广东技术师范大学
Guangdong Polytechnic Normal University

《系统分析与设计》课程

Instructor: Wen Jianfeng
Email: wjfgdin@qq.com

系统分析与设计

第1讲：系统分析与设计概述

提纲

- 软件工程知识体系（SWEBOK）
- 软件开发生命周期（SDLC）
- 软件开发过程（Process）

- 通过小案例阐述分析与设计过程

- 面向对象概念（OO）
- 统一建模语言（UML）

软件工程知识体系（SWEBOK）

SWEBOK

- 2014年IEEE发布的SWEBOK（V3.0）（Software Engineering Body of Knowledge）中提到的15个知识领域（Knowledge Area, KA）

1. Software Requirements 【软件需求】
2. Software Design 【软件设计】
3. Software Construction 【软件构建】
4. Software Testing 【软件测试】
5. Software Maintenance 【软件维护】
6. Software Configuration Management 【软件配置管理】
7. Software Engineering Management 【软件工程管理】
8. Software Engineering Process 【软件工程过程】
9. Software Engineering Models and Methods 【软件工程模型和方法】
10. Software Quality 【软件质量】

SWEBOK

11. Software Engineering Professional Practice 【软件工程专业实践】
12. Software Engineering Economics 【软件工程经济学】
13. Computing Foundations 【计算基础】
14. Mathematical Foundations 【数学基础】
15. Engineering Foundations 【工程基础】

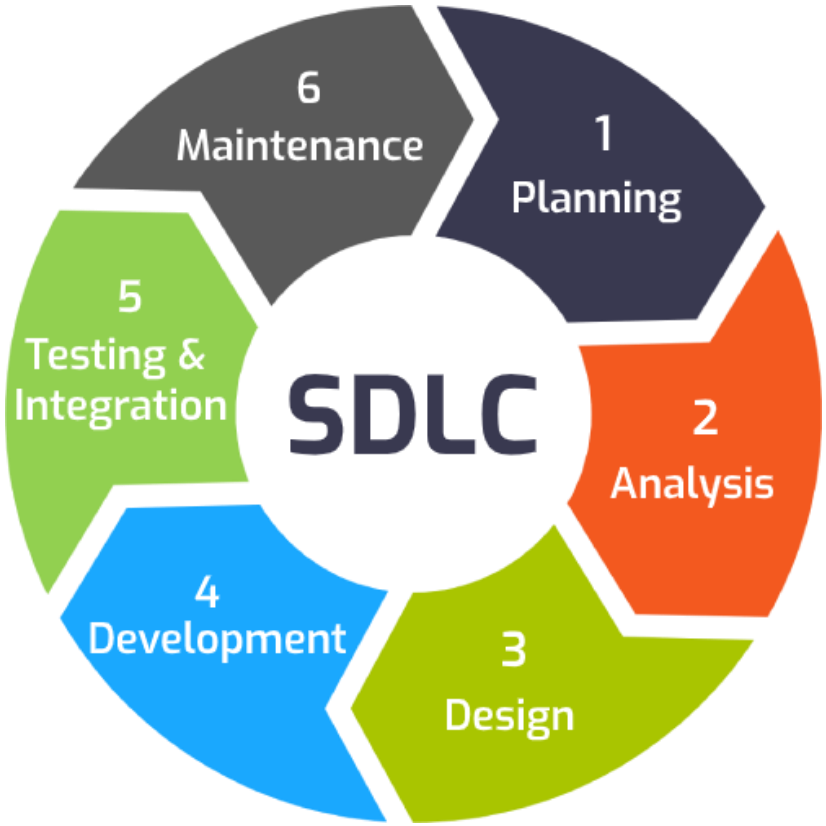
注：详细内容请参见第1讲
附件：SWEBOKv3.pdf）

SWEBOK下载地址：https://www.computer.org/education/bodies-of-knowledge/software-engineering

软件开发生命周期（SDLC）

软件开发生命周期

- SDLC【Software Development Life Cycle】
  - SDLC是一个框架，它指定了规划、分析、设计、实现、测试、部署和维护一个软件系统所需的所有活动



Systems Development Life Cycle (SDLC)
Life-Cycle Phases

Initiation

Begins when a sponsor identifies a need or an opportunity. Concept Proposal is created.

System Concept Development

Defines the scope or boundary of the concept. Includes Systems Boundary Document, Cost Benefit Analysis, Risk Management Plan and Feasibility Study.

Planning

Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a solution.

Requirements Analysis

Analyzes user needs and develops user requirements. Creates a detailed Functional Requirements Document.

Design

Transforms detailed requirements into complete, detailed System Design Document. Focuses on how to deliver the required functionality.

Development

Converts a design into a complete information system. Includes acquiring and installing systems environment; creating and testing databases/ preparing test case procedures; preparing test files; coding, compiling, refining programs; performing test readiness review and procurement activities.

Integration and Test

Demonstrates that the developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

Implementation

Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phase.

Operations and Maintenance

Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

Disposition

Describes end-of-system activities. emphasis is given to proper preservation of data.

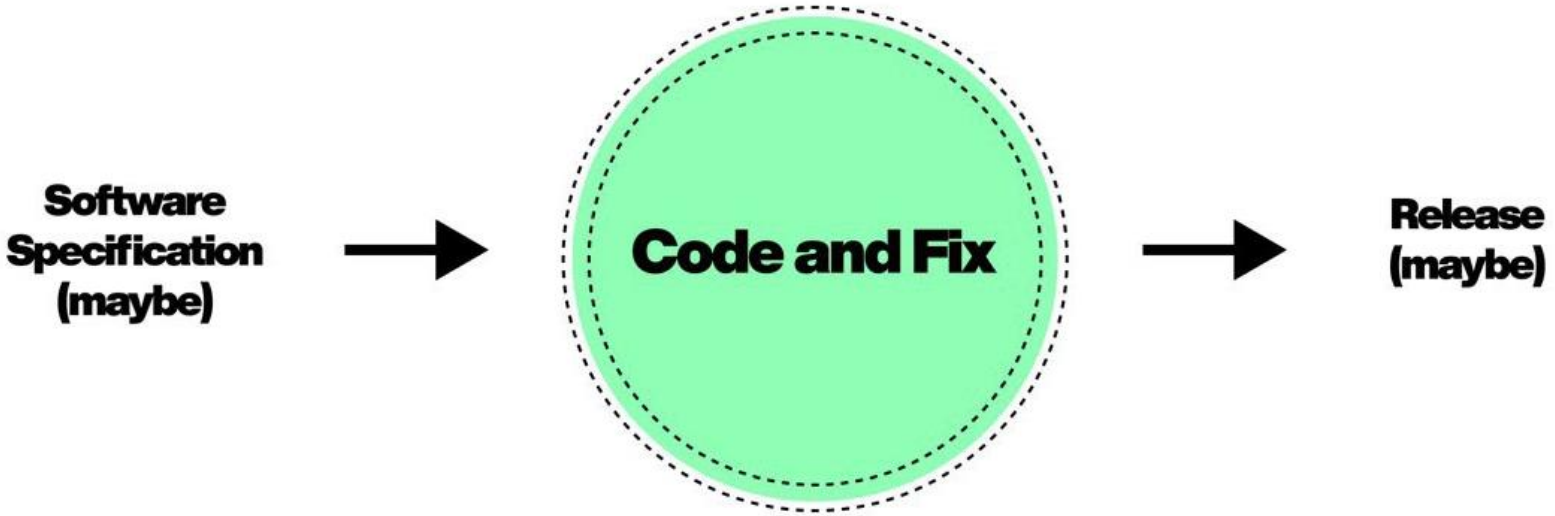
软件开发过程（Process）

软件开发过程

- 开发过程【Software Development Process】
  - 为实现SDLC的每个核心过程的所有活动而指定的一组广泛的指导准则
- 过程模型【Process Model】
  - Code and Fix Model（写了再改模型）
  - Waterfall Model（瀑布模型）
  - Sashimi Waterfall Model（生鱼片瀑布模型）
  - Waterfall with Subprojects（带子项目的瀑布模型）
  - Spiral Model（螺旋模型）
  - Rational Unified Process（RUP统一过程）
  - Prototyping（原型）
  - ... ...
  - Agile Process（敏捷过程）

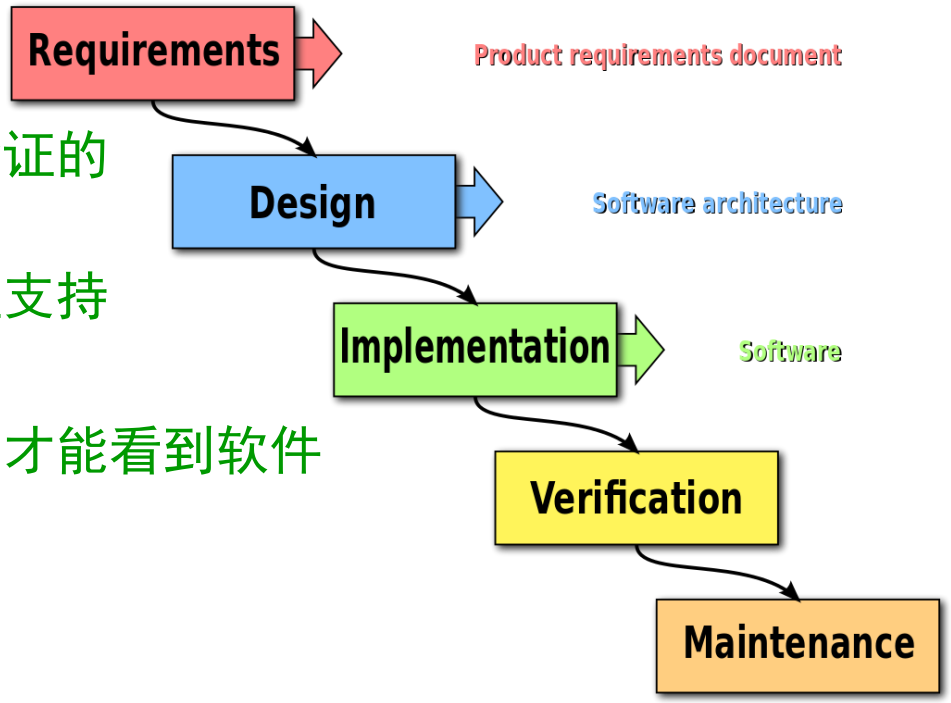
过程模型

- Code and Fix Model（写了再改模型）
  - 优点：无需太多专业知识；无需太多开销
  - 缺点：无进度评估；无质量评估
  - 适用于：小项目、概念验证、Demo演示、原型



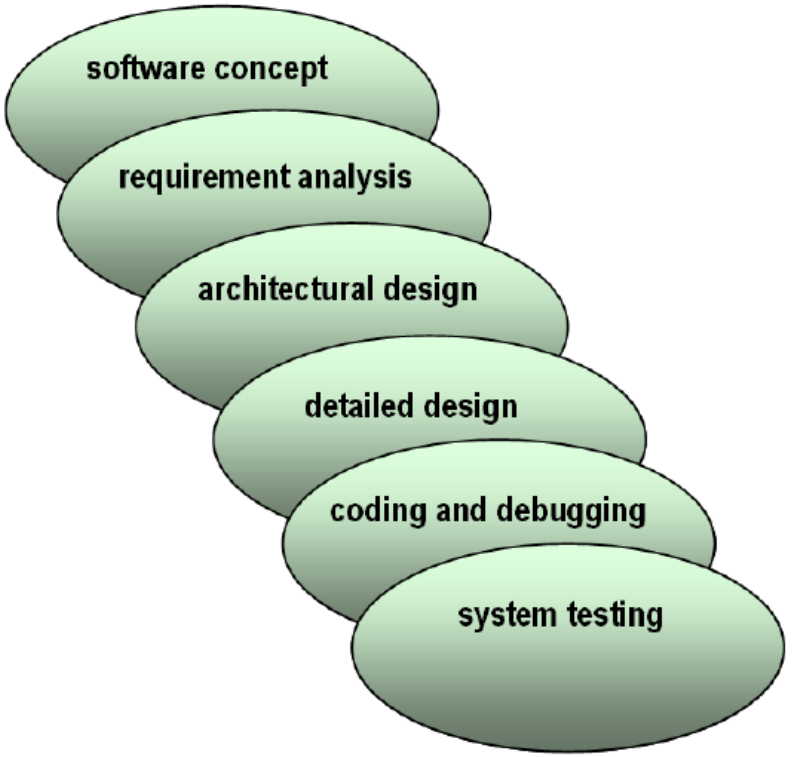
过程模型

- Waterfall Model（瀑布模型）
  - 它反映了软件开发的思维模型，它是分离的、串行的、文档驱动的模型
  - 优点：
    - 每一步的结果都是可验证的
    - 减少风险
    - 给团队提供稳定的流程支持
  - 缺点：
    - 要等到开发过程结束后才能看到软件
  - 适用于：
    - 产品需求比较稳定的
    - 使用的技术是娴熟的
    - 复杂的、被充分理解的项目



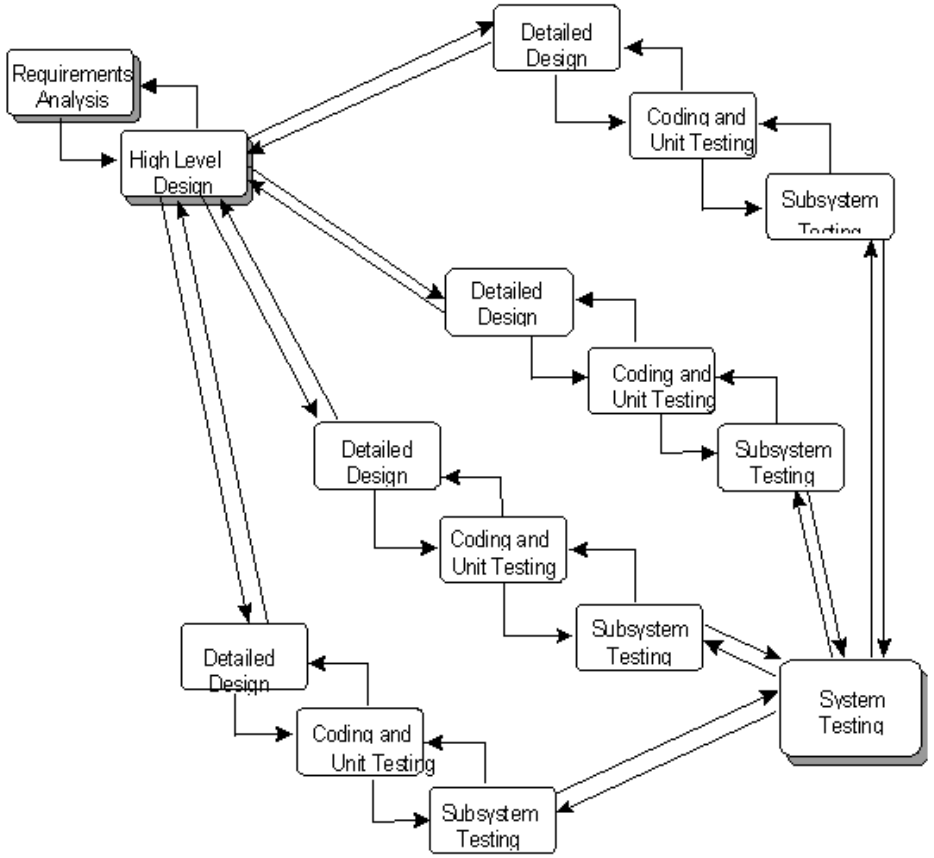
过程模型

- Sashimi Waterfall Model（生鱼片瀑布模型）
  - 即：Waterfall with Overlapping（带层叠的瀑布模型）
  - 相邻阶段有层叠
  - 优点：
    - 降低成本
  - 缺点：
    - 里程碑重叠
    - 阶段重叠并行易导致沟通问题



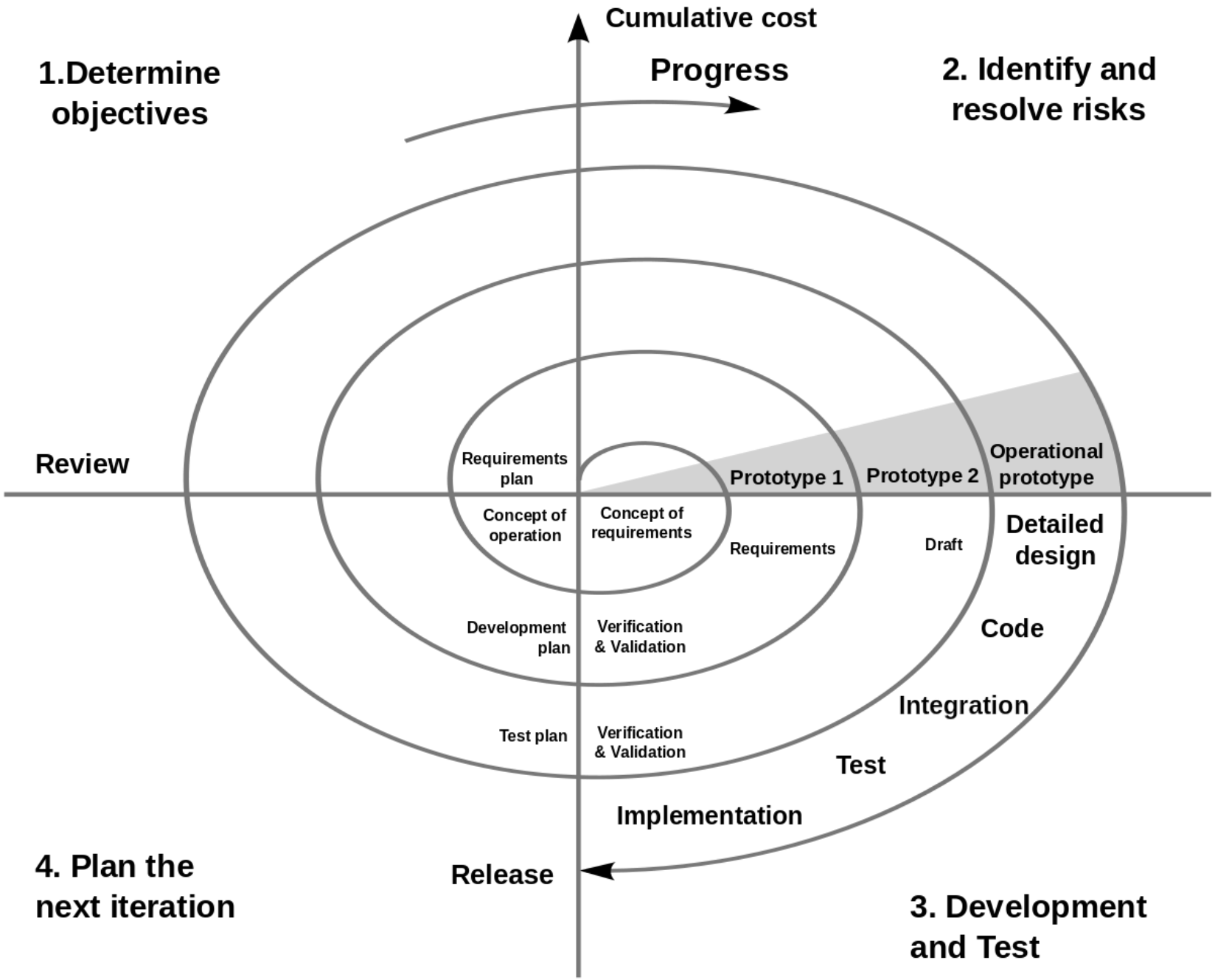
过程模型

- Waterfall with Subprojects（带子项目的瀑布模型）
  - 优点：分而治之
  - 缺点：不可预见的互相依赖的风险、集成成本增加



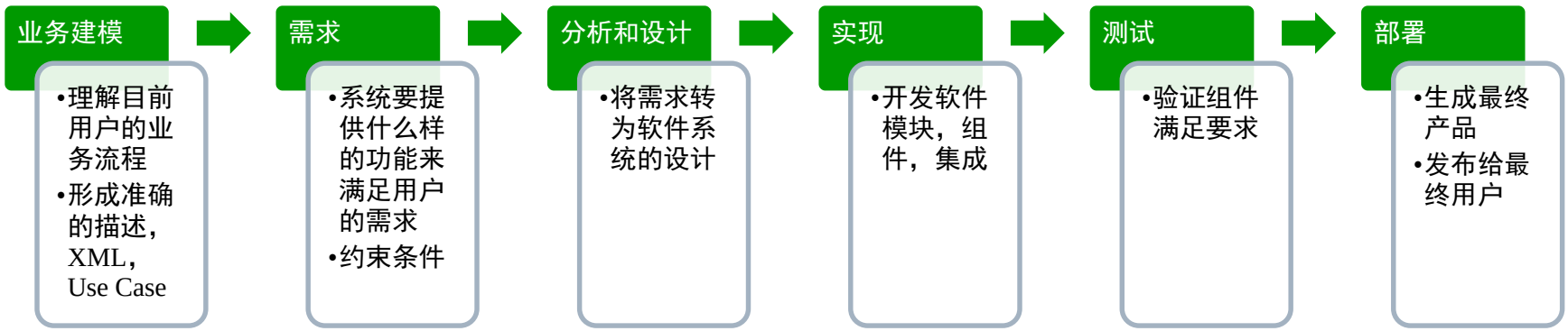
过程模型

- Spiral Model（螺旋模型）
  - 优点：
    - 在每一个周期都能衡量并控制风险
    - 随着投入的增加和产品的运行，产品失败的风险在降低
  - 缺点：
    - 需要高水平的管理团队
    - 很难明确定义目标和可验证的里程碑
  - 适用于：
    - 复杂的项目且有许多方面仍不明确
    - 知识丰富、经验丰富的团队

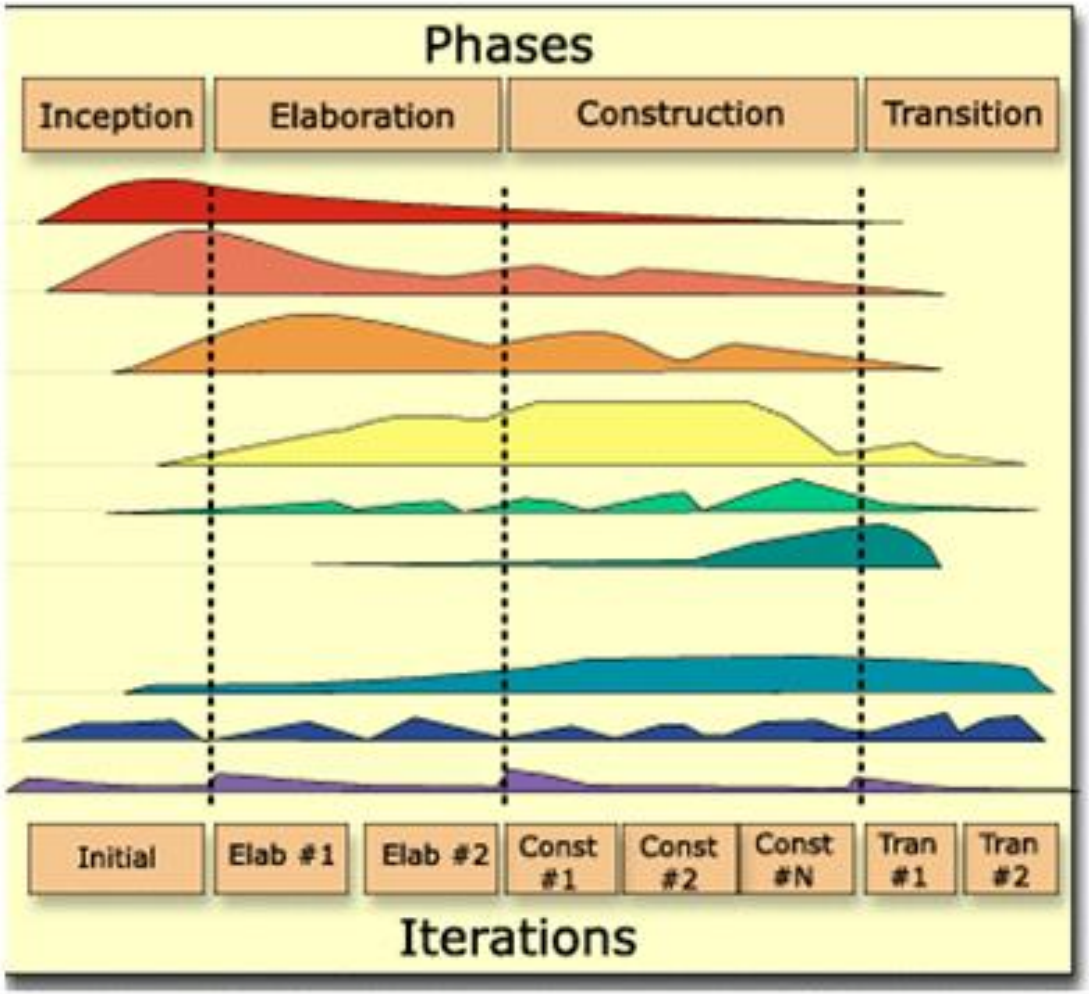

Cumulative cost
Progress
1.Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Prototype 1
Prototype 2
Operational prototype
Concept of operation
Concept of requirements
Requirements
Draft
Detailed design
Development plan
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
4. Plan the next iteration
Release
3. Development and Test

过程模型

- Rational Unified Process（RUP统一过程）
  - 前面从瀑布模型开始的各种模型的共同点：
    - 重计划
    - 重事先设计
    - 重文档表达
  - 这一类的方法中集大成者要算RUP，它把软件开发的各个阶段整合在一个统一的框架里
  - RUP的具体流程



RUP Phase Model



过程模型

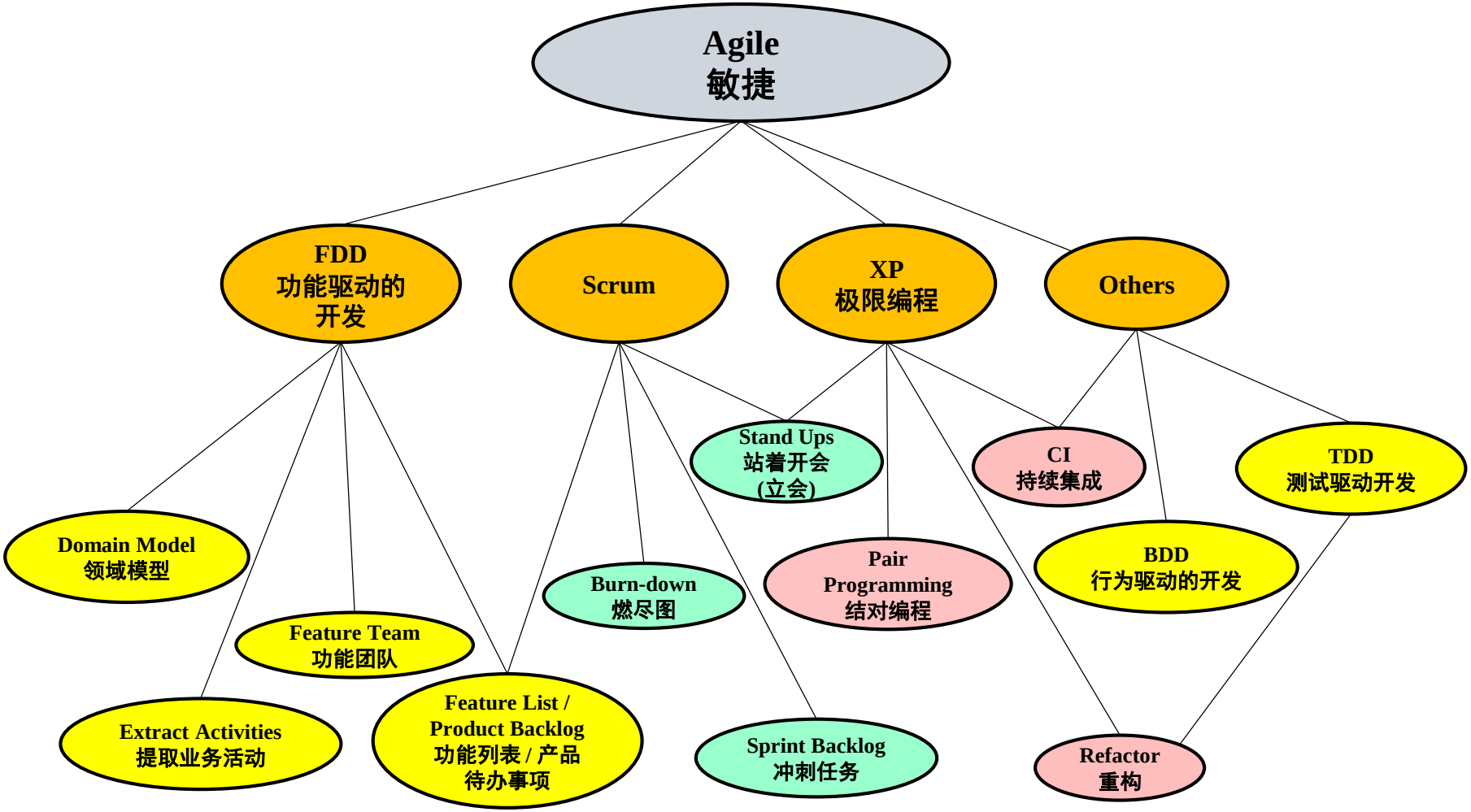
- Prototyping（原型）
  - 步骤：
    - 确定基本的需求
    - 开发初始的原型
    - 评审
    - 修改和增强原型
  - 原型的类别
    - Throwaway prototyping（丢弃式原型）
    - Evolutionary prototyping（进化式原型）
    - Incremental prototyping（增量式原型）
  - 优点：
    - 降低时间和成本；提高用户参与度
  - 缺点：
    - 分析不足；原型与最终系统的不同；开发者对用户目标的误解；原型的额外开发费用

过程模型

- 敏捷软件开发【Agile Software Development】
  - 基本思想：在开发过程中灵活、快速地响应新的和变化的需求
  - 敏捷是一组软件开发思想的统称
  - 以一组软件开发方法论为代表
  - 具体体现为许多互相支援的概念，工具和实践经验

敏捷软件开发

- 什么是敏捷？



敏捷软件开发

- 敏捷产生的背景
  - 最初的软件（20 世纪六七十年代）的顾客都是大型研究机构、军方、美国航空航天局、大型股票交易公司，他们需要通过软件系统来搞科学计算、军方项目、登月项目、股票交易系统等超级复杂的项目。这些项目对功能的要求非常严格，对计算的准确度要求相当高。
  - 20 世纪八九十年代，软件进入桌面软件时代，开发周期明显缩短，各种新的方法开始进入实用阶段。但是软件发布的媒介还是软盘、CD、DVD，做好一个发布需要较大的经济投入，不能频繁更新版本。
  - 互联网时代，大部分的服务是通过网络服务器端实现，在客户端有各种方便的推送（Push）渠道。一般消费者成为主要用户。网络的传播速度和广度，使得知识的获取变得更加容易，很多软件服务可以由一个小团队来实现。同时，技术更新的速度在加快，用户需求的变化也在加快，开发流程必须跟上这些快速变化的节奏。于是敏捷就产生了。

敏捷软件开发

- 敏捷和现有做法的区别
  - 从2001年开始，一些软件界的专家开始倡导“敏捷”的价值观和流程，他们肯定了流行做法的价值（左列），但是强调敏捷的做法（右列）更能带来价值。

Existing Practices（现有做法）	**Agile Approach（敏捷方法）**
Processes and Tools 流程和工具	Individual and Interaction 个人和交流
Comprehensive Documentation 完备的文档	Working Software 可用的软件
Contract Negotiation 为合同谈判	Customer Collaboration 与客户合作
Following a plan 执行原定计划	Respond to Change 响应变化

敏捷软件开发

- Agile Principles（敏捷原则）
  1. 尽早并持续地交付有价值的软件以满足顾客需求
  2. 敏捷过程欢迎需求的变化，并利用这种变化来提高用户的竞争优势
  3. 经常发布可用的软件，发布间隔可以从几周到几个月，能短则短
  4. 业务人员和开发人员在项目开发过程中应该每天共同工作
  5. 以有进取心的人为项目核心，充分支持信任他们
  6. 无论团队内外，面对面的交流始终是最有效的沟通方式
  7. 可用的软件是衡量项目进展的主要指标

敏捷软件开发

- Agile Principles（敏捷原则）（续）
  8. 敏捷过程倡导可持续的开发。领导、团队和用户应该能按照目前步调持续合作下去
  9. 坚持不懈地追求技术卓越和良好设计，以增强敏捷能力
  10. 以简洁为本，它是极力减少不必要工作量的艺术
  11. 最好的架构、需求和设计出自自组织团队
  12. 团队定期地反思如何能提高成效，并依此调整自身的举止表现

敏捷软件开发

- XP（eXtreme Programming，极限编程）
  - 把好办法（best practice）发挥到极致（extreme）

如果……	发挥到极致就变成……
满足顾客的需求很重要	那就用顾客的语言和行为来指导功能的开发 ——Behavior Driven Development
顾客表达能力不强	那就请顾客代表和团队人员一起工作
测试/单元测试能帮助提高质量	那就先写单元测试，从测试开始写程序 ——Test Driven Development
代码复审可以找到错误	从一开始就处于“复审”状态：结对编程
计划没有变化快	那就别做详细的设计和文档。通过增量开发、重构和频繁地发布来满足用户的需求
代码重构会提高质量	那就持续不断地重构
其他好方法……	发挥到极限的做法……

敏捷软件开发

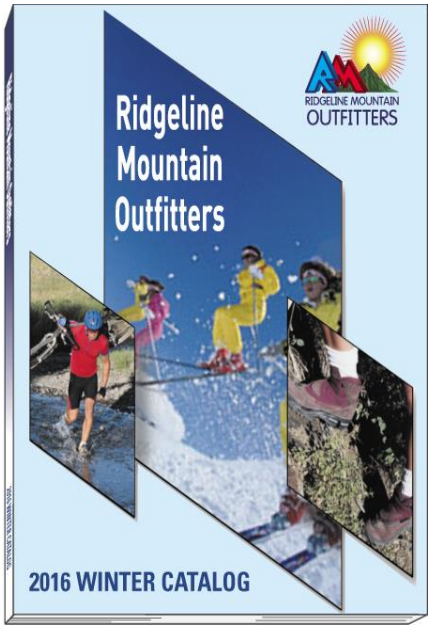
□ 用瀑布还是敏捷？

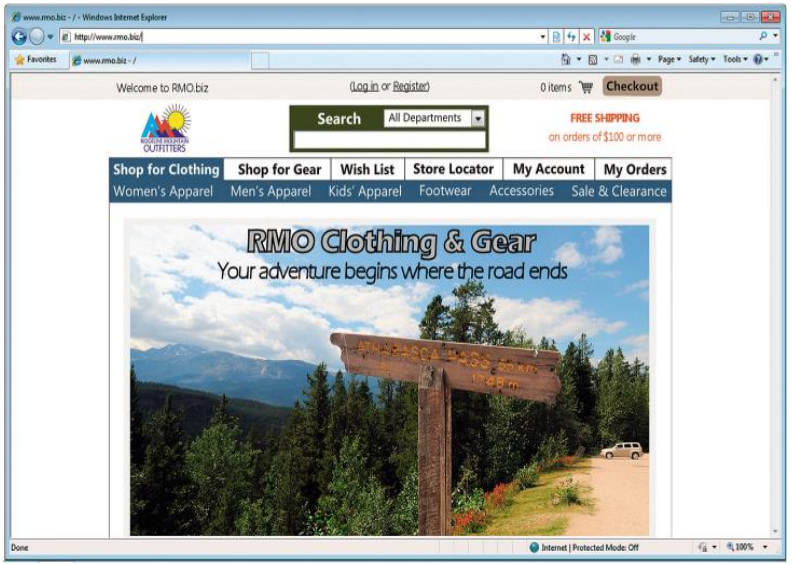
问题	Yes —— 偏向传统的瀑布 + 文档的流程	No —— 偏向敏捷流程
1. 项目需要有明确的 spec 么？		
2. 项目没有明确的用户，也无法联系用户进行沟通。		
3. 软件系统是大型的么？		
4. 软件系统是复杂的么？例如实时系统。		
5. 软件的生命周期很长么？		
6. 你使用比较差的软件工具么？		
7. 软件项目成员是分布在不同的地区么？		
8. 团队是否有“文档为先”的传统？		
9. 团队的编程技术较差么？		
10. 要交付的软件系统是否要通过某种行业规定或行政法规的批准？		

通过小案例阐述分析与设计过程

一个小案例项目

- 公司：Ridgeline Mountain Outfitters (RMO)
  - 是一家大型的零售商，主要销售各种户外和运动服饰
  - 销售途径：实体店、邮件订单、电话订单、网站





一个小案例项目

- 项目：RMO Tradeshow System（商品展销会系统）
  - 小型信息系统
  - 此系统将被加入到更大的供应链管理系统中
  - 这里演示该项目开发的一个迭代

RMO Tradeshow System

- 问题【Problem】
  - 采购经理参加世界各地的服装展销会向供应商订购新产品
- 需求【Need】
  - 信息系统在贸易展会期间实时收集和跟踪供应商及其新产品的信息
- 建议立项开发贸易展销会系统【Tradeshow Project is Proposed】
  - 供应商信息子系统
  - 产品信息子系统

核心过程一【Core Process 1】

- 初始活动【Initial Project Activities】
  - 识别问题和编写系统愿景文档
    - 初步调查
    - 系统愿景文档
  - 项目获得批准
    - 与关键相关人员会面，包括管理层
    - 决策一致后，批准计划和预算

System Vision Document【系统愿景】

- Problem description
  【问题描述】
- System capabilities
  【系统功能】
- Business benefits
  【商业收益】

RMO Tradeshow System

Problem Description

Trade shows have become an important information source for new products, new fashions, and new fabrics. In addition to the large providers of outdoor clothing and fabrics, there are many smaller providers. It is important for RMO to capture information about these suppliers while the trade show is in progress. It is also important to obtain information about specific merchandise products that RMO plans to purchase. Additionally, if quality photographs of the products can be obtained while at the trade show, then the creation of online product pages is greatly facilitated.

It is recommended that a new system be developed and deployed so field purchasing agents can communicate more rapidly with the home office about suppliers and specific products of interest. This system should be deployed on portable equipment.

System Capabilities

The new system should be capable of:

- Collecting and storing information about the manufacturer/wholesaler (suppliers)
- Collecting and storing information about sales representatives and other key personnel for each supplier
- Collecting information about products
- Taking pictures of products (and/or uploading stock images of products)
- Functioning as a stand-alone without connection
- Connecting via Wi-Fi (Internet) and transmitting data
- Connecting via telephone and transmitting data

Business Benefits

It is anticipated that the deployment of this new system will provide the following business benefits to RMO:

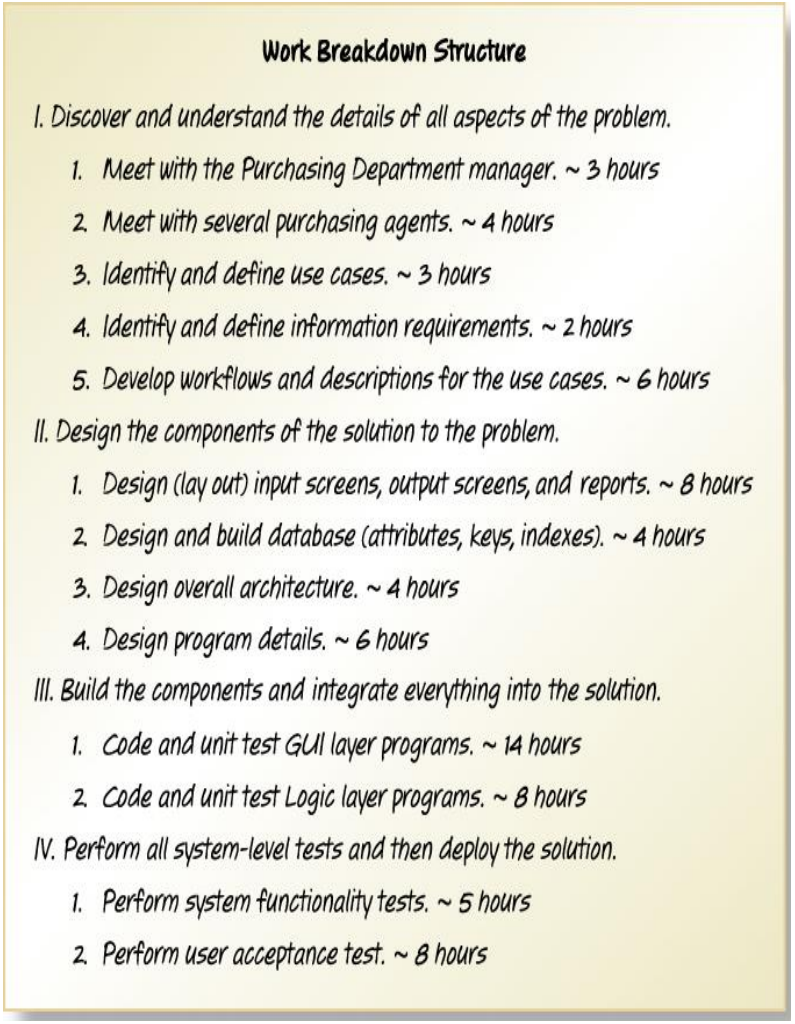
- Increase timely communication between trade show attendees and home office, thereby improving the quality and speed of purchase order decisions
- Maintain correct and current information about suppliers and their key personnel, thereby facilitating rapid communication with suppliers
- Maintain correct and rapid information and images about new products, thereby facilitating the development of catalogs and Web pages
- Expedite the placing of purchase orders for new merchandise, thereby catching trends more rapidly and speeding up product availability

核心过程二【Core Process 2】

- 规划项目【Plan the Project】
  - 确定主要功能模块
    - 供应商信息子系统
    - 产品信息子系统
  - 确定迭代的安排
    - 决定先开发供应商子系统
    - 计划一个小型直接的迭代
  - 确定项目团队成员和职责

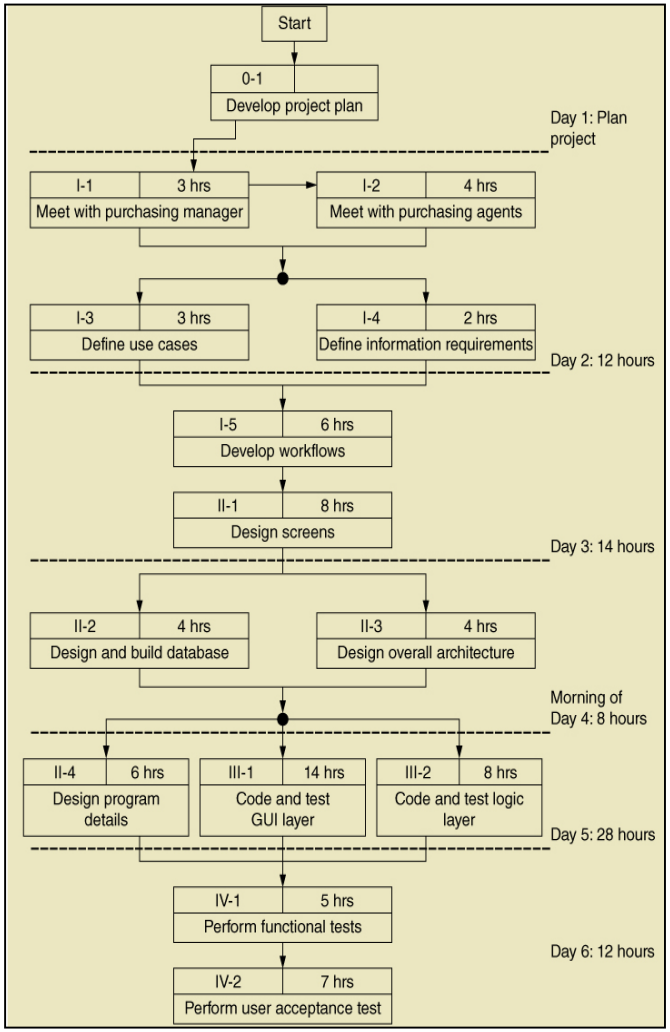
WBS【工作分解结构】

- WBS【Work Breakdown Structure】



Iteration Schedule【迭代计划】

- 迭代计划是对WBS的详细描述



核心过程三【Core Process 3】

- 发现和理解需求【Discover and Understand Details】
  - 调研以理解需求
  - 识别用例
  - 识别类
  - 深入调研以理解每个用例的细节
  - 理解并描述每个用例的具体工作流

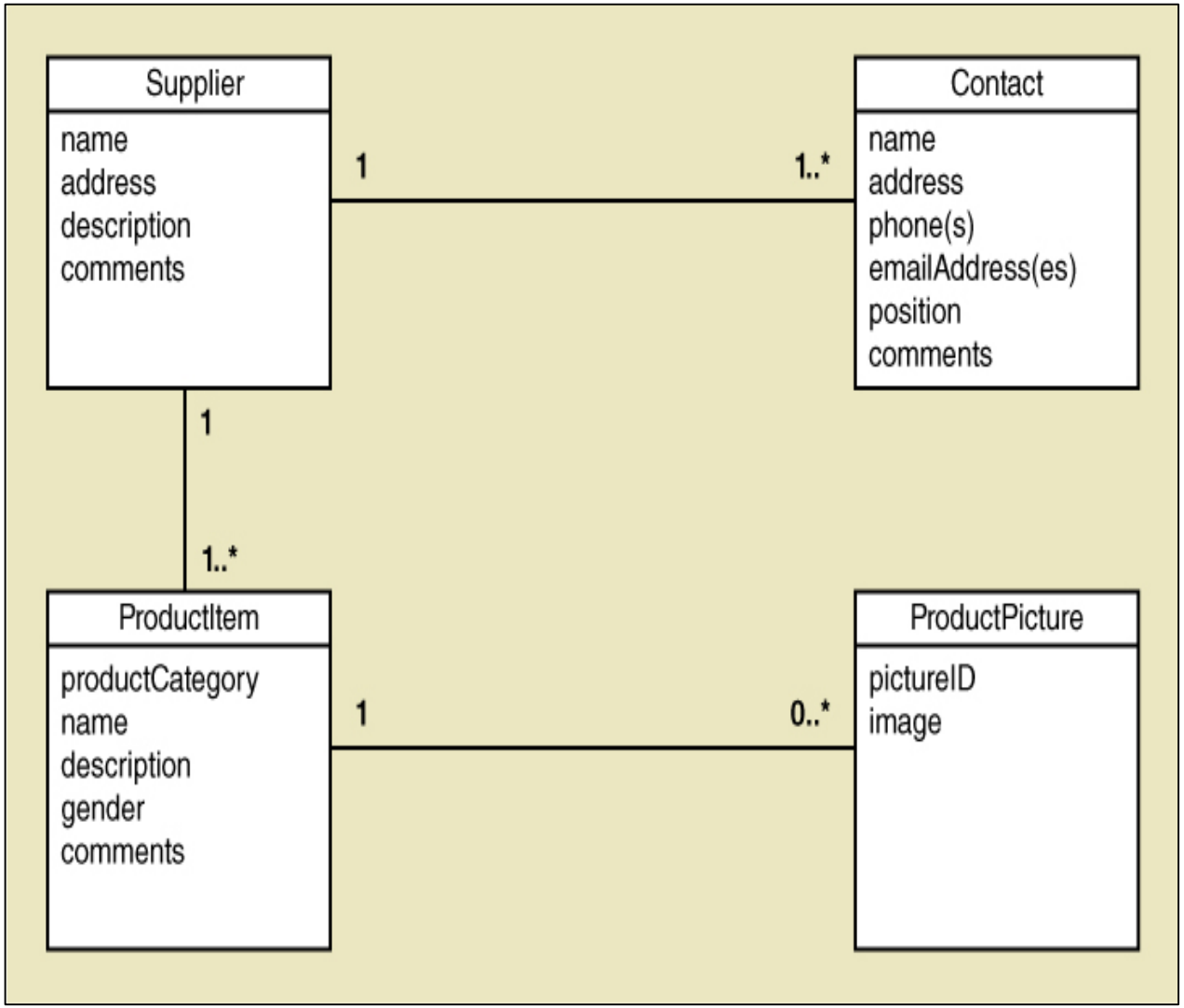
Identify Use Cases【识别用例】

Use Case	Description
Look up supplier	Using supplier name, find supplier information and contacts
Enter/update supplier information	Enter (new) or update (existing) supplier information
Look up contact	Using contact name, find contact information
Enter/update contact information	Enter (new) or update (existing) contact information
Look up product information	Using description or supplier name, look up product information
Enter/update product information	Enter (new) or update (existing) product information
Upload product image	Upload images of the merchandise product

Identify Domain Classes【识别类】

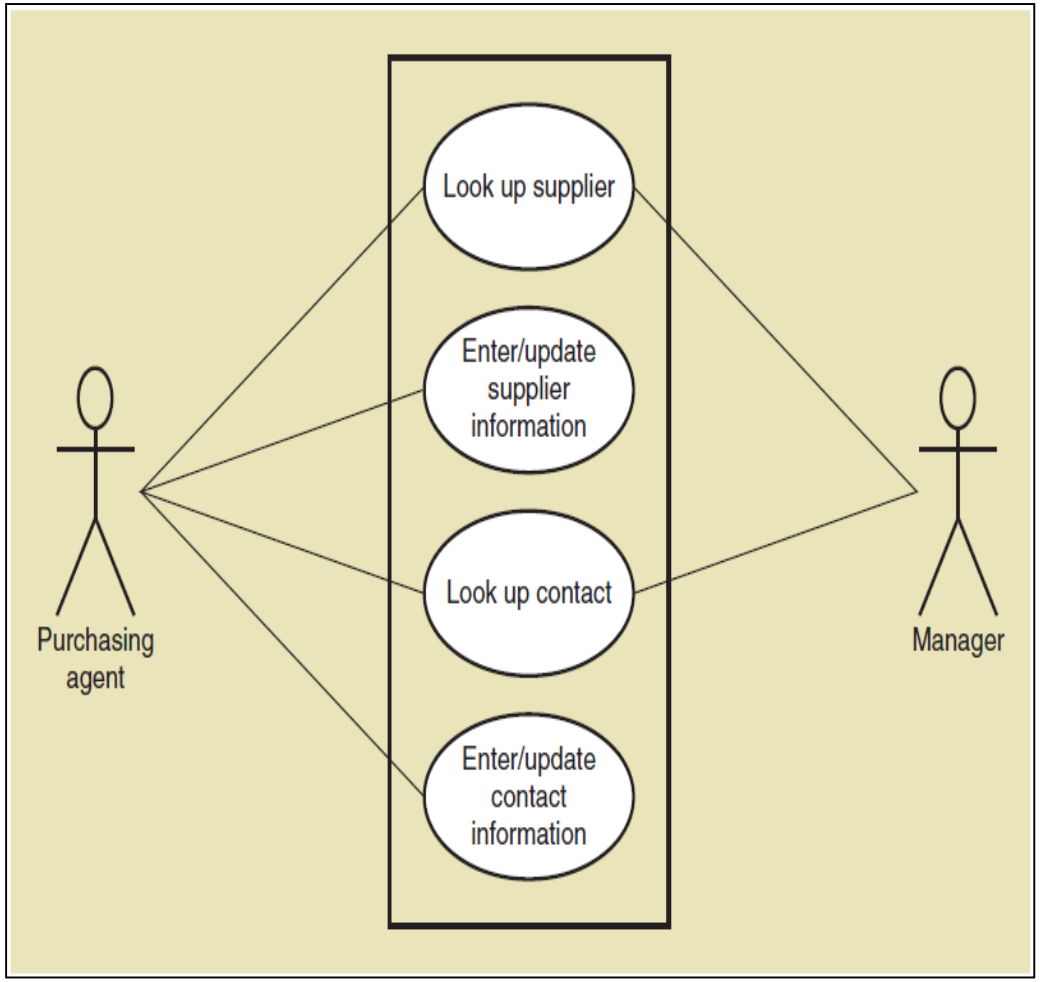
Object Classes	Attributes
Supplier	supplier name, address, description, comments
Contact	name, address, phone(s), e-mail address(es), position, comments
Product	category, name, description, gender, comments
ProductPicture	ID, image

Domain Class Diagram【领域类图】



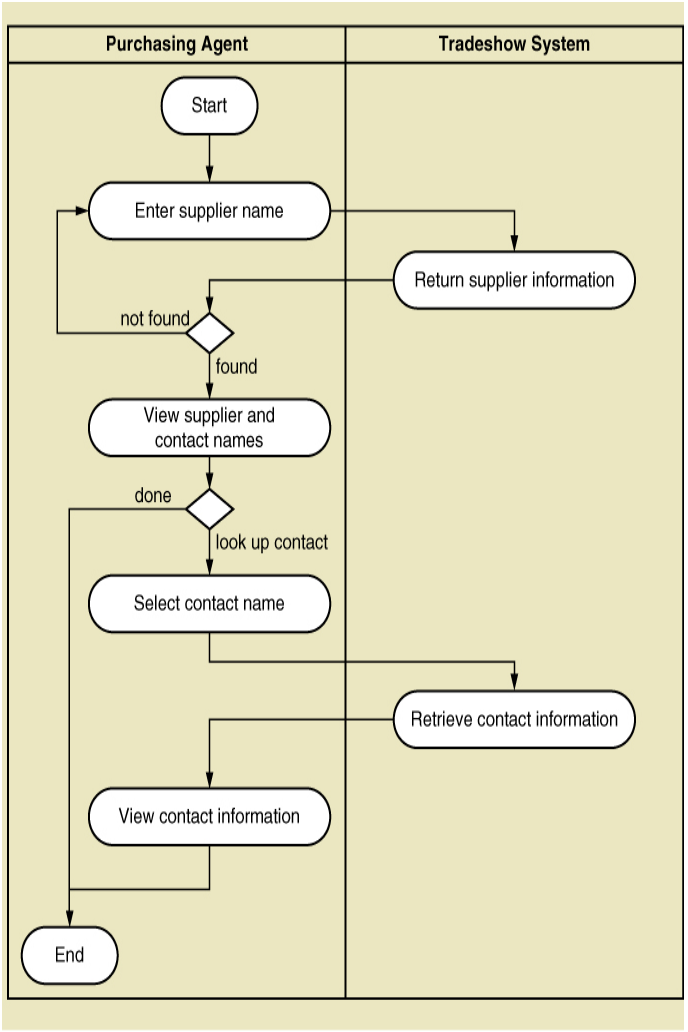
Use Case Diagram【用例图】

- 注：Supplier Information Subsystem的用例图



Activity Diagram【活动图】

- 注：Look up supplier用例的活动图

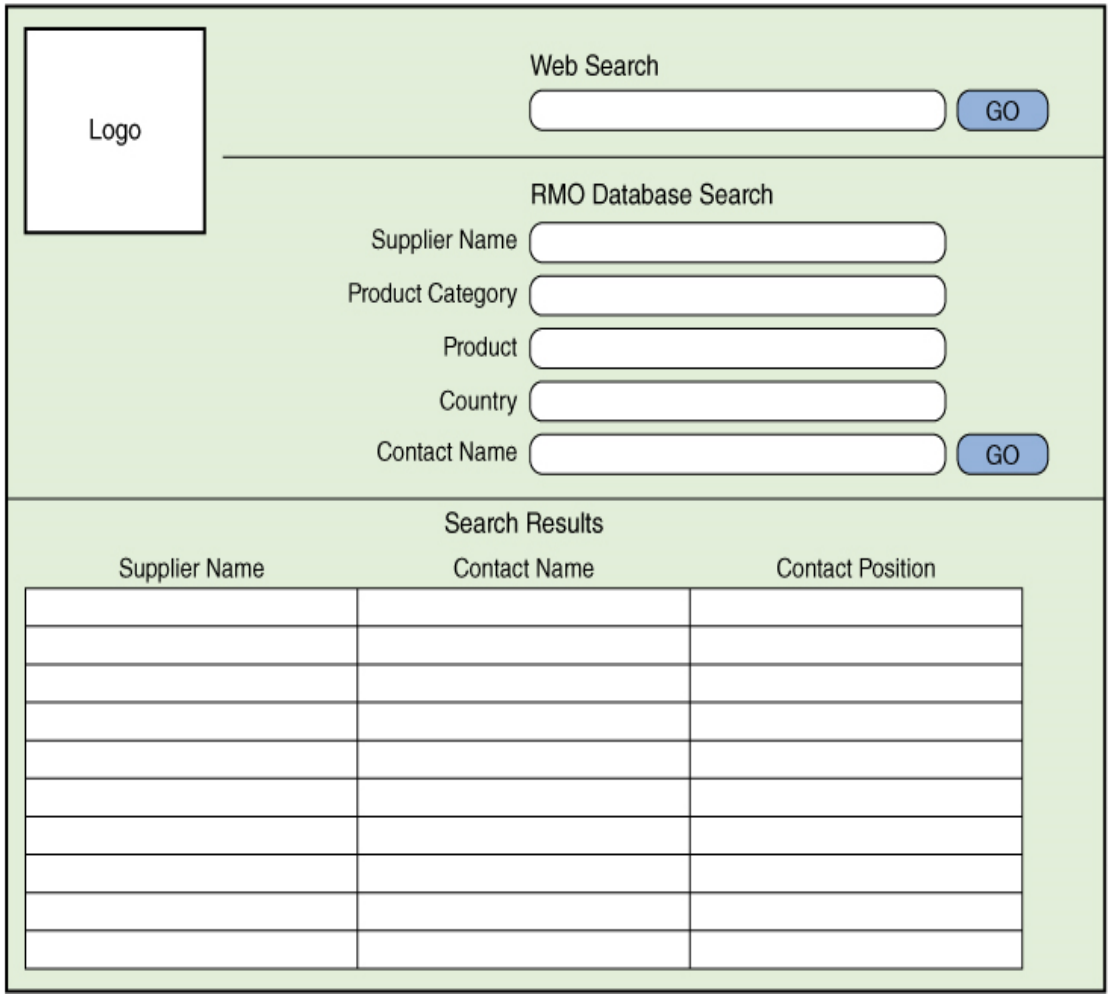


核心过程四【Core Process 4】

- Design System Components【设计系统】
  - 设计用户界面
  - 设计数据库
  - 设计系统高层结构
    - 设计软件组件
    - 定义初步的设计类图
    - 设计子系统的软件架构

Defining Screen Layout【设计界面】

- 注：Look up supplier用例的用户界面

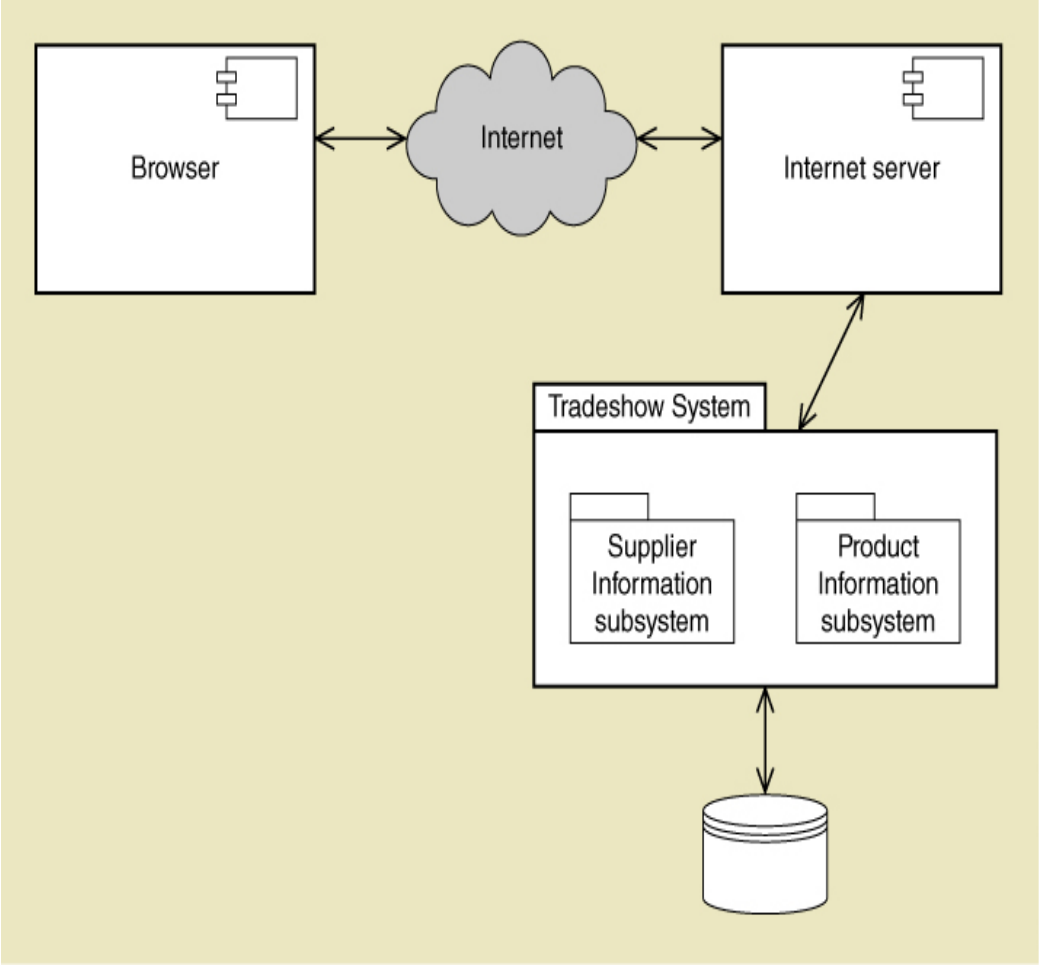


Designing the Database【设计数据库】

Table Name	Attributes
Supplier	SupplierID: integer {key} Name: string {index} Address1: string Address1: string City: string State-province: string Postal-code: string Country: string SupplierWebURL: string Comments: string
Contact	ContactID: integer {key} SupplierID: integer {foreign key} Name: string {index} Title: string WorkAddress1: string WorkAddress2: string WorkCity: string WorkState: string WorkPostal-code: string WorkCountry: string WorkPhone: string MobilePhone: string EmailAddress1: string EmailAddress2: string Comments: string

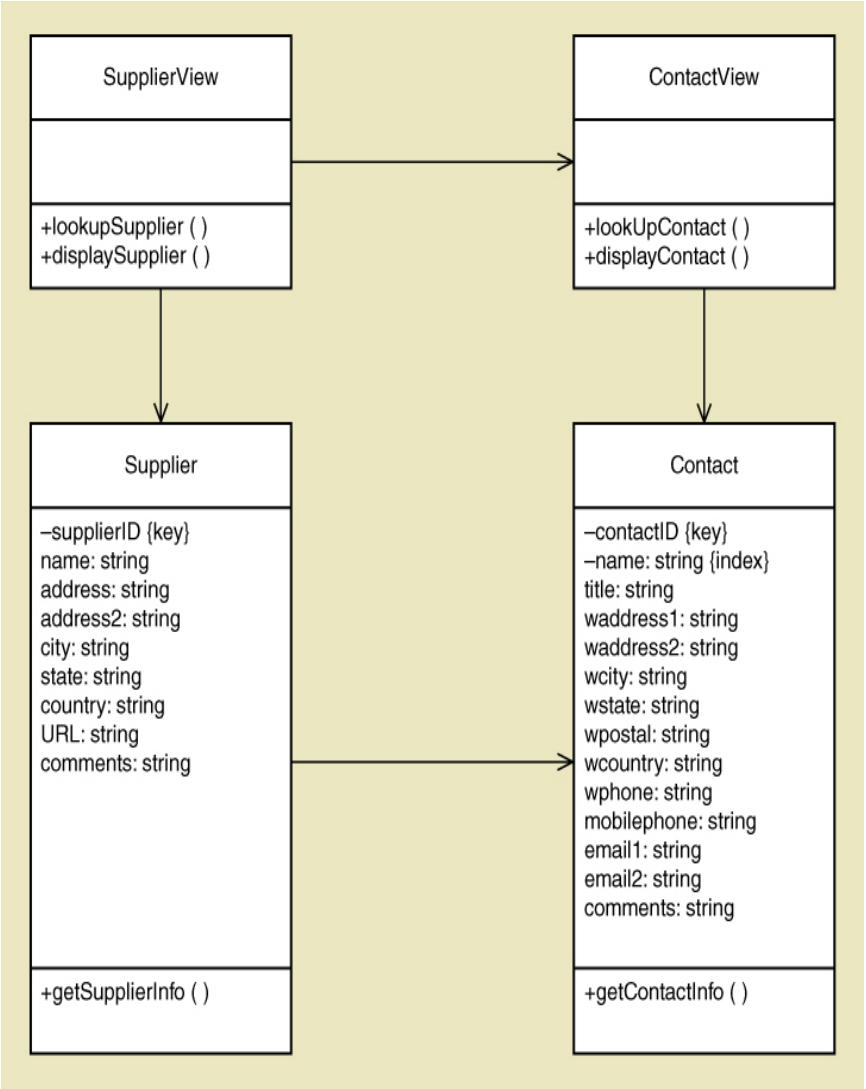
Designing the Software Components

- Software Components Diagram【软件组件图】

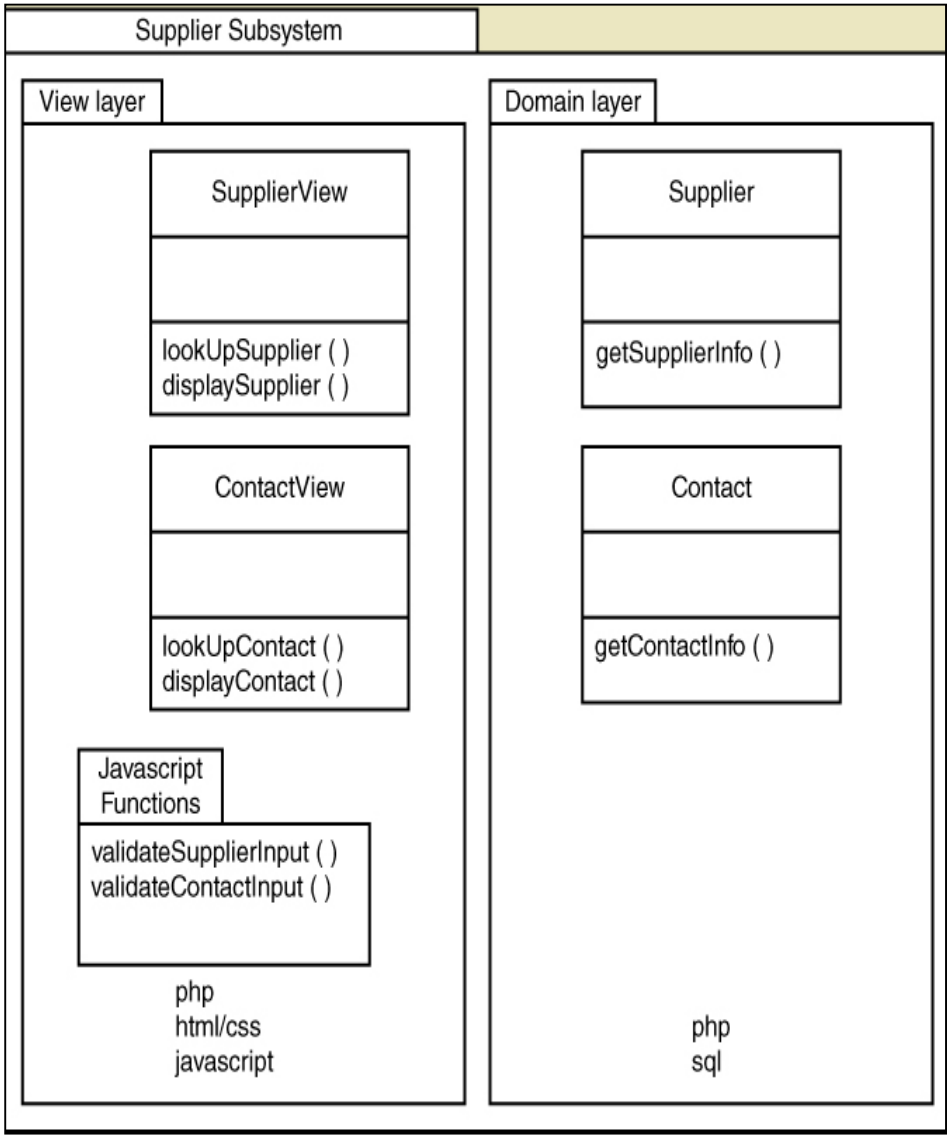


Preliminary Design Class Diagram

- 初步的设计类图



Software Architecture【体系结构】



核心过程五【Core Process 5】

- Build, Test, and Integrate System Components
  【构建、测试和集成系统组件】
  - 详细设计
  - 编码
  - 单元测试和集成测试

Code Example for One Class

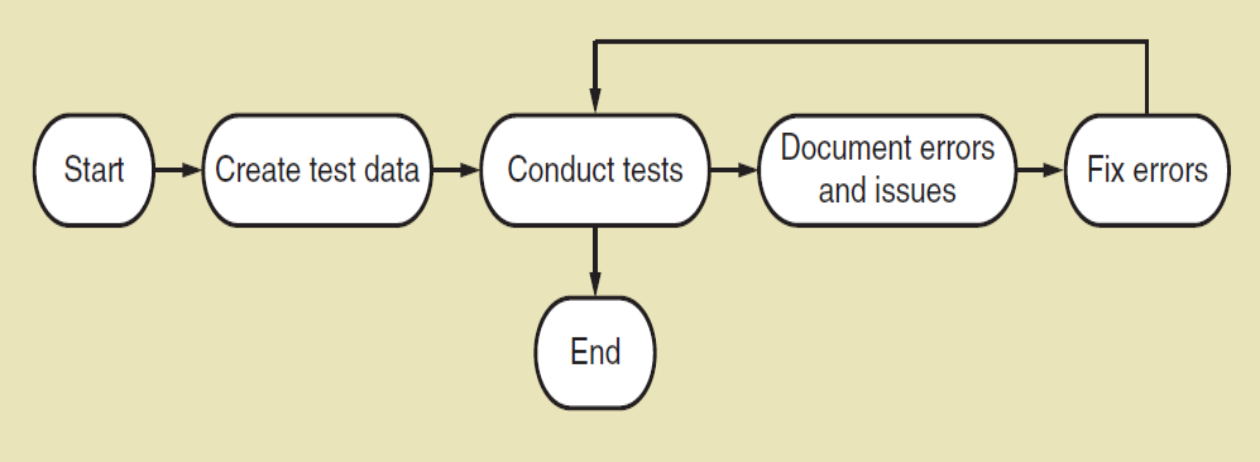
```
<?php
    class SupplierView
    {
        private Supplier $theSupplier;

        function __construct()
        {
            $this->theSupplier = new Supplier();
        }

        function lookupSupplier()
        {
          include('lookupSupplier.inc.html');
        }

        function displaySupplier()
        {
              include('displaySupplierTop.inc.html');
              extract($_REQUEST);   // get Form data
            //Call Supplier class to retrieve the data
            $results = $theSupplier->getSupplierInfo($supplier, $category,
                                              $product, $country, $contact);

           foreach ($results as $resultItem){
           ?>
                    <tr>
                        <td style="border:1px solid black">
                              <?php echo $resultItem->supplierName?></td>
                        <td style="border:1px solid black">
                               <?php echo $resultItem->contactName?></td>
                        <td style="border:1px solid black">
                           <?php echo $resultItem->contactPosition?></td>
                    </tr>
            <?php }
            include('displaySupplierFoot.inc.html');
        }
    }
?>
```

核心过程六【Core Process 6】

- Complete System Testing and Deploy the System 【系统测试及部署系统】
  - 系统功能测试
  - 用户验收测试
  - 部署系统

Flowchart for Testing Tasks

- 测试流程



软件开发的核心过程小结

- 软件开发的6个核心过程
  1. Identify the problem and obtain project approval
     识别问题并立项获得批准
  2. Plan and monitor the project
     规划和监督项目
  3. Discover and understand the details of the problem
     发现和理解问题的细节
  4. Design the system components that solve the problem
     设计系统组件解决问题
  5. Build, test, and integrate system components
     构建、测试和集成系统组件
  6. Complete system tests and then deploy the solution
     完成系统测试并部署解决方案

面向对象概念

面向对象基本概念回顾

- 基本概念
  - Object / Class（对象/类）
  - Method / Message（方法/消息）
  - Encapsulation（封装）
  - Inheritance（继承）
  - Interface / Implement（接口/实现）
  - Polymorphism（多态）
  - Composition / aggregation（组合/聚合）
  - Abstraction（抽象）

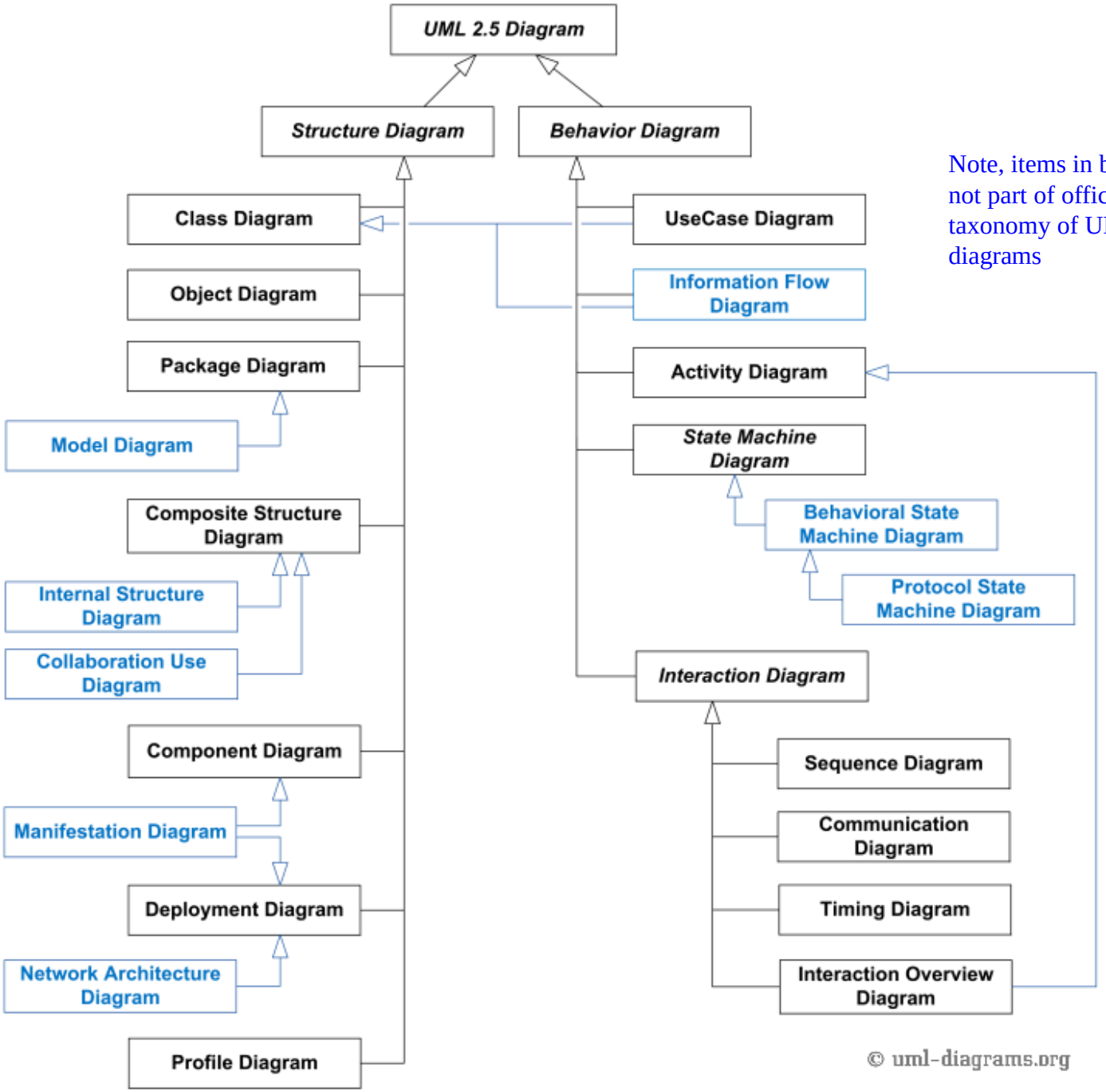
统一建模语言（UML）

UML是什么不是什么

- “统一建模语言（UML）是一种用来对软件密集型系统制品进行可视化、详述、构造和建档的图形语言，也可用于业务建模以及其它非软件系统。”
  - 是一种建模语言，不是一种建模方法
  - 用于建立系统的分析模型和设计模型，而不是用于编程
  - 是一种已被OMG采纳的建模语言规范（specification）
  - 部分地采用了形式化语言的定义方式，但并不严格

UML2模型图

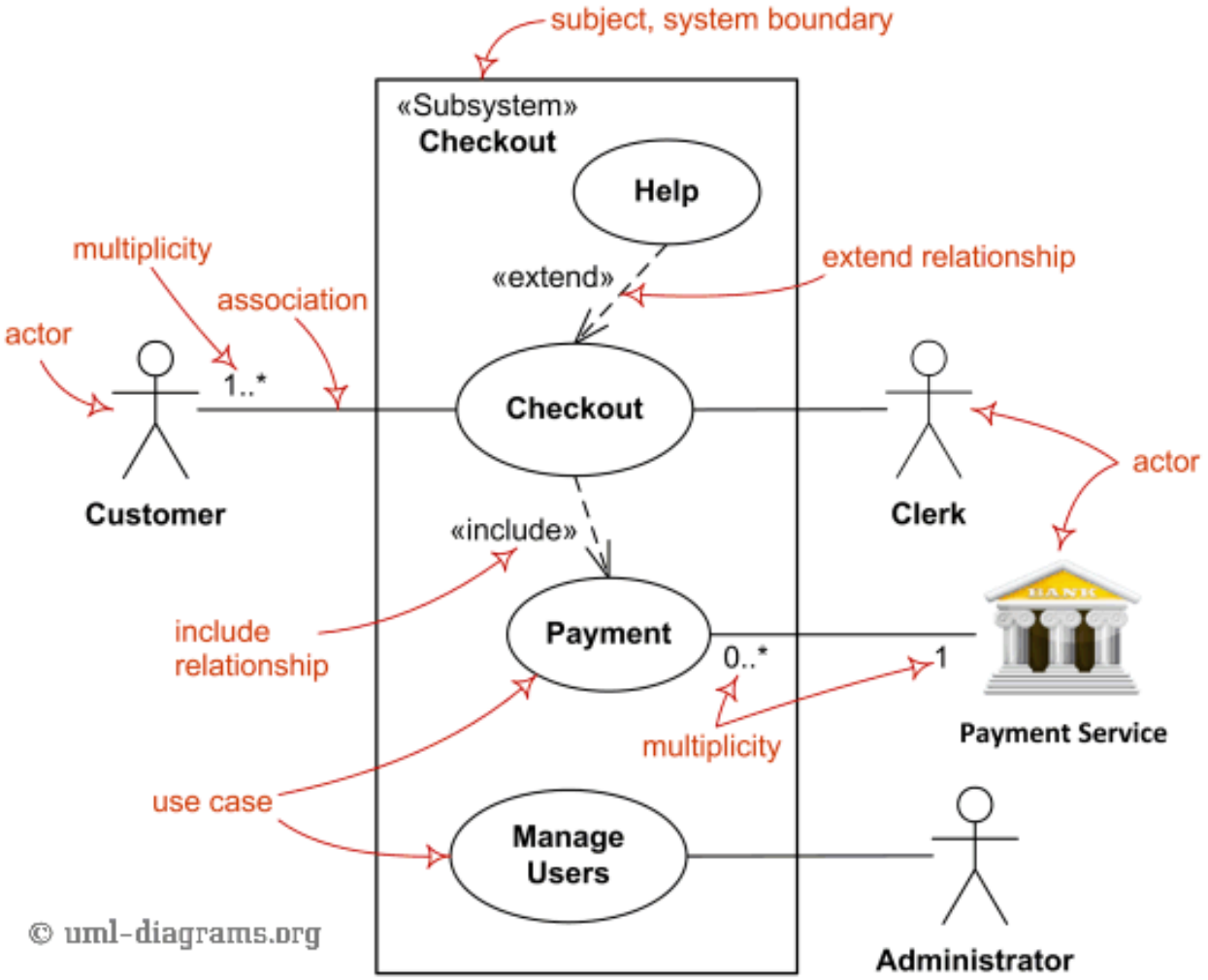
模型图	描　　述
用例图	描述系统与外部系统和用户的交互。换句话说，它们以图形化的方式描述了谁将使用系统，以及用户期望以什么方式与系统交互。用例描述也用于以文本化的方式描述每个交互步骤的顺序。
活动图	描述一个业务过程或者一个用例的活动的顺序流。它也可以用于系统的建模逻辑。
类图	描述系统的对象结构，它们显示构成系统的对象类，以及那些对象类之间的关系。
对象图	类似于类图，但并不描述对象类，它们对实际的对象实例进行建模——显示实例的属性的当前值。对象图为开发人员提供对象在某个时间点上的“快照”。
状态机图	用于建模在生命周期中事件如何改变对象的状态——对象可以经历的各种状态，以及引起对象从一个状态向另一个状态转换的事件。
组合结构图	分解类、组件或用例的内部结构。
顺序图	以图形化的方式描述了在一个用例或操作的执行过程中对象如何通过消息互相交互，说明了消息如何在对象之间被发送和接收以及发送的顺序。
通信图	在 UML 1. X 中称为协作图，描述了对象通过消息的交互。因此，它类似于顺序图。但顺序图关注消息的定时或“顺序”，通信图则以一种网络格式表现对象之间的结构化组织。
交互图	组合了顺序图和活动图的特征，显示在每个用例的活动中对象如何交互。
定时图	另一种交互图，它关注一个对象或一组对象在改变状态时的时间约束条件。当为设备设计嵌入式软件时，定时图特别有用。
组件图	描述程序代码分解成组件的组织结构，以及组件之间的交互。
部署图	描述软件组件在系统的硬件“节点”的物理体系结构中的配置。
包图	描述类或其他 UML 构件如何组织成包(对应 Java 的包或者 C++ 和 . NET 的名字空间)，以及这些包之间的依赖关系。



Note, items in blue are not part of official taxonomy of UML 2.5 diagrams

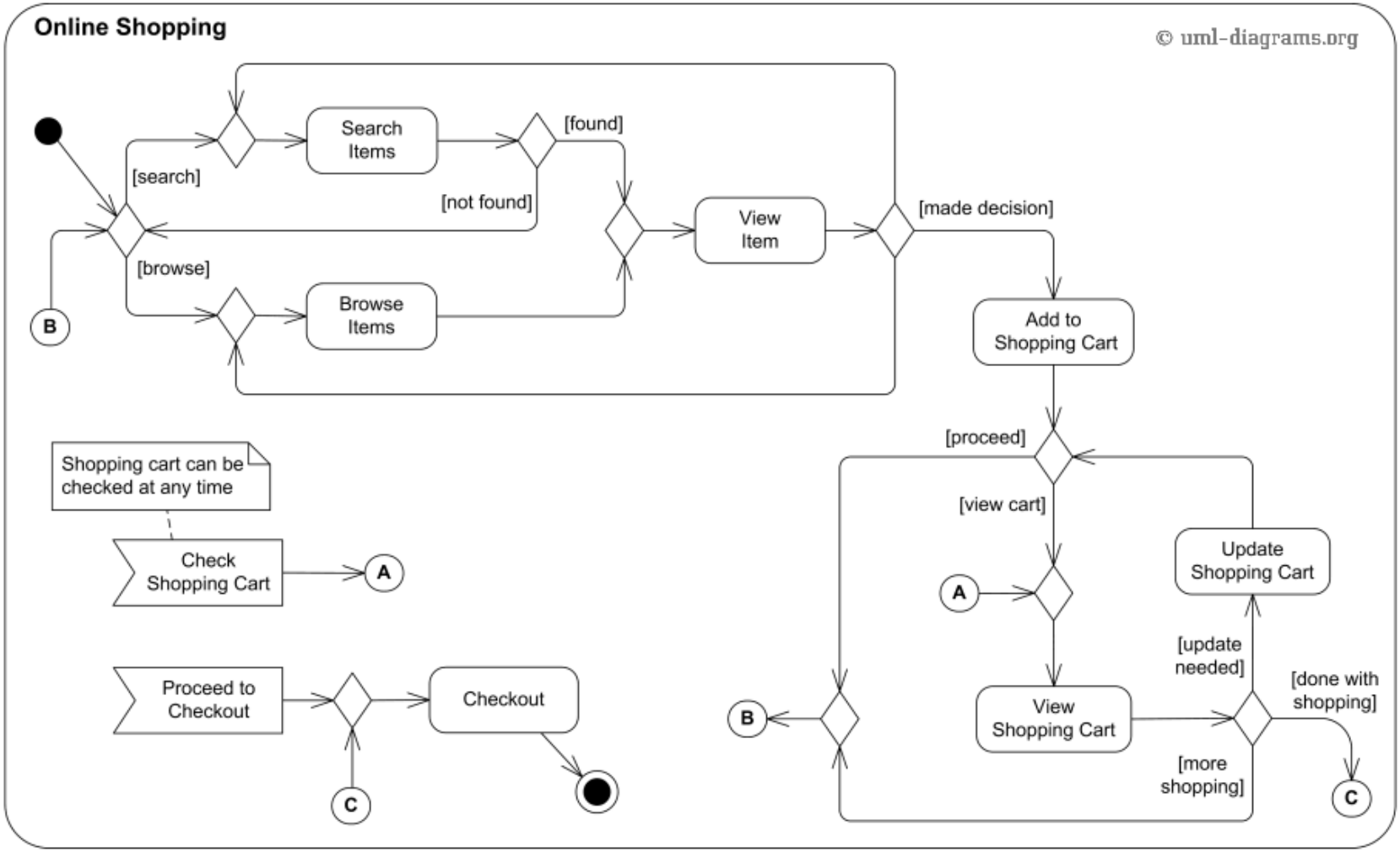
用例图【Use Case】

- 描述系统与外部系统和用户的交互



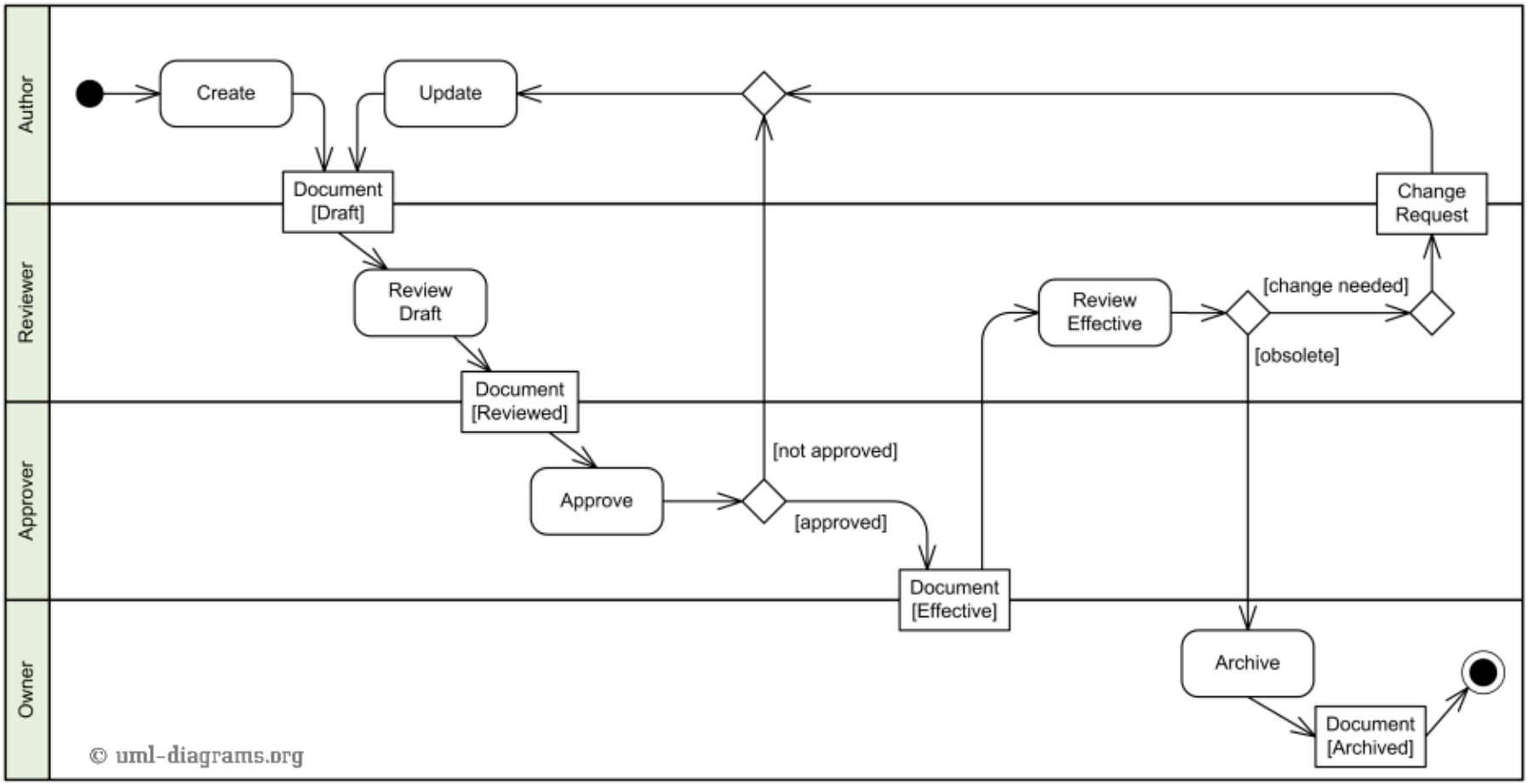
活动图【Activity】

- 描述一个业务过程或者一个用例的活动的顺序流



活动图【Activity】

- 例：带泳道的活动图（Document Management Process）

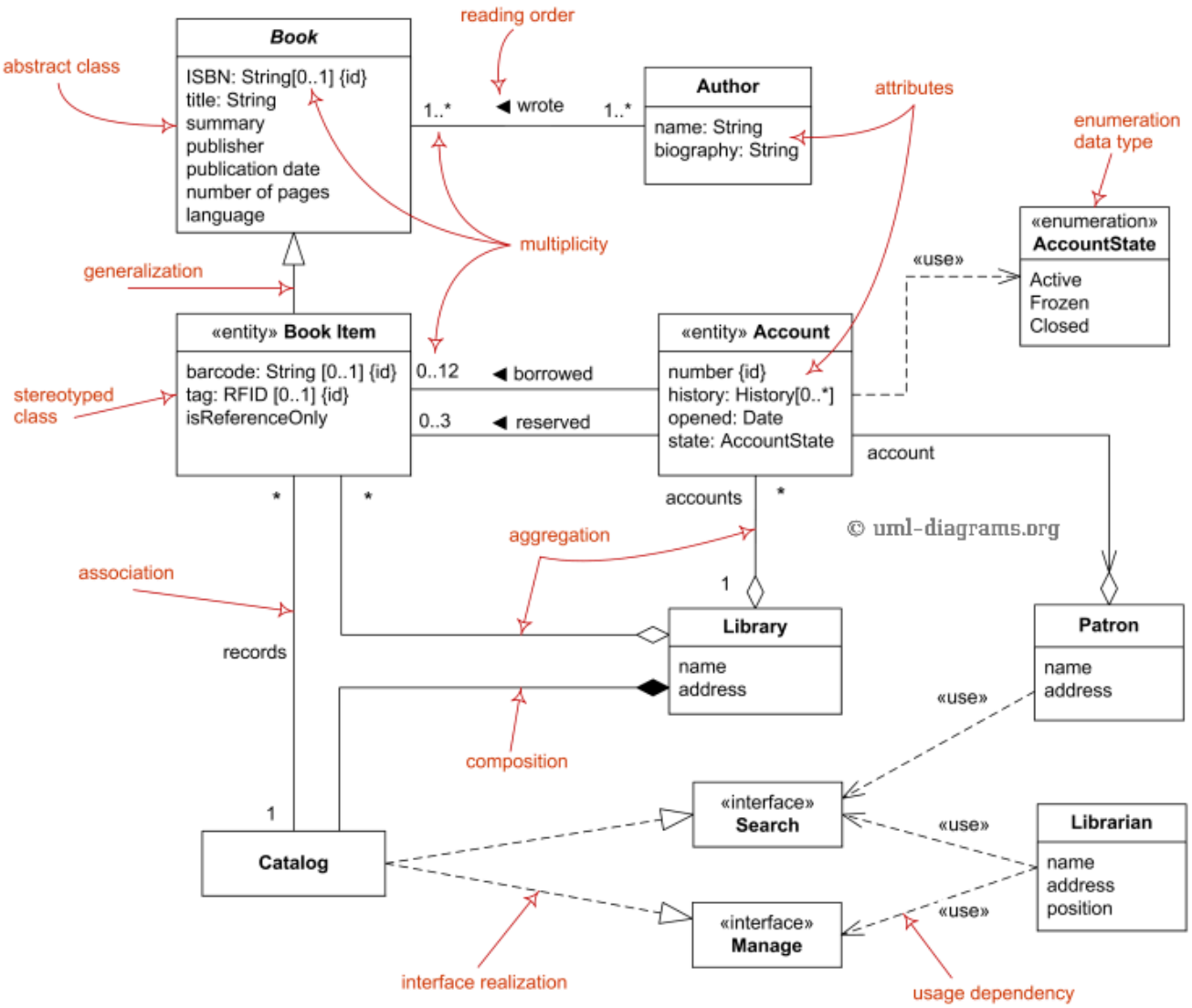


类图【Class】

- 领域类

Library domain model

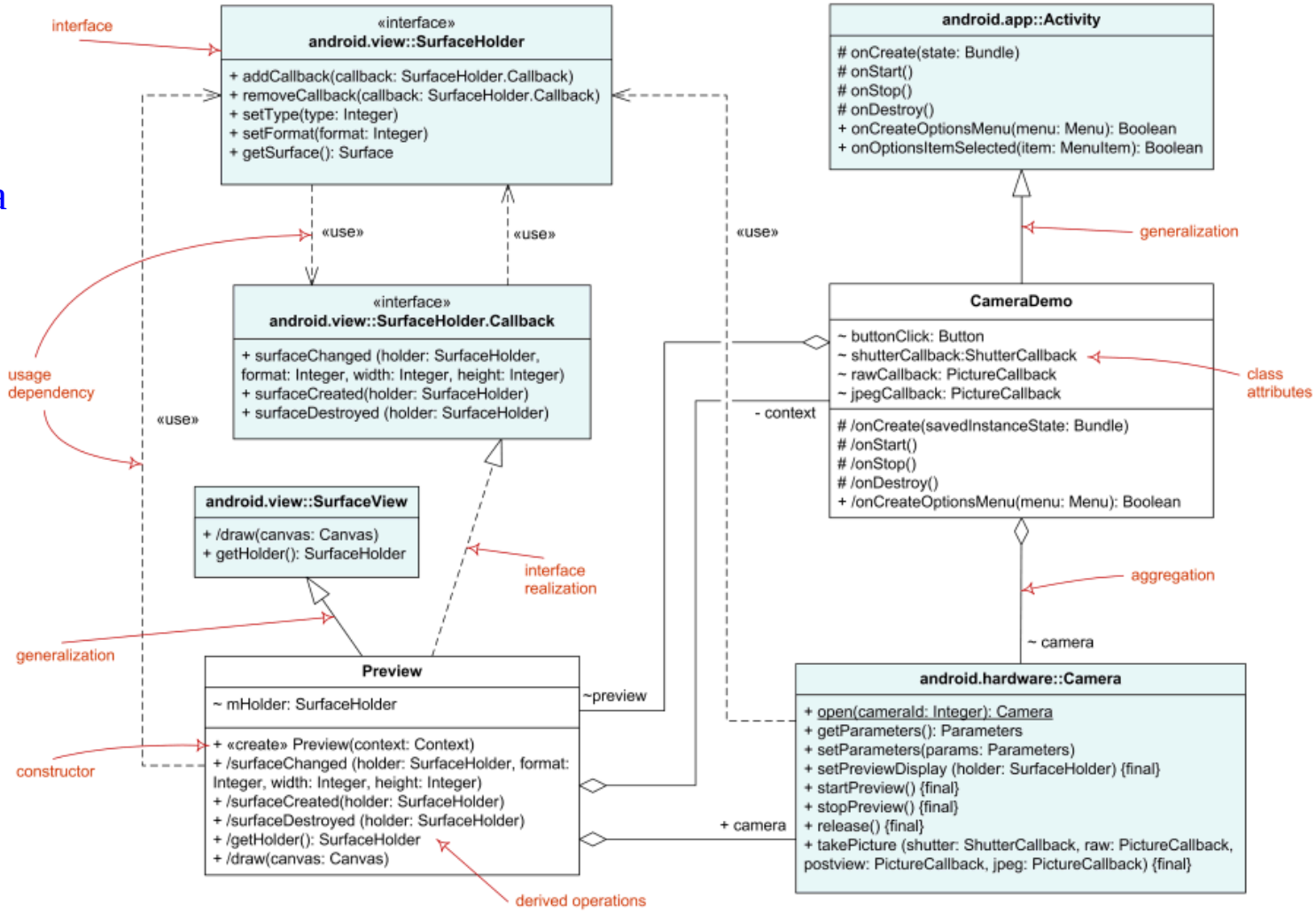
描述系统的对象结构，它们显示构成系统的对象类，以及那些对象类之间的关系



类图【Class】

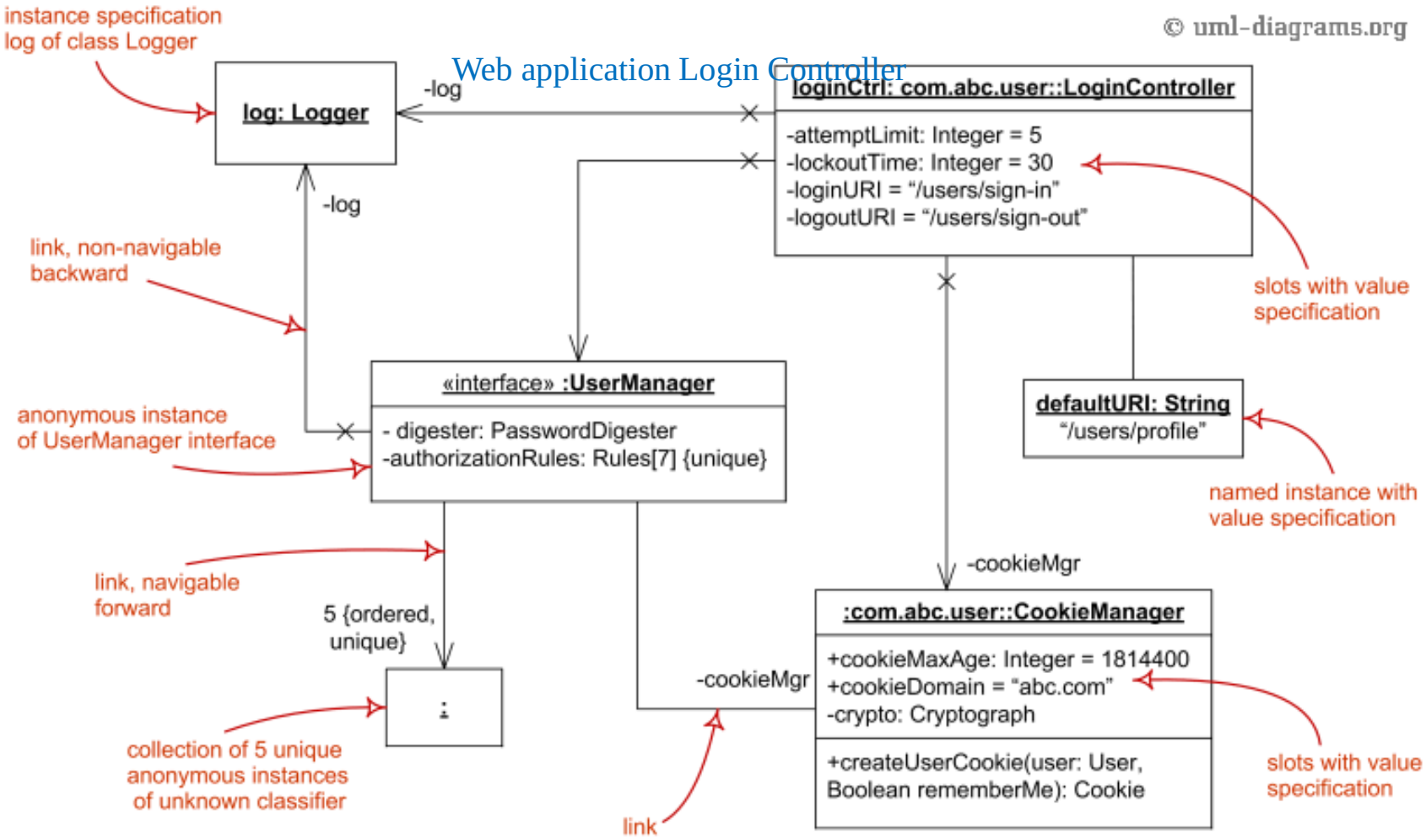
- 实现类

Android Camera implementation classes



对象图【Object】

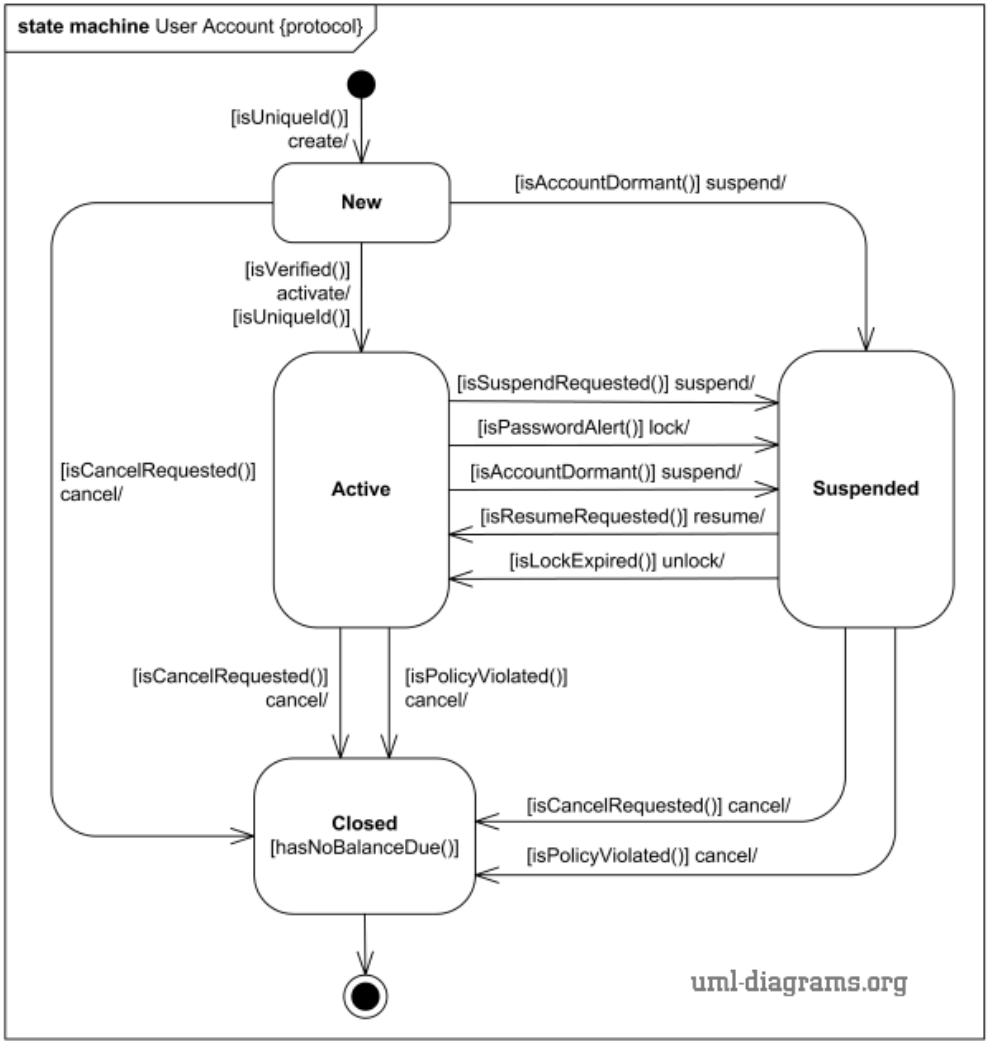
- 类似于类图，显示实例的属性的当前值



状态机图【State Machine】

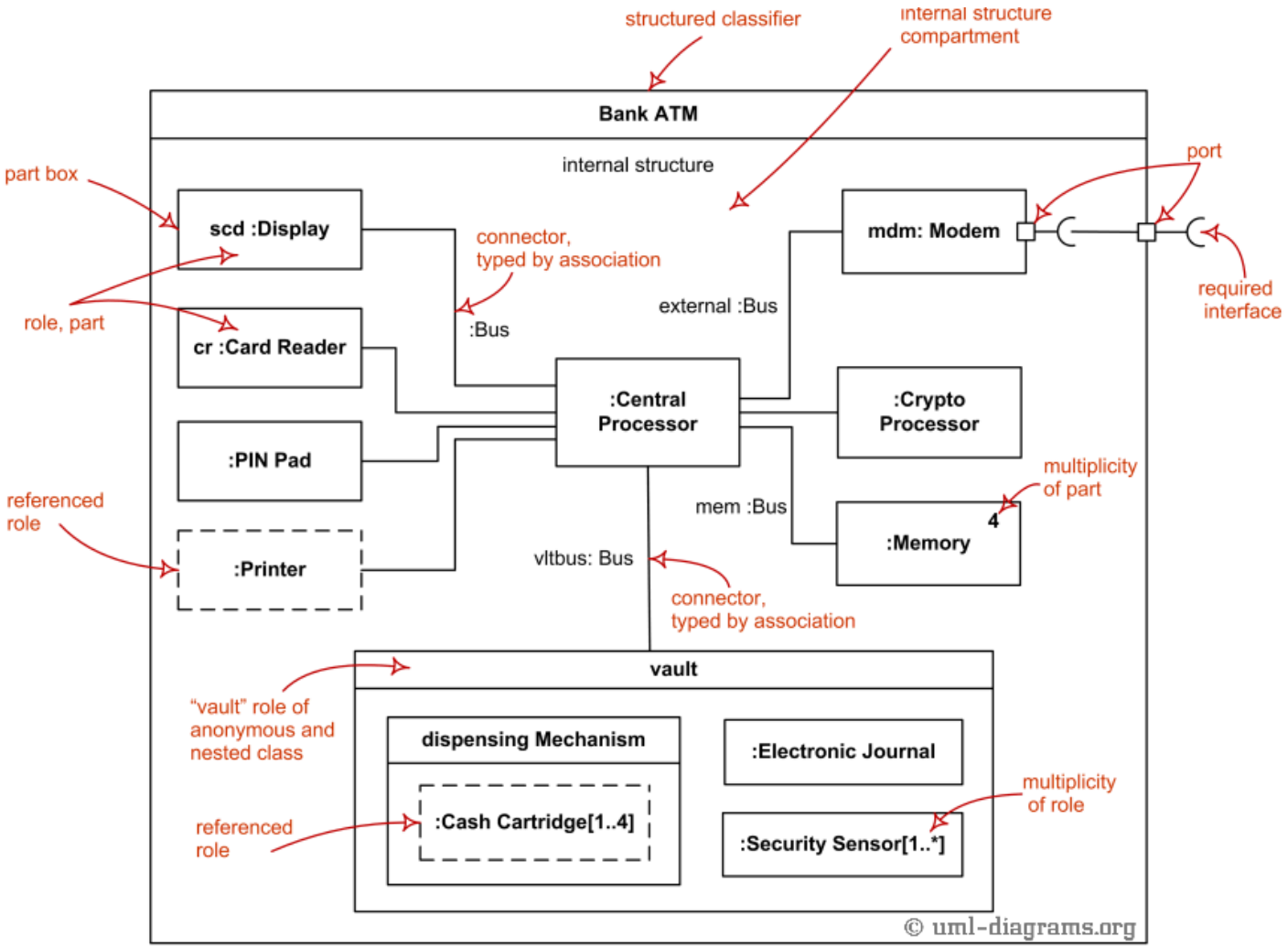
- 用于建模在生命周期中事件如何改变对象的状态

Online Shopping
- User Account



组合结构图【Composite Structure】

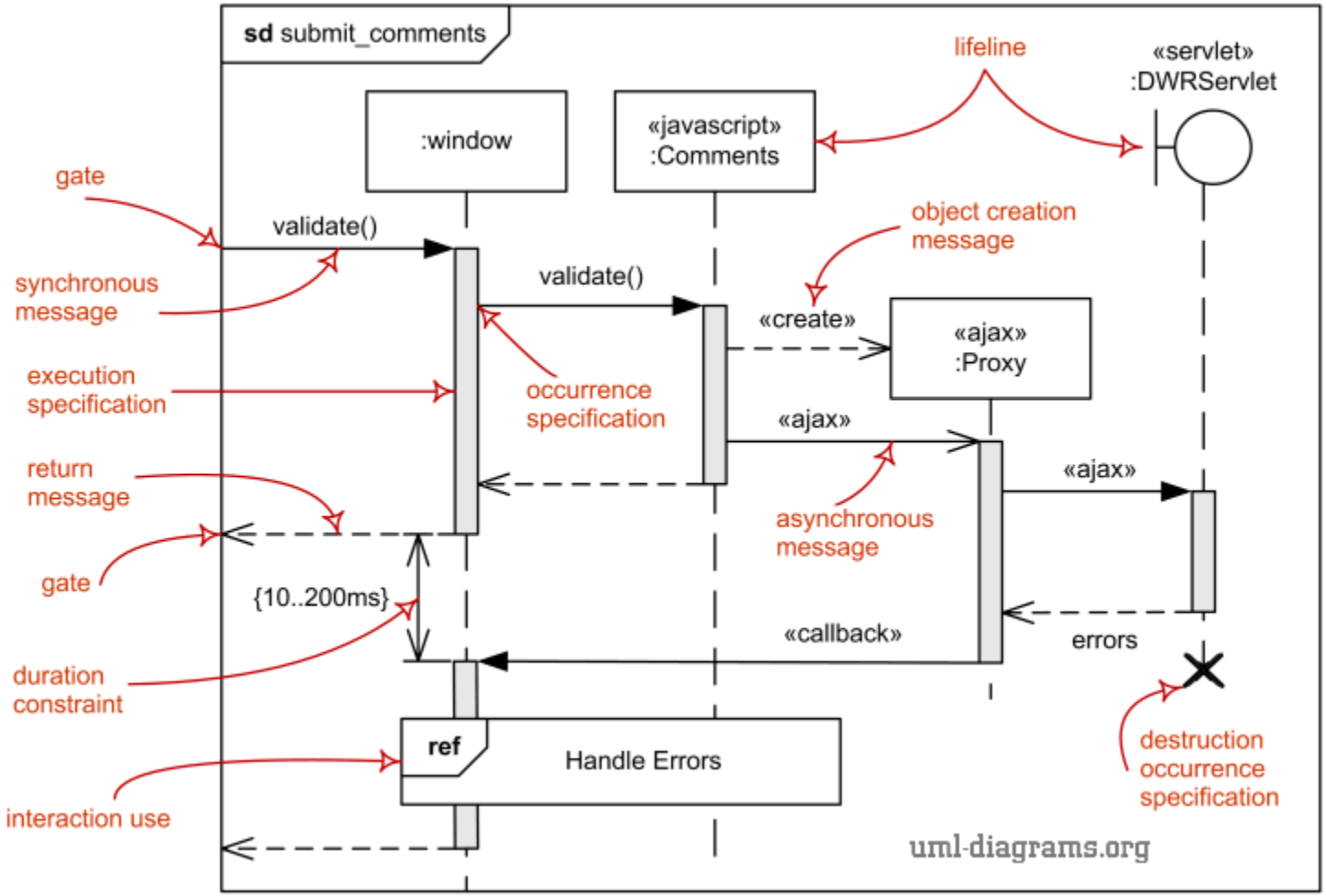
- 分解类、组件或用例的内部结构



顺序图【Sequence】

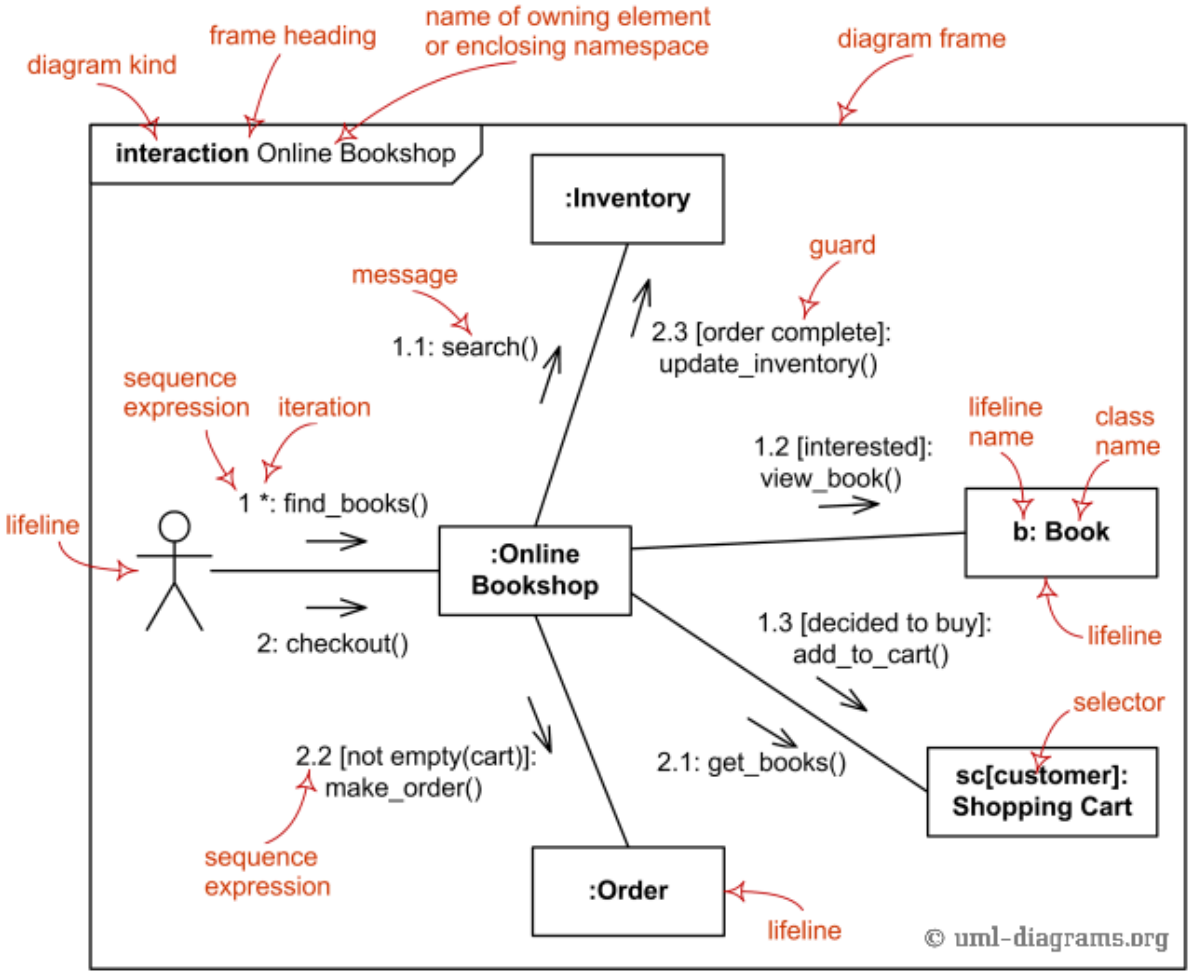
- 描述了在一个用例或操作的执行过程中对象如何通过消息互相交互

Submit Comments



通信图【Communication】

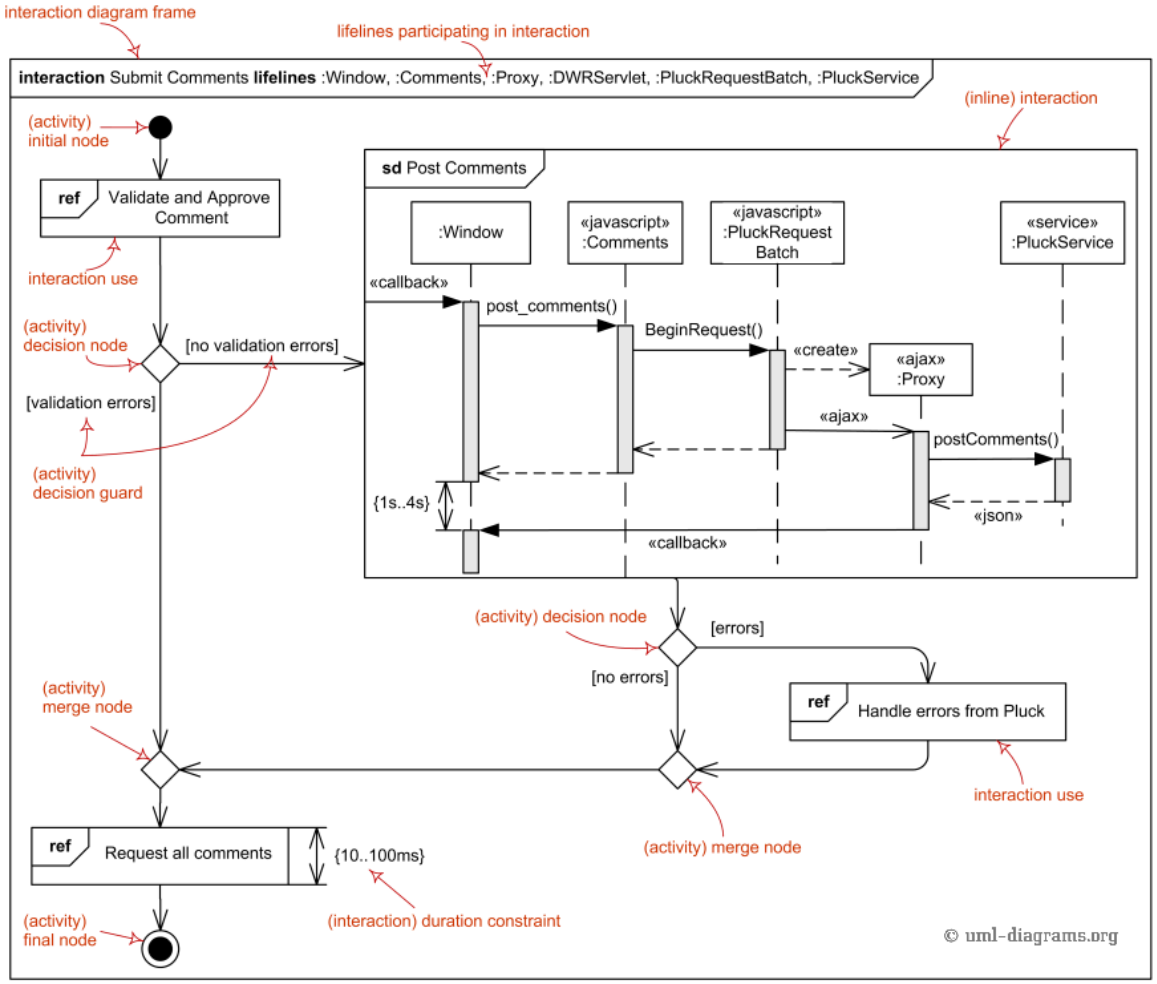
- 描述了对象通过消息的交互，以一种网络格式表现对象之间的结构化组织



交互概览图【Interaction Overview】

- 组合了顺序图和活动图的特征，显示在每个用例的活动中对象如何交互

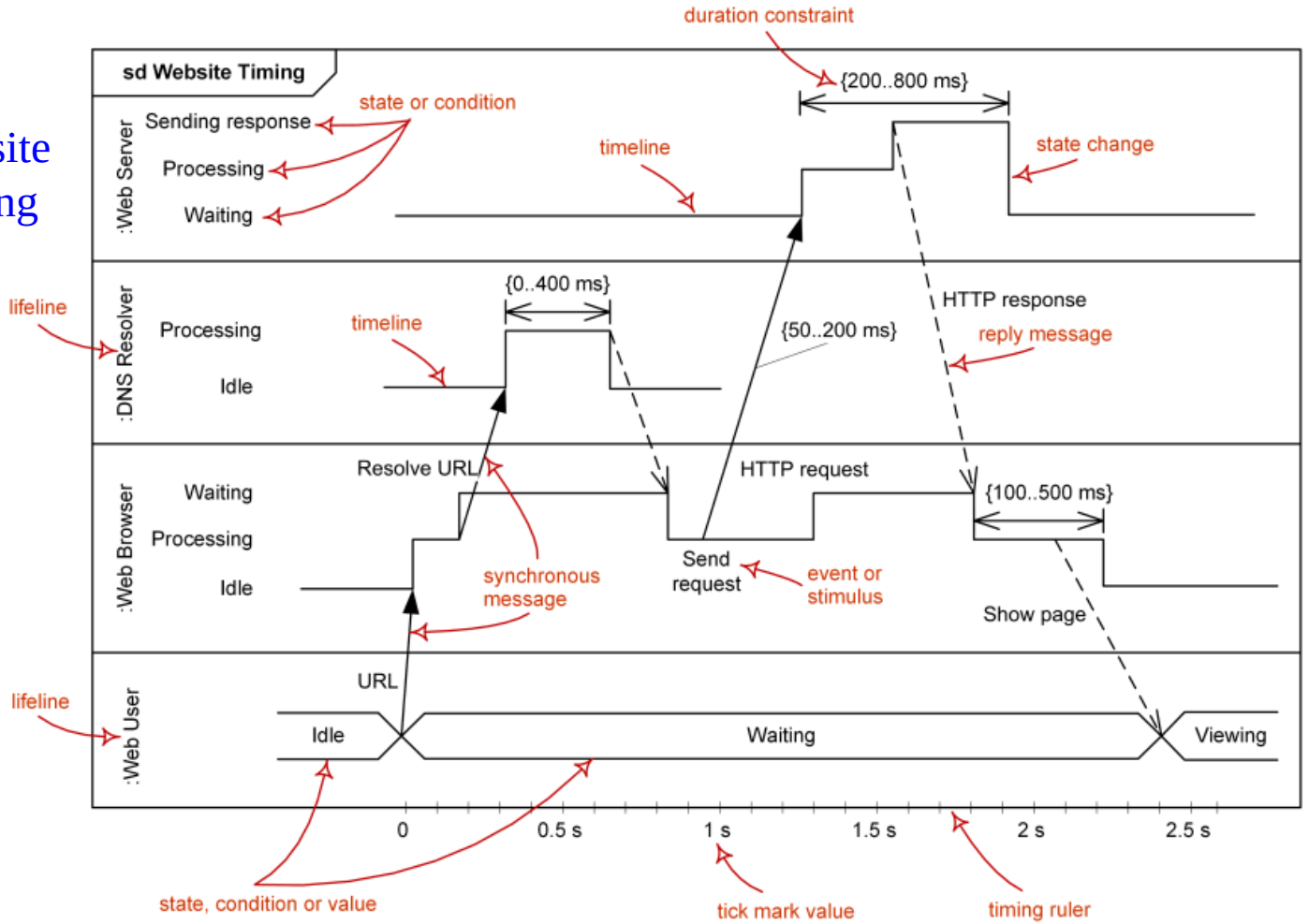
Submit Comments



定时图【Timing】

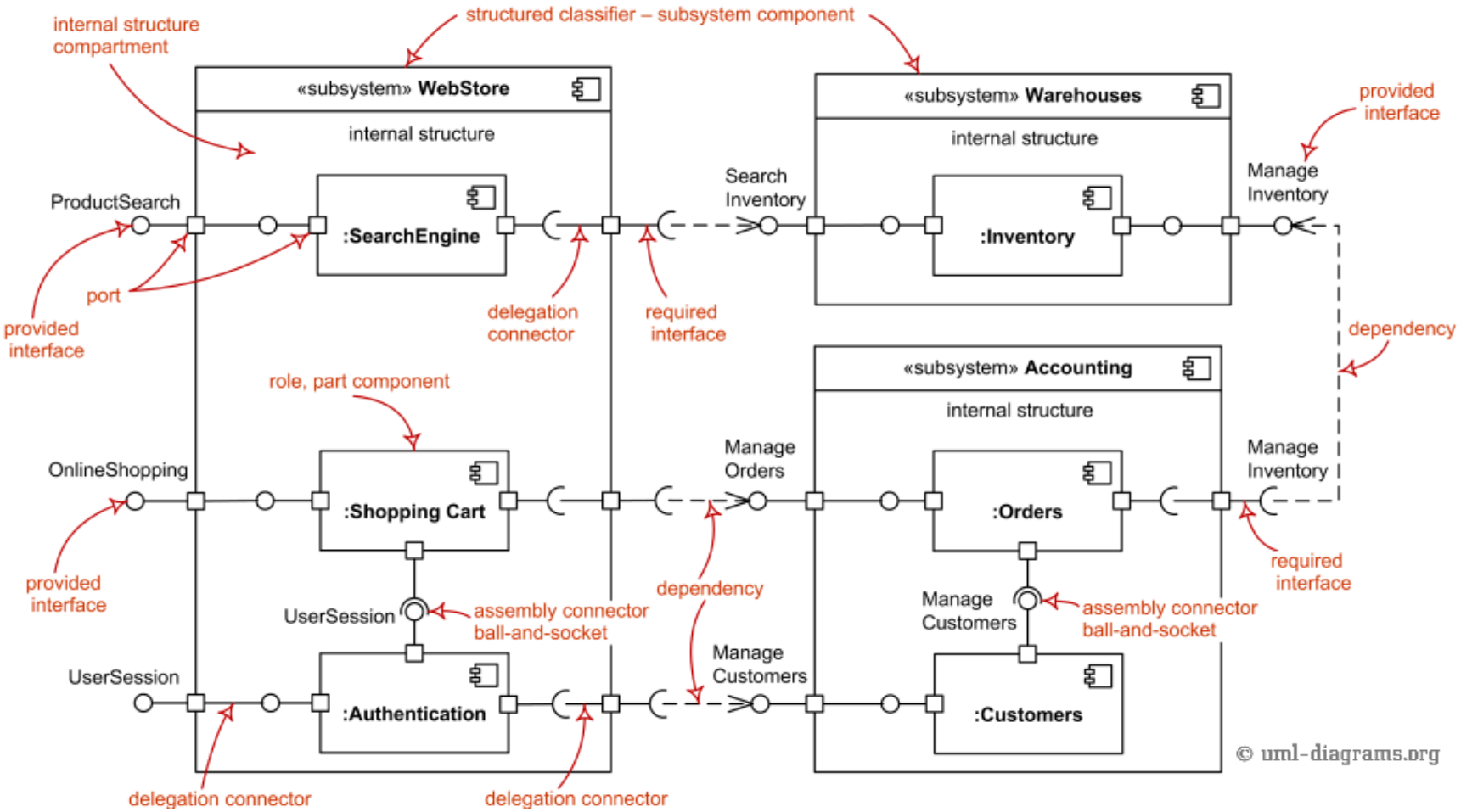
- 一种交互图，它关注一个对象或一组对象在改变状态时的时间约束条件

Website Timing



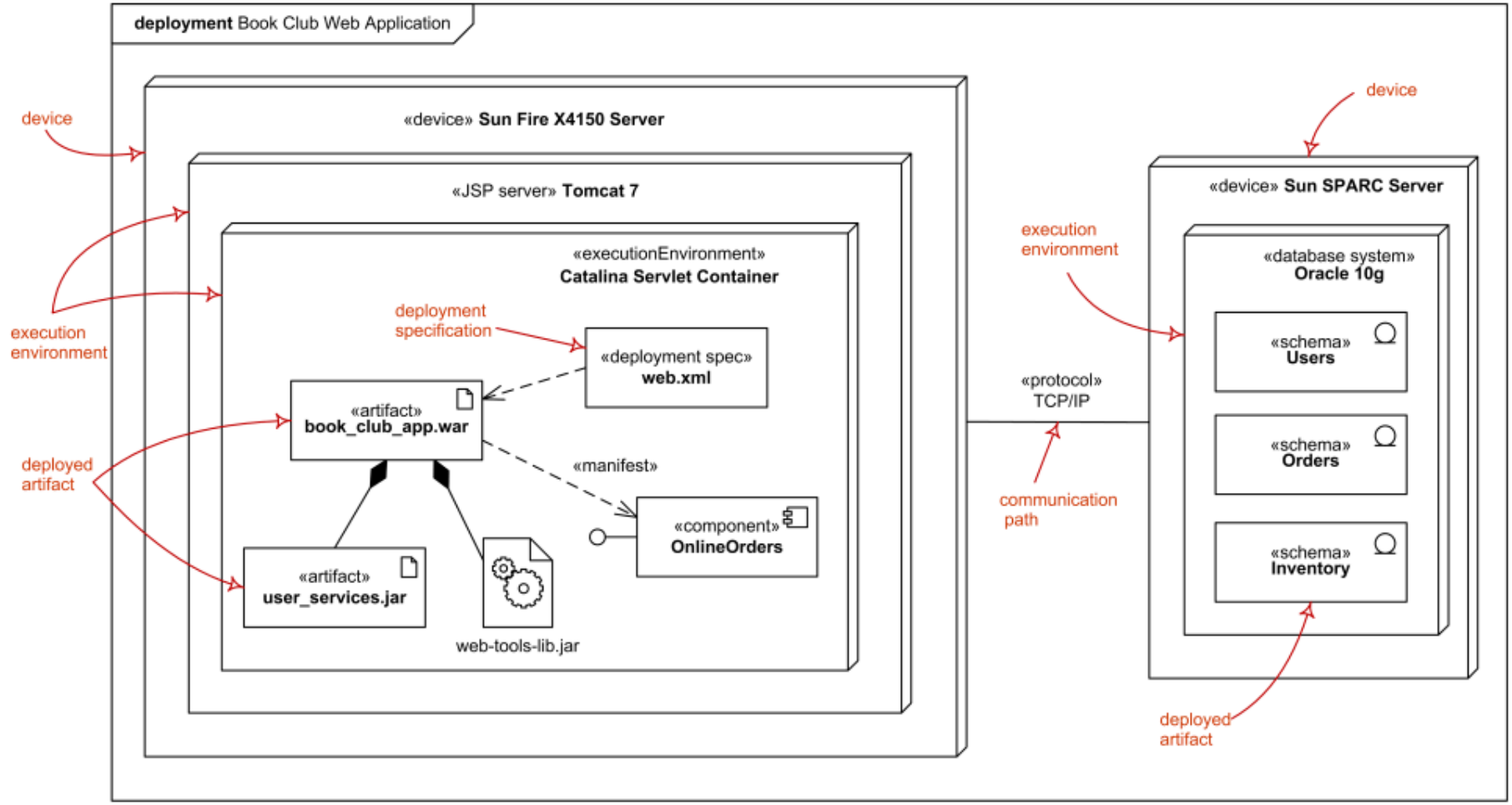
组件图【Component】

- 描述程序代码分解成组件的组织结构，以及组件之间的交互



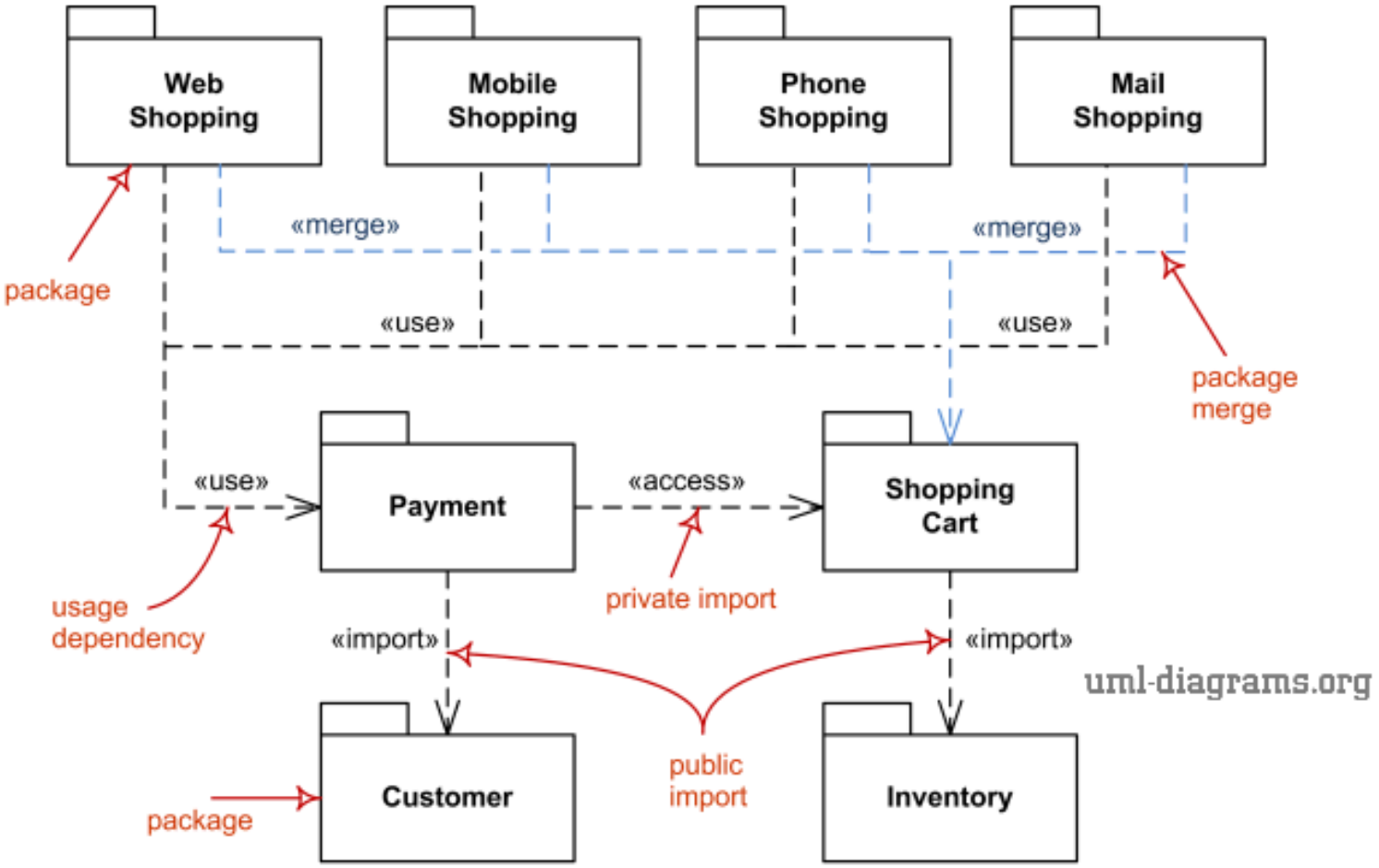
部署图【Deployment】

- 描述软件组件在系统的硬件“节点”的物理体系结构中的配置



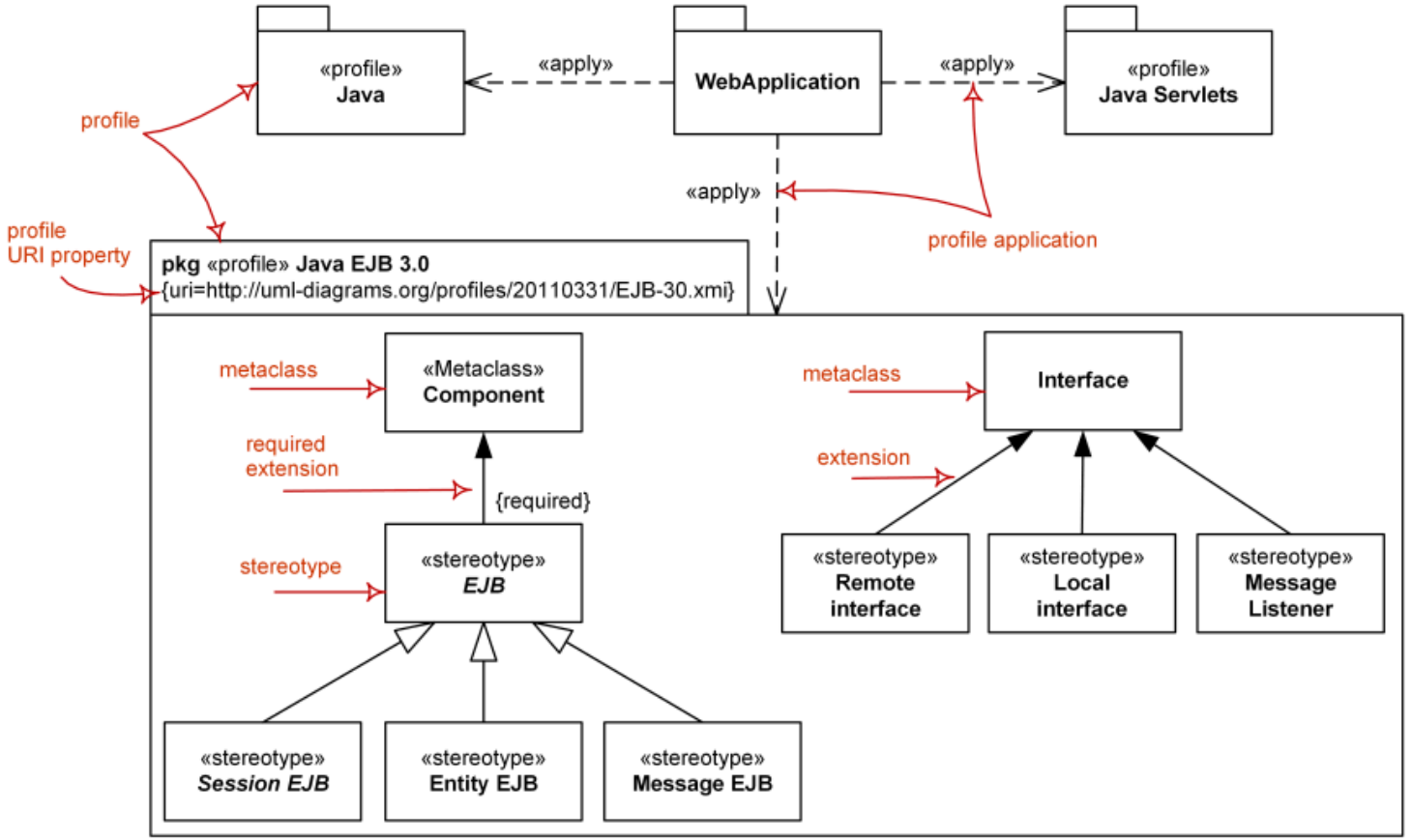
包图【Package】

- 描述类或其他UML构建如何组织成包，以及这些包之间的依赖关系



Profile图【Profile】

- Profile是一个包含了为特定的领域和目的定制的模型元素的构造型（Stereotype）包，他利用构造型、标注值，约束来扩展元模型



UML建模工具

- StarUML
- PowerDesigner
- PlantUML
  - https://plantuml.com/zh/
- ZenUML
  - https://app.zenuml.com/

Thank you!