IBM Software Group

2004 WDI / WBIC Customer Conference

Global Business Transformation

WebSphere. software

Service Oriented Architecture
Raghu Thiagarajan

Raghu Thiagarajan
Product Portfolio Strategist, Business Integration
WebSphere

e business software

© 2004 IBM Corporation

Business evolution

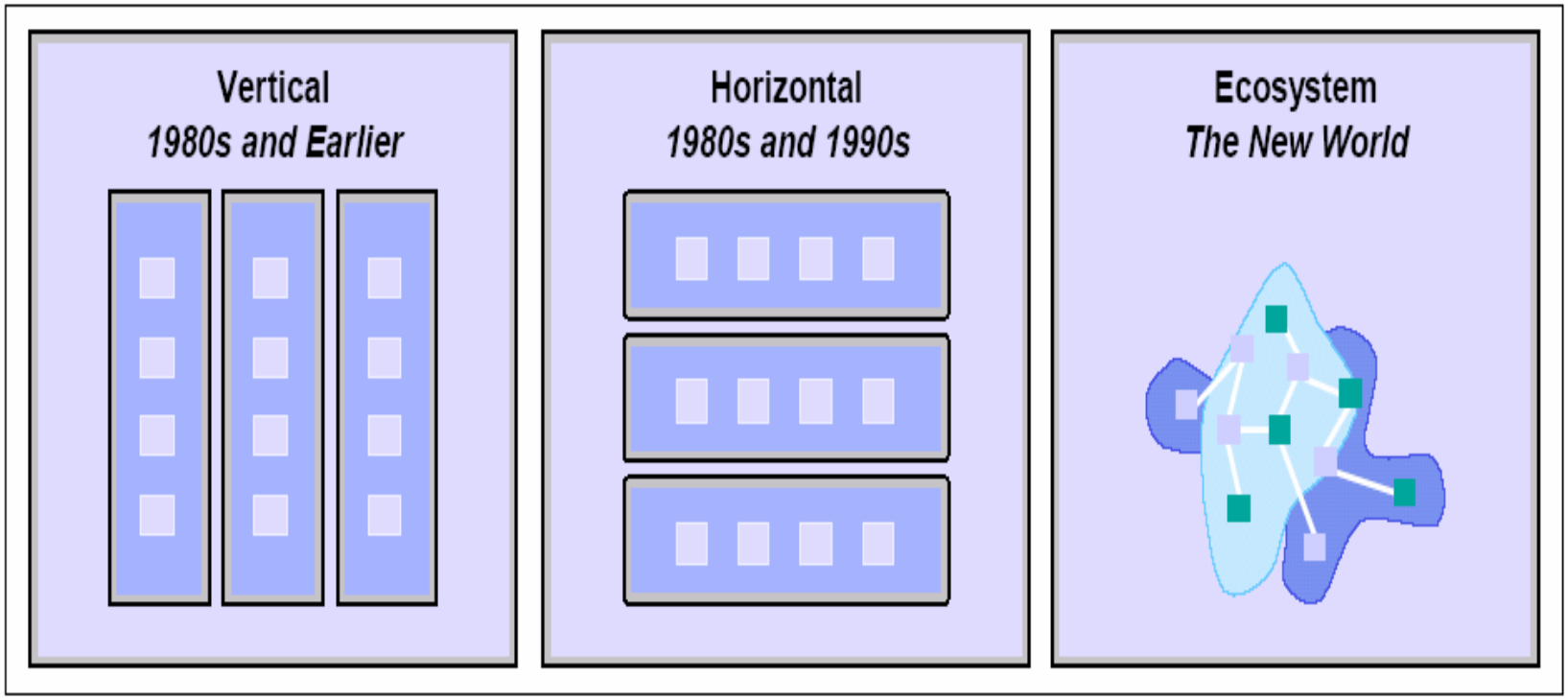


Figure 2-1 The evolution of business

© 2004 IBM Corporation

IT evolution

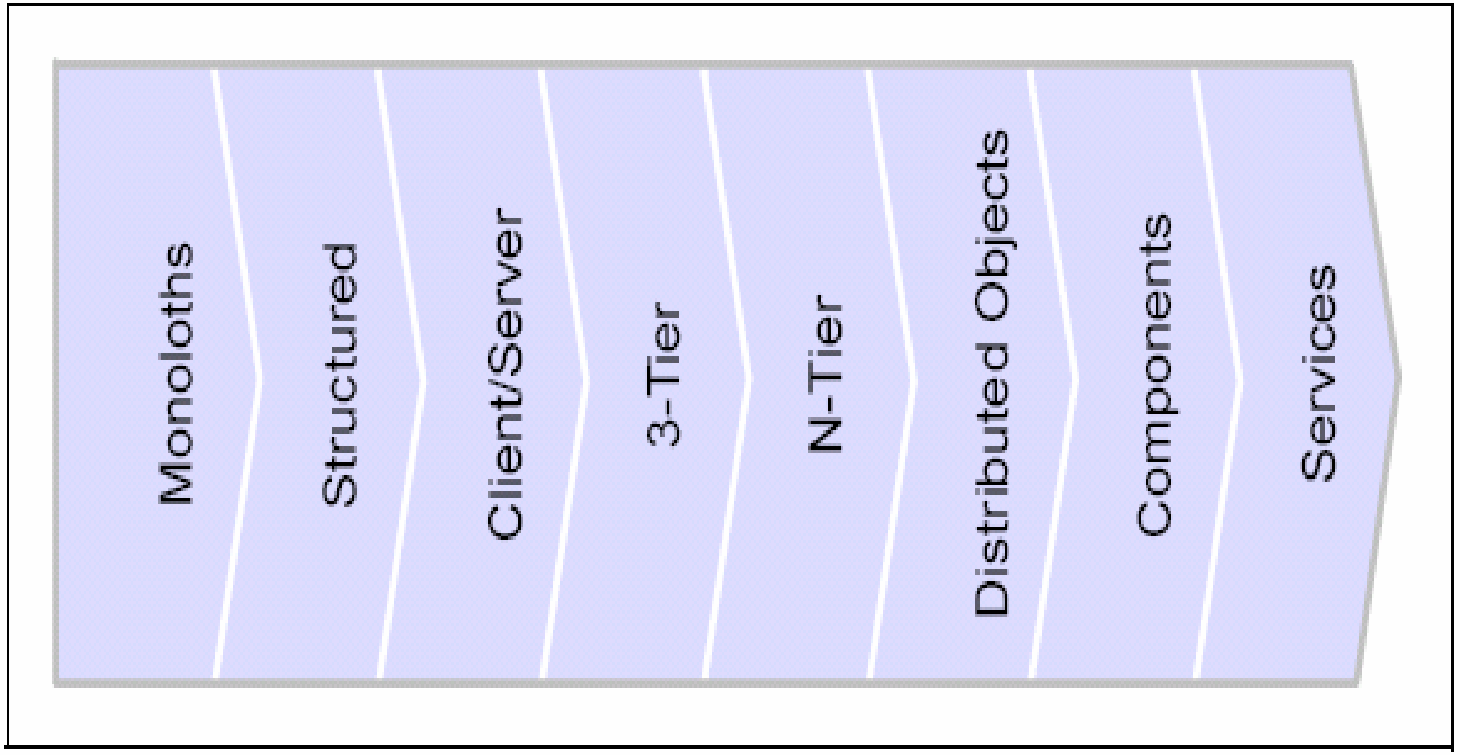


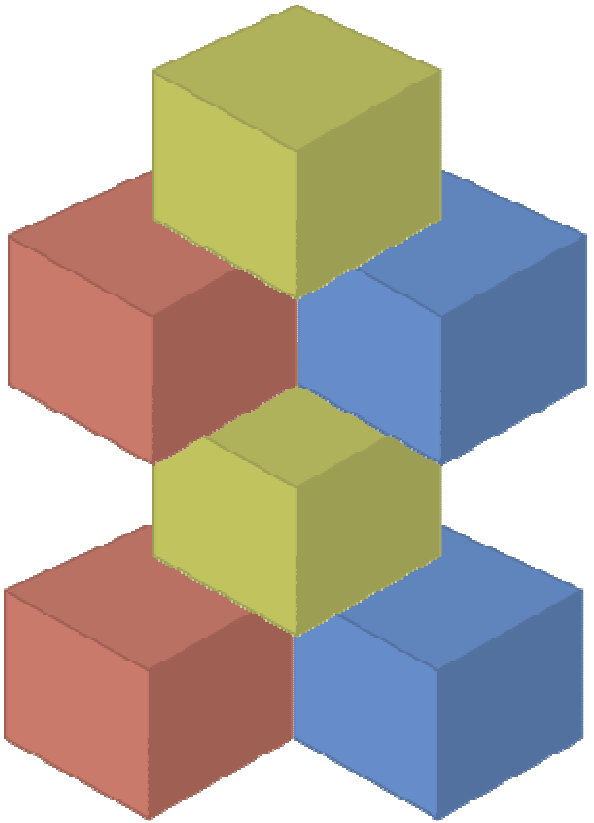
Figure 2-2 The evolution of architecture

© 2004 IBM Corporation

What is Service Oriented Architecture?

- An **approach** for **building** distributed systems that deliver application functionality as **services** to either end-user applications or other services

- It defines :
  - An architecture that leverages open standards to represent software assets as services.
  - Provides a standard way of representing and interacting with software assets
  - Individual software assets become building blocks that can be reused in developing other applications
  - Shifts focus to application assembly rather than implementation details
  - Used externally to integrate with applications outside of the enterprise

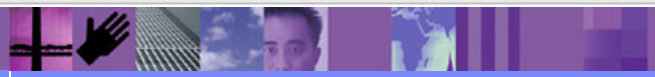


Protocol independence.

© 2004 IBM Corporation

Why is SOA Important?

Business Benefits	▪ Business flexibility provided by increased granularity of processes enabled through services ▪ Ability to quickly create business processes and composite applications to respond to changes in the marketplace ▪ Improved customer service using services without having to worry about the underlying IT infrastructure
IT Benefits	▪ Becoming a more responsive IT organization with a secure and managed integration environment ▪ Decrease development and deployment cycle times through the use of pre-built, reusable services building blocks. ▪ Reducing complexity and maintenance costs with common services ▪ Enhancing existing IT systems rather than replacing them



© 2004 IBM Corporation

How did we get here:

- Process (programs) exchange Information
- Processes need to expose interfaces for Data exchange.
- Processes need to expose entry points.

Key interaction patterns
- Synchronous or asynchronous
- Tightly or loosely coupled. (Loose or tight typing)
- Conversational or stateless.

- Remote Procedure Call (RPC)
  - Stock_price = get_stock_price(stock_symbol)
  - Or
  - historical_stock_price = get_historical_stock_price(Date, stock_symbol)
  - Cons:
  - Hard coded types -> Date has to be understood by type system/language of both requester and provider. Not easily extensible. Parameterization is inflexible.
  - Corba provides for extensible object typing, but changes need to “recompiled” into both client and server systems.
- Message based
  Loosely typed and disconnected.
  Asynchronous.

© 2004 IBM Corporation

SOA terminology

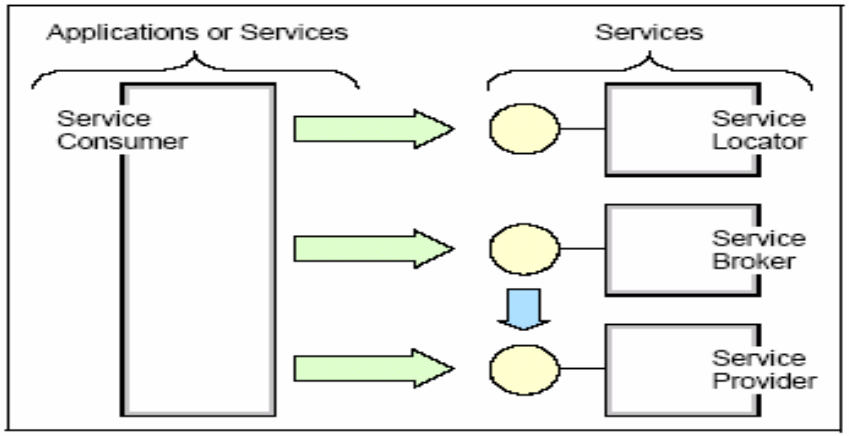


Figure 2-3 Service-oriented terminology

- **Services**: Logical entities, the contracts defined by one or more published interfaces.
- **Service provider**: The software entity that implements a service specification.
- **Service consumer** (or requestor): The software entity that calls a service provider. Traditionally, this is termed a “client”. A service consumer can be an end-user application or another service.
- **Service locator**: A specific kind of service provider that acts as a registry and allows for the lookup of service provider interfaces and service locations.
- **Service broker:** A specific kind of service provider that can pass on service requests to one or more additional service providers.

© 2004 IBM Corporation

SOA invocation style

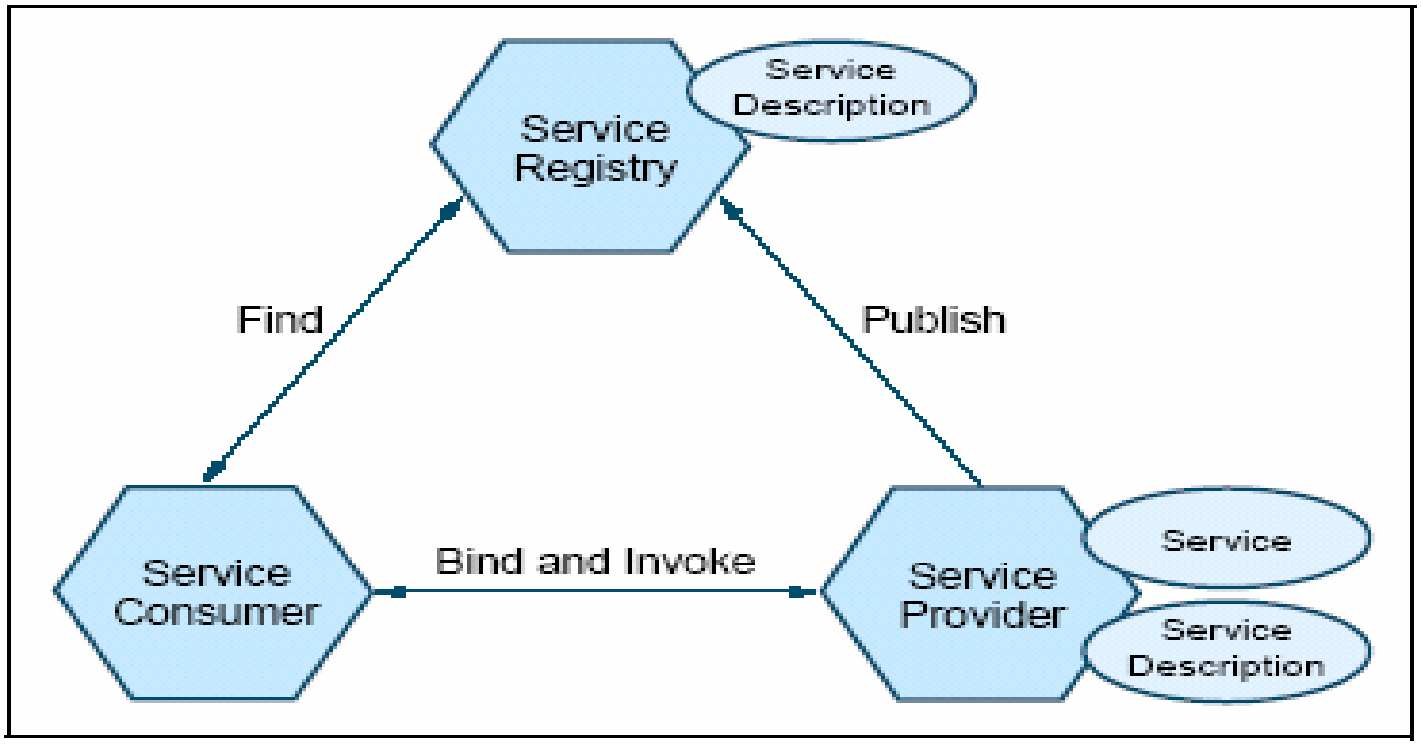


Figure 2-7 Collaborations in a service-oriented architecture

© 2004 IBM Corporation

Before SOA: Component based architectures presume homogeneity

- Tight coupling between requester and provider
  1. Parameters order and structure is hard-coded.
  2. Synchronous invocation style constrains interaction.
  3. Developers exposed to multiple API styles for differing service provider modules.
- Suitable for tightly knit project teams building on homogeneous application frameworks (.NET or J2EE)

© 2004 IBM Corporation

Before SOA: Distributed object architectures presume homogeneity

- Tight coupling between requester and provider
  1. Parameters order and structure is hard-coded.
  2. Synchronous invocation style constrains interaction.
  3. Developers exposed to multiple API styles for differing service provider modules.
  4. Microsoft did not sign up for CORBA.

- Suitable for intra-departmental projects
  - applications on homogeneous application frameworks

© 2004 IBM Corporation

IT architectural relationship: Objects, Components, Services

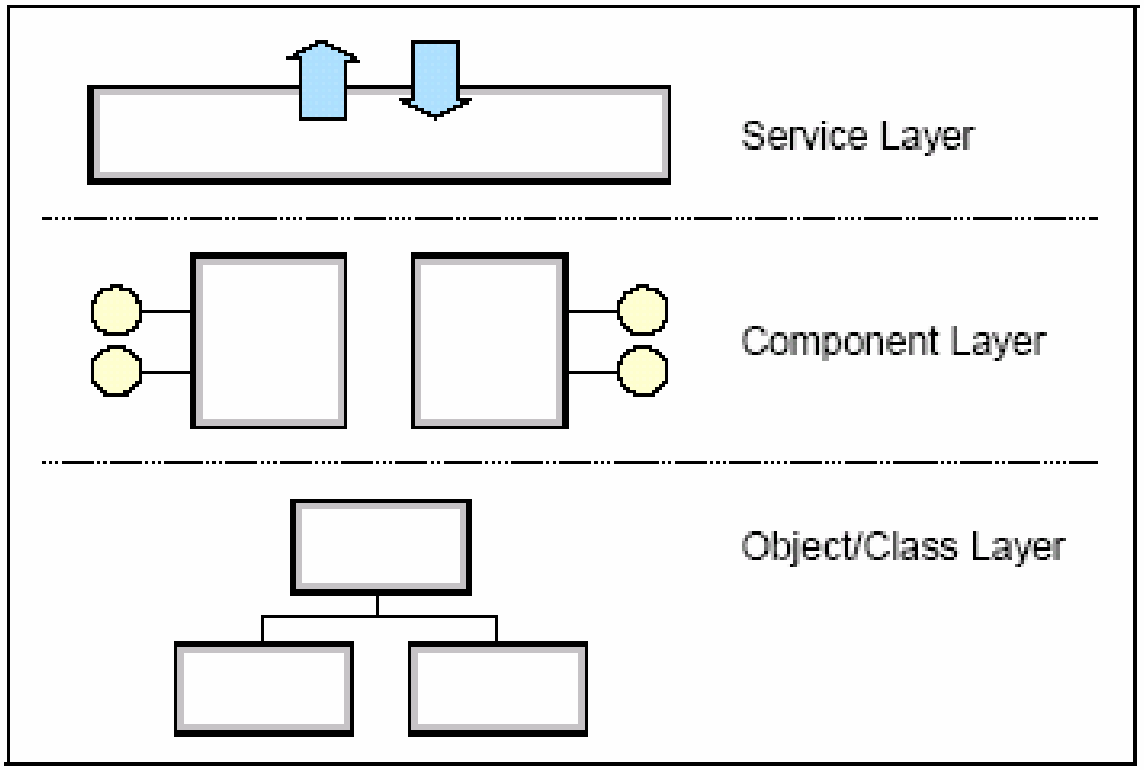
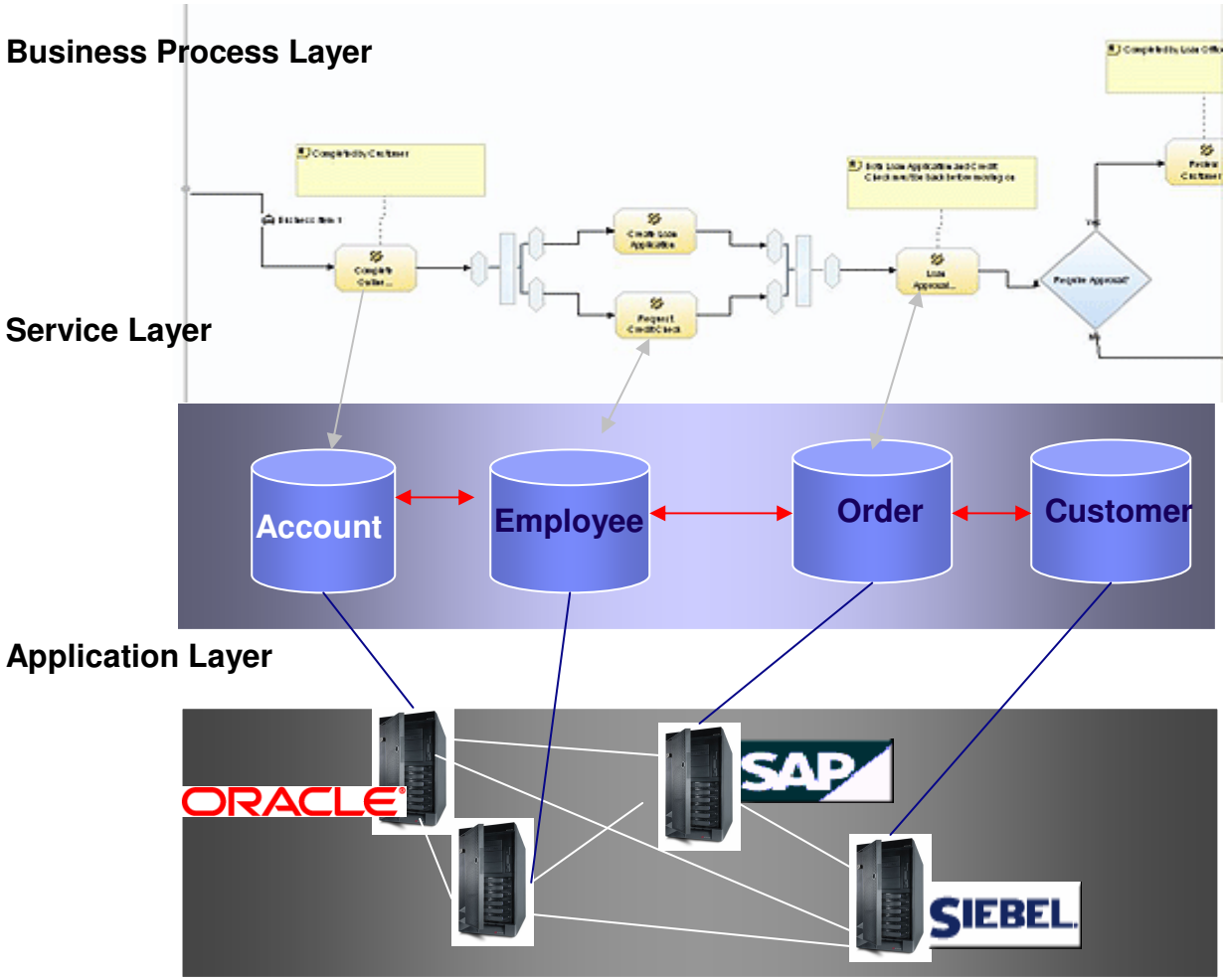


Figure 2-5 Application implementation layers: Services, components, objects

© 2004 IBM Corporation

Architectural Layers



Context of SOA.

Integration is performed at **various layers** in IT architecture.

1. **Application Layer** contains applications and components deployed to the technology below. E.g. it might consider how **SAP ERP** is **connected** to **Siebel CRM** for example.

- Integration challenge at these layers – hard wired to **specific instances** such as J2EE to .NET, or SAP to Siebel. If these change then integration must change too.

2. **Service Layer** abstracts resources away from lower layers. Focus shifts to **connecting Customers** with **Orders** – regardless of the underlying implementation.

3. **Business Process Layer** solutions are constructed that use the **Services**, not the underlying applications and technology.

© 2004 IBM Corporation

SOA stack

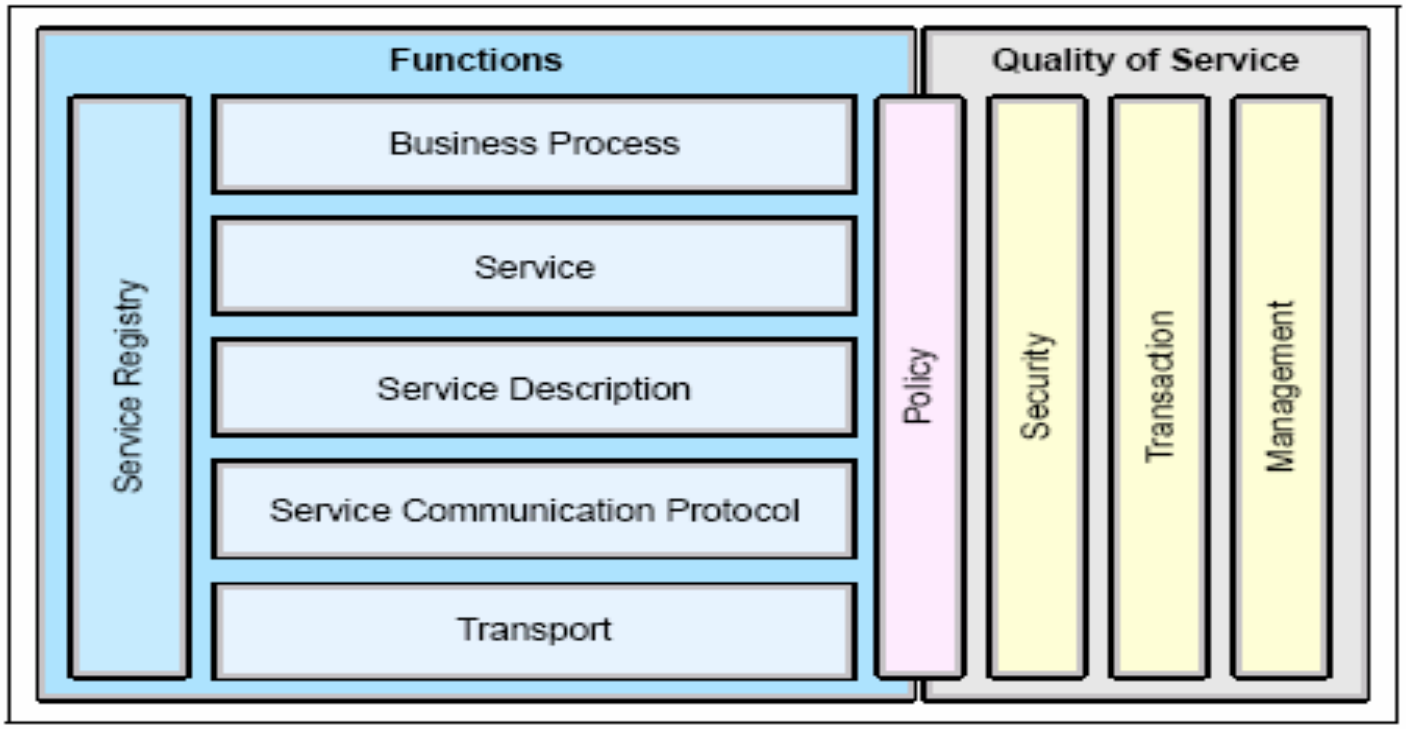


Figure 2-6 Elements of a service-oriented architecture

© 2004 IBM Corporation

What is a Web service?

- **An Open Standard for accessing component based applications**
- **Core Web services standards include XML, WSDL, SOAP and UDDI.**
- **Advanced Web services provide support for Security, Transactions, Reliability, Business Processes and Management..**

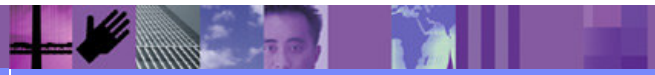
© 2004 IBM Corporation

Web Services: Review

- "A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols."

- Web Services are based on

1. eXtensible Markup Language (XML)
2. Simple Object Access Protocol (SOAP)
3. Universal Description, Discovery and Integration (UDDI)
4. Web Services Description Language (WSDL)



© 2004 IBM Corporation

XML

HTML was designed for human / computer interaction

XML was designed for computer / computer interaction

© 2004 IBM Corporation

Web Services: User interface for computers

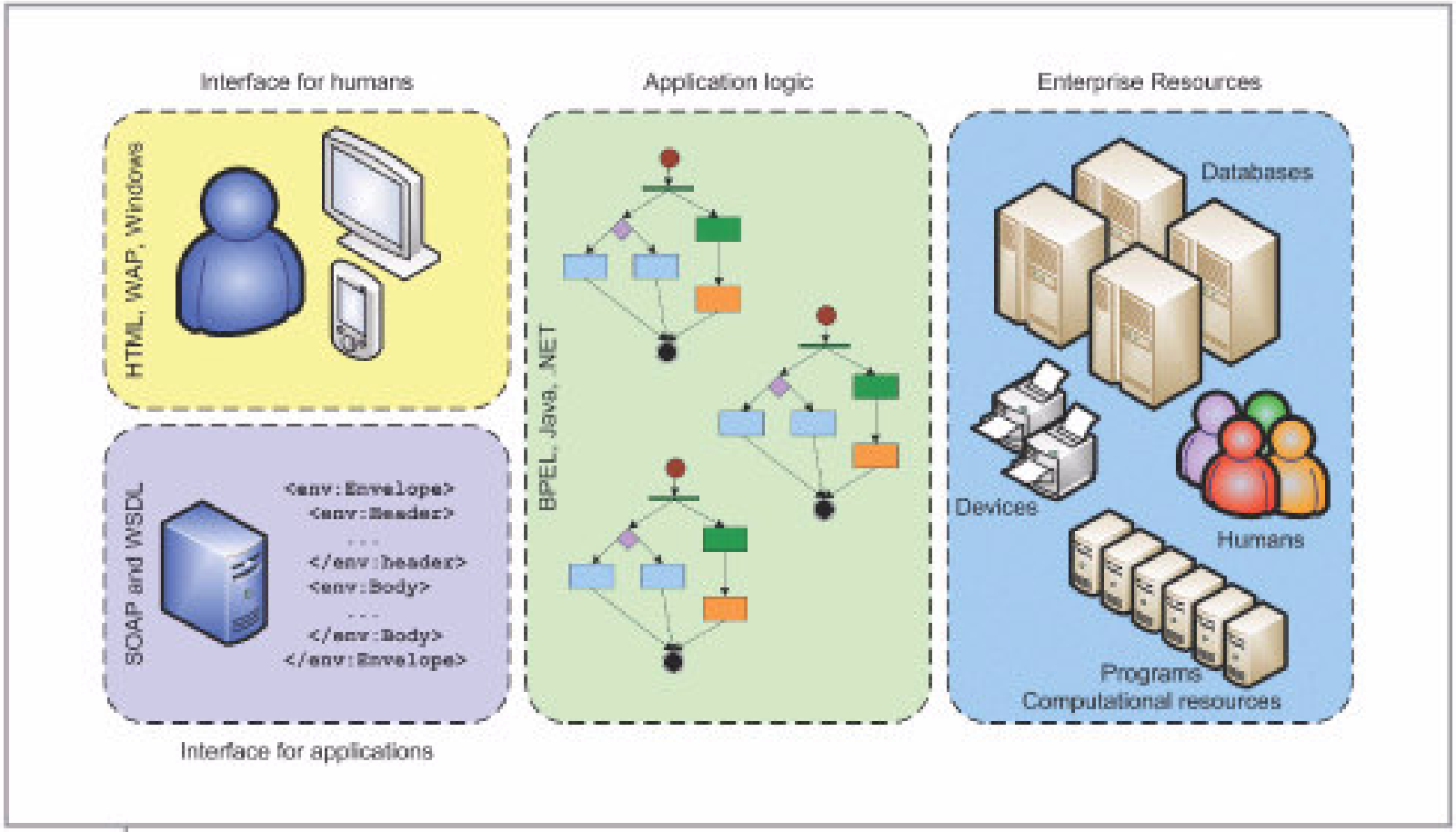


FIGURE 1 | **Managing the problems**

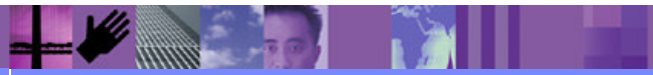
© 2004 IBM Corporation

XML

HTML was designed with humans in mind.

The tags don't tell what the information is.

```
<p><b>Mr. Marshall Barnes</b>
<br>
13023 Partridge Bend
<br>
Austin, TX 78729</P>
```



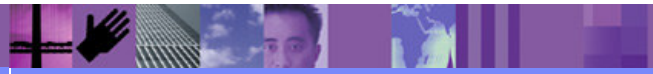
© 2004 IBM Corporation

XML

XML lets you create your own tags,

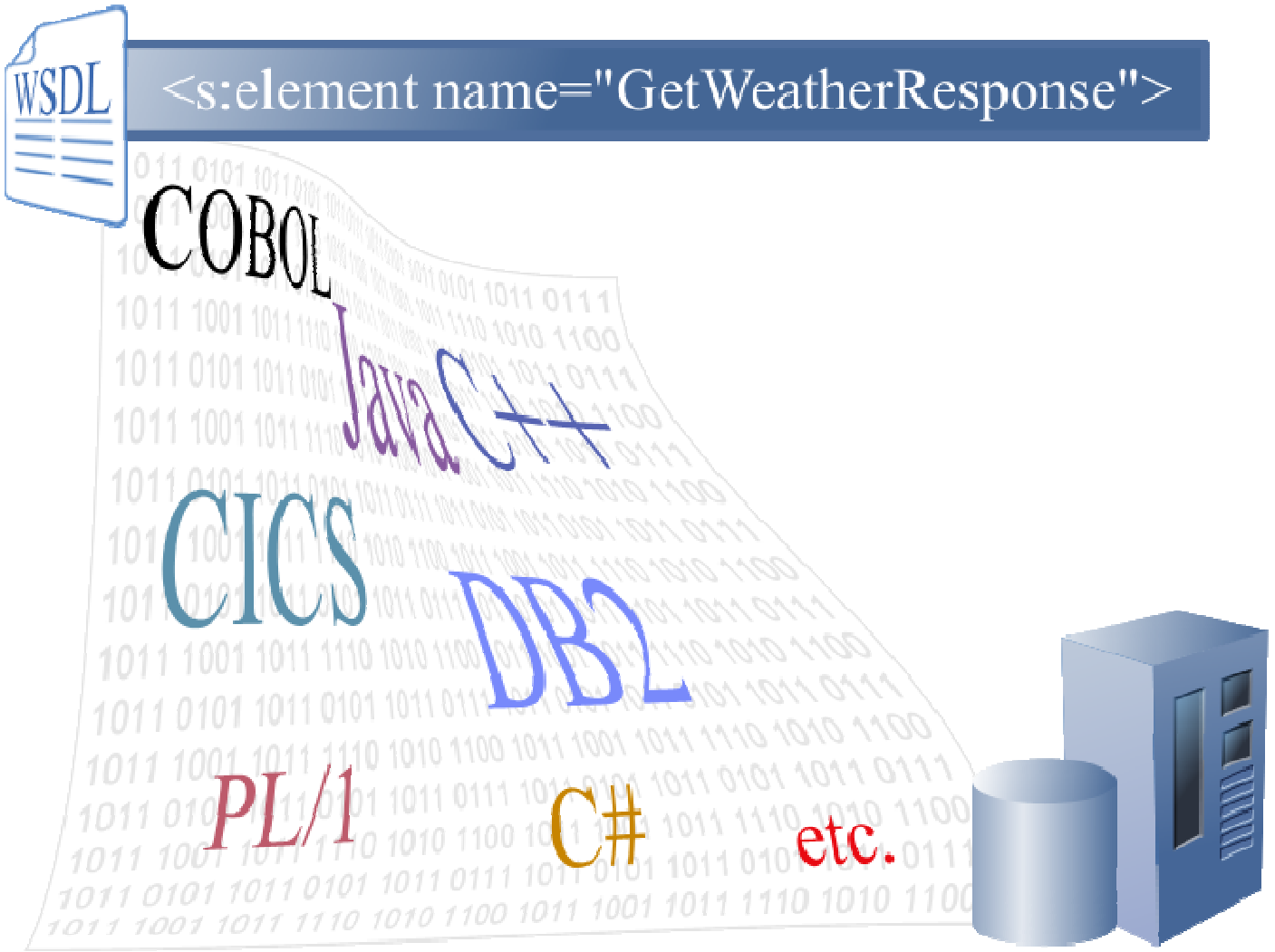
and was designed with the Web in mind.

```
<address>
   <name>
      <title>Mr.</title>
      <first-name>
         Marshall
      </first-name>
      <last-name>
         Barnes
      </last-name>
   </name>
```



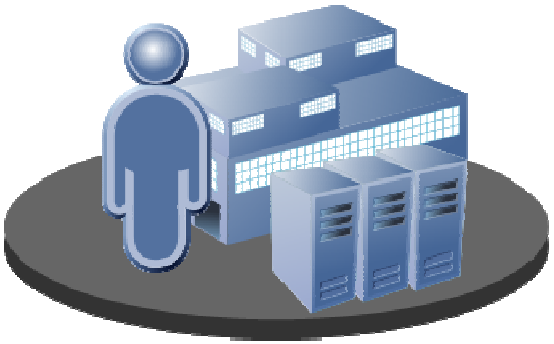
© 2004 IBM Corporation

WSDL describes the Web service



© 2004 IBM Corporation

SOAP specification defines an “envelope”



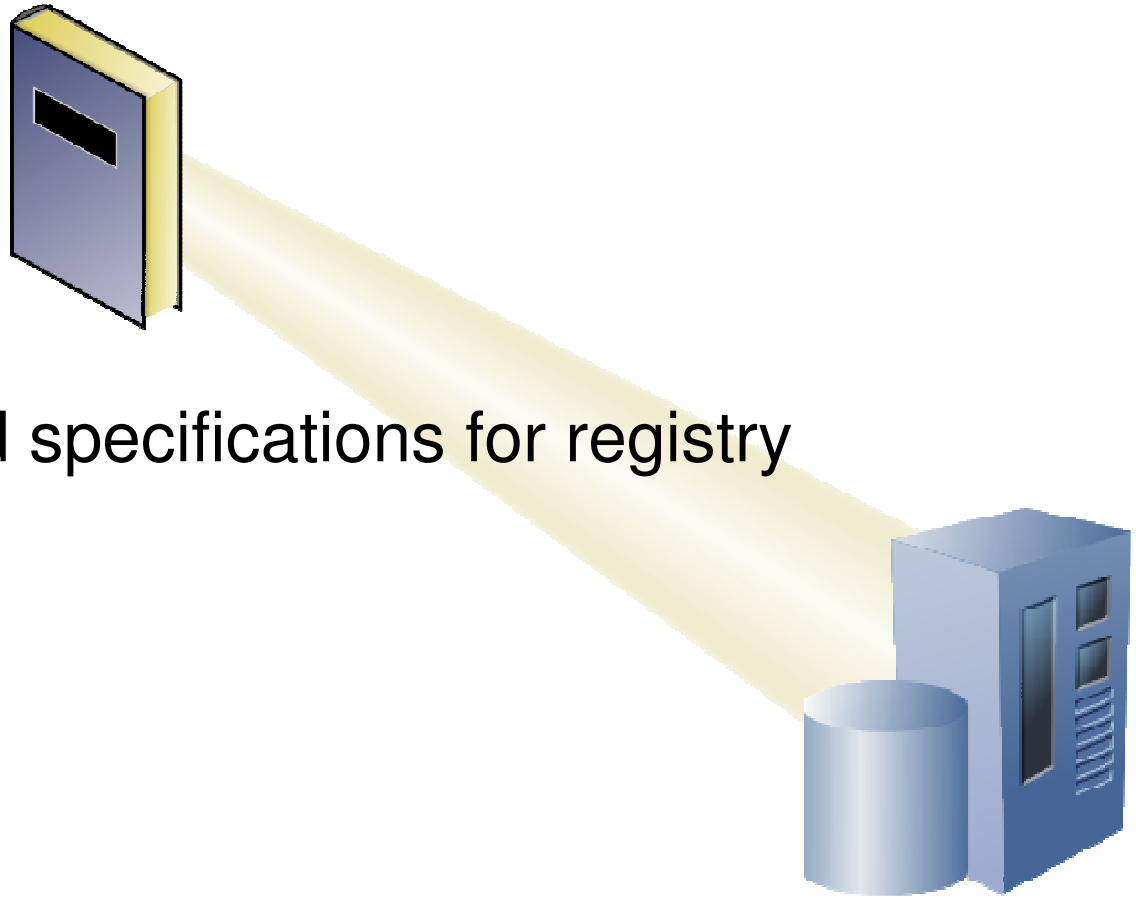
- Application-specific message vocabulary

- SOAP envelope vocabulary

© 2004 IBM Corporation

UDDI Servers can be used to locate available Web services

Universal
Description,
Discovery, and
Integration

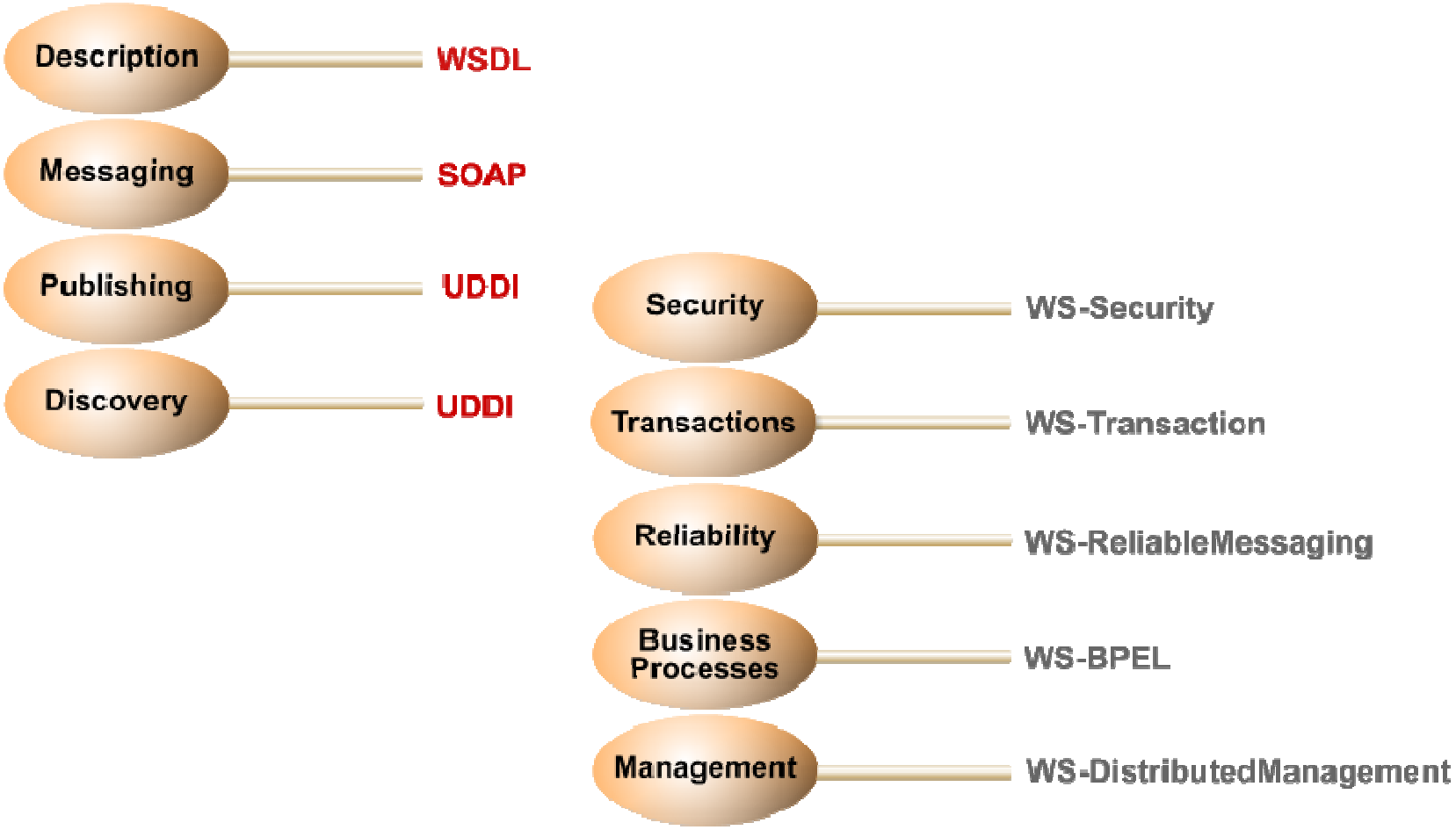


Standards-based specifications for registry and discovery

UDDI Server

© 2004 IBM Corporation

Web Services Specifications



© 2004 IBM Corporation

Industry standards

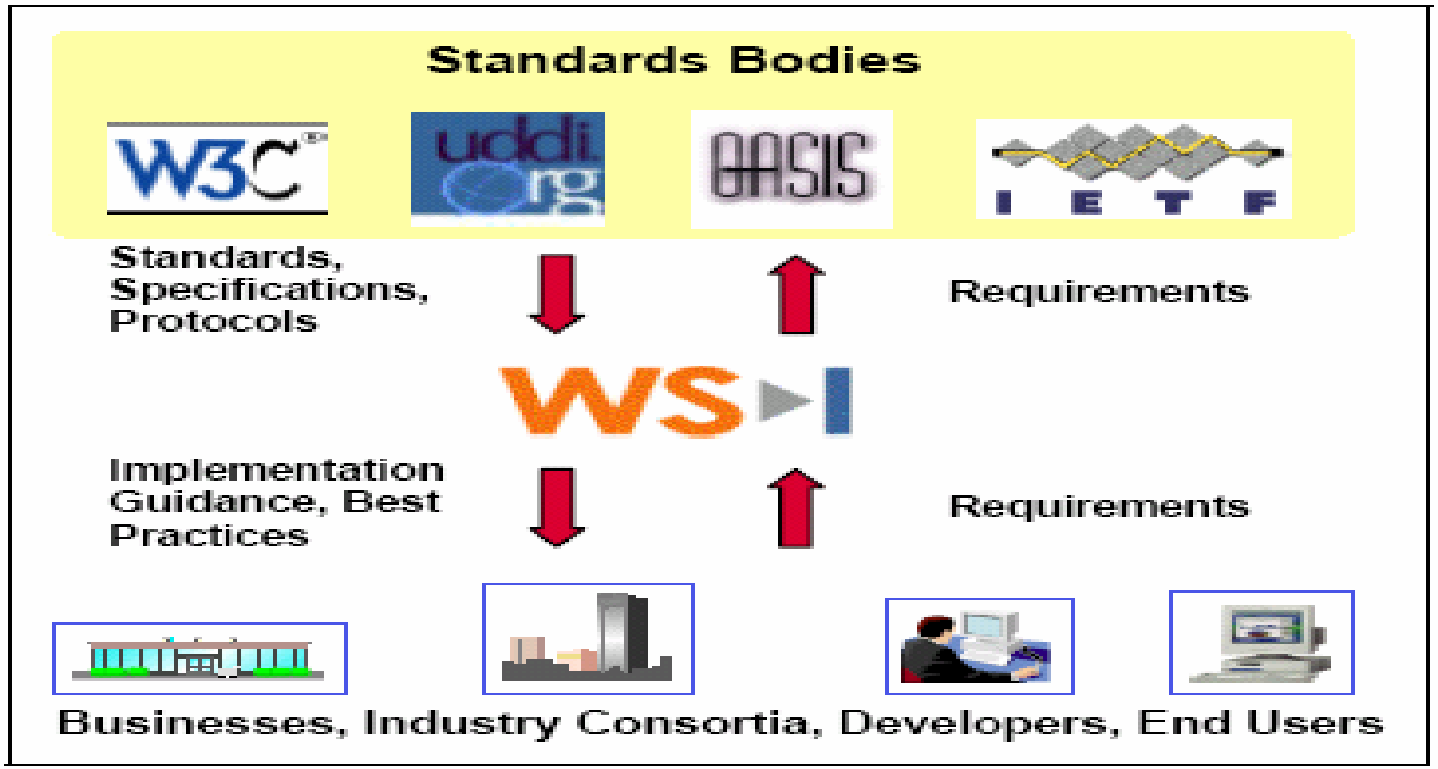
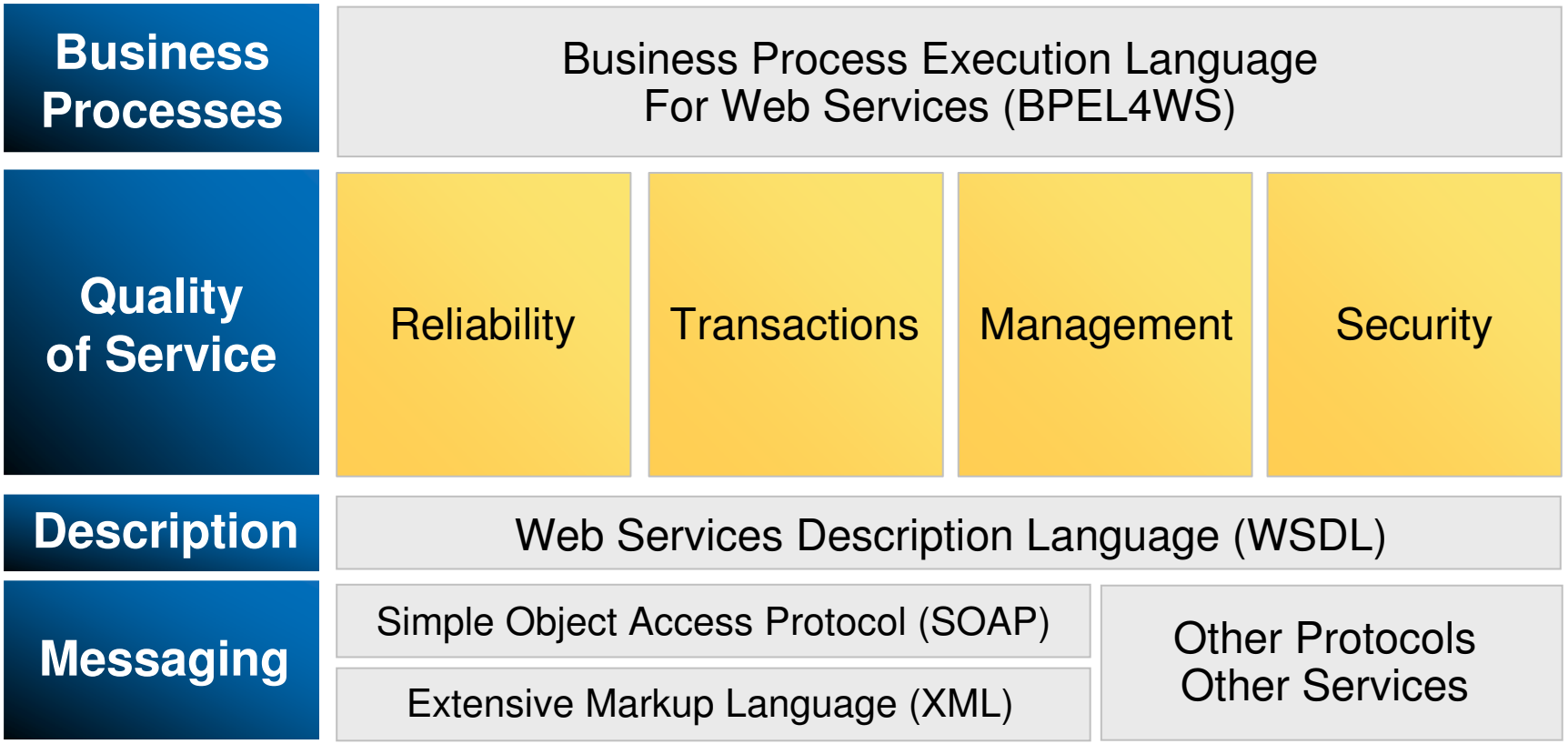


Figure 2-11 WS-I, standards and industry

© 2004 IBM Corporation

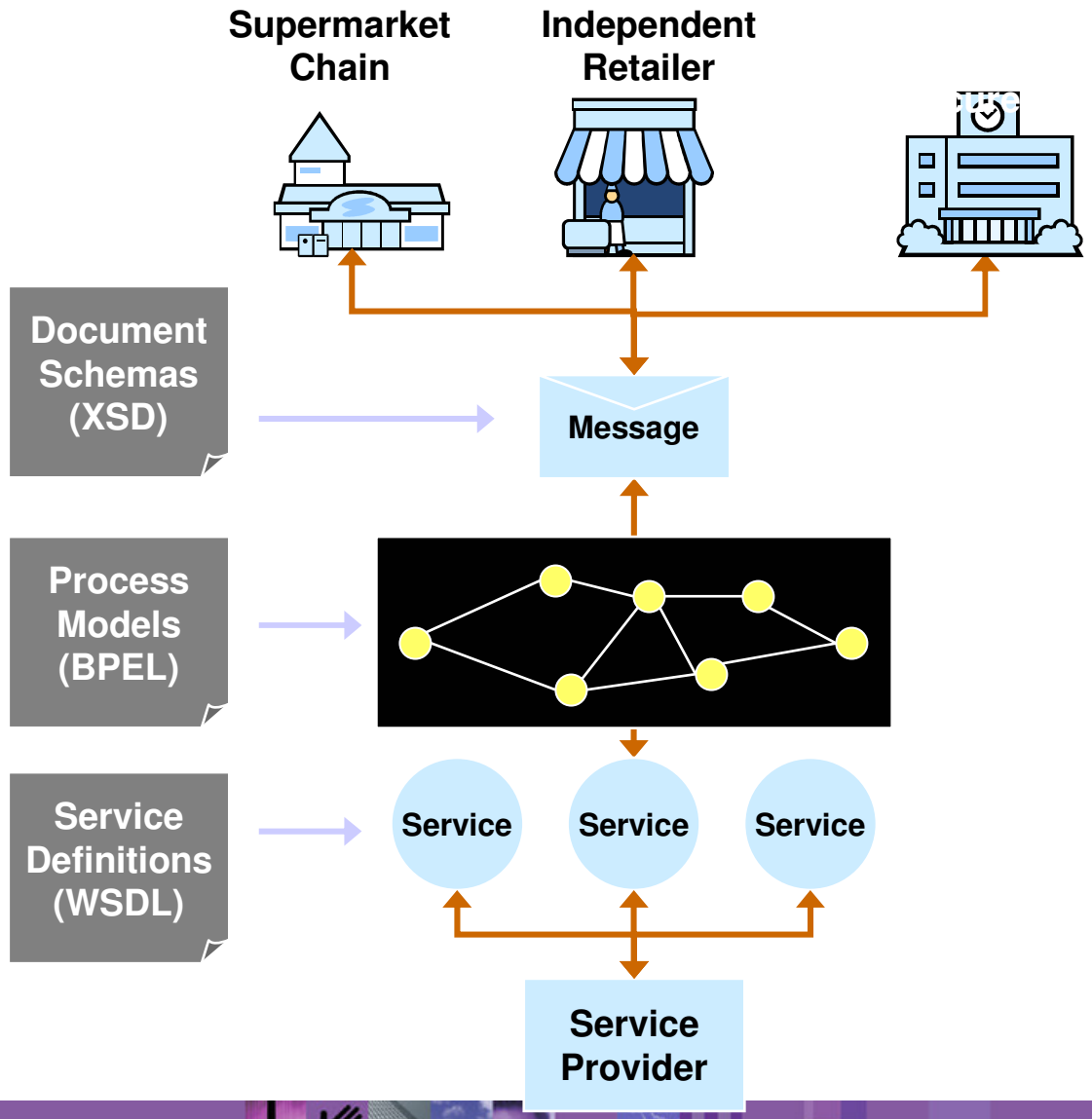
Open Standards Progress

Web services collaboration makes "secure, reliable, managed and transacted" a reality



© 2004 IBM Corporation

EDI and B2B



- Though Web Services will be delivered onto an increasing range of end-customer devices, the **external application** of Web Service today is **most likely** in an **EDI** or **B2B** context.
- Today **proprietary EDI** and **B2B gateways** are widely used by large organizations but the **high cost** of implementation has **prohibited** their many **smaller suppliers** and partners from utilizing this approach.
- The **EDI community** and various B2B groups are already working on evolving their existing standards to support Web Services. The use of Web Services will look little different to the other scenarios covered here. However, they will overlay **standards** that facilitate B2B activity. These will provide standards for the
  - **Document Schemas**
  - **Service Definitions**
  - **Process Models**
- Key **advantages** of Web Service B2B include
  - **Lower technology costs**, enabling all participants regardless of size
  - **Open standards** rather than proprietary gateways and networks

© 2004 IBM Corporation

IBM view: The Enterprise Service Bus

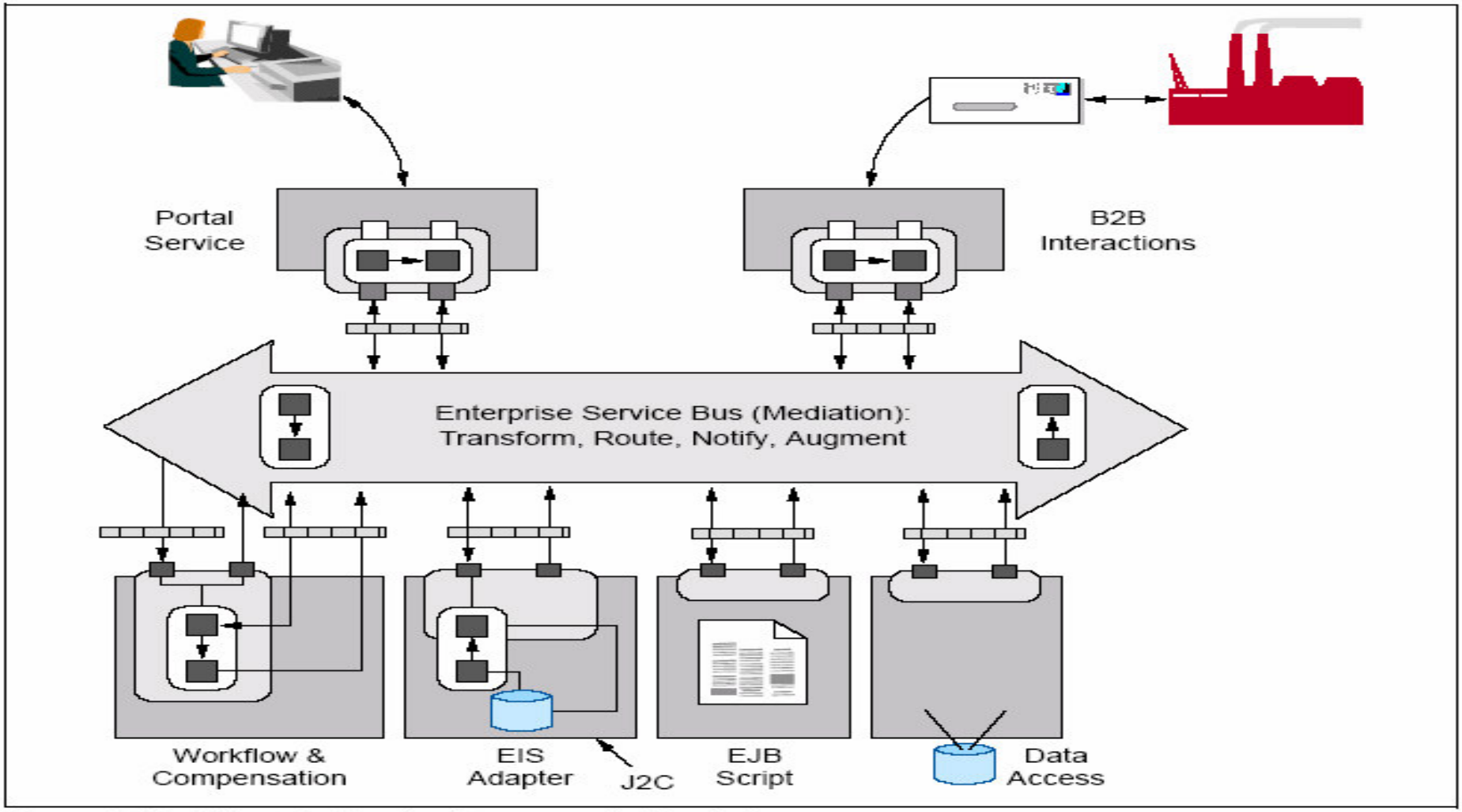
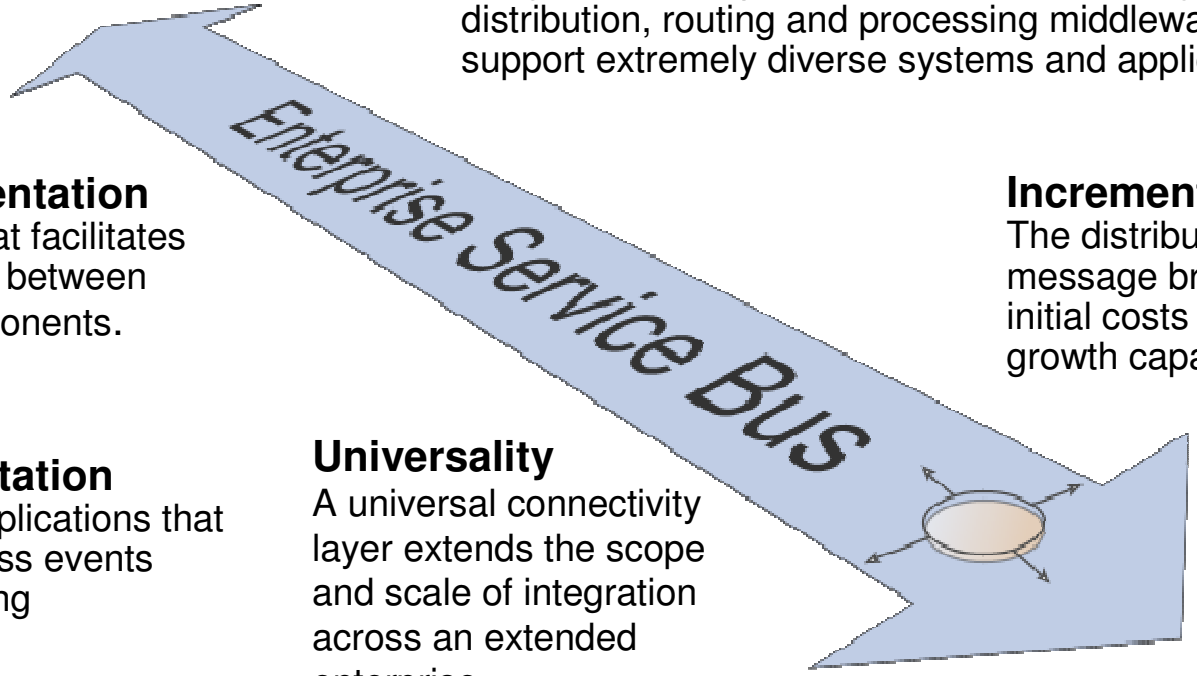


Figure 2-13 Enterprise Service Bus conceptual model

© 2004 IBM Corporation

Enterprise Service Bus Defining Concepts – the IBM View

Heterogeneity
Integrated, multi-protocol, multi-API, message distribution, routing and processing middleware to support extremely diverse systems and applications.



Service Orientation
Middleware that facilitates loose coupling between software components.

Incremental Integration
The distribution of low-function message brokers to provide low initial costs and step by step growth capabilities.

Event Orientation
Decoupling applications that publish business events from subscribing applications.

Universality
A universal connectivity layer extends the scope and scale of integration across an extended enterprise

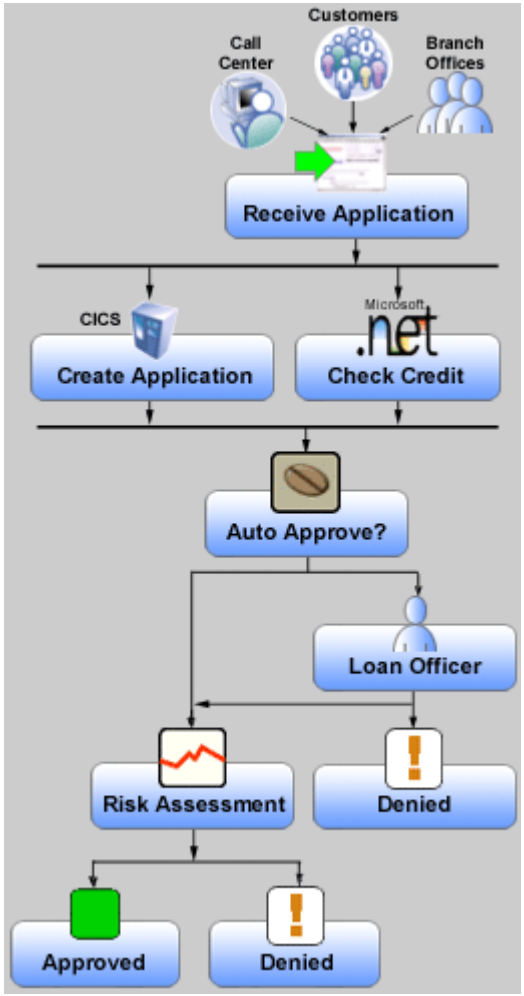
Qualities of Service
A variety of qualities of persistence, performance, reliability, security, availability and transactionality.

Interoperability
Open protocols support the interoperability of middleware from multiple vendors.

Substitution
Open APIs allow the substitution of middleware from different vendors.

© 2004 IBM Corporation

Process Choreography in a Service oriented world



BPEL4WS benefits

- Open standard, Portable
- State and Context Management
- Parallel processing
- Events/notifications
- Compensating heterogeneous transactions.

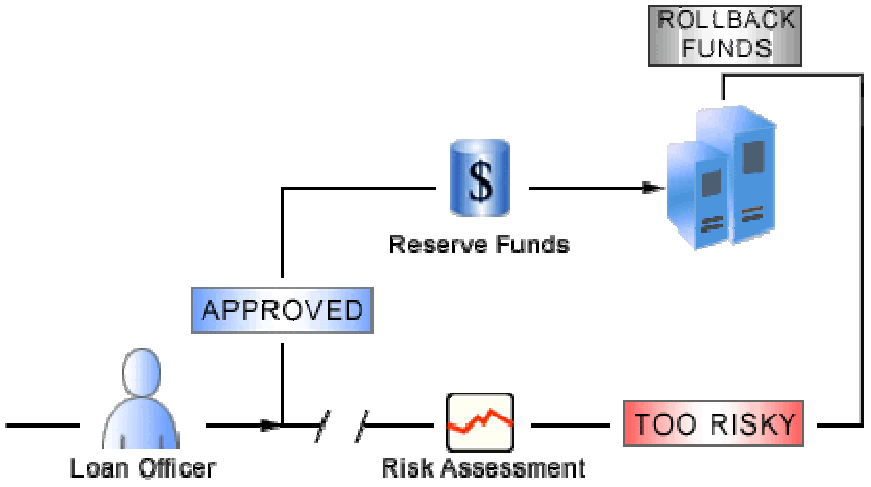
BPEL4WS constructs

Interactions

- <invoke>, <receive>, <reply>
- <wait>
- <assign>

Structure

- <flow>, <sequence>
- <switch>, <while>, <pick>



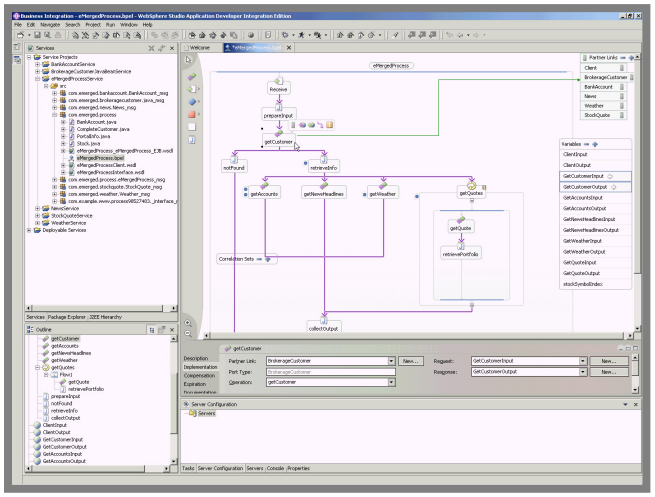
© 2004 IBM Corporation

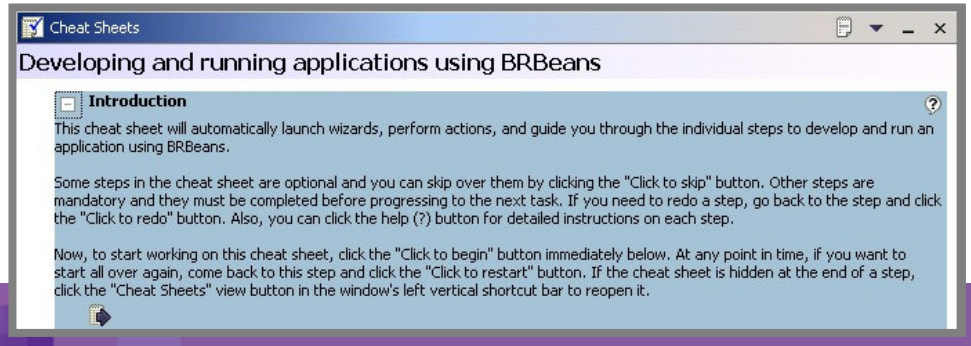
IBM provides …

The next generation integration platform for building, deploying, and choreographing Web services to form composite applications within a service oriented architecture

IBM WebSphere Business Integration Server Foundation &
IBM WebSphere Studio Application Developer Integration Edition

- Service oriented architecture
- BPEL4WS process choreography
- Human workflow support
- Business rules support
- Application adapters
- Programming model extensions
- Support for WebSphere Business Integration Modeler and Monitor (2H04)
- Common Event Infrastructure*
- J2EE Application Server
- Integrated J2EE development environment





*Tech preview in V5.1, GA delivery later this year

© 2004 IBM Corporation

© 2004 IBM Corporation