

SC2006 SOFTWARE ENGINEERING

1 OVERVIEW

Requirements elicitation -> Design it -> Build it -> Test it -> Release it and maintain it.

The problem of volatility: the problem always changes.

2 REQUIREMENTS ELICITATION (REQUIREMENT DOCUMENT PART)

2.1 PROJECT MISSION STATEMENT

这个 project 要做什么，需要包含：

1. 需要解决什么问题
2. The stakeholders (利益相关者) , developers and users
3. The outcomes and benefits of the project

E.g., The GoFast team will develop a website that enables airline travellers to rate their travel experiences. This project will be considered complete when the website has been tested and approved for release by the FactFinding Organisation. This project supports the International Travel Watchdogs' objective to ensure air passengers can openly compare airlines.

2.2 TYPE OF REQUIREMENTS

2.2.1 Functional Requirements

需要实现一定的功能

Examples:

1. System functionality to be performed
 - a. e.g., The library member must be able to search the library catalogue.
 - b. e.g., The bank customer must be able to withdraw cash from the ATM.
2. Information to be processed
 - a. e.g. The system must display the current time in 24-hour format.
 - b. e.g. The system must display the temperature in degrees centigrade in the range -10C to +130C to one decimal place of accuracy.
3. Interface with other systems
 - a. e.g. The system must be able to use wifi to communicate all transactions with a client's secure database.
 - b. e.g. The system must be able to control up to six robot arms simultaneously.

2.2.2 Non-Functional Requirements

可操作性、可拓展性等等

Examples:

1. Usability: Help messages must be displayed in the local language according to the user's locale.
2. Reliability: The full system functionality must be restored within 5 minutes after a system reboot.
3. Performance: When a book is placed in the checkout pad, the system must detect it within 2 seconds.
4. Supportability: The database must be replaceable with any commercial product supporting standard SQL queries.

2.3 DOCUMENTING THE REQUIREMENTS

要求：

1. Atomic: 每句话只能实现一个功能，可以有很多重复
2. Verifiable:
 - a. 可以承受大于 1 次：那测试的时候到底是多少次呢？=> 可以承受 1-7 次
 - b. User-friendly => 80%的新用户两分钟内能知道怎么用这东西
3. Traceable: 代码和 Requirement 要同步修改，这样能通过 Requirement 的改变来追溯代码 (If you want to change, you should have a reason)
4. Unambiguous

2.4 REQUIREMENTS VALIDATION

目标：

1. 对 Stakeholders：能满足所有的需求和条件
2. 对 Development Team：这些 requirements 有没有被误解

方法：

1. Review（需要对 Stakeholders 和 Development Team 都进行）：
 - a. Walkthrough, inspection, critical review
 - b. Checklist for completeness, consistency, unambiguity, correctness
2. Prototype

2.5 DATA DICTIONARY

Glossary（词汇表）

这个 dictionary 主要是在 problem domain 的，而非 implementation terms

Term	Definition
Elevator Manager	Human user of system to manage grain storage
Grain	Delivered by truck, stored in silo, distributed by train 6 types: wheat, barley, long grain rice, short grain rice, oats, hops 2 grades: high, low
Railroad Car	Carries 1 type of grain of certain grade from silo to processing plant
Report	Requested by Elevator Manager to reflect status of grain elevator, can be presented per silo or per grain type
Sensor	Monitors humidity and temperature in silo Sends data and alert to the System
Shipment	Arrival Shipment or Departure Shipment Records the type / grade of grain & quantity, together with transport information
Silo	Holds 1 type of grain of certain grade 2 sizes: 8000 bushels, 12000 bushels
Transaction Log	List of Shipments, can be sorted in chronological order
Truck	Carries 1 type of grain of certain grade from farm to silo

3 REQUIREMENTS ELICITATION (USE CASE PART)

Use case: 一些特定的功能，显示出用户和系统是怎么交互的。

Use case diagram 是静态的，use case description 是动态的。

3.1 REVIEW OF REQUIREMENT ELICITATION PROCESS

执行 Step 1 和 2 直到大家达成共识

3.1.1 Step 1: Project Team 自己搞

1. 分解需求，将其原子化。
2. 确定参与者。
3. 确定主要的用例（功能需求）。
4. 确定非功能性需求。
5. 启动数据字典的生成。
6. 确定需要与客户澄清的不确定性（待定）。不要猜测用户需求。

3.1.2 Step 2: Project Team meet Client（客户）

1. 展示和解释初步需求和数据字典。获得一致同意。
2. 澄清不确定性（待定问题）。
3. 讨论客户提出的任何新需求。

4 REQUIREMENTS ANALYSIS

- Conceptual model (structural aspect): 做什么（这些 objects 在系统中是干什么的）
- Dynamic model (dynamic aspect): 怎么做（这些 objects 到底是怎么做实现这些动作的）

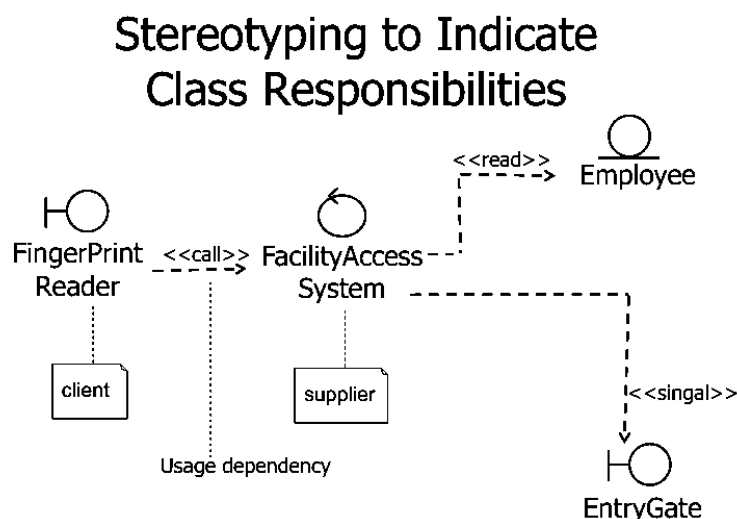
4.1 CONCEPTUAL MODEL (CLASS DIAGRAM)

Objects / Operations (Methods) / Attributes

Class Diagrams 有三种作用， requirements analysis/system design/object design.

SC2002 用的是 object design，现在用的是 requirements analysis。

这里 Class Diagram 的 Example:



4.2 DYNAMIC MODEL

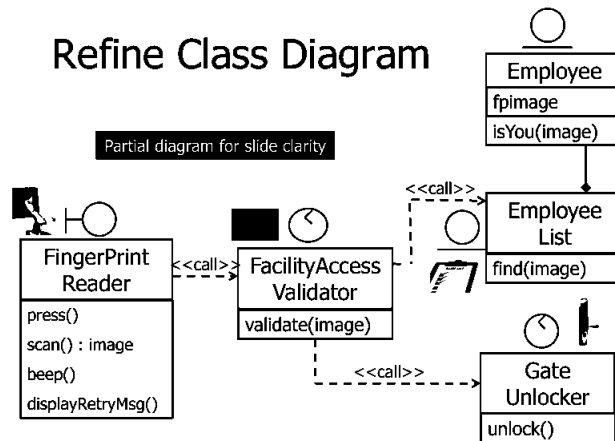
4.2.1 Diagrams Learned

1. Single Use Case
 - a. Sequence Diagram: 描述 class 之间的关系，用来设计
 - b. Communication Diagram: 通过以不同的方式呈现消息序列，可以实现类似的目的，用来展示

2. Multiple Use Cases

- State Machine Diagram / Dialog Map: 描述的是这个 system 或者 object 的状态
- Activity Diagram: 描述 workflow

4.2.2 Refine Class Diagram



5 SOFTWARE PROCESSES

5.1 BASIC KNOWLEDGE

一种结构化的方法

几乎不同的 process 都有这些:

- Specification (确定要做什么)
- Design and implementation
- Validation
- Evolution (修改)

Software Process Descriptions

- Products (要做出什么)
- Roles (每个人要干什么)
- Pre- and post-conditions (每一项活动之后我们需要做出什么东西)

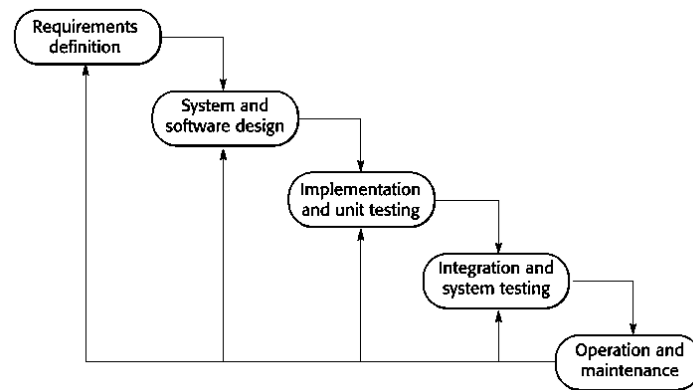
Plan-Driven vs Agile: Plan-Driven 是先全都计划好, Agile 是先把屎山代码堆出来再说

Incremental 和 Integration 也有可能是 Plan-Driven 的。

5.2 SOFTWARE PROCESS MODELS

1. Waterfall Model: 一步一步来, 分析, 计划, 写代码
2. Incremental Development: 一边写代码一边计划, 随缘发展
3. Integration and Configuration (COTS: Commercial Off the Shelf): 把别人堆好的小屎山堆成自己的大屎山

5.2.1 Waterfall



缺点：（堆得好，但要修改很难）The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be completed before moving on to the next phase. Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

适合上古大型项目

5.2.2 Incremental Development

优点：

- ✧ The cost of accommodating changing customer requirements is reduced.
 - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- ✧ It is easier to get customer feedback on the development work that has been done.
 - Customers can comment on demonstrations of the software and see how much has been implemented.
- ✧ More rapid delivery and deployment of useful software to the customer is possible.
 - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

缺点：

- ✧ The process is not visible.
 - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- ✧ System structure tends to degrade as new increments are added.
 - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

5.2.3 Integration and Configuration

优点	缺点
<ul style="list-style-type: none">Reduced costs and risks as less software is developed from scratch.Faster delivery and deployment of the system.	<ul style="list-style-type: none">requirements compromises are inevitable so the system may not meet the real needs of users. (别人的屎山可能对不上)Loss of control over evolution of reused system elements. (别人的东西版本影响到了你的版本)

5.3 SOFTWARE PROCESSES ACTIVITIES

Specification, development, validation, evolution

6 AGILE SOFTWARE DEVELOPMENT

6.1 PRINCIPLES

Customer Involvement	Customers should be closely involved throughout development to provide and prioritise new system requirements and evaluate iterations of the system.
Incremental Delivery	The software is developed in increments; the customer specifies the requirements to be included in each increment.
People not process	The skills of the development team should be recognised, team members are left to develop their own ways of working without prescriptive processes.
Embrace Change	System requirements are expected to change, the system should be designed to accommodate these changes.
Maintain Simplicity	Focus on simplicity in both the software and development process, actively eliminate complexity from the system.

6.2 MANIFESTO (宣言)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

6.3 适用条件 (APPLICABILITY)

项目小	Product development is where a software company is developing a small or medium-sized product for sale.
用户参与开发	Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external rules and regulations that affect the software.

6.4 EXTREME PROGRAMMING (XP)

什么时候用:

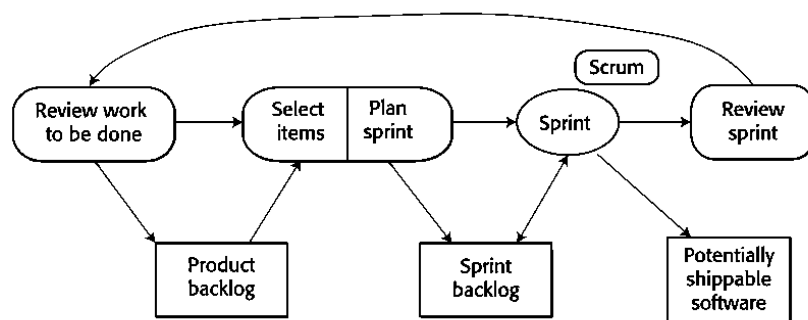
- Involve new or prototype technology, where the requirements change rapidly, or some development is required to discover unforeseen implementation problems.
- Are research projects, where the resulting work is not the software product itself, but domain knowledge.
- Are small and more easily managed through informal methods.

User stories for specification	1. 需求被记录在“Story Card”上, 供开发人员选择
--------------------------------	---------------------------------

	2. 被描述为 stories or scenarios
Refactoring	<ol style="list-style-type: none"> 1. Re-organization of a class hierarchy to remove duplicate code. 2. Tidying up and renaming attributes and methods to make them easier to understand. 3. The replacement of inline code with calls to methods that have been included in a program library.
Test-first development	Usually relies on a testing framework such as JUnit.
Pair programming	All production software in XP is built by two programmers, sitting side by side, at the same machine. Fast knowledge
Customer Involvement	用户加入开发

6.5 SCRUM

Development team	7 人以下小队
Potentially shippable product increment	一个有可能可交付的软件增量
Product backlog	待办事项
Product owner	一个人（或可能是一个小团队），其工作是识别产品特性或需求，为开发这些特性进行优先排序，并不断审查产品待办清单，以确保项目继续满足关键业务需求。产品负责人可以是客户，但也可能是软件公司的产品经理或其他利益相关者代表。
Scrum	Scrum 团队的每日会议，审查进展情况并优先安排当天要完成的工作。理想情况下，这应该是一个短暂的面对面会议，包括整个团队。
Scrum Master	Scrum Master 负责确保 Scrum 流程的遵循并指导团队有效地使用 Scrum。他/她负责与公司的其他部门进行接口，并确保 Scrum 团队不会被外部干扰所分散。
Sprint	一个开发迭代。Sprint 通常持续 2-4 周。
Velocity	一个团队在单个 Sprint 中可以完成多少产品待办工作量的估计。



7 ARCHITECTURE DESIGN

Software Architecture = {Components, Connectors}

Components: 可以对应为具体的代码

Connectors: functions

Architecture Diagram 和 Class Diagram 的区别：Architecture Diagram 讲的是 class 与 class 之间的 communication，而 Class Diagram 讲的是 class 与 class 之间的联系（更注重 implementation）

Low coupling and High Cohesion: Module 之间关联弱，但一个 Module 里的元素之间联系强

7.1 ARCHITECTURE STYLES (只讲 LAYER ARCHITECTURE)

One Example: Presentation – App Logic – Persistent Data.

Require:

1. Everything in the one layer is similar. (Every layer has a label)
2. The communications are always from the upper layer to the lower layer.

Benefits (Android):

1. 把 app 分在同一层，方便 app 开发（容易增加新功能）
2. 增加了底层（kernel 层）的重用率（增加代码重用率 reuse）
3. 层层设计，结构化

Disadvantages:

1. Lower performance.
2. Difficult to find the right levels of abstraction.
3. Not all systems can be easily structured in a layer fashion.

8 OBJECT DESIGN

8.1 STEP

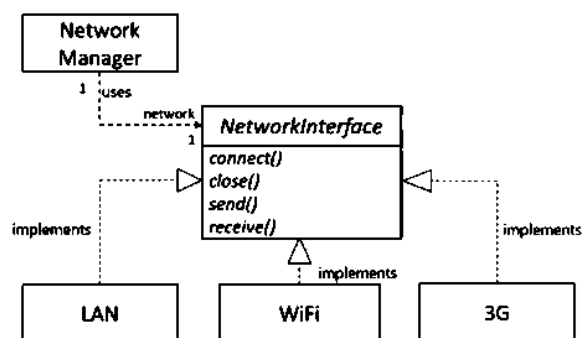
- Reuse: Existing Libraries and Designed Patterns (major decisions)
- Interface Specification
- Restructuring
- Optimization

8.2 DESIGN PATTERN

8.2.1 分类

1. Creational Patterns
2. Structural Patterns
3. Behavioural Patterns

8.2.2 Strategy Pattern



NetworkManager: Context
NetworkInterface: Strategy interface
LAN, WiFi, 3G, etc.: Concrete strategy objects

Pros:

1. Provides encapsulation
2. Hides implementation
3. Allows behaviour change at runtime

Cons:

1. Results in complex, hard to understand code if overused

8.2.3 Observer Pattern

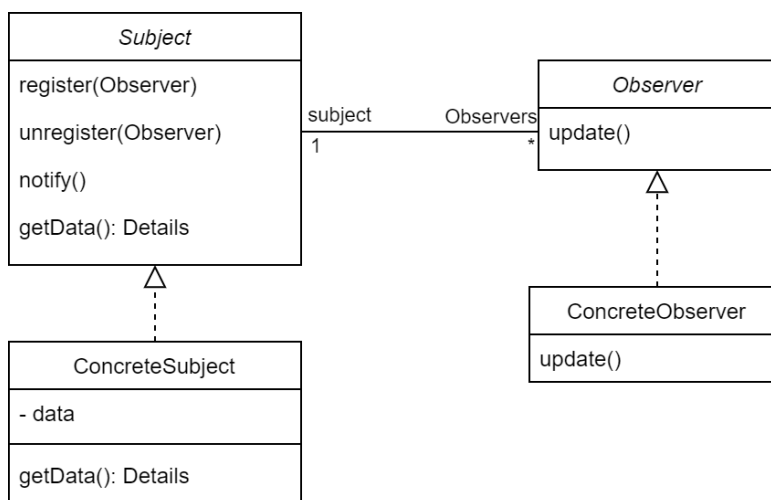
Subject – Observer

需要完成两个操作：

1. Subscription mechanism: observer 可以自由地注册/取消注册对主题的兴趣。
2. Notification mechanism: 当发生变化时，subject 将变化传播给 observer。

三种方式：

- Pull based: 用户可以选择性 pull，当信息多的时候，用户可以自己选择（好多个 get method 自己选）
- Push based: 全塞给你，答辩也得吃
- Merge: 先塞给你一部分，如果你吃着觉得挺香，再全 pull



Pros	Cons
<ul style="list-style-type: none">• Abstracts coupling between Subject and Observer• Supports broadcast communication• Enables reusability of subjects and observers independently of each other	<ul style="list-style-type: none">• Slower performance• If not used carefully the observer pattern can add unnecessary complexity

8.2.4 Factory Pattern

```
public static DataStoreInterface getDatastore(String classname) {
    DataStoreInterface datastore = null;

    try {
        Class datastoreTmp =
            ClassLoader.getSystemClassLoader().loadClass(classname);
        datastore = (DataStoreInterface) datastoreTmp.newInstance();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }

    return datastore;
}
```

Pros	Cons
<ol style="list-style-type: none">1. Encapsulate object creation2. Easy to extend, replace, or add new subclasses3. Easy to change object creation logic	<ol style="list-style-type: none">1. Complexity: Factory Pattern can add complexity to the codebase, especially with many object types.2. Coupling: The pattern can create tight coupling between factory and product classes, limiting flexibility.3. Overhead: The pattern can add overhead in terms of memory usage and performance, especially when creating many small objects.

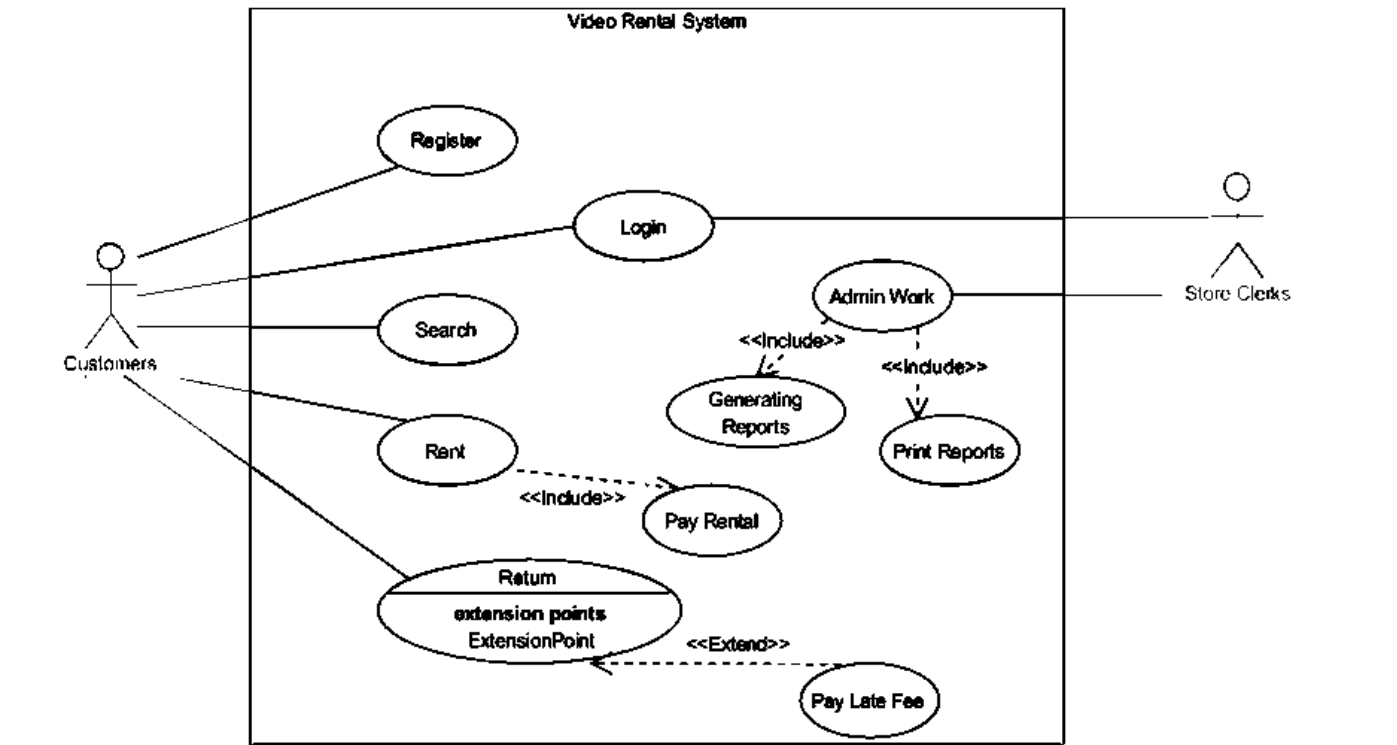
Use Case Diagram / Description

1 USE CASE DIAGRAM

1.1 ARROW

<< include >>	<< extend >>
虚线箭头，如果 A 使用了 B，那么应该是 A->B。	虚线箭头，如果 A 有一个分支去 B，那么是 B->A。
吃饭指向张嘴	洗手指向吃饭
Reuse functionality	Add (optional) functionality
一定会用到所有的 include	不一定用到所有的 extend（因为是分支）

1.2 EXAMPLES



2 USE CASE DESCRIPTION

2.1 TEMPLATE

Use Case ID			
Use Case Name			
Created By		Last Updated By	
Date Created		Date Last Updated	

Actor	谁参与了这个活动（Diagram 上连的人） 如果没人，那么填 NULL
Description	
Preconditions	执行这个 Use Case 之前，需要满足哪些条件
Postconditions	完成这个 Use Case 之后，系统会变成什么样（在执行 Use Case 之后可以 assert）
Priority	(Optional)

Frequency of Use	(Optional)
Flow of Events	Step 1: ... Step 2: ...
Alternative Flows	出现情况之后，这个 use case 自己能解决，那么就在这里。 AF-S5 表示 Step 5 的时候出现的分支（比如密码输错之类的）
Exceptions	出现情况之后，需要跳出这个 use case 的 直接按照 EX1, EX2 命名即可
Includes	
Extends	
Special Requirements	
Assumptions	
Notes and Issues	

2.2 常用语句

2.2.1 Priority

Optional，如果真的要写 Priority，那就 1~5，1 is the highest priority, and 5 is the lowest priority.

2.2.2 Flow of Events

2.2.2.1 使用别的 use case

- The system uses the included use case *Login* to verify the Registrar.

2.2.2.2 选择列表

- The system prompts the Registrar to select the desired activity: ADD, DELETE, REVIEW, or QUIT.
- If the Registrar selects the activity ADD, then he uses the included use case *AddCourse* to add a course.
- If the Registrar selects the activity DELETE, then he uses the included use case *DeleteCourse* to delete a course.
- If the Registrar selects the activity REVIEW, then he uses the included use case *ReviewCourses* to review existing courses.
- If the Registrar selects the activity QUIT, the system returns to the semester selection screen.

2.3 EXAMPLES

2.3.1 The Bank Customer Withdraw Money Using ATM

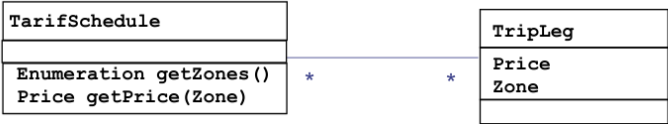
Actor	Bank Customer (Initiating Actor), Bank
Description	The Bank Customer Withdraw Money Using an ATM
Preconditions	<ol style="list-style-type: none"> Bank Customer has a Bank Account with the Bank Bank Customer has an ATM Card and PIN
Postconditions	<ol style="list-style-type: none"> Bank Customer receives the requested cash. Bank Customer receives an explanation from the ATM why the cash could not be dispensed.
Flow of Events	<ol style="list-style-type: none"> The Bank Customer inputs the card into the ATM. The ATM requests the input of a six-digit PIN. The Bank Customer enters a PIN. The ATM verifies the card and PIN with the account information in the Bank. If the card and PIN are verified the ATM requests the amount to withdraw. The Bank Customer enters an amount. The ATM verifies the amount is available in the customer account. If the customer account has sufficient funds, the ATM outputs the money, card, and receipt. The Bank Customer removes card, receipt and money.
Alternative Flows	AF-S5: If the card and PIN are invalid.

	<ol style="list-style-type: none"> 1. The ATM displays the message “Invalid card and PIN. Please try again!” for 2 seconds. 2. The ATM returns to step 2. <p>AF-S8: If the customer account has insufficient funds.</p> <ol style="list-style-type: none"> 1. The ATM displays the message “Insufficient funds in your account! Please enter a smaller amount!” for 2 seconds. 2. The ATM returns to step 6.
Exceptions	<p>EX1: If the Bank Customer enters an invalid PIN three times.</p> <ol style="list-style-type: none"> 1. The ATM displays the message “Card is suspended! Please call customer service (1633) to reactive the card” for 2 seconds. 2. The ATM outputs the card. <p>EX2: If the ATM has insufficient funds.</p> <ol style="list-style-type: none"> 1. The ATM displays the message “This ATM is off service due to insufficient funds!” at the start screen. 2. The ATM does not accept the card.

Class Diagram

1 RELATIONSHIP

1.1 ASSOCIATION



有多个：*

1~6 个：1 ... 6

1.2 GENERALIZATION (INHERITANCE)

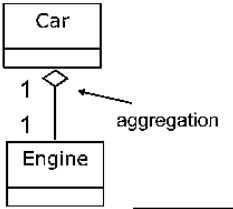
类型	线条	箭头
class	实线	黑色箭头
abstract class		白色空心箭头
interface	虚线	

1.3 组合

1.3.1 Aggregation

第一种说法：“is part of”

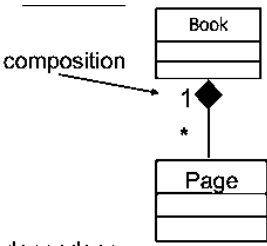
第二种说法：班级和学生，没有班级，学生依然能存在。没有汽车，engine 依然能存在。



1.3.2 Composition

第一种说法：“is entirely made of”

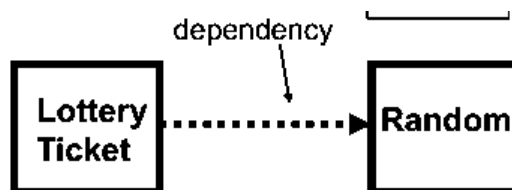
第二种说法：没有书，页数就变得没有意义



1.4 DEPENDENCY

“暂时使用”，相当于 << use >>

献上可以写字，比如 << call >>, << read >>, << signal >>



1.5 COMMUNICATION ALLOWED

	Entity	Boundary	Control
Entity	1		1
Boundary			1
Control	1	1	1

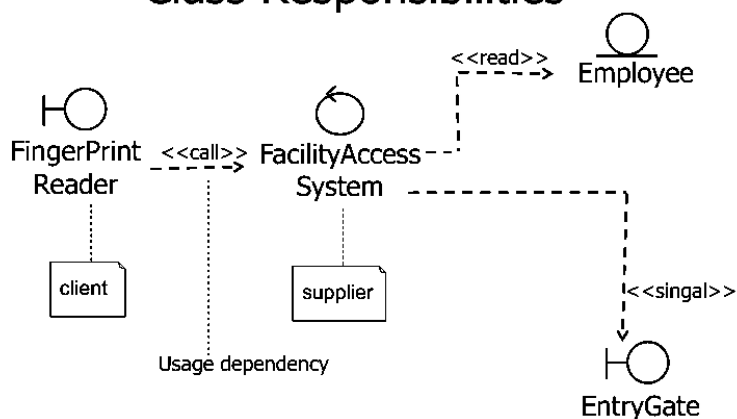
2 STEREOTYPES OF CLASSES (CLASS 的分类)

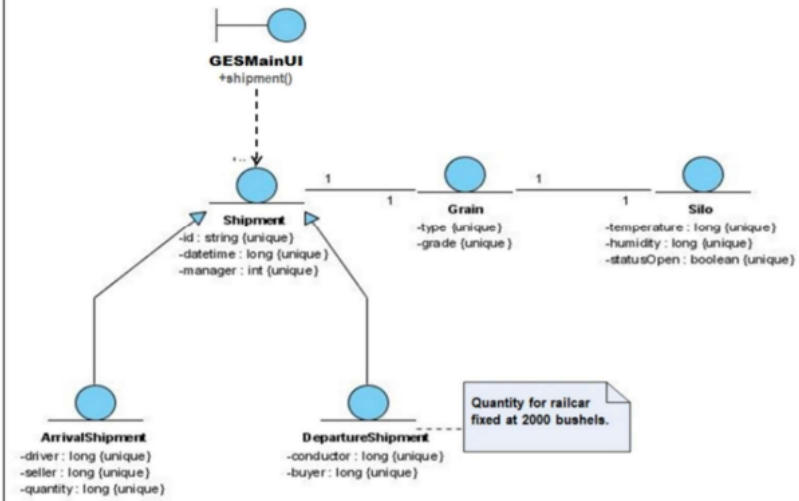
- ⊢○ Boundary Class <<boundary>>
– interaction between Actor & System
- Control Class <<control>>
– logic to realize use case
- Entity Class <<entity>>
– information tracked by System

3 EXAMPLES

3.1 REQUIREMENTS ANALYSIS PART (CONCEPTUAL CLASS DIAGRAM)

Stereotyping to Indicate Class Responsibilities







Sequence Diagram

1 ARROW

1.1 线

主动方传过去的（发送的信息）	实线
被动方传回来的（return 的信息）	虚线

1.2 箭头

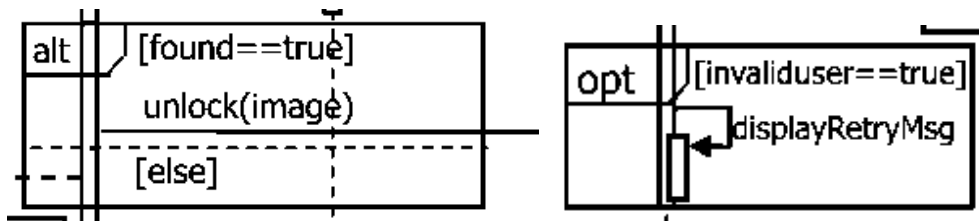
Async	发完信息就不管了	
Sync	发完信息等待回复	

2 TYPES OF INTERACTION FRAMES

2.1 MEANING

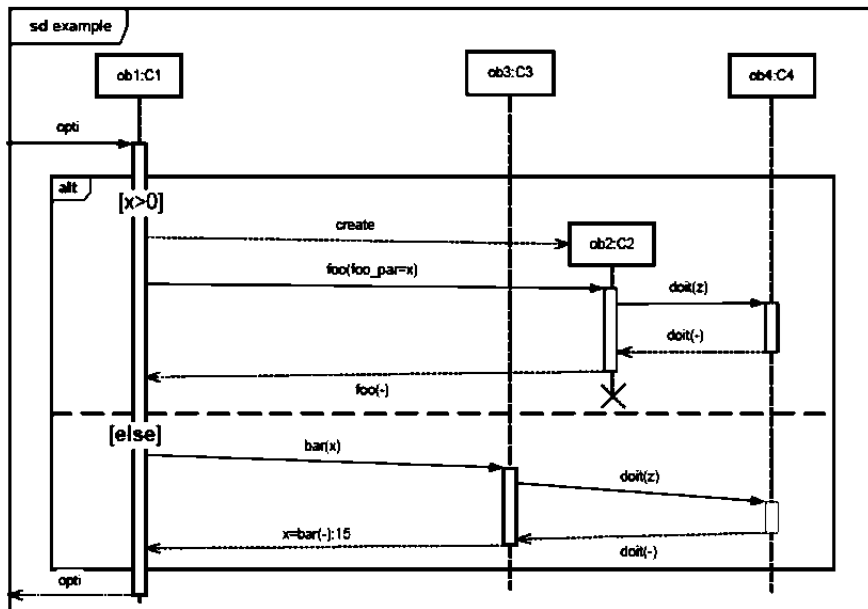
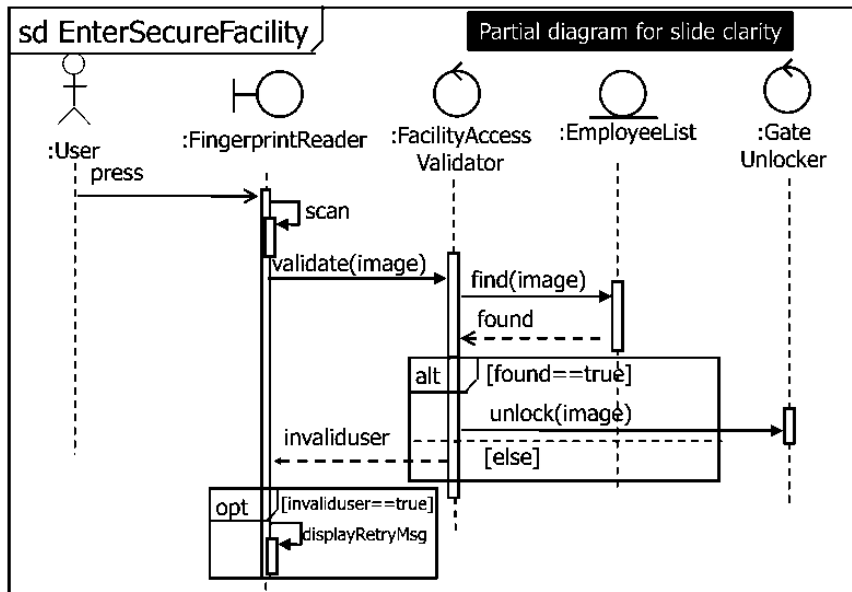
sd	Sequence diagram
ref	Reference another sequence diagram（子图）
loop	循环（for, while）
opt / alt	分支结构（if-then, if-then-else, switch-case）
par	每个 action 是并行的

2.2 EXAMPLES



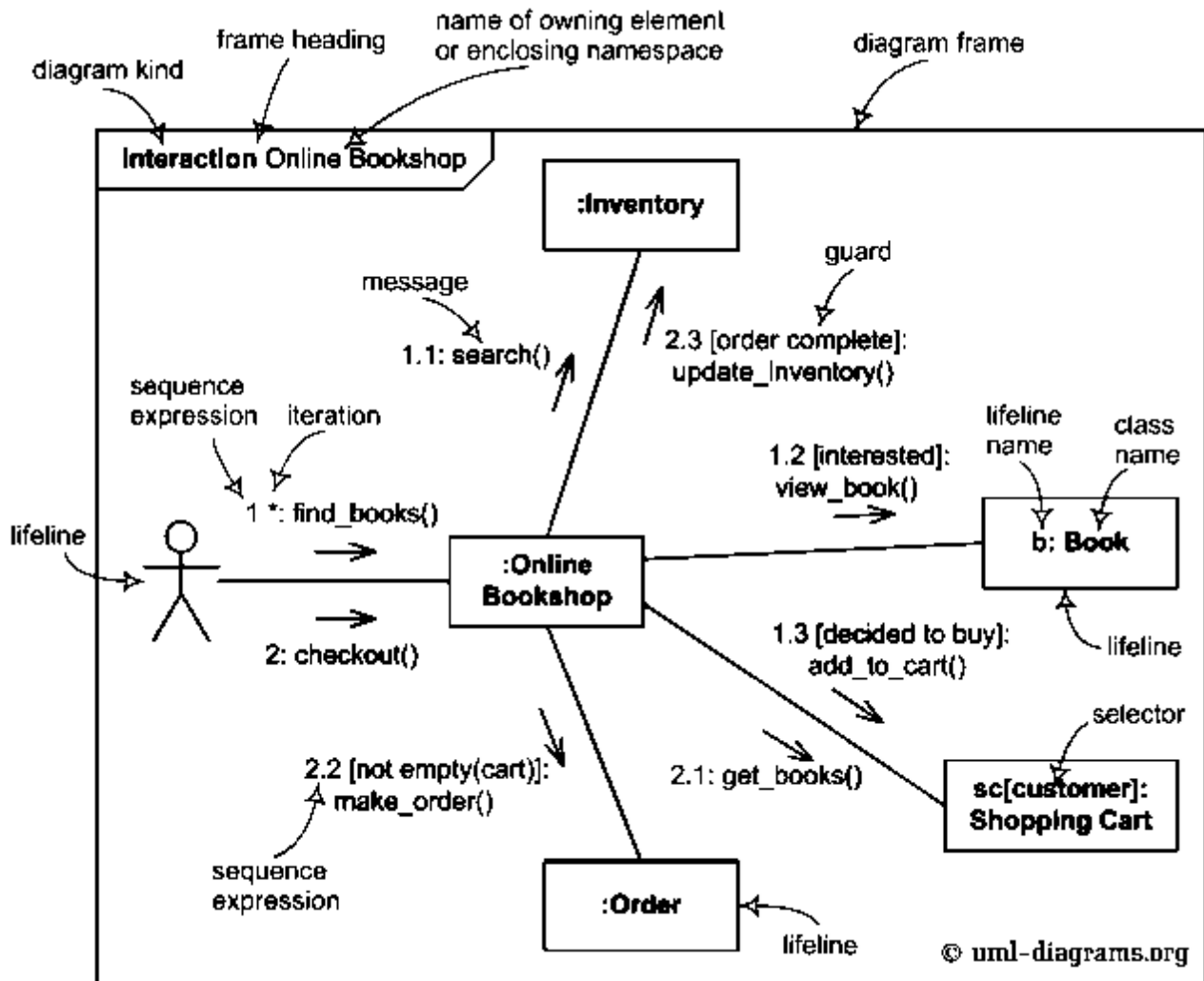
3 EXAMPLES

Refine EnterSecureFacility Sequence Diagram



Communication Diagram

1 DETAILS

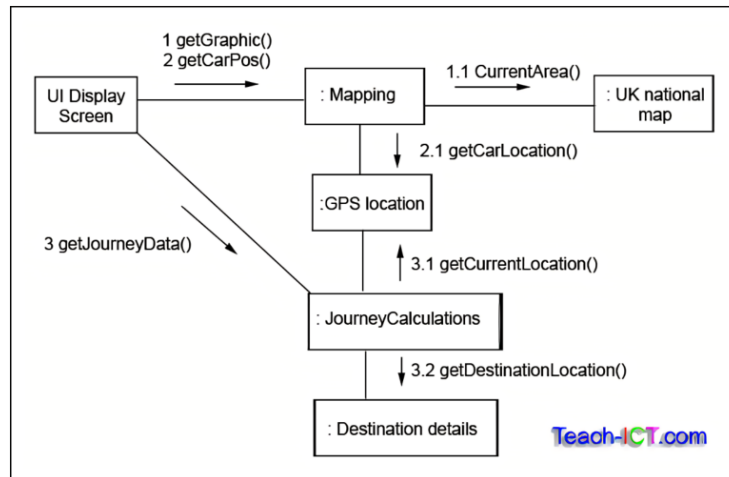


1 *: find_books(), 这个*表示的是可能这个操作可能会经历多次

1.2 [interested] view_book(), 这个方括号表示的是 if (guard condition)

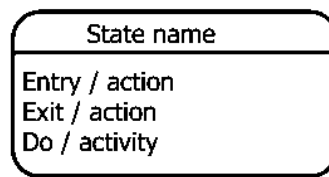
sc[customer]: Shopping Cart, 这个方括号也表示选择, 也就是这个 class 是给 customer 用的

2 EXAMPLES



State Machine Diagram / Dialog Map

1 STATE



Entry: 刚开始到这个状态那一刻的操作

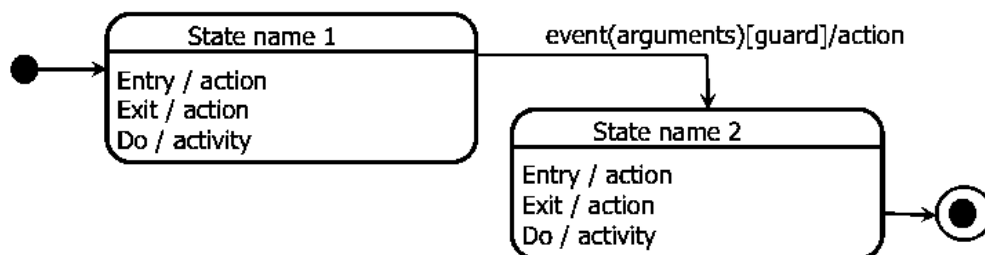
Exit: 离开这个状态那一刻的操作

Do: 停留在这个状态时的操作

起点: ●

终点: →●

2 ARROW



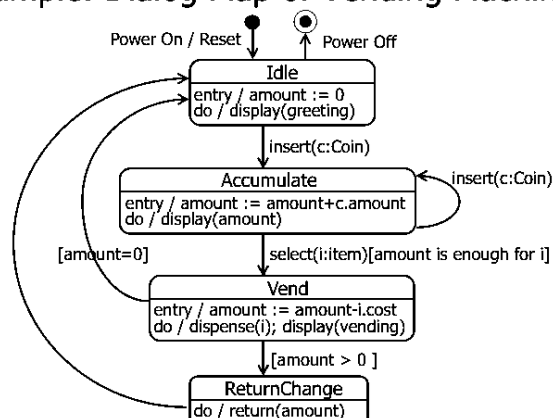
Event: 触发转换的事件

Guard: 必须为真才能执行转换的条件

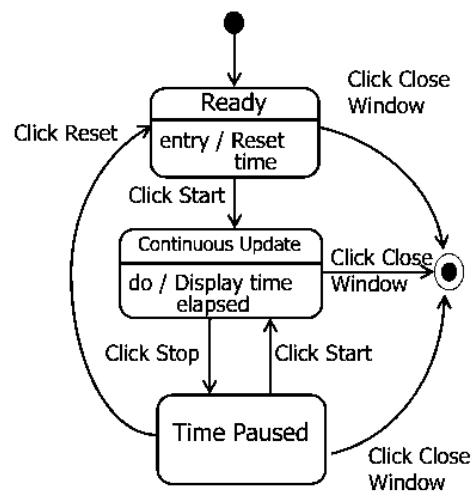
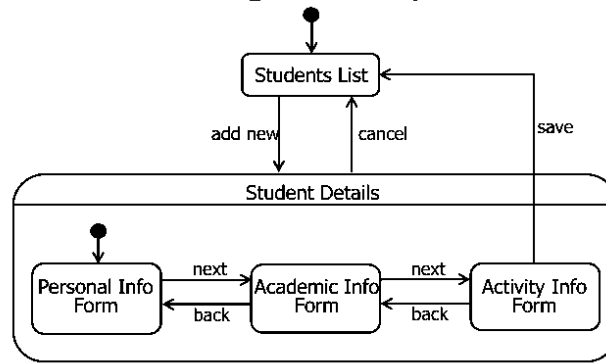
Action: 在转换过程中要执行的操作

3 EXAMPLES

Example: Dialog Map of Vending Machine



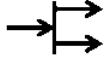






Example: Dialog Map of Student Management System



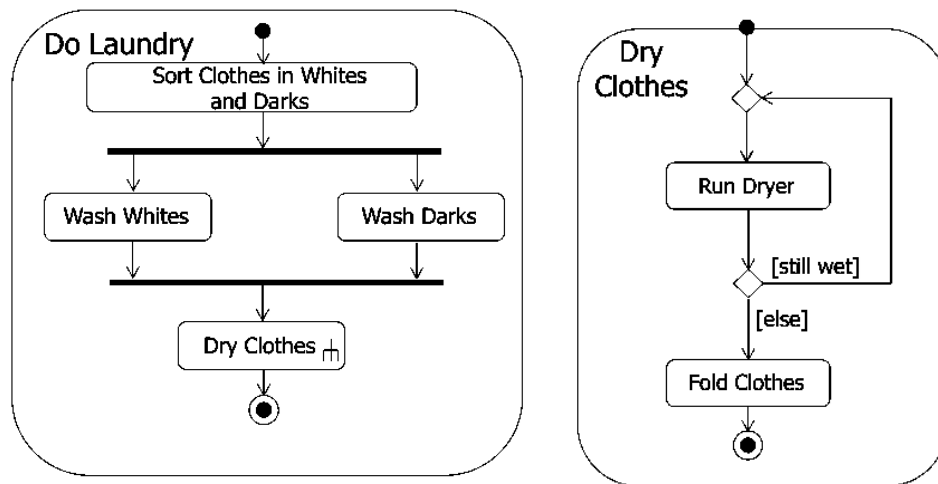
Activity Diagram

1 SYMBOL

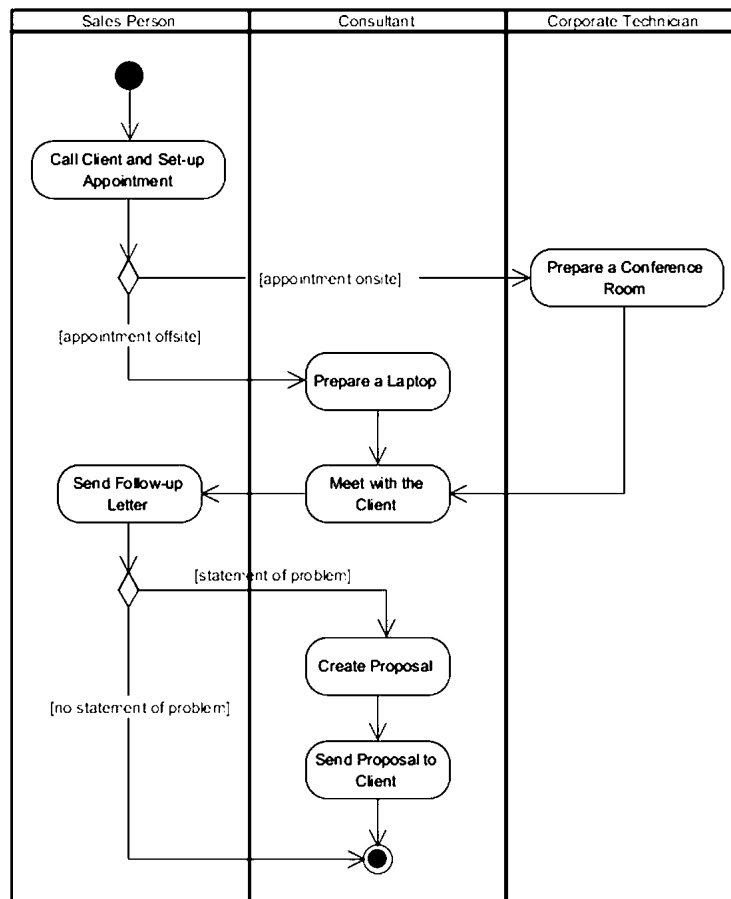
	Activity 是执行一个或多个基本操作的过程，通常用圆角矩形来表示。
	Activity 之间的控制流通常通过箭头来表示。
	把分成两条支线并行做
	把两条并行的支线合在一起（必须两条支线都做完了才能继续）
	分支 相当于 if
	两条进入的线中，只要有一条完成了就继续相当于是这两条线做完之后都是做这个任务。
	表示有子图

2 EXAMPLES

2.1 NORMAL



2.2 SLIMLINE STYLE



Control Flow Graph

1 EXAMPLE

