2.4 development(s) – mashup semantic(s)

The old media have a hard time to catch up with the new media. While TV still may be considered a mass-medium, it seems to be loosing ground to online games and, indeed, *youtube.com*. In a panel of experts, gathered to discuss the notion of crossmedia, all agreed that the development(s) commonly referred to as *web 2.0* are here to stay:

web 2.0

video sharing / online gaming / social networking

Not only do these application areas appeal to the user(s), but moreover they seem to be fruitful from an entrepeneurial perspective as well. In other words, there is money in it!

The spirit of the shift of culture that characterizes these developments is well expressed in the following poem/rap from a local group, called *daft punk*:

daft punk – technologic

Buy it, use it, break it, fix it.
Trash it, change it, melt – upgrade it.
Change it, point it, zoom it, press it.
Snap it, work it, quick – erase it.
Write it, out it, paste it, save it.
Load it, check it, quick – rewrite it.
Plug it, play it, burn it, rip it.
Drag and drop it, zip – unzip it.
Look it, fill it, curl it, find it.
View it, coat it, jam – unlock it.
Surf it, scroll it, pose it, click it.
Cross it, crack it, twitch – update it.
Name it, rate it, tune it, print it.
Scan it, send it, fax – rename it.
Touch it, bring it. Pay it, watch it.
Turn it, leave it, stop – format it.

From a more objective perspective, we may observe that information has become a commodity, that is easily re-used, or put together in different combinations, for different purposes.

In an extremely well-readible article[1], entitled: *What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software*, Tim O'Reilly, ponders on the question(s), what makes these things work, and why are they profitable? When we look at many of these new applications or *mashups*, for example those using google maps, that these are:

mashup(s)

- substituting a single pragmatism for ideal design
- using light weight programming models

[1]www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

In other words, where the original visions of *hypertext* and *hypermedia* suffered from megalomaniac ambitions such as *boosting the human intellect*, many mashups simply provide a useful service or entertaining content. And in the same vein, where software engineering principles dominated the early hypermedia systems, the new mashups are often no more than a simple hack, exploiting existing services in a clever way. With great effect!

O'Reilly also sketches the shift that characterizes the underlying economic model of these development(s), that is the growth of the original web into the *web 2.0*, and beyond:

web 2.0 design pattern(s)

- web 1.0 – the web as platform
- web 2.0 – architecture of participation
- web 3.0 – data is the (intel) inside

The gist of these characterizations should be clear, service-oriented, and with a clear eye to the data that makes service(s) worthwhile, and profitable.

In a study, investigating how to use web services to enhance Second Life, Mashups, we wrote: *by now the phrase* web 2.0 *as well as applications representing it, such as Flickr and YouTube, are well established, and enjoyed by a wide community.* Each day new items are added to the growing list of mashups[2], and the number of web services that constitute the building blocks of mashups also shows a steady growth. Mashups seem to be the easy way to start up a company, since the technology is relatively easy and, making use of appropriate services, initial investment costs can be low. Cf. Amazon.

What *web 2.0* stands for, from a technical perspective, is succinctly expressed in Dorai's:

Learnlog[3]: *XML Is The Fabric Of Web 2.0 Applications*

- the client side is AJAX (Asynchronous Javascript and XML)
- the server application typically exposes data through XML
- the interaction model is web services
- mashups combine multiple webservices to create new types of applications

And eventhough many alternative representations, such as JSON[4] (Javascript Object Notation) are increasingly being used, all in all XML may be regarded as the *interlingua* of the Web 2.0.

Before taking a closer look at the communication protocol(s) underlying *web 2.0* and de-construct the tight link of AJAX to HTML in-page formatting, it is worthwhile, following Amazon, to give an overview of a selected number of services, that may be used to create mashups:

service(s)

- google – code.google.com/
- yahoo – developer.yahoo.com/

[2]www.programmableweb.com/mashuplist/
[3]dorai.wordpress.com/tag/mashups/
[4]www.json.org/

- del.icio.us – del.icio.us/help/api/
- flickr – www.flickr.com/services/
- bbc – www0.rdthdo.bbc.co.uk/services/
- youtube – www.youtube.com/dev

Although mashups featuring google maps seem to be the dominant mashup type, other services such as offered by del.ici.us, Flickr and BBC might prove to be more worthwhile for 'serious' applications. For example, for developing e-commerce applications Amazon[5] offers services for *product operations*, such as item search and similarity lookup, *remote shopping carts*, to create and manage purchase collections, *customer content*, to access information contributed by customers, and *third party listings*, to find related resellers. It is important to note that many of these services, as for example the *shoppong cart* services, may be used independently of the commercial offerings of Amazon!

Most of the service providers and services mentioned above are accessible using a choice of protocols, including WSDL, SOAP, XML-RPC and the REST protocol. The REST protocol seems to be most widespread and as we will discuss in the next section, it seems to be tho most appropriate protocol in Second Life.

REST stands for *Representational State Transfer*. In essence, the REST protocol uses the url as a command-line for stateless RPC invocations, which allows for services to be executed by typing in the address box of a web browser. A great tutorial about the REST protocol can be found in Joe Gregorio's column column[6]: *The Restful Web*. As fully explained in Web, the phrases *representation*, *state* and *transfer*, respectively, stand for:

REST[7]

- representation – encoding in a particular format
- state – data encapsulated in an object
- transfer – using HTTP methods

In practice, the use of REST means that the state associated with a resource or service must be managed by the client. Together with mechanisms such as content-negotiation and URL-rewriting, REST provides a simple, yet powerful method to invoke services using HTTP requests.

A common misunderstanding is that AJAX is intimately tied to web browsers and in-page HTML formatting. This misunderstanding is due to the fact that AJAX is often used to improve the user experience of web pages bij emulating RIA (Rich Internet Applications) using DHTML and CSS. However, the real meaning of AJAX in our view is that AJAX allows for asynchronous client-controlled server requests, that are executed without an immediate visible effect for the user.

The *web 2.0* offers a lively arena for consumers and developers alike, with a multitude of blogs discussing the future of the web. For example, in Dion Hinchcliffe rebuttal[8] of Jeffrey Zeldman's *Web 3.0 Ũ Web 1.0 = Web 2.0* blog,

[5]aws.amazon.com
[6]www.xml.com/pub/a/2004/12/01/restful-web.html
[7]www.xml.com/pub/a/2004/12/01/restful-web.html
[8]web2.sys-con.com/read/172417.htm

entitled *Is Web 2.0 Entering "The Trough of Disillusionment"?* it is suggested that *our services could even be more powerful* by creating *semantic mashups*[9]. Although the notion of *sematic web technology* is widely known and accepted, we include for reference a characterization of Nova Spivack quoted from Dan Farber and Larry Dignan's blog[10] *Web 2.0 isnŠt dead, but Web 3.0 is bubbling up*:

> The Semantic Web is a set of technologies which are designed to enable a particular vision for the future of the Web Ũ a future in which all knowledge exists on the Web in a format that software applications can understand and reason about. By making knowledge more accessible to software, software will essentially become able to understand knowledge, think about knowledge, and create new knowledge. In other words, software will be able to be more intelligent, not as intelligent as humans perhaps, but more intelligent than say, your word processor is today.

But even in the semantic web community the discussion whether to go for *folksonomies* or *formal ontologies* rages, Folk, and it is not clear at this stage what will prove to be more powerful, HTML-scraping, tags, microformats, or full ontologies.

Instead of joining this perhaps endless discussion, let us explore what is involved in incorporating web services in Second Life, and how to realize meaningful mashups in 3D virtual environments. Nevertheless, to conclude this brief overview of web services and mashups I wish to give another quote from Dorai's Learnlog, this time from Jon Udell, in his blog on his move to Microsoft:

> *the most powerful mashups don't just mix code and data, they mix cultures.*

which provides a challenge that trancends all issues of mere technological correctness.

using web services in Second Life Second Life offers an advanced scripting language with a C-like syntax and an extensive library of built-in functionality. Although is has support for objects, LSL (the Linden Scripting Language) is not object-oriented. Cf. OO. Scripts in Second Life are server-based, that is all scripts are executed at the server, to allow sharing between visitors. Characteristic for LSL are the notions of *state* and *eventhandler*, which react to events in the environments.

Among the built-in functions there are functions to connect to a (web) server, and obtain a response, in particular (with reference to their wiki page):

built-in(s)

- request – wiki.secondlife.com/wiki/LlHTTPRequest
- escape – wiki.secondlife.com/wiki/LlEscapeURL
- response – wiki.secondlife.com/wiki/Http_response

[9] www.web2journal.com/read/361294.htm
[10] blogs.zdnet.com/BTL/?p=3934

Other functions to connect to the world include *sensors*, for example to detect the presence of (visitors') avatars, and chat and instant messaging functions to communicate with other avatars using scripts. In addition, LSL offers functions to control the behavior and appearance of objects, including functions to make objects react to physical laws, to apply force to objects, to activate objects attached to an avatar (as for example the phantom Mario sprites mentioned earlier), and functions to animate textures, that can be used to present slide shows in Second Life.

On the Mashable[11] *Social Networking News* site a brief overview is given of the use of web services in Second Life, entitled *Second Life + Web 2.0 = Virtual World Mashups*. To access Second Life from outside-in (that is from a web browser), so-called *slurls* may be used, for example to reach VU[12] @ Second Life, and all slurls listed in del.icio.us under *slurlmarker*[13] may be used, also to activate in-world teleporting using scraping techniques.

As remarked in the *hackdiary*[14] by Matt Biddulph, Second Life (currently) lacks the ability to parse XML or JSON, so the best way to incorporate web services is to set up a web server with adequate resources. As Matt Biddulph indicates, to access *flickr* photographs for a particular user (avatar), a web server may contain the following resources:

resource(s)

- /seen?user=SomeAvatar – records the presence of SomeAvatar
- /touched?user=SomeAvatar – invokes flickr API with users tag
- /set_tag?user=SomeAvatar&tag=FavoriteTag – records SomeAvatar's favourite tag

For example, in response to a 'touch' event, invoking *touch* results in consulting the database for the user's tag and asking the Flickr API for a random photo with that tag. It then returns a string containing the url for a particular photograph. LSL functions used in this application include *sensors*, to check for presence, *listen* functions, to respond to spoken commands, and *touch* events, for the physical interface. In addition to supporting strings and lists, LSL provides a perl-like split function to convert a string into a list of strings, thus allowing for processing multiple items in response to a server request.

Another example of using web services in Second Life is writing blogs[15] from within Second Life using the BlogHUD[16] developed by Koz Farina who also is reported to have found a flash hack that allows for reading RSS feeds. As explained by Koz Farina:

flash/quicktime in SL

> Quicktime supports Flash, but only up to Flash version 5. We're up to version 9 on that now! Luckily, I have been dabbling with Flash since the

[11]mashable.com/2006/05/30/second-life-web-20-virtual-world-mashups/
[12]slurl.com/secondlife/VU%20University%20NL/29/151
[13]del.icio.us/tag/slurlmarker
[14]www.hackdiary.com/archives/000085.html
[15]nwn.blogs.com/nwn/2006/10/really_simple_s.html
[16]bloghud.com/

> early days, so already knew how to do this 'the old way'... So, Flash is doing all the work. No LSL at all... I heart feeds. Did I say 'I heart feeds?

The RSS display uses the ability to stream Quicktime video in Second Life, and again the mashup is not created in Second Life but by appropriate server support.

In a similar vein we may incorporate live streaming video[17], for example by using WireCast[18] to capture and organize live camera input, possibly together the screen output of other applications such as *powerpoint*, which must then be sent to a streaming server supporting Quicktime, such as Apple's Darwin[19], which may then be accessed from Second Life to texture a display object.

Finally, as another *Web 2.0 to Web 3D* phenomenon, announced in New World Notes[20], we may mention the used of Twitter[21] messages, that allow residents to send and receive message about ongoing activities. A similar service is reported to exist for *jaiku*[22] messages.

Referring to section 7.4 for a more detailed discussion, we may observe that there is no meaning in merely putting things together. Without mechanisms of personalization and recommendation we would simply be flooded by data and information, in a way that even search would not be able to cope with. Context, narratives and personalized presentation(s), notions from the past, reappear as keywords for the future of the *web 2.0* and beyond.

[17]blogs.electricsheepcompany.com/chris/?p=206
[18]www.varasoftware.com/products/wirecast/
[19]developer.apple.com/opensource/server/streaming/
[20]nwn.blogs.com/nwn/2007/03/post_1.html
[21]twitter.com/
[22]devku.org/docs