Middleware Interoperability

Dr. Seán Baker, Feb 2004

Middleware Interoperability

The nice thing about middleware is that it integrates applications

Middleware choices: CORBA, J2EE, Tuxedo, MQSeries, ….

The bad thing is that different middleware doesn’t interoperate

This results in “islands of integration”. Ad-hoc techniques can be used to integrate these, but none of these are really satisfactory; and most enterprises are too busy to research how best to do their own integrations.

We don’t believe that this is a passing phenomenon

- Different middleware has different advantages
- Departments may still insist on making their own decisions
- A “super-middleware” is unlikely
  - It’s such a big technical area, that a “super-middleware” would be unwieldy
  - Some of the properties are mutually exclusive
- The cost of changing to new technology is high
  - And you may be half way there when the next new thing arrives

More than one middleware: why?

Here are some of the reasons:

- Historical reasons in the enterprise
  - Middleware products have been on sale for many years
- Sophisticated users:
  - Select middleware that best matches each integration need
    - Synchronous request-reply (possibly RPC-based API)
    - Asynchronous messages (possibly fixed API)
    - Pub-sub
    - EAI (messages, transformation, routing)
    - QoS trade-offs: high-throughput, logging, transactions, security, …
- Each department/division of a large enterprise may make its own decisions
- Some of the applications that the enterprise has purchased will have introduced new middleware

Service Oriented Architecture

Let’s start with an “ideal solution” to integrate the islands

Instead of ad-hoc integration:

SOA is a set of principles for building systems. It helps you to master complexity.

...let's assume that all applications provide services that can be used by all others (that have security clearance) – even across the “islands of middleware”:
Services
- Each service has a well-defined interface
  - Listing the operations it provides (or view this as the set of messages it accepts and sends in response)
- SOA works well at the business level
  - The services are high-level entities meaningful to the business
- It also offers much to the technical level
  - The services are meaningful to the implementer (objects, components)
  - A business level service can be implemented by multiple objects/components, or by a non-OO approach
- SOA is not a new idea, but it's very much in vogue today
  - Analysts recommend that companies move towards it

SOA – zoom in on one interaction

SOA and middleware
- SOA can be achieved with many types of middleware
  - But some provide direct support
    - The middleware works in terms of services
  - Any MoM has the basics in place for SOA.
  - CORBA and EJBs are closer still
    - They provide native interface definition languages
    - EAI could struggle.
  - Web Services offers direct support
    - And maybe it's good that we have a SOA-middleware that is aimed at a higher level than CORBA and EJBs
    - This may help to keep the business and implementation levels separate for the designers and implementers

The SOA bus
- This is a complex piece of software:
  - Quality of Service
    - Security
    - Transactions
    - Routing
    - Logging
    - SNMP message generation
    - Failover
    - And of course: protocols, message formats, ...
  - Which brings us back to reality. How can a SOA bus integrate the software islands?
  - SOA is the principle that we want to aim for, but it's not a solution until we can solve the "islands of integration" problem.
  - Put another way: we want an enterprise-wide SOA.

To get an enterprise-wide SOA, we need to have middleware interoperability
- But middleware doesn’t interoperate
- Even the “service” interfaces are very different
  - A service has an interface, and this is the key to hiding details such as the operating system, programming language, network addresses, etc
  - But this interface doesn’t hide/abstract the middleware itself
  - Some examples…

Middleware and Web services
We need ESB that can work across different middleware products and standards,

Artix builds an extensible ESB from underlying middleware
More details of using a middleware switch

- Recall that WSDL has logical and physical parts

<table>
<thead>
<tr>
<th>Logical Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>PortType</td>
</tr>
<tr>
<td>Operation</td>
</tr>
<tr>
<td>Message</td>
</tr>
<tr>
<td>Part</td>
</tr>
<tr>
<td>Soap Body</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
</tr>
<tr>
<td>Port</td>
</tr>
<tr>
<td>Binding</td>
</tr>
<tr>
<td>Part</td>
</tr>
</tbody>
</table>

The use of WSDL means that the core of the switch has to deal with only one way of defining interfaces. Plug-ins allow it to understand WSDL specialized to different middlewares (MQ, Tuxedo, CORBA, J2EE, TIBCO/Rendezvous, …), and other plug-ins allow it to send/receive messages using these middlewares.