IBM Software Group

SOA 开发第二步 – 设计SOA架构

IBM Rational　姚炳雄 yaobx@cn.ibm.com

e business on demand.

SOA 开发三步曲

第一步 业务分析员

业务目标
业务模型
业务需求

- 分析并记录现有的业务流程及其概念
- 优化改进业务流程

Rational. software
Rational Requisite Pro
Rational Software Modeler

第二步 架构设计师

服务模型
软件架构
企业架构

- 精练服务架构
- 设计服务

Rational. software
Rational Software Architect

集成开发员

服务流程模型
服务组装模型

- 实现业务流程和组合应用
- 定义服务

WebSphere software
Websphere Integration Developer

第三步 开发人员

设计模型
实现模型

- 设计服务组件
- 编码实现服务

Rational. software
Rational Application Developer

资源组合
共享资产
公共流程

Rational. software
Rational Portfolio Manager
Rational Unified Process
Rational ClearCase
Rational ClearQuest

内容

- SOA与架构设计师
- SOA in RUP
- RSA平台功能概览
- Service概要文件
- 开始SOA架构设计

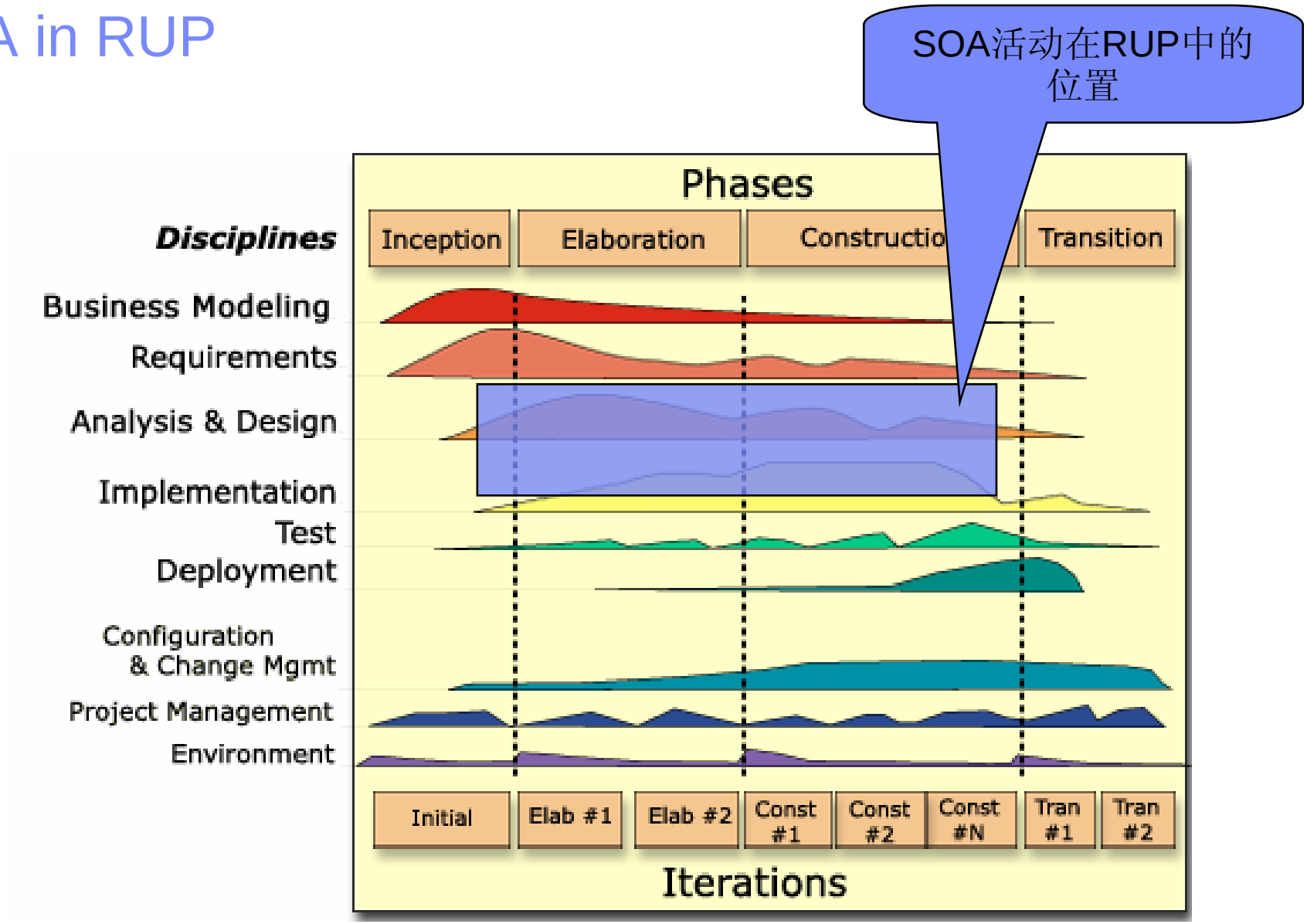
SOA与架构设计师

- SOA架构设计师能力要求
  - 良好的技术背景、经验丰富、善于快速学习……
  - 领域知识、管理知识、沟通能力、洞察力、责任心……
- SOA架构设计师职责要求
  - 充分理解商业需求和非功能性服务级别
  - 识别并降低各种技术风险
  - 可用性、安全性、可重用性、可扩展性、可管理性、可维护性、可靠性、性能、容错、美学全方位思考和决策
- SOA架构设计师任务要求
  - 体系结构整体决策
  - 构建整个系统解决方案轮廓
  - 组件、操作的建模
  - 系统组件的逻辑和物理布局设计
- SOA架构设计师要注意的关键问题
  - 着手体系结构设计时必须全面考虑所有可能的集成需求
  - 服务粒度的控制以及无状态服务的设计

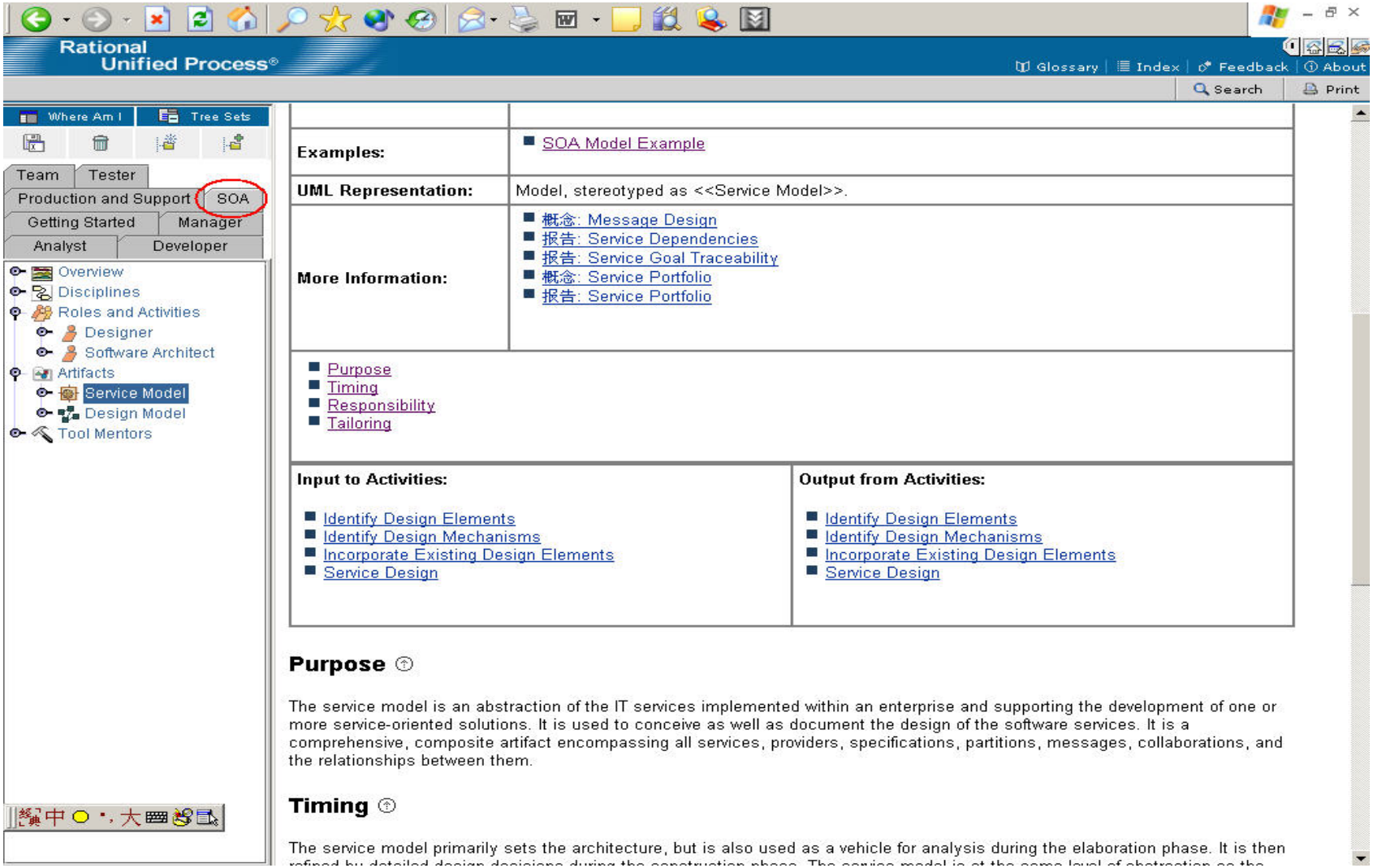
内容

- SOA与架构设计师
- SOA in RUP
- RSA平台功能概览
- Service概要文件
- 开始SOA架构设计

SOA in RUP


SOA活动在RUP中的位置
Phases
Disciplines
Inception
Elaboration
Constructio
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

SOA in RUP


Rational Unified Process®
Glossary
Index
Feedback
About
Search
Print
Where Am I
Tree Sets
Team
Tester
Production and Support
SOA
Getting Started
Manager
Analyst
Developer
Overview
Disciplines
Roles and Activities
Designer
Software Architect
Artifacts
Service Model
Design Model
Tool Mentors
Examples:
SOA Model Example
UML Representation:
Model, stereotyped as <<Service Model>>.
More Information:
概念: Message Design
报告: Service Dependencies
报告: Service Goal Traceability
概念: Service Portfolio
报告: Service Portfolio
Purpose
Timing
Responsibility
Tailoring
Input to Activities:
Identify Design Elements
Identify Design Mechanisms
Incorporate Existing Design Elements
Service Design
Output from Activities:
Identify Design Elements
Identify Design Mechanisms
Incorporate Existing Design Elements
Service Design
Purpose
The service model is an abstraction of the IT services implemented within an enterprise and supporting the development of one or more service-oriented solutions. It is used to conceive as well as document the design of the software services. It is a comprehensive, composite artifact encompassing all services, providers, specifications, partitions, messages, collaborations, and the relationships between them.
Timing
The service model primarily sets the architecture, but is also used as a vehicle for analysis during the elaboration phase. It is then

内容

- SOA与架构设计师
- SOA in RUP
- RSA平台功能概览
- Service概要文件
- 开始SOA架构设计

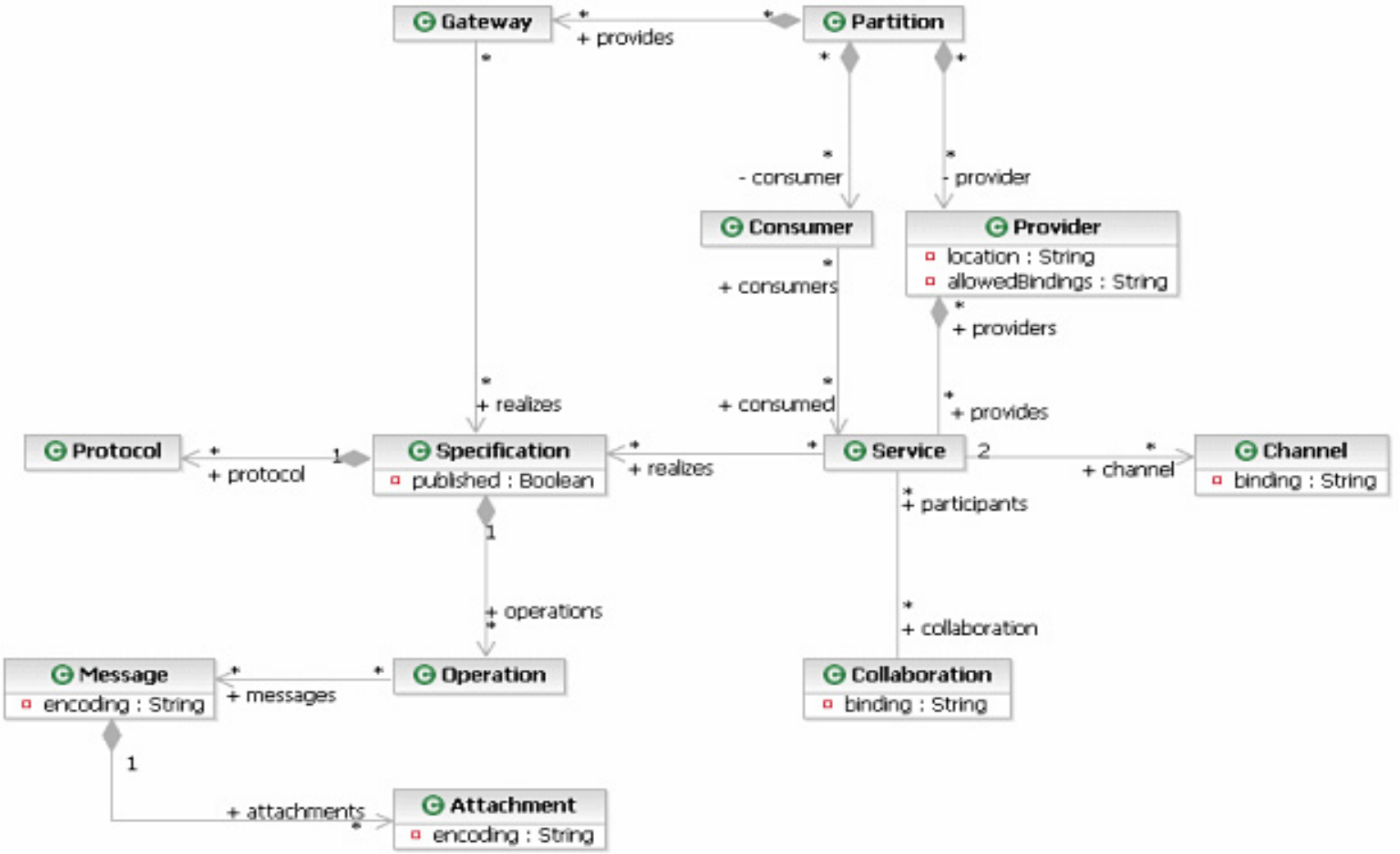
内容

- SOA与架构设计师
- SOA in RUP
- RSA平台功能概览
- Service概要文件
- 开始SOA架构设计

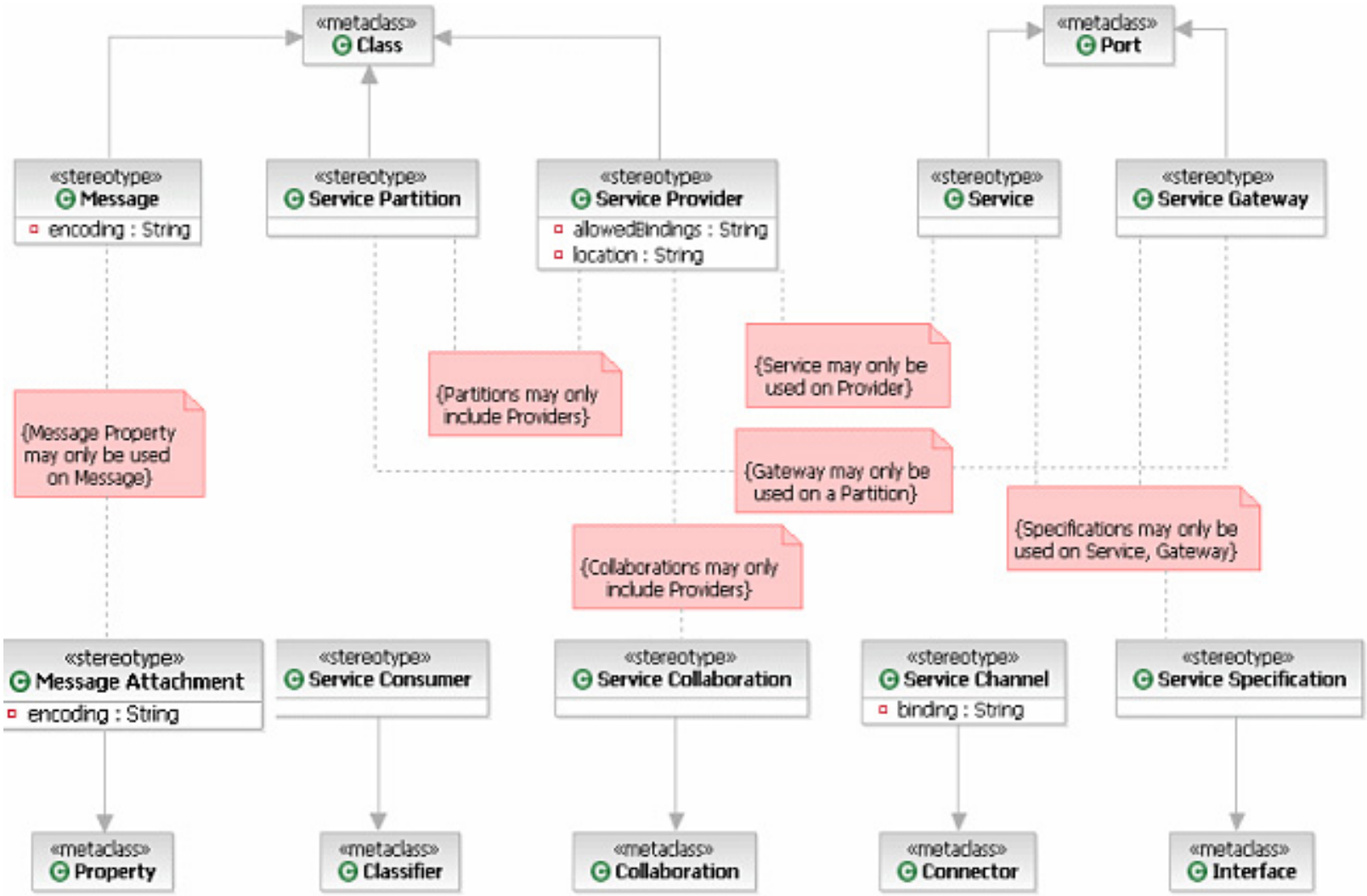
Software Services的UML概要文件(Profile)

- 一个**RSA Plug-In**，目的是为描述**Service**提供一个共同语言
  - 用于为服务(service)、面向服务的架构(SOA)和面向服务的解决方案(service-oriented solutions)建模
  - 适合顾客复杂场景(scenario)建模，帮助人们解决在开发面向服务的解决方案过程中关心的问题
  - 包括了在开发生命周期内的很多活动并且为不同的涉众提供了视图
  - 包括下列文件：
    - SoftwareServices.epx – Software Services的UML概要文件的具体实现
    - Service Design Model.emx – 一个模板，用于创建新模型
    - Retail Example Service Model.emx – 一个例子，演示概要文件的用法
- 下载网址
  - http://www-128.ibm.com/developerworks/rational/library/05/510_svc/
- 安装
  - 需将RSA升级至6.0.0.1以上
  - 用IBM Rational Product Updater 工具安装

Software Services的UML概要文件(Profile)



Software Services的UML概要文件(Profile)



Service

继承自	端口（Port）
语义	●Service模型元素提供Service交互作用的一个端点，而这些交互作用的定义是Service Specification的一部分
属性	无
约束	●应该只在原型为<<Service Provider>>的类中使用 ●应该被作为原型为<<Service Specification>>的接口

Service Specification

继承自	接口（Interface）
语义	●对接口的使用表示服务提供的一个操作集合，一个Service可能实现多个接口 ●描述和定义Service实例的结构和行为，可以看成是控制服务存取或者使用的一组规则 ●可以将一个协议状态机或者UML2.0 Collaboration附加到Service Specification上来表示Service Specification上操作调用的顺序 ●Service Specification只能提供公开的操作，而且每个操作应当只能消耗至多一条消息、产生至多一条消息
属性	●Published：Boolean 表示Service是否被发布到一个Service存储库中，这和UML中提供的公有/私有的属性是不同的概念
约束	●所有的操作都应当被标记为公有的 ●只能用于原型为 <<Service>> 或 <<Service Gateway>>端口的类型

Service Partition

继承自	类（Class）
语义	● 一个Service Partition代表系统的某个逻辑或者物理的边界 ● 建模Service Partition是可选的但是有用的 ● 一个Service Partition可能只具有表示嵌套部分的属性，成为Service或者其他的Service Partition ● 一个Service Partition的符号也比较严格，如果一个Service Partition表示它和其他所有Service Partition的全部通信必须直接通过指定的Gateway，那么它被称为是一个严格的(Strict) Service Partition。
属性	无
约束	● 不应当有任何自己的属性 ● 不应当有任何自己的操作 ● 不应当有任何自己的行为 ● 任何自己的部分应当是类原型<<Service Provider>>

Service Collaboration

继承自	协作（Collaboration）
语义	●Service Collaboration代表Service之间通信，这些Service通常被封装成一个新的Service，新的Service的实现是一组现存Service的简单的协作 ●Service Collaboration是一种指定一个实现的Service作为一个其他Service的协作的方式 ●在定义Service Specification阶段，通过Service Collaboration理解Service所承担的职责；在Design Service阶段，通过Service Collaboration表达Service在整个业务流程中是如何互相协作的
属性	无
约束	●Service Collaboration参与者应当只应该是原型为<<Service Provider >>的类

Service Provider

继承自	类（Class）
语义	●Service Provider是一个提供一个或者多个Service的元素 ●一个Service Provider拥有一个属性能够捕捉他的位置信息，但它的含义随具体的实现而定 ●Service Provider可能不直接暴露任何属性和操作 ●只有公开的端口可能被提供(原型为Service)，Service Specification中会有所记录
属性	●Location：String Service Provider的位置，它可以被用作创建端点的名字。 ●AllowedBindings：String 表示允许的平台绑定机制，一个Service Channel可能被用在连接Service中
约束	●不应当有任何自己的操作 ●任何自己的部分应当是原型为 <<Service>>的一个类

Service Consumer

继承自	分类器（Classifier）
语义	●任何分类器(类、组件等等)可能既充当Service的Consumer，也包括其他的Service
属性	无
约束	无

Service Channel

继承自	连接器（Connector）
语义	●一个Service Channel表示两个Service之间的通讯路径 ●交互作用可能出现在一个Service Channel，但Service Channel并不表示任何特殊的交互作用 ●在Web Service世界里，每个Service表示与之相联系的绑定从而一个客户可以访问它
属性	●Binding：String 表示在 Web 服务描述语言（WSDL）规范中产生服务绑定的平台绑定机制，例如 SOAP-RPC、SOAP-Doc、HTTP-Get等等
约束	●连接器至少一端应当连接到原型为<<Service >>的端口 ●连接器至多一端能够连接到原型为<<Service Gateway>>的端口，一个原型为<<Service Provider>>的类或者<<Service Consumer>>的分类器

Service Gateway

继承自	端口（Port）
语义	●Service Gateway看起来像一个Service，但它却只是对Service Partition可用的，而对Service Provider是不可用的 ●一个Service Gateway充当一个代理服务，可以使用它协调协议或者表示一个Service Partition上的可以利用的接口 ●如果只有某些Service对于Service Partition之外是可用的，则使用Service Gateway来提供这些服务
属性	无
约束	●应当只用于原型为<<Service Partition>>的类 ●应该被作为原型为<<Service Specification>>的接口

Message

继承自	类（Class）
语义	●Message是实际数据对Service和Consumer有意义一个容器 ●Message应采取值传递方式，其中不应定义任何行为，但是它可能有属性和与其他类的关联(很可能是某个领域模型)
属性	●Encoding：String 代表了用于产生消息模式的平台编码机制，如SOAP-literal、SOAP-rpc、ASN.1等等
约束	●不应有任何自己的操作 ●不应有任何自己的行为

Message Attachment

继承自	属性（Property）
语义	●表示Message的某些组件是该Message的一个附件(相对于消息的直接部分本身) ●在高级的设计活动中一般不太可能经常使用这个原型，但是对于许多过程，区分Attachment数据和内嵌的Message数据还是很重要的
属性	●Encoding：String 代表了用于产生消息模式的平台编码机制，如SOAP-RPC、Doc-Literal、ASN.11等等
约束	●只能用于属于原型为<<Message>>的类的属性

内容

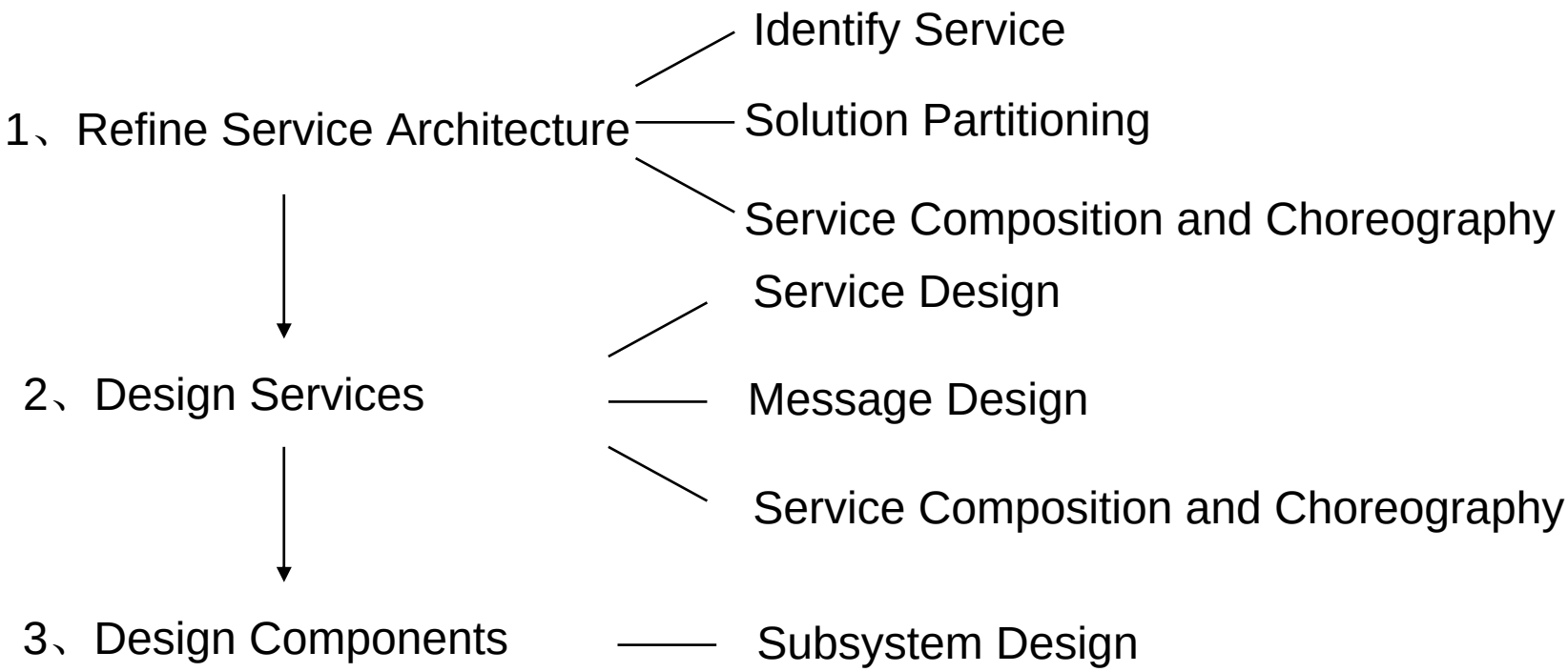
- SOA与架构设计师
- SOA in RUP
- RSA平台功能概览
- Service概要文件
- 开始SOA架构设计

开始SOA 架构设计

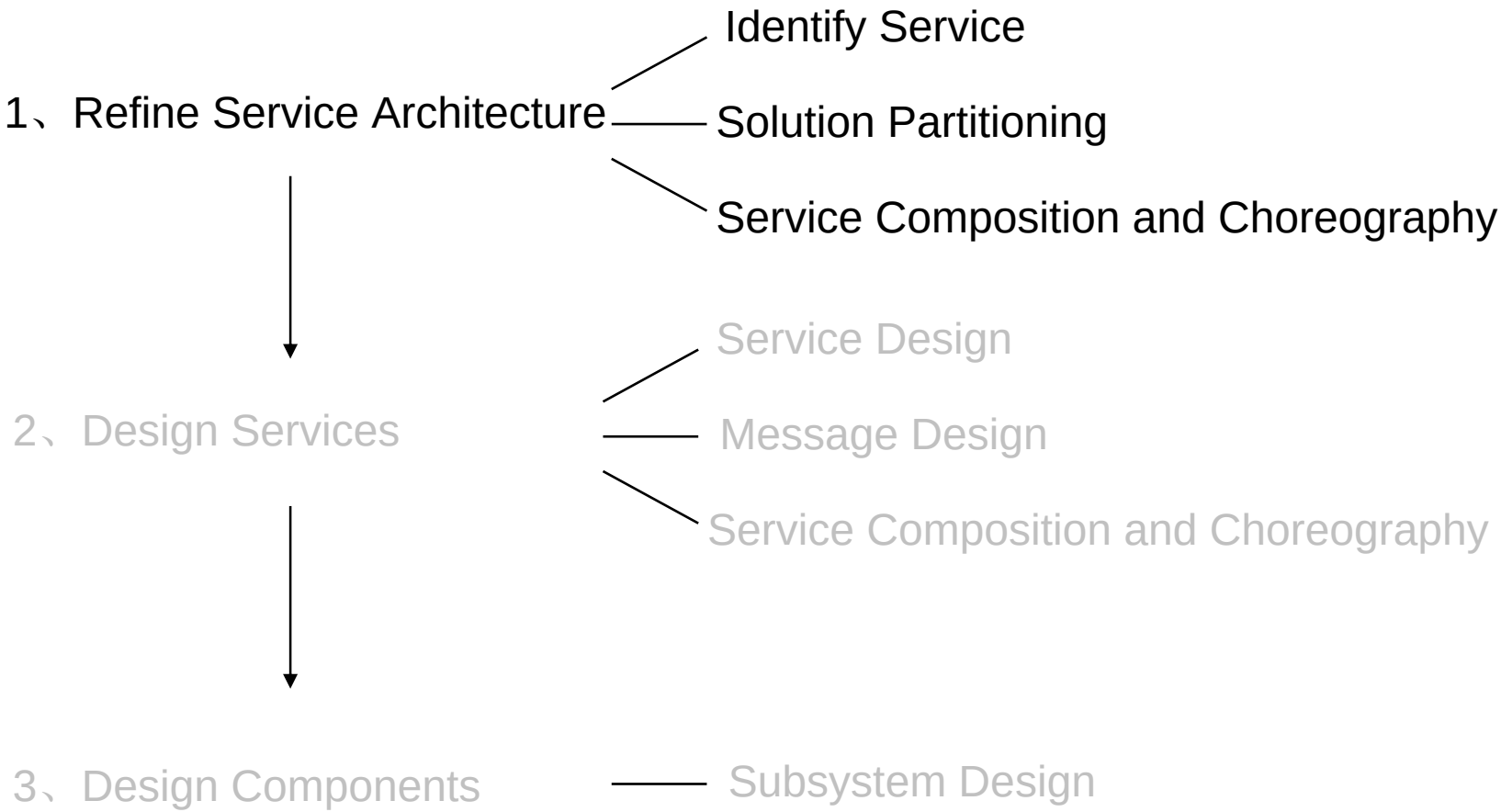
一、问题描述

（关于Demo的背景要补充在这里）

二、设计步骤

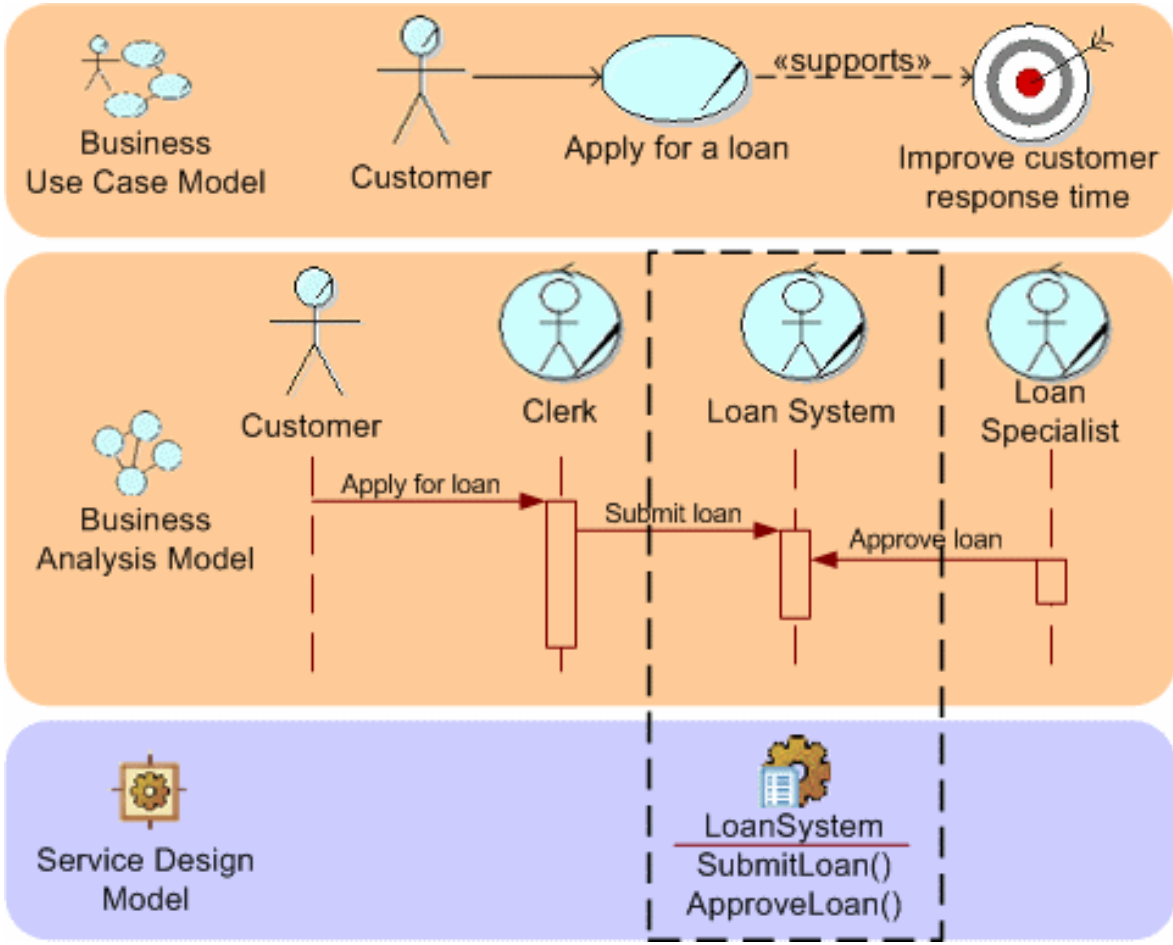


开始SOA 架构设计


1、Refine Service Architecture
Identify Service
Solution Partitioning
Service Composition and Choreography
2、Design Services
Service Design
Message Design
Service Composition and Choreography
3、Design Components
Subsystem Design

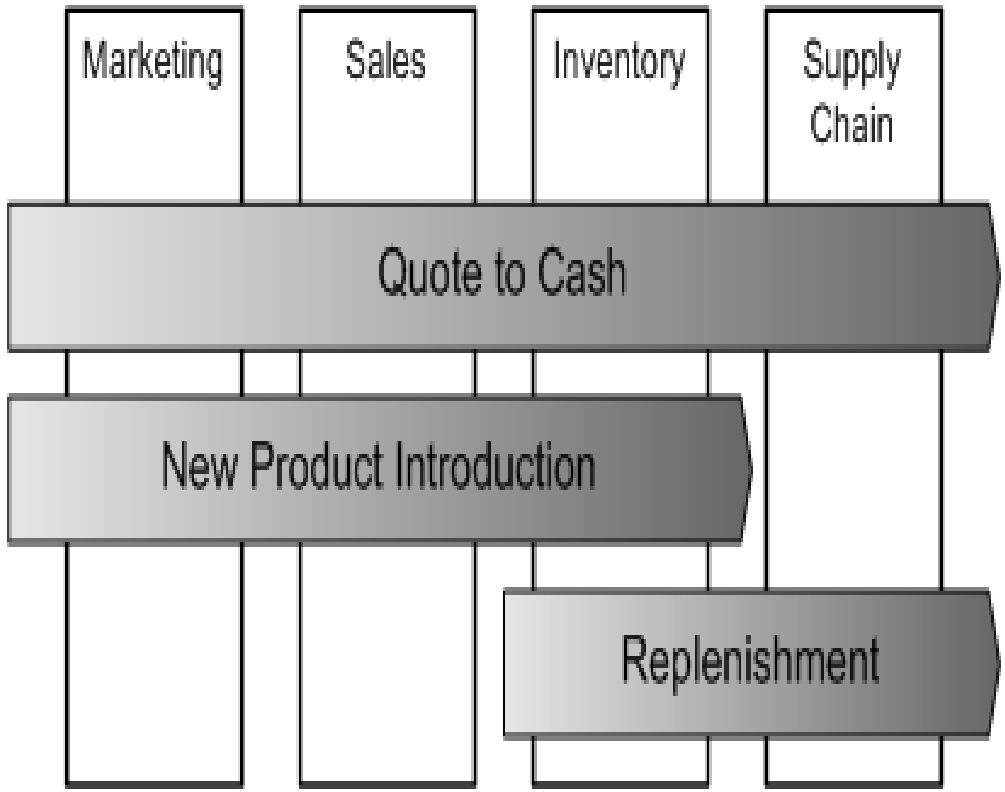
Identify Service

- 活动: Identify Services
  - 流程驱动
  - 用例驱动
  - 数据驱动
  - 规则驱动
  - 由底向上，通过现有的资产产生服务



Solution Partitioning

- 活动: Solution Partitioning
- 架构师通过Solution Partitioning来对架构中比较关注的地方进行深入研究
  - 如右图中，架构师描绘出Service在企业不同部门的应用情况

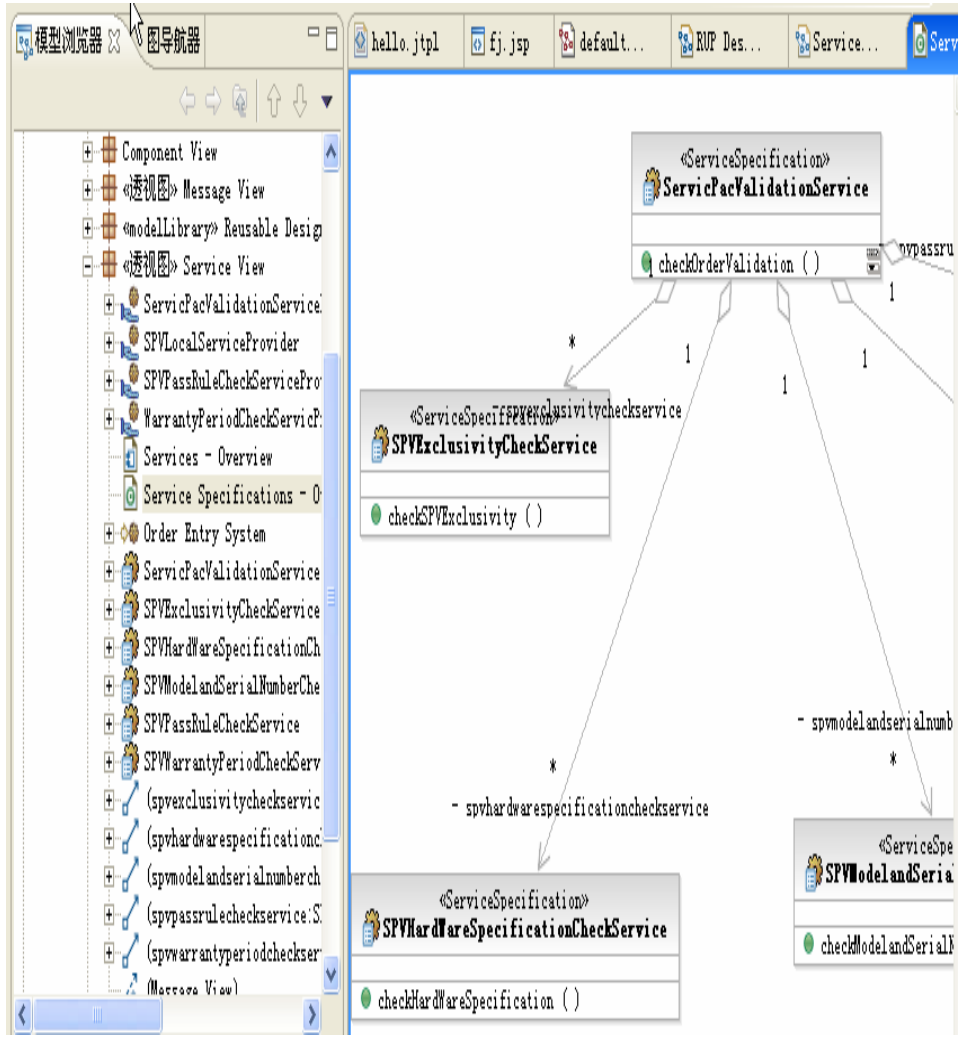


Service Composition and Choreography

- 活动: Service Composition and Choreography
- 如果一个ServiceProvider内部是由多个Service组成，可以通过Service Composition来描绘组成关系
- 如果要了解Service之间的协作关系，或者了解Service是如何一起实现整个业务流程的，可以通过Service Collaboration描绘

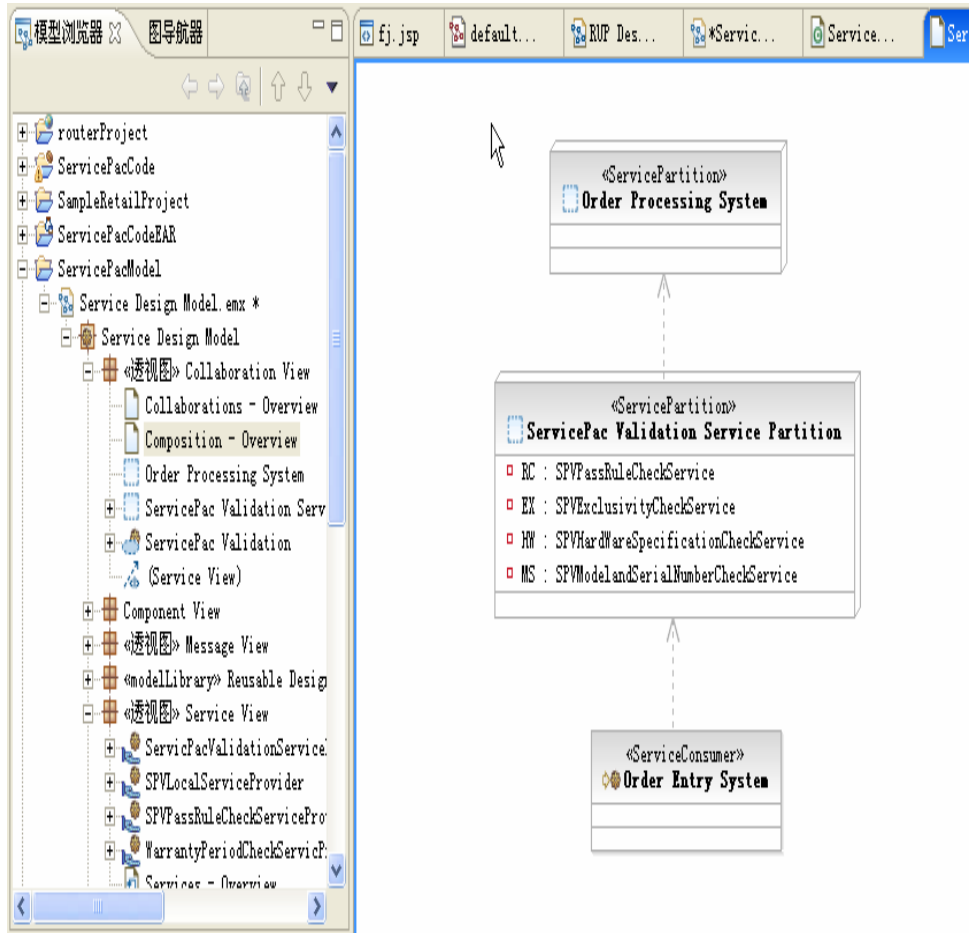
Demo1 - Service Specification

根据在业务需求阶段找出的Service，定义其Service Specification



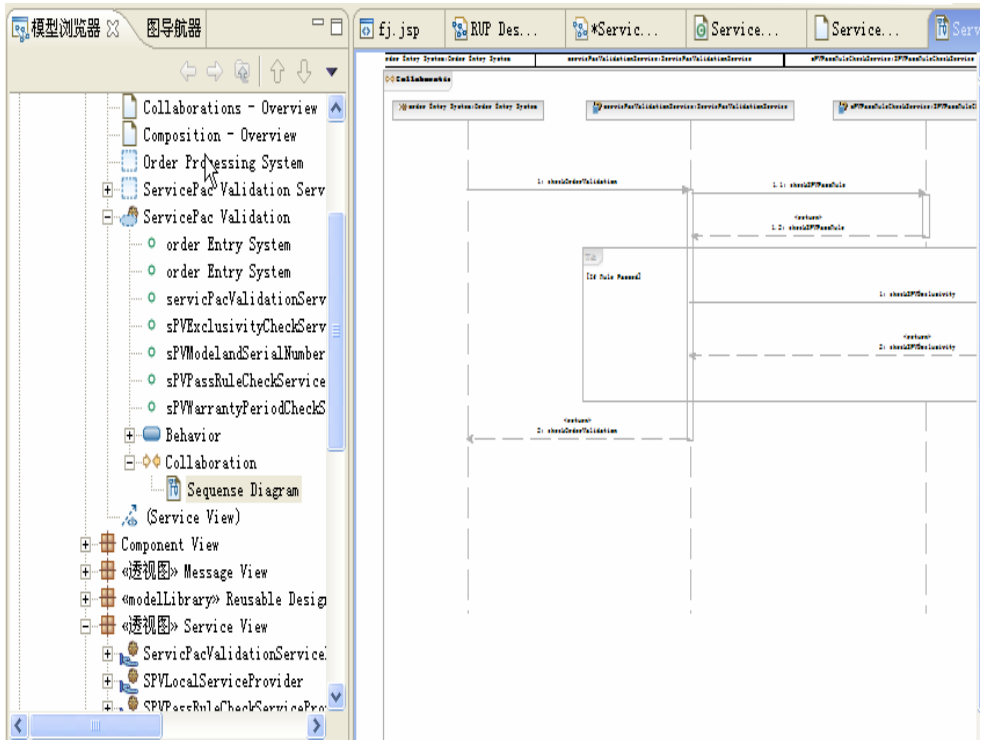
Demo2 - Service Partition

通过Service Partition描述Validation Service系统与其它系统的关系

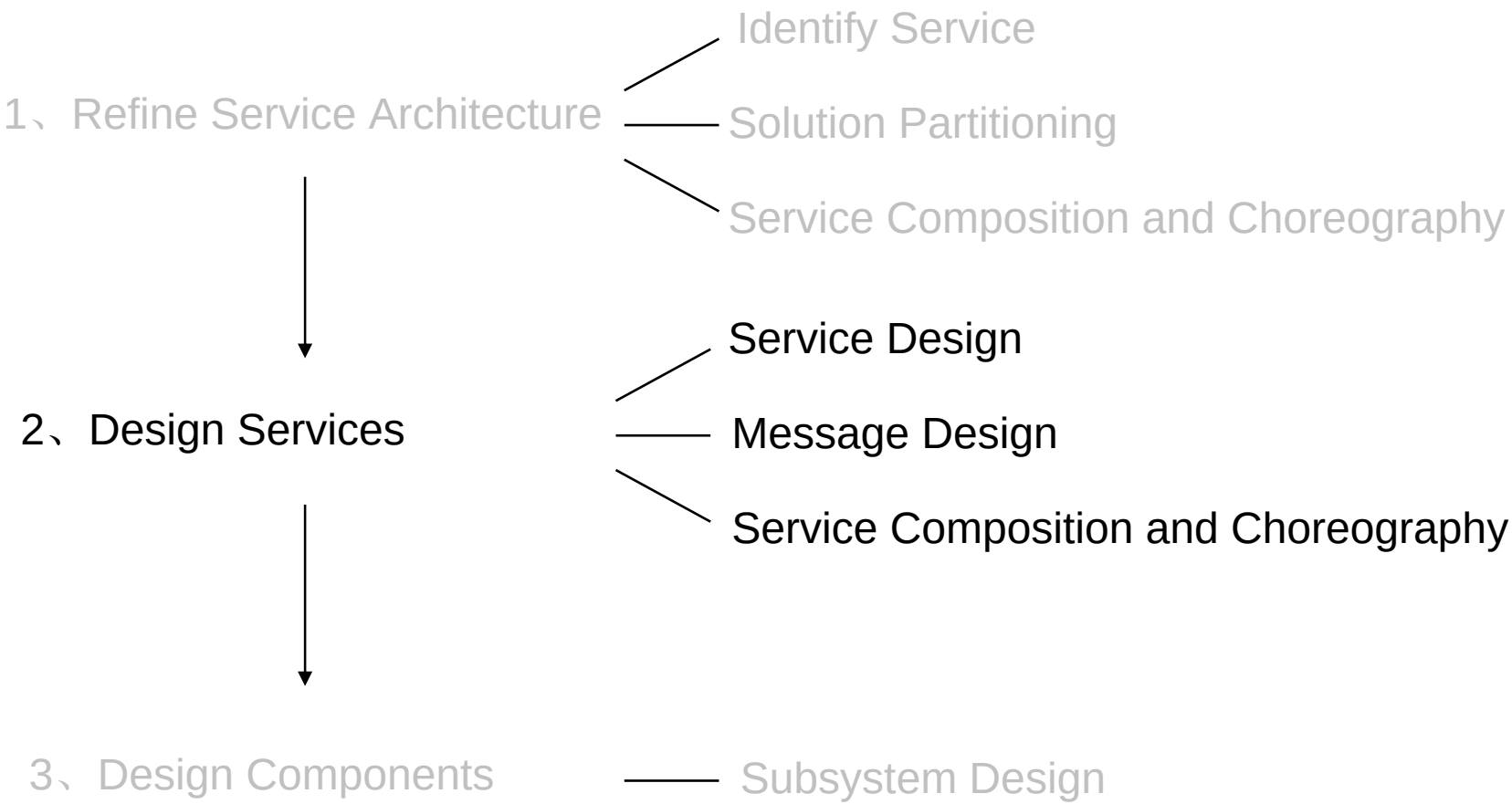


Demo3 - Service Collaboration in Identify Service

通过Service Collaboration描述Service Rule Validation场景，从而找出Service的职责

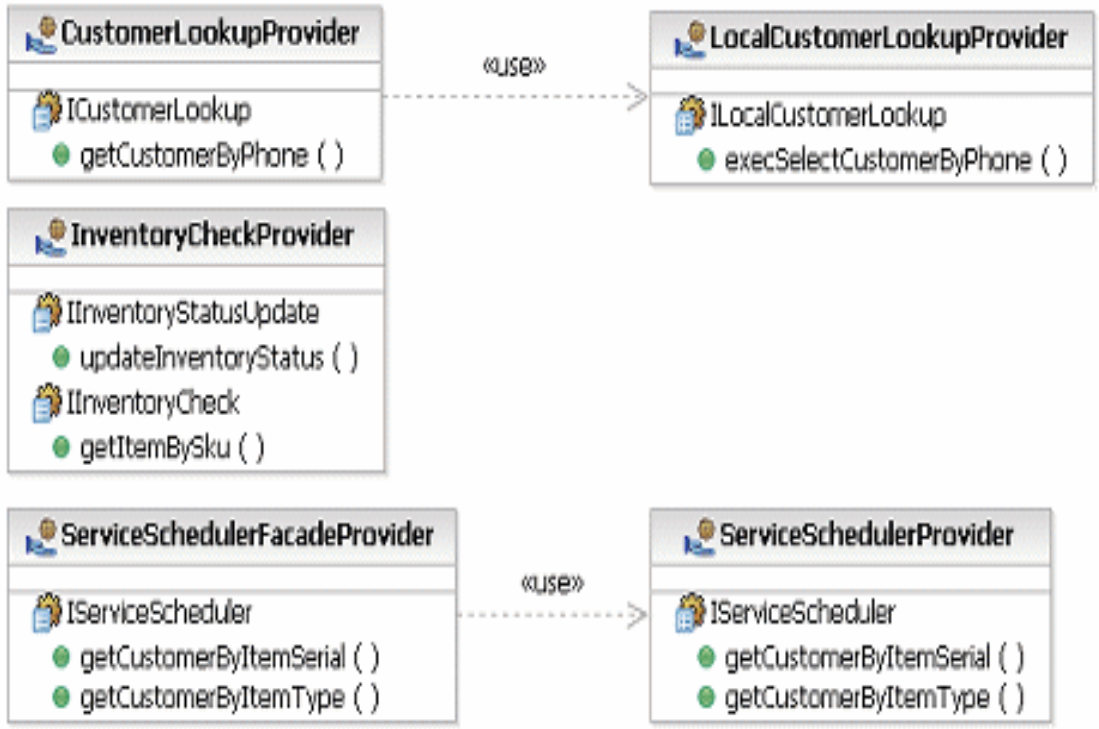


开始SOA 架构设计


1、Refine Service Architecture
Identify Service
Solution Partitioning
Service Composition and Choreography
2、Design Services
Service Design
Message Design
Service Composition and Choreography
3、Design Components
Subsystem Design

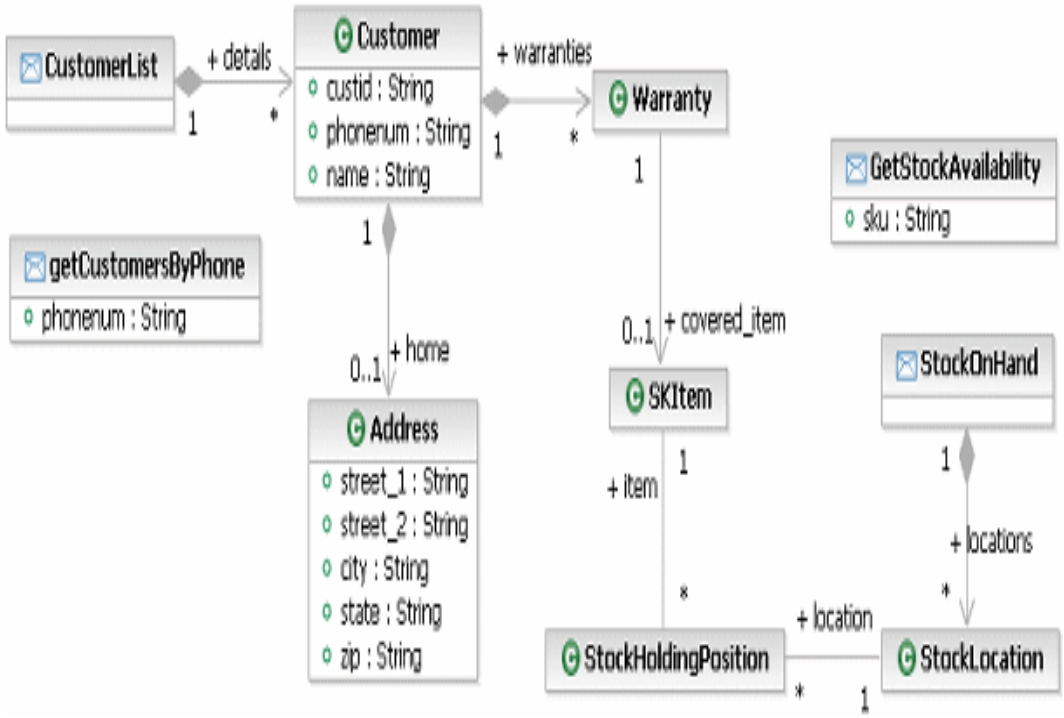
Service Design

- 活动: Service Design
- 在这一步通常要考虑如何具体的实现之前定义的Service Specification - 根据实际部署的情况定义Service Provider，合并相应的Service Specification



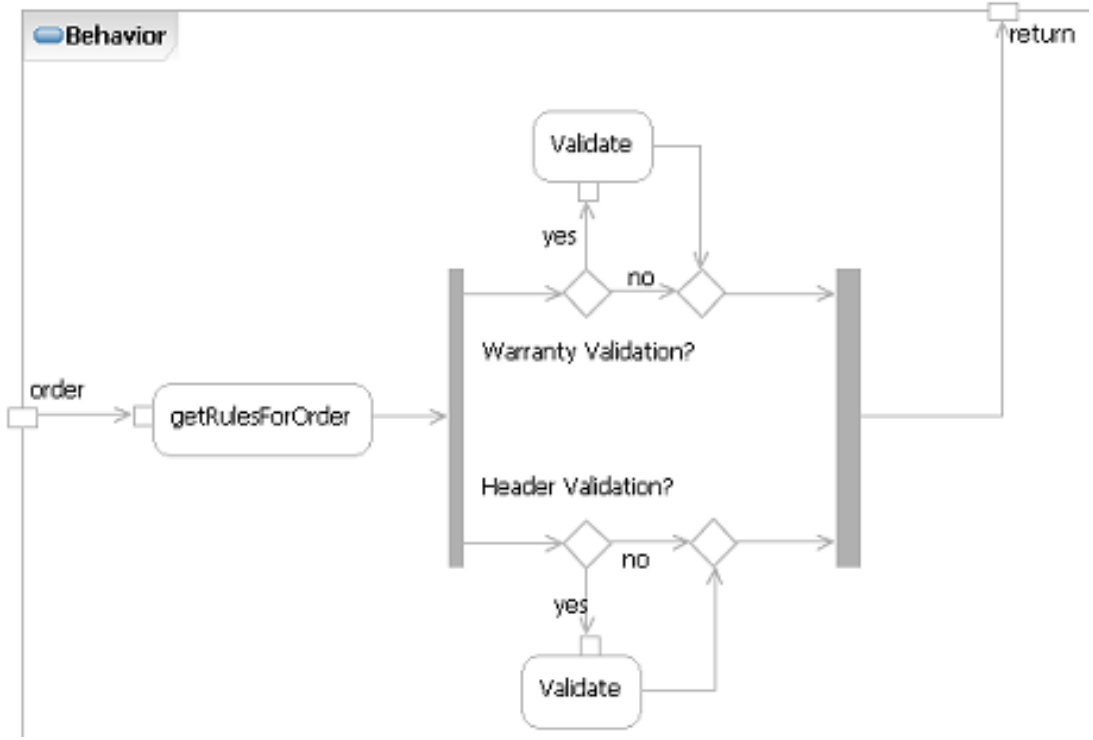
Message Design

- 活动: Message Design
- 首先定义领域的实体模型，然后通过Message定义实体类之间的数据传递



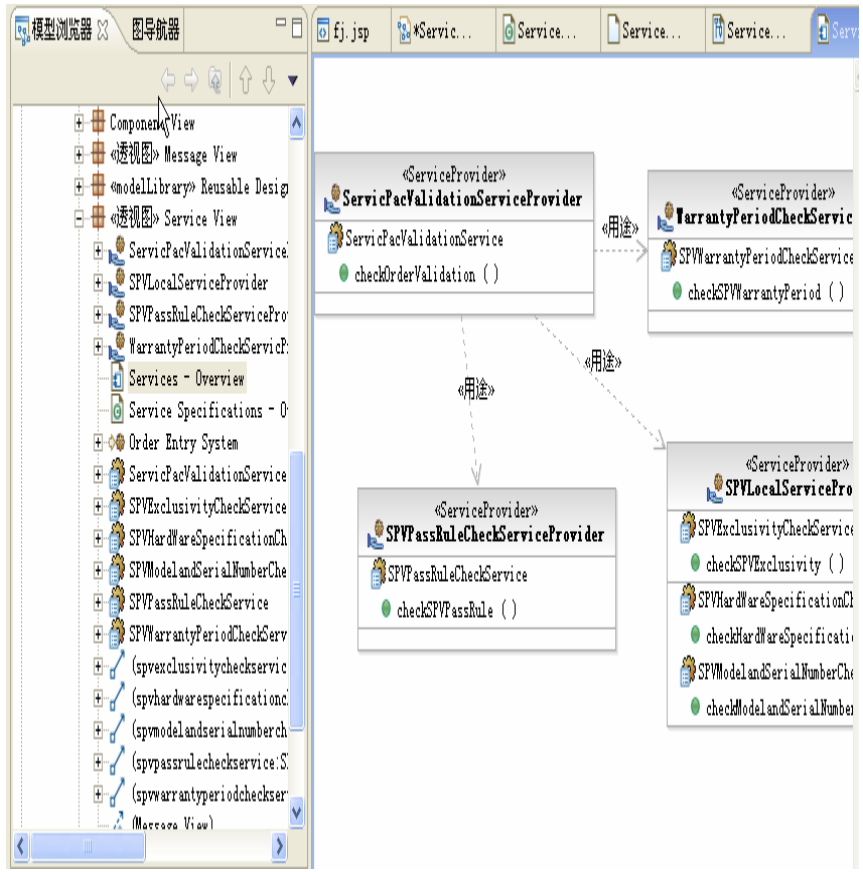
Service Composition and Choreography

- 活动: Service Composition and Choreography
- 通过Service Collaboration描绘Service是如何一起实现整个业务流程的，



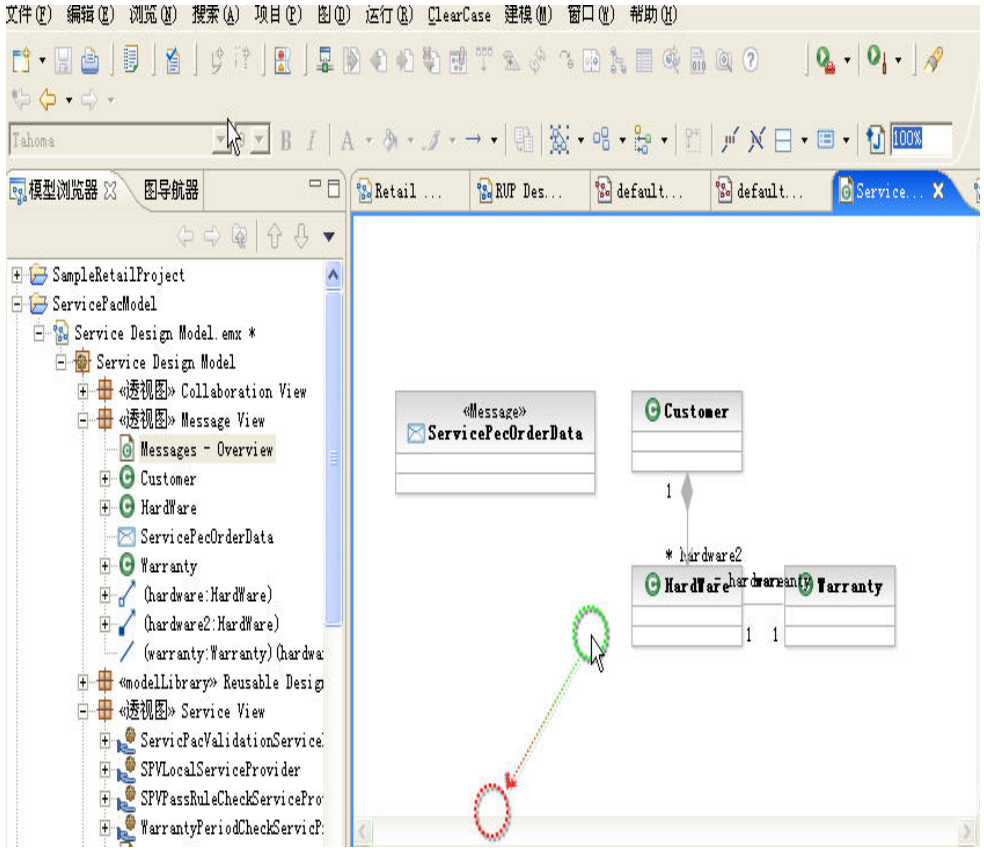
Demo4 - Service Provider

把第三方系统提供的Service由单独的ServiceProvider提供，而local Service由一个localServiceProvider提供



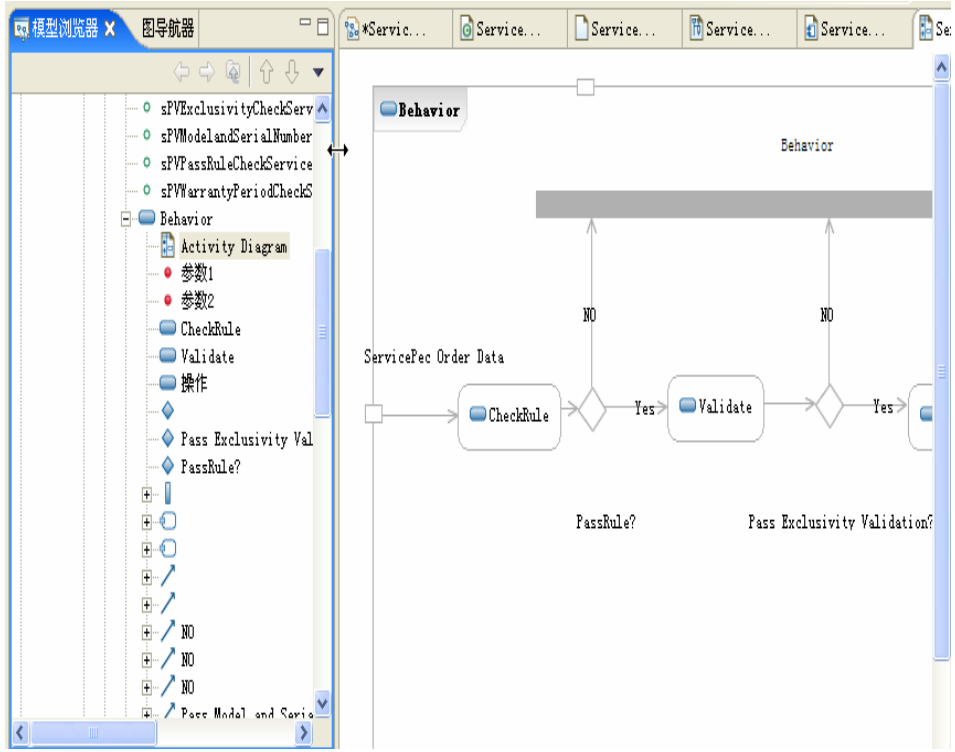
Demo5 – Message Design

- 首先定义领域的实体模型，然后通过Message定义实体类之间的数据传递

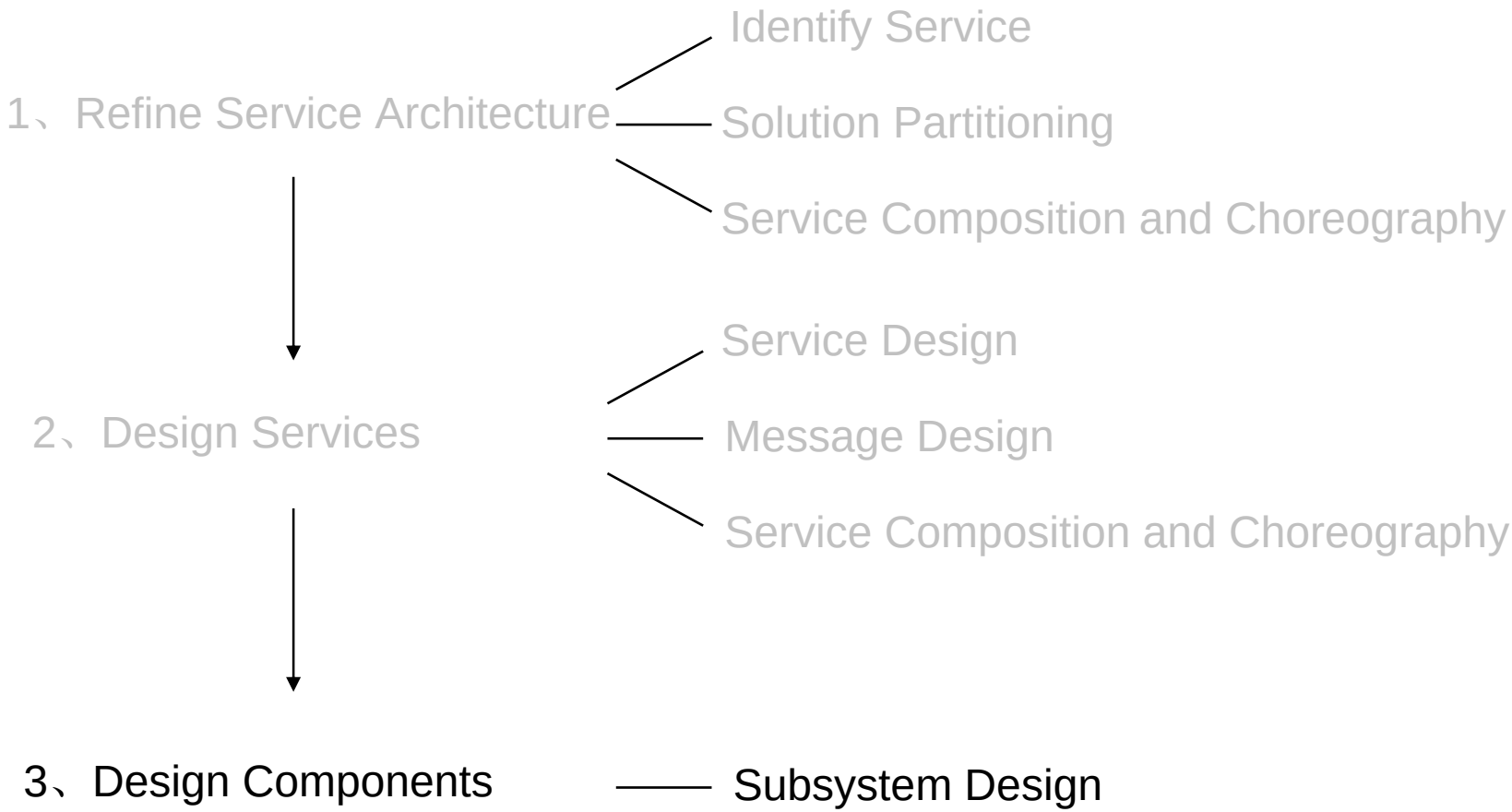


Demo6 - Service Collaboration in Design Service

通过业务流程将定义好的ServiceProvider串起来，以复审ServiceProvider的职责是否正确，以及为转变为BPEL流程打下基础

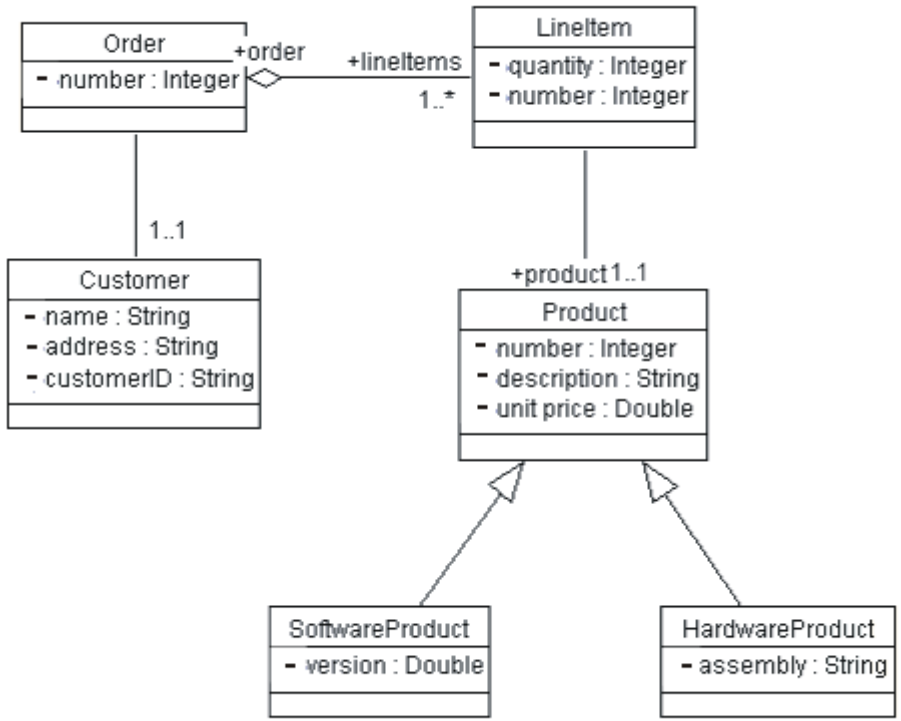


开始SOA 架构设计


1、Refine Service Architecture
Identify Service
Solution Partitioning
Service Composition and Choreography
2、Design Services
Service Design
Message Design
Service Composition and Choreography
3、Design Components
Subsystem Design

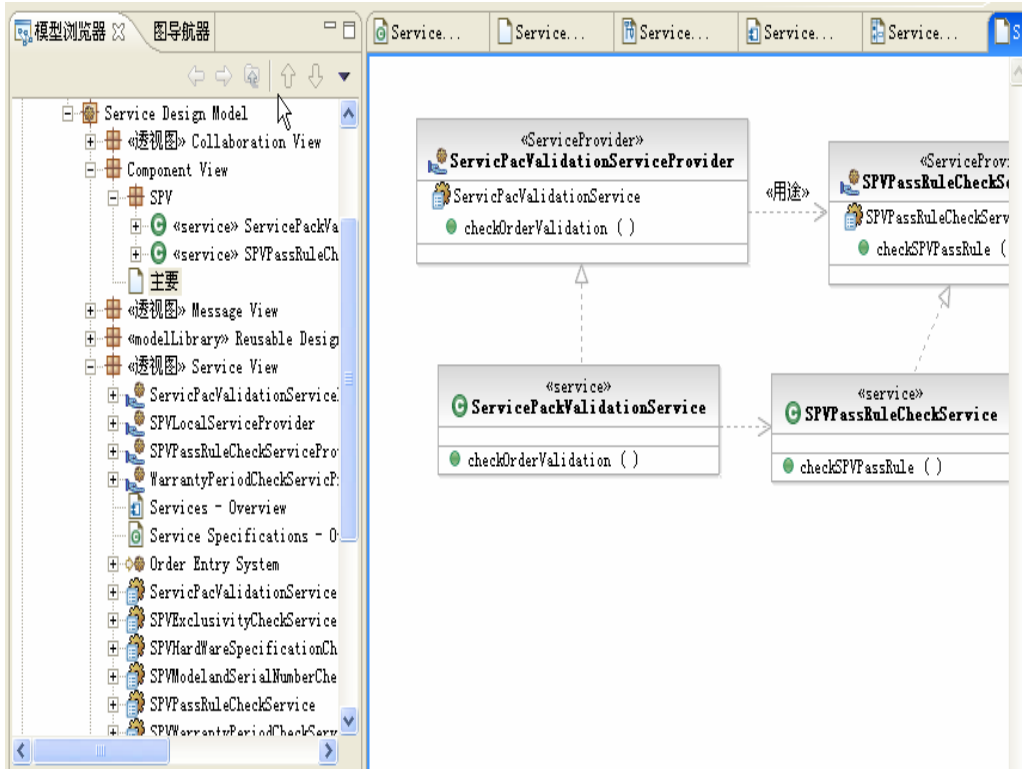
Subsystem Design

在此步具体设计实现ServiceProvider的设计类



Demo7 - Subsystem Design

在Demo里面每个ServiceProvider都由相应的Session Bean来实现




Thank
You