# <WEB 2.  >

*Report for*

COEN 332 Wireless Mobile/Multimedia Networks

*by*

**Ronald Bhuleskar**

**Vineet Agarwal**

Santa Clara University

Spring 2010

Ronald & Vineet

Digitally signed by Ronald & Vineet
DN: cn=Ronald & Vineet, o, ou, email=bhuleskar@gmail.com, c=US
Date: 2010.06.13 14:12:27 -07'00'

# Audience

This primary focus in this paper is a general audience having a fundamental knowledge of Web. Though we recommend reading the entire report to gain a full-fledged understanding of Web 2.0 and its related aspects, you may choose to read the topic of your interest. Following table will help you decide the contents to be read as recommended by the authors:

| Part | Audience |
|---|---|
| **Introduction** | Everyone |
| **Web 2.0 Architecture** | Students, Architects, Developer |
| **Mobile Web 2.0** | Students, Mobile Architects, Mobile Application Developer |
| **Security in Web 2.0** | Those interested in Web Security, Web/Appication Developer |
| **Developers Guide** | Developer who intend to build Web 2.0 based Application |

Specialized readers, who are interested in an in-depth study of specific patterns, may need not go through all the defined patterns listed in this report.

# Preface

To know the knowledge level of the audience, the authors conducted a small survey and found amazing results. In this Web 2.0 era almost 80% of the people are still unaware what Web 2.0 is. I am sure all those who participated in the survey has a facebook account or follow someone on twitter or have a professional connection with his boss or peer member of his/her team on LinkedIn, still many of those don't know what applications they are already using since years is a Web 2.0 application. Many answered Web 2.0 is soon take over the web with a new version of Internet. Is it true? We were quite amazed with many replies similar like this and a straight forward reply like *I don't know, what the hell is it? Another piece of crap!*

So we have decided to educate the young minds what technologies are they using today. With an introduction on how everything evolved i.e. Web 1.0, phasing out to Web 2.0, fading of semantic web and when to coin a term as Web 2.0. Web 2.0 is all about richer and interactive applications. It could be defined as enclosure of one or many patterns as defined in Chapter 3. Starting from building models to patterns, Patterns like collaboration which is used by Wikipedia, a Local Motor Company to develop a new Web 2.0 car and more.

As Tim OR'ielly coined the most important characteristics of Web 2.0 applications as platform independency hence a secondary focus has also been chosen as Mobile Web 2.0 among Web 2.0 architects and developers. Chapter 5 highlights the architecture of Mobile Web 2.0.

Considering security as another major aspect in Web 2.0 because of the loopholes in already existing system a high level of protection is desired for the Web 2.0 applications. These applications are openly available to the Internet but privacy and protection is a prime factor to be considered. Chapter 6 deals with most of the security aspect needed for protecting Web 2.0.

Also a developer guide has been included which attempts to teach some basic API's to interact with Facebook Web 2.0 application.

After you are done reading with the report you should have a firm understanding of Web 2.0 architecture, building Web 2.0 applications and more importantly determining that you are in the *You Era*, where you are the one who creates information and you are the one who uses it!

## Table of Contents

## Table of Figures

## Table of Tables

# PART I


WEB 2.0 Landscape
WEB APPLICATION
Widget/ component
Aggregation/ recombination
CONTENT SHARING
RECOMMENDATIONS/ FILTERING
Rating/ Tagging
Collaborative filtering
SOCIAL NETWORK
FeedBurner
Jetapot
ZOHO
jumpcut
Blogger
pbwiki
Basecamp
Ning
thinkfree
BlogLines
netvibes
snap
Rojo
Zillow.com
HYPE MACHINE
trulia
PANDORA
Findory
ourmedia
YouTube
edgeio
imeem
LinkedIn
flickr
LIFE
meebo
REVVER
FURL
myspace.com
Frappr!
blinx.com
reddit
movielens
PLAZES
Technorati
Opinity
SHOUTWIRE
digg
lost-fm
ROLLYO
www.futureexploration.net

# Chapter 1: Introduction

## 1.1 Introduction to Web 1.0

Tim Berners-Lee, the inventor of World Wide Web proposed his idea in 1989 and lately in 1990 an implementation of an HTTP client over the Internet was seen. Before the dot com bubble, the advent of Web 2.0 released in 2004 [1] any web design style previously used was termed as Web 1.0. There isn't a standard definition to coin what is web 1.0 but here I illustrate by some means for you to understand what Web 1.0 is before you start your reading further in this report on Web 2.0.

- Static Behavior
  Contents on the web site don't change unless the owner of the site wishes to.
- Non Interactive
  Users does not contribute to the contents of the site nor is able to alter the looks of his choice
- Proprietary
  Companies develop Web 1.0 application where the users are not aware how the application works nor alter its behavior. Web 1.0 not being open source developers even if they wish to, are not authorized how to mould the application to fit their needs.

## 1.2 Evolution of Web 2.0

Tim O'Rielly coined out a term Web 2.0 as a new generation of Internet. Though he suggested a new version for the web, no technical specification has been changed but it differs only in a way software developers build their applications and how the users use it. It is hard to tell in concrete what technical changes have brought up Web 2.0. Many had a debate does Web 2.0 really features a new version to Web or is just a marketing term. Tim Berners-Lee called the term as "just another piece of jargon".

People believed Web 1.0 as static, boring, passive read-only web where web is a source of information. Tim's idea to have it more interactive for users to have them involved in the web evolved the new version of the web.

## 1.3 Web 1.0 versus Web 2.0

As said it is a lot difficult to define what Web 2.0 actually is but it is not at all difficult to compare it with the earlier version of web i.e. Web 1.0. To prove this fact you can just search Web 2.0 on you tube and you will find tons of videos each defining Web 2.0 differently. Even today you will find people running behind Tim interviewing him and asking the same question "Tim, Could you please define what Web 2.0 really is?" This report as focused towards interested readers we will help you settle down supporting with the new term Web 2.0

Joe Drumgoole, founder of many startups like CloudSplit.com, PutPlace.com and more wrote on his blog[2] some notable difference presented to you in a tabular format to figure out the differences:

| *Web 1.0 was about ….* | *Web 2.0 is about ….* |
|---|---|
| Reading | Writing |
| Companies | Communities |
| Client-server | Peer to peer |
| HTML | XML |
| Home pages | Blogs |
| Portals | RSS |
| Taxonomy | Tags |
| Wires | Wireless |
| Owning | Sharing |
| IPOs | Trade sales |
| Netscape | Google |
| Web forms | Web applications |
| Screen scraping | APIs |
| Dialup | Broadband |
| Hardware costs | Bandwidth costs |

**Table 1: What Web 1.0 and Web 2.0 is about?**

Table 2 shows some examples [1] of websites in Web 1.0 and the corresponding site built in Web 2.0

| Web 1.0 | | Web 2.0 |
|---|---|---|
| **DoubleClick** | --> | Google AdSense |
| **Ofoto** | --> | Flickr |
| **Akamai** | --> | BitTorrent |
| **mp3.com** | --> | Napster |
| **Britannica Online** | --> | Wikipedia |
| **personal websites** | --> | blogging |
| **evite** | --> | upcoming.org and EVDB |
| **domain name speculation** | --> | search engine optimization |
| **page views** | --> | cost per click |
| **screen scraping** | --> | web services |
| **publishing** | --> | participation |
| **content management systems** | --> | wikis |
| **directories (taxonomy)** | --> | tagging ("folksonomy") |
| **stickiness** | --> | syndication |

**Table 2: Sense of Web 2.0 by examples**

## 1.4 Seven Principles of Web 2.0

In the paper [1] by Tim O'Reilly, he described that encapsulation of the below listed seven essential principles to have a Web 2.0 application

- Web as a platform
- Harnessing collective intelligence
- Data as the next Intel inside
- End of software release cycle
- Light weight programming models
- Software above the level of single device

- Rich user experience

Later in 2008, Andrew Yang, Dan Kim, Vishal Shalwani and Tri Vu in their paper [3] said that the defined Seven principles, which they defined as 7C's as: *Context, Content, Community, Customization, Communication, Connection, Commerce* are not enough and an addition '*C*'is desired to be added on to the 7C framework to complete the Web 2.0 core principles. The additional principle added was *Collaboration,* more details this new principle is described later in Section 4.3.3.

Table 3 describes the 8C Framework, in the context of Web 1.0 and Web 2.0[3]:

| Interface Elements | Meaning/Types in Web 1.0 | Meaning/Types in Web 2.0 |
|---|---|---|
| **1: Context** | How the site is organized, and how the content is presented to the users?<br>a. Functionalities: layout, performance<br>b. Aesthetics (look-and-feel): color schemes, visual themes | The Web 2.0 Web sites have layouts that are more<br>dynamic. The performance and dynamism increase<br>greatly by the use of technologies such as AJAX<br>and FLASH. |
| **2: Content** | What are offered by the site?<br>Offering mix is the mix of product and service information on a Web site;<br>Appeal mix refers to promotional and communication messaging, Multimedia mix deals with the choice of media;<br>Content type refers to the degree of time-sensitivity. | Collective Intelligence mix is the new addition<br>which deals with all traditional three "mixes" with<br>users participating in the generation of the content.<br>This is typical of Web 2.0 applications. |
| **3: Community** | Non-interactive communication;<br>Interactive communication (instant messaging, message boards, memberto-member emailing lists) | Collaborative communication may be enabled via<br>non-interactive and, most likely, interactive<br>communication mechanisms. |
| **4: Customization** | Refers to the site's ability to tailor itself (tailoring) or to be tailored by each user (personalization). | The content of the site can now be tailored in a<br>collaborative manner, since the content will be<br>user-generated. Also the customization can be<br>done in more dynamic fashion (desktop-like feel). |
| **5: Communication** | Site-to-user communications:<br>Broadcast, Interactive, and Hybrid | Site-to-user communications:<br>Broadcast,<br>Interactive, Hybrid, and Push/Pull (e.g., RSS) |
| **6: Connection** | Refers to the extent of formal linkage from one site to others: outsourced content, percent of home site content, and pathways of connection. | Lots of content from external sites may be pulled<br>in the form of blogs, advertisements, mash-ups,<br>etc. |
| **7: Commerce** | Deals with the interface that supports | Deals with the interface that supports |

| | | |
|---|---|---|
| | the various aspects of e-commerce, such as shopping carts, security, order tracking, etc. | the various aspects of e-commerce, such as shopping carts, security, order tracking, affiliates and advertisements, etc. |
| **8: Collaboration** | Generally in the form of feedback forms, forums, and bulletin boards. | Refers to the site's ability to provide users with interface and services to carry out high degree of collaboration, such as collaborative editing, project managements, etc. |

**Table 3: The 8C Framework, in the context of Web 1.0 and Web 2.0**

In the later part of this report you will not just understand what these principles are but also how to achieve them.

## 1.5 Semantic Web

Semantic web is an evolving development of WWW in which the information and services in the web is defined such that the machines and users can utilize the web content. Semantic web is expressed in formal specifications. The purpose of semantic web is to let computers understand the information, so that the computers can perform more of the tedious work in finding, combing and acting upon information on the work [4]. The Semantic web is regarded as an integrator across different content and information applications and systems, and provides mechanisms for the realization of enterprise information systems. Semantic web didn't take off well because of the machine readability and less user friendliness and also it was difficult for the developers. Some more limitation of Semantic web includes:

- Vastness
- Vagueness
- Uncertainty
- Inconsistency
- Deceit

For more detailed information on limitation refer [4].

## 1.6 Technology and Standards

Traditional way to think software applications to run is to install it on user's machine. With the era of Web 2.0 primary focus of the developers is to have a Web as a platform to distribute and run the applications [5]. With the latest inventions in browsing technology, it has swiftly moved its focus to Rich Internet Applications aka RIA. Today now the primary technology for delivering RIA's is AJAX. Recently in December of 2009 Adobe has also released an alternative to flash, Adobe AIR to build richer applications easily. With gaining popularity with Flash developers and a recent collaboration with Intel to use Adobe AIR with their Atom release recommends Atom developers to use Adobe AIR [6] as their user interface with the native C/C++ as a back end development toolkit.

Few Web development and architectures are listed below:

### 1.6.1 AJAX

Asynchronous Javascript + XML – a term first coined by Jesse James Garrett in 2005. AJAX uses not just one but various technologies to eliminate the frustrations for users of traditional HTML-based websites to reload the entire web page.

Several attempts have been made over the years to improve the dynamism of webpages through individual techniques such as Javascript, hidden frames, Dynamic HTML (DHTML), CSS and Microsoft's XMLHttpRequest ActiveX tool. However, it is really only with the introduction of Ajax that this has come together successfully [5]. With Ajax, only small amounts of information pass to and from the server once the page has first been loaded.

The Ajax technologies are:

- HTML/XHTML (a standardsbased way of presenting information within the browser)
- Cascaded Style Sheet (CSS)
- Document Object Model (DOM) (a way of dynamically controlling the document)
- XML (data interchange and manipulation)
- XSLT (data interchange and manipulation)
- XMLHttpRequest (asynchronous data retrieval from the server)
- Javascript (or ECMA script)

A detailed overview of Ajax and its application in Web 2.0 services is provided by the Open Ajax group [7]

### 1.6.2 Alternatives to AJAX

Once AJAX is out of your mind let me explain you what could substitute it. With the use of Flash—the ubiquitous graphics plug-in from Macromedia (now Adobe) that first appeared in the 1990s. It allowed sophisticated, but quick-to-download, vector graphics and animation to be displayed in the browser window. Flash requires a browser plug-in to work, although within only a few years of its launch 99% of computers [5] had the necessary addition to support it.

Flash is still being used to deliver compelling content within the browser (in fact the Flash video player is beginning to take off because YouTube have adopted it).It has been used as the basis of other RIA development tools, including Adobe's Flex, Adobe's AIR and OpenLaszlo.

Other than the Flash-based systems there are several emerging technologies such as Microsoft's WPF/E32, XBAP, Mozilla's XUL and the related XAML33, which focus on displaying rich graphics within the browser window.

### 1.6.3 SOAP vs REST

*'At the heart of REST is the idea that the web works precisely because it uses a small number of verbs applied to a large number of nouns.'*- McGrath, 2006.

There is an on-going debate within these technologies that circles over simplicity vs. sophistication.

A further strand in the development of Web technology is the use of what are called lightweight or simplified programming models, which facilitate the creation of loosely coupled systems. This flexibility is a source of

debate since, the lightweight 'ideal' is often viewed in contrast to the production of more robust Web Services which use what are seen as the 'heavyweight' and rather formal techniques of SOAP and WS-*.

Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of web service. It relies on XML as its message format relying on application layer such as HTTP for message transmission [8]. On the other hand Representational State Transfer (REST) is a style for software architecture built around the transfer of "representations" of "resources". It constraints certain interactions called as macro-interactions [9] between the four web components viz., servers, gateway, proxies and clients. More characteristics of REST can be obtained at [10]

In REST, every resource is identified by a URI and the use of HTTP lets you communicate your intentions through GET, POST, PUT, and DELETE command requests. SOAP and WS-*, on the other hand, are more formal and use messaging, complex protocols and Web Services Description Language (WSDL).

Sean McGrath describes the Web as an enormous information space, littered with nouns (that can be located with URIs) and a small number of verbs (GET, POST etc). Where SOAP is more of a Verb Noun system, he argues that SOAP/WSDL allows the creation of too many (irregular) verbs [5].

### 1.6.4 Microformats

Microformats are used by web developers to embed semi-structured semantic information buried within certain XHTML tags (such as 'class' or 'div') or attributes (such as 'rel' or 'rev'). These additional information are not used by the browser s but can be effectively used by applications such as search engines as a factor to search on.

hCard is a perfect example where personal or an organizational contact information are stored based on the standard vCard to be embedded in a webpage. XHTML Friends Network (XFN) is an HTML microformat that allows representing a human relationships [11] using links.

Similarly to microformats, there are also semantic web technologies like Friend of a Friend (FOAF) [12] that is machine-readable ontology describing persons, activities and their relation to other person or activities. This allows social network to work without a need of a central database.

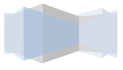
The use of microformats is not without its detractors and debates around this subject tend to be centered around whether they help or hinder the process of moving Web content towards the Semantic Web vision (they are sometimes referred to as the 'lowercase semantic web') have bearing on the on-going and wide-ranging discussions over the merits or otherwise of the use of lightweight (REST etc.) or heavyweight (SOA etc.) approaches and solutions [5].

### 1.6.5 Open APIs

An Application Programming Interface (API) features a way for programmers to make use of the functionality without having access to the source code. An API is often described as open, that doesn't require the programmer to license or pay royalties. Such 'open' APIs have helped Web 2.0 services develop rapidly and have facilitated the creation of mash-ups of data from various sources. The application of Open API's is seen in section 4.3.6.

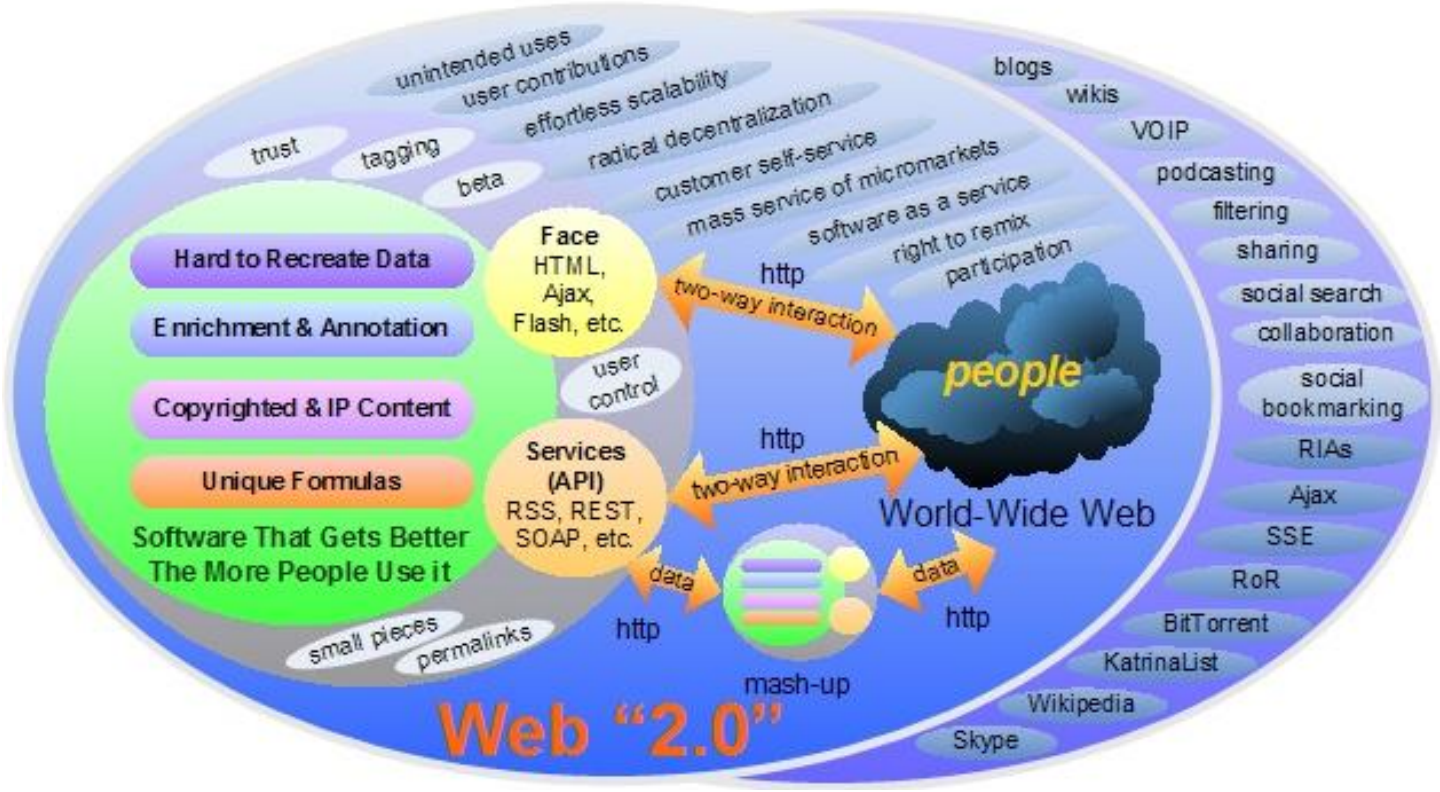


Typically the API's are defined set of HTTP request messages and a structured response expressed in JSON or XML. More information on Web API's can be found in [5][13].



One way of finding out what APIs are available is to look at the Programmable Web website (http://programmableweb.com/), which keeps track of the number of APIs and what people are doing with them.

# PART II


unintended uses
user contributions
effortless scalability
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
trust
tagging
beta
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype
Hard to Recreate Data
Enrichment & Annotation
Copyrighted & IP Content
Unique Formulas
Software That Gets Better The More People Use it
Face
HTML, Ajax, Flash, etc.
http
two-way interaction
user control
people
Services (API)
RSS, REST, SOAP, etc.
http
two-way interaction
World-Wide Web
data
data
http
http
mash-up
small pieces
permalinks
Web "2.0"

# Chapter 2: Modeling Web 2.0

## 2.1 Introduction to Models

Model are different from architecture and patterns, they are independent of an implementation. A model is an abstract representation of a set of concepts or components of a process, system, or structure, generally developed to aid understanding and analysis of a class of things. It helps to capture knowledge the components and concepts and the relationships between them. As said because of their abstractness they can't be directly implemented. The Figure 1 helps you understand the relationship between the model, patterns and architecture.

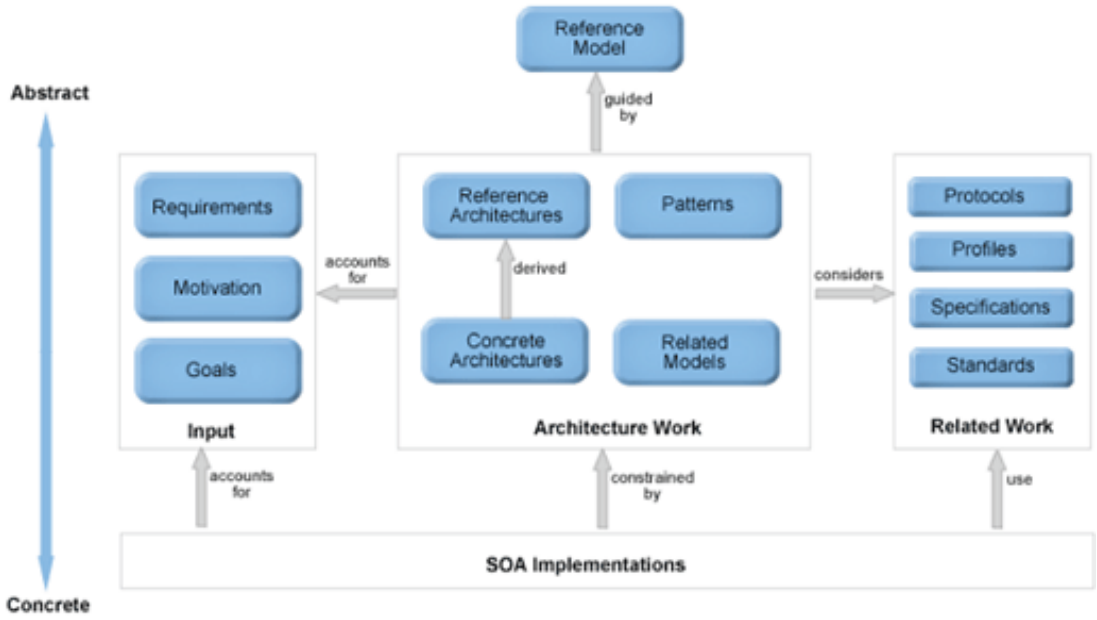



**Figure 1: Guiding developers to use a reference model**

The above reference model is defined by Organization for the advancement of Structures Information Standards (OASIS) to guide architecture projects. They worked on defining an abstract model, which can quickly confuse non-architects who have trouble distinguishing between concrete architecture and abstract models. A reference model (RM) is an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. A reference model is not directly tied to any standards, technologies, or other concrete implementation details. Hence, a good reference model provides common semantics that can be used unambiguously across and between different implementations.

## 2.2 How to use Models and Patterns

The Figure 2 shown below shows a process of mining patterns from examples and then reconstructing models and architecture based on those patterns. Many of the people do such type of activity, they don't know how

to solve a particular problem but they know what they can use to solve it without really knowing the process. The following figure defines concept maps as proposed by OASIS [14]


**Figure 2: Concept Maps**

The concept shown in Figure 2 is easy to understand and use. Concept 2 is what you want to build while you don't know how to exactly do it but you are aware that if you use Concept 1 your task gets easier and simpler. Developing Concep2 is just utilizing the features existing with Concept1 for your own benefit. The following figure shows a lineage of mining patterns from examples and then constructing models and architectures based upon those patterns:

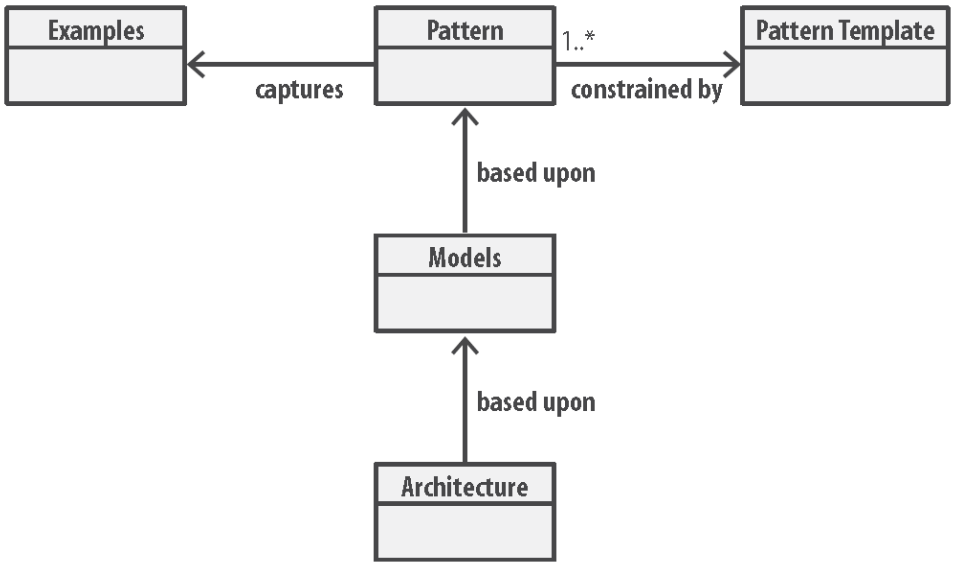



**Figure 3: Constructing models and architecture based on commonalities in the pattern.**

The Figure 3 shows a typical example of a Web 2.0 development approach. This methodology is used to mine patterns from an existing repository (in the above case examples), capturing knowledge and then reconstructing models and architecture based on commonalities in the pattern.

## 2.3 Client/Server Model for Web 2.0

This model of interaction differs widely from the traditional client/server interaction wherein the client communicates with the server with the request for services, server receives the request - processes it – and sends back the response to the request as a complete HTML build contents which the clients browser displays as a new web page in response to a single request sent. Web 2.0 patterns of interactions are more elaborate than the simple request/response interaction pattern. Web 2.0 practices requires interactions reach deeper and into more capabilities than the tradition one used in Web 1.0. Service Oriented Architecture (SOA), one of

the widely used patterns in Web 2.0 allows capabilities to be exposed and consumed via services across disparate domains of ownership. This makes it easier for other platforms to operate.

The Figure 4 demonstrates the five-tier model for Web 2.0. This model can be extended over time as new patterns and models are introduced.

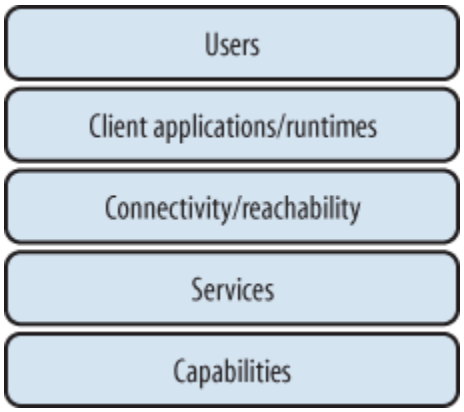



**Figure 4: A model for Web 2.0**

This model focuses on how Web 2.0 connects capabilities and users going far beyond the traditional client/server model used in earlier web development. A different model may be applied for a peer-to-peer network. This model could be applied even beyond the Internet itself. Tim O'Reilly mentions that software developers could use this model to expose the functionality encapsulated in a single class to the other software components via a defined interface throughout the specific environment (connectivity/reachability) in which the interface exists.

The Internet is a platform used to connect devices (basically via "services"), but much more. As more and more devices are connected to the Internet, the services have to more carefully think out in terms of their architecture, implementation and descriptions. The client in the traditional model is broken down into three specific aspects of the client: applications, runtime and the users - that are the ultimate targets, largely the humans as part of the machine, the model must reflect their existence [15].

Let's have a deeper look on each layer in the 5 tier model:

### 2.3.1 Capabilities

Any functionality that leads to a real-world effect can be termed as a capability. It doesn't necessarily reside in a computer; it can be any device that can respond/serve to users or to its request. Capabilities can be owned or provided by government departments, businesses of any size, or individuals with some backend functionality they wish to share with users (for example, the ability to share a text file with another person).

In typical client/server architecture, during certain phases of communication a client assumes some of the roles of a server. Similarly, a user can be (or become) the provider of capabilities. Figure 5 shows a classic client/server interaction with client requesting for services:

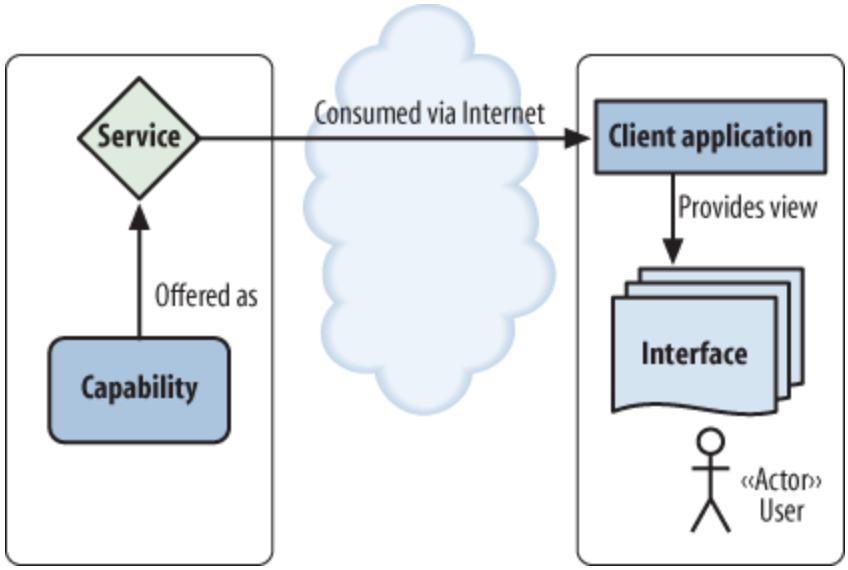



**Figure 5: Basic service-consumer pattern**

The next pattern shows how all of that may change if the user is working with a torrent application or similar P2P applications and protocols: the "client" in one transaction may subsequently become the provider of a capability that others consume.

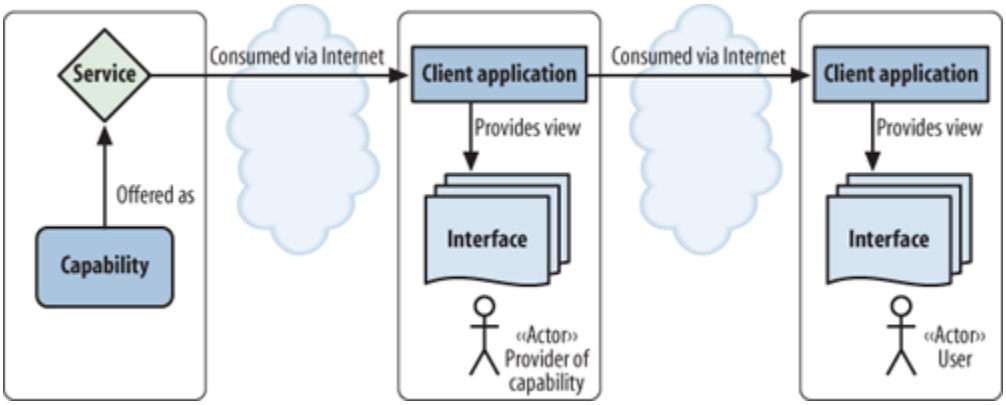



**Figure 6: A service consumer acting as a service provider or "intermediary"**

In the above figure, consider that you are a actor who consumed service by downloading a file from the service agent (i.e. capability). During the same time you are a leecher to someone else who is downloading the same file from your machine. Here you are contributing your resource (a file) to someone else.

### 2.3.2 Services

The services tier makes capabilities available for people and programs to consume via a common set of protocols and standards. Service consumers do not necessarily know how the services are being fulfilled, as the service boundaries may be opaque. Services have a data model that manifests itself as data (payloads) going in and coming out of the service during its invocation life cycle. Even though the payloads themselves vary, the service pattern itself remains unchanged.



The architectural model behind the services tier is generally referred to as Service-Oriented Architecture. Although it is often considered an enterprise pattern, SOA is an essential concept for the Internet as a whole. SOA is a paradigm for organizing and using distributed capabilities that may be under the control of different

ownership domains. The reference model for SOA emphasizes that entities have created capabilities to solve or support solutions to the problems they face in the course of their business. The most important use of services goes this way: One entity's needs could be fulfilled by capabilities offered by someone else; or, in the world of distributed computing; one computer agent's requirements can be met by a computer agent belonging to a different entity. This can be thought of a student writing an assignment and publishing it online while other students can refer to the same solutions reducing the overall efforts he needs to put-in.

SOA accounts for the fact that these systems are distributed over disparate domains of ownership, which means that they require additional mechanisms in order to consume each other's services. The granularity of needs and capabilities varies from fundamental to complex, and any given need may require combining numerous capabilities, whereas any single capability may address more than one need. SOA is an architectural discipline that provides a powerful framework for identifying capabilities to address specific needs and matching up services and consumers. [15]

Within SOA, if service interactions are to take place, visibility and reachability are the keys. Visibility refers to the fact that what services could be used with the given set of capabilities. This requirement can be met by using the standards and protocols common to the Internet. A Service interaction helps to provide with the service description, which return details the service's functions and technical requirements, related constraints and policies, and mechanisms for access or response. Services can be called from outside the system. For services to be invoked from different domains, these descriptions need to be written in a form in which their syntax and semantics will be interpretable. For Example Facebook themselves uses their own API's for their own website www.facebook.com while other sites uses their services to access services.

The concept of interaction is the key activity of using a capability often carried out by the exchange of messages. It is not mandatory to have message based services for e.g., a service could be invoked based on some event, such as a timeout event or a no-response event. A series of information exchanges and invoked actions may be processed for interactions with capabilities.

Certainly, not all services are to be visible. Facebook does not want the other developers to use all of their services to build their own platform with enhanced services and earn more than what facebook does. This drives architects to distinguish between public and private actions and realms of visibility. Private actions are inherently unknowable by other parties, yet may be triggered by publicly visible service interfaces. Public actions result in changes to the state that is shared between those involved in the current execution context and, possibly, others. Support for private actions is one of the key characteristics of services. Such actions are handled with "managed transparency." A service performing private functions that consumers cannot see hides the implementation of the capability it's delivering from the service consumers or users. This opacity makes it possible to swap underlying technologies when necessary without affecting the consumer or user tier. When implementing managed transparency, the service designers consider the minimal set of things consumers need to know regarding what lies beyond the service interface to make the service suit their needs. The declaration of what is behind a service is largely just a set of claims and cannot necessarily be monitored during runtime.

A drastic movement in the working of the Internet has been seen during the development in the services tier which forms the building blocks of Web 2.0. Clients consume multiple services to facilitate their functionality using the Mashup pattern. Software as a Service (SaaS) is a variation of the basic service packages in which functional units of computing are consumed via the service interface. Likewise, the rush to harness collective intelligence often involves making some capabilities available as services.

### 2.3.3 Connectivity/Reachability

There is a need to have some level of agreement on a set of protocols, standards, and technologies for interoperability on the media to be possible. The Internet became a visual tool usable by most of the computer users when browsers that supported the Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML) appeared. HTTP enabled data to be transmitted and HTML provided a foundation for declaring how data should be rendered visually and treated.

In cyberspace, protocols, standards, and technologies fulfill a major part of the infrastructure requirements to operate the Internet. Like any network, Internet has to have a common fabric for nodes to connect with each other. From a user's standpoint, search engines have become increasingly important in terms of visibility and reachability.

Several web services standards have evolved, along with standards to serialize and declare information. HTTP's core functionality, in addition to being the basic way for browsers and servers to communicate, is to provide a solid foundation on which to implement more complex patterns than the basic request/response cycle. The advancement of web services standards have been proposed to facilitate a new type of connectivity between users and capabilities. These standards include Asynchronous JavaScript and XML (AJAX), the Simple Object Access Protocol (SOAP), Message Transmission Optimization (MTOM), the use of SOAP over the Universal Datagram Package (UDP), Web Services-Security (WS-S), Web Services Reliable Exchange (WS-RX), and many others [15].

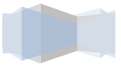
Its lot more than the plain HTML used on the Web today. Many sites now offer RSS for syndication as well as Atom, and microformats make HTML at least somewhat machine-readable without requiring developers to go all the way to XML or Resource Description Framework (RDF).

The most commonly used technology that most people describe with AJAX. It breaks away from the old pattern of complete web pages as single entities, AJAX allows developers to build sites that can dynamically update a web page or part of a view. Though AJAX builds on technologies such as JavaScript, the Document Object Model (DOM), and Cascading Style Sheets (CSS), as well as the XMLHTTPRequest object that had been lurking in browsers for years, the realization that these pieces could be combined to change the web-browsing experience was revolutionary. Developers were freed from the cage of pages, and users experienced improved interaction with resources without having to constantly move from page to page. AJAX shattered the previous model in which each page request was treated as being stateless, and no other requests were considered part of its context.

Unlike the commonly used wired connections for connecting devices, wireless communication is becoming by far the most popular connectivity option. Many devices rely on technologies such as Wi-Fi, Bluetooth, and other protocols to connect to the Internet. This evolution is important to understand from an architectural perspective, given that it's a strong driver for separating the actual Internet protocols (such as HTTP) from the underlying physical means of communication which drives us to focus on more enhanced study in chapter 5 of this paper on Mobile Web 2.0.

### 2.3.4 Client Applications/Runtimes





The traditional client tier of the client/server model remains with slight changes in the new Web 2.0 model, but some new issues are raised by Tim O'Reilly. The idea of SOA begs the question, "Why is everyone concentrating on the servers and not the clients for this new paradigm?" It's very important to mitigate the client-side connectivity considerations for online and offline tolerance to deliver a rich user experience [15]. A

hybrid approach for building applications between browser and a desktop application could be a better model for developers in the near future. A better model could be seen from the developments such as Adobe Systems's Adobe Integrated Runtime (AIR) and Sun Microsystems's Java FX.

The following figure shows the shifts on the part of both web-dependent and native applications to a new breed of client-tier applications that embody the best of both worlds.

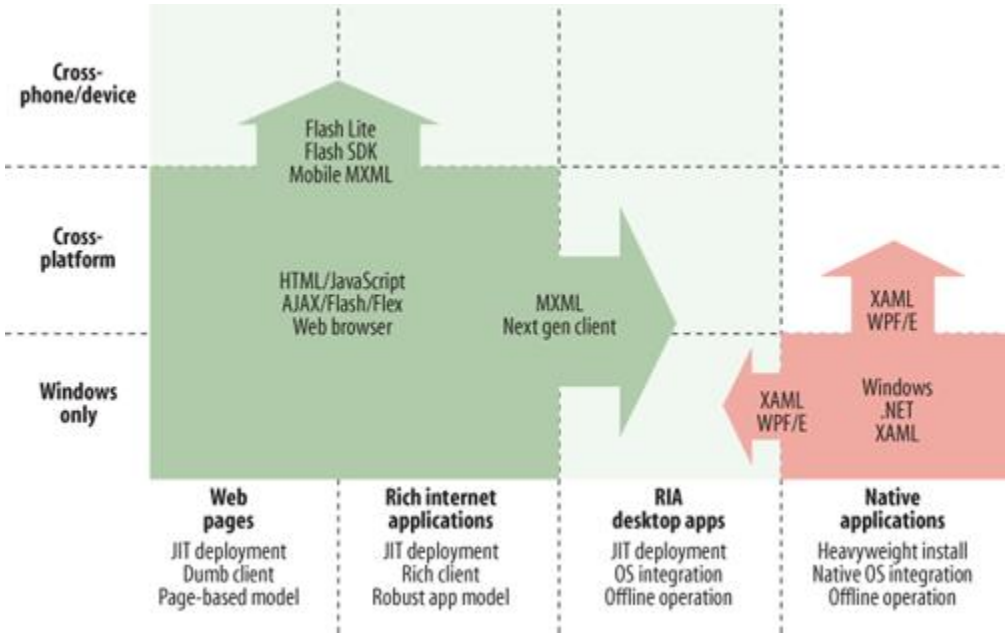


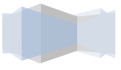

**Figure 7: "The landscape leading to hybrid online/offline development platforms"**

"RIA desktop apps" include traits from both old-school desktop applications and the latest web-delivered Web 2.0 applications, yet they also add some unique functionality. Typically, security models have kept applications delivered over the Internet from interacting with local system resources. The hybrid RIA ability to expand the basic security model to accomplish things such as reading and writing to local hard drives gives Web 2.0 developers an extended set of capabilities. AIR, formerly named "Apollo," was the first development platform to reach into this void, and Sun's Java FX and Google Gears soon followed.

### 2.3.5 Users

Unlike the users in Web 1.0, Web 2.0 tagged users differently. In Web 1.0 users were the focused as to view the information what someone has provided to the users. In Web 2.0 the source of information is the user itself. Including the user as a core part of the model is one of the main factors that distinguish Web 2.0 from previous revolutions in the technology space. A Web 2.0 user can be an individual who is on his computer, or looking at a PDA screen, interacting with information via a telephone keypad, using a large IBM mainframe, using a GPS-aware smart device where data triggers events, or uploading video content to YouTube, among many other things. The user is the provider of capabilities that the enterprise consumes, as well as the consumer of the interaction with the enterprise. Users may also share the results of an interaction with other users, taking on the capabilities role in that exchange.





Users have become an integral part and a primary focus of the Internet's current generation. In most early interactions and exchanges, users were typically anonymous actors who triggered events that resulted in

interaction requests. A person opening a web browser on a home computer visited sites to read their content but didn't exactly participate in them. Over time, websites made these interactions contextually specialized, taking advantage of knowing something the user and making the user an important part of the overall model for Web 2.0. Even users who were just visiting provided information their level of interest. Delivering a rich user experience requires understanding something the user; otherwise, you're just anonymously sending content that will likely be too generic to deliver such an experience. Examining the concept of harnessing collective intelligence illustrates one of the most notable aspects of how important the user is to Web 2.0.

Here is a list of five great ways to harness collective intelligence from your users:

**Be the hub of a data source that is hard to recreate**

Successes for web sites like Wikipedia and eBay are entirely dependent on the content their users contribute. Success in this context often requires first entry with a good implementation. It is said better not to wait until your technology is perfect; better get a collective intelligence technique that creates a user base virtually on its own from the innate usefulness of its data, in a niche where users are seeking opportunities to share what they have. A perfect example of this could be read in Section [16]

**Gather existing collective intelligence**

This is what Google does. There is an infinite number of existing information waiting out there on the Web to be analyzed, derived, and leveraged. You get thousand of results on when you google out for topics like "Web 2.0" but it gives you the most important result first and less relevant results later. Google uses hyperlink analysis to determine the relevance of any given page and ranks in its own database of content that it then shares through its own search engine.

**Trigger large-scale network effects**

The network effect is the most crucial factor of Web 2.0. When you can anyhow convince that your site is a great source for someone. You can possibly ask him to refer their friends to get some bonus points or some benefits in your plan. With 1.2 billion connected users on the Web[17], the potential network effects are theoretically almost limitless. Smaller examples can be found in things such as the Million Dollar Pixel Page[18], an attempt to raise $1 million by building web pages of 1,000 by 1,000 pixels and selling each of them for $1. Network effects can cut both ways and are not reliably repeatable, but when they happen, they can have huge profits.

**Provide a Folksonomy**

This is something Web 2.0 design patterns strongly encourage. Self-organization by your users can be a beneficial to allow the content on your site or your social software to be used in a way that better befits your community. Letting users tag the data they contribute or find, and then make those tags available to others so they can discover and access resources in dynamically evolving categorization schemes, can be a great boon to your website. Use real-time feedback to display tag clouds of the most popular tags and data; you'll be amazed at how much better your software works. This type of intelligence system works perfectly well for Flickr, Delicious and more, and it can work for you too. (Even if you don't provide an explicit tagging system, you may find that your users create one, like the #subject notation in Twitter. But you need to have a system to recognize the tags and use them)

**Create a reverse intelligence filter**

Blogosphere is the greatest example of a reverse intelligence filter, and sites such as Memeorandum have been using this to great effect. There are several factors which can be considered such as hyperlinks on the site, trackbacks to the site, and other references can be counted and used to determine what is important, with or without further human intervention and editing. Combine them with temporal filters and other techniques, and you can easily create situation-awareness engines. It sounds similar to the method of seeking collective intelligence, but it's different in that you can use it with or without external data sources. Plus, the filter is aimed not at finding, but at eliminating the irrelevant.

## 2.4 The You Era:

The major shift today has moved away from the traditional focus on data to user. The Figure 8 below describes the two approaches together with a set of users (on right) connected on the web which publishes media for users. While on the other hand infinite number of users (on left) which participate in a process of development and all the users use the data. The users of  the primary focus as they are the one who creates the data (participation) and they are the one who uses it (consumpotion).

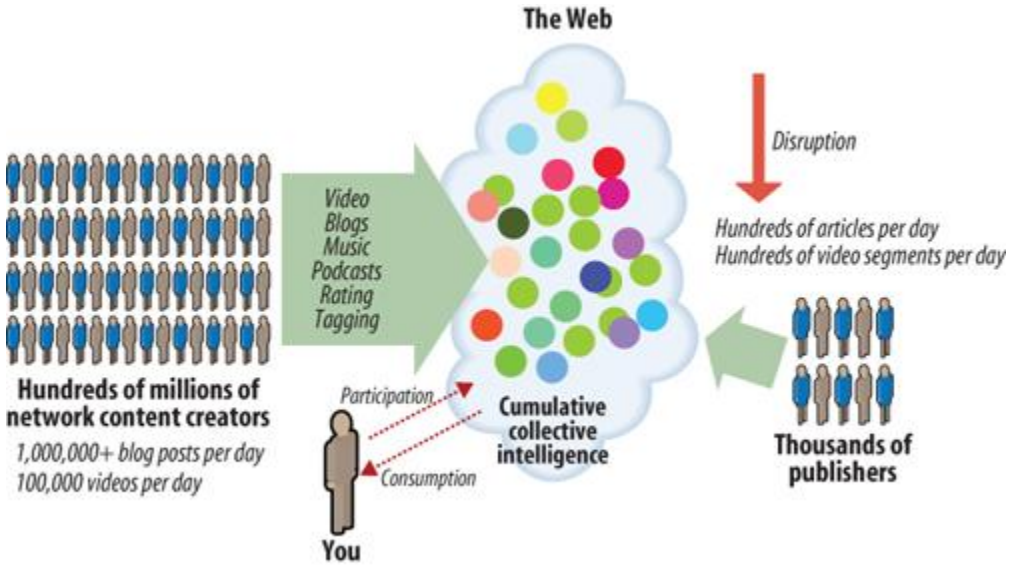



**Figure 8: The You Era: Consumer-generated content swamping and disrupting traditional media**

# Chapter 3: Architecture of Web 2.0

## 3.1 What is Architecture?

Architects design, describe and document systems and their structure, whether it is an internal or an externally visible properties. It also determines what relationship exists between them. There may be a data model view, technical infrastructure view, views from various "actors" who will interact with the system. Actors could be any one whether a person or any resource which interact with the system.

A building guideline for a good software development approach is to have a stable framework just like a blueprint of a building which determines the components or technological view of the software. It documents aspects that might not be visible to the naked eye. not just static architecture of the Internet; it is the patterns of usage between variety of things, whether human or machine, that wishes to use Internet as a platform.

## 3.2 Introduction to Architectural Patterns

Patterns are abstract design that can be applied as a solution for a common problem. There is no standard approach to define problem but many of them today uses following 3 elements:

- A Problem
- Problem Context
- A Solution

We will describe it in more details in the subsequent chapters.

## 3.3 Reference Architecture

Web 2.0 Reference Architecture does not provide any restriction on implementation, it just provides building blocks for the developers, architects and entrepreneurs to help them design and build Web 2.0 applications.

For software architects and developers, layered reference architecture serves to align their technical views regarding various aspects. It offers a good starting place for anyone wishing to develop applications based on the topic covered by the reference architecture. As with the model seen earlier in this report, you should view this reference architecture as a starting point for your technology road maps, not the one true normative architecture for Web 2.0 application development.

## 3.4 The Web 2.0 Reference Architecture

Till now we have seen the models of Web 2.0, now let us have a look on the reference architecture. The Web 2.0 Reference Architecture is an evolution of the abstract Web 2.0 model discussed in earlier. It is modeling Web 2.0, with more detail added for developers and architects to be considered during implementation. Many components exist at each layer, but at the top level of abstraction, they provide a foundation that applies to many different kinds of application.

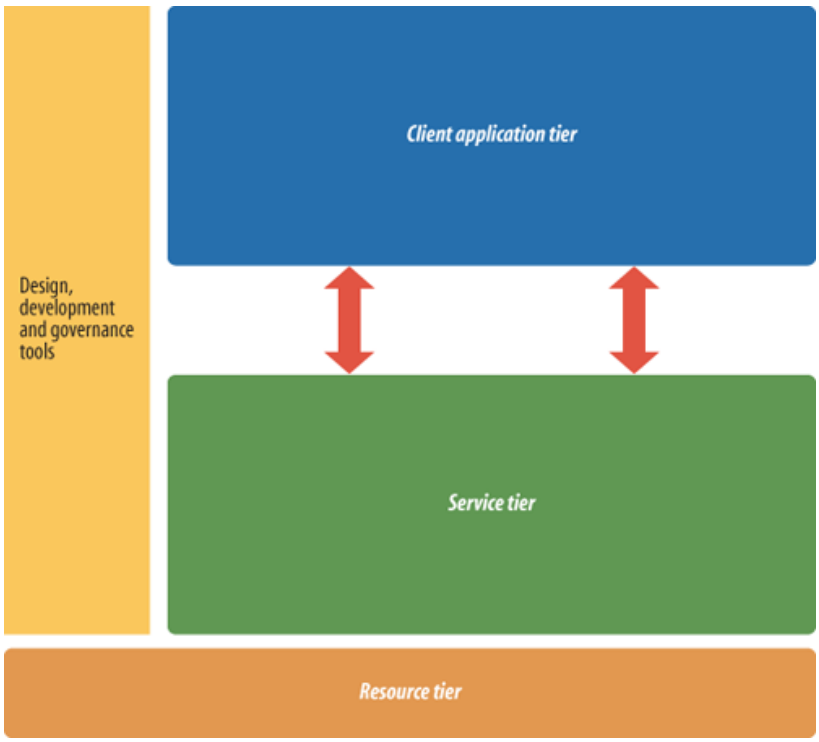



**Figure 9: Basic Web 2.0 Reference Architecture diagram**

As seen from the above diagram the major components of Web 2.0 architecture are:

- Resource tier
- Service tier
- Connectivity
- Client Application tier

Let us get an overview of each of them.

### 3.4.1 Resource tier

The lowest tier is the resource tier. It includes capabilities or backend systems that can support services that will be consumed over the Internet, i.e. the data or processing needed for creating a rich user experience. This typically includes files, databases, enterprise resource planning (ERP) and customer relationship management (CRM) systems, directories, and other common applications an enterprise, site, or individual may have within its domain [15].

### 3.4.2 Service tier

It lies between the client and the resource tier and all the functionality are packaged together which can be accessed as a service. This provides better control what goes in and out. Within enterprises, the classic examples of this functionality are J2EE application servers deploying SOAP or EJB endpoints. Web developers may be more familiar with PHP, Rails, ASP, and a wide variety of other frameworks for connecting resources to the Web.



### 3.4.3 Connectivity

In the above diagram this is not a block but the arrows which connect the client tier with the service tier. It provides a means to invoke the service to the upper tier. For any service to be consumed, it must be visible to

and reachable by the service consumer. There are also ways to invoke these services. Standards and protocols such as XML over HTTP, or other formats and protocols are used to handle connectivity.

### 3.4.4 Client tier

Client-tier software helps users to consume services and displays graphical views of service calls to users. Examples of client-side implementations include web browsers, Adobe Flash Player, Microsoft Silverlight, Acrobat, iTunes, and many more.

### 3.4.5 Design, development, and governance tools

There are a set of tools available with the designers and the developer to help them guide the development of the client and services tier of the Web 2.0 application. There are various tools available for the client and the service tiers. The client tier development tools usually plays with the graphical user interface and the way the user interact with the application. Various tools available today are Adobe Dreamweaver , Apple's developer tools xCode, DashCode, and more though there are many integrated development environments (IDEs) available like Eclipse, IBM WebSphere , etc and many developers have their own custom sets of tools.

## 3.5 The Detailed Architecture

Each of these tiers can contain a wide variety of components in them. The following Fig shows many more possibilities in greater detail.

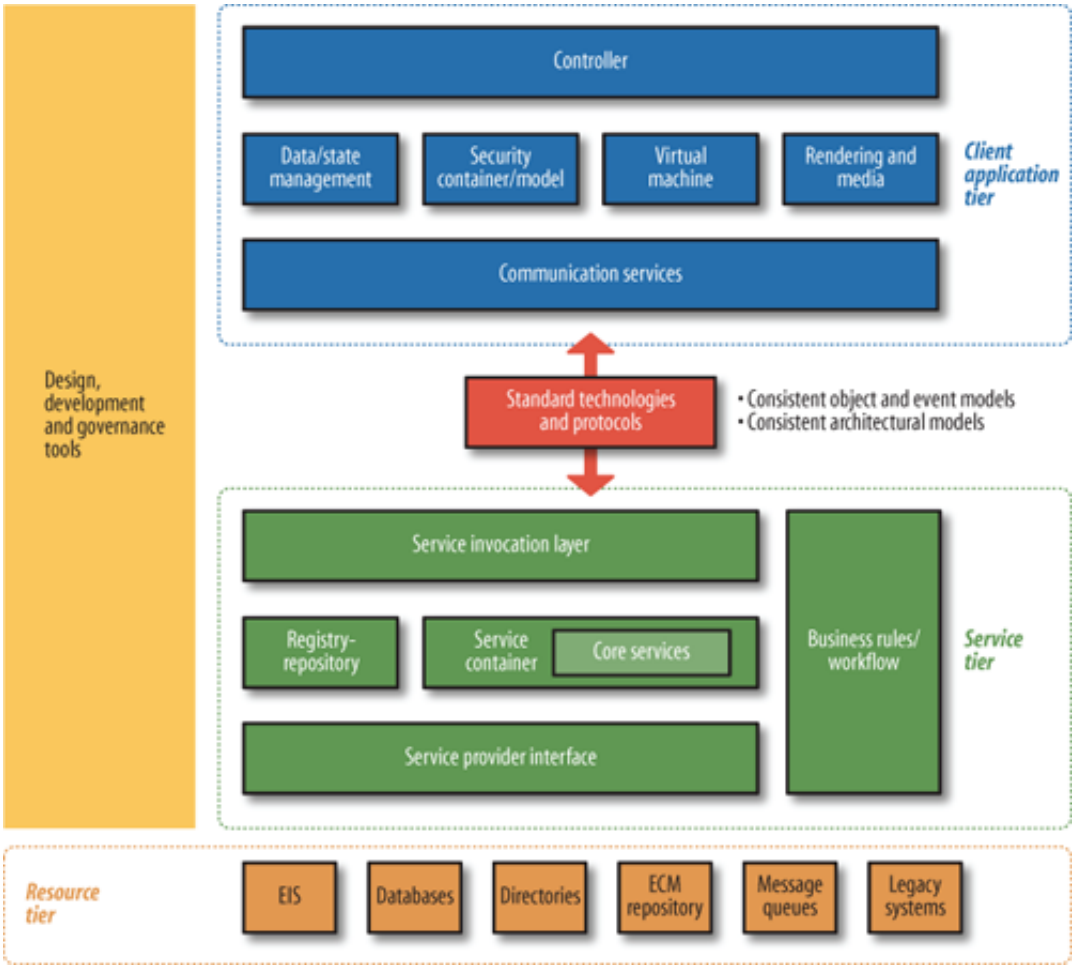



**Figure 10: Detailed reference architecture for Web 2.0**

Figure 10 above shows the general architecture with components involved in each tier. This reference architecture should not be considered "the" sole authoritative Web 2.0 Reference Architecture. It is meant as a reference architecture that decomposes each of the concepts in Figure 9, into more detail. It provides a starting point to the software architects or businesspeople when designing a way to implement a certain set of design or architectural patterns over the Internet.

Just to recollect once again, let us revise how technologies and protocols are tied with Web 2.0 architecture. Infact the truth is Web 2.0 Reference Architecture is not tied to any specific technologies or standards nor is it dependent upon them. Architects and entrepreneurs can later decide how to implement it using standards and technologies specific to their needs. For example, HTTP can be used if services need to be reachable by and visible to the largest possible segment of users, achieving its advantages likes simplicity, ability to pass through firewalls, and its widespread adoption. They are not just limited to using HTTP and could also opt for other messaging protocols to meet special requirements, such as Web Services Reliable Exchange (WS-RX) for reliable messaging, Web Services Secure Exchange (WS-SX) for enhanced security and efficiency with security, or BitTorrent for rapid distribution of multimedia content depending on the sole decision of architect in which would serve the purpose and what would work better.

One important thing to note is that it is not at all mandatory to implement all the individual components in each of the tiers. It is depending on the software architect to decide what is required and the software engineer to decide how to build it. The Figure 10 is based on a commonly used set of components to enable the patterns described later in this report in section 4.3.

### 3.5.1 The Resource Tier

This tier contains core functionality and capabilities and can be implemented in many ways depending upon the context. For example, a large enterprise might have an ERP system, an employee's directory, a CRM system, and several other systems that can be leveraged and made available as services via the service tier. A general overview on resource tier was seen in the previous section. The following Figure 11 maps few components used at the resource tier.


**Figure 11: Detail view of the resource tier**

The resource tier is increasingly being integrated into web applications in order to build rich user experiences. Many a times the developer prefers to move a part of the resource at the client for itself to handle. This increases the performance because of the client side local processing. This benefits users with high performance as well as rich user experience.

While the above figure shows few components at the resource tier, these are meant only as exemplars of potential resources, we'll look at each in detail to help you understand the overall architecture and some common capabilities:

**EIS**

Enterprise Information System (EIS) is an abstract name for a component common in most IT systems. EISs typically hold various types of data for use by those who run the company [15]. An example might be a short-term storage database or sensors feeding information into a common repository.

**Databases**

Databases are typically used to persist data in a centralized repository designed in compliance with a relational model. Other types include hierarchal databases and native XML databases.

**Directories**

Directories are lookup mechanisms that persist and maintain records containing information users. Such records may include additional details for authentication or even business data pertaining to their purpose.

**ECM repository**

Enterprise content management (ECM) repositories are specialized types of EIS and database systems. While they typically use databases for long-term persistence, most ECM systems are free to use multiple data-persistence mechanisms, all managed by a centralized interface.

**Message queues**

Message queues are ordered lists of messages for inter-component communications within many enterprises. Messages are passed via an asynchronous communications protocol, meaning that the message's sender and receiver do not need to interact with the message queue at the same time.

**Legacy systems**

The last component is a catchall generally used to denote anything that has existed through one or more IT revolutions. While some view legacy systems as outdated or slated-to-be-replaced systems, the truth is that most systems become legacies as a result of working well and reliably over a long period of time [15].

### 3.5.2 The Service Tier

At the core of the service tier is a service container, where service invocation requests are handled and routed to the capabilities that will fulfill the requests, as well as routing the responses or other events, as required.

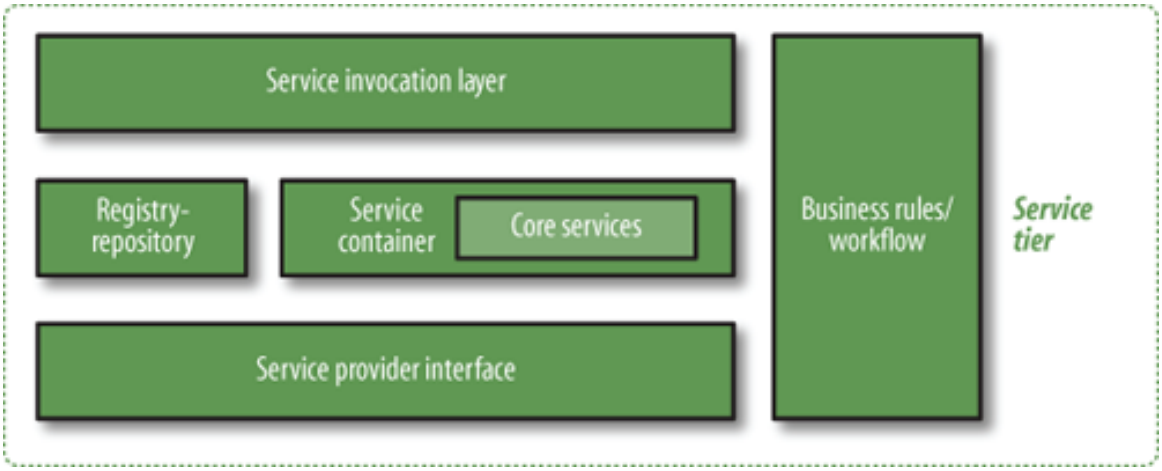



**Figure 12: Detail view of the service tier**

The service container is a major component in the service tier which controls the service invocation request at run time, and orchestrates the current state, data validation, workflow, security, authentication, and other core functions required to fulfill service requests. It coordinates and direct service invocation requests until the inevitable conclusion, whether successful or not.

We now illustrate several elements within the service tier:

**Service invocation layer**

At this layer all the listeners are plugged in which capture events, this may trigger services to perform certain actions. It may utilize several types of adapters or event listeners to allow invocation of services; for example, communication endpoints may provide the triggers (typically SOAP or XML over HTTP), or the services may be invoked temporally via timeout events, or even via internal events such as a change in the state of some piece of data within the resource tier. While many articles and papers focus on incoming messages arriving via SOAP endpoints, other forms of invocation are often used.

**Service container**

Once the service invocation layer invokes a service to be executed, a container instance is spawned which is asked to carry out the service invocation request. Two possible outcomes are possible: either successfully concluded or hits a fatal error. Service containers may be either short or long-lived instances, have permissions to access many common and specific services (core services) within the service tier, and can communicate with external capabilities via the service provider interface.

**Business rules and workflow**

Each service request needs to be traversed in a certain path in order to reach its final state. All service invocation requests are subject to internal workflow constraints and business rules. Examples include forms being routed to the correct personnel for approval or parameters for a request for data being authenticated to determine if the request meets the business's acceptance criteria.

**Registry/repository**

A central component that keeps track of services is the registry. This can be thought of a simple database possibly with a support to store multiple versions of services. A repository is a component used to persist resources or data needed during the execution of short or long-running service invocation requests. Data

stored in a repository is often referenced from the registry. The registry and the repository can be used both during design time, to orchestrate amongst multiple services, and at runtime, to handle dynamic computational tasks such as load balancing and resource-instance spawning and to provide a place where governance and monitoring tools can use data to give insight into the state of the entire service tier.

**Service provider interface (SPI)**

The SPI might be required to communicate with several different types of applications, that provide the capabilities to power a service. An SPI is required to connect to the resource tier, as the service tier makes existing capabilities available to be consumed as services. The resource tier in this case is a generic descriptor for a virtual collection of capabilities.

Let us have a look on the service tier in detail. The service invocation layer is where service invocation requests are passed to the core service container. The service invocation layer can hook into messaging endpoints (SOAP nodes, Representational State Transfer interfaces, HTTP sockets, JMS queues, and so on), but service invocations may also be based on events such as a timeouts, system failures and subsequent powerups, or other events that can be trapped. Client software development kits (SDKs), customer libraries, or other human or application actor interactions can also initiate invocation requests. In short, remember that several potential types of invocations are inherent in SOA design and to fulfill the patterns of Web 2.0 flexibility should be maintained by using the service invocation layer as a sort of "bus" to kick off service invocations. Realizing that patterns other than request/response via SOAP may be employed to invoke services will result in a much more flexible architecture.

### 3.5.3 The Client Application Tier

The core application logic at the client application tier of the Web 2.0 Reference Architecture is a controller, which manages several functional components. Every client application has some form of top-level control. The concept used here is in alignment with the controller concept in the Model-View-Controller (MVC) pattern.

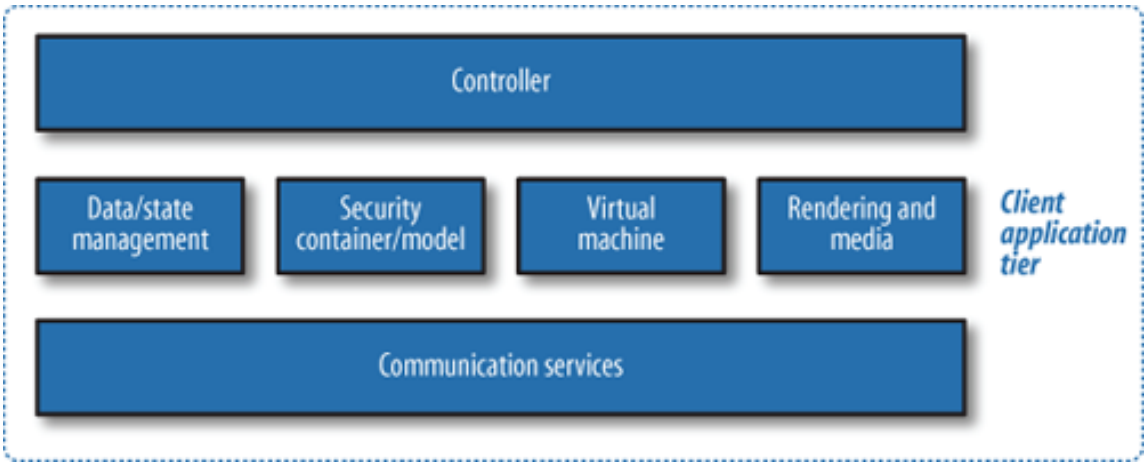



**Figure 13: Detail view of the client application tier**

It is possible for Web 2.0 clients to have several runtime environments running at a same time. Each runtime is contained and facilitated by a virtual machine as shown in the figure above. Thus, while the runtime environments are launched and managed by a single controller, they remain somewhat autonomous with

respect to executing scripts or bytecode to control certain aspects of an application. The virtual machine itself is a specialized type of controller, but it's controlled by a master controller that's responsible for launching and monitoring virtual machines and runtime environments as they're required.

To clarify the relationships, let's break out each component in the client application tier diagram, starting at the top:

**Controller**

The controller contains the master logic that runs all aspects of the client tier.

**Data/state management**

Any data used or mutated by the client tier may need to be held in multiple states to allow rollback to a previous state or for other auditing purposes.

**Security container/model**

A security model expresses how components are constrained to prevent malicious code from performing harmful actions on the client tier.

**Virtual machines**

Virtual machines (VMs) are plug-ins that can emulate a specific runtime environment for various client-side technologies.

**Rendering and media**

Management of the media and rendering processes is required to present a graphical interface to users (assuming they are humans).

**Communications**

The communications aspect of the client tier requires incorporation of various stacks and protocols so that it can communicate via HTTP and HTTPS, support the XMLHTTPRequest object, and more. With every client-tier application, communication services are required.

The client-side rendering engine handles all the "view" behavior for GUIs, as well as media integration. Rendering engines also pass information to the virtual machines and are controlled by the master controller. The data/state management mechanisms in the client tier control transformations, state synchronizations, transitions, and state change event generation during the object life cycle.

The communication services manage all communications, including between the client and server, with the host environment, and with resources in other tiers. Data/state management services, needs to keep aware of the connection status and understand when to locally cache data and where to go to synchronize data states once interrupted connections are reestablished.

## 3.6 Architectural Models That Span Tiers

To provide a modular interaction with the Web 2.0 system it is always advisable to build the client and the service tier using a common design principle. Architectural models such as SOA and MVC helps you achieve that. The objective of using MVC is to have an application built in such a way that allows reusing datasets for multiple views separating data from application logic and views. The components are tied together to build one single application. The following figure provides you with a graphical view on coupling of the 3 components of the MVC model:

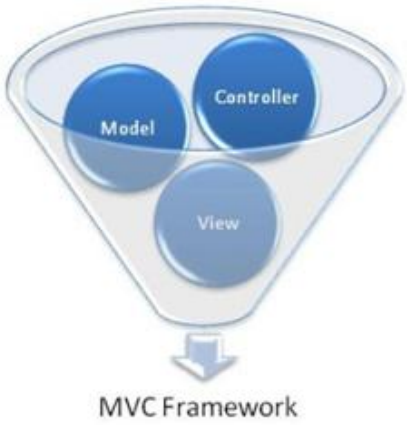



**Figure 14: The Components of MVC Model**

Let us have a look on each of them in detail:

### 3.6.1 Model-View-Controller (MVC)

In simple words MVC helps out separating the application logic from what the data is and how that looks (presentation). Web 1.0 mixed all the 3 components and MVC is a deployment pattern where the application code is broken down into three areas:

- **Model**

  This component is concerned with data of the application. It interacts with databases, file, etc maintaining the actual data needed by the business logic. It provides response to the controller for its request.

**View**

It is concerned with the graphical aspect of the application. Controller supplies the data received from the Model to the View component to build a graphical view for a user's browser. This component does no logical task instead just places out the data obtained to have a better look for the user.

**Controller**

The most crucial component of the MVC model is the Controller component which handles the actual business logic. It processes the users' request, determines what Model is required to execute it, retrieves the data and passes it to the View component before sending out the actual data to be displayed by the user.



There are different variations of the MVC model where different components can be laid out at various systems. Some believe that handling views at the user is better than the server to handle it. AJAX uses this

concept to build the views dynamically at the browser. Some system model handles Models with specialized systems to increase the efficiency of data processing.

The inter communication between components is a major responsibility of the system. The Figure 15 shows the communication between the three components and the messages that flows between them.

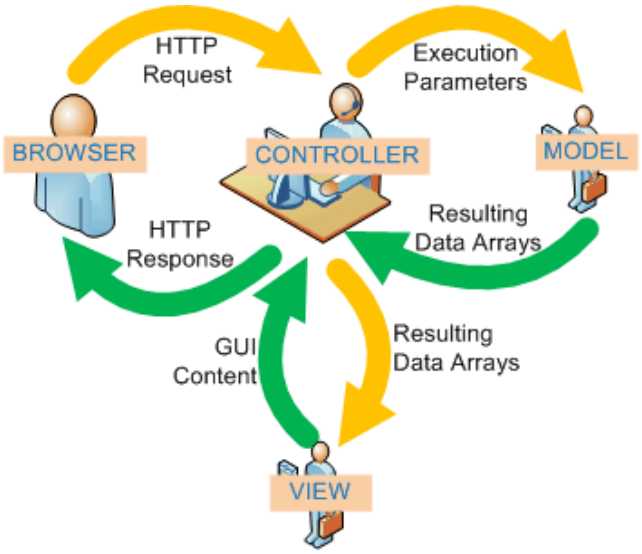



**Figure 15: Communication in MVC Model**

Web application development has evolved, however. On the client, HTML still provides a basic framework, but it has become more of a container for parts that support the different aspects of MVC more cleanly. While the HTML itself still often contains much information, XML and JSON offer data-only formats that clients and servers can use to communicate their data more precisely, according to models created early in development. Controllers have also advanced. Script languages such as JavaScript and ActionScript are great for client-side processing. The View components can be realized with a multitude of technologies; however, the most common are HTML/XHTML, while using variety of formats like JPEG, GIF, PNG, SVG, and Flash.

On the server side, new frameworks supporting MVC approaches have helped create more consistent applications that integrate more easily with a variety of clients and servers. There are variety of framework available today for e.g. .NET MVC Framework, simple implementation using servlets and JSP's for MVC, XForms and more.

### 3.6.2 Service-Oriented Architecture (SOA)

SOA replaces the traditional fashion of client/server communication where limited standard and protocols existed and there were no feature where services belonging to different owners to be utilized. SOA booms up replacing the traditional client/server communication architecture facilitating a better architectural pattern for the designers of Web 2.0 to connect to their applications to Internet.

SOA refers to an architectural style for software architecture, in much the same way that REST is an architectural style. SOA does not mean "web services," nor would every implementation of web services be automatically considered an SOA [15].

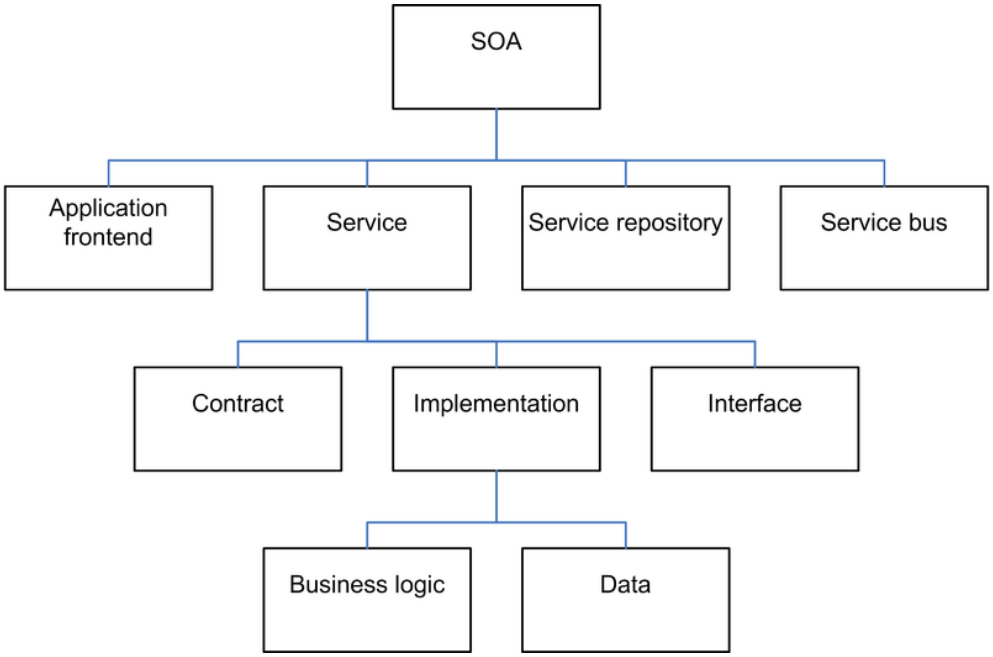



**Figure 16: Elements in SOA**

The figure above entirely describes "what SOA is?" By looking at the elements in SOA we are now clear of the ambiguity that SOA is not just a web service but web service is simply a part of SOA architecture.

SOA is not coupled with any standards or technologies. SOA also generally provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. Several disparate departments within a company may develop and deploy SOA services in different implementation languages; their respective clients will benefit from a well understood, well defined interface to access them. SOA defines how to integrate widely disparate applications for a world that is Web based and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality. An endpoint is the entry point for such an SOA implementation [19].

We will see more details on SOA in section 4.3.

## 3.7 Consistent Object and Event Models

There should be a consistency in message communication when someone outside the tier of the reference model needs to communicate. This leads to a design of consistent event and object models. Consider an Adobe Flex client side frontend application which couples a .NET backend as a server. If the client needs to capture events, the model of how events are generated and detected and how messages are dispatched has to be consistent throughout the entire application. SOA makes this somewhat easier by providing a clearly defined interface to the objects; however, the high-level models need to be aligned.

The Document Object Model (DOM) is the base technology used to address many XML and HTML pages, and the XML Infoset provides a further layer of abstraction. Even Adobe's Portable Document Format (PDF), Microsoft's Office format, and the Organization for Advancement of Structured Information Systems's Open Document Format (OASIS ODF) largely correspond to the same conceptual model for a document object. Most programming and scripting languages have also evolved in a similar manner to have a roughly consistent view of events and objects [15].

Some patterns today use events and objects across both client and server. Also advancement in Web 2.0 implementations is the ability to capture events on objects persisting on multiple systems, and triggers something in another application. These event/triggers can accumulate the results and syndicate them back to the client. AJAX applications support this model across systems with thousands of clients. For example, the Google personalized home page [20], provides you create your own template where you can customize your page and have informative blocks of your choice and position. The corresponding server then syndicates data accordingly for your browser to display in the blocks you chose. Small updates within the boxes are communicated via events asynchronously via the AJAX framework to refresh the view.

I am sure you now have the heads-up the models and architecture of Web 2.0. Next, we will start discussing patterns, but before that let's get an overview on the metamodel for architectural patterns.

# Chapter 4: Patterns of Web 2.0

## 4.1 Metamodel for Architectural Patterns

It is always good to document patterns as use cases and solution template for software architecture. It is not a standard; but simplifies the work by using the available patterns as shown in section 2.2.

## 4.2 What is Pattern?

Pattern is the recurring solution to the recurring problem and can be generally decomposed into 3 major components:

**Context**

Context is a set of situation in which the problem occurs. It may be highly generalized to ensure maximum applicability of the pattern or can be specialized to deal with only them specific problem.

**Problem**

The detailed issues which needs to be solved including the requirements, constraints, and desirable properties of a possible solution.

**Solution**

It solves the defined problem weather by its own or by using a solution of another problem as a subset of its solution. This can be displayed either by using UML's or an easier way of concept map as described in section 2.2. It also contains implementation guidelines for the developer.

A more detailed version of this model exists. It is out of scope for the readers of this report to understand it. You can refer [15] for more readings.

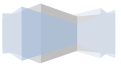
## 4.3 Patterns of Web 2.0

It is important to know that there is no finite number of patterns available today and it is feasible for the authors of this report to say that this report does not cover all the possible patterns instead gives you with the glance of the most important patters which are widely used today. There might be someone developing a new model while you are reading this. There are some terms which I would like you to be friendly with before proceeding to the details of SOA.

### 4.3.1 Service Oriented Architecture Pattern

SOA are often used with the term Event-Driven Architecture (EDA). Event-Driven Architecture is a style of software architecture that is built around the model of event processing; that is, a model in which the movement of information is facilitated by the detection, notification, and processing of events. Events of relevance include notable changes of state and discovery of facts.





It is, on the whole, a more powerful architecture. It removes from the originator of an event the burden of knowing what to do, and whom to call to do it, when action is required. It also removes the burden from the middleware of preserving the availability of and connectivity between parties so that the response can be

delivered and the context of the encompassing unit of work can be preserved. By pairing two event postings, EDA can easily simulate interactive SOA.

The authors of the paper would always recommend the software developers and architects to use the Event driven architecture instead of the traditional request/response driven forms of interaction because of its scalability, and resiliency and less costly over long run

**Problem**

Let's consider a real life issue to understand SOA.

An enterprise has a set of systems or applications that its employees use to accomplish various tasks. Each system is large and encapsulates a lot of functionality. Some of the functionality is common to more than one system, such as authenticating users before granting them access privileges to the system. The Figure 17 shows a set of systems, each containing its own module for logging in and authenticating users, a persistent record of the users' names and addresses, functionality for changing their names and addresses, and some human resources (HR) information. Each vertical grouping represents an application that fulfills a process. The only thing that differs between them is their central focus: payroll, insurance, or employee relations (ER).

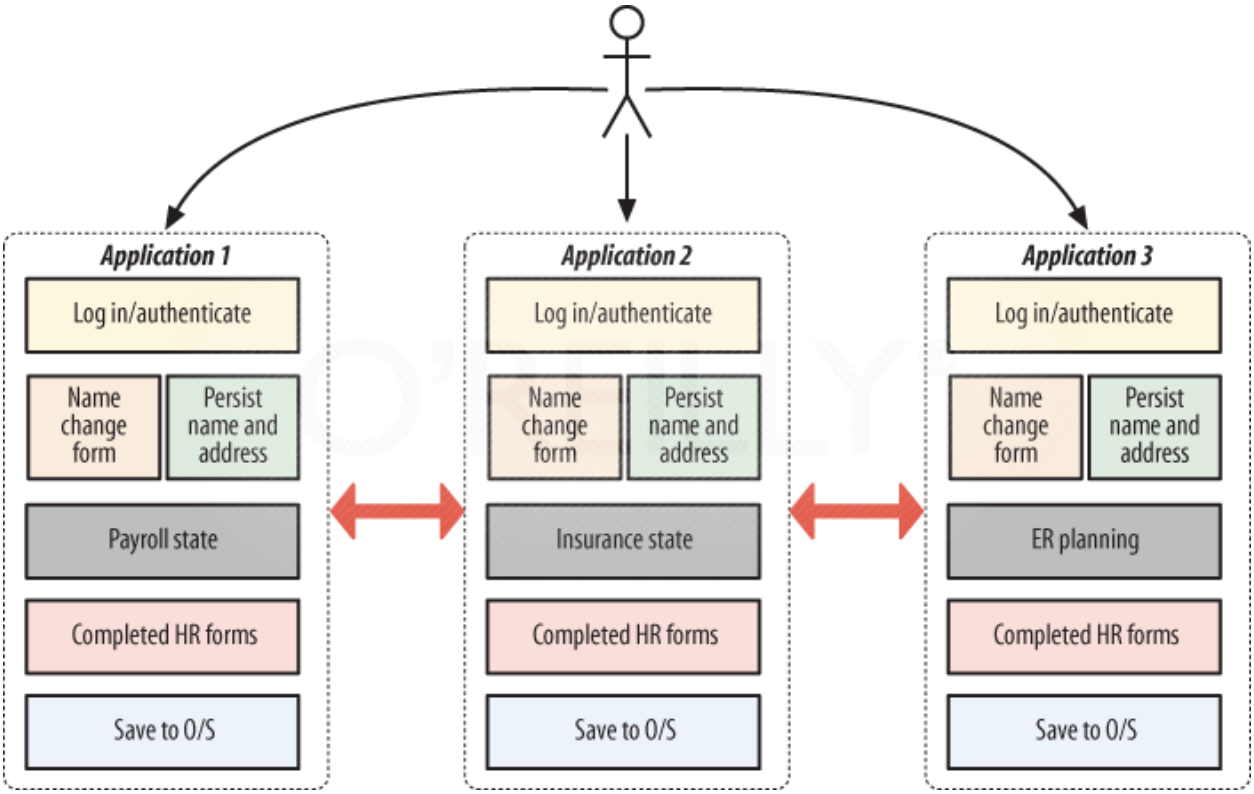



**Figure 17: A view of the business problem**

Intersystem communication are possible but at the consent of the system with the defined rules. Maintenance of the system is an expensive task as each time an employee changes her password, she has to reset that password on all the system which she has registered with after successful authentication to each system. Replacing a system or upgrading it to a newer version requires an extensive series of studies to understand what impact this upgrade might have on other connected systems.

Corporate acquisitions and mergers raise similar issues. When two companies come together, it makes little sense for the combined enterprise to maintain two customer relationship management (CRM) systems, two authentication directories, two payroll systems, two enterprise resource planning (ERP) systems, and so on.

**Derived Requirements for the given problem statement**

Each system that the enterprise owns and maintains must have a clearly defined interface so that it can be repurposed and used by multiple other applications, including external systems. Each interface should be linked to a specific policy whose terms a consumer must agree to comply with in order to use the service. The services can exist in different domains and can be owned by different owners, several factors must be considered to facilitate interoperation with the enterprise's guidelines.

A common set of protocols and standards must be used alongside a standard model for the services to facilitate coupling to the services. Ideally, use of common protocols and standards will create a service bus within the organization, enabling most applications to talk to each other without being hardwired together.

It's also important that all consumers use the same data formats as the service providers. This requirement logically encompasses the artifacts that describe various aspects of a service (e.g., the use of Web Service Description Language (WSDL) as the format for describing a web service). Service consumers should not know how the functionality is being fulfilled except the service interfaces i.e. the handle to use those services. This makes it easier to manage the functionality and capabilities behind the services, including replacing systems or changing external providers, without endangering the functionality of the systems using the services.

**Solution**

SOA is an architectural paradigm used to organize capabilities and functionality under different domains of ownership and to allow interactions between the consumers of capabilities and the capabilities themselves via a service. It can also be defined as an abstract action boundary that facilitates interactions between those with needs and those with capabilities. Organizations adopting this strategy can make all of their core functionality that is shared by two or more consumers available as opaque services. Opacity isolates the service consumer from the internal details of how a capability is fulfilled and helps keep the components of a system from becoming too tightly bound to each other.

The Organization for Advancement of Structured Information Systems (OASIS) Reference Model for Service-Oriented Architecture depicts SOA as an abstract pattern—not tied to any specific technologies, standards, or protocols—built from the components shown in Figure 18.

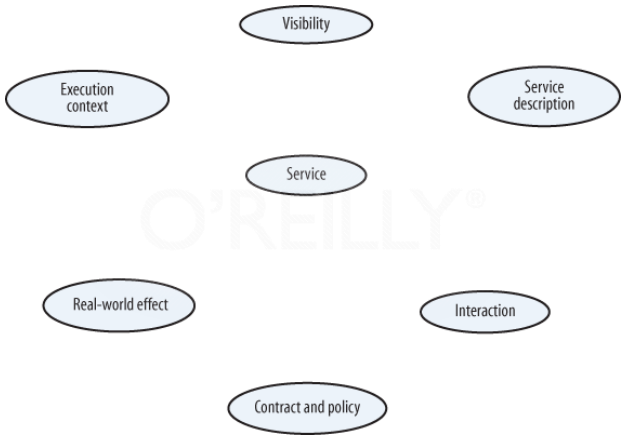



**Figure 18: The core reference model for SOA**

Because services fall under various domains of ownership, each service has a set of policies in place that govern its use. Failure to comply with the terms of those policies may result in service invocation requests being denied.

For a service to be consumed, it must be reachable by and visible to the consumer. This is typically accomplished via some common network bus, such as the Internet. The most common implementations of SOA use web services over HTTP, Asynchronous JavaScript and XML (AJAX), and several variations of Representational State Transfer (REST)-style (referred to as "RESTful SOA") services using different technologies, including plain old HTTP.

**Dynamic Behavior**

The dynamic pattern of interaction with a service describes the behavior of all actors working with the service at runtime. Before a consumer can consume the services he must be aware of the service's existence and purpose, including its real-world effects. It's best to use a specific, standard set of protocols and technologies across all service and consumer interfaces, creating an ad hoc service bus. There are many interaction models with services, but those described here are the most common.

- **Request/Response**

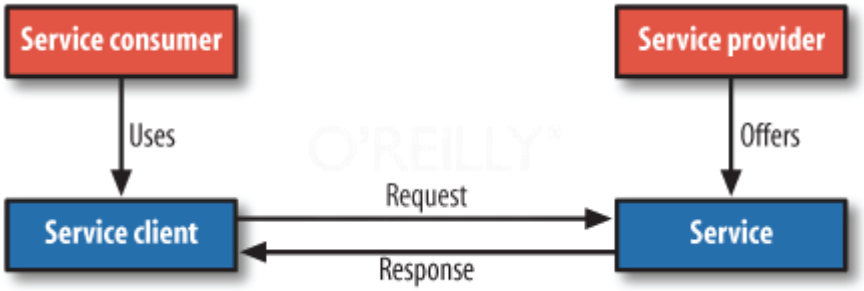



**Figure 19: Request/Response pattern in SOA**

Request/Response is by far the most common interaction pattern on the public Web. The service consumer uses configured client software to issue an invocation request to a service provided by the service provider which responds to the users request.

- **Request/Response via service registry**

Here the client request for service with the service provider. A response may be obtained to it and any necessary update to the client may also provided depending what services were registered with the service provider. Hence multiple asynchronous responses may be obtained for each service request.

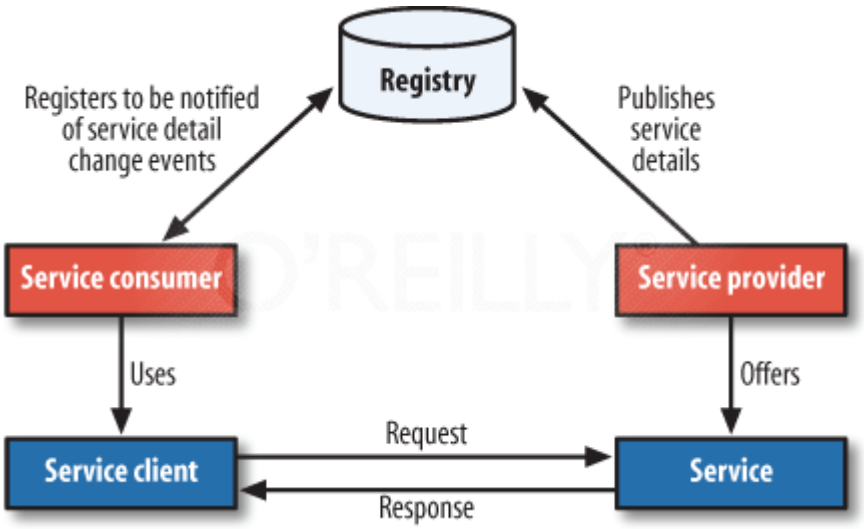



**Figure 20: SOA Request/Response pattern with a service registry**

- **Subscribe/Push**

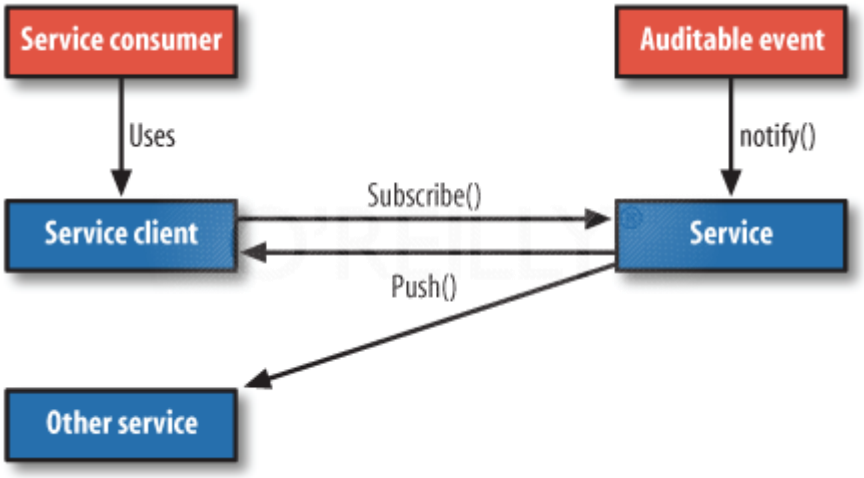



**Figure 21: SOA Subscribe/Push Pattern**

Another pattern for interaction is called Subscribe/Push. Multiple clients can register with the service agent called as subscribe. On any event changes what the service agent wishes to broadcast to its registered listeners it broadcast i.e. pushes the event notification to all the registered service clients or other services.

- **Probe and Match**

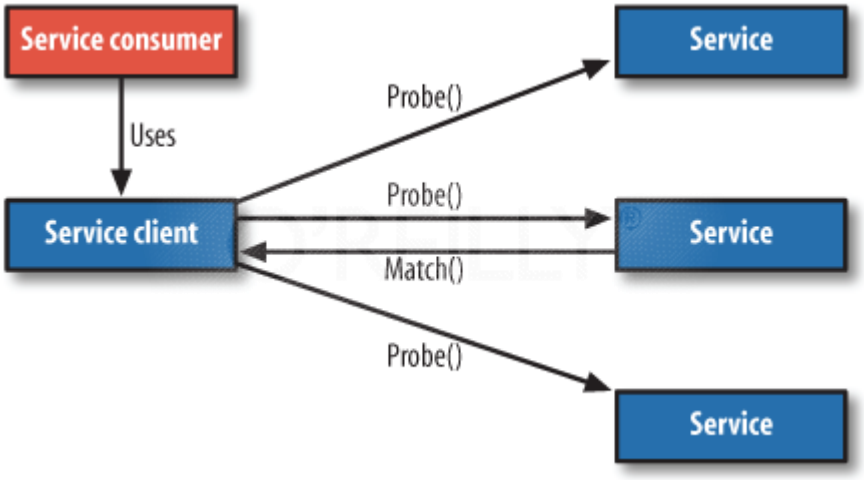



**Figure 22: SOA Probe and Match pattern**

This variation of the SOA message exchange pattern may also be used to discover services. A client may broadcast message to several endpoints on the network. Responses may be obtained from one or from multiple services. The amount of messages transferred eats up lot of bandwidth, using a registry may be a better option, because it does not require sending probe messages to all the endpoints instead registry can then discover the service and inform the client.

**Components of an SOA and its implementation**

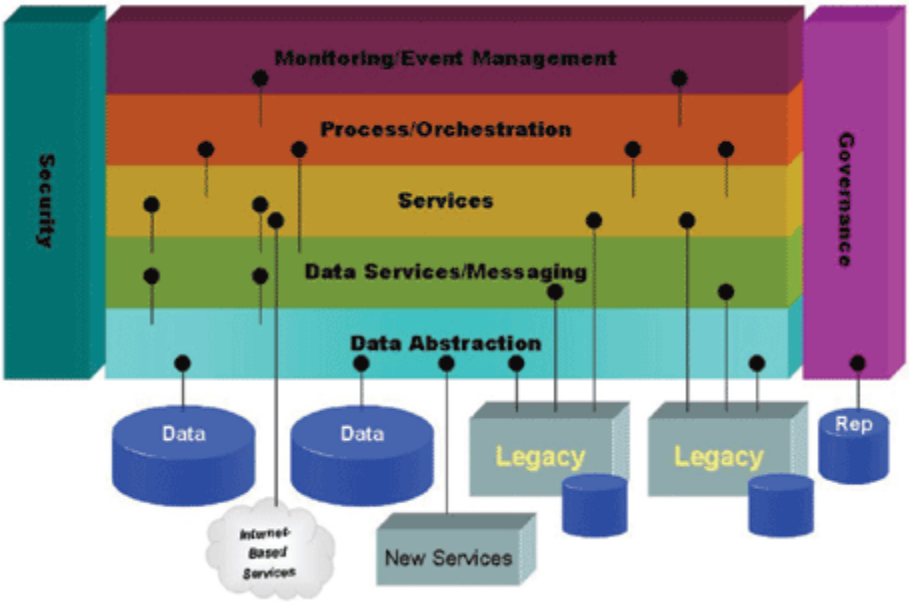



**Figure 23: Components of an SOA [21]**

In the previous section we have studies the monitoring and event management process. When implementing SOA, architects have to deal with several key issues as described in [15]:



- Protocols and standards
  Implementers need to make sure that a common set of protocols and standards is employed to ensure that users can communicate with all services without requiring custom client software for each

service. For example, if all of a provider's services use Simple Object Access Protocol (SOAP) over HTTP to communicate, it's much easier to wire together multiple services for a common purpose. If each service uses a different protocol and the service responses came back in differing data formats (e.g., one in XML, another in EDI, another in CSV, and another in plain ASCII), the work required to make the service bus operate might be disproportionately greater than the work involved in building a self-contained application from the ground up.

- Security
  Some services are open to anyone, but most have a limited number of acceptable users and roles, and service providers try to limit use to that list.

- Denial-of-service (DoS) and other attacks
  Implementers must work to minimize the impact of potential DoS attacks. One safeguard may be to require every service consumer to use a registry to get the endpoint coordinates for the service and to configure its service client appropriately before talking to the service. In the event of a DoS attack, this will allow the service provider to dynamically redirect legitimate traffic to a new service endpoint that is unknown to the attacker in order to avoid interruptions in service provision.

- Governance
  Service providers monitor service invocation requests during their life cycles to make sure they can scale the number of active service endpoints to meet demand in peak times. Additionally, service providers need to monitor the real-world effects of what their services are allowing consumers to do.

**Business Problem Resolved**

The architecture for the system described in Figure 17Figure 17 has been remodeled to apply the SOA pattern to make it service-oriented.

The human actor may simply use a form to log into a system or systems. Note that, unlike in earlier model, now a single authentication service is shared across all applications within the enterprise's platform. This is known as single sign-on (SSO). Likewise, there is only one data persistence component; this requires update and management of only one system, rather than several.

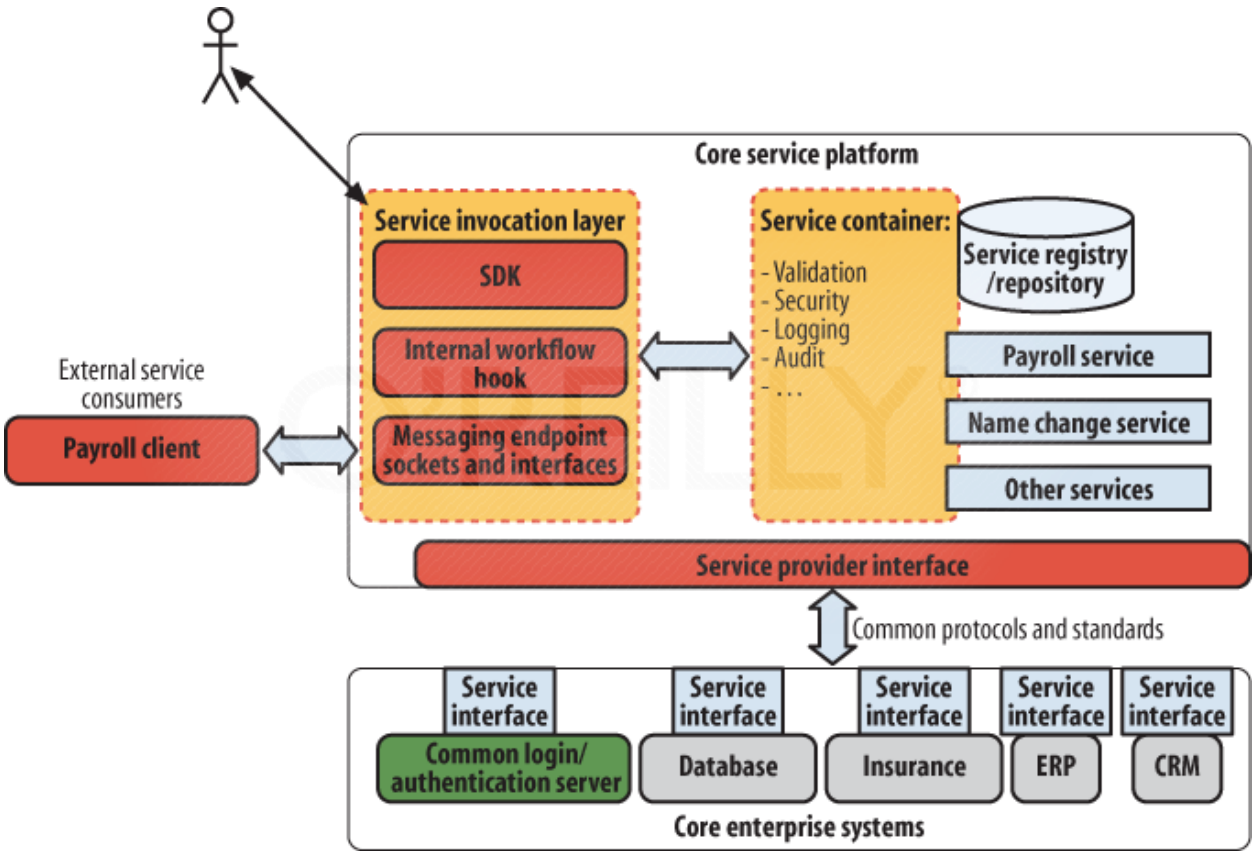



**Figure 24: The enterprise systems model, redefined using an SOA**

The enterprise can also outsource some functionality by allowing a third party to hook into its system via a messaging endpoint. Similar hooks could be added should the company decide to outsource other functions or act as a resource for other enterprises in the future.

The SOA infrastructure is very flexible and should be easy to add or drop functionalities. If an application has to be replaced, as long as the new application supports the existing service interface, there will be no interruptions to the rest of the system.

### 4.3.2 The Software as a Service (SaaS) Pattern

To know in detail of SaaS lets get acquainted with following terms:

- **Utility Computing and Cloud Computing**
  Cloud and Cloud Computing is a specialized pattern of virtualization and should not be confused with SaaS.

- **On-demand applications**
  Demanding for a service when required is called on-demand applications. A real time example is a on demand pdf creator, you can use http://createpdf.adobe.com service to create a PDF document online rather than having to download and install a version of Acrobat.

- **Software Above the Level of a Single Device**
  Applications are not bounded specifically for a device and should be capable to be used across all devices (not platforms). A real time example is using facebook on your mobile as well as your computers.

- **Model-View-Controller (MVC)**
  Some people consider SaaS a specialization of the MVC pattern that distributes the Model, View, and Controller (or parts thereof) over multiple resources located on the Internet. It is strongly related to SaaS, and most SaaS providers follow the MVC pattern when implementing SaaS.

**Business Problem**

To provide a real time example and how to build SaaS application and measure its performance we considered a "Hybrid Spam Email Filtering" model defined in [22]. The model described in [22] is personalized to each user as user can use integrated spam filtering software to classify their spam emails. Isn't it better if they are said you don't need to download anything still getting the spam filtering service and at a greater efficiency of eliminating spam's. Isn't that great?

Functional components of the core computing resources must be usable via a well-defined interface. Such an interface should not be bound to a single client or single model for delivery and should support multiple options for building the user interface.

Clients enjoy the full benefits of the software from remote locations. SaaS is a model of "functionality delivery" rather than "software distribution." Most of the functionality can be delivered over the Internet or made available in such a way that the end user can interact with the application to get that functionality without having to install the software on her machine.

**Deployment Pattern of SaaS**

The basic deployment pattern for SaaS involves deploying different aspects of the model, view, and control components of an application to multiple physical locations. The deployment approach may vary greatly depending on the software and its complexity and dependence on other aspects. The Figure 25 shows how the basic deployment pattern for SaaS differs from traditional software distribution.

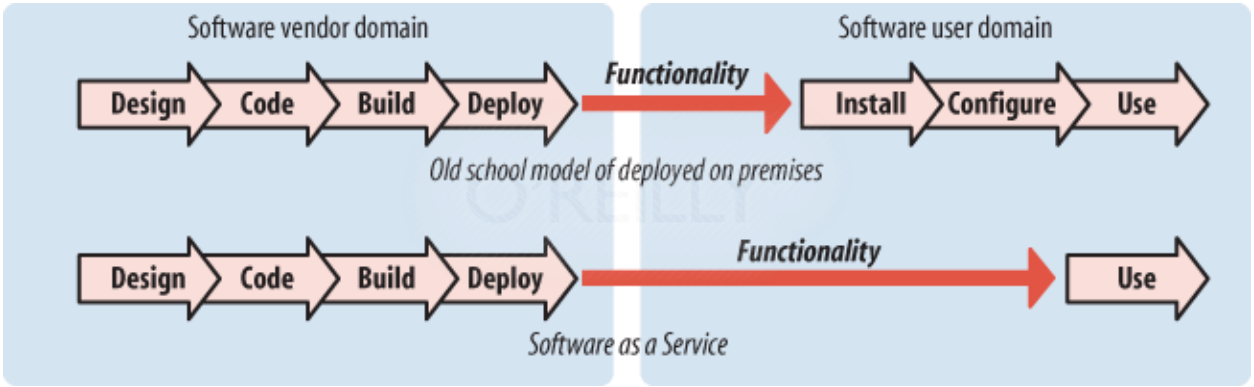



**Figure 25: Deployment pattern for SaaS versus conventional approach**



The main benefit centralizing the Spam Filter application is that it can learn from the results obtained from other users. If you receive an email from junk@spams.com and you determined that its spam then other users receiving it could also receive emails from such source as spams. Not only it can track illegitimate users but

can also effectively track different patterns of keywords which are to be avoided. For e.g. An email is considered to be legitimate by your spam filter with the subject line "\/i/\G/\R/\" even though you have blacklisted the word "Viagara". This is a simple example of collective intelligence. This application gets better and better as more and more people use it. This algorithm would be more sophisticated than the one described in [22]

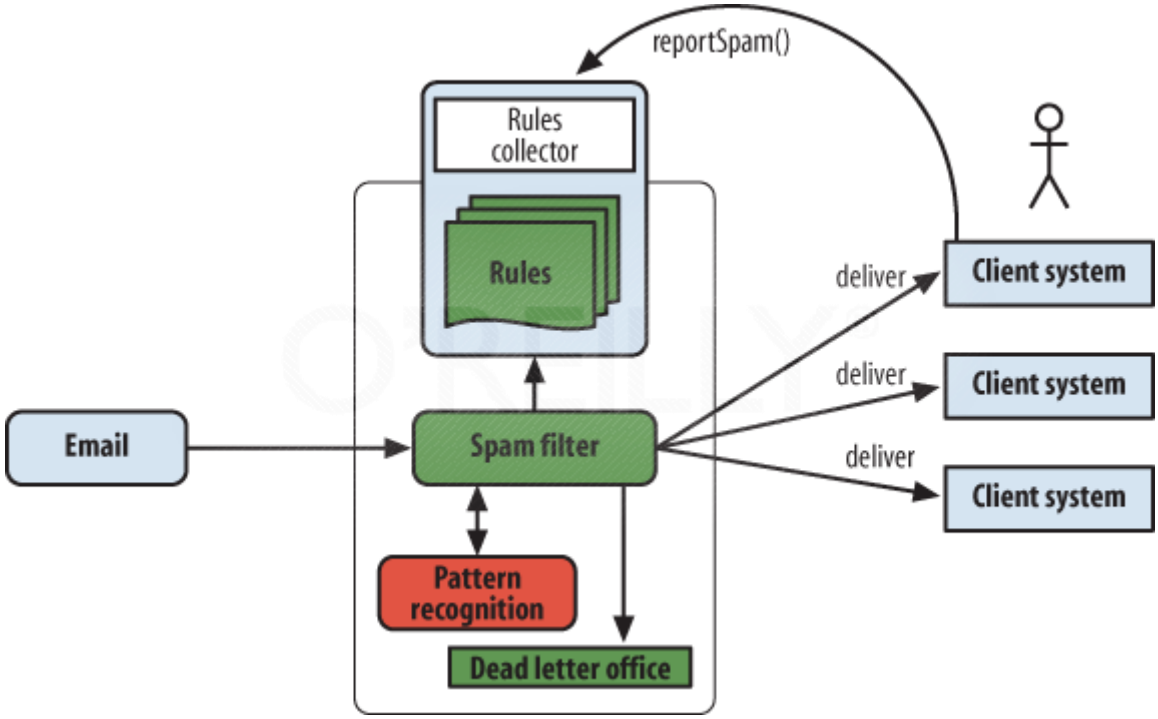



**Figure 26: Modeling HSF Spam filter described in [22] as software as a service**

**Specializations in SaaS**

SaaS may be specialized by using advanced computer algorithms to perform reasoning tasks such as inference. These ultra-advanced systems may one day be able to use cognitive skills to recognize and act on certain patterns of events. For example, these hybrid systems could use a combination of the Bayesian Theorem (conditional probability) and lexical heuristic filtering (finding evidence to support a hypothesis) to dynamically change their functionality. Such specializations will also benefit from adoption of the Collaborative Tagging (a.k.a. folksonomy) pattern, as it will help foster computational intelligence applications that can reason and infer hypotheses.

**Uses of SaaS**

As discussed earlier in this section a better spam filtering model could be proposed using SaaS. Apple's iTunes music application is perhaps one of the most prominent examples of a hybrid approach to the Software as a Service pattern. The iTunes application has some predetermined functionality and user interfaces (the "V" and "C" components of "Model-View-Controller"); however, much of the information presented to the user is based on information (the "M" in "MVC") the application receives from the iTunes servers during runtime. This hybrid approach can link user profiles to service calls to offer users better experiences.



Adobe Systems recently launched a service that allows people to manually create small numbers of PDF documents online and optionally link to them other functionality for things such as enabling rights management. Google continues to expand its SaaS offerings. Initially, its search service used algorithms to vary

search result rankings based on user interaction patterns. Recently, Google has added multiple other SaaS offerings, including creation and manipulation of online documents, spreadsheets, and more. Note that most of Gmail has always been provided as a service rather than as an application.

**Consequences**

The negative consequences of using the SaaS pattern are minimal, but they need to be addressed from the outset. In addition, authentication—especially when used to force compliance with software licensing—can be difficult to implement and to police.

The most crucial dependency of having connected to the internet while using SaaS, if the connection does not work, neither will the software. When such a mechanism is possible and makes sense, the best way to avoid such issues is to employ client-side caching. The appropriateness of this strategy varies by application. Caching Google Search is difficult, and would be mostly useless in an environment where readers couldn't go to the linked articles anyway. On the other hand, technologies like the Adobe Integrated Runtime (AIR) and Google Gears are steps in the right direction. Gmail desktop provides a model to cache emails at the client where you can surf your emails, and even send without internet connection. When an internet connection is senses all queued emails are pushed on the network.

Denial-of-service attacks can also be a threat. A malicious user may be able to overpower the bandwidth or computational capabilities of software implemented as a service and effectively deny or greatly slow down other users' experiences. Because SaaS applications are not hosted or controlled by the user, the user might be subjected to occasional outages when service upgrades or other events take place on the provider side. Also, personal security risks are inherent in passing information back and forth on the open Internet. Users should carefully assess what risks are acceptable for their purposes.

### 4.3.3 The Participation-Collaboration Pattern

The Participation-Collaboration pattern is related to two other patterns:

- Harnessing Collective Intelligence
  This pattern, discusses aspects of the Participation-Collaboration pattern where inputs from many users over time are combined into composite works.

- Innovation in Assembly
  This pattern, is focused on the ability to build platforms where remixing of data and services creates new opportunities and markets.

Until recently, the easiest way to compose and distribute textual content was to have a small group of authors write some material, print it on paper, bind that paper, and sell it or give it away or creating a soft copy of which could be distributed online. Once something is published, if the material becomes obsolete or errors are found, there is no quick or inexpensive way to change it. The high costs of either new print or requiring the site contents to be updated by the owner were expensive.

**Context**

Allowing not just the author of the document to contribute towards publishing but also involving regular participants to add their ideas removes the constraints as discussed above. The participation is possible when a group of people leading on a common interest is willing to share some information. This is always better rather than only few present their knowledge.

A most common real life example of this pattern is Wikipedia. I am well 100% sure those reading this document are aware of what Wikipedia is and what does it facilitates to the users. But were you aware that it is based on participation and collaboration pattern of Wed 2.0?

There will always be issues to decide if the contents are valid or not. A malicious user can come in and write something incrorrect leading to loss of the actual contents. Hence version control over content is also a common requirement for those implementing this pattern. Keeping track of multiple versions of shared edits is very crucial and helps participants to edit anything they like but at the consent of the owner he can rollover the edits.

**Static Structure**

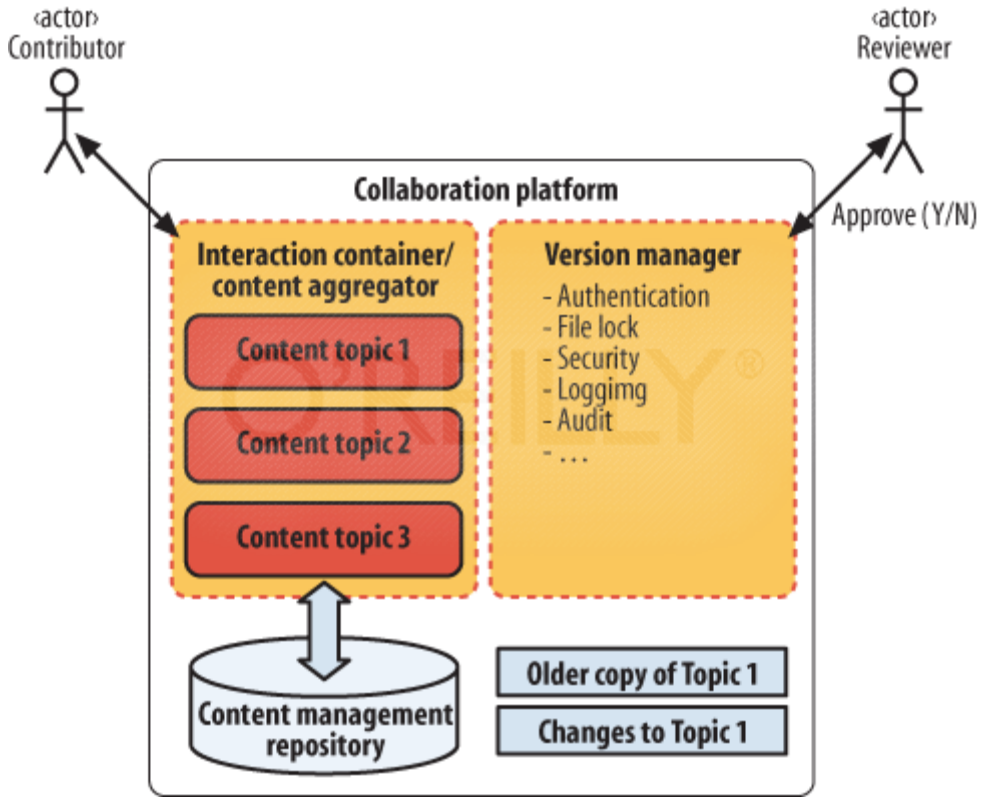



**Figure 27: The Participation-Collaboration pattern**

A contributor is allowed to update, edit or add the contents to the already existing page on the collaborated platform. It is believed that the user corrects the contents available for others benefit but even if they don't there is a mechanism to roll back the page. The user when updates an event is generated which informs the volunteers the updates, on which they can either allow to update the contents or when directly updated they can roll back the contents from the repository. The Figure 28 shows a self explanatory UML sequence diagram explaining the process in participation collaboration model.

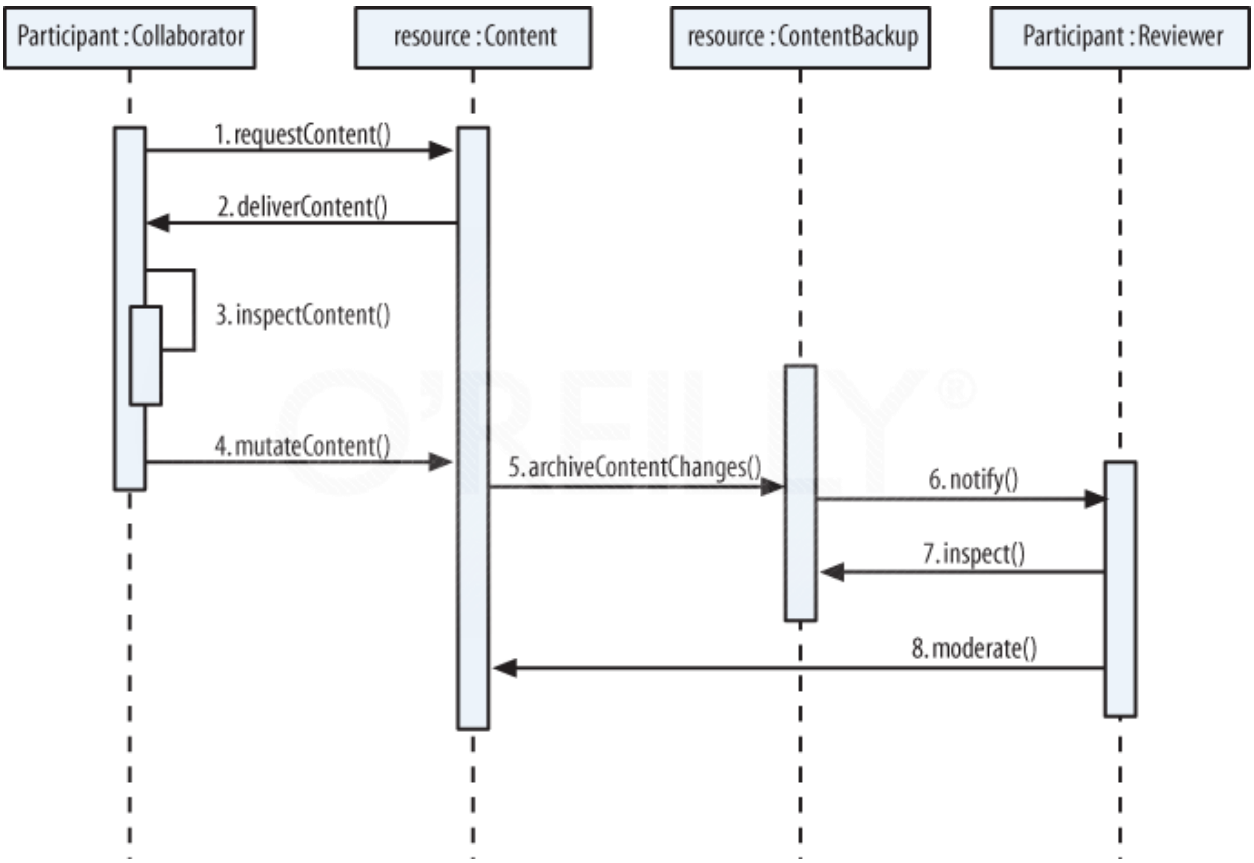



**Figure 28: The sequence of the Participation-Collaboration pattern**

These are not limited to text contents but to media format whether it be audio, video or any customized application. A recent article on Web 2.0 Car by Local Motors by Jay Rogers pioneers a model where first car based on Web 2.0 was developed, applying thoughts and ideas. As stated in [16] 60,000 car models have been collected yet contributed from the young minds on Internet community.

One criticism of Wikipedia is that a select few editors end up in control of the collective knowledge that was originally contributed by many diverse contributors. Some Wikipedia pages have disappeared completely, much to the dismay of subjects of those pages, and some content has been ruled unworthy or improper. Trusting one or two people to control the content related to a specific topic may lead to contributor frustration. Proponents of this argument claim that mechanisms like Wikipedia represent a clear danger to society: if considered authoritative and beyond reproach, they could effectively be used to rewrite portions of history, at least for people who don't look beyond these sources [15].

There are many specializations of the Participation-Collaboration pattern, including blogs (web logs), blikis (blogs with wiki support), blooks (books that are collaboratively written), moblogs (mobile blogs), and vblogs (video blogs), among others. Each uses the same basic pattern, where participation and collaboration in a shared work is a core theme. The same pattern is also used for open source software development, where many programmers contribute code to evolving projects.

**Known Uses**

Wikis—including many specialized wikis such as Wikipedia is probably the best-known use cases of this pattern. SourceForge, Apache, and other open source software processes use the same pattern for code rather than text.

Some cool new websites are applying this pattern in a different manner. Mix2r.com, Gluejam, and MixMatchMusic—have collaborative sites where music is open sourced and new tracks can be mixed with existing ones to create new audio projects. Artists and musicians can contribute loops and original audio tracks to works that others can download.

In the video world, similar companies are pursuing this pattern. MixerCast.com, MovieSet.com, Brightcove.com, and others have used the same approach with video, letting people remix video clips and even provide audio clips to create new works. MovieSet.com also allows users to view the behind-the-scenes aspects of content creation, and in the future it may allow the audience to provide input regarding the plot.

### 4.3.4 The Asynchronous Particle Update Pattern

The Asynchronous Particle Update patter is related to two well-known architectural styles:

- Asynchronous JavaScript and XML (AJAX)
  AJAX is primarily used in many Web 2.0 applications, which allows a more dynamic and interactive web page.

- Representational State Transfer (REST)
  REST is software architecture paradigm for distributed hypermedia systems. The term Representational State Transfer was introduced and defined by Roy Fielding in 2000 [9]. Conforming to the REST constraints is referred to as being 'RESTful'.
  This architecture style consists of clients and servers. Just like the traditional client server approach Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource. The term is often used in a looser sense to describe any simple interface that transmits domain-specific data over HTTP without an additional messaging layer such as SOAP or session tracking via HTTP cookies. In true pattern style, it is possible to design any interchange in accordance with REST without using HTTP or the Internet. It is likewise possible to design simple XML and HTTP interfaces that do not conform to REST principles.

The Asynchronous Particle Update pattern is likely to be useful in any situation in which exchanging a small number of bytes rather than an entire document will save both server and client (and owner) resources. During the first iteration of the Internet, most clients retrieved content using HTTP GET requests, which returned complete HTML web pages. In true REST style, the web pages were built with information that represented snapshots of the state of information at a certain point in time.

Much information was needed to be dynamically updated without the user consent to click on reload. For example a website displaying you the live score of a cricket match. If you opened a page you don't certainly want yourself to manually update it each time to check out the live score. Asynchronous particle update pattern with the help of AJAX or RESTful pattern would definitely help you refresh your display each time a score is updated. We will look it in more details later in this section.

**Derived Requirements**

With the help of asynchronous request the part of the document object model (DOM) should be able to update its contents without requiring updating the entire document. This requires the browser to have additional form of processing capability to update in such fashion.

Being able to trigger updates on events such as user interactions, server events, state changes, or timeout events will provide maximum flexibility for this pattern to be applied over a variety of contexts.

Consistent model for objects (including document objects), events, event listeners, and event dispatchers across multiple tiers of any solution pattern is required by the architecture. Building an application that adheres to the MVC pattern is also likely to be an advantage for developers and architects who can manipulate both the server and the client frameworks and code.

**Solution with different possible static structure and dynamic behavior**

Several clients should be able to subscribe to event or state changes, perhaps with one change federating out to several clients. The following structure illustrates several ways how this could be achieved.

### 4.3.5 Asynchronous Particle Update pattern

The Figure 29 depicts an asynchronous request/response message exchange based on a user event trigger. In this case, the user clicks the mouse over a button labeled “Update stock quote” a small request() message is dispatched to the server. The server retrieves the stock quote from the stock quote data provider and returns it to the client side via the response. The service client uses the information to update the browser view.

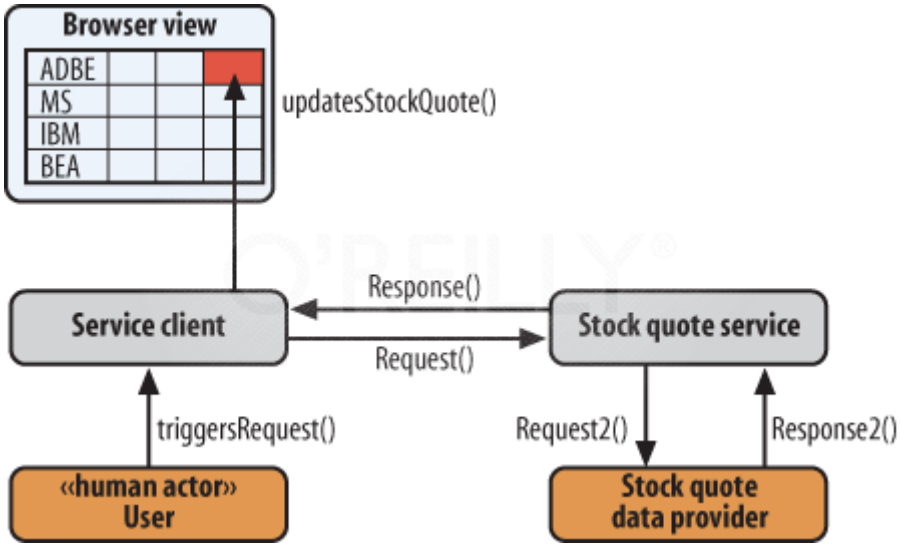



**Figure 29: Asynchronous Particle Update pattern**

- **Asynchronous Particle Update pattern, based on an elapsed time event on the client**
  Asynchronous Particle Update pattern, based on an elapsed time event on the client operates the same way but triggered after every elapsed time as specified with the timeElapsed(), which automatically updates the stock quote at given intervals. In this case, the timeout(90) event is triggered from the client to update the node of the DOM every 90 seconds.

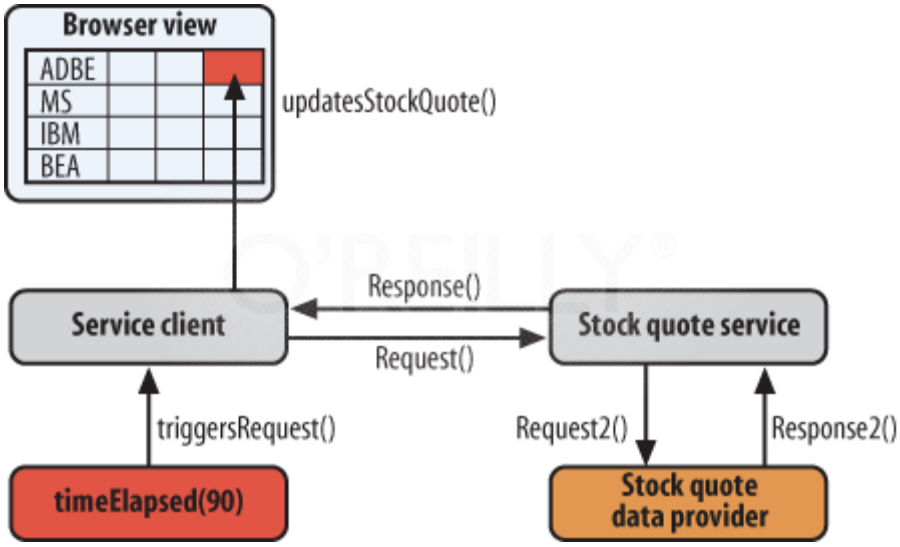



**Figure 30: Asynchronous Particle Update pattern, based on an elapsed time event on the client**

- **Asynchronous Particle Update pattern, based on an elapsed time event on the server**
  Another variation exists in which the service decides when the contents needs to be sent to the client. This is better than the earlier model as multiple requests to be transferred from client are eliminated. This is something not presently supported by AJAX today. It is based on the assumption that client registers only once and the updates the sent at every elapsed time as decided by the service running at the server. The server-side request then triggers the messages to be pushed to each client that has registered to receive messages of that type or for that event.

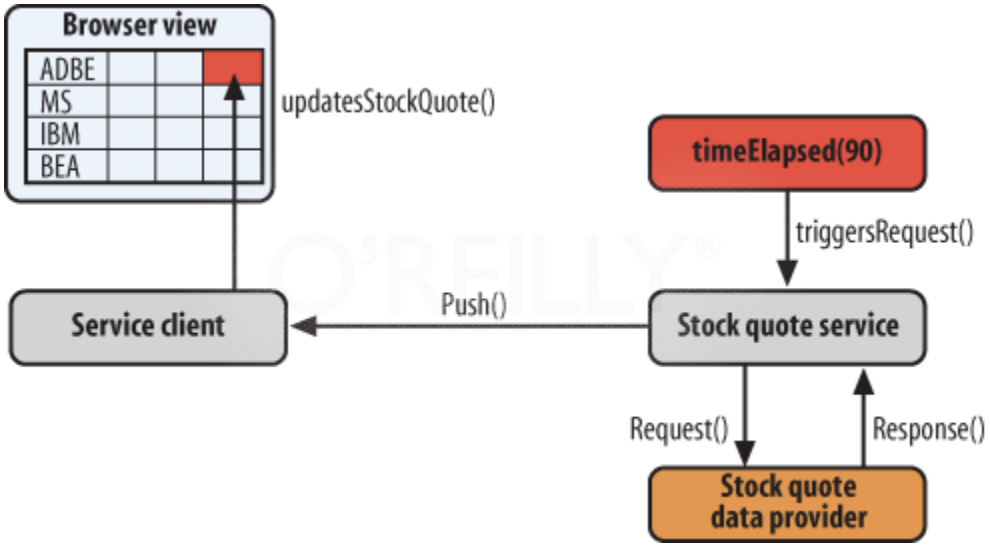



**Figure 31: Asynchronous Particle Update pattern, based on an elapsed time event on the server**

- **Asynchronous Particle Update pattern, based on a state change event on the server side**
  It is very similar in architecture to the previous model seen but the trigger point is not static instead is based on the state change event. This would save a lot of network bandwidth as client sends request message once, server registers it and updates all registered client not periodically but only in a case where new information is to be declared. Figure 32 illustrates a scenario where clients are updated only when the stock value changes.

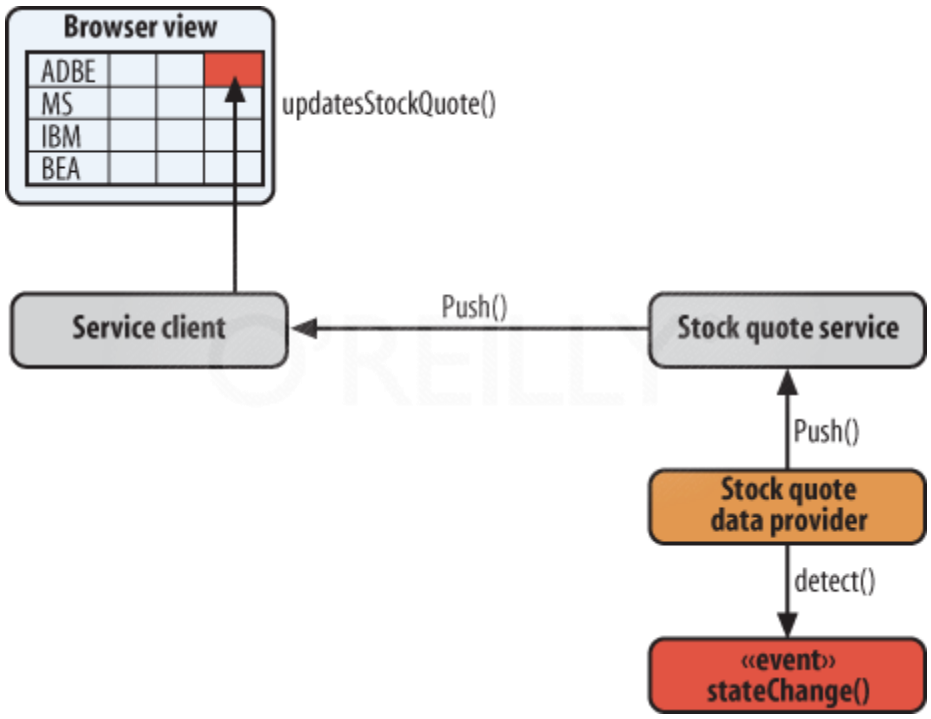



**Figure 32: Asynchronous Particle Update pattern, based on a state change event on the server side**

**Implementation**

Factors that need to be considered while deciding to develop this pattern are as follows:

- Number of requests
- Size of the particle
- Frequency of updates
- How to present changes to users.

The overall goal here is to provide the best possible quality of service while using the least possible amount of network bandwidth and generating minimal processing overhead.

**Known Uses**

The AJAX manner of implementing this pattern is widely known and used in the software industry today. Although the specifics regarding how it can be implemented are largely left to each developer, the technologies used are very particular.

Adobe's Flex and AIR, for example, implement the pattern via different remote messaging methodologies, standards, and protocols. The Action Message Format (AMF) transmission protocol may be used within a Flex application. In addition, Adobe LiveCycle data services can be used to push data to multiple subscribers.

REST is also a style of software architecture that is frequently used to support implementations of this pattern (REST does not mean XML over HTTP). Although not specifically dependent upon HTTP and XML, it is a frequently quoted implementation of this pattern.



### 4.3.6 The Mashup Pattern

I would like to start explaining you this by thinking of a real world example what most of you have used. Sites like 511.org which most of you use today to know the public transport available on the route. Have you ever

thought how it can intelligently obtain the schedule of all the transport modes. What if the schedule changes are they notified for the same. Do they build all maps to show you? What if there is a new route defined today, how will they get a map for the new route.

Let's look into this! This is not a small problem when dealt entirely on your own. A pattern like Mashup will help you integrate services from different sites and help you define your own contents as required. The solution to the above problem is quite simple if dealt this way. 511.org calls the services built by transport authority like VTA, Caltrain to have their routes and schedules. As required it can parse those as per the users query and supply those for a View generator to show various itinerary to the user and at the same time a request to the Google map service is made to have a user with a map for the user selected route. Here your site is not just what 511.org provides you but a mashup of other services like from vta.org, caltrain.com, maps.google.com, etc.

You are given with multiple routes and selecting each route obtains you a different map for the same. Selecting any new route just updates the map while the whole page remains same. This is performed via the asynchronous particle update pattern as seen earlier in this section.

Mashups can be thought of multiple sub-windows inside one window which is displayed, where each sub-window constitutes for a specific data. These data are delivered via services over a network connection or other communication formats and protocols.

**Requirements for Mashups**

The Mashup pattern depends on the SOA pattern (as seen earlier in this section); services that can be consumed must exist. Data encoding should generally be in a format that does not depend on any specific operating system, browser, or other hardware or platform. Languages such as XML are commonly used to encode data for mashups, as these languages are easy to interpret and manipulate.

On the client side, there should usually be some way for the user to manipulate the mashup content. However, this is not always necessary. Where major computational resources are required to manipulate data, a third tier may be introduced to help reduce client-side processing.

**Solution for Mashups**

An application is built to load a map and template onto a user's system upon request. The application then detects remote data sources that are online and presents the user with a control button object to allow the user to render the data sets over the top of the map application. The data is retrieved using web-based protocols and standards to connect to a set of services that supply the data.

The data sets may alternatively be cached locally on the client to allow more efficient launching and operation of the client-side application.

**Static Structure**

A mashup uses one or more services and mashes together aspects of those services. The Figure 33 demonstrates a simple model where a single client runtime is fed with data from multiple disparate services.

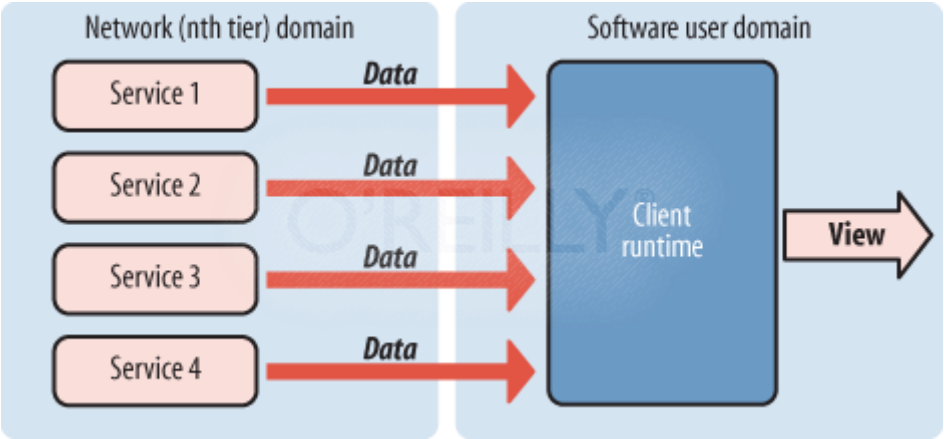



**Figure 33: The basic Mashup pattern**

A specialization of the Mashup pattern involves letting users configure the mashup's view component. Another way to think of this is just to recollect the time when you had dragged and dropped certain windows in your browser to have it a personalize view in your iGoogle or your profile in linkedIn or if you have used a content management system (CMS) you could edit the way users see the contents. A simplified UML diagram shown below helps you determine the relationship between service, mashup and view.

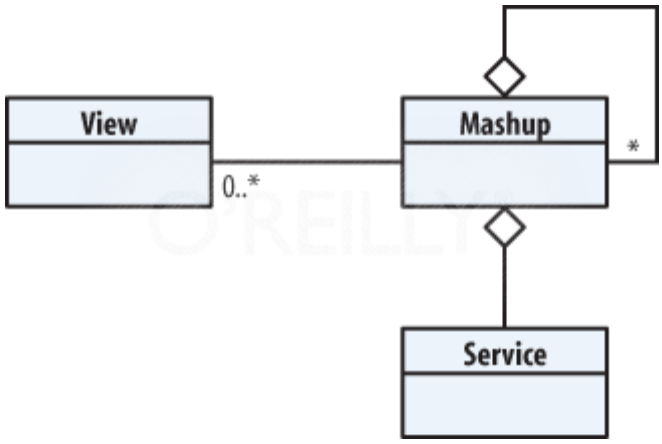



**Figure 34: A UML class view diagram of the Mashup pattern**

Another way to digest this fact is to have a look at the below figure and try to map yourself till what you read in this report yet.

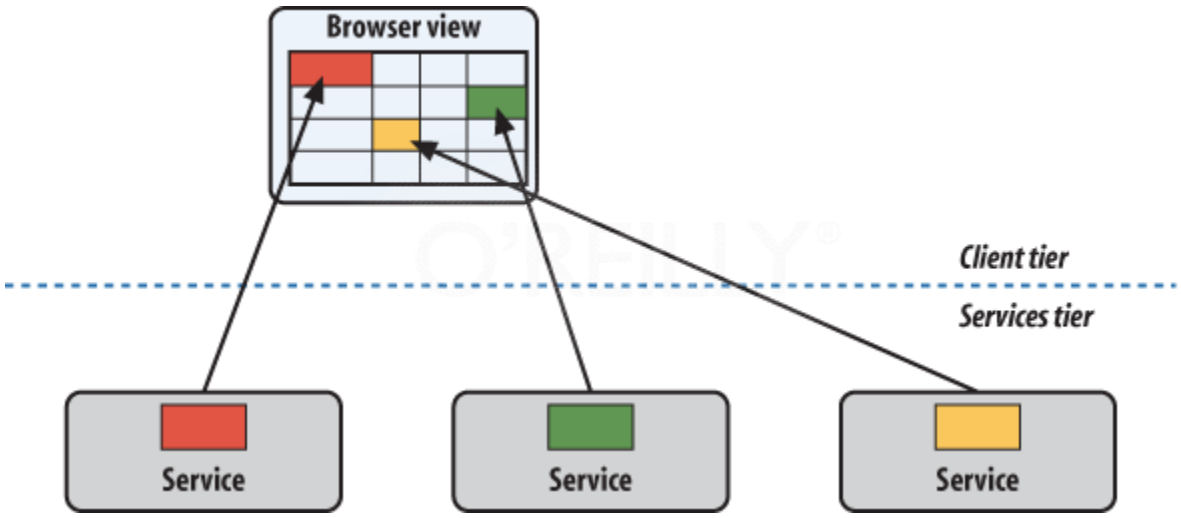



**Figure 35: A simple two-tier mashup pattern**

Let me explain you what happens in the Figure 35 though it is self explanatory. Each portion of the view in the client tier are driven by a disparate service running at the service tier. Another model for a mashup is a multitier fashion explained with a self explanatory figure shown below.

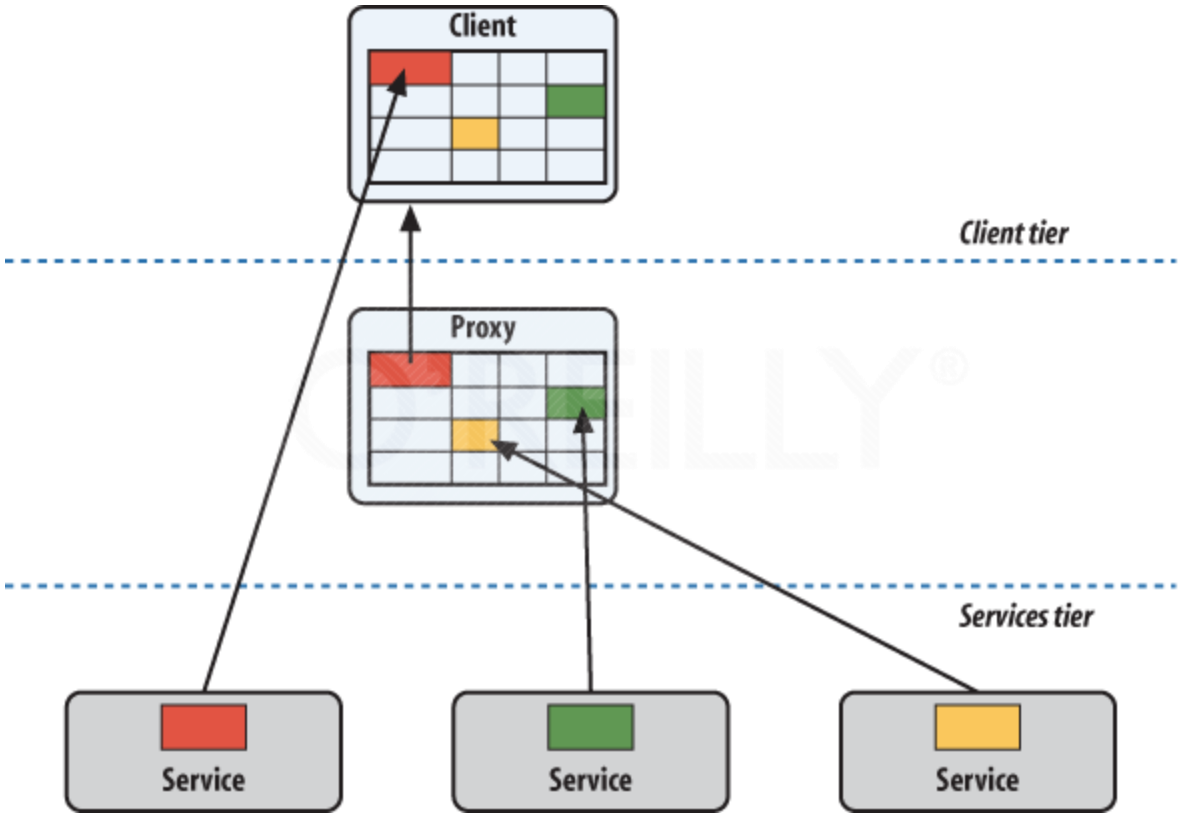



**Figure 36: A hybrid multitier mashup**



Contents are mashed up in a proxy middle tier or a server before it is delivered to the client. The client is completely transparent of any mashups he obtained from the entity as a whole. The proxy helps achieve such transparency where client's not supporting mashup can utilize mashup via proxy with no additional task.

**Implementation**

Developers need to consider many standards and protocols which constitute to build a complete application. Developers of services that want to provide mashable data also should strongly consider using common data formats such as XML, as well as the client-side overhead of processing various standards and protocols.

Browsers don't, however, provide much support for SOAP with extensions such as the OASIS WS-SX and WS-RX standards. It is often believed that the same mashup are available on resource limited hardware machines and the processing overhead needed to comply with the SOAP model which might eliminate the possibility of services using this model. In these cases, the Representational State Transfer architectural style, implemented by using XML over HTTP, may be a natural fit for many mashups. RESTafarians (the name given to those who enthusiastically evangelize the benefits of REST) [15] promote the view that dividing application state and functionality into resources and giving them unique identifiers makes content easier to mash up.

On the client side, architects and developers will have to choose their target technologies carefully. In many implementations, a consistent rendering experience or manner in which data can be shown to end users will be desirable. It is desired to have your contents displayed correctly in variety of browsers such as Internet Explorer, Opera, Google Chrome, Firefox, and Safari and their versions hence a better testing can have a great impact on the quality of your product. To add to the complexity, all of these browsers may have to be tested on multiple platforms or operating systems.

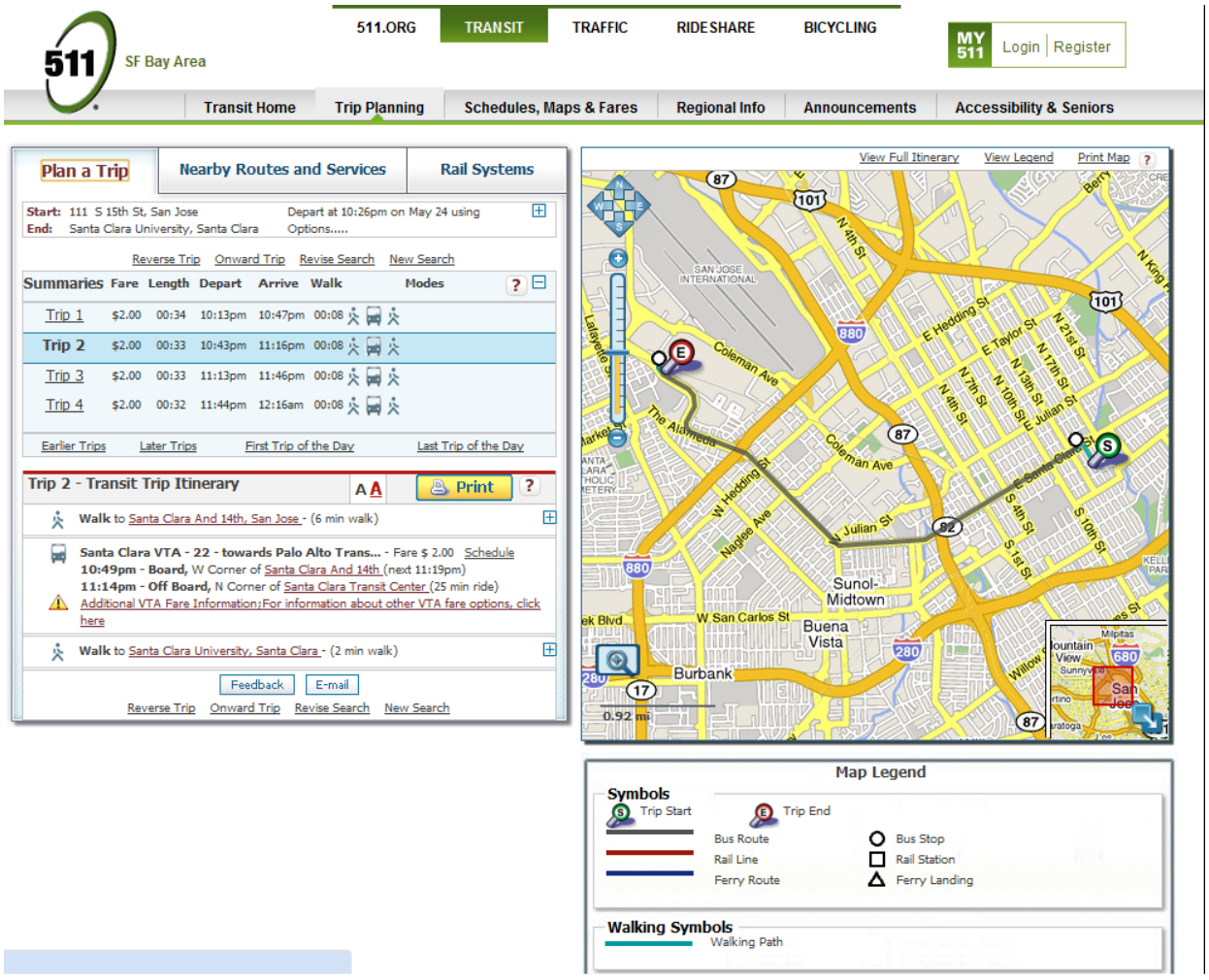



**Figure 37. An excellent example of a mashedup webpage: 511.org**

**Limitations**

Mashups entirely depends on services and the failure by the service to feed the contents will fail some contents in the mashup to be loaded. The services are externally provided and have no control on the mashup developer. Content used in mashups may carry with it certain copyright privileges or restrictions. Material licensed through Creative Commons under a license allowing free distribution is a perfect fit for mashups.

Just to recollect what we have seen so far. We have studies few patterns like SOA, SaaS, asynchronous particle update and mashup pattern. As said there are unlimited number of patterns that could be used. I will just provide an overview of few more patterns below:

### 4.3.7 Rich User Experience Pattern

A Rich User Experience Pattern (RUE) conforms for a high level of formatting and interaction; is often seen to work much better than the applications with naïve layout. I don't want to argue on this but just would like to provide a fact from yourself, how many time do you spend on facebook even when you don't want to?

### 4.3.8 Synchronized Update Pattern

Multiple clients may be synchronized to a single object's state or multiple states. Although this pattern may appear to be enterprise-oriented, it is equally applicable to individual servers or even peer-to-peer interactions.

Figure 38 and Figure 39 demonstrates how objects are synchronized. The clients subscribe for files to be synchronized. The object when modified, the state change updates are sent to all other clients who have subscribed to synchronize the object's contents. This is depicted in Figure 39.

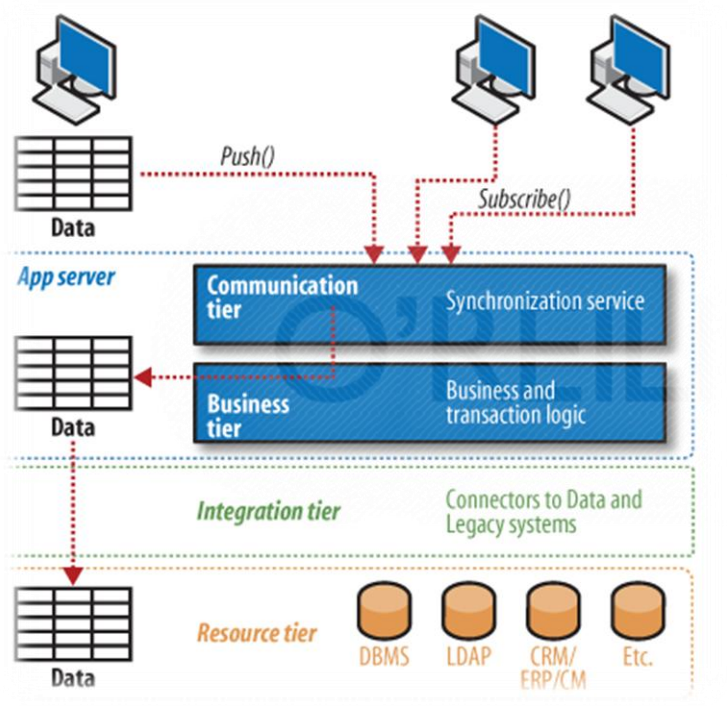



**Figure 38: Synchronized web clients registering to an object's state**

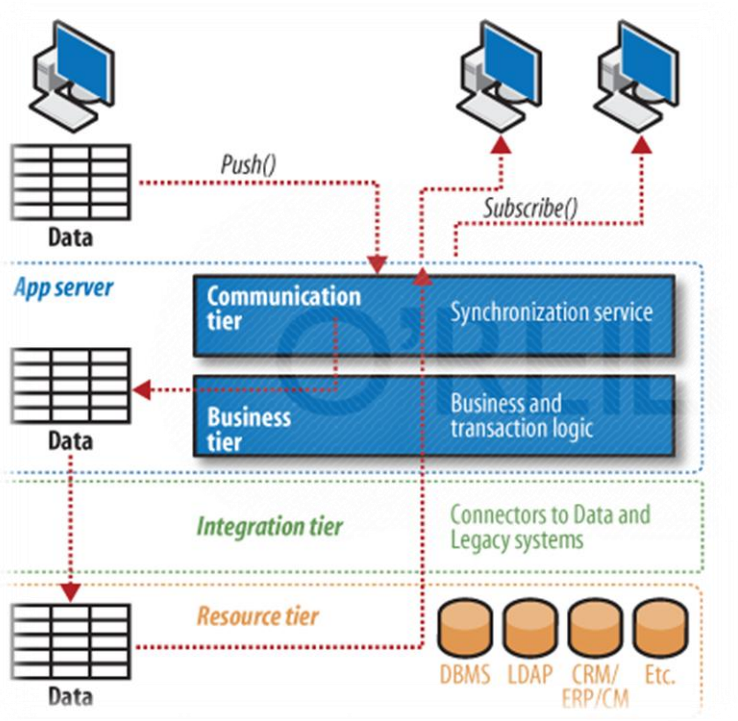



**Figure 39: Multiple interaction services being synchronized via one application**

A perfect example of this is a dropbox service which allows synchronization of files from your computer to various other machines.

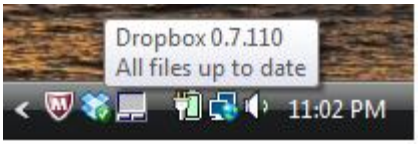



**Figure 40: Dropbox synchronization service verifying all files that are synched.**

There are several other patterns like collaborative tagging pattern for an application like folksonomy, declarative living and tag gardening pattern used for rating something, Persistent rights management, etc. For more details you can refer [15].

# PART III

# Chapter 5: Mobile Web 2.0

Till now we have read about Web 2.0 applications to run on your desktop machines or your laptop. We now take the web one step forward to the mobile devices and name it the Mobile Web 2.0. By Mobile Web 2.0 we mean that the users can now access internet or web applications through their browsers installed on their mobile device such as a smart phone. It is very important for the device to be connected to a wireless network. The challenge faced by mobile Web 2.0 is that it suffers from interoperability and usability platforms. We will consider each of these limitations in later sections.

Providing high speed data on mobile devices is a big milestone for the internet and content providers because the user is constantly on move from one network to other. Hence, it is very important that the mobile web uses lightweight pages that load faster. These pages are written in XHTML (Extensible Hypertext Markup Language) or WML (Wireless Markup Language) to deliver content to mobile devices.

## 5.1 Development

Let us first have a look at the evolution of mobile web standards dating from dating from 1990 to 2007

The evolution of mobile web standards [23] is shown in Figure 41.

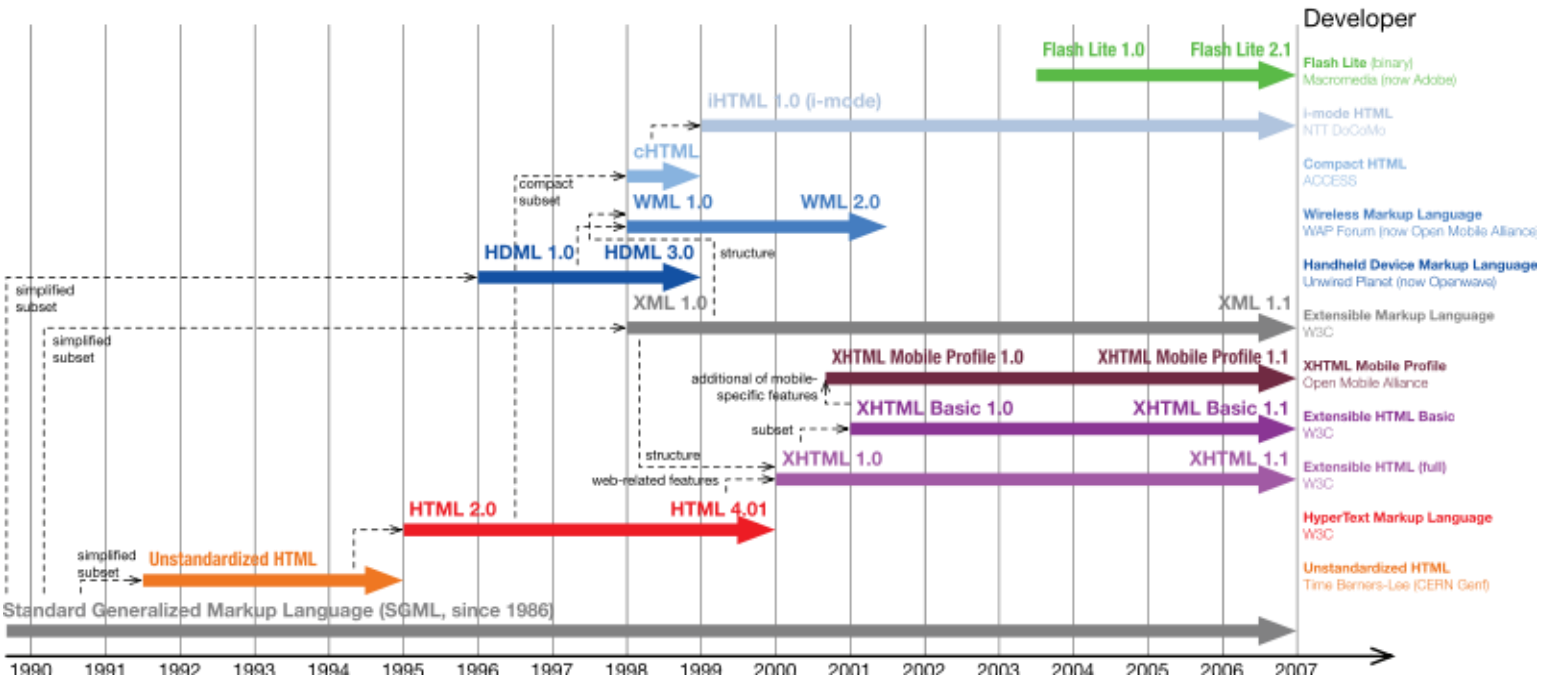



**Figure 41: Evolution of Mobile Web Standards**

The first access to the mobile web was commercially offered in Finland in 1996 on the Nokia 9000 Communicator phone via the Sonera and Radiolinja networks. This was access to the real internet. The first commercial launch of a mobile-specific browser-based web service was in 1999 in Japan when i-mode was launched by NTT DoCoMo [23].

As the technology advanced more efficient and powerful Web 2.0 technologies evolved such as the XMLHttpRequest and Adobe's Flash plug in.

## 5.2 HTML 5:

Most of the web applications are based on HTML and JavaScript hence, in 2006 a group of web developers from Apple, Mozilla and Opera began the Web Hypertext Application Technology Working Group (WHATWG) came up with a new version of HTML called as the HTML 5. HTML 5 is intended to provide additional support for Web applications. HTML 5 is a large specification and it introduces a new document structure, too many new elements and attributes to list, several APIs to new features explicitly intended for richer web applications [24]. HTML also provides new document structure elements as shown in Figure 42.

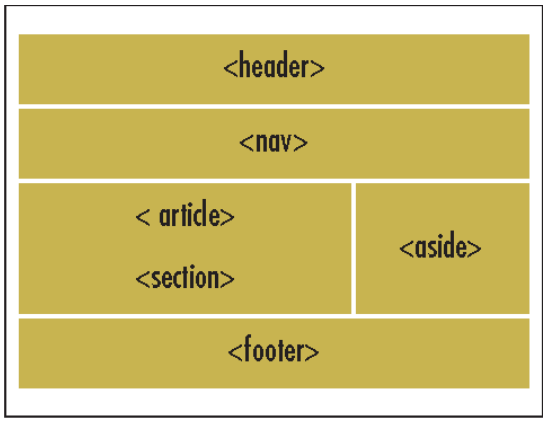



**Figure 42: HTML 5 provides new elements to help make document structure explicit**

HTML 5 also provides Web Forms 2.0 and a new networking service via the WebSocket interface. HTML 5 includes explicit support for offline execution of Web applications. The most important features are the application cache and the application cache manifest. The manifest is (among other things) the list of everything the server needs to send to the user-agent cache so that, at some time in the future, the application can be launched using the cache's content instead of contacting the originating server [24].

## 5.3 Evolution Path towards Mobile Web 2.0

In this section we will see how Mobile Web 2.0 evolved from two different paths. Path 1 is through Mobile Web 1.0: the mobile internet needs to address Web 1.0 issues before joining Web 2.0 dynamism and Path 2 is through Web 2.0: the mobile internet needs to address Web 2.0 issues to boost its full coverage before Mobile Web 1.0 is established. These paths are shown in Figure 43. The path 1 is denoted by MW1-path and path 2 is denoted by W2-path.

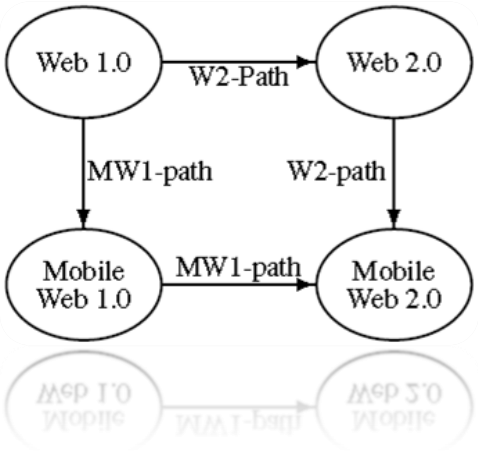



**Figure 43: Two different transitions to mobile Web 2.0**

Some of the application difference in Mobile Web 2.0 and Web 2.0 is shown in Table 4.

| Mobile Web 2.0 | Web 2.0 |
|---|---|
| mobile Google | Google AdSense |
| (mobilized version) | Flickr |
| not yet | BitTorrent |
| not yet | Napster |
| mobile Wikis | Wikipedia |
| mobile blogging | blogging |
| (mobilized version) | upcoming.org, EVDB |
| mobile search engine | search engine |
| -speculation | -optimization |
| page per click | cost per click |
| mobile web services | web services |
| m-participation | participation |
| mobile wikis | wikis |
| m-tag | Tag("folksonomy") |
| m-syndication | syndication |

**Table 4: Comparison of Mobile Web 2.0 and Web 2.0**

The difference between Web 1.0 and Mobile Web 1.0 is shown in Table 5.

| Web 1.0 | Mobile Web 1.0 |
|---|---|
| DoubleClick | QR-code [3] |
| Ofoto | MobShare [6] |
| Akamai | not yet |
| mp3.com | not yet |
| BritannicaOnline | not yet |
| personal websites | (partially) |
| evite | (notification only) |
| domain name | not yet |
| -speculation | |
| page views | subscription |
| screen scraping | not yet |
| publishing | publishing(partially) |
| content management systems | specialized one |
| directories (taxonomy) | carrier portal |
| stickiness | email |

**Table 5: Comparison of Web 1.0 and Mobile Web 1.0**

## 5.4 Context - Aware Mobile Web 2.0 Service Architecture

In this section a context-aware mobile Web 2.0 architecture is proposed [25] that connects user context and community information with web services. For the delivery of this information from a mobile device to web services a Mobile middleware is needed. Four different models for this delivery of information are explained. "A context is any information that can be used to characterize the situation" – Dey. An example where context is useful is an instant messaging service where user can decide whether or not to initiate a conversation based on the availability in the contact list.

Context-aware mobile Web 2.0 service architecture:

### 5.4.1 Elements of the service architecture

As stated earlier a mobile middleware component is required to collect information related to user context, community and authentication. The operation should be secure and security is controlled by a trusted entity i.e. a specific control service. The elements of service architecture are shown in Figure 44:

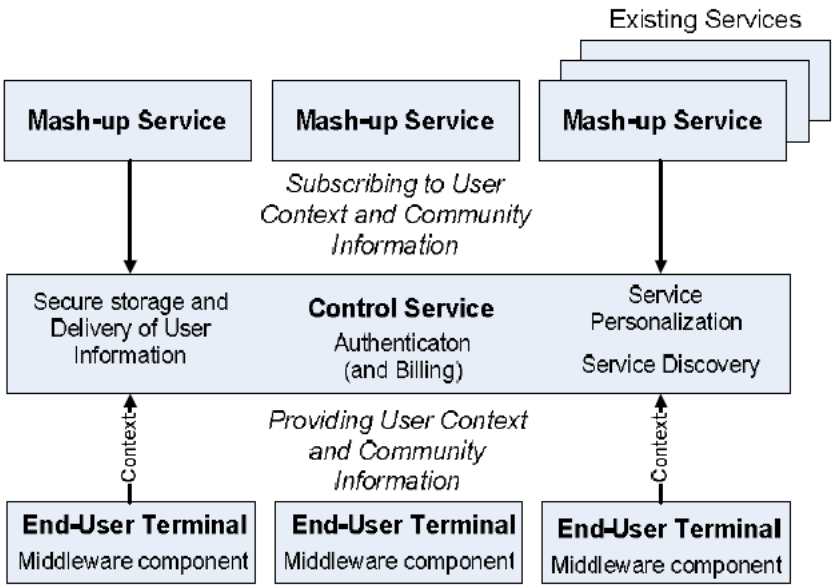



**Figure 44: Elements of the service architecture**

It is clear from the figure that the web services can utilize the features from other web services forming mash-ups. These web services are subscribed to user context and community information via the control service. Mashup services are actually services that are visible to the end-user. Every user and a mashup service have a unique identifier that are used to link the service sessions specific context and community information. The control service is used for service discovery and acts as an entry point to available mashup services. The control service is also responsible for providing tools for the users to define the mashup services and other users that are allowed to access the context and community information. The mobile middleware component prevents delivery of certain type of information. The information delivery can also be adjusted according to local context.



### 5.4.2 Mobile middleware

Figure 45 shows a middleware design and it is clear that every end-user's mobile terminal runs the middleware component that controls P2P access and the context information delivery. The middleware could

be implemented as part of the browser or as a separate process. Probably the latter choice is the easiest to implement, but, if the middleware's functionality is tightly coupled with the browser's functionality, it might be necessary to tie the two together [25].

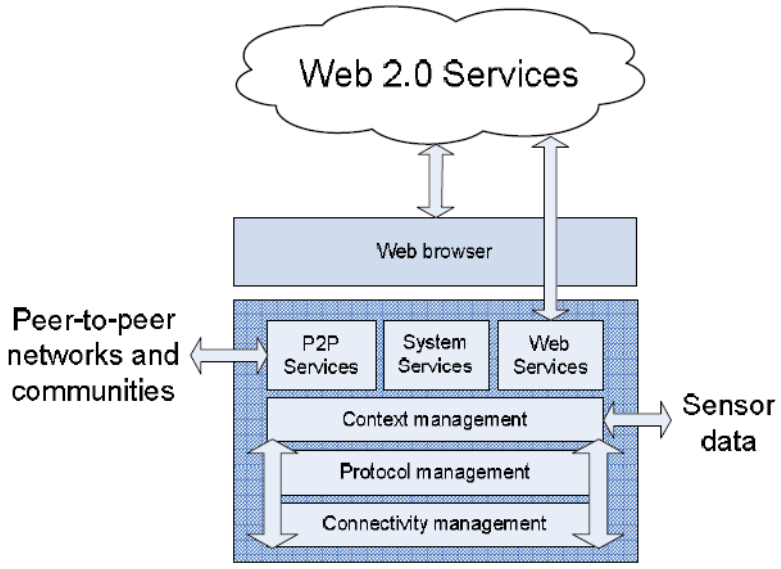



**Figure 45: Mobile middleware design**

The job of the middleware is to gather local context information, refine it and then share it with the network. By network we mean the control service or to a P2P storage network. Refining expresses essential information that is extracted from the raw local context data in a format that is recognized by the other nodes. The middleware also manages P2P networking protocols for content browsing or community management. It could also combine existing P2P-based community information, making the control service and mash-ups aware of social groups, to a certain extent [25].

### 5.4.3 Communication models for services and delivery of context and community information

There are two types of information, first is the Web 2.0 mashup services and second is the user context and community information. Also there are two methods to deliver this information, first is using a centralized manner and second in a P2P manner. Hence, there are four different combinations as shown in Table 6.

| | Services | Context and community information |
|---|---|---|
| Centralized Control | Centralized | Centralized |
| Centralized services | Centralized | P2P |
| P2P services | P2P | Centralized |
| Pure P2P | P2P | P2P |

**Table 6: Models for delivering services and user context and community information**

Now let us deal with each of the models one by one.

- **Centralized Control (CC)**
  In this model the delivery of user context and community information is achieved through database capabilities. The mobile middleware would then update the user information to the control service at the appropriate intervals as shown in the Figure 46. The mashup services are also connected to the control service and the mashup service then can query for the user information using a provided interface.

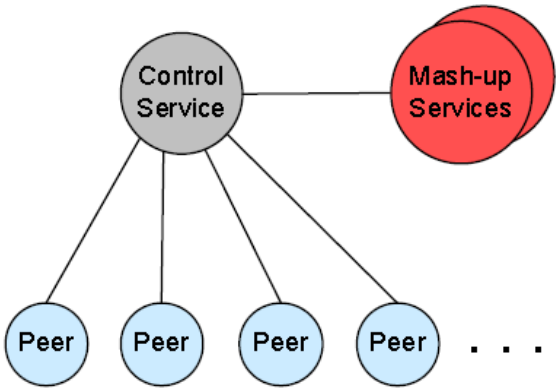



**Figure 46: Centralized control model**

The problem with this approach is that at peak usage time (i.e. massive amount of mobile users updating their user information and also when the mashup services request continuously for user content and community information) there could be a possibility of a bottleneck. The advantage of this approach is the secure storage and exact query results.

- **Centralized Services (CS)**
  In this model the user information would be made available in P2P networks by the mobile middleware. As it is clear from the Figure 47 there are multiple control services and the mashup services can query the P2P network through several control services that act as gateways to user context and community information. Each control service is bind to varying P2P networks. For gaining the most out of the network design the peers should be organized hierarchically so that the network load is divided evenly among all peers. The superpeers are the peers with great processing power and bandwidth, these superpeers is responsible for message routing and storage of information.

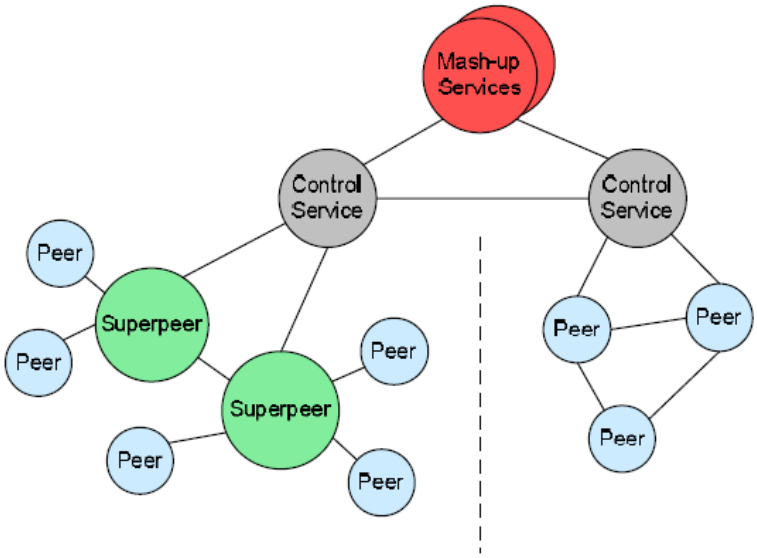



**Figure 47: Centralized services model**

- **Peer-to-Peer Services (PS)**
  In this model the user context and community information is centralized and the services are decentralized but searchable via P2P networks. This model takes the advantage of both the centralized approach (i.e. providing security) and decentralized approach (i.e. services can be indexed robustly). The model is represented in the following Figure 48.

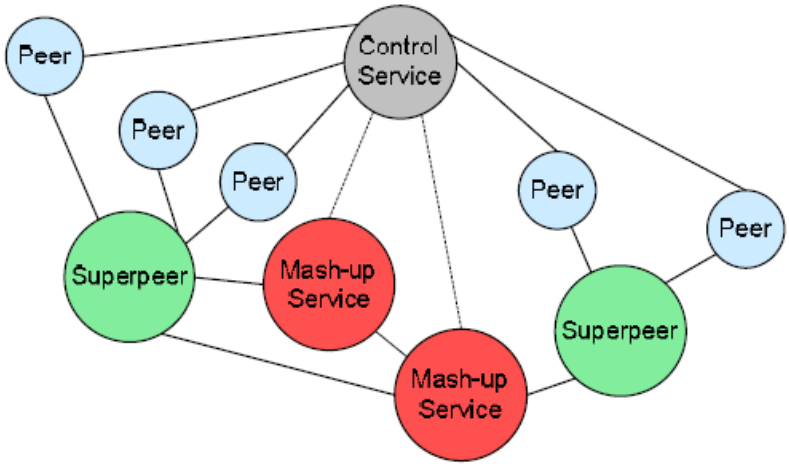



**Figure 48: Peer-to-peer services model**

- **Pure Peer-to-Peer (PP)**
  As per the name this network is organized completely in a P2P manner and the web services also being part of the existing P2P networks as shown in Figure 49. For a better understanding of this model consider a multifunctional instant messenger (IM) type client and would act as a service portal. The client can be enhanced with service discovery and rights control capabilities. Service discovery could be managed by using the IM client to make search queries in the P2P network, or users could pass links to new services among each other. The IM client would set the access rights for mash-up services (and communities). If the user discovers a new mash-up service while surfing the web, the

mash-up service could request for the appropriate access rights via the P2P network. The request would pop up in the IM client, where the user might accept or decline the request. The mash-up service would require the user's unique ID to be able to make the request. Passing the ID from the user to the mash-up should be effortless, but still implemented in a secure way [25]. The main advantage of this approach is the low need for centralized management.

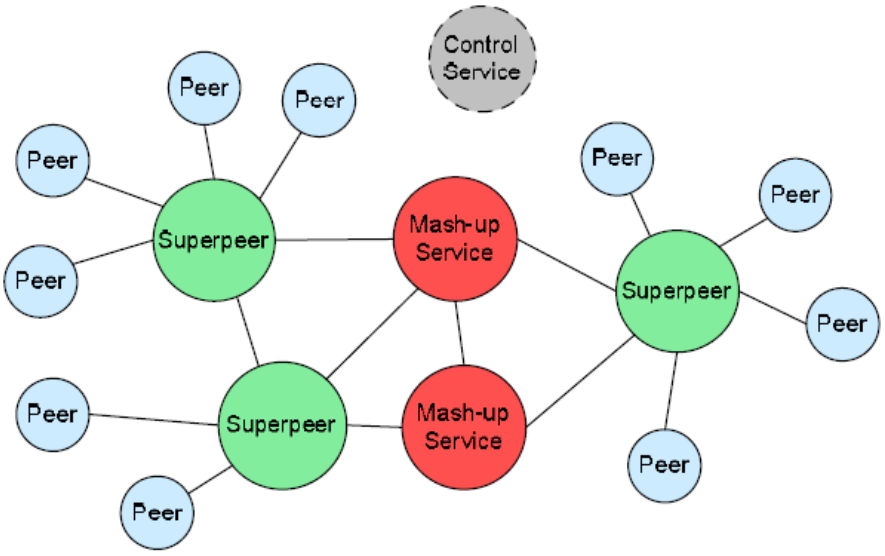



**Figure 49: Pure peer-to-peer model**

## 5.5 Mobile SOA: A Service Oriented Web 2.0 Framework for Context – Aware Lightweight and Flexible Mobile Applications

In this section we will discuss about a framework that allows mobile applications to easily interface with enterprise backend and at the same time be lightweight and flexible. This framework will have a Web 2.0 as its front end. Enterprises these days are using mobile devices to access their workflows and systems hence giving rise to trend wherein the enterprises can take real time decisions by reacting to changes in the business context. Also, the enterprise backend systems are moving to adopt Service Oriented Architectures (SOA) as they provide advantages of flexibility, implementation abstraction and interoperability all of which are important attributes for an application to run on a mobile device. Hence, this framework will allow users and enterprises to execute remote services through their devices and also the enterprises will be able to create and execute lightweight dynamic applications. With the enterprises using more of mobile technologies they have coined a new term "Enterprise Web 2.0" [26]. But before we move on to address the mobile access to enterprise systems there are a lot of issues to be considered. The issues are as follows [26]:

- Web service protocols, such as SOAP, WSDL etc are designed keeping enterprise backend systems in mind and thus are heavyweight. By heavy weight we mean that they involve a lot of discovery, packaging and communication requirements. Hence, making unsuitable for the mobile devices.

- Since the web services are used between backend systems there is no user interface thus making end user Web 2.0 applications difficult to interface with them.

- The loose coupling of services, the independent nature and fixed contract-based mode of operation of the traditional web services framework make it difficult for the mobile application since the context of mobile users change continuously.

- A large fraction of enterprise tasks are collaborative and information intensive. Mobile device user interfaces are not good at supporting exploratory data search or multiple application access from different windows. Mobile users need just-in-time access to relevant applications, services and data from a unified user interface; and traditional approaches to mobilizing enterprise applications do not address or support this.

- Accessing enterprise systems from mobile devices adds many new security concerns for enterprises. Mobile devices work over heterogeneous networks, and poor connectivity regions which makes security and confidentiality levels of intermediaries a big concern. Web service messaging between applications and services over mobile links need to be secure. Service discovery procedures also need to be secure so that rogue services do not exploit the situation. Not all of these are addressed by existing web service security models. Local data security is another concern as devices do get lost.

- The current web services have no provisions for integrating services such as SMS, camera, etc hence, not making them worthwhile for the mobile applications.

### 5.5.1 Aspects to be considered and related work

In this section we discuss the aspects that need to be considered while mobilizing applications and designing a service oriented framework for mobile devices.

- **SOA for the client**

  SOA was designed [26] to promote

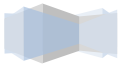
  - A clean "separation of concerns" between various components,
  - Loose coupling and contract-based coupling between service providers and consumers, and
  - Reusability, implementation abstraction, discoverability and statelessness of service providers.

  As we mentioned earlier that SOA is a heavyweight protocol thus practitioners came up with Representational State Transfer (REST) as an alternate model for realizing SOA. REST is more suitable [26] as it promotes

  - A focus on design for consumption instead of design for integration,
  - Treatment of the client as a first class citizen in an SOA ecosystem,
  - Lightweight and flexible contracts and interfaces,
  - A uniform way to interconnect with Web resources,
  - A balance between security and ease of consumption/usage of resources.

  Thus, because of these reasons SOA seems to be a better architecture for mobile devices.





- **Mobile application model**

  Applications on mobile devices follow two models:

- Thick Client
  This model requires custom development for the client and includes mechanisms for local caching of data, synchronization with backend data, and an application specific user interface.

- Portal-based
  This model instead requires backend application development and deployment, and maintenance of an enterprise portal.

Both these models satisfy the mobile application attributes i.e. they are very cost-effective for application that have a large number of users , have well-defined data and service requirements and user requirements are relatively static.

- **Architectural requirements**
  Before we move on to describing the structure for MobileSOA let us first consider the architectural requirements. As we mentioned earlier that enterprises should be able to take real time business decisions over their mobile devices, hence for this to be successful there are a set of requirements that have to be fulfilled. The requirements are listed below [10]:

  - Make intelligent use of asynchronous, semi-synchronous and push-based communication models to provide pseudo-continuous application/service level connectivity to the enterprise. This includes limited operations in non-connected scenarios also.
  - Utilize context of users and devices to make just enough information, applications, services, collaboration mechanisms, and means of effecting a decision available on the device, thus enabling fine grained device management by the enterprise.
  - Use lightweight service oriented architecture embodiment for the device and provide a clear separation of concerns between service functionality and service implementation.
  - Expose individual application functionality as services to enable effective service orchestration. Allow loose syndication of application functions to enable composite applications and seamless handoffs between applications.
  - Enable quality of service (QoS) and trust-based usage of ambient devices and services.

### 5.5.2 MobileSOA Framework

It's now time to present the MobileSOA framework [10] which will satisfy the above mentioned architectural requirements. This framework will allow services to be accessed by Web 2.0 frontend in a lightweight manner. The backend systems i.e. the enterprise will have full control over the device and can control availability and execution of services. It takes advantages of mobile and enterprise context to provide just-in-time targeted applications, services and data at the device. The framework functions as follows:
At the startup the device/user session is established with the backend and also device context is updated at regular intervals. Now, based on the current context the backend can provision remote services and policies that are applicable to device or users. The Web 2.0 applications interact with Mobile SOA framework through an asynchronous interface (Mailbox) to resolve and access various services required to execute the said application in a RESTful manner. Based on changing context backend can push new services/supplications as required.
The framework is presented in Figure 50 and it has four main classes of components explained as follows:

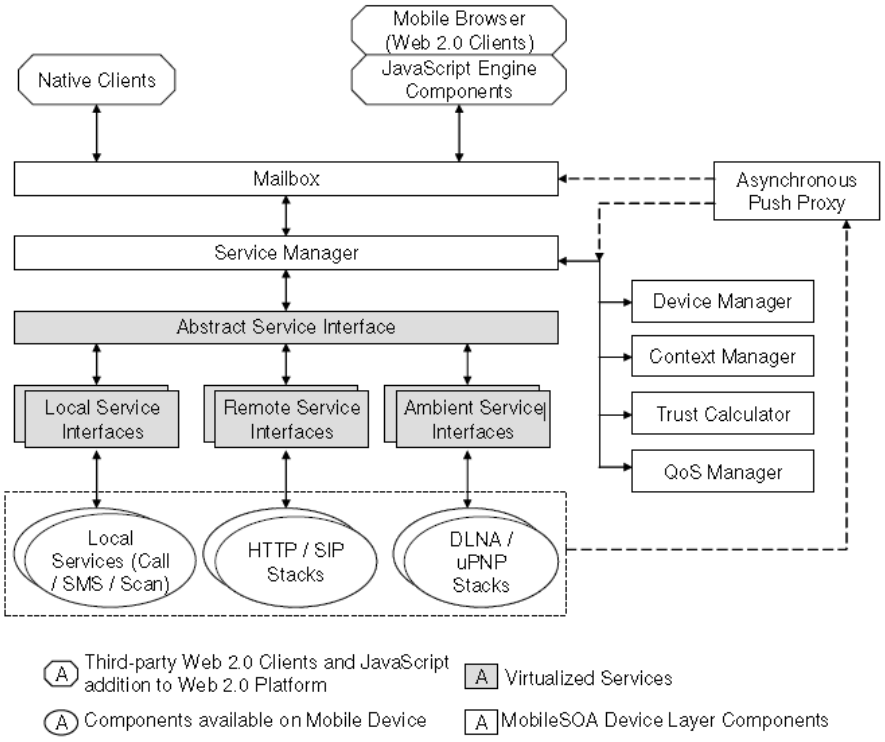



**Figure 50: Mobile SOA System Architecture**

- **Clients:**

  The clients of this framework are nothing but the Web 2.0 applications i.e. they possess the features of Web 2.0 (For features refer to paper [10]). The framework supports two types of clients. First, Native Web 2.0 clients. These clients directly use the framework's mailbox interface explained in following section. Second, Browser-based Web 2.0 clients. These clients use JavaScript object that provide JavaScript interfaces for Mailbox.

- **Mobile SOA Device Layer**

  This component is the main element of the MobileSOA framework. It consists of the following components.

  - Mailbox

    As shown in the Figure 50 mailbox acts as an interface between the client and the device layer. It receives requests from the clients and forwards it to the service manager. Reponses and asynchronous push messages are delivered to client by invoking registered handlers via callbacks. Mailbox also provides functions to store messages till the client deletes them. Another function of mailbox is that it allows applications to run scripts like JavaScript without blocking the user interface and also receive push messages from backend.

  - Device Manager

    The device, the user and the client are authenticated with the enterprise by the device manager upon startup. The device manager at regular intervals receives policy updates regarding new application provisioning and security/QoS policy from the enterprise or backend. Upon startup the device manager logs in to the enterprise and provides device and/or user credentials and the enterprise in turn sends the preliminary policy including context update URI and other information

required. The device manager initiates context update through context manager explained in the next section. Enterprise evaluates received context, generates applicable policy update and sends it to the device manager. The policy update contains information like certificates required, session information, security tokens etc. along with service information like new services provisioning and service access control.

- Context Manager
  The task of the context manager of is to update context information to the enterprise/backend at regular intervals and in turn receive contextual policy updates and then forward these to device manager. Enterprise context policies decide context parameters to be updated and frequency/conditions of update. Some of the context parameters are device location and connectivity level.

- Trust Calculator
  It is an important element in the framework as it decides the level of trust of a discovered service or deice based on enterprise specified policies and mechanisms. The mechanisms work in a way by checking the history of trusted interactions of the user with the enterprise, the recommendations from third party service recommenders and also from current context of the user. This allows in a sense only for the trusted users to use and access the services.

- QoS Manager
  This element is very important as well because it is responsible for determining whether the current quality of service is sufficient for completing the current operation. It is also determines whether a service handoff is required [10]. The operations of the QoS manager are determined by policies provisioned dynamically by the enterprise.

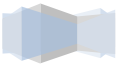
- Service Manager:
  As it is clear from Figure 50 all service accesses by client and other components are done through service manager only. It manages access to both local and remote service and also notifies the service responses to the calling modules. Service manager has a mechanism by which exposed services can register with it providing their REST based URI and readiness status and an optional service description in a standard format (like WADL). Policy updates are used to register remote services, as well as restrict service access for registered services. The service manager also provides a service resolution service that allows clients to determine which of the operations available in the received application can be actually invoked, which allows them to tailor presentation of application to user.

- Asynchronous Push Proxy
  It is possible for the backend to asynchronously invoke the clients, or framework components. It is also possible for client to invoke other client and provide startup data to the invoked clients.



➢ **Virtualized Services**

As we mentioned as earlier that all services, local or client, are virtualized as service classes and made available to clients and framework components as REST based URIs. Call service and web service can be



perceived as two different service classes. Each service class has syntax for REST URI interface invocation that is of the generic format:

*<class>:<invocation uri>?<invocation parameters>*

The service manager receives a service request from the caller via the mailbox and the service manager at that instant maps the service by invoking it through the abstract service interface. This interface provides mechanisms to invoke, terminate and restart a service.

- **Mobile Device Platform Components**

  These components are nothing but the actual lower layer device components that are used by the concrete services. The actual components used may vary a lot depending upon the specific service.

### 5.5.3 Advantages and limitations of Mobile SOA Framework

The MobileSOA framework has the following advantages [10]:

- Service Discovery

  MobileSOA framework provides an optimized service discovery mechanism by maintaining an updated list of allowable local, remote and ambient concrete service bindings in the service manager component.

- Service Binding

  MobileSOA framework allows binding of services at three places – at application generation time in the enterprise backend, at application instantiation time on the mobile device, and at service invocation time on the mobile device. This enables enhanced flexibility and use of context in determining an appropriate service binding.

- Security

  MobileSOA framework provides lightweight application level secure sessions which do not require secure network connections for the entire duration of usage of an application. Secure connection is required during session setup, when application session details and keys are communicated to the mobile device. Packets are encrypted using session keys eliminating the need for secure connection. There is a session timeout to ensure removal of stale application sessions and keys; and re-verification of credentials at regular intervals. Application session setup involves three levels of security: Device authentication, User authentication, Application authentication – which involves verifying if application is authorized to access enterprise systems, using application-level keys.

- Asynchronous Push

  Enterprise systems can now asynchronously start applications on user devices. Also one application can invoke another through the push proxy. Usage of this mechanism can lead interleaved execution of client and server applications, with asynchronous invocation from either side. Enterprise applications may also be distributed between client and server, instead of being completely on one side.

# PART IV

# Chapter 6: Web 2.0 Security

## 6.1 Web 2.0 creates security challenges

Due to the open accessibility and dynamically generated content the Web 2.0 applications seems to be interested but they also posse's bigger security risks. It can possibly happen that the attackers or hackers can exploit the vulnerabilities and add malicious content to Web 2.0 sites. This usually happens because developers focus more on functionality rather than security. One of the security mechanisms is ensuring that each page validates user input and continuous monitoring of user content. These threats and risks create a concern for the vendors and website operators because it violates the privacy of the website users.

Now we move on to show some of the security threats in the development technologies used in Web 2.0

### 6.1.1 Web 2.0 and Security Threats

It is clear that Web 2.0 websites carry more risk than the traditional websites because Web 2.0 websites contain more user uploaded content. These user uploaded content require scripting capabilities which in turn can run code or carry malware which is a potential risk to the website. It becomes easy for the hackers to exploit the Web 2.0 websites and launch worms that execute harmful operations outside the browser, leaving users unaware of their activities. Also, the hackers upload some legitimate looking content (for example: some link to anti-virus software) on websites and the users get tricked and may land up installing a virus, worm or Trojan horse on the machines. With these malicious content the hackers could possibly turn victim's machines into remote-controlled zombies which in turn launching of spam, denial-of-service, or other attacks.

Hackers take advantage of several Web 2.0 elements and applications.

- **Ajax**

  We have read about Ajax in our introduction section. We will now see how attackers exploit Ajax to perform malicious activities. Much of the processing and most of the data requests occur outside the browser window in the background rendering engine. Hence, the users might not detect security problems with Ajax code [27]. The hackers take advantage of this security hole in the browser and download code through their software and perform harmful operations.

  JavaScript let developers embed behavior such as opening windows and changing images-into web pages can run arbitrary server-provided code on a client with full privileges [27]. Hence, this vulnerability in JavaScript allows the attackers to impersonate as legitimate users and run malicious code.

  Today, about 70 percent of malicious code in the wild is downloaded via Ajax [27]. Attacks by exploiting Ajax vulnerabilities are more common because developers are not well versed with the Ajax technology leading them to design programs with inadequate security.

- **XML Syndication**

  Many Web 2.0 applications like blogs, wikis, etc use the RSS (Really Simple Syndication) feed to update the subscribed users about the changes in the web sites. Syndication systems distribute this code directly into browsers or encapsulate it into browser based or freestanding online newsreader applications. Newsfeed authors can create content themselves or let third parties upload material into

the feed. Hackers could insert code either directly into a feed or via a compromised newsfeed or via a compromised newsfeed server. The client then automatically accepts the malicious content [27].

- **Mashups**

  As we have seen in our previous sections that mashups combine data or services from multiple websites into one user experience. Mashups typically work via a set of APIs published by the website provider. The APIs let the mashup take information from the various sites and mix features from multiple sources and also the mashups can connect dynamically to the website [27]. Hence, the content provider is responsible for the security and also they should secure their servers and validate content.

- **Social Networking**

  Most of the social networking sites like YouTube, Flickr, MySpace, etc let the users upload files containing audio, video, photo, text, etc on the websites which in turn can be downloaded by other users. Hence, the attackers can include malicious code in the uploaded files which may result into harmful damages.

### 6.1.2 Typical Attack Techniques and Targets:

The majority of the attacks use some form of cross-site scripting (XSS) or cross site request forgery (CSRF)

- **XSS**

  In XSS attacks, hackers inject their own executable code into existing, legitimate, dynamically generated Web pages, as Figure 51 shows.

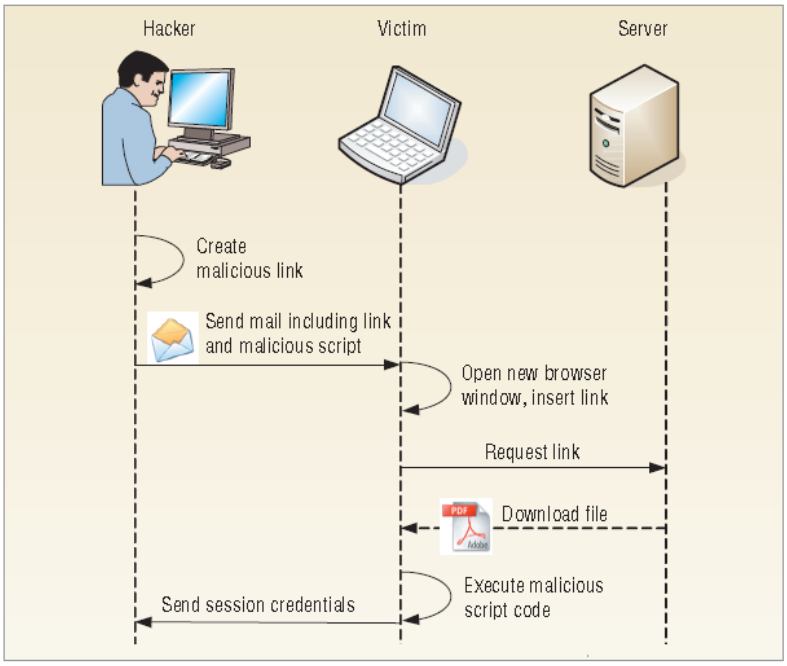



**Figure 51: Cross Site Scripting (XSS) Attack**

In the above figure a cross-site-scripting attack is seen, hacker inject their own executable code into a legitimate Web page. In this case, hacker launch an XSS attack to use a victim's computer to get a Web page from a server that the victim is authorized to access. The hacker include a malicious script that,

when the requested page downloads, executes and then sends the attackers the session credential for the victim's Web site visit. The hacker can use the credential to access and take actions on the site.

When someone downloads the page, the embedded programming accompanies the requested page and can execute on the user's computer. The code generally lets attackers gain elevated access privileges to the victim's system and, for example, steal data or change user settings. XSS can thus have disastrous consequences. One way that hackers sometimes inject the harmful code is via many popular online guestbook and forum programs that let users submit posts that include embedded HTML and JavaScript.
In particular, XSS takes advantage of Ajax-style applications, which let code execute outside a user's browser [27].

- **CSRF**

The CSRF exploits the trust a website has in a user. A typical CSRF attack can be performed in the following ways. A hacker gains access to an unsuspecting user's computer and sends unauthorized requests to an e-commerce or other Web site to which the victim has been authenticated. The hacker can also send requests via the user's computer to a company intranet to which the victim has access, thereby bypassing firewall protection. To authenticate and thus gain access to a Web site or corporate intranet, a hacker uses either the compromised computer's IP address or cookies that the site placed on the machine.

The hacker then uses either the compromised computer's IP address or cookies that the site placed on the machine and thus can authenticate and gain access to a website or corporate intranet. In this manner the hacker can impersonate the computer's owner and initiate harmful actions such as taking money from the person's bank account, ordering products from an e-commerce site, stealing data from a company intranet, or changing settings on a local firewall or router.

- **Dynamic code obfuscation**

The antivirus and anti-malware tools use pattern matching software to look for known malware's code signatures. Hackers sometimes conceal the signatures via dynamic code obfuscation. DCO uses algorithms to add randomly generated code to a JavaScript-based Web page that includes malware. The code doesn't affect the way browsers render the page and doesn't make the malicious code the page contains less harmful. But it keeps pattern-matching software from recognizing the malware.
In addition, DCO keeps mutating the code it adds to malicious files. With no single set of added code to key on, antivirus products can't develop subsequent pattern-matching software to recognize DCO altered malware.
To lure victims to a compromised Web page, hackers use techniques such as spam or phishing, which can include e-mail messages with links to malicious sites. The links have URLs that are similar to those of popular legitimate sites or that appear to advertise sexual or other content that appeals to some people. They can also work via hidden redirect code that hijacks visitors from their apparent destination to the attacker's Web site. The attack exploits browser vulnerabilities, such as buffer overflows, to place the payload—which can be a virus, Trojan horse, worm, or other type of malware—on a victim's computer.
Because DCO affects visitors to sites written in JavaScript, it is a particular threat to Web 2.0 sites [27].

- **Web 2.0 worms**

  The Web 2.0 worms can propagate in the background for a user's browser without being displayed in an open window if someone visits an infected site. The two major victims of these worms are MySpace and Yahoo Mail. MySpace was hit by a Samy worm. This worm was a JavaScript code that loaded into a browser whenever someone visited an infected MySpace page and Yammer worm hit Yahoo Mail by running itself outside the browser whenever user downloaded any attachment and the copy of the worm was sent to all the persons in the contact list of that user.

- **Exploiting Web 2.0 attacks**

  Once the attacker is successful in compromising the client with the Web 2.0 attack, the attacker can then use the, malicious code running into the browser to fool the victim's browser into believing that requests are coming from a local user. This in turn allows an attacker to reprogram the router or firewall to let the outside access to local services and then a lot of damage can be caused once the attacker has access to all the local resources. One such incident is of a hacker who could use victim's browser to initiate requests to a company's internal servers and could then access sensitive company information despite being protected by a firewall.

- **Not just browsers**

  The attacks are not just limited to browser invulnerabilities but attackers are also using applications such as Flash, QuickTime and WinZip which are most commonly used applications.

### 6.1.3 Fighting back

With the involvement of users, content providers and operators, it is their responsibility to protect their machines and websites from potential attacks. Some of the methods are: keeping browsers up to date with security patches, use application firewalls as well as application and source-code vulnerability scanners.

## 6.2 Enterprise Security for Web 2.0

In the previous section we discussed the various threats and vulnerabilities in the tools and applications used for Web 2.0 and also some of the security mechanism for each of the threat. In this section we focus on the area of enterprise security for Web 2.0 and term it as Security 2.0 [28]. Security 2.0 means securing all data, end points and networks and perimeters.

### 6.2.1 Adaptive Security

A key element to security 2.0 is the adaptive security. The security model will facilitate collaboration without making it fatally weak link in the security chain and must adapt to reflect the new reality that the data is no longer locked up. The security 2.0 will be essential to all areas pertaining to the IT industry ranging from data security to device security (on all end points) to connectivity security (all networks and perimeters).

Network perimeters haven't vanished; in fact, they must—paradoxically— both expand and shrink. They must expand to encompass all network players, even temporary ones, instead of relying on a few trusted gatekeepers. At the same time, every network component must have its own defensible security perimeter, and each perimeter must therefore shrink to fit the size of the item protected, whether that is an individual

laptop, smart phone, or any other end point. Connectivity in Security 2.0 will also follow the collaborative usage patterns of Web 2.0. No longer will it be sufficient for enterprises to maintain the traditional hub-and-spoke approach to networking. Remote users will want to connect simply and securely to other remote users, peer to- peer and point-to-point, ensuring the confidentiality and integrity of the communication, whether data, video, voice, or images.

Strong authentication and encryption technologies must operate in peer-to-peer formats and be able to scale to orders of magnitude—many tens of millions of devices, far larger than the operational threshold of current technologies [28].

### 6.2.2 Securing End Points

By endpoint we mean the end devices which clients or users use to perform operations or application like the cell phones. Today most of the business is done over the cell phones hence it is very important that these devices themselves are secured, including encryption of basic voice communications. People usually carry just once device and perform all functions on the same device, hence it becomes very crucial for security 2.0 to enable a single device to serve both personal and professional needs and one way to do this is by using firewalls on smart end points. Also security 2.0 must check the end point every time it connects to either a network or another device.

### 6.2.3 Securing Data

Securing the data itself will need active protection as it migrates from device to device. There are three ways in which the data can be secured [28].

- Intelligent search engines
  Search engines will be both smarter and more selective inside the enterprise than generic web search engines. Smarter in terms that the search engine should not record or store the keywords or patterns for confidential information or even for shopping activities. Also the intelligent search engine could be used in a privileged mode.

- Information rights management
  This type of security applies to “all data, all the time” and applies to multiple forms of information exchange-email, instant messaging, documents, spreadsheets, presentations, etc-regardless of where data lives, how and where it travels, and what devices it resides on.

- Data migration
  It is very important to secure all the network elements, such as operating systems, routers, etc because the mashups can cause a lot of mess since it might be possible that one site is secured but other is not. Hence, just having access to an application module or to privileged data through that application module doesn’t mean the user should be able to pass that access to another person through a mashup. Enterprises don’t want critical data to migrate like a cold virus—that is, to pass with every “data touch.” If the system enforces security at the data level, wherever and however it migrates, controlled collaboration doesn’t become uncontrolled data leakage.

In the enterprise computing realm, Security 2.0 means securing all end points; establishing simple, secure peer-to-peer connectivity between these end points; and controlling all data, in whatever form it lives and morphs, throughout its entire life cycle, wherever it goes and however it gets there. It also means that each network entity must self-defend because network and application perimeters themselves have become

mutable and collaborative [28]. Many a times users bypass security just because it is too complicated hence security 2.0 should be easy to use for all kind of users.

## 6.3 A Security Architecture for Web 2.0 Applications

In this section we will learn about an efficient and user-friendly architecture based on the popular tagging paradigm that connects user-defined tags with security policies, rules and social network information to ensure access control, data integrity and confidence also in derived and syndicated data.

There are two aspects of social web security: secure consumption and secure publishing. This architecture will be based on new and existing protection information on social web. Some of the techniques used are specifying access control policies by using popular and user-friendly tagging mechanisms and issuing digital signatures with user-generated content, which enable content to be related to its source even after syndication or processing. These security mechanisms will be useful only with its widespread deployment and for this there should be a high level of user acceptance and interoperability among platforms. For this reason, we also discuss a different security architecture based on semantic description. This mechanism allows decentralized definitions of security features building on top of semantic descriptions of other user and context information. Only this kind of architecture can retain the pace of application development on the Social Web, and give Social Web participants the amount of control needed for processing business transactions [29].

Before moving on to the security architecture let us first understand the different type's data and their conventional security mechanisms. Previously the content on WWW comprised of *unstructured data*. This type of data is not accessible by machines except for information retrieval and high level services. Then came the *semi-structured* data and was based on XML language and it was used on the social web. The data provided by databases or web services is called the *structured data* and this along with semi-structured data make up the mashup application. The level of security was decided by the structured data, such as the digital signatures, or list of authorized users, to decide whether a given set of data is trustworthy or which access control policy is to be applied.

The different types of web content is shown in Figure 52.

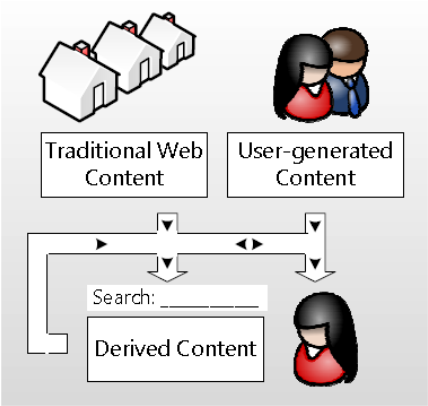



**Figure 52: Classification of Web content**

- **Traditional web content**

  This web content was used in the first decade of WWW. The scenario was such that few people were running the websites and many users consuming this content. Hence, the access to the traditional content is controlled by its owner or publisher, while Transport Layer Security (TLS) ensures that the information cannot be altered on the way from the publisher to the recipient.

- **User generated web content**

  The content is used in the social web arbitrary users can publish their content. Hence, the users themselves want to control who might access their data and in which was the access is granted. Access control might depend on the recipient's location, payment status, and other criteria. Similarly, TLS still helps protecting user-generated content on the way from the Web site to the recipient. However, TLS, which uses client or server certificates, cannot ensure that a resource has not been modified on the way to or within the Web site, and cannot convey prove of the resource owner's identity in the way digital signatures do [29].

- **Automatically generated or derived content**

  As the name suggests this type of web content is derived from other content such as search query results or aggregated data from statistical analysis or data mining technologies. Also the content in the mashup application is referred to as derived content. Until now there is no full proof security mechanism for the derived content.

We now consider several security aspects and technologies relevant for the social web. The two most common aspects are:

- **Quality of information**

  The users expect services with high quality output and this introduces the demand for technologies *measuring* information quality, *transporting* the results, and *behaving* according to the desired level of information quality.

  The two issues with the quality of information are:

  - Trust and reputation

    Trust is either configured directly, i.e. users decide to trust other users, or configured via Trusted Third Parties (TTP) such as trust centers, i.e. users decide to trust all identities provided by a verified trust center. There are also *reputation systems* which provide heuristics for propagating the trust relation between two users to a third user.

  - Trusted site syndication

    Another way of ensuring trusted information is by deploying XML signatures on the XML based information on the social web. This way the information remains connected to an identity even after distribution by syndication or web service technologies.

- **Access control**

  As we have discussed earlier that each user is itself responsible for the access control of the data on social web.

  - **Access control models:**

    Some of the most common access control models are:

- Identity-based access control (IBAC): access control depends only on the identity itself
- Role-based or Task-based access control (RBAC or TBAC): access control based on roles or task assignments in an organization
- Attribute-based access control: access control for attributes such as information obtained from digital signatures, network protocols. Application context, etc.

- **Semantic description of the security mechanisms:**
  Now let us consider the semantic description of the security mechanisms for access control. Semantic descriptions of often used patterns are proposed and can be labeled with a simple tag. This tag helps understanding the meaning of the underlying security mechanisms. Such semantic descriptions can rely on a complex security implementation in the background. With the help of these semantic descriptions, the mechanisms are transparent to users—given a clear user interface—and to services, and thus, form a basis for interoperable security architecture [29].

➢ **The Social Web Security Architecture**

This architecture represents security mechanism [29] to protect information from the moment it is published on the internet to the moment it is used by the users. Figure 53 shows the social web security architecture.

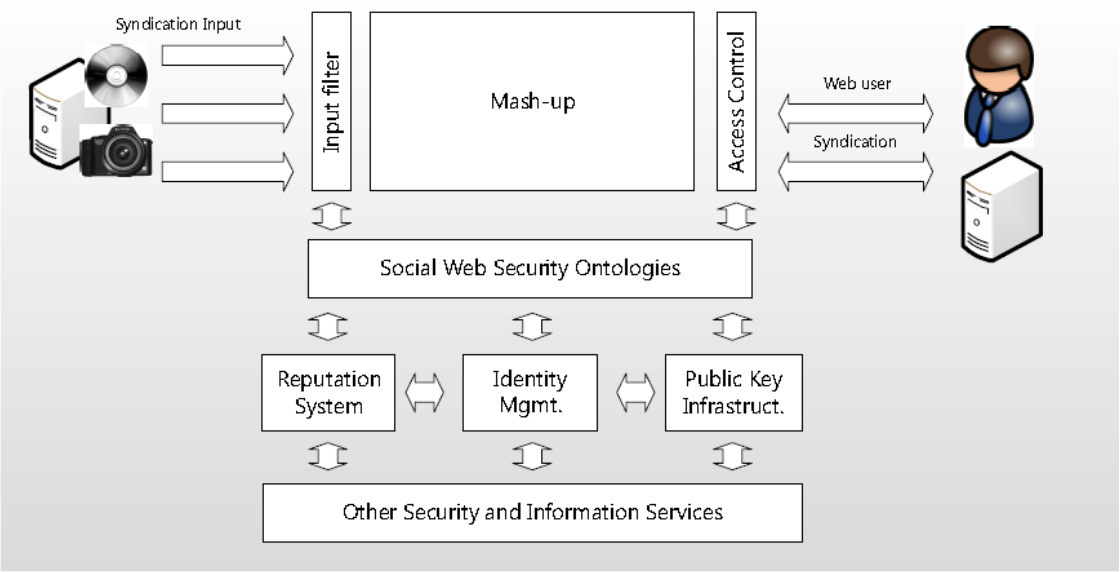



**Figure 53: Relation of participants, information and security mechanisms on the Social Web.**

The areas that are to be focused for security in web 2.0 or social web are:

- The universal widespread of users over the web, hence resulting in huge number of different identities to be considered.
- Decoupling of content from the originating sites and identities when employing Web service and syndication technologies.
- Unanticipated possibilities of transforming information using Web service and syndication technologies, leading to dramatically reduced confidence in the processed information.

Let us now consider each of the above for achieving a web of trust into four milestones.

- **Input of user generated content**

  With such a large number of users and user generated content there is no single identity management system.

  Some of upcoming identity management systems are OpenID [30] and CardSpace [31] are better than the traditional identity management systems such as LDAP directories or X.509 trust centers. This solution which is a decentralized identity management system and trust centers is not enough. However, X.509 is the most accepted standard for Public Key Infrastructures (PKI).

  Another possible solution for identity management system would be combining anonymity solutions with reputation interference by semantic web technology. With this the quality of the content can be estimated without necessarily revealing the author's identity. These steps, combined with XML signatures for trusted syndication of HTML content, will help users and services in judging the quality of the content they produce [29].

- **Output to human users and to other services**

  For the output to human users and to other services a tag based access control has been proposed [29]. First let us start with the meaning of a tag. A tag is a rule to be applied when performing access control and can be specified using OWL ontologies (Web Ontology Language) and SWRL rules (Semantic Web Rule Language) (Explained briefly in the Semantic Web section). These rules can be mapped into to a single word which would describe the rules hence enabling user friendliness, interoperability and machine readable descriptions.

  To create this system for user-centric access control [29], it takes the following:

  - Specification of access control mechanisms based on existing domain ontologies, describing security properties network protocols and organizational structures.
  - An implementation translating resource requests to the objects from the respective domain model, and reacting according to the access control decision inferred from the SWRL rules that define the access control conditions.
  - User interfaces for the configuration of security mechanisms and the association of tags and security mechanisms.

  Example architecture for realizing the issues [29] mentioned above is shown in Figure 54.

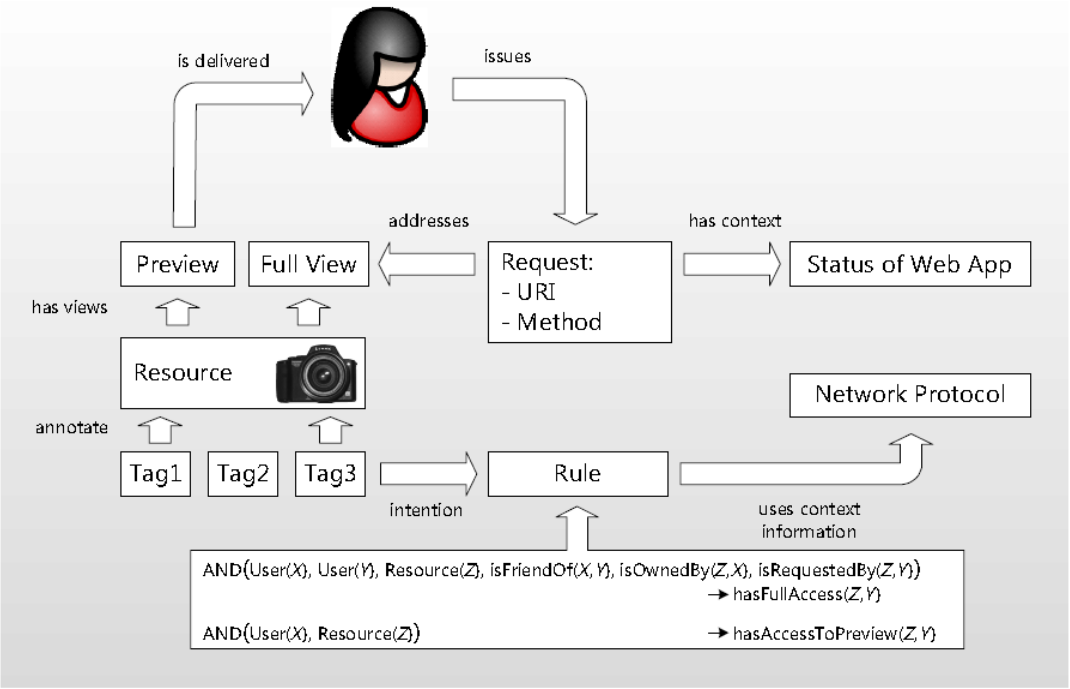



**Figure 54: Access control based on tags. Resource access is granted, if the requesting user and the resource owner are friends according to the underlying identity management.**

- Syndication and web service invocation:
  Syndication used alone creates a lot of complexity both the trust side and confidentiality side. Hence, along with syndication access control information can be passed. For this to be successful, the receiving side needs to be trusted to actually perform the access control requested and this trust can be established using trust and reputation technologies mention in the previous section.

Hence, the proposed architecture is a conceptual architecture will bring together the abstract, yet important goal of information security and the already existing user-friendly, lightweight products and services emerging in the Social Web.

# PART V

# Chapter 7: Facebook - F8 Platform

Well, we have now seen what and how it could be applied in a real world. We have also seen various patterns in section 4.3 that could be used with Web 2.0. Here we will have a closer look how to integrate a well known social networking Web 2.0 site which I am sure all of you have an account with. Yes guys! Its facebook! If you don't have one, sorry guys, you need to create an account before continuing further in this section. Just kidding. If you have atleast use any of the social networking sites then okay. Go ahead!

First we would showcase you how different website integrates with facebook and the reason behind it. Various website listed at http://developers.facebook.com/showcase/ integrates with facebook for their own benefit. Further in this section we will help you teach how you can integrate some basic features of facebook in your own site.

## 7.1 Showcasing Facebook

Why many developers want to integrate facebook with their application. According to facebook [32] there are already more than 400 million active users on their site and the dream of most website developers is to have most users to visit and use their website. Website like facebook where people network with other users, possibly his/her friends, and in most cases his/her friends believe that if his friend liked something then it would be most probably interesting for him/her too. This is a simple marketing strategy what most people do today.

Integrating facebook with your site allows you to connect with the friends of the user using facebook API's helping the user to connect his facebook friends to the site which most probably drag his friends to your site. You can also allow posting some information on the users Wall [33] or sharing some information with the users friends with openly available protocols as discussed later in this section.

Let's look into some website integrating with facebook:

### 7.1.1 Simply Hired

Simply Hired has released the first job search experience combining the world's largest social graph with the largest jobs database. With extended beta integration with Facebook Connect, job seekers can now utilize their Facebook profile and friends on SimplyHired.com to receive customized job recommendations and land a job. With this functionality, authenticated Facebook users will be able to discover jobs based on their current or previous work titles, location, interests and their friends' companies on the home page of SimplyHired.com. From the search results page, job seekers will be able to browse friends' companies and search for job openings. The goal of this functionality is to help users leverage their Facebook friends to get an inside track on a job.

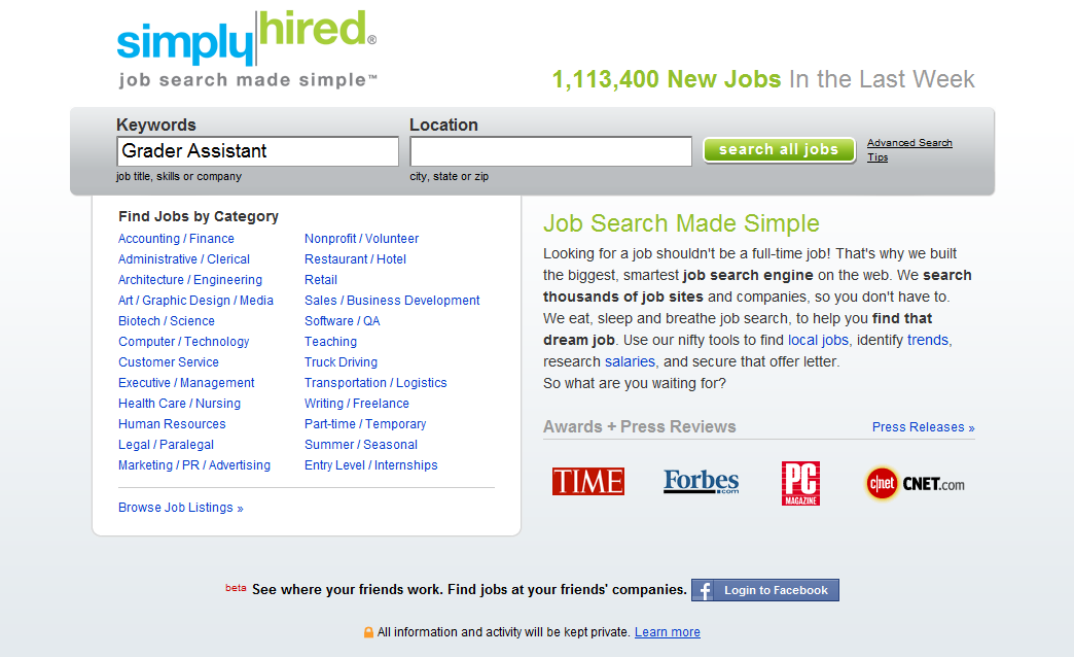



**Figure 55: SimplyHired website with login to facebook**

## 7.1.2 Cric Info

Cricinfo is the world's leading cricket web site and provides live coverage of matches, detailed cricket statistics, and features written by some of the world's best cricketers and cricket writers. Implementing social plugins, such as the Like button, allows Cricinfo to create a socially relevant and engaging experience for its over 10 million monthly users. Since implementing social plugins, Cricinfo has experienced a two fold increase in referral traffic from Facebook.


**Figure 56: CricInfo and facebook integration with Like button**

### 7.1.3 CNN

In 2009, CNN and Facebook enabled millions to experience the Inauguration of Barack Obama with their friends directly on CNN.com and set a new record for the largest live video event in Internet history. Today CNN.com has launched a site-wide integration of Facebook's new social plugins. The enhancements will incorporate multiple ways for CNN.com users to seamlessly recommend, share or comment on the news site's content with their friends on Facebook, as well as see when their friends have recommended, shared or commented on CNN.com content including articles, videos, blogs and iReports.

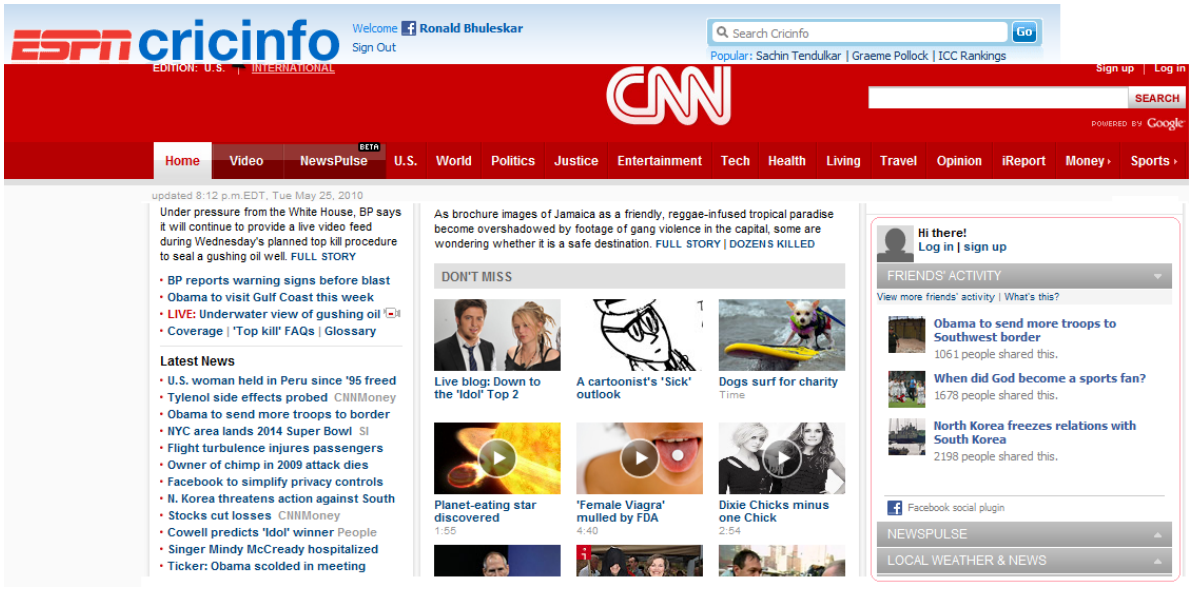



**Figure 57: ESPN providing users' friends activity**

There are countless number of application which I can go on listing, even a thousand page report would not be sufficient to list all the site which integrates with facebook's F8 developer's platform to grow the number of users to their site. Recent figures buy Justin Osofsky, on May 11, 2010 declared that 100,000+ sites uses facebook's social plugins [34], Many others uses Graph API and Open Graph Protocol.

## 7.2 F8 Developers Platform

The core of facebook is their social graph, without which facebook would not be a million dollar company. It is all people and their connection with other people what they need to maintain.

Facebook F8 Platform is the set of APIs and tools which enable you to integrate with the social graph to drive registration, personalization, and traffic — whether through applications on Facebook.com or external websites and devices [34]. Facebook provides API's for your websites, apps on Facebook.com and for your mobile apps.

The F8 platform lets you achieve the following tasks:

### 7.2.1 Registration + Login

With a single dialog, you can access data including a user's real name, email address, profile picture and list of friends. Replace or supplement your user account system with Facebook to help drive signups and improve data quality.

### 7.2.2 Engagement

With Facebook users comes with their friends; you could incorporate connections to make your product more engaging. Social plugins like the Like button and the activity feed enable you to offer social experiences with just a line of HTML. The Graph API enables you to integrate the social graph into your site in deep and compelling ways.

### 7.2.3 Growth

You can publish content from your site into the social graph to reach your users' friends. The Like button enables users to share your site's content back to their Facebook stream with one click. In addition, you can integrate pages deeply into the social graph via the Open Graph protocol.

## 7.3 Features of F8 Platform

Let's have an overview of the F8's platform what features it provides to the developers:

- **Graph API**

  This API enables you to read and write data to Facebook. It provides a simple and consistent view of the social graph, uniformly representing objects (like people, photos, events, and pages) and the connections between them (like friendships, likes, and photo tags).

- **Authentication**

  Facebook authentication enables your application to interact with the Graph API on behalf of Facebook users, and it provides a powerful single-sign on mechanism across Web, mobile, and desktop apps.

- **Social plugins**

  Social plugins enable you to provide engaging social experiences to your users with just a line of HTML. Because the plugins are served by Facebook, the content is personalized to the viewer whether or not they have signed into your site.

- **Open Graph protocol**

  The Open Graph protocol enables you to integrate your pages into the social graph. These pages gain the functionality of other graph objects including profile links and stream updates for connected users.

### 7.3.1 Graph API

The new Graph API updated on April 21 attempts to drastically simplify the way developers read and write data to Facebook. It presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph and the connections between them

Every object in the social graph has a unique ID. You can fetch the data associated with an object by fetching https://graph.facebook.com/ID. For example, the official page for the Facebook Platform has id 19292868552, so you can fetch the object at https://graph.facebook.com/19292868552 which returns you something like this:

```
{
  "name": "Facebook Platform",
  "type": "page",
  "website": "http://developers.facebook.com",
  "username": "platform",
  "founded": "May 2007",
  "company_overview": "Facebook Platform enables anyone to
build...",
  "mission": "To make the web more open and social.",
  "products": "Facebook Application Programming Interface
(API)...",
  "fan_count": 449921,
  "id": 19292868552,
  "category": "Technology"
}
```


As said everything in facebook can be accessed as JSON objects people and pages with usernames can be fetched using their username as an ID. Since "platform" is the username for the page above, https://graph.facebook.com/platform will return what you expect.

To retrieve users profile given a object id for a user is as follows:
https://graph.facebook.com/661050362


```
{
  "id": "661050362",
  "name": "Ronald Bhuleskar",
  "first_name": "Ronald",
  "last_name": "Bhuleskar",
  "link": "http://www.facebook.com/people/Ronald-
Bhuleskar/661050362",
  "gender": "male"
}
```


All objects in Facebook can be accessed in the same way some uses are listed below:

| |
|---|
| **Users**: https://graph.facebook.com/btaylor (Bret Taylor) |
| **Pages**: https://graph.facebook.com/cocacola (Coca-Cola page) |
| **Events**: https://graph.facebook.com/251906384206 (Facebook Developer Garage Austin) |
| **Groups**: https://graph.facebook.com/2204501798 (Emacs users group) |
| **Applications**: https://graph.facebook.com/2439131959 (the Graffiti app) |
| **Status messages**: https://graph.facebook.com/367501354973 (A status message from Bret) |
| **Photos**: https://graph.facebook.com/98423808305 (A photo from the Coca-Cola page) |
| **Photo albums**: https://graph.facebook.com/99394368305 (Coca-Cola's wall photos) |
| **Videos**: https://graph.facebook.com/614004947048 (A Facebook tech talk on Tornado) |
| **Notes**: https://graph.facebook.com/122788341354 (Note announcing Facebook for iPhone 3.0) |

All of the objects in the Facebook social graph are connected to each other via relationships. Bret Taylor is a fan of the Coca-Cola page, and Bret Taylor and Arjun Banker are friends. We call those relationships connections in our API. You can examine the connections between objects using the URL structure https://graph.facebook.com/ID/CONNECTION_TYPE.

The connections supported for people and pages include:

| |
|---|
| **Friends**: https://graph.facebook.com/me/friends |
| **News feed**: https://graph.facebook.com/me/home |
| **Profile feed (Wall)**: https://graph.facebook.com/me/feed |
| **Likes**: https://graph.facebook.com/me/likes |
| **Movies**: https://graph.facebook.com/me/movies |
| **Books**: https://graph.facebook.com/me/books |
| **Notes**: https://graph.facebook.com/me/notes |
| **Photos**: https://graph.facebook.com/me/photos |
| **Videos**: https://graph.facebook.com/me/videos |
| **Events**: https://graph.facebook.com/me/events |
| **Groups**: https://graph.facebook.com/me/groups |

Facebook support different connection types for different objects. For example, you can get the list of all the people attending COEN 332 Wireless Mobile and Multimedia Network Final Exam by just mentioning the Event ID (For example ID #24560554565) by fetching https://graph.facebook.com/24560554565/attending.

Now you have a clear picture how you can obtain personal information your user just by having the user's name or the actual object id for the user. Not only user's information but also the events, pages, etc. You might wonder why the relationships API's doesn't work. It is because before you ask for the private information you need to authenticate to facebook through authentication API as discussed next.

### 7.3.2 Authentication

Facebook Platform uses the OAuth 2.0 protocol for authentication and authorization. It provides number of ways with which you can authenticate users in web applications via redirects, in JavaScript, or in desktop and mobile applications.

When a Facebook user authorizes your application, your application gets access to the user's Facebook ID. By default, your application can access all general information in a user's profile, including her name, profile picture, gender, and friend list. If your application needs to access other parts of the user's profile that may be private, your application can request extended permissions. For example, if you want to incorporate a user's photos into your application, you would request the user photos extended permission. During the authentication process, the user is presented with a UI in which the user can authorize your application to access that specific part of her profile:

Extended permissions are also available to authorize write access to a user's profile, so your application can, e.g., publish new posts to a user's wall. You can find more information on how to authenticate using extended authentication and single sign-on at http://developers.facebook.com/docs/authentication/

### 7.3.3 Social Plugins

Social plugins let you see what your friends have liked, commented on or shared on sites across the web. All social plugins are extensions of Facebook and are specifically designed so none of your data is shared with the sites on which they appear [35].

All social plugins are available to your site with just a line of HTML. Most of the code according to your choice could be obtained directly from the facebook site http://developers.facebook.com/plugins Some of the social plugins available to the developers are:

**Like Button**


The Like button lets users share pages from your site back to their Facebook profile with one click.

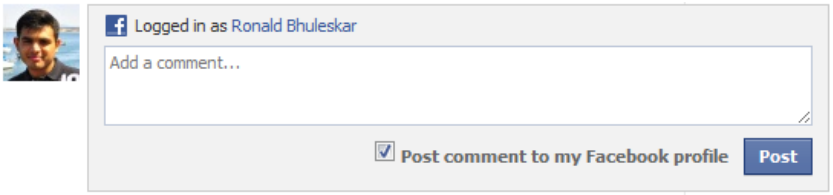
**Like Box**

The Like box enables users to like your Facebook Page and view its stream directly from your website.


**Login with Faces**


The Login with Faces plugin shows profile pictures of the user's friends who have already signed up for your site in addition to a login button.

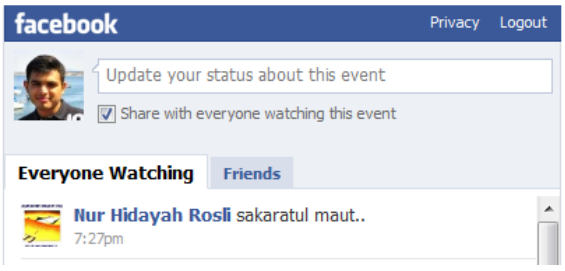
**Comments**



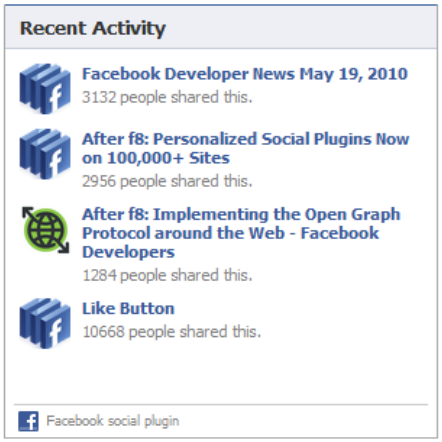

The Comments plugin lets users comment on any piece of content on your site.

**Live Stream**

The Live Stream plugin lets your usersshare activity and comments in real-time as they interact during a live event.

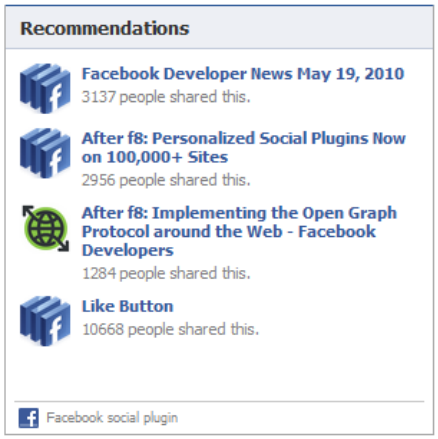


**Activity Feed**




The Activity Feed plugin shows users what their friends are doing on your site through likes and comments.

**Recommendations**




The Recommendations plugin gives users personalized suggestions they might like

**Facepile**

The Facepile plugin shows profile pictures of the user's friends who have already signed up for your site.

## 7.4 Facebook SDKs

There are several SDK's available in variety of languages as listed below:

- JavaScript SDK
- PHP SDK
- Python SDK
- iPhone SDK
- Android SDK (unofficial)

## 7.5 Advanced APIs

Some recent updates by facebook has released with few new API's, some of which are:

- Facebook Query Language (FQL)
- Facebook Markup Language (FBML)
- Old REST API
- Old JavaScript Client Library

## 7.6 New Permission Model

A new permission model has been defined effective June 1, 2010, where only user's public data would be allowed for access unless the user grants your application the extended permissions it needs.

To access to private information user must explicitly authenticate the application. During the authentication process, the user is presented with a UI in which the user can authorize your application to access that specific part of her profile. The UI also displays for the users to know what information that would be fetched after a successful authentication. The sample authentication of the new model is shown below:

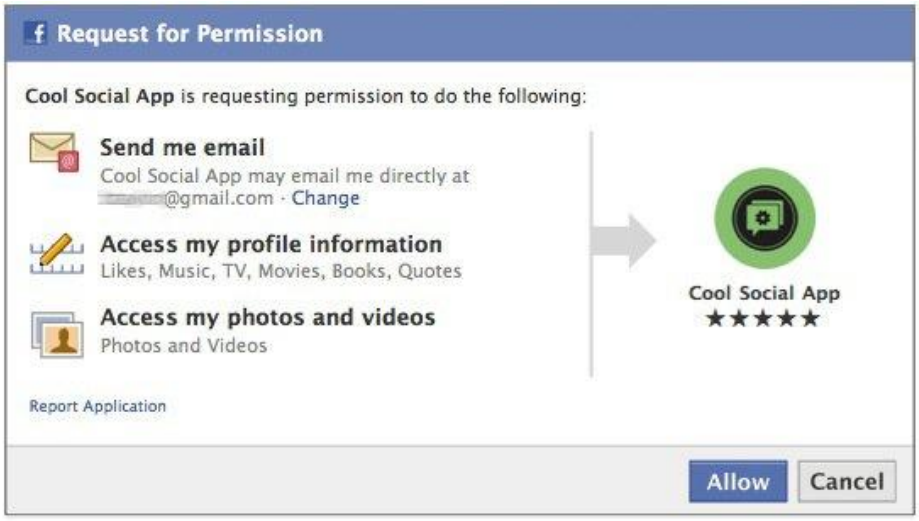



**Figure 58: A new authentication model**

If by June 1, 2010 your application has not been upgraded to the new permissions model as defined, it will be automatically upgraded. What this means is that your application will be provided with only public information by default with the single authorization dialog.

# Chapter 8: Applying SOA and Web 2.0 to Telecom

So far we have discussed about the SOA architecture and the Web 2.o models and patterns. In this chapter we will apply both of these to telecom i.e. to legacy and IMS next generation architectures. First we will start with the "SOA-lization" of telecom architectures and then move explain "Web 2.0-ification" of the same [36]. IMS (IP Multimedia Subsystem) is a set of specifications that describes the Next Generation Networking (NGN) architectures for implementing IP based telephony and multimedia services. For more on IMS refer [37]. IMS is very important to support SOA as it enables to reuse existing service platforms and enrich them with capabilities provided by the emerging IMS service platform. Figure 59 illustrates the overall architecture diagram.

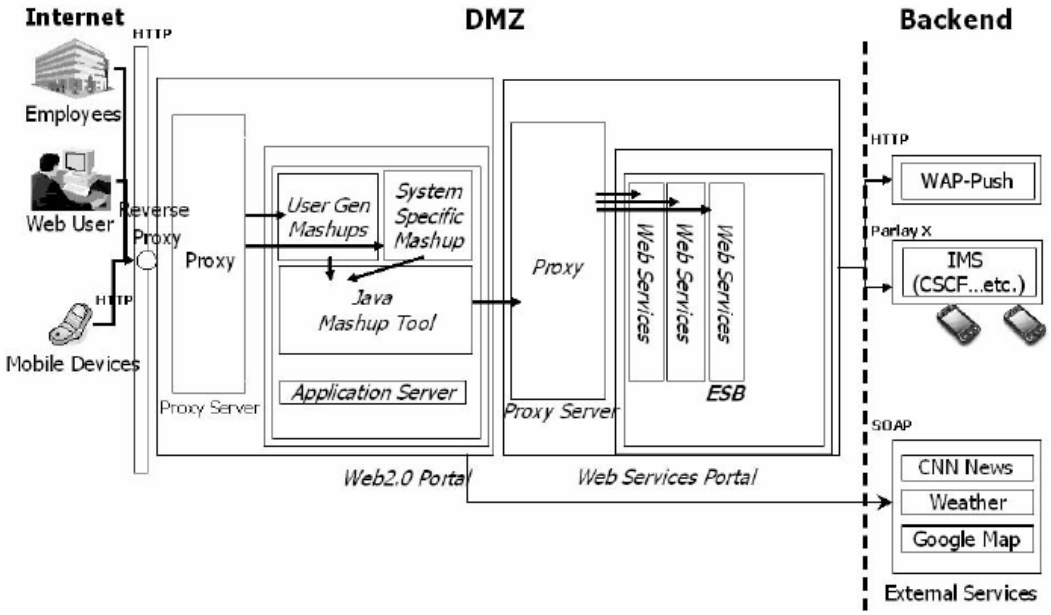



**Figure 59: Overall Architecture**

## 8.1 Migrating traditional telecom interfaces to SOA

In this section we will discuss out first part i.e. SOA-lization of telecom architectures. Following are some issues and criteria to be considered to achieving this migration.

- **Why SOA in telecom**

  SOA holds some important features that enable the creation of Web 2.0 applications easily. Some of the features are: fast service creation, deployment and composition. SOA also posses certain characteristics that support and enhance architectures for exposing telecom services. Some of the characteristics are:

  - Loose coupling, with well-defined interfaces
  - Stateless service design
  - Service granularity/atomicity
  - Quality of service considerations

- **Choosing services for SOA enablement**
  There are several options for services that can benefit from integration into a SOA architecture. These include:
    - Native or legacy telecom services such as SMS/MMS messaging, presence, and terminal status and location, either through direct integration with network elements, or usage of a communications gateway (Parlay gateway)
    - Next-generation services in IMS such as SIP-based call control, SIP-based presence, or XML Document Management (XDM)

- **Interface type selection**
  There are many options for choosing the type of interface like the RESTful web services, RPC, J2EE and others, but here we consider the use of SOAP-based web services [15] for building SOA-enables services as it is robust, weel understood and well supported in SOA based tooling.

Parlay X [38] standard can be used as a Web service for the telecom and IBM WebSphere Telecom Web Service Server (TWSS) [39] is a platform for building and deplpying SOA-based services and for centralizing AAA (Authentication, Authorization & Accounting), policy and QoS for those services. A SOA-lization of services involving the above two is illustrated in Figure 60.

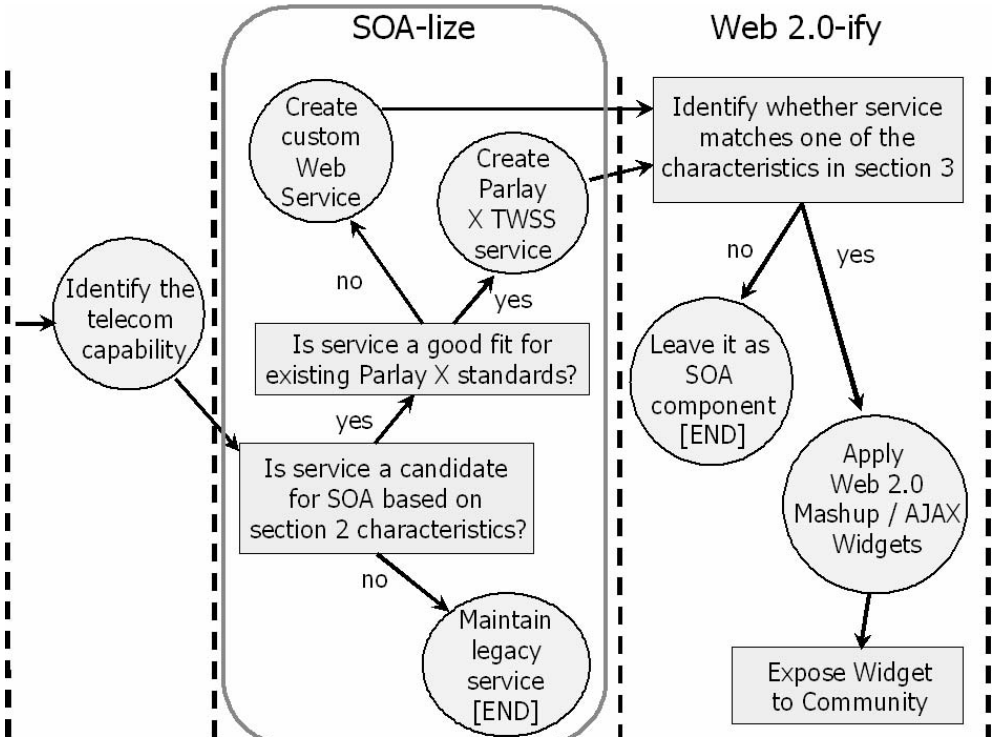



**Figure 60: SOA-lization of services**

## 8.2 Enabling a SOA-lized telecom capability to Web 2.0

In the previous section we have seen the SOA-lization of telecom and it is depicted in Figure 60. We now extend the figure to enable SOA-lized telecom capability to Web 2.0 and it is depicted in Figure 61.

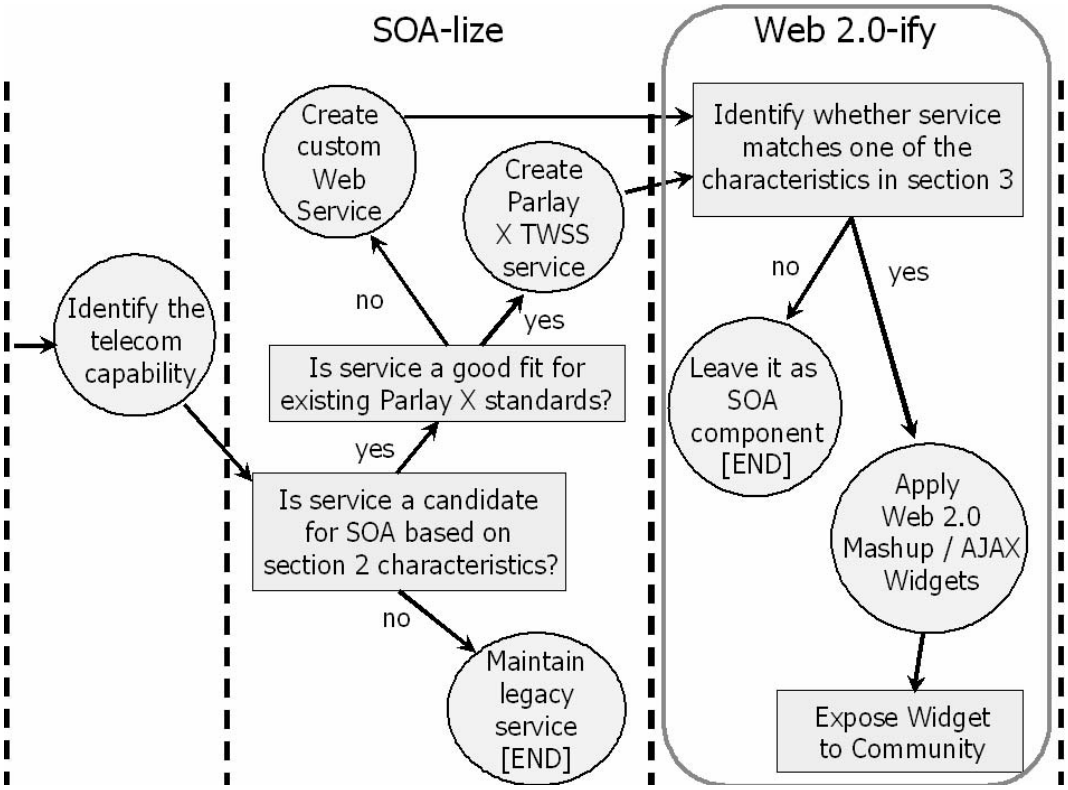



**Figure 61: Web 2.0-ification**

**Identify which telecom capabilities fit to be Web 2.0-fied.**

Before we move forward we should note one thing that Web 2.0 is unlike SOA. Web 2.0 is not about technology it is about social behavior. The following are some of the additional characteristics needed to identify whether the components (Components used to realize the SOA-lization) are right for Web 2.0:

- **Assemble**
  - Subject Matter experts who may not be programmers can create web applications to address just-in-time ad-hoc situational needs
  - Integrate data and markup using widgets to create new utilities

- **Wire**
  - Bind rich content from disparate sources to create new ways to view information
  - Add behavior and relationships to disparate widgets to create a rich interactive application experience

- **Share**
  - Quickly promote your mashup for use by others
  - Enable multi-user collaboration on the development of a mashup

Now if you see that at a widget level, the Assemble and Wire steps are applied. A widget can wither work independently or along with other widgets. Multiple Web 2.0 widgets can be "assembled" together to form a bigger Mashup, and their data can be also "wired" together so that the widgets are capable of talking to each other. Hence, we can conclude that the criteria for turning a SOA compoenet into a web 2.0 widget is determined by whether the component has the above mentioned three characteristics.

The following Figure 62 illustrates the flow of how to enable a telecom system first for SOA Web Services, and then to build on those services for Web 2.0. The very first step is to identify the capabilities of the telecom which are a good fit to expose through the SOA web services. Once the telecom architecture is SOA-lized then we have a next generation telecom solution that can leverage legacy and IMS environments. Next, apply those same capabilities to see if any benefits through Web 2.0 can be achieved.

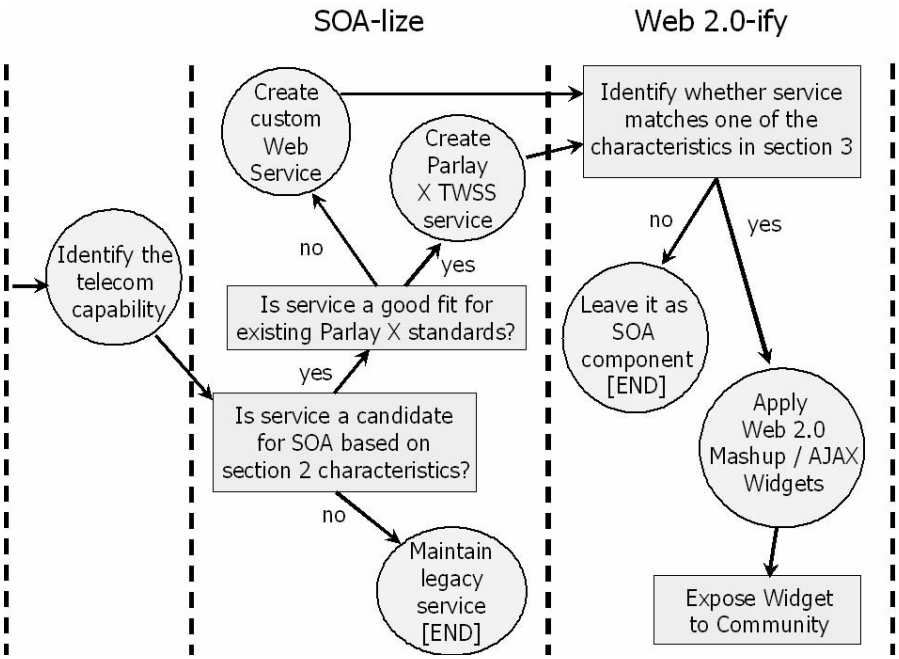



**Figure 62: End-to-end decision tree**

# PART VI

# Chapter 9: Future Scope

## Web 3.0

The shift from Web 1.0 to Web 2.0 has taken a decade long, but next generation of Web will be known as the Web 3.0 and the shift from Web 2.0 to Web 3.0 will be faster and very soon. Web 3.0 is denoted as a real personal web [40] and has started creating many questions among the stakeholders of the web. It is noted that Semantic Web will be used for implementing Web 3.0. Semantic web as we have seen in the section 1.5, which possess many limitations due to which it did not gain recognition. One of the major drawbacks of Semantic web is that it is machine readable. But now Web 3.0 will bring users and producers (create contents for users) closer to machines for more dynamic, interactive and efficient creation of the contents and its management and it will be possible by using the Semantic web.

Figure 63 shows the evolution of web from 1.0 to 3.0:

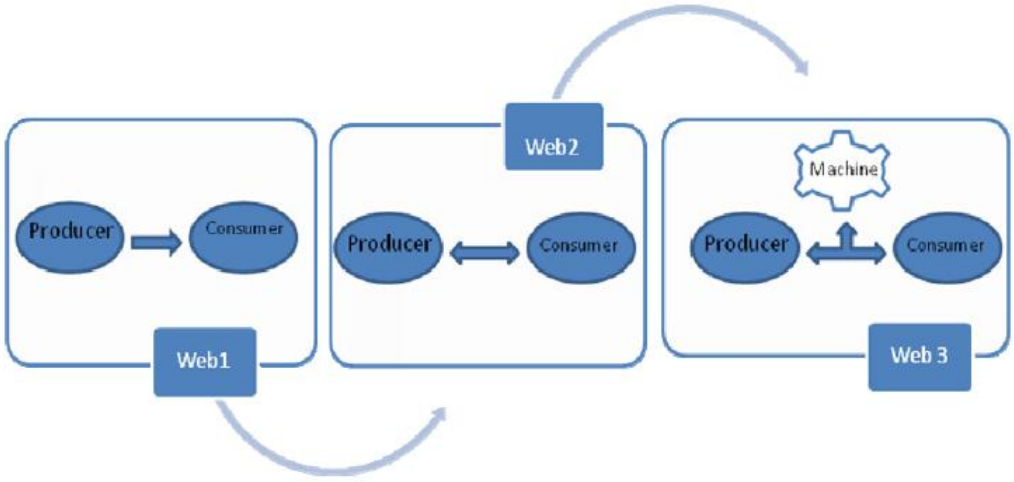



**Figure 63: Web Evolution**

The following Table 7 shows the differences between Web 1.0, 2.0 & 3.0

| Web 1.0 | Web 2.0 | Web 3.0 |
|---|---|---|
| **Front Page** | MySpace | SIOC-Project.org |
| **Encarta** | Wikipedia | Dbpedia |
| **StreetMap/MapQuest** | Google Earth | 3-D Street View |
| **PC Games** | Online Games | Online 3-D Games |
| **Home Video** | YouTube | Yet to come |
| **Mp3.com** | iTunes | Yet to come |
| **Microsoft Office** | Google Docs | Yet to come |

**Table 7: Distinctive Web applications**

In Web 3.0, the principle is based upon linking, integrating and analyzing data from various data sources into new information streams. In short, the main aim of Web 3.0 of personalization of web. Linking of data in Web 3.0 is achieved with the help of semantic technologies like Resource Distribution Framework (RDF) [41] and SPARQL [42]. The two main building blocks of Web 3.0 are the data from Web 2.0 and technology from Semantic web. Semantic technologies have the open standards and can be applied on the top of the existing web as shown in the Figure 64.


**Figure 64: Semantic Technologies, Web 2.0 and Web 3.0**

Social computing environment means Web 3.0 focuses on the human-machine relationship and desires to organize a large number of current social web communities. These Semantic technologies can play a key role in the future Web 3.0 applications [40].

Web 3.0 seems to be quite interesting and has the level of interest but it does posses security challenges. Some of them are listed below:

- credibility of the data
- control over the metadata,
- privacy implications of massive data mash-ups

For more on limitations kindly refer to [40].

# Conclusion

After reading the report, you would be well versed with answers for the basic questions: what is Web 2.0? How different it is from the initial web specifications Web 1.0. What are its benefits, its application and how various patterns could be applied?

We come up with the conclusion that Web 2.0 is essential in today's world as it eliminates the basic read-only web and brings-in write on web feature. Information sharing in today's world is very important and with the high demand for open source technologies for better and rapid software development. There is no enforcement that could be provided for your application to be with Web 2.0 standard, it is solely on the designers, architects and the developers to adapt it according to the need. It is also not necessary to have all applications run as Web 2.0, but it entirely depends on what kind of application it is and if a better model could be achieved.

We defined how to approach for a given problem with the basic Metamodel as discussed in chapter 2. Also a detailed Web 2.0 architecture has been studies in chapter 3. Chapter 4 catalogs several patterns which would be useful to identify a similar requirement for your Web 2.0 application.

Mobility issues of Web 2.0 have been studied and a complete architecture for Mobile Web 2.0 using SOA has been discussed in chapter 5. Security being an vital aspect of any technology hence there could be a breakthrough even in Web 2.0. Hence a security model has to be mandated to avoid various possible attacks on the application or system. Chapter 6 talks about the security breaches, how an enterprise security in Web 2.0 can be achieved and Web 2.0 security architecture.

A more focused, developers guide for Facebook F8 platform is described in chapter 7. Though this chapter is focused to the developers of Web 2.0 integrating with facebook this provides other readers with a perfect example the advantages of integrating with another Web 2.0 application. Also it provides with an inner look with different open API's facebook provides with how to use them.

We applied Web 2.0 and SOA to telecom i.e. to legacy and IMS next generation architectures in chapter 8. Starting with the "SOA-lization" of telecom architectures and then move forward explaining "Web 2.0-ification" of the same.

Finally we conclude our topic with our future scope of Web 2.0 i.e. a next generation Web 3.0 which will focus more towards personalization of Web. Still it is not what level of personalization is required and sure how users will react to it.

The authors would like to add a note at this concluding section of the report: Though many say Web 2.0 is just a marketing term and no new standard has been evolved specifically to be used with Web 2.0 but a model defined and designed with a proper pattern would definitely be a huge affect in terms of users acceptability, because of rich user experience (RUE ), low bandwidth requirements, providing open API's for extensibility with other applications and a lot more users' participating in the web rather than just be a dummy entities to the Web.

# Acronyms & Definitions

| Acronym | Full Form | Description |
|---|---|---|
| **AIR** | Adobe Integrated Runtime | The **Adobe AIR runtime** lets developers use proven web desktop application technologies to build rich Internet applications that run outside the browser on multiple operating systems. |
| **AJAX** | Asynchronous JavaScript And XML | **Ajax** is a group of interrelated web development techniques used on the client-side to create interactive web applications. |
| **AMF** | Action Message Format | **Action Message Format** (**AMF**) is a binary format used to serialize ActionScript objects. It is used primarily to exchange data between an Adobe Flash application and a remote service, usually over the internet. |
| **API** | Application Programming Interface | An **Application Programming Interface** (**API**) is an interface implemented by a software program which enables it to interact with other software. It is similar to the way the user interface facilitates interaction between humans and computers. |
| **ASP** | Active Server Pages | **Active Server Pages** (**ASP**), also known as *Classic ASP* or *ASP Classic*, was Microsoft's first server-side script-engine for dynamically-generated web pages. |
| **CRM** | Customer Relationship Management | **Customer relationship management** is a broadly recognized, widely-implemented strategy for managing and nurturing a company's interactions with clients and sales prospects. |
| **CSRF** | Cross-Site Request Forgery | **Cross-site request forgery**, also known as a **one-click attack** or **session riding** and abbreviated as **CSRF** or **XSRF**, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser. |
| **CSS** | Cascading Style Sheets | **Cascading Style Sheets** (**CSS**) is a style sheet language used to describe the presentation semantics (that is, the look and formatting) of a document written in a markup language. |
| **DCO** | Dynamic Code Obfuscation | DCO uses algorithms to add randomly generated code to a JavaScript-based Web page that includes malware. |
| **DHTML** | Dynamic HTML | **Dynamic HTML**, or **DHTML**, is an umbrella term for a collection of technologies used together to create interactive and animated web sites by using a combination of a static markup language (such as HTML), a client-side scripting language (such as JavaScript), a presentation |

| | | |
|---|---|---|
| | | definition language (such as CSS), and the Document Object Model. |
| **DOM** | Document Object Model | The **Document Object Model** (**DOM**) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. |
| **DoS** | Denial-Of-Service | A **denial-of-service attack** (**DoS attack**) or **distributed denial-of-service attack** (**DDoS attack**) is an attempt to make a computer resource unavailable to its intended users. |
| **ECM** | Enterprise Content Management | **enterprise content management** (**ECM**) refers to the technologies, strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to an organization and its processes. |
| **EDA** | Event-Driven Architecture | **Event-driven architecture** (**EDA**) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. |
| **EIS** | Enterprise Information System | An **Enterprise Information System** is generally any kind of computing system that is of "enterprise class". This means typically offering high quality of service, dealing with large volumes of data and capable of supporting some large organization ("an enterprise"). |
| **EJB** | Enterprise Javabeans | **Enterprise JavaBeans** (**EJB**) is a managed, server-side component architecture for modular construction of enterprise applications. |
| **ERP** | Enterprise Resource Planning | **Enterprise resource planning** (**ERP**) is an Integrated computer-based system used to manage internal and external resources including tangible assets, financial resources, materials, and human resources. |
| **FBML** | Facebook Markup Language | **Facebook Markup Language** (**FBML**) enables you to build full Facebook Platform applications that deeply integrate into a user's Facebook experience. You can hook into several Facebook integration points, including the profile, profile actions, Facebook canvas, News Feed and Mini-Feed. |
| **FOAF** | Friend Of A Friend | **FOAF** (an acronym of **Friend of a friend**) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe him or herself. FOAF allows groups of people to describe social networks without the need for a centralized database. |
| **FQL** | Facebook Query Language | Facebook Query Language, or FQL, allows you to use a SQL-style interface to more easily query the same Facebook social data that you can access through other Facebook API methods (assuming your application has access!). |

| | | |
|---|---|---|
| **GIF** | Graphics Interchange Format | The **Graphics Interchange Format** (**GIF**) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability. |
| **HSF** | Hybrid Spam Filter | A spam filtering approach using various participating filters each defining the legitimacy or spam characteristics. |
| **HTML** | Hypertext Markup Language | **HTML**, which stands for **HyperText Markup Language**, is the predominant markup language for web pages. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. |
| **HTTP** | Hypertext Transfer Protocol | The **Hypertext Transfer Protocol** (**HTTP**) is an Application Layer protocol for distributed, collaborative, hypermedia information systems. |
| **IDE** | Integrated Development Environment | An **integrated development environment (IDE)** also known as *integrated design environment* or *integrated debugging environment* is asoftware application that provides comprehensive facilities to computer programmers for software development. |
| **J2EE** | Java Platform, Enterprise Edition | **Java Platform, Enterprise Edition** or **Java EE** is a widely used platform for server programming in the Java programming language. |
| **JMS** | Java Message Service | The **Java Message Service** (**JMS**) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. |
| **JPEG** | Joint Photographic Experts Group | **JPEG** is a commonly used method of lossy compression for photographic images. |
| **JSON** | Javascript Object Notation | **JSON** is a lightweight text-based open standard designed for human-readable data interchange. |
| **JSP** | Java Server Pages | **JavaServer Pages** (**JSP**) is a Java technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. |
| **LDAP** | Lightweight Directory Access Protocol | The **Lightweight Directory Access Protocol**, or **LDAP**, is an application protocol for querying and modifying data using directory services running over TCP/IP. |
| **MMS** | Multimedia Messaging Service | **Multimedia Messaging Service**, or **MMS**, is a standard way to send messages that include multimedia content to and from mobile phones. |
| **MTOM** | *Message Transmission Optimization Mechanism* | **MTOM** is the W3C *Message Transmission Optimization Mechanism*, a method of efficiently sending binary data to and from Web services. |
| **MVC** | Model–View–Controller | **Model–View–Controller** (**MVC**) is a software architecture, currently considered an architectural pattern used in software engineering. |

| | | |
|---|---|---|
| **NET** | Microsoft .Net Framework | The **Microsoft .NET Framework** is a software framework that can be installed on computers running Microsoft Windows operating systems. |
| **OASIS** | Organization For Advancement Of Structured Information Systems | A non-profit consortium that drives development, convergence and adoption of open standards for global information society. |
| **ODF** | Open Document Format | The **Open Document Format for Office Applications** (also known as **OpenDocument** or **ODF**) is an XML-based file format for representing electronic documents such as spreadsheets, charts, presentations and word processing documents. |
| **OWL** | Web Ontology Language | The **Web Ontology Language** (**OWL**) is a family of knowledge representation languages for authoring ontologies endorsed by the World Wide Web Consortium. |
| **P2P** | Peer-To-Peer | A **peer-to-peer**, commonly abbreviated to **P2P**, is any distributed network architecture composed of participants that make a portion of their resources (such as processing power, disk storage or network bandwidth) directly available to other network participants, without the need for central coordination instances (such as servers or stable hosts). |
| **PHP** | Hypertext Preprocessor | **PHP: Hypertext Preprocessor** is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. |
| **PKI** | Public Key Infrastructure | **Public Key Infrastructure** (**PKI**) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates |
| **PNG** | Portable Network Graphics | **Portable Network Graphics** (**PNG**) is a bitmapped image format that employs lossless data compression. |
| **QoS** | Quality Of Service | Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. |
| **RDF** | Resource Description Framework | The **Resource Description Framework (RDF)** is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. |
| **REST** | Representational State Transfer | **Representational State Transfer** (**REST**) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. |
| **RIA** | Rich Internet Applications | **Rich Internet Applications** (**RIA**s) are web applications that have many of the characteristics of desktop applications, typically delivered either by way of a site-specific browser, |

| | | |
|---|---|---|
| | | via a browser plug-in, or independently via sandboxes or virtual machines. |
| **RUE** | Rich User Experience Pattern | A basic pattern highly recommended for all applications in Web 2.0 to have a better user experience for user. |
| **SaaS** | Software As A Service | **Software as a service** is software that is deployed over the internet. With SaaS, a provider licenses an application to customers as a service on demand, through a subscription or a "pay-as-you-go" model. |
| **SDK** | Software Development Kit | A **software development kit** (**SDK** or "**devkit**") is typically a set of development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform. |
| **SMS** | Short Message Service | **Short Message Service** (**SMS**) is a communication service component of the GSM mobile communication system, using standardized communications protocols that allow the exchange of short text messages between mobile phone devices. |
| **SOA** | Service-Oriented Architecture | In computing, a **service-oriented architecture** (**SOA**) is a flexible set of design principles used during the phases of systems development and integration. A deployed SOA-based architecture will provide a loosely-integrated suite of *services* that can be used within multiple business domains. |
| **SOAP** | Simple Object Access Protocol | **SOAP**, originally defined as **Simple Object Access Protocol**, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. |
| **SVG** | Scalable Vector Graphics | **Scalable Vector Graphics** (**SVG**) is a family of specifications of an XML-based file format for describing two-dimensional vector graphics, both static and dynamic. |
| **SWRL** | Semantic Web Rule Language | **SWRL** (**S**emantic **W**eb **R**ule **L**anguage) is a proposal for a Semantic Web rules-language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language |
| **TLS** | Transport Layer Security | **Transport Layer Security** (**TLS**) and its predecessor, **Secure Socket Layer** (**SSL**), are cryptographic protocols that provide security for communications over networks such as the Internet. |
| **TTP** | Trusted Third Parties | In cryptography, a **trusted third party (TTP)** is an entity which facilitates interactions between two parties who both trust the third party; The Third Party reviews all critical transaction communications between the parties, based on the ease of creating fraudulent digital content. |

| | | |
|---|---|---|
| **UDP** | User Datagram Protocol | The **User Datagram Protocol** (**UDP**) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. |
| **UML** | Unified Modeling Language | **Unified Modeling Language** (**UML**) is a standardized general-purpose modeling language in the field of software engineering. |
| **URI** | Uniform Resource Identifier | **Uniform Resource Identifier** (**URI**) is a string of characters used to identify a name or a resource on the Internet. |
| **WHATWG** | Web Hypertext Application Technology Working Group | The **Web Hypertext Application Technology Working Group** (**WHATWG**) is a community of people interested in evolving HTML and related technologies. |
| **WML** | Wireless Markup Language | **Wireless Markup Language**, based on XML, is a markup language intended for devices that implement the Wireless Application Protocol(WAP) specification, such as mobile phones, and preceded the use of other markup languages now used with WAP, such as HTML/XHTML(which are gaining in popularity as processing power in mobile devices increases). |
| **WS** | Web Services | **Web services** are typically application programming interfaces (API) or **web APIs** that are accessed via Hypertext Transfer Protocol and executed on a remote system hosting the requested services. |
| **WSDL** | Web Services Description Language | The **Web Services Description Language** is an XML-based language that provides a model for describing Web services. |
| **WS-RX** | Web Services Reliable Exchange | |
| **WS-S** | Web Services-Security | Security definitions required to access a web service |
| **WS-SX** | Web Services Secure Exchange | |
| **WWW** | World Wide Web | The **World Wide Web**, abbreviated as **WWW** and commonly known as **the Web**, is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them by using hyperlinks. |
| **XBAP** | XAML Browser Applications | **XAML Browser Applications** are Windows Presentation Foundation (.xbap) applications that are hosted and run inside a web browser such as Firefox or Internet Explorer. |
| **XDM** | XML Document Management | |
| **XHTML** | Extensible Hypertext Markup Language | **XHTML (Extensible Hypertext Markup Language)** is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written. |
| **XML** | Extensible Markup Language | **XML** (**Extensible Markup Language**) is a set of rules for encoding documents electronically. It is defined in the XML |

| | | |
|---|---|---|
| | | 1.0 Specification produced by theW3C, and several other related specifications, all gratis open standards. |
| **XSLT** | Extensible Stylesheet Language Transformations | **XSLT** (**Extensible Stylesheet Language Transformations**) is a declarative, XML-based language used for the transformation of XML documents into other XML documents. |
| **XSS** | Cross-Site Scripting | **Cross-site scripting** (**XSS**) is a type of computer security vulnerability typically found in web applications that enables malicious attackers to inject client-side script into web pages viewed by other users. |

# References


[1] *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, Tim O'Reilly, O'Reilly Media, Communications & Strategies, no. 65, 1st quarter 2007, p. 17.

[2] Joe Drumgoole blog, May 2006, available from http://joedrumgoole.com/blog/2006/05/29/web-20-vs-web-10/

[3] T. Andrew Yang, Dan J. Kim, Vishal Dhalwani, Tri K. Vu, "The 8C Framework as a Reference Model for Collaborative Value Webs in the Context of Web 2.0," hicss, pp.319, Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 2008

[4] *Semantic Web*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/Semantic_Web

[5] Paul Anderson, "*What is Web 2.0? Ideas, technologies and implications for Education*", JISC Technology and Standards Watch, Feb. 2007

[6] "*Intel Atom Developer Program with Adobe AIR*", available from http://appdeveloper.intel.com/en-us/develop

[7] Introducing Ajax and Open Ajax, *OpenAjax Alliance whitepaper*, available from http://www.openajax.org/whitepaper.html

[8] *SOAP*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/SOAP

[9] *REST*, Wikipedia, March 2010, available at http://en.wikipedia.org/wiki/Representational_State_Transfer

[10] Chirag Mehta, "*Lecture notes on Semantic Web*", SCU, available from http://www.cse.scu.edu/~cmehta/ws_references.html

[11] *XFN*, Wikipedia, April 2010, available at http://en.wikipedia.org/wiki/XHTML_Friends_Network

[12] *FOAF*, Wikipedia, March 2010, available at http://en.wikipedia.org/wiki/FOAF

[13] *Web API*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/Web_api

[14] Reference Model for Service Oriented Architecture 1.0, OASIS Standard, 12 October 2006, available at http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html

[15] James Governor, Dion Hinchcliffe, Duane Nickull, *Web 2.0 Architectures*, First Edition, O'Reilly Media, May 2009, Chapter 1, 4, 5, 6, 7

[16] Thomas Hillenbrand, "Cars on Web 2.0", May 2010, available from http://www.spiegel.de/auto/fahrkultur/0,1518,695366,00.html

[17] "*Internet usage statistics*", available from http://www.internetworldstats.com/stats.htm,

[18] "*Million dollar homepage*", available from http://www.milliondollarhomepage.com/faq.php

[19] *SOA*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/Service-oriented_architecture

[20] "*Google IG Personalized home page*", http://www.google.com/ig

[21] David Linthicum, *"SOA & WOA: Article, SOA World - Approaching SOA Testing"*, August 2007, available from http://soa.sys-con.com/node/417750

[22] Ronald Bhuleskar, Anoop Sherlekar, Anala Pandit, "Hybrid Spam E-mail Filtering," cicsyn, pp.302-307, 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, 2009

[23] *Mobile Web*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/Mobile_Web

[24] Franklin Reynolds, "Web 2.0-In Your Hand," *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 86-88, Jan.-Mar. 2009

[25] Timo Koskela, Nonna Kostamo, Otso Kassinen, Juuso Ohtonen, Mika Ylianttila, "Towards Context-Aware Mobile Web 2.0 Service Architecture," ubicomm, pp.41-48, International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'07), 2007

[26] Anuraj Ennai, Siddhartha Bose, "MobileSOA: A Service Oriented Web 2.0 Framework for Context-Aware, Lightweight and Flexible Mobile Applications," edocw, pp.345-352, 2008 12th Enterprise Distributed Object Computing Conference Workshops, 2008

[27] Lawton, G. 2007. Web 2.0 Creates Security Challenges. Computer 40, 10 (Oct. 2007), 13-16.

[28] Davidson, M.A.; Yoran, E.; , "Enterprise Security for Web 2.0," Computer , vol.40, no.11, pp.117-119, Nov. 2007

[29] Quasthoff, M.; Sack, H.; Meinel, C.; , "Who Reads and Writes the Social Web? A Security Architecture for Web 2.0 Applications," Internet and Web Applications and Services, 2008. ICIW '08. Third International Conference on, vol., no., pp.576-582, 8-13 June 2008

[30] *OpenID*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/OpenID

[31] *Windows CardSpace*, Wikipedia, January 2010, available at http://en.wikipedia.org/wiki/Windows_CardSpace

[32] "*Facebook Statistics*", http://www.facebook.com/press/info.php?statistics

[33] "*Facebook Wall*", available from http://www.facebook.com/help/?page=820

[34] Justin Osofsky, "*After f8: Personalized Social Plugins Now on 100,000+ Sites*", May 2010, available from http://developers.facebook.com/blog/post/382

[35] "*Facebook Social plugins*", available from http://developers.facebook.com/plugins

[36] Rebecca Chen, Victor Shen, Tony Wrobel, Christina Lin, "Applying SOA and Web 2.0 to Telecom: Legacy and IMS Next-Generation Architectures," icebe, pp.374-379, 2008 IEEE International Conference on e-Business Engineering, 2008

[37] *IMS*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

[38] *Parlay X*, Wikipedia, May 2010, available at http://en.wikipedia.org/wiki/Parlay_X

[39] "*WebSphere Telecom Web Services Server*", available from http://www-01.ibm.com/software/pervasive/serviceserver/

[40] Malik Muhammad Imran Pattal, Yuan Li, Jianqiu Zeng, "Web 3.0: A Real Personal Web! More Opportunities and More Threats," ngmast, pp.125-128, 2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies, 2009

[41] "*Resource Description Framework*", March 2010, available from http://www.w3.org/RDF/

[42] "*SPARQL Query Language for RDF*", W3C Recommendation, 15 January 2008, available from http://www.w3.org/TR/rdf-sparql-query/