THE UNFOLDING OF THE WEB SERVICES PARADIGM

Akhil Sahai[1], Sven Graupner[2], Wooyoung Kim[3]

[1,2] Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo-Alto, CA 94034

[3] University of Illinois at Urbana-Champaign, Department of Computer Science, 1304 West Springfield, Urbana, IL 61801

akhil_sahai@hp.com sven_graupner@hp.com wooyoung@cs.uiuc.edu

OUTLINE

ABSTRACT

The past decade has seen the unfolding of the web services paradigm as a logical extension to the distributed system approach of the Internet and the increasingly popular “software as a service” approach. In this paper, we discuss the genesis of web services, the state of the art and the promises that the web services paradigm holds.

INTRODUCTION

There were two predominant trends in computing over the past decade – (i) a movement from monolithic software to objects and distributed components and (ii) an increasing focus on software for the Internet. Web services (or e-services) are a result of these two trends.

Web services are described as distributed services that are identified by URI's, whose interfaces and binding can be defined, described and discovered by XML artifacts, and that support direct XML message-based interactions with other software applications via internet-based protocols. Web services that perform useful task would often exhibit the following properties:

1. Discoverable: One of the foremost requirements for a web-service to be useful in a commercial scenario is that it be discovered by consumers (humans or other web services).
2. Communicable: Web services follow a message-driven operational model exchanging messages in XML syntax. The operational model is thus also referred to as Document Object Model. Various communication patterns are used between web services: synchronous, and asynchronous as well as transactional communication.
3. Conversational: Sending a document or invoking a method and getting a reply are the basic communication primitives. However, complex interactions between web services involve multiple steps of communication that are related to each other.
4. Secure and Manageable: Properties such as security, manageability, availability, and fault tolerance are critical for a commercial web-service as well as transactions, quality of service, and reliability.

A gradual paradigm shift from object-oriented monolithic software to software available as a service via the Internet is taking place.

THE GENESIS OF WEB SERVICES

Contrary to public perception, the development of web services followed a rather modest evolutionary path. The underpinning technologies of web services borrow heavily from object-based distributed computing and development of World-Wide Web (Lee 1996). In the chapter, we review related technologies that help shape the notion of web services.

Tightly Coupled Distributed Software Architecture

The study on various aspects of distributed computing can be dated back as early as the invention of time-shared multiprocessing. Despite the early start, distributed computing remained difficult until the introduction of Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) and Microsoft's Distributed Component Object Model (DCOM), a distributed extension to Component Object Model (COM). Both CORBA and DCOM create an illusion of a single machine over a network of (heterogeneous) computers and allow objects to invoke remote objects as if they are on the same machine, thereby vastly simplifying object sharing among applications. They do so by building their abstractions on an OS-independent, PL-independent middleware. In these software architectures, objects define a number of interfaces and advertise their services by registering the interfaces. Discovery of interfaces and objects implementing them is done by using unique identifiers assigned to them when they are created. In addition, CORBA supports discovery of objects using descriptions of the services they provide. Sun Microsystems' Java RMI provides a similar functionality, where a network of platform-neutral Java virtual machines provides the illusion of a single machine. Java RMI is a language-dependent solution though Java Native Interface (JNI) provides language independence to some extent.

The software architectures supported by CORBA and DCOM are said tightly coupled because they define their own binary message encoding and thus objects are interoperable only with objects that are defined in the same software architecture; for example, CORBA objects cannot invoke methods on DCOM objects. Also, it is worth noting that security was a secondary concern, though some form of access control are highly desirable, partly because method-level/object-level access control is too fine-grained and incurs too much overhead, and partly because these software architectures were developed for use within the boundary of a single administrative control, typically small local are network.

Loosely Coupled Distributed Software Architectures

Computing has become more pervasive, more intelligent and autonomous devices with powerful computing capabilities have sprung up in the market. The fundamental changes in computing landscape make it rather easy to reason about virtualizing device usage in terms of object invocation. For instance, viewing document printing as a printing service provided by a printer can be viewed as a method invocation on a proxy object of a printer. Thus, the notion of service-centric view of computing has emerged.

These services tend to be dispersed over wider area, often crossing administrative boundaries for better resource utilization through load balancing and exploitation of locality. Such physical distribution called for more loosely coupled software architectures where scalable advertising and discovery are a must and low-latency, high-bandwidth inter-processor communication is highly desired. Specifying and enforcing security policies as well as protecting data themselves at every corner of a system can no longer be put off.

A slew of service-centric middleware developments have come to light. We note three distinctive systems from computer industry's research laboratories, namely, HP's client utility (e-speak), Sun Microsystems' Jini, and IBM's T-spaces (here listed in the alphabetic order). These have been implemented in Java for platform independence.

Client Utility Systems

HP's client utility is a somewhat under-publicized system that became the launching pad for HP's e-Speak. Its architecture represents one of the earlier forms of the peer-to-peer system suitable for web service registration, discovery and invocation (Kim, 2002). The fundamental idea is to abstract every element in computing as a uniform representation called “service (or resource)”. Using the abstraction as a building block, it provides facilities for advertising and discovery, dynamic service composition, mediation and management, and fine-grain capability-based security. What distinguishes client utility from the other systems is the fact that it makes advertisement and discovery visible to clients. Clients can describe their services using vocabularies and can specifically state what services they want to discover.

Jini
Jini technology at Sun Microsystems is a set of protocol specifications, which allows services (in particular, devices with computing capability) to announce their presence and discover other services in their vicinity. It prevails a network-centric view of computing. However, it relies on availability of multicast capability, practically limiting its applicability to services/devices connected with a local area network (such as home network). Jini exploits Java's code mobility and allows a service to export stub code implementing a private communication protocol using Java RMI. Joining, advertisement, and discovery are done transparently from client services. It has been developed mainly for collaboration within a small, trusted workgroup and offers limited security and scalability supports.

T-Spaces
IBM's T-Spaces (T-Spaces 1999) is network middleware that aims to enable communication between applications and devices in a network of heterogeneous computers and operating systems. It is a network communication buffer with database capabilities, which extends Linda's Tuple space communication model with asynchrony. T-Spaces supports hierarchical access control on the Tuple space level. Advertisement and discovery are implicit in T-Spaces and provided indirectly through shared Tuple spaces.

Convergence of the Two Independent Trends
Web services are defined at the cross point of the evolution paths of service centric computing and World Wide Web. The idea is to provide service centric computing by using the Internet as the platform. Services are delivered over the Internet (or intranet). While World Wide Web has strived to become a distributed, decentralized all pervasive infrastructure where information is put out for other users to retrieve. It is this decentralized, distributed paradigm of information dissemination that on meeting the concept of service centric computing has led to the germination of the concept of web services.

The web services paradigm has caught the fancy of the research and developer community. A large number of efforts in industry and universities are actively pursuing to define standards, platforms and concepts that will determine how web services are created, deployed, register, discover and interact with each other.

WEB SERVICES TODAY

Web Services are appearing on the Internet in the form of e-business sites and portal sites. For example, *priceline.com* and *expedia.com* act as the broker for airlines, hotel and car booking respectively. They are statically composed web services that have pre-negotiated understanding with certain airlines and hotels and broker their services through their portal sites. These are mostly B2C kind of web services. A large number of technologies and platforms are appearing and are being standardized upon so as to enable the paradigm of web services, for satisfying B2B and B2C scenarios alike in a uniform manner. These standards and platforms enable creation and deployment, description, discovery and communication amongst them. Web Services Description Language (WSDL) is used to publish services' access points (i.e., bindings) and supported interfaces, both of which are described in an XML-based description language. UDDI is used for registration and description of web services. After having discovered its partners, web-services use the document model to asynchronously exchange documents, and Simple Object Access Protocol (SOAP) for messaging (which is an incarnation of remote procedure call (RPC) in eXtensible markup language (XML)) over hypertext transfer protocol (HTTP). Most services are implemented using platform independent languages such as Java and C# on platforms like J2EE and .Net. The primary means for enforcing security are digital signature and strong encryption with public/private key pairs. Standards like SAML and XKMS are appearing in this area. A large number of payment mechanisms are being defined, too.

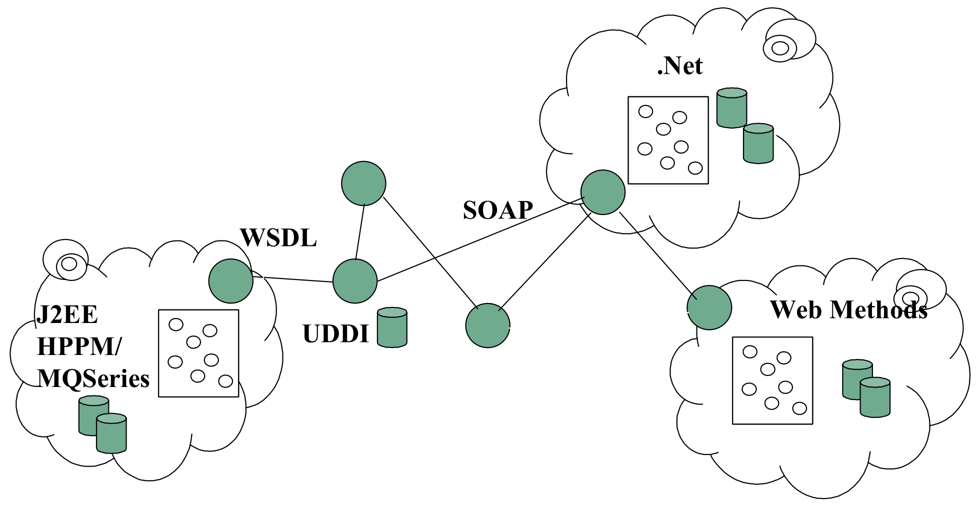


Figure 1: Web Services.

Web Services Description

In traditional software development environment, software component interfaces are defined through interface definition languages (IDL). The interfaces describe the operations the software component supports, their inputs and the expected outputs. This enables the interfaces to be decoupled from the actual implementation. As web services are envisaged as software available on the web that other web services or users will use, they need to be described so that other components can easily use them without coupling the interfaces with the implementations. Web Services Description Language (WSDL) is an attempt to describe the web service interfaces.

Web Services Definition Language (WSDL) enables creation of flexible and generic interaction models for web services. WSDL enables description of web services irrespective of the message formats and network protocols used. For example, in WSDL a service is described through a set of endpoints. An endpoint is in turn a set of operations. An operation is defined in terms of messages received or sent out by the web service:

- Message – an abstract definition of data being communicated consisting of message parts,
- Operation – an abstract definition of an action supported by the service. Operations are of the following type namely, one-way, request-response, solicit-response, notification,
- Port type – an abstract set of operations supported by one or more end points,
- Binding – a concrete protocol and data format specification for a particular port type,
- Port – a single end point defined as a combination of a binding and a network address,

- Service – a collection of related end-points.

As the implementation of the service changes or evolves in time, the WSDL definitions have to be continuously updated and versioning of the descriptions have to be tracked.

Web Services Discovery

While web browsing for information, search-engines are used to find relevant web sites based on keywords. However, this leads to lot of unnecessary links that need to be sifted through, before arriving at the relevant sites. Similarly, before web services can interact with each other, they need to discover other compatible web services with which they can undertake business. The registration and discovery of web services necessitate other entities that act as intermediaries. Universal Description Discovery Integration (UDDI) is such an initiative supported by IBM, Microsoft, and HP. It is a group of web-based registries (operator sites) maintained by these organizations that expose information about a business and its technical interfaces and APIs. The core component of the UDDI is the registration, an XML file used to define business and the web services they provide. There are three parts to the registration, namely *white pages* for name, address, contact and other identifiers, *yellow pages* for classification of business under standardized taxonomies and *green pages* that contain technical information about the web services that are exposed. It also exposes a set of APIs for inquiry and publication of information related to web services. The inquiry APIs enable browsing of the information in the repository site (e.g. find_business) and also to drill down (e.g. get_businessDetail). The publication APIs are for publishers to put their information on these repositories.

Marketplaces have been proposed as virtual meeting place managed by an intermediary supplying a series of benefits to participating web services in addition to the basic registration and discovery functionality, namely:

- Enabling inter web service interaction with or without the direct participation of the e-marketplace in the actual interaction after the discovery.
- Enabling supply and demand mechanisms like traditional catalogue purchasing, RFP or through more dynamic auctions, exchanges.
- Providing value added services, such as rating, secured payment, financial handling, certification services, notification services etc; and,
- Supply-chain management through collaborative planning and inventory handling.

Marketplaces could thus be developed as entities that use UDDI operator sites and provide value-added services on top of the basic functionality of registration and discovery. Vertical and horizontal marketplaces have been steadily coming up. VerticalNet, GlobalNetXchange, and Retailers Market Exchange are examples targeting a specific section of the industry with key players performing B2B transactions. Other examples like Chemdex, E-Steel and DirectAg.com have been successful in their respective industries. Horizontal exchanges in turn are directed at a broad range of players such as e-Bay targeting a range of clients and businesses.

Web Services Orchestration

Web Services usually expose a sub-set of enterprise business processes and activities to the external world through a set of operations defined in their WSDL. The enterprise business processes have to be defined and some of their activities have to be linked to the WSDL operations. This requires modeling of web service's back-end business processes. This necessitates intra-web service modeling and interaction. In addition, web services need to interact with other web services. This interaction happens through a sequence of message exchanges. A sequence of message exchange is termed a *conversation*. The conversations can be described in independently of the internal flows of the web services and could be simply described by sequencing the exposed operations and messages of the web services (as mentioned in their WSDLs). However, such an inter-web service interaction automatically leads to coupling of the internal processes of the web services to form what is called a *global process*. The participating web services may or may not be aware of the whole global process depending on their understanding with each other and the internal information they are willing to expose to each other.

Intra-Web Service Modeling and Interaction

Modeling of processes is usually done through languages like XLANG/WSFL. The Web Services Flow Language (WSFL) introduces the notion of activities and flows – which are useful for describing both local business process flows and global flow of messages between multiple web services. XLANG is another technology from Microsoft that provides a mechanism for process definition and global flow coordination.

WSFL (WSFL, 2001) models business processes as set of activities and links. An activity is a unit of useful work. The links could be control links where decisions are made to follow one activity or another, data links where data is fed into an activity from another. These activities could be exposed through one or more operations that are grouped through end-points (as defined in WSDL). A service is comprised of a set of end-points. A service provider can provide multiple services. Just like internal flows, global flows can be defined. Global flow consists of plug links that link up operations of two service providers. This helps in creation of complex services that can be recursively defined.

XLANG defines services by extending WSDL. The extension elements describe the behavioral aspects. A behavior spans multiple operations. A behavior has a header and a body. An Action is an atomic component of a behavior. The action elements could be an *operation*, a *delay* element or a *raise* element. A delay element could be of types delayFor and delayUntil. The delayFor and delayUntil introduce delays in the execution of the process to either wait for something to happen (for example a timeout) or to wait till an absolute date-time has been reached respectively. Exceptions are flagged through raise constructs. Raise handle the exceptions by calling the handlers registered with the raise definition. Processes combine actions together in useful ways. A process form could be a sequence, switch, while, All, Pick, Context, Compensate, Empty.

Inter-Web Service Modeling and Interaction

Web Service to Web Service interactions need to follow certain business protocols to actually undertake business on the web. X-EDI, ebXML, BTP, TPA-ML, cXML, CBL are some of the B2B technologies that have been proposed to enable this paradigm with web services.

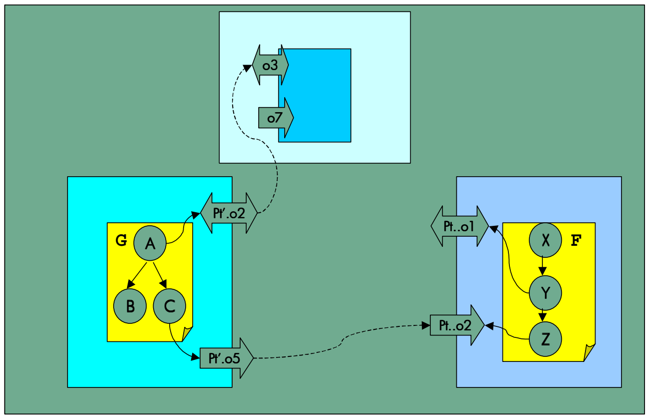


Figure 2: Intra and Inter-web Service Modeling and Interaction.

In ebXML (ebXML 2001) parties that engage in business have Collaboration Protocol Profiles (CPP) that they register at ebXML registries. Each CPP is assigned a GUID by the ebXML registry. Once a party discovers another party's CPP they form a Collaboration Protocol Agreement (CPA). CPAs are formed after negotiation between the parties. The intent of the CPA is not to expose business process internals of parties but to expose the visible process that involves interactions between parties. The messages exchanged between the involved parties or business partners may utilize ebXML Messaging Service (ebMS). The CPA and the business process specification document it references define a *conversation* between parties. This conversation involves multiple *Business Transactions*. A Business Transaction may involve exchange of messages as request and replies. The CPA may refer to multiple business process specification documents. Any one conversation will involve only a single process specification document however. Conceptually, the B2B server at the parties is responsible for managing the CPAs and for keeping track of the conversations. It also interfaces the functions defined in the CPA with the internal business processes. The CPP contains the following:

The CPP contains the following:

- Process specification Layer: This details the business transactions that form the collaboration. It also specifies the order of business transactions.
- Delivery Channels: describes party's message receiving and sending characteristics. A specification can contain more than one delivery channels.
- Document Exchange Layer: deals with processing of the business documents like digital signatures, encryption, reliable delivery.

- Transport layer: The transport layer identifies the transport protocols to be used the end point addresses, along with other properties of the transport layer. The transport protocols could be SMTP, HTTP, and FTP.

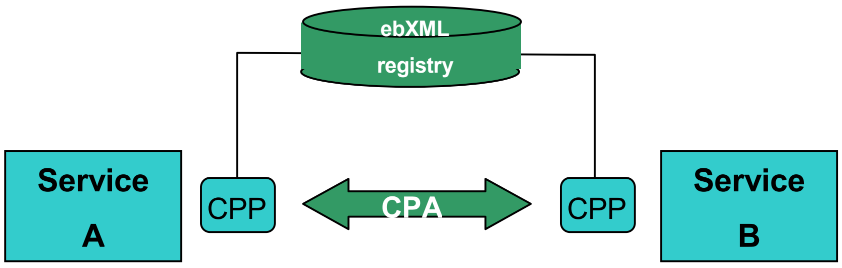


Figure 3: ebXML Service-to-Service Interaction.

WEB SERVICES PLATFORMS

Web services platforms are the technologies, means and methods that are available to build and operate web services. Platforms have changed and developed over the course of time. A classification into four generations of platform technology should help to structure the space:

- First Generation – HTML/CGI: characterized by web servers, static HTML pages, HTML FORMS for simple dialogs, and the CGI (Common Gateway Interface) to connect web servers to application programs, mostly Perl or Shell scripts.
- Second Generation – Java: server-side dynamic generation of HTML pages, user session support, the Java servlet interface became popular to connect to application programs,
- Third Generation – Richer development and run-time environments: J2EE as foundation for application servers.
- Fourth Generation – XML web services platforms: characterized by the introduction of XML and WSDL interfaces for web services with SOAP-based messaging, and a global service infrastructures for service registration and discovery emerged – UDDI.
- Fifth Generation – dynamic web services aggregation, characterized by flow systems, business negotiations, agent technology, etc.

Technically, web services have been built according to a pattern of an n-tier architecture that consists of: a front-end tier: firewall (FW), load balancer (LB), a web-server tier (WS), an application(server) (AS) tier, and a backend tier for persistent data, or the database tier (DB).

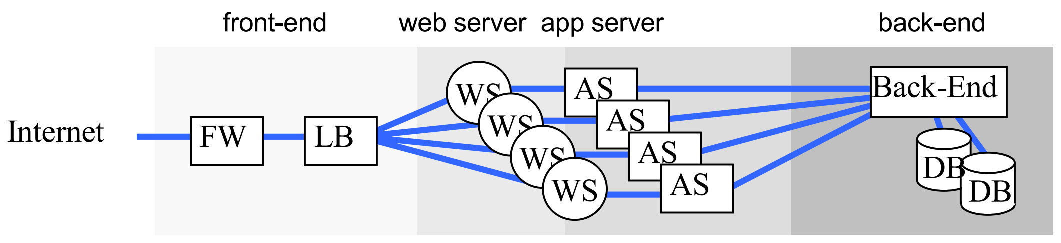


Figure 4: Basic 4-tier Architecture for Web Services.

First Generation: CGI and Perl

The emergence of the World Wide Web facilitated the easy access and decent appearance of linked HTML mark-up pages in user's browsers. In the early days, it was mostly static HTML content. Passive information services could be built that provided users with the only capability of navigating though static pages. However, HTML supported from the very beginning FORMS that allowed users to enter text or select from multiple choices menus. FORMS were treated specially from the web server. They were passed on to an interface, CGI – the Common Gateway Interface, behind which small applications, mostly Perl or Shell scripts, could read the user's input, perform respective actions, and return a HTML page according to the user's input that could then be displayed in the browser. This primitive mechanism enabled a first generation of services in the web beyond pure navigation through static content.

Second Generation: Java

With the growth of the web and the desire for richer services such as online shopping or booking, the initial means to build web services quickly became too primitive. Java applets also brought graphical interactiveness to the browser side. Java appeared as language of choice for web services. Servlets provided a better interface between the web server and the application. Technology was introduced to support the dynamic generation of HTML pages at the server side: JSP (Java Server Pages) by Sun Microsystems, ASP (Active Server Pages) by Microsoft or PHP pages in the Linux world enabled the separation of presentation, the appearance of pages in browsers, from content data. Templates and content was then merged on the fly at the server in order to generate the final page returned to the browser. Since user identification was critical for business services, user login and user sessions were introduced. Applications were becoming more complex, and it turned out that there was a significant overlap in common functions needed for many services such as session support, connectivity to persistent data bases, security functions etc.

Third Generation: Richer Development and Run-time Environments

The observation that many functions were shared and common between web services drove the development towards richer development environments based on the Java language and Java libraries. A cornerstone of these environments became J2EE. J2EE (Java 2 Platform, Enterprise Edition) is a Java platform designed for enterprise-scale computing. Sun Microsystems (together with industry partners such as IBM) designed J2EE to simplify application development for web services by decreasing the need for programming through reusable modular components and by providing standard functions such as session support and database connectivity.

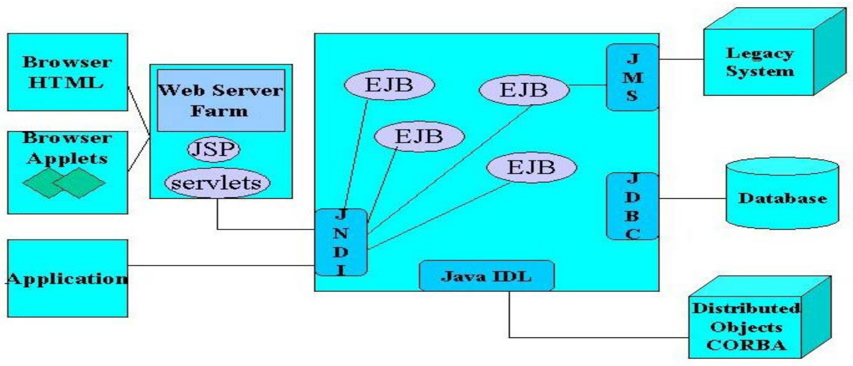


Figure 5: The J2EE Platform.

Application Server: J2EE primarily manifests in a set of libraries used by application programs performing the various functions. Web service developers still had to assemble all the pieces, link them together, connect them to the web server, and manage the various configurations etc. This lead to the emergence of pre-assembled packages that could easier be deployed on a variety of machines. These packages became known as application servers. They significantly reduced the amount of configuration work during service deployment such that service developers could spend more time on business logic and the actual function of the service. Most application server are based on J2EE technology. Examples are HP Application Server, IBM's WebSphere suite, BEA's WebLogic environment, Sun's iPlanet Application Framework or Oracle's 9*i* application server.

Fourth Generation: Web Services Platforms

Prior generations of web services were mostly focused on end-users, people accessing services from web browsers. However, it turned out being extremely difficult to access services from other services, a desirable function when services should be built that make use of other services, also known as service aggregation. The primary reason was that the external "interface language" for web services was HTML, a language that is hard to parse and overloaded with information how information is presented in user's browsers. XML emerged providing a clean syntax and the capability to describe the syntax in its own language as XML Schema. XML thus became the language of choice for web services to provide programmatic interfaces that can not only be accessed by users but also by other services. XML is being pervasively used for messaging (SOAP) and for web service interface descriptions (WSDL). In regard to platforms, XML enhancements were added to J2EE and application servers. The introduction of XML is the major differentiator between web services platforms of the 3rd and the 4th generation in this classification.

A major step towards the service-to-service integration was the introduction of an Universal Description, Discovery, and Integration service (UDDI), a global, XML-based registry infrastructure for businesses offering web services. Its goal is to enable companies and individuals to find one another on the web in a much more structured and classified manner than it is possible through search engines. Microsoft, IBM, and Ariba spearheaded UDDI.

Two major platforms are currently offered that explicitly aim for further web services interaction and integration: Sun Microsystems' SUN ONE (Open Net Environment) and Microsoft's .NET.

SUN ONE: is Sun's standards-based software architecture and platform for building and deploying services on demand. SUN ONE's architecture is build around business needs: Data, Applications, Reports, and Transactions referred to as the DART model. Major standards are supported: XML, SOAP, J2EE, UDDI, LDAP, and ebXML. The architecture is comprised of three product lines: the iPlanet Application Framework (JATO), Sun's J2EE application framework for enterprise web services development, the Solaris Operating Environment, and the Forte Development tools.

Microsoft .NET: Microsoft's .NET platform aims for providing lead technology for future distributed applications that are inherently seen as web services. With Microsoft .NET, web services' application code is built in discrete units, XML web services, that handle a specified set of tasks. Because standard interfaces based on XML simplify communication among software, XML web services can be linked together into highly specific applications and experiences. The vision is that the best XML web services from any provider from around the globe can be used to quickly and easily create a needed solution. Microsoft will provide a core set of XML web services, called Microsoft .NET My Services, to provide functions such as user identification and calendar access.

SECURITY AND WEB SERVICES

Due to their public nature, security is vital for web services. Security attacks can be classified as threats of information disclosure, unauthorized alteration of data, denial of use, misuse or abuse of services and, more rarely considered, repudiation of access. Since web services link networks together with businesses, further attacks need to be considered such as masquerading, steeling or duplicating identity and conducting business under false identity, or accessing or transferring funds from or to unauthorized accounts.

Security is vital for establishing the legal basis for businesses done over networks. Identification and authentication of business partners is the basic requirement. Integrity and authenticity of electronic documents is another. Electronic contracts must have the same binding legal status as conventional contracts. Refuse and repudiation of electronic contracts must provable in order to be legally valid. Finally, payment and transferring funds between accounts must be safe and secure.

Security architectures in networks are typically comprised of several layers:

- secure data communication: IPsec (Internet Protocol Security), SSL (Secure Socket Layer), TLS (Transport Layer Security),
- secured networks: VPN (Virtual Private Networks),
- authenticity of electronic documents and issuing individuals: digital signatures,
- secure and authenticated access: digital certificates,
- secure authentication and certification: PKI (Public Key Infrastructure),
- single sign-on and digital passports.

Single Sign-On and Digital Passports

Digital passport emerged from the desire to provide individual's identity information from a trusted and secure centralized place rather then repeatedly establishing this information with each collaborating partner and maintaining separate access credentials for each pair of collaboration. Individuals only need one such credential, the passport, in order to provide collaborating partners with certain parts of an individual's identity information. It can be avoided that individual have to maintain separate identity relationships with all other collaborating individuals such as other people or businesses. Digital passports provide an authenticated access to a centralized place where individuals have registered their identity information such as phone numbers, social security numbers, addresses, credit records, payment information. Participating individuals, both people and businesses, will access the same authenticated information assuming trust to the authority providing the passport service. Two initiatives have emerged: Microsoft's .NET Passport and The Liberty Alliance Project, initiated by Sun Microsystems.

Microsoft .NET Passport (Microsoft .NET, 2002) is a single sign-on mechanism for users in the Internet. Instead of creating separate accounts and passwords with every e-commerce site, users only need to authenticate with a single Passport server. Then, through a series of authentications and encrypted cookie certificates, the user is able to purchase items at any participating e-commerce site without re-verifying the user's identity. .NET Passport is an online service that enables to use an e-mail address and a single (passport server) password to securely sign in to any .NET Passport participating Web site or service. It allows users to easily move among participating sites without the need to re-verify their identity. The Microsoft .NET Passport has been seen critical in the public since Microsoft initially planned to be the only provider of the passport service. This concern raised another initiative that is widely support in the industry and public.

The Liberty Alliance Project (The Liberty Alliance Project, 2002) is an organization being formed to create an open, federated, single sign-on identity solution for the digital economy via any device connected to the Internet. Membership is open to all commercial and non-commercial organizations. The Alliance has three main objectives. 1) To enable consumers and businesses to maintain personal information securely. 2) To provide a universal, open standard for single sign-on with decentralized authentication and open authorization from multiple providers. 3) To provide an open standard for network identity spanning all network-connected devices.

With the emergence of web services, specific security technology is emerging. Two major classes of technologies are emerging:

- Java-Based Security Technology, and
- XML-Based Security Technology.

Both classes basically provide mappings of security technologies as introduced in the first section such as authentication and authorization, encryption, signatures, etc. into respective environments.

Java-based Security Technology for Web Services Java-based Security Technology is primarily available through the Java 2 SDK and J2EE environments in form of sets of libraries:

- Encryption: JSSE (Java Secure Socket Extension), the JCE (Java Cryptography Extension) provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers. The software also supports secure streams and sealed objects.
- Secure messaging: Java GSS-API is used for securely exchanging messages between communicating applications. The Java GSS-API contains the Java bindings for the Generic Security Services Application Program Interface (GSS-API) defined in RFC 2853. GSS-API offers application programmers uniform access to security services atop a variety of underlying security mechanisms, including Kerberos.
- Single sign on using Kerberos.
- Authentication and Authorization: JAAS (Java Authentication and Authorization Service) for authentication of users, to reliably and securely determine who is currently executing Java code, and for authorization of users to ensure they have the access control rights (permissions) required to do security-sensitive operations.
- Certification: Java Certification Path API.
- X.509 Certificates and Certificate Revocation Lists (CRLs) and Security Managers.

These libraries are available for use when web services are built using Java. They are usually used when building individual web services with application servers.

For web services interaction, XML technology eliminates the tied binding to Java. Consequently, a similar set of XML-based security technologies is emerging enabling cross-services interactions.

SAML-Based Security Technology for Web Services (SAML, 2002)The Organization for the Advancement of Structured Information Standards (OASIS) drives merging security into web services at a higher level than the common Internet security mechanisms and practices described above. Proposals are primarily directed towards providing XML specifications for documents and protocols suitable for cross-organizational web services interactions. XML-based security technology can be classified into:

- XML Document-Level Security: encryption and digitally signing XML documents,
- Protocol-Level Security for XML Document Exchanges: exchanging XML documents for authentication and authorization of peers,
- XML-Based Security Frameworks: infrastructures for establishing secure relationships among parties.

XML Document-Level Security: Encryption and Signature The (preliminary) XML Encryption Specification (XML Encryption, 2000) specifies requirements how to a digitally encrypt a Web resource in general, and an XML document in particular. XML Encryption can be applied to a part or complete XML document. The granularity of encryption can be reduced to an element, attributes or text content. Encryption can be recursive. The specification does not address confidence or trust relationships and key establishment. The specification addresses both key-encrypting-keys and data keys. The specification will not address the expression of access control policies associated with portions of the XML document.

XML Signature defines the XML Schema and processing rules for creating and representing digital signatures in any digital content (data object), including XML. An XML Signature may be applied to the content of one or more documents. Enveloped or enveloping signatures are over data within the same XML document as the signature; detached signatures are over data external to the signature element. More specifically, this specification defines an XML signature element type and an XML signature application; conformance requirements for each are specified by way of schema definitions and prose respectively. This specification also includes other useful types that identify methods for referencing collections of resources, algorithms, and keying and management information.

The XML Signature (XML Signature, 2002) is a method of associating a key with referenced data (octets); it does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed. Consequently, while this specification is an important component of secure XML applications, it itself is not sufficient to address all application security/trust concerns, particularly with respect to using signed XML (or other data formats) as a basis of human-to-human communication and agreement. Such an application must specify additional key, algorithm, processing and rendering requirements.

The SOAP Digital Signature Extensions defines how specifically SOAP messages can be digitally signed. The following fragment shows how message digest information, obtained from algorithms such as MD5 applied on a certain region in a document, can be included in the document according to the specification.

Protocol-Level Security for XML Document Exchanges Protocol-level security defines document exchanges with the purpose of establishing secure relationships among parties, typically providing well-defined interfaces and XML bindings to an existing Public Key Infrastructure. Protocol-level security can be built upon document-level security.

The XML Key Management Specification (XKMS, 2001) defines protocols for distributing and registering public keys, suitable for use in conjunction with the proposed standard for XML Signature developed by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) and an anticipated companion standard for XML encryption. The XML Key Management Specification (XKMS) comprises two parts: the XML Key Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS).

The X-KISS specification defines a protocol for a trust service that resolves public key information contained in XML-SIG document elements. The X-KISS protocol allows a client of such a service to delegate part or all of the tasks required to process <ds:KeyInfo> elements embedded in a document. A key objective of the protocol design is to minimize the complexity of application implementations by allowing them to become clients and thereby shielded from the complexity and syntax of an underlying Public Key Infrastructure (PKI Forum, 2002) used to establish trust relationships based specifications such as X.509/PKIX, or SPKI (Simple Public Key Infrastructure, 1999).

The X-KRSS specification defines a protocol for a web service that accepts registration of public key information. Once registered, the public key may be used in conjunction with other web services including X-KISS.

XML-Based Security Frameworks XML-based security frameworks go one step further than the above. They do not only built upon secure documents and document exchanges, they also provide native security services or XML bindings in existing security services.

The Security Assertion Markup Language (SAML), developed under the guidance of OASIS (OASIS, 2002), is an XML-based framework for exchanging security information with established, SAML-compliant security services. This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain. A typical example of a subject is a person, identified by his or her email address in a particular Internet DNS domain.

Assertions can convey information about authentication acts performed by subjects, attributes of subjects, and authorization decisions about whether subjects are allowed to access certain resources. Assertions are represented as XML constructs and have a nested structure, whereby a single assertion might contain several different internal statements about authentication, authorization, and attributes. Assertions containing authentication statements merely describe acts of authentication that happened previously.

Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities, and policy decision points. SAML defines a protocol by which clients can request assertions from SAML authorities and get a response from them. This protocol, consisting of XML-based request and response message formats, can be bound to many different underlying communications and transport protocols, currently it defines one binding: SOAP over HTTP.

SAML authorities can use various sources of information, such as external policy stores and assertions that were received as input in requests, in creating their responses. Thus, while clients always consume assertions, SAML authorities can be both producers and consumers of assertions.

PAYMENT SYSTEMS FOR WEB SERVICES

Effective payment systems are a precondition for business with web services. This section introduces and classifies different approaches for payment systems that have been developed over the passed years. However, payments in the Internet are mostly conducted through the existing payment infrastructure that has developed before the Internet became pervasive. End-consumer retail business in the Internet primarily relies on credit card transactions. Other traditional payment methods are offered as well: personal checks, money orders or invoice billing. In the business-to-business segment, traditional invoice billing is still the major payment method. An overview is given in (Weber, 1998). W3C has adopted payment standards (W3C e-commerce micro payment, 2002).

Payments by Credit Cards

The reason why credit card payments are well accepted is that credit card providers act as intermediators between payers and recipients of payments (payees). They do also guarantee payments up to a limit (important to the payee), and they carry the risk of misuse. All parties have to register accounts before transfers can be conducted. Another important service is the verification of creditability of a person or a business before opening and account.

SET – The Secure Electronic Transaction Standard

SET (Secure Electronic Transaction, 2002) is an open technical standard for the commerce industry initially developed by two major credit card providers, Visa and MasterCard, as a way to facilitate secure payment card transactions over the Internet. Digital certificates (Digital Certificates, 1988) create a trust chain throughout the transaction, verifying cardholders' and merchants' identity. SET is a system for ensuring the security of financial transactions over credit card providers or bank accounts. Its main objective is to provide a higher security standard for credit card payments in the Internet. A major enhancement compared to traditional credit card payments is that neither credit card credentials nor payers' identity are revealed to merchants. With SET, a user is given an electronic wallet (digital certificate). A transaction is conducted and verified using a combination of digital certificates and digital signatures among the purchaser, a merchant, and the purchaser's bank in a way that ensures privacy and confidentiality.

Not all payments required by web services can be conducted through credit card transactions. First, credit card transactions are typically directed from an end customer, a person, to a business that can receive such payments. Second, the amounts transferred through a credit card transaction are limited to a range between currency equivalents of >$0.10 up to several thousand dollars depending on an individuals' credit limits. Micro payments, <$0.10, as well as macro payments, >$10,000, are typically not provided. The lower payment bound is also caused by the cost per transaction model credit card providers use. Third, payments among persons, as for instance required for auctions among people or for buying and selling used goods, cannot be conducted through credit card accounts. Traditional payment methods are used here: personal checks, money orders, or cash settlement. Fourth, only individuals with registered accounts can participate in credit card payments. Individuals that do not qualify are excluded. This restriction is also a major barrier for web service business in developing countries.

Micro-Payments

The purpose of micro-payments is primarily for "pay-per-use" models where the usage is measured and immediately charged to customers in very small amounts. Transaction costs for micro payment systems need to be significantly lower, and the number of transactions may be significantly higher compared with credit card payments. Accurate, fine-grained charging is enabled. These are the two major differentiators of micro payment systems. W3C proposes a Common Markup for micro payment "per-fee-links".

Micro-payments involve a buyer or customer, a vendor or merchant, and potentially one or more additional parties that keep accounts in order to aggregate micro payments for final charge. These mediators are called brokers (in Millicent), billing servers (in IBM MicroPayments), or intermediaries (in France Telecom Micropayments), to name a few.

Millicent: One micro-payment system is Millicent (Millicent, 2002). The MilliCent Microcommerce Network provides new pay-per-click / earn-per-click functionality for Internet users. It allows buying and selling digital products costing from 1/10th of a cent to up to $10.00 or more. MilliCent can be used by web services to build any number of parallel revenue streams through the simultaneous use of pay-per-click purchases, subscriptions, and advertising. It can also be used to make direct monetary payments to users. MilliCent is optimized for buying and selling digital products or services over the Internet such as articles, newsletter, real-time data, streaming audio, electronic postage, video streams, maps, financial data, multimedia objects, interactive games, software, and hyperlinks to other sites.

NetBill: NetBill is a Carnegie-Mellon University Internet billing server project, which is used as payment method for buying information goods and services via the Internet. It aims at secure payment for and delivery of information goods, e.g. library services, journal articles, CPU cycles etc. The NetBill system charges for transactions and requires customers to have a prepaid NetBill account from which all payments are deducted. The NetBill payment system uses both symmetric key and public key cryptography. It relies on Kerberos for authentication. An account server, called NetBill server, maintains accounts for both customers and merchants. NetBill acts as an aggregator to combine many small transactions into larger conventional transactions, thus amortizing conventional overhead fees. customers and merchants have to trust the NetBill server.

Digital Money and Digital Coins

In contrast to account-based payment systems, such as credit card-based systems, where amounts are transferred between accounts inside or between credit card or bank providers, digital money represents a value amount flowing from a payer to a payee across the network. Establishing accounts with providers before services can actually be used is unnecessary. Advantages are the same as for cash money: no mutual accounts need to be established before a payment can be conducted. No mutual authentication is needed improving convenience for both parties. And, like for cash money, the payer does not need to reveal any identity credentials to the payee or someone else. Payments are anonymous and non-traceable. Major hurdles for this approach is the prevention of duplication and forging of digital money since no physical security marks such as watermarks can be applied to digitized bit strings.

The basic idea behind digital money is that a consumer purchases "digital coins" from an issuer using a regular payment method such as a credit card. The issuer generates an account for that customer and deposits the amount to it. It then hands out a set of digital coins to the customer that he or she can use for payments. For a payment, the customer transfers coins to the merchant or service provider. The provider then transfers coins to the issuer and deposits them into his account. The merchant, however, may also use these coins to pay its suppliers. Digital coins will thus flow among participants similarly like cash money flows among people.

Following requirements need to be met by digital money systems:

- digital money must be protected from duplication or forging, and
- digital money should neither contain nor reveal identity credentials of any involved party in order to be anonymous.

The first requirement is achieved by not actually representing an amount by a digital coin, but rather a reference to an allocated amount in the possessor's account with the issuer. When digital coins are copied, the reference is copied, not the amount itself. However, the first individual redeeming a coin with the issuer will receive the amount. Identity at redemption cannot be verified since digital coins do not carry identifying credentials of the possessor. The only term the issuer can verify is whether or not a coin has already been redeemed. By thus, theft of digital money is possible, and parties have an interest in keeping their coins protected.

Achieving complete anonymity between an issuer and subsequent receivers of digital money is a key characteristics of digital money. It is basically achieved by blinded signatures (Chaum85, 1985) that guarantee to uniquely assign coins with allocated amounts within the issuer's account system and without revealing any identification information of the holder of that account.

eCash: eCash (eCash, 2002) stands for "electronic cash", a system developed by DigiCash that went under field tests in the late 1990's. eCash is a legal form of computer-generated currency. This currency can be securely purchased with conventional means: credit cards, checks, money orders or wire transfers. MicroMint: MicroMint is a proposal by Rivest and Shamir about coins that can only efficiently be produced in very large quantities and are hard to produce in small quantities. The validity of a coin is easily checked. MicroMint is optimized for unrelated low-value payments. It uses no public key operations. However the scheme is very complex and would require a lot of initial and operational effort. Therefore it is unlikely that it ever will gain any practical importance.

A broker will issue new coins at the beginning of a period and will revoke those of the prior period. Coins consist of multiple hash collisions, i.e. different values that all hash to the same value. The broker mints coins by computing such hash collisions. For that process many computations are required, but more and more hash collisions are detected with continued computation. The broker sells these MicroMint coins in batches to customers. Unused coins can be returned to the broker at the end of a period, e.g. a month. Customers render MicroMint coins as payment to merchants.

THE FUTURE OF WEB SERVICES

In future we will see the unleashing of web services phenomenon. This would involve the fulfillment of dynamic web service composition and orchestration vision, appearance of personalized web services, concepts of end-to-end guarantees on web services and the development of web service infrastructure as a reusable, re-configurable, self-healing, self-managing, large-scale system.

Dynamic Web Services Composition and Orchestration

The vision of web services intelligently interacting one with another and performing useful tasks automatically and seamlessly remains to become reality. Major milestones have been achieved: XML as syntactic framework and data representation language for interacting web services, the web infrastructure itself providing ubiquitous access to web services, the emergence of global registration and discovery services, the technology to support the creation and maintenance of web services, just to name a few. However, major pieces are still missing such as the formalization and description of service semantic. The effort of creating a semantic web (Semantic Web 2001) is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. Ontologies define the structure, relationships and meaning of terms appearing in service descriptions. The semantic web vision is that these ontologies can be registered, discovered and used for reasoning about web service selection before undertaking business.

In addition, sending a document or invoking a method and getting a reply are the basic communication primitive. However, complex interactions between web-services will involve multiple steps of communication that are related to each other. A conversation definition is a sequencing of document exchanges (method invocations in the network object model) that together accomplish some business functionality. In addition to agreeing upon vocabularies and document formats, conversational web-services also agree upon conversation definitions before communicating with each other. A conversation definition consists of descriptions of interactions and transitions. Interactions define the atomic units of information interchange between web-services. Essentially, each service describes each interaction in terms of the documents that it will accept as input, or will produce as output. The interactions are the building blocks of the conversation definition. Transitions specify the ordering amongst the interactions. Web-services need to introspect other web-services and obtain each other's descriptions before they start communicating and collaborating (WSCL, 2002).

RosettaNet (RosettaNet, 2002) is a non-profit consortium of major Information Technology, Electronic Components and Semiconductor Manufacturing companies working to create and implement industry-wide, open e-business process standards, particularly targeting business-to-business market places, workflow and supply-chain management solutions. These standards form a common e-business language, aligning processes between supply chain partners on a global basis. Several examples exist. The centerpiece of the RosettaNet model is the partner interface process (PIP). The PIP defines the activities, decisions, and interactions that each e-business trading participant is responsible for.

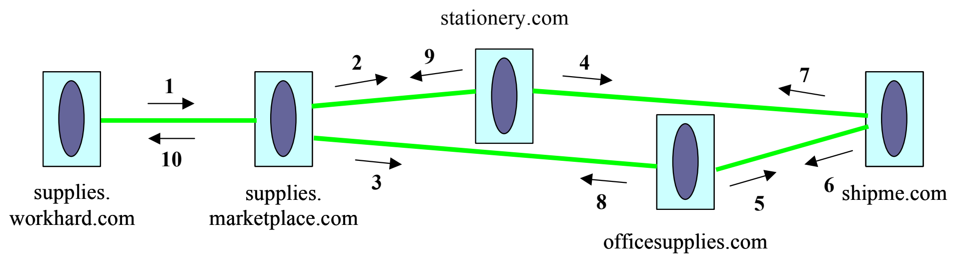
Once these hurdles are overcome, the basis and platform for true web services would exist that would enable agent technologies merging into web services providing the envisioned dynamic web service aggregation on demand according to users specifications.

Personalized Web Services

As web service technology evolves, we anticipate that they will become increasingly sophisticated, and that the challenges web service community will face will also evolve to meet their new capabilities. One of the most important of these challenges is the question of what it means to personalize web services. Personalization can be achieved by using user (other web services or humans) profiles, monitoring user behavior, devices and context to customize web services (Kuno 2002) for achieving metrics like Quality of Experience (QoE) (van Moorsel 2001). This would involve providing and meeting guarantees of service performance on user's side. Personalization could also result in creation of third-party rating agencies that will register user experiences and could be instructive for other first-time users. These rating mechanisms already exist in ad-hoc manner, e.g. e-bay, Amazon allow users to rate sellers and commodities (books) respectively. Salcentral.com and bizrate.com are third-party rating agencies that rate businesses. These services could be also developed as extended UDDI services. These mechanisms will also render web services more "customer-friendly".

End-to-End Web Service Interactions

Web services are federated in nature as they interact across management domains and enterprise networks. Their implementations can be vastly different in nature. When two web services connect to each other, they have to agree on a document exchange protocol and the appropriate document formats. From then on they can interoperate with each other exchanging documents. SOAP defines a common layer for document exchange. Services can define their own service-specific content on the top of SOAP. Often, these web service transactions will span multiple web services. A request originating at a particular web service can lead to transactions on a set of web services. For example, a purchase order transaction that begins when an employee orders supplies and ends when he or she receives a confirmation could result in ten messages being exchanged between various services as shown in Figure 6.



1. purchase order
2. part of purchase order
3. the other part of the purchase order
4. shipping request
5. shipping request
6. shipping confirmation
7. shipping confirmation
8. order confirmation
9. order confirmation
10. purchase order confirmation

Figure 6: SOAP Messages Exchanged Between Web Services.

The exchange of messages between web services could be asynchronous. Services sending a request message need not be blocked waiting for a response message. In some cases, all the participating services are like peers, in which case there is no notion of a request or a response. Some of the message flow patterns that result from this asynchrony are shown in Figure 7. The first example in Figure 7 shows a single request resulting in multiple responses. The second example shows a broker-scenario, in which a request is sent to a broker but responses are received directly from a set of suppliers.

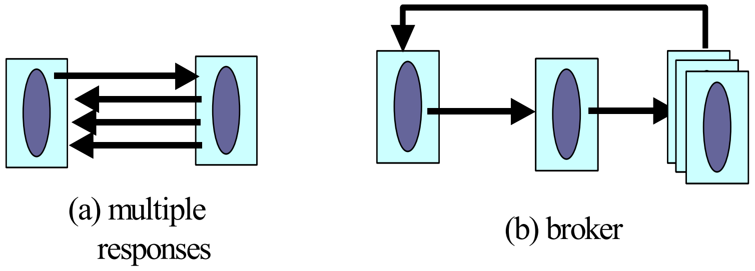


Figure 7: Asynchronous Message Patterns Between Web Services.

These web services also interact with a complex web of business processes at their back-ends. Some of these business processes are exposed as web service operations. A business processes comprises of sequence of activities and links as defined by WSFL and XLANG. These business processes have to be managed so as to manage web service interactions. Management of web services thus is a challenging task because of their heterogeneity, asynchrony, and federation. Managing web services involves managing business transactions by correlation of messages across enterprises (Sahai 2001), and managing the business processes.

Also, in order to real-business on the web, they will need to specify, agree and monitor service level agreements with each other. As web services will invariably have large number of SLAs and little human intervention is desirable it would necessitate automating the process as much as possible (Sahai 2002).

Web Service to Web Service interaction management can also be done through mediation. Web Service Networks vision is to mediate web service interactions, so as to make it secure, manageable and reliable. Such networks enable versioning management, reliable messaging, and monitoring of message flows (Flamenco Networks, GrandCentral, Transact Plus, Talking Blocks).

Future Web Services Infrastructures

Deployment and operational costs are determinants in the balance sheets for web service providers. Web service providers are optimizing their IT infrastructures to allow faster provisioning of new services and more reliable operation. Platforms and management solutions are emerging that reduce web services' deployment and operational costs. Those platforms support the deployment of web service (installation and configuration of software and content data), the virtual wiring of machines into application environments independently of the physical wiring in a data center. They allow rearrangements of web services' applications among machines, the dynamic sizing of service capacities according to fluctuations in demands, and the isolation of service environments hosted in the same data center.

HP's Utility Data Center (UDC) (HP Utility Data Center, 2001) represents such a platform. The HP Utility Data Center with its Utility Controller Software creates and runs virtual IT environments as a highly automated service optimizing asset utilization and reducing staffing loads. Resource virtualization is transparent to applications, sitting underneath the abstractions of operating systems.

Two types of resources are virtualized:

- virtualized network resources, permitting the rewiring of servers and related assets to create entire virtual IT environments,
- virtualized storage resources, for secure, effective storage partitioning, and with disk images containing persistent states of application environments such as file systems, bootable operating system images, application software, etc.

Figure 8 shows the basic building blocks of such a Utility Data Center with the two fabrics for network virtualization and storage virtualization.

The storage virtualization fabric with the Storage Area Network attaches storage elements (disks) to processing elements (machines). The network virtualization fabric then allows linking processing elements together in a virtual LAN.

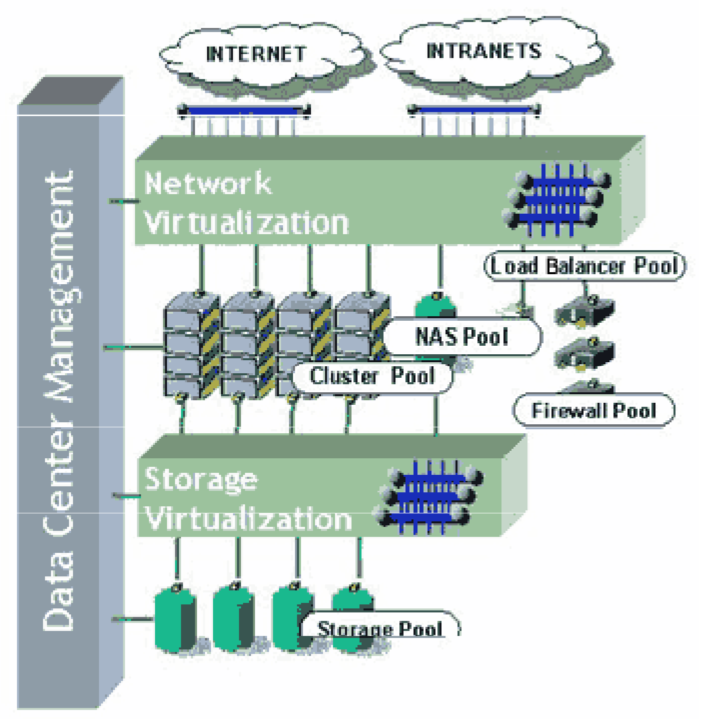


Figure: 8: Architecture of a Utility Data Center Web Services Platform Infrastructure.

The two major benefits for web services management can be achieved on top of that infrastructure:

- Automated Web Services Deployment: by entirely maintaining persistent web services' states in the storage system and conducting programmatic control over storage containing deployed service arrangements, and
- Dynamic Capacity Sizing of Web Services: by the ability to automatically launch additional service instances absorbing additional load occurring to the service. Service instances are launched by first allocating spare machines from the pool maintained in the data center, wiring them into the specific environment of the web service, attaching appropriate storage to those machines and launching the applications obtained from that storage. Web server farms are a predestine example for such a "breathing" (meaning dynamically adjustable) configuration (Andrzejak, 2002) (Graupner, 2002).

IBM's Autonomic Computing vision is to provide for self-managing systems. The intent is to create systems that respond to capacity demands and system glitches without human intervention. These systems intend to be self-configuring, self-healing, self-protecting and self-optimizing (IBM, 2002).

CONCLUSION

The web services paradigm has evolved substantially because of concerted efforts by the software community. While the genesis of web services can be traced to projects like e-speak, Jini, and T-spaces, it has received a boost recently because of standardization of related technologies for description (WSDL), and discovery (UDDI). Platforms like J2EE compatible application servers and .Net have appeared for the creation, and deployment of these web services. Standards for security and payment are being agreed upon. The full potential of web services however remains unrealized. The future will see the realization of web services as a means of doing business on the web, the vision of dynamic composition of web services, personalized web services, end-to-end management of web service interactions and of having dynamically reusable service infrastructure that will be adapt to variations in resource consumption.

BIBLIOGRAPHY

Andrzejak, A., Graupner, S., Kotov, V., Trinks, H.: Self-Organizing Control in Planetary-Scale Computing, IEEE International Symposium on Cluster Computing and the Grid (CCGrid), 2nd Workshop on Agent-based Cluster and Grid Computing (ACGC), May 21-24, 2002, Berlin.

Chaum, D.: Security Without Identification: Transaction Systems to Make Big Brother Obsolete, Communications of the ACM, Vol. 28, October 1985.

Digital Certificates, CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

eCash, http://www.cryptologic.com/faq/faq-ecash.html, 2002.

ebXML http://www.ebxml.org.

Graupner, S., Kotov, V., Trinks, H.: Resource-Sharing and Service Deployment in Virtual Data Centers, IEEE Workshop on Resource Sharing in Massively Distributed Systems (RESH'02), Vienna, Austria, July 2002.

HP Utility Data Center (UDC), http://www.hp.com/go/hpudc, November 2001.

Kim, W., Graupner, S., Sahai, A.: A Secure Platform for Peer-to-Peer Computing in the Internet, 35th Hawaii International Conference on System Science (HICSS-35), Island of Hawaii, January 7-10, 2002.

Kuno H, Sahai A. My Agent Wants to Talk to your Service: Personalizing Web Services through Agents. HPL-2002-114.

IBM Autonomic Computing: http://www.research.ibm.com/autonomic/.

Lee T. B. The World Wide Web: Past, Present and Future. http://www.w3.org/People/Berners-Lee/1996/ppf.html.

Microsoft .NET Passport, http://www.passport.com/, 2002.

Millicent, http://www.millicent.com/home.html, 2002.

Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org, 2002.

PKI Forum, http://www.pkiforum.org/, 2002.

RosettaNet, http://www.rosettanet.org, 2002.

Sahai A, et al. Automated SLA Monitoring for Web Services. HPL-2002-191.

http://www.hpl.hp.com/techreports/2002/HPL-2002-191.html

Sahai A, Machiraju V, Ouyang J, Wurster K. Message Tracking in SOAP-Based Web Services. IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), April 2002, Florence, Italy.

Sahai A, Machiraju V, Wurster K. Monitoring and Controlling Internet based Services. The Second IEEE Workshop on Internet Applications (WIAPP'01), San Jose, July 2001 (also as HP Technical Report HPL-2000-120).

Sahai A, Ouyang J, Machiraju, V. End-to-End Transaction management for Web Based Services. Third International Workshop on Advanced issues of E-Commerce and Web based Information Systems (WECWIS), June 21-22 2000, San Jose USA (also as HP Technical Report HPL-2000-168).

SAML 1.0 Specification Set, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-29.pdf, 2002.

Semantic Web, http://www.w3.org/2001/sw/

SET Secure Electronic Transactions LLC, http://www.setco.org/, 2002.

Simple Public Key Infrastructure (SPKI), SPKI Certificate Theory, RFC 2693, 1999.

T-Space at IBM. http://www.almaden.ibm.com/cs/TSpaces/. 1999.

The Liberty Alliance Project, http://www.projectliberty.org/, 2002.

The W3C Micro Payment, http://www.w3.org/ECommerce/Micropayments/, 2002.

Weber, R.: Chablis - Market Analysis of Digital Payment Systems, University of Munich, http://chablis.informatik.tu-muenchen.de/MStudy/x-a-marketpay.html, 1998.

Van Moorsel, A. Metrics for the Internet Age-Quality of Experience and Quality of Business. HPL-2001-179. http://www.hpl.hp.com/techreports/2001/HPL-2001-179.html

WSCL Web Services Conversation Language, http://www.w3.org/TR/wscl10, 2002.

WSFL Web Services Flow Language (WSFL 1.0), Edited by F. Leymann, IBM, May 2001.

XML Key Management Specification (XKMS), http://www.w3.org/TR/xkms, March 2001.

XML Encryption Requirements, http://lists.w3.org/Archives/Public/xml-encryption/2000Oct/att-0003/01-06-xml-encryption-req.html, December 2000.

XML Signature Syntax and Processing, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212, 2002.