



Service Oriented Architecture

目录

- 背景
- SOA概念
- SOA构建中的三个要素
- 注意的问题

背景

- 面向服务的社会
- SOA & 构件技术
- SOA & 传统架构
- EAI

背景一面向服务的社会

- 现代社会中我们是不是要亲自做任何事情？
- **NO**，只需表达意图，而不用说明特点的执行过程
- 显示的社会场景提供了**SOA**架构设计的灵感，思考一下面向过程和面向对象的架构设计演化。

背景一 工匠到服务提供者

- Our society has become what it is today through the forces of
 - Specialization
 - Standardization
 - Scalability
- It is now almost exclusively “service” oriented
 - Transportation
 - Telecommunication
 - Retail
 - Healthcare
 - Financial services
 - ...

背景—服务的属性

- Well defined, easy-to-use, somewhat **standardized interface**
- **Self-contained** with no visible dependencies to other services
- (almost) **Always available** but idle until requests come
- Easily accessible and **usable readily**, no “integration” required
- **Coarse grain**
- **Independent of consumer context**,
 - but a service can have a context
- New services can be offered by **combining existing services**

背景—服务接口

- Non proprietary
 - All service providers offer somewhat the same interface
- Highly Polymorphic
 - Intent is enough
- Implementation can be changed in ways that do not break all the service consumers
 - Real world services interact with thousands of consumers
 - Service providers cannot afford to “break” the context of their consumers

背景—意图和提供

- Service consumer expresses “intent”
- Service providers define “offers”
- Sometimes a mediator will:
 - Find the best offer matching an intent
 - Advertise an offer in different ways such that it matches different intent
- Request / Response is just a very particular case of an Intent / Offer protocol

背景—服务目录

- Our society could not be “service oriented” without the “Yellow Pages”
- Directories and addressing mechanisms are at the center of our service oriented society
- Imagine
 - Being able to reach a service just by using longitude and latitude coordinates as an addressing mechanism?
 - Only being able to use a service if you can remember its location, phone or fax number?

背景—SOA&构件技术

- 满足不同用户的定制要求
- 各个功能模块独立升级，实现，不影响整体
- 提供标准的构件接口和框架，软件开发变成了基于构件的组合，这种方法相对于传统的面向机器、面向数据、面向过程、面向功能、面向数据流的方法是更高层次上的复用
- 在面向对象的体系种，对象是最基本的构件单元
- **SOA**是一种基于对象的构件计算模型，它将不同的功能单元通过预先定义好的接口和契约联系起来

背景—SOA&传统架构

- 软件编程理念经历了随意编程、面向结构、面向对象、面向构件，功能重用主要是通过源代码级的封装、继承等特性来实现
- **Web**服务则是通过基于动态目标代码级的封装、继承，及元数据的自描述技术等来实现的

背景—SOA&传统架构

- 传统软件架构为了保持自由开放的特性，对软件的执行没有限制，一旦软件开始执行，就会获得相应的系统资源，并且认为软件的每个组成部分都是可靠的
- 在SOA中，软件的执行过程中不再像过去那样将要在本地安装所有的部分，而是动态地从网上根据需要请求服务，软件的各个部分分散在Internet中，需要时进行动态地组合。这种分散的软件结构，将安全责任由传统的软件使用者全部负担，转换为由软件的使用者和提供者共同负担，因而明确了责任，降低了单个用户的安全风险

背景—EAI

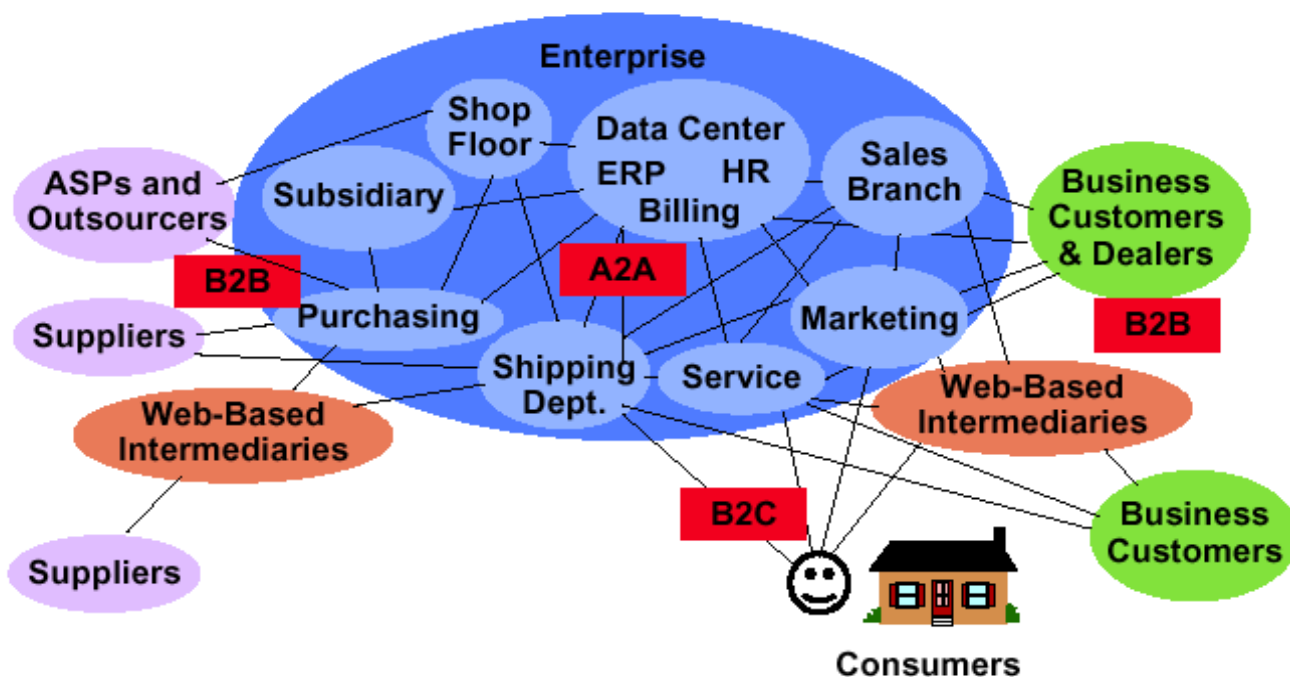
企业面临的问题

- 信息孤岛
- 系统间的业务协作由手工完成
- 重复信息与信息不一致
- 缺乏完整的信息全貌
- 系统过于庞大而难于调整
- 无法面对新的改变而不得不开发新的系统
- 多系统的局面无法避免
- 如何协同多系统？

背景—EAI

- **Enterprise Application Integration**，即企业应用整合，仅指企业内部不同应用系统之间的互连，以期通过应用整合实现数据在多个系统之间的同步和共享。（狭义）
- 业务整合(**Business Integration**)的范畴，业务整合相对**EAI**来说是一个更宽泛的概念，它将应用整合进一步拓展到业务流程整合的级别。业务整合不仅要提供底层应用支撑系统之间的互连，同时要实现存在于企业内部应用与应用之间，本企业和其他合作伙伴之间的端到端的业务流程的管理，它包括应用整合，**B2B**整合，自动化业务流程管理，人工流程管理，企业门户以及对所有应用系统和流程的管理和监控等方方面面（广义）。

企业现状



Gartner

Source: Gartner Research

一家典型的大型企业平均拥有49个应用系统，33%的IT预算是花在系统的集成上(让系统之间交换信息)。

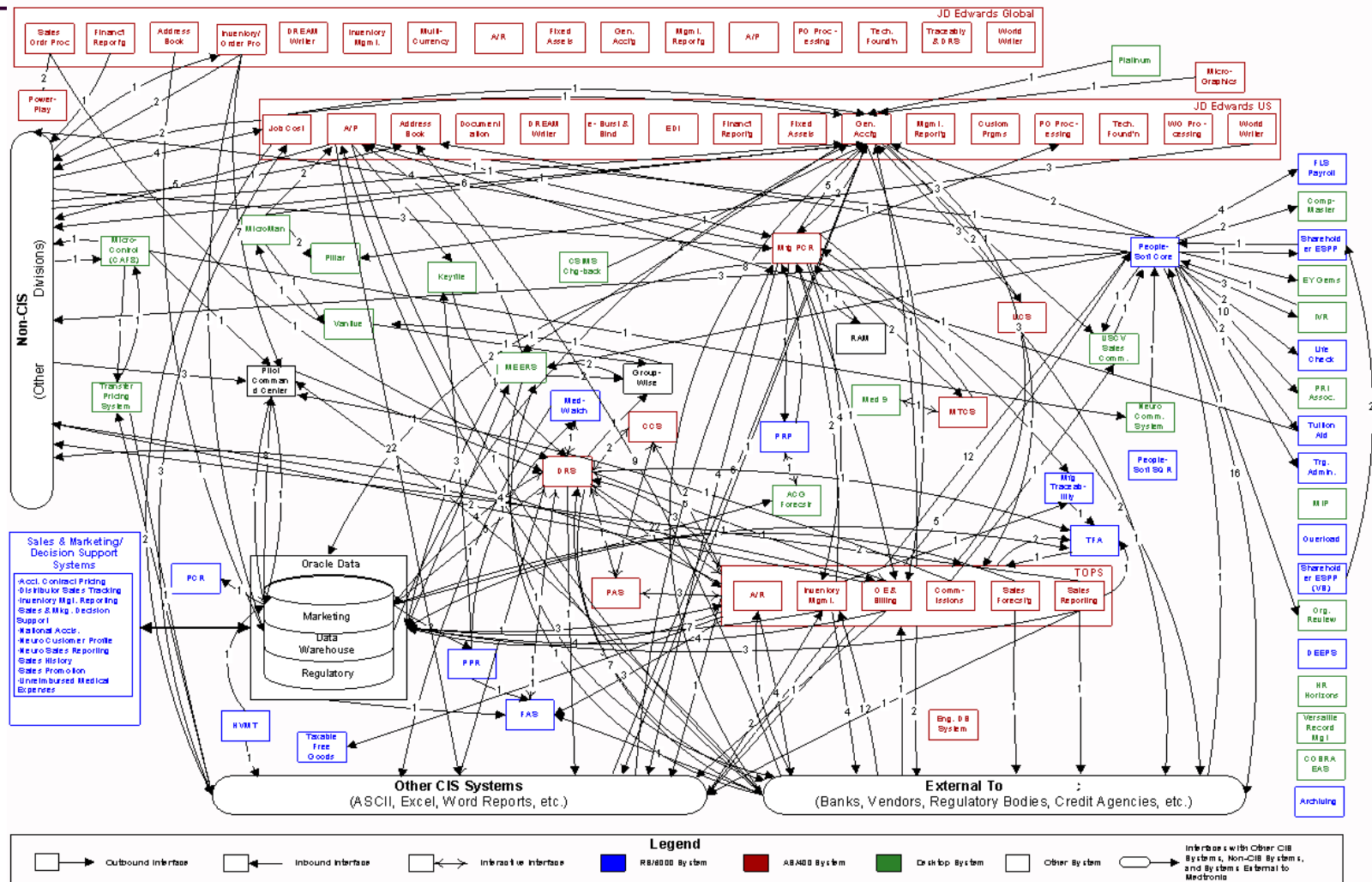
- Meta Group

点对点集成带来的结果

- 大量编程、脆弱、维护困难
- 高级QoS特征必须被在每个endpoint编码实现
- 一处改变影响波及各个系统，新系统加入最多要维护 $N(N-1)$ 个集成



蜘蛛网式的企业应用接口导致最终无法维护

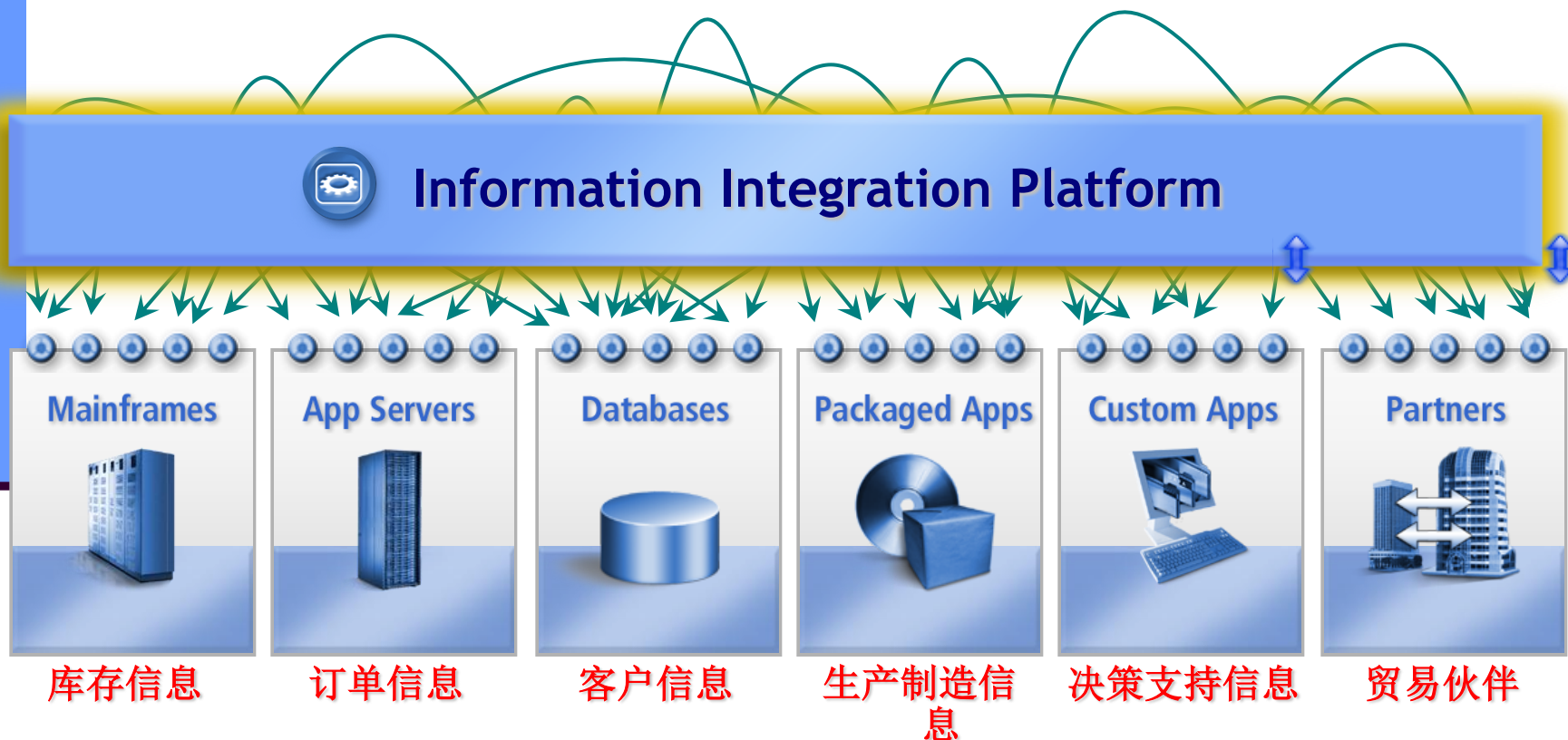


通过平台实现集成体系的革命

单一的集成平台取代网状集成连接

所有系统仅仅与集成平台交互，不关心其它系统的连接情况

信息集成平台负责与各个系统的接口和格式转换

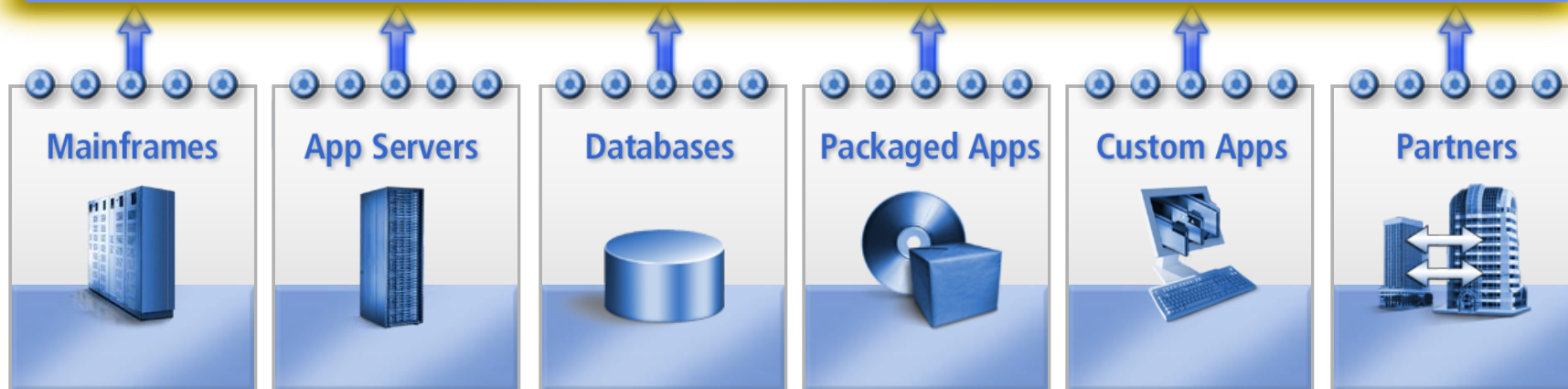


信息集成平台的功能

- 多种预建的适配器
- 图形化数据格式转换
- 事务机制
- 基于订阅/发布的消息机制
- 可靠的消息传递
- 消息路由
- 动态部署
- 系统管理
- 安全性保证
- B2Bi, PKI支持



Information Integration Platform



集成的四个技术层面

应用接口层

解决的问题是独立应用系统之间的连接，传统的应用系统之间的连接方式包括了：CORBA, SOCKET通讯, RMI, RPC, EJB, COM/COM+, HTTP和FTP等，数据库系统之间常见的连接规范包括：ODBC, JDBC。上述这些规范在企业应用系统或数据库系统之间传统的点对点的连接中得以广泛应用。【通讯】

应用整合层

EAI技术层次体系中的核心层次，该层次是连接业务流程管理层和应用接口层的桥梁。数据信息在业务流程中的流转以及在各个应用系统之间的交互必须建立在数据源和数据目的地都能理解该数据信息的基础之上。在应用整合层我们定义了能为数据产生源、数据处理地、数据投送地都能理解的信息处理规范方式、方法和规则，包括：数据格式定义、数据转换和消息路由。【数据】

集成的四个技术层面

■ 流程整合层

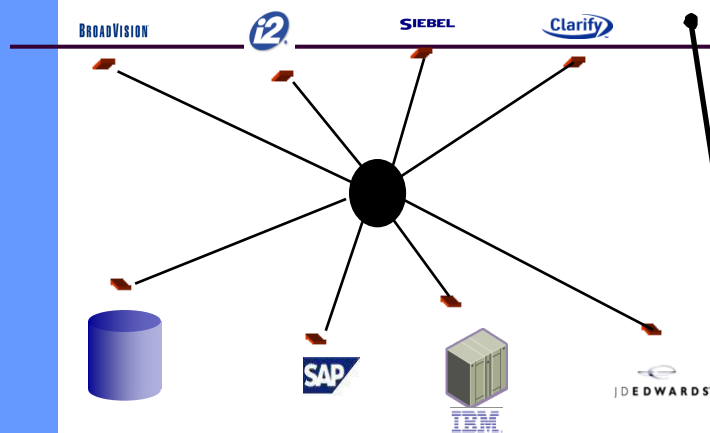
采用成熟的技术可以成功地创建模型，自动化流程处理过程，监控和管理这些业务流程，从而满足业务变化的需求。一个完整的业务流程整合方案应该包括BPM（业务流程管理）、BAM（业务活动监控）、B2Bi（企业间的整合）三个主要方面，只有具备了这三方面的能力，企业才能真正从业务整合中受益，实现按需应变的电子商务。

■ 用户交互层

用户交互层是EAI与用户实现人机交互在表示层面上的扩展。涉及的内容包括展示内容的集成（门户应用）、单点登陆（Single Sign On）、用户统一管理、用户认证授权的管理等。现今很多EAI产品都提供了对用户集成这几方面内容的支持。

【门户】

典型的集成体系架构分析

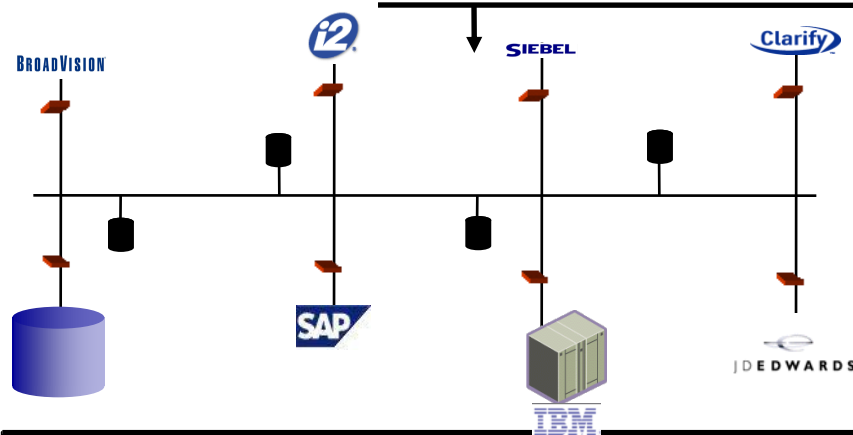


集线器型

- 优势
 - 集中管理
 - 部署方便
- 缺点
 - 部门级的应用
 - 扩展性差

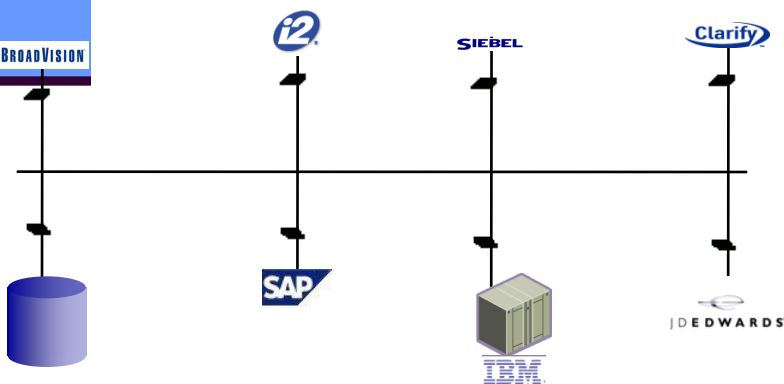
基于消息队列

- 优势
 - 扩展性较好
 - 基于队列主题
- 缺点
 - 每个应用都对应多个队列
 - 没有充分利用硬件资源
 - 维护困难

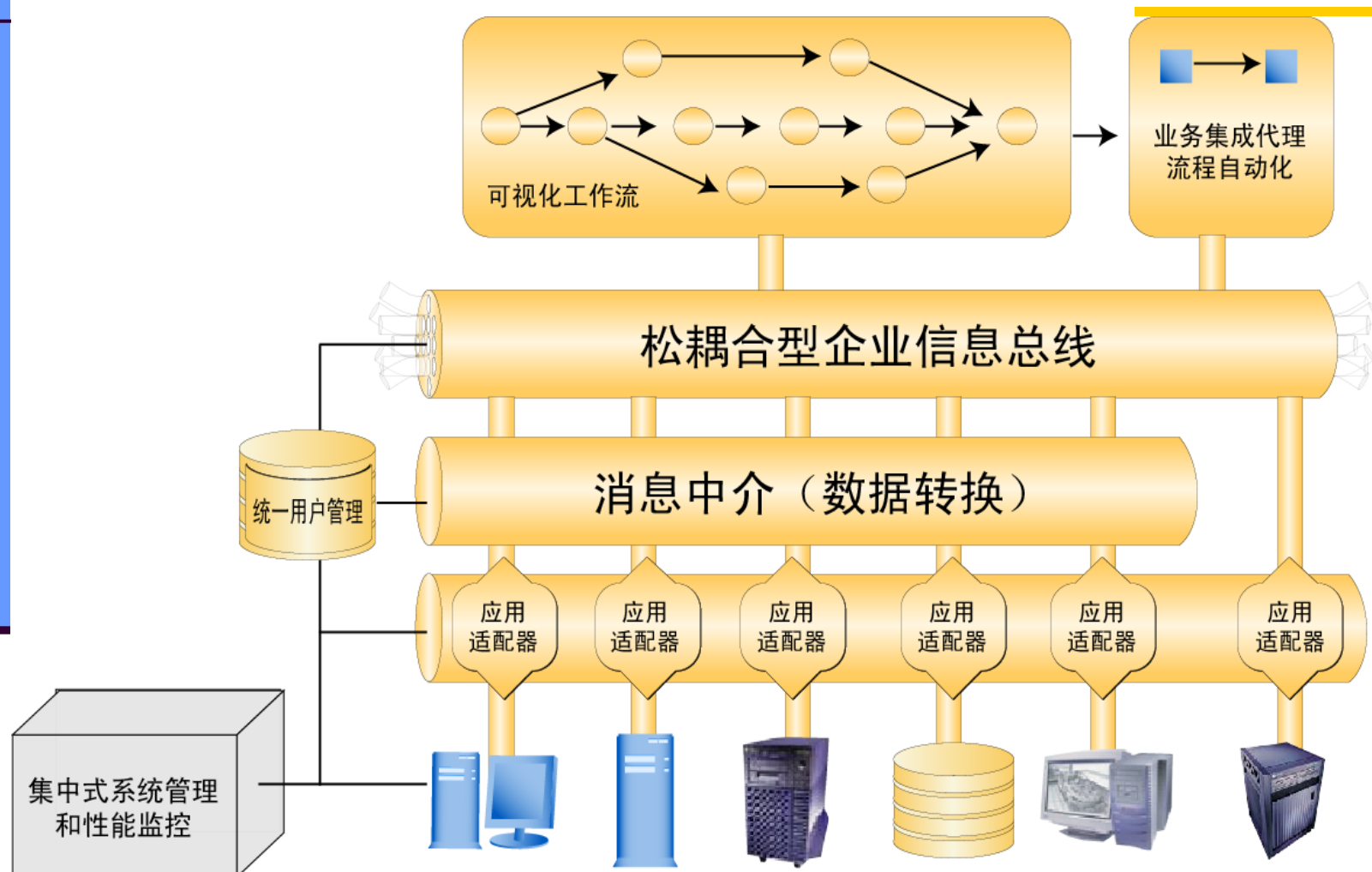


以网络为中心的总线型结构

- 优势
 - 扩展性好
 - 实现了“即插即用”的集成方式
 - 充分利用了现有的硬件资源
- 缺点
 - 只适合简单的“Push”应用集成
 - 网络负载大



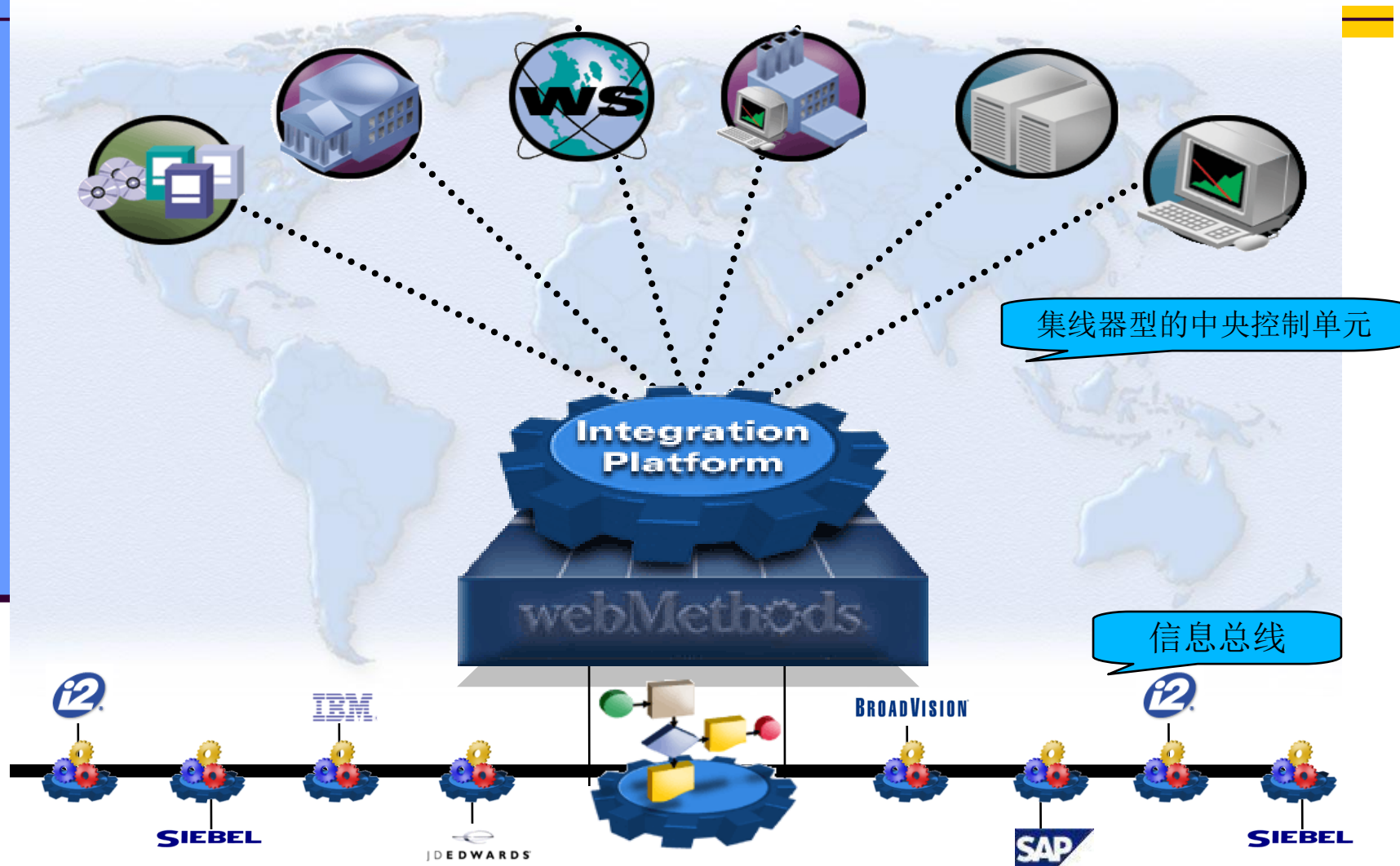
“企业应用集成” 平台的构成



目录

- 背景
- SOA概念
- SOA构建中的三个要素
- 注意的问题

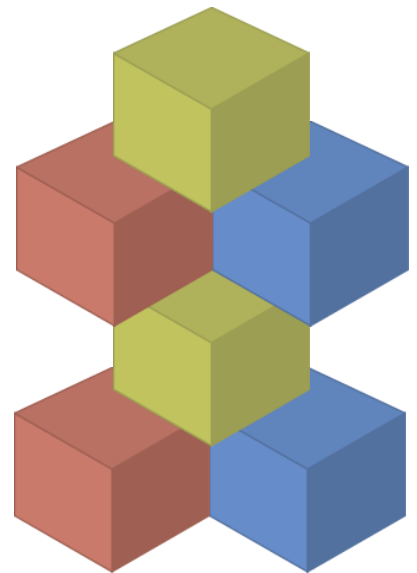
面向服务的集成体系架构（SOA）



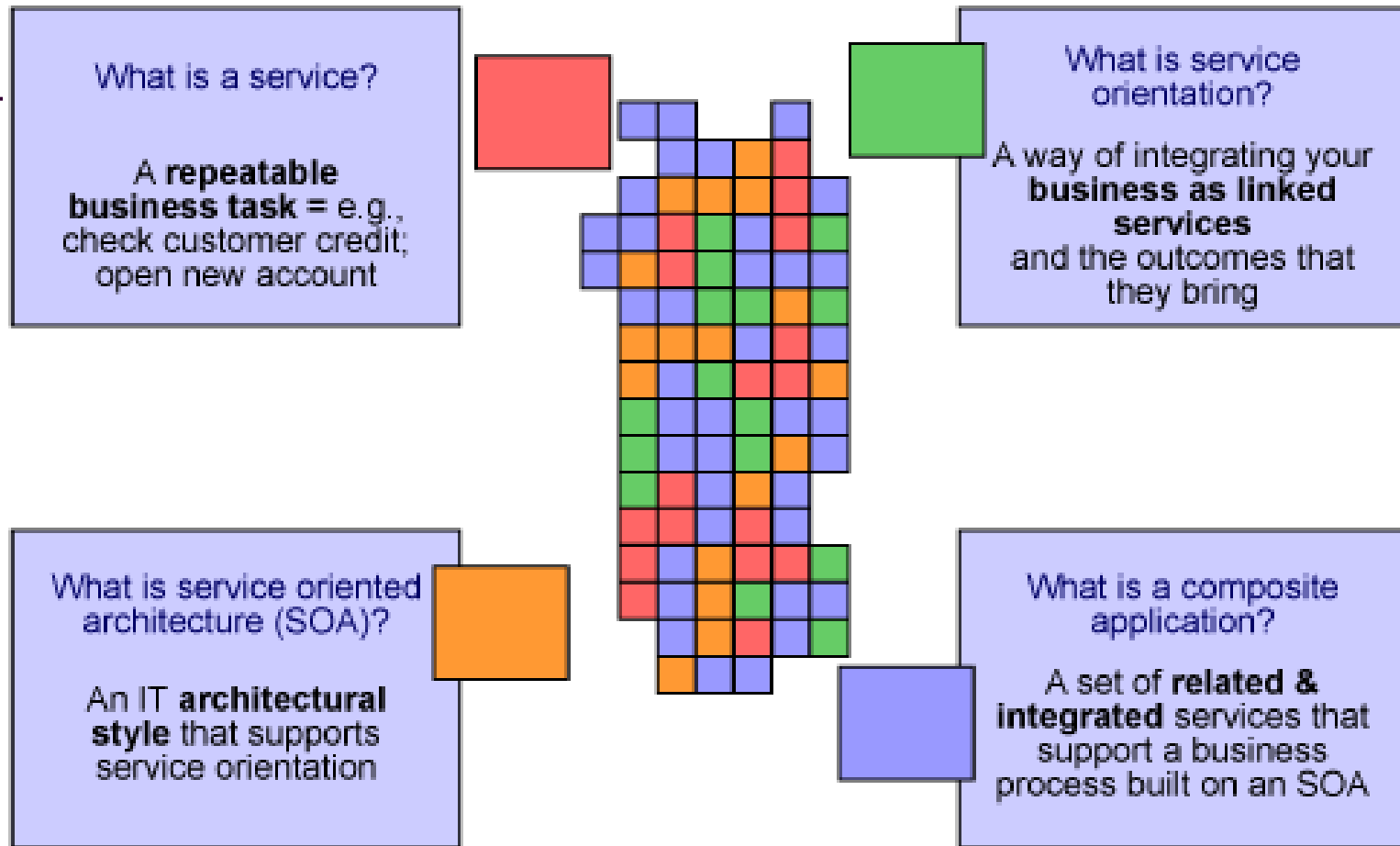
What is a Service-Oriented Architecture?

An approach to building distributed systems that delivers application functionality as services to end-user applications or to other services.

- A flexible and open architecture
 - Represent software assets as services
 - Integrate with applications outside of the enterprise
 - Well defined ways of representing and interacting with software components
- Software components become reusable building blocks
 - Focus on application assembly rather than development
 - Create new internal apps out of existing components



SOA step by step



Gartner Group预计，到2008年，SOA将成为占有绝对优势的软件工程实践方法，它将很可能结束传统的整体软件体系架构长达40年的统治地位，届时将有70%的企业在进行企业IT建设时会转向SOA

SOA VS Traditional Architecture

- 传统软件包是被编写为独立的（self-contained）软件，即在一个完整的软件包中将许多应用程序功能整合在一起。实现整合应用程序功能的代码通常与功能本身的代码混合在一起。我们将这种方式称作软件设计“单一应用程序”。与此密切相关的是，更改一部分代码将对使用该代码的代码具有重大影响，这会造成系统的复杂性，并增加维护系统的成本。而且还使重新使用应用程序功能变得较困难，因为这些功能不是为了重新使用而打的包
- SOA旨在将单个应用程序功能彼此分开，以便这些功能可以单独用作单个的应用程序功能或“组件”。这些组件可以用于在企业内部创建各种其他的应用程序，或者如有需要，对外向合作伙伴公开，以便用于合作伙伴的应用程序。“服务”的概念是要使用与实施细节无关的标准化接口来构建这些“组件”
- 服务装配化使企业随需而变

SOA基本特征

■ 独立的功能实体

SOA非常强调架构中提供服务的功能实体的完全独立自主的能力。传统的组件技术，如.NET Remoting, EJB, 都需要有一个宿主来存放和管理这些功能实体；当宿主运行结束时这些组件的寿命也随之结束。当宿主本身或者其它功能部分出现问题的时候，在该宿主上运行的其它应用服务就会受到影响。

■ 大数据量低频率访问

对于.NET Remoting, EJB这些传统的分布式计算模型而言，服务提供都是通过函数调用的方式进行的，一个功能的完成往往需要通过客户端和服务端多次函数调用才能完成。在Intranet的环境下，这些调用给系统的响应速度和稳定性带来的影响都可以忽略不计，但是在Internet环境下这些因素往往是决定整个系统是否能正常工作的一個关键决定因素。因此SOA系统推荐采用大数据量的方式一次性进行信息交换。

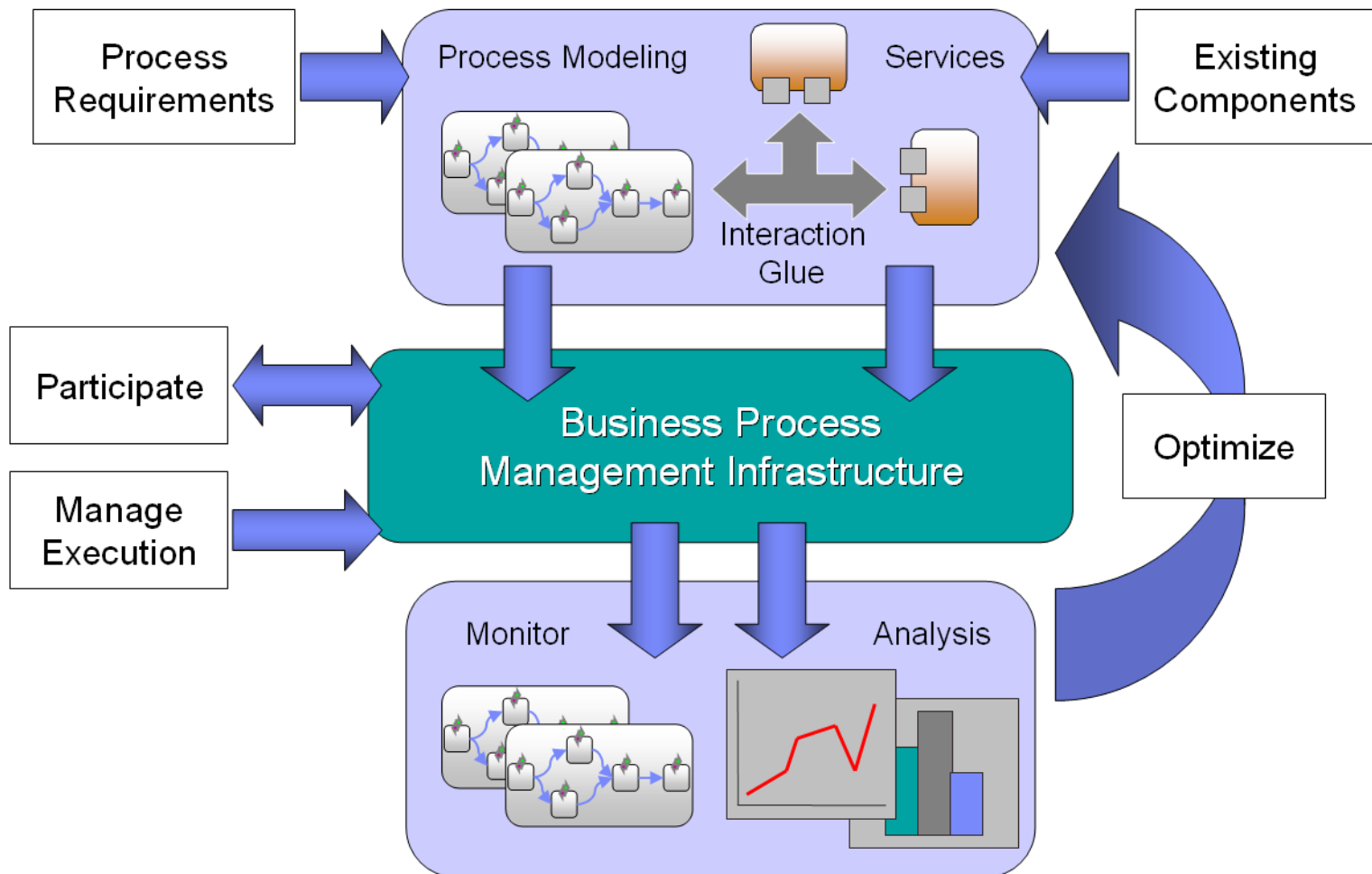
■ 基于文本的消息传递

由于Internet中大量异构系统的存在决定了SOA系统必须采用基于文本而非二进制的消息传递方式。在COM、CORBA这些传统的组件模型中，从服务器端传往客户端的是一个二进制编码的对象，在客户端通过调用这个方法来完成某些功能；由于基于文本的消息本身是不包含任何处理逻辑和数据类型的，因此服务间只传递文本，对数据的处理依赖于接收端的方式可以帮忙绕过兼容性这个的大泥坑。

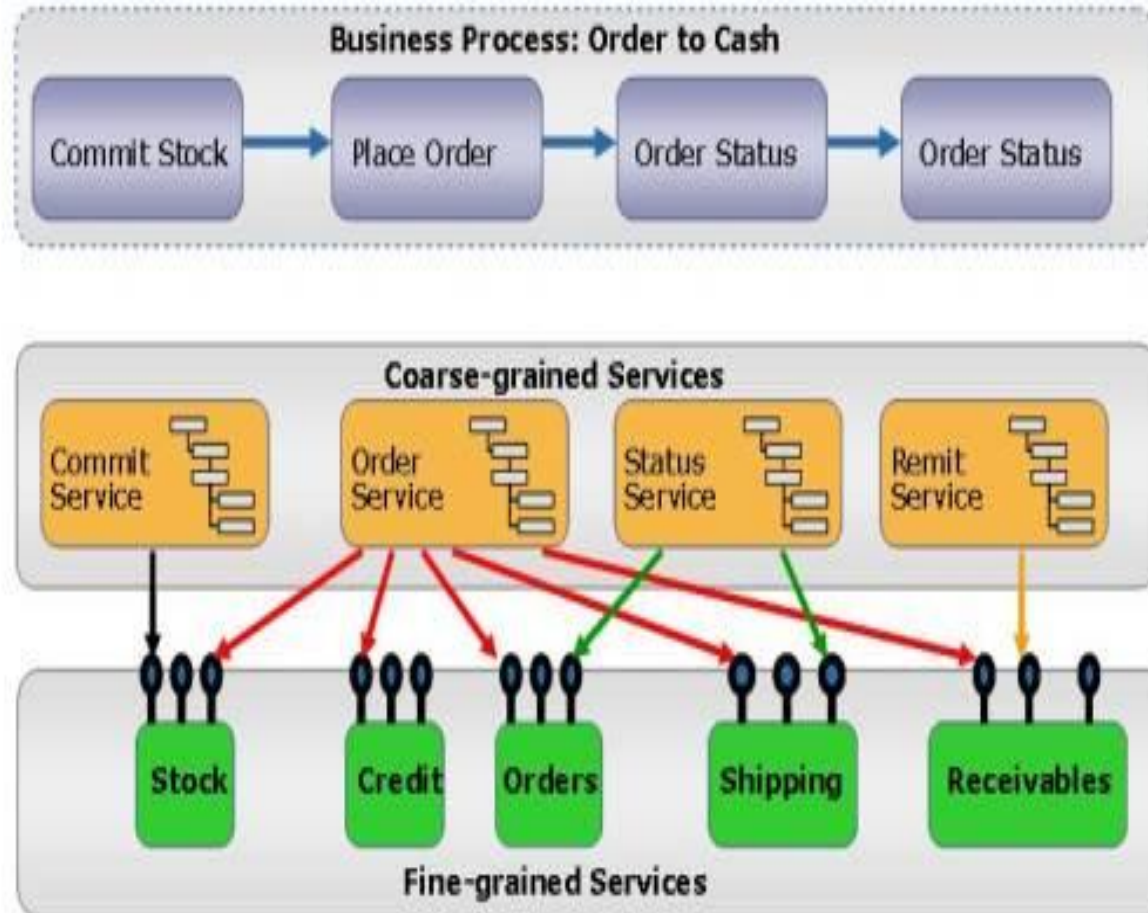
SOA核心理念

- 让企业应用彻底摆脱面向技术的解决方案的束缚，轻松应对企业商业服务变化、发展的需要
- 业务决定服务，服务决定技术

SOA Dev Lifecycle



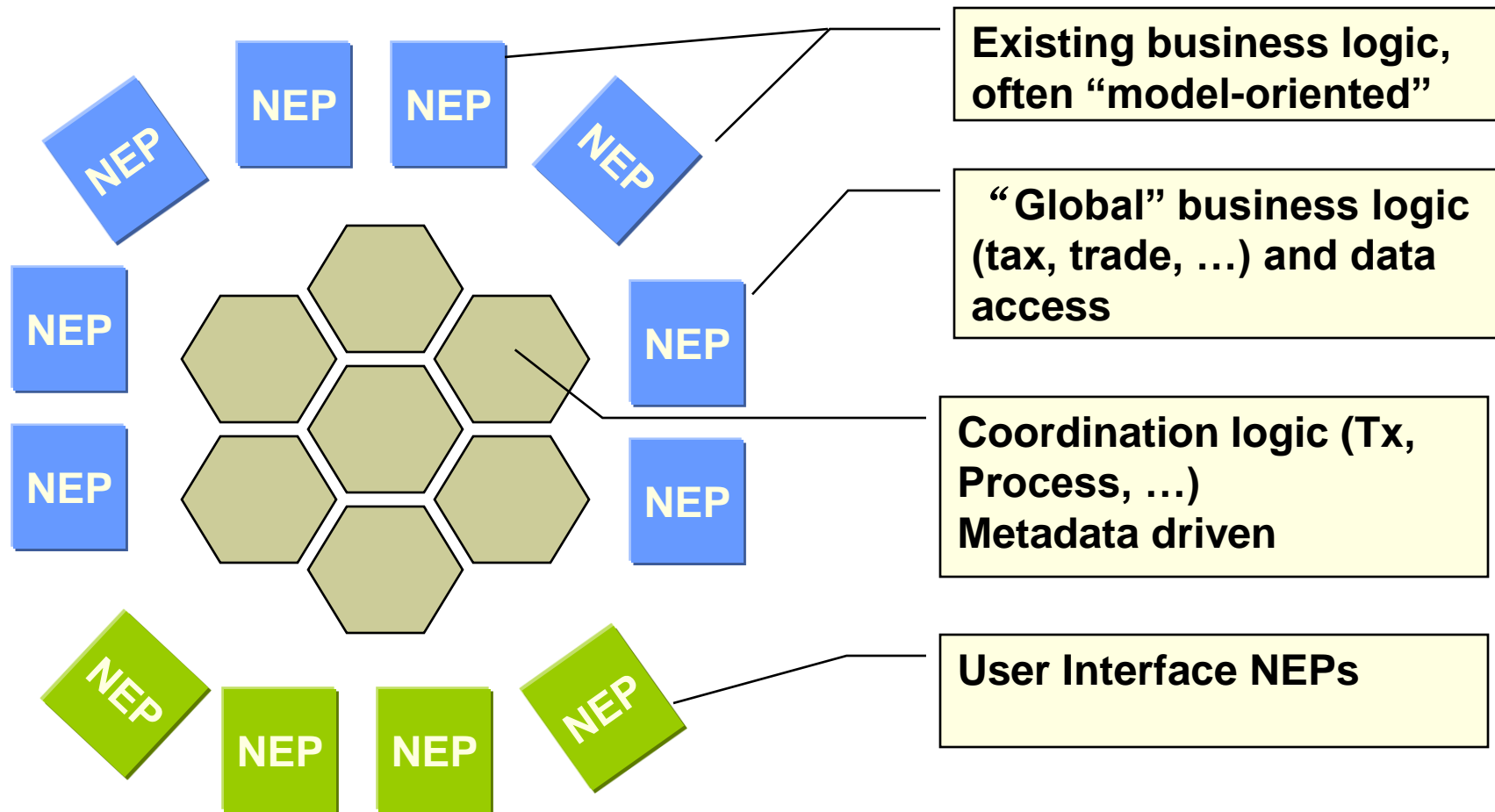
Principal: Fine-grained Coarse-grained BP →



目录

- 背景
- SOA概念
- SOA构建中的三个要素
- 结论

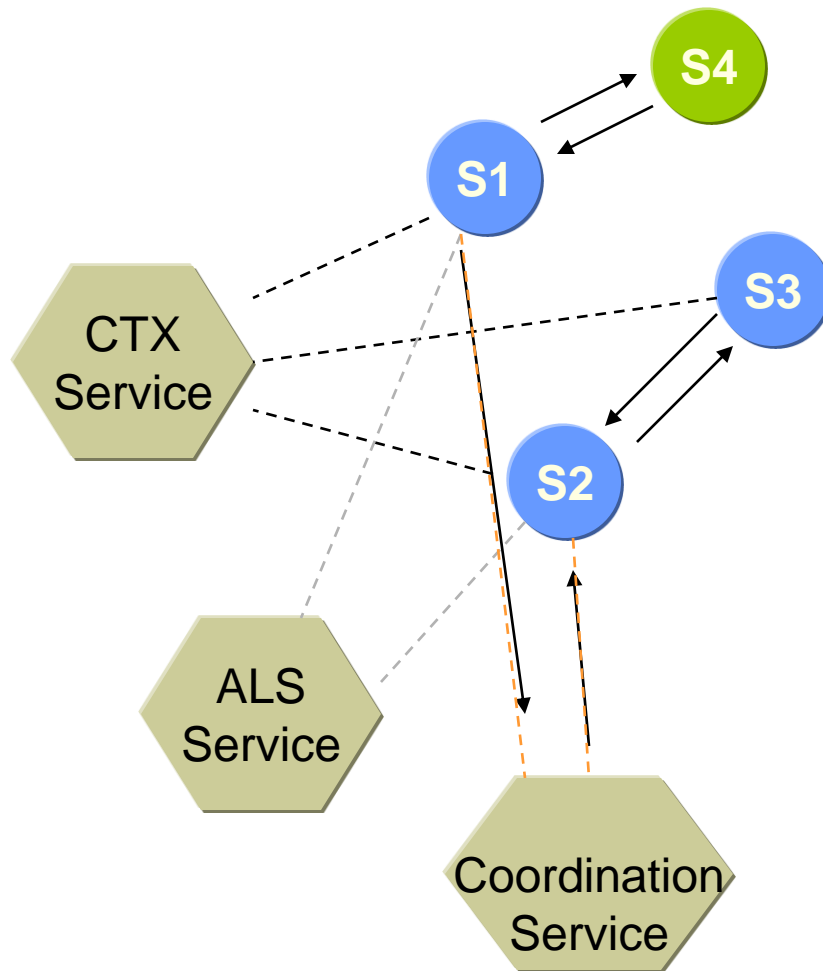
SOA中考虑的问题



SOA三要素

- The coordination layer
- Information Entities
- The relationship between BPM and SOA

Coordination Layer



A coordinator is an active component of the architecture

It can support a service or provide services itself

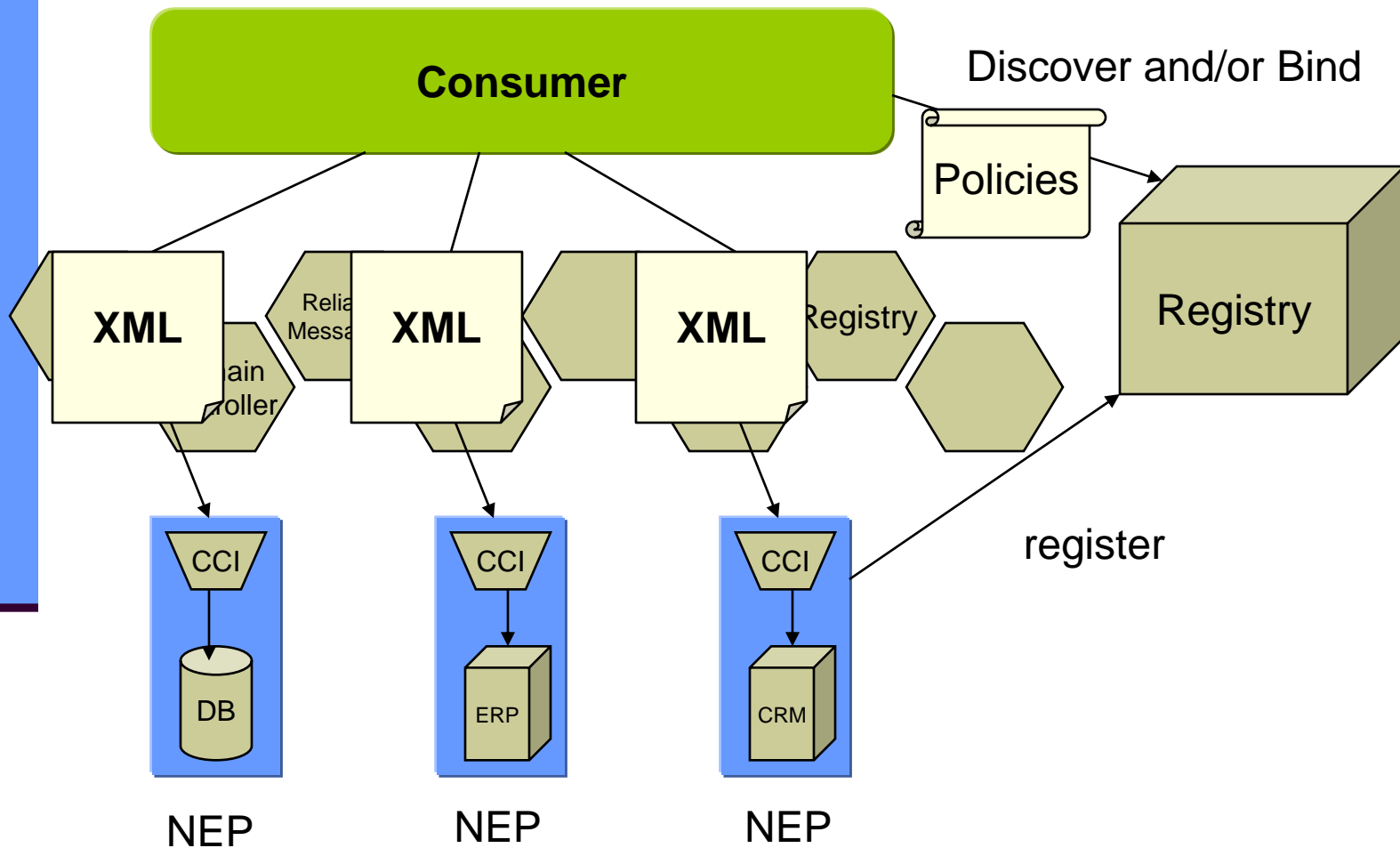
Multiple purposes:

- Transaction
- Orchestration
- Choreography ...

传统应用服务器提供的服务

- Transaction
 - Security
 - Connection pooling
 - Naming service
 - Scalability and failover
 - ...
-
- They become the “Service Fabric”

带有容器和适配器的服务总线



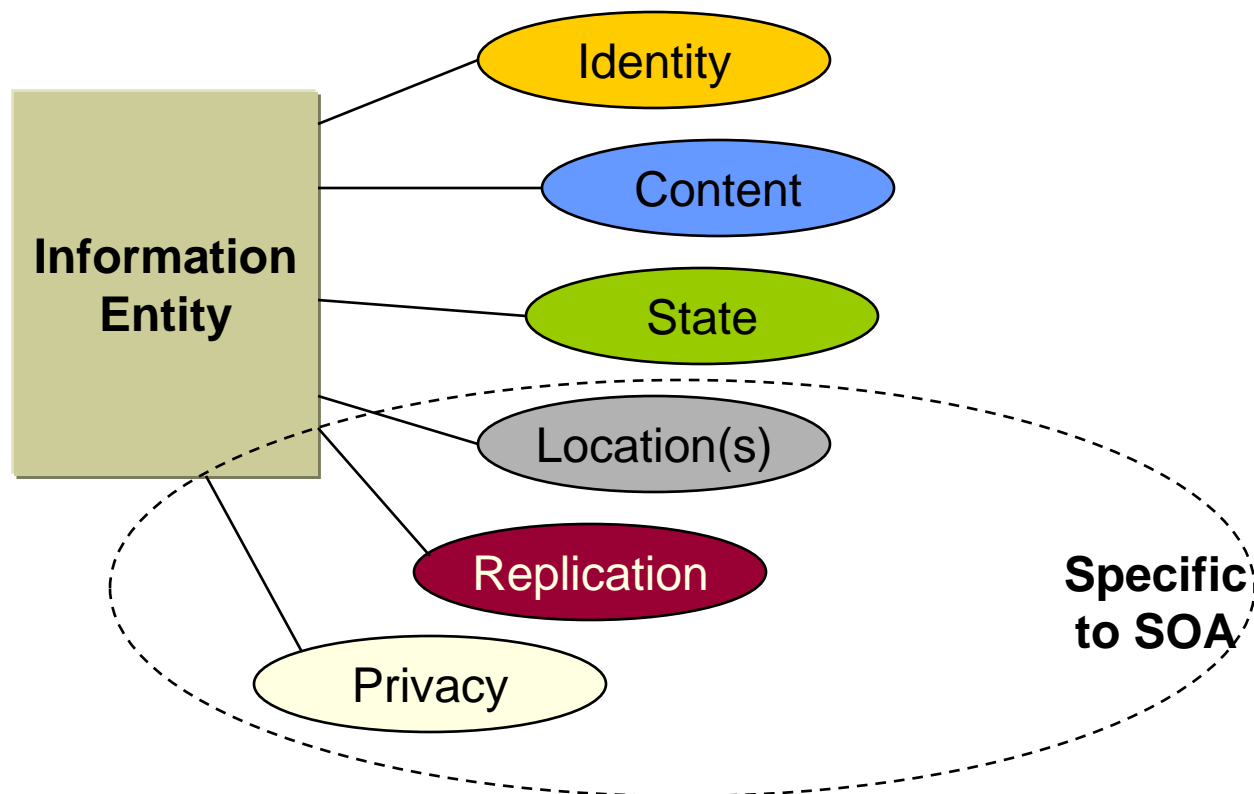
Information Entity in SOA

- “at the heart of Web services is a very complex problem: with distributed applications comes the need for distributed data sharing”
 - Identification and equivalence
 - authentication
 - Authorization and privacy
 - mediation
 - synchronization

Source: The Dataweb: An Introduction to XDI, Drummond Reed et al.

Information Entities in SOA

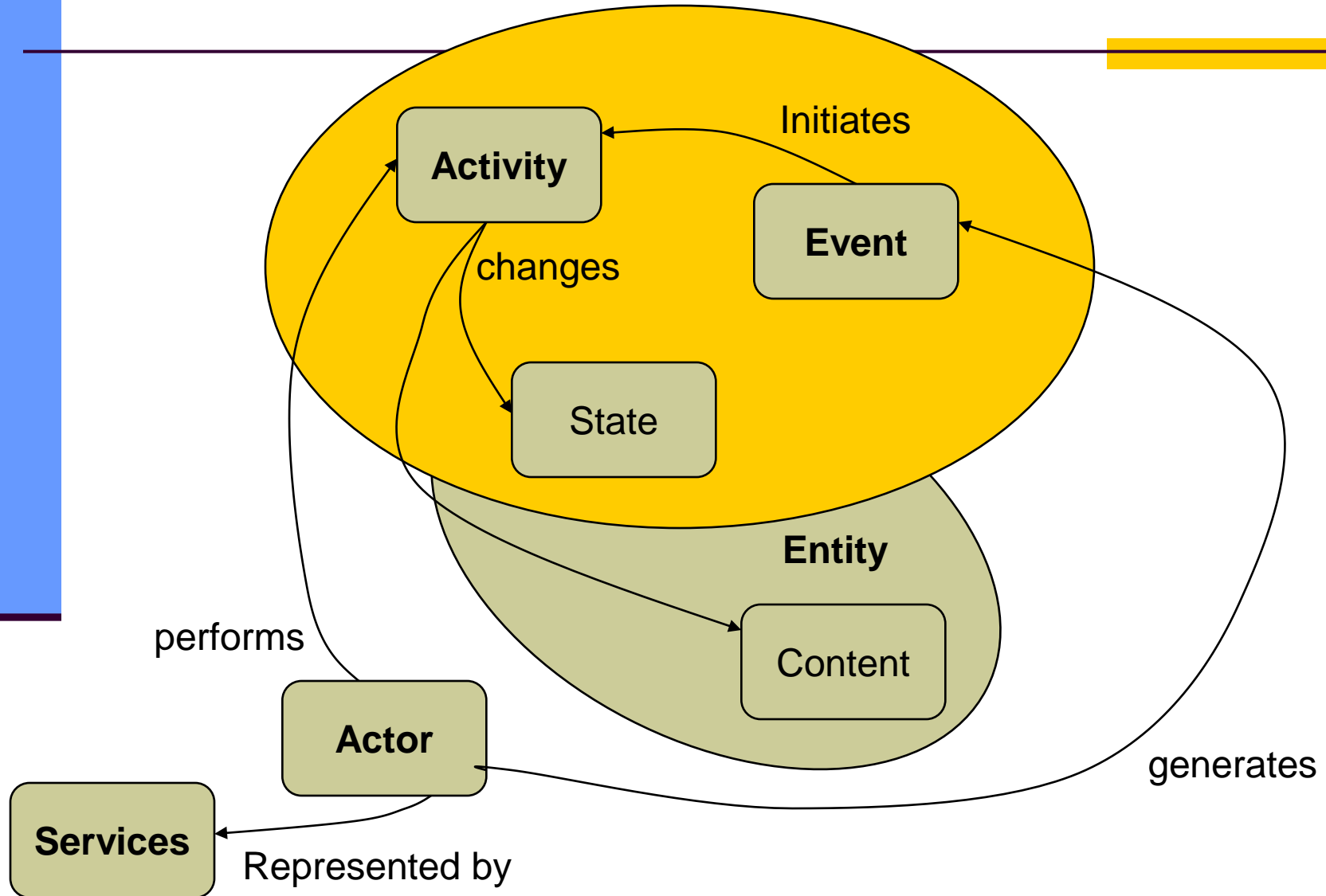
- Several dimensions appear when managing an Information Aggregate in a SOA



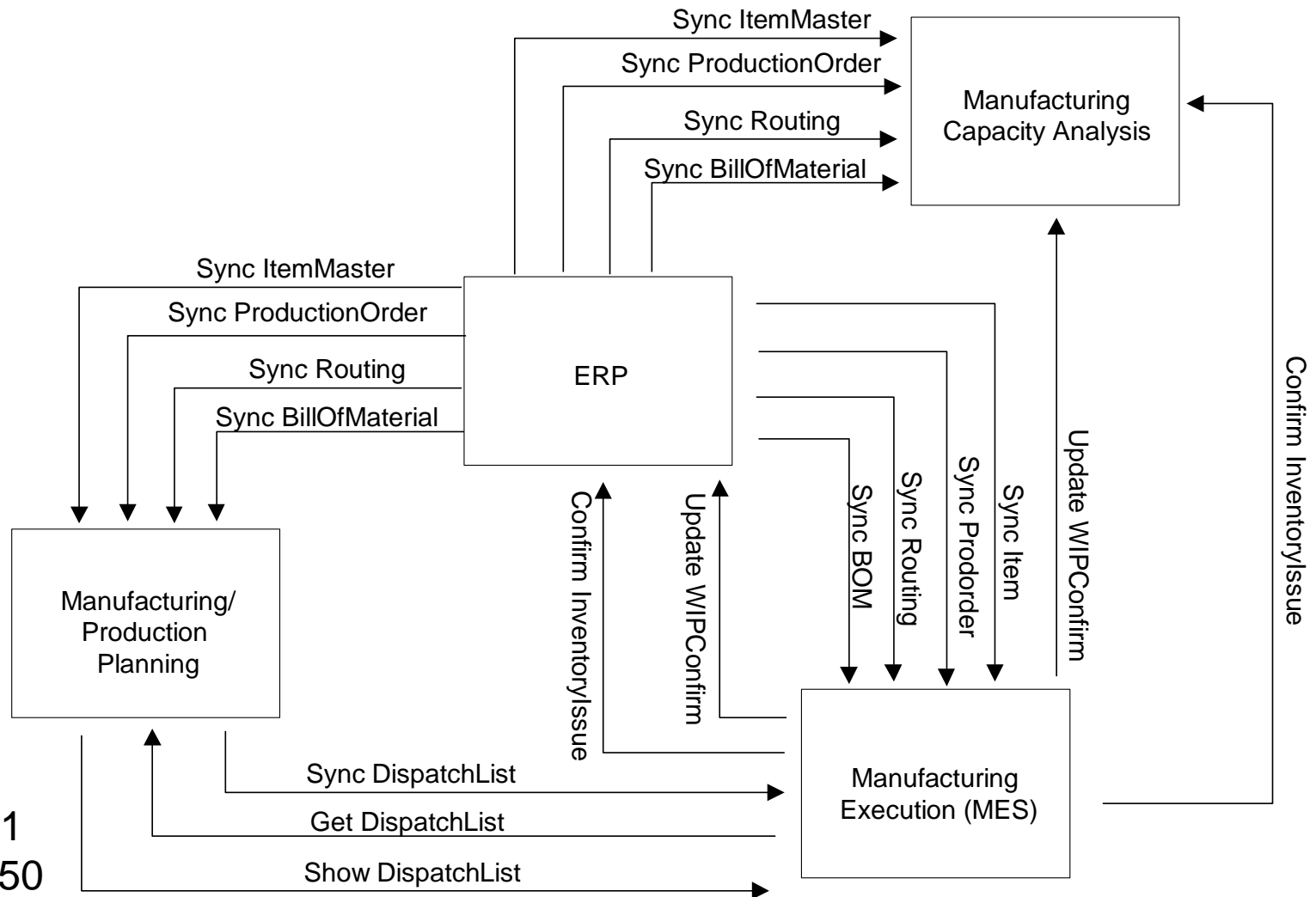
SOA and BPM

- SOA is about constructing software components that can be reused in context unknown at design time
 - Composition versus Extension (OO)
- BPM is about being able to precisely model and possibly change the context in which enterprise components are used
- But how the two meet?

Elements of BPM



A business process does not have a “center”...it is *de facto* “peer-to-peer”



OAGIS 8.1
Scenario 50

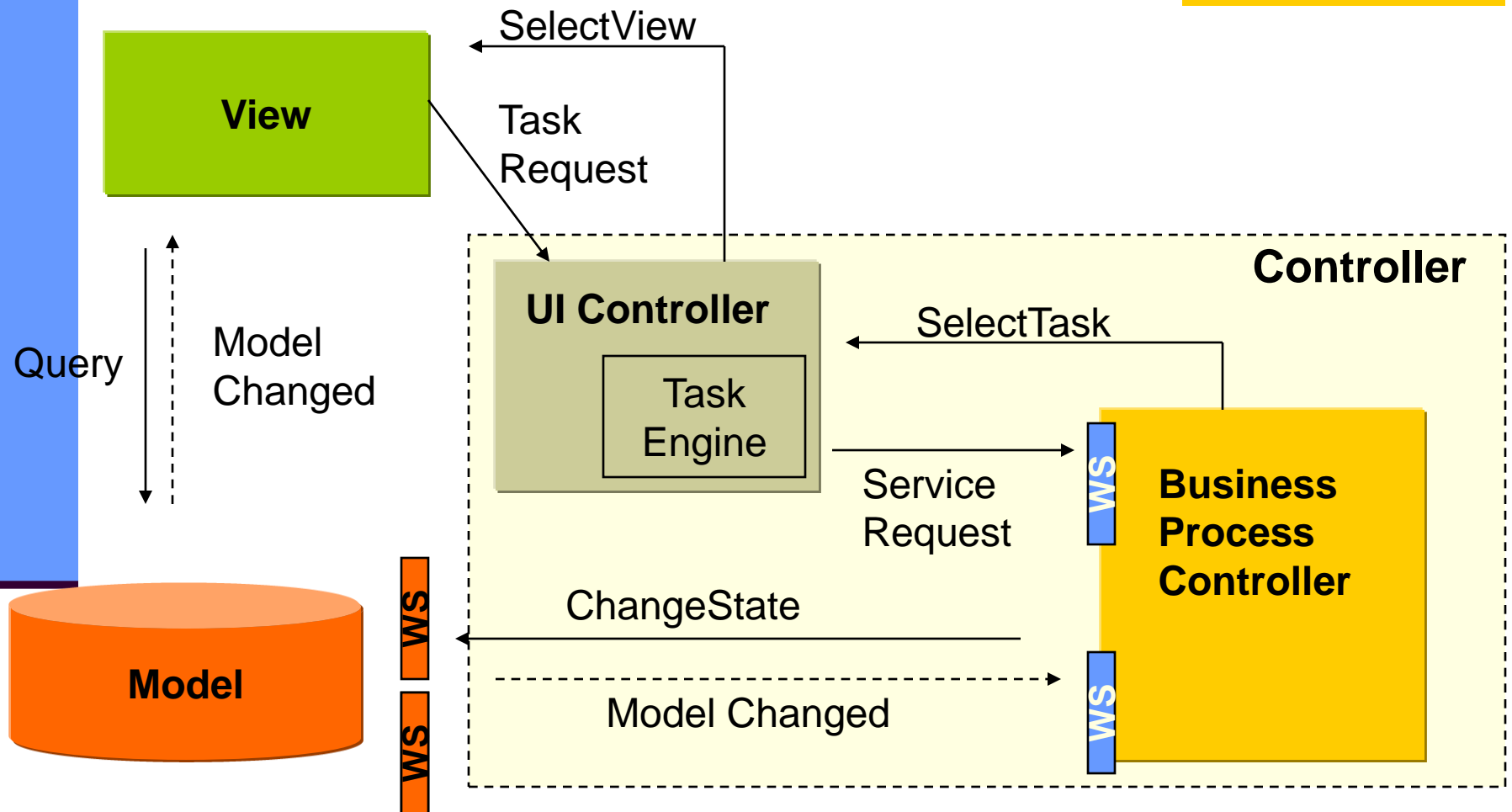
Bringing BPM and SOA together

- The foundation is becoming sound with strong theoretical support
 - A big piece still missing: “state” (IMHO)
 - Shift from Orchestration to Business Process Definition
- Once the foundation is in place we should see Domain Specific Languages (DSL) appear that are a lot closer to the way the users (not the programmers) think about a Business Process

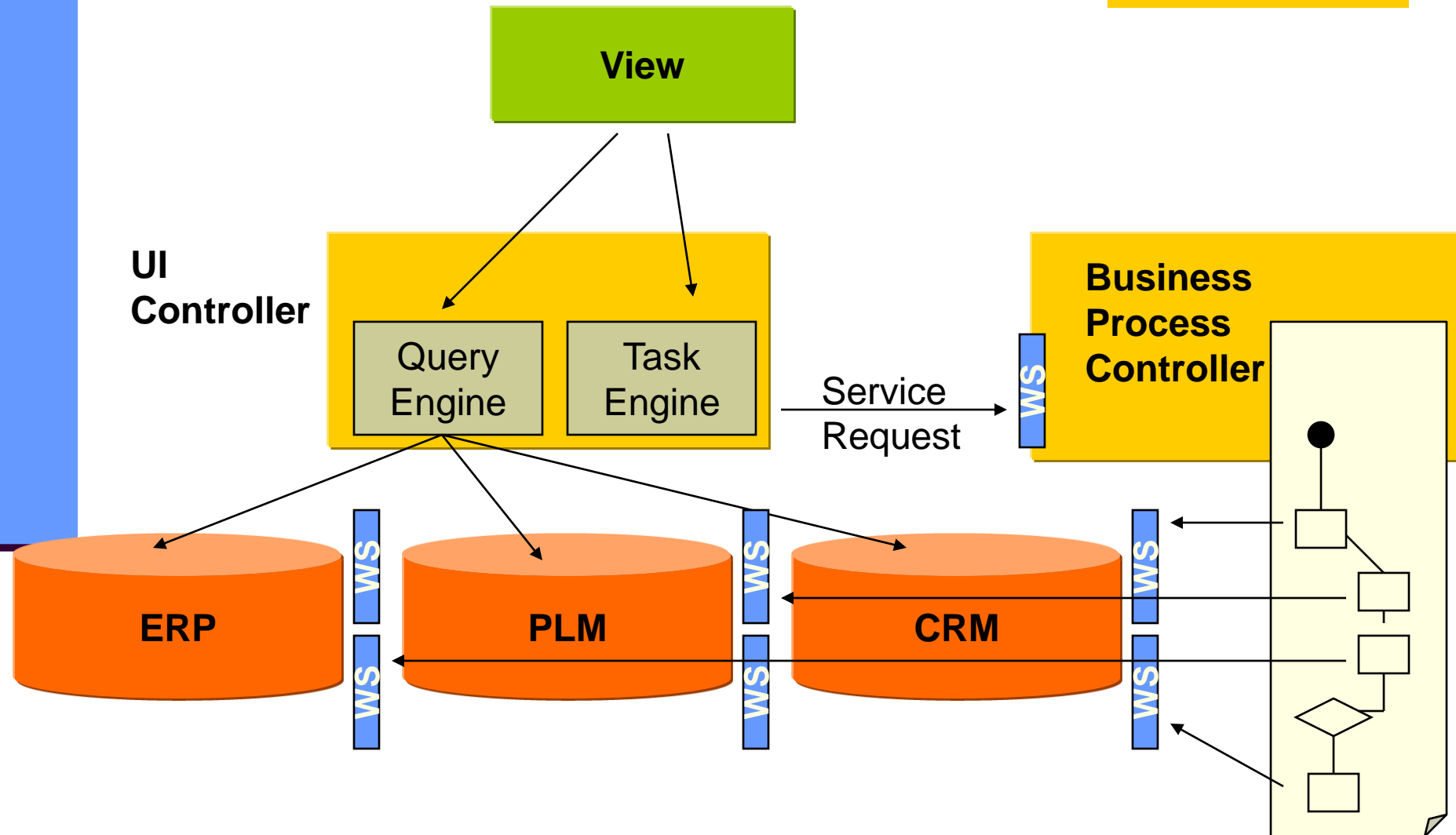
A Technical Model for Constructing Software in SOA

- We need to adapt the foundation of modern application architecture
 - Model-View-Controller
- Model → Services
- Controller → Coordination
- View → View

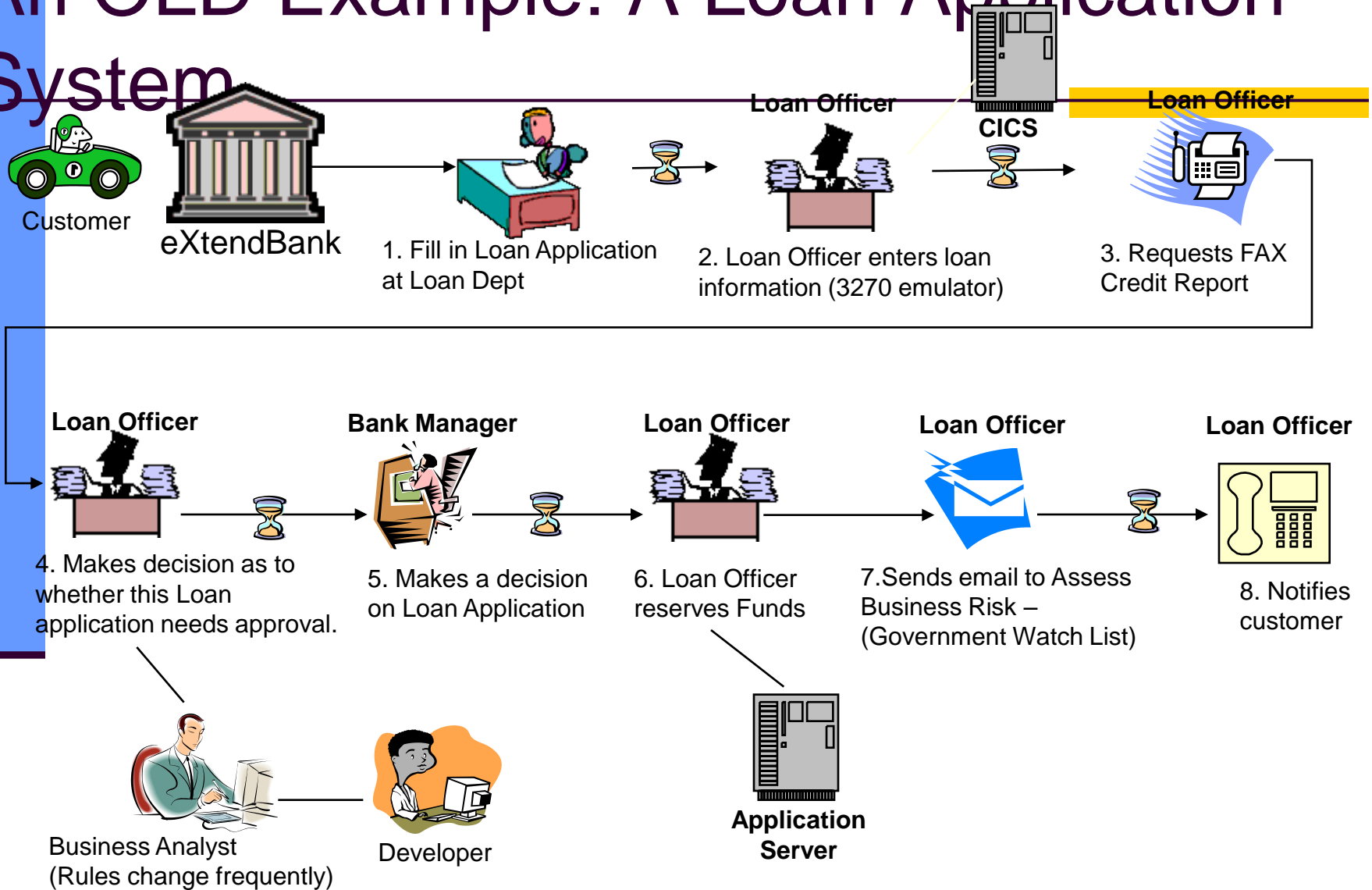
The Model-View-Controller pattern Revisited



SOA requires a Complete Separation of the Business Logic and UI



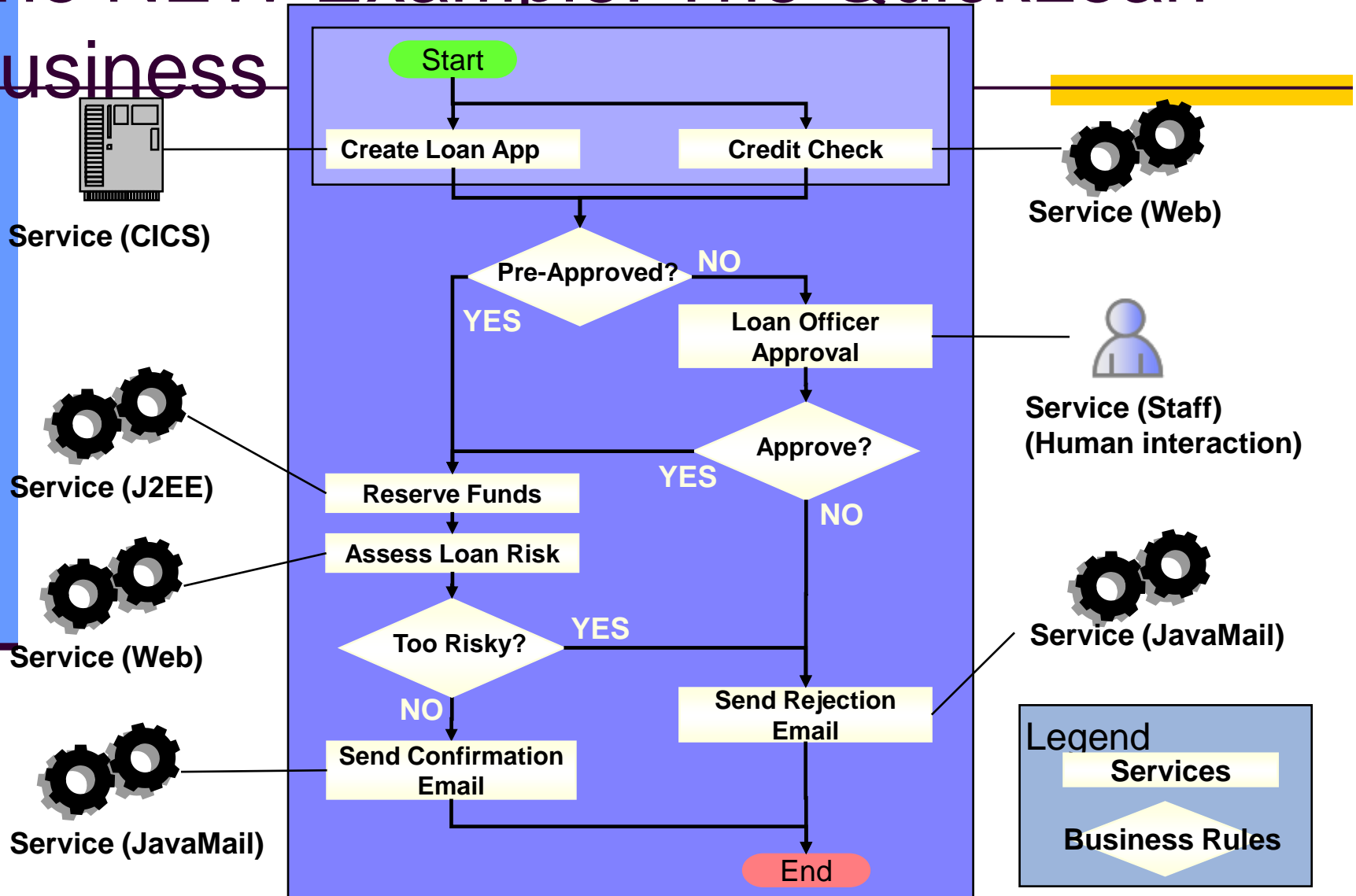
An OLD Example: A Loan Application System



Challenges with the OLD System

- Takes too long to process loan applications
- Paper based human interaction in loan processing is error prone
- Many different technologies are involved (legacy, app servers, email apps, ...)
- Integrating people, processes, and information is difficult
- Manual work is needed to “undo” work performed if there is a process failure
- Difficult to react to business rule changes

The NEW Example: The QuickLoan Business





目录

- 背景
- SOA概念
- SOA构建中的三个要素
- 注意的问题

注意的问题

- SOA相关的技术尚不成熟，可靠性、安全性、事务、编制(Orchestration)、遗留系统(Legacy support)支持和语义(Semantics)方面均还存在不足。
- SOA主要用于解决其余企业和企业之间的业务之间的整合，国内许多中小公司信息化基础差，大公司改造自己的业务系统，牵涉很多人员和投入，阻力大。
- 目前国内用户整体上还没有到规模化的推广、应用基于Web服务和SOA架构开发B2B应用的阶段，即使是企业内部的应用整合SOA适用的情况也不多，仅有的案例其实际意义也非常有限。企业是否要部署和应用SOA，根本上还是要看业务上的需要和要解决的业务问题以及要通过IT系统达到的目的。在技术的选择上，无论是在SOA出现之前、现在、还是SOA之后，最重要的是要看什么技术和产品能够最有效、最可靠、最方便地解决用户的现实业务问题和相关的技术问题。国内宣称实施了EAI的厂商大都集成在门户整合和信息整合这两个方面。



谢谢！