

Open Source and the Service Age

By ALAIN LEFEBVRE, Vice-President of Groupe SQLI (alefebvre@sqli.fr)



In the past, customers knew that the computers (boxes!) they bought were not going to run the company. Today – despite wishes to the contrary – they’re realizing that the software they buy is not going to create applications. In the end, customers want results – not just applications that run on computers. How are these results delivered? Through the provision of services. We have seen the hardware and the software ages; we have now entered the service age.

From hardware to service

The hardware age was based on mainframes at the architecture level and on COBOL at the tool level; the former provided a standard infrastructure, while the latter was a standard tool known by everybody. The hardware age also marked the golden age of manufacturers, with IBM dominating the market. The emergence of PCs heralded the software age, which was based on client/ server architecture and

on software packages – a relatively new concept – at the tool level. This time around, software vendors dictated the market (Microsoft and Oracle being two of the better known).

Today, the service age is based on the Web and Internet (architecture/infrastructure) and on Open Source software (tool/technology). In this new service age, service providers – natural integrators of Open Source projects – are the new players to be reckoned with.

The Open Source movement drives change

The fact that Open Source code projects are determining standard technologies indicates that the changes underway run deep, and the Open Source movement is driving these changes. This has already proven true for Internet architecture, and it is now extending to the whole area of software in general. The Simple Object Access Protocol (SOAP) is the latest example confirming this change. Although originally a Microsoft creation, Oracle has adopted the SOAP standard due in large part to its acceptance by the Open Source

(Continued on page 3)

→	INSIDE THE FEBRUARY ISSUE OF TRENDMARKERS	
	Open Source and the Service Age By ALAIN LEFEBVRE	1
	Editorial By JEAN-CHRISTOPHE CIMETIERE	2
	Close-up on JDO By TECHMETRIX RESEARCH	5
	Web Services: SOAP Interoperability By JOHANN DUMSER and JEAN-CHRISTOPHE CIMETIERE	9
	Microsoft SOAP Toolkit 2.0 By JOHANN DUMSER	12

Editorial: A Few Words on Development Practices

By JEAN-CHRISTOPHE CIMETIERE, CEO (jcc@techmetrix.com)

The changing face of application development

From monolithic applications on mainframes using character mode for the user interface, to the Windows event-driven model, there has been a huge change in the way applications are developed. Every aspect has been affected, with the user interface being hit first. Windows brought a very enhanced graphical user interface with a host of new possibilities, and as many potential sources for mistakes in usability design. The client-server programming model brought heterogeneous systems to tie together. And then Internet computing brought us great flexibility in deploying applications, but also more complexity in architecture design.

Ten years ago, a good software developer had to know one language (Cobol or C, say) and usually didn't have to pay attention to its mainframe environment. Five years later, the same person had to learn specific 4GLs (VB, PowerBuilder, Delphi, ...), SQL in general and then specific features for each DBMS (i.e. Oracle PL/SQL or Sybase Transac SQL for stored procedures).

Beside these technical languages, the software developer had to understand the client-server computing model, which was not the easiest phase to deal with. If I refer to my own experience and look back to many of the clients whom we helped choose and design solutions, I remember that for most of our consulting assignments, we had to start with the very basic principles of the client-server computing model, before talking about any product or solution.

Now, we are living in the world of Web applications, whether it be for Internet or intranet usage. The ideal software developer has now to cope with specific languages for the presentation layer (HTML, Style Sheets, XML, XSL and JavaScript), for the business objects layer (Java or any other language) and for the data layer (object/relational mapping, persistence and SQL – again!). In addition, since Web applications are increasingly part of the day-to-day enterprise business, they require integration with existing back-end systems (ERPs, legacy applications, Message Oriented Middleware...).

So, we all have to recognize that the ideal software developer has to be a real superman. Sadly, in reality, such a figure doesn't exist – or else is extremely rare. Instead, skills are spread between many people

using many different products and technologies. In software design, coordinating these different technologies and layers is a tough challenge, meaning that nowadays, the notion of Software Architect really makes sense. But what tools and methods can we use to streamline the architecture design process?

Solutions for enhancing quality and productivity?

Frameworks

The most common strategy for achieving better quality and productivity when developing applications is to create a framework or to extend an existing one. The creation of a framework may seem a long and expensive process as it aims to resolve problems by offering development teams a simple interface. Above all else, documentation and examples are crucial.

"The most profoundly elegant framework will never be reused unless the cost of understanding it and then using its abstractions is lower than the programmer's perceived cost of writing them from scratch."(Grady Booch, *Dr Dobbs Journal*, February 1994.)

The development of a framework is an iterative process which should evolve as and when needed. It requires the presence of application experts who have experience of similar problems. Sophisticated development demands abstraction and good object skills.

If the above description of a framework hasn't put you off and you are still reading, here are a few useful tips we'd like to share with you.

Team Organization

When creating a framework, the technical expertise of the team is only one ingredient for success. Team organization is the real cornerstone. First organize your team: separate roles, and try to give to developers a focus related to their core competencies, then promote and measure.

Promoting should work in two directions. Of course, you have to promote reusable pieces of source code, components, frameworks, and so on; but it is just as important to promote people. A good developer, like most human beings, does appreciate being rewarded. Rewarding people is not just about giving a bonus – it

(Continued on page 16)

(Continued from page 1)

community in the framework of the XML-Apache project. No, you're not imagining things – Oracle has indeed adopted a Microsoft standard! And it is precisely the validation of SOAP by the Open Source community that makes it a standard.

Bear in mind that the Internet is mostly built on elements which themselves are Open Source projects. Software vendors do not dictate the ways and means by which standards are adopted. This is done by independent projects and contributing players (such as Red Hat with Linux). For SOAP to become a true Internet standard, it needed to be released into the Open Source community.

Future changes in the organization of the IT industry will be significant: we will witness a weakening in the position of software vendors, a strong comeback by manufacturers with the right attitude (IBM and VA Linux are leading the way), and the emergence of a new category of service players: Open Source integrators.

Already, certain software vendors offer their products as Open Source Software (OSS) in an effort to ensure better distribution, and this behavior will become commonplace. This new trend is known as "abandonware."

Progress, Ilog, Prolifics and SAP are the latest vendors to adopt Open Source publication for all or part of their products. Yes, even a key player like SAP understands there is much to gain from Open Source. SAP has put its Relational Database Management System (RDBMS) at the disposal of the community of developers. Let us look at SAP more closely and examine why the company decided to enter the world of Open Source.

SAP

In 1998, speculation and rumors ran rife on the RDBMS that SAP could or would buy. Sybase and Informix were most frequently mentioned, as competition between the two vendors was fierce. At the time, SAP/R3 Enterprise Resource Planning (ERP) used Oracle as its primary database. It seemed perfectly normal that SAP would wish to buy a RDBMS vendor in order to offer its own database, rather than depending on Oracle – an ERP vendor itself.

Finally, SAP settled an agreement with Software AG (another German company) for the Adabas DBMS. Under the agreement, SAP has the right to develop

and sell its own Adabas version called SAPdb. Initially, the ramifications of this agreement were obscure. SAP never put much marketing support behind SAPdb, and in a database world dominated by Oracle and DB2 (IBM), it never gained much credibility. In short, the move led to nothing and SAP was back to square one... until in October 2000, when SAP announced that SAPdb was to become an Open Source product.

This change of tactic has two goals: to make a complete product popular and credible in a sector where the competition has not yet provided substantial products, and to share the development costs of SAPdb with the community of developers. This may indeed work, as the RDBMS sector of Open Source is not congested. The MySQL project is the current leader in this category, closely followed by PostgreSQL. MySQL may bring a smile to your face at first glance, but don't be quick to dismiss such projects. They may seem insignificant at first, but they can easily become standards as developers adopt them. After all, that's exactly how Linux rose to prominence.

In addition to these two well-known, well-supported projects (there are specialists in both support and publishing for MySQL and PostgreSQL distributions), we can mention a number of outsiders such as Interbase, which Borland has had hanging around for years. There is a gap in the market for the "Linux" of the RDBMS world, and SAP has set its sights on this.

If SAPdb gains ground in the Open Source world, we can expect Oracle to challenge it on the issue of credibility. In response, SAP will have to push its support for the Open Source development of SAPdb. This could be a blessing: if SAPdb becomes popular more money will be invested in its development, which will spread costs and contribute to its growth. In the end, SAP will have exchanged a few license fees for the kind of popularity and credibility that could generate greater revenue down the line.

IBM

IBM has also understood that the Open Source movement is a key weapon in the battle for market share. Let us look again at the example of the RDBMS market, to illustrate how IBM intends to use the Open Source movement to nibble at Oracle's piece of the pie. Over the past few months IBM has claimed that it is greatly contributing to the development of MySQL. But why?

(Continued on page 4)

(Continued from page 3)

IBM's thinking is as follows: by contributing large portions of DB2 code to MySQL, IBM will influence the foundations of MySQL. Little by little, both the Application Program Interface (API) and storage structures will become compatible with – if not identical to – DB2. Down the road, MySQL users who need an upgrade will be more inclined to opt for DB2 than for Oracle.

The decision facing vendors is clear: either enter the world of OSS and use the service sector to make up for lost revenue from licenses or be progressively marginalized (an easy choice for IBM as services – through IBM Global Services – represent half the company's revenue and its fastest growing, most profitable form of business).

It could be argued that the name "abandonware" indicates a certain degree of hopelessness on the part of vendors that are quickly losing ground. This may be true for some vendors, but SAP and IBM do not see the OSS movement as a last resort.

Service providers, systems integrators move towards Open Source

Even more surprising than the abandonware trend adopted by software vendors is the increasing numbers of service providers that are turning to Open Source. Almost all of them back a particular project, often born from an internal development that has been published: Microstate, Mortbay Consulting, Diamond Technology Partner, Egrail inc, Metatdot inc, Akopia, ArsDigita, Semiotek, and so on. Projects produced by these entities vary from Java application servers to groupware tools and failover management.

Undoubtedly something major is happening in the whole area of Open Source. We are seeing only the first signs, but more and more people are convinced that change will come from this direction. In May 2000, a TechMetrix poll (157 respondents) showed that 54 percent of those questioned believe that growth in the Open Source movement will lead to significant change in the IT industry. The same question was asked in January 2001 – and this time 70 percent of the 130 respondents believe the impact of Open Source will change IT as we know it (see all polls at <http://www.techmetrix.com/scripts/lastpolls.php> 3). There are already more than 250,000 contributors (developers, testers, information officers) worldwide, with the US and Germany most highly represented (<http://www.ibiblio.org/osrt/develop.html>).

As for the likely consequences, we can predict that

Linux will supplant manufacturers' Unix, and all software vendors will have to adopt an Open Source strategy. Therefore we can conclude that, just like in 1996/97 when everybody had to prove that they understood the Internet (and the corresponding strategy), each vendor will soon have to show that they have understood the mechanisms of the Open Source movement and explain how they plan to use it to their advantage. And this phenomenon does not only affect the development community, but the whole software industry.

Linux replaces proprietary Unix

The replacement of proprietary Unix by Linux is a popular trend and has already had some effects. SGI was the first to cross the Rubicon and the others (IBM, HP...) will follow sooner or later. Why should a manufacturer continue to spend money and use resources to maintain an increasingly unattractive version of Unix while Linux support gains ground?

Sun should be the last to hold out as it has two unique features that other leading manufacturers do not: it has the most popular Unix version (Solaris) and it has the most to lose. But apart from this one exception it will be Linux for everybody. And the winner? VA Linux. As this server manufacturer integrated Linux as a system platform from the beginning, it is therefore in the best position to avail of it.

When a major innovation enjoys rapid success, skeptics always say that it won't last long (today they claim that Open Source is limited to system layers and will not produce anything significant outside Linux). Fifteen years ago people said, "FSF (Free Software Foundation) enthusiasts have developed some good demonstrations but nothing serious or useful." The GNU project proved the contrary. Detractors then said, "OK, the GNU toolkit is useful but nonetheless is far from being a complete and operational system." Then Linux appeared. And today these same people claim that there is nothing beyond the system layers. Should we really believe these pessimists, given their track record?

The fact remains that Open Source projects are filling the gaps in more and more market sectors. Even in the demanding space of office suites, Star Office is increasingly considered an alternative to MS Office and GIMP as a competitor for PhotoShop! In a sector that I know well, I see that Java-based application servers available in Open Source are growing in number and gaining in credibility. Layer after layer, progress continues – no area is safe!

(Continued on page 5)

(Continued from page 4)

Conclusion

The Open Source movement is real because companies now see how it can foster their business goals. Today, companies adopting an open source strategy want to:

- ♦ produce better final products
- ♦ reduce development costs
- ♦ gain backdoor access to increased market share
- ♦ generate credibility and popularity
- ♦ sacrifice license fees for future service fees.

At this point, people will say to me "OK, just because developers like using source code projects doesn't necessarily mean that customers – the real users – will adopt them en masse!"

But don't kid yourselves – this has already begun and mainly for two reasons: the quality (and therefore reliability) of Open Source projects is superior to the quality of proprietary projects, and their durability is unequalled. ♦

ABOUT ALAIN LEFEBVRE

Alain Lefebvre is co-founder and vice president of Groupe SQLI, TechMetrix Research's parent company, a total-Internet agency specialized in defining and activating a useful and effective Internet presence for its clients. A major new technology player for over a decade, Groupe SQLI offers full service and continuing coaching to enable companies to move towards profitable total-Internet solutions.

Author of three books dedicated to client/server and intranet, Mr. Lefebvre is also a regular contributor to many specialized publications.

Close-up on JDO: A Standard for Persistence of Java Business Objects

By TECHMETRIX RESEARCH (info@techmetrix.com)

Introduction

In July 2000, Sun came up with a new data access technology which could have a major impact on the way Java applications are created. Standardized under the name JDO (Java Data Objects), this technology will enable real business object modeling to be carried out by automatically managing interaction between objects and resource managers. Traditional development using a specific, non-object access API such as JDBC will now, with JDO, have a standardized alternative.

In this article we shall explain some of the basics of JDO.

Persistence of business objects

In order to implement an enterprise application in Java, it is often necessary to access persistent data, traditionally stored in databases. Today, the most widely used method involves using the JDBC API within the application code, which enables SQL queries to be sent to a database. When data is found in a hierarchical database, mainframe or ERP, it is common practice to use an API specific to the resource manager in question. (By "resource manager" we mean any subsystem

managing access to resources located outside the Java virtual machine, in which applications are executed.) Typically, this covers databases (relational,

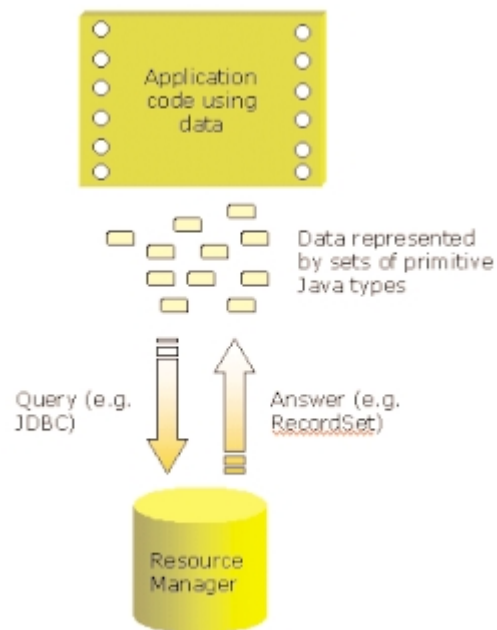


Fig. 1. Classical Data Modeling

(Continued on page 6)

(Continued from page 5)

object or other), ERP or transaction access systems on mainframes.) Data is fed back to the application in the form of character strings and/or simple types (Date, int, etc.) and processed in this format (see fig. 1 on page 4).

While this approach remains the most commonly used today, there is already a growing number of users of the business object approach. This involves defining an object representation of the system data. In practice, it requires programming Java classes to represent the different data found in the resource managers, at the application level. When the application is executed, the data appears in memory in the form of complex Java objects, displaying high-level behavior (fig. 2).

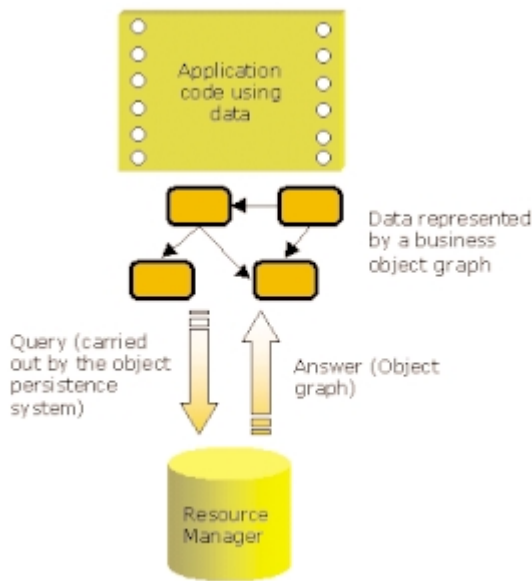


Fig. 2. Object Oriented Data Modeling

This approach makes it possible to enjoy the benefits of object technology for entity modeling: modularity, extensibility, encapsulation, etc. It requires an object persistence manager whose role is to establish correspondence between the business objects in memory and the persistence engine (resource manager). This subsystem is indispensable; it becomes clear that manually managing persistence of each object becomes impractical as soon as the system becomes even slightly complex.

There is an exception to this architecture, however: if the resource manager used is a Java object data-

base, then object persistence is already available, without the need for an additional module. When using a relational database, the object persistence manager will be a so-called "object/relational mapping" module (the best known products in this field include TopLink and CocoBase).

Java Data Objects (JDO)

JDO is a specification developed by a number of industry players, including Sun, within the scope of the Java community process. The aim is to provide a standard API which enables a business objects persistence manager to be used. It therefore concerns both the object/relational mapping tool and the object database or mapping module/ERP. To an extent, it also standardizes the way in which such an object persistence service can be integrated with an application server.

JDO therefore unifies data access techniques, keeping the peculiarities of resource managers masked, to an extent at least, behind the paradigm of business object persistence.

Technically speaking, the two major functionalities underpinning such a technology are as follows:

- ◆Object persistence: JDO automatically manages synchronization of the data represented by the business objects layer and the underlying resource managers.
- ◆Transaction support: transaction support is closely linked with persistence and gives application developers the capacity to manage object graphs transactionally (commit and rollback on the object graph) with automatic management of transactions to be performed on resource managers.

With JDO, application development can be broken down into three stages:

- ◆Business object development
- ◆Configuration of mapping between objects and data manager.
- ◆Creation of application code using business objects.

With a technology like JDO, the developer does not have to directly manipulate native interfaces for

(Continued on page 7)

(Continued from page 6)

access to difference resource managers; the JDO module manages interaction between the object layer and the underlying managers. Above all, the developer can benefit from the advantages of the object-oriented approach in terms of software engineering.

Mapping configuration is an important stage and it can prove complex. It requires an understanding of the operating principles governing the data manager used. For example, in the case of a relational database, for each object field we indicate the columns in the persistence database, specify the way in which inheritance is represented in the database, how the relational integrity constraints should be taken into account in the object layer, and so on. On the other hand, if we use an object database, this stage is almost completely removed because the data model of the object base directly matches the Java object model. It should be noted that JDO does not specify the way in which mapping configuration is performed. This can therefore vary between the different JDO implementations on the market.

Support for business object modeling

Providing support for business object modeling is JDO's number one goal. In this context, JDO offers a relatively transparent persistence model, demonstrated the fact that class developers do not have to write specific code to make objects persist. On execution, persistence by reference is used: an object becomes persistent as soon as it is referenced by another object that is already persistent. Evidently, a bootstrap model is provided for these "root" objects. Some objects cannot be made persistent, however; system objects such as Exception, Thread and Socket, for instance, cannot be made persistent via JDO.

JDO supports complex associations between objects – a fundamental feature of object modeling. In addition, objects are saved to memory incrementally: only objects actually used by the application are saved to memory from the resource manager. This is crucial if acceptable performances are to be obtained.

With a technology like JDO, which enables enterprise data to be represented as objects, the different processes are written in Java. For this to be possible, however, it is necessary to have access to the persistent objects representing the data, in the

programs. In fact, you must at least have access to one object from the graph enabling you to access other objects. This pre-process object retrieval is carried out by a query language defined by JDO. The language can be used, for example, to select all "Employ" objects which meet a particular criterion. The query language in question uses a syntax similar to Java, but imposes significant restrictions (no method call, for example). It expresses conditions on objects, which are automatically translated by JDO in order of selection in the underlying resource managers (for example, an SQL "select" order if a relational database is being used).

Unlike EJB technology, JDO supports fine-grained objects.

Other objectives of JDO

JDO provides for a data access technique which does not require knowledge of a specific access language, as is currently the case for JDBC which requires SQL for data access.

The JDO architecture is designed to simplify development of scalable systems able to bear heavier loads. Likewise, one of the goals of JDO is to simplify development of applications operating in a secure transactional context. Support for ACID transactions lies at the heart of JDO operation; JDO offers coordination and synchronization between the object layer and underlying resource managers, in a transactional context. In this context, it can be said that JDO integrates the functionality of an Object Transaction Manager (OTM).

JDO is designed to be able to operate with a broad range of resource managers: file systems, all types of databases, ERP, mainframe, etc. JDO's designers hope to see many implementations of this technology emerge, offering a vast selection of resource managers or features specific to a field of application or particular environment.

JDO is not intended to be restricted to the enterprise applications present in application servers, but is also aimed at embedded applications and stand-alone applications.

The specification is designed to enable JDO implementations to achieve major optimizations specific to target environments (in particular, specific to

(Continued on page 8)

The Pros	The Cons
<ul style="list-style-type: none"> ◆ Support for complex relationships between objects (association, inheritance). ◆ Support for fine-grained objects. ◆ Uniform persistence mechanisms. 	<ul style="list-style-type: none"> ◆ JDO does not specify how mapping configuration is carried out; this may therefore vary. ◆ No significant JDO implementations as yet.

(Continued from page 7)

resource managers), while giving application developers a unique API.

JDO is conceived so as to be able to integrate with J2EE application servers. This integration is made possible through JDO's use of the J2EE Connector Architecture. What is more, a JDO implementation should be able to be integrated directly within different application servers.

Conclusion

JDO is opening up the way for a real standardization of object/relational mapping tools and object databases. The fact that it is positioned outside the bounds of J2EE is particularly significant; JDO will integrate with J2EE platforms but will not be limited to these alone. What is more, JDO is not mandatory to obtain J2EE certification, so this specification can go well beyond what J2EE editors are able to provide uniformly. In short, JDO is a well thought out specification, designed by object experts and with a bright future ahead of it.

Since the emergence of JDO, the issue of JDO versus EJB has been arising more and more often. JDO seems to be a useful solution for business object modeling. Unlike Entity EJBs, JDO supports complex relationships between objects (association, inheritance), as well as fine-grained objects. JDO provides simple Java objects with persistence and transaction services, thereby making object models easier to develop. Meanwhile, Entity EJBs provide extra services (distribution and security) that are not always useful.

Nonetheless, joint use of EJB Session Beans for security and distribution, with a persistence and transaction framework for simple Java objects like JDO, is a useful option for enterprise projects. It enables certain complex, redundant aspects of the EJB standard to be avoided (JNDI calls,

deployment descriptors on Entity EJBs...) and makes it possible to benefit from the semantics of Java objects to bring a real level of transparency.

In conclusion, we remember that the JDO specification is geared first and foremost towards vendors of application servers, O/R mapping tools, software packages and middleware. For developers and architects within the enterprise, it is still too early to base development of strategic applications on JDO, for two key reasons: JDO implementations (integrated with application servers or standalone) are not ready yet, and there are not any JDO Resource Adaptors available. However, in the quest for O/R Mapping solutions, it is important to consider software vendors' position with regard to JDO.◆

References:

JDO Specification:

<http://java.sun.com/aboutJava/communityprocess/review/jsr012/index.html>

Debate on JDO vs EJB (www.theserverside.com):

http://theserverside.com/discussion/thread.jsp?thead_id=771

Persistence management tools (O/R Mapping Tools) :

TopLink:

<http://www.webgain.com/products/toplink/>

Cocobase:

http://www.thoughtinc.com/cber_index.html

CastorJDO:

<http://castor.exolab.org/jdo.html> - Open Source Solution (Not directly related to JDO, see FAQ <http://castor.exolab.org/jdo-faq.html>)

Web Services: SOAP Interoperability

By JOHANN DUMSER, Junior Consultant (jdumser@techmetrix.net)
and JEAN-CHRISTOPHE CIMETIERE, CEO (jcc@techmetrix.com)

The growing number of SOAP implementations is concrete proof of the real boom being enjoyed, in recent months, by Web Services based on this protocol. And each of the implementations has its own objects, methods, environments and deployment.

In an earlier article we introduced the set of concepts and technologies that are linked with Web Services:

- ◆XML: a technology used to describe information
- ◆UDDI: to find the necessary services
- ◆WSDL: to describe how Web Services work
- ◆SOAP: to remotely execute Web Services.

(For related articles, visit: <http://www.techmetrix.com/trendmarkers/topics/tmktopic.asp.php3>).

We are now entering a much more pragmatic phase, which involves testing and checking the actual interoperability promised by the SOAP protocol.

In this article, we present the results of our study on interoperability between the three main SOAP implementations, which are:

- ◆SOAP Toolkit2.0 Beta 1 (Microsoft): C#, Visual Basic and any other language of the .NET platform
- ◆Apache SOAP (Jakarta Apache project): JAVA language

- ◆SOAP::Lite (Paul Kulchenko): Perl language

These three implementations were tested both in SOAP Client and SOAP Server.

By "SOAP Client" we mean the program that will create the SOAP message, send it and receive the response from the service called. It is important in this context not to confuse a SOAP Client and an Internet Client, which can call on a SOAP Client to call a Web service.

By "SOAP Server" we mean the component to which the SOAP messages are sent. A SOAP Server deserializes SOAP queries from a SOAP Client, provides the service and sends the message back to the SOAP Client.

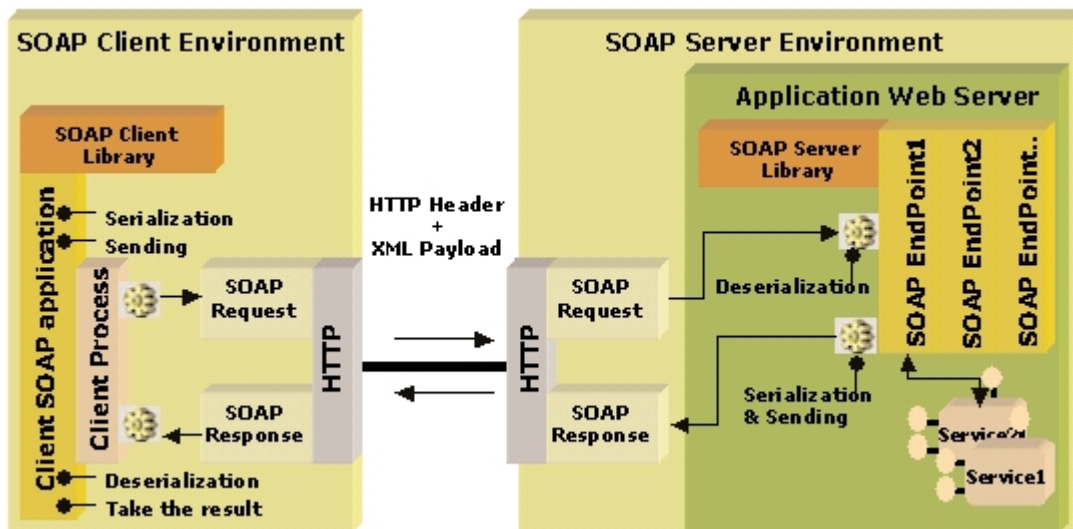
The SOAP implementations that we tested provide both the SOAP Client and Server environment. The diagram below shows a traditional architecture with a SOAP Client calling Web Services on a SOAP Server.

Interoperability tests:

The first table on page 10 shows the interoperability tests carried out.

Server side, we tested Web Services deployed on Internet and locally on our computers. Our SOAP messages were created using simple type parameters (string, integer).

(Continued on page 10)



Communication between the SOAP Client and SOAP Server

Tests on the TechMetrix Platform for SOAP Clients, using SOAP Servers hosted by XMethods and ITFinity				
Soap Client - TechMetrix Platform		Soap Servers: Implementation (hosting location)		
		Apache SOAP (XMethods)	SOAP::Lite (XMethods)	Microsoft SOAP Server Toolkit 2.0 (ITFinity)
Apache SOAP	Medium API	Service [Traffic Info California] => OK	Service: [BabelFish translation] => OK	# Datatype problem
SOAP::Lite	High API	Service [Traffic Info California] => OK	Service: [BabelFish translation] => OK	Service: [Generation of a Globally Unique Identifier] => OK
	Medium API	Service [Traffic Info California] => OK	Service: [BabelFish translation] => OK	Service: [Generation of a Globally Unique Identifier] => OK
MS SOAP Toolkit 2.0 Beta 1	High API	# Datatype problem	Service: [BabelFish translation] => OK	Service: [Generation of a Globally Unique Identifier] => OK
	Low API	# Datatype problem	Service: [BabelFish translation] => OK	Service: [Generation of a Globally Unique Identifier] => OK

(Continued from page 9)

All our tests were performed in a Windows 2000 environment, for SOAP Clients and for SOAP Servers

deployed locally. For Web Services (SOAP Server) hosted by XMethods, the platform used was UNIX (Solaris 8). For Web Services (SOAP Server) hosted by ITFinity, the platform used was Windows 2000.

Tests on the TechMetrix platform (SOAP Clients and Servers, Windows 2000)				
Soap Client - TechMetrix Platform		Soap Servers, TechMetrix Platform: Implementation		
		Apache SOAP (Unit Test Application Server)	SOAP::Lite	Microsoft SOAP Server Toolkit 2.0
Apache SOAP	Medium API	WSDE example [StockQuote] => OK	Not tested	# Datatype problem
SOAP::Lite	High API	WSDE example [StockQuote] => OK	SOAP::Lite examples [incrementation operation] => OK	Toolkit example: [Algebraic operation] => OK
	Medium API	WSDE Example [StockQuote] => OK	Not tested	Toolkit example: [Algebraic operation] => OK
MS SOAP Toolkit 2.0 Beta 1	High API	# Datatype problem	Not tested	Toolkit example: [Algebraic operation] => OK
	Low API	# Datatype problem	Not tested	Toolkit example: [Algebraic operation] => OK

[The services used are described briefly at the end of this article]

(Continued on page 11)

References

API levels:

- ♦High API: Interoperability checked by using high implementation: WSDL call to reach Web service
- ♦Medium API: Interoperability checked by using a medium API: instantiation of key parameters of SOAP message.
- ♦Low API: Interoperability checked by using low implementation: manual construction of SOAP message.
- ♦Not tested: Interoperability test not carried out
- ♦#Datatype problem: Interoperability tested and not checked. In fact, the Apache system requires all the arguments to be typed by using the xsi:type attribute, while the Microsoft system does not require this. The next version of Apache SOAP (2.1) should be able to guarantee interoperability between these two implementations.

Services hosted:

- ♦XMethods: www.xmethods.com
- ♦ITFinity: www.itfinity.com, service referenced with XMethods (www.xmethods.com)

Conclusion

Overall, these tests demonstrate that interoperability between different implementations is making progress. We can see that out of the number of examples tested, we managed to call services using different languages for their SOAP implementation. Only the Apache SOAP and SOAP::Lite implementations here enable UNIX platforms be equipped with the necessary SOAP modules for dialoging with this protocol.

What is more, the introduction of WSDL (see www.tech-metrix.com/trendmarkers/tmk0101/tmk0101-4.php3) in the implementations seems to improve and enhance full interoperability between Web Services. WSDL describes the information required in order to call the services offered, and it is one of the best ways to guarantee interoperability of SOAP services.

These tests reveal excellent potential. Even though the Web Services actually available at the moment are few and far between, and rather basic in nature, professional offerings with more complete solutions are beginning to emerge, such as that offered by Sevina Technologies (www.sevina.com) which features CRM (Customer Relationship Management) services.

Lastly, those wishing to look in more detail at the technical aspects can consult our feature (page 12) on the tests we carried out on the MS SOAP implementation.

Description of services used as examples

Service: Traffic Info California

Description: Information on traffic conditions on California's highways. Information source from the Web site Caltrans.

Input: Number of Californian highway

Output: Info on traffic conditions on highway in question.

WSDL: Yes

SOAP Implementation: Apache SOAP

Service: BabelFish translation

Description: Translates a text up to 5K in size. For more information, see the BabelFish homepage:

www.babelfish.altavista.com/translate/dyn

Inputs: Translation mode ["en_fr", "en_de", "en_it", "en_pt", "en_es", "fr_en", "de_en", "it_en", "pt_en", "ru_en", "es_en"]; string to be translated in source language.

Output: Translation in target language

WSDL: Yes

SOAP Implementation: SOAP::Lite

Service: GUID Generator (Globally Unique Identifier)

Description: Assigns a unique ID. Service delivered by ITFinity

Input: None

Output: UID

WSDL: Yes

SOAP Implementation: MS SOAP 2.0

Service: StockQuote

Description: Information on stock quotes from finance.yahoo.com which draws its resources from Reuters. The example is given by XMLToday.

Input: Symbol of listed company

Output: Company stock quote

WSDL: Yes

SOAP Implementation: Apache SOAP

Service: Incrementation operation

Description: Adds 1 to the integer entered

Input: Integer

Output: Integer+1

WSDL: No

SOAP Implementation: SOAP::Lite

Service: Algebraic operations

Description: Carries out algebraic operations on 2 algebraic numbers. Example from SOAP Toolkit 2.0

Input: Method ["AddNumbers", "SubtractNumbers"]; Number1; Number2

Output: Result of operation

WSDL: Yes

SOAP Implementation: MS SOAP2.0 ♦

Microsoft SOAP Toolkit 2.0 Beta 1

By JOHANN DUMSER, Junior Consultant (jdumser@techmetrix.net)

An ASP-based SOAP Implementation

in Windows 2000 and NT4.0 Sp6

On December 19, 2000, Microsoft published a new version of its SOAP Toolkit (2.0 Beta 1), supporting WSDL. This latest version SOAP Toolkit provides:

- ◆client-side components for invoking Web Services described by a WSDL document
- ◆server-side components which use the WSDL and WSML (Web Service Meta Language) documents residing on the server to perform mapping between the Web Services invoked and the COM object methods.
- ◆components for deserializing (breaking down), transmitting and reserializing (putting back together) SOAP messages.

SOAP Toolkit 2.0 Beta 1 makes it possible to call services and publish them according to two Application Program Interface (API) levels: high or low. The choice will depend on the characteristics of the SOAP message that you wish to send or the level of monitoring of the service to be called.

The toolkit also provides a means to generate WSDL from the services described in a COM object. You give the service endpoint, the location of the related DLL, the name of the WSDL file that you wish to build... and the wizard does the rest. For unrecognized parameter types, it sends back '?????' and you will then need to specify the types yourself.

As the old version of SOAP Toolkit for Visual Basic 6.0 used proprietary format SDL files, Microsoft was not making any progress in boosting interoperability of Web Services. But by releasing this new version of the toolkit, Microsoft is changing its strategy and taking a step towards standardization or, at least, interoperability of implementations. Our tests revealed a fair level of interoperability, although some problems were encountered with the Apache implementation.

The purpose of this document is to explain precisely how Microsoft's SOAP Toolkit is used.

Platforms supported

Client and server objects run in a Windows environment.

- ◆SOAP client objects run in Windows 98, ME, NT4.0 and 2000
- ◆SOAP server objects run in Active Server Pages

When the Toolkit is installed, the XML parser (MSXML3.0) is saved in the system. New DLLs are also registered if you have been using the old version of SOAP Toolkit released in November.

The SOAP Toolkit features two approaches for using SOAP: a Usage approach (High API) and a development approach (Low API)

Usage approach: High API

This approach makes programming easy for developers by keeping all the technical details hidden.

Client side: Calling a Web Service with high API

The following example features a Web Service available on <http://www.xmethods.com> and performs a ping from the Xmethods server to a host using the WSDL of the related service. First of all, the client application instantiates a SoapClient object which is implemented in msoap1.dll:

```
set soapclient =  
CreateObject("MSSOAP.Soap Client")
```

Once instantiation is complete, the SOAP object can call the msoapinit() method. This method has three parameters: the URI of the WSDL file, the name of the service (same name as the SERVICE tag in the WSDL) and the port name (the same name as the PORT tag in the WSDL).

```
Call soapclient.msoapinit  
("http://www.xmethods.net/sd/PingService  
.wsdl", "PingService", "PingPort")
```

When this call is completed, all the WSDL methods become available (the names are those of the OPERATION tag in the WSDL), and all that remains to be done is to choose the required method(s).

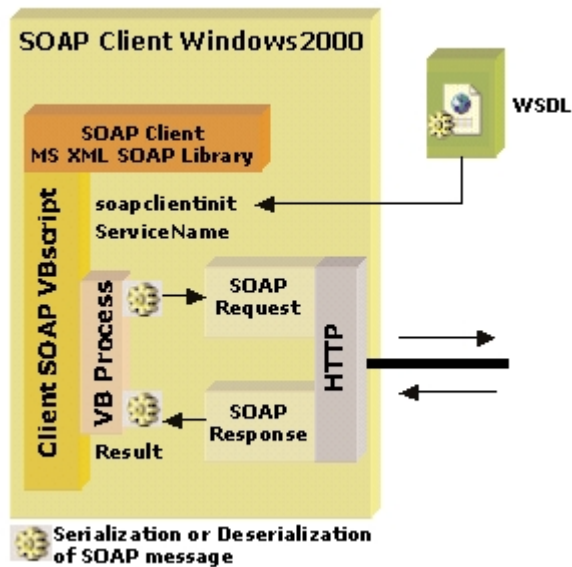
```
wscript.echo soapclient.pingHost  
("www.yahoo.fr")
```

Server side: Offering a Web Service with high API

To illustrate the server side of the high API, we have directly reproduced an example from Microsoft's SOAP Toolkit 2.0 Beta1, included in the download

(Continued on page 13)

(Continued from page 12)



High-API SOAP communication: client side

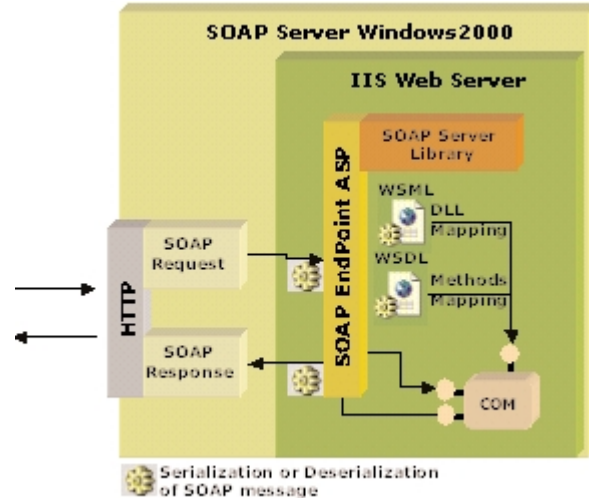
file. It shows the standard server-side model for processing a SOAP message requiring WSDL for execution. MS SOAP Toolkit 2.0 is based on ASPs. It starts by invoking a SoapServer object which creates an instance of a COM object defined in the WSMML document (this document can be generated by a Toolkit, which requests the COM object in which the services are located, the URL of the SOAP server and the name of the required document).

The SoapServer object decodes the parameters of the SOAP request, and uses them to call the right methods. It retrieves the result of the execution, places the result in a SOAP message and sends it back to the client.

The ASP scripts below correspond to the SOAP Endpoint ASP scripts on the server side.

```
<%@ LANGUAGE = VBScript %>
<% Response.ContentType = "text/xml" %>
<%
set soapserver =
CreateObject("MSSOAP.SoapServer")
wsdl = Server.MapPath("Sample.wsdl")
wsml = Server.MapPath("Sample.wsml")
call soapserver.init(wsdl, wsml)
call soapserver.SoapInvoke(request,
response)
%>
```

By using a high API you can send and receive SOAP information without having to write code. The Toolkit uses a WSDL file as a method to reach the requested service. The advantage is to be able to map an existing COM object by making it SOAP-enabled.



High-API SOAP Communication: server side

Development Approach: Low API

This approach exposes the technical details of the SOAP message. The Toolkit offers several interfaces for low-level transactions. These low-level interfaces enable the client to generate SOAP messages, to send them to the server and to process the responses.

Client side: Calling a Web Service with low API

The different objects used to execute SOAP messages are as follows:

- ◆ SoapConnector – sends and receives SOAP messages
- ◆ SoapSerializer – builds the SOAP message and sends it to the server
- ◆ SoapReader – reads the response from the server

Let us use the same service to be called as for the high API (XMethods Ping). First of all, the Soap-Connector object is created which manages the sending and receiving of SOAP messages. The EndPoint property is then instantiated with the service address. To initialize and prepare the connection, you can call the Connect method. The SoapConnector's SoapAction property provides the value required for the SOAPAction field in the SOAP header.

```
Connector.Property("EndPointURL") =
"http://services.xmethods.net:80/perl/so
aplite.cgi"
Connector.Connect Nothing
Connector.Property("SoapAction") =
"urn:xmethodsSoapPing#pingHost"
The BeginMessage method of the
SoapConnector is then ready to send the
SOAP message.
```

(Continued on page 14)

(Continued from page 13)

The SoapSerializer object is used to manually create the message. This creation process is very intuitive and involves creating the payload step by step. The first step is to create the envelope, followed by the body, then an element for which we define:

- ♦the name of the service that will use this element
- ♦URI, means of coding
- ♦tag prefix.

Next, the name of the element and its value must be indicated. All the Serializer properties have end properties, which means that corresponding to Serializer.startEnvelope is Serializer.endEnvelope. This structure corresponds to the generation of an XML payload.

The EndMessage method of the Connector indicates the end of the SOAP message to be sent to the server.

```
Serializer.Init Connector.InputStream

Serializer.startEnvelope
  Serializer.startBody
    Serializer.startElement "pingHost",
    "urn:xmethodsSoapPing", , "namesp01"
    Serializer.startElement "hostname"
    Serializer.writeString
    CStr("www.yahoo.com")
    Serializer.endElement
    Serializer.endElement
    Serializer.endBody
  Serializer.endEnvelope

Connector.EndMessage
```

The server executes the requested operation and sends back the SOAP message. The returned Response contains either the result of the operation, or a Fault element describing the error. The Response object can be read with the SoapReader object.

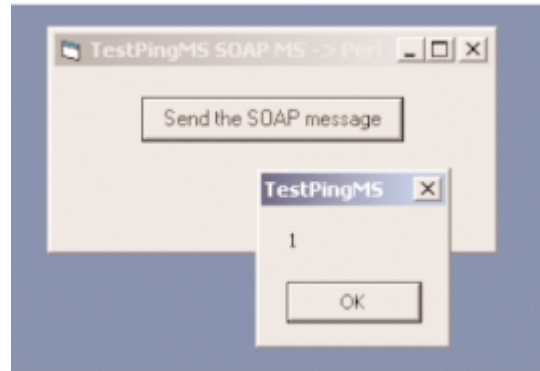
The Reader.Load Connector.OutputStream object parses the Document Object Model (DOM) tree and reads the OutputStream message returned. The result can then be displayed.

```
Reader.Load Connector.OutputStream
MsgBox Reader.RPCResult.Text
```

Server side: Offering a Web Service with low API

The different elements used to process SOAP messages are as follows:

- ♦ SoapReader – reads the incoming SOAP message request and builds a DOM tree



SOAP Interoperability: SOAP Client (MSSOAP) – SOAP Server (SOAP ::Lite hosted by Xmethods)

- ♦ SoapSerializer – builds the SOAP message containing the result of the operation and sends it to the client

This example, taken from those provided in Microsoft's SOAP Toolkit 2.0 Beta1, sends 'Hello' or 'Bye Bye' depending on the name of the requested method. The endpoint here is therefore an ASP which manages client requests. Each time the server receives a SOAP request, it instantiates a COM object. Then the server invokes the Process method of this object. Its two parameters are the incoming and outgoing SOAP messages. The Endpoint ASP script is as follows:

```
<%
Set SpeechSrv =
Server.CreateObject("TestHello.SpeechSrv")
SpeechSrv.Process Request, Response
%>
```

On the server side, the process in the COM object involves three steps: retrieving the information, performing the operation, and sending the response.

The following VB code for the COM object gives the key elements required to carry out the operation.

```
` Process function which performs the
operation
Public Sub Process(ByVal Request As
ASPTypelibrary.Request, _
ByVal Response As
ASPTypelibrary.Response)

` Load request in Reader object
Reader.Load Request

` Retrieve required information
` Base name - with RPCStruc.baseName
property
MethodName = Reader.RPCStruct.baseName

` Value of element A - with property
```

(Continued on page 15)

(Continued from page 14)

```

RPCStruc.selectSingleNode("Tag_Name").text
xt
A =
CDBl(Reader.RPCStruct.selectSingleNode("
A").text)

` Operation
If (MethodName = "Morning") Then
Answer = "Hello!"
Else
Answer = "Bye Bye!"
End If

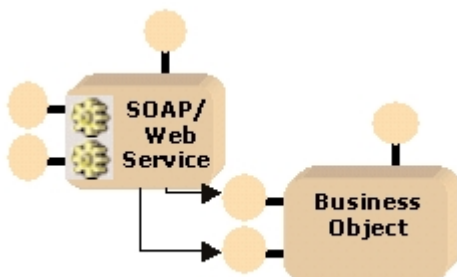
` Construction of response header
Response.ContentType = "text/xml"

` Construction of response XML payload
Serializer.Init Response
Serializer.startEnvelope
  Serializer.startBody
    Serializer.startElement MethodName &
"Response", "uri:Speech"
      Serializer.startElement "Answer",
"uri:Speech"
        Serializer.writeString Answer
      Serializer.endElement
    Serializer.endBody
  Serializer.endEnvelope

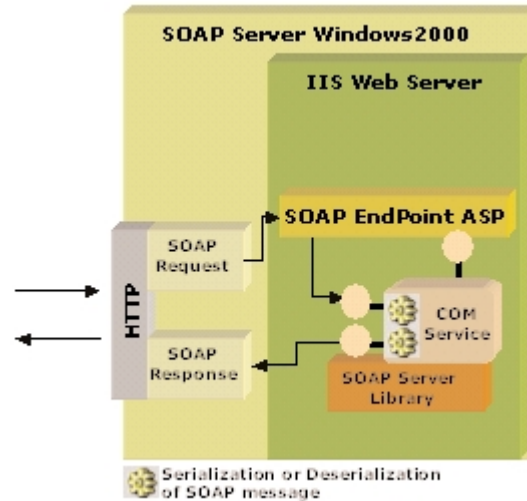
```

The difference between the low API (illustrated opposite) and the high API is that in the former case, processing of incoming and outgoing messages are directly processed by the COM object. For a High API, the SOAP Server automatically takes care of these operations.

The advantage of this programming method is that it makes it possible to control the message generator very precisely, and allows customized error management to be developed. The major disadvantage is that such COM objects are solely designed to be called via SOAP. So when using this method, it would be wise to externalize all the business logic to another COM object and to keep only the SOAP processes in the object presented as a Web Service, as the following diagram illustrates:



Recommended use with COM objects in low API



Low API SOAP communication: server side

Conclusion

Microsoft's SOAP Toolkit 2.0 Beta 1 enables us to call or create SOAP services using two API levels.

The high API enables the client to make simple calls using the WSDL of the service called. Interoperability is not 100 percent, however, as these clients cannot as yet use services developed with the Apache SOAP implementation, which requires the type of the parameters sent.

Those wishing to offer services can do so via an ASP and a COM object. A WSMIL file makes the connection between the WSDL and the service itself, which is described in a COM object. Mapping is carried out in the ASP using methods contained in the new libraries.

The low API enables the client to control the message independently (or not) from the service described by the WSDL. Server side, the implementation makes it possible to offer services and monitor messages. This low API means that developers can process messages with the methods made available. The advantage is therefore that flows can be controlled and errors managed.

Microsoft is carving itself out a good position by offering an easily accessed, well detailed product for calling Web Services using SOAP. The examples given by Microsoft on this new toolkit include insertion of SOAP in Office applications such as Excel, so Office users can now have the current Euro-Dollar exchange rate in their Excel functions, for example. However, few Web Services except those on Microsoft's www.gotdotnet.com site are currently deployed on the Web with this new technology. Nevertheless, the chances are that the advent of .Net will see a whole plethora of Web Services blossom on the Internet. ♦

also means giving them the opportunity to be recognized as experts, making them feel important within your organization. Promoting people through their level of expertise is also a very good tool from the Human Resources point of view. In our business, keeping good developers on board is a hard and expensive task. Developing that kind of personal reward and recognition will help to motivate your best employees.

Measuring the reusability level of a framework or a piece of code has always been challenging, and no tool can solve this issue, since this information is hard to collect.

Tools

Initiatives for tooling reusability emerge regularly in the software industry. A new one has just shown up in its second version: Flashline Component Manger Enterprise Edition (CMEE).

Originally proposed as a hosted service (version 1), Flashline now offers its CMEE as a product you can install in-house. Flashline's CMEE offers an infrastructure and end-user tools to organize, promote and measure component reusability within your enterprise.

CMEE and Flashline associated services are focused on Java development. Even though it can accept any other component in its repository, you will really benefit from the features if your developments are Java-based. The client-side application of CMEE (a Java application) integrates with most of the popular Java IDEs such as WebGain.

While still a young product, CMEE doesn't yet have a long track record to prove the efficiency of reusability, but it introduces some very interesting concepts. First, the toolset is not only a static product: it allows interaction with other Flashline

services such as Code Testing/Profiling, Component Certification and the Component Marketplace. Incidentally, this unique aspect explains the price tag, which stands at \$60,000 for an unlimited server license and then \$150 per user per month.

What is more, the system allows you to track a lot of information, such as the keywords searched, real usage of a component (extraction, under testing, deployment and rejection). Another feature is that CMEE is designed to leverage people. As Charles Stack, President and CEO of Flashline, puts it: "One of the design goals of CMEE, is to leverage 'Star Programmers.'" This tool may facilitate the creation of a real community among the developers within your enterprise. And finally, the way this product is designed enables non-intrusive integration with the company's existing practices and toolsets.

Conclusion

The era of monolithic and independent applications is over. Development should be looked at in terms of maintenance, scalability and communication.

Development via frameworks follows this logic and the structure and experience encapsulated in the framework may be reused in similar developments. This is merely the first step on the road to the industrialization of development.

Capitalizing on knowledge is a very important asset in the enterprise and frameworks can provide excellent, concrete means of doing so. In coming issues of TrendMarkers we will continue to share with you both our vision of how Web application development should be done, and the feedback we've got from the real-world projects that TechMetrix's parent company, SQLI (www.sqli.com), is delivering to its clients. ♦



TechMetrix Research
6 New England Executive Park, # 400
Burlington, MA, 01803 ♦ Phone/Fax: 781-270-7486/87

Assessments and conclusions rendered by TechMetrix Research are proprietary. TechMetrix Research and/or TechMetrix Research analysts cannot be held liable for any damages directly or indirectly caused by decisions made using any TechMetrix Research material.

Names appearing in this document that are registered trademarks are not mentioned as being so, nor is the trademark symbol inserted with each mention of these registered trademarks. This document uses these trademarks for editorial purposes only. In no way does TechMetrix Research have the intention of infringing on any registered trademark mentioned in editorial.