Business Process Standards For Web Services

The convergence of two major trends is creating a rapidly growing demand for a new breed of software that facilitates automation of business processes both between enterprises and within the enterprise.

The first of these trends is Web Services technology: a collection of XML-based standards that provide a means for passing information between applications using XML documents. The ability of Web Services to reach beyond the firewall, the loose coupling between applications encouraged by Web Service interfaces, and the wide support for core Web Service standards by major enterprise software vendors are the key reasons why Web Services technology promises to make integration of applications both within the enterprise and between different enterprises significantly easier and cheaper than before. Loose coupling means that not only can applications be implemented on different platforms and operating systems, but also that the implementations can readily be changed without affecting the interfaces.

The second of these trends is a business driver. In order to increase an organization's agility in responding to customer, market, and strategic requirements, the information flow between the IT systems that carry out these business operations must be streamlined. This includes not only the organization's own IT systems but also those of its partners. It is the task of electronic business integration to automate this information flow as much as possible in order to streamline operations. Historically, organizations have generally focused on integrating their own IT systems.

If, however, the information flow between the organization's own IT systems and those of its partners (particularly in the supply chain) is not also streamlined, then the overall agility of the business is still restricted. Therefore, many enterprises also strive to integrate their partner's IT systems with their own in order to more fully automate critical business processes such as sales, procurement, and research and development. The benefits of the increased agility resulting from business process automation are extensive. For example, operational costs are decreased, inventories are reduced, customer satisfaction is increased, and products are brought to market faster.

A whole new set of tools has arisen to facilitate the integration and automation of business processes. These include graphical process modeling tools, middleware technologies such as CORBA and JMS, integration brokers, Business Process Management Systems (BPMS), and B2B servers. Unfortunately, until recently the investment required by organizations to integrate the IT systems both inside their organization and across the firewall has been very high. This is mainly because the different proprietary interfaces and data formats used by each application have meant that integration projects have had to invest considerable resources in expensive integration tools as well as in the time and expertise to perform the integration.

Web Services technology promises to change this by replacing proprietary interfaces and data formats with low-cost, ubiquitously supported standards for interfaces and data that work as well across the firewall as within it. The first generation of Web Services technology, though, has largely focused on the messaging foundation supported by SOAP and WSDL. While this foundation is sufficient for some internal application integration needs, it is not sufficient to support the complete automation of critical business processes. This requires the ability to specify workflow, security requirements, transaction management, and other critical information related to the business process context. Such information is generally specified in a business process model.

The Need for Business Process Standards

We require standards for business process models that are built on Web Service architectures. These would enable processes to be modeled, deployed, executed, and managed by software from various vendors. Without such standards, a number of undesirable consequences arise. These include:

- Vendors are likely to offer support for such features as proprietary extensions to Web Service standards, leading to vendor lock-in.
- Collaborating enterprises may choose incompatible means of defining the shared process models, leading to inefficiencies and error-prone operations.
- Reuse of proven processes and patterns across products from different vendors is difficult if these can't be specified in a standard way.
- The emergence of best-of-breed tools for modeling and for execution of processes will be hampered.

B2B and EAI Processes

Business processes can be divided into two distinct but converging domains:

- **Public processes** are those that an enterprise shares with its customers, suppliers, or other partners. This is the **business-to-business integration (B2Bi)** domain.
- **Private processes** are those that are internal to the enterprise. This is the **enterprise application integration (EAI)** domain.

Solutions for these two domains share many common characteristics. For example XML document exchange between applications is used in both the EAI and B2B domains for loosely coupled integration of applications. Additionally, in any enterprise, public and private business processes combine to perform the overall operations of the business. These facts drive the demand for a single business process standard that encompasses both the B2B and EAI domains.

There are, however, some important differences between the domains. For example, stricter legal and security requirements will apply to public processes. On the other hand private process models stipulate execution details that are not present in public process models, such as how a purchase order is actually processed by various enterprise applications.

Business Process Features

A business process standard that provides comprehensive support for both public and private processes should consider the following features:

- **Collaboration-Based Process Models**
  Experience in both EAI and B2B process modeling has led to the increasing adoption of collaboration-based process models, usually based on UML. In collaboration-based process models, processes are described as a set of collaborations between various participants, including organizations, applications, employees, and other business processes. Usually participants can be abstracted in model descriptions using roles. The ability to recursively decompose process models is generally required.
- **Workflow**
  The workflow defines how the participants in a process work together to execute a process from start to finish, and is also called choreography or orchestration. Most workflow standards support subprocesses, which allow activities within a workflow to be implemented as another workflow. Workflow descriptions can be generated from collaboration models, or specified independently. Recursively decomposed process models can be mapped to workflow descriptions using subprocesses.

There are two complementary parts to workflow: the **control flow** and the **data flow**. The control flow defines the sequencing of different activities in the process. The data flow defines how information flows between activities.

- **Transaction Management**
  Transactions are crucial building blocks of any business process and a comprehensive business process standard must provide a means for specifying how transactions are managed. Long-running transactions that may take hours or weeks to complete must be supported. If an enclosing transaction fails after an enclosed transaction is completed, some compensating actions may be needed. For example if a hotel reservation is canceled after a payment has been authorized, a compensating action may be required to cancel the payment. Time constraints for receiving responses or acknowledgements may also be required.

- **Exception Handling**
  If an exception is raised during the course of a business process, then it is important that the model allow appropriate recovery actions to be taken.

- **Service Interfaces**
  Web Services provide a basis for passing messages between participants in collaboration-based processes. Some recently proposed business process standards such as WSFL and XLANG use WSDL interfaces to describe the loosely coupled services exposed by participants.

- **Message Security and Reliability**
  For mission-critical processes, reliable and secure message delivery is required. Additionally, B2B messages may need to be digitally signed and authenticated. These quality-of-service semantics may vary for different transactions.

- **Audit Trail**
  It is generally very important for legal purposes in B2B processes that an audit trail of certain business transactions is kept. This means that a trading partner is unable to claim that a transaction was not accepted when in fact it was; that is, it ensures non-repudiation of the transaction by the partner. Digitally signed receipt acknowledgements of messages may be demanded.

- **Agreements**
  The notion of agreements is specifically for B2B processes. An agreement represents a contract between two or more partners to carry out specific functions (identified by roles) in a public business process.

- **Execution**
  Public processes describe only how information should flow between organizations. In order to be able to fully automate the execution of the business process within an organization, the complete information flow within that organization as well as across its firewalls must be specified. This requires the process models to fully describe the private as well as the public activities of the organization.

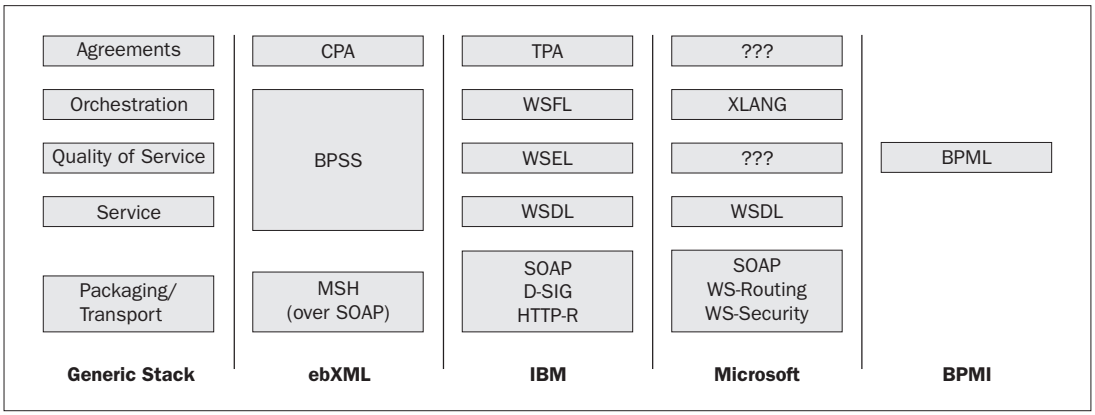
A powerful approach supported by some standards is Web Service aggregation, whereby one Web Service is used in the implementation of another. For example an organizational workflow that handles purchase orders might receive the orders from customers via one Web Service and then call an internal ERP application via another Web Service to help process the order. Such an approach should become significantly less expensive than traditional EAI methods.

The Web Services Stack

In order to describe how Web Service standards relate to the above features, it is useful to begin by looking at a representative Web Services architecture.

Web Services architecture is built from layers of technology and standards on which services can be implemented and deployed. Each layer on this Web Services stack depends on the layers below it. There are many variations of this architecture, but each variation generally includes the features described in the previous section in addition to the basic messaging and service description foundation layers.

The following diagram illustrates a generic Web Services architecture, and how it maps to specific architectures from prominent organizations or companies. The next section examines some of the business process specifications in more detail:



In this generic architecture we have the following layers:

- **Packaging/Transport**
  This enables information to be packaged into messages and transported reliably and securely between participants. It is sometimes just called the **messaging layer**.
- **Service**
  This layer describes the operational interfaces of a Web Service.

- **Quality Of Service**
  This layer describes non-operational aspects of services, including reliability and security characteristics.
- **Orchestration**
  This layer describes how services interact with each other in business processes using workflow descriptions. This layer is also sometimes referred to as the **choreography layer**.
- **Agreements**
  This layer describes how specific trading partners will collaborate to perform some shared business process.

This generic architecture is of course a highly simplified representation. It omits some important elements that are not the focus of this article, for example service discovery.

It should be noted that BPMI deliberately only defines the process layer, as it is intended that BPML process models can bind to complementary standards from other stacks (see *BMPL* for more details) .

The Candidates

Now let's examine those specifications that address the orchestration layer of the Web Services stack, the core layer that describes business process semantics. These are ebXML BPSS, XLANG, WSFL, and BPML. Each supports some subset of the aforementioned features, depending largely on the domain they are addressing.

ebXML BPSS

ebXML BPSS (Business Process Specification Schema) is part of the comprehensive ebXML B2B suite of specifications, which also includes core specifications for reliable and secure messaging based on SOAP, collaboration agreements and profiles, a registry/repository, and core components.

BPSS is a relatively simple but effective schema that describes public processes only. In a BPSS model different roles (seller, buyer, etc.) collaborate to carry out a set of transactions. The orchestration of the transactions is defined using a control flow based on UML activity graph semantics. There is no explicit support for describing how data flows between transactions.

The transaction part of the model is based on a proven, robust model for long-lived e-commerce business transactions used by previous B2B standards such as RosettaNet. There is explicit support for specifying quality-of-service semantics for transactions such as authentication, acknowledgements, non-repudiation, and timeouts:

Feature	Support
Collaboration-Based Modeling	BPSS describes public processes as collaborations between roles, with each role abstractly representing a trading partner. There are two types of collaborations: binary collaborations between two roles, and multi-party collaborations between three or more roles. Multi-party collaborations are decomposed to binary collaborations. Recursive decomposition is further supported through nesting binary collaborations inside other binary collaborations, making for a flexible solution.
Workflow	BPSS workflow is described by assigning a public control flow based on UML activity graph semantics to each binary collaboration. The control flow describes the sequencing of business transactions between the two roles. The control flow can specify sequential, parallel, and conditional execution of business transactions. There is also a limited facility for describing control flow across multi-party collaborations.
Transaction Management	BPSS supports a long-running business transaction model based on robust, proven e-commerce transaction patterns used by previous standards such as RosettaNet. A business transaction consists of a request and optionally a response. Each request or response may require that a receipt acknowledgement be returned to the sender. Additionally for contract-forming transactions such as purchase order requests, an acceptance acknowledgement may need to be returned to the requester. Time constraints can be applied to the return of responses and acknowledgements. If a business transaction fails on either side, the other side is notified so that both sides can carry out any actions necessary to process the failure in their internal systems. Transactions are not nested and there is no support for specifying compensating transactions.
Exception Handling	BPSS defines a number of possible exceptions and prescribes how these are communicated and how they affect the state of the transaction. They generally cause the transaction to fail. Transitions exiting from a transaction can be enabled based on whether the transaction failed or succeeded. For example if a quote request transaction fails, a procurement process might transition to completion, whereas if it succeeds the process might transition to a purchase order transaction.

Table continued on following page

Feature	Support
Service Interfaces	BPSS process models implicitly contain service interface descriptions for each role. The service interfaces support specific asynchronous request and response operations, each with a defined message content. That content can consist of any number of specified XML document types and MIME attachments. The service interface also implicitly supports generic acknowledgement and exception messages. Organizations can advertise their support for particular roles (service interfaces) in ebXML collaboration profiles and agreements, which include the location of the services. WSDL descriptions of the service interfaces for each role could be readily generated although there is no standard mapping at this time.
Message Security And Reliability	BPSS assumes that processes will use reliable and secure messaging services such as the ebXML messaging service. For each request or response, it can be stipulated that the identity of the originator must be checked for authorization purposes. For document security, it can be stipulated whether each document or attachment in a request or response must be encrypted, whether it must contain a message digest to prevent tampering, and whether a digital certificate is required. For each transaction it is possible to specify whether guaranteed delivery of messages is required. Default settings for these properties can be specified as attributes of transactions in a BPSS model, and these defaults can then be overridden in a CPA (collaboration protocol agreement) between two partners.
Audit Trail	For each request or response, it can be stipulated that the sender must save a copy of the message contents. Additionally it can be stipulated that a digitally signed receipt acknowledgement must be returned to the sender, who then saves it. This provides a high degree of non-repudiation of transactions. Default settings for these properties can be specified as attributes of transactions in a BPSS model, and these defaults can then be overridden in a CPA between two partners.
Agreements	A BPSS process model can be referenced in an ebXML CPA. This provides details on which trading partner supports which role in a specified process model in the context of some business agreement.
Execution	As a public process schema, BPSS provides no support for internal execution semantics.

See also http://www.ebxml.org/specs/ebBPSS.pdf.

XLANG

XLANG is Microsoft's proposal in this space, and like BPSS is currently focused entirely on public processes.

XLANG uses WSDL to describe the service interfaces of each participant. The behavior is specified with a control flow that choreographs the WSDL operations. There is no means for specifying data flow between operations. Long-running transactions encompassing multiple operations are supported and can be nested. Compensating operations for transactions can be specified. Exceptions can be caught and recovery operations specified. Acknowledgements and timeouts can be flexibly incorporated. Some support for agreements is provided in XLANG by contracts, which defines how to stitch together Web Services of collaborating partners.

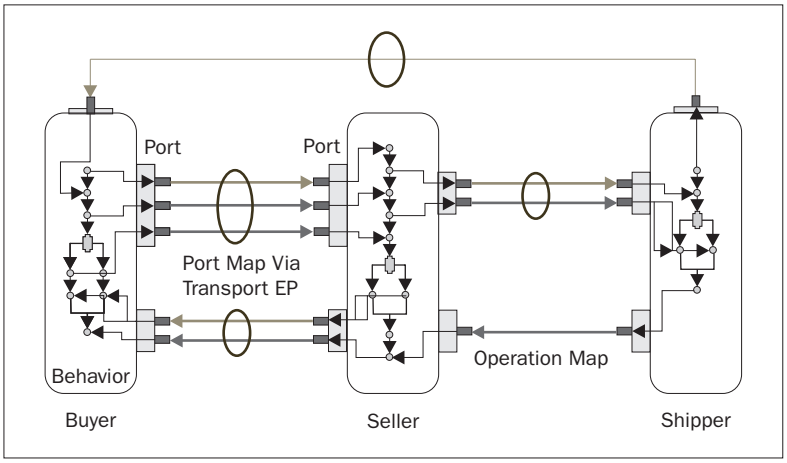
XLANG does not define quality-of-service characteristics of Web Services such as non-repudiation and authentication, or guaranteed messaging requirements.

Feature	Support
Collaboration-Based Modeling	XLANG describes processes as interactions between Web Service providers so collaboration-based process modeling tools are possible. The block-structured control flow descriptions of XLANG are more suitable for generation from flow-chart tools than UML tools, but the latter is possible. Recursive decomposition of XLANG processes is facilitated by actions that are implemented by subprocesses.
Workflow	In XLANG the workflow associated with each Web Service is defined by an XML `<behavior>` element. This defines a control flow based on a block-structured approach. The control flow supports sequential, parallel, and conditional actions. Actions can include WSDL operations, timed waits, and the raising of exceptions. There is no support for specifying data flow between actions.
Transaction Management	XLANG provides a flexible and comprehensive long-running transaction model. Transactions are scoped by context blocks, within which any number of actions can be defined. Transactions can be nested to any level. Compensating blocks can be associated with each transaction context. If a fault occurs in a transaction then the compensating actions of all nested transactions that have completed will usually need to be executed. XLANG allows flexible specification of the order in which such actions will be executed, but the default is reverse order.

Table continued on following page

Feature	Support
Exception Handling	XLANG provides flexible exception-handling facilities. Exception handlers can be specified for any block of actions, and explicit recovery actions specified including the compensating blocks of specified transactions. Exceptions can also be raised at any point in the control flow.
Service Interfaces	XLANG uses WSDL to describe the service interfaces for each participating Web Service.
Message Security And Reliability	There is no support for security and reliability semantics in XLANG.
Audit Trail	There is no support for non-repudiation semantics in XLANG.
Agreements	XLANG supports the notion of business process contracts, which could provide the foundation for business agreements. These specify how two or more XLANG-enabled Web Services are stitched together to describe a shared process between particular participants.
Execution	XLANG is focused on public processes and omits some details required to automate execution of a process, for example data flow constructs.

The following diagram illustrates a sample three-party contract in XLANG:



See also http://www.gotdotnet.com/team/xml_wssspecs/xlang-c/default.htm.

WSFL

WSFL (Web Services Flow Language) is IBM's proposal in this area. It covers both public and private processes. WSFL is primarily focused on describing Web Service compositions, and like XLANG uses WSDL to describe the service interfaces.

A **flow model** describes the workflow for a process. Both control flow and data flow can be defined using a state-transition model. Transactions and exception handling are not explicitly supported, but some of the semantics can be implemented using conditional transitions. Activities in a workflow can be exported as Web Service operations, and activities can also be implemented by delegation to a Web Service. In this way WSFL supports Web Service aggregation.

A **global model** defines how the various Web Services are linked together in the process. It is similar therefore to the business process contracts of XLANG.

Quality-of-service characteristics are delegated to a separate specification called WSEL (Web Services Endpoint Language).

Feature	Support
Collaboration-Based Modeling	WSFL describes processes as interactions between Web Service providers, which can be abstracted using roles so collaboration-based process modeling tools could certainly be used to generate WSFL descriptions. Recursive decomposition of WSFL processes is facilitated because WSFL flow models can be exposed as Web Services, which in turn can be used in the implementation of activities in other flow models.
Workflow	In WSFL, a flow model defines the workflow associated with each service provider (collaboration role). This defines both a control flow and a data model. The control flow is based on transitions between activities. Transitions can specify XPATH conditions on particular messages that enable or disable them, thus directing the process flow to different activities depending on the content of the messages. Data flows can extract data from different activities using XPATH expressions, transform them using XSLT, and aggregate them for input into other activities.
Transaction Management	WSFL doesn't support transactions. Transactional characteristics of Web Services are being addressed in another IBM project (WSTx), which might end up contributing to the complementary WSEL specification. See http://www.research.ibm.com/AEM/wstx.html for more details on WSTx.

Table continued on following page

Feature	Support
Exception Handling	WSFL can support handling different exceptions that are indicated in the content of messages by specifying transition conditions that examine the message for these exceptions. In this way the process flow can be directed to different activities for different exceptions.
Service Interfaces	WSFL explicitly uses WSDL to describe the service interfaces for each participating Web Service.
Message Security And Reliability	There is no support for security and reliability semantics in WSFL. This is delegated to the separate WSEL specification.
Audit Trail	There is no support for non-repudiation semantics in WSFL. This is delegated to the separate WSEL specification.
Agreements	In the IBM Web Services stack, agreements are a separate component (TPA) but WSFL global models give a foundation that could be used for business agreements.
Execution	WSFL provides execution capabilities for activities through Web Service invocations or through Java, CICS, or EXE/CMD-based implementation.

See also http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

BPML

BPML (Business Process Management Language) is a specification from the BPMI.org (Business Process Management Initiative) organization. BPML aims to provide a comprehensive means of specifying the processes of an enterprise. It is positioned as complementary to public process standards such as ebXML BPSS – the BPMI FAQ (from the BPMI.org web site, http://www.bpmi.org/faq.esp) states:

"**What is the relationship between BPMI.org and ebXML?**
BPMI.org and ebXML are addressing complementary aspects of e-Business process management. While ebXML provides a standard way to describe the Public Interface of e-Business processes, BPMI.org provides a standard way to describe their Private Implementation."

BPML describes comprehensive control flow and data flow constructs. It supports both short- and long-running transactions with compensating activities. It also supports exception handling and timeouts. It does not provide a means to specify characteristics that are important to B2B processes, such as authentication and non-repudiation.

Feature	Support
Collaboration-Based Modeling	BPML describes processes as XML message exchanges between participants. Participants can be abstracted using roles and can represent organizations, applications, employees, or other processes. Participants can be assigned statically or determined dynamically at runtime. In essence the private process represented by a BPML process interacts with participants through a set of collaborations. Such descriptions are amenable to generation from collaboration modeling tools. Recursive decomposition is supported through nested processes.
Workflow	BPML provides comprehensive control and data flow support. A process consists of a simple or a complex activity. Simple activities include sending or receiving an XML message, invoking a Web Service operation, or raising an exception. Complex activities include block-structured control flow constructs for sequential, parallel, and conditional execution of other simple or complex activities. Activities can be scheduled to start at a future date, and time constraints can be assigned to the duration of the activity. Data flow between activities is accomplished by assigning data from messages to state variables and vice-versa. Rule sets express complex conditions based on XPATH expressions that can be used to filter input messages to activities.
Transaction Management	BPML provides comprehensive support for both ACID (coordinated) and long-running (extended) transactions. A transaction can be associated with any complex activity. This implies that transactions can be nested. Compensating activities can be associated with both coordinated and extended transactions. If a transaction is aborted, any compensating activities within the same context will be executed in reverse order.
Exception Handling	The exception-handling capabilities supported by BPML are robust and quite similar to XLANG. Exceptions are propagated upwards to enclosing activities until caught. If not handled within a transaction, the transaction is aborted.
Service Interfaces	The service interfaces exposed by participants in collaborations can be described in BPML using abstract processes. An abstract process need not fully specify how the participant implements the process, but does specify aspects of their behavior relevant to the overall process model. Thus BPML abstract processes are analogous to descriptions of participant behavior in purely public process models such as ebXML or XLANG. Conceivably, mappings could be performed between these standards and BPML abstract processes. The service interface part of abstract processes is very similar to WSDL so that part of a mapping should be quite straightforward.

Table continued on following page

Feature	Support
Message Security And Reliability	There is no support for security and reliability semantics in BPML.
Audit Trail	There is no support for non-repudiation semantics in BPML.
Agreements	There is no support for agreements in BPML.
Execution	Participants in BPML processes can represent IT systems, applications, or users within an organization, or external service providers. Thus, by exchanging messages with these participants the detailed implementation steps of a process can be specified. BPML does not specify all details for binding such participants, for example messaging transports or application programming interface bindings. Such details are left to vendors. If legacy applications are already exposed as WSDL Web Services, then they can be incorporated as participants in BPML processes by vendor tools that map the WSDL interfaces to BPML abstract processes and route the messages at run-time using SOAP. Such processes would then look very similar to the Web Service composition approach facilitated by WSFL. It is conceivable that such an approach could be standardized in a future version of BPML.

See also http://www.bpmi.org/.

Convergence

As outlined above, the business drivers point to a convergence of private and public business process model standards based on Web Services. How might this convergence occur in practice?

It seems likely that both ebXML BPSS and BPML will remain focused on their complementary domains for the time being, which are the B2B and EAI domains respectively.

On the other hand, Microsoft and IBM are clearly moving towards a set of specifications that would address both B2B and EAI requirements. It has been widely speculated that they will collaborate to produce a single proposal or set of proposals in this space that could then be submitted to the W3C for inclusion in its Web Services architecture stack in the process layer.

There are, however, significant obstacles to be overcome for this to happen. Technical obstacles include the different approaches to control flow modeling (in XLANG control flow is described using a block-structured approach best represented graphically using flow charts, while WSFL uses a state-transition approach best represented graphically using UML activity or state graphs). This is not just an argument about the technical merits of the respective approaches – both vendors have significant investments in these technologies in their respective product lines (WebSphere from IBM and BizTalk from Microsoft).

Given these obstacles and the time it takes for any new proposal to become widely supported in products and in the marketplace, the widespread adoption of a single Web Services-based standard for B2B and EAI processes is some time away.

Although the standards convergence process is ongoing, this does not necessarily mean that enterprises should wait before adopting one or more of these standards. The potential return on investment from automating business processes means that it might be quite costly for enterprises to wait until the standardization process has settled before adopting business process modeling and automation tools. The best way to protect investment in such tools is to ensure that the vendors are committed to a standards-based approach. For the moment an enterprise should focus on the standards that best support the domain that it is most urgently seeking to automate. If seeking to integrate public and private processes, an approach based on using complementary existing standards should be considered.

OMG EDOC

At this point it is worth mentioning another relevant emerging standard that applies to the modeling of business processes for Web Services. This is the EDOC (Extended Distributed Object Computing) standard from the OMG (Object Management Group, http://www.omg.org/).

EDOC essentially defines a modeling framework that supports the OMG MDA (model-driven architecture). It aims to support collaborations between loosely coupled systems in both the B2B and EAI domain, and to enable the reuse of business components from different distributed object technologies in these collaborations, such as CORBA, EJB, and Web Services.

EDOC is based on UML and defines several complementary subprofiles, including a Component Collaboration Architecture. This profile defines the core concepts that can be used to describe collaboration-based process models. Such models could be mapped to the different business process standards described above that are then used to drive the execution of the collaborations. Thus EDOC is clearly complementary to these standards.

Conclusion

It is clear that businesses are increasingly moving towards comprehensive automation and integration of their private and public processes, and that Web Services is becoming increasingly popular for use as the integration infrastructure. This scenario drives the demand for Web Services-based business process standards. Over the next couple of years we can expect to see continuing activity to address this demand.