C/S结构或两层体系结构。

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

分析与设计规程的任务是研究欲采用的实现环境和系统构建的效用，结果是产生一个设计模型。设计模型包含了用例的实现，可以表现对象是如何相互通信和运作实现用例的。在设计模型中可能包含对象类和子系统的接口定义，规定它们提供操作服务的责任。这个对象模型也可以在实施环境中采用，那就是用程序设计语言，分布等来表达。对于大型系统的项目，有时为系统分析的结果建立单独的分析模型也是很有用的。



分析模型是概念模型，因为是系统的一个抽象并回避了实现问题；设计模型是物理模型，因为它是实现的蓝图。

分析模型对设计是通用的，即适用于多种设计；设计模型对设计不是通用的，针对特定的实现

分析模型不太形式化；设计模型比较形式化

分析模型开发费用比较低；设计模型开发费用比较高，是5倍的分析模型

分析模型层数少； 设计模型层数多

分析模型勾画系统的设计轮廓，包括系统架构；设计模型是进行系统的设计，包括系统架构

分析模型不需要在整个软件生命周期内做维护； 设计模型需要在整个软件生命周期内做维护

分析模型定义作为构造系统基本输入的架构，包括创建设计模型；设计模型在尽可能保持需求模型所定义结构的前提下构造系统

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

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

Entity Class:Key abstractions of the system..

Control Class: Use-case behavior coordinator. 】

分析类是这样的类：

它代表问题域中的简洁抽象； 应该映射到真实世界的业务概念（并且据此仔细命名）。

boundary 类，中介本系统与其环境之间的协作。

control 类，封装特定用例的行为。

entity 类，用于建模事物的永久信息。

举例：

boundary?类：

1. 用户界面类─人与系统之间的接口类；

2. 系统接口类─同其他系统之间的接口类；

3. 设备接口类─同外部设备，例如传感器之间的接口类。

控制类：

设计课程注册系统，开始你可能引入了控制类 CourseRegistrationController 来协调整个过程。

实体类：表由系统所管理的主要事物（例如，客户(customer)）；

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

【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】

【use-case realization的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 occured in the use-case analysis.

【Supplement the Use-Case Description

For each Use-Case Realization: Find Classes from Use-Case Behavior, Distribute Use-Case Behavior to Classes

For each resulting analysis class: Describe Responsibilities, Describe Attributes and Associations,Qualify Analysis Mechanisms

Unify Analysis Classes

Checkpoints】

补充用例说明。

找出用例中的行为，把行为合理分配给各个类。

对每一个分析出来的类，描述其职责，属性，和类间的关联，限定分析机制。

统一分析类。

查检分析过程和结果。

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

【A general purpose mechanism for organizing elements into groups.

A model element that can contain other model elements. 】

【A package can be used:

To organize the model under development.

As a unit of configuration management. 】

包是用来对元素分组的一种机制，是一种包含其它模型元素的模型元素。

包能用来：在开发过程中组织模型。

是配置管理的单元。

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

【Is a “cross between”a package and a class

Realizes one or more interfaces that define

[子系统](http://doc.qiuhao.com/phrase/200604161433025.html)是一种模型元素，它具有包（其中可包含其他模型元素）和[类](http://doc.qiuhao.com/phrase/200603090857555.html)（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。

包：一旦模型变得很大，不能再维持平面结构，就需要对包分割。

子系统：如果某个协作中的各个类只是在相互之间进行交互，并且可生成一组定义明确的结果，就应将该协作和它的类封装在一个子系统中。这一规则同样适用于协作的子集。可以对协作的任何部分或全部进行封装和简化，作为一个子系统。

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

【Analyze concurrency requirements.

Identify processes and threads

Identify process lifecycles

Map processes onto the implementation

Distribute model elements among processes】

分析并发需求。

识别进程和线程。

识别进程生命周期。

从进程到实现的映射。

在进程间分配模型的元素。

【Processes can be modeled using Class Diagrams, Interaction Diagrams and Components. 】

进程可以通过类图，交互图和构件建模。

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

【Reduce processor load

Special processing requirements

Scaling concerns Economic concerns

Distributed access to the system】

减轻处理器负载。

满足特殊的处理要求。

减少经济开支。

分布式地访问系统。

部署图建模指南：

在特定的项目图上注明软件组件；集中在企业级图上的结点和通信关联

结点和组件：用描述性术语命名结点；仅仅建模重要的软件组件；为组件一致地应用一致版型；把可视化的版型应用到结点

依赖和通信关联：用版型来注明通信协议；仅仅建模组件间的关键性依赖

1. Discribe the two typical distribution patterns, C/S and P2P

【Client/Server

3-tier

Fat Client

Fat Server

Distributed Client/Server】

[C/S结构](http://doc.qiuhao.com/phrase/200604291145475.html)（[Client/Server结构](http://doc.qiuhao.com/phrase/200604291147015.html)）是[软件系统](http://doc.qiuhao.com/phrase/200602281706245.html)[体系结构](http://doc.qiuhao.com/phrase/200603122156385.html)，通过它可以充分利用两端硬件环境的优势，将任务合理分配到[Client](http://doc.qiuhao.com/phrase/200604231337375.html)端和Server端来实现，降低了系统的通讯开销。服务器通常采用高性能的PC、工作站或小型机，并采用大型[数据库系统](http://doc.qiuhao.com/phrase/200603011056245.html)，如[ORACLE](http://doc.qiuhao.com/phrase/200604040935115.html)。[客户端](http://doc.qiuhao.com/phrase/200603082208195.html)需要安装专用的客户端软件。

P2P直接将人们联系起来，让人们通过互联网直接交互。P2P使得网络上的沟通变得容易、更直接共享和交互，真正地消除中间商。P2P就是人可以直接连接到其他用户的计算机、交换文件，而不是像过去那样连接到服务器去浏览与下载。P2P另一个重要特点是改变互联网现在的以大网站为中心的状态、重返"非中心化"，并把权力交还给用户。

如果将操作的定义比作“黑盒”，那么方法（ Method ）就是相应的“白盒”内容。方法是对操作的实现，由具体的程序设计语言完成。方法说明了实现操作的具体方式。其内容通常会涉及操作的参数，类的属性及关系在方法中的使用。如果方法的内容需要采用特定的算法，应给出相应的文字或图示说明。

Four Principles of Modeling

The model you create influences how the

problem is attacked.

Every model may be expressed at different

levels of precision.

The best models are connected to reality.

No single model is sufficient

stereotype是UML提供的一种扩展方式 有实体类，边界类，控制类

定义uml的新元素

One of the main weakness of object technology: high-level manipulation of data.

Object technology is low-level when manipulating whole sets of data

OO缺点

(1)需要一定的软件支持环境。

(2)不太适宜大型的MIS开发，若缺乏整体系统设计划分，易造成系统结构不合理、各部分关系失调等问题。

(3)只能在现有业务基础上进行分类整理，不能从科学管理角度进行理顺和优化。

(4)初学者不易接受、难学。

# 复习题

1. On the scut campus, students can apply for a consumed-card .The card administrator would create a new consumed-card when receiving an application. The consumed-card can be used in the campus super-market. When the students consume with the card , the salesperson would deduct the money from the card. Also , The card administrator can deposit money into the card for the students . Imagine that you have been hired to develop the supporting software.

1) Draw a UML use case diagram for the system.

2) Produce a sequence diagram to model the scenario of consuming with card.

3) Draw a class diagram for the system.

4) Produce a statechart diagram for the consumed-card Class.

1）



2）



3）



4）



2. Consider developing a campus classroom facilities management system

The campus has a collection of classrooms. Each classroom has a collection of equipments, which may include microphone, blackboard, overhead projector, TV, Internet connection, etc. Each equipment has information such as history, condition, as well as information specific to the equipment, such as size for blackboard and TV, mobile or not for microphone, and speed of connection to Internet. Each classroom has a list of courses scheduled to take place in it, the instructor for each course, requested equipments and setups for each course, and a manager in charge of setting up the equipments. The system should allow instructors to look up the list of equipments in a classroom, request equipments and setups, and query the setup status. The system should send request for equipments and setups in a classroom to the classroom manager. The manager can update equipment information and setup status. In case of an equipment can not be setup according to the request, the manager should notify the instructor as soon as possible. The system should also allow anyone to send comments to the manager.

Draw the use case diagrams for this system.

Draw the class diagrams for this system.

1）



2）

