

Software Component Quality Assessment in Practice: Successes and Practical Impediments

Ian Gorton
Chief Architect,
Pacific Northwest National Laboratory
Richland WA 99352, USA
+1 509 375-4367
Ian.Gorton@pnl.gov

Anna Liu
Senior Research Engineer
CSIRO, North Ryde,
Sydney NSW 2113, Australia
+61 2 325 3216
anna.liu@csiro.au

ABSTRACT

This paper describes the authors' experiences of initiating and sustaining a project at CSIRO aimed at accelerating the successful adoption of COTS middleware technologies in large business and scientific information systems. The projects aims are described, along with example outcomes and an assessment of what is needed for wide-scale software component quality assessments to succeed.

1. INTRODUCTION

Software reuse is one of the holy grails of software engineering. The ability to assemble new applications from pre-fabricated, pre-tested software components has the potential to deliver enormous economic and engineering benefits to the software industry. In many expert views, wide-scale reuse is one of the cornerstones required to transition the building of software systems from a craft to an engineering discipline.

Many of the problems associated with wide-scale reuse of software components are well documented, in both research and practice. However, much of this work focuses on the reuse of small-scale, general-purpose software components. Examples of this are C and FORTRAN libraries, C++ classes, Visual Basic widgets, and so on.

Interestingly, some parts of the software industry are already rather successful at reusing software components. For example, large-scale information systems are routinely built from assemblies of interoperable software components that are customized for a specific application purpose. What distinguishes the reuse in this context is that the vast majority of components are large-scale, complex, commercial off-the-shelf (COTS) products. In this category are relational databases, ERP applications¹, transaction processing monitors, middleware, workflow systems and a multitude of business-specific packages.

Whilst large-scale COTS components solve many problems, their adoption and use is by no means straightforward. These components typically provide business-critical services, which make the selection of a component a key activity. The industry is littered with numerous examples of systems and ultimately business failures because a large-sale COTS component did not work as expected. In addition, competition amongst several products that superficially offer the same set of services complicates the COTS component selection problem even further. This is especially true when the competing products provide differing implementations of standard-based technologies, such as CORBA and J2EE.

This paper firstly describes some of the major experiences of CSIRO's Middleware Technology Evaluation (MTE) project. This project started in 1998 and performs rigorous technical evaluations of various COTS middleware components and technologies. The project outputs include technical evaluation reports of key COTS middleware products and a repeatable methodology for performing these evaluations. By providing reliable, comprehensive product assessments, it becomes possible for organizations to significantly reduce the up-front costs incurred in evaluating and selecting suitable COTS middleware components. This in turn significantly reduces the downstream costs of system development by eliminating at an early stage the majority of the risks associated with complex technology selection and procurement.

Second, the paper discusses some of the key issues regarding the success and viability of this approach. These lessons have been learned through the in-house MTE project work at CSIRO, as well as in on-going projects in which the authors help external CSIRO customers and project teams within PNNL with various architectural issues relating to COTS middleware components.

2. THE MTE METHODOLOGY

The rigorous evaluation methodology for the MTE project revolves around two different but complementary activities, namely:

- Qualitative architectural analysis
- Quantitative performance analysis

Both activities are driven by working with every COTS middleware product to build applications. With each product, at the very minimum an example test case known as *Stock-Online* is built. *Stock-Online* imitates a stock trading system, and has seven

¹ Enterprise Resource Planning applications (e.g. SAP, PeopleSoft)

business transactions for tasks such as buying and selling stock, and creating user account details. The system requires a database to store the account and stock data, and this database is preloaded before each test with a known initial data set. As tests are run, clients execute a test script in which a known mix of transactions is requested. However, each client randomizes the order in which the transactions are executed so that locks on data are also randomized.

Stock-Online is specifically designed for testing COTS middleware that operates synchronously. These technologies, commonly known as *application servers*, use a remote method invocation style of interaction, where client calls block until the server completes processing the request and returns data to the client. Also, in contrast with industry benchmarks, (e.g., TPC-C³ and ECPe⁴) *Stock-Online* is deliberately much simpler in terms of business logic, and less demanding in terms of database resources. This makes it a lower cost proposition to both test and support, while still providing significant insight in to the behavior of a middleware technology. We have other test cases designed for asynchronous, message-oriented middleware and integration technology, but these are beyond the scope of this paper.

Stock-Online has a number of key ingredients that are tested for each COTS middleware product. These are:

- Middleware infrastructure for implementing an N-tier application
- A transaction service
- A name or directory service
- A client request load-balancing mechanism

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

When the test application is up and running, the various architectural alternatives and deployment options that each product offers are examined. By experimenting with these different configurations, we gain an in-depth understanding of the product architecture, the various strengths and weaknesses, and areas where the technology constrains application design [1]. The results of these activities feed in to the qualitative architectural analysis provided for each product in the MTE reports.

Next, the test application is put through a series of demanding performance tests. All the tests utilize the same hardware and software infrastructure (e.g., machines, network, and database) for each COTS middleware product evaluated. As far as possible, identical application code is also used across product tests. In all tests, the COTS middleware-dependent components are isolated in their own modules. This makes it easy to replace these dependencies as products are changed, and means we can attain complete reuse of the code that has no middleware dependencies. This includes the database access layer, business logic, test drivers and performance analysis and measurement components.

The aim of these tests is to measure how a technology performs under both normal and heavy transaction loads. The results obtained give deep insights in to the performance and scalability potential of the various products. They also make it possible to do direct, meaningful comparisons about the relative performance merits of various COTS middleware products. These results also feed in to the quantitative analysis provided for each product in the MTE reports.

This whole approach is driven by some fundamental principles to which the project adheres:

Technological Agnosticism: There is no fundamental belief that any technology is for some reason *better* or *worthier* than any others. Different technologies have different strengths and weaknesses, and these vary in relevance depending on application requirements. By striving to expose these strengths and weaknesses, it should be possible for IT organizations to achieve a better fit between products and their use in enterprise solutions.

Rigorous Investigations: The findings contained in the MTE reports are all based on detailed and rigorous explorations of the technologies. Applications are tested, their performance is analyzed and anomalies resolved, often with the help of the vendor's product support organizations.

Factual: The reports contain observations based on solely on fact. These cover product descriptions, configurations and performance results obtained in our test environment. The limitations of what is observed are also clearly stated so that results and conclusions are not interpreted out of context. There is occasional speculation about the behavior of components or configurations that have not been fully explored. This is however always explicitly stated, and only done when there is a very high degree of probability of the validity of the speculations.

3. SAMPLE RESULTS

One of the major parts of the MTE evaluations is performance analysis. Anecdotal experience suggests that there are often major differences in performance between middleware components. For many applications, performance and scalability are 'make-or-break' factors. It is therefore crucial to select a middleware technology that can provide the necessary sustained and peak performance needed by the application.

One of the MTE studies has in September 2001 evaluated six major application server technologies, namely:

- BEA WebLogic Server (WLS) v6.0
- Borland Application Server (BAS) v4.5
- IBM WebSphere (WAS) v3.5.3
- JBoss v2.4.3
- SilverStream Application Server (SS) v3.7
- Fujitsu INTERSTAGE Application Server (IAS) v4.0

All support the J2EE specification, and all are superficially similar and competitive products. Using the *Stock-Online* test case, the same code is deployed and run upon each of the five products. Each product is then used for a series of test runs with various client loads. All tests are run on the same hardware test

³ www.tpc.org

⁴ <http://jcp.org/aboutJava/communityprocess/review/jsr004/>

environment, with machine configurations stable during all tests. The only variability is the COTS middleware product that is used.

In the test configuration used to gather the example results in Figure 1, three machines were used: one for the application server COTS middleware, one for the client processes and one for the database. The results show the measured application performance in terms of transactions processed per second.

First, it's worth noting that many applications that use application server technology do not require high, sustained throughput levels. For many systems, 10 transactions per second are perfectly adequate, and all these technologies are likely to suffice for such applications. However, the most notable effect is the approximately three to four times performance difference between the best and worst performing application server technologies. This difference in raw performance will dramatically affect the behavior of an application for a high performance application.

Also notable is the amount the performance of each product diminishes as the client load increases. Ideally, a perfectly implemented technology should show minimal decreases in throughput from peak levels when handling large client loads (of course client response time will increase linearly with client load). This would indicate a well-engineered technology with no resource bottlenecks, and one that is capable of withstanding sudden bursts of activity.

In this respect, both WebLogic Server and Borland Application Server come close to the ideal, with less than 10% throughput reduction at 1000 clients. Importantly, the throughput of JBoss, SilverStream and INTERSTAGE is trending downwards quite rapidly with 500 clients. It is unlikely that these products would handle large increases in requests in a reliable manner.

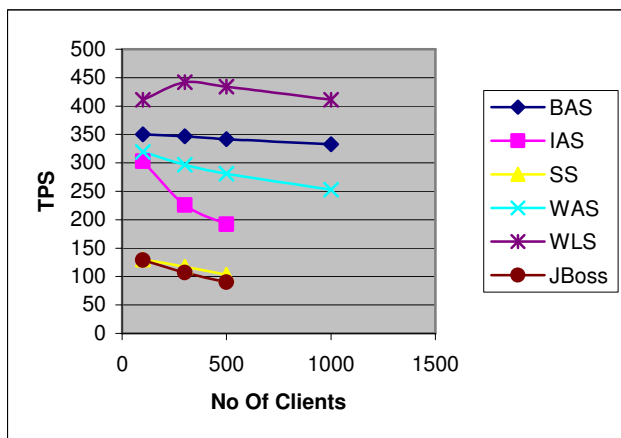


Figure 1 Application Throughput – Single Application Server Machine

It is also instructive to test the scale-out capability of a COTS middleware component. With J2EE technologies, scale-out is supported through a feature known as clustering. Clustering makes it possible to deploy replicated application server components across multiple hosts. Ideally then, adding a new host of equivalent resources should double application a throughput.

In Figure 2, the results of introducing clustering in to the tests using two identical application server machines are depicted. Note

that JBoss does not support clustering in version 2.x, and hence is not represented in these results. Again, WebLogic Server and Borland Application Server stand out in absolute terms, with the Borland technology demonstrating around a 90% performance increase over a single machine. SilverStream also achieves excellent scalability, but still trails in absolute performance. All the other three products are able to exploit the increased hardware resources, with the amount varying between approximately 15% and 50%.

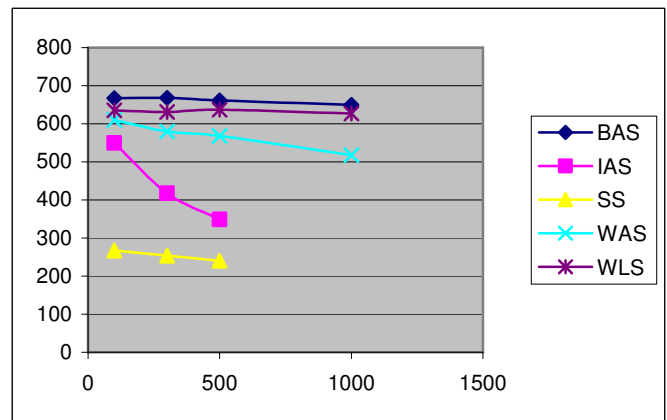


Figure 2 Application Throughput – Dual Application Server Machine

Overall, these results are significant and instructive. They tell IT organizations that are adopting COTS middleware components how each technology is likely to behave in a deployed application. They give technology acquirers a sound, rigorously derived basis on which to make decisions about the capability of a particular product to meet their application's needs.

4. PRACTICAL ISSUES

The MTE project has provided some significant insights in to the difficulties of performing rigorous COTS middleware evaluations. Some of these are discussed below.

Scale: COTS middleware technologies are invariably large and complex products. A typical application server product will comprise around four to six major core components, and numerous other components that offer services useful in some application scenarios. This sheer scale makes it impractical, in terms of time and cost, to quantitatively evaluate all the components in every product. Hence the approach taken has been to focus on carrying out performance analysis of the key, core components, and only qualitative analysis of the common additional components and services. This strategy has proven to be successful in discriminating between the implementation qualities of these technologies. However, CSIRO is constantly asked for more information about the services that have not extensively tested, and are regularly contracted to carry out specific in-depth component evaluations for clients.

Complexity: Scale and complexity do not always correlate in software technologies, but they certainly do in middleware. Middleware products are invariably difficult to get to work well – occasionally they are difficult to get to work at all. Performance

tuning is always complex, as every product offers various 'knobs and dials' that can be set to effect performance and resource usage. This requires product specific knowledge, and extensive experimentation to achieve near-optimal performance. It can also be guaranteed if a product has subtle bugs or behavioral *quirks* that are hard to explain, the stress testing will reveal these. CSIRO consequently often enter in to protracted discussions with the vendors to try to diagnose unexplained performance and behavior. This is usually news to the vendors as well.

Moving Targets: On average, a COTS middleware vendor will release a new major version of their product every 12 to 18 months. This places pressure on the evaluations to keep pace with new releases, as vendors rightly want to see their latest version evaluated. By working closely with vendors, CSIRO are able to get pre-release versions, and test these in a timely fashion. On occasions though, new releases have caught the MTE project by surprise, and necessitated considerable rework to keep evaluations up to date. A further interesting note on this issue is that the performance tests often show new product releases perform slower than older versions. On more than one occasion, the stress tests have brought out significant problems with new releases, and fixes have had to be supplied before testing can continue. Not surprisingly, this type of news is not greeted well by vendors!

Vendor Participation: This level of evaluation work can only succeed with the involvement of the product vendors. After all, it is their technology, they understand it better than anyone, and are able to offer insights in to its behavior that no one else can. The MTE process entails that the product evaluation is released to the vendor when completed, and feedback on this can often lead us to modify tests or qualitative evaluation scores. However, all the vendors only see their relative rankings when the MTE evaluation reports are released. Not surprisingly, some are less than happy, and this can cause problems with the on-going relationship. So far however, no vendor has asked to be removed from future studies. They see value in what CSIRO are doing, and vendors regularly request that their products be included in the MTE studies.

Cost: Since its inception in 1998, the MTE project has recovered about 50% of its costs through commercial sales of evaluation reports, professional seminars and client-specific consulting engagements. This level of cost-recovery satisfies the current demands of the CSIRO environment. Undoubtedly though, undertaking this level of deep technical analysis is a costly proposition. As both the collective experience and the value of the MTE project outcomes increase, then the ratio of cost recovery should grow. However, making a profit is likely to require a somewhat different business and delivery model.

5. IS THIS VIABLE?

In terms of its impact on the software industry, research community and attracting commercial funding, the MTE project has proven a success in its first three years. The aim is to now build upon this success and increase the impact by broadening the technology basis and reaching a wider audience. A number of factors, as described below, could be influential in achieving these aims.

Direct User Involvement: The MTE evaluation reports tend to sell to medium and large IT organizations that are deploying applications using COTS middleware products. The diverse

nature of these organizations' requirements make it impossible for the evaluations to cover all the factors they wish to see addressed. CSIRO would therefore like to establish some mechanism whereby an organization can request that the MTE project test and evaluate some features in more detail, and the resulting intellectual property remains with the MTE project. These 'add-on' evaluations could then be performed cost-effectively for the originator of the request, and the results distributed as a report update via the MTE WWW site. This improves the coverage and our evaluations, and directly satisfies needs generated by actual users.

Vendor Investment: Product vendor investment in this project has been mostly limited to staff resources and software licenses. For most, if not all vendors, this has worked out very economically. For vendors whose products rate well in the reports, the MTE project effort translates in to sales for them. In addition, the feedback we give them on specific features, bugs and overall product quality is undoubtedly valuable. In fact, in some cases it has felt like the project team has been performing system testing for the vendors, which is certainly not desirable from CSIRO's perspective.

For these reasons, the MTE project is seeking to find some sort of arrangement whereby the vendors invest more directly in the MTE evaluation work. Options are direct funding of evaluations, purchase of the resulting intellectual property, or support through staff secondment or additional resources to aid testing.

Independent, Rigorous Testing: 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. This approach 'lets the products speak for themselves'. It produces evaluations that are consequently more reliable and trustworthy than those from other analysts. Maintaining and enforcing this positioning, and the independence of the evaluations, remains critical to future viability. Any compromising of the perceived, never mind the actual independence of the MTE results could fatally harm the project.

6. ACKNOWLEDGMENTS

CSIRO would like to acknowledge the support of the TEST-IT program administered by the Department of Communications, Information Technology and the Arts in Australia.

7. REFERENCES

- [1] Ian Gorton, Anna Liu, P. Tran, The Devil is in the Detail, A Comparison of CORBA Object Transaction Services, the 6th International Conference on Object-Oriented Information Systems, pages 211-221, 18-20 December 2000, London
- [2] Ian Gorton, Anna Liu, Streamlining the Acquisition Process for Large-Scale COTS Middleware Components, International Conference on COTS-based Software Systems, Florida Feb 2002, Springer-Verlag LNCS Series

