服务计算
Service Computing

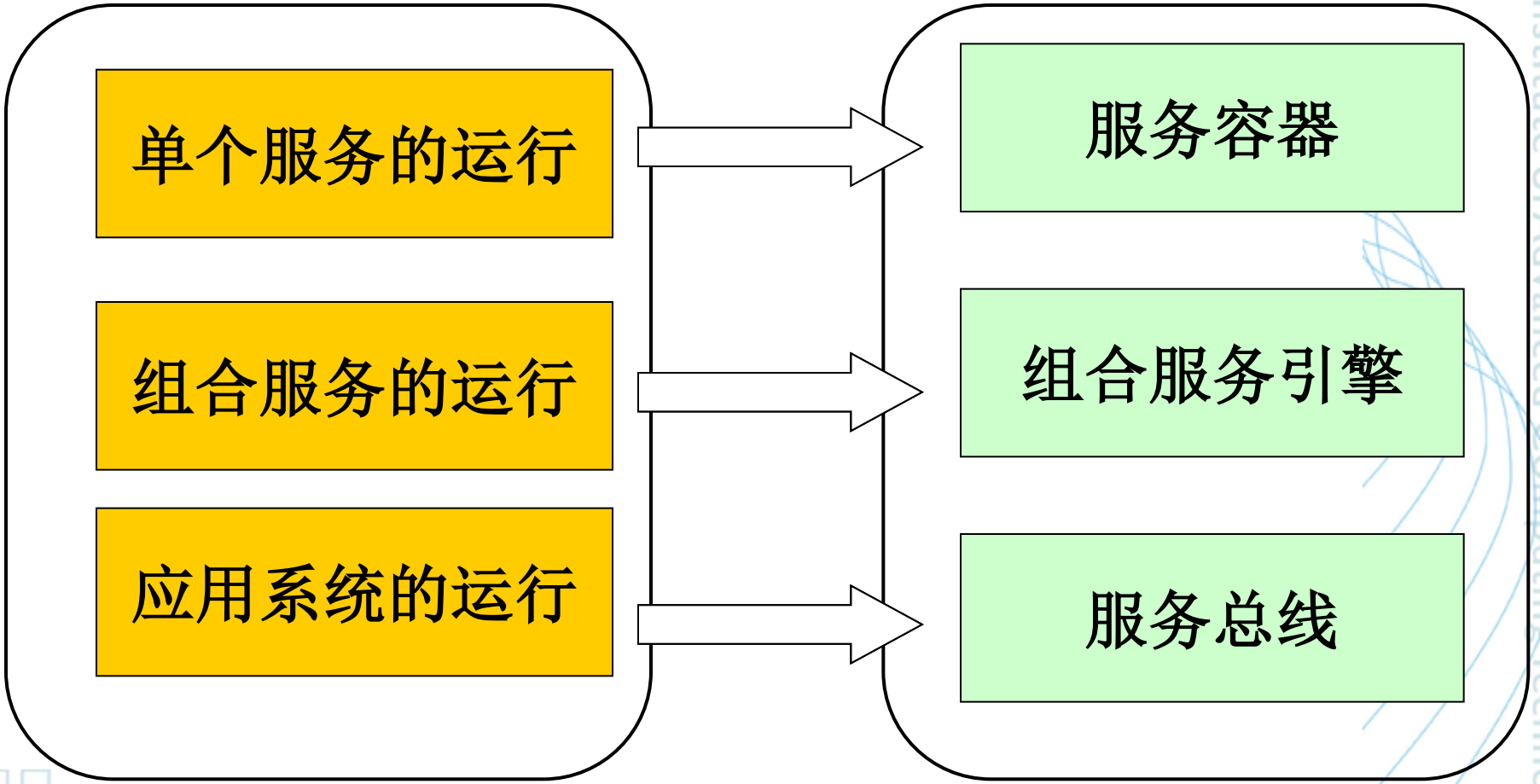
王旭
wangxu@buaa.edu.cn

七、服务运行

提纲

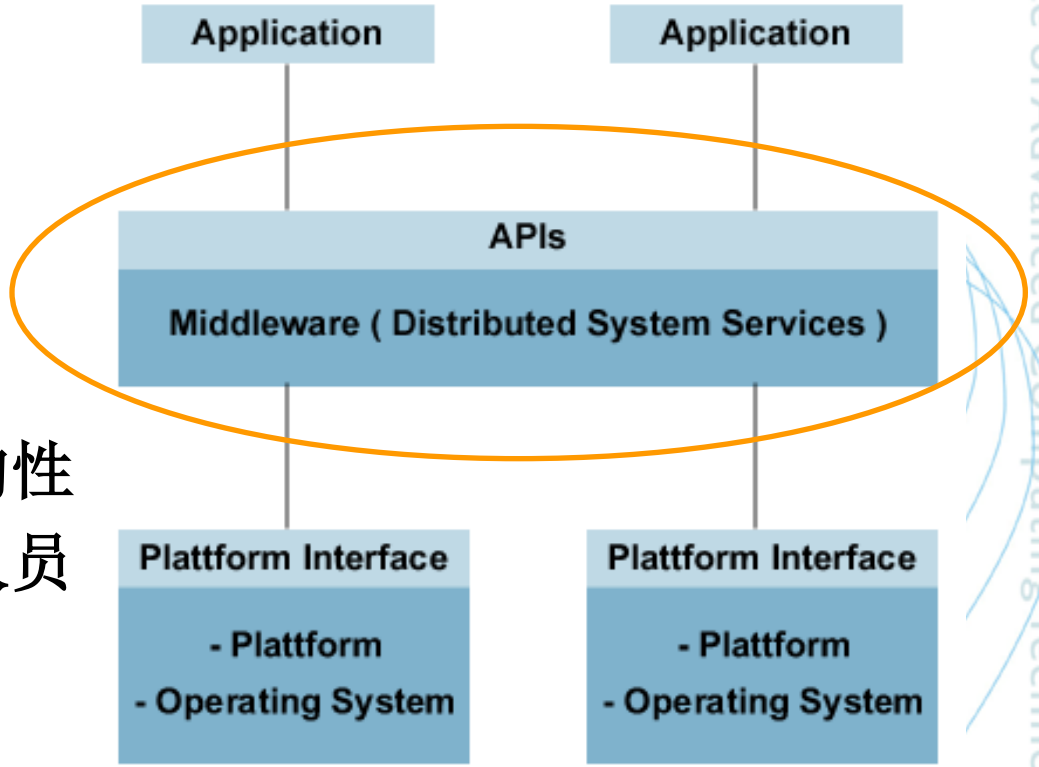
- 运行支撑中间件
- 服务的安全管理
- 服务的事务管理
- 服务的动态演化

服务的运行支撑



中间件技术

- 位于应用和操作系统之间的软件层
- 产生的动机
  - 应用整合
  - 分布式计算
- 中间件的功能
  - 屏蔽分布性
  - 屏蔽基础软件的异构性
  - 向上层应用的开发人员提供统一接口
  - 提供一组通用服务



服务容器: 支撑单个服务运行的中间件

- 容器：管理组件及组件交互的中间件

- 服务容器类别
  - RPC引擎
  - HTTP服务器
  - Web服务容器

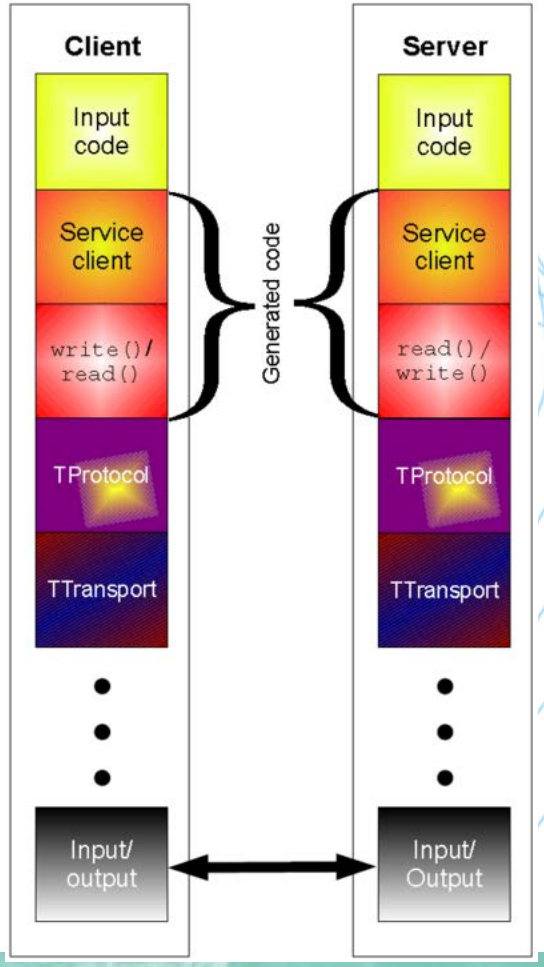
RPC引擎

- **RPC引擎**
  - 支持参数编码和传递
    - JSON, ProtoBuf, XML等等
  - 支持网络传输协议
    - Socket, HTTP以及专有协议
- 代表性框架
  - **Apache Dubbo:** 多协议，支持**SOA**
  - **Apache Thrift:** 多语言支持
  - **XML-RPC**
  - **JSON-RPC**
  - **Microsoft DCOM**

Apache Thrift

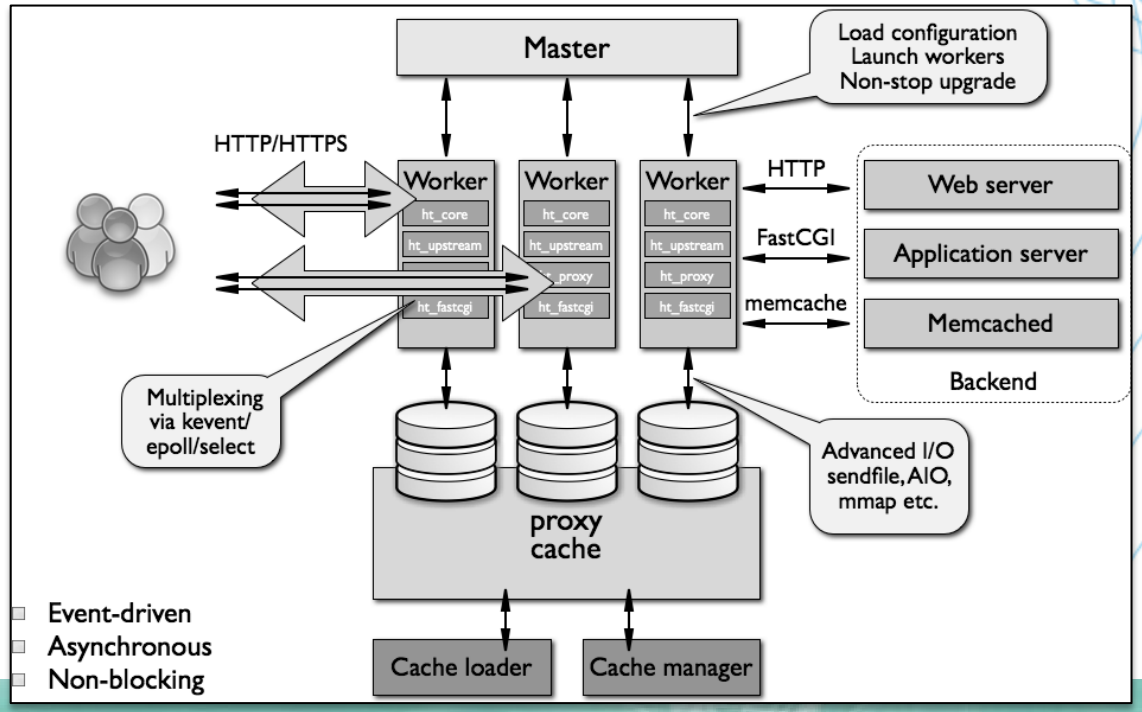
- 自动生成不同语言的RPC客户端和服务器端代码

- **TProtocol:数据编码**
  - **TBinaryProtocol**
  - **TCompactProtocol**
  - **TJSONProtocol**
- **Tranport:网络协议**
  - **Tsocket**
  - **TFramedTransport**



HTTP服务器

- 实现对HTTP协议的接收、解析与应答
- 支持不同服务器语言协议
  - **Apache HTTP Server, Nginx, IIS：HTTP, HTML**
  - **Apache Tomcat, JBoss: java/jsp/servlet**
  - **Node.js: js**

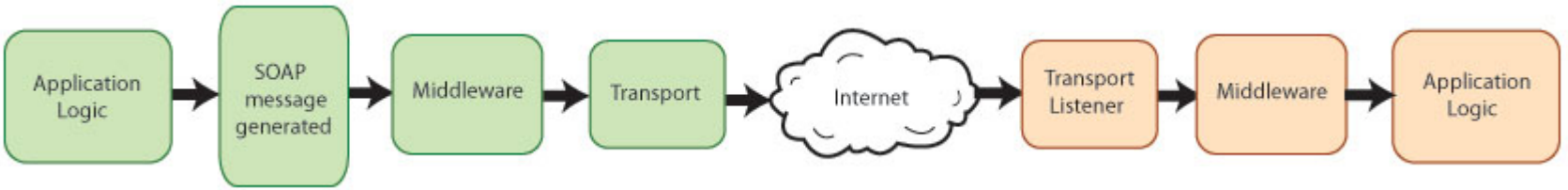


Web服务容器

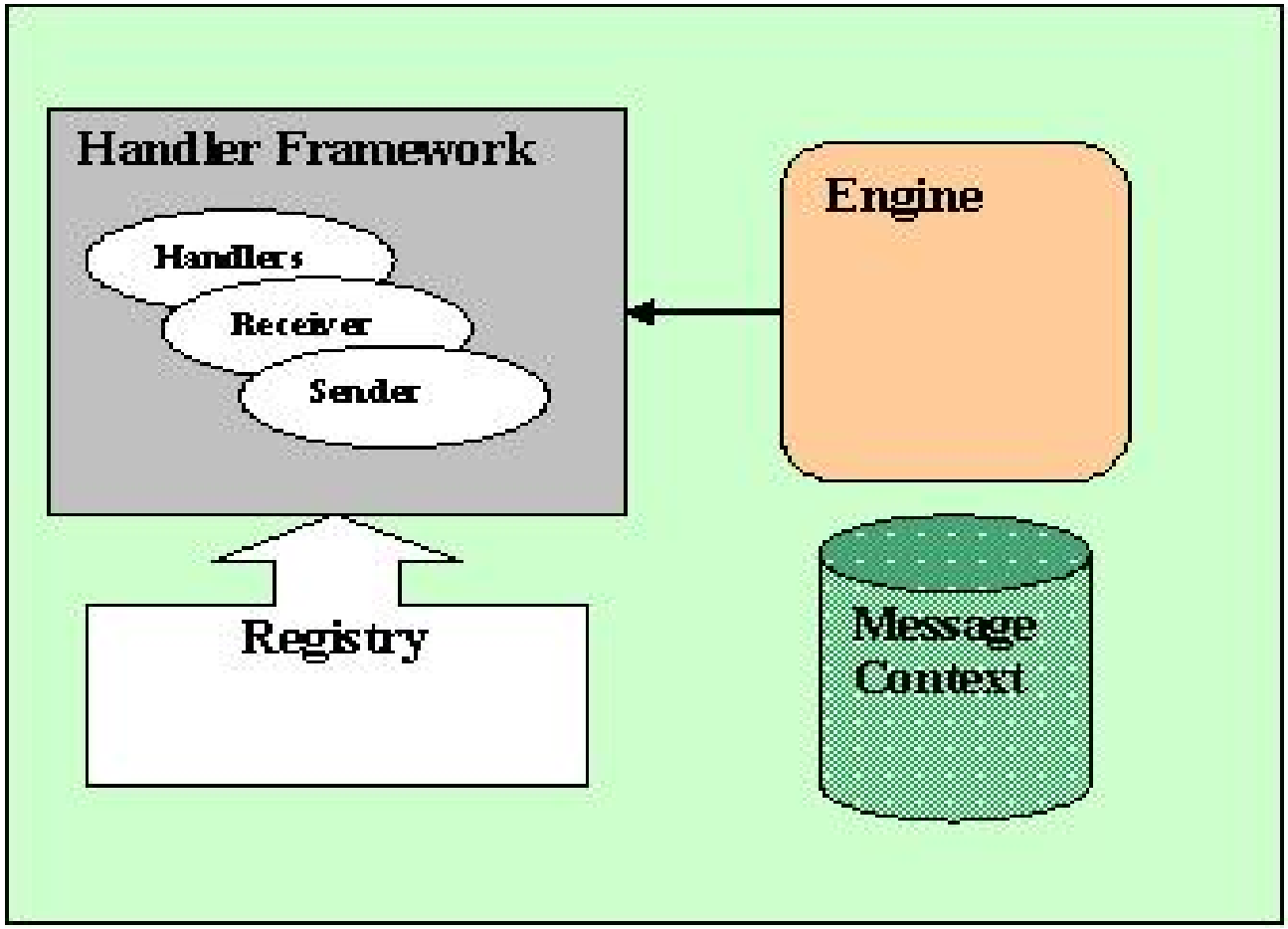
- Web服务容器：支持Web服务运行的中间件环境，提供所有服务的运行所需要的共性支撑
  - SOAP消息的处理
  - 和遗留系统的接口
  - 数据类型的编码/解码
  - 用户请求的调度执行
  - 服务的部署/反部署
  - 监控及管理
  - 其他功能：日志、安全

Apache Axis2

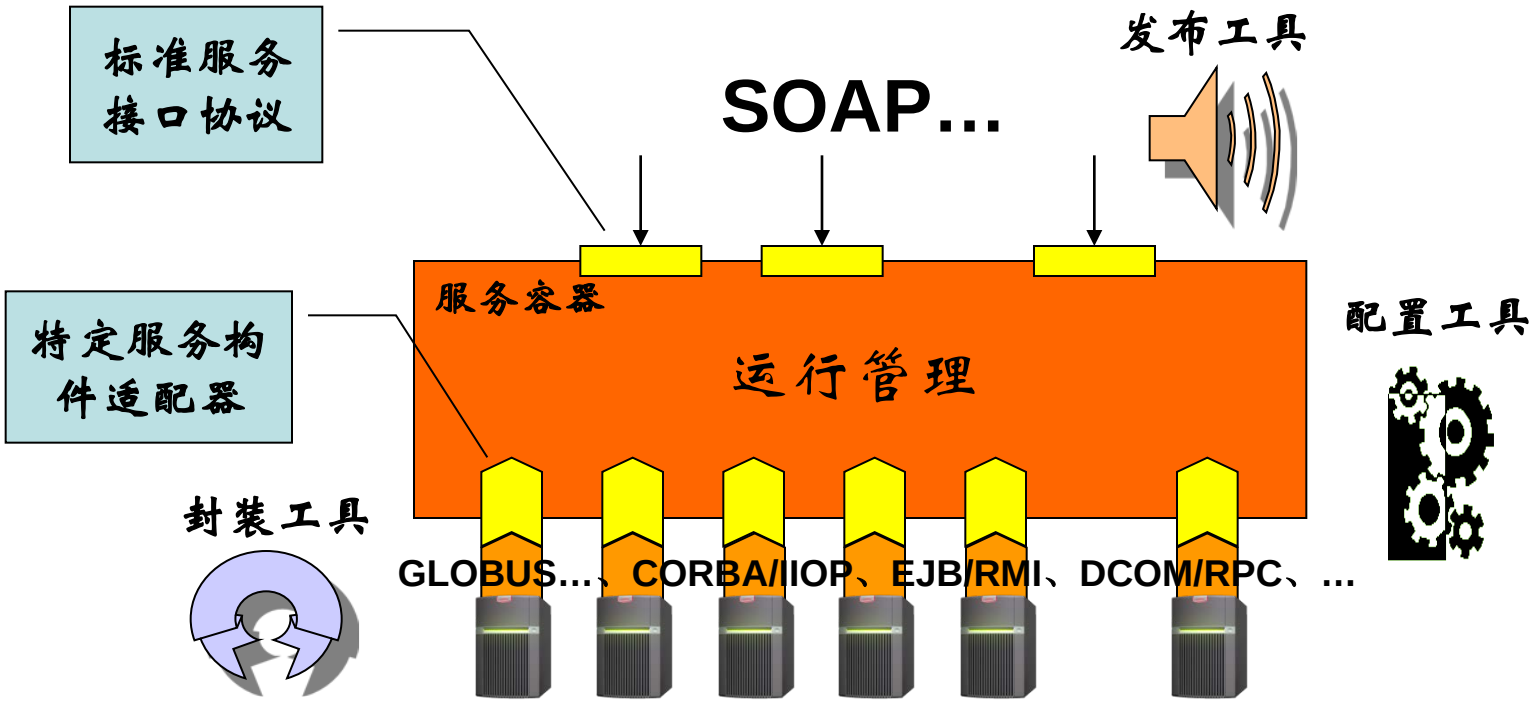
- Axis2的新特性
  - more efficient, more modular and more XML-oriented
- 主要功能
  - SOAP消息处理：普通SOAP消息、SOAP附件
  - WSDL相关处理
  - Java类的服务封装
  - WS-*规范的支持: Security, RM, Addressing, Coordination, Atomic Transaction
  - RESTful 服务的支持



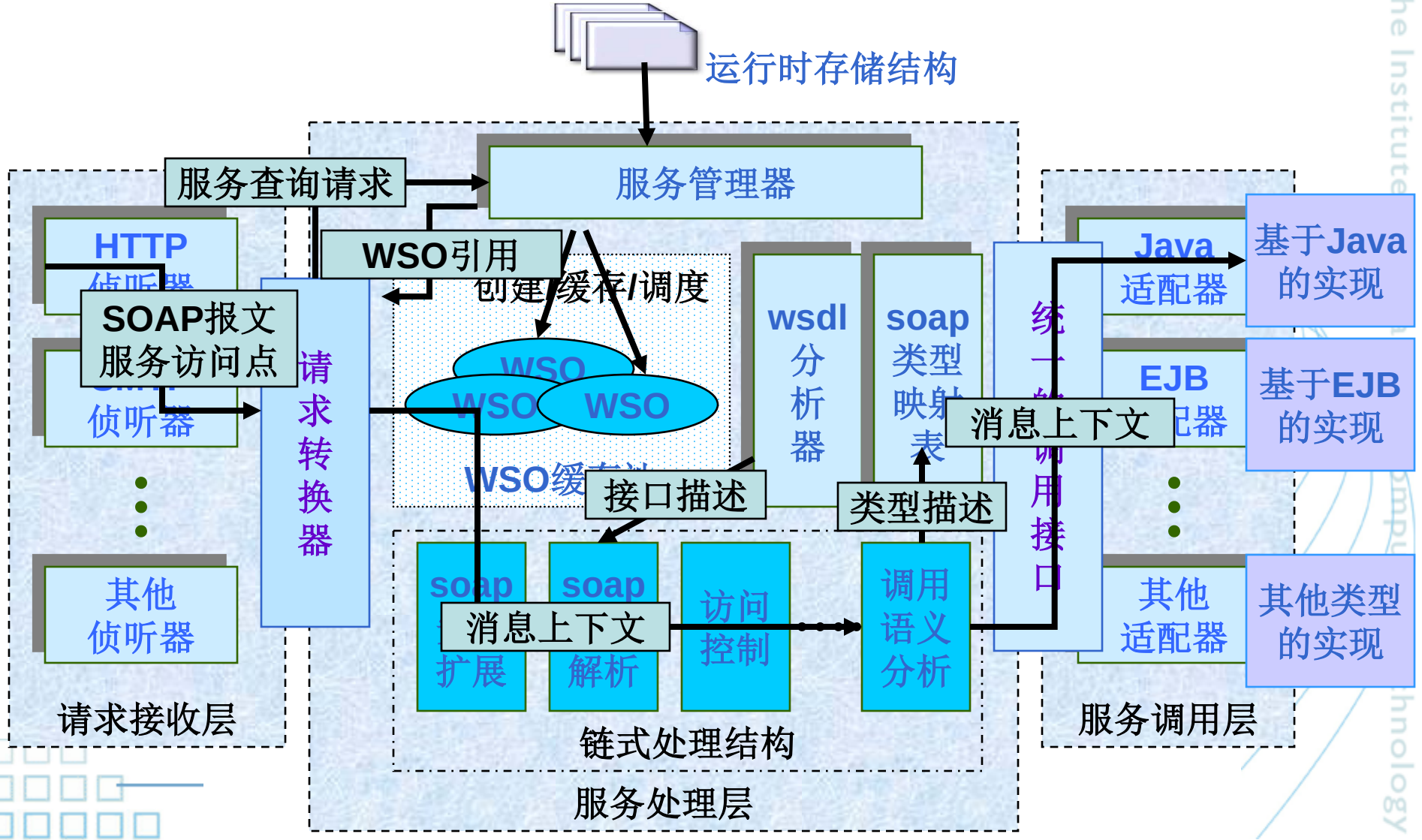
Axis2 核心组件



服务容器的基本功能结构

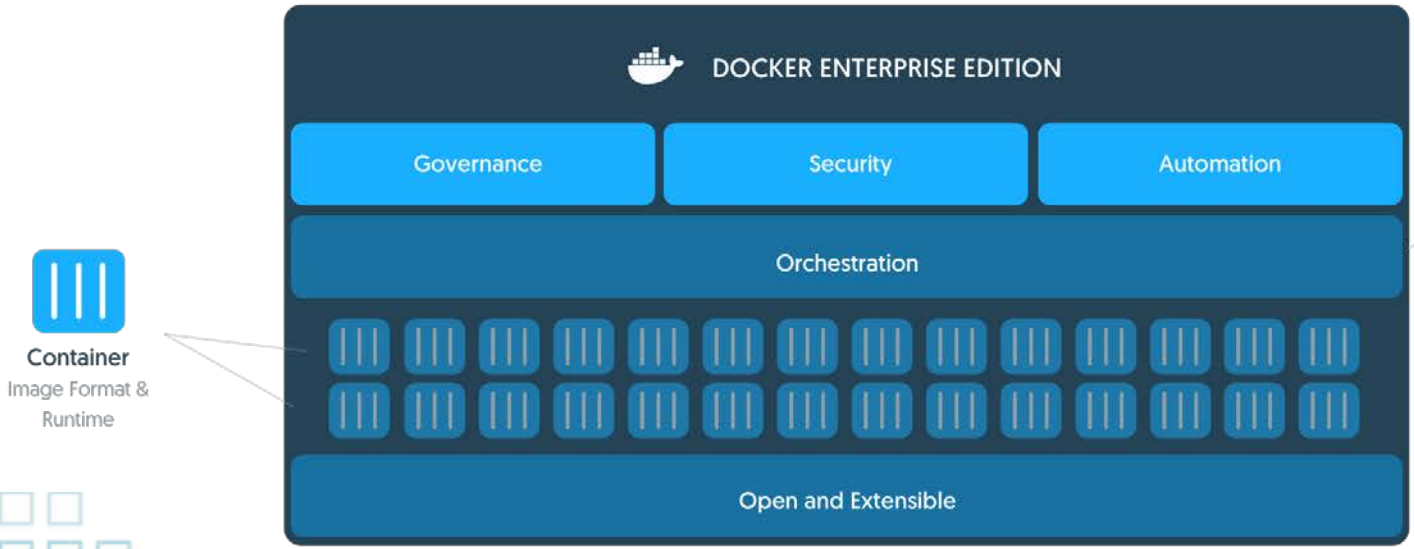


Web服务容器的另一个实例



Docker容器及其集群

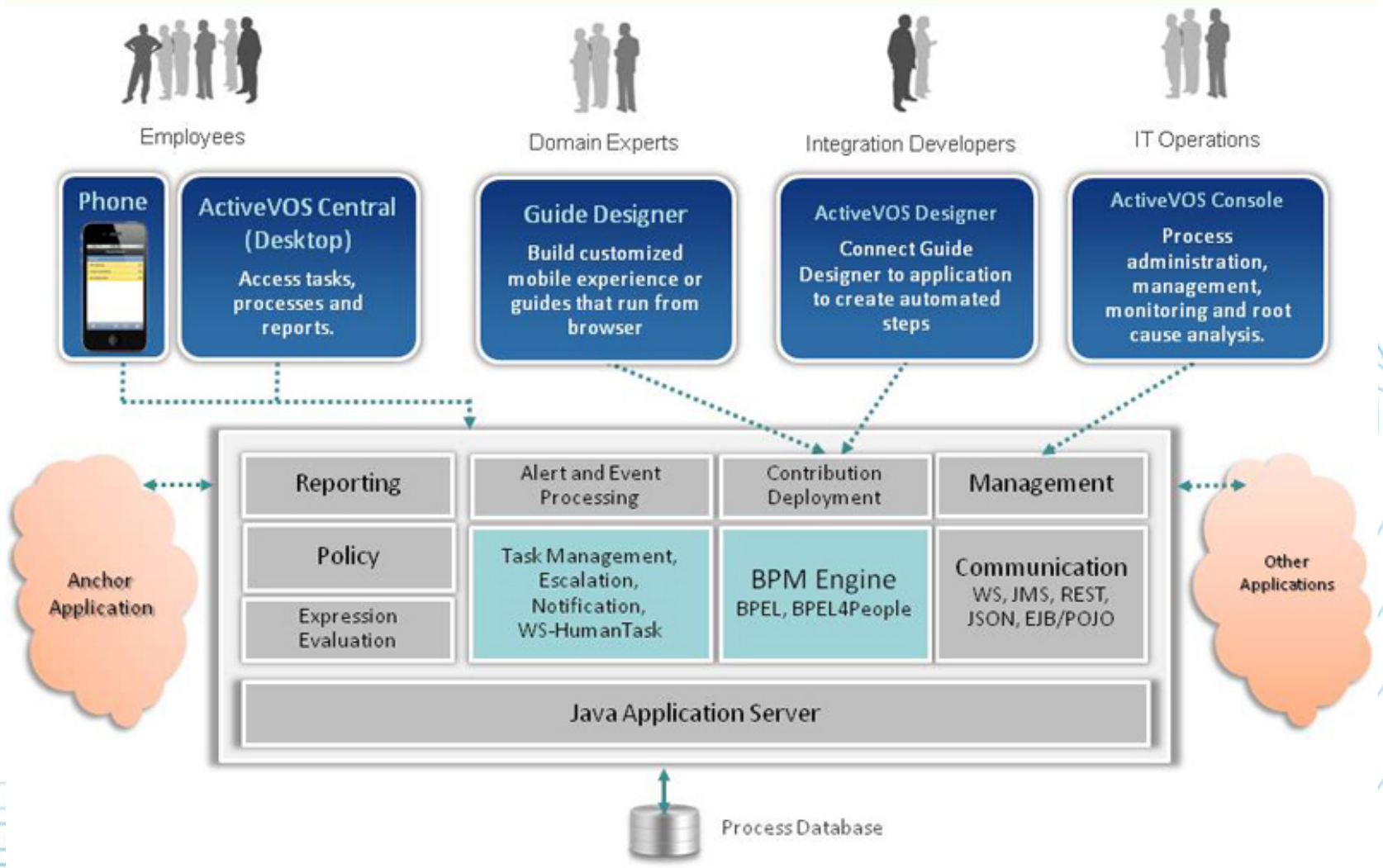
- **Docker容器**
  - 可为各种服务器创建轻量、独立、可移植的容器
  - 包括虚拟机、各类服务器及中间件
  - 容器集群：管理大量容器构成的网络集群



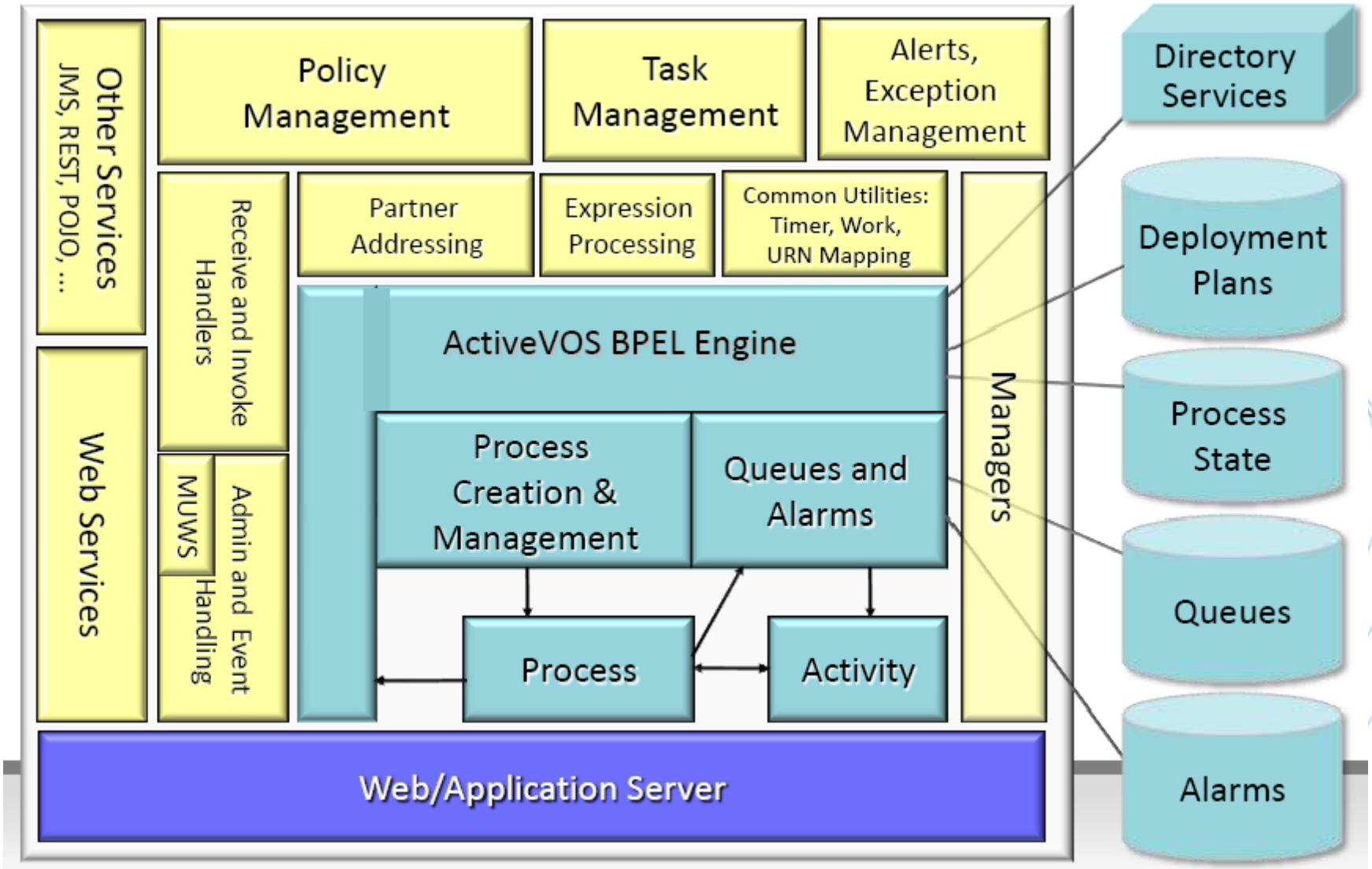
组合服务执行引擎

- 主要功能
  - SOAP消息处理
  - 解析服务组合规范
  - 按照组合逻辑，解释执行应用处理逻辑
  - 多个执行实例的管理
  - 监控管理

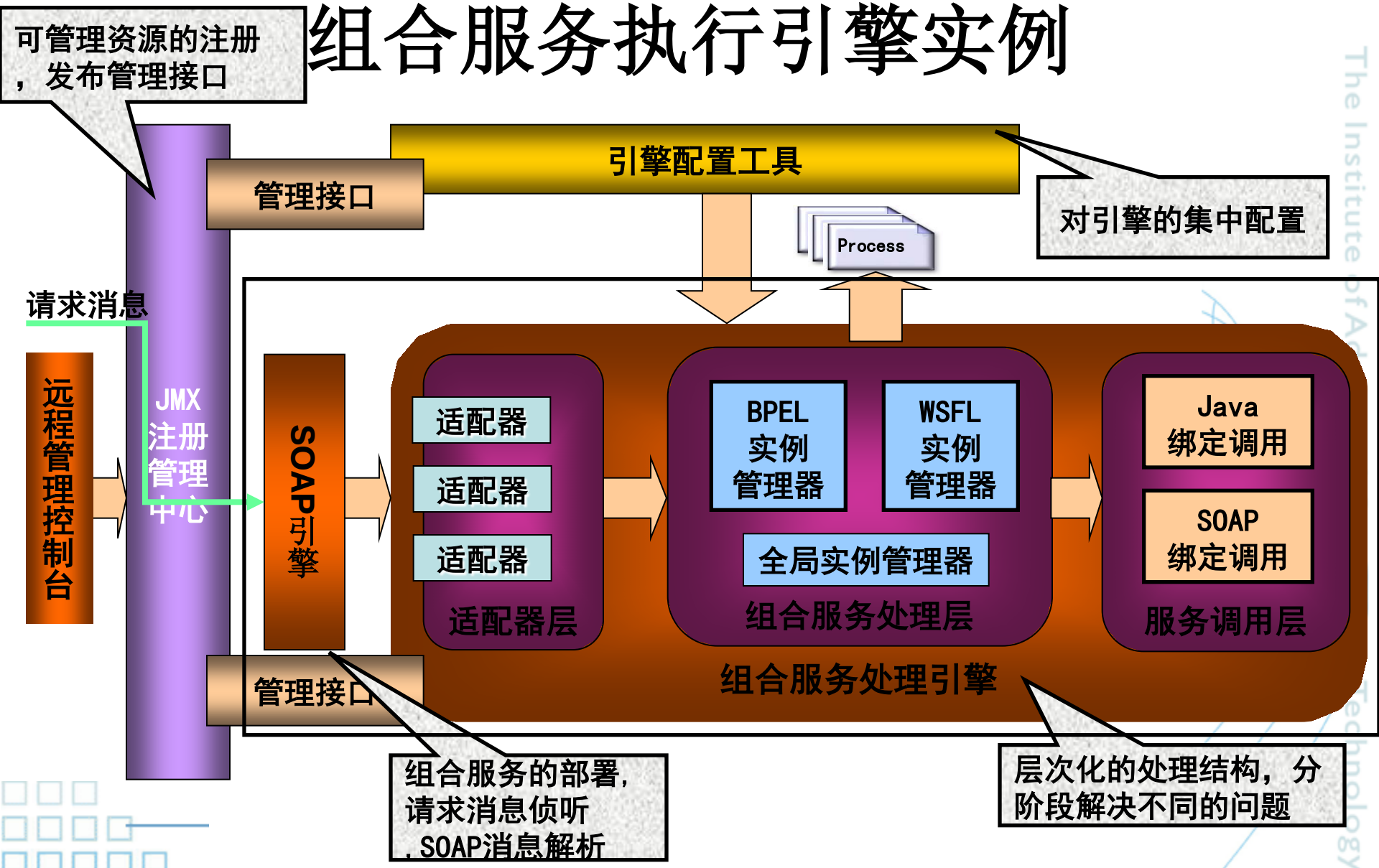
ActiveBPEL→ActiveVOS



ActiveVOS Server

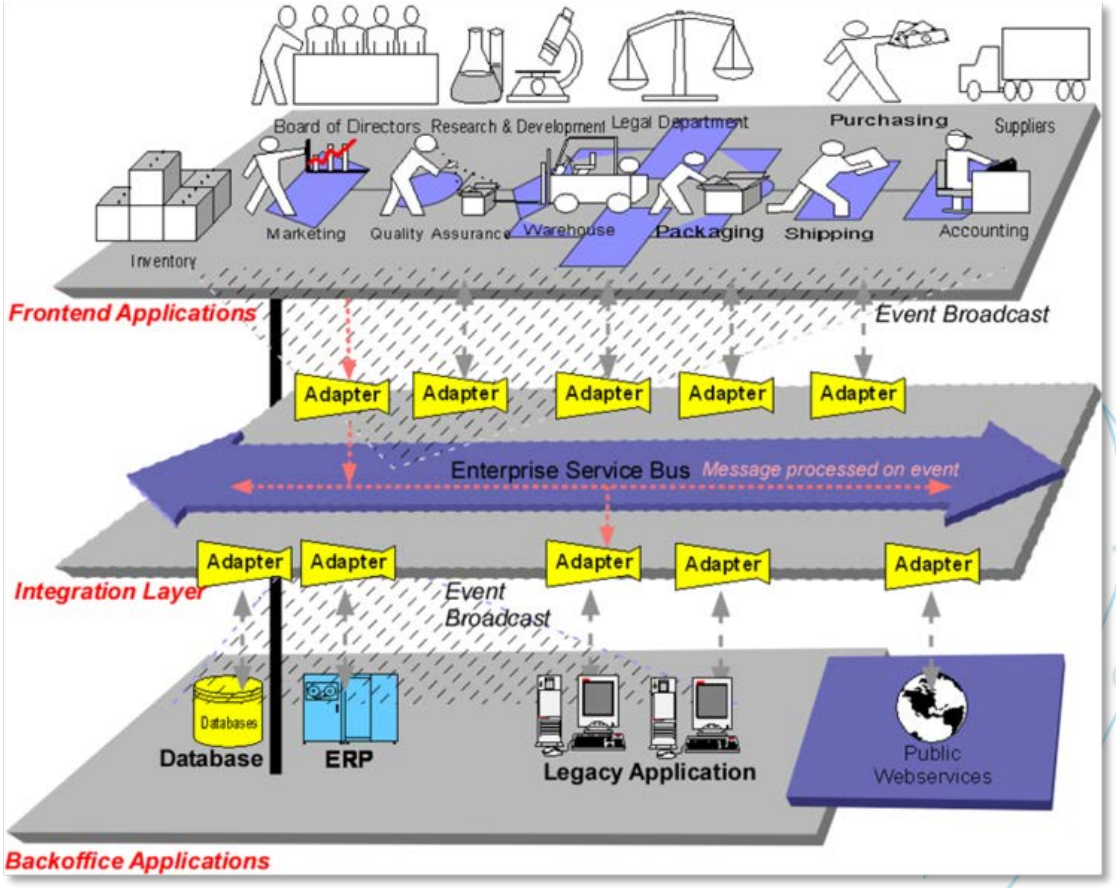


组合服务执行引擎实例

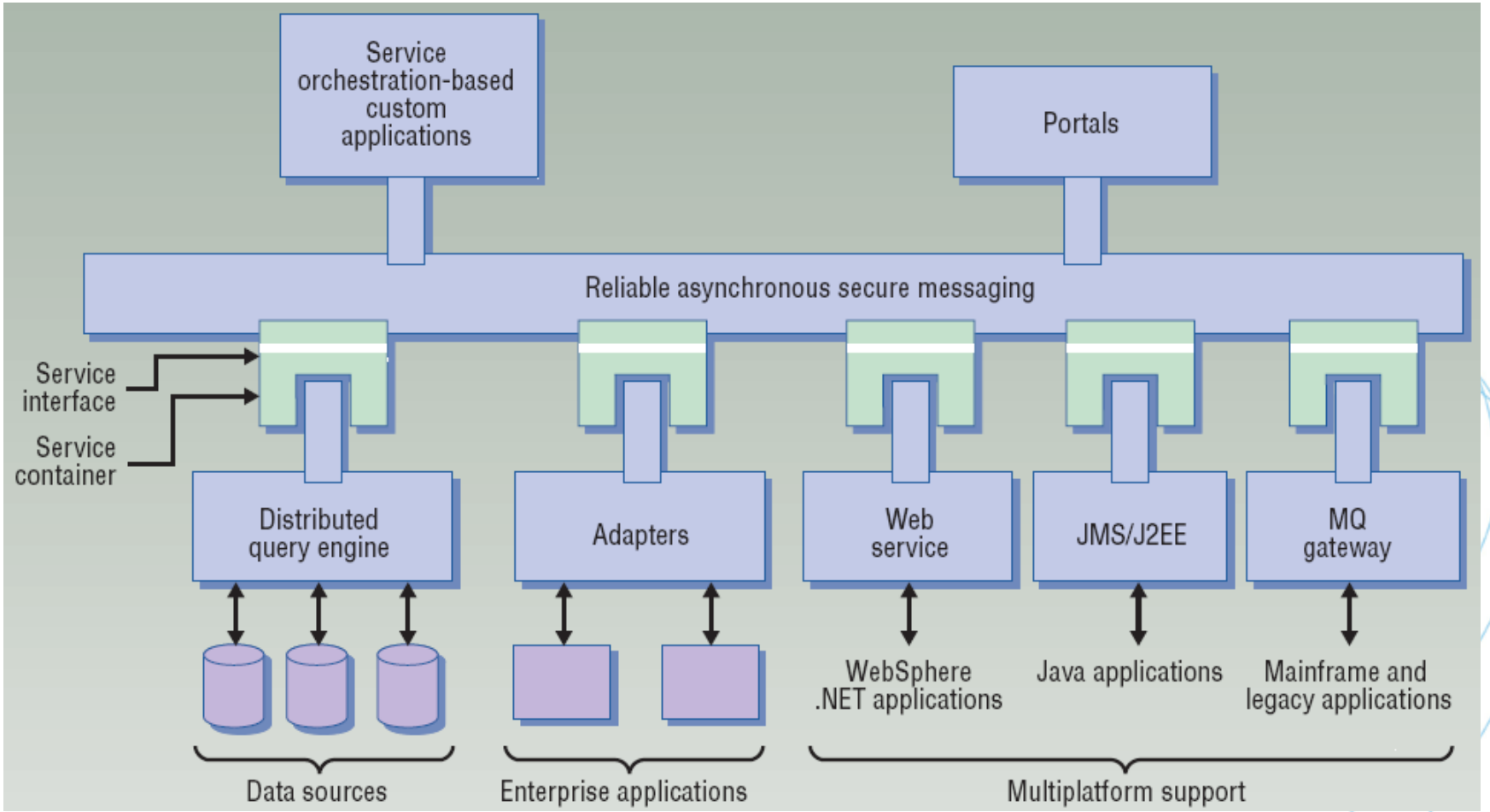

可管理资源的注册，发布管理接口
管理接口
引擎配置工具
对引擎的集中配置
Process
请求消息
远程管理控制台
JMX注册管理中心
SOAP引擎
适配器
适配器
适配器
适配器层
BPEL实例管理器
WSFL实例管理器
全局实例管理器
组合服务处理层
Java绑定调用
SOAP绑定调用
服务调用层
管理接口
组合服务处理引擎
组合服务的部署，请求消息侦听，SOAP消息解析
层次化的处理结构，分阶段解决不同的问题

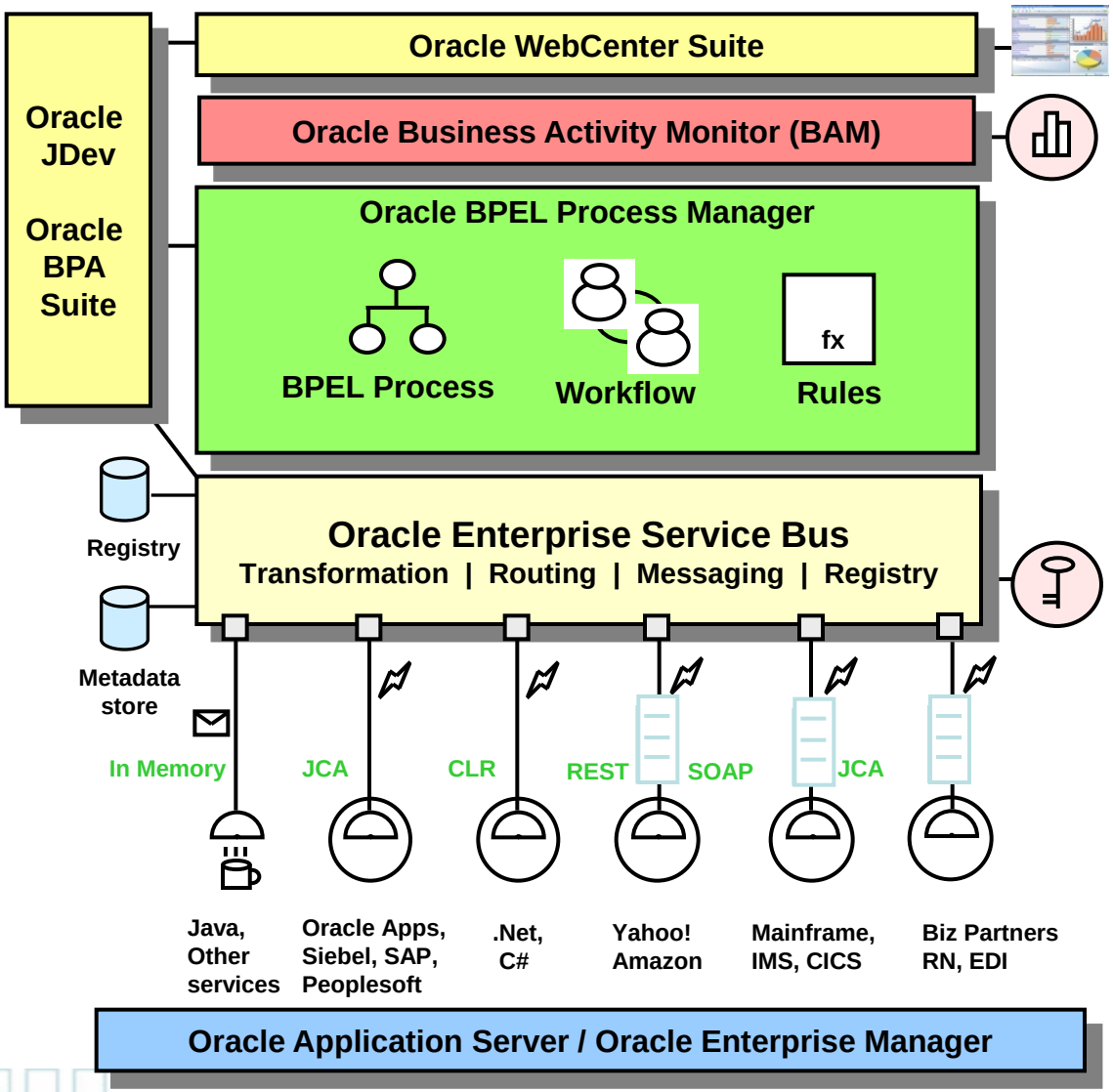
服务运行总线

- 主要的功能
  - 消息交换的路由控制与监控
  - 通信服务组件的协调
  - 服务的部署及版本管理
  - 冗余服务的调度使用
  - 共性服务



ESB的基本组成部分





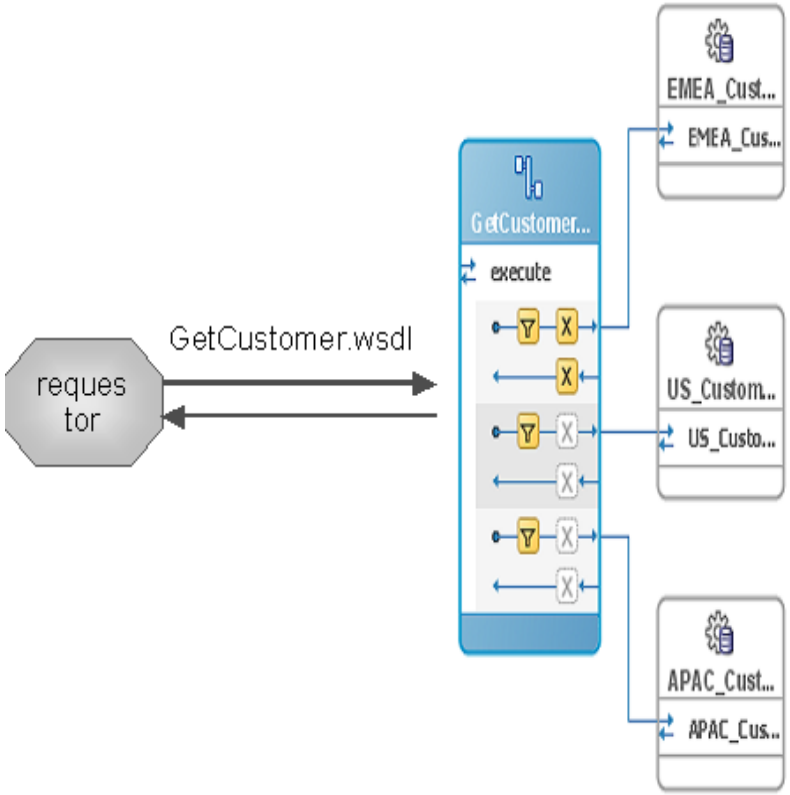
Oracle's SOA Platform

Key Features

- 100% BPEL Support
- Extensible Human Workflow
- Flexible Rules Integration
- Integrated Business Activity Monitor
- JCA/WSIF Binding Framework
- Integrated ESB, Registry, WSM
- SOA enabled user interaction layer
- Unified enterprise management
- Integrated development environment

Oracle Enterprise Service Bus (ESB)

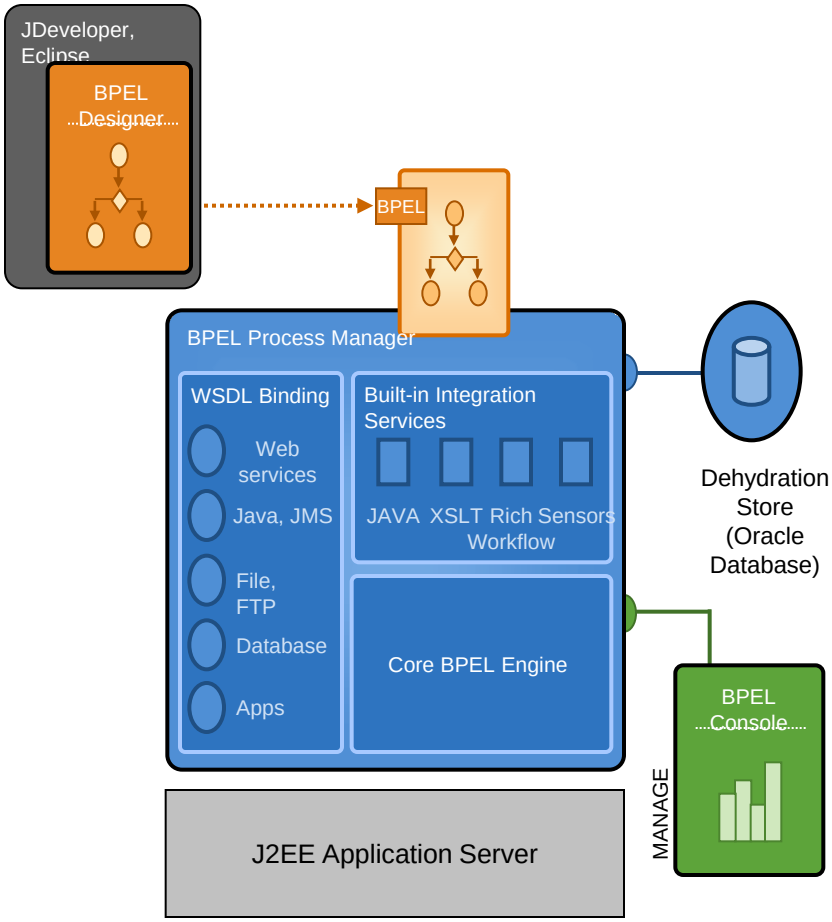
An ESB is a multi-protocol fabric to separate integration concerns from applications and business logic



- Virtualized Endpoints: **From resources to services.**
- Transform: **Convert data to target formats.**
- Route: **Reliable transport over a variety of protocols.**
- Standards Based: **XSLT, SOAP, XPATH, JMS, JCA, …**
- Hot Pluggable: **Java, J2EE, .NET, database, application server, …**

Oracle BPEL Process Manager

Enterprise-strength infrastructure for designing, deploying and managing BPEL business processes



- Comprehensive **BPEL implementation.**
- Easy-to-Use **Modeling tool**
- Reliable and Scalable **process engine.**
- Flexible **binding framework**
- Rich **management and monitoring**

提纲

- 运行支撑中间件
- 服务的安全管理
- 服务的事务管理
- 服务的动态演化

服务化软件的安全挑战

- 更加侧重应用层的安全
  - 未经授权的访问：消息内的信息被非授权者获悉，例如信用卡号
  - 未经授权的消息篡改：删除、修改、增加附加信息
  - 中间人攻击：在请求者和服务提供者之间进行消息拦截，实施破坏和攻击行为
  - 拒绝服务攻击：大量的明文/密文消息会占用大量信息资源，从而影响用户的正常使用
- 应用层安全的基本需求
  - 认证、授权、消息完整性、机密性、操作防御和不可抵赖

XML安全

- XML数据安全的特殊需求
  - 实际应用要求对文档的全部或部分内容(元素、属性等)加密/签字
  - XML文档是结构化的数据，加密/签字后的文档仍然是格式良好的XML文档
- 主要的XML安全规范
  - XML Signature
  - XML Encryption
  - XKMS
  - SAML
  - XACML

XML 签名和加密实例

```
<PurchaseOrderDocument>
<PurchaseOrder id="po1">
<SKU>12366</SKU>
<Quantity>17</SKU>
</PurchaseOrder>
<Signature
xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<Reference URI="#po1" />
</SignedInfo>
<SignatureValue>...</SignatureValue>
<KeyInfo>...</KeyInfo>
</Signature>
</PurchaseOrderDocument>
```

```
<Employee>
<Name>Dave Remy</Name>
<SocialSecurityNumber>
<EncryptedData id="socsecnum"
Type="http://www.w3.org/2000/09/xm
ldsig#content">
<EncryptionMethod Algorithm=". . ."
/>
<CipherData>
<CipherValue>. . .</CipherValue>
</CipherData>
</EncryptedData>
</SocialSecurityNumber>
</Employee>
```

XKMS规范

- **XKMS: 简化对PKI的集成，以及XML应用中数字证书的管理；使得认证、数字签名和加密服务更容易集成到应用中。**
  - **XML密钥信息服务规范(X-KISS): 注册服务、定位服务**
  - **XML密钥注册服务规范(X-KRSS): 验证服务**
    - 密钥的注册
    - 密钥的注销
    - 密钥的恢复
    - 密钥的重发

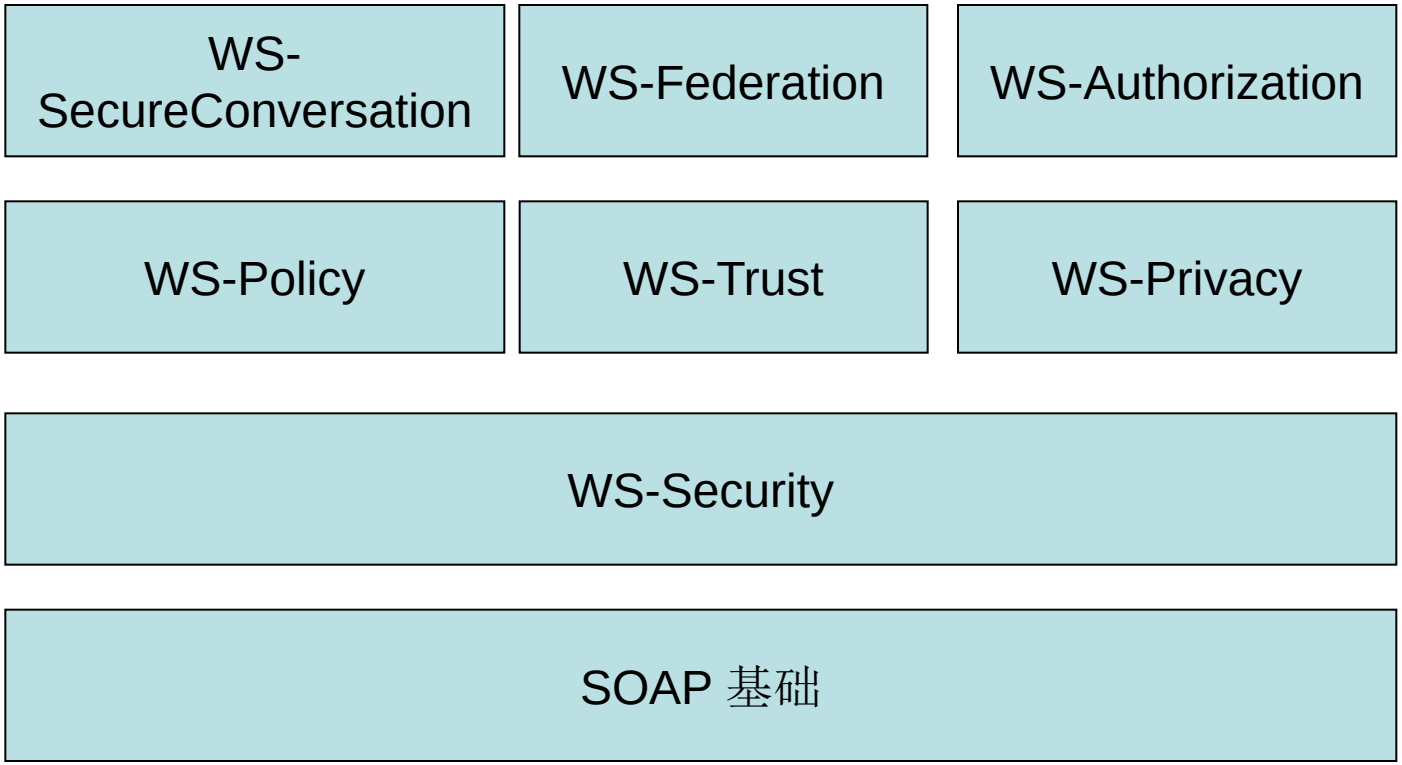
SAML规范

- **SAML：安全声明标记语言**
  - **用于单点登录中的用户认证：用户在多个应用中仅需提交一次身份标识，并可将身份标识从一个应用传送到另一个应用。**
  - **厂商中立，基于XML的标准框架，用于描述、交换与安全性相关的信息（声明/断言）**
  - **主要组件**
    - 断言：认证断言、属性断言和授权断言
    - 请求/响应协议
    - 绑定
    - 描述文件

XACML规范

- XML访问控制语言
  - 通用的访问控制策略语言，提供了管理授权决策的语法
  - 主要组件
    - 访问规则：谁能访问/访问什么/何时访问
    - 请求/响应语言：描述访问请求

Web服务安全



WS-*安全规范

- **WS-Security: SOAP**消息安全，实现消息内容的完整性和机密性
- **WS-Policy:** 描述安全性需求，如加密/签名算法、保密/隐私属性，以及这些信息符合绑定到**Web**服务
- **WS-Trust:**描述使 **Web** 服务能够安全地进行互操作的信任模型的框架，包括对于安全性令牌的请求与发放以及信任关系的管理
- **WS-Privacy:**描述**Web** 服务和请求者如何声明主体隐私权首选项和组织隐私权实践声明的模型
- **WS-SecureConversation:**描述如何管理和认证各方之间的消息交换，定义了能够提供安全通信的扩展
- **WS-Federation:** 用于跨不同信任域的身份标识、属性、认证和授权联邦
- **WS-Authorization:**如何管理和指定**Web**服务的访问策略

提纲

- 运行支撑中间件
- 服务的安全管理
- 服务的事务管理
- 服务的动态演化

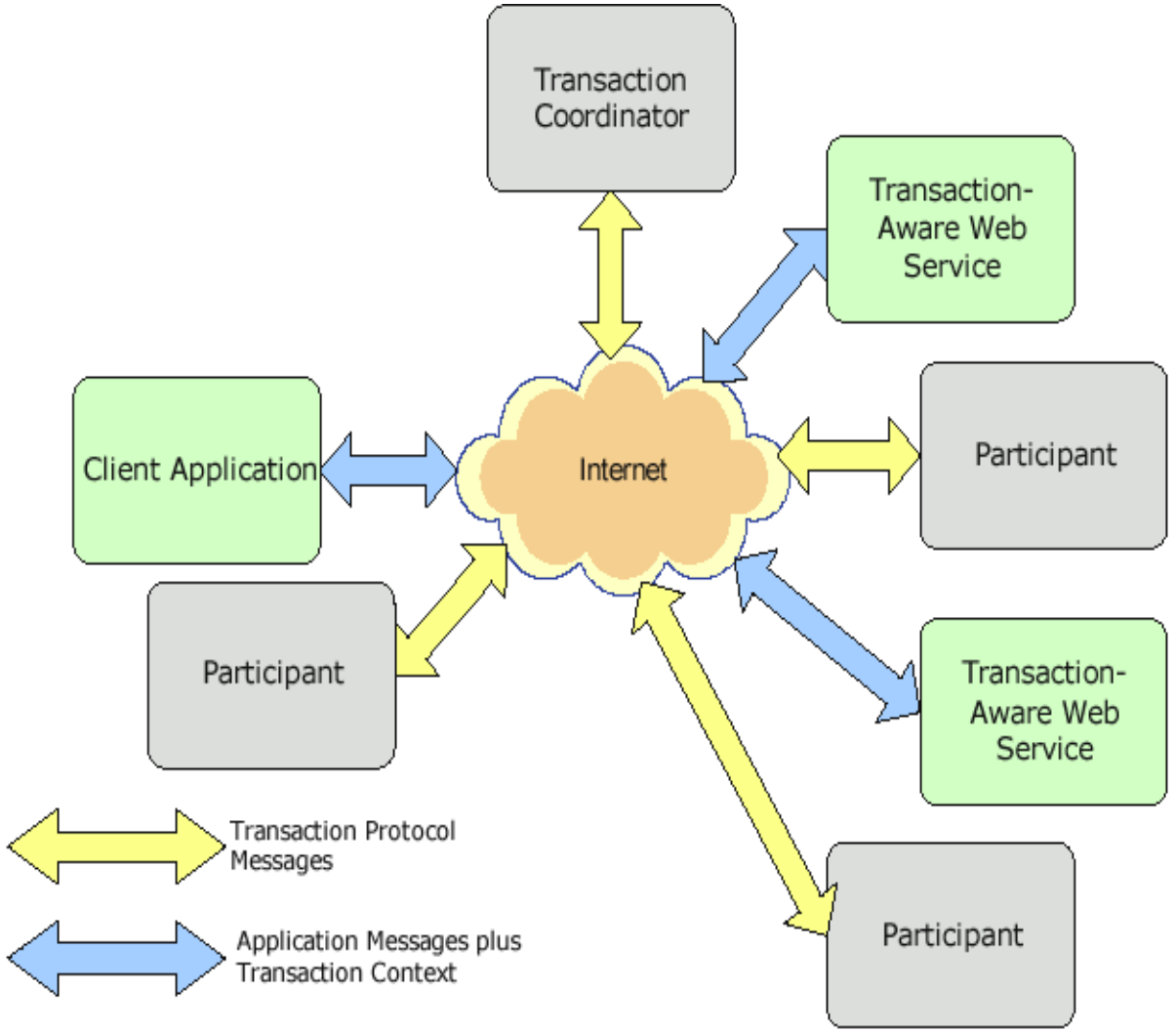
Web服务的事务处理

- 事务
  - 事务是一种用来确保应用程序中的所有参与者都能达到彼此已达成协定的输出结果的机制。作为一个完整工作单元的一系列操作，或者全部执行，或者失败。
  - 基本属性：ACID
    - Atomicity:事务所有的操作必须作为一个整体，要么全部成功，要么全部失败。
    - Consistency:事务的成功完成将使一个一致状态转变为另一个一致状态。
    - Isolation:执行事务时产生的中间状态，对其他事务是不可见的，即并发执行的各个事务间不会相互干扰。
    - Durability:一旦事务提交，那么它的影响将是持久的，无论发生任何机器和系统故障。
- Web服务中的事务
  - 涉及到多个参与方
  - 持续时间长
- ACID属性过于严格

OASIS规范

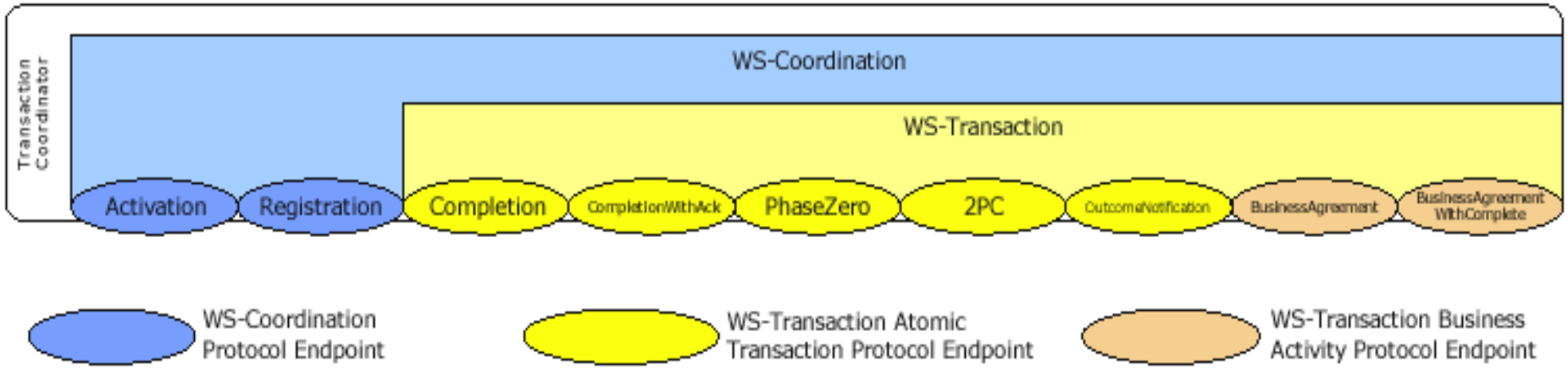
- **Web Service Transaction: WS-Transaction**
  - **WS-Coordination v1.2**
  - **WS-AtomicTransaction v1.2**
  - **WS-BusinessActivity v1.2**

Web服务的事务处理



WS-Cooridnation

- **WS-Coordination:** 提供描述分布式应用协同处理的可扩展框架，利用协调器（**Coordinator**）和一系列协调协议（**Coordination Protocols**）来协调活动，使得参与者就分布式活动的输出能够达成一致。协调协议支持多种活动，包括简单的短期操作和复杂的长时间运行的业务活动。



WS-Coordination

- 目标：该框架使参与者对分布式活动的结果达成一致的协议。协调协议可以在框架中定义，框架可以容纳很多活动，包括简单、短暂、操作的协议和复杂、长期、交易活动的协议。包括下列服务
  - 一个激活服务
  - 一个注册服务
  - 一个协调类型

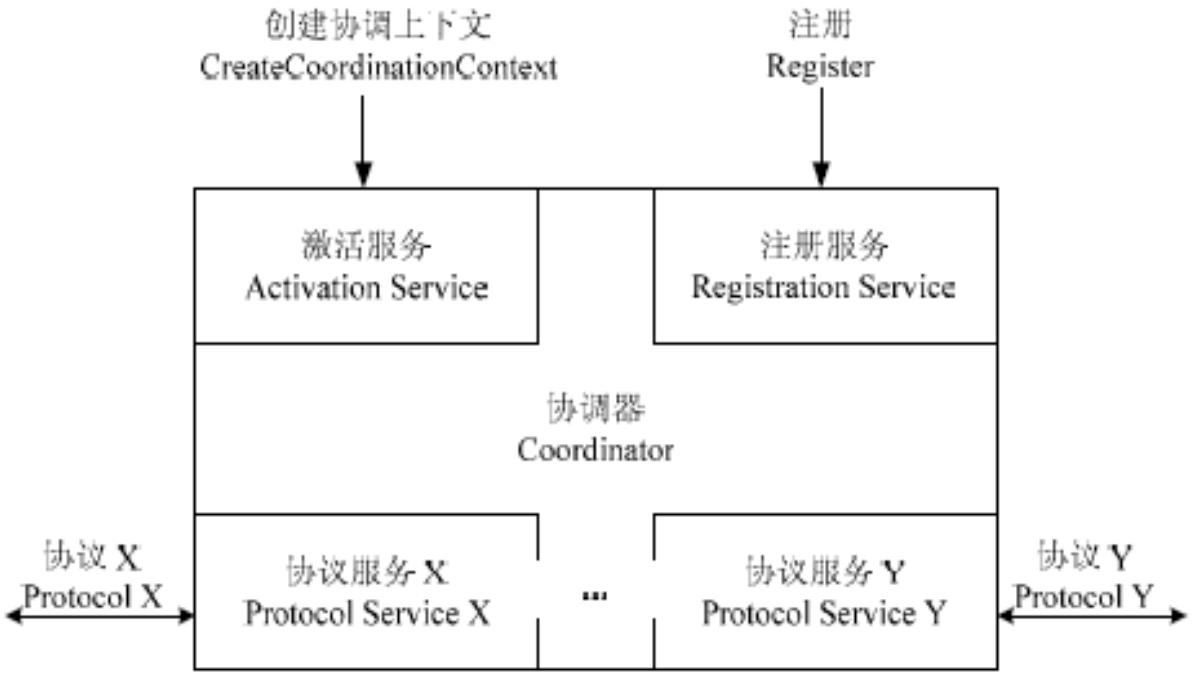
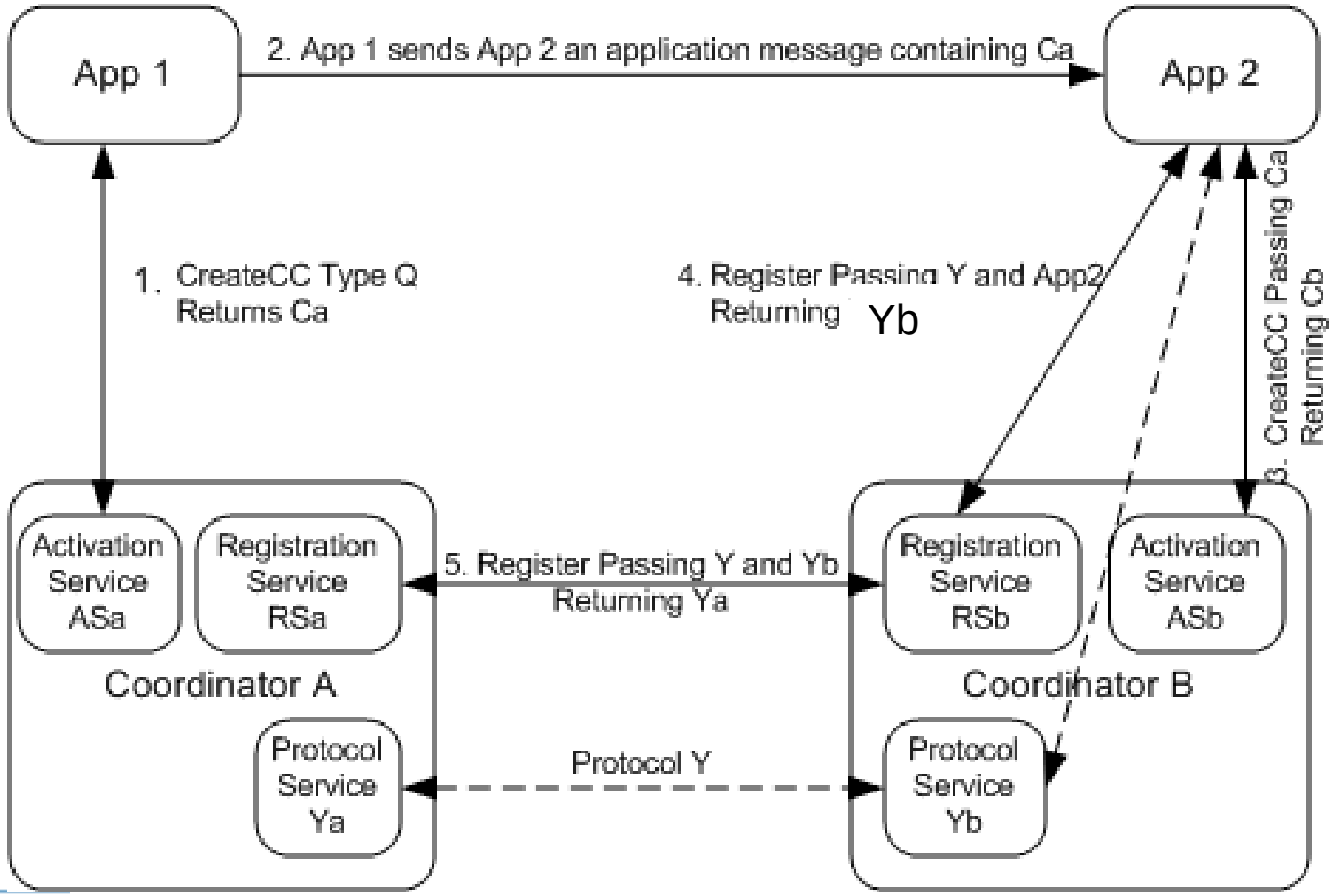


图 1　WS-Coordination 的协调服务模型

WS-Coordination(续)

- 激活服务：协调者提供激活服务，定义一个**CreateCoordinationContext**操作，允许创建**CoordinationContext**。
- 注册服务：协调者提供注册服务，允许参与者注册自己的地址、协调协议等参与到协调中。参与者可以通过发出多个注册操作，注册到多个协调协议



WS-AtomicTransaction

- **WS-AtomicTransaction：原子事务(Atomic Transaction，AT)具有“全做或全不做（all or nothing）”的特性，也就是说要么所有的操作全部成功，要么所有的操作失败并终止。WS-AtomicTransaction 用于协调持续时间短并且只能在有限的信任域内执行的原子事务。**
- **给出了原子事务协调类型的定义，将与 WS-Coordination 中的可扩展协调框架一起使用。**

WS-AtomicTransaction

- **WS-AtomicTransaction 为原子事务定义了下列协调协议：**
  - **（1）完成（Completion）：用于提交或放弃原子事务的情形。**
  - **（2）两阶段提交（2PC）：两阶段提交协议是一个协调有多个参与者参与的原子事务的协议。**

两阶段提交协议是指参与者例如资源管理者注册，因此在所有资源管理者中，协调者可以管理提交放弃决定，若包含多于一个的两阶段提交协议的参与者，先执行PhaseOne后执行PhaseTwo。若只有一个2PC参与者，OnePhaseCommit被用来代表参与者的提交放弃决定。

WS-AtomicTransaction

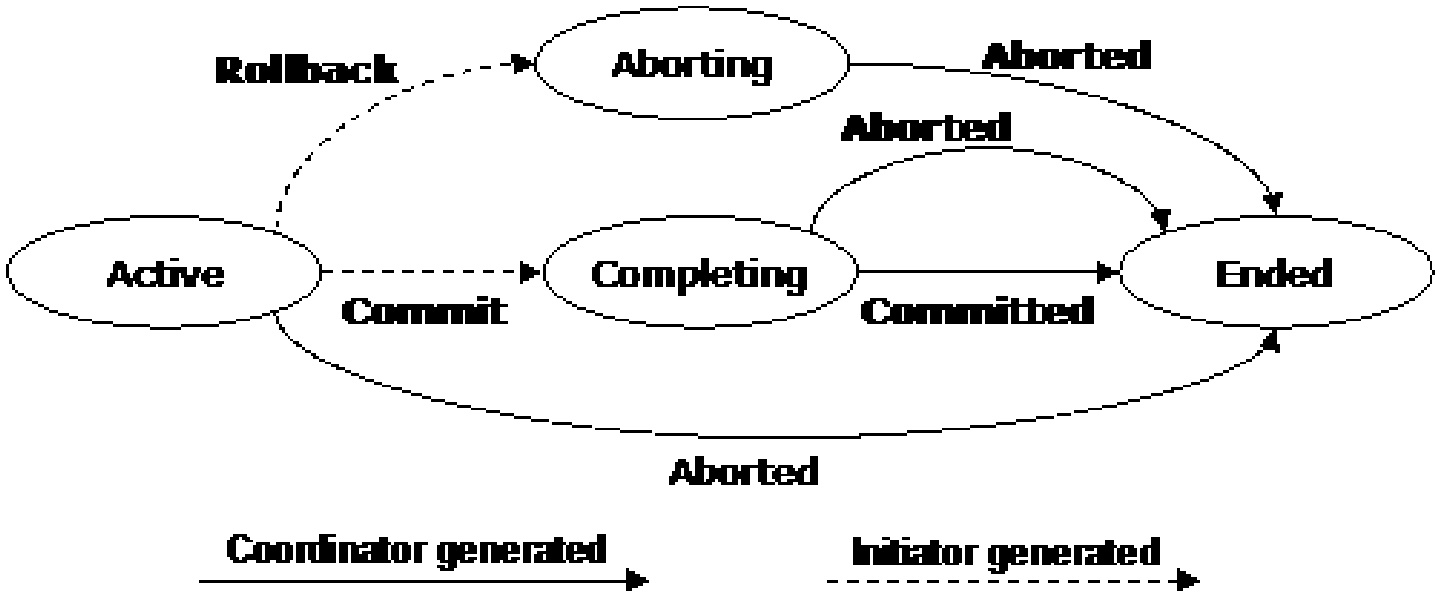
- **两种2PC 协议：**

  A.易失的两阶段提交（volatile 2PC）：用于那些管理着易失资源（例如缓存、内存）的参与者。

  B.持久的两阶段提交（durable 2PC）：用于那些管理着持久的资源（例如磁盘、数据库）的参与者。

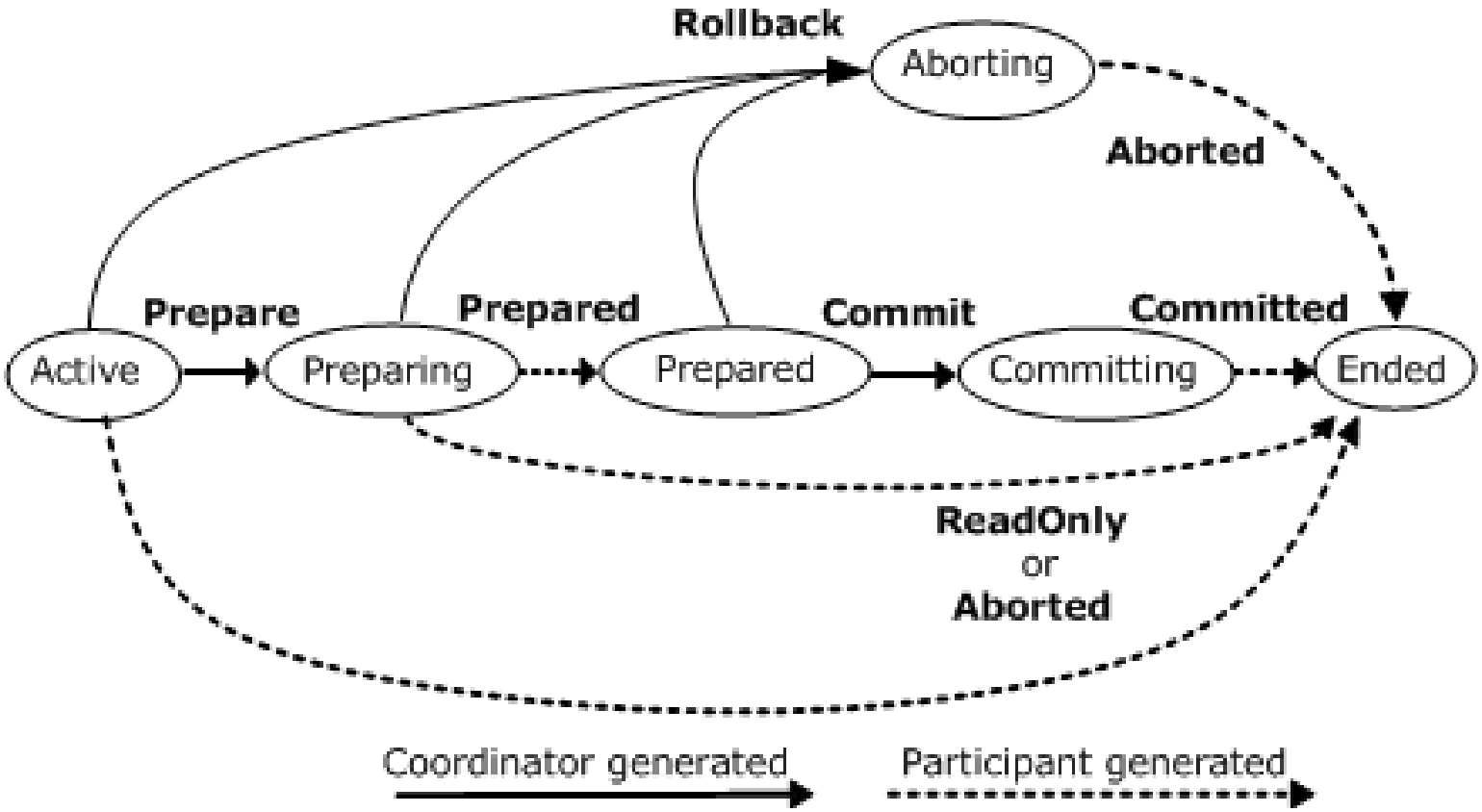
WS-AtomicTransaction

- **WS-AtomicTransaction**
  - **Completion protocol**
  - **Two-phase commit protocol**



Completion协议

Two-phase commit protocol



WS-BusinessActivity

- 业务活动(Business Activity, BA)会长时间使用很多资源，并且涉及相当多的原子事务。WS-BusinessActivity 引入了故障处理机制和补偿处理机制，以将前面已经完成的业务活动的效果恢复成原来的样子。在长期运行的业务活动完成前，嵌入的原子事务的动作就被提交并且成为可见的，万一长期运行的业务活动失败了，则需要对这些原子事务的效果进行补偿。

WS-BusinessActivity

WS-BusinessActivity 支持两种协调协议，指示参与者如何在业务活动中进行活动：

（1）参与者完成业务协定（BusinessAgreementWithParticipantCompletion）：参与者自己就知道它什么时候可以完成它在业务活动中的所有工作。

（2）协调器完成业务协定（BusinessAgreementWithCoordinatorCompletion）：参与者要依赖业务活动的协调器来告诉它什么时候已经接收完业务活动中的所有请求。

WS-BusinessActivity

- 面向Web服务应用中的长事务管理
  - 两种协调类型
    - AtomicOutcome: 每一个参与者都close或者compensate
    - MixedOutcome：不保证每个参与者都close或compensate
  - 两类协议

BusinessAgreementWithParticipantCompletion: A participant registers for this protocol with its coordinator, so that its coordinator can manage it. A participant knows when it has completed all work for a business activity.

BusinessAgreementWithCoordinatorCompletion: A participant registers for this protocol with its coordinator, so that its coordinator can manage it. A participant relies on its coordinator to tell it when it has received all requests to perform work within the business activity.

BusinessAgreementWithParticipantCompletion

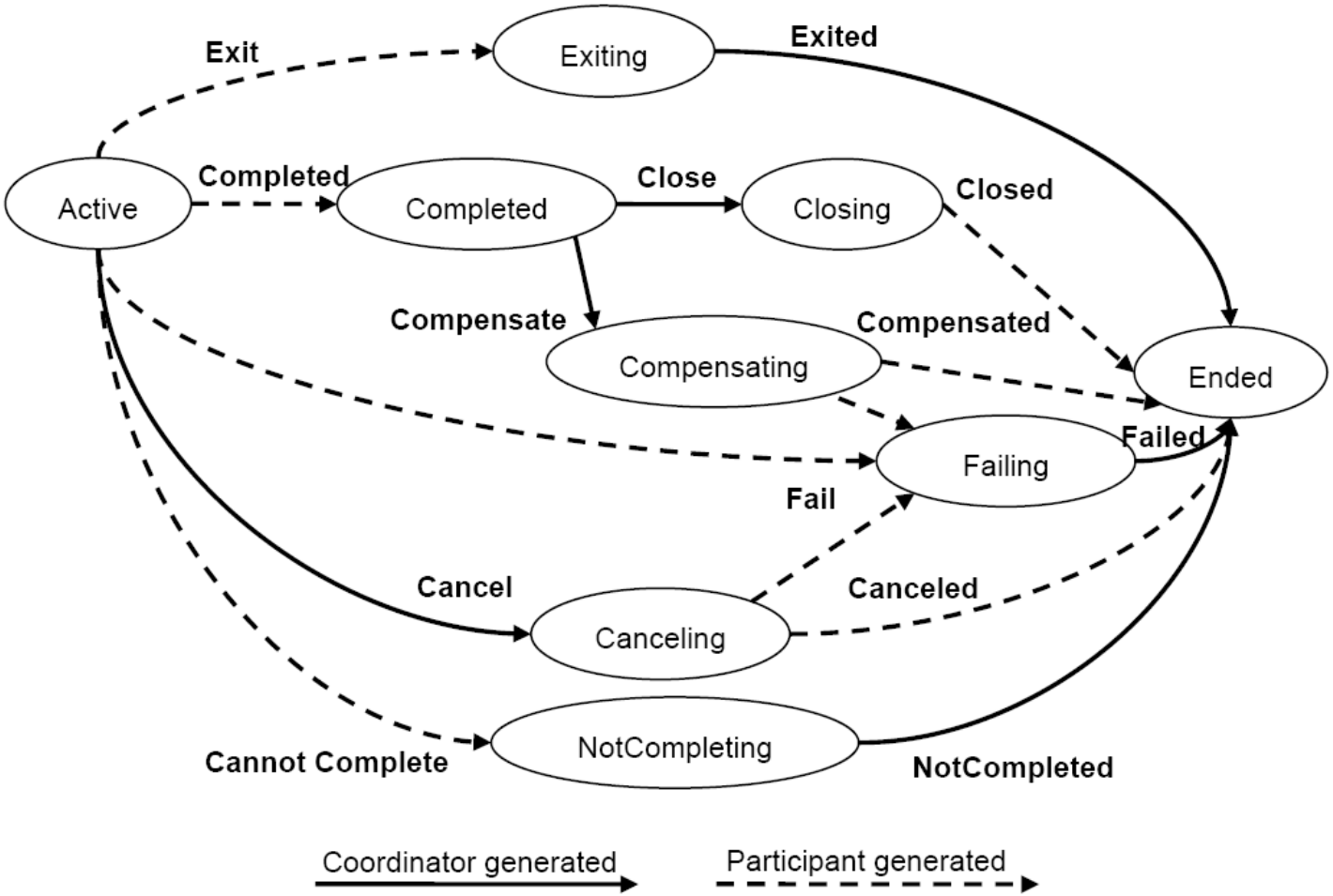


Figure 1: BusinessAgreementWithParticipantCompletion abstract state diagram

BusinessAgreementWithCoordinatorCompletion

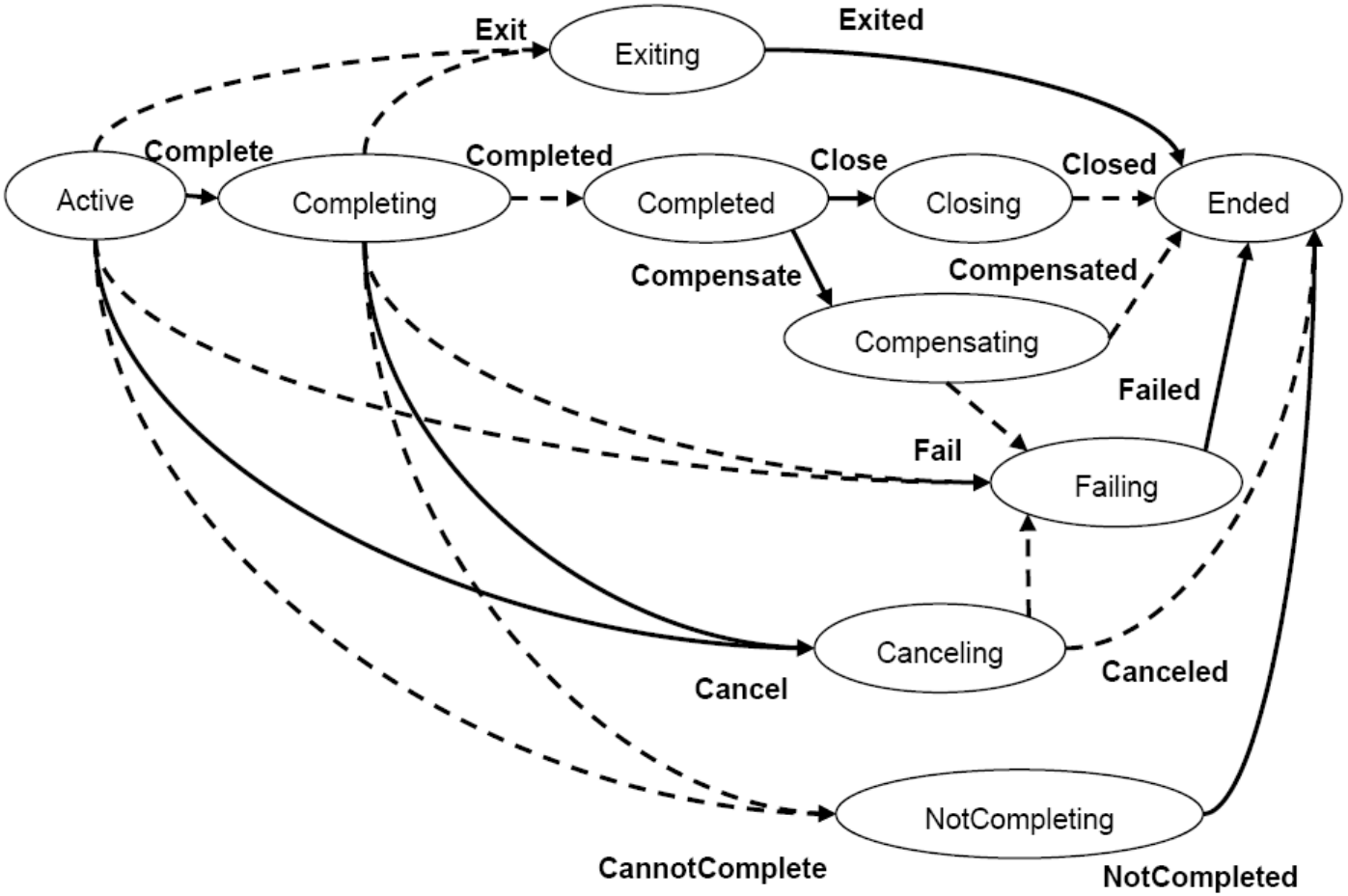


Figure 2: BusinessAgreementWithCoordinatorCompletion abstract state diagram

提纲

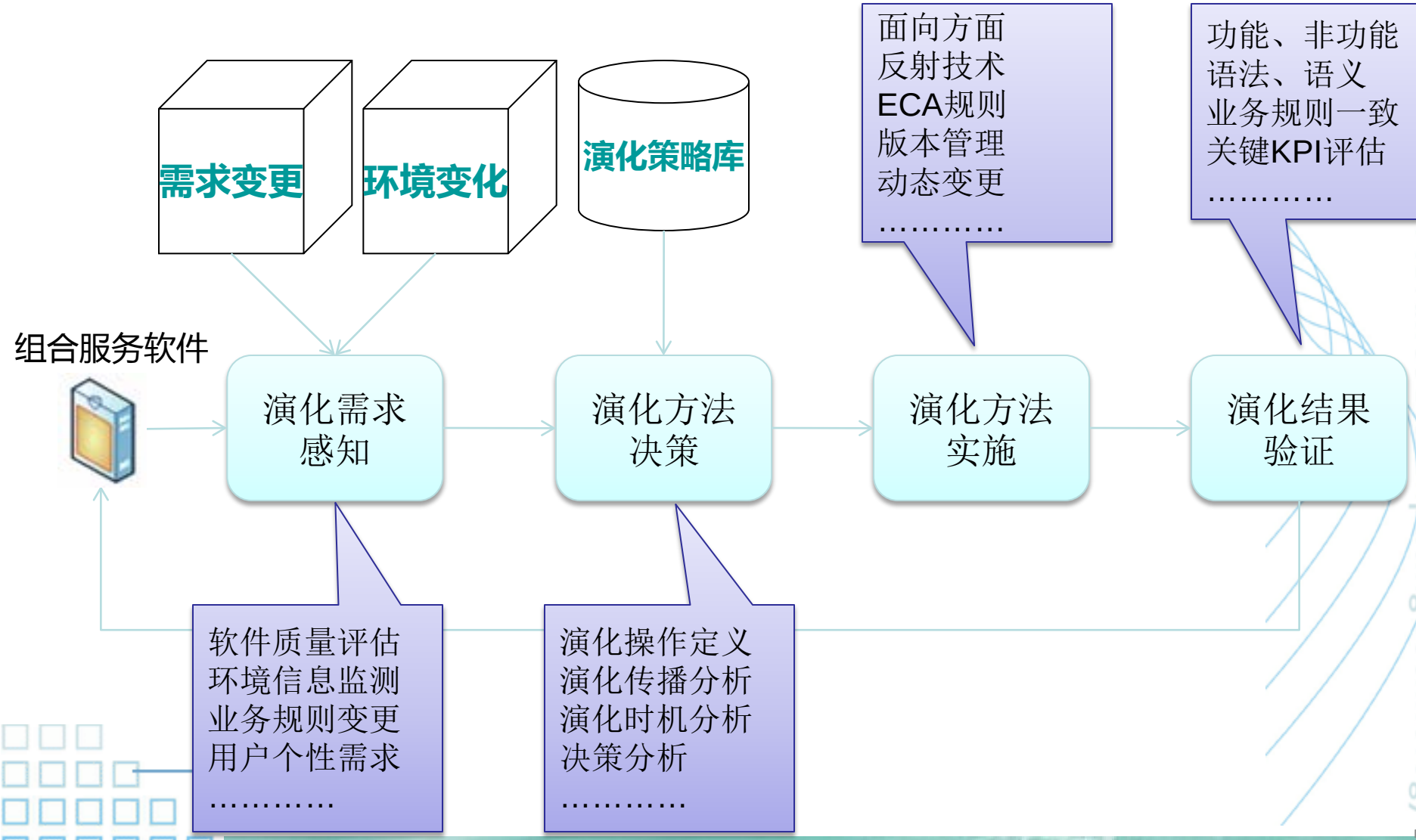
- 运行支撑中间件
- 服务的安全管理
- 服务的事务管理
- 服务的动态演化

网络化软件演化的意义

- **Internet**的自治性和不可控性使得网络化软件难以保证稳定的服务质量
- 市场竞争日益激烈，业务规则和客户需求都在快速变化之中，要求网络化软件具备更强的灵活性，实现“随需应变”的商业模式

主要表现在其实体元素数目的可变性、结构关系的可调节性和结构形态的动态可配置性上

网络化软件演化


需求变更
环境变化
演化策略库
面向方面
反射技术
ECA规则
版本管理
动态变更
……
功能、非功能
语法、语义
业务规则一致
关键KPI评估
……
组合服务软件
演化需求
感知
演化方法
决策
演化方法
实施
演化结果
验证
软件质量评估
环境信息监测
业务规则变更
用户个性需求
……
演化操作定义
演化传播分析
演化时机分析
决策分析
……

服务演化

- 服务演化：服务通过一致和清晰的一系列变化而进行的连续开发过程。
- 服务演化的表达：生命周期中不同版本的创建和消亡。
- 版本必须彼此一致，这样能够使得服务设计人员能够追踪服务的修改以及对服务本身的影响结果。
- 需要可靠的版本策略来支持服务的多个版本。

服务演化的驱动力

Innovation in products & processes

Extended products

Globalization

Corporate & business strategy decisions => Redesign of business models

Technological advancements

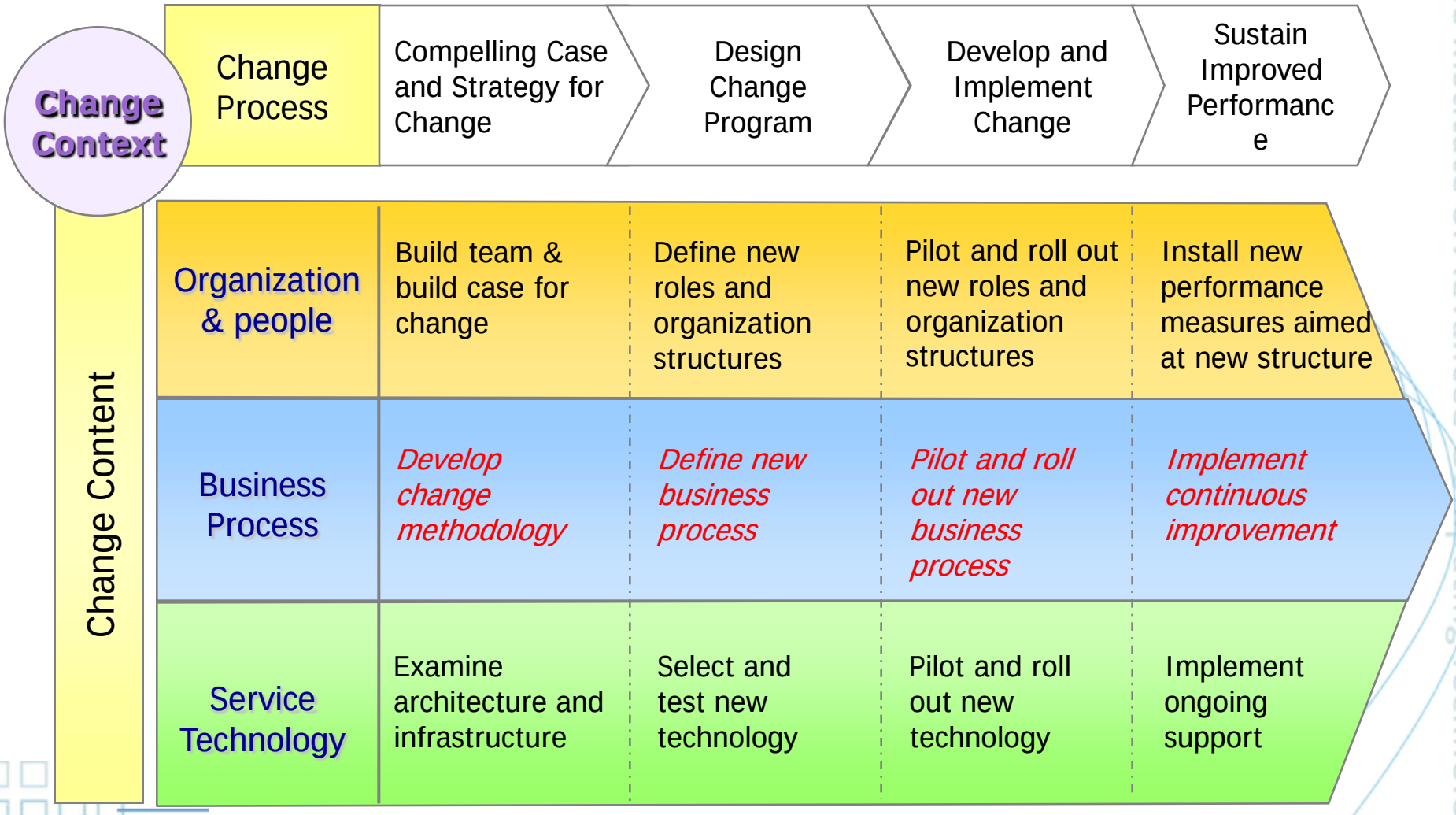
Regulation

Cross-organizational service networks

New market opportunities

Continuous Need for Service Evolution!

演化: 源于变化



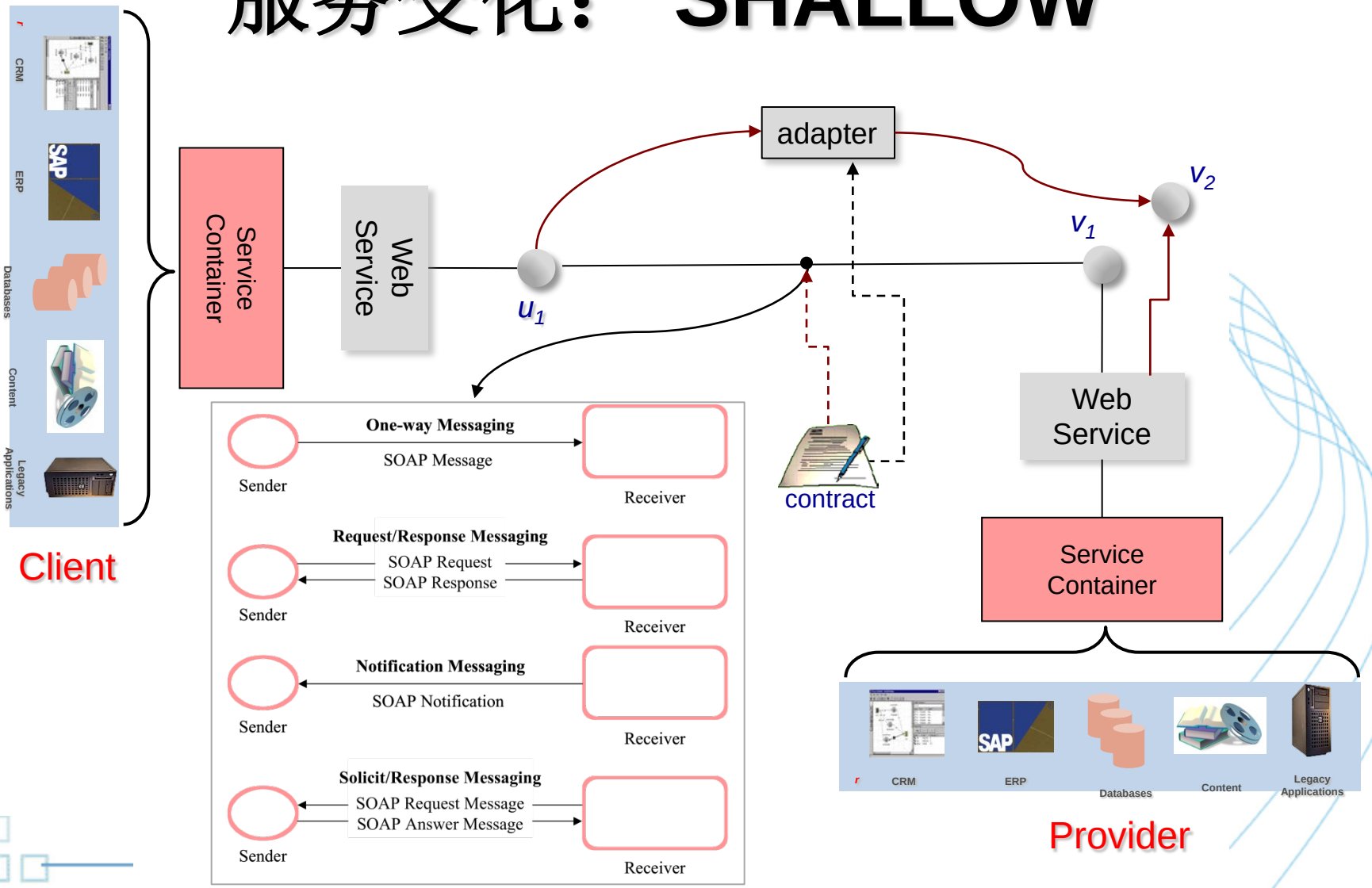
服务变化的分类（一）：特点

- 结构性变化：**signature & interfaces**
- 业务协议变化：**external messaging behaviour of services**
- 策略引发的变化：**changes in policy assertions, business rules, & regulatory compliance**
- 操作行为变化：**effects of changing the meaning & behaviour of service operations-*operational semantics***

服务变化的分类（二）：效果

- Shallow Change (小规模的增量式变化)：变化的效果仅仅影响服务本身或者服务的用户
  - **Structural changes (service types, messages, and operations)**
  - **Business protocol changes (external messaging behaviour of services)**
- Deep Change (大规模的革新式变化)：**端到端的业务流程**
  - **策略引发的变化：changes in policy assertions, business rules**
  - **操作行为语义的变化：effects of changing the meaning & behaviour of service operations – operational semantics**

服务变化： SHALLOW



Shallow Change

- **Structural changes**
  - **Compatibility:变更后的服务和/或客户端能够继续处理旧版本的调用消息**
    - 向后兼容: Backward compatibility
    - 向前兼容: Forward compatibility
- **Business protocol changes**
  - **Static protocol evolution：**
    - a set of change operations to allow the incremental modification of an existing protocol
  - **Dynamic protocol evolution**
    - requires a protocol to migrate active instances running under an old protocol version to meet the new protocol requirements.

Deep change: regulatory compliance

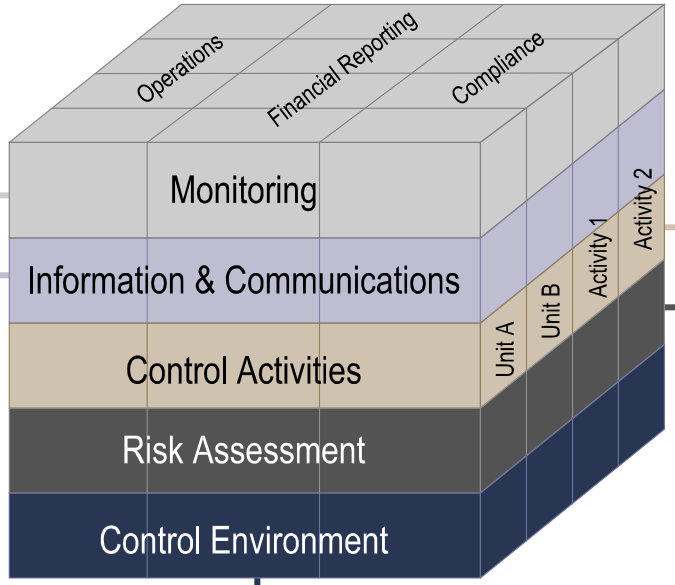
COSO is a standard ICT framework providing guidance on organizational governance, business ethics, internal control, enterprise risk management, fraud, & financial reporting.

Monitoring

- Assessment of a control system's performance over time
- Combination of ongoing and separate evaluation
- Management and supervisory activities
- Internal audit activities

Information & Communication

- Pertinent information identified, captured and communicated in a timely manner
- Access to internally and externally generated information
- Flow of information that allows for successful control actions from instructions on responsibilities to summary of findings for management action



Control Activities

- Policies/procedures that ensure management directives are carried out
- Range of activities including approvals, authorizations, verifications, recommendations, performance reviews, asset security and segregation of duties

Control Environment

- Sets tone of organization, influencing control consciousness of its people
- Factors include integrity, ethical values, competence, authority, responsibility, organization structure, HR policies and IT control environment
- Foundation for all other components of control

Risk Assessment

- Risk assessment is the identification and analysis of relevant risks to achieving the entity's objectives – forming the basis for determining control activities

DEEP CHANGE:非功能性变化

- **Deal with non-functional changes & *guarantee end-to-end consistency*. Typical KPIS include:**
  - Delivery performance
  - Fill rates
  - Order fulfilment
  - Production efficiency
  - Logistic costs to sales
  - Inventory days of supply
  - Quality thresholds
  - Service velocity
  - Service volumes
  - Transaction volumes
  - Cost baseline
  - Six - Sigma
  - Yield mgt
  - Asset turn over ratio
- ***Translate* into SLAs, *predict* impact of KPI changes to process structure & *detect* KPI violations during run-time.**

DEEP CHANGE:非功能性变化

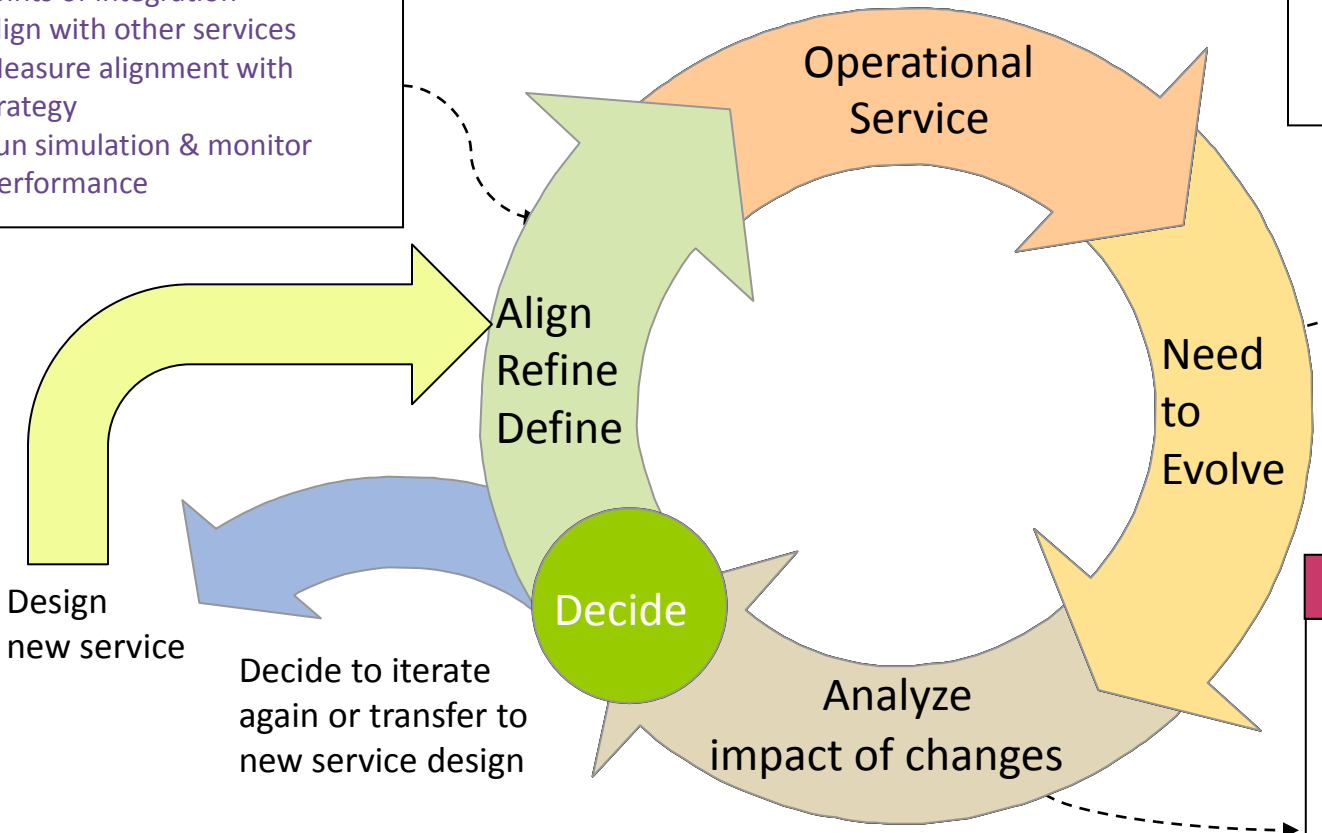
- **Innovation driven by an entire supply-chain may result in introducing *new services*:**
  - **Determine that *overall KPI meets internal company KPIs***
  - **Find out which *public process fragments* are affected by introduction of new service(s).**

- **Innovation driven by *a firm within the supply-chain* which may introduce new services:**
  - **Determine that *potential change of internal KPI meets overall KPI***
  - **Find out which *public process fragments* are affected by introduction of new service.**

CHANGE –ORIENTED SERVICE LIFE CYCLE

Broader Change Context

- Test service interfaces
- Define/refine interfaces & points of integration
- Align with other services
- Measure alignment with strategy
- Run simulation & monitor performance



Understand Change Logic

- Determine causes
- Scope extend of change
- Identify services in-scope
- Collect detailed service metrics

Change Impact Analysis

- Analyze changes
- Determine functionality
- Determine changes to inter-dependent processes
- Determine whether KPIs are satisfied
- Determine compliance with regulations, business-rules
- Estimate costs (ROI)

典型的问题

- ***Service flow problems:*** **problems with the logical completeness of a service upgrade, problems with sequencing and duplication of activities, decision-making problems and lack of service measures.**
- ***Service control problems:*** **problems with policies and business rules and problems with external services.**
- ***Overlapping services functionality:*** **a service-in-scope may (partially) share identical business logic and rules with other related services.**
- ***Conflicting services functionality:*** **(including bottlenecks / constraint violation in the service value stream)**
- ***Service input and output problems:*** **e.g., QoS i/p or o/p is low, and timeliness i/p or o/p problems where the needed inputs/outputs are not produced when they are needed.**

服务的演化

- 为什么要演化：变化！
  - 现有功能不满足新的业务需求
  - 系统运行故障（设计bug、环境影响）
  - 开发运行一体化
- 演化什么
  - 服务组件增加、删除和替换
  - 逻辑依赖和数据依赖的调整
- 演化的目标
  - 提高组合服务的对需求变更和环境变化的适应能力
- 特点: 动态、在线
  - 允许不停机的情况下对组合服务及其运行实例进行演化

面向流程服务组合的演化问题

- 演化的现实需求
  - 功能层面：
    - 灵活、可配置、[illegible]性
  - [illegible]功能层面：
    - 持续、稳定的提[illegible]户需求的服务质量

组合服务适应环境变化和需求变更的要求

组合服务中刚性的业务流程结构

- 业务流程的技术局限

手工开发和配置，效率[illegible]言难以修改，灵活性差，维护成本高

因此，需要一种方法和技术来支持服务组合动态适应环境变化和需求变更，降低运维成本，保障软件质量，进一步提高开发效率。

关键问题

- 组合服务演化模型
  - 组合服务描述模型
  - 变更操作集合
- 感知演化需求
  - 确定演化目标
  - 分析演化影响
- 演化方法实施
  - 组合服务的动态重配置
  - 正确性验证
  - 版本管理
- 运行实例处理
  - 实例迁移
  - 状态补偿

组合服务演化

环境变化

需求变更

1、监测评估

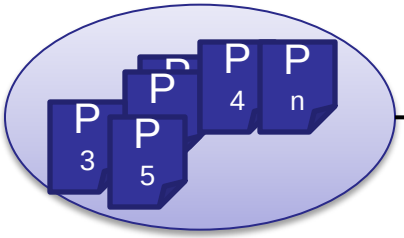
组合服务约束

流程结构

QoS质量

……

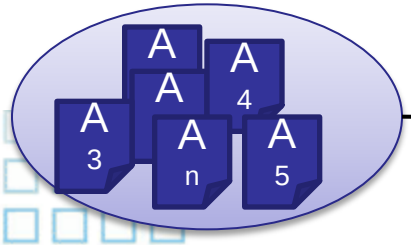
Why：什么变化会导致组合服务演化，如何检测这种变化，这种变化在组合服务的原始约束下如何产生演化目标。



演化策略

2、演化方法决策

What：实现同一个演化目标的方法有多个，如何在成本、影响范围等因素的制约下决策出最优的演化动作集。



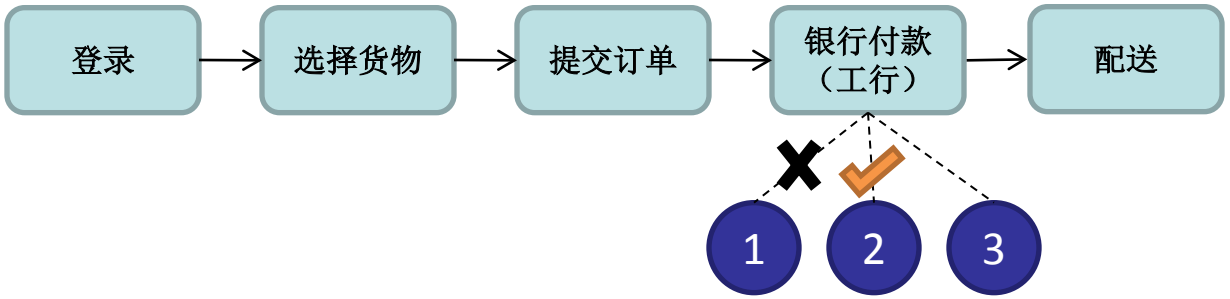
演化机制

3、演化动作执行

How：不同的演化动作需要不同的执行机制，如何实际的执行演化动作集，并且保证组合服务原有的性质。

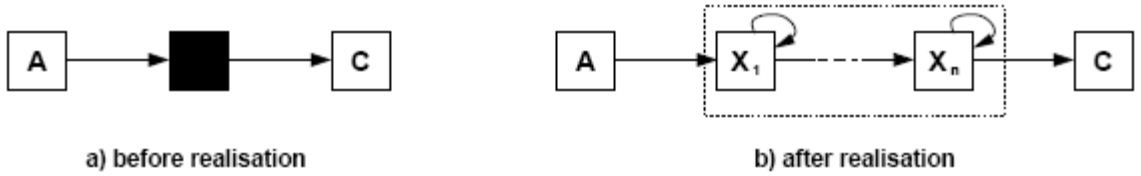
场景举例

- 对组合服务的一个实例（Momentary或ad-hoc change）
  - 业务流程的定义不改变，针对某些特定用户的个性需求或运行bug临时的改变部分运行实例
    - 组件服务失效的副本替换



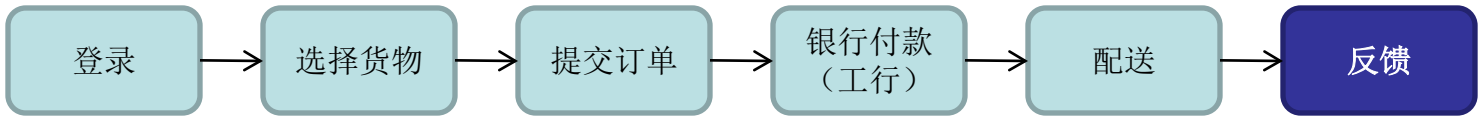
    - 对于已经选择了货物的用户可以直接提交订单

    - 流程结构迟建模、服务动态绑定

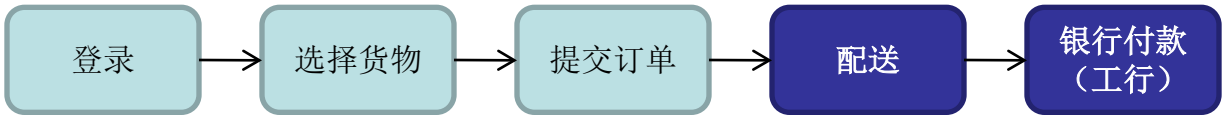


场景举例

- 对组合服务定义（**Evolutionary change**）
  - 增加新功能
    - B2C购物的业务流程增加一个用户反馈功能



  - 改变现有功能
    - B2C购物的业务流程中调整部分结构逻辑



    - B2C购物的业务流程中替换了某个组件

Q&A