Web Service Oriented Architecture

Jørgen Thelin
Chief Scientist
Cape Clear Software Inc.

Agenda

- Distributed System Types
  - Request / Response
  - Message passing
- Architecture styles
  - Object-oriented architecture
  - Service-oriented architecture
- Choosing Architecture Style and Implementation Technology

Distributed System Types

Distributed Systems Types

- Two main types of distributed software systems:
  - Request / Response type systems
    - Also known as “Call & Return” type systems
  - Message passing type systems

Distributed System Type #1 – Request / Response Systems

- Request / Response type systems are:
  - Call oriented systems
  - Usually synchronous in nature
- Approach:
  - Operations have input parameters and output / return values
- Focus is on:
  - The particular operation to be invoked
  - The set of input values
  - The set of output values
  - The correlation of replies with requests
- No real focus on:
  - How the individual values are marshalled as a unit
  - How the output values are produced from the input values (assuming the correct output is produced!)

Distributed Systems Type #2 – Message Passing Systems

- Message passing type systems are:
  - Data oriented systems
  - Usually asynchronous in nature
- Approach:
  - Messages are constructed and send to a destination
- Focus is on:
  - Constructing the message payload in the correct format
  - How to dispatch the message (transport medium)
  - Where to dispatch the messages to (endpoint)
- No real focus on:
  - What will happen to messages after they are dispatched
  - Whether there will be a corresponding reply message

Web Services

- Web Service technology can support both system types simultaneously
  - Request / response ~= “RPC” style
  - Message passing ~= “Document” style

Architecture Styles

Object-Oriented Architectures - 1

- Involve communicating with
  - A particular object instance
- Specific operations for object lifecycle management
  - E.g. EJB create, EJB remove methods
- Communications are implicitly **stateful**
  - Talking to a particular previously-created object instance

Object-Oriented Architectures - 2

- Use middleware specific protocols for communication
  - For example: IIOP, DCOM or RMI
  - Usually not Internet-friendly protocols
- Usually require **pass-by-reference** facilities
- Marshalling object references generally precludes using different types of software on client-side and server-side

Service-Oriented Architectures - 1

- Involve communicating with
  - A specific application service
    - All messages/requests are sent to the service "endpoint"
- **No** operations for service lifecycle management
- Communications are implicitly **stateless**
  - All requests are sent to the same service endpoint
- SOA are generally more **scalable** due to stateless nature

Service-Oriented Architectures - 2

- Service endpoint decides how to process request
  - Inspects the message data content – either an "envelope" or the actual "payload" itself
- Each service has an **interface description**
  - Completely defines the message and payload formats (for example, a WSDL file)
  - Creates a **loosely-coupled** contract between client and server due to late binding

What Makes a Good [Web] Service?

- John McDowall comments in his weblog:
  - There is a lot of confusion about what is a good service - it is not an object, it is not a component, it is an ***interface*** that provides a ***business service***.
  - A good heuristic is:
    - *If the service cannot be described such that a business person understands the value - go back to the drawing board!*
  - [John McDowall, CTO, Grand Central]
  - http://www.mcdowall.com/webservices/2003_02_11_archive.html#90307530

Choosing Architecture Style and Implementation Technology

Comparison of Architecture Styles

Attribute	Object-oriented	Service-oriented
Granularity	Object instances	Service instances
Main Focus	Marshalling of parameter values	Creation and formatting of request payloads
Request routing	Routed to unique object instance	One endpoint address per service
Invocation style	Stateful	Stateless
Lifecycle management operations	Yes	No
Application interface	Specific to the object / class Description is middleware specific (e.g. IDL)	Specific to the service Description is standard specific (e.g. WSDL)
Payload / data format description	Usually middleware specific (e.g. IDL)	Part of service description (e.g. XML Schema in WSDL)

Choosing Between Architecture Styles

- Object-oriented architectures
  - Involve **tight coupling** between client and server, due to:
    - Object reference semantics
    - Object serialization
    - Early binding to interfaces
  - Usually best for "closed" systems controlled by a single organization
- Service-oriented architectures
  - Involve **loose-coupling** between client and server, due to:
    - Late binding to service interface
    - Full interface and payload descriptions in interface contract
  - Generally the most flexible
    – can support request/response and message passing systems
  - Tend to scale better due to their stateless nature.
  - Usually best for "shared" systems crossing org boundaries

Choosing an Implementation Technology

System Type / Architecture Style	Request / response	Message passing
Object-oriented	CORBA RMI EJB DCOM	
Service-oriented	**Web Services**	JMS / MQSeries **Web Services**

Evolution Stage #1

Evolution Stage #2

Summary and Conclusion

Summary

- Two main system types are:
  - Request / Response
  - Message passing
- Two main architecture styles are:
  - Object-oriented
  - Service-oriented
- The choice of architecture style is an important decision for any software system
- Choice of architecture style can have implications on scalability, re-usability and ease of interconnection with other systems

Conclusion

- A major shift is under way in the industry
  - from object-oriented request-response systems
  - towards service-oriented message-passing systems
- Web Services technology provides an ideal way to achieve this architectural shift
  - Web Services allow a Service-oriented approach supporting both Request/Response and Message Passing systems

Further Info

- CapeScience
  - Web Services developer website
    - Articles on Web Services technology
    - Web Services discussion forum
  - http://www.CapeScience.com/
- Jorgen Thelin's weblog
  - http://www.TheArchitect.co.uk/weblog/