Software Architecture 12
Service Based Software Development

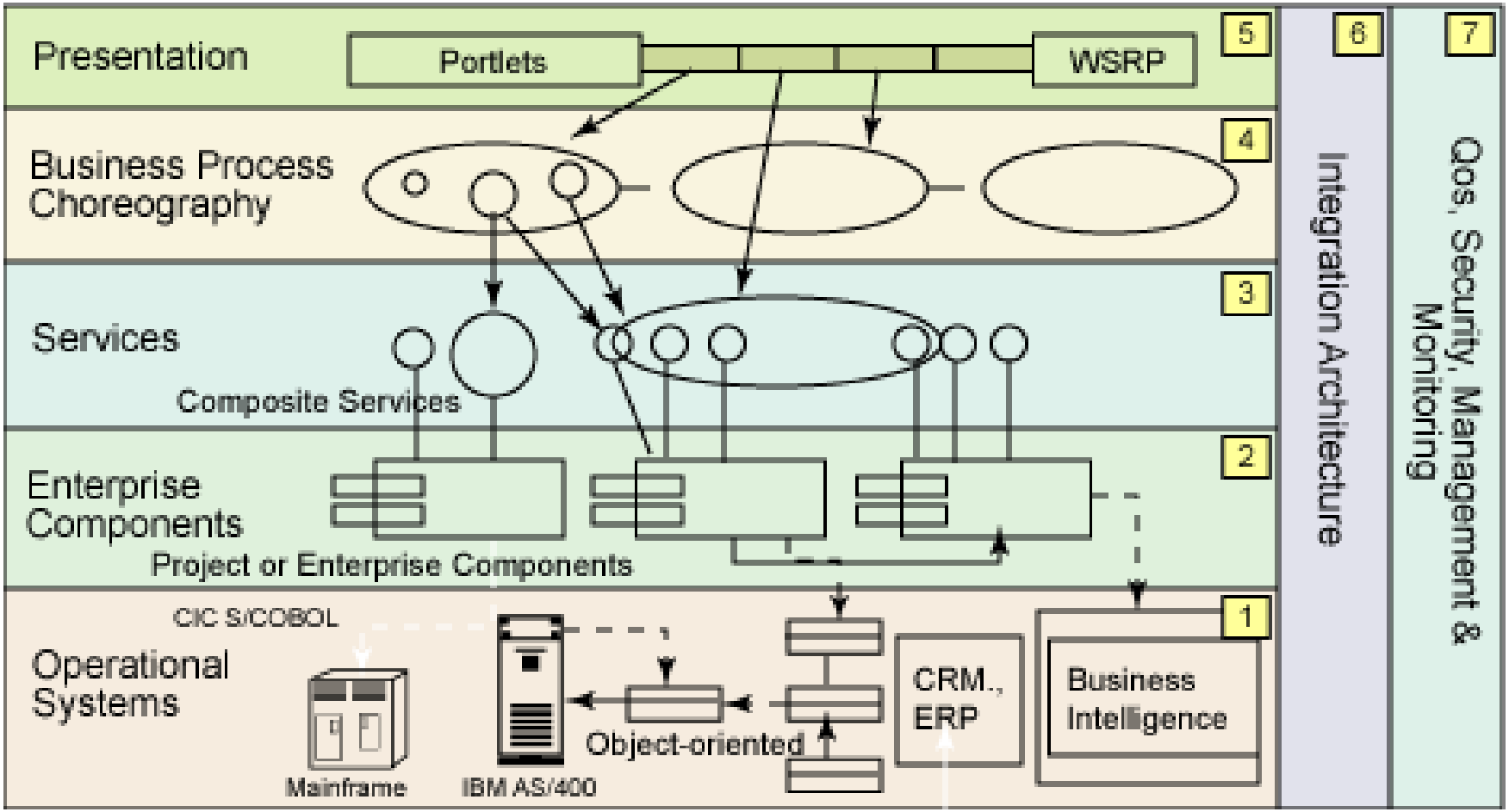
Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
e-mail: chen-hp@sjtu.edu.cn

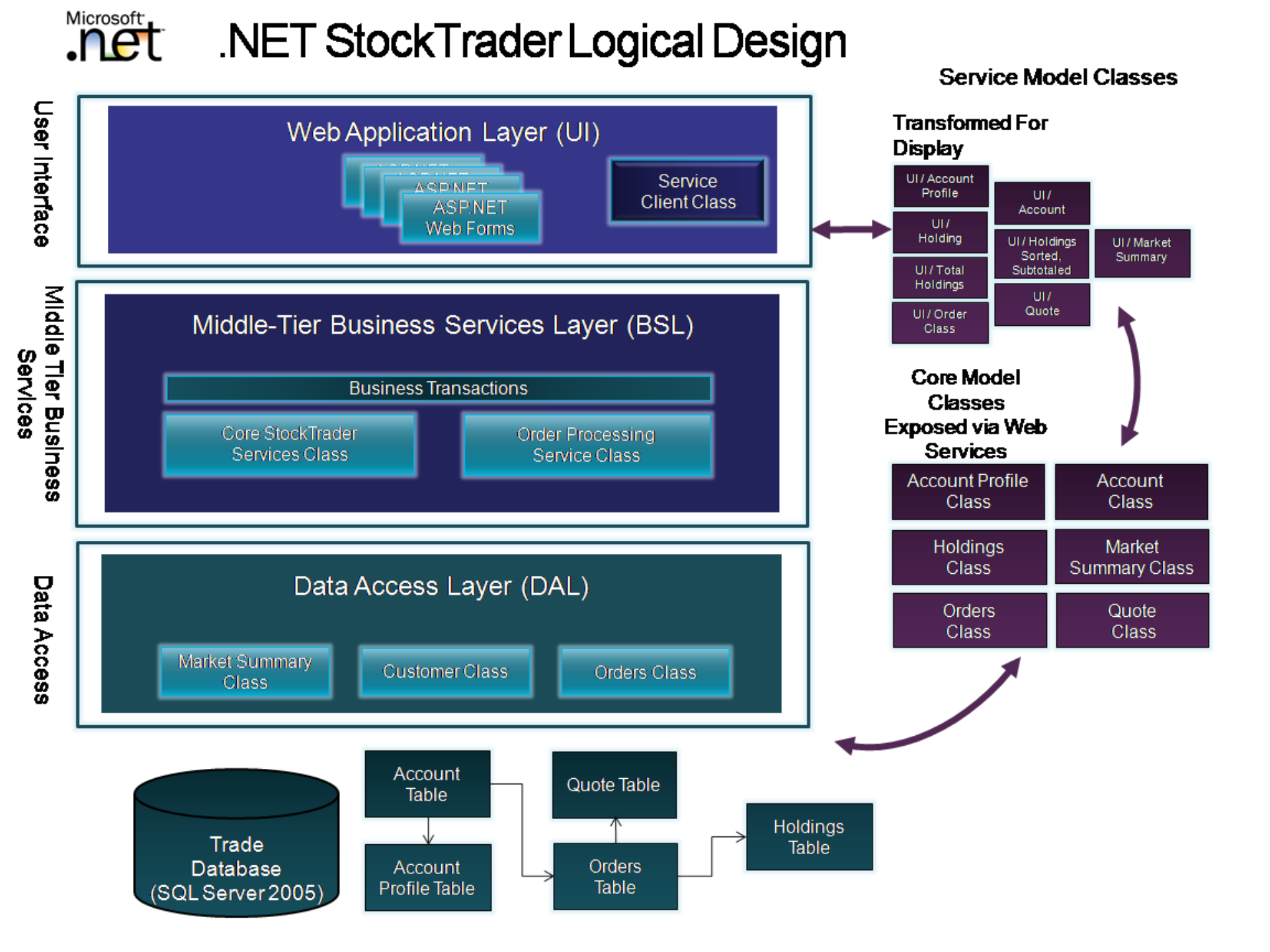
Agenda

- Model of SOAD
  - JBI
  - SCA/SDO
  - .NET
- ESB
  - ESB feature
  - MoM
  - ESB Core
  - Service Endpoint
  - Intelligent routing
- Mashup

Classic SOA Layers



.NET StockTrader Logical Design



Web Service Interoperability

IBM WebSphere J2EE Trade Application

IBM Trade 6.1 Web Application

Java Server Pages

Web Service Client

Web Service Interface

Enterprise Java Beans

JDBC

- DB2 V9
- Oracle 10G

MS .NET StockTrader Application

.NET StockTrader Web Application

ASP.NET

Web Service Client

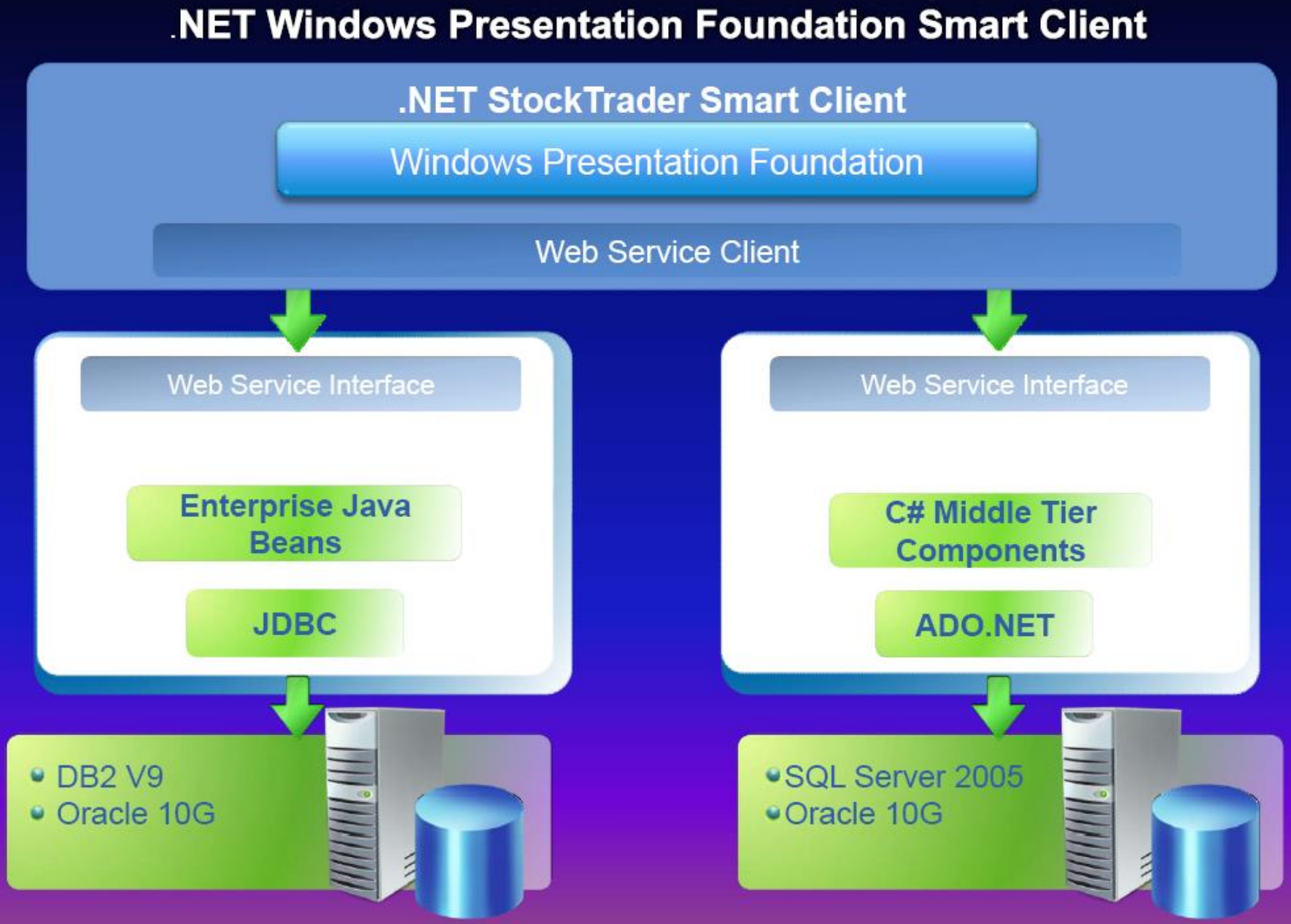
Web Service Interface

C# Middle Tier Components

ADO.NET

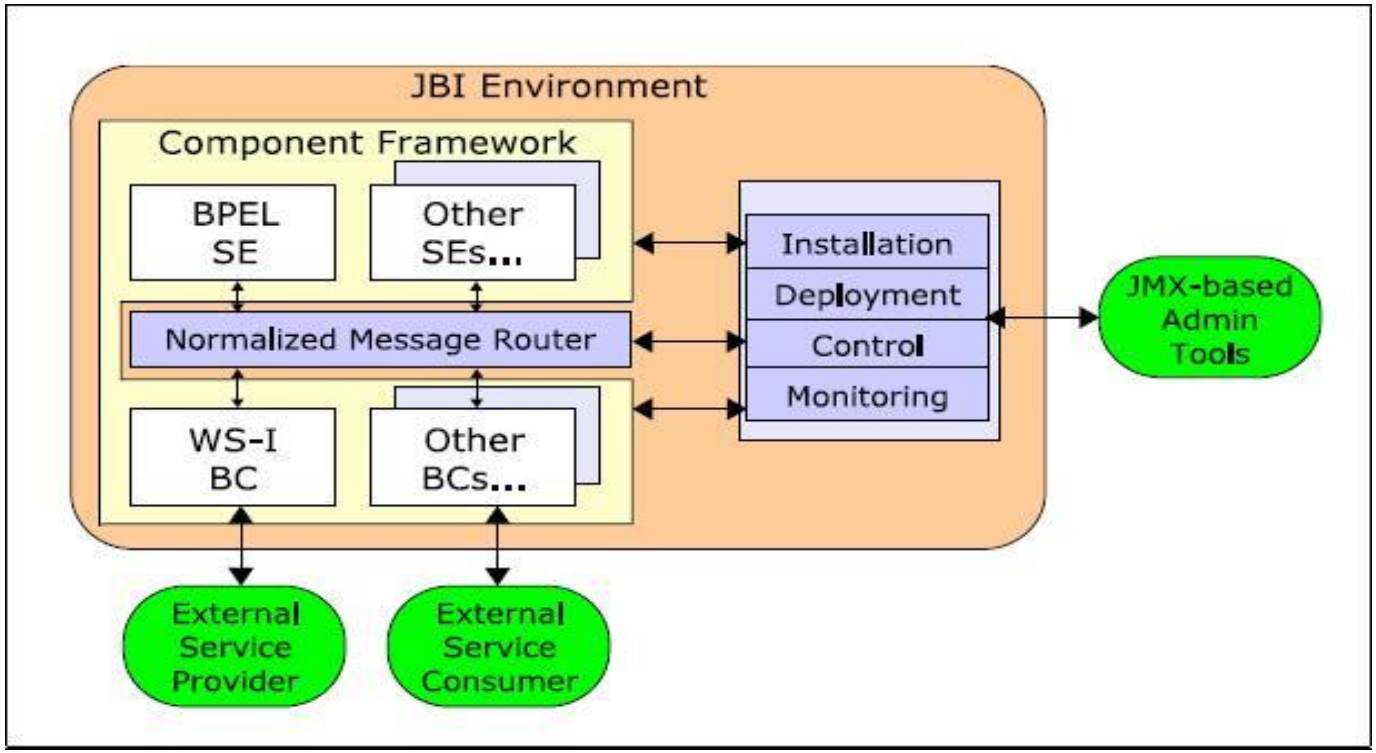
- SQL Server 2005
- Oracle 10G

Web Service Interoperability



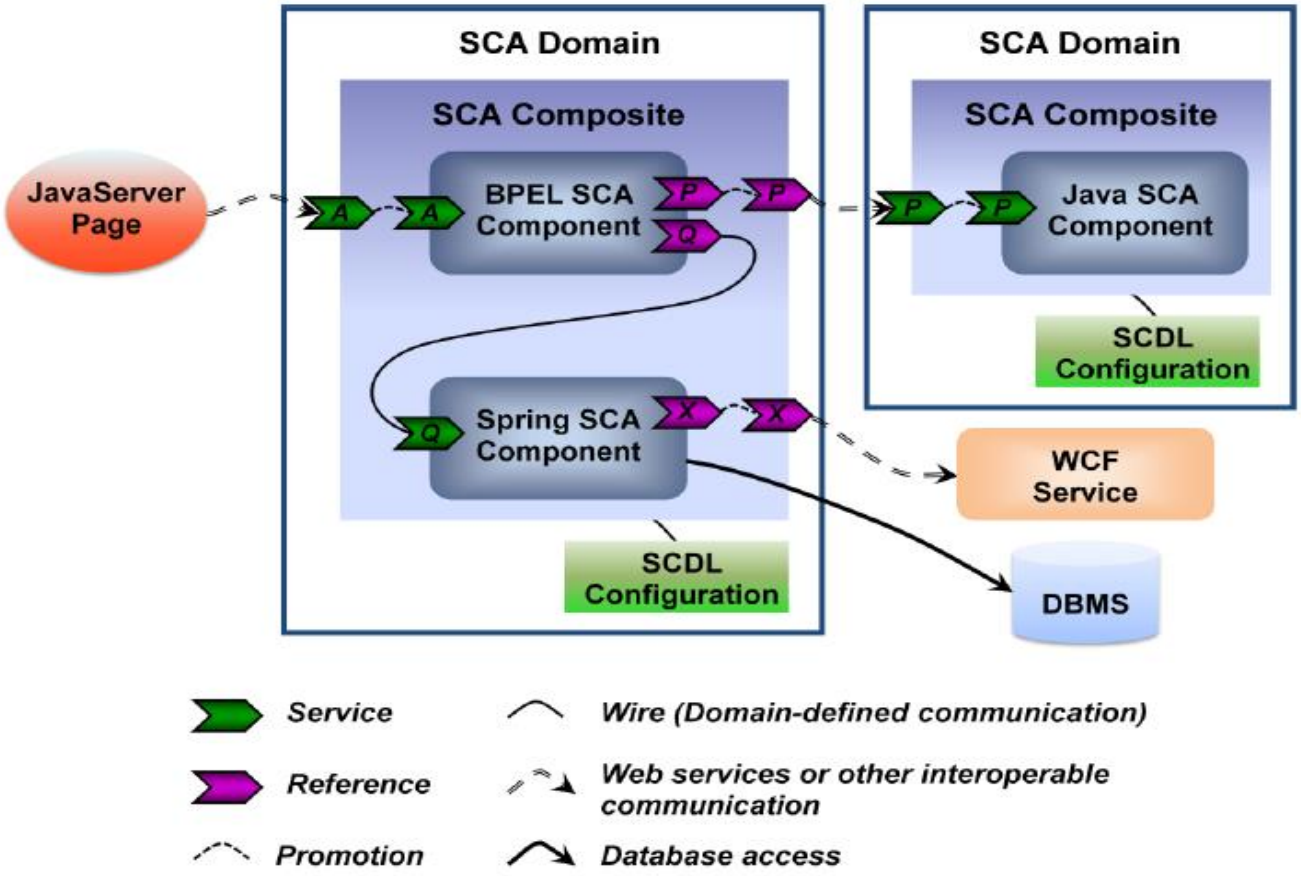
Models of SOAD

- **JBI（Java Business Integration）JSR 208**



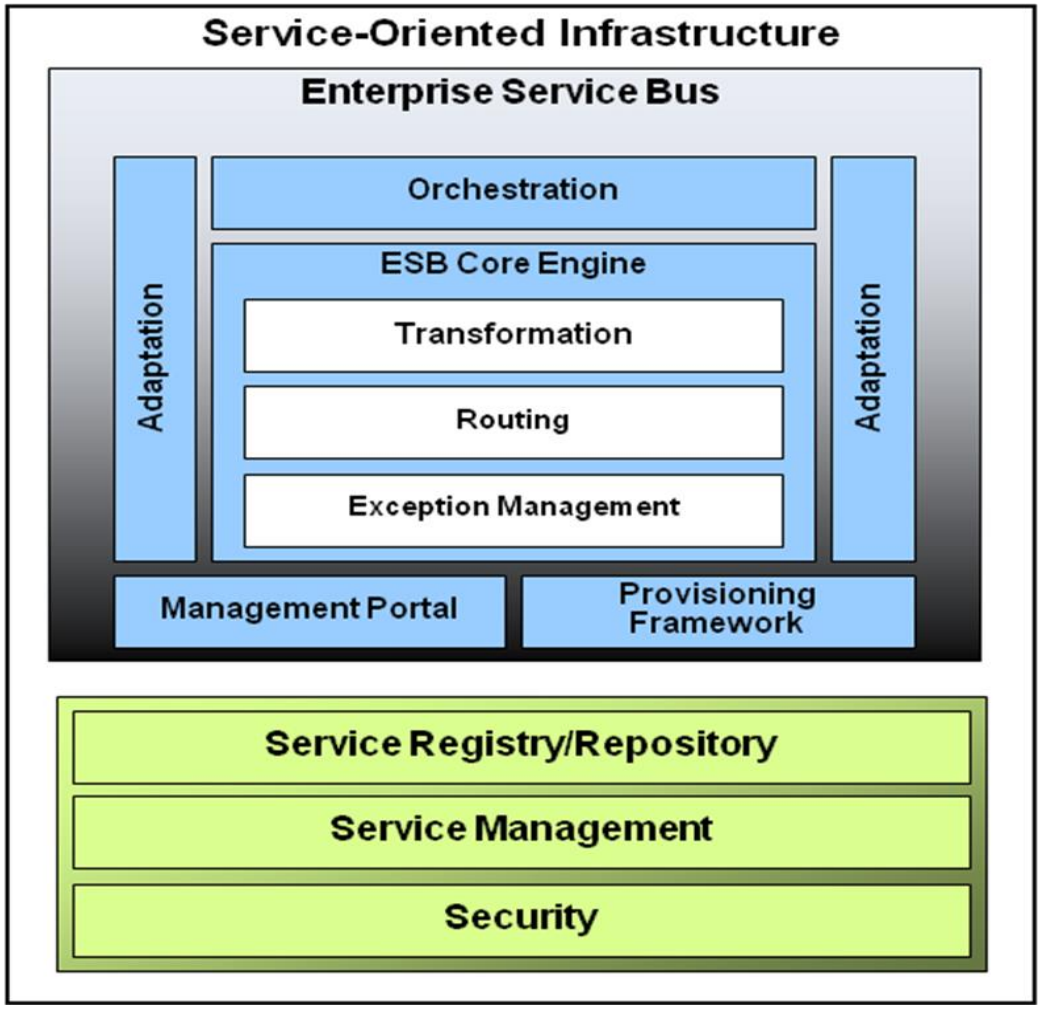
Models of SOAD

- **SCA(Service Component Architecture)/SDO(Service Data Object)**

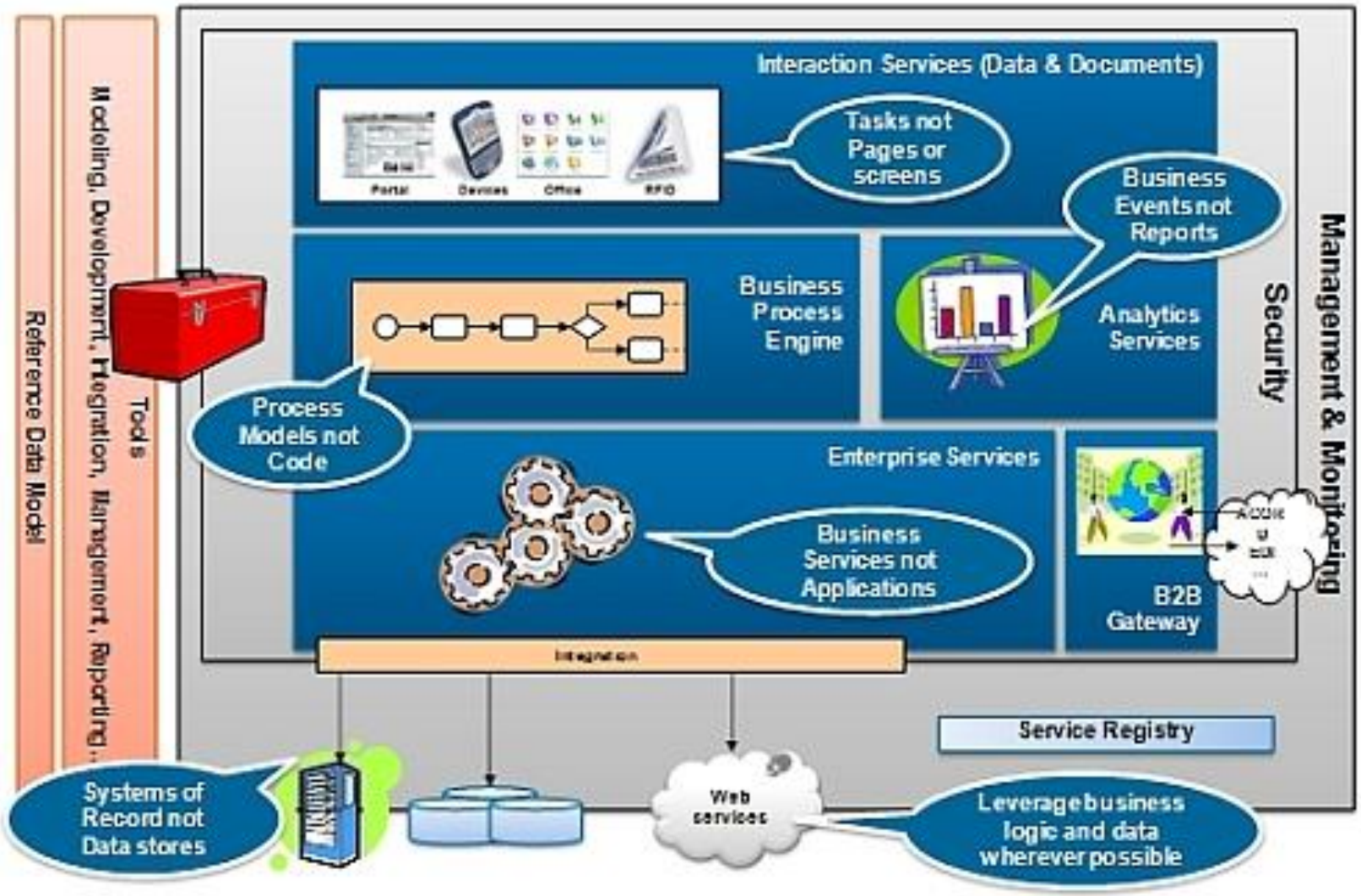


Models of SOAD

- **Windows Server + .NET Framework + BizTalk Server**

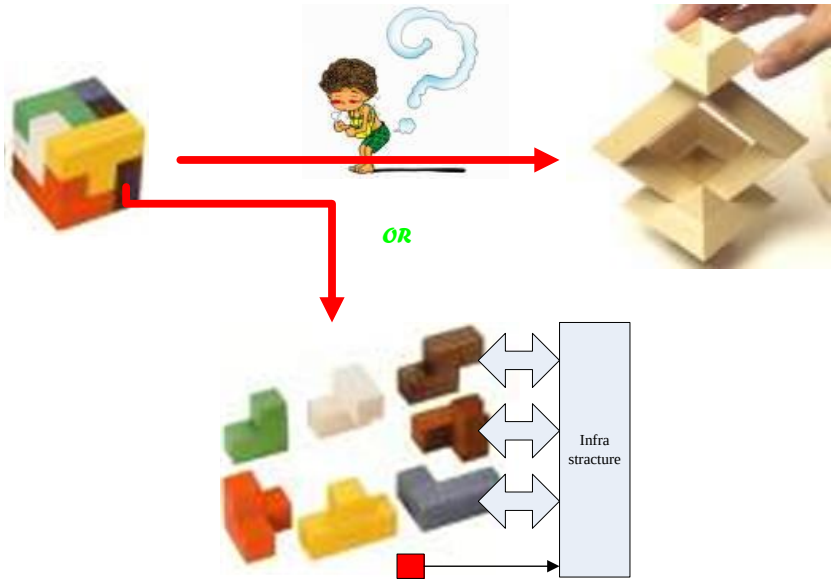


Typical SOA APPLICATIONS



Problems in IT Integration

- Isolated information in different Business Unit
- Various Info system platforms and applications
- Fixed business process
- New information systems are introduced
- .....



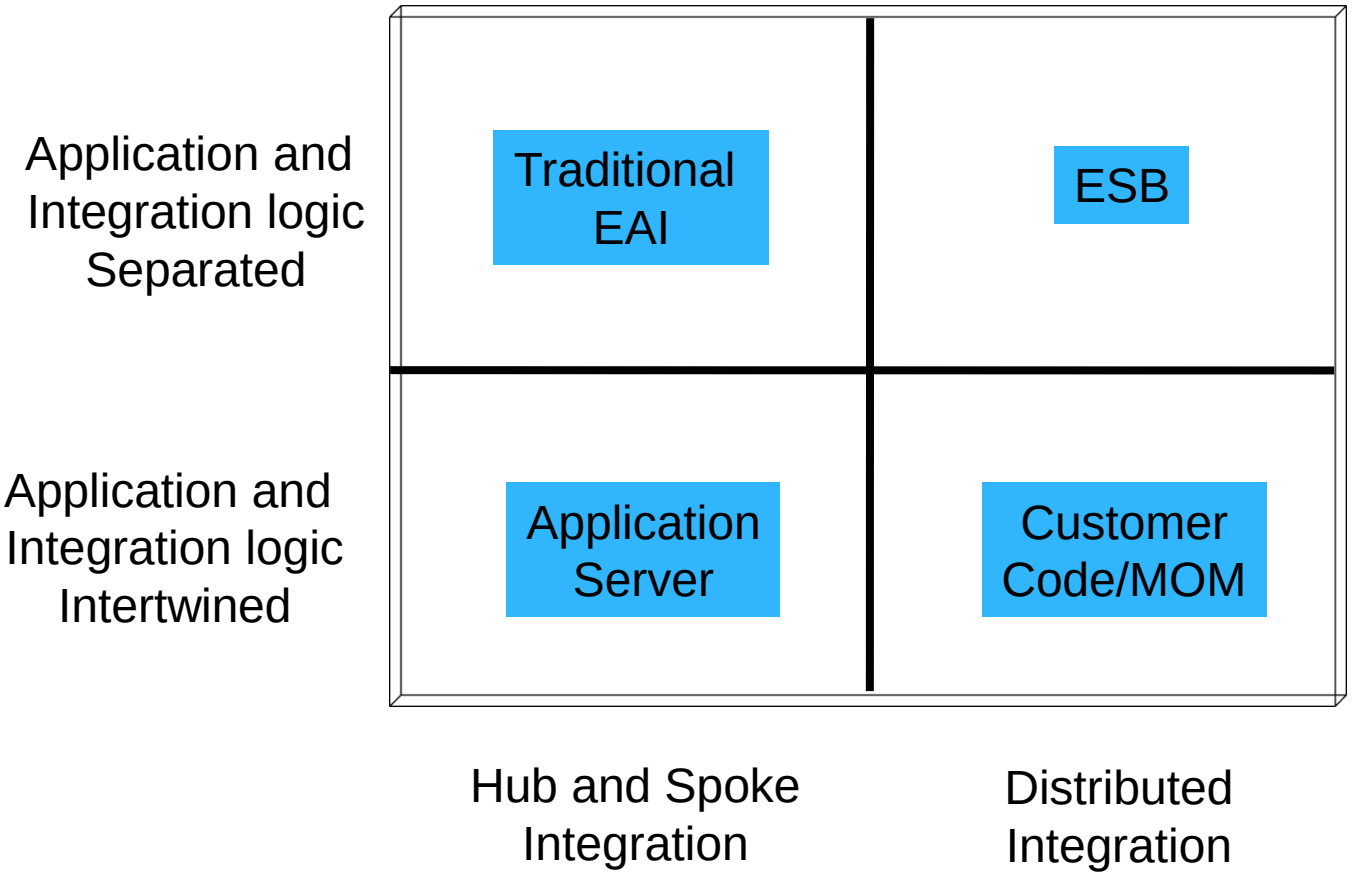
The Evolution of Web Services

	Today's Web services	Tomorrow's Web services infrastructure
Architecture	Tightly coupled, endpoint-centric	Loosely coupled, message-oriented
Security	Provided by every endpoint	Provided by a message gateway or firewall
Performance	Dependent on the application server	Load balanced across endpoint instances
Data transformation	Part of every endpoint and application	Data transformation engine attached to the network
Business processes	Processes tied to a single application	Reusable processes hosted in a process engine
Version control	Handled at the endpoint	Message router handles based on requester, header, and content
Service level agreements	Provided by the application	Implemented and monitored by a Qos engine

Forrester Research

Conventional Integration Approaches

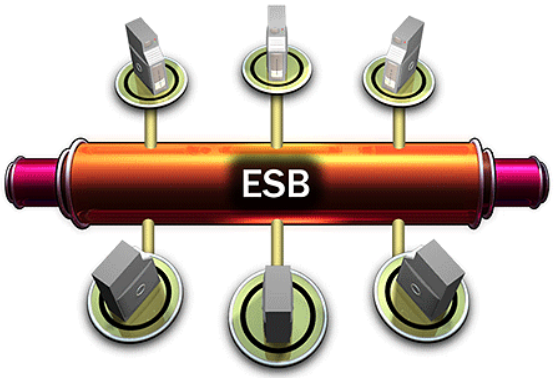
REliable, INtelligent & Scalable Systems



Necessity is the mother of invention

- ESB does not come of nowhere; many catalysts helped it develop and evolve. Lessons were learned from past technology approaches that extend back more than a decade.

- ESB is not merely an academic experience; it was born out of necessary, based on real requirements arising from difficult integration problems that couldn't be solved by any preexisting integration technology.

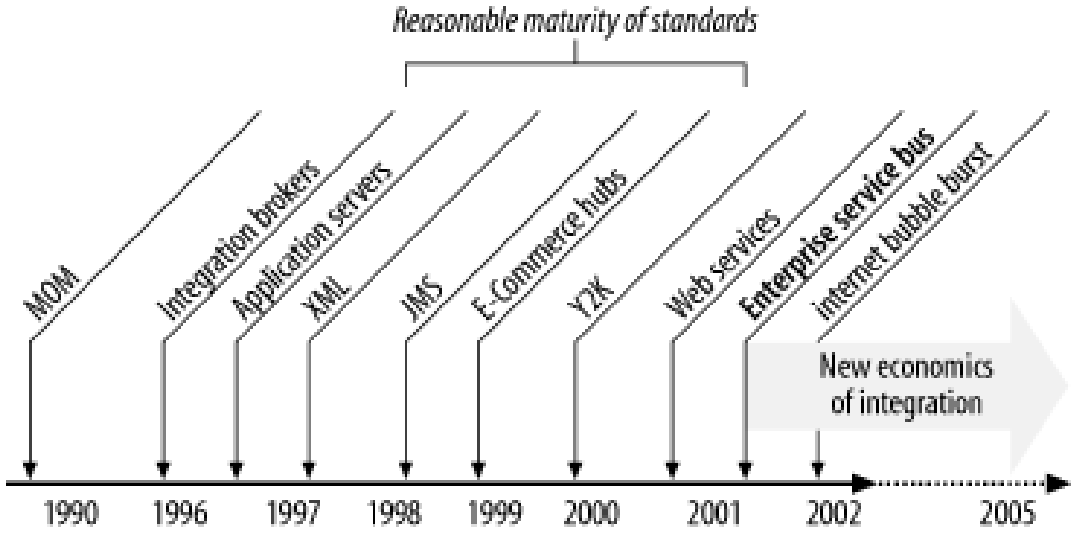


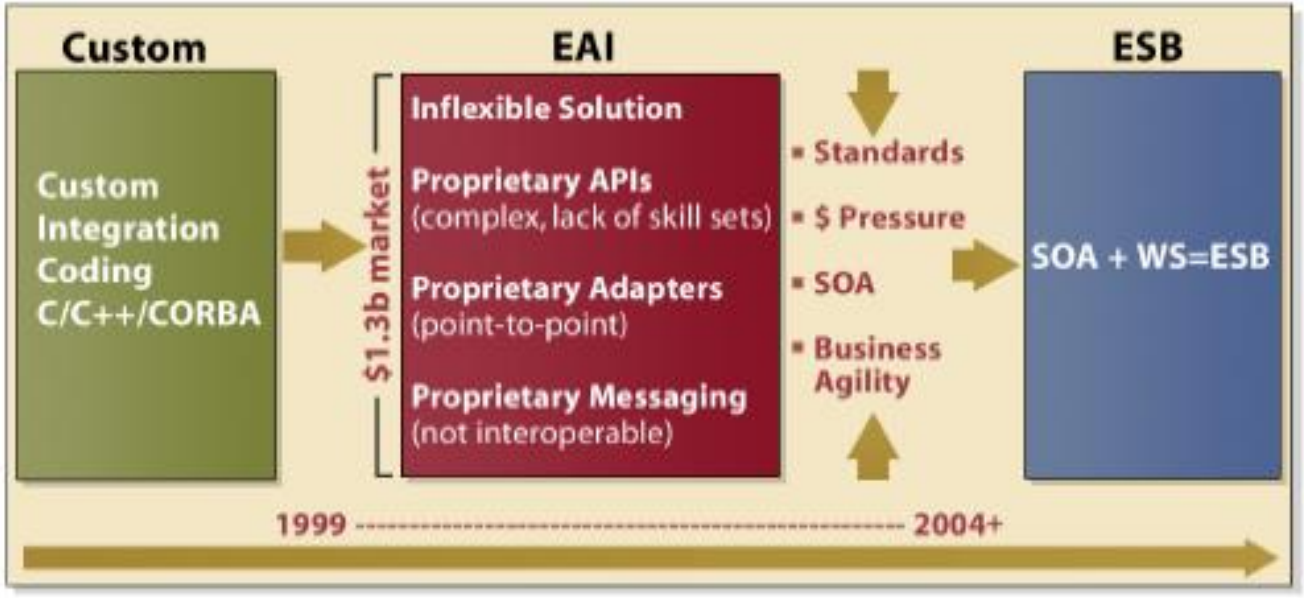
The Necessity of ESB

- Enterprise that wish to implement SOA need a more sophisticated, manageable infrastructure that can support high volumes of individual interactions.
- Such infrastructure should support more established integration styles
  - Message-oriented
  - Event-driven
  - Legacy integration
- Such infrastructure should support enterprise-level quality of service.

ESB is emerging as the unifying concept for such infrastructure

The Evolution of ESB

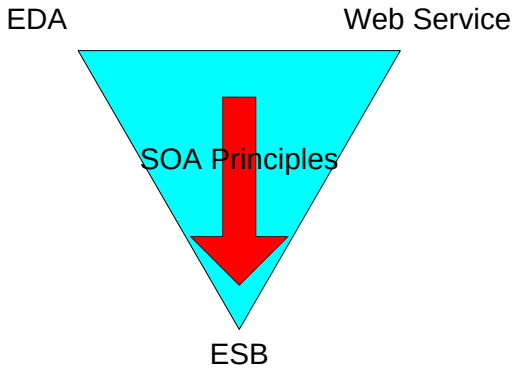




What is ESB

"A new form of *enterprise service bus (ESB)* infrastructure – combing *message-oriented middleware*, *web services*, *transformation* and *routing intelligence* – will be running in the majority of enterprise by 2005.

These high-function, low-cost ESBs are well suited to be the *backbone* for service-oriented architectures and the enterprise nervous system."



Roy Schulte, Gartner Report, 2002.

What is ESB

- ESB definition is still evolving.
  - *"software infrastructure* that enables *SOA* by acting as an intermediary layer of *middleware* through which a set of *reusable* business services are made widely available. "

    -- Forrester August 13, 2004

  - "Infrastructure software that makes *reusable* business services widely available to users, applications, business processes, and other services. "

    -- Forrester Q2,2006

What is ESB

- "(Enterprise Service Bus) providing a set of infrastructure capabilities, implemented by middleware technology, that enable the integration of services in an SOA.
  - Decoupling the consumer's view of a service from the actual implementation of the service
  - Decoupling technical aspects of service interactions
  - Integrating and managing services in the enterprise "

-- IBM, 2004

ESB Minimum Capabilities

Category	Capabilities	Reasons
Communications	► Routing ► Addressing ► At least one messaging style (request / response, pub/sub) ► At least one transport protocol that is or can be made widely available	Provide location transparency and support service substitution
Integration	► Several integration styles or adapters ► Protocol transformation	Support integration in heterogeneous environments and support service substitution
Service interaction	► Service interface definition ► Service messaging model ► Substitution of service implementation	Support SOA principles, separating application code from specific service protocols and implementations
Management	► Administration capability	A point of control over service addressing and naming

-- "Getting Started with Websphere ESB",
IBM Red Book (SG24-7212-00), 2006

ESB Extended Capabilities

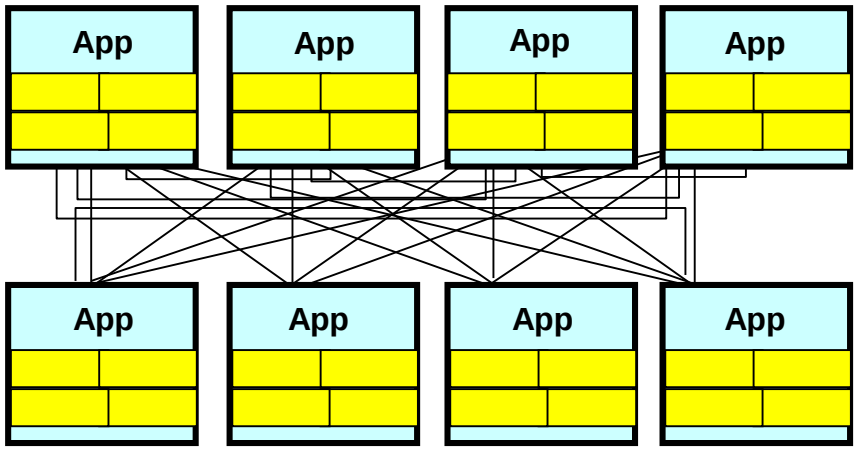
Communication	Service interaction
► Routing ► Addressing ► Protocols and standards (HTTP, HTTPS) ► Publish/subscribe ► Response/request ► Fire and forget, events ► Synchronous and asynchronous messaging	► Service interface definition (WSDL) ► Substitution of service implementation ► Service messaging models required for communication and integration (SOAP, XML, or proprietary Enterprise Application Integration models) ► Service directory and discovery
Integration	**Quality of service**
► Database ► Existing and application adapters ► Connectivity to enterprise application integration middleware ► Service mapping ► Protocol transformation ► Data enrichment ► Application server environments (J2EE and .Net) ► Language interfaces for service invocation (Java, C/C++/C#)	► Transactions (atomic transactions, compensation, WS-Transaction) ► Various assured delivery paradigms (WS-ReliableMessaging or support for Enterprise Application Integration middleware)
Security	**Service level**
► Authentication ► Authorization ► Non-repudiation ► Confidentiality ► Security standards (Kerberos, WS-Security)	► Performance (response time, throughput, and capacity) ► Availability ► Other continuous measures that might form the basis of contracts or agreements

The ESB Story

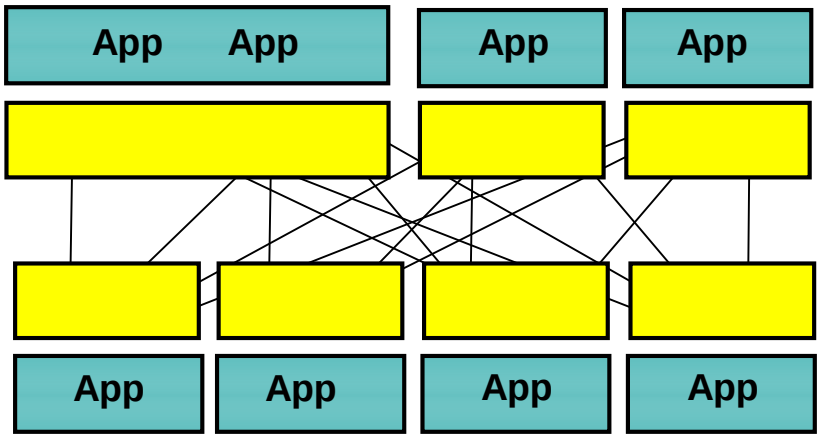
REliable, INtelligent & Scalable Systems

Loose-coupled, Standard-Based Integration

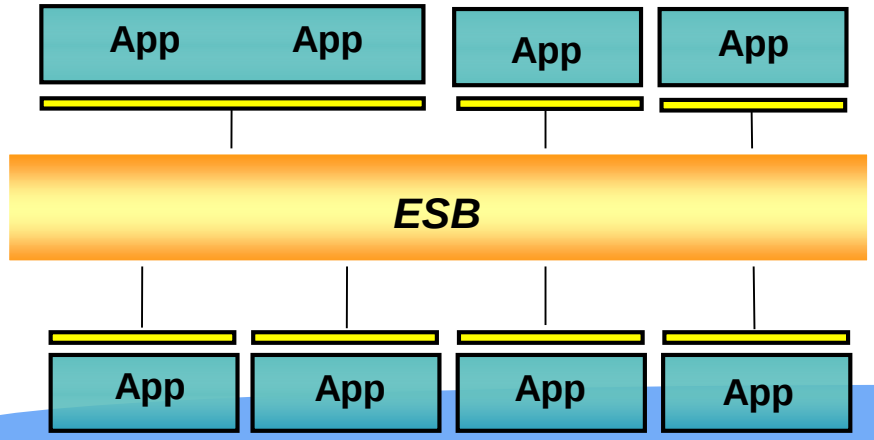
Integration any which way



Integration though Interfaces



Integration though ESB

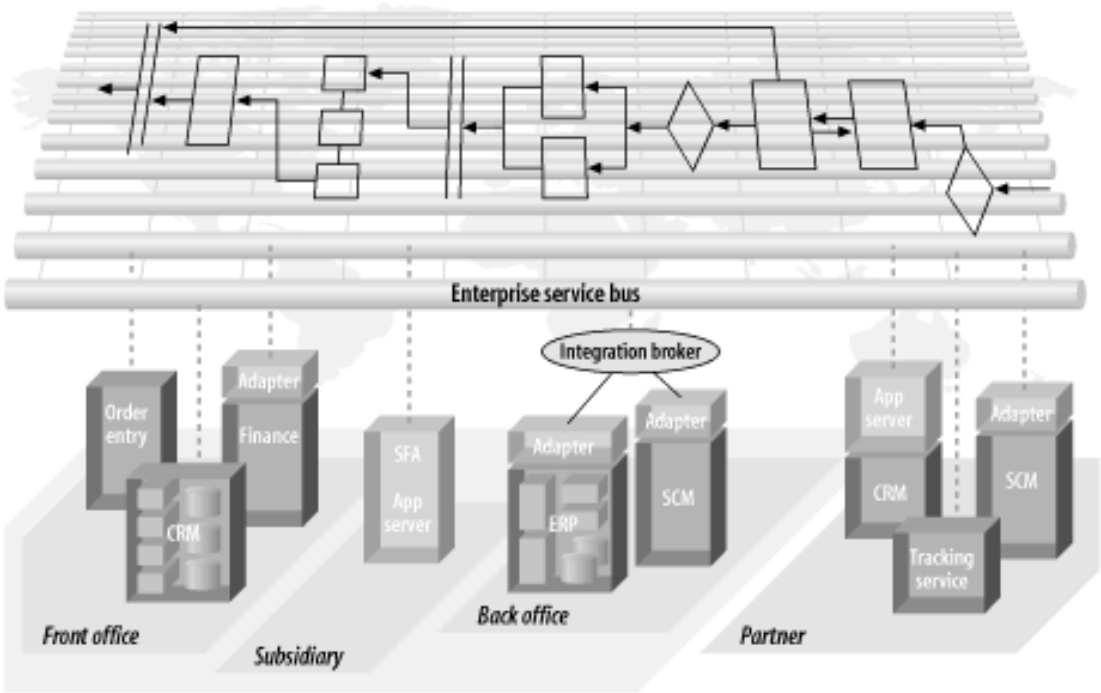


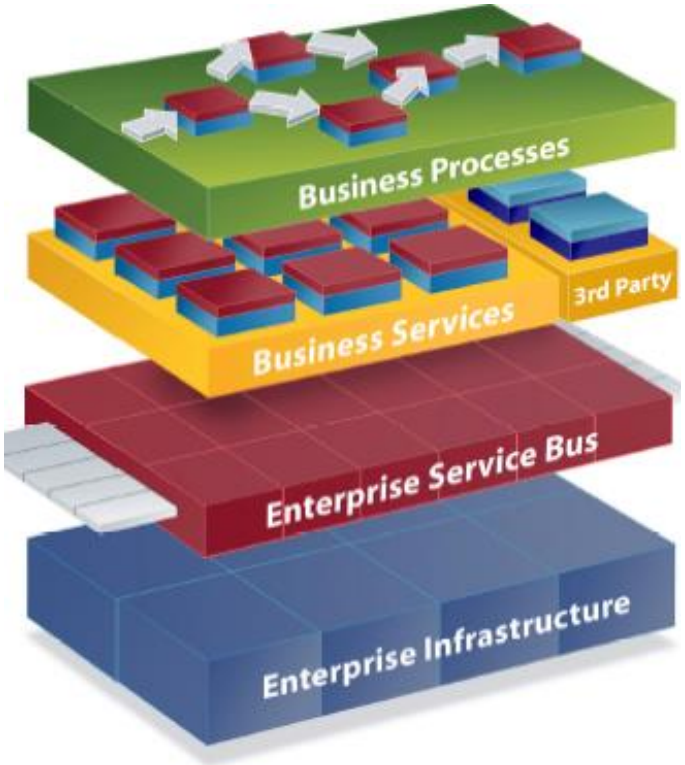
interface

The ESB Story

Highly Distributed Integration And Selective Deployment

Orchestration and process flow spanning highly distributed deployment topologies across physical and logical boundaries

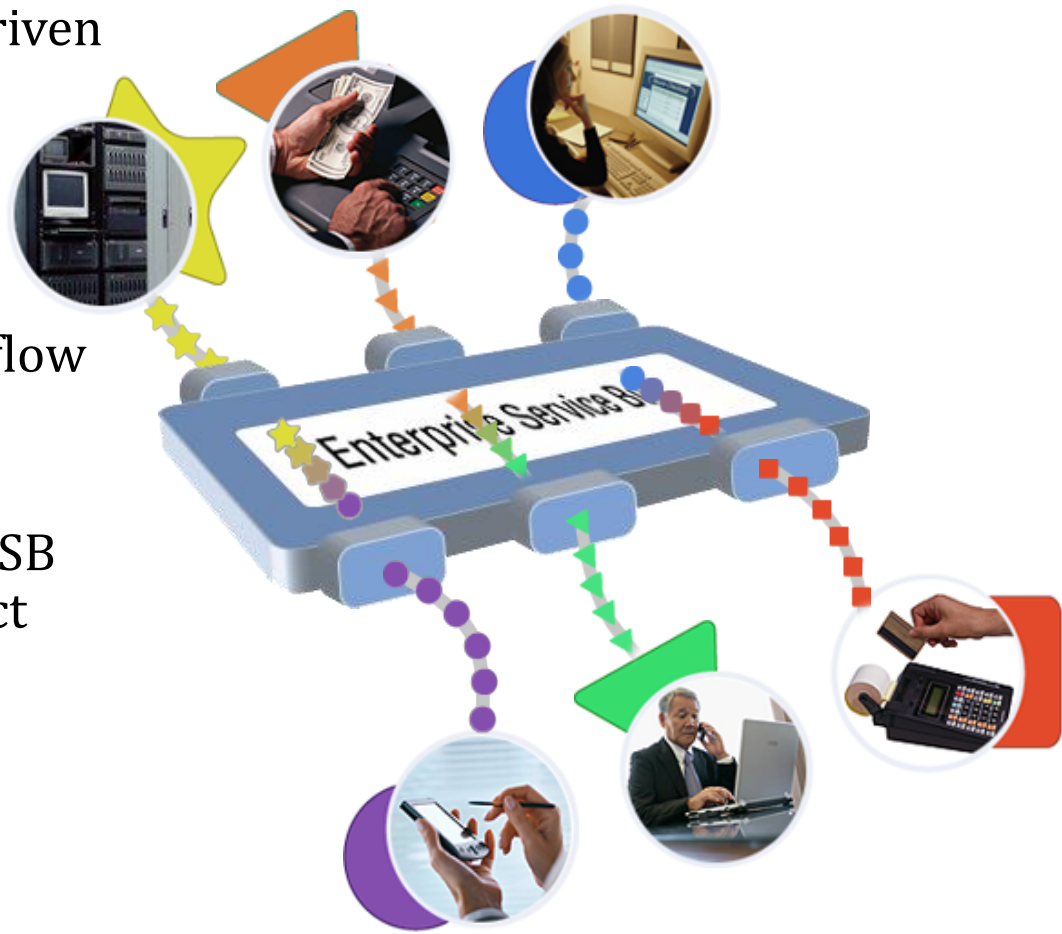




Extensibility through layered services

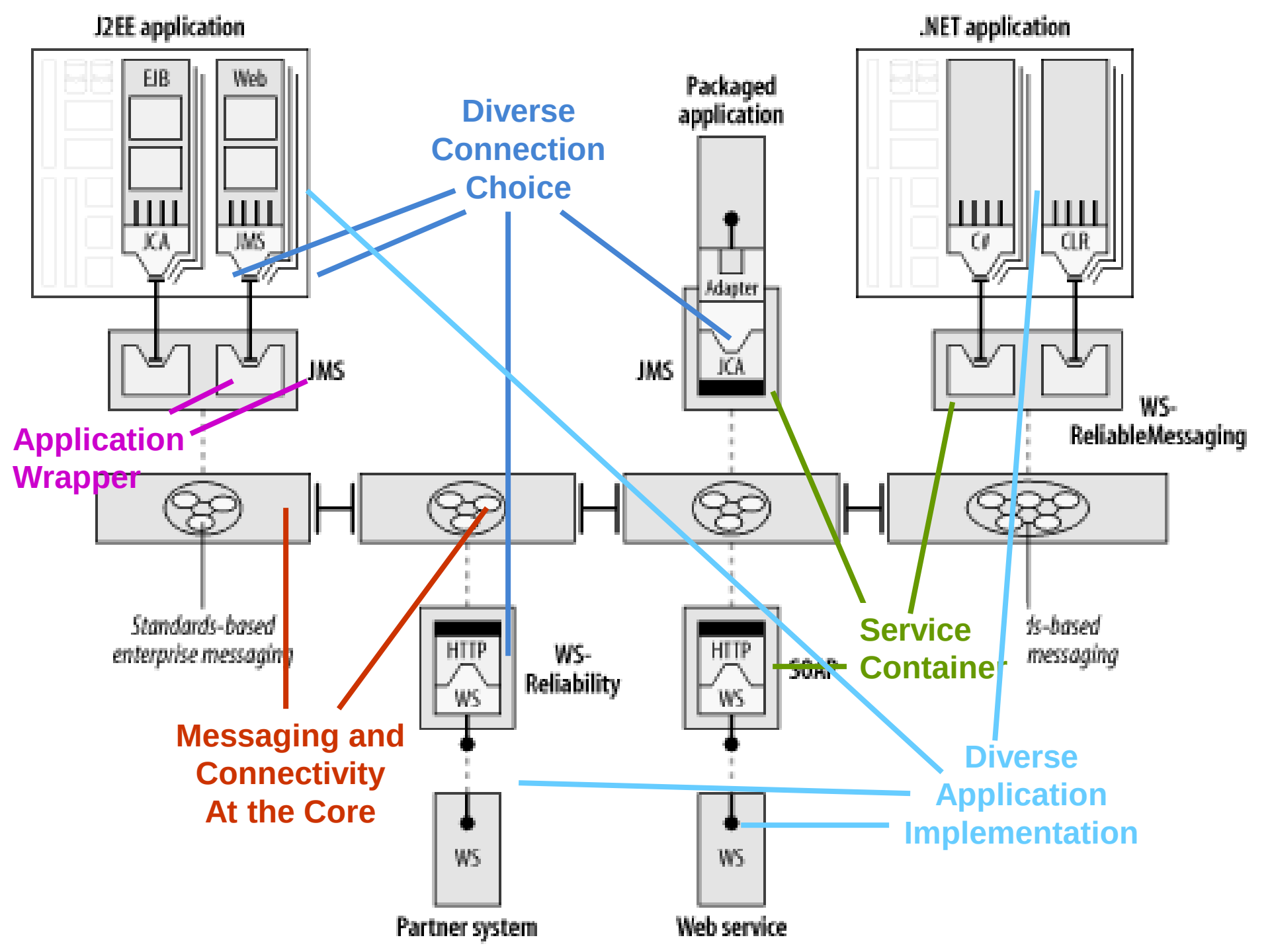
ESB Characteristics

- Pervasiveness
- Highly distributed, event-driven SOA
- Selective deployment of integration components
- Security and reliability
- Orchestration and process flow
- Autonomous yet federated managed environment
- Incremental adoption. An ESB can be used for small project
- XML support
- Real-time insight



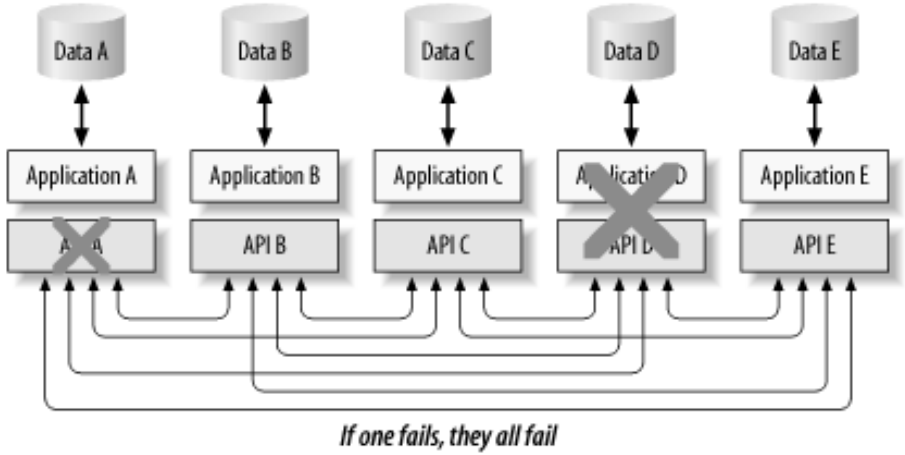
ESB Key Components

- Message Oriented Middleware
  - Robust, reliable transport
  - Efficient movement of data across abstract data channels
  - End-to-end reliability
- Service Container and Abstract Endpoints
  - Endpoints
    - Logical abstraction, representing remote services in various implementations
  - Container
    - The physical manifestation of the endpoints
    - Distributed and lightweight
- Intelligent routing
  - Message routing based on content and context
  - Message routing based on business process rules
  - Business process orchestration based on a rules language such BPEL4WS


J2EE application
EJB
Web
JCA
JMS
.NET application
C#
CLR
Packaged application
Adapter
JCA
JMS
JMS
WS-ReliableMessaging
Diverse Connection Choice
Application Wrapper
Standards-based enterprise messaging
HTTP
WS
WS-Reliability
HTTP
WS
SOAP
Service Container
ls-based messaging
Messaging and Connectivity At the Core
WS
WS
Diverse Application Implementation
Partner system
Web service

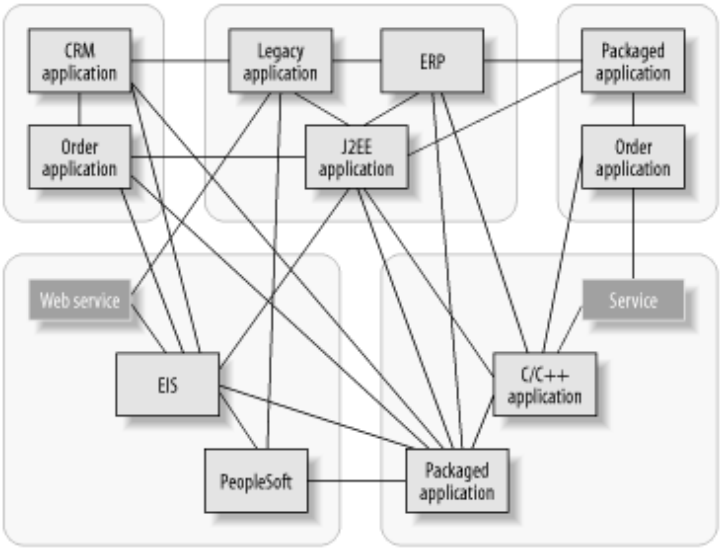
Tightly Coupled Versus Loosely Coupled Interfaces

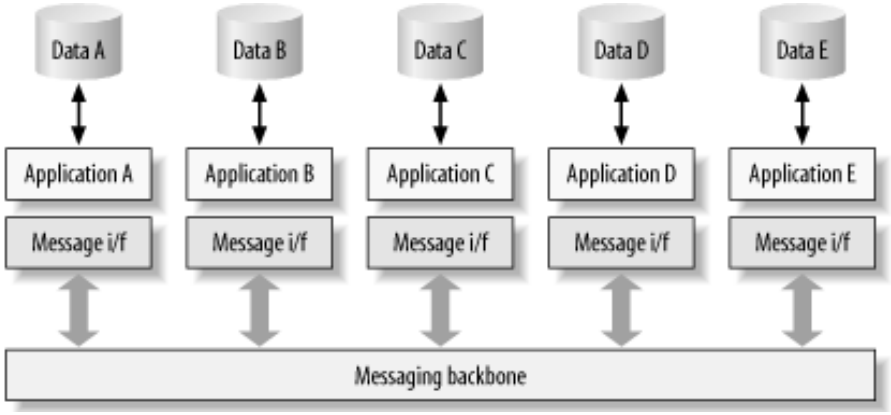
- Remote Procedure Call
  - is a protocol that allows a computer program running on one computer to cause a subroutine on another computer to be executed without the programmer explicitly coding the details for this interaction.
  - an easy and popular paradigm for implementing the client-server model of distributed computing
  - A synchronous operation across multiple processes
    - All-or-Nothing



Tightly Coupled Versus Loosely Coupled Interfaces

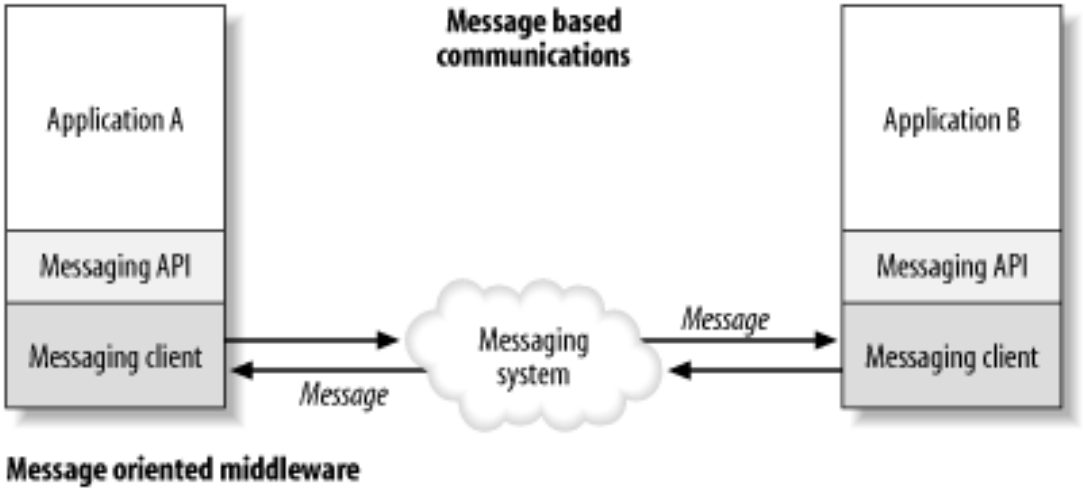
- Tightly coupled interfaces
  - Each application needs to know the intimate details of how every other application wants to be communicated with
  - The number of interfaces becomes unwieldy when the system scale up: n(n-1)/2
- Loosely coupled interface
  - Self-contained, stand alone unit
  - Asynchronous message:
  - Reduces the number of interfaces from $O(n^2)$ to $O(n)$





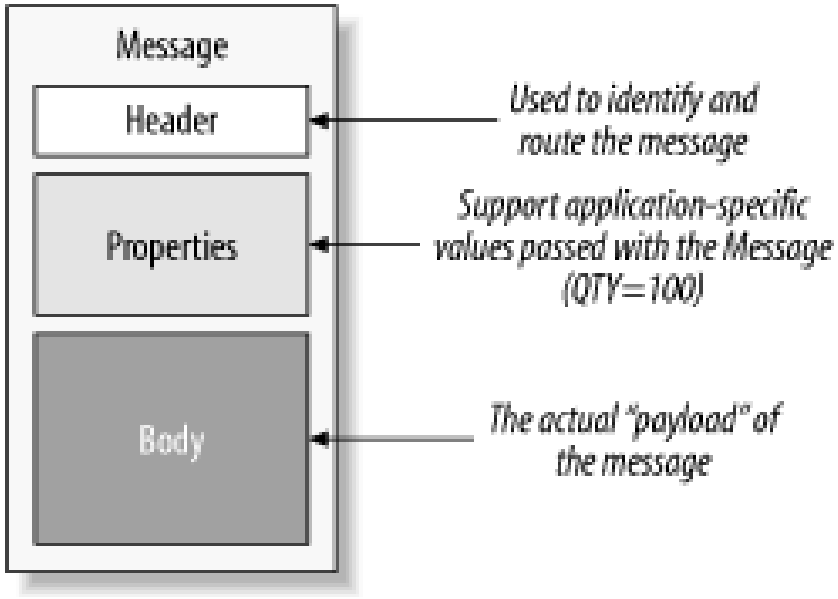
Message Oriented Middleware (MOM)

- Virtual channels that an ESB uses to route messages
- Self-contained units of information (messages)
- Asynchronous communication
- Applications are abstractly decoupled
- Messaging system supports the management of connection points between multiple messaging clients, and of multiple channels of communication between connection points.
  - Message server
  - Message broker



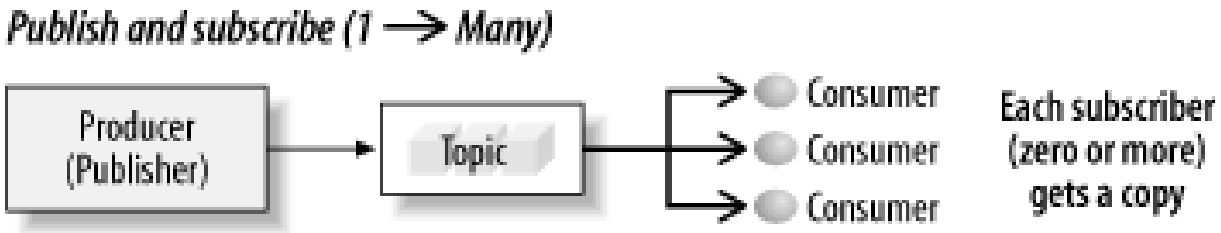
Message

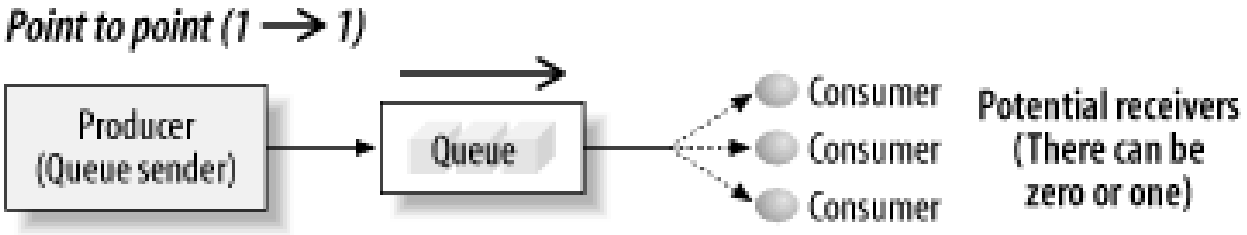
- The header
  - Basic information
    - Destination, reply-to, message type, etc.
- Properties
  - Application-defined name/value pairs
  - For filtering by consumer or routers
- Body
  - Plain text, raw data, XML message



MOM Concepts

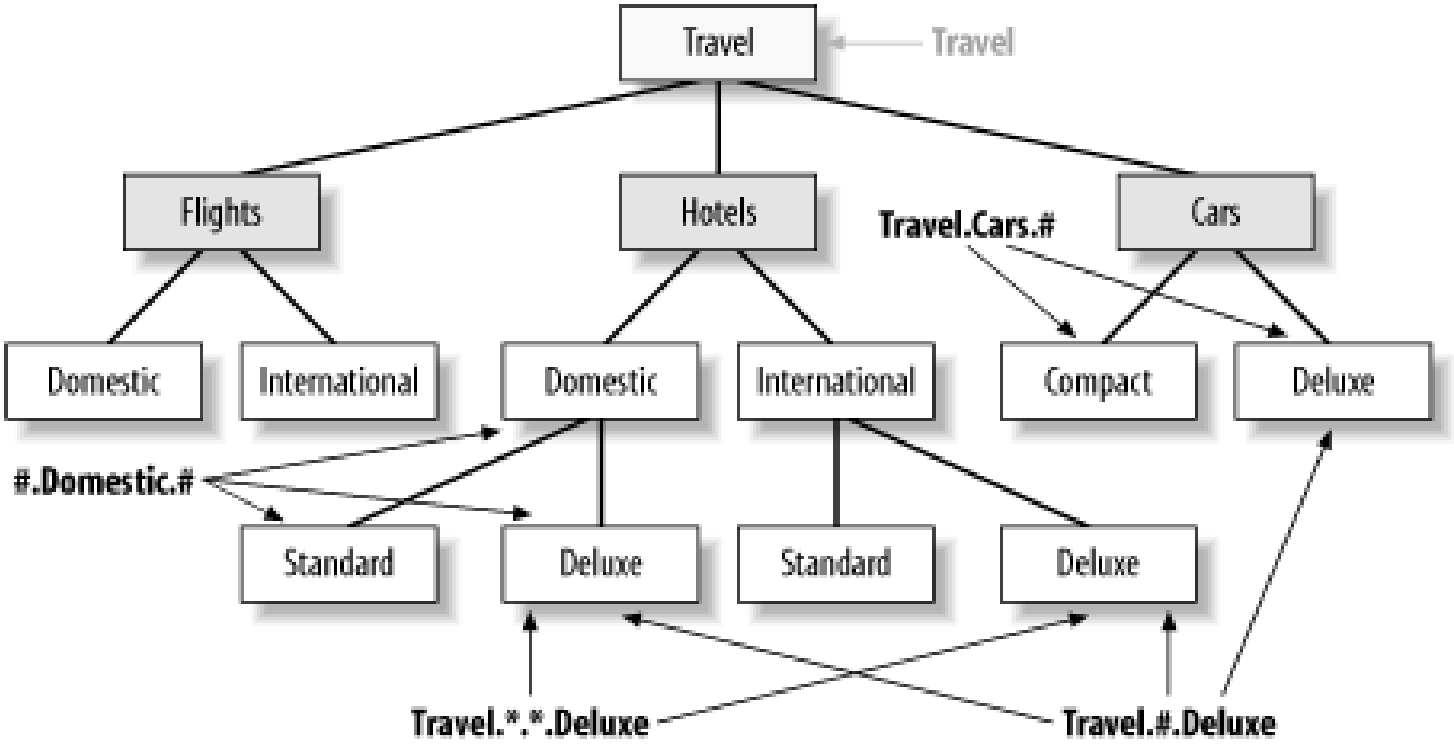
- Abstract Decoupling
  - The producer and consumer do not know each other
- Messaging Models
  - Publish-and-Subscribe
    - one to many broadcast of information
  - Point-to-Point
    - One-to-one communication between two specific applications





MOM Concepts

- Topic Hierarchies
  - Allow for wildcard-based subscriptions at any level
- Access Control Lists
  - Grant access to different levels of the hierarchy



Asynchronous Reliability

- Message autonomy
  - Messages are self-contained, autonomous entities
  - Producer sends a message
  - Messaging system guarantees that it is received by any interested parties
  - ESB ensures that it arrives in the desired data format

Asynchronous Reliability

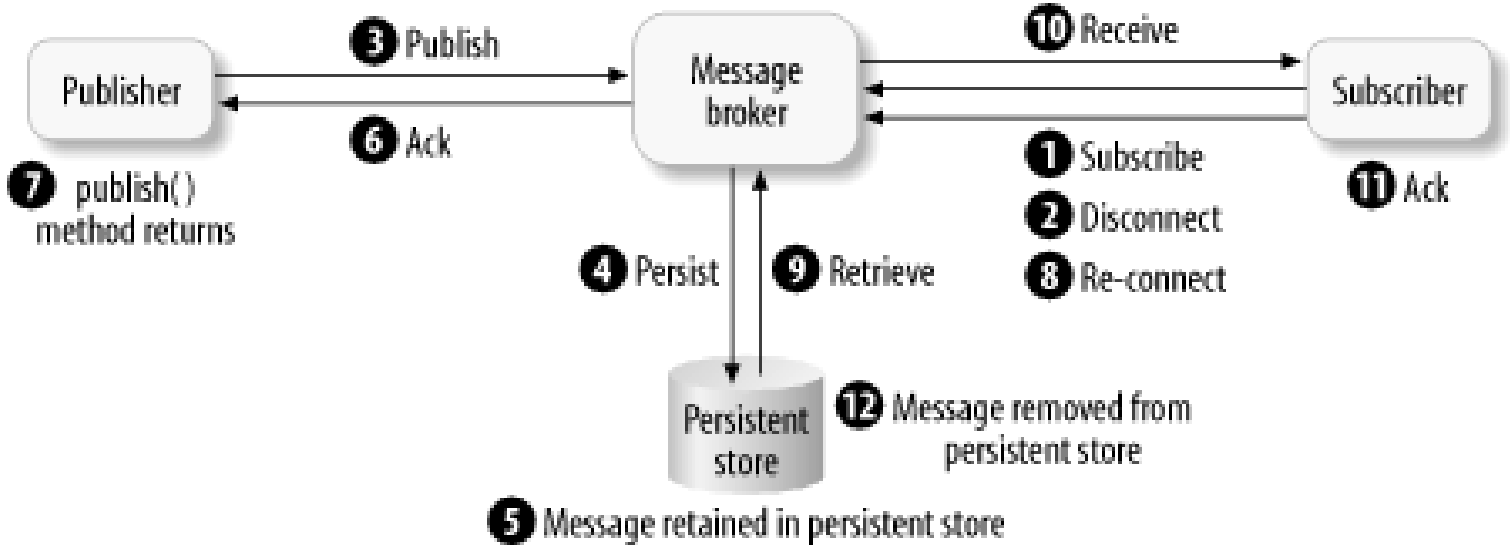
- Store and Forward
  - Message queuing and guaranteed delivery
    - Exactly once
    - At least once
    - At most once
  - Message ordering
    - Messages are delivered to the receiver in the same order in which they are sent by the sender

Asynchronous Reliability

- Message Acknowledgment
  - Allow the messaging system to monitor the progress of a message so that it knows whether the message was successfully produced and consumed.

Reliable Messaging Models

- Reliable Publish-and-Subscribe
  - Persistent messages
  - Durable subscriptions

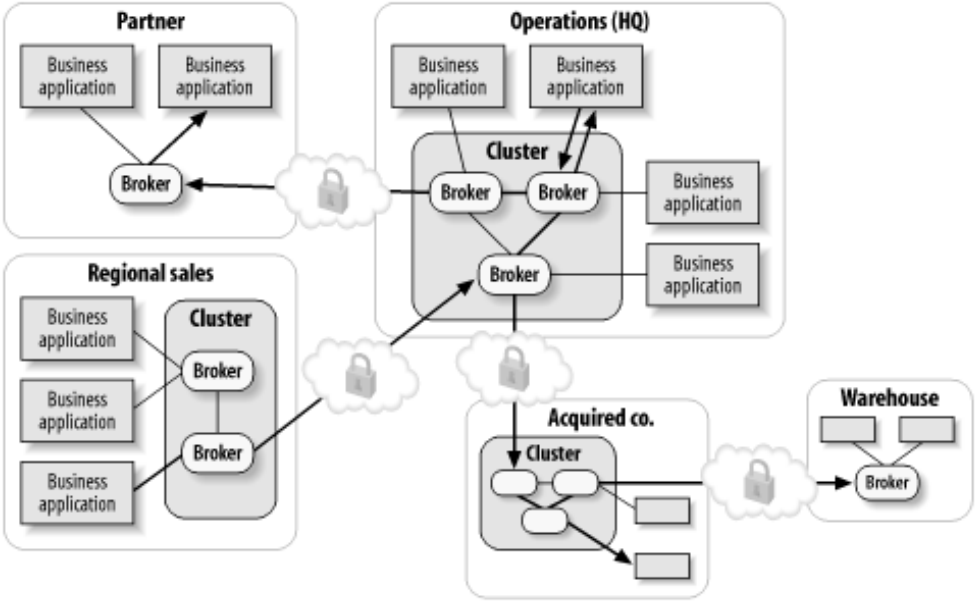


Reliable Messaging Models

- Reliable Point-to-Point Queues
  - A persistent message stays in the queue until it is delivered to the consumer or it expires.
  - A non-persistent message also stays in the queue until it is delivered or expired, but it is not guaranteed to survive a failure and recovery of the messaging server.

Reliable Messaging Models

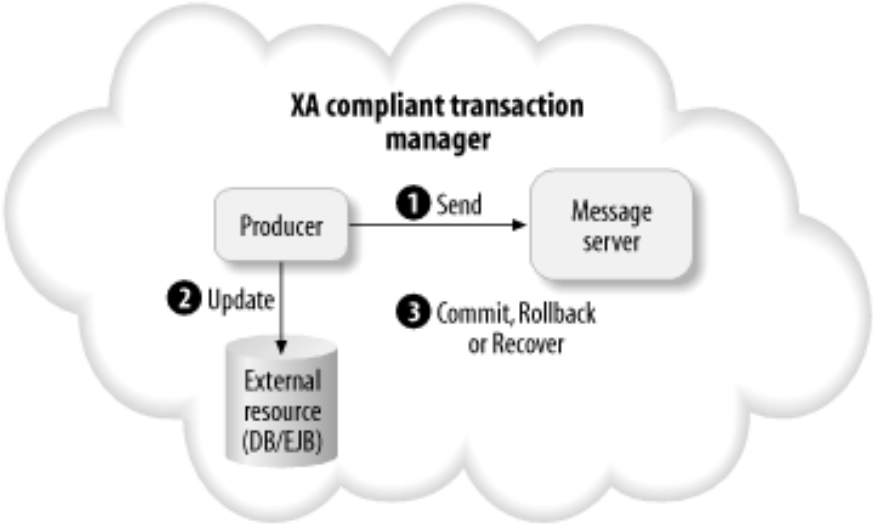
- Multi-step store-and-forward
  - Each message server uses store-and-forward and message acknowledgements to get the message to the next server in the chain.
  - Each link can be secure, authenticated and capable of traversing through firewall boundaries.



Transacted Messages

Receive and Send operations can be grouped together in a single local transaction

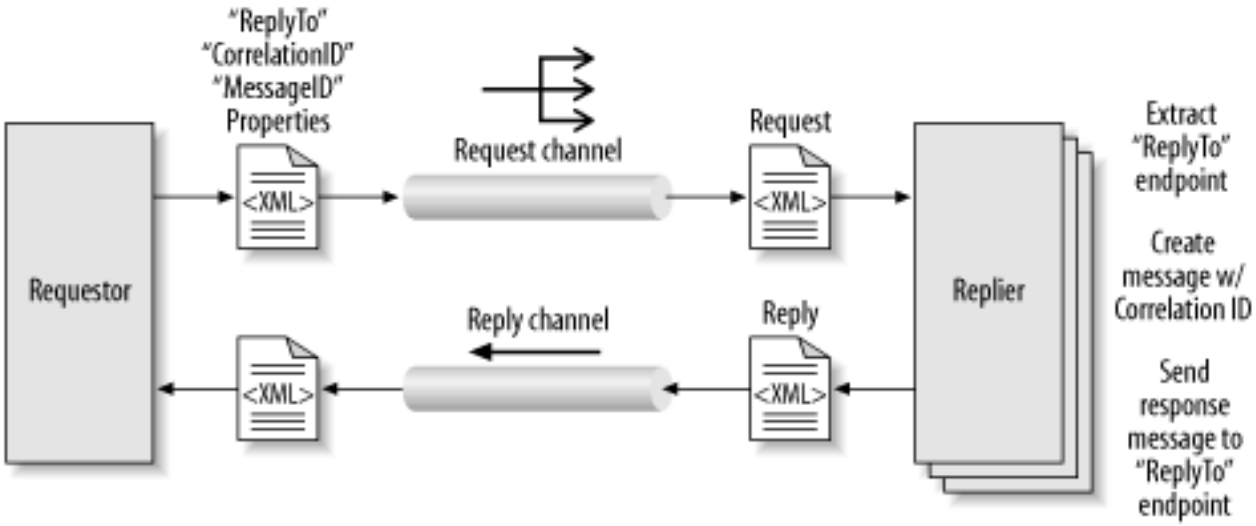
Multiple resources participate in a two-phase-commit Transaction



An ESB Removes the Low-Level Complexities

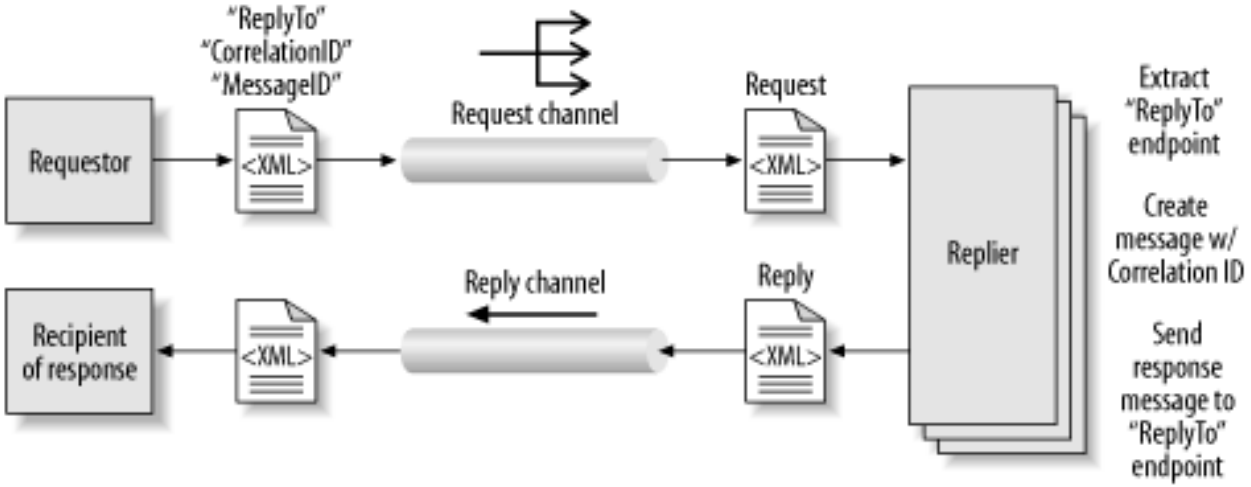
The Request/Reply Messaging Pattern

- The patterns can be built on top of a MOM to perform synchronous request/reply or asynchronous request/reply.
- An ESB can further automate this process by managing the details in ESB container.



The Request/Reply Messaging Pattern

- The patterns can be built on top of a MOM to perform synchronous request/reply or asynchronous request/reply.
- An ESB can further automate this process by managing the details in ESB container.



Messaging Standards

- Java Message Service (JMS)
  - Messaging Specification, 1998
  - Defines the API and a set of rules that govern message delivery semantics in a MOM environment for both reliable and unreliable messaging.
- Simple Object Access Protocol (SOAP)

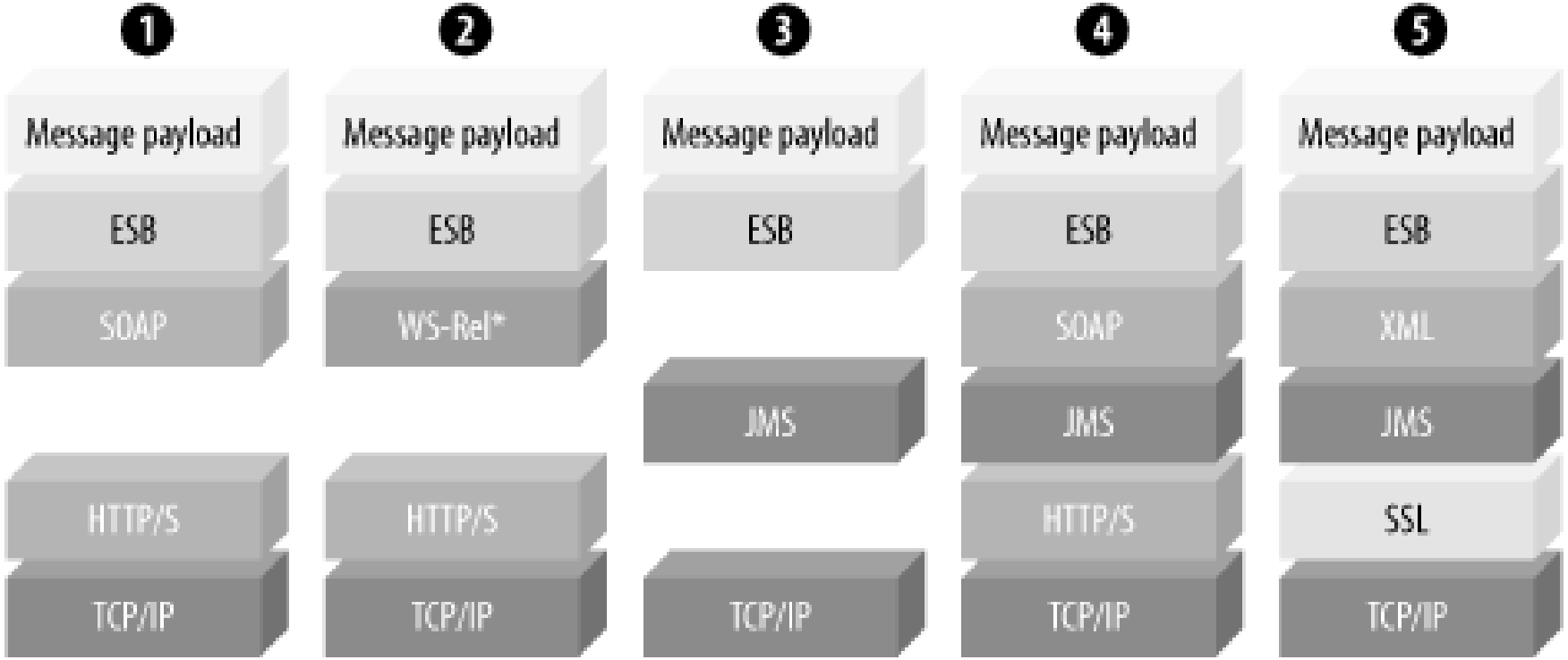
Messaging Standards

- Web Services Events (WS-Eventing)
  - Defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties
  - April 2004, submit to OASIS, similar to WS-BaseNotification
- Web Services Notifications (WS-Notification)
  - Define a standard Web services approach to notification using a topic-based publish/subscribe pattern
  - WS-BaseNotification, WS-BrokeredNotification and WS-Topics specifications
  - October 2006, OASIS standards

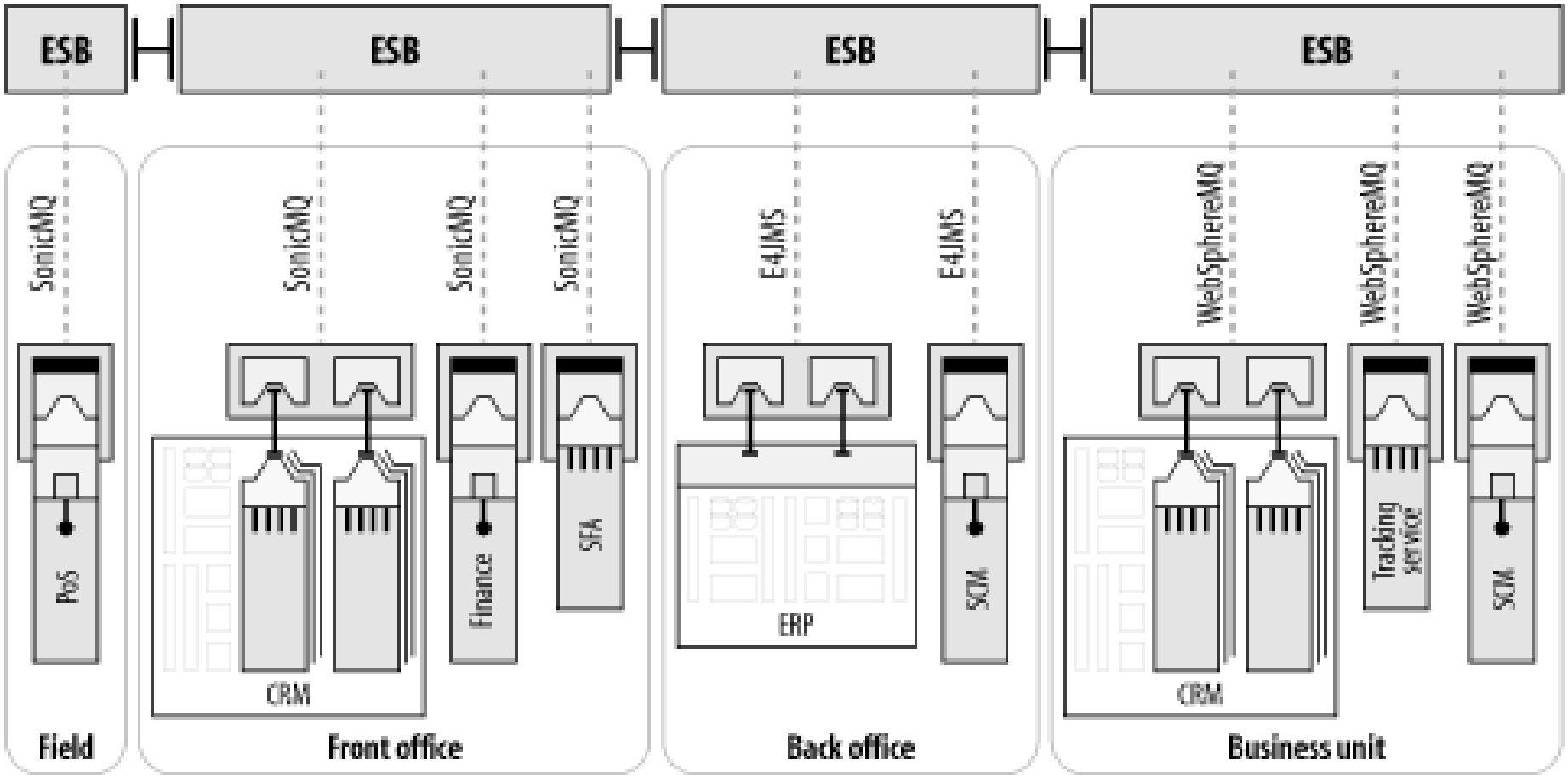
Summary

- MOM provides the backbone for enterprise data exchange
  - Message acknowledgement, message persistence, and transactions.
- Message systems ensure reliable message exchange
  - Contract between producer and message server: ensure reliable message delivery
  - Contract between message server and consumer: ensure reliable message acceptance
- Messaging middleware may be an appropriate transport protocol when there is a requirement for Web services to communicate:
  - Asynchronously, where the sender of a message does not wait for a reply to the message
  - Reliably, where the sender is assured that the message will be delivered
- ESB can encapsulate the low-level details in a container-managed environment

A Generic Message Invocation Framework

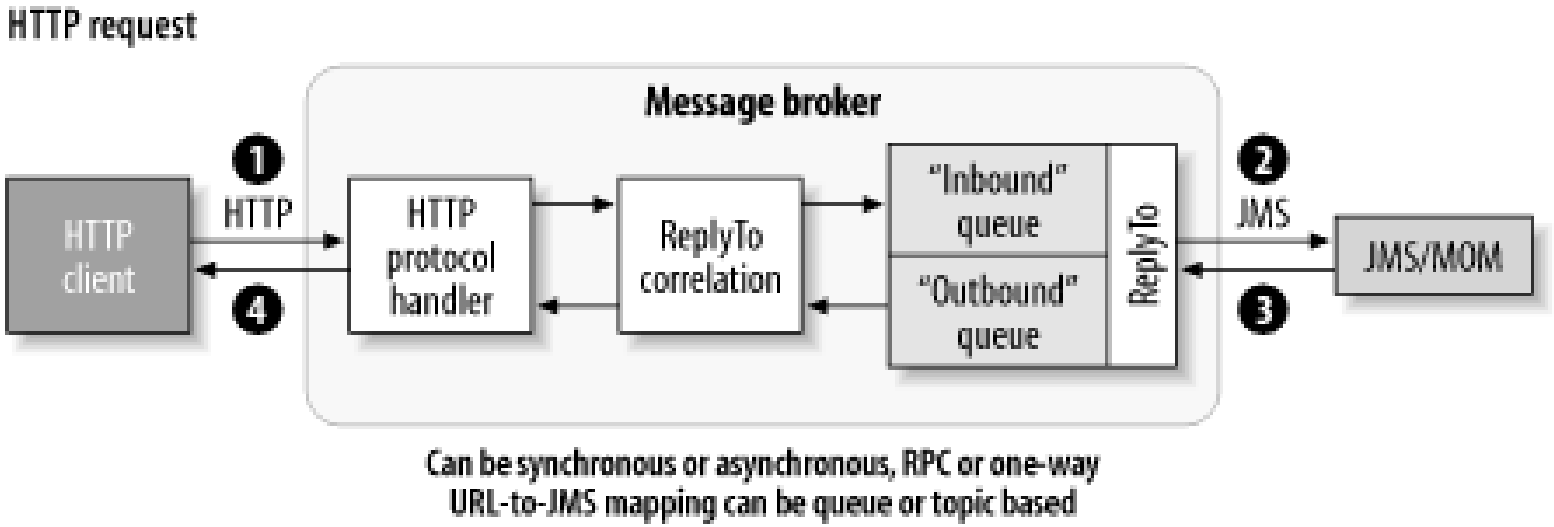


MOM Bridging



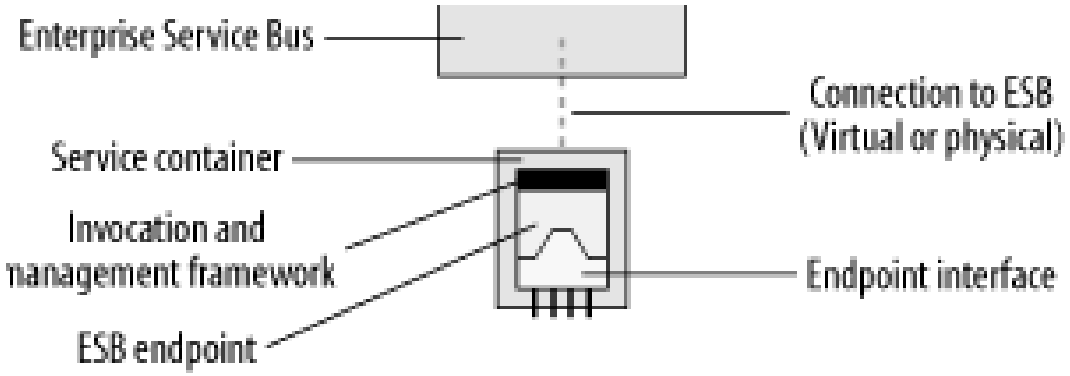
Direct Protocol Handlers

- The message broker inside ESB provides the bridging, or mapping between the external protocols and the internal MOM channels.



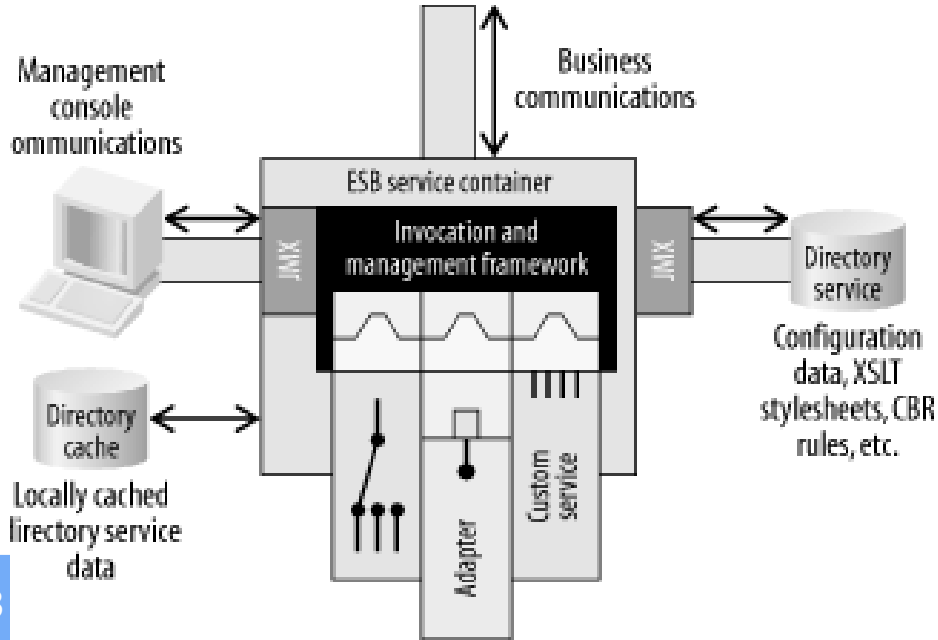
Service Endpoints

- Endpoints are logic Abstractions of services that are plugged into the bus.
- The actual representation of an endpoint could be diverse
  - A single application
  - A suite of applications
  - A business unit
- The underlying implementation of an endpoint is hidden from the integration point of view
  - A local binding to an application adapter
  - A callout to an external service

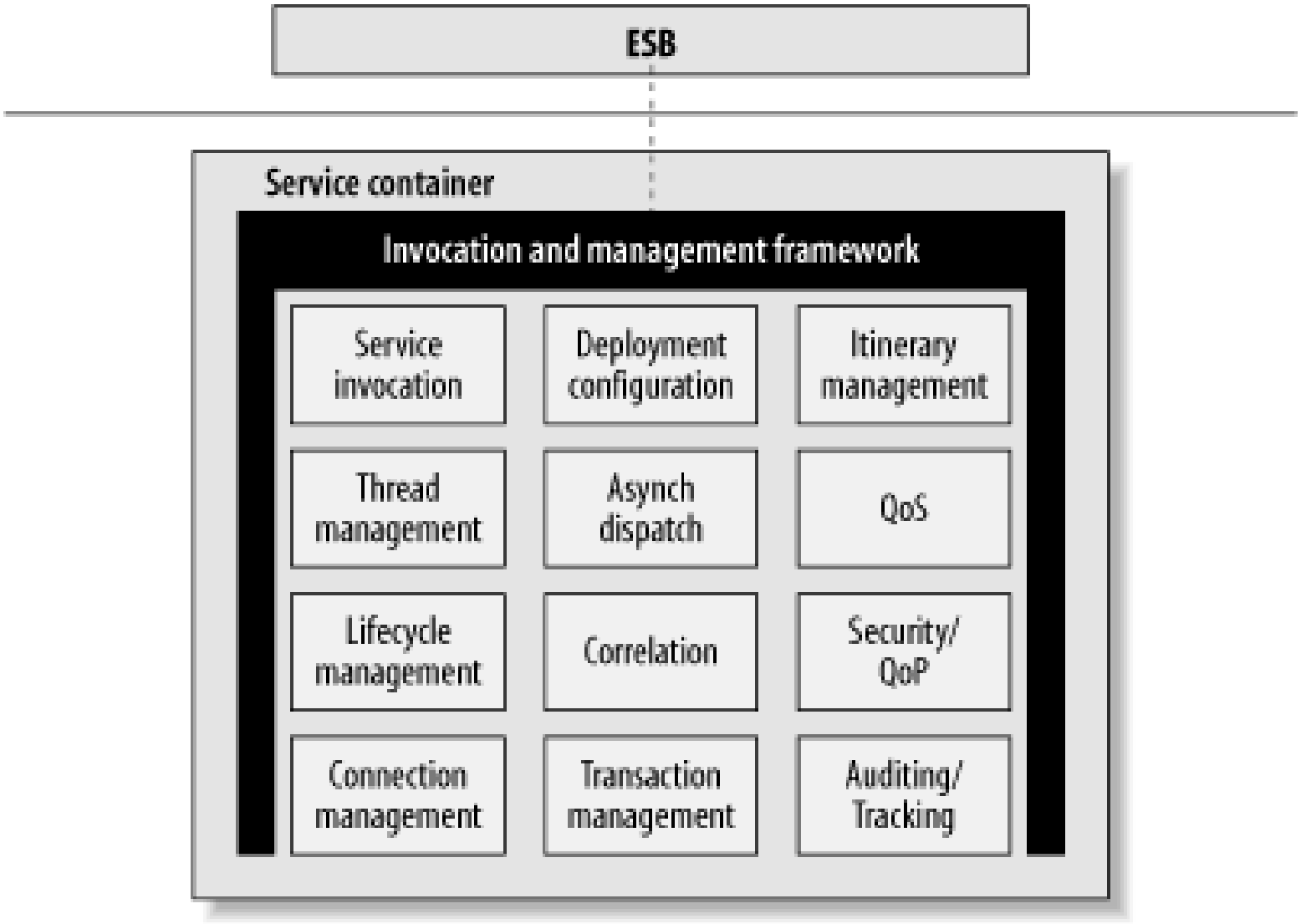


Service Container

- A service container is the physical manifestation of the abstract endpoint.
- A simple and lightweight process, compared with application container and EAI broker.
- Host services and provide support for selective deployment, service invocation and lifecycle management.

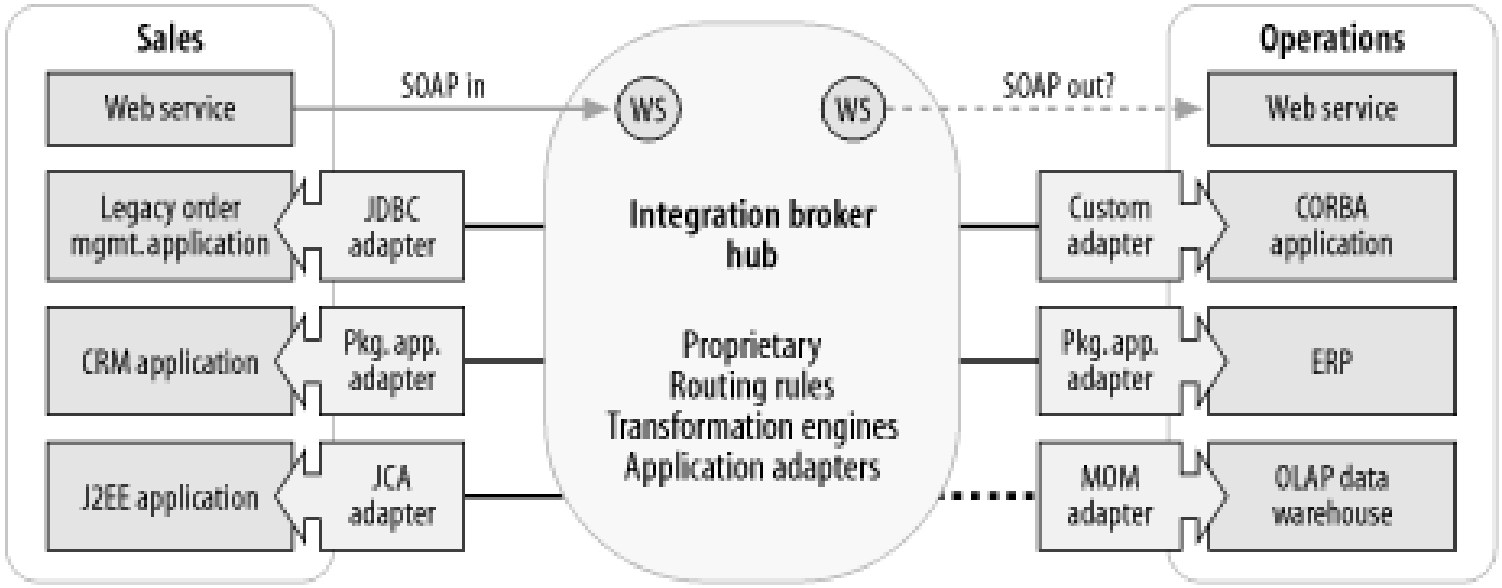


ESB Service Container Facilities



Service Container vs. Integration Broker

- Service Container
  - Highly distributed
  - Standards-based
- Integration Broker
  - Centralized, monolithic
  - Proprietary



Service Container vs. Application Server

- Container managed environment
  - Lifecycle issues
  - Instance management
  - Thread management
  - Timer service
  - Security and transaction services
  - Etc.

- Service Container
  - Host services

- Application Server
  - Host applications

Service Container vs. Application Server

REin
REliable, INtelligent & Scalable Systems

	J2EE Appserver	ESB
Web client proxy	Session Bean	HTTP endpoint
Process definition	Compile class (JPD)	Process / itinerary
Deployment parameters	Bean descriptor	DS config params
Routing rules	Message bean	Routing rules
Web service call	WS proxy class	WS endpoint
Aggregation service	Custom service	Process JOIN
Logging service	Entity bean	XML service
Transformation	XML bean	Transform service
Custom service	EJBean	ESB service

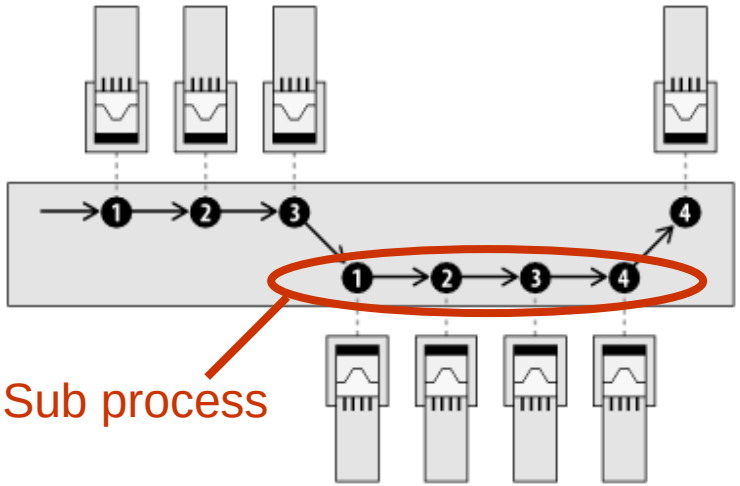
Compiled Class Declarative artifact

Why routing

- The key importance of the ESB approach to SOA is that the service definition is separated from the mechanism for location and invoking services.
- ESB = Router
  - Service routing of requests from service requesters to the relevant service provider based on a routing table
  - Protocol transformation, to allow the decoupling of the protocol that is used between the service requesters and service providers.

Itinerary-Based Routing

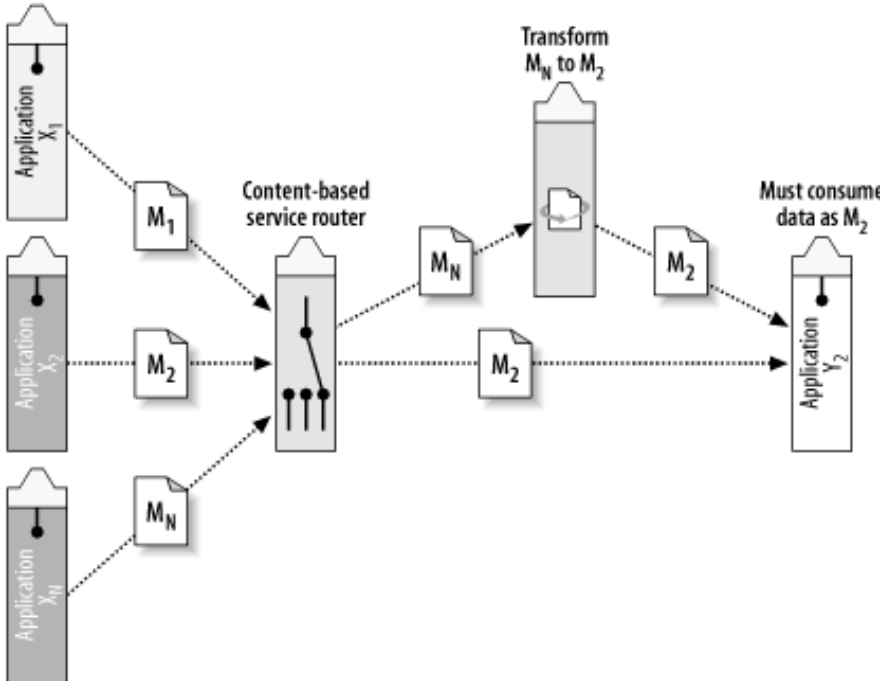
- A message itinerary is analogous to a travel itinerary that you carry when going on a trip.
- the itinerary are stored as XML metadata and carried with the message as it travels across the bus from one service container to the next
- The itinerary represents a set of discrete message routing operations (endpoints)



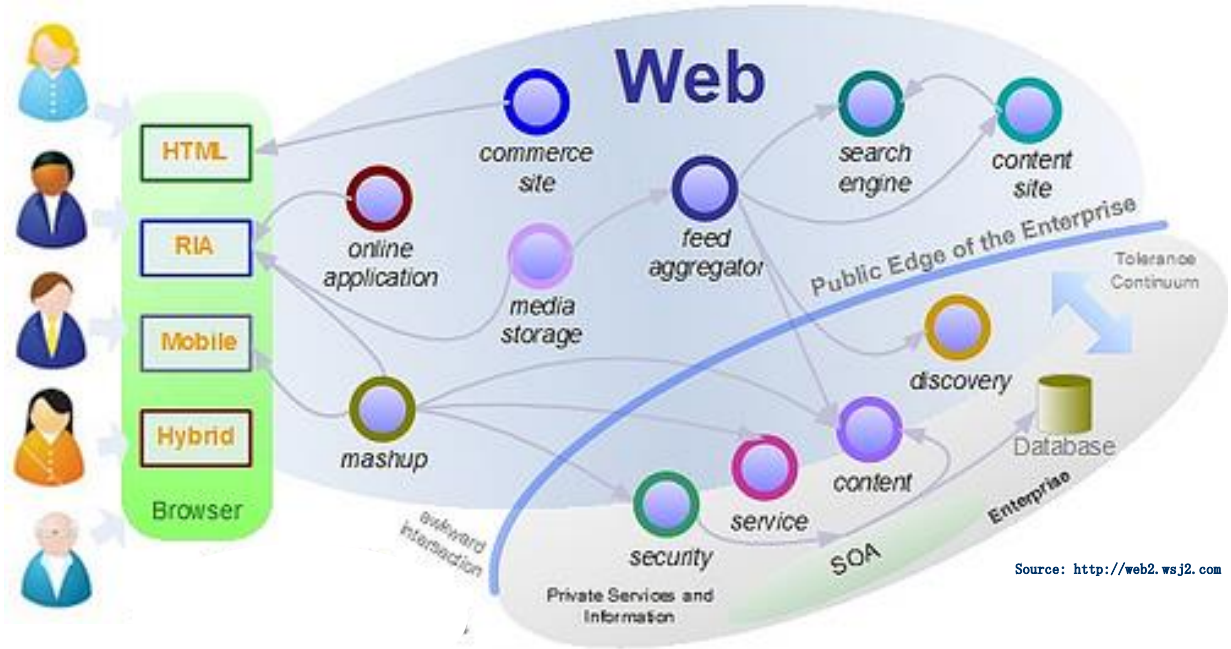
Content-Based Routing

- To control where messages go based on message properties or message content
- Metadata describing the possible branches and rules are evaluated at remote container, not by a centralized rules engine

```
<rule:routingRule>
    <rule:condition>GetProperty("orderCount")
           &gt;= defaultCount</rule:condition>
    <rule:endpoint endpoint_ref="CheckAvailabilityJCAEndpoint"
           type="ENDPOINT"/>
</rule:routingRule>
```



The Rise of the Mashup



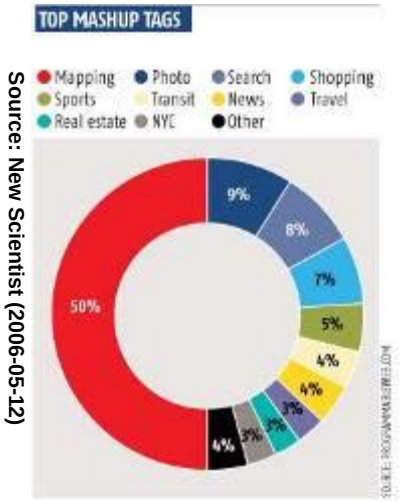
- Mashup = a flexible composition of Services within a rich user interface environment
- In essence, a Mashup is a SOBA interface

An Explosion of Mashups

REin
REliable, INtelligent & Scalable Systems

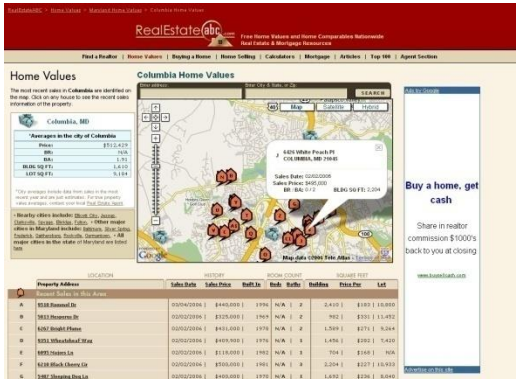
A mashup is a website or web application that uses content from more than one source to create a completely new service.

Source: Wikipedia -- http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

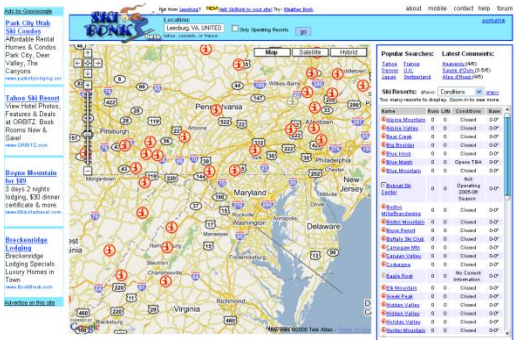


Source: New Scientist (2006-05-12)

Check Real Estate Value



Track Ski Conditions



Track Storms

Mashups are HOT

- An explosion of "mashups" is fueled by
  - (1) ubiquitous Web Service API (esp. Google Maps API)
  - (2) the idea that "everyone can create new applications by reusing the existing parts"
  - (3) the rediscovery of the power of "maps"

You Mashup?

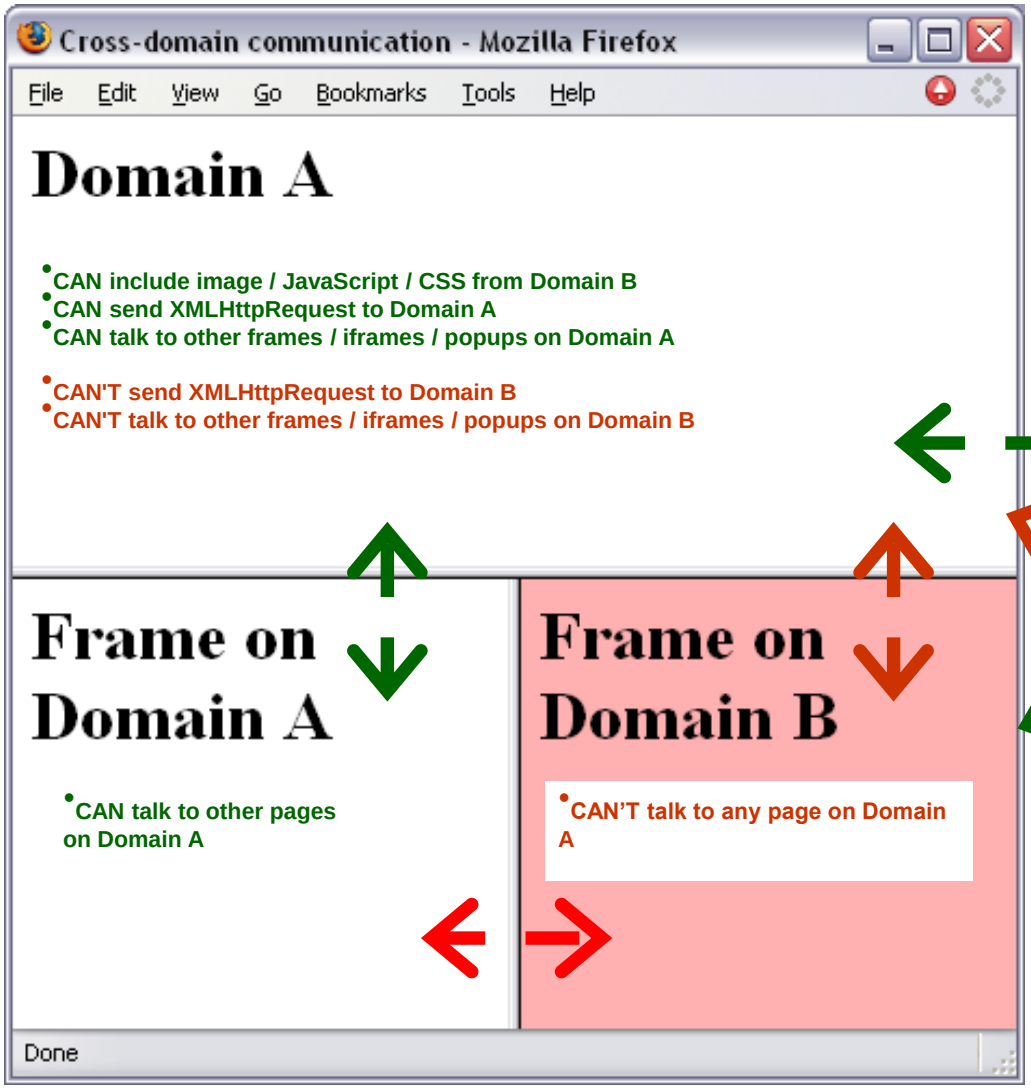
By Cathy Wilcox, the Sydney Morning Herald

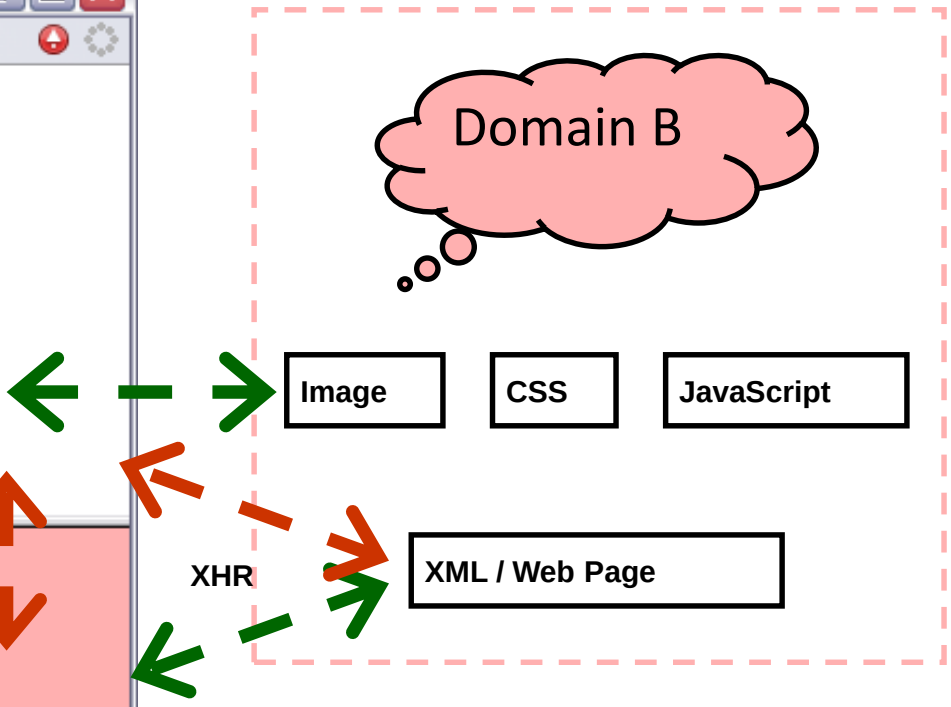
Talking between web components

- Normal situation: all on the same web site
  - Communicate across frames / iframes / popups
  - Talk to server using AJAX (XMLHttpRequest)
- Problem: doesn't work when components live at different domains (mashup situation)
  - *Same-origin policy*
    - Can include JavaScript / images from another domain
    - Can't talk to frame / iframe popup or send an XHR
    - Prevents snooping on secret web pages (e.g. intranet)

Talking between web components

REliable, INtelligent & Scalable Systems

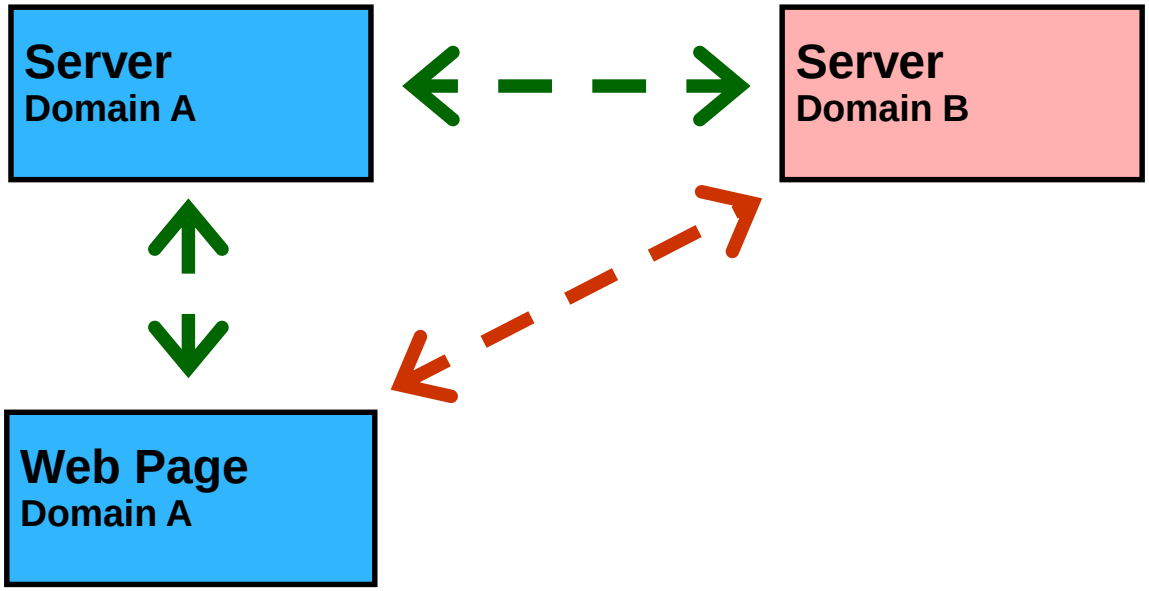




- So, how do mashups communicate?

How do mashups communicate?

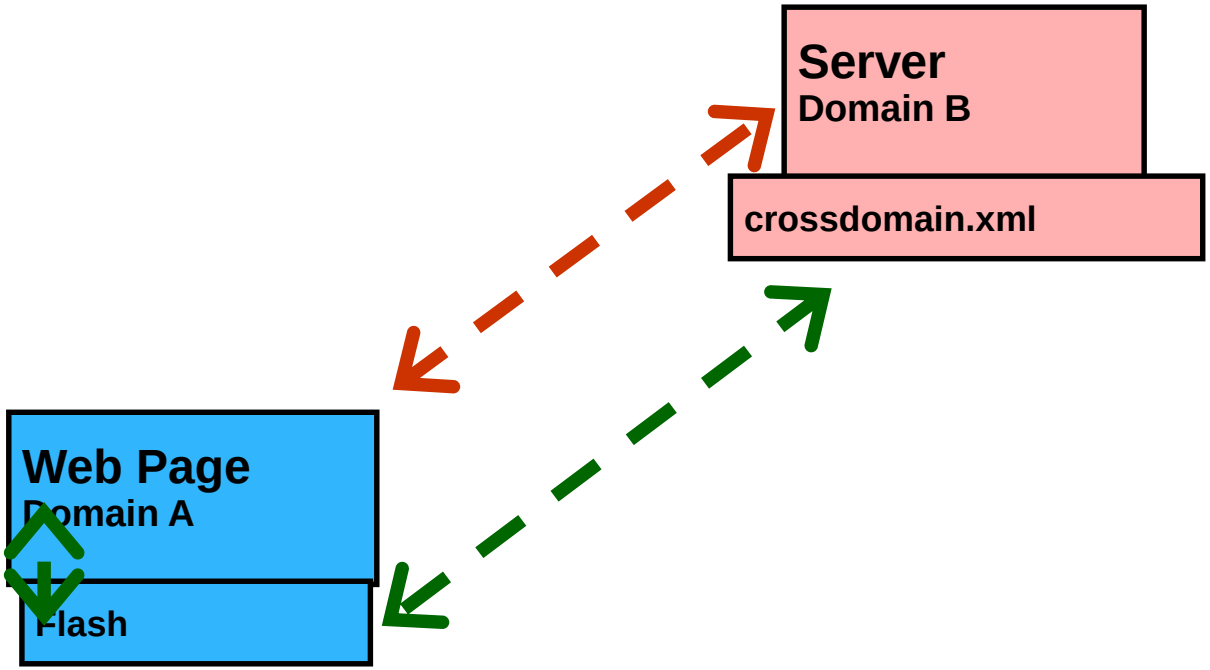
Server-side proxy



Server-side proxy (HousingMaps, Flickr)

- Talk to your server → it talks to the foreign site
- Problem: bottleneck; barrier to mass deployment

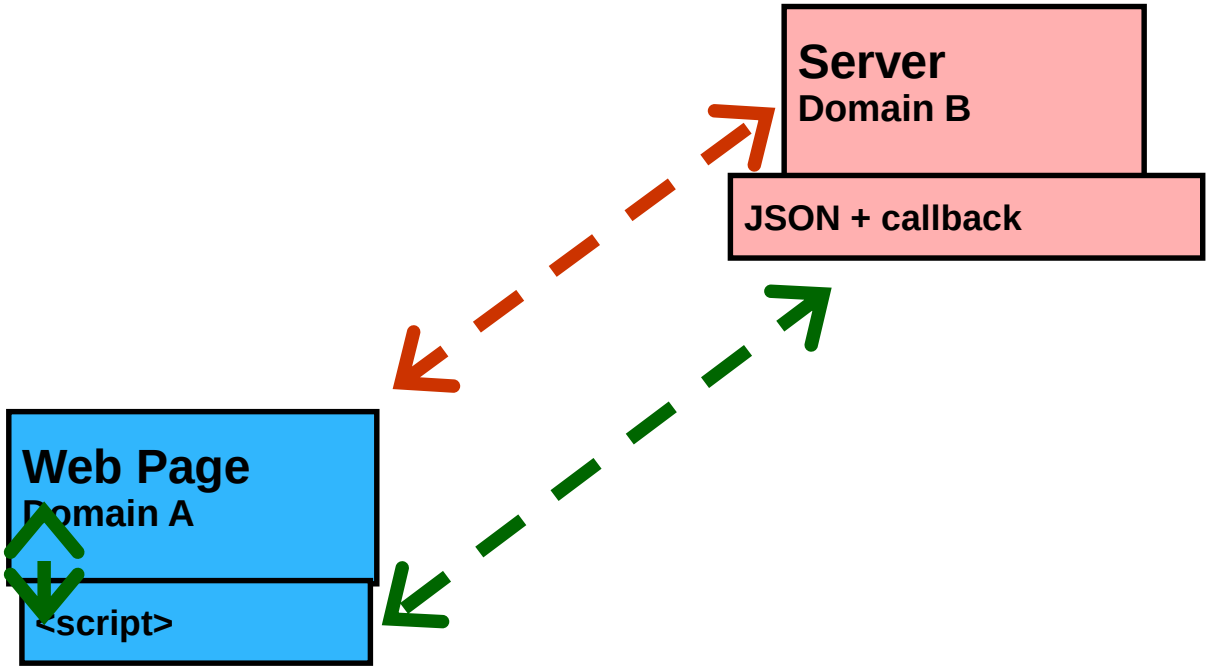
Flash proxy



Use flash for cross-domain communication

- Can use crossdomain.xml to allow foreign sites
- Yahoo maps uses this to do geo-coding

JSON-P



JSON-P (Script injection + callback)

- Dynamically load JavaScript file with dataClever trick: specify callback function on URL
- Works well in practice, but some drawbacks

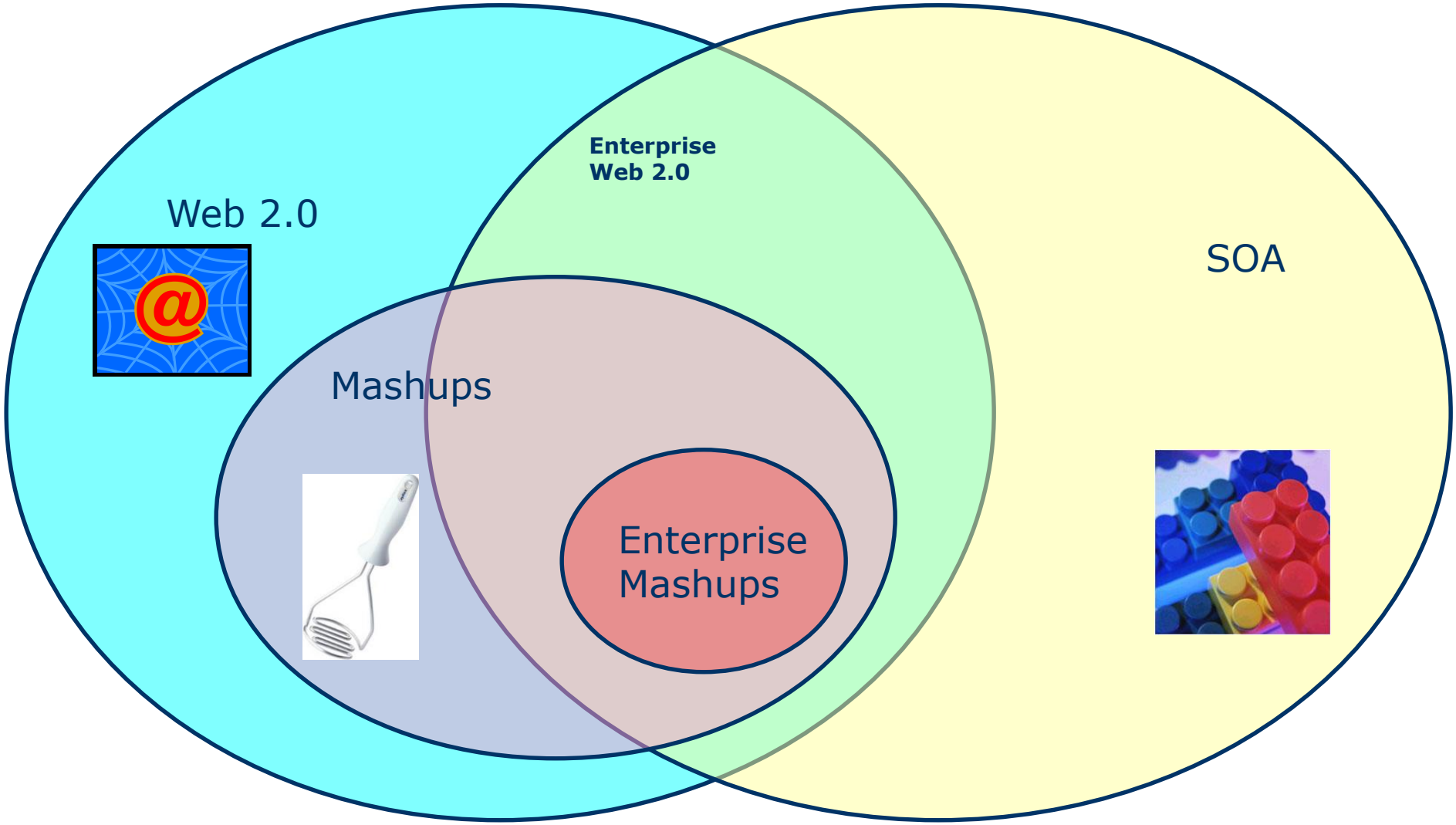
Without Governance, Mashups are *Dangerous*

REin
REliable, INtelligent & Scalable Systems

- Mashups enable *unpredictable* SOBAs

- Risks:
  - Confidentiality breaches
  - Unauthorized capabilities
  - Fraud

Web 2.0 vs. SOA



References

1. IBM Redbook: *Patterns Service Oriented Architecture and Web Services*
2. Pages 13-60 are from the courseware of Prof. Xiaoying Bai, Department of Computer Science and Technology, Tsinghua University

Thank You!