

Accelerating COTS Middleware Acquisition: The i-Mate Process

Anna Liu, *Commonwealth Scientific and Industrial Research Organisation, Australia*

Ian Gorton, *Pacific Northwest National Laboratory*

Most major organizations now use some commercial-off-the-shelf middleware components to run their businesses. Key drivers behind this growth include ever-increasing Internet usage and the ongoing need to integrate heterogeneous legacy systems to streamline business processes.¹ As organizations do more business online, they need scalable, high-performance software infrastructures to handle transactions and provide access to core systems. Legacy system access

in particular requires well-managed integration of disparate systems not originally designed to interoperate.

COTS middleware components provide core software infrastructures that help developers build Internet-enabled business applications.² Middleware offers prebuilt services that support, for example, off-the-shelf distributed transaction processing, security, and directory and naming services. It also provides specialized components that let developers integrate many different legacy systems and design and deploy new business processes that integrate multiple distributed applications. As the COTS middleware market grows, however, product selection becomes complex, and organizations risk investing in a product that ultimately doesn't meet their requirements.³

We developed i-Mate (Middleware Architecture and Technology Evaluation for the Internet) to help organizations accelerate and improve the reliability of the COTS middleware acquisition process. This spe-

cialized software engineering process takes a three-tiered approach to COTS middleware evaluation that provides

1. A prefabricated, reusable set of generic requirements based on analysis of middleware component characteristics
2. A process for incorporating application-specific requirements, weighting individual requirements, and ranking specific COTS middleware products against these requirements
3. Tool support for capturing and rapidly exploring requirement tradeoffs and generating reports showing how the middleware products compare against the requirements

We designed the i-Mate process for organizations operating at Level 3 in the Software Engineering Institute's Software Acquisition Capability Maturity Model,⁴ especially in its support for the User Requirements and Acquisition Risk Management key process ar-

COTS middleware speeds e-business application deployment but can be difficult to select. i-Mate provides a structured software engineering process for reliable COTS middleware acquisition.

eas. We discuss here our experiences helping financial, telecommunications, manufacturing, and large government organizations evaluate, build, and deploy enterprise-strength Internet systems that use COTS middleware technology.

What makes COTS middleware selection difficult?

COTS middleware supports Internet applications by providing a distributed infrastructure for application-level components to carry out business-specific processing. If the middleware can't handle the required transaction volume, is unreliable, is insecure, or lacks key features, it won't effectively support an e-business site and the venture could fail. Selecting appropriate middleware greatly reduces the risks of application and business failure, making this a crucial step in the software engineering process.

Unfortunately, most IT organizations have difficulty selecting an appropriate COTS middleware technology.³ We see three main reasons for this.

- *Technology complexity.* Middleware products are large, diverse, and complex, and have thousands of features and application programming interfaces. Low-level, often obscure details can greatly affect product behavior.
- *Product differentiation.* Superficially, many competing COTS middleware products seem to have almost identical features and capabilities, yet both prices and real features vary widely, further complicating selection and acquisition. (The sidebar "Middleware Technologies" briefly reviews the major types on the market.)
- *IT organization knowledge.* End-user organizations rarely have architects and engineers who fully comprehend the full range of middleware technologies and products. Choosing a middleware product thus becomes a time-consuming, expensive exercise that distracts key engineering staff from their primary tasks.

Also, application-level components rely on the middleware components for life-cycle and execution management and off-the-shelf services such as transactions and security. Hence, an application-level component cannot execute outside of a suitable middle-

Current middleware technologies fall into four broad categories.

Corba-based technology (www.omg.com) enables transparent, synchronous interobject communication across different processes, written in different languages, running on different hosts. These technologies hide the complexities of low-level networking details from the application programmer and provide easy integration across heterogeneous computing platforms. Further, the IIOP (Internet Inter-ORB Protocol) enables integration of systems built on different Corba products. Leading products include Iona Technology's Orbix 2000 (www.iona.com) and Borland's Visibroker (www.borland.com).

Java-based technologies have evolved through Sun's Java 2 Enterprise Edition specification (J2EE; <http://java.sun.com/j2ee>). This defines the Enterprise Java Beans programming model, which is built on the Java Remote Method Invocation (RMI) communication protocol. It enables communication between different Java components across different machines and simplifies many distributed-programming issues by incorporating transaction services, component life-cycle, integration and persistence services, and security.

Messaging and integration technologies include message-oriented middleware such as IBM's Websphere MQ family (www-3.ibm.com/software/ts/mqseries) and Tibco's Rendezvous (www.tibco.com/solutions/products/active_enterprise/rv/default.jsp). These widely deployed messaging products provide asynchronous communication models for autonomous systems to exchange messages in both point-to-point and multicast modes. Add-on services such as message transformation and translation engines simplify business integration by unifying message formats and offering event-triggering services.

Proprietary technologies include Microsoft's .Net platform (www.microsoft.com/net), which provides component-based technology for deploying applications on the Windows platform. .Net includes class libraries that give components access to base-level middleware services. All .Net compilers then generate a common intermediate code representation that is translated to machine code as the application executes. .Net also includes extensive support for Web services technologies, which provide a nonproprietary access mechanism for .Net applications.

ware environment—application component behavior depends completely on that of the infrastructure components (see Figure 1).

This interdependence profoundly influences technology acquisition and application engineering. Regardless of application component quality, the COTS middleware infrastructure becomes critical in most systems. If the middleware is naively architected or implemented, has subtle errors in some services, or is simply inefficient and lacks features, the application components inevitably pay the price. The two are tightly coupled.

Interestingly, vendors supporting open-standard COTS middleware infrastructures such as Corba and J2EE can actually exacerbate this problem. Vendors implement their own versions of these middleware infrastructures, so each product performs differently and offers varying levels of reliability, scalability, and other quality attributes.⁵ This means, for example, that a J2EE com-

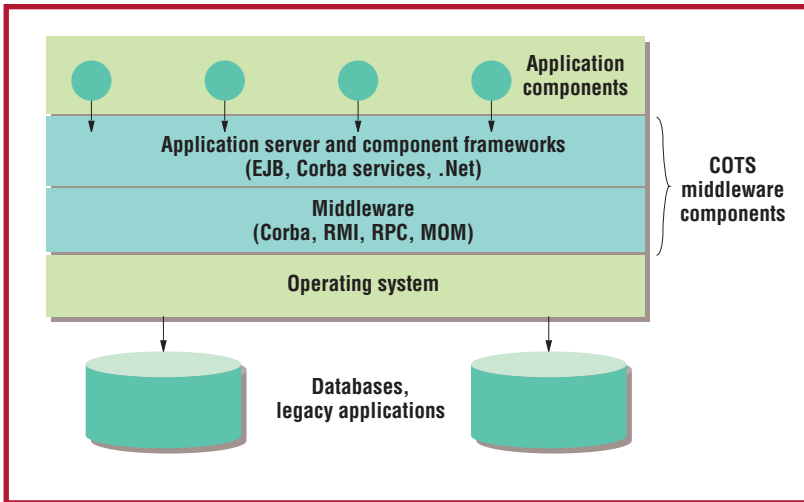


Figure 1. Anatomy of a COTS middleware application.

ponent's performance depends on the J2EE product it runs on—the same component might perform differently on two different J2EE implementations.⁶ This doesn't occur, of course, with single-source component technologies such as .Net or MQSeries.

Most organizations acquiring new middleware have no prebuilt application-level components to run—developers build application components after they select middleware. If the middleware infrastructure selected can't adequately support application component behavior, the application is doomed before it's even designed. Unfortunately, COTS middleware products are large and complex, typically having a thousand or more API calls and assembling integrated services of varying importance to different applications. Rarely does one size fit all in acquiring these technologies.

Other issues also complicate the acquisition of COTS middleware. Numerous competing middleware products make it difficult to narrow the competition down to a manageable shortlist. Vendors' vague product specifications and sometimes inflated feature and functionality claims obscure serious comparison of these many products beyond a superficial level. And middleware evolves quickly, making it difficult to discern tested from promised functionality.

Any organization can tackle these issues with sufficient time and resources. Unfortunately, the accelerated pace of e-business doesn't permit protracted product selection and acquisition processes. It also magnifies the consequences of choosing an inadequate product, as poor application performance and reliability translate directly to reduced revenues. These conflicting demands of reduced time and increased risk create significant software engineering problems.

i-Mate

The i-Mate process offers three tools to help IT organizations more effectively engineer Internet-based applications that require middleware technology. i-Mate provides a defined process for gathering, ranking, and weighting application requirements that the middleware must meet. Its knowledge base contains several hundred generic requirements for various COTS middleware product classes and rates existing products' ability to address these specific requirements. The requirements analysis tool enables rapid assessment, experimentation, and presentation of how the middleware products under evaluation compare against the project requirements.

The process

The i-Mate process, described in detail elsewhere,^{3,7} defines the steps taken, decisions made, and artifacts produced at each stage of middleware evaluation and selection (see Figure 2).

Elaborate customer requirements. This first step produces a document capturing customer requirements. Customers rarely understand the requirements in depth because the technology and application problems are complex. We therefore hold several workshops with the application stakeholders—ideally including both IT and business groups—to elicit the requirements. The resulting document details the business and technical requirements specific to the need for COTS middleware technology in this application environment. It expresses each requirement as a single item that we can evaluate against a specific middleware technology.

Augment with generic requirements. We then introduce the i-Mate knowledge base of over 200 generic, broadly applicable requirements for middleware technologies to augment the application-specific requirements. It produces overall middleware application requirements represented as individual requirement points.

Rank overall requirements. Working with the key application stakeholders, we rank the overall set of requirements. At a coarse level, we label each requirement mandatory, desirable, low priority, or not applicable; then,

within each category, we assign importance weightings for fine-grained control over requirement rankings.^{7,8} This step provides a collection of weighted requirements that the i-Mate requirements analysis tool stores.

Identify candidate products. This step identifies three to five middleware products most likely to meet the overall application requirements. In some cases, the customer has already identified a short list based on both technical and business criteria. If there is no short list, we work with the customer to identify candidates based on our knowledge and experience.

Evaluate products. Workshops with the key customer stakeholders and product vendor representatives help us evaluate each candidate product against the overall requirements. We score each product on each requirement point and capture this in the requirements analysis tool. This involves in-depth technical discussions and stepping through relevant application scenarios to understand precisely how the COTS middleware products behave. In some cases, product capabilities and features can cause the process to iterate and refine the requirement rankings. Once we've evaluated all products, the requirements analysis tool automatically calculates weighted summary scores⁷ based on individual requirement point scores and weightings. It also automatically creates summary charts to support efficient presentation and reporting.

Product evaluation generally takes one to three days per product, depending on the i-Mate team's familiarity with each product. These efforts, and the final stage of prototyping, consume most of the team's resources.

Analyze scenarios. By varying requirement weightings, the requirements analysis tool simplifies exploring various "what-if" scenarios and tradeoffs that can help us further differentiate between candidate products or confirm a product's appropriateness under varying requirements. This step yields a recommendation of one or more products that appear able to satisfy the application requirements.

Develop a proof-of-technology prototype. If product evaluation results are inconclusive,

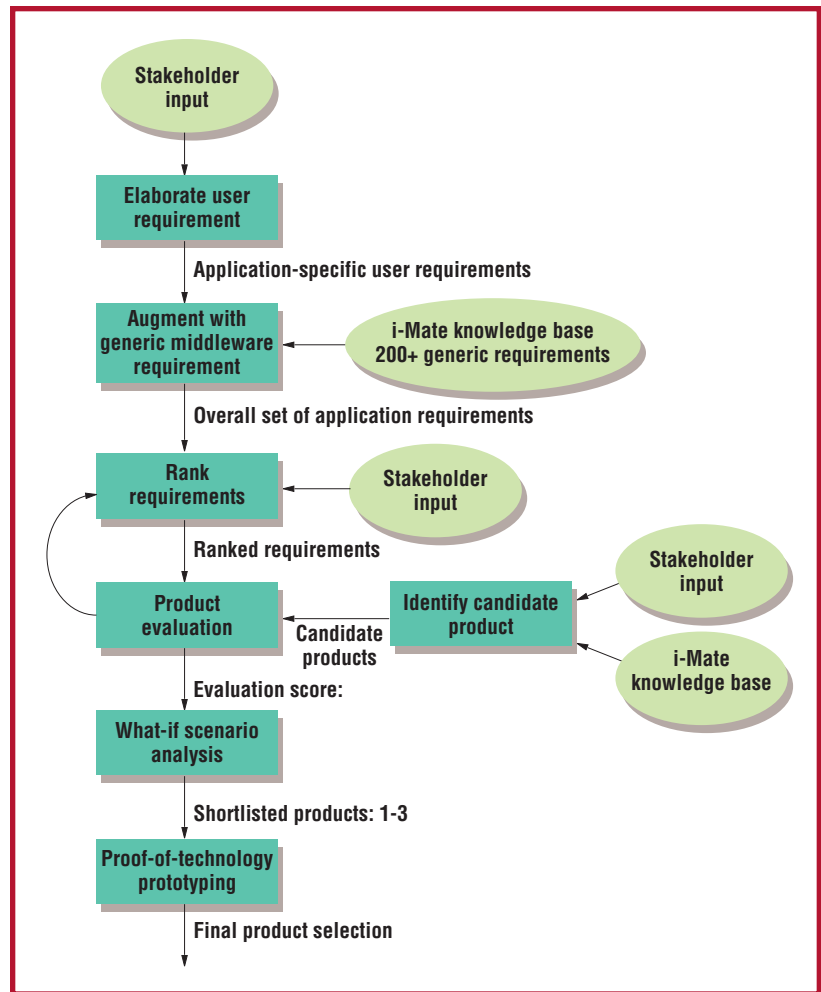


Figure 2. The iMate middleware selection process.

we develop a rapid proof-of-technology prototype that typically implements one critical scenario to exercise or stress the highest-priority requirements. Even very simple prototypes usually give concrete, indisputable evidence of product capabilities. In several i-Mate projects, prototype results provided the differentiation needed for final product selection. In fact, we always recommend a prototyping phase in i-Mate, even if one product emerges from the process as a clear leader. When considering only one product, prototyping can help validate key application requirements.

Prototyping depends on the desired prototype's complexity: for a simple system, the prototyping stage lasts less than a week, but for higher-risk applications, it can last a month or more.

The knowledge base

The i-Mate knowledge base contains an extensive set of generic requirements for COTS middleware technologies. We derived these generic requirements from our practical experiences with the CSIRO (Australia's Commonwealth Scientific and Industrial Research Organisation) Middleware Tech-

In addition to the categorized requirements, the i-Mate knowledge base includes evaluations of current major middleware product versions.

nology Evaluation project,⁹ working with product vendors and on consulting engagements for clients.¹⁰

The MTE project created a different instantiation of the overall knowledge base for each middleware product class, including Enterprise Application Integration,¹ J2EE, and Corba technologies. We discuss the EAI version here to illustrate how the knowledge base operates. For each version, detailed analysis of the generic requirements generates a knowledge base structured as a set of high-level categories encapsulating several individual requirement items. The EAI version's high-level evaluation categories include

- *Costing*: basic technology, services, and training costs
- *Message management*: facilities available for message format definition, management
- *Integration architecture*: core architectural features, flexibility
- *Adapters*: range and quality of adapters available for integration to external systems
- *Rules engine*: how are transformation rules built?
- *Send and receive operations*: fundamentals of messaging, including mode of operation, quality of service
- *Development and support*: how are applications developed, debugged?
- *Performance*: raw performance and scalability issues
- *Security*: authentication, authorization, encryption facilities
- *Transaction services*: facilities available for supporting transactional behavior
- *Workflow*: business process management and automation features
- *System management*: how are applications deployed, managed, versioned?
- *Technical*: miscellaneous technical requirements

Each high-level category contains 10 to 20 individual requirements. For example, the send and receive operations category specifies that

- The product must support programmable quality-of-service levels (includes send and forget, persistent, transactional).
- Receivers must be able to programmatically register interest in message group

or type, and wait synchronously for messages to arrive.

- Receivers must be able to programmatically register interest in a message group or type, and receive the message asynchronously through a callback mechanism.
- The product must support programmable wait periods on message receive operations.
- Receivers must be able to programmatically deregister interest in a message group or topic.
- A sender must be able to send a message to many receivers.
- Message destinations must be logical, not physical.
- A sender must be able to send multiple message types to a single logical destination or queue.
- The product must support sending a message and waiting for a response from the receiver.
- The product must handle data representation differences across different hardware and operating system platforms.
- The product must support the ability to send files as well as messages.
- The product must support handling up to 64-Mbyte messages (larger size requirements should be added as additional requirements).

These requirements cover low-level middleware features, and accompanying notes and explanations provide further details and examples. All applications will inevitably require some or all of these capabilities. During an i-Mate project, we lead the client through the knowledge base contents and determine each requirement's importance to the client application. With technologically cognizant clients, the process is fast and straightforward, taking less than a day. In other projects, the client relies on the i-Mate team to explain many of the requirements' implications, and we set their relative importance collaboratively.

In addition to the categorized requirements, the i-Mate knowledge base includes evaluations of current major middleware product versions, ranking each product on a scale of one to five against individual requirements. Two mechanisms generate and update rankings (see Figure 3). First, the MTE project rigorously evaluates middle-

ware technologies using a defined, repeatable approach.⁹ The MTE evaluation outputs feed directly into the i-Mate knowledge base evaluations. Second, during i-Mate projects, clients often request an assessment of a middleware product or version not previously evaluated. The i-Mate team then works with the product vendor to rank the product features, and the resulting evaluation extends the knowledge base's product coverage.

Reusing i-Mate's generic requirements saves organizations the cost of developing their own middleware requirements set. We can then focus on capturing the client's application-specific requirements and adding these to the off-the-shelf set in i-Mate's knowledge base. This saves both time and effort and helps ensure a low-risk outcome.

Requirements analysis tool

To support i-Mate, we built a custom requirements analysis tool that

- Captures individual requirements points, both generic and application-specific, structured as high-level categories
- Captures product rankings and requirement weightings
- Instantly calculates weighted averages for requirement categories
- Instantly calculates and reports evaluation outcomes using charts and graphs

Figure 4 shows a screenshot of the tool (with anonymous products). We built the tool on top of a COTS spreadsheet package. Scripts and formulas extend the spreadsheet's core features to create a low-cost, extensible tool that helps accelerate i-Mate evaluation projects. This approach proves especially useful during the product evaluation and what-if scenario analysis phases. Because the spreadsheet is live, the graphs depicting the evaluation scores immediately reflect any changes made to category rankings or requirement item weighting.

For example, Figure 5 shows the screen for setting requirement category weightings. In this project, the rules engine, development, and support and system management categories have highest priority. These settings generate a set of graphs representing product rankings. We can now explore how the overall evaluation results might vary if we reduce one of these high-priority categories to

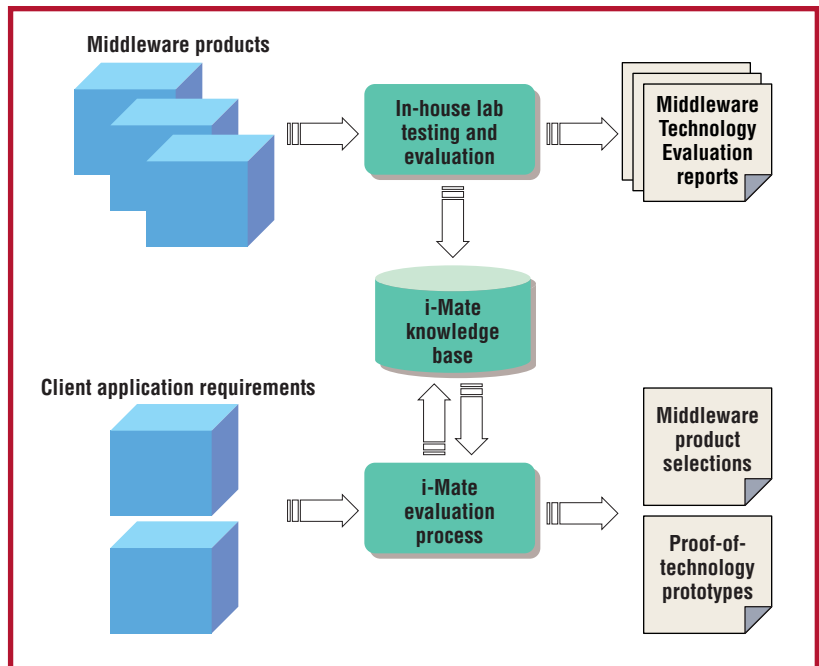


Figure 3. Populating the i-Mate knowledge base.

medium priority. This lets us rapidly explore alternatives and confirm the evaluation results under various alternative scenarios.

Case studies

A large financial organization planned to replace its existing custom messaging middleware with a COTS product. The custom technology gave them a fast messaging layer that offered a high level of guaranteed transaction behavior. However, as Internet-based applications placed greater transaction loads on this messaging layer, maintenance and upgrading became too costly. This, in turn, affected application upgrade cycles, especially because developers couldn't easily leverage the growing range of COTS application integration components.

Figure 4. The i-Mate requirements analysis tool.

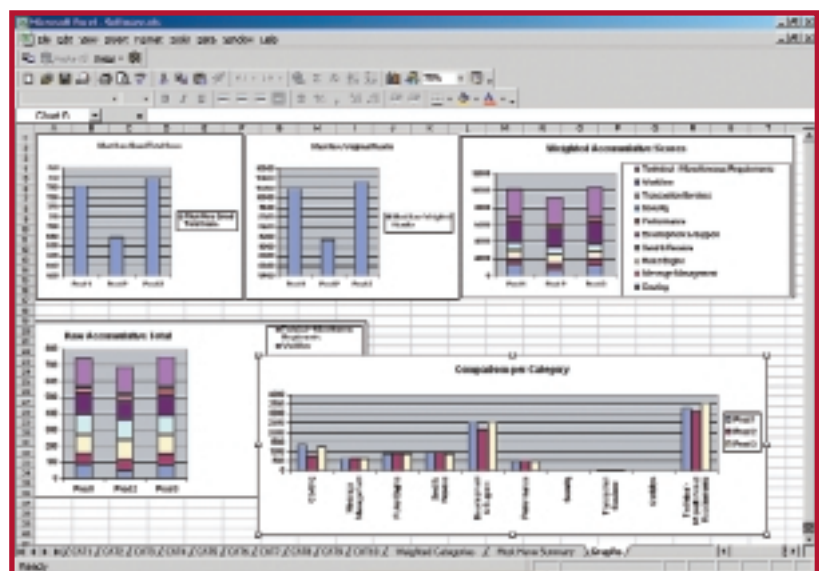
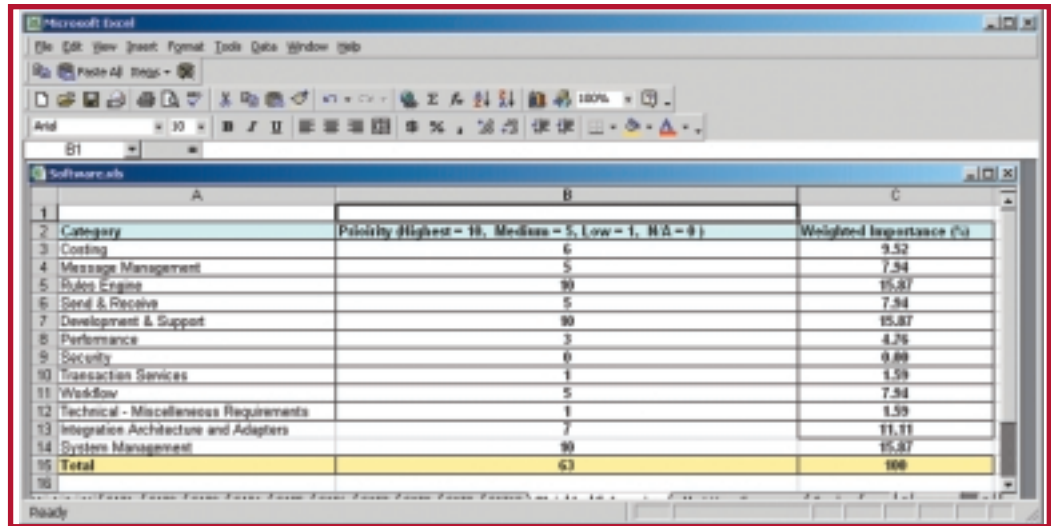


Figure 5. Requirement category weightings.



Category	Priority (Highest = 10, Medium = 5, Low = 1, N/A = 0)	Weighted Importance (%)
Costing	6	9.52
Message Management	5	7.54
Rules Engine	90	15.87
Send & Receive	5	7.54
Development & Support	90	15.87
Performance	3	4.76
Security	0	0.00
Transaction Services	1	1.59
Workflow	5	7.54
Technical - Miscellaneous Requirements	1	1.59
Integration Architecture and Adapters	7	11.11
System Management	90	15.87
Total	63	100

The organization had drawn up a short list of three COTS middleware products. These differed in nature, but superficially all offered both high-speed best-effort messaging and optional messaging with strong transactional guarantees of delivery. At this stage, the organization engaged CSIRO to help them differentiate between these three products.

The i-Mate process added the application-specific requirements to the generic middleware technology requirements already in the knowledge base. After ranking all requirements, we noted the paramount importance of two requirement areas: transactional messaging and message transformation.

We then arranged vendor workshops to elicit each product's capabilities against the full requirements set. In most cases, the issues involved deep technical features, as we discovered precisely how the products worked "under the covers." Several vendor workshops, close examination of product documentation and, in two instances, telephone conferences with the product development groups helped us establish the implications of certain architectural mechanisms.


This process enabled us to reconcile the respective vendor and user organization rankings for each product against the requirement points. Once we completed the overall product ranking set, we used the analysis tool to see which products best matched the application requirements. We saw that one product excelled in transactional and message transformation features. By varying the requirement weightings, we explored several "what if" scenarios to see the ramifications of possible requirements changes. These essentially confirmed that the leading product did not become less attractive under varying conditions and assumptions.

We completed this project in three weeks, and the i-Mate process took 22 person-days, most consumed by vendor workshops needed to understand technical details and eliminate technical risk. The client organization accepted the i-Mate process recommendation, then conducted a four-week prototyping exercise to confirm the product could meet the application's performance requirements. This was successful, and the application is now operational.

In another case, a major Australasian manufacturing business sought middleware to provide a consistent interface between their 30-plus back-end systems.¹⁰ Maintaining over 150 point-to-point interfaces between applications had become costly and unmanageable. The i-Mate process helped the organization rapidly differentiate between seven competing products. We based the final technology selection on a three-day prototyping exercise that helped us discover the benefits of one of the two short-listed products.

In both projects, the i-Mate process helped application stakeholders quickly explore numerous technical issues and differentiate between COTS middleware products. The process proved agile and able to address application-specific requirements, and the analysis tools led to highly reliable recommendations that IT managers can easily justify and present to management.

The next challenge is to leverage the i-Mate process and knowledge base across many more projects and organizations. One possible mechanism is to Internet-enable the process, providing access to the knowledge base contents and process deliverable templates through a Web browser.

This would make core parts of i-Mate widely available and, when associated with value-added advisory services, create a valuable and reliable IT industry resource. 

References

1. R. Zahavi and D.S. Linthicum, *Enterprise Application Integration with Corba Component and Web-Based Solutions*, John Wiley & Sons, New York, 1999.
2. G.T. Heineman and W.T. Councill, *Component-Based Software Engineering*, Addison-Wesley, Boston, 2001.
3. S. Comella-Dorda et al., "A Process for COTS Software Product Evaluation," *Proc. 1st Int'l Conf. COTS-Based Systems (ICCBSS 2002)*, Lecture Notes in Computer Science, vol. 2255, Springer-Verlag, Berlin, 2002, pp.86–96.
4. J. Cooper and M. Fisher, *Software Acquisition Capability Maturity Model (SA-CMM)*, Version 1.03, CMU/SEI-2002-TR-010, Software Eng. Inst., Pittsburgh, Mar. 2002; www.sei.cmu.edu/publications/documents/02-reports/02tr010.html.
5. I. Gorton, A. Liu, and P. Tran, "The Devil Is in the Details: A Comparison of Corba Object Transaction Services," *Proc. 6th Int'l Conf. Object-Oriented Information Systems*, Springer-Verlag, New York, 2000, pp. 211–221.
6. S. Ran, P. Brebner, and I. Gorton, "The Rigorous Evaluation of Enterprise Java Bean Technology," *Proc. 15th Int'l Conf. Information Networking (ICOIN-15)*, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 93–100.
7. J. Kontio, "A Case Study in Applying a Systematic Method for COTS Selection," *Proc. 18th Int'l Conf.*

About the Authors



Anna Liu is an enterprise architect at Microsoft Australia, where she works with the financial industry to address enterprise application integration problems using commodity middleware technologies. She was a senior researcher at CSIRO in Sydney, Australia, at the time of this work. She is also a part-time visiting scientist at the Software Engineering Institute, Carnegie Mellon University. Her research interests include software architectures, patterns and best practices, and COTS software evaluation and acquisition methods. She holds a BEng (Hons) and PhD in computer engineering from the University of New South Wales, Australia. Contact her at Microsoft Australia, 1 Epping Rd., North Ryde NSW 2113, Australia; annali@microsoft.com.

Ian Gorton is the chief architect in information sciences and engineering at the US Dept. of Energy's Pacific Northwest National Laboratory. He is also an adjunct professor of computer science at the University of Sydney. Previously he was a senior researcher at CSIRO in Sydney, Australia, and has worked in both academia and industry since obtaining his PhD from Sheffield Hallam University. His research interests include software architectures, particularly designing large-scale, high-performance information systems that make extensive use of COTS middleware technologies. Contact him at Pacific Northwest National Laboratory, Battelle, PO Box 999, MS K7-28, Richland, WA 99352; ian.gorton@pnl.gov.



Software Eng., IEEE CS Press, Los Alamitos, Calif., 1996, pp. 201–209.

8. P.K. Lawlis et al., "A Formal Process for Evaluating COTS Software Products," *Computer*, vol. 34, no. 5, May 2001, pp. 58–63.
9. I. Gorton and A. Liu, "Software Component Quality Assessment in Practice: Successes and Practical Impediments," *Proc. 24th Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 2002, pp. 555–559.
10. I. Gorton and A. Liu, "Streamlining the Acquisition Process for Large-Scale COTS Middleware Components," *Proc. Int'l Conf. COTS-Based Software Systems*, Lecture Notes in Computer Science, vol. 2255, Springer-Verlag, Berlin, 2002, pp. 122–131.

For more information on this or any other computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

1/2 PAGE FILL