

Net.ObjectDays 2002

Web Services between demand and reality

Author: Dr. Torsten Schmale
CEO inubit AG

Abstract: By using powerful base technologies such as XML, SOAP and UDDI, Web Services today are state-of-the-art for electronic data exchange between applications, and for interaction between information and business transaction services on the net. Due to their flexibility and neutrality regarding programming languages and providers, Web Services are becoming increasingly important in all areas of commercial technology, and thus may be viewed as a mainstream development within IT.

Despite of all optimism and the obvious fundamental possibilities, the introduction barrier of this technology, which is created by established - even though often inefficient – procedures, and the inclusion of legacy systems should not be underestimated. Effective migration approaches and mechanisms to comprise today's IT landscape are required. Only then Web Service technology will experience a true market breakthrough after initial success reports of early adopters.

This article gives an overview of the current situation with Web Services both from a technical and an economic point of view. After a brief introduction of the basic principles, a closer look will be taken into what is available right now and what is under work or can be expected as far as the level of standards in the near future is concerned. Furthermore, Web Services sometimes considered as the ultimate way of process and system integration, are compared with other XML-based approaches like ebXML and RosettaNet.

With our explanations we try to bridge the arch of today's Web Services technology and previously used mechanisms, and demonstrate how, by the help of modern integration tools, the hurdles of migration and mixed environments can be overcome, and how the required integration of new and legacy systems can efficiently be realized.

1. Introduction

After the exaggerated hype of the e-Business market, nearly everyone has come back down to earth recognizing that the internet and all of the accompanying technologies will change business sooner or later. Nowadays, however, it is no longer a question of how do we have to change our business processes so that they fit to what the internet provides, but more a question of how can we make use of internet technology, doing what we always did, but more efficiently, quicker, more reliable using less resources, so that competitive advantages and reduced costs can be realized.

Although globalization was already a driving factor, the internet and web activities have once more triggered companies in all industries and markets to intensify usage of IT in order to improve the way they are linked with their business partners along the supply chain. While in the past, companies were mainly focused on their internal IT, they are now slowly but steadily realizing that the focus on where the race will be won in the future is shifting to externalization and integration of business processes, i.e. linking themselves electronically with their suppliers, partners, and customers [SCH02].

E-Collaboration is the new buzzword from an economic point of view, and it does not mean anything but the continuous migration from a closed community of function-oriented working companies to an interlocked network of process-oriented companies. These make up a supply chain network, where systems, users, and processes are the building blocks handled and managed in a joined approach – the age of Web Services has started – that's what you think…

Well, before we get too theoretical and visionary, let's take a brief look at how we got to where we are today. Traditionally, software applications, as well as organizations in their internal structure, have been realized mostly along the line of a functional model. For example, applications in the past were developed to support the selling of goods and services (CRM, SFA, ERP etc.).

With the internet, or better, with web technology on top of it, the business community realized that this might be a good new infrastructure to let in particular customers from every place in the world, at any time of day or night, take notice of the offered goods and services. Quickly, everybody tried to attract these customers with websites, e-shops, e-marketplaces, and other types of portal applications, again with a pure functional view, and a pure person-to-system (P2S) interaction, disregarding backend integration with systems on both sides of the fence. The result of this dotcom approach was that most of these things did not work as expected, and people and companies did business as usual, i.e. without using these brave new portals – the end of the hype.

On the other side, connecting backend applications system-to-system (S2S) on lower technical levels was developing slowly, and nearly without attention from the business community, since it was always viewed as a necessary evil of importing and exporting on a pure data level.

With applications becoming more complex, distributed, and in consequence interacting more often, the client-server paradigm entered the scenery, and took more and more room from the classical mainframe and server-centric applications. Numerous methods for application interaction following the request-response or fire-and-forget model where developed and introduced to the market, such as SUN-RPC, DCE-RPC, CORBA, DCOM, RMI to name only a few of them.

In the late 90's, exchanging business data such as pricing catalogs, invoices, and orders between companies via EDI were introduced mainly in the automotive and the financial industries. This was already a quite advanced approach from today's point of view, but could

not penetrate the industry as a whole due to weak and thus quickly blurred standards, and due to closed transportation networks (VAN) being expensive to operate and to use. The effect of this early externalization attempt was that mostly large enterprises made use of it, while small and medium sized companies, the larger part of in the supply chain, did not.

To complete the view, we should not forget the influence that the printing and publishing industry had trying to separate content and presentation in a standardized way – the birth of SGML, as an ancestor of today's XML, which we all love due to its simplicity, vendor and system independence, wide range of use, and tremendously strong acceptance worldwide.

2. Web Services – a combination of proven technologies

In this context, Web Services have been born as a natural evolution of different technologies, approaches and demands from communities in business and technology. Web Services are now lining up to tie everything together, and to become the leading way of exchanging information, making use of system and business services along the web, in a vendor and system-independent way. Web Services, as acting on the S2S-level, are going to let systems and services find, bind and execute each other with the minimum of user invocation – let's take a look at where we are today…

2.1. Definition

It has proven to be far from easy to agree on a definition of what Web Services are. There are many different definitions and we will confine to just two. According to the W3C's Web Service Architecture Group [JEN02]:

> "a Web Service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered by XML artifacts and which supports direct interactions with other software applications using XML-based messages via internet-based protocols".

The Web Services Interoperability Organization (WS-I) seems to have a more generic view. The FAQ states somewhat fuzzy that

> "WS-I views Web Services as having a general purpose architecture with inherent interoperability that supports B2B scenarios, but extends far beyond in scope and functionality".

The second definition implicitly carries the notion of support for document-centric, loosely coupled business collaboration. Web Services are based on the request-response-model with no business process semantics. This is in keeping with today's three specifications that enjoy broad acceptance in the vendor community: SOAP, WSDL, and UDDI.

2.2. Communication and transport via SOAP

SOAP, the Simple Object Access Protocol, is an enhanced version of the classical RPC (Remote Procedure Calls) based on the request-response principle. Besides RPC/encoded style, Web Services support document/literal style, meaning that an XML document is sent as the payload of the SOAP message. The requester sends a message to the service, the service processes the message, and sends back a response message. In essence, the requester (client) has no knowledge of how the service is implemented, or how it processes the message (the XML document). It is particularly suitable for passing business documents asynchronously between loosely coupled services.

Similar to the Open Source competitor XML-RPC, SOAP makes use of HTTP(S) as the underlying transport protocol, which was quite a good decision from the architects. Compared to CORBA, being based on its own IIOP not passing through firewalls easily, SOAP lets systems interact through their respective electronic barrier without changing firewall configurations, as only standard web sockets are used. Furthermore, Web Services based on SOAP do not require integration of an ORB (Object Request Broker), a task that is not for the faint of heart.

Aside from that RPC, CORBA and Web Services do share the principle of an interface description language (IDL) to specify the server interface out of which the appropriate client and server stubs and marshalling procedures can be generated.

2.3. Interface definition with WSDL

WSDL is this IDL for Web Services. It is an XML definition of the server interface and properties in an object-oriented manner. Unlike XML-RPC, SOAP makes use of some advanced mechanisms originally introduced with CORBA such as exceptions for the indication of errors raised during execution on the server side.

Furthermore, it allows to define own data types transmitted between client and server based on XML tags. Finally, WSDL is completely independent and variable in terms of encoding rules and the used transport protocol, which at least in theory can be applied by use of protocols other than HTTP(S).

2.4. Registration and localization via UDDI

Now, having described the mechanisms for the server interface, parameter passing and transport protocol, the question remains how these Web Services can be located, bound and executed in a common and machine readable manner?

Well, even this has been solved to a great extend in before by the former technologies like DCE (Distributed Computing Environment), CORBA and DCOM. With their concepts of a global and distributed directory service, now described with XML and enhanced with human readable information that everyone knows from the yellow pages, UDDI was born – the Universal Description, Discovery and Integration service.

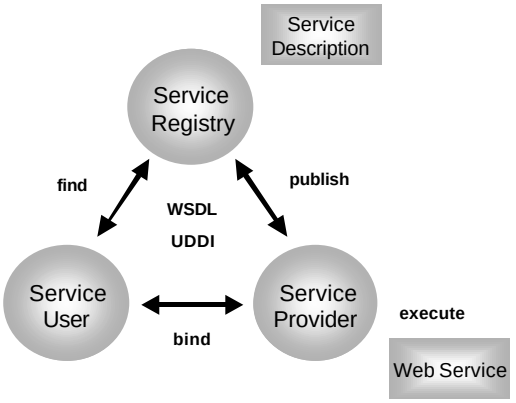


figure 1 Principle interaction model of Web Services

Primarily, UDDI is a global business registry with its root directory under www.uddi.org. Here, companies and services can register their Web Services and offerings and UDDI client applications can look up for desired information and services.

The UDDI-registry is made up of three types of information:

- White Pages, with information regarding the companies
- Yellow Pages, for the lines of business in different markets and segments
- Green Pages, with information regarding business model, processes and services

The UDDI specification now released in version 3 is quite mature and easy to use due to several application kits being available to work with this registry. In figure 1 the principle interaction model of a Web Service user, provider and the registry is shown.

3. Web Services under work – extending the basis

Well, the basic mechanisms of Web Services have now been introduced with SOAP, WSDL and UDDI and it can be seen as a bare minimum only of an infrastructure for distributed computing.

Being familiar with what DCE and CORBA have already standardized, live and kicking in numerous products and projects, we still miss a lot more in the future to give Web Services a lasting effect in this market.

3.1. Security Mechanism

Web Services still lack security mechanisms, which is a major gap today, and limits the application for B2B due to the absence of services for authentication, encryption, access control, and data integrity.

Currently, Web Services cannot certify the identity of the publisher or consumer of a Web Service. In addition, there are no facilities to restrict access to Web Service to a group of authorized users. Several initiatives have been launched to address some of the shortcomings.

The XML-based Security Services Technical Committee from OASIS (Organization for the Advancement of Structured Information Standards) is working on SAML (Security Assertion Markup Language). This specification allows organizations to exchange authentication, authorization and profile information securely with their partners. In parallel, OASIS is also working on XACML (extensible Access Control Markup Language), a specification that enables organizations to limit access to services to authenticated, authorized users.

On lower levels, other system standards are under development, such as XKMS (XML Key Management), which aims to reduce the complexity of PKI (Public Key Infrastructures) by making trust utilities available on the open internet.

3.2. Interaction, process modeling and transactions

Web Services today are API-centric and are mainly defined by the static description of the server interfaces with WSDL. As such, there are functionally very close to the pure client-server communication like the good old RPC.

However, integrating on a business process level means to have a view of these processes and the ability to model and manage also the dynamic in between them. Several specifications are

currently under way all describing the sequence and interaction of business processes between Web Services. They can be viewed as a type of glue between individual Web Services acting as distributed subroutines. Ton name only some of them: **WSFL** (Web Service Flow Language) by IBM; **XLang** (Microsoft's XML Language); **BPML** (Business Process Modeling Language) and **WSCI** (Web Services Choreography Interface) backed by SUN.

Brand new and after a first shake-out in this zoo, **BPEL4WS** (Business Process Execution Language for Web Services) now supported by Microsoft, IBM and BEA is now becoming successor of WSFL and XLang. BPEL4WS is also co-related with two other specifications for the message coordination and transaction handling (WS-Coordination, WS-Transaction). As there is obviously still a lot of initial work to be done here, we quickly pass over to the next working area.

3.3. Presentation layer for Web Services

Web Services currently lack a mechanism to encapsulate a user interface, which is the key to being able to package an application and embed it into another application. It is similar to WebEDI, which is nowadays used to let users view, create, and maintain electronic business documents such as orders, invoices, etc. Since they are completely based on XML data, Web Services are well prepared to be extended by XSL/XSLT-based views to transform the results of a Web Service invocation into display code such as HTML or WML (Wireless Markup Language).

The Web Service User Interface (WSUI) initiative, announced in June 2001, defines the concept of a view, which represents a single display screen of a Web Service. WSUI follows the MVC (Model View Controller) principle, and uses a simple schema for describing a WSUI "component" that calls backend SOAP and XML services. By adopting WSUI, vendors, customers, and developers gain the ability to dynamically share applications across web sites without being concerned with the backend implementation of service interfaces. WSUI is now in the state of a first draft.

On 28 January 2002, OASIS, the XML interoperability consortium, announced its members have formed the OASIS Web Services for Remote Portals (WSRP) Technical Committee to create an XML and Web Services standard that will allow the plug-n-play of visual, user-facing Web Services with portals or other intermediary web applications. These WSRP services will enable businesses to provide content or applications in a form that does not require any manual or application-specific adaptation by consuming portals and applications. WSRP will enable distributed portal systems, where portals share portlets as visual, user-facing Web Services for integration with other portals. WSRP will allow remote portlet Web Services to be implemented in a variety of ways, including Java/J2EE and Microsoft's .NET platform.

1.1 Other challenges

In absence of the mentioned standards, whose establishment is probably only a matter of time, one could also argument about principle issues not being addressed by Web Services such as the question about how to cope with potential failure conditions of invoked Web Services [JEN02]:

- What happens if a Web Service implementation is not available? For example, if the service provider has revoked the Web Service implementation or the system is down, the requesting application may be unable to proceed. There is no way to discover unavailability before service invocation.
- What happens if the service provider changes the implementation in a way that it is no longer compatible with the interface and requires a modification of the interface?

In that case, the service requester must also migrate to the new interface. If the transition goes unnoticed, the service requester will receive an error. Analysis and resolution may require considerable time and may be awkward in time-critical situations.

- How do service requester and service provider understand each other? WSDL and SOAP provide for technical interoperability, but not for semantic interoperability. There is no guarantee that service requester and service provider have a common understanding of the message content. For example, the provider returns a value in the "Total amount" field. Now, does it include VAT or not?

Web Services technology is about loose coupling in the technical sense. However, it is tight coupling in a conceptual sense. If anything changes that affects the availability of a Web Service, the interface of a Web Service or the semantics of a message, the service provider would need to proactively notify service requesters. This, however, would be far from easy, since a service provider cannot tell who will use a particular Web Service and when. The use of a Web Service does not require an agreement between requester and provider.

There are obviously numerous fields of activities, where the young Web Service technology can be improved in the future. However, we should not forget what the current situation in B2B information exchange is looking alike: people send orders and invoices via fax, or still via good old snail mail, other information is exchanged via simple phone calls or personal meetings – all this is far away from being reliable, quick, and secure with transactional ACID properties as one might ask for.

So, we should take one step at a time, and take care not to be too keen to get all this within the next few months. However, Web Services standardization committees are not the only groups trying to address these needs. In the last couple of years, ebXML and RosettaNet, both based on XML document exchange as well, have evolved to serious competitors of Web Services, or are they simply add-ons?

4. Web Services, ebXML, RosettaNet – friends or foes?

As explained in [JEN02], ebXML and Web Services seems to significantly overlap in their objectives to enable enterprises of any size, in any location to exchange business documents.

ebXML represents a set of modular business collaboration-oriented specifications. Business collaboration as understood here, requires a solid and consistent conceptual foundation, encompassing the concept of inter-enterprise business collaboration based on mutually accepted trading partner agreements, as well as the concept of a technical infrastructure, which enables businesses to find each other, and provides for reliable and secure exchange of business messages between collaborating business partners.

Like Web Services technology, ebXML provides technical interoperability through a vendor-neutral protocol. However, there is no such thing as a service interface description, as with Web Services in the form of WSDL. Instead, ebXML uses Collaboration Protocol Agreements (CPAs) to declare bindings to business collaboration specifications.

ebXML requires collaborating partners to mutually agree upon the formats and semantics of business documents, which are XML-encoded. However, it is not an actual constraint to only allow XML-encoded messages within ebXML, which would even leave room for transmitting EDI messages. In fact, one of today's largest-scale ebXML application in North America combines ebXML and EDI technologies in exactly this way [KOT02b].

In an inter-enterprise business collaboration scenario, both business partners would use the ebXML Message Service to securely and reliably transport business documents. However, the ebXML Message Service just represents the message envelope. It requires an additional content standard to define the semantics of a business document (which represents the content of the envelope, i.e. the message payload). This is still a weak point in ebXML due to several different proposals.

Of course, ebXML-based business collaboration has to take failure conditions into account as well. Since the ebXML Message Service caters for reliable and recoverable message exchange, there is not too much to worry about. Transport level failures are completely taken into account with the ebXML Message Service. However, there are certain situations when the ebXML Message Service is not reliable enough, which may happen under system failure. This is nothing new, but a known scenario from the EDI world as well.

Business level failures are completely taken into account with the Business Process Specification Schema (BPSS). For example, if a party fails to respond within a pre-defined time period, then the BPSS reverts to the previously known secure state.

As shown before, Web Services address the business process specification currently only vendor-specific with WSFL, XLANG and other specifications. However, things are moving quite fast here and with BPEL4WS a unified specification supported by the major vendors seems to substitute them together with BPML, another approach from a non profit organization.

ebXML and Web Services are today complementary technologies – but we expect this will change very soon. In general terms ebXML is about loosely coupled document-centric business collaboration, while in Web Services, the API-centric service interface dominates rather than the document-centric view in ebXML. Web Services offer some of the same business functions of ebXML such as messaging, service descriptions and registries – but in a more component-oriented and more open form.

In terms of standardization, today ebXML is more advanced than Web Services. It provides for reliable business collaboration and inter-enterprise process automation. In the Web Services space, we currently just have SOAP, WSDL and UDDI as widely accepted specifications, although it seems that the missing parts are brought together with an amazing speed. Of course, enhancing Web Services in this way will result in lots of duplications compared to ebXML, however, wasn't that also the case with DCE, CORBA, DCOM and EJB?

As the past has shown too often, simple and modular solutions are always preferred by the market and will finally make the race (compare to SNMP versus CMISE/CMIP, SMTP versus X.400 or TCP/IP versus OSI 7-layer stack to name only some of them). Thus, ebXML was probably to early to become widely accepted in relation to Web Services. According to recent stories, e.g. [KNOR02], the momentum for ebXML has stalled, while Web Services seem to have just started a passing maneuver that will be decided quickly.

The same could be true for RosettaNet, another process integration framework like ebXML. RosettaNet again has own definitions of standard processes (here called PIPs) and a slightly modified transport protocol based on HTTPS. However, while Web Services and ebXML are still in their infancy, RosettaNet is already up and running in and between a continuously increasing community of companies. As such, RosettaNet is ahead of all others from that point of view.

It will be interesting to see how all three approaches will position themselves in the market and against each other. But we can be pretty sure, that even if one will position better than the others, we will have to live with all of them over the next years. Together with outdated but still widely accepted EDI technology (according to [KOT02b] still used in projects in a 6:1 ratio compared with the others!) the complexity of mixed environments is again increasing and

needs to be addressed when we really want to solve integration demands which the market requires today.

5. No Web Services without integration

Although Web Services aim to be the ultimate solution for all aspects of internal and external integration, no one really believes in this dream to become true in a short term – if at all.

Besides the pure feasibility, 'fully interoperable standard-based Web Services' are even not desired by major players in the market. We still haven't forgotten the battle in the web browser arena.

This kind of political and business tactical issues will most likely also lead to proprietary elements and features within available Web Service platforms such as SUN Java One and Microsoft .NET Myservices and other activities such as HP Netaction and E-Speak.

5.1. Technology mix in the next year

As a consequence system integrators who take care of the real project business out in the market, already know for sure that Web Services, although being the most reasonable way to integrate, won't solve their problems in short or mid term.

Each new technology goes along with several variations and proprietary enhancements that considered even without other technologies do result in a bunch of incompatible applications, e.g. the EDI world with more than 100 variations of the 'standard' not counted specific customizations made as 'company standards'.

Business integration along the supply chain is coping with the most heterogeneous IT environment one can imagine (e.g. in Germany alone, more than 2000 vendors of different ERP systems are known). Thus, introducing Web Services as one more technology to exchange information cross and in between enterprises will not lead to a big bang revolution.

The only way to overcome this situation as long as legacy systems have not been removed from the market – which will never happen – is to accept the world as it is – heterogeneous. This in mind, new developments in the integration server market are quite important and do finally also open the way for a real breakthrough of Web Service technology.

5.2. Breakthrough via modern integration servers

Today's integration market is separated into different segments, which are [MAR02]:

EAI (Enterprise Application Integration), integration platform that provides the core infrastructure to integrate business processes between applications within the enterprise. Traditionally, EAI has focused on integrating at the application level, and lacks functionality on the process level, as well as the desired functionality to integrate across the electronic border of companies.

EDI (Electronic Data Interchange), definition and exchange of business documents, in rather complicated and only machine-readable document structures, in order to support integration on a pure data level.

ETML (Extract, Transform, Load and Manage), an integration approach used to prepare typically mass data from different sources for analytical evaluation in BI (business intelligence) tools like data warehouses.

IEI (Inter-Enterprise Application Integration), a type of integration platform to integrate and automate business processes between applications of different companies along the supply chain.

While EAI and IEI focus already on process integration, ETML and EDI are more on the data level that can be seen as a former and more basic integration layer that will probably be replaced in the future by products from the two other departments.

The necessary convergence of these traditionally different approaches is obvious. No customer wants to invest in four different platforms all going in the same direction. However, most customers are not yet aware of investing already in these different types of integration tools, due to the fact that different teams in the company try to solve different aspects (e.g. on the sell side and buy side of integration, or internally to overcome limitations of legacy systems still requiring paper-based information exchange). Thus, most customers follow a bottom-up approach when selecting tools for their specific problem.

In the future, integration will be more and more understood as a common problem that can be addressed with standard integration products in a top-down manner. Those few companies already having recognized this major trend do carefully look at the integration vendors, and decide in favor of products that have already been designed as universal integration servers built upon future-proof XML and J2EE technology.

Such products will quickly become important key elements in the IT structure, solving all kinds of internal and external integration demands, including support for Web Services, ebXML, EDI etc. in a mixed environment.

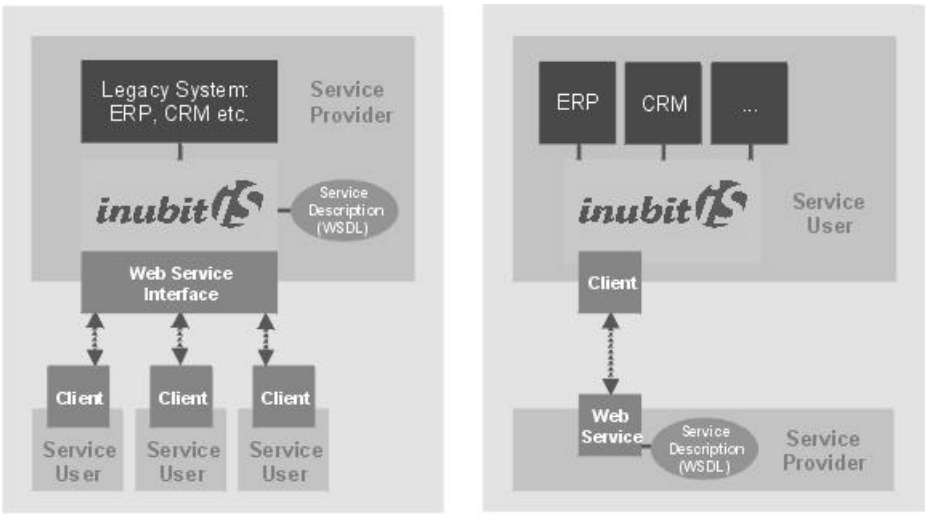


figure 2: Web Service with legacy systems

This fact is an important issue, and in the end, it will also be responsible for a real breakthrough of the Web Service technology as the de facto standard for application integration and application modularization in the sense of component-oriented software architectures.

If, and only if, the critical mass of installed applications fully supporting interoperable Web Service interfaces, transport, registration and lookup can be reached, a breakthrough will happen. The role of integration servers in this context will be to overcome the restrictions caused by the pure amount of older systems being incompatible with the world of Web Services.

In this case, integration servers will be the mediation technology, which on one hand can add a modern Web Service interface to a legacy application, and on the other hand, hook other Web Services into the execution chain of nearly arbitrary legacy applications. State-of-the-art integration servers like the inubit IS [INUBIT02] provides this already as shown in figure 2.

In addition to this migration demand, integration products acting on the process level also extend in the direction of another domain commonly known as BPM (Business Process Modeling). This is no surprise, since modeling business processes is of course required, when the integration of such processes will be addressed across all involved applications.

Finally, integration considered in a holistic manner also require the integration of user interaction at one point or another. Although manual interaction with the user should be reduced to a minimum for a maximum of process automation, the user will still be involved in releasing and supervising certain activities. Examples for this are releasing automatic orders above a certain financial value, or suppliers updating their pricing lists, which could cause the responsible user on the procurement side to look for another supplier with lower prices and so on.

As a result, modern integration servers will either add support for this kind of user interaction, probably close to standard approaches like WSUI or WSRP, or by considering this as another application that is to be integrated. Nevertheless, integration servers can be seen as the most likely infrastructure leading the way for Web Service technology to become really a mainstream within the near future.

6. Conclusion

A substantial amount of IT resources is spent each year on developing and maintaining point-to-point connections and interaction code between applications. Although Web Services technology is still in the realm of early adopters, it represents an attractive new direction in the development of the integration market and will probably be the crowd-puller to unify other approaches such as ebXML or RosettaNet.

Vendors and standards organizations have identified the key technical issues that inhibit Web Services from supporting true application integration in B2B environments today. Efforts are under way to address these issues, and key standards have been or will be proposed, but they are still months away from being finalized, and years away from being widely used in the industry.

Therefore, the already stable parts of Web Services (WSDL, SOAP, and UDDI) are all that can be used today for integration purposes on a wider scope. But even here, legacy systems are typically to be integrated and do not support even this basic technology today; instead, they come with ugly proprietary application interfaces.

Nevertheless, integration is required right now independent of Web Service, EDI or whatever technology. Fortunately, today's state-of-the-art integration servers such as the inubit IS are based on modern XML and J2EE technology already, while still fully supporting the legacy world. Therefore, integration servers will fill the integration gap between the old and the new world for at least 3-5 years from now on.

Even then, when the majority of systems in the field do have Web Service interfaces, integration servers currently evolving from EAI middleware to real process integration and modeling tools will play an important part in this scenery driving Web Services via standardized flow languages like BPEL4WS to orchestrate, administrate and supervise them.

7. References

[MAR02] Wolfgang Martin Team, IT-Research,
EAI im Wandel – Wertschöpfungsnetze durch kollaborative Geschäftsprozesse, Strategic Bulletin – April 2002,
www.it-research.net

[SCH02] Dr. Torsten Schmale, inubit AG,
IT am Scheidepunkt – Veränderte Anforderungen an Supply Chains im Zeichen elektronischer Märkte, page 218-230, Modelle im E-Business, 1. Aufl. – Paderborn, ALB-HNI-Verlagsschriftenreihe, April 2002

[KRUE01] Stefan Krüger,
Automatisierte Geschäftsprozesse,
E-Business Magazine, Oct. 2001

[WONG01] Sam Wong, VP Patkai Networks,
Web Services: The Next Evolution of Application Integration,
ebiz, http://e-serv.ebizq.net, 08/20/01

[JEN02] Dieter E. Jenz,
ebXML and Web Services - Friends or Foes?,
paper published at www.webservices.org, 06/27/02

[BAU01] Michael Bauer,
Web Services – Methoden und Tools für verteiltes Computing,
Computerwoche Magazine, 12/10/01

[INUBIT02] inubit AG,
Technical Whitepaper of the inubit Business Integration Server,
Release 2.1, www.inubit.com, published January 2002

[KOT02] Alan Kotok,
The e-business continuum: Web services, ebXML and EDI,
paper published at www.webservices.org, 06/27/02

[KOT02b] Alan Kotok,
Why doesn't EDI just die?,
paper published at www.webservices.org, 06/11/02

[KAL02] Muhammad F. Kaleem,
Transactions over web services – an introduction to the business transaction protocol, paper published at www.webservices.org, 05/17/02

Dr. Torsten Schmale, CEO inubit AG



is one of two founders of the inubit AG. From 1999 to 2001, he began as general technology manager and then became board member, responsible for the areas of strategy, development and professional services. In the beginning of 2002, he took over as CEO of the successful software development firm in Berlin.

From 1993 to 1999, Dr. Schmale worked as leading system architect in the software engineering department of Siemens AG in Munich, later in Berlin. He was responsible for the development of a new software generation for the management of continental communication networks - TMN. During this time, he was an active member of international and European standardization committees, such as CCITT today known as ITU-T and ETSI, where he supported normative definitions for worldwide telecommunication standards.

Dr. Schmale holds a master of science in electronics and computer science from the german university of Wuppertal and Duisburg in 1993, and was awarded a PhD at the university of Duisburg in 1998 on the subject of 'Object-Oriented Network and Systems Management in a Distributed Environment'.

As a recognized IT and business process expert, he has published numerous articles in several professional media on various subjects, such as electronic commerce, the optimization of business processes, as well as the escalation of companies' competitive advantages by means of process integration.

inubit AG

inubit is the leading supplier of process-oriented integration software for electronic commerce. The inubit Business Integration Server significantly increases the competitive advantage of companies, and guarantees a quick return on investment, by efficiently integrating business processes, enterprise applications, old systems, databases, and work processes with XML, EDI and web services technology, and by electronically connecting these with the systems of partners and clients.

Companies can thus drastically reduce their costs for procurement and sales, develop new areas of business, and improve their competitive position of the market noticeably. In addition, they are able to strengthen their customer ties, streamline their supply chains more efficiently, and thus optimize their existing business processes tremendously.