MDA Journal

December 2003

David S. Frankel
David Frankel Consulting

df@DavidFrankelConsulting.com

When the concept of web services first emerged, I remember reading an article in a popular IT trade journal stating that the 20 million or so programmers in the world will have to retool in order to be able to code applications using SOAP, WSDL, and UDDI. I cringed when I thought of legions of developers explicitly hand coding WSDL, SOAP, and UDDI wiring in their application programs.

Since then, service-oriented architecture (SOA) built around Web services is gaining increasing traction in B2B and A2A integration. Thankfully, many savvy IT people are realizing that the mechanical, technical artifacts of SOA-based systems are the sorts of things that computers should produce rather than people. The SOA approach automates labor-intensive work, makes it easier to apply a consistent wiring architecture throughout a system, and makes it easier to change the wiring technique as technologies evolve.

We see many web service tools moving in this positive direction. Some tools generate much, if not all, of the XML and Java code that implements this technical wiring. Better still, some encapsulate much of this capability in middleware, and generate the bindings between the middleware and applications.

Most tools of this nature provide wizards where developers can make design decisions about the wiring. A model-driven approach goes one step farther, by capturing these design decisions in a model that persists and that can be reviewed, and possibly altered, at various stages in the life cycle of a system.

Mike Rosen is among those who have embarked on this path. Mike is a highly skilled enterprise architect with vast experience in building scalable distributed systems. In this month's MDA Journal, Mike takes us on a tour of the technologies that underpin SOA, and provides an overview of how to apply MDA in a SOA context. It's food for thought.

See you after the New Year …
David Frankel

MDA Journal

December 2003

Michael Rosen
CTO
M2VP, Inc.

Contents

MDA, SOA and Technology Convergence

Introduction

Today, we are hearing a lot about Service Oriented Architecture (SOA) and how it will enable the agile enterprise. These marketing claims are based on a convenient convergence of new technologies, namely Web services and Business Process Management (BPM), which together have the potential to improve the agility of enterprise applications. Unfortunately, the half-life of these technologies is extremely short. So how can we take advantage of what they offer without buying ourselves an incompatibility and upgrading nightmare? We will show how Model Driven Architecture (MDA) provides a safe path to both the adoption of these technologies, and to the efficient implementation of SOA based on them.

WebServices

What is a Web service? There have been many different definitions proposed, but when explaining them to my customers, I like to use: “A Web service is a software construct that exposes business functionality using Internet-related technologies.” Most definitions focus on the lower level protocols of Web services, specifically the use of SOAP (Simple Object Access Protocol). While this appears to be a common ingredient of all definitions, I think it misses the point of what’s really important about Web services. My definition stresses two things. One is that Web services employ common Internet technologies; the business implications being that standards-based, commodity infrastructure can be used. But more important is the fact that Web services are used to expose valuable business functionality. In other words, what Web services can do for us is more important than how they do it. Let’s examine the definition a little more closely.

In the context of a Web service, “exposing business functionality” means:

- Identifying valuable business processes within the enterprise
- Defining loosely-coupled, service-oriented interfaces to those processes

And, in this context, “using Internet-related technologies” means:

- Describing those interfaces in a Web-based, industry-standard format
- Publishing a description of the interface so that it can be discovered and used
- Accepting requests and returning replies
- Bridging between the outward-facing interface (accessible via standard Web protocols) and the internal implementation of the service (typically using standard as well as proprietary enterprise protocols)

Although from a business standpoint, the technical aspects are not as important, we must still understand them in order to implement a Web service. To publish a Web service requires the interface to be expressed in an industry-standard,

XML-based format called Web Service Description Language (WSDL). WSDL supports a complete description of the operations available and the parameters required to use those business services. Once a Web service is defined, its description is published in a repository that conforms to the Universal Description, Discovery and Integration (UDDI) specification. Clients can query the repository in a variety of ways to discover an appropriate service. For example, a client might request a service that is known by name (white pages style), or the client might search for a service based on a category (yellow pages style).

Accepting requests and returning replies requires that the request be formatted as a message according to the Simple Object Access Protocol (SOAP), which is a modular protocol built on top of XML. On the Internet, SOAP messages are sent using HTTP or HTTP/S (although SOAP is actually protocol-independent). SOAP's XML foundation is important because XML provides an extensible mechanism for creating self-describing messages and content. This extension capability allows for a loosely coupled relationship between sender and receiver, which is especially important over the Internet where two parties may be in different organizations or enterprises. In addition, XML is represented in ASCII text, which can easily pass through corporate firewalls over the HTTP protocol.

Finally, bridging between the Web service interface and the internal implementation of the service means that the SOAP request is received by a run-time component that accepts the SOAP message, extracts the XML message body, transforms the XML message into a native protocol, and delegates the request to the actual business process (or software component) within the enterprise. These run-time extraction and transformation capabilities may be hosted within a *Web services container*, which provides scalability, load balancing, and other enterprise qualities-of-service for the Web service itself.

Figure 1 illustrates a basic Web services architecture that addresses the requirements implied by the Web services definition described above.

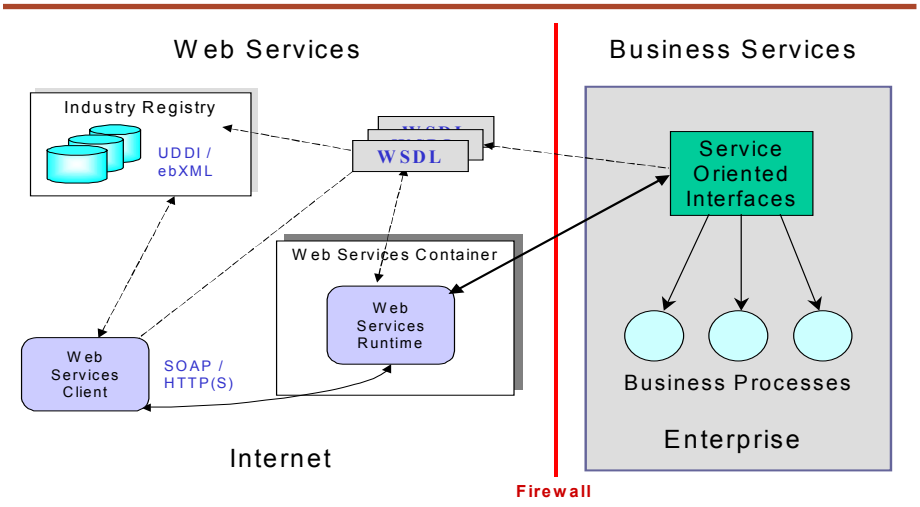

Figure 1 - Basic Web Service Architecture

Although Web services are similar in many ways to other distributed software constructs, the characteristics of Web service usage are very different. Among other things, the designer must deal specifically with the network latency and overhead of Web service invocations, which can be orders of magnitude greater than that of LAN based systems. So, whereas the basic architecture is sufficient for connectivity, a different client interaction model is required for efficient use of these technologies. One essential technique is to increase the granularity of Web service requests.

This can be achieved by defining Web service interfaces that enable business functions to take place via a single exchange of messages. We can create these higher-level business services by combining a set of more primitive or fundamental business processes. One of the major advantages of this approach is that we can create many different services from the same set of primitive functions. Business services can use an XML document-based approach for organizing data into a single request message. The document contains all of the information required for processing the business service. Because it is formatted in XML, the document's structure is self-describing, which makes it easy for subsets of information to be extracted for different processing steps. Since these documents can contain a great deal of data, document processing can be complex. A superb paradigm for both document processing and business service construction is using a business process model as shown in Figure 2. The different information that makes up the request document corresponds to the different steps in a business service, which is defined by a Business Process Model. The process model expresses a sequence of steps, conditional branches, transformations, and so on. A business process execution engine (not pictured) invokes the steps in the model.

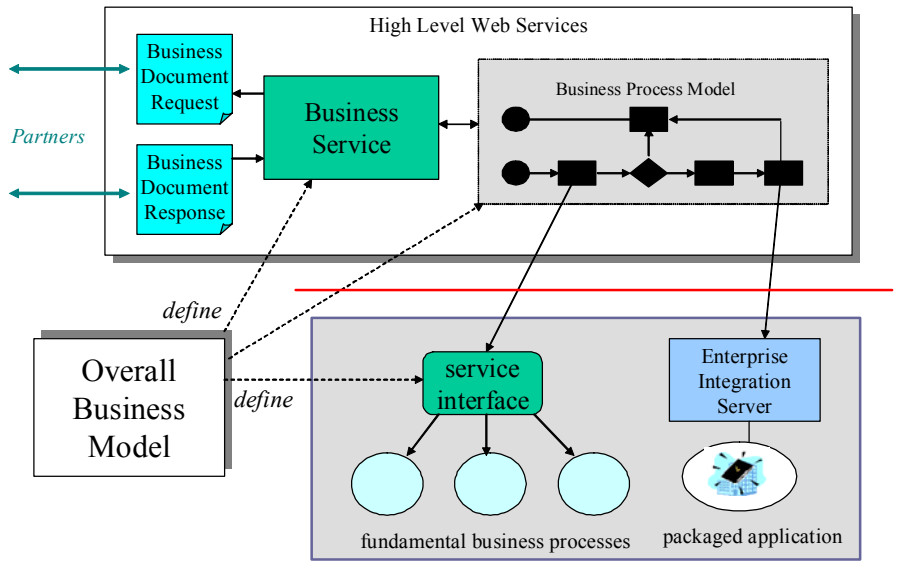

Figure 2 - Business Document based Web Service

Steps within the model can invoke fundamental/primitive business processes. Steps may also invoke processes within packaged applications, such as an ERP system, typically through an enterprise integration server or through specialized adapters.

Information resulting from the request is then formatted into the response business document and sent back to the partner. Note the critical role of the business process model in the implementation of the business services. Also notice that the business process model, the business service definition, the lower-level service interface definitions, and the relationships among them are all part of the overall business model. The overall business model also defines the information view of the enterprise.

Business process models can be used at several different levels within the enterprise to compose higher-level services, so it is important to distinguish between the different kinds of services and processes. We use the following definitions throughout this article:

- *Service* – A specific style of providing functionality based on a 'service contract' as expressed in the service interface. A service is also normally characterized by dynamic discovery and loose coupling.
- *Process* – A process is also a kind of business functionality. The interaction style with a process is less strictly defined than that of a service.
- *Fundamental business process or service* – the lowest level of business functionality within the system. Fundamental processes are frequently provided by a legacy or packaged application.
- *Business Service* – A business service offers a high granularity of business value. It is composed of several lower-level or finer-grained processes and services (as illustrated in Figure 2).
- *Business Process Model* – A model used to define the execution and composition of higher level services from lower level process and services. Both business services and enterprise business processes are defined by business process models.

BPM

In his September 2003 *MDA Journal* article "Software Industrialization and the New IT", David Frankel makes an argument for the importance of Business Process Management (BPM) in future IT systems. The pages of this site are filled with valuable information on the many different aspects of BPM, which goes well beyond process models and beyond IT considerations. However, in this article, we will address only those aspects related to Web services and the implementation of the processing steps described above.

With Web services, we typically talk about the "orchestration" of different services. Web services orchestration is just business process automation applied to the particular environment and technologies of Web services. In other words, if we want to expose a business service as a Web service, we can use a Web service orchestration mechanism to define and execute the business process model.

Some initial approaches to orchestration were proposed in the XLANG specification put forth by Microsoft and the Web Services Flow Language (WSFL) sponsored by IBM. Today, the concepts have been expanded to include the possible transactional coordination of Web services within an orchestration and are expressed in the following proposed specifications.

BPEL4WS

Business Process Execution Language for Web Services (BPEL4WS) is an XML-based language that describes how business processes interact. A business process describes the flow of tasks, the order in which they need to be performed, the type of data shared and how other partners are involved. BPEL4WS allows companies to describe business processes that include multiple Web services and to standardize message exchange internally and between partners. To paraphrase the specification published jointly by IBM, Microsoft, BEA, SAP, and Seibel:

> "…Business Process Execution Language for Web Services (BPEL4WS) is a notation for specifying business process behavior. Processes in BPEL4WS export and import functionality by using Web Service interfaces exclusively. BPEL4WS is intended to describe business processes in two ways. Executable business processes describe the actual behavior of participants in a business interaction. Business protocols, or abstract processes, describe the mutually visible message exchange of the parties involved without revealing their internal behavior. BPEL4WS provides a language for the formal specification of business processes and business."

Once the business process and the connections with customers, partners and internal entities are defined using BPEL4WS, the next step is to coordinate the various activities that occur within a business process, in the right order and at the right time. Two additional specifications provide solutions for this problem. WS-Coordination and WS-Transaction complement BPEL4WS by providing a way for companies to coordinate and integrate a number of distinct Web services and business processes, consistently and reliably, across a variety of implementation environments.

WS-Coordination

WS-Coordination provides a way to manage the operations related to a business activity. A business process may involve a number of Web services working together. Each service needs to be able to coordinate its activities with those of other services for the overall process to succeed. Coordination involves the sequencing of operations to reach an agreement on the overall outcome of the business process.

WS-Coordination provides the structure under which this coordination can take place. The specification supplies standard mechanisms to create processes and register them with transaction protocols that coordinate the execution of distributed operations in a Web services environment.

WS-Transaction

WS-Transaction allows businesses to monitor the success or failure of each

specific, coordinated activity in a business process. It provides businesses with a flexible transaction protocol to help enable consistent and reliable operations across distributed organizations in a Web services environment. The specification also allows the business process to react to faults detected during execution. WS-Transaction provides support for transactions that encompass short- and long-running business processes. In both cases, WS-Transaction takes advantage of the structure WS-Coordination provides to enable all participating Web services to end the business process with a shared understanding of its outcome.

Several products have already announced support for BPEL4WS including: IBM BPEL4WS Editor and Java Run-Time, Microsoft BizTalk Server, Collaxa BPEL Orchestration Server, and Momentum Software ChoreoServer for ISVs. Clearly a cast of heavy hitters agrees about the importance of this and we can expect them to incorporate more capabilities of BPM over time.

SOA

We've stated that Web services and business process orchestration are being brought together and used as building blocks for Service Oriented Architecture (SOA). But, what exactly is SOA? Simply put, SOA describes how to build systems composed of services. A SOA enables different groups, organizations, or enterprises to independently build services that can be combined in both anticipated and unanticipated ways to provide higher levels of business functionality within the context of the enterprise. In this way, SOA helps to make enterprises more agile.

SOA requires more than just a Web service infrastructure. A SOA should describe the following aspects of services within an enterprise:

- The granularity and types of services
- How services are constructed
- How existing packaged and legacy systems are integrated into the service environment
- How services are combined together
- How services communicate at a technical level (i.e., how they connect to each other and pass information)
- How services interoperate at a semantic level (i.e. how they share common meanings for that information)

Figure 3 illustrates these aspects of a Web service based SOA. In the middle of the diagram is the service bus, the technical infrastructure that is used to provide connectivity and messaging between services. We use Web services technologies to implement this bus. In the top middle, we see a business service connected to the service bus, as well as the business process model that defines the execution of the services. The service is connected to the bus using a Web service layer to provide the technical connectivity. Multiple business services are connected to the bus. One important special case is that of supporting legacy systems and packaged applications. We explicitly single out this type

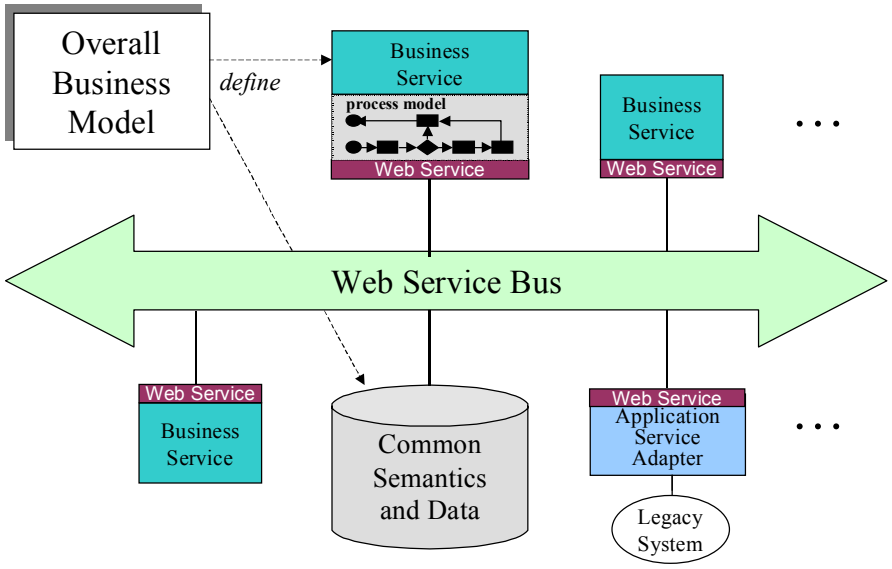

Figure 3 - Web Services Based SOA

of service because of the importance of existing applications within the enterprise. Again, we use a Web service layer to connect it to the service bus, and an application adapter to go between the Web service and the system itself (though many package applications now provide native Web service interfaces.)

At the bottom middle we see common semantics and metadata. This generally consists of several parts: the shared semantic definitions, mappings to other internal data models, and connectivity to the shared data. The common semantics are defined by the business model. The mappings and connectivity implement them operationally. An Operational Data Store (ODS) is a typical mechanism for connectivity to the shared information.

Last, but not least, at the top left is the overall business model. As in the previous example, the business model provides the definition of processes, services, data and semantics throughout the enterprise. The business model is critical to creating a set of services that can actually work together to provide higher-level functions. You can implement a SOA without an overall business model, but what you will end up with is just a pile of incompatible services, rather than enabling an agile, flexible enterprise.

SOAs are not new. There are prominent examples of enterprises that have successfully implemented them and that have realized numerous advantages including reduced costs and improved competitiveness. Some of the world's most successful applications are based on SOA, built on mature infrastructure technologies such as CORBA.

Unfortunately, these examples are few and far between because implementing a successful SOA is extremely difficult. In the past, few organizations had the skill to overcome the architectural, technical, business and organizational

challenges required. And while many of those challenges still exist, the technologies we have been discussing are converging to make SOA more obtainable to enterprises staffed by 'mere mortals'. For example:

- Web services provide the infrastructure to connect services together, even if these services are based on incompatible technologies. Web services have several advantages over previous technologies:
  - Web services naturally provide service-based interfaces
  - Web services are an interfacing technology designed to bridge between an external service interface and the internal implementation
  - Web services are standards-based, platform-independent, and supported by all major players.

- BPEL4WS is explicitly designed to work with Web services and provide coordination and integration of business services into higher-level business processes. Business process capabilities allow organizations to think about and implement IT systems in business terms and to correctly identify business services.

Let's review Web service-based SOA for a moment. The technologies required include:

- XML, including basic XML 1.0 (second edition) and XML Schemas 1.0
- The Simple Object Access Protocol(SOAP), which defines how to use XML as an application-to-application protocol. The most commonly used version of SOAP is V1.1
- Web Services Description Language (WSDL), which is a specific type of XML Schema used to describe Web service messages, operations, and protocol mappings. The current version of WSDL is V1.1.
- Universal Description, Discovery, and Integration (UDDI), which defines a standard repository for registering and discovering Web service descriptions. The current versions for interoperability are UDDI API V2.04, UDDI Data Representation V2.03, UDDI XML Schema V2.0
  (The above bullets are based on the WS-I basic profile V1.0)
- BPEL4WS, V1.1
- WS-Coordination, draft specification, September 2003
- WS-Transactions, draft specification, August 2002.

There are at least two things wrong with this picture. First, the set of technologies is complex and constantly evolving. In the WS-I Basic Profile V1.0, the WS-I organization has made an attempt to define a set of Web service technologies that can work together to provide interoperability between implementations. Web service projects would be well advised to follow these recommendations. But at the same time, those underlying technologies are evolving and maturing. What happens when the Basic Profile V1.1 or V2.0 comes out? Also, what happens when you need the business process capabilities? Those specifications are in an even greater state of flux.

The second problem is that all of these technologies are low level. Developing systems directly to these technologies is tedious, complex, and error prone. We need to develop systems to a higher-level of abstraction that makes application programming more productive. And, we need to put our computers to work to handle the tedious details of these many low-level technologies. Raising this level of abstraction is a well-established trend in computing and is one of the main goals of MDA. (David Frankel provides an excellent discussion of these topics in his article).

The Role of MDA

As a quick review, with MDA, we design our systems using models defined in an industry standard notation. This concept is not new, and is in fact how database implementations are typically designed today. An Entity-Relationship or other diagram is developed using a tool like ERWIN, and the lower level schema and database specifics are generated from that model. Nobody involved in data design would question the productivity gains that have been achieved. MDA attempts to bring these same gains to other areas of computing as well.

This raising of the level of abstraction is the major advantage of MDA. We separate the concerns of the business specification from the details of the technology implementation. Then we use tools to generate the technology details from the models. Thus, when a technology changes or evolves, for example when the WS-I Basic Profile V2.0 comes out, we don't have to change our business specification; instead, we generate new versions of the Web service technology artifacts from our PIMs.

The typical MDA process involves the creation of three different types of models as shown in Figure 4. The highest-level model, the Computational Independent Model (CIM) is used to define 'business level' information independent of technology considerations. This model is then refined into what is called a Platform Independent Model (PIM), which specifies the high-level design details of the IT system. Although the PIM assumes a certain architectural style (such as n-tier, Web based applications), it remains independent of the specific technical details of the platform (such as J2EE or .NET). Finally, the PIM is transformed into a Platform Specific Model (PSM), which adds the technology details necessary for implementation on the specific platform. MDA tools can be employed to help automate the transformation between these models and to ultimately generate much of the code. [1]

So, how do we apply MDA to develop and evolve our Web services and service oriented architecture applications? We design our systems using industry-standard, technology-neutral models, and then generate the specific technology artifacts from them. The core value of our systems is in the business capabilities, not the technology details.

---

[1] Platform-independence and platform-specificity are relative concepts. When you claim platform-independence you have to specify what that means in the given context. Typically when people talk about MDA Platform-Independent Models in the context of SOA they mean such models are independent of specific middleware such as J2EE and CORBA and of interface protocols such as WSDL.

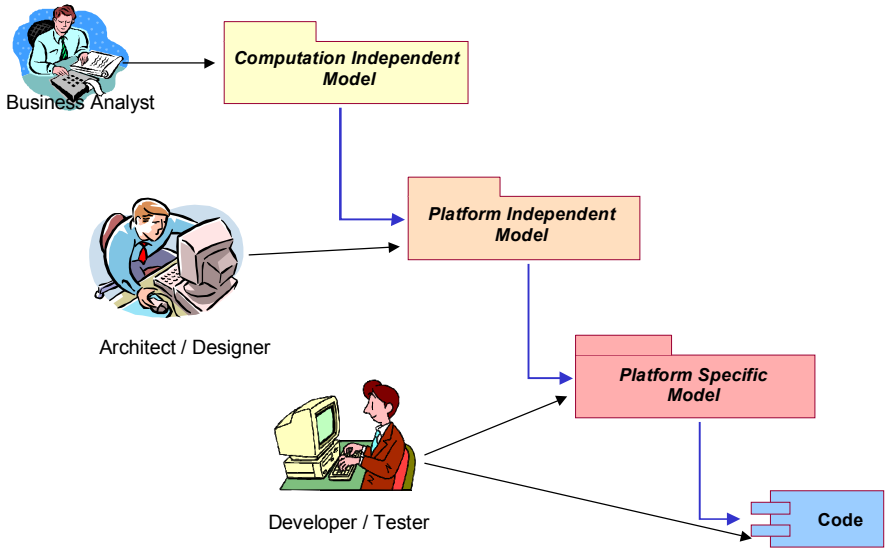

Figure 4 - MDA Process

Figure 5 illustrates some of the artifacts that can be generated. We begin by expressing our designs as MDA models in UML. (A different, but compatible modeling language may also be defined for BPM). We then use tools based on MDA standards to generate the following artifacts from the models:

- *XML Business Documents* – There are two important parts to the documents; one is the schema that defines them, and the other is the documents themselves. Using XML Metadata Interchange (XMI), we can create either, or both. The XML Schema can be generated from the model of the document. Thus, we don't have to (and should not) write the schema for the XML message payloads directly.
- *WSDL* – The WSDL defines the interface to the services that are being exposed as Web services. Using a UML-WSDL mapping, we can generate the WSDL directly from the UML model that describes the services.

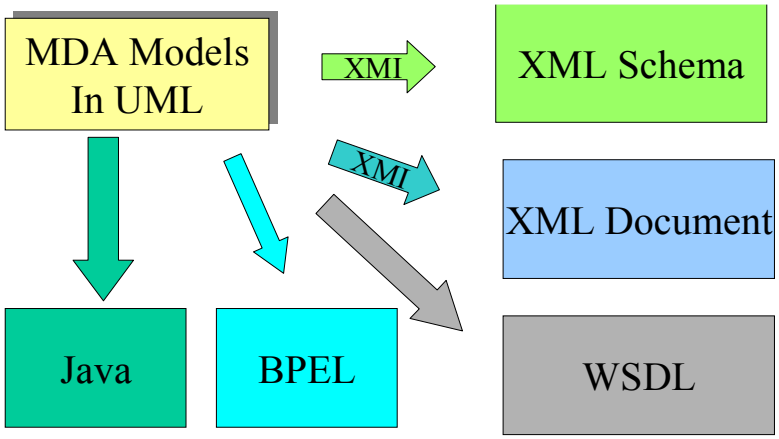

Figure 5 - Generating Web Service Artifacts with MDA

- *Java (or .NET)* – We can also generate much of the structural skeleton code that will be used to implement the services using mappings to Java, .NET and other technologies. Numerous MDA and UML tools are available today that offer code generation and implementations of these mappings.
- *BPEL4WS* – If we define our business process execution in UML, we can also generate the BPEL documents that correspond to those processes. IBM has implemented a mapping for BPEL and is in the process of standardizing it. More about this exciting work can be obtained from the article "From UML to BPEL", by Keith Mantell at: http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/
- *SOAP and UDDI* - We typically rely on the Web services toolkit to generate the SOAP and UDDI information from either the WSDL or object class (Java or .NET object) definition. With the MDA approach, we take advantage of that, but go one step further to generate the WSDL and object classes directly from our models.

As important as insulation from technology churn is with today's volatile Web service technologies, it is not the only advantage that MDA brings. MDA also provides processes and tools for implementing model-based development. But just as important, it allows us to specify and customize the different CIM, PIM and PSMs through the use of UML profiles. Profiles provide a standard way to focus a set of models toward specific architectural styles, applications and environments. Each model that conforms to a profile essentially inherits the architectural framework and decisions that went into defining the profile. A commonly know profile is the UML Profile for EJB (at the PSM level), but in MDA we can have one or more profiles for each of the CIM, PIM and PSM types. These profiles can be based on standards, or on our own custom architectures and requirements.

Thus, we can formally specify SOA in terms of MDA profiles. Each profile would specify the constructs and constraints of the different levels of MDA models. For example we could have:

- SOA Profile for Business Processing – This CIM level profile would be used to describe higher-level business services and would include specific constructs for BPM concepts.
- SOA Profile for Applications – This PIM level profile would be used to describe applications that expose services and that utilize specific infrastructure features such as dynamic discovery.
- SOA Profile for Web services – This PSM level profile would be used to define an SOA infrastructure using specific Web service technologies.

We can't go into the details of profiles in this article, but we will describe the concepts and techniques in an upcoming article on using MDA to implement enterprise architectures.

Conclusion

Service Oriented Architecture has the potential to help enterprises become more efficient, agile and competitive. Up until now, however, the complexity of implementing it successfully has limited these benefits to only a select few. But, luckily for everyone else, several new technologies are converging to facilitate at least the technical aspects of implementing a SOA.

Web services provide a convenient technology for the publication of services, and offers several advantages over previous technologies. Foremost, is the fact that all of the major vendors seem to agree on the importance of Web services and support the standards. However, the nature of distributed systems dictates that services be defined at a higher-level of granularity. Business services that are designed to exchange XML business documents take this into account, but present new challenges in terms of orchestrating business processes and processing business documents. BPM helps to provide a solution and complements the implementation of services and SOA.

But there is still an important problem to be considered. All of these technologies are new and in flux, and addressing this problem is where MDA shines. MDA enables us to develop systems at a higher level of abstraction. This not only makes us more productive, but also insulates us from the lower-level technology specifics and the inevitable changes that happen at this level. As these technologies mature and evolve, we are protected from obsolescence because the technology details are generated from our business-driven designs, not intertwined with them.

If you have already implemented a SOA in your organization, more power to you. If you're just considering it, now is the time to get started. If you're not considering it, think again. In any case, the convergence of MDA, BPM and Web services provide an intelligent approach that is available to you now.

Michael Rosen is CTO of M2VP, Inc., a consultancy specializing in Enterprise Architecture, MDA and SOA. His current emphasis is providing expert architecture consulting to Global 1000 clients in insurance, finance, government and telecommunications. He has more than 20 years' technical leadership experience architecting, designing, and developing software products and applications, including Web services, Java, CORBA, COM, Messaging, Transaction Processing, DCE, networking, and operating systems. Mr. Rosen has been a frequent technical speaker and author, presenting keynotes, tutorials and seminars at conferences in the US, Europe and Asia and has articles in publications including *Cutter Enterprise Architecture Reports*, *Web Services Journal*, *EAI Journal*, *Software Magazine* and *Enterprise Development*. He is coauthor of the book *Developing E-Business Systems and Architectures: A Manager's Guide* and *Integrating CORBA and COM Applications.*