What's the Fuss about Fusion? SOA and Oracle

John Jay King
King Training Resources
john@kingtraining.com

Download this paper and code examples from:
http://www.kingtraining.com

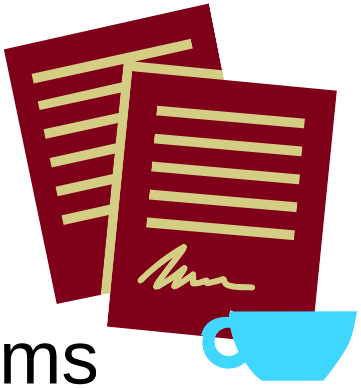
Where We Are Going

- What is Service Oriented Architecture (SOA)?
  - SOA Principles
  - What Enables SOA?
- Oracle Fusion Architecture
- Oracle Fusion Middleware
- Oracle Fusion Applications
- **Note:** *I am not and never have been an employee of Oracle Corporation, anything I say here represents personal conjecture based upon experience, practice, and observation*

What is SOA?

- Business functions as shared, reusable services
- Business processes, and the IT infrastructure that supports them, as secure, standardized, components (services) that may be combined to address changing business priorities
- SOA brings a new way to look at applications
  - SOA separates messages from processing
  - SOA services provide consistent results via predefined messages delivered by any number of mechanisms

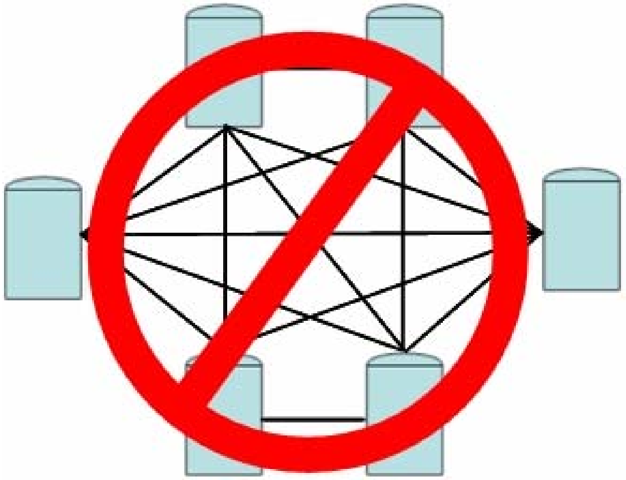
Menu (Service) Choices



- When ordering from a menu
  (a list of restaurant services)
  you have a predefined set of choices:
  main courses, side dishes, desserts, beverages, etc…
- However, when you order you specify the items and quantity desired, you don’t (usually) tell the chef how to prepare the meal
- The restaurant (kitchen and staff) receives the message, prepares, then delivers the food according to the menu’s description (“contract”)
- The menu is a directory of available choices, each choice showing the description and specifications for that item

SOA: Not Just Technology!

- SOA provides policies, practices, and frameworks used to ensure real synchronization between the business and IT to provide the right services
- SOA allows users of services to leverage their functionality independently from the IT infrastructure technology in use
- SOA requires a different mindset for IT professionals: reuse, modularity and architectural principles are central to SOA design
- Successful SOA requires that IT organization and processes must align with SOA principles, sharing resources and breaking down isolated "silos" of information



“Service” Definition

- SOA simplifies application development by focusing on standardized services; here's a definition of services from the Oxford English Dictionary:

service

noun 1 the action or process of serving. 2 a period of employment with an organization. 3 an act of assistance. 4 a ceremony of religious worship according to a prescribed form.
5 a system supplying a public need such as transport, or utilities such as water. 6 a public department or organization run by the state: the probation service. 7 (the services) the armed forces. 8 (often in phrase in service) employment as a servant. 9 a set of matching crockery used for serving a particular meal. (in tennis, badminton, etc.) a serve. a periodic routine inspection and maintenance of a vehicle or other machine. (verb definition omitted)

- “**a system supplying a public need**”; services are not designed for a specific customer, services provide functionality reusable in a variety of environments

Why Services?

- Services are reusable units providing business functionality that are:
  - Clearly defined using standard policies, practices, and frameworks
  - Clearly described (usually with XML)
  - Autonomous
  - Abstractions of the underlying business logic and functionality
- Business people understand services; one party delivers a service according to a contract with the other party

Services and Components

- Generally, services use one or more software components to satisfy some business functionality

  For example, the "Schedule Mortgage Closing" service may involve execution of many components (modules) in the underlying IT systems

Why Architecture?

- Architecture is necessary to properly manage the design and implementation of complex systems
- Like a building's blueprints and plans, IT architecture provides a high-level view of a system's structure:
  - Parts needed
  - Standards used
  - Responsibilities of each part
  - Interrelationships between various parts
  - Orchestration of activities to complete the plan
- Using an architecture does not guarantee success, but attempting to build something complex without an architecture will guarantee failure

Application Integration Evolves

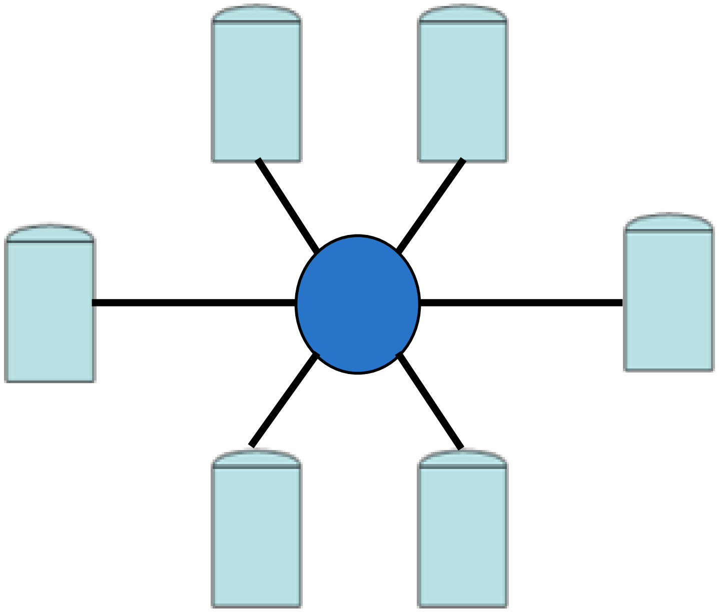
- IT has been creating complex computer application systems for many years
- SOA provides the next logical step in the integration of systems
- The flexibility and adaptability of SOA provides the ability to create first-quality solutions quickly
- By design SOA solutions are reusable, reducing ongoing costs and making IT more agile when changes occur to the business environment

Integration Today

- Early systems were application-specific offering point-to-point integration
- Application Integration (EAI) solutions provide a hub-and-spoke ability to interact between systems
- The SOA model extends EAI integrating applications using ***standards-based*** technology to define an “Enterprise Service Bus” (ESB)

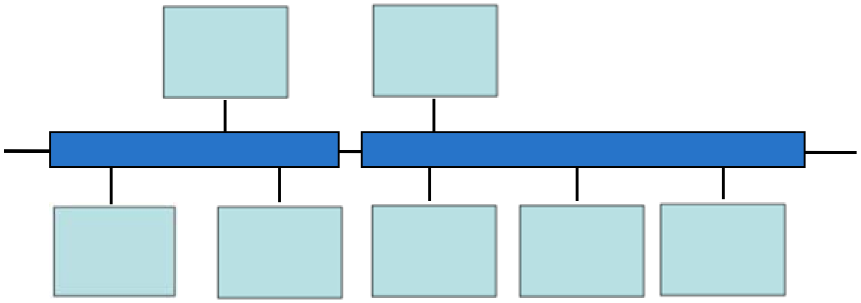
Integration: Hub-and-Spoke

- Hub-and-spoke (Integration Brokers) standardized interactions reduced the number of interfaces bringing some reuse and savings



“Enterprise Service Bus”

- Web services technology provides a “less expensive” and standards based approach to solve integration issues
- An Enterprise Service Bus (ESB) provides self-directing interfaces



  - Standards-based, reducing vendor lock-in and Total Cost of Ownership (TCO)
  - Distributed, allowing flexibility of deployment and business agility
  - Federated, allowing appropriate local autonomy and extension across enterprise business-to-business (B2B) boundaries
  - Fully scalable

Business Agility Benefits

- A fully implemented SOA provides the ability to rapidly and dynamically compose applications, services and even complete processes
- SOA allows new business opportunities to be grasped quickly providing quicker response to competitive business challenges
- Using SOA views of information from the same service may be used on a variety of devices including: desktops, laptops, PDAs, IVR, etc...
- SOA provides the ability to upgrade/replace components or even whole systems with less impact on other systems

Business Alignment Benefits

- SOA more closely aligns IT applications and business functionality:
  - Services are described at a business level
  - Services are closely aligned with meaningful business activities and business strategy
  - SOA design and document/message-based interactions closely mimic actual business processes and interactions
  - Service architecture and commitment to open standards allows services from multiple partners to be orchestrated into collaborative services

Other SOA Benefits

- Simplified interactive B2B (Business to Business) processes
- Cross-application consistency of business-process execution
- Increased “real time” interactions
- Independent services may be combined into new services
- Redundant development work reduced through greater reuse
- Standards-based implementations lead to:
  - Reduced infrastructure purchase costs
  - Reduced maintenance costs
  - Improved interoperability
- Service based approaches are quicker and easier to use than traditional integration and development mechanisms
- Internals of an application or service can be changed without impacting other applications

Benefits of Standards-Basis

- The standards-based nature of SOA reduces platform and tool vendor “lock-in”
- Consistency in methods and technology reduces the overall skill set required for development

SOA Principles

- A key strength of SOA is simplicity
- Basic principles guiding SOA:
  - Standard set of enterprise service definitions described in a registry
  - Central management of service definitions
  - Loose coupling

Central Management

- Central management of service definitions ensures that
  - Duplicate services are not created
  - Developers follow organization standards
  - Developers can find (and use) services

Loose Coupling

- Loose coupling is a long-standing IT term meaning that the internals of an application or business service must be able to be changed without impacting client applications;

  Specifically, a service consumer should not be required to know any more about a service than what is contained in the published contract

- Loose coupling means that message(s) used to interact with a service are 100% involved with the business function being performed ***without regard to how*** the function is being performed

Loose Coupling Everyday

- Here is an everyday example of Loose Coupling
- Most of you have driven or ridden in an automobile recently…
  - Was the accelerator pedal connected to the motor using a chain, cable, mechanical links, hydraulics, or electronics?
  - *IT DOES NOT MATTER!*
  - As long as the car “goes” when the accelerator is used, only car enthusiasts really care how it happens
  - The “Acceleration Service” is loosely-coupled, the automobile operator accelerates or decelerates without concern to exactly how the service is performed!

Common Language

- Interactions within an SOA should use a common business “language” to increase reusability
- Today, typically using XML (eXtensible Markup Language) defined by a common set of “schema” definitions
- Most common language XML Schema definitions cross-reference to Enterprise-level data definitions

Why XML Schema?

- XML Schema is probably the best technology available for representing the “common” language
- Before XML, organizations with SOAs created their own customized message syntax
- Each custom syntax requires:
  - Custom parser
  - Validation tool
  - Message serialization/deserialization
  - and more…

Coarse-Grained Communication

- Services are more highly coarse-grained than typical IT objects and components; frequently services map directly to a business function or activity
- Coarse-grained interactions are simpler and require fewer messages to use the service (less “clutter”)
- Designing services and interactions may be complex; different aspects of providers and consumers must be reconciled into a simple set of course grained communications
- Pass the entire “Purchase Order” as a coarse-grained unit rather than breaking it into PO Header and PO Detail Lines

Standards-Basis

- Architecture and technology choices need to be based upon open standards as much as possible and practical
  - – The number of tools that work with open standards increases daily
  - – Open standards ensure a supply of available tools
  - – Open standards increase the pool of available people
  - – Open standards make a ready supply of “pluggable” components and services likely
  - – Open standards increase the likelihood that tools from different vendors will “play nice” together

Proprietary Problems

- Proprietary technology and protocols should be avoided where possible to avoid vendor lock-in
  - Proprietary technology and protocols often requires specialized tools (and the tools are usually proprietary too)
  - When building a home: Would you rather your builder used standardized electrical and plumbing fixtures? - or - Would you rather use the "really cool" switches and drains the builder's cousin designed and created in his garage?

Well-Defined Contracts

- In order to interact successfully with a service, you must know at least two things:
  - What you expect to get from the service
  - What information you have to provide the service
- A well-defined “contract” from the service provider spells out business and technology requirements for using a service (the “interface”) and how to invoke the service
  - A service contract reflects specific business knowledge and is the basis for sharing and reusing services
  - Maintenance of service “contracts” becomes critical over time
  - Contracts are stored in a service registry

Enabling SOA

- What enables SOA?
  - Services
    Software components or sets of components
  - Service Providers
    Location where Services are available
  - Service Consumers
    Software actually using Services (sometimes user-facing)
  - Service Registries
    Contains “contracts” describing available services
  - Messaging
    Communications between Service and Service Consumer

Service Providers

- Service Providers provide a service that performs a business function at the request of the Service Consumer
- Service Providers:
  - Create or purchase the service
  - Expose the service
  - Describe the service using a contract
  - Post the contract and any pertinent meta-data on a registry available to the service consumer
    (Real World Intrusion: Service providers might also communicate the contract directly to the service consumer; hopefully the exception, but often the rule)

Service Consumer

- Services are meant for use between two pieces of software
- Service Consumers might be other services, components, or programs
- It is not “normal” for services to interact directly with humans; web ***consumers*** might provide with users on browsers or via personal electronics
- Service Consumer developers do not design, build, or test services, they just use them
- Service Consumers receive Contracts for Interaction from a Service Registry or directly from the Service Provider
- Because Services are designed to be reusable, any Service Consumer properly accessing a service will get predictable results

Service Registries

- Service Registries provide a mechanism for storing, managing, and accessing service contracts
- Various mechanisms have been developed for Service Registries over time; it is probably best to stick with an open standard solution using UDDI (Universal Description Discovery and Integration)
- A registry serves as a directory of sorts containing:
  - Names of services
  - Description of services
  - Service “Contracts” describing service parameters, results, and access methods
  - Documentation (meta-data) for the services

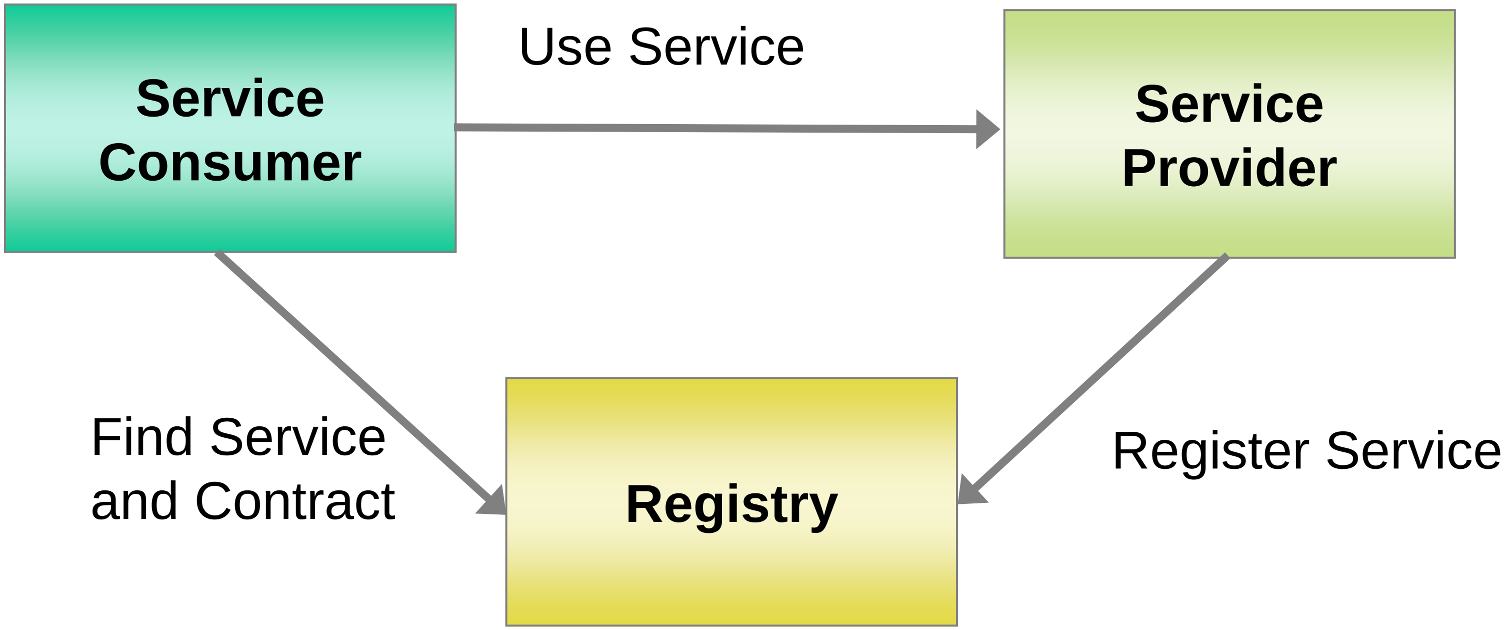
Service Messaging

- Services on their own are not much use; a messaging infrastructure is needed so that services may be:
  - Contacted
  - Passed the appropriate information
  - Allowed to respond to the service consumer
- Messaging is also important for requesting information about services from a registry, and returning service contracts from the registry
- Many successful messaging technologies are in place today that can support SOA
- In keeping with the principle of using Open Standards, SOA is most often implemented using SOAP (a W3C standard)

How the Players Interrelate

- The ***Contract*** describing the ***service***, its inputs and outputs, location, and method of invocation is placed in the ***Registry*** by the ***Service Provider***
- The ***Service Consumer*** locates a ***Service*** using the specifications found in the service's ***contract*** from a ***Registry***
- ***Service Consumers*** use ***Services*** provided by a ***Service Provider*** to perform all or part of some business function

SOA “Players”


Service Consumer
Use Service
Service Provider
Find Service and Contract
Registry
Register Service

Designing & Creating Services

- SOA design needs to create services that represent business process logic and process control
- SOA design is not merely a return to a functional design methodology from a distributed object methodology
- Services representing business functionality will be much more complex than those that merely provide simple data access
- Services must be designed for reuse
- Loose coupling is a requirement, not a suggestion
- Invocation should not require language or environment specific features
- Interfaces should be defined using industry-standard mechanisms (including XML) where possible

SOA and Existing Applications

- Many existing applications already expose some of their functionality as services
- Most vendors of software now provide mechanisms to create services to access existing functionality
- Business processes and data that are currently “trapped in silos” can be exposed via SOA and made available as services without regard to their actual location, language, or environment

Moving into the Future

- Design new applications from a service perspective:
  - Service orientation involves business departments
  - Breaking business functionality into manageable components (services) aids planning
  - Service orientation allows incremental roll out, allowing quick focus on the top 80% (20% of the effort)
  - Standards-based integration simplifies incorporation of new functionality into existing systems (especially via ESB)
  - Developers focus on creating web services; reducing the need to retrain staff in the “language de jour”
  - Services solidify project objectives by abstracting the gap between business requirements and technical details
  - Developers are **obligated** to see if existing services meet project needs before making or buying custom solutions

Orchestration

- Sometimes process flows are complex and require control
- Orchestration allows the combination of service executions in a specific sequence to satisfy a business process
  - Orchestration services usually require that a special processing engine be plugged into the ESB
  - Orchestration is typically scripted using BPEL4WS (Business Process Execution Language for Web Services)
  - Orchestration allows the creation of services using other services as building blocks increasing re-use and allowing IT to more flexibly adapt to business changes
- Because services are autonomous, it is possible to invoke multiple related services in parallel

BPEL Orchestration

- BPEL consists of specific steps:
  - Invoking web services <invoke>
  - Waiting for client to invoke a web service via a message <receive>
  - Generating responses for synchronous operations <reply>
  - Manipulating variables <assign>
  - Signaling faults and exceptions <throw>
  - Pausing for selected time <wait>
  - Ending the process <terminate>

BPEL “Coding”

- BPEL provides common programming constructs:
  - Sequence <sequence>
  - Flow <flow>
  - Path selection <pick>
  - Case construct <switch>
  - Looping <while>
- BPEL tools provide a pretty, graphical, paint-by-the-numbers interface

Contracts, Access, and ESB

- Service contracts need to be modeled and reviewed before being created, tested, and added to a Registry
- Ideally, all access to service contracts should be through the ESB (Enterprise Service Bus)
- A properly constructed ESB allows access to all services regardless of their origin
- Use of the ESB ensures that all Service Consumers will be able to access a service in a standardized fashion
- Access of a service through the ESB implies that the service lives up to its contract simplifying the task of developers (presumes governance)

Need for Governance

- All the Services in the world are **useless** unless:
  - We know what they are named
  - We know where to find them
  - We know the expected inputs and outputs
  - We trust them to work as specified in their contract

Governance

- Governance is needed to:
  - Make sure multiple services don’t provide the same functionality
  - Understand who is responsible for a given service
  - Prioritize and control change requests
  - Determine that services conform to standards
  - Ensure that contracts are accurate
  - Provide a level of comfort that advertised services work and can be accessed as described by their contract
  - Be sure that services are cataloged and can be located

Key SOA Best Practices

1. Explicitly define business processes and services
2. Design application and system functionality as accessible and reusable business services
3. Expose service functionality through well defined interfaces
4. Describe service interfaces consistently
5. Align services with business functions and processes
6. Manage service definitions across the enterprise
7. Advertise and discover services using standards-based registries
8. Communicate with services using coarse grained messages over industry standard protocols
9. Use a common language for interactions (probably XML)
10. Adhere to SOA principles

What are Web Services?

- Web Services are:
  - Components of services representing all or part of a business process
  - Designed to be invoked via a network
  - Typically invoked using SOAP over HTTP (the same standard that enables the Web)
- Web Services are self-contained and self-describing
- Web Services can be used by any Service Consumer residing on any platform with web connectivity (if designed properly)

Standards and Web Services

- Web Services are language and technology independent (if designed properly)
- A Service Consumer written in any language may use a Web Service no matter what language the Service Provider used to create it (if designed to)
- Web Services are built using well-known and vendor-independent standards and protocols including: HTTP, XML, SOAP, WSDL, and UDDI
- Web Services are not the only way to implement SOA; however, most successful SOA implementations use Web Services due to the ubiquitous nature of HTTP, XML, and the other standards-based tools available today

Is Everything on the Web a Web Service?

- No!
- The World Wide Web provides a convenient standards-based mechanism for people and organizations to share information via networks
- Many Web pages provide “services” to the user that are supported by a variety of software, most of this functionality is not currently created with Web Services
- Web Services are built upon the same basic open standards that support all web pages and also upon SOA-oriented standards like SOAP, WSDL, UDDI, and others

SOA is Standards-Based

- The Web (WWW) works because of widely adopted standards; Service Oriented Architecture (SOA) is language and platform independent; following standards makes integration easier
- Most Enterprise Service Bus (ESB) products convert non-standard data and messaging to standardized messages reusable throughout the bus
- Standards central to SOA with Web Services:
  - SOAP
  - WSDL
  - UDDI

SOAP

- SOAP is an XML application used for messaging or other services; since XML is environment and language independent SOAP messages may be used anywhere
  - SOAP used to be an inaccurate acronym (Simple Object Access Protocol)
  - Today SOAP is simply “SOAP”
- SOAP is supported by most networking mechanisms, operating systems, and servers available today

SOAP Messages

- SOAP messages consist of:
  - SOAP envelope containing message header and message body
  - SOAP attachments (optionally non-XML content)
- The message header is used to include message routing, delivery, security, and encryption settings
- The message body (sometimes called “payload”) is any XML that might be useful to send in a message
- More information may be found at: http://www.w3.org/tr/soap

SOAP Envelope

- SOAP envelope contents are dictated by standards
  - Element name must be "Envelope"
  - Element may use namespace and other attributes, attributes must be qualified using namespaces
  - Subelements are allowed, they must also be qualified using namespaces

```
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   soapenv:encodingStyle="http://schemaws.xmlsoap.org/soap/encoding/">
   <soapenv:Header>
     <!-- Header XML goes here -->
   </soapenv:Header>
   <soapenv:Body>
     <!-- Application Data XML goes here: (e.g. an element naming a service
            with subelements naming arguments/parameters and their values) -->
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP Message

- Here is an example SOAP message used to request web service execution

```
<soapenv:Envelope
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header/>
<soapenv:Body>
    <p864:fStrToCstr xmlns:p864="http://example">
        <inFahrenheitStr>32</inFahrenheitStr>
    </p864:fStrToCstr>
</soapenv:Body>
</soapenv:Envelope>
```

WSDL

- Web Services Description Language (WSDL) is an XML language describing how programs interact with Web Services
- Service “contracts” are usually WSDL (wiz-dull) documents representing the contract
- For each service, the WSDL contract specifies:
  - Address of the Web Service
  - Available operations
  - Format of operation arguments
  - Exceptions that may be thrown
- More WSDL information may be found at: http://www.w3.org/wsdl

UDDI

- Universal Description Discovery and Integration (UDDI) provides a standard registry of web services so that others may find and use them
- UDDI is one way to implement a Service Registry
  - Originally, the creators of Web Services intended for application developers and Service Consumers to search the internet for services that might be applicable and then use them
  - Security and other concerns have brought UDDI registries inside the firewall in most installations

UDDI Contents

- UDDI contains three kinds of entries:
  - White Pages
    - Business address
    - Contacts
    - Identifiers
  - Yellow Pages
    - Service Categories (industry-specific, geographic, or other codes)
  - Green Pages
    - Technical documentation about services
- More information about UDDI may be found at: http://www.uddi.org

Enterprise Service Bus (ESB)

- The Enterprise Service Bus (ESB) is the backbone of SOA
- ESB is a new approach to integration
- An ESB is a standards-based integration platform combining messaging, web services, data transformation, and dynamic routing
- ESBs improve on the Hub-and-Spoke Integration Broker concept because of the standards-based orientation
- The ESB provides a “transparent pipeline” for SOA through which messages and events flow via multi-protocol message bus routing dynamically across the available network

Using ESB

- Applications and Event-Driven Service components expose course grained loosely coupled interfaces to the ESB making them reusable more-broadly to the enterprise
- Like a passenger bus, an ESB collects a rider (Service Consumer request) and takes it to a destination (Service Provider) without telling the driver (ESB) how to get there
- The Service Consumer and the Service Provider only need to know how to talk to the ESB
(usually using standards-based tools)

ESB Capabilities and Services

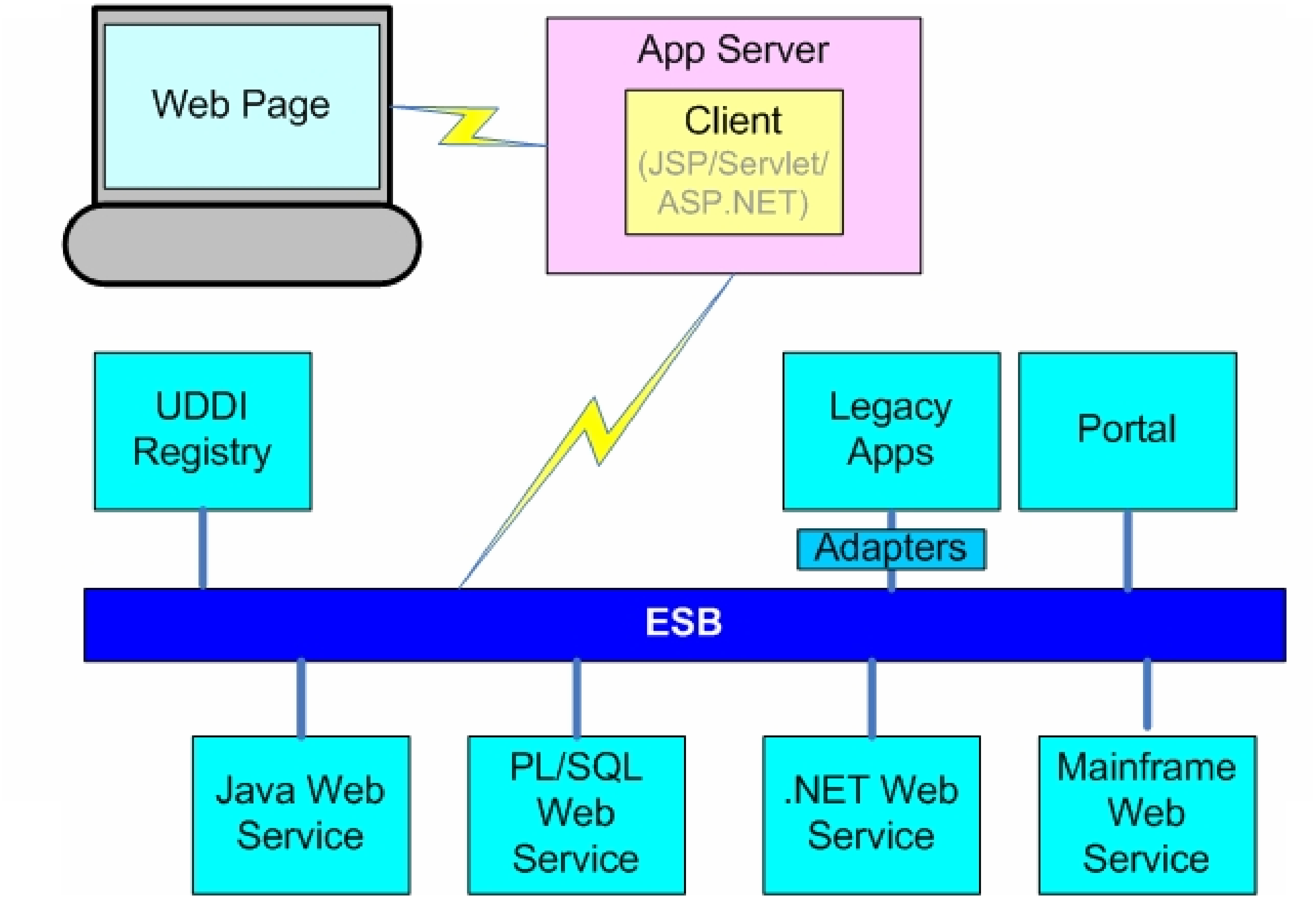
- Here are some capabilities and features that are usually part of the ESB or accessed by it:

ESB Messaging	Robust, reliable messaging
Routing	Routing anywhere on the ESB
Transformation	Data transformed/translated anywhere on the ESB
Security	Unified security and access model
Management	Centralized local and remote ESB management
Availability	Software to ensure availability and reliability
Others	Service Location, Process Execution/Monitoring, Monitoring, Policy Management, and Subscription Management

EAI or “Pure” ESB?

- Enterprise Application Integration (EAI) has been around for years, many vendors are adding SOA-awareness to their offerings
- Reuse of “legacy” assets is one of the much-sought but frequently unrealized goals of EAI
- Enterprise Service Bus (ESB) vendors hope to provide a standards-based mechanism allowing more agility than current EAI implementations

ESB in Use


Web Page
App Server
Client
(JSP/Servlet/
ASP.NET)
UDDI
Registry
Legacy
Apps
Portal
Adapters
ESB
Java Web
Service
PL/SQL
Web
Service
.NET Web
Service
Mainframe
Web
Service

ESB Vendors

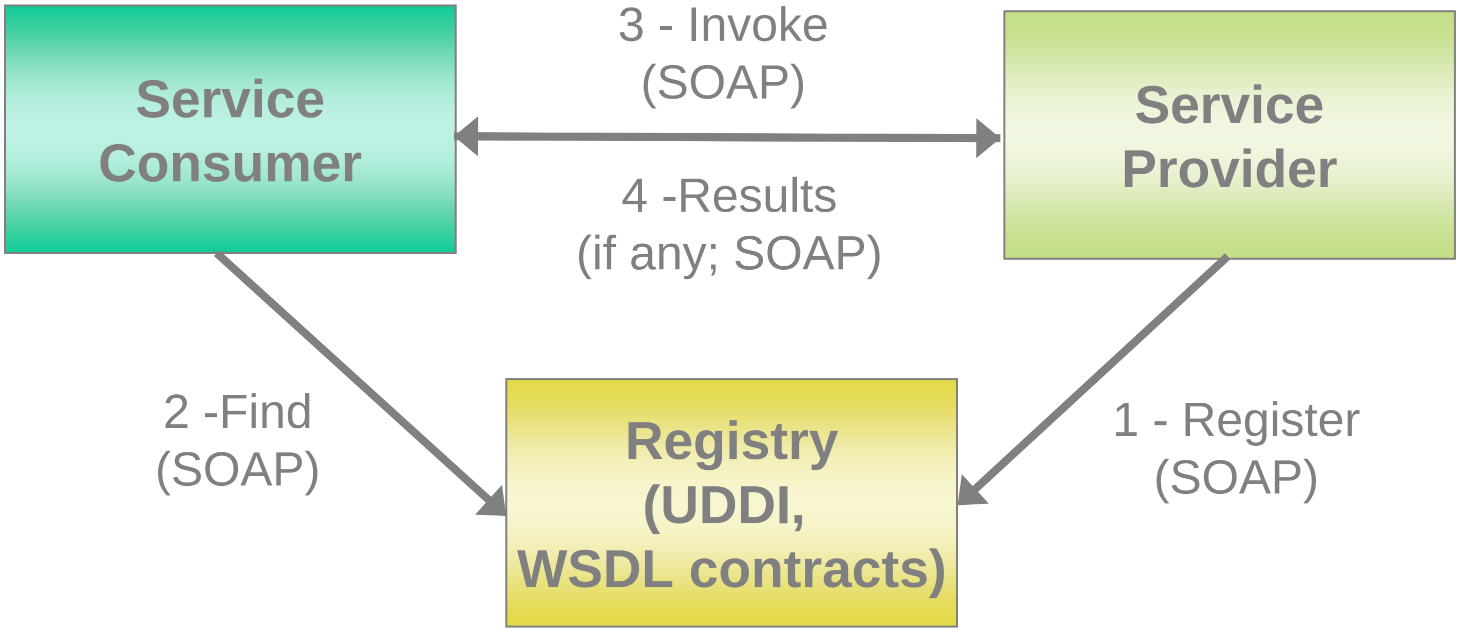
- Some of the major vendors of ESBs include:
  - BEA Systems AquaLogic
  - **Oracle SOA Suite**
  - TIBCO Software Business Works
  - Fiorano Software SOA 2006 Platform
  - Cape Clear Software ESB
  - IBM WebSphere Message Broker
  - Software AG Enterprise Service Integrator
  - Sonic Software SOA Suite
- Several open source ESBs becoming available too

Emerging Standards

- Several “WS-*” standards are ***emerging*** that will eventually have impact on SOA and ESB implementations
  (see http://www.oasis-open.org for more)

- These standards might impact SOA and ESB first:
  – WS-ReliableMessaging and WS-Reliability specify standards for SOAP header elements, behavior, acknowledgements, retries, and fault handling (WS-ReliableMessaging was created by a consortium including Microsoft, and IBM to improve the OASIS standard WS-Reliability)
  – WS-Security describes security enhancements to SOAP including tokens, encryption, and other security features

Players Revisited


Service Consumer

Service Provider

Registry (UDDI, WSDL contracts)

3 - Invoke (SOAP)

4 -Results (if any; SOAP)

2 -Find (SOAP)

1 - Register (SOAP)

What About REST?

- REST is another way organizations are implementing Web Services using a simpler technology stack
- REST (Representational State Transfer) describes a different architecture than SOA (the term REST originated in a doctoral dissertation about the web written in 2000 by Roy Fielding) "Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."
- REST is often used to (loosely) describe transmission of domain-specific data via HTTP without SOAP or any type of session tracking; REST works well with Java HTTP objects
- Web Service invocation via Ruby on Rails is often performed using REST rather than SOA architecture

Oracle Fusion Architecture

- So how does Oracle's Fusion fit in?
- Oracle uses the title "Fusion" to unify its SOA-directed offerings and highlight the integration features incorporated in their products
- Two major legs of Oracle Fusion Architecture identified so far are:
  - Oracle Fusion Middleware
  - Oracle Fusion Applications

Fusion Architecture Core Principles

- Oracle outlines five core principles to Fusion Architecture:

– Model Driven	Following business processes
– Service and Event-Enabled	Loosely-coupled, modular, and flexible
– Information-Centric	Providing complete, actionable information
– Grid-ready	Scalable via low-cost hardware
– Standards-based	Based upon open standards allowing easy interaction with other products

Oracle Fusion Middleware

- Built upon Oracle Application Server 10G
- Oracle Fusion Middleware builds on the solid Java EE and open-source architecture of Oracle Application Server improving application integration
- SOA emphasis on business processes in Oracle Fusion Middleware leads to better coordination between Information Technology groups and Business units
- Oracle Fusion Middleware comes complete with over 250 adapters to existing application systems including (but not limited to): Oracle E-Business Suite, PeopleSoft, and JD Edwards

Oracle Fusion SOA Suite

- Oracle Fusion Middleware includes the SOA Suite:
  - BAM (Business Activity Monitoring) providing real-time access to business performance information
  - BPEL (Business Process Execution Language) Process Manager for defining and executing business processes
  - Business Rules Engine to manage business rules
  - Web Services Manager for security (Oracle Directory, Active Directory, LDAP) and management
  - ESB (Enterprise Service Bus) to provide routing and messaging
  - JDeveloper 10g provides a unified SOA Suite toolset
- In addition the Oracle Service Registry, Oracle Portal, and other products are also available

SOA Suite Everywhere

- Oracle Fusion SOA Suite works with any Java EE Application Server including Oracle Application Server or any other Java EE Application Server such as BEA WebLogic, IBM WebSphere, JBOSS, or others

  (caveat, some Oracle-specific features will require Oracle Application Server or additional licensing)

Fusion Applications & Middleware

- Oracle Fusion Applications rely heavily on Oracle Fusion Middleware; the opposite is not true
- An organization may use Oracle Fusion Middleware and its wide array of tools even if Oracle Fusion Applications are not installed
- Oracle Fusion Middleware's reliance on industry standards (like SOAP, WSDL, and UDDI) and SOA makes it an excellent choice no matter how applications are supported in an organization

Fusion Middleware is already good, but will get better and better since it is the lynchpin that thousands of Oracle's developers are using to build Fusion Applications. ***We all win!***

Oracle Fusion Applications

- Fusion Applications are the next generation of Oracle's Applications products
  - Oracle E-Business Suite
  - PeopleSoft
  - JD Edwards Enterprise
  - JD Edwards World
  - Siebel
  - Retek
  - more…
- Rather than “stitching together” disparate technologies, Fusion uses a service-oriented architecture to make the functionality of the various tools available

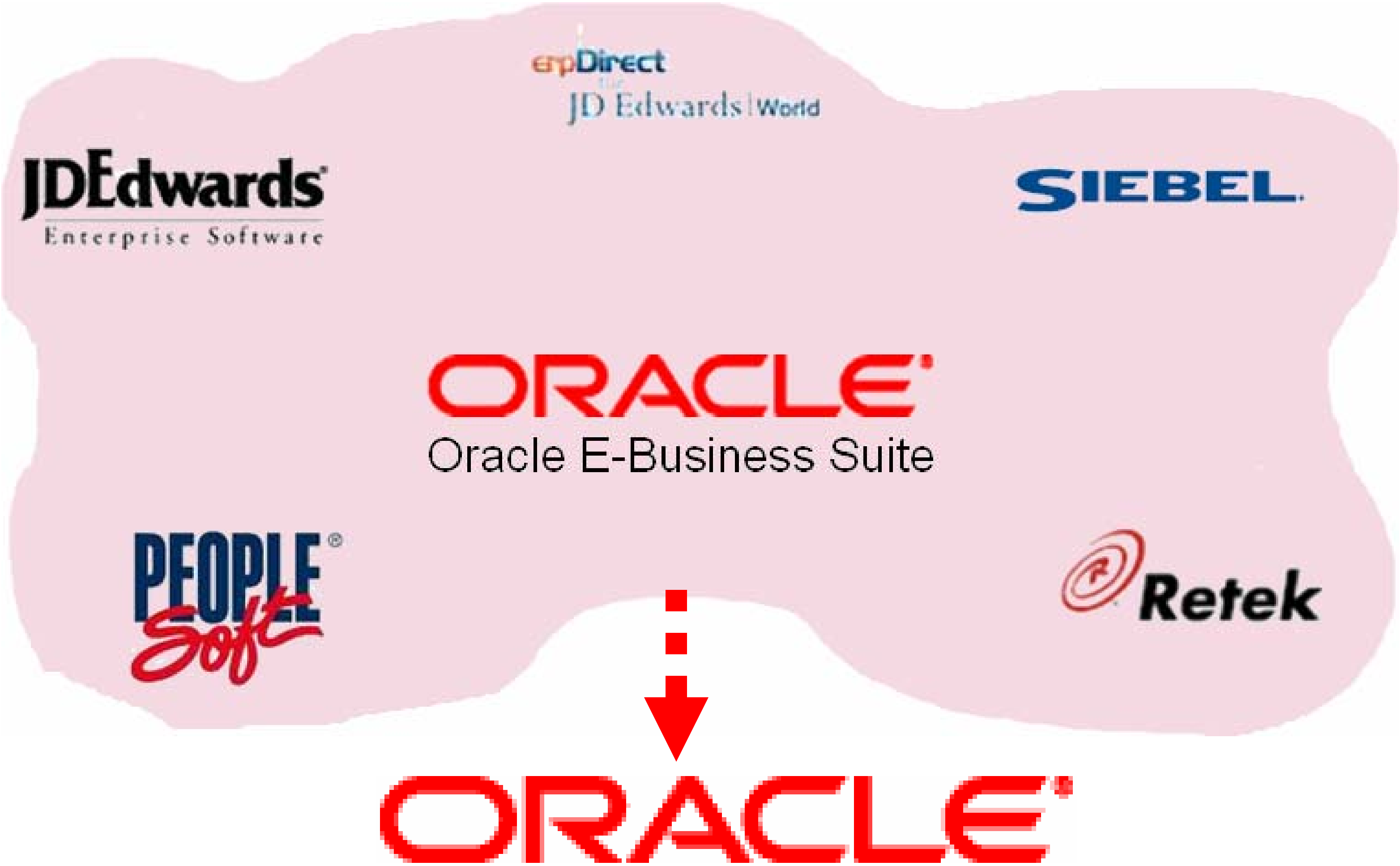
Fusion: Moving Forward

- Rather than customizing applications, business process modeling may be used to orchestrate existing functions to handle issues not addressed by current applications
  (building upon a foundation of work first begun at PeopleSoft)
- As the Oracle Applications products move forward they want to take advantage of the best features of each environment, making future products better
- Oracle Fusion is really a destination point

Fusion and SOA

- Oracle is relying on SOA to develop the next generation of Oracle applications
- SOA provides a standard method of integrating and evolving business application functionality from the Oracle E-Business Suite, PeopleSoft, JD Edwards, and other Oracle application products
- The industry-standard nature of SOA will allow third-parties to extend Oracle Fusion Applications adding new functionality or customized services

Fusion in Action



Fusion Applications

Recap: Oracle and “Fusion”

- Oracle chose the “Fusion” name to highlight the synergy of components developed by Oracle and others *(I think…)*
- Fusion Middleware is key to Fusion Applications; many key components came from Fusion Application requirements
- Fusion Middleware can be used without Fusion Applications
- Those of use who use the Oracle Middleware tools will benefit greatly due to Oracle’s use and dependence upon the Oracle Middleware products

"Fusion Profusion" (just for grins…)

RMOUG

ROCKY MOUNTAIN ORACLE USERS GROUP

Training Days 2008

February 13-14 2008!

2 Days – Denver Colorado

$240 for RMOUG, IOUG, ODTUG

(or other Oracle user-group members)

$300 for non-user-group attendees

http://www.rmoug.org

(make it a Valentines weekend in the Rockies!)

IOUG-Collaborate 2008

IOUG-Collaborate April 13-17 2008

Denver Colorado!

Oracle Development Tools User Group

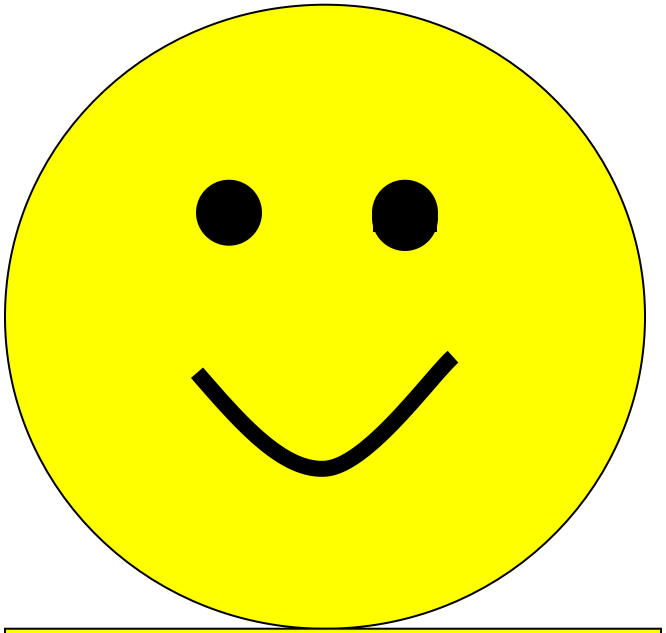
WWW.ODTUG.COM

A *Real World* User Group

For *Real World* Developers

ODTUG Returns to . . .
THE BIG EASY!
June 15-19, 2008
Sheraton New Orleans

What's the Fuss about Fusion? SOA and Oracle



Thanks for your attention!

To contact the author:

John King

King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com

Today's slides are on the web:

http://www.kingtraining.com