



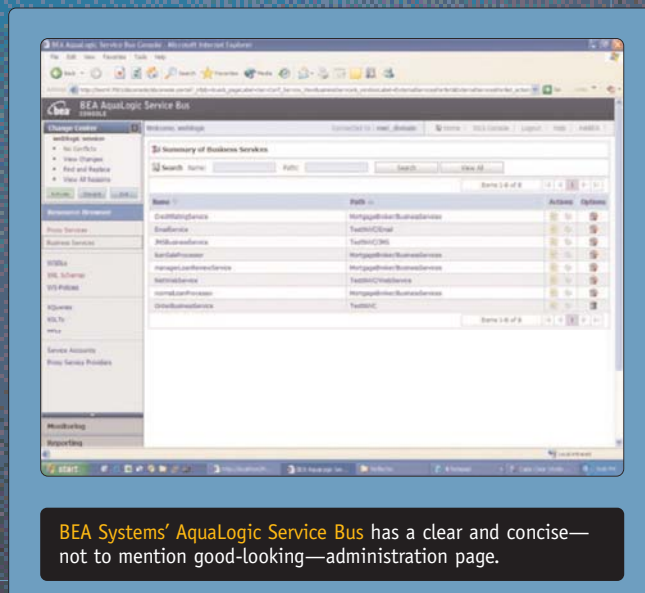
MAKE WAY FOR THE ESB

We test 8 ESB suites to see which one offers the smoothest ride. BEA's AquaLogic Service Bus cruises into first place thanks to its powerful Web-based orchestration, monitoring and reporting

BY LORI MACVITTIE

It's no wonder so many enterprises are looking to build out their SOAs—the agility a service-oriented architecture promises, within both IT and the business, is enticing and financially rewarding. The reuse of legacy assets, if not necessarily code, is a benefit long promised by EAI (enterprise application integration) but rarely realized. Granted, agility is one of those marketing words we try to avoid. But in this case it's warranted to describe the ability to modify business processes without code, as well as to describe the capability to modify the inner workings of a business service without affecting the clients and services that rely on the modified service. In a truly agile system, for example, a customer-lookup service can be changed to use an Oracle database from a COBOL copybook, and the client should be none the wiser.

Problem is, once you start digging, you'll find SOAs are a lot like your skeletal system: There are



BEA Systems' AquaLogic Service Bus has a clear and concise—not to mention good-looking—administration page.

many moving parts, some of them essential and others—like the coccyx!—you may be able to do without. And like your skeletal system with its bones and ligaments, SOAs aren't just a single product. It's an architectural concept that must be implemented with many types of products, each designed to fulfill a purpose within the overall architecture.

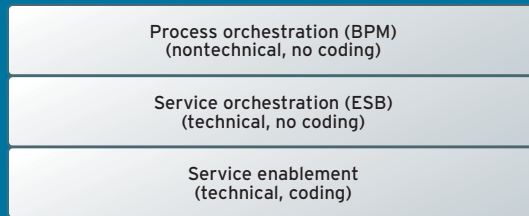
To build a solid SOA infrastructure, you need an enterprise service bus. If the SOA is your skeletal system, the ESB is your backbone. It's the application infrastructure that joins disparate services and ensures availability of those services. It enables business services to be turned into business processes by BPM (business process management) suites. It's connected to everything else, and it protects the information flowing from the SOA brain (the registry/repository) to everything else. It's the middle tier of the SOA infrastructure stack, the glue that binds process orchestration to services. An ESB isn't super glue, though—services aren't tightly bound together by the ESB because that would thwart the SOA's promise of agility. An ESB should link services lightly to a business service that can change rapidly with the needs of the business and the ever-evolving technological landscape of IT.

Stack It Up, I'll Take It

The SOA infrastructure stack comprises three tiers:

» **The service-enablement layer.** Applications are fitted, or retrofitted, with Web service interfaces. Con-

SOA Infrastructure Stack



ventional EAI offerings will fit the bill here, but we prefer products from companies like IBM, Iona and Software AG that specialize in exposing legacy technologies as Web services for use in more modern architectures. Playing on this layer requires both technical competency and coding chops.

» **The service-orchestration layer.** The ESB is an essential piece of this layer, providing availability and routing of services exposed at the service-enablement layer and orchestrating individual services into business services for use at the next layer. The registry/repository is also an essential player in this layer and will be required if SOA nirvana is ever to be reached. This layer requires technical competency but should not involve coding skills.

» **The process-orchestration layer.** BPM and BAM (business activity monitoring) take their place in the SOA stack at this abstract layer, providing support for business processes built on the business services integrated in the service orchestration layer. This is the

REAL-WORLD
LABS®

REPORT CARD

ESB Suites

| | BEA Systems AquaLogic Service Bus 2.1 | Oracle SOA Suite | TIBCO Software Business- Works 5.3 | Fiorano Software SOA 2006 Platform 3.7 | Cape Clear Software ESB 6.5 | IBM WebSphere Message Broker 6.0 | Software AG Enterprise Service Integrator 2.1 | Sonic Software SOA Suite 6.1 |
|---|--|---------------------|---|---|-----------------------------------|---|---|---------------------------------------|
| CORE BUS FEATURES | | | | | | | | |
| Routing (10%) | 5 | 4 | 4 | 3 | 3 | 3 | 3 | 4 |
| Transformations/mapping (15%) | 4 | 5 | 4 | 5 | 4 | 3 | 2 | 3 |
| Orchestration (20%) | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 2 |
| Protocol support (15%) | 4 | 4 | 4 | 4 | 3 | 4 | 3 | 3 |
| Management/configuration (10%) | 5 | 5 | 4 | 3 | 3 | 4 | 3 | 2 |
| INTEGRATION | | | | | | | | |
| Adapter support (5%) | 2 | 4 | 5 | 4 | 2 | 5 | 3 | 3 |
| Web services (15%) | 5 | 4 | 4 | 3 | 5 | 3 | 2 | 3 |
| Management/configuration (5%) | 4 | 4 | 3 | 3 | 3 | 4 | 3 | 2 |
| PRICE (5%) | 5 | 3 | 4 | 2 | 3 | 2 | 3 | 4 |
| TOTAL SCORE (100%) | 4.30 | 4.20 | 4.00 | 3.65 | 3.60 | 3.15 | 2.90 | 2.80 |
| A≥4.3, B≥3.5, C≥2.5, D≥1.5, F<1.5 A-C GRADES INCLUDE + OR - IN THEIR RANGES. TOTAL SCORES AND WEIGHTED SCORES ARE BASED ON A SCALE OF 0-5. | A- | B+ | B+ | B- | B- | C+ | C | C |

CORE BUS FEATURES grades are based on the product's support of standard protocols, transformation and mapping capabilities, and overall orchestration mechanisms.

INTEGRATION scores are based on ability to integrate Web services (including security considerations) and support for typical enterprise integration needs (including RDBMSs and enterprise applications).

PRICE scores are based on NWC Inc.'s testing scenario.

Customize the results of this report card using the Interactive Report Card®, a Java applet, at www.nwc.com.

layer where business analysts can interact with services and where, one day, they'll be able to compose SOBAs (service-oriented business applications) by using a drag-and-drop mechanism to orchestrate business services in a way that best suits their needs. This layer should support nontechnical business-oriented users.

Three layers, three sets of products. Some vendors, including Oracle and Sonic Software, claim to combine all three layers into a single SOA suite. But make no mistake: Even if all the requisite technology is packed into a single box, there are still three disparate layers in the stack, and your SOA should reflect that to maintain a loosely coupled architecture and realize the benefits of a well-constructed SOA.

The ESB should orchestrate services into business services—small orchestrations that bind lower-layer integrations into a business purpose. CustomerInfo-Retrieval, for example, is a business service, but under the covers it may comprise several lower-layer services that communicate with multiple systems to retrieve the data required to present higher layer process-management systems or applications with all the right data.

Although you could shoehorn some ESB products into taking on the role of orchestrating business processes, most ESB suites are missing components—such as process management, workflow and BAM—necessary to manage and develop true BPM systems. An ESB should focus on providing business services to higher-layer applications and on ensuring the availability of those services.

Come Ride Our Bus

To evaluate current ESB offerings we invited a lengthy list of vendors to make the trek to our to Green Bay, Wis., business applications lab and help NWC Inc. begin its journey toward SOA nirvana. BEA Systems, Cape Clear Software, Fiorano Software, IBM, Oracle, Sonic Software, Software AG and TIBCO agreed.

We grew tired and sore from all the bumps and turns during our ride on each of the eight ESB implementations we tested. All eight entries drove us from Point A to Point B, and all hit a few potholes on the way. Overall the products from BEA, Oracle and TIBCO gave us the smoothest ride, while those from Cape Clear, IBM and Sonic Software actually made us break out the Mountain Dew and code for a while. Silly us,

we didn't stipulate "no coding" in our invitation.

WebMethods and Sun responded positively but must have taken a left turn at Albuquerque; neither sent us a product. PolarLake fell victim to internal miscommunications and had to decline. Microsoft never replied at all, not surprising given its current marketing hype of having something "better than an ESB" and our requirement to support SOAP over JMS (Java Message Service); support of anything Java is necessarily absent from Microsoft products.

Iona joined us in the lab, but after we tested version 3.0.3 of its Artix product we decided it didn't meet our requirements. The company was in the middle of a major release cycle, with version 4 due only a few weeks after our testing window closed. Version 4 would have met our requirements, and we hope to get a chance to review it down the road.

We evaluated each product primarily on its core ESB features, examining routing capabilities, transformations and mapping, protocol support and orchestration

We evaluated each ESB suite by attempting to orchestrate a service to handle a purchase order.

mechanisms, as well as conventional management facilities. We also peeked at how well each ESB performs conventional integration functions for common enterprise tasks, such as interfacing with enterprise applications (like those from PeopleSoft, SAP and Siebel) and database interactions. Although such integrations would be achieved over Web services in the SOA purist's world, some situations and performance considerations require that typical integrations such as these be performed at the service-orchestration layer.

With that in mind, we evaluated each product by attempting to orchestrate a service to handle a purchase order. The service had to interact with NWC Inc.'s Oracle 9i database, be able to publish to a JMS queue, act as a Web services consumer and send a simple e-mail notification message to the customer. The ESB was required to transform data when publishing to the queue as well as when creating the appropriate e-mail notification. We used content-based routing to

ESB Transport Methods

| | Server | Bus transport | Metadata |
|--|-------------------|-------------------|----------|
| BEA Systems AquaLogic Service Bus 2.1 | J2EE ¹ | JMS | Other |
| Cape Clear Software ESB 6.5 | J2EE | HTTP | BPPEL |
| Fiorano Software SOA 2006 Platform 3.7 | J2EE | JMS | Other |
| IBM WebSphere Message Broker 6.0 | OS Native | User-defined | Other |
| Oracle SOA Suite | J2EE | HTTP ² | BPPEL |
| Software AG Enterprise Service Integrator 2.1 | J2EE | JMS | Other |
| Sonic Software SOA Suite 6.1 | J2EE1 | Sonic MQ | Other |
| TIBCO Software BusinessWorks 5.3 | OS Native | TIBCO EMS | Other |

¹Standalone J2EE product ²Uses in-memory transfer when in same container

determine whether a message needed to be published to the JMS queue used by manufacturing processes. For more detail on our NWC Inc. scenario, see “How We Tested ESB Suites,” below.

Under the Hood

Architectural differences—most specifically those involving routing and management capabilities—among the products we tested affected much of our evaluation. There is no standard architecture, no stan-

dard transport mechanism on the bus, no standards to address how Web services are implemented ... all of these decisions are left to the vendor, and no two vendors appear to be of like mind (see “ESB Transport Methods,” page 44).

Although Sonic Software’s SOA Suite 6.1 and BEA’s AquaLogic Service Bus 2.1 are both, technically, J2EE applications, neither is deployable on enterprise service platforms other than the ones with which they ship. Sonic’s suite requires only a JDK and ships by

HOW WE TESTED ESB SUITES

We jumped on each ESB by installing it on a dual Xeon 2.6-GHz Dell 2650 with 1 GB of RAM, running Windows 2003. Where out-of-the-box integration with NWC Inc.’s Oracle 9i database was not provided, we used the appropriate JAR (Java Archive) files from Oracle. Where coding was required, we used Java, even if support was offered for multiple languages. All SOAP messages were required to be DOC/LIT encoded, as per WS-I Basic Profile interoperability guidelines.

We attempted to orchestrate a business service to perform order fulfillment for NWC Inc.’s flourishing widget enterprise. The scenario was crafted to let us fairly evaluate core ESB features, such as transformation, protocol support, orchestration and content-routing capabilities. We used OpenJMS as an exercise in configuration and management of a JMS queue because it is not generally supported out of the box. We, therefore, could evaluate the mechanisms provided by each ESB for configuring foreign JMS servers,

which, like the JDBC requirement, let us better understand the underlying architecture of each product.

While implementing our scenario we became familiar with each product’s design-time interface and deployment mechanisms. Where BPEL 1.1 support was offered we exported and imported among vendors to evaluate compatibility (see “What’s Up With BPEL?,” page 58, for our results). We also examined the integration features offered by each vendor to meet common enterprise needs, such as database and application (SAP, PeopleSoft, Siebel) integration. We found a great deal of differentiation in this area, lending credence to the perception that the definition of an ESB is far from agreed upon.

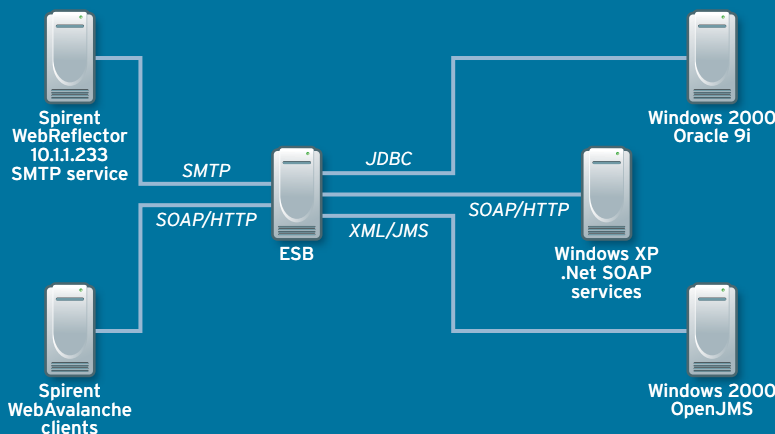
Our implementation required each ESB to be both a consumer and producer of Web services, and we used the development of services and inclusion of external services as a mechanism to evaluate integration and use of a registry/repository. Although most vendors whose prod-

ucts we tested take dual advantage of external UDDI-based registries and embedding of a registry within the bus (usually Systinet, but sometimes the vendor’s own implementation), we discovered that the design and run-time role of the registry as it pertains to the bus is still somewhat in flux.



NWC Inc. implementation scenario logic

1. Accept a SOAP-over-HTTP purchase order from the client
2. Call out to an Oracle 9i database via JDBC to determine the quantity in stock of the widget being purchased
3. Based on the quantity of the widget, perform a content-based route
 - a. If the widget is in stock
 - i. Call out using SOAP over HTTP to a shipping service
 - ii. Send an e-mail to the customer using SMTP
 - b. If the widget is not in stock
 - i. Put an XML message on a JMS queue that notifies manufacturing
 - ii. Send an e-mail to the customer using SMTP



All NETWORK COMPUTING product reviews are conducted by current or former IT professionals in our own Real-World Labs®, according to our own test criteria. Vendor involvement is limited to assistance in configuration and troubleshooting. NETWORK COMPUTING schedules reviews based solely on our editorial judgment of reader needs, and we conduct tests and publish results without vendor influence.

default with IBM's JDK 1.4. BEA's ESB automatically installs into a lightweight WebLogic Server (WLS) 9.1 container, and the only time we needed to manage the WLS instance was to add a connection to get to our remote OpenJMS queue.

Oracle's SOA Suite and Cape Clear's ESB 6.5 requirements were similarly light on the enterprise service platform administration front, but both can take advantage of existing servers, supporting deployment on several J2EE containers including BEA WebLogic, IBM WebSphere and OracleAS.

What's interesting about products that have grown into ESBs from the conventional EAI area, such as those from TIBCO and Fiorano, is the requirement for an external service to provide Web services support. Although the products from BEA and Oracle also use external containers for Web services, both have integrated the process so well that it is transparent to the administrator and developer. In contrast, Fiorano's SOA 2006 Platform 3.7 and TIBCO's BusinessWorks 5.3 make it painfully obvious that the Web service container is an adjunct to the core ESB. Fiorano's container is Tomcat/Axis, and services with a Web service endpoint must be specifically deployed as such, separate from the orchestrated service. We also had to download and install Sun's JDK 1.5 to get Tomcat up and running. If it's a requirement, the JDK should be distributed with the product. TIBCO's solution to Web services is similar to Fiorano's, though we found its deployment process a bit more fluid; it didn't require as much in the way of installing external products.

Software AG's Enterprise Service Integrator employs JBoss, though it can be deployed in other enterprise service platforms, and its Web services support is in line with TIBCO's implementation, with one exception: In Enterprise Service Integrator we had to go to the Web administration console and define a "portal" (an entry point based on protocol, like SOAP or JMS) that is tied to a specific

sequence (orchestration) configuration. As with Sonic's product, the line between development and administration isn't quite clear, and there's no definitive demarcation of responsibility for configuration between administrators and developers.

IBM's WebSphere Message Broker 6.0 was unique in that we had only to define a URI within the orchestration for its message broker to start listening for and handling Web services requests. IBM does not distinguish a SOAP listener from an HTTP listener, choosing instead to determine whether the request coming in on an HTTP port is SOAP or plain-old XML as part of the configuration of the

Lack of FTP support may be problematic: FTP is still the main transport for many transactions.

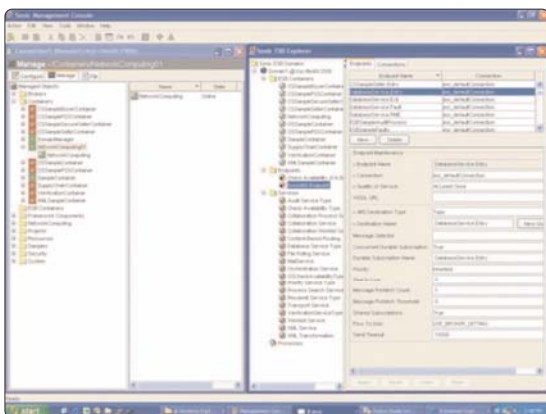
orchestration. This was an interesting approach that let us define an orchestration to process a single SOAP message using both MQ and HTTP with little effort.

After IBM, configuring Sonic's SOA Suite 6.1 was like stepping into an alternate universe. Everything is based on JMS, with endpoints and exit points required for every service, including Web services (see screen at left). Endpoints are configured within the fat client-management console, and then set as properties on steps within an orchestration in the design-time environment. Although it wasn't too difficult to set up a process to be accessible over Web services, it was a bit daunting to force Web services into a JMS endpoint-oriented configuration paradigm.

The SOAP Bone's Connected ...

Protocol support is a primary consideration when choosing an ESB. We found much differentiation among products in terms of inbound protocol support. Although every product supported HTTP and JMS, the similarities ended there. We could not, for example, easily implement inbound FTP or SMTP requests with every product. This lack of FTP support may be problematic, especially in terms of partner interactions, because FTP is still the main transport for a variety of business transactions.

Even more disconcerting was the lack of support for some fairly common protocols within an orchestration. Although JMS and HTTP were consistently supported, SMTP was frequently missing from the list of communication mechanisms (IBM, Cape Clear) as was support for simple queries to an RDBMS (Cape Clear, BEA, IBM). In all three of the latter cases, we had to write code to access our Oracle 9i database and expose the code as services before we could include them in our orchestration.



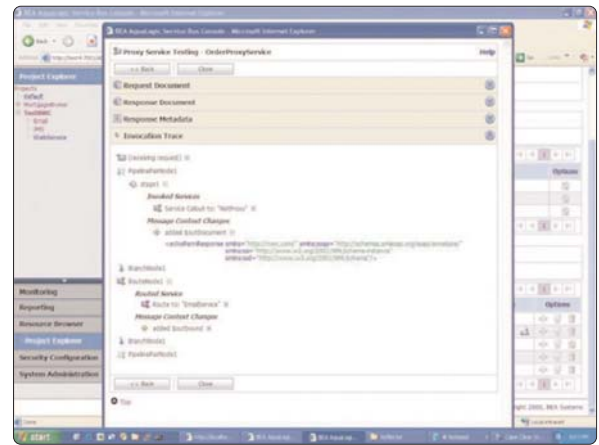
Sonic SOA Suite's JMS-centric administration can make managing Web services confusing if you're not a Java expert.

IBM's solution to support RDBMSs includes its proprietary compute nodes that require coding in Java or ESQL (Extended SQL), which is very BASIC-like in nature. The only other product that required us to write code was Sonic's, which needs JavaScript to do any type of content-based routing within an orchestration. We preferred the XPath-based methods of the other products we tested, even those that required us to know more XPath than we'd like.

Sonic has a wide variety of protocol support, provided you can translate protocol requirements into a JMS world. All protocols must be turned into services—not unusual in the service orchestration sphere—but all protocol support also must be JMS-like entry and exit points, which was confusing at times and difficult to configure.

This varying support of protocols from within a service orchestration is at the heart of a controversy over just what constitutes an ESB. One camp asserts that ESB is the natural evolution of EAI and therefore must be able to integrate more than just Web services, which means supporting more than just SOAP over JMS and HTTP. Others argue that an ESB is a new creature, and while it serves a purpose similar to integration, it is at a higher layer in the SOA stack than pure integration technologies.

We're going to meet somewhere in the middle. Although we agree integration of legacy technology (CICS, COBOL, CORBA) should be handled at the service-enablement layer, some integration is so commoditized that it should be handled at the service-orchestration layer. Database access, for example, shouldn't require coding—aside from the SQL necessary to craft a query—and therefore should not require the overhead of being service-enabled in addition to the already performance-penalizing JDBC connection. Nor should pervasive enterprise applications that are generally integrated with application-specific database connectors or specialized API calls, such as PeopleSoft, SAP or Siebel, be placed at the service-enablement layer. Such connectivity is, or is rapidly becoming, commoditized and should be easily accommodated by enterprise applications whose purpose it is to communicate and orchestrate their services. Our grading of adapter support within the "integration" category reflects this view. Although it wasn't a high



BEA's AquaLogic Service Bus tester provides a wealth of easily assembled information at any point in the process.

percentage of the products' grades—which reflects the small number of adapters we expect to see supported at this layer—it is part of our criteria because we believe you expect the same.

Draw a Map

Core to the function of ESB are transformation and mapping. Data arriving in one format needs to be transformed before being sent to other systems, necessitating the mapping of fields within one document to another. The mechanisms to achieve this varied wildly, from Fiorano's most elegant visual mapping tool to IBM's multistep process.

We preferred Fiorano's and Oracle's visual tools for mapping data—being able to drag and drop fields and XSLT functions to construct both simple and complex transformations is a big plus (see Fiorano screen, page 55). BEA's drag-and-drop interface was good, but it entailed an understanding of XQuery/XPath that should not be required at the service-orchestration layer. TIBCO's BusinessWorks, like most products, relies on XPath and, like BEA AquaLogic, it constructed the XPath for us. But unlike AquaLogic, BusinessWorks doesn't necessitate an understanding of XPath to implement your scenario. Much better.

IBM's setup required us to import the appropriate schemas, create a message map between the two—

XML/XSLT Support

| | XML parser | XSLT engine | Pluggable |
|--|-------------|-------------|-----------|
| BEA Systems AquaLogic Service Bus 2.1 | Proprietary | Proprietary | Y |
| Cape Clear Software ESB 6.5 | Proprietary | Proprietary | N |
| Fiorano Software SOA 2006 Platform 3.7 | Xerces | Xalan | Y |
| IBM WebSphere Message Broker 6.0 | XML4C | Xalan | N |
| Oracle SOA Suite | Xerces | Xalan | Y |
| Software AG Enterprise Service Integrator 2.1 | Xerces | Xalan | N |
| Sonic Software SOA Suite 6.1 | Xerces | Saxon | N |
| TIBCO Software BusinessWorks 5.3 | Proprietary | Proprietary | N |

Y=Yes, N=No

using a visual mechanism—and then generate an XSLT from the map that we could use in our orchestration. Although the process was seamless, we were a bit put off. IBM has two formats for data, the protocol and the wire. Wire format is how data is formatted as it travels through the bus. Protocol format is how it's parsed when it comes in and "rendered" when it goes out. We found IBM's flexibility in this regard impressive, and its ability to distinguish between wire format and protocol format is useful, but we'd like to see something less complex for com-

mon scenarios, such as mapping XML to XML.

Software AG has further to go. With Service Integrator, we had to construct our XSLT from scratch, though it did provide an XPath expression builder that was accessible from most places XPath was required, but not all. More than once a copy-and-paste approach was necessary to provide the expected XPath expression, something we'd like to see rectified in future releases. Even so, Service Integrator is light-years ahead of Sonic's Stylus Studio, which required us to write XPath from scratch or attempt a copy-and-

ESB SUITE VENDORS AT A GLANCE

PARTICIPATING COMPANIES

| Company name | Year founded | Service name | Year launched | Market capitalization as of Feb. 10 | Other products |
|--|--------------|-----------------------------------|---------------|-------------------------------------|--|
| BEA SYSTEMS (BEAS) www.bea.com | 1995 | BEA AquaLogic Service Bus 2.1 | 2005 | \$4,720,000 | WebLogic, Tuxedo |
| CAPE CLEAR SOFTWARE (private) www.capeclear.com | 1999 | Cape Clear ESB 6.5 | 2000 | N/A | N/A |
| FIORANO SOFTWARE (private) www.fiorano.com | 1995 | Fiorano SOA 2006 Platform 3.7 | 2005 | N/A | MQ 2006 JMS server, ESB 2006 Enterprise Service Bus, BPEL 2006, SOA 2006 Platform |
| IBM (IBM) www.ibm.com | 1911 | IBM WebSphere Message Broker 6.0 | 1999 | \$128,460,000 | WebSphere ESB |
| ORACLE (ORCL) www.oracle.com/soa | 1977 | Oracle SOA Suite | 2004 | \$65,510,000 | Components of Oracle SOA Suite: BPEL Process Manager, Business Activity Monitoring, Web Services Manager, Enterprise Service Bus, Business Rules, JDeveloper |
| SOFTWARE AG www.softwareagusa.com | 1969 | Enterprise Service Integrator 2.1 | 2004 | N/A | CentraSite, Business Process Manager, Information Integrator, Application Composer, Legacy Integrator, Tamino, Adabas, Natural, ApplinX, EntireX |
| SONIC SOFTWARE (unit of Progress Software) (PRGS) www.sonicsoftware.com | 2001 | Sonic SOA Suite 6.1 | 2002 | \$1,120,000 | Actional Looking Glass 6.0, Actional SOAPStation 6.0, SonicMQ 6.1 |
| TIBCO SOFTWARE (TIBX) www.tibco.com | 1985 | TIBCO BusinessWorks 5.3 | 2001 | \$1,780,000 | BusinessFactor, BusinessEvents, PortalBuilder, General Interface |

NONPARTICIPATING COMPANIES

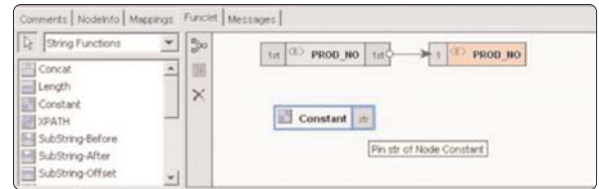
| Company name | Year founded | Service name | Year launched |
|---|--------------|--|---------------|
| IONA TECHNOLOGIES (IONA) www.iona.com | 1991 | Artix 3.0 | 2003 |
| MICROSOFT (MSFT) www.microsoft.com | 1975 | BizTalk Server 2004 | Undisclosed |
| POLARLAKE (private) www.polarlake.com | 2000 | PolarLake Integration Suite 5.0 | 2001 |
| SUN MICROSYSTEMS (SUNW) www.sun.com | 1982 | Sun Java ESB Suite | 2006 |
| WEBMETHODS (WEBM) www.webmethods.com | 1996 | webMethods Enterprise Service Platform 6.5 | 2001 |

Note: We invited 13 vendors to participate in our tests of ESB suites. The companies that declined or did not fit our criteria are listed here as a service to our readers.

paste routine from within a hierarchical-tree-structure view of the XML document we were working on. Mapping the “PROD_NO” value from the inbound XML document to the database parameter, for example, required constructing an XPath query and then adding all appropriate name spaces to the step in the orchestration.

Direct Us, Maestro

Orchestration and service management are at the heart of ESB management. Not only do developers



Fiorano's visual mapping tool makes transformations painless.

need to orchestrate services, they also need to debug and test, and administrators then need to manage those services. Orchestration was not an issue for any of the products, though we were frustrated by IBM's complex modeling requirements. But debugging and testing the services we orchestrated was a mixed bag.

We were thrilled with the capabilities included in many of the products for debugging service orchestrations. BEA's testing tool offers a wealth of options as well as a detailed trace of each step in the orchestration and the data, broken down by headers and payload, right from the browser. Oracle's test tool is also browser-based, and it provides even more details than BEA's, including execution times on a per-orchestration-step basis. TIBCO BusinessWorks' debugging facilities are excellent, and we like its conditional breakpoint functionality, something we didn't see in other products.

Software AG includes a debugger within its XML Mediator Studio that let us set breakpoints and even modify values within documents as they moved through the orchestration. Software AG also provides a testing panel to execute Web services—including a nifty checkbox for WS-I compliance testing—but it still isn't as impressive as the test tool offered by BEA. Fiorano's test capabilities let us send and receive messages, but we couldn't view the steps in between. Hey, that's better than nothing, which is what we got with most products.

Although Cape Clear Orchestration manager also offers a view of execution times on a per-step basis, it doesn't provide a mechanism for viewing the actual message being passed to each step, something BEA and Oracle provide. Note that these testing tools and debuggers actually stopped the messages at each breakpoint we set, in real time, allowing for real-time analysis and modification (see BEA's AquaLogic screen, page 53).

TIBCO BusinessWorks provides details on a number of executions and execution times, but it did so on a process-by-process basis, which meant that we had to drill in and out of processes to find out the number of times our JDBC query had executed, and what the average execution time had been. Even though the orchestration containing the JDBC query is called from within another process, there are no direct drill-down capabilities in the user interface to get us from an outer orchestration to an inner orches-

| Key customers | News |
|---|---|
| America Online, City of Chicago, Covad Communications | Preliminary Q4 2005 revenue is expected to reach \$340 million |
| Alliance Trusts, AT&T, British Sky Broadcasting, JP Morgan Chase, Lockheed Martin, Northrop Grumman | Announced 75 new customers and 100% revenue growth in 2005 |
| Norwegian Cruise Line, The Sports Authority, Quicken Loans, North California Power Agency | Reported \$5 million in revenue for fiscal year 2005 |
| The Bekins Co., Merrill Lynch | Reported Q4 2005 revenue of \$24.4 billion, a 12% decrease from Q4 2004 |
| Undisclosed | Completed acquisition of Siebel Systems |
| American Fidelity, Calif. Public Employees Retirement System (CalPERS), Gov't. of Switzerland, Nissan, SIRVA | Reported total revenue of \$438 million, a 6.5% increase |
| ABNA, AltaGas, Ameriquest Mortgage, Axfood, Boehringer Ingelheim, CARQUEST, City of Seattle, Data Interchanged Standards Assoc. (DISA), First Command Financial Planning, Gillette, SBC | Merged with Actional, which was acquired for \$32 million |
| British Petroleum, Con-Way Transportation Services, Delta Airlines, Deutsche Bank, KPN Telecom, Lehman Brothers, Proctor & Gamble, Telecom Italia | Reported Q4 2005 earnings of \$134.4 million, up from \$105 million in Q3 |

Source: Company reports, Yahoo.com

| Reason for declining |
|---|
| New version not available within the review time frame |
| Does not consider ESB a standalone product category |
| Thinks its product is best evaluated on-site, within an appropriate customer environment, rather than as part of a general ESB overview |
| New version not available within the review time frame |
| New version not available within the review time frame |

Source: Company reports, Yahoo.com

tration. We like the attention to detail provided, but would like to see contextual drill-down capabilities within the process details in the future.

Other products tested simply don't provide a mechanism for viewing service-orchestration details, and only Oracle SOA Suite and TIBCO BusinessWorks gave us the option to terminate long-running or errant orchestrations. Cape Clear offers a minor configuration option during deployment that let us specify how the server should react to failed or exceptionally long-running orchestrations—deletion or

attempted recovery—but that was it. We'd like to see more control over service orchestrations at the server level in all products.

Speed Limits

We started this ride on the enterprise service bus with the intention of testing the performance of our scenario using NWC Inc.'s existing infrastructure. By the end of the trip we discovered that the ESB market has generally agreed on the protocols and standards that an ESB must support ... with two exceptions:

ESB Suite Features

| | BEA Systems AquaLogic Service Bus 2.1 | Cape Clear Software ESB 6.5 | Fiorano Software SOA 2006 Platform 3.7 | IBM WebSphere Message Broker 6.0 | Oracle SOA Suite | Software AG Enterprise Service Integrator 2.1 | Sonic Software SOA Suite 6.1 | TIBCO Software BusinessWorks 5.3 |
|---------------------------------------|---------------------------------------|-----------------------------|--|----------------------------------|------------------|---|------------------------------|----------------------------------|
| Platforms | | | | | | | | |
| Windows | Y | Y | Y | Y | Y | Y | Y | Y |
| Linux | Y | Y | Y | Y | Y | Y | Y | Y |
| Solaris | Y | Y | Y | Y | Y | Y | Y | Y |
| AIX | N | Y | Y | Y | Y | Y | N | Y |
| Features | | | | | | | | |
| AXIS Web services | N | Y | Y | N | Y | Y | N | Y |
| XML Native Persistent Store | N | Y | N | N | N | Y | Y | N |
| SOAP test client | Y | Y | N | N | Y | Y | N | N |
| Bus integration points | | | | | | | | |
| Generic JMS | Y | Y | Y | Y | Y | Y | Y | Y |
| SOAP Consumer | Y | Y | Y | Y | Y | Y | Y | Y |
| SOAP Producer | Y | Y | Y | Y | Y | Y | Y | Y |
| Generic JDBC/ODBC | N | N | Y | Y | Y | L | Y | Y |
| Enterprise apps | L | L | Y (SAP), T (iWay) | L | L | L | L | L |
| MQ Series | Y | Y | Y | Y | Y | L | L | Y |
| TIBCO RV | N | Y | Y | T | Y | N | L | Y |
| HTTP/S | Y | Y | Y | Y | Y | Y | Y | Y |
| File | Y | Y | Y | A | Y | Y | Y | Y |
| E-mail (SMTP/POP3) | Y | N | Y | A | Y | Y | A | Y |
| FTP | Y | Y | Y | A | Y | Y | A | Y |
| Oracle | N | N | Y | Y | Y | L | L | Y |
| DB2 | N | N | Y | Y | Y | L | L | Y |
| Sybase | N | N | Y | Y | Y | L | L | Y |
| SQL Server | N | N | Y | Y | Y | L | L | Y |
| Mainframe/CICS | N | N | T (iWay) | Y | L | L | L | L |
| SOA registry/repository (UDDI) | | | | | | | | |
| Design time | Y | Y | Y | N | Y | Y | N | N |
| Run time | Y | Y | Y | N | Y | N | N | N |
| Included in product/suite | E | Y | E | N | E | Y | N | N |
| Standards | | | | | | | | |
| BPEL | N | Y | Y | N | Y | Y, free third-party product | N | N |
| SOAP | Y | Y | Y | Y | Y | Y | Y | Y |
| WS-I Basic Profile | Y | Y | Y | Y | Y | Y | N | Y |
| WS-Security | Y | Y | Y | A | Y | N | N (soon to be released) | Y |
| WSDM | N | N | N | N | Y | N | N | Y |

Y=Yes, N=No, L=Licensed separately, T= Third-party purchase required, A=Available but not included, E=Embedded Systinet

JDBC and SMTP are hit and miss. Many suites support both, but some require the development of compute nodes (IBM) and Java Web services (BEA and Cape Clear). After much hemming and hawing we were forced to remove performance testing as a grading criterion because we were unable to perform a true apples-to-apples comparison.

Those products on which we could performance-test convinced us that, regardless of how the product performs out-of-the-box, there's plenty of tuning to be done. Take a look at JVM (Java Virtual Machine) memory tweaks, the number of connections allowed for HTTP connections and the code implementing a service; all are factors in the overall performance of any given scenario. In addition, the ability to perform in-memory calls between orchestrated services is imperative, especially when the orchestration involves SOAP over HTTP. Removing the overhead of a TCP session when services reside on the same machine is a huge performance benefit, and you should inquire about this capability to ensure it's available.

Another impediment to performance is the execution of complex style-sheet transformations or the use of encryption/decryption as per the WS-I Basic Security Profile. Being able to perform transformations quickly and efficiently will have a direct impact on the overall performance of any ESB. The choice of XML parsers, too, will have an impact, as well as imposing possible limitations on the size of XML documents that can ride the bus. For that purpose we asked each vendor whether the XML parser and XSLT engine provided with the product were pluggable—that is, could they be replaced or pointed at external hardware, such as a DataPower or Forum Systems device. As IBM has recently purchased DataPower, we hope to see support for pluggable XSLT engines in the near future in IBM's offering (see "XML/XSLT Support," page 53).

BEA Drives the Bus

At the end of the ride, BEA System's AquaLogic Service Bus got our Editor's Choice stamp of approval, with Oracle coming in a close second, hindered by a large gap in pricing. Although AquaLogic Service Bus lacks the integration of Oracle's SOA Suite, BEA makes up for that with its awesome Web-based orchestration environment and elegant monitoring and reporting features.

TIBCO's BusinessWorks isn't far behind the leaders, held back primarily because of management and its less-elegant transformation and mapping capabilities. BusinessWorks did come in several heads higher than Fiorano's SOA 2006 Platform and Cape Clear's ESB, which both managed to take spots in our Top 5.

We set pricing grades based on our NWC Inc. scenario: Two processors, running Windows 2003 Server; adapters for Oracle 9i database, SQL Server database, JMS, SOAP consumer, SOAP producer and e-mail; five administrators, 10 developers, 100 users. The total cost to deploy ranged from BEA's low quote of \$40,000, to IBM and Fiorano posting the highest prices, \$170,000 and \$170,995, respectively. In the midrange were TIBCO at \$75,000, Sonic at \$76,250, Oracle and Software AG both at \$100,000, and Cape Clear at \$105,000. **NWC**



FIND THE INDIVIDUAL PRODUCT REVIEWS FOR THIS STORY
AT WWW.NETWORKCOMPUTING.COM/1705/1705F3.HTML.



LORI MACVITTIE is a NETWORK COMPUTING senior technology editor working in our Green Bay, Wis., labs. She has been a software developer, a network administrator and a member of the technical architecture team for a global transportation and logistics organization.

Write to her at Imacvittie@nwc.com. Post a comment or question on this story at www.nwc.com/go/ask.html.

WHAT'S UP WITH BPEL?

Analysts like to shout a lot about BPEL and claim an ESB should support the Business Process Execution Language spec, period. But there's some contention in the industry about BPEL, for several good reasons. First, BPEL 1.1 is merely a specification, not a standard. The second, more compelling, reason to ignore BPEL is that version 2.0 is vastly different from 1.1—and it's version 2.0 that's expected to be adopted as the standard.

Interestingly enough, BPEL 1.1 is often used as the orchestration mechanism for both services and

business processes, even though BPEL lacks the notion of human workflow and focuses solely on machine-to-machine interaction. This makes it particularly useful for orchestrating services, as we noted with Oracle and Cape Clear, but not nearly as useful at the business process orchestration layer.

In addition, testing showed that BPEL as a specification implementation is not yet portable. Even though Fiorano, Oracle and Cape Clear all support BPEL 1.1, none was able to successfully import and use another's BPEL without modifica-

tion. Oracle came closest to being successful—we cut and pasted the source of Fiorano's BPEL into its design-time environment and with a few changes, got it working.

Other attempts were not successful, and we came to the conclusion that, while BPEL 1.1 is just fine for service orchestration, it's not the be-all and end-all of service orchestration. Interoperability among BPEL engines will have to wait until a standard is accepted, and then perhaps we'll agree that BPEL is a definitive requirement to be considered an ESB.

Copyright (c) 2006, CMP Media LLC. Important note: This PDF is provided solely as a reader service. It is not intended for reproduction or public distribution. For more information on obtaining a Reprint, please contact a Reprint Services Rep at 516.562.7026 or visit www.cmpreprints.com/faxback.jhtml

REVIEWS: ESB TECHNOLOGY

By Lori MacVittie

BEA Systems AquaLogic Service Bus 2.1 Several years ago, BEA began to lag behind competitors Oracle

A- and IBM in the integration and enterprise platform space. Its goal was to focus in on SOA, and its strategy was to take that market by storm. AquaLogic Service Bus 2.1 proves that BEA's hiatus from the competitive market was well worth the wait.

AquaLogic is installed within a stripped down, lightweight version of WebLogic Server (WLS) 9.1. It's essentially a stand-alone J2EE product, capable of being installed as a module within any WLS container, but it does not support deployment within other J2EE containers, such as JBoss or IBM WebSphere.

AquaLogic is stateless, using JMS for persistence of state only when necessary. It does not require a metadata repository unless you want to take advantage of its included reporting and monitoring capabilities, which can then be configured to use an existing RDBMS or the default embedded Pointbase database.

Deployment to the bus is instantaneous because the AquaLogic design-time environment is completely Web-based, and it takes advantage of the HTTP session-management capabilities of WLS to keep track of what the designer is doing. The design-time environment is drag-and-drop and worked equally well in both Internet Explorer and Mozilla Firefox 1.4. In fact, AquaLogic is unique in its use of a Web browser as its design-time environment. We easily defined services, which can be published to BEA's embedded Systinet Business Service Registry or to an external registry, and orchestrated those services from within a composite service.

AquaLogic distinguishes business services from proxy services, emphasizing that external clients will always connect to a proxy while internal services may communicate with either. Business services can communicate without requiring the overhead of an HTTP

session. Proxies, by definition, require an HTTP session. That's overhead that can degrade performance—every TCP intermediary adds latency and a point of failure. AquaLogic supports e-mail, HTTP/S, various file formats, JMS, and FTP out of the box. We created an SMTP service to send e-mail and a JMS service to call out to our OpenJMS queue, and included a call-out to a Web service hosted on an external .Net server within NWC Inc.'s infrastructure. The orchestration notation used within AquaLogic is proprietary, but we found the notation intuitive and close enough to BPMN (Business Process Modeling Notation) that users should experience very little trouble orchestrating services with it.

Web service support within AquaLogic is excellent; BEA supports the WS-I Basic Security Profile requirements as well as SAML 2.0 and the Web Services Security Username/Certificate profile. WS-I compliance testing is also a checkbox item when orchestrating Web services, though as with other products supporting this



The screenshot shows the BEA AquaLogic Service Bus console in a Microsoft Internet Explorer browser. The main window displays a 'Summary of Business Services' table with columns for Name, Path, and Actions. The table lists several services such as CreditRatingService, EmailService, and FileUploadService. A sidebar on the left contains navigation options like 'Home Overview', 'New Changes', and 'View All Services'. At the bottom of the screenshot, there is a black callout box with white text.

BEA Systems' AquaLogic Service Bus has a clear and concise—not to mention good-looking—administration page.

particular test, we don't suggest using it in production as it imparts a heavy performance penalty.

After orchestrating the service, we used AquaLogic's Web-based proxy tester to test the resulting proxy service; we dug into the flow and examined the requests and responses as they passed through. The one thing missing from the tester is the execution time of each step in the flow, a feature provided by competitors Tibco and Oracle that we thought gave them a bit of an edge.

Another nit with AquaLogic is its lack of support for integration with external data sources, such as our Oracle9i database. AquaLogic requires that such integration be service-enabled using some other mechanism, such as Web Logic Integration, and doesn't let you create a simple JDBC integration as part of the service orchestration. Cape Clear and IBM also omit this simple integration as part of their offerings, but provided the means to code a solution and include it as a service within their orchestrations. We'd like to see simple database access offered at the service bus level in all products in the future.

Configuration of foreign JMS queues is accomplished with the WLS 9.1 administration console—this was the only time we had to configure anything in WLS 9. Obviously, if database connectivity were coded at the service-enablement layer and deployed as part of the server, the JDBC configuration also would need to be configured within WLS 9 and not AquaLogic. We view this as a simple task, easily accomplished by any experienced enterprise service platform administrator.

Once we completed our orchestration we activated the session we'd created, and a new version of the orchestration was automatically deployed on our server. AquaLogic provides simple versioning and change tracking, and we could roll back changes at any time with the click of a mouse. Obviously, we don't anticipate or suggest that developers create and test orchestrations on a production server, and neither does BEA. To deploy to a production server we exported the orchestration and its requisite dependencies—automatically determined by AquaLogic—to an archive file and then imported it into the production environment, all from within BEA's Web-based design and administrative environment. Note that all the Eclipse-based products integrate with CVS and some external version control systems. Proprietary environments used a variety of "version control" mechanisms, most of them proprietary and not a good choice for large-scale code management.

We also tried out AquaLogic's monitoring and reporting capabilities, which can be configured at the service layer and include the exposure of process-level variables ... a poor man's BAM, if you will. We created a variable to be carried along in our service orchestration and assign a value to it using AquaLogic's drag-and-drop XQuery/XPath editor. By specifying that the variable should be monitored, we could see the value of the variable in every orchestration. This feature can be turned on and off with the click of a button, making it useful for debugging or tracing through a service orchestration.

We were pleasantly surprised by BEA's pricing of AquaLogic—\$20,000 per CPU is more than equitable for the features and functionality offered. Of course, you should factor in that you'll require some way to service-enable your database and enterprise applications, and we're certain BEA knows you will, too.

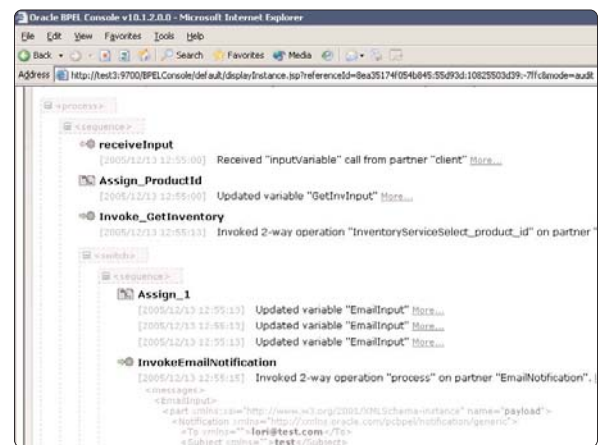
BEA AquaLogic Service Bus 2.1, \$40,000 as tested.
BEA Systems, (800) 817-4BEA. www.bea.com

Oracle SOA Suite Oracle has spent several years acquiring technology, and just as long putting the

B+ pieces together into a cohesive offering—including Collaxa for BPEL, Oblix for WS-Security and WS-Management, and branded Systinet BSR 6.0 for its registry/repository. Toss in a rules engine and the result is SOA Suite, a nearly seamlessly integrated set of tools designed to enable SOA implementations in the enterprise.

The primary component in SOA Suite is Oracle's BPEL PM 10.1.2. SOA Suite is a J2EE offering that can be deployed in a number of containers, including IBM's WebSphere, BEA Systems' WebLogic, JBoss and OracleAS. By default, the suite installs in an OracleAS container with an embedded OracleLite meta-data repository.

SOA Suite differs from most of the other products we tested in that all of its instances of a service orchestration are technically stateless, with session information stored in the repository. Although OracleLite is the default repository, we could (thankfully) configure SOA Suite to use an enterprise-class RDBMS, such as DB2, SQL Server or, of course, Oracle. Instances of SOA Suite do not communicate, so we couldn't designate backup nodes as we could with Fiorano's product, but because all state information is stored in a shared repository, instances should be able to re-create themselves, regardless of which node they were instantiated on. Failover requires that load balancing be configured at the container level, or that you use an external, hardware load balancer such as those from F5 Networks or Citrix.



The audit trail in Oracle SOA Suite's administrative console makes tracing messages a snap.

Failover using BEA AquaLogic is accomplished using WebLogic Server clustering technology, while Cape Clear relies on the clustering capabilities of the J2EE enterprise service platform in which it is deployed, with some routing-based failover available at the ESB layer—provided the container handling the routing isn't the one that fails, of course. We prefer to see failover at the ESB layer, because it knows about services.

Like TIBCO's product, Oracle's allows for profiling individual orchestrations, providing execution times on a per-activity basis. But unlike TIBCO's, Oracle's presentation of the data in SOA Suite's Web-based console was easy to navigate. Unique to SOA Suite is the ability to fully manage in-flight orchestrations and execute multiple versions of the same orchestration. We were particularly pleased with the audit-trail capabilities of Oracle's and BEA's products, though Oracle's audit trail is available for all instances. Still, SOA Suite lacks AquaLogic's granularity of monitoring in this regard, and exposing process-variable data through AquaLogic's monitoring capabilities is far easier than accomplishing the same task with Oracle SOA Suite.

Configuring SOA Suite to communicate with OpenJMS was as expected and is on par with other products in our review. Gaining RDBMS access, as well, was a painless task we accomplished easily. Oracle distributes DataDirect drivers for connecting to most popular RDBMSs, and we were pleased we didn't need to copy any driver files to the system to enable connectivity to NWC Inc.'s Oracle9i database. Given the sheer hell we went through with Sonic Software's SOA Suite to configure simple access to the same database, we greatly appreciated the ease with which this task was accomplished using Oracle's.

We were likewise pleased with Oracle's default in-memory communication between processes of like service types. HTTP-to-HTTP and JMS-to-JMS services in the same container, for example, can communicate without requiring the overhead of setting up the underlying TCP session for an HTTP exchange. This feature was offered by BEA's suite as well—except for SOAP/HTTP services, which always required us to set up the underlying TCP session. Oracle also claims to have an optimized SOAP stack to speed up communication, but we couldn't test performance across all products so we can't confirm this. All communication between services within an SOA Suite orchestration is through SOAP/HTTP or SOAP/JMS, as is the case with the products from BEA and Cape Clear.

SOA Suite's design-time environment is available as a JDeveloper or Eclipse plug-in. By default, it installs JDeveloper and its plug-in. Service orchestration is accomplished using BPEL, and we preferred Oracle's BPEL editor over that from Cape Clear, the only other vendor to offer a BPEL-only modeling environment. Oracle uses the same sort of environment for building transformations that Fiorano employs, but Oracle's is a bit easier to use; we found it less developer-oriented than Fiorano's mapper, which is somewhat surprising as many of Oracle's tools are highly developer-oriented. Although both

are graphical and use a drag-and-drop mechanism for building up expressions, Oracle presents the various functions, like concat, without exposing you to developer-oriented parameterization requirements.

SOA Suite's default XSLT engine is Xalan, with Xerces as its default XML parser, but both use a pluggable model and can be replaced with another parser and engine if so desired. We modeled our scenario for NWC Inc. in short order and quickly deployed through the JDeveloper plug-in, though Apache Ant scripts are also available, and likely preferable in an enterprise setting. The only nit here is that Ant scripts must be created by hand, so your mileage may vary.

We were pleased with the ability to route within an orchestration based on payload as well as headers and process variables. Although most of the products we tested provide this feature, IBM's cannot route messages based on payload and provides only the means to route based on headers. Like the other products in our review, SOA Suite's routing and decisions are based on XPath expressions. While creating the XPath required to route our process we found Oracle's XPath builder a boon; it didn't require the cutting-and-pasting demanded by the products from Sonic Software and Software AG.

Oracle's pricing scheme—cut-and-dried compared with several other products—is based solely on number of licensed CPUs. Existing Oracle Application Server customers will shell out \$20,000 per CPU, while new customers can compute the cost at \$50,000 per CPU.

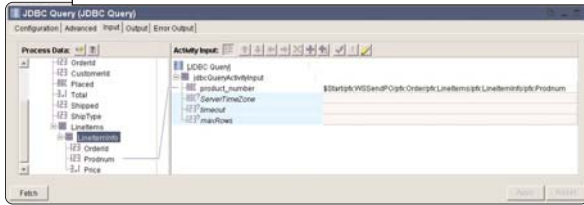
Oracle SOA Suite, \$100,000 as tested. Oracle, (800) ORACLE1, (650) 506-7000. www.oracle.com/soa

TIBCO BusinessWorks 5.3 As with IBM and Sonic Software, messaging is a large part of TIBCO's SOA

message. We didn't find this surprising—TIBCO's Rendezvous, along with IBM's and Sonic's MQ implementations, are well known and widely implemented across a variety of industries. TIBCO, like Sonic and IBM, considers SOA a deployment strategy and an extension of integration rather than a replacement for conventional EAI implementations. Again not surprising, given all three companies' heavy investments in the EAI and messaging markets.

TIBCO's bus backbone, like Sonic's, is JMS-based, taking advantage of TIBCO's EMS product. Unlike Sonic, TIBCO lets the customer choose the underlying transport; we could replace EMS with another messaging backbone, such as SonicMQ or IBM's MQSeries, or go with a completely HTTP-based backbone. Flexibility, thy name is TIBCO.

BusinessWorks requires a runtime agent, as do all TIBCO products, to enable centralized management from its Web administration console. BW Designer, TIBCO's design-time environment is, like most non-Eclipse-based design-time products we tested, moving to Eclipse in a forthcoming release. Like Sonic and BEA, TIBCO uses the notion of "domains" for management,



Mapping data within TIBCO's Business Works 5.3 is easier to do than to read.

grouping BW servers together for administrative purposes. This is different from clustering, which is used for availability and failover. Unique to TIBCO is its support of management over WSDM, specifically using the MUWS (Management Using Web Services) profile.

BusinessWorks is a standalone, OS-native application (meaning it's a compiled app and not interpreted) that uses Apache/Tomcat for its graphical administration environment. The Apache/Tomcat portion of the product is bundled and used expressly for BusinessWorks based on specific version needs. There is no external management or installation required; TIBCO handles it all under the covers. Apache is used to handle Web services endpoints, and it communicates over an internal channel with the BusinessWorks server to pass client requests. Essentially, a servlet peeled off SOAP and policy information, then passed the core message to the TIBCO engine for processing.

Unlike Sonic, Software AG, Oracle and Cape Clear, whose entries all required steps within an orchestrated service to be atomic services, TIBCO takes a more integration-oriented approach and provides myriad options for integrating services, databases and enterprise applications. We easily dragged-and-dropped components into our orchestration. The only odd configuration requirement is that we had to separately define a series of connections—one for JDBC, one for HTTP and one for JMS—and then specify in the component which connection was required for each step. What we really liked about this oddity was that the connection could be reused—unlike other products that tightly coupled an SQL statement to a specific JDBC connection, TIBCO let us reuse the JDBC connection and specify different SQL statements at will.

The requirement to use XPath to map variable data from step to step in an orchestration was eased somewhat by TIBCO's automatic generation of the appropriate XPath expression. But we found the display of configured XPath expressions on a field-by-field basis painful because of the verbosity of XPath expressions and the length of fields within the TIBCO user interface. A minor nit, we grant, but a nit nonetheless.

Once our service was orchestrated, we still had to generate a Web service interface so that our test clients could communicate with the soon-to-be-deployed service. Only the products from Oracle, Cape Clear and BEA automatically generate a Web service endpoint for orchestrated

services; the others, including BusinessWorks, require the specific generation of a Web service. We see the logic behind this decision, but wish that the process was a bit less painful than it tends to be with products that require this step. Unlike the suites from Sonic and Software AG, BusinessWorks generated all the information necessary for the WSDL and gave us the option to modify it before generating the service. We think even the presentation of the data required to generate the appropriate WSDL is a bit much given the WSDL-specific knowledge necessary for the designer to modify the values; we'd prefer to see this information left for more advanced designers. We chose to use the default values supplied by BusinessWorks and successfully generated the service.

Deploying orchestrated services required the creation of an EAR (Enterprise Archive), which was a bit more complicated than the process used by IBM and Cape Clear to accomplish effectively this same task. We had to first create an EAR, then a Process Archive, and then add our OrderFillProcess to the archive. To give TIBCO credit, BusinessWorks automatically determined all dependencies and added them to the archive without our help.

We pushed a button, appropriately labeled "Build Archive," and our EAR file was ready for deployment. TIBCO is unique in its decision to not allow deployment from BusinessWorks Designer. Rather, deployment requires either scripting or manual deployment through its Web console, TIBCO Administrator. We chose the latter and were faced with a simple process of specifying the EAR file to deploy from within TIBCO Administrator, then choosing a few configuration options, including which BusinessWorks Engine to target.

Testing deployed service orchestrations was more difficult because of the absence of built-in tools like those provided by Oracle, Cape Clear and BEA. Even Sonic and Software AG provided JMS client test tools, which at least let us test our orchestrated process before deployment using SOAP/JMS. Testing is possible with BusinessWorks, it just required that we orchestrate a little test service that called the deployed service. This isn't a bad way to go, but we'd at least like to see an option to automatically generate a test service, as all the requisite information is available within BusinessWorks. TIBCO told us it is working toward automating this process in a future release.

TIBCO's pricing was a bit lower than average, offering up a two-CPU starter pack for \$75,000 that included everything we needed to implement NWC Inc.'s scenario. JDBC connectivity is included at no extra charge.

TIBCO BusinessWorks 5.3, \$75,000 as tested.
TIBCO Software, (800) 420-8450, (650) 846-1000. www.tibco.com

Fiorano SOA Platform 2006 3.7 Fiorano SOA Platform 2006 is a J2EE ESB with several moving parts. All



administration is accomplished over Java fat clients, and there are several to choose from depending on your role in the organization.

no's consumer-side support of Web services, though incorporating an external Web service into our orchestration revealed a few problems, such as the lack of user feedback when importing external WSDL files. Fiorano indicates only errors, not successes, and subscribes to the "no news is good news" camp when dealing with WSDL files. Despite the lack of feedback, we easily incorporated the external service required by our scenario.

Deployment of orchestrated services was just a button click away, and we liked Fiorano's ability to synchronize changes in an orchestrated service with processes already running on the server. Fiorano's capabilities in terms of managing services running on our server were more limited than those provided by Oracle, but we did view running processes as well as stop and start individual components at run time. We started the components required by our orchestration, stopped one of them and launched the associated service. We could verify that the message had persisted in Fiorano's file-based repository because the next component in the orchestration wasn't available. We could configure the repository to use an external RDBMS instead of the default file-based system; if an external RDBMS is used, messages are persisted in the RDBMS instead of the file. After verifying that the message was being properly persisted we restarted the component in question, and the message was correctly routed.

Fiorano's prices its product on a per-CPU basis for the server (\$40,000 per CPU) and a subscription-based developer model (\$5,000 annually per developer). Needless to say, we are not pleased with the per-developer, per-year part of the equation. The company also charges on a per-CPU basis for adapters, so we added in \$10,000 per CPU for our database adapter, and \$4,000 per CPU for each SMTP and JMS. We also had to add in \$995 per administrator for the use of Fiorano Tools (BPEL Editor, ESB and admin tools). Ouch.

Fiorano SOA 2006 Platform, \$170,995 as tested.
Fiorano Software, (800) 663-3621, (408) 354-3210. fiorano.com/frontpage.htm

Cape Clear ESB 6.5 Cape Clear's entry is inter-

Besting because of its origins: ESB 6.5 didn't evolve from an enterprise service platform, like BEA's AquaLogic and Oracle's SOA Suite did; it didn't grow out of a conventional EAI product, like Tibco's and Software AG's products did; nor did it bloom from a messaging world, like Sonic's and IBM's products did. Rather, Cape Clear's software came out of a pure Web services world, and its ESB approach reflects these roots as much as its competitors reflect theirs.

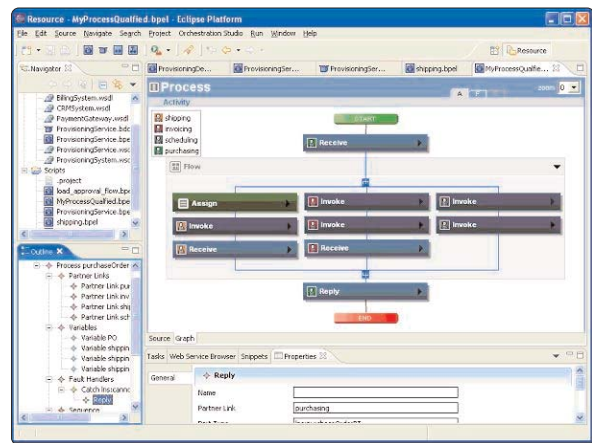
Cape Clear and BEA both view the ESB as a service orchestration mechanism and don't include conventional integration as part of their visions for their individual ESB products. Cape Clear takes the same metadata-driven orchestration approach to implementing its functionality as BEA does, though it has

chosen to tie its success to BPEL, as has Oracle, instead of a proprietary orchestration notation.

Cape Clear ESB 6.5 is J2EE-based and can be deployed within any number of enterprise service platforms, including IBM WebSphere and BEA WebLogic. It's distributed with JBoss, which is what we used for testing. Orchestration requires a BPEL repository, and we used ESB 6.5's embedded Hibernate database for this as well as for configuring Cape Clear ESB to use our Oracle9i database. The configuration can be determined during setup, and we suggest this as the best course of action if you want to use an external RDBMS as ESB 6.5's BPEL and WS-RM (Web Services Reliable Messaging) repositories. Regardless of which database you're using, if the system is already installed, and you migrate to a different database, you will have to modify the configuration files manually, or reinstall the application. Cape Clear recommends a reinstallation of its software if you need to migrate.

Cape Clear's design-time environment is an Eclipse 3.1 plug-in—something we are now intimately familiar with, as it was distributed with almost all the products we tested. The design-time environment provides the means to develop both individual services and orchestrations. We easily created services integrating JMS queues and databases; the latter required all the necessary code—including connectivity—to build. We missed having an easy mechanism for dealing with SMTP and, like Software AG's and IBM's exclusion of SMTP, we found this lack problematic. Again, coding was necessary.

Integration with a foreign JMS was a breeze, requiring that we specify an "integration" from Cape Clear's Web administrative console and then copy the appropriate JAR files to the proper directory on the server. Once an integration was defined, all we had to do was add an adapter and then call the JMS queue from our orchestration, which was presented as just another service by Cape Clear.



Mapping variables from one step to another is a simple process using Cape Clear's design-time environment.

Orchestrating our services using Cape Clear's BPEL editor was painless, though we much preferred Oracle's BPEL editor. As we published each process to the server, it was added to ESB 6.5's included UDDI 2 registry. ESB 6.5 is one of the few products that doesn't support UDDI 3 or allow access to external registries for its run-time environment. Support for external and internal registries is completely integrated into the design-time environment, however, so including services in an orchestration requires only that you choose services from the registry and drop them into the orchestration.

Once our services were created and orchestrated, deployment was a simple matter of publishing the created archives to the server. Our created archives included all requisite external JARs—such as those needed for database access—and could easily be deployed to any existing Cape Clear ESB server. Unlike the products from BEA and Oracle, Cape Clear's generates code for every orchestration/service, and it is this code that is compiled and bundled into a WAR (Web Archive) file for deployment.

Testing our service was a breeze with Cape Clear's included Web services tester, accessible directly from within Eclipse. We preferred Cape Clear's tester over BEA's simply because Cape Clear's creates the necessary SOAP environment and headers as well as the body—we had only to fill out the necessary elements and hit "send" in order to run our test.

Cape Clear ESB's process-management features are a plus, though the capabilities are much more limited than those in Oracle's product. The Web orchestration manager console gave us a view of processes, messages sent and timing information as well as the exposure of process variables. Like BEA's presentation of process variables, they had to be of a primitive type (string, integer), but we were impressed that such functionality was included at all.

There were fewer capabilities for process management than we would have liked. Additionally, we encountered errors while running against the HSQL database that prevented us from managing processes within the Cape Clear System. We initiated thousands of processes, many of which ended up hung. Although a delete option was available, it failed with a less-than-helpful error message, and we couldn't resolve the error working with Cape Clear's engineers. Cape Clear asserts that this is an issue with HSQL, and we agree that HSQL should not be used in production.

Cape Clear came in at \$35,000 per processor, and unlike many of its competitors it charges for its design-time environment—\$3,500 per developer. That drove the price of Cape Clear ESB 6.5 for NWC Inc.'s scenario over \$100,000, which put Cape Clear in the middle of the pack.

Cape Clear ESB 6.5, \$105,000 as tested. Cape Clear, (888) CAPE 439, (781) 622-2258. www.capeclear.com

IBM WebSphere Message Broker 6.0 IBM spent several years claiming enterprises didn't need an ESB,



and then appeared on the scene with not one, but two, ESB products.

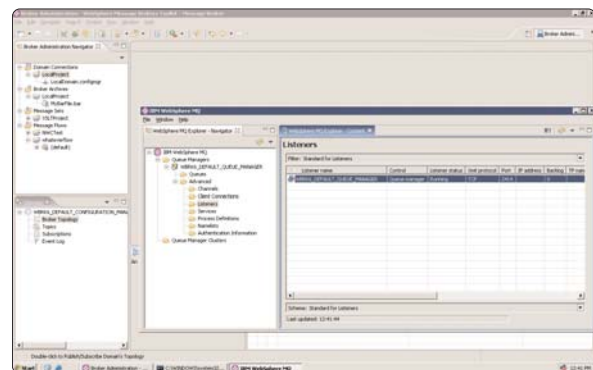
IBM sent us the more flexible of its offerings, WebSphere Message Broker 6.0. The other IBM offering, WebSphere ESB, sounds more like BEA's AquaLogic, with a stronger focus on Web services orchestration and less on support for integration of conventional legacy protocols and data formats.

WebSphere Message Broker 6.0 is a C/C++ OS-native application that takes advantage of IBM's MQ Series messaging middleware. Each Broker requires an underlying MQ instance, which is used for management and deployment. WebSphere Message Broker takes advantage of MQ as the underlying mechanism for most management functions, including the deployment of orchestrated services from its Eclipse-based Message Broker Toolkit.

We found little management is required for MQ, as it isn't needed for service orchestration. The deployment of the archive files containing service orchestrations, accomplished over MQ, was more reliable than the WebDAV- and Web-services-based deployment mechanisms of Software AG and Cape Clear, respectively.

WebSphere Message Broker persists the metadata describing the orchestration in a local database, Cloudscape or DB2 by default, but can be configured to use any number of supported RDBMSs, including Oracle and SQL Server. The configuration is read when the broker starts or when the configuration changes; while the broker is running, the configuration is cached in memory.

IBM uses terminology similar to that of BEA—we orchestrated services as a "message flow," not an orchestration, a model or a process, as other vendors termed the orchestration. IBM models in terms of input and output nodes, parsers and renderers. We especially liked IBM's flexibility in its differentiation of document format and wire format—we could determine how documents were transported on the "wire," or bus, and choose which protocol data would be used. This was a



Management of IBM Message Broker is made easy with the administration perspective from within its Eclipse-based toolkit.

marked difference from all other tested products, which predetermined the format and protocol, usually XML or native format and JMS. This flexibility translates into speed, fewer required transformations and a better ability to natively deal with formats.

We created a new message flow and specified the input format to be XML. Doing so forced the system to use the default Xerces parser, though IBM's architecture is pluggable and allows the configuration and use of other XML parsers and XSLT engines. We could have chosen any of a number of data formats, including COBOL copybook, CSV and CICS. Such flexibility in data formats was matched only by Software AG's and Fiorano's products, both of which support myriad legacy data formats.

Incoming data was parsed into a neutral, hierarchal format unless we specified otherwise, and all steps in the message flow refer to the data in its "natural" state. This requires the use of message sets and transformations for even the simplest of XML-to-XML data mappings. Although we were impressed with IBM's depth and breadth of support for transforming data, we were simultaneously dismayed at the depth and breadth of knowledge required to orchestrate our simple scenario.

The XSLT we needed to map our incoming data to other formats during execution of our orchestration was generated automatically from the message map we created using WebSphere Message Broker's drag-and-drop mapping capabilities. We were pleased with this autogeneration, because unlike most other products we tested, IBM doesn't provide a mechanism for building XSLT using a visual tool. Our biggest nit while using message maps: The map to be used for each activity in our orchestration had to be specified in the properties of the activity by selecting it from its project. We'd prefer to just drag-and-drop an existing message map onto the appropriate node and let the tool do the rest.

Creating a consumer-side connection to a Web service was a painful process, though we admit it would have been much simpler if NWC Inc. had an existing registry. WebSphere Message Broker does not directly support an incoming SOAP message as an entry point, instead simply requiring that it be marked as XML. Not a bad assumption, as SOAP is, in fact, an XML format. Unlike the products from TIBCO and Fiorano, which required us to export our orchestration to an AXIS container, all we needed to do to expose our orchestration as a Web service was specify our chosen URI within the configuration, and then deploy the orchestration. WebSphere Message Broker took care of the rest.

Accessing a database from within our orchestration was an adventure. After combing through the documentation, we created the ODBC DSN required, but discovered it was still necessary to write ESQL (Extended SQL) or Java to access the database for a simple lookup. Configuring the connection within the design-time environment was painless and familiar—it uses the same tools found in Web Sphere's application design

environment—but using the connection and its associated SQL statement was not so painless.

To deploy our orchestration we had to create an archive file (BAR, in IBM speak). Then, we simply dragged the BAR file and dropped it on an execution group in our configured WebSphere Message Broker, and the orchestration was deployed. WebSphere Message Broker groups deploy archives together by execution group, which is akin to a process on the server. Each message flow is executed in its own thread.

Overall, we were pleased with IBM's WebSphere Message Broker 6.0 as an ESB, though we shudder a bit at the amount of training you may require before you'll be productive and orchestrating services. It's flexible and offers tons and tons of options, but just like eating at a buffet, sometimes too many choices can be overwhelming.

IBM's pricing is straightforward and based on a flat \$85,000 per CPU. IBM was quick to point out that standard discounting rates apply (12 percent), but even if we did take that into consideration—which we did not, since all other vendors' pricing is list—the price would be \$149,600.

IBM WebSphere Message Broker 6.0, \$170,000 as tested. IBM, (800) IBM-4YOU. www.ibm.com

Software AG Enterprise Service Integrator (ESI)

7.2 Software AG has been around since I was born, and in the integration game nearly as long.



The software company, which is based in Germany, has undergone some changes lately and is jumping into the SOA game with both feet.

Although most products we tested had nearly one-click installs (one-click as in a single installation, as opposed to multiple installations of products in a particular order while burning incense and chanting about dead chickens under the light of a full moon), Enterprise Service Integrator (ESI) stipulated a few updates to our system before we could install all the moving parts: Microsoft Installer 3.1 was required only to install an update to Tamino (SAG's XML database), and Tamino required an external Apache installation before it could be installed. Other than that, installation was smooth.

SAG's ESI is a J2EE system deployed by default into a JBoss container. Like the products from Oracle and Cape Clear, it can be deployed within a BEA WebLogic or IBM WebSphere container. Its Tamino XML database is used to persist process data and enable load balancing/failover configurations. ESI is unique among our participants in its use of Java RMI (Remote Method Invocation) as the communication protocol between its standalone Connection Factories and the ESI core server, the Host Manager. As with several other products we tested, messages are transported on the bus in XML format. SAG uses the term "sequence" to refer to orchestrated services, and these sequences are stored as XML-based metadata in its CentraSite SOA repository. Sequences are called through portals, which clients can communicate with using SOAP/HTTP or JMS.

Portals should be considered entry points into orchestrated services; one portal is defined per orchestrated service and supports a variety of inbound protocols: JMS, HTTP and SOAP/HTTP. SAG says it plans to support File- and e-mail-based portals in future releases. To deploy our service orchestration, we had to create a SOAP/HTTP portal and start it up.

In our testing, we kicked off a sequence by sending a SOAP/HTTP document to a portal, which in turn initiated the appropriate sequence (as configured in SAG's design-time environment, XML Mediator Studio) in the component factory. Calls out to the appropriate Gateways—one for NWC Inc.'s Oracle9i database, which we defined, and a predefined mail gateway provided out-of-the-box with SAG's product—were accomplished over SOAP/HTTP.

SAG's been in the integration business for a while, but seems to live in a world of its own. Adapters are referred to as "XML gateways" and are configured in a Java application, separate from the Web console. Once we'd configured an adapter to connect to our test database and written the requisite SQL for our desired query, the gateway was deployed automatically to the system and appeared in the Web console under the "XML Gateways" heading.

We accomplished most administration and configuration using ESI's Web console; the only oddity was that we had to separately log in to the repository through the console—using a different login by default than the one we used to log in to the console—to manage the XML Gateways. Our biggest criticism regarding ESI's Web administrative hub, as it's called, is that we couldn't access our log files or even determine their location from the console. Our first attempts at deploying a Web service failed, and it took forever to find the proper logs to troubleshoot the problem. We much preferred log access over the Web console as provided by Cape Clear and Oracle.

Configuring integration with OpenJMS was easy ... until we got to the part about adding JAR files to the classpath. Needless to say, we were not amused when we were required to hand-enter each and every JAR file into a short text field on the Web console. And when we screwed up, we had to carefully arrow through the text to find the mistyped entry and fix it. Adding support for OpenJMS also required creation of a specific ComponentFactory; this task required domain-specific expertise. SAG's and Sonic Software's products have much in common in this area, and we'd be happy to see the domain-specific-expertise requirement disappear in future releases of both.

We also were disappointed by the inclusion of a registry/repository for design time, dubbed CentraSite, that communicated with the design-time environment, XML Mediator Studio, over WebDAV. We don't like WebDAV, and it doesn't like us. 'Nuff said. We'd prefer a more up-to-date solution taking advantage of standards-based protocols, such as UDDI version 2 or 3, as was provided by most of the other products we tested.

Also problematic was the lack of integration of Gateways with the repository.

Orchestration is accomplished within SAG's design-time environment, XML Mediator Studio, a standalone Java application the company says will be moving to Eclipse in a future release. Orchestrating our service was painless at first, but became problematic when it came time to configure a transformation. SAG includes a transformation "step," but requires that the associated XSLT be coded by hand. Mapping data from step to step required a guru's understanding of XPath and XSLT—we much preferred the visual mechanisms offered by just about every other product we tested. Although an included XPath editor was easy to use, it wasn't well-integrated and wasn't always available when we needed it. Setting a variable in our sequence required retrieving a value from the input document using an XPath expression, for example, but the XPath editor couldn't build the expression. We had to launch the XPath builder, load the appropriate document, create the expression, and then we could cut and paste the XPath into the appropriate field in the property sheet.

On the other hand, SAG's debugging tool was mondo-cool, and we found we liked it better than even Oracle's offering. We could modify values at run time and watch variables during sequence execution, which reduced the amount of time we spent debugging errant XPath expressions, always a good thing.

SAG's product was undergoing major changes in naming and pricing structure during our review. Pricing as of January for ESI was based on per-CPU licensing (\$50,000 for a two-CPU license), and database Gateway charges (\$50,000 for gateways connecting to two unique database types). At \$100,000, its total price to implement NWC Inc.'s scenario was comparable to the average total price of implementation for other products in our review. SAG says the charge for Gateways will be eliminated in May 2006.

Enterprise Service Orchestrator 2.1, \$100,000 as tested. Software AG, (800) 241-9905, (703) 860-5050. www.softwareagusa.com

Sonic Software SOA Suite 6.1 Sonic Software's SOA Suite is an evolutionary step on the integration



ladder, building on the messaging expertise acquired over years from Sonic's presence in the messaging market. Unfortunately, its heavy reliance on messaging terminology and architecture has led to the creation of an SOA suite that isn't nearly as SOA-oriented as it is queuing-oriented.

Sonic excelled in areas in which messaging expertise could be parleyed into an advantage—routing and messaging protocol support, for example. The standalone, J2EE-based Sonic SOA Suite comprises a limited number of moving parts. Sonic relies on its messaging bus, SonicMQ, for its SOA Suite's backbone, using multipart messages transported over JMS to move messages along the bus. Management is accomplished over JMX, as it is

with Fiorano's product, but Sonic has done a much better job of hiding the complexity and ugliness of JMX (Java Management Extensions) behind its user interface.

Sonic's SOA Suite supports the SOA paradigm by turning everything into services on the bus and routing messages to services according to orchestrations. We often found ourselves scratching our heads in the service-creation areas as we tried to understand the relationship between entry and exit points on the bus and the definition of a service. Messages come in an entry point and leave on an exit point. No relationship other than that. Fine for JMS, odd in an SOA world.

To create a service to access our Oracle 9.1 database, we had to define a connection (a common task), write the desired SQL, and then define the origins of the parameters. Although we enjoyed the visual mechanisms available in other products for crafting SQL queries, we weren't put off by the requirement to define SQL by hand with Sonic. We were, however, put off that each and every database service we created required a license. Sonic is a Progress Software company, and so is DataDirect—whose drivers Sonic uses—so why all the license malarkey? We had to enter a license key to define a database connection within Sonic's Management Console to create our service. We were not—and still aren't—amused.

Mapping data in an orchestration requires choosing the appropriate variable by message part, index/name, and header or by assigning a constant value. The process requires too much domain expertise in the area of messaging for an SOA-oriented orchestration designer. We prefer using XPath to extract values from an incoming document, even when we had to craft the XPath expressions manually. We were pleased with simply wrapping the desired input variable using a manually crafted XPath expression or within an XML element name. We also had to add the namespaces of the incoming XML document (the SOAP document) to the service—including the default namespaces for SOAP-ENV and SOAP-ENC. This blew us away—no other SOA product has ever required us to provide these namespaces, because they don't change often enough to warrant this.

Once our services were created, we included them in a service orchestration by selecting them from a list of services available from the domain manager. Sonic says it plans integration with Systinet's Business Service Registry in its upcoming release and didn't include any registry support in the version we tested. Stylus Studio, which reportedly will be replaced in forthcoming release with an Eclipse-based plug-in, uses SonicMQ to communicate with the appropriate message broker, from which it pulls configuration information and available services and processes. Unlike BPEL-based products from Oracle and Cape Clear, and proprietary products from BEA, Software AG, TIBCO and Fiorano, Sonic's required us to configure content-based routing as a separate service and add it to the service orchestration as a step!


To add insult to injury, we had to write JavaScript rules to determine how to route our message based on whether a desired widget was in stock (quantity > 0). Writing JavaScript and XPath, all at one sitting, is just cruel and unusual. The JavaScript we wrote generated a simple Boolean return, which seemed easy enough to deal with, but then we realized we had to learn how to use it to route the message. This is likely to give less-technical users fits.

After the rule was defined, we could use it to specify the condition (`isQuantityZero()`) and route to a service or process of our choosing. All the other products we tested let us configure content-based routing within a service orchestration without an external service call. This anomaly is partially due to Sonic's unique service-routing architecture, which is highly distributable, letting each individual step in a service orchestration be executed on local or remote brokers, depending on the enterprise's needs.

We thought this was great ... until we started putting a service orchestration together and discovered some rather annoying messaging quirks again creeping into our SOA. All services required JMS- or HTTP-based entry, exit and fault endpoints. Every endpoint defined required a specific QoS (Quality of Service) setting, and within an orchestration all endpoints assigned to services in that orchestration had to have the same QoS or the orchestration would not execute. The notion of QoS on every endpoint permeates Sonic's SOA Suite and caused us no end of trouble. We were also not pleased with the requirement to manually specify endpoint names within a service definition, as it was easy to mistype the name of an endpoint.

Services and processes (an orchestration of services) are assigned to an ESB container for deployment. This is a somewhat tedious process, but it did allow flexibility in determining where services and processes are deployed and also allowed for grouping of like services into specific, manageable containers. Brokers manage the endpoints (the ones assigned to services, which are reusable) as conventional message queues.

Sonic's overall pricing isn't bad, though we still balk at the mechanisms used to enforce its data-access licenses. The enterprise edition of Sonic's SOA Suite is a mere \$10,000 per CPU, with the bulk of NWC Inc.'s total scenario cost coming from developer (\$3,750 per developer for Sonic Workbench) and database service licensing (\$18,750 for a two-CPU machine per database type).

SOA Suite 6.1, \$76,250 as tested. Sonic Software , (866)GET SONIC, (781) 999-7000. www.sonicsoftware.com 



LORI MACVITTIE is a NETWORK COMPUTING senior technology editor working in our Green Bay, Wis., labs. She has been a software developer, a network administrator and a member of the technical architecture team for a global transportation and logistics organization.

Write to her at lmacvittie@nwc.com. Post a comment or question on this story at www.nwc.com/go/ask.html.