Implementing a Ground Service-Oriented Architecture (SOA)

March 28, 2006

John Hohwald

Implementing a Ground SOA

Definitions and Terminology

- *What is SOA?*
  - *SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners. [xml.com]*
  - *Characteristics*
    - Documented and discoverable interfaces (APIs)
    - Standards-Based
    - Infrastructure neutral – use of H/W, OS, and S/W used to implement the service is encapsulated from the service consumer
- *What is a "Web Service"?*
  - *A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols [w3.org]*
- *"Web-Enabled" does NOT EQUAL Web Service nor SOA*

Implementing a Ground SOA

Key Standards (WS-I)

Additional Capabilities	Management	Portals	
Business Process Orchestration	Composition/Orchestration		
Composable Service Elements	WS-Security	Reliable Messaging	Transactionality
Messaging	Endpoint Identification, Publish/Subscribe		
Description	XML Schema, WSDL, UDDI, SOAP with Attachments		
Invocation	XML, SOAP		
Transports	HTTP, HTTPS, Others		

Implementing a Ground SOA

WSDL Styles

- A WSDL document describes a web service
- A SOAP message is used to invoke a web service
- WSDL-to-SOAP Binding Style includes the following
  - RPC/Encoded
  - RPC/Literal
  - Document/Literal
  - Document/Literal wrapped (design pattern)
    - Better performance; no overloaded operations; WS-I Basic Profile compliant

Transport Protocols

- Synchronous
  - HTTP, HTTPS, RMI/IIOP, SMTP
  - More appropriate for Request/Response message pattern
- Asynchronous
  - JMS, IBM MQSeries, MS Messaging (MSMQ)
  - Reliable delivery at transport level – needed for "guaranteed delivery"

Implementing a Ground SOA

- Messaging Exchange Patterns
  - Based Upon WSDL 1.1 Specification
    - One-Way: Endpoint receives a message
    - Request/Response: Endpoint receives a message, and sends a correlated message (usually correlated-id)
    - Solicit/Response: Endpoint sends a message, and receives a correlated message
    - Notification: Endpoint sends a message
- More complex patterns can be constructed from these four patterns
  - Application-level logic agreed to among provider and consumers (e.g. who will provide the correlator_id)

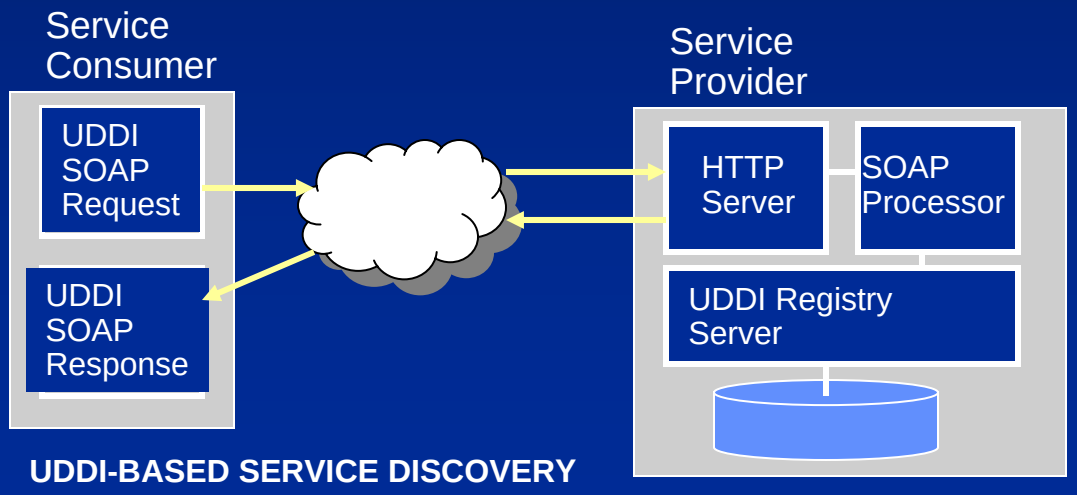
Implementing a Ground SOA

- SOAP, REST
  - Simple Object Access Protocol (SOAP) Envelope for XML-Document Payload
    - Protocol for encapsulating and passing data in a platform-neutral manner
    - Uses POST command when transport is HTTP; other transports protocols possible
    - Supports discovery and documentation via Universal Description, Discovery and Integration (UDDI) compliant registry
  - Representational State (REST)
    - REST → HTTP
    - Describes a style for creating web services using HTTP request methods (GET, POST, PUT, DELETE)
    - Easier to code and develop: Similar to common web pages
    - Do not lend themselves to self-discovery or description in a UDDI-compliant registry
    - More suited to data access (query) services on a reliable, internal Intranet

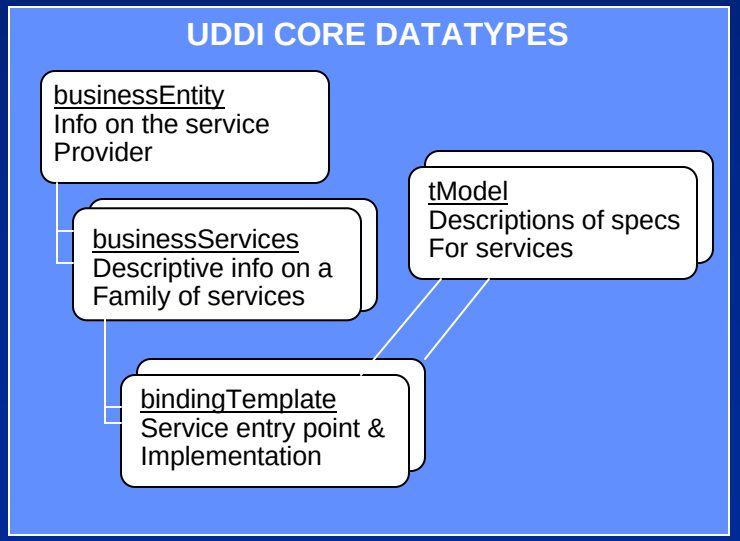
Implementing a Ground SOA

UDDI and Service Registries

- UDDI Standard
  - One of the core Web Services standards. It is designed to be interrogated by SOAP messages and to provide access to WSDL documents describing the protocol bindings and message formats required to interact with the web services listed in its directory.
- Service Registry Implementation Design Principles:
  - Deploy both Public and Private Registries
  - Employ both Design-Time Discovery, but Run-Time Binding



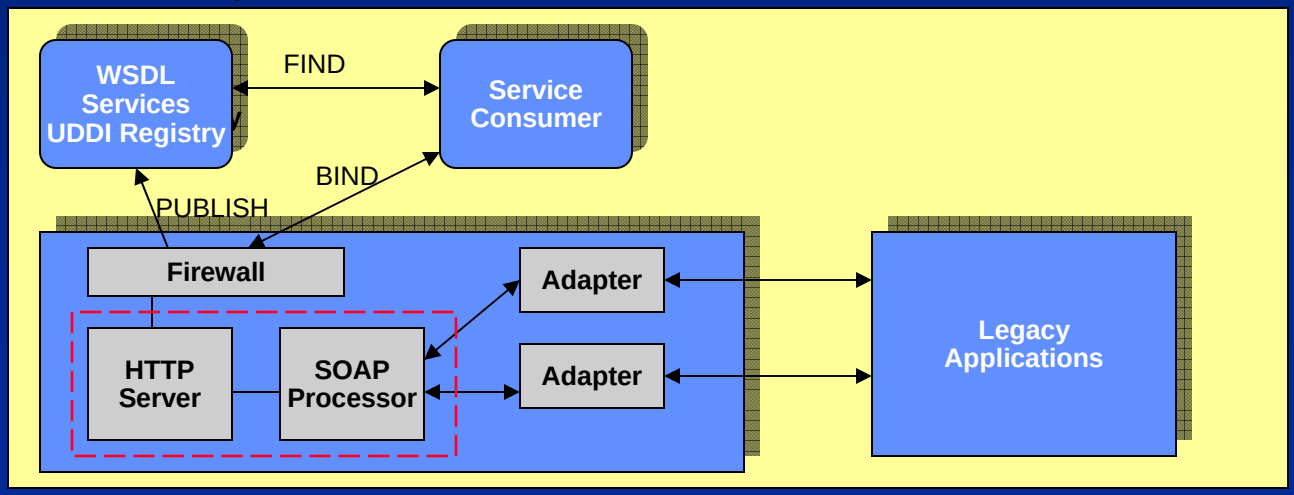
UDDI-BASED SERVICE DISCOVERY



Implementing a Ground SOA

Utilizing Existing Software

- SOA is nearly ideal for leverage legacy software investments
- SOA COTS tools (e.g. IBM Websphere, BEA Weblogic, etc.) on the market are JAVA-centric (aside from Microsoft IIS which is .NET)
- Wrapping existing software, e.g. C or C++ code
  - gSOAP, AXIS provide tools for automatically generating SOAP/XML – C/C++ language bindings
- Using Software Adapters
  - Separate module which proxies between legacy application and ESB (Enterprise Service Bus) infrastructure



Implementing a Ground SOA

- Deciding what is and what is not a service
  - Requires domain expertise – art, not a science
    - Coarse-grained versus fine-grained determination
  - Not all functionality should become a service
    - Some functionality remains internal in an infrastructure-aware "framework" focused on a given functional area (e.g. Command & Control)
- Functionality Taxonomy in an SOA

Software Category	Software/Service Category Description
1	In-Line Software Functionality
2	Utility common to a given functionality framework: exposed in a shared library accessible to the framework
3	Service common to a given functionality framework: exposed in a Service Registry accessible to that framework
4	Utility common to the entire ground enterprise
5	Service common to or exposed across the ground enterprise: exposed in an internal Services Registry
6	Service common to or exposed to external partners outside the ground enterprise: exposed in an external Services Registry
A utility represents a common piece of linked-in software used by multiple applications (historic software reuse). A service represents a separately hosted and invoked piece of functionality accessed through a predefined and published API. Primarily this access mechanism will be web-based.	

Implementing a Ground SOA

- Development of Web-Based Services and Applications
  - Generally, coarse-grained services are to be preferred
    - Fine-grained, "chatty" or conversational services can be performance-challenged
    - Maintaining "state" over a multi-exchange conversation requires additional complexity
    - Simple, stateless, and coarse-grained are the easiest to debug and deploy
  - Techniques to help Fine-Grained: e.g. AJAX
    - Asynchronous JavaScript and XML (AJAX) works with the service consumer being a browser-supported display
- Mission-Based Applications from Simple Services
  - Simple, stateless services, in themselves are often not sufficient to accomplish realistic mission needs
  - Orchestration of services into useful mission applications accomplished via tools utilizing Business Process Execution Language (BPEL)

Implementing a Ground SOA

- **Security**
  - Critical component of any enterprise level or cross-enterprise web service
  - Existing transport level security mechanisms
    - Firewalls, PKI, SSL provide for user authentication and point-to-point security between web-browser and web service: data is protected only while in transit
    - Web services, with potential for intermediaries demand end-to-end security
  - Needed: Message level security for SOAP/XML documents
    - Signing and encrypting an XML document or portions of a document
    - SOAP message filtering based upon content examination: XML Firewall/Guard
  - Standards
    - WS-Security: OASIS standard for SOAP extensions to provide end-to-end security framework
    - SAML 2.0: OASIS standard for exchanging tokens in a vendor neutral format for authentication and authorization
    - XML Encryption: W3C standard to allow encryptions of portions of an XML document
    - XML Signature: W3C standard to digitally sign portions of an XML document
  - XML Firewalls
    - Content level examination; e.g. DataPower, Reactivity, etc.

Implementing a Ground SOA

- **Performance Considerations**
  - XML tagging greatly expands message size
    - 12X expansion factor experienced, although this can vary depending upon the complexity of the XSD associated with the payload content
    - Impacts performance via message parsing time and transport time
    - Vendors addressing via XML compression and XML Gateways (e.g. Reactivity, DataPower)
    - Need to prototype and benchmark critical timelines
- **Handling Large Data Objects**
  - Web Services are not ideal for sending/distributing large binary files
  - Standards are evolving to help address this shortcoming
    - XML-binary Optimized Packaging (XOP), Message Transmission Optimization Mechanism (MTOM), and Resource Representation SOAP Header Block (RRSHB)
      - W3C Standards
    - Enables SOAP bindings to optimize the transmission or wire format used to transfer a SOAP enveloped message [SOAP 1.2 Standard]

Implementing a Ground SOA

- Handling Heterogeneous Infrastructure
  - Large Benefit of SOA
    - Choose the infrastructure (H/W, OS, S/W, COTS) for a given framework area (Command & Control, Planning & Schedule, Data Processing, Tools & Analysis, etc.) most appropriate to the functional demands in that area
    - Exchange data and communicate between frameworks via infrastructure-neutral services over an ESB
  - Interoperability is the key to data sharing and collaboration
    - Standards-driven interoperability based upon universal adoption of key technologies: WSDL, SOAP, XML, UDDI; not adopting a common infrastructure
    - Standards-driven interoperability is distinct from and superior to "Infrastructure Franchising" which requires a core set of identical infrastructure (S/W, COTS, etc.) in order to interoperate

Implementing a Ground SOA

- Services Governance
  - Use of services within an ground enterprise creates greater interdependencies
  - Interdependencies need to be managed by governance policies
    - Testing and deployment of new services
    - Service Level Agreements (SLA) relative to Quality of Service (QOS)
      - Performance, Availability, Accuracy, etc.
      - Tools for measurement and enforcement SLAs
    - Upgrading or modifying existing services
    - Retirement of obsolete services
  - Increased need for joint testing and validation

Implementing a Ground SOA

- Summary and Conclusions of SOA (Pros/Cons)
  - PROS(+)
    - Standards driven
    - Infrastructure-Neutral
      - Supports ground systems recapitalization efforts
      - Avoids vendor "lock-in"
    - Promotes data sharing and interoperability
    - Cost benefits due to common service reuse
    - Good for encapsulating and exposing legacy functionality
  - CONS(-)
    - Message size increases due to XML expansion and associated performance
    - Ability to handle large binary data files
    - Intense standards competition and evolution
    - Greater interdependencies drives need for extra governance policies