

Rigorous Evaluation of COTS Middleware Technology



To help developers assess competing packages accurately, the Middleware Technology Evaluation project provides technical evaluations of commercial-off-the-shelf middleware products and a repeatable evaluation methodology.

Ian Gorton
Pacific Northwest
National
Laboratory

Anna Liu
Paul Brebner
CSIRO
Mathematical
and Information
Sciences

Middleware refers to a broad class of software infrastructure technologies that use high-level abstractions to simplify construction of distributed systems. Over the past decade, the adoption of commercial off-the-shelf middleware products across the software industry has gathered significant momentum. COTS products make it possible to deploy application components in high-performance, scalable, and fault-tolerant environments. Most major organizations rely on some form of COTS middleware components to integrate new Internet applications with existing core business systems and to manage evolving business processes.

While COTS middleware products demonstrably solve many significant problems, their adoption and use is by no means straightforward. Because these technologies typically provide business-critical services, selecting the appropriate middleware platform is essential.¹ System and, ultimately, business failures—caused by large-scale COTS middleware products that didn't work as anticipated—litter the industry. Competition among products that appear to offer identical services further complicates the COTS middleware selection process,² especially when competing products provide different implementations of standards-based technologies, such as Corba (www.omg.org) and Sun Microsystems' Java 2 Enterprise Edition (J2EE, <http://java.sun.com/j2ee/>).

The Middleware Technology Evaluation (MTE) project rigorously evaluates COTS middleware components and technologies. As the "MTE's Fundamental Principles" sidebar describes, the project bases its evaluations on three basic principles. Project out-

put includes technical evaluation reports of key COTS middleware packages and a repeatable methodology for evaluating these products. By providing reliable, comprehensive product assessments, the MTE project helps organizations reduce the up-front costs they incur in selecting suitable COTS middleware products. This in turn should significantly reduce the downstream costs of system development by eliminating most risks associated with complex technology selection and procurement early in the process.

COTS MIDDLEWARE COMPONENTS

Middleware technologies consist of various components that form the infrastructure, or *plumbing*, of distributed applications. In effect this infrastructure provides a distributed environment, often called a *container*, for deploying application-level components that perform application-specific processing. These application components rely on the middleware infrastructure to manage their life cycles and execution, and to provide off-the-shelf services such as transactions and security. This means the two are tightly coupled: An application-level component cannot execute outside a suitable COTS middleware container. The application component behavior—the business logic—depends completely on the infrastructure component behavior.

This relationship between application and infrastructure components profoundly affects middleware technology assessment, certification, and engineering. No matter how high-quality the application components, the middleware infrastructure provides the most crucial component in most systems. If the middleware is naively architected or

MTE's Fundamental Principles

In its evaluation of COTS middleware components and technologies, the Middleware Technology Evaluation project adheres to three fundamental principles:

- *Technological agnosticism.* No technology is better than any other. Rather, technologies have individual strengths and weaknesses, which vary in relevance depending on application requirements. Knowledge of these strengths and weaknesses can help organizations achieve a better fit between products and their intended use.
- *Rigorous investigation.* MTE findings are based on detailed and rigorous explorations of the technologies. Evaluation teams test the applications, analyze their performance, and resolve their anomalies, often with the help of the vendor's product support groups.
- *Factual observations.* The evaluations contain observations based solely on fact. These cover product descriptions, configurations, and performance results obtained in the test environment. The limitations of what is to be observed are also clearly stated so that results and conclusions are not interpreted out of context.

This approach adheres to scientific methods for experimentation and analysis. This is necessary to guarantee evaluations that are as reliable and trustworthy as possible.

implemented, has subtle errors in some services, or is simply inefficient and lacking in features, the application components will inevitably fail or perform poorly. Organizations typically build the application-level components after selecting the COTS middleware infrastructure. Thus, if the infrastructure is of insufficient quality, the application is doomed to failure before it has even been designed.

COTS middleware products are large and complex. They typically have several thousand API calls and a collection of integrated services and tools of varying importance levels to different applications. Rarely, if ever, does "one size fit all."

Other issues add to the challenge of acquiring COTS middleware:

- Many products compete in the middleware arena, thus narrowing the competition to a manageable short list is a challenge in itself.
- Vendors provide vague specifications and claim superior features, performance, and functionality for their COTS products, making it difficult to perform a serious comparison of the technologies.
- Middleware technology is evolving quickly, making it difficult to discern tested from promised functionality.

MIDDLEWARE TECHNOLOGY EVALUATION PROJECT

Figure 1 shows the complete set of MTE activities. The project consists of two specific, highly synergistic middleware evaluation activities:

- laboratory-based generic product evaluation, and
- organization-specific product evaluation.

The laboratory-based activity involves building testbed applications with the middleware products under evaluation. We then stress-test the applications on the same hardware environment to produce directly comparable quantitative performance results. Our experience working with the products also helps populate a generic qualitative evaluation framework for each class of middleware technology. Detailed technology evaluation reports provide the major output.

The Middleware Architecture and Technology Evaluation (i-Mate) process¹ provides a clearly defined set of steps that helps organizations select a middleware product suitable for their application requirements. i-Mate leverages the generic qualita-

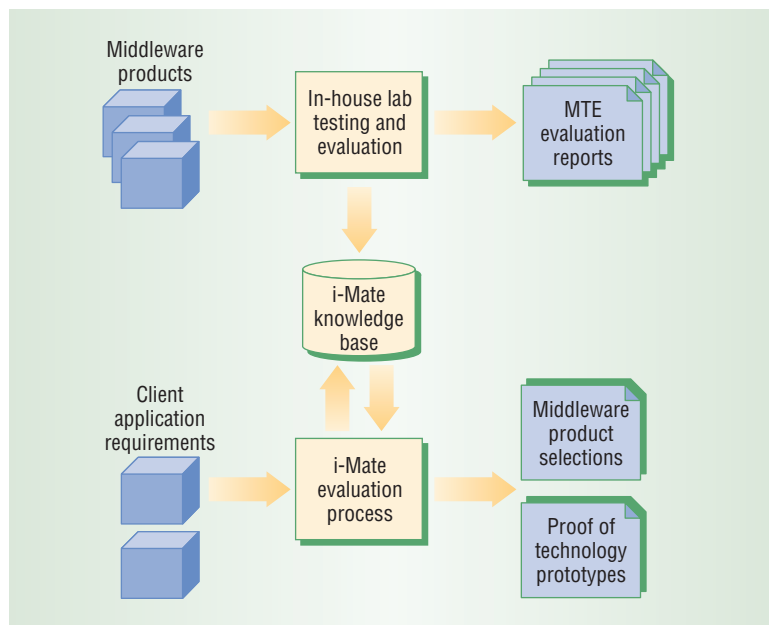


Figure 1. The Middleware Technology Evaluation (MTE) project. The process consists of two steps: in-house laboratory testing and evaluation and the i-Mate evaluation process, which focuses on specific organization needs.

tive and quantitative evaluations from in-house testing, augmenting them with requirements specific to the organization. The process resembles other COTS evaluation schemes,³⁻⁵ but its knowledge base tailors it to middleware technology evaluations.

The i-Mate knowledge base contains several hundred generic requirements for COTS middleware technologies,⁶ which we derived from in-house testing and working with vendors and organizations

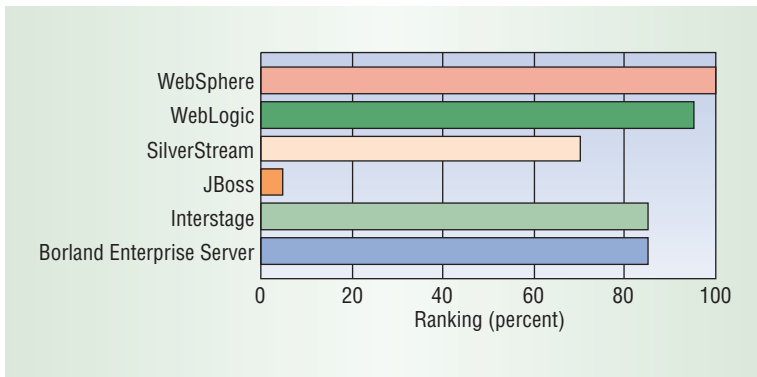


Figure 2. J2EE product rankings for the “Scalability and Availability” category. Because the J2EE specification doesn’t define features in this category, product behavior varies widely.

during product evaluation. We rank each product’s capabilities against all relevant requirements in the knowledge base. We consider cost factors in the i-Mate process, but not in the generic laboratory-based evaluations. Although all products have a nominal standard price, competition among vendors means that businesses rarely pay this price. Cost is therefore only considered in the context of a specific organization’s requirements and is based on actual vendor quotes.

The in-house evaluations populate the i-Mate knowledge base so that organizations can instantly leverage them during the i-Mate process.^{1,6} Also, when organizations request an evaluation of middleware products or versions that the knowledge base doesn’t cover, we can perform new i-Mate evaluations and use those results for subsequent requests.

Laboratory-based evaluation

The MTE project’s laboratory-based evaluation methodology revolves around two complementary activities:

- qualitative architectural analysis, and
- quantitative performance analysis.

The project bases the qualitative analysis on detailed feature matrices that capture generic product features and requirements. It groups the features into high-level categories of evaluation criteria that are important to organizations interested in acquiring middleware products. For J2EE products, for example, categories such as J2EE support are important, while for enterprise application-integration products, a message-transformation engine category is crucial. We rank each product on a numeric scale against each feature, using an iterative process that involves all evaluation team members as well as vendor representatives who help ensure that the evaluations are based on a complete and thorough understanding of the technologies.

The project also builds a sample test case for each product, using different test cases for different mid-

dleware types, such as asynchronous messaging, integration brokers, and application servers.

Stock-Online, a test case for J2EE application servers, imitates a stock-trading system, with business transactions such as buying and selling stock and creating user accounts. The system preloads a database before each test with an initial data set that includes account and stock data. During the tests, clients execute a test script that requests a known mix of transactions. However, each client randomizes the order in which the transactions execute to also randomize locks on data.

In terms of business logic, Stock-Online is much simpler than industry benchmarks such as TPC-C (www.tpc.org) and ECperf (<http://java.sun.com/j2ee/ecperf/>), and it is less demanding in terms of database resources. This makes testing and supporting Stock-Online less expensive while still providing significant insight into the middleware’s behavior.

Stock-Online exercises a number of key COTS middleware components:

- middleware infrastructure for implementing an *N*-tier application,
- transaction service,
- name or directory service, and
- load-balancing mechanism for client requests.

These components form the core of most multi-tier applications constructed using COTS middleware. They provide the message transport, database coordination and connectivity, location transparency, and scalability required for systems that operate in demanding application environments.

When the test application is running, we experiment with each product’s architectural alternatives and deployment options.⁷ Exploring these configuration options provides us with an in-depth understanding of the product architecture, its strengths and weaknesses, and areas where the technology constrains application design. The project also uses the results of these activities in its qualitative analysis of each product.

Next, we run the test application through a series of demanding performance tests. All tests use the same hardware and software infrastructure and, as much as possible, the same application code for each COTS middleware product evaluated. The test application code isolates the COTS middleware-dependent components—for example, the database access layer, business logic, test drivers, and performance analysis and measurement components—in their own modules, making them easy to replace as products are changed. This also allows us to

reuse code without middleware dependencies across different product tests.

The tests measure a technology's performance under both normal and heavy transaction loads. Test results provide insight into the products' performance and scalability potential and allow a direct comparison of their relative performance merits. They also feed into the quantitative analysis.

Sample results

J2EE is one of the most technically advanced COTS middleware technologies on the market. Its integration with Internet-enabled components makes it attractive as a back-end platform for business systems. In fact, product vendors offer more than 40 implementations of the technology. A recent MTE study evaluated six J2EE application server technologies:

- Borland Enterprise Server (BES) v. 5.0.2
- SilverStream Application Server (SS) v. 3.7.4
- WebLogic Application Server (WLS) v. 6.1
- WebSphere Application Server (WAS) v. 4.0
- Interstage Application Server (IAS) v. 4.0
- JBoss v. 2.4.3

All six support the J2EE specification, and all are superficially similar and competitive products.

Figure 2 summarizes the qualitative evaluation result rankings for the "Scalability and Availability" category. In J2EE, this category relates to a product's clustering features that both scale up and scale out an application, as well as provide replicated services to increase availability when failures occur. Because clustering, failover, and load balancing fall outside the J2EE specification's scope, the products vary widely in these areas. Some, for example, support only simple session-level load balancing—a client uses only the first allocated server. More sophisticated products support method-level load balancing and failover—a client may be allocated to multiple servers.

The quantitative evaluation deploys identical Stock-Online application code for each of the six products. We then test each product with various client loads using the same hardware test environment and stable machine configurations. Across all tests, the only variability is the COTS middleware product used.

Figure 3 shows the measured application performance of the six products in terms of transactions processed per second. Three machines were used: one for the application server middleware, one for the client processes, and one for the database. The

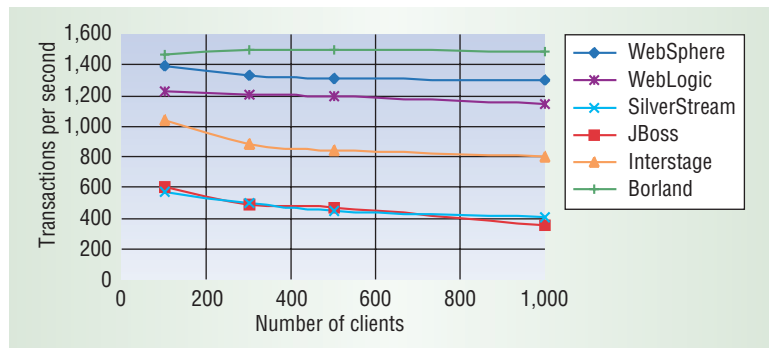


Figure 3. Application throughput for a single application server machine. The performance of the products varies widely.

tests executed the test client drivers and application servers on quad 700-MHz CPU Windows 2000 machines with 1 Gbyte of memory using an Oracle 8.1.6 database running on a separate 8-CPU Windows 2000 machine. A 100-Mbit isolated Ethernet network connected the three machines.

Many applications that use application server technology do not require high, sustained throughput. For such applications, on the order of 10 transactions per second is adequate, and all six tested technologies will suffice for such applications. With 1,000 clients, however, the best-performing products provide roughly four times the throughput of the worst performers. This difference in raw performance will dramatically affect application behavior.

Also notable is the amount by which product performance diminishes as client load increases. Ideally, a perfectly implemented technology should show minor decreases in throughput when handling large client loads, even though client response time will increase linearly with client load. Such a technology would have no resource bottlenecks and could withstand sudden bursts of activity. In this respect, the leading three products—Borland, WebSphere, and WebLogic—come close to the ideal, with less than 10 percent throughput reduction at 1,000 clients. The others show from 20 to 35 percent degradation, which does not bode well for further increases in client loads. It is therefore unlikely that these products would reliably handle large increases in request volume.

We also tested the scale-out capability of the J2EE application servers. In J2EE, clustering supports scale-out, allowing deployment of replicated application server components across multiple hosts. Ideally, adding a new host of equivalent resources will double application throughput.

Figure 4 depicts the results of introducing clustering into the tests using two identical application server machines; the JBoss version tested did not support clustering. Borland and WebSphere clearly stand out in absolute performance terms, with both demonstrating an 80 percent average increase over a single machine. Significantly, the product that scales best, Interstage, achieves superlinear scala-

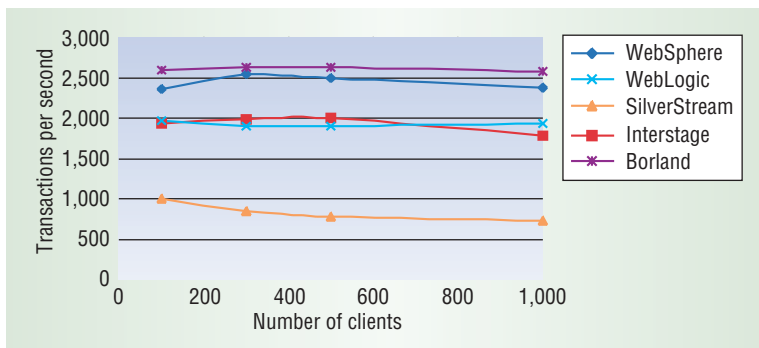


Figure 4. Application throughput for a dual-application server machine. Scalability varies significantly across products.

bility for client loads above 100. This indicates that Interstage is resource-limited on a single machine. This important result impacts capacity planning and costing for Interstage solutions.

Overall, these results indicate how each J2EE middleware technology is likely to behave in a deployed application. They also provide a sound, rigorously derived basis for deciding about a particular product's ability to meet an IT organization's needs.

PRACTICAL ISSUES AND IMPEDIMENTS

While performing COTS middleware evaluations, the MTE project uncovered difficulties relating to scale, complexity, product evolution, vendor participation, and cost. COTS middleware technologies are invariably large and complex. A typical application server product comprises both core and other components offering services that are useful in some applications. The sheer scale makes analyzing the features and performance of all components in every product impractical in terms of time and cost. Hence, the MTE project focuses on analyzing the performance of key core components and performs a qualitative evaluation of additional components and services. This strategy, essentially a practical application of the 80-20 rule,⁸ has successfully discriminated between the implementation qualities of these technologies.

Scale and complexity do not always correlate in software technologies, but they do in middleware. It is invariably difficult to get middleware products to work well—and occasionally it is difficult to get them to work at all. Performance tuning is always complex because every product offers various “knobs and dials” that can affect performance and resource usage. Achieving near-optimal performance requires product-specific knowledge and extensive experimentation. If a product has subtle bugs or behavioral quirks that defy explanation, stress testing will usually reveal them. The evaluation team consequently must often work with the vendors to diagnose unexplained performance and behavior.

The level of evaluation that MTE provides can only succeed with vendor involvement. After all, product vendors understand their technology and

can offer insights into its behavior. Because COTS middleware vendors average a major new version every 12 to 18 months, evaluations must keep pace, as vendors rightly want to have the latest version of their product evaluated. Analysts usually work closely with vendors to get prerelease versions that they can test promptly. Occasionally, however, new releases appear without warning, necessitating considerable updating. Performance tests may show that new product releases perform more slowly than older versions—news that vendors do not greet warmly.

The MTE process requires release of the product evaluation to the vendor upon completion, and vendor feedback often leads to test modification or changes to qualitative evaluation scores. However, vendors only see their relative rankings when MTE releases its evaluation reports. Although some vendors do not like the final outcome, thus far, no vendors have asked to be removed from future studies.

MTE VIABILITY

In terms of its impact on the software industry and research community, the MTE project has proven a success in its first four years. The aim now is to build upon this success and increase its impact by broadening the technology base and reaching a wider audience. Several factors will influence the achievement of these goals. They include

- direct user involvement,
- vendor investment, and
- independent, rigorous testing.

The MTE evaluation reports are useful mostly to medium and large IT organizations deploying applications that use COTS middleware products. The diversity of their requirements makes it impossible for the evaluations to cover all the factors these organizations would like to see addressed. It would therefore be useful to establish a cost-effective mechanism that an organization could use to request that we test and evaluate a feature in more detail. We could then perform an “add-on” evaluation, retaining the resulting intellectual property and distributing the results as a report update via the project's Web site.

Since its inception in 1998, the MTE project has recovered about 50 percent of its approximately US\$4 million costs. Although this level of technical analysis is a costly proposition, the current cost recovery level satisfies the requirements of the Commonwealth Scientific and Industrial Research Organization (CSIRO). There is no guarantee that this situation will persist, however.

Vendor investment in the MTE project has been mostly limited to staff resources and software licenses. For most vendors, this arrangement has been economically advantageous. Some vendors whose products rate well in the reports have reported increased sales. In addition, vendors receive valuable feedback on specific features, bugs, and overall product quality. Although it is not an MTE objective, an unanticipated benefit to vendors is that, in some cases, the project seems to have helped them perform system testing for their products. For these reasons, MTE is seeking a viable model for increased vendor involvement and investment in the project.

Rigorous performance analysis and independent reporting on a range of COTS middleware products differentiates the MTE project from the work of other software technology analysts. Other research groups, such as TechMetrix Research (www.techmetrix.com), perform similar although more limited evaluations.^{9,10} Because MTE reports “let the products speak for themselves,” we believe that the project provides evaluations that are more reliable and trustworthy than evaluations from other analysts.

The MTE project represents a significant attempt to provide rigorously derived, in-depth technology evaluations for use by middleware product adopters. The project outcomes—evaluation reports, the i-Mate process, and research papers—had a positive impact on both the software industry and research community. The MTE project proves that successfully applying rigor and science to technology evaluation is possible.

We encourage others to expand on our approach in other software technology domains, helping to create a reliable body of knowledge for use by the entire software industry. ■

References

1. I. Gorton and A. Liu, “Streamlining the Acquisition Process for Large-Scale COTS Middleware Components,” *Proc. 1st Int’l Conf. COTS-based Software Systems*, LNCS 2255, Springer-Verlag, 2002, pp. 122-131.
2. W. Emmerich, E. Ellmer, and H. Fieglein, “TIGRA—An Architectural Style for Enterprise Application Integration,” *Proc. 23rd Int’l Conf. Software Eng.*, ACM Press, 2001, pp. 567-576.
3. J. Kontio, “A Case Study in Applying a Systematic Method for COTS Selection,” *Proc. 18th Int’l Conf. Software Eng.*, IEEE CS Press, Mar. 1996, pp. 201-209.
4. P.K. Lawlis et al., “A Formal Process for Evaluating COTS Software Products,” *Computer*, May 2001, pp. 58-63.
5. S. Comella-Dorda et al., “A Process for COTS Software Product Evaluation,” *Proc. 1st Int’l Conf. COTS-Based Software Systems (ICCBSS 2002)*, LNCS 2255, Springer-Verlag, 2002, pp. 86-96.
6. A. Liu and I. Gorton, “Accelerating COTS Middleware Technology Acquisition: The i-Mate Process,” to be published in *IEEE Software*, Apr./May 2003.
7. P. Brebner and S. Ran, “Entity Bean A, B, C’s: Enterprise Java Beans Commit Options and Caching,” *Proc. Middleware 2001, IFIP/ACM Int’l Conf. Distributed Systems Platforms*, LNCS 2218, Springer-Verlag, 2001, pp. 36-55.
8. R. Koch, *The 80/20 Principle: The Secret of Achieving More With Less*, Doubleday, 1999.
9. E. Cecchet, J. Marguerite, and W. Zwaenepoel, “Performance and Scalability of EJB Applications,” *Proc. OOPSLA 2002*, ACM Press, 2002, pp. 246-261.
10. M. Arlitt, D. Krishnamurthy, and J. Rolia, “Characterizing the Scalability of a Large Web-Based Shopping System,” *ACM Trans. Internet Technology*, Aug. 2001, pp. 44-69.

Ian Gorton is the chief architect at Pacific Northwest National Laboratory. His research interests include software architectures and component technologies. He received a PhD in computer science from Sheffield Hallam University. Gorton is a member of the IEEE Computer Society. Contact him at Ian.Gorton@pnl.gov.

Anna Liu is a software architect at Microsoft Australia. Her research interests include software architectures and application integration. Liu received a PhD in computer engineering from the University of New South Wales in Australia. She is a member of the IEEE Computer Society. Contact her at Annali@microsoft.com.

Paul Brebner is a research engineer in the Software Architectures and Component Technologies Group at CSIRO Mathematical and Information Sciences. His research interests include component technologies and software engineering. Brebner received an MSc in computer science/artificial intelligence from Waikato University, New Zealand. Contact him at paul.brebner@csiro.au.