

Web Services vs. ebXML

An Evaluation of Web Services and ebXML for e-Business Applications

Yuhong Yan¹ and Matthias Klein²

¹NRC-IIT, Fredericton, NB, Canada,
yuhong.yan@nrc.gc.ca

² Faculty of Computer Science,
University of New Brunswick, Canada
matthias@cmklein.de

ABSTRACT

Web Services and ebXML are modern integration technologies that represent the latest developments in the line of middleware technologies and business-related integration paradigms respectively. In this paper, we discuss relevant aspects of the two technologies and compare their capabilities from an e-Business point of view.

KEYWORDS

Web services, ebXML, distributed information system, e-commerce, information infrastructure,

INTRODUCTION

For companies operating in an increasingly globalized business environment, e-Business means online transactions, automated business collaborations and system integration. This means not only to provide products through supply chains, but also to delivery services and information through networks. The e-Business tools and standards come from two domains known as **Web Services** and **e-Business XML** (ebXML¹).

Web Services is a technology-oriented approach. Its ancestors include COBRA² and other middleware technologies such as TPM³ and RPC⁴. W3C⁵ is a big sponsor of Web Service technologies. Many Web Services standards, such as SOAP⁶, WSDL⁷, UDDI⁸ etc., are W3C standards or recommendations. Many world-level IT companies currently support Web Service technology. Web Services is moving from a middleware solution to a tool of business process integration by adding more functions for business entity description and business process management.

In comparison, ebXML is the successor of EDI⁹. ebXML is sponsored by UN/CEFACT¹⁰ and OASIS¹¹. It is the latest achievement in a long line of business integration paradigms that include EDI, ANSI X.12¹², EDIFACT¹³, EAI¹⁴, XML-EDI, B2Bi¹⁵ or BPI¹⁶. Compared to Web Services, ebXML is more at the executive business level (Alonso, *et. al.*, 2003). Though currently there is lack of software tools implementing ebXML specifications, existing Web Service software can be modified as implementation of ebXML specifications through binding.

In this chapter, we discuss relevant aspects of the two technologies and compare their capabilities from an e-Business point of view. We see a B2B process as following. Before doing business

with someone, a business needs to *find its partner*. While negotiating with this potential partner, *documents and messages* must be processed *via reliable and secure channels*, such as post or courier services. Those documents must be designed in a *semantic* fashion that both partners understand. In order to ensure smooth business operation, the companies will have to agree upon the *processes* the resulting transactions are to follow. Ultimately, a contract or *trading partner agreement* must be signed to establish this new business relationship. Therefore, we compare the two technologies from the above aspects. We point out the capabilities and the limitations of both and discuss trends in the future development of both technologies. This also helps the readers to make right decisions of choosing the specifications and implementation software when facing a new B2B integration project.

OVERALL FUNCTIONALITY

Both Web Services and ebXML put their service entities on a network and have means for service description, service discovery and service invocation. For Web Services, it adopts a **Service-Oriented Architecture (SOA)** with three kinds of parties – service providers, service requesters, and service registries (as shown in Figure 1). The service providers register their service descriptions in the service registries for service discovery purposes. The service requesters search the service registries for services that meet their requirements. The service requesters then can communicate with the service providers directly and use their services. Similar to Web Services, ebXML also has a service register to collect the service descriptions. Different from Web Services, the business partners are not distinguished as service providers or requesters, but are treated as the same role of business partners. The service discovery and invocation are similar to Web Service (details in this section). For some people in ebXML community, ebXML is not a SOA solution. If we consider SOA is a kind of architecture in computing technology, this argument is true that SOA is a solution to software component reuse, analogy to object-oriented architecture. But we can expect that the implementation of ebXML should be a kind of SOAs that comprises loosely joined, highly interoperable application services. In fact, the current practice shows that ebXML adopts some SOA technology such as SOAP.

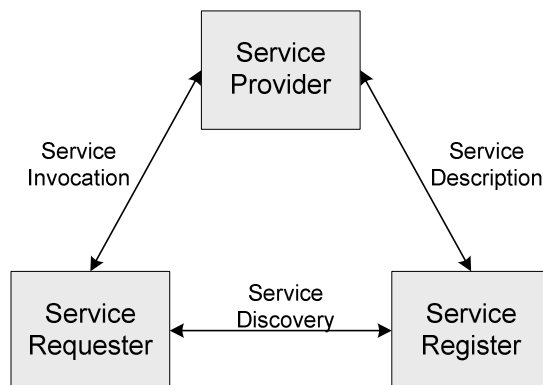


Figure 1: Both Web Services and ebXML have means for service description, service discovery and service invocation.

In Web Services, the interactions among the parties are implemented in a straight forward manner. The communications between any parties use *SOAP* which is based on Internet protocols and XML¹⁷ technology. It is exactly SOAP that makes Web Services interoperable across the platforms and programming languages. *UDDI* is the protocol used by service registry to describe the information of the services. One important piece of information in the business descriptions is the URI¹⁸ for the *WSDL* file. WSDL is an XML file describing how the service can be invoked from a software engineering point of view.

Web Services invocation is similar to an RPC call (Figure 2). The client side wraps the parameters for a remote function call into a SOAP message using the encoding convention. (Marshalling) The SOAP message is transported to the server end and unwrapped. The parameter information is used to invoke the service. The same method is used to send information back to the client side. Many companies and W3C are working to move Web Services beyond the function of RPC. For example, standards are being suggested for business process modelling (see the section for Business Process Modelling).

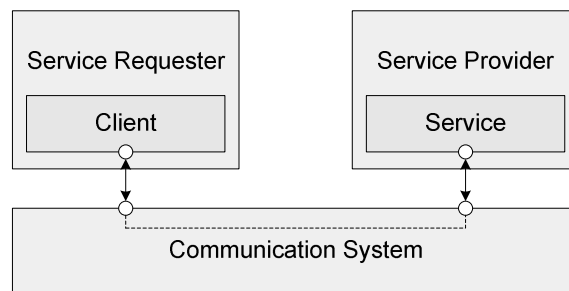


Figure 2: A Web Service follows a simple RPC-like communication pattern

The interactions among the parties are far more complicated in an ebXML-enabled system than for Web Services. ebXML is geared towards business-oriented collaboration of arbitrary partners. It works in two phases:

1. Implementation Phase (Figure 3)

A company that wishes to enter a new business sector queries the ebXML Registry to determine if third parties, such as existing vertical standardization organizations (e.g. ODETTE¹⁹ for the European automotive industry) have already placed an industry profile there. This profile contains business processes, conventions of this sector, the specific documents and forms used or rules on how to do business in this industry.

If such a profile already exists, the “new” company downloads it and adapts its own system to comply with these rules and processes. This is a manual step that is needed only once when a business enters a new business sector as supposed to once per business partner when using Web Services. One can reasonably assume that a company changes its business sectors far less often than its business partners. Nevertheless, this manual adaptation can be further reduced if the provider of the company’s business system (e.g., SAP²⁰) provides templates for all existing business sectors. Even then, however, the newcomer has to decide which of the many processes described in the industry profile it wishes to support. The technical parameters of the message exchange capabilities are described by *Collaboration Protocol Profile* (CPP). This CPP is uploaded to the ebXML Registry so that other companies can find it.

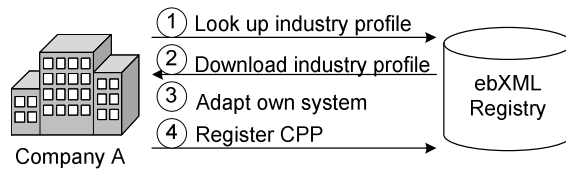


Figure 3: The implementation phase of ebXML prepares the system of a joining company for business collaboration within a vertical industry branch

2. Runtime Phase (Figure 4)

Any other company can now download this CPP from the Registry. Assume Company B downloads the CPP of Company A. Company B can compare those constraints to its own guidelines and rules and propose a **Collaboration Protocol Agreement (CPA)** which is the agreed technical parameters for message exchange. If the proposal complies with the rules defined in the CPP, Company A will agree to it and business transactions can begin. A CPA does not cover all aspects the companies may want to agree on. It is just the technical part of Trading Partner Agreement (TPA). ebXML currently defines only CPP and CPA. The business –related agreements seem to involve paper work and human being.

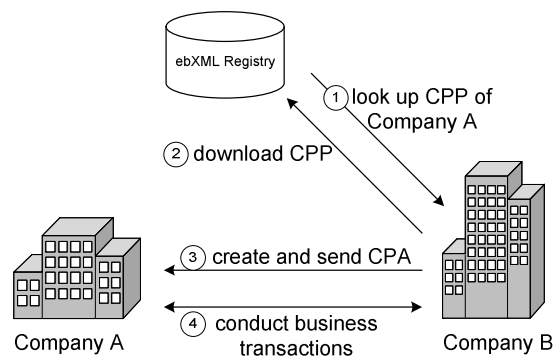


Figure 4: The steps of the runtime phase of ebXML can be carried out automatically. However, this does not mean that manual intervention is not possible.

We point out that we do not include the design phase of ebXML in this chapter. It is because there is no explicit equivalent process in Web Service standards. In short, the design phase in ebXML standards defines the workflows and worksheets that used for business process acquisition and modelling. Any organization can describe ebXML-compliant business processes. One can check (Chappell, *et. al.* 2001) for more information on the design phase.

From the implementation phase and run time phase, one can see that ebXML is a much more complex system than Web Services. Indeed, it is true that Web Services-enabled systems can be implemented very quickly if developers use the existing, powerful libraries. Those libraries allow for developers to accomplish technology-centric integration assignments quickly. But from a business integration point of view, Web Services have two major drawbacks:

1. Web Services rely on stubs. A client must implement one stub for each Service with which it is to interact. In a rapidly changing business environment where companies maintain collaborations

with hundreds or thousands of arbitrary international partners, such technology-oriented bottom-up architecture might prove to be too limited.

2. Web Services describe systems, not business. More precisely, Web Services describe the parameter types for service invocation from the software engineering point of view, but not the semantics of the parameters from the business point of view. While this is valid for simple Web Services, the limitations when dealing with more complex business scenarios are quite evident. New standards and research efforts are trying to change this.

On the other hand, *the major drawbacks for ebXML are its complexity and the fact that implementations are still rare* (see below). While it is possible to have a simple Web Service up and running within minutes, a simple ebXML system will require much more efforts and time. Accordingly, small technology-oriented integration scenarios remain the strength of Web Services.

MESSAGE TRANSPORT

In order to convey messages between service providers and requestors, both ebXML and Web Services use SOAP. SOAP can be transferred via arbitrary protocols, yet the only binding defined in SOAP Specification 1.2 (Gudgin, *et.al.*, 2003) is of SOAP to HTTP.

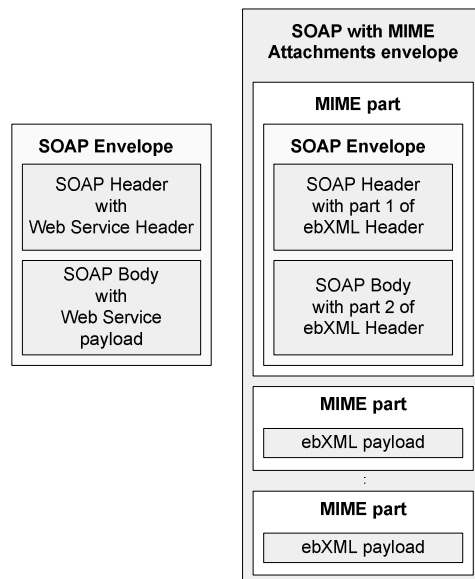


Figure 5: the ebXML message container (right) is more complex than the rather simple Web Services envelope (left).

All SOAP messages are XML documents, but the structures of the SOAP messages are different between Web Services and ebXML (see Figure 5) (Barton, Thatte, and Nielsen, 2000).

SOAP messages for Web Services always contain a SOAP *envelope* and a SOAP *body* inside the envelope. The payload is in the SOAP body. The SOAP messages may contain an optional SOAP *header* which provides a mechanism for adding information about the message. For example,

information about message routing, authentication and transaction management can be contained in the SOAP header.

Since Web Services are built on top of the XML serialization architecture, the data types are limited to those defined in the XML schema. Though some implementations from companies provide mechanisms to define complex data types, it should not be encouraged because it introduces potential interoperability problems. The transfer of binary data, such as images, can only be performed using something like Byte Arrays. But the communicating parties must agree on the format before being able to parse the data. Floating point data can also bring in interoperability problems between different programming languages (Cohen, 2002).

The biggest difference in SOAP messages for ebXML as compared to those for Web Services is that ebXML uses multipart MIME²¹ Attachments for the payload. The ebXML *Message Header* is divided into the two parts that are placed into the SOAP header and SOAP body respectively. Part 1 of the ebXML Message Header contains mandatory information, such as routine information, and optional information such as error control or signatures. Part 2 of the ebXML Message Header contains the manifest information which is mainly an index of the payload. The ebXML payload is entirely stored in one or more MIME Attachment. This enables ebXML to easily transmit binary data such as picture catalogues. We point out that SOAP messages for Web Services also support MIME Attachments. And it is exactly the reason that ebXML adopts SOAP instead of developing its own messaging mechanism.

Since ebXML itself relies on its Core Components (see chapter “Semantics”), interoperability issues are not likely to occur. However, it is possible to use any type of payload within ebXML, so it is necessary to be aware of the common issues associated with the chosen payload format. For migration purposes, it is necessary to be able to process discretionary payloads.

The basic SOAP specifications do not fully satisfy e-Business demands and requirements for security and reliability. This is why OASIS (OASIS, 2002a) defines a set of layered extensions to the basic SOAP specifications for ebXML. Those extensions define a *Message Reliability Layer* which handles the delivery and acknowledgement of ebXML messages to ensure QoS²² and transactions.

SECURITY

The Web Services specification does not define a security framework, but refers to the SOAP Extensibility model. The W3C has specified those extensions for digital signatures, encryption, credentials, and authentications. It remains the responsibility of the provider of a Web Service to implement security services. This may not always be done.

Since business transactions require integrity, confidentiality and availability of all participating systems, the ebXML Security Team conducted a risk assessment in (UN/CEFACT and OASIS, 2001b) and proposed further specifications for ebXML. Even though there is no complete security model defined for the overall ebXML specification yet, available security technologies are integral parts of the sub-specifications defining Business Processes, Trading Partners, Registry & Repository and the Messaging Layer. For example, the CPP/CPA defines

authorization, authentication, and confidentiality. It also provides the means to create tamper-proof documents.

SERVICE DISCOVERY

To locate a service, Web Services use UDDI, while ebXML relies on a Registry with Repositories.

The platform-independent, XML-based UDDI is a directory service that allows access to WSDL information using a SOAP interface. A UDDI business registration consists of three main components:

- White Pages: Business names, contact information, human-readable description and identifiers such as tax IDs
 - Yellow Pages: Services and products index, industry codes and geographic index
 - Green Pages: e-Business rules, service descriptions, application invocation and data binding
- The contents of this register can be found using API calls.

In ebXML, the registry contains descriptions of business artefacts and the (distributed) repositories actually store them. Those business artefacts are usually (OASIS, 2001):

- Business Process & Information Meta Models
- Business Library
- Core Library
- Collaboration Protocol Profiles
- List of Scenarios
- Messaging Constraints
- Security Constraints

Apart from those objects, the ebXML Registry is designed to handle arbitrary information fragments such as XML schema, documents, process descriptions, ebXML Core Components, context descriptions, UML²³ models, information about parties and software components (OASIS, 2002b). Compared to UDDI, the ebXML Repositories are intended for more general purpose storage. UDDI is more specialized and geared towards the type of information that can be stored in the White, Yellow and Green pages. While UDDI stores mostly flat lists, ebXML is capable of handling classification information and information about relationships between business artefacts.

Generally, the UDDI model focuses on middleware connectivity and describes the systems companies use through XML. In contrast, ebXML standardizes the way XML is used in B2B²⁴ integration.

SEMANTICS

When carrying out business transactions, business information (e.g. documents) must be processed. The Semantic Triangle (Figure 6) illustrates that the term associated with a concept

must be defined according to the context of the communication. There is ambiguity of linguistic terms and the objects to which they refer. In a bilateral environment, it might be acceptable for two partners to define proprietary and non-reusable document formats for their transactions. If we want the business process and the business information to be reusable on a global basis, we need to add a layer of Core Components.

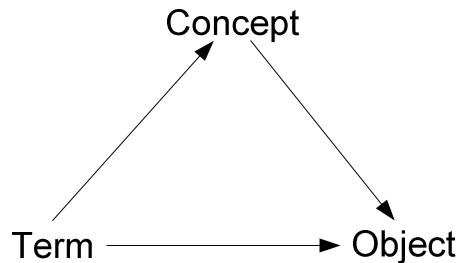


Figure 6: the semantic triangle describes, among other things, the fact that several linguistic terms might describe the same object.

In the example in Figure 7, companies use different terms, “supplier” or “contractor” for the same concept. If both companies map their concepts to an object whose identifier is “CC 0815”, they may talk to each other without ambiguity. Such an agreed upon use of common descriptors is vital to industries or businesses if they want their systems to carry out cross-company processes.

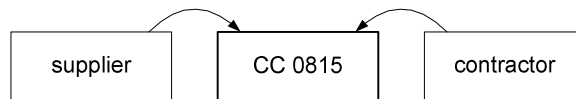


Figure 7: unique identifiers can solve semantic problems by introducing a common vocabulary so that all business partners refer to the same unique object, even if they use different terms.

Core Components in ebXML are the standardised data elements that are used for constructing electronic business documents. In other word, Core Components are building blocks that serve as the basis to assemble business documents so that the business document can be mutually understandable in business collaboration. Simply, a core component is the object identified as “CC0815” in figure 7.

Core Components are in fact the generic representations of information on UML object classes (UN/CEFACT, 2003). Because UML class diagrams have four categories of elements, there are four categories of core components: Aggregate Core Components (ACC), that represent Object Classes; Basic Core Components (BCC), that represent simple properties of Object Classes; Association Core Components (ASCC), that represent relations between Object Classes, where one Object Class is the (complex) property of another Object Class; Core Component Types (CCT), that define the type of information that a Basic Core Component may contain, like text, a number or a date. Each Aggregate Core Component, Basic Core Component and Association Core Component is given a unique name, under which the Core Component can be found in a registry or dictionary. This name is therefore called a “Dictionary Entry Name”. The Dictionary

Entry Name consists in principle of three parts or “terms”: the object class term (the name of the object class), the property term (the property the core component is representing) and the representation term (the name of the data type that is derived from the core component type).

The so-called *context driver* defines the environment where the business process is engaged. The specific Business Information Entities which are contained by a business document can be derived contextually from the more generic Core Components.

Example: when a business process/document contains a “date of order” item, its North American (ISO²⁵) representation will be YYYY-MM-DD, while the European representation of the same component will be DD-MM-YYYY. A context driver can translate the “date of order” core component into proper format according to Geographical Context = “Europe” or “North America”.

Being able to use Core Components to create new documents that are mutually understandable is a very powerful semantic instrument. This flexible tool can help diminish the semantic gap of EDI technologies, but only if it is globally accepted and widely adopted. At the same time, the EDI history suggests that Core Components alone might not be able to close the semantic gap entirely (Kelz, 2004).

Since the WSDL standard for Web Services only defines syntax and doesn’t include any semantic definitions, it is the responsibility of the service provider to deal with the resulting problems. To close this semantic gap, one can use the recent OASIS standard *UBL*²⁶, which is based on *xCBL*²⁷ and is harmonized with ebXML Core Component specifications (OASIS, 2004). UBL defines a set of standard business documents that build a common business vocabulary. Those documents can be used as a semantic layer for existing technologies such as Web Services, even though the EDI history suggests that it is unlikely that UBL will be the lingua franca of e-Business. Nevertheless, UBL can be used to add interoperability to Web Services (Gertner, 2004) or to migrate from Web Services to ebXML.

BUSINESS PROCESS MODELLING

Business transactions of any kind follow certain processes to ensure smooth business operation with predictable and agreed upon behaviour of the participating parties. In the past, those processes were usually not formalized, Modern companies use modeling tools such as ARIS²⁸ to represent, formalize, understand and ultimately optimize the processes relevant to their own organization.

Though it might be possible to develop and enforce a proprietary business model for internal processes (e.g. by using an integrated platform such as SAP), this is not feasible for transactions that go beyond company boundaries. Therefore, the goal of BPI²⁹ is to integrate the systems of individual companies to carry out business processes smoothly based on changing customer requirements and with varying partners. Figure 8 shows how the applications of different companies are integrated to work cooperatively on the same business process.

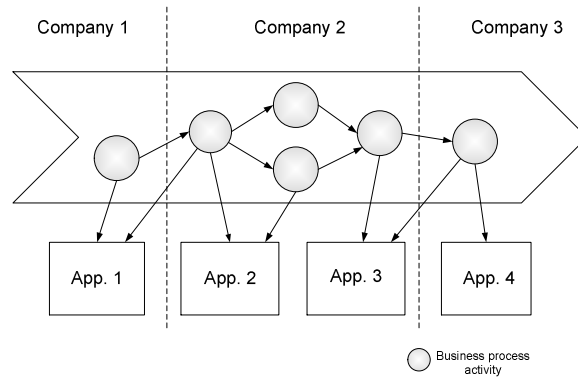


Figure 8: The goal of Business Process Integration is to integrate the existing systems of individual companies into a single cooperative operating system.

The great challenge of BPI is to find and introduce a global and cross-industry standard to formalize business processes so that individual companies can interact in this manner. Following the general movement in the e-Business community, such a standard should create a machine-readable definition of interactions between business partners to build a declarative system rather than a procedural one (Chappell, 2001). In addition, the transactions between partners have to be non-repudiable, legally binding and transmitted in a reliable manner.

The innovative *Business Process Specification Schema* (BPSS) among ebXML standards promise to solve the above problems. BPSS “*provides a standard framework by which business systems may be configured to support the execution of business collaborations consisting of business transactions. ... The Specification Schema supports the specification of Business Transactions and the choreography of Business Transactions into Business Collaboration*” (UN/CEFACT and OASIS, 2001a).

BPSS provides the semantics, elements, and properties necessary to define business collaborations rather than business processes. BPSS defines the roles that partners may fulfill. It consists of one or more choreographed business transactions and describes the type of business information that needs to be exchanged. BPSS can be used independent of ebXML to capture and communicate business processes that can be understood by all participating parties without ambiguity.

A BPSS instance is composed of:

- business documents
- business transactions (protocol to exchange the documents)
- binary collaborations (collaboration of transactions)
- multiparty collaborations (composition of one or more binary collaborations)
- substitution sets (replacing existing document definitions for the purpose of specializing collaboration definitions for a specific industry)

In summary, a BPSS instance specifies all business messages and their content, sequence and timing.

BPSS is designed to accommodate any kind of payload, so it is possible to use the ebXML Core Component Framework to design machine-readable business documents. In order to ensure

message reliability, BPSS provides a message reliability layer which is distinct from the ebXML Messaging Service layer. The aspect of non-repudiation is based on digital signatures as specified by the W3C XML-DSIG, while legally binding transactions are created by simply using an associated property within a binary collaboration. Substitution sets allow for existing vertical standardization organizations to define reusable process specifications.

The Web Services community also works hard to enable Business Modelling and Workflow Management. Some of those standards are ***Business Process Execution Language*** (BPEL) and ***Business Process Modelling Language*** (BPML) - languages that enable Web Service Composition and Web Service Choreography.

BPEL describes:

- the sequence of activities
- the triggering conditions of activities
- the consequences of executing activities
- partners for external activities
- the composition of Web Services
- the binding to WSDL

The abilities and scopes of BPEL and BPML do not differ significantly (Mendling *et al.*, 2003). One of the major disadvantages for both is that both can automate a sequence of messages but not execute actual transactions. While the ability to automate transactions is essential for a full-scaled e-Business system, such as one that uses ebXML, even the automation of a few steps leading to a transaction can be a big cost-saver. For smaller-scaled systems, BPEL or BPML might just be the tools to add some aspects of e-Business to existing Web Services systems (Fogarty, 2004).

Since they do not provide data transformations, human workflow, trading partner agreements or the support of existing business protocols, BPEL and BPML could certainly be seen as inferior when compared to ebXML. But those standards do not promise to provide full-scale e-Business over Web Services. They aim to compose Web Services, which is precisely what they do. There are other standards, such as Web Services Choreography Interface (WSCCI), Web Services Conversation Language (WSCL), and DARPA Agent Markup Language-Service (DAML-S), that aim to solve particular problems in the field of business process modelling.

The big difference between BPEL and BPSS is the point of view from where the collaboration is described. BPSS describes the collaboration from a neutral view, i.e. it describes how party A and party B interact. BPEL describes a collaboration from the point of view of the involved partners, i.e. how party A interacts with party B and party C. If B and C interact in the same multi-party collaboration as well, this cannot be seen from the BPEL file of party A. Currently, the W3C conducts the work on WS-CDL which describes a choreography of Web Services from a neutral perspective. From above, one can see that BPEL supports multi-party definitions. For BPSS, though there is a tag for multiparty collaboration, it is composed by several binary collaborations.

Currently all the modelling languages in Web Services have software implementation. BPSS has no direct implementations. But it is possible that by binding existing implementations from Web Services to BPSS specification, BPSS can be implemented. (Chappell, *et al.* 2001) gives binding between BPML and BPSS, and binding between XLANG and BPSS.

TRADING PARTNER AGREEMENTS

Most operational e-Business infrastructures focus on the automation of established (static) business relationships where the partners already know each other and have made arrangements with which to carry out business. The e-Business system simply automates those existing arrangements. But the e-Business community suggests the development of systems that support highly dynamic business relations. Such system must be able to automate the process of setting up new collaboration agreements on an ad-hoc and time-limited basis.

Currently, ebXML defines CPP and CPA which are the technical part of Trading Partner Agreement. Or more specifically, CPP and CPA define the technical runtime environment.

Within ebXML, this demand is addressed through the *Collaboration Protocol Profiles* (CPP) and *Collaboration Protocol Agreements* (CPA). A CPP defines the technical parameters of the message exchange capabilities, and a CPA is the agree technical parameters for message exchange. We described how they are used when an ebXML forms a process in the second section. CPP and the CAP define the technical runtime environment of collaboration.

Web Services specifications do not allow the descriptions similar to CPP and there is no agreement between partners like CPA. The protocol binding is fixed by the service provider. It is a simpler, but less flexible solution.

INDUSTRIAL SUPPORT AND COMPLIANCE

Web Services are well accepted and supported by industrial companies and W3C. Many large companies, such as SUN, IBM, Microsoft, HP and SAP have their implementation of Web Services specifications, such as SOAP, WSDL and UDDI. Information about these software packages are not difficult to find from their Web sites. Many other service providing companies, such as Amazon.com, Google, and eBay use SOAP as an interface to their platform. Obviously, Web Services become a strategic direction in e-business companies. (Hogan, 2003) reports that IDC predicts global spending for Web Services will be US\$15.2 billion in 2008, up from US\$3 billion in 2003. (Correia, 2003) reports that by 2006, 99 percent of all new products for Application Integration will have some level of support for Web Services, while the market for Web Services-enabled IT professional services will be US \$29 billion.

Compared to Web Services, ebXML is less accepted. UN/CEFACT TMG estimates that the acceptance rate of ebXML is only about 3% of that of Web Service. Especially ebXML is less accepted by small and medium enterprises. However, there are still many implementation projects from various organizations and companies. Here we just list some of the players.

- Sun Microsystems (<http://www.sun.com/software/xml/developers/regrep/>)
- Korea Institute of eCommerce (<http://www.ebxml.or.kr/>)
- Korea Trade Network (<http://www.GXMLHub.com/english/index.html>)
- XML Global (<http://www.xmlglobal.com>)
- XML.gov registry (<http://xml.gov/registries.htm>)

- DISA: Open Travel Alliance and Interactive Financial Exchange Forum (<http://www.disa.org/drive/>)
- Seeburger (<http://www.seeburger.com>)
- Drummond Group (<http://www.drummondgroup.com/>)
- Sterling Commerce (<http://www.stercomm.com/>)

Yet many other companies, such as bTrade, CDC, Cyclone Commerce, eXcelon, Fujitsu, GXS, IPNet Solutions, Sybase), have their ebXML projects.

While Web Services are a well-adopted standard for system integration throughout business sectors, ebXML still lacks industry support. But it is quite evident that soon, ebXML will be the state-of-the-art technology for global cross-company and cross-industry system integration. When a business is planning its overall system integration strategy or specific integration tasks these days, it is advisable to keep emerging standards such as ebXML in mind. In order to reduce the cost for system integration and interface building, companies might want to aim for a consistent integration strategy that leads to uniformity of system interfaces. Existing strategies might have focused on in-house applications only, treating gateway systems as a “whole different world”. But as indicated earlier, it is possible to merge both realms.

Since Web Services and ebXML use the same technological foundations, the task of (slowly) migrating from one technology to the other does not require exchanging the underlying infrastructure. At the same time, even a step-by-step migration is possible. Standards such as UBL can add ebXML-compatible semantics to Web Services, while the implementation of the ebXML Messaging Service allows for Web Services to use secure and reliable message transfer. Since ebXML is modular and uses the same technologies as Web Services, businesses can pick individual modules to deal with the integration tasks at hand. At the same time, they protect their investments because they ensure that the modules they implement now for use with existing Web Service interfaces can still be used if the system is switched entirely to ebXML in the future.

But even if no such full migration is wanted, companies can take advantage of the fact that, if they use Web Services for in-house integration and ebXML for cross-company integration, they use compatible technologies. Plus, they can always upgrade individual modules and without the need to use different experts for internal and external interfaces.

CONCLUSION AND OUTLOOK

Web Services and ebXML have many things in common and can complement each other. Both technologies provide solutions to integration problems, both use XML over Internet for Message interchange, and both approaches share a common high-level architecture. Observing the e-Business world reveals the evolution from tactical systems with limited scope to strategic e-Business initiatives. This does not mean, however, that Web Services will soon be abolished and replaced by ebXML.

Web Services are a well established and widely adopted standard. A multitude of experienced developers use the numerous available libraries and frameworks to guarantee short time-to-market for their products. In addition to those strengths, the Web Services domain is much

broader than that of ebXML and its architecture is simpler and easier to handle. **As a successor of other middleware technologies, Web Services excel in intra-enterprise request/response type application integration environments.**

At the same time, real-life business – especially in the B2B domain – is far more complicated than a collection of request/response pairs. This is why many initiatives have begun to add layers of powerful business functionality, such as reliable messaging, security and business process orchestration, to Web Services. But while these aspects were successfully defined within ebXML, the Web Services community could endanger all its efforts through divergence over those technologies.

If Web Services want to be more than a middleware standard for intra-enterprise application integration, the Web Services community will have to specify the layers of business standards used to support the complex and collaborative business transactions that organizations demand.

On the other hand, ebXML is a complete solution focused on B2B integration scenarios. It is not surprising that **ebXML excels whenever it comes to inter-enterprise business process integration.** But ebXML is also suitable for intra-enterprise business process integration, especially when departments of large enterprises are treated as separate companies. Moreover, since ebXML is modular, an enterprise could use single ebXML modules for in-house application integration projects (e.g. pick the ebXML Messaging Service to add reliable and secure message transfer to an Enterprise Application Integration project).

The major drawbacks of ebXML are that the specification is not entirely complete and that industry support is still lacking. If industry fails to provide affordable implementations of ebXML, this standard might follow the destiny of EDIFACT, which was not widely adopted due largely to its cost. Since ebXML is powerful, implementations are likely to be complex and might not be easy to handle. Templates for the most common demands of companies might help to decrease the time-to-market for system providers that use ebXML implementations.

For the global community, an Open-ebXML initiative is likely to trigger a whole new industry that could have the potential to change the way we view system integration. So far, several attempts have been made to provide an open source implementation of ebXML, but none has reached a level of maturity that suggests use in commercial applications.

While ebXML is always intended for e-Business, Web Services is a bottom-up technology that focuses on the technical aspects of middleware functionality. However, for many integrational projects (especially in-house) companies do not need full grown e-Business suites. Instead, they need smaller, more reliable, and easier to handle technologies that have reached a sufficient level of maturity.

One interesting topic for system architects might be to create migration paths between Web Services and ebXML by taking the modules of ebXML and enabling them to be used with Web Services while at the same time suggesting a step-by-step migration path. Companies that already use Web Services might be more interested in using certain aspects of ebXML in conjunction with their existing Web Services infrastructure. As their products evolve, they might consider adding more modules until their product is, in fact, a full ebXML framework. If such a migration follows a specified plan, migration issues can be reduced.

REFERENCES

- Alonso, G., et. al., (2003). *Web Services – Concepts, Architectures and Applications*. Springer Verlag, Heidelberg, Germany.
- Barton, J., Thatte, S. & Nielsen, H.S. (2000). *SOAP Messages with Attachments*. Retrieved January 29, 2005, from <http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211>.
- Chappell, D., et. al. (2001). *Professional ebXML Foundations*, Birmingham, Wrox Press Ltd. Birminham, UK.
- Cohen, F. (2002). *Understanding Web Service Interoperability*, Retrieved December, 2004, from <http://www-106.ibm.com/developerworks/webservices/library/we-inter.html#4>.
- Correia, J. et al. (2003). *Gartner Sheds Light on Developer Opps in Web Service*, Corte Madera (USA), Integration Developers News LLC. Retrieved January 29, 2005, from <http://idevnews.com/IntegrationNews.asp?ID=69>.
- Fogarty, K. (2004). *Business Process Execution Language*, Ziff Davis Media. Retrieved January 29, 2005, from http://www.baselinemag.com/print_article2/0,2533,a=123575,00.asp
- Gertner, M. (2002). UBL and Web Services. *XML-Journal*, 3(6), 16-19.
- Gudgin, M. (2003). SOAP Version 1.2 Part 2: Adjuncts, W3C, Retrieved January 29, from www.w3.org/TR/2003/REC-soap12-part2-20030624/.
- Gudgin, M., et.al., (2003), SOAP Specification 1.2, <http://www.w3.org/TR/soap12-part1/>
- Hogan, J. (2003). *Gartner: Web services projects riding out budget cuts*. Retrieved January 29, from WebServices.com.
- Kelz, W. (2004). Allheilmittel? Die Universal Business Language, XML magazine & Web Services. Retrieved January 29, 2005, from http://www.xmlmagazin.de/itr/online_artikel/psecom,id,571,nodeid,69.html.
- Mendling, J. et al. (2003). *A Comparison of BPML and BPEL4WS*, Retrieved January 29, from wi.wu-wien.ac.at/~mendling/talks/BXML2003.pdf.
- OASIS (2001). *ebXML Technical Architecture Specification v. 1.0.4*, ebXML Technical Architecture Project Team, OASIS. Retrieved January 29, 2005, from www.ebxml.org/specs/ebTA.pdf.
- OASIS (2002a). *Message Service Specification Version 2.0*, OASIS, ebXML Messaging Services Technical Committee. Retrieved January 29, 2005 from www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
- OASIS (2002b). *ebXML Registry Information Model*. ebXML Registry Technical Committee OASIS. Retrieved January 29, 2005, from www.oasis-open.org/committees/repreg/documents/2.1/specs/ebrim_v2.1.pdf.
- OASIS. (2004). *Universal Business Language 1.0*, Retrieved January 29, 2004, from <http://docs.oasis-open.org/ubl/cd-UBL-1.0/>.
- UN/CEFACT & OASIS (2001b). *ebXML Technical Architecture Risk Assessment Version 1.0*, ebXML Security Team, UN/CEFACT and OASIS. Retrieved January 29, 2005, from lists.oasis-open.org/archives/security-consider/200103/pdf00000.pdf.
- UN/CEFACT & OASIS, (2001a). *ebXML Business Process Specification Schema Version 1.01*, UN/CEFACT, OASIS. Retrieved January 29, from www.ebxml.org/specs/ebBPSS.pdf.
- UN/CEFACT (2003). *Core Components User's Guide*, Retrieved January 29, 2005, from http://www.ecp.nl/ebxml/docs/cc_ug_oct03.pdf

¹ ebXML: Electronic Business using eXtensible Markup Language

² CORBA: Common Object Request Broker Architecture

-
- ³ TPM: Transaction Processing Monitor
- ⁴ RPC: Remote Procedure Call
- ⁵ W3C: World Wide Web Consortium
- ⁶ SOAP: Simple Object Access Protocol
- ⁷ WSDL: Web Service Description Language
- ⁸ UDDI: Universal Description, Discovery and Integration
- ⁹ EDI: Electronic Data Interchange
- ¹⁰ UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business
- ¹¹ OASIS: Organization for Advancement of Structured Information Standards
- ¹² ANSI X12: “American National Standards Institute X12”, X12 stands for the originator of this standard: the Accredited Standards Committee X12 (ASC X12)
- ¹³ EDIFACT: Electronic Data Interchange For Administration, Commerce and Transport
- ¹⁴ EAI: Enterprise Application Integration
- ¹⁵ B2Bi: Business-to-Business Integration
- ¹⁶ BPI: Business Process Integration
- ¹⁷ XML: eXtensible Markup Language
- ¹⁸ URI: Uniform Resource Indicator
- ¹⁹ ODETTE: Organisation for Data Exchange by Tele Transmission in Europe
- ²⁰ SAP: Systeme, Anwendungen, Produkte in der Datenverarbeitung. The SAP AG is the third largest independent software supplier in the world and known for its enterprise software products. (see www.sap.com)
- ²¹ MIME: Multipurpose Internet Mail Extensions
- ²² QoS: Quality of Service
- ²³ UML: Unified Modeling Language
- ²⁴ B2B: Business to Business
- ²⁵ ISO: International Organization for Standardization
- ²⁶ UBL: Universal Business Language
- ²⁷ xCBL: XML Common Business Library
- ²⁸ ARIS: ARIS is an integrated product of the IDS-Scheer AG (www.ids-scheer.de) for design, implementation and controlling of business processes
- ²⁹ BPI: Business Process Integration