Service Oriented Architectures and the JBoss SOA Platform

Dr Mark Little
Technical Development Manager, Red Hat
Overview

• SOA in a nutshell
  – Degrees of coupling
  – The component triad

• Relationship to WS-*

• The JBoss SOA Platform
  – Registries and repositories
    • BRMS
  – Message delivery and transformation
  – Service orchestration

• Futures
  – Transaction processing in a SOA
What is SOA?

• An SOA is a specific type of distributed system in which the agents are "services" (http://www.w3.org/TR/2003/WD-ws-arch-20030808/#id2617708)

• Adopting SOA is essential to delivering the business agility and IT flexibility promised by Web Services.

• But SOA is not a technology and does not come in a shrink-wrapped box
  – It takes a different development methodology
  – It’s not about exposing individual objects on the “bus”
Services

• **Services represent building blocks for applications**
  – Allowing developers to organize their capabilities in ways that are natural to the application and the environment in which they operate.

• **A Service provides information as well as behaviour and it does not expose implementation (back-end) choices to the user.**
  – Furthermore a service presents a relatively simple interface to other services/users.
Tightly coupled

• A distributed application consists of several distinct components

• Traditional client and server technologies based on RPC
  – Hide distribution
  – Make remote service invocation look the same as local component invocation

• Unfortunately this tightly coupled applications
  – Such applications can be brittle
Loosely coupled

• SOA is an architectural style to achieve loose coupling
  – A service is a unit of work done by a service provider to achieve desired end results for a consumer.

• SOA is deliberately not prescriptive about what happens behind service endpoints
  – We are only concerned with the transfer of structured data between parties.

• SOA turns business functions into services that can be reused and accessed through standard interfaces.
  – Should be accessible through different applications over a variety of channels.
But …

• There are degrees of coupling and you should choose the level that is right for you

• At the one extreme
  – Defining specific service interfaces, akin to IDL
    • Easier to reason about the service
    • Limits the amount of freedom in changing the implementation

• At the other extreme
  – Single operation (e.g., doWork)
    • More flexibility is changing the implementation
      – Well, almost …
    • More difficult to determine service functionality a priori
      – Need more service metadata
What about Web Services?

• Popular integration approach
  – XML
  – HTTP
  – Pretty much universal acceptance (see bullets above!)

• Not specific to SOA
  – Web Services began life as CORBA-over-HTTP
  – XML-RPC

• Web Services+SOA gives benefits
  – Loose coupling
  – Interoperability
  – Enterprise capabilities, e.g., security and transactions
Relationship to WS-*

Web Services Standards Overview
Fortunately ...

- SOA is technology agnostic
- WS-* offers the potential for interoperable SOA
- But it is just as easy to develop closely-coupled applications in WS-*
- Most vendor WS-* tools are direct mappings of distributed object tools
  - SOA != distributed objects with angle brackets
- A SOA infrastructure should support and encourage SOA principles
  - Sometimes it is easier said than done
The JBoss SOA Platform

• A Service Oriented Infrastructure
  – Based on JBossESB, Drools, JBossWS, JBossTS, JBoss Messaging and jBPM
  – Can run stand-alone or be deployed into JBossAS

• JBossESB acts as the glue
  – Supported protocols and capabilities make it more of an Internet Service Bus
  – Currently uses the “doWork” service definition approach

• Encourages an incremental approach to SOA
  – You don’t need to be a domain expert to benefit from it
  – Build up your knowledge in step with your requirements
Relationship to JBossESB

• Messages and services are key to architecture
• Inherently asynchronous
  – Correlated one-way messages for RPC
• Support for Web Services
• Support for task management
• Adapters
  – JCA
  – Gateways
• Flexible architecture
  – Multi-implementation approach
Where does it fit?
SOA components

• The key components of a Service Oriented Architecture are
  – The messages that are exchanged
  – The agents that act as service requesters and service providers
  – The shared transport mechanisms that allow the flow of messages

• A description of a service that exists within an SOA is essentially just a description of the message exchange pattern between itself and its users
Component triad

- **UDDI Registry**
  - Service Description (WSDL document)
  - SOAP Discover
  - SOAP Publish
  - SOAP Bind
  - Service Requestor
  - Service Provider
Repository

- **Service metadata, which is important for contract definitions**
  - Functional and non-functional aspects
    - Transactional, secure, QoS, …
    - Policies
  - MEPs
    - One-way
    - Request-response
  - Message structure
    - Where data resides
  - Governance
- **Service binaries**
- **Business rules**
- **Workflow tasks or process control information**
Services and messages

- Within the SOA-P everything is a service
- All services are interacted with via messages
  - Messages are part of the contract between client and service
- Messages do not imply specific implementations of carrier-protocol
- Services do not need to be bound to specific implementations of carrier-protocol
  - Email, S-FTP, JMS, File, etc.
  - More can be added as required
The Message envelope

- Attachments
- Body
- Context
- Header

- Payload
- Payload
- Sessions
- Routing
Message implementations

- On-the-wire representation may be tailored for environment
  - E.g., binary versus text
- Only the structure of the Message is mandated
- Two wire-formats provided
  - Java Serialized
  - XML
- Others can be added statically or dynamically
Message delivery in the SOA-P

• Addressed via WS-Addressing Endpoint References
  – Transport agnostic

• Supports request-response as well as one-way MEP

• Mandatory to define the recipient address

• Optional
  – Reply address
  – Message relationship information
  – Fault address
Gateway Services

- Need to allow legacy services to plug-in to the bus
- Need to allow legacy clients to plug-in to the bus
- Neither have concept of Message or EPR
- Must bridge from ESB-aware to ESB-unaware domains
  - Gateways perform this role
- This allows the bus to be extended across the enterprise without perturbing existing infrastructure
Service registration

• Services are identified by Service Name but addressed by EPR
  – Can be clustered for high availability and load balancing
• Registry associates <Service Name, EPRs>
• Service may be available on more than one EPR
  – E.g., different qualities of service
• Services are expected to store EPR when activated
• Senders look up EPR(s) using Service Name
  – May select on other criteria
Content based routing

- Intermediary services can redirect messages based on content
  - Hiding federating service implementations
  - Business logic choices
  - Fault tolerance
- Not a requirement for SOA
  - But does help loose coupling and legacy integration
- SOA-P has a CBR Service
  - Supports JBoss Rules and XPath expressions
Web Service example

Content Based Router

Service

Service

Service
SOA Platform example
rule "Routing Rule - Serialized based message"

    when
        Message( type == MessageType.JAVA_SERIALIZED)
    then
        System.out.println("Serialized");
        destinationServices.add("test_category:Serialized_ServiceDestination");

end

rule "Routing Rule - XML based message"

    when
        Message( type == MessageType.JBOSS_XML)
    then
        System.out.println("JBOSS_XML");
        destinationServices.add("test_category:JBOSS_XMLDestination");

end
Message transformation

• Different services may communicate in different vocabularies
  – Particularly with dynamic service registration/updates

• Data may need to be restructured based on recipient, time of day, etc.

• Several ways to do transformation

• Transformation Service
  – Smooks
  – XSLT
  – Others can be plugged in
Message store

- Messages can be durable recorded
- Useful for audit trail, debugging, replay etc.
  - Sometimes mandated by local laws
- Separate service
- Flexible implementations possible
  - Service API does not impose implementation restrictions
  - Out-of-the-box uses JDBC
Service orchestration

- Orchestration (e.g., BPM or workflow) is important in many distributed environments
  - More so as the scale and complexity increases
- Need to have intra service task orchestration
  - Control the transition of the state of a service as it executes tasks
- Need to have inter service orchestration
  - Control the invocations of services as messages flow through the infrastructure
- SOA-P supports both approaches
  - jBPM
  - WS-BPEL
Orchestrating message flows
Fault tolerance

• Machines and software fail
  – Fundamental universal law (entropy increases)
  – Things get better with each generation, but still statistically significant

• Failures of centralized systems difficult to handle

• Failures of distributed systems are much more difficult
Fault tolerance techniques

• Replication of resources
  – Increase availability
    • Probability is that a critical number of resources remain operational
    • “Guarantee” forward progress
  – Tolerate programmer errors by heterogeneous implementations

• Spheres of control
  – “Guarantee” no partial completion of work in the presence of failures
What is a transaction?

- Mechanistic aid to achieving correctness
- Provides an “all-or-nothing” property to work that is conducted within its scope
  - Even in the presence of failures
- Ensures that shared resources are protected from multiple users
- “Guarantees” the notion of shared global consensus
  - Different parties in different locales have the same view of the transaction outcome
SOA characteristics

• Business-to-business interactions may be complex
  – involving many parties
  – spanning many different organisations
  – potentially lasting for hours or days

• Cannot afford to lock resources on behalf of an individual indefinitely

• May need to undo only a subset of work

• Need to relax ACID properties
Transaction interoperability

- Web Services are as much about interoperability as they are about the Web
- In the short term will be about interoperability between existing TP systems
  - Achievable with JBossTS
Transactions for SOA

• **Relax isolation**
  – Internal isolation or resources should be a decision for the service provider
    • E.g., commit early and define compensation activities
    • However, it does impact applications
      – Some users may want to know a priori what isolation policies are used
  – Undo can be whatever is required

• **Relax atomicity**
  – Sometimes it may be desirable to cancel some work without affecting the remainder
    • E.g., prefer to get airline seat now even without travel insurance
  – Similar to nested transactions
    • Work performed within scope of a nested transaction is provisional
    • Failure does not affect enclosing transaction
Heisenberg’s Uncertainty Principle

• Cannot accurately measure both position and momentum of sub-atomic particles
  – Can know one with certainty, but not the other
  – Non-deterministic measurements

• Large-scale/loosely-coupled transactional applications suffer the same effect
  – Can know that all services will eventually see same state, just not when
  – Or at known time can determine state within model/application specific degree of uncertainty

• Or another way of thinking about it …
  – No such thing as simultaneity in data space as there isn't in space-time
    • “Data on the Outside vs. Data on the Inside”, by Pat Helland
Conclusions

- **SOA is an important design-time and use-time approach**
  - SOA is NOT a product
  - Requires changes to organizational view of software components (services)
- **Web Services are important**
  - Interoperability
  - Internet-scale computing
  - But SOA applications are not inherent in WS-*
- **JBoss SOA-P can bridge the divide**
  - A single infrastructure that provides SOA support
- **Get involved**
  - Start by downloading JBossESB and give it a try (http://labs.jboss.com/jbossesb)
  - Lots of examples
  - Contribute