

**Seminar Report**

**On**

**Service Oriented Architecture & Web 2.0**

**Submitted By:** Gurpreet Singh Modi (048/GTBIT/CSE/2004)



**Department of Computer Science and Engineering**  
**Guru Tegh Bahadur Institute of Technology, New Delhi**

## Table of Contents

1. Introduction	01
2. Service Oriented Architecture	02
2.1 Creation of a Service Oriented Environment ...	05
2.2 SOA Elements ...	06
2.3 Benefits ...	07
3. Web 2.0	09
3.1 Web 2.0 Buzzwords ...	10
3.2 Web 1.0 vs. Web 2.0 ...	13
4. Convergence of SOA & Web 2.0	14
5. Software as a Service	15
6. Web Technologies	16
6.1 Client Side Technologies ...	16
6.2 Server Side Technologies ...	17
6.3 Web Services ...	18
6.4 Mashups ...	19
7. Tools for SOA & Web 2.0	20
References	22

## **1. Introduction**

---

---

Service Oriented Architecture (SOA) is an architectural style that guides all aspects of creating and using business processes, packaged as services, throughout their lifecycle, as well as defining and provisioning the IT infrastructure that allows different applications to exchange data and participate in business processes regardless of the operating systems or programming languages underlying those applications. SOA represents a model in which functionality is decomposed into small, distinct units (services), which can be distributed over a network and can be combined together and reused to create business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between one or more services. It is often seen as an evolution of distributed computing and modular programming.

Web 2.0 is the network as platform, spanning all connected devices; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a form that allows remixing by others, creating network effects through an "architecture of participation," and going beyond the page metaphor of Web 1.0 to deliver rich user experiences.

Two of the industry's hottest buzzwords are combining to fuel one of the hottest emerging trends in the industry—the use of Web 2.0 technologies acting as front ends to SOA back-end environments. This trend touches on RIAs (rich Internet applications), mashups, AJAX, RSS and other Web 2.0 areas. Now being referred to as Enterprise 2.0, the Web 2.0 technologies are helping to create rich interactive front ends to SOA back-end systems. In addition, line-of-business users who typically are non developers can take services and build mashups without IT involvement.

## **2. Service Oriented Architecture**

---

---

Companies have long sought to integrate existing systems in order to implement information technology (IT) support for business processes that cover all present and prospective systems requirements needed to run the business end-to-end. A variety of designs can be used to this end, ranging from rigid point-to-point electronic data interchange (EDI) interactions to Web auctions. By updating older technologies, such as Internet-enabling EDI-based systems, companies can make their IT systems available to internal or external customers; but the resulting systems have not proven to be flexible enough to meet business demands.

A flexible, standardized architecture is required to better support the connection of various applications and the sharing of data. SOA is one such architecture. It unifies business processes by structuring large applications as an ad-hoc collection of smaller modules called services. These applications can be used by different groups of people both inside and outside the company, and new applications built from a mix of services from the global pool exhibit greater flexibility and uniformity. One should not, for example, have to provide redundantly the same personal information to open an online checking, savings or IRA account, and further, the interfaces one interacts with should have the same look and feel and use the same level and type of input data validation. Building all applications from the same pool of services makes achieving this goal much easier and more deployable to affiliate companies. An example of this might be interacting with a rental car company's reservation system even though you are doing so from an airline's reservation system.

SOAs build applications out of software services. Services are relatively large, intrinsically unassociated units of functionality, which have no calls to each other embedded in them. They typically implement functionalities most humans would recognize as a service, such as filling out an online application for an account, viewing an

online bank statement, or placing an online book or airline ticket order. Instead of services embedding calls to each other in their source code, protocols are defined which describe how one or more services can talk to each other. This architecture then relies on a business process expert to link and sequence services, in a process known as orchestration, to meet a new or existing business system requirement.

Relative to earlier attempts to promote software reuse via modularity of functions, or by use of predefined groups of functions known as classes, SOA's atomic level objects are 100 to 1,000 times larger, and are associated by an application designer or engineer using orchestration. In the process of orchestration, relatively large chunks of software functionality (services) are associated in a non-hierarchical arrangement (in contrast to a class's hierarchies) by a software engineer, or process engineer, using a special software tool which contains an exhaustive list of all of the services, their characteristics, and a means to record the designer's choices which the designer can manage and the software system can consume and use at run-time.

Underlying and enabling all of this is metadata which is sufficient to describe not only the characteristics of these services, but also the data that drives them. XML has been used extensively in SOA to create data which is wrapped in a nearly exhaustive description container. Analogously, the services themselves are typically described by WSDL, and communications protocols by SOAP. Whether these description languages are the best possible for the job, and whether they will remain the favorites going forward, is at present an open question. What is certain is that SOA is utterly dependent on data and services that are described using some implementation of metadata which meets two criteria. The metadata must be in a form which software systems can consume to dynamically configure to maintain coherence and integrity, and in a form which system designers can understand and use to manage that metadata.

The goal of SOA is to allow fairly large chunks of functionality to be strung together to form ad-hoc applications which are built almost entirely from existing software services. The larger the chunks, the fewer the interface points required to implement any given set of functionality; however, very large chunks of functionality may not be granular enough to be easily reused. Each interface brings with it some amount of processing overhead, so there is a performance consideration in choosing the granularity of services. The great promise of SOA is that the marginal cost of creating the n-th application is zero, as all of the software required already exists to satisfy the requirements of other applications. Only orchestration is required to produce a new application.

The key is that there are no interactions between the chunks specified within the chunks themselves. Instead, the interaction of services (all of whom are unassociated peers) is specified by humans in a relatively ad-hoc way with the intent driven by newly emergent business requirements. Thus the need for services to be much larger units of functionality than traditional functions or classes, lest the sheer complexity of thousands of such granular objects overwhelm the application designer. The services themselves are developed using traditional languages like Java, C#, C++, C or COBOL.

SOA services are loosely coupled, in contrast to the functions a linker binds together to form an executable, a DLL, or an assembly. SOA services also run in "safe" wrappers such as Java or .NET, which manage memory allocation and reclamation, allow ad-hoc and late binding, and provide some degree of indeterminate data typing.

Increasing numbers of third-party software companies are offering software services for a fee. In the future, SOA systems may consist of such third-party services combined with others created in-house. This has the potential to spread costs over many customers, and customer uses, and promotes standardization both in and across industries. In particular, the travel industry now has a well-defined and documented set of both services and data, sufficient to allow any reasonably competent software engineer to create travel agency

software using entirely off-the-shelf software services. Other industries, such as the finance industry, are also making significant progress in this direction.

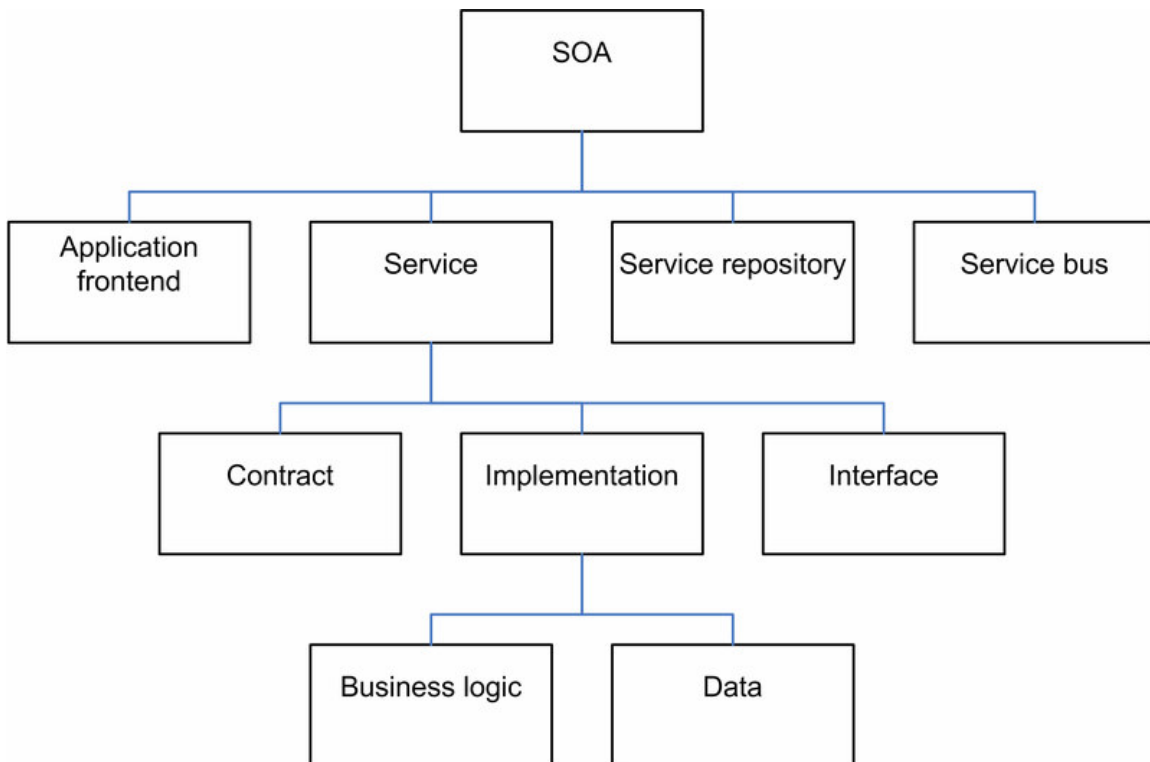
SOA is an architecture that relies on service-orientation as its fundamental design principle. In an SOA environment independent services can be accessed without knowledge of their underlying platform implementation.

## **2.1 Creation of a Service Oriented Environment**

The goal for a SOA is a world wide mesh of collaborating services, which are published and available for invocation on the Service Bus. Adopting SOA is essential to deliver the business agility and IT flexibility promised by Web Services. These benefits are delivered not by just viewing service architecture from a technology perspective and the adoption of Web Service protocols, but require the creation of a Service Oriented Environment that is based on the following key principals:

- Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form.
- SOA is not just an architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the *right* services are provided and consumed.
- With SOA it is critical to implement processes that ensure that there are at least two different and separate processes—for provider and consumer.
- Rather than leaving developers to discover individual services and put them into context, the Business Service Bus is instead their starting point that guides them to a coherent set that has been assembled for their domain.

## 2.2 SOA Elements



An SOA is based on four abstraction elements: *application frontend*, *service*, *service repository*, *service bus*. These abstractions are the key elements involved in a Service Oriented Architecture. Application frontend is the user interface through which the owner of the business process interacts with the system. Services provide business functionality that can be used by the application frontends and other services. The service contains an implementation which contains the business logic and data; a contract defining the functionality and constraints for users; and an interface that exposes the functionality. The service repositories store the service contracts of the individual services and acts as a meta base for the services. The service bus interconnects the services to the application frontends.

### **2.3 Benefits**

SOA enables development of a new generation of dynamic applications that address a number of top-level business concerns that are central to grow and competitiveness. It provides the framework through which to simplify the creation and management of integrated systems and applications, and a way to align IT assets with the business model and changing business needs. The benefits of implementing SOA include:

- **Enhanced business decision making.** By aggregating access to business services and information into a set of dynamic, composite business applications, decision makers gain more accurate and more comprehensive information.
- **Greater employee productivity.** By providing streamlined access to systems and information and enabling business process improvement, businesses can drive greater employee productivity.
- **Stronger connections with customers and suppliers.** The benefits of SOA extend beyond organizational boundaries. Mergers and acquisitions become more profitable, since it is easier to integrate disparate systems and applications. Integration with trading partners and streamlining of supply chain processes are readily attainable goals.
- **More productive, more flexible applications.** The service oriented approach enables IT to make existing IT assets—including legacy systems and applications—more productive and more profitable to the business without the need for custom-coded one-off integration solutions.

- **Faster, more cost-effective application development.** Standards-based service design enables IT to create a repository of reusable services that can be combined into higher level services and composite applications as new business needs arise.
- **More manageable and secure applications.** Service oriented solutions provide a common infrastructure (and documentation) for developing secure, monitored, and predictable services. As business needs change, SOA makes it easier to add in new services and capabilities that map onto critical business processes.

### **3. Web 2.0**

---

---

Web 2.0 refers to a perceived second generation of web-based communities and hosted services — such as social-networking sites, wikis, and folksonomies — which aim to facilitate creativity, collaboration, and sharing between users. The term gained currency following the first O'Reilly Media Web 2.0 conference in 2004. Although the term suggests a new version of the World Wide Web, it does not refer to an update to any technical specifications, but to changes in the ways software developers and end-users use webs. According to Tim O'Reilly,

"Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform."

The idea of "Web 2.0" can also relate to a transition of some websites from isolated information silos to interlinked computing platforms that function like locally-available software in the perception of the user. Web 2.0 also includes a social element where users generate and distribute content, often with freedom to share and re-use. This can allegedly result in a rise in the economic value of the web as users can do more online. Tim O'Reilly regards Web 2.0 as business embracing the web as a platform and utilizing its strengths (global audiences, for example). O'Reilly considers that Eric Schmidt's abridged slogan, don't fight the Internet, encompasses the essence of Web 2.0 — building applications and services around the unique features of the Internet, as opposed to building applications and expecting the Internet to suit as a platform (effectively "fighting the Internet").

Web 2.0 websites allow the user to do more than just retrieve information. They can provide "Network as platform" computing, allowing users to run software applications entirely through a browser. Users can own the data on a Web 2.0 site and exercising

control over that data. These sites may have an "Architecture of participation" that encourages users to add value to the application as they use it. This stands in sharp contrast to hierarchical access-control in applications, in which systems categorize users into roles with varying degrees of functionality. Web 2.0 sites often feature a rich, user-friendly interface based on Ajax, Flex or similar rich media. The sites may also have social-networking aspects.

Technologies such as weblogs (blogs), social bookmarking, wikis, podcasts, RSS feeds (and other forms of many-to-many publishing), social software, web application programming interfaces (APIs), and online web services such as eBay provide enhancements over read-only websites.

### **3.1 Web 2.0 Buzzwords**

Here are a few of the buzzwords that are driving the Web 2.0 era:

**API:** Application Programming Interface is a code interface over which programmers can built their own applications. APIs make it possible for the programmers to use some great features on the web without any complex coding.

**Atom:** Atom is a commonly used syndication format for web feeds. It is based on XML and is supported by most standard feed readers.

**Blog:** Blog or a Weblog is an online journal, which provides an easy to use interface for users to publish content. Blogs enable people to have an online presence without having any technical knowledge.

**Feeds:** Feeds allow you to get updates from the websites. It allows you to check the recent articles and content from web pages on the internet. A person may subscribe to a news feed to get the latest news right on his desktop.

**Folksonomy:** Folksonomy is the practice of collaboratively categorizing content by tagging the content under various connotations. This method is popularly applied in social bookmarking and tagging webpages and photos.

**Mashup:** Mashups are a interactive genre of web applications which accumulates data from various sources and conflates it in a single integrated application. Mashup generally collects data from Feeds and Web Services.

**Microformats:** Microformats are a set of simple open data formats, which allows the normal data in HTML and XHTML to be categorized by providing annotations in the existing markup. It aims at easier access of information such as contact addresses, locations etc making it easily placeable by the searching softwares.

**Perpetual Beta:** Perpetual Beta is the software stage where the software is launched and is always under constant updates, which could be monthly, weekly or even daily. It follows the principle of "release early and release often", thus a software is released at a premature state and new features are added frequently.

**Podcast:** Podcast is a form of a feed carrying digital media files which could be downloaded or streamed on media players. Online radio is a well comprehensible example of a podcast.

**RSS:** Really Simple Syndication is a collection of feed formats. It is used to syndicate updated content from a website.

**SEO:** Search Engine Optimization is a process of improving your rankings on the search engines by using a set of techniques, which include ameliorating the quality of content and code of the website, placing links at strategic locations on the web and adhering to some web standards.

**Social Bookmarking:** Social Bookmarking is a folksonomy practice through which users bookmark pages on the web and create custom tags to annotate these pages.

**Social Networking:** Social networking is a method to promote online collaboration of people by creating communities of (net)citizens with similar interests. It is a popular method to connect with friends online.

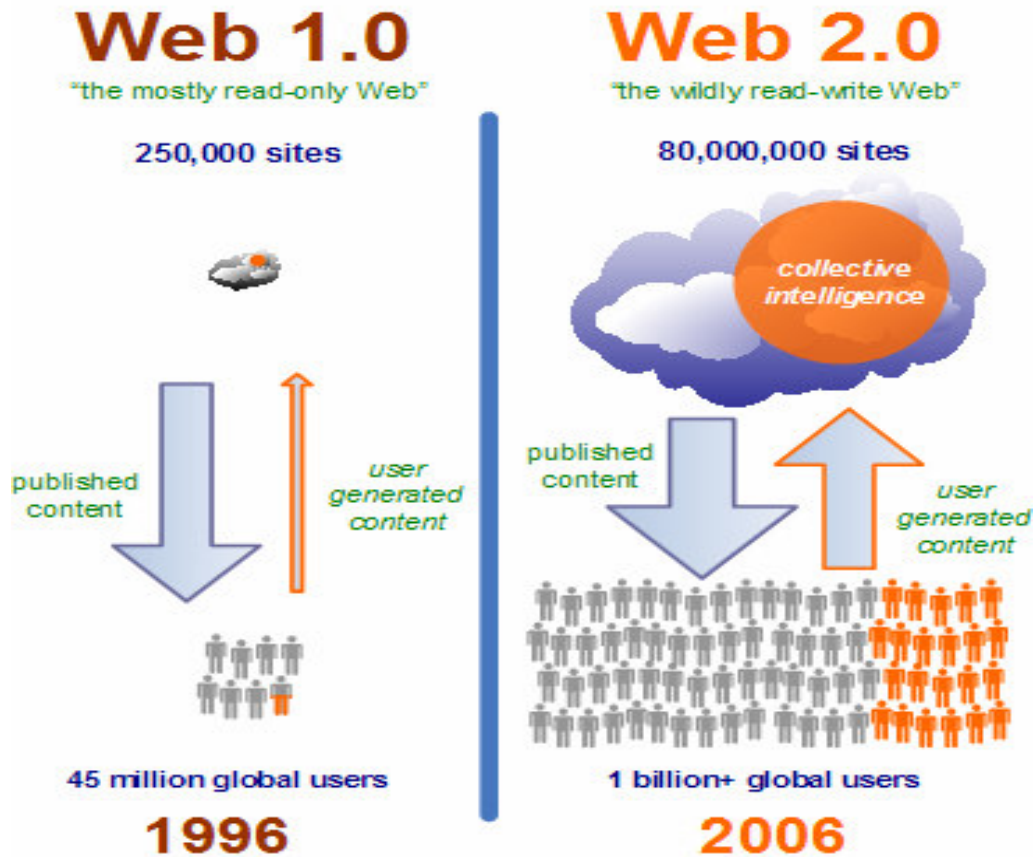
**Tagging:** Tagging is a practice of annotating the web content by creating tags or keywords to identify them. Also see: Folksonomy

**Web Service:** Web Services are the modules enable a programmer to use these modules in their programs without actually coding them. It allows machine to machine data transfer between applications in XML format that follow the SOAP standard.

**Wiki:** Wiki is a web application which allows you to easily create and edit content on the web. It is used to create collaborative websites and is a great tool for social content creation and organization.

**XML:** eXtensible Markup Language is a markup language which allows users to generate their own tags. It enables creation of structured data that could be easily exchanged over the web.

### 3.2 Web 1.0 vs. Web 2.0



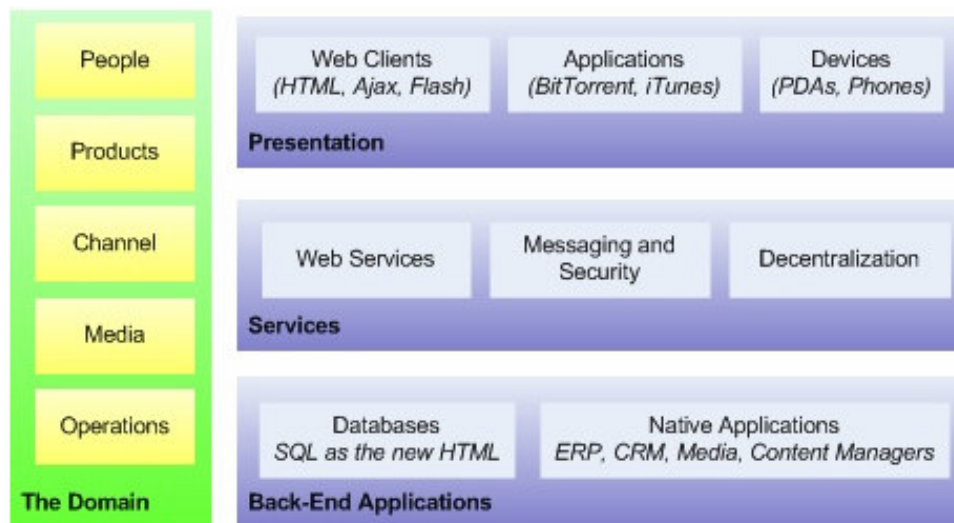
The transition from Web 1.0 to Web 2.0 can be effectively presented by the following transformations:

- Read only to read/write web
- Less user generated content to more
- 250,000 sites to about 80 million sites
- 45 million users to 1 billion users
- Static published content to user contributed dynamic published content

#### 4. Convergence of SOA and Web 2.0

---

There are at least two significant connection points which relate Web 2.0 and SOA. One is that Web 2.0 can be conceptualized as a global SOA. Two, that many traditional brick-and-mortar business that are currently using SOA as their architectural model will want to connect their Web/Web 2.0 faces up to their SOA. This makes Web 2.0 not just being the Global SOA but makes meeting smaller SOAs everywhere not just likely, but inevitable. Note that the respected industry analysis firm, Gartner, recently said that by 2008 that 80% of all software development will be based on SOA. And interestingly by 2006, Gartner believes that 60% of the \$527 billion IT professional services industry will be based on exploiting Web services and technology. We're talking serious convergence of focus here folks. If true, this means that more than half of all software development, SOA and otherwise, will revolve around the Web, inside or outside organization boundaries. All of this means Web 2.0 and SOA will meet each other both coming and going, and begin to become each other as well.



## **5. Software as a Service**

---

---

Another concept closely related to SOA is the notion of Software as a Service (or SaaS). Simply put, SaaS can be defined as "software deployed as a hosted service and accessed over the Internet."

SaaS as a concept is often associated with the application service providers (ASPs) of the 1990s, which provided "shrink-wrap" applications to business users over the Internet. These early attempts at Internet-delivered software had more in common with traditional on-premise applications than with modern SaaS applications in some ways, such as licensing and architecture. Because these applications were originally built as single-tenant applications, their ability to share data and processes with other applications was limited, and they tended to offer few economic benefits over their locally installed counterparts.

Today, SaaS applications are expected to take advantage of the benefits of centralization through a single-instance, multi-tenant architecture, and to provide a feature-rich experience competitive with comparable on-premise applications. A typical SaaS application is offered either directly by the vendor or by an intermediary party called an aggregator, which bundles SaaS offerings from different vendors and offers them as part of a unified application platform.

In contrast to the one-time licensing model commonly used for on-premise software, SaaS application access is frequently sold using a subscription model, with customers paying an ongoing fee to use the application. Fee structures vary from application to application; some providers charge a flat rate for unlimited access to some or all of the application's features, while others charge varying rates that are based on usage.

## 6. Web Technologies

---

---

The web technologies used for SOA and Web 2.0 can be divided into four classes: Client Side, Server Side, Web Services and Mashups. The subsequent sections will provide the details about these.

### 6.1 Client Side Technologies

The client side technologies are a set of programming methodologies which are sent as raw code to the client and are rendered by the application running on the user's system. Some of the commonly used technologies are:

**XHTML:** *eXtensible HyperText Markup Language* is a standard W3C markup language, which is a reformation of HTML adhering to the XML standard.

**CSS:** *Cascading Style Sheets* is a W3C standard for defining styles of the elements on a web page. It enables the separation of content from the layout, thus providing more flexibility and control of the elements on a web page. Also, it enables layering of the content on the page.

**AJAX:** *Asynchronous JavaScript and XML* is a scripting technique to develop interactive and fast web applications with enhanced functionality. The most commendable feature of AJAX is its ability of partial exchange of data with the server, enabling the changes to appear on the browser without reloading the complete page.

**VRML:** *Virtual Reality Modeling Language* is a standard file format for representing 3-dimensional (3D) interactive vector graphics, designed particularly for the Internet.

## 6.2 Server Side Technologies

Server Side technologies are a group of languages used to program the functionalities which have to be executed on the server. The most common examples of such functionality are the applications which require accessing the database. Some commonly used server side technologies are as follows:

**ASP.NET:** ASP.NET is a web application framework marketed by Microsoft that programmers can use to build dynamic web sites, web applications and XML web services. It is part of Microsoft's .NET platform and is the successor to Microsoft's *Active Server Pages* (ASP) technology. ASP.NET is built on the Common Language Runtime, allowing programmers to write ASP.NET code using any Microsoft .NET language.

**PHP:** *PHP Hypertext Preprocessor* is an open-source web scripting language to generate dynamic web pages. It can also be used from a command line interface or in standalone graphical applications.

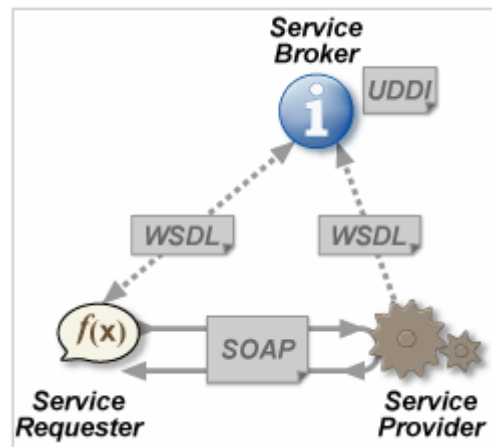
**CGI:** *Common Gateway Interface* is a standard protocol for interfacing external application software with an information server, commonly a web server. This allows the server to pass requests from a client web browser to the external application. The web server can then return the output from the application to the web browser. The external application may be written in any high level language like C++, Perl etc.

**JSP:** *Java Server Pages* is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

### 6.3 Web Services

The W3C defines a Web Service as "a software system designed to support interoperable Machine to Machine interaction over a network." Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The W3C Web service definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate using XML messages that follow the SOAP standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server written in the Web Services Description Language (WSDL).



**Simple Object Access Protocol (SOAP):** An XML-based, extensible message envelope format with "bindings" to underlying protocols. The primary protocols are HTTP and HTTPS, although bindings for others, including SMTP and XMPP, have been written.

**Web Services Description Language (WSDL):** An XML format that allows service interfaces to be described along with the details of their bindings to specific protocols. Typically used to generate server and client code, and for configuration.

**Universal Description Discovery and Integration (UDDI):** A protocol for publishing and discovering metadata about Web services that enables applications to find them, either at design time or runtime.

## **6.4 Mashups**

Mashups are an interactive genre of web applications which accumulates data from various sources and conflates it in a single integrated application. Mashup generally collects data from Web feeds (e.g. RSS or Atom), web services and screen scraping.

A mashup application is architecturally comprised of three different participants that are logically and physically disjoint (they are likely separated by both network and organizational boundaries): API/content providers, the mashup site, and the client's Web browser.

Much like how blogs revolutionized online publishing, mashups are revolutionizing web development by giving creative power to the masses. Many mashups are relatively easy to design with minimal technical knowledge, and thus custom mashups are being designed by unlikely innovators, utilizing data in creative and unique ways. While there are several useful mashups, many are simple novelties or gimmicks, with minimal practical utility.

Mashups are certainly an exciting new genre of Web applications. The combination of data modeling technologies stemming from the Semantic Web domain and the maturation of loosely-coupled, service-oriented, platform-agnostic communication protocols is finally providing the infrastructure needed to start developing applications that can leverage and integrate the massive amount of information that is available on the Web. As mashup applications gain higher visibility, it will be interesting to see how the genre impacts social issues such as fair-use and intellectual property as well as other application domains that integrate data across organizational boundaries, such as grid computing and business-to-business workflow management.

## **7. Tools for SOA and Web 2.0**

---

---

There are various tools available in the market for developing SOA and Web 2.0 based applications. Here are some of the Microsoft Products for developers:

### **Microsoft BizTalk Server**

BizTalk Server is a server product targeted at IT Professionals and Architects that enables customers to integrate systems, employees, and trading partners. With its core architecture based on XML and the .NET Framework, BizTalk Server fully supports all the open standards upon which Web services are built. BizTalk acts as the management layer that orchestrates Web services, controlling the flow and interaction between them and aggregating individual services into a larger composite solution.

### **Microsoft SharePoint Server**

By providing a simple, consistent user experience through familiar client applications, Microsoft Office SharePoint Server 2007 makes human-centric business process initiation, participation, tracking, and reporting easier and more flexible. Designed to help empower users to optimize the way people, content, and processes interact within and across organizations, Office SharePoint Server enables users to take advantage of workflows to automate and gain more visibility into common business activities like document review and approval, issue tracking, and signature collection.

### **Microsoft Visual Studio**

Visual Studio is the professional development environment for building applications on the Windows platform. Visual Studio enables consumption of Web services in Windows, Web, Mobile and Office-based applications. It also makes it easier for developers to publish and locate new Web services across the enterprise, and supports unit and load testing of Web services.

### **Microsoft SilverLight**

Microsoft Silverlight is a proprietary runtime for browser-based Rich Internet Applications, providing a subset of the animation, vector graphics, and video playback capabilities of Windows Presentation Foundation. Silverlight provides a retained mode graphics system and integrates multimedia, graphics, animations and interactivity into a single runtime.

### **ASP.Net AJAX (Atlas)**

ASP.NET AJAX, formerly code-named Atlas, is a set of extensions to ASP.NET developed by Microsoft for implementing Ajax functionality. Including both client-side and server-side components, ASP.NET AJAX allows the developer to create web applications in ASP.NET 2.0 (and to a limited extent in other environments) which can update data on the web page without a complete reload of the page.

### **Microsoft Expression Studio**

Microsoft Expression Studio is a suite of design and media applications aimed at developers and designers. It consists of: Expression Web - WYSIWYG website designer and HTML editor; Expression Blend - visual user interface builder for Windows Presentation Foundation applications; Expression Design - raster and vector graphics editor; Expression Media - digital asset and media manager; and Expression Encoder - VC-1 content professional encoder

### **Windows Live APIs**

It is a suite of APIs for programmatic access of various Windows Live services, from other web sites or applications. There are currently 13 services available on Windows Live Dev for developers to help build their applications on the Windows Live Platform: Alerts, Contacts, Custom Domains, Expo, Gadgets, ID, Messenger, Spaces, Spaces Photo Control, Writer, Search, Silverlight Streaming and Virtual Earth.

## References

---

1. [Book] *Dirk Krafzig, Karl Banke, and Dirk Slama. Enterprise SOA: Service Oriented Architecture Best Practices.* Prentice Hall, 2005
2. [Book] *Dan Woods, Thomas Mattern. Enterprise SOA: Designing IT for Business Innovation.* O'Reilly, 2006
3. [White Paper] **Enabling “Real World SOA” through the Microsoft Platform.** Microsoft Corporation, Dec 2006
4. [Article] *Darryl K. Taft. The Merging of SOA and Web 2.0* Jul 2007  
[http://midmarket.eweek.com/article/The+Merging+of+SOA+and+Web+2.0/211573\\_1.aspx](http://midmarket.eweek.com/article/The+Merging+of+SOA+and+Web+2.0/211573_1.aspx)
5. [Journal] **Web 2.0 Journal**  
<http://www.web2journal.com>
6. [Industry Resource] **Microsoft SOA**  
<http://www.microsoft.com/soa>