Mashups

ITEC810: Final Project Report

Student Name: Mathindri Pathiraja (41364465)

Supervisor's Name: Diego Molla-Aliod

Summary

Mashups are a current hype that is attracting high interest by academia and industry now and in the next years. The idea behind a mashup is to create new content by reusing and combining existing content from heterogeneous sources. Advantages of mashups are that even people with no knowledge of programming languages can easily build new Web applications and create new forms of visualizations. To support the mashup construction process several tools have been proposed with easy-to-use functionalities. However, from the research perspective it is dissatisfying that neither a clear definition and classification model for mashups nor a separation between mashups and other forms of application integrations exist. The aim of this paper is to elucidate the mashup hype by providing a definition as well as tools and languages supporting the mashup creation are presented.

1.0 Introduction

The idea behind the term mashup is not new. The integration of different resources is an issue usually faced during software development. In fact, most engineering methods take into account the fact that some data and functionality is provided by external systems and provide mechanisms to specify them properly.

The main reason why mashups are gaining tremendous popularity is that even non-technical people are able to create new content and representations of resources without much effort or knowledge of programming languages. Another advantage is that the execution of mashups is not performed by black-box systems (like in the service-oriented execution) but rather user-driven, which makes the resource integration process much more transparent in comparison to conventional application integration platforms.

When undertaking a literature review about mashups then the search results about methodologies or infrastructures for mashup constructions are disillusioning. The amount of comprehensible approaches proposed so far is minimal and no clear line exists between mashups and technologies such as Service-oriented Architecture (SOA). The confusion even starts when looking for a generic definition of mashups. The definitions proposed in the literature span technical, business (in sense of economic) or industry perspectives. The next confusion follows when searching for a classification model for mashups. Too many references [for instance see 16, 19, 9] have different perspectives on mashups. In particular, the work of Hoyer & Fischer [16] is similar to the work of this paper. But Hoyer & Fischer limit their focus on enterprise mashups, proposes a more specific classification model and does neither discuss challenges for mashups nor investigate a guide for a methodical construction of mashups.

The intention of this paper is to provide a generic definition and a classification model for mashups. Furthermore, we aim at discussing challenges of mashups and to present methodical concerns

that should be considered when developing mashups. Finally, our paper presents an excerpt of tools and languages supporting the creation of mashups.

The structure of the paper is as follows. The next section discusses the different definitions for mashups and points out one clear definition. Section 2 describes a classification model for mashups. An excerpt of mashup tools is given in Section 3. In Section 4 challenges are discussed that have to be faced with mashups. The last section summarizes our work and motivates the need for more research on mashups.

1.1 Definitions

Over the past decade, computer applications have become more comprehensive, and the Internet has become more accessible – to users and, increasingly, to amateur programmers. More and more, the Web is being used not only as a portal for information but also for application-to-application communication. Web services is the general term for the interfaces available that connect different applications to each other, or any technology that supports machine-to-machine interaction over a network.[1] Web services were developed as a mechanism for connecting otherwise un-interoperable systems, and they can make it much easier for applications to communicate,[2] especially with XML to code and decode the data and SOAP (Simple Object Access Protocol) to transport it using open protocols. The term "Web services" as we use it, however, encompasses more than these messaging technologies, including application-specific programming interfaces and any other software capabilities that facilitate data transfer between applications on the Web. This set of technologies provides an interoperable framework for communications between applications, but the overall interoperability depends greatly on the applications and the data exposed to them.[3]

Web 2.0

In the aftermath of the Web 1.0 crash, the glut of infrastructure kept the costs of going online low. That simple fact helped attract even more users to come online. A few companies began to figure out how to leverage the Web without going bankrupt. Collectively, their embrace of the Internet represented the slow expansion of the Web from that last primordial blast. New marketplaces evolved as sites like eBay linked buyers and sellers from around the globe. These online flea markets, in turn, spawned communities that helped pioneer the concepts behind new social networking sites like MySpace and Facebook. By 2006, the firms that had simultaneously feared and tried to control Web 1.0 looked up from licking their wounds and saw the dawn of a new paradigm. In a symbolic changing of the

[1] See, e.g., World Wide Web Consortium, Web Service Definition Language, http://www.w3.org/TR/wsdl (last updated 15 March 2001); Wikipedia, Web service, http://en.wikipedia.org/wiki/Web_service (last visited 20/03/2009).

[2] W3Schools, Web Services Tutorial, http://www.w3schools.com/webservices/default.asp (last visited 31/04/2009).

[3] Paul Anderson, What is Web 2.0? Ideas, technologies, and implications for education, http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf, at 25; Björn Hartmann et al., Hacking, Mashing, Gluing: A Study of Opportunistic Design and Development, http://hci.stanford.edu/publications/2006/MashUps-TR.pdf, at 4; Tim O'Reilly, What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, at 3.

guard, "old media" giant *Time* magazine announced the Person of the Year was "You."[4] There was no great single occurrence that made this milestone possible. Rather, the driving force was the confluence of many events: the spread of cheap broadband access, the Web enabling of multiple devices, the arrival of new communication environments, and the emergence of cooperative environments for organizing information. Collaborators were finally running the show.

Industry figurehead Tim O'Reilly is credited with popularizing the term "Web 2.0" to define this new age: *Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform.*[5]

A simpler working definition is that Web 2.0 is a shift from *transactionbased* Web pages to *interaction-based* ones. This is how the power of "You" is mashed, mixed, and multiplied to create value. Social-networking sites, folksonomies (collaborative tagging, social bookmarking), wikis, blogs, and mashups are just some of the components that make this possible. The success of sites such as Facebook, wikipedia, flikr, and digg has demonstrated that democratization of content creation and manipulation is powering the latest wave of Internet growth. The underlying driver of Web 2.0 is flexibility. The one trait technologies slapped with the Web 2.0 moniker share is that they are extremely (and perhaps sometimes unintentionally) malleable. The successful products don't break when a user tries to extend them beyond their original design; they bend to accept new uses. Two success stories of the new Web illustrate this principle:

flickr was started by Caterina Fake and Stewart Butterfield as an add-on feature for a video game they were developing. The idea was to allow players to save and share photos during gameplay. When they realized that bloggers needed a convenient way to store and share photos, Fake and Butterfield started adding blog-friendly features. Opening up their architecture to allow users of the site to create custom enhancements fueled their viral spread. The original game was ultimately shelved and flickr was sold to Yahoo! a year later for an undisclosed sum.
Deli.cio.us grew from a simple text file that its founder, Joshua Schachter, used to keep track of his personal collection of tens of thousands of Web site links. When the site went public in 2003, it spawned a host of add-ons. The concept of associating data with simple keywords to aid in organization wasn't new, but the cooperative "social tagging" aspect of deli.cio.us resonated with the frustrations of other Internet users.

Enterprise 2.0

The Web has been a staple of IT departments for more than a decade. Intranets, extranets and Internet sites are common for companies of all sizes, in all industries and in all geographies. Similar to the first wave of enterprise Web investment, much of the inspiration for functionality is being driven by the consumer Web culture.

Traditional Web sites continue to be the mainstay of most enterprise user interface (UI) strategies. The ubiquity of the Web browser makes it the most appropriate tool for interacting with customers, trading partners and employees. Although the Web enjoys a significant presence in enterprises, there are still many opportunities for improvement, including better processes and tools for content management, deployment and the management of international, decentralized and multilingual Web sites, as well as Web sites for handicapped users. Infusing traditional Web architectures with rich

[4] *Time* magazine, December 13, 2006.
[5] http://radar.oreilly.com/archives/2006/12/web-20-compact-definition-tryi.html (Last accessed 01/06/2009)

Internet application technologies, including Ajax, is becoming increasingly common for employee-facing Web environments, and is a best practice for customer-facing environments.

1.2 The Origin of Mashups

"Defiant Downloads Rise from Underground" headlined an article in the New York Times on February 25, 2004, describing how hundreds of websites and blogs had united in "a day of coordinated civil disobedience." Under protest was record label EMI's crackdown on the free and online distribution of The Grey Album that had become a flashpoint for debates on copyright and authorship. The work catalysed controversy because its artist, a DJ named Brian Burton, who goes by the stage name Danger Mouse, had combined vocals from rapper Jay-Z's Black Album with instrumentation from the Beatles' White Album. Although the work was never intended for commercial release, its circulation online was so widespread that it came to the attention of EMI, owner of the rights to the Beatles' album. Seeing the album as a case of copyright infringement, EMI took legal action against the artist and its major distribution outlets on the web. For the protesting groups, EMI's copyright claims were irrelevant because they considered the album to be, in fact, a new work known as a *mashup*, creatively reassembled from elements of pre-existing works. However, the debate around copyright and authorship in this case was not strictly two sided because Jay-Z's label, Roc-A-Fella, released an *a cappella* version of the album's vocal tracks to DJs for the express purpose of remixing, complicating the relationship between mashups author and the creator of the original work and suggesting that perhaps this relationship can be mutually beneficial. While not the first artist to make mashups,[6] nor the first to face legal consequences,[7] Danger Mouse's situation gained significant media coverage and widespread support on the internet.

In the coordinated, distributed protests online—known as "Grey Tuesday"—websites attempted to spread digital copies of the album as widely as possible. While never released commercially, this "illegal" album became a cultural phenomenon (at least among those active on the internet or fans of hip hop and DJ music), later ranked by Entertainment Weekly as the best album of the 2004. With "The Grey Album", the term "**Mashup**" entered the public consciousness.

The question of ownership, authorship and agency have resurfaced in recent years, as an explosion of World Wide Web mashups have appeared, in which users either mix data from various web sites or integrate their own data into existing webpages. Among the most popular web mashups are

6 Despite the fame and controversy surrounding the "Grey Album", it is important to note that it is only one of many examples of the mashup form, ranging from remixes by contemporary artists such as 2 Many DJs, The Kleptones, and Girl Talk, and descended from earlier examples of artists such as Negativland and John Oswald's Plunderphonics. While in these works the practice of "sampling" (reusing snippets of existing songs) sits front and center, the use of sampling as an instrumental backing to vocals has had a long tradition in Hip Hop. This sampling culture can be traced to Jamaican Dub music pioneered by King Tubby and Lee "Scratch" Perry, who used remixes of existing recordings to create "versions" of songs. Beyond the field of music, examples of sampling and remixing can be found in William S. Burroughs' literary "cutups," in the collage art and poetry of the Dadaists and the "found art" of Marcel Duchamp, and the détournement of the Situationists.

7 In 1991 rapper Biz Markie was sued (in "Grand Upright Music, Ltd v. Warner Bros. Records, Inc.") for sampling the song "Alone Again (Naturally)" by Gilbert O'Sullivan. The resulting injunction radically limited the use of sampling in mainstream Hip Hop, although underground sampling and mashups continued.

those based on the mapping services of Google, Yahoo! and Microsoft. These user-created web maps cover all manner of subjects, including personal daily experiences and travelogues (such as annotating a map with photos taken at that location); real estate and shopping guides (such as mashing classified ads onto a map); and political and community activism (such as mapping reported crimes in the neighbourhood).

Strictly speaking, mashups online are hybrid web applications, combining disparate data sources and web services in ways that are not how they were intended. This definition, which mirrors the way the term "mashup" is used in music, appears to cover a range of online mapping activity, but leaves other kinds of online mapping outside the category of mashups. However, due to the newness of the field, the usage of these terms remains in flux, and common usage tends to employ the word "mashup" to describe any use of online mapping services, or sometimes only to refer to the use of Google Maps. One of the purposes of this paper is to disentangle the confusion of terminology, and, to the extent possible, retain a narrow definition of what constitutes a mashup in order to better understand the implications of this specific practice of remixing and recombination. To begin this inquiry, I will begin with a description of two of the earliest and best-known mashups, HousingMaps and Chicago Crime.

1.3 The First Mashups

In July 2005, Google released an API (or Application Programming Interface) for its Google Maps service, effectively unveiling a blank geographic canvas on which web programmers could begin to paint. As a detailed description of all the functions and instructions of the software behind the service, the Google Maps API provided developers with the means and opportunity to appropriate its data, interface, and iconography, which included among other things its pushpin icons, text balloons, as well as its zooming and scrolling frames and functions. Users could, therefore, generate new maps without having to write or host the mapping software themselves. Yahoo! followed Google's lead and released its mapping service API a day later, arguably inaugurating the era of mashups, in which customized mapping became available to anyone on the internet. Significantly, the first true mashups had in fact emerged months *before* this announcement, created by savvy programmers who had managed to decode parts of the then unpublished but still accessible APIs. In response to these initial mashup activities, Google Maps product manager Bret Taylor explained that his company had ultimately published the API "because they were already doing it" (Singel 2005) [23].

One of the earliest Mashups was Paul Rademacher's HousingMaps.com, a combination of apartment rentals and houses for sale extracted from the online classifieds site craigslist.org, and displayed on Google Maps. On this site, housing ads are enriched through geo-referencing, both in terms of pragmatic utility (helping users find accommodations in a given area more easily) but also in terms of the demographic insight which emerges from the large dataset on craigslist that documents a spatially significant human activity, namely the processes involved in the acquisition of housing. HousingMaps was simple to create and instantly filled a need not provided by the original data source craigslist, exemplifying a new paradigm called Web 2.0 by Tim O'Reilly, a leading publisher of technical books and a trend-spotter in the computer industry. For O'Reilly, Web 2.0 refers to new interoperable, interlocking type of services, where web sites would provide components (either data itself or data processing services) rather than finite, one-stop experiences. Users in this paradigm would, therefore, be free to combine online services in any way they choose. "Google Maps with craigslist is the first Web

2.0 application," O'Reilly explained at his 10 geospatial Where 2.0 conference in 2005 (Singel 2005)[23]. In this way, HousingMaps may yet foreshadow the way users interact with the web in the coming years.

Adrian Holovaty's Chicago Crime[8] was, along with HousingMaps, perhaps the best known of the early mashups. The site displays crime reports from a feed available on the Chicago Police Department's web page and geo-coded the reports onto a Google Map. Later, the site moved beyond simply identifying crime locations with coloured pushpins and began linking the locations with stories from the Chicago Journal's police blotter, widening representations of space beyond plotted points to include narrative. The site received significant press coverage, winning an award for journalism for what one judge called "a pioneering integration of geo-mapping and a public database,[that] delivers one of the most comprehensive crime sites online." [9] Both mashups originally relied on screen-scraped data from websites and were developed before the release of Google's mapping API, meaning the points were plotted by reverse engineering Google's javascript. As such, despite their mainstream accolades, they relied on a hacker mindset. The need for a hacker's skill set also meant that creating these mashups was an elite venture, only possible for those with programming knowledge.

1.4 Terminology

In fact, these early Mashups were also called "map hacks" or "mapping hacks,"[10] terms that were already in use in 2004 (the year before Housing Maps and Chicago Crime were released) by programmers Schuyler Erle, Rich Gibson and Jo Walsh in the website[11] for their book *Mapping Hacks* (subsequently published in 2005). [12] These map hacks ranged from personal mapping of one's own GPS tracks to tricks for geo-coding addresses; some of the map hacks could be considered mashups in that they plotted data feeds from other sources, such as earthquakes re-ported by the USGS or health code violations, but usually using their own simple base maps, rather than mashing the data onto existing maps such as Google's.

While *Mapping Hacks* doesn't mention the word "mashups," a subsequent book by Gibson and Erle (2006) [10], clearly defines mashups as a subset of mapping hacks, involving, as in music, combining two existing data sources, such as a news feed and a web map. The remainder of the book is devoted to mapping one's own data, or describing advanced techniques for dealing with Google Maps as a user, none of which are described as mashups. This continues with the strict definition of mashup: it must combine two or more existing data sources, even if one is a base map (Google Maps, in this case) and one is a "content" layer (such as the data scraped from Craigslist or the Chicago Police Department). Mashup creators pull data dynamically from one source and integrate it with another. The trend toward

[8] On January 31, 2008, Holovaty shut down the original Chicago Crime mashup, redirecting it to EveryBlock, a new project with expanded functionality and coverage of more cities. http://www.everyblock.com

[9] 2005 Batten Award Winners: http://www.j-lab.org/batten05winners.shtml (accessed March 22, 2009)

[10] On the original website of HousingMaps, Paul Rademacher called his project a "Craigslist-GoogleMaps combo site", avoiding the emotional connotations of "hack" or "mashup."

[11] http://www.mappinghacks.com

mashups is already visible in other domains. In web terminology, a *mashup* is a web site that combines ("mashes up") content from more than one source (from multiple web sites) into an integrated experience. Mashups are content aggregates that leverage the power of the Web to support worldwide sharing of content that conventionally would not have been easily accessible or reusable in different contexts or from different locations.

Bloggers and mainstream news sources have also shown inconsistent and shifting use of terminology. Mike Pegg's blog Google Maps Mania[12] has tracked the Google Maps hacking phenomenon since it began, becoming the preeminent index of Google-based mashups. Pegg's post switched from using "Google Maps hack" to "mashup" around June 2005, and today employs a loose classification system of "websites, mashups and tools being influenced by Google Maps". Another influential aggregator of information about mashups is ProgrammableWeb,[13] which, Through its focus on exhaustively indexing all available APIs (not only those of the major mapping sites, but also blogging, social networking, e-commerce APIs, and others) maintains a consistent use of "mashup" as a "web page or application that combines data from two or more external online sources." However, ProgrammableWeb's archive includes numerous examples of sites that only use the Google Maps API, not mashing it with APIs or feeds from any other sites. In a recent article, Jack Dangermond (2008) [2], CEO of GIS software company ESRI, makes a distinction between "mashups" (combining web services) and "consumer map visualization" (that is, using a map API to plot your own data on a map; Dangermond is careful not to call this "map making"). In this sense, the projects on ProgrammableWeb that use no service other than Google Maps would not fit Dangermond's definition of a mashup (nor, strictly, ProgrammableWeb's own definition).

Goodchild (2007b) [14] also followed the increasingly common usage where in a mashup is considered any overlay of data (whether sourced from another web service, or contributed by the user) on a web map, but in a later article he expresses more appropriate use of the word: "The term mashup has become the preferred way of referring to the linking of Web services through common references, and geo-references are clearly one of the most powerful and ubiquitous bases for what is essentially a generalization of the concept of a relational join" (Goodchild 2008) [15].

Anthony Bradley and David Gootzit have defined Mashups in H*ype Cycle for Web and User Interaction Technologies, 2008* Gartner Publications as follows [4]:

Definition: *"A "Mashup" is a lightweight, tactical presentation layer integration of multisourced applications or content into a single, browser-compatible offering. It is a lightweight variant of the older notion of a composite application ("composite app") and the heavier service-oriented architecture orchestration approach to composite applications. In the usual use of the term, composite apps are built on enterprise platforms, internal facing and not necessarily Web-based. A mashup is a Web-based composite application that leverages systems (data, business logic and presentation) to create a new capability."*

As you can see there are lot of definitions in the Internet defining Mashups and there is no official definition of Mashups available. We regard these definitions as too tight since it does not take

[12] http://googlemapsmania.blogspot.com

[13] http://www.programmableweb.com

into account, for example, aspects such as source heterogeneity. In fact, the integration/combination of different sources is not limited to data but also to functionality and layout styles. Therefore, in order to provide a more precise and complete definition of this term we have extracted the most relevant terms included in several definitions found in the Internet. These terms include *Web page or application, integration, combination, reuse, data sources, APIs, third party data, Web 2.0 and data processing.*

Taking into account these terms we propose to define a Mashup as a Web-based application that is created by combining and processing on-line third party resources, which contribute with data, presentation or functionality. It is important to note that in this definition, on-line third party resources refer to any type of resource available in the Internet, independently of the format in which this is provided (by means of an API, Web Feeds or screen scraping techniques). As a result, a Mashup provides a new resource not conceived by the original combined resources. Finally, based on the nature of the original combined resources Mashups can be categorized in different groups as explained later. I will call this the "**narrow**" definition of mashups.

1.5 The Web 2.0 & Mashups:

As we know by now, Web 2.0 continues to be one of the most hyped areas of the Web, so a backlash is likely. It is also an area with the most opportunities for enterprises. Web 2.0 is a collection of methodologies, technologies, and social and business models highlighted by openness, participation, the use of lightweight technologies (many times, using open-source software), and decentralized, distributed processes. Web 2.0 applications (such as blogs, wikis, social networks and Mashups) have enjoyed great success on the public Internet, and are seeing early adoption among Type A enterprises. Methodologies such as REST/POX and Ajax not only have seen good adoption by enterprises, but are also enjoying the focus of the ISV community, with many mashup servers, toolkits, enablers and Ajax toolkits on the market. Unfortunately, the hype surrounding Web 2.0 can lead enterprises to focus on the technologies of Web 2.0, rather than on the community, business and social implications — where true breakthrough value can be realized.

According to Gartner Emerging Technologies Hype Circle 2008 research, *"Web 2.0 is on its way toward the Trough of Disillusionment, and a backlash should be expected. Some enterprise applications of Web 2.0 concepts, including mashup applications and social software suites, are coming up on the Peak of Inflated expectations. Although some interest around Web 3.0 has started to garner attention, we believe that, as a concept, Web 3.0 is a false start." (Figure 1)*

As per Gartner's Priority Matrix**:** Many Web-oriented technologies and methodologies will see mainstream adoption soon. Some of the more-impactful items include enterprise portals, Web 2.0 and Mashup applications. Following matrix will show a clear picture on where Mashup and their applications stand on the Technology Hype Circle as for July 2008. (Figure 2). We can see that Mashup and it's applications are rated as high beneficial to industry at the moment.

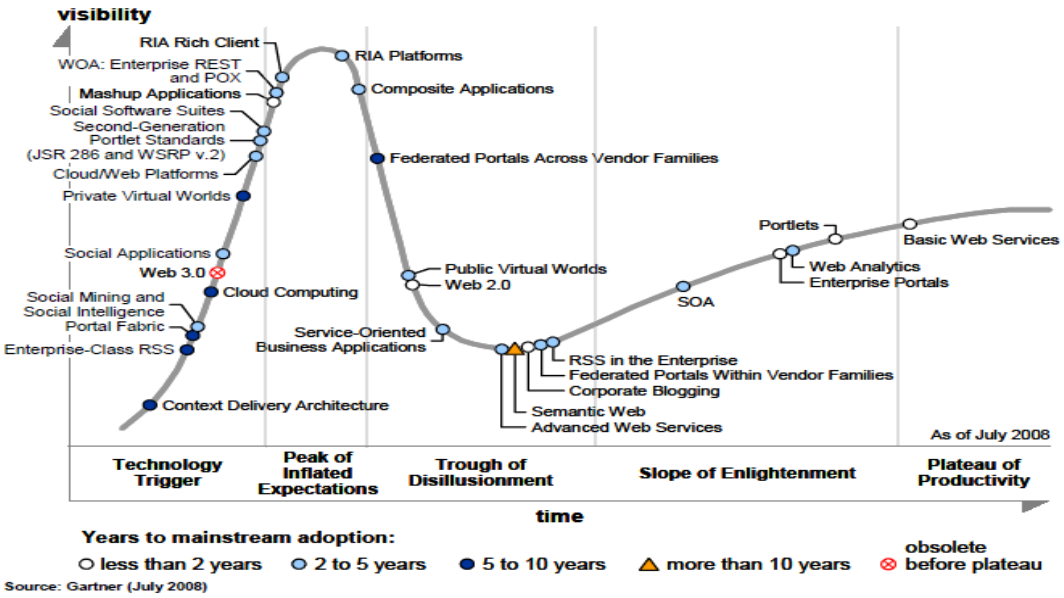


Figure1: Hype Cycle for Web and User Interaction on Technologies, 2008

benefit	years to mainstream adoption			
	less than 2 years	2 to 5 years	5 to 10 years	more than 10 years
transformational	Enterprise Portals Web 2.0	Cloud/Web Platforms Public Virtual Worlds SOA Social Applications	Cloud Computing Context Delivery Architecture	
high	Mashup Applications Portlets	Composite Applications Service-Oriented Business Applications Social Mining and Social Intelligence Social Software Suites WOA: Enterprise REST and POX	Federated Portals Across Vendor Families Portal Fabric Private Virtual Worlds	Semantic Web
moderate	Basic Web Services Corporate Blogging	Advanced Web Services Federated Portals Within Vendor Families RIA Platforms RIA Rich Client RSS in the Enterprise Second-Generation Portlet Standards (JSR 286 and WSRP v.2) Web Analytics	Enterprise-Class RSS	
low				

As of July 2008

Source: Gartner (July 2008)

Figure 2: Priority Matrix for Web and User Interaction Technologies, 2008

1.5.1 Why Mashups are Popular?

The Web 2.0 bandwagon is an important reason why Mashups are popular now. Mashups have been identified explicitly (under the phrases “re-mixable data source” and “the right to remix”) by Tim O’Reilly in “What is Web 2.0?”[14]Added to this, we have the development of what might be accurately thought of as “Web 2.0 technologies/mind-sets” to remix/reuse data, web services, and micro-applications to create hybrid applications. Recent developments bring us closer to enabling users to recombine digital content and services:

Enterprises are beginning to transform Mashups from Web entertainment to enterprise relevance. Even in the enterprise, Mashups leverage data and services from Web sites, such as Google Maps, craigslist, eBay, Amazon.com and others. Because Mashups leverage content and logic from other Web sites and Web applications, they're lightweight in implementation and are built with a minimal amount of code (which can be client-side JavaScript or server-side scripting languages, such as PHP or Python). These are not fixed requirements, but they reflect the original implementation of the mashup concept in Web 2.0 startup companies, which typically do not use enterprise-oriented platforms, such as Java or .NET.

Mashups exploit lightweight mechanisms, such as representational state transfer-based application programming interfaces (APIs), to Web services, as well as Ajax "snippets" and “widgets”. Mashups aren't intended to be strategic, systematically built, industrial-strength enterprise applications; rather, they're created quickly or opportunistically to meet a focused tactical need.

Mashups generally are personalized to fulfil personal productivity needs rather than the requirements of a long-standing corporate role. The cultural context of mashups involves the confluence of many innovations: Web APIs, lightweight client-side scripting, delivery of content via Really Simple Syndication (RSS), wikis, Ajax, social networking and the explosion of Web based communities. For a long time, the closest thing to mashup creation tools for "civilians" (users who do not write code) was an RSS feed reader or podcasting client, which enabled them to "mash" content from more than one site. This situation has improved, with more-powerful tools such as Yahoo Pipes and Microsoft Popfly.

Position and Adoption Speed Justification: Mashups are driven by the Web culture. There are thousands of mashups on the Web, often built by nonprofessional programmers. For example, HousingMaps.com combines data from Google Maps with apartment rental information from craigslist to create a new application that shows the location of available apartments in a given city — all ccomplished without the direct participation of Google or craigslist staffs.

In addition, Web sites such as Facebook, Pageflakes, Netvibes and iGoogle have enabled thousands of nontechnical users to build mashups using prebuilt mashable assets (for example, gadgets and widgets). Web APIs and mashups are available in an online directory at ProgrammableWeb**.** In keeping with the consumerization trend, the popularity of mashups on the Web is driving enterprise interest and experimentation in how mashups deliver value inside the firewall. Although this Web mashup movement is significant, it's importance should not be overstated to enterprises, because the vast

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

majority of Web mashups is more for entertainment value rather than business value. Greater business value and the realization of mashup hype are growing as more enterprise implementations mature.

User Advice: Mashups quickly integrate content or functions from multiple sources and present easily understandable items of interest. The trade-off is faster time to market and reduced development costs over application robustness and longevity. An enterprise mashup environment is only as valuable as the quantity and quality of the mashable assets available for users and professional or nonprofessional developers to assemble into mashup applications. The role of IT developers is to provide this high-quality repository of mashable assets.

Business Impact: Mashups can deliver significant application agility by bridging the gap between how quickly the business situation changes and how slowly enterprise applications evolve. Mashups provide power users and knowledge workers with the means to quickly build software solutions without the direct involvement of developers. Mashups also extend the reach of the applications group and provide greater developer productivity through a potentially highly leveraged repository of mashable assets.
Also due to following technology awakening along with Web 2.0:

- Increasing availability of XML data sources and data formats in business, personal, and consumer applications (including office suites)
- Wide deployment of XML web services
- Widespread current interest in data remixing or mashups
- Ajax and the availability of JavaScript-based widgets and micro-applications
- Evolution of web browsers to enable greater extensibility (for example, Firefox extensions and Greasemonkey scripts)
- Explosive growth in “user-generated content” or “lead-user innovation”
- Wider conceptualization of the Internet as a platform (“Web 2.0”)
- Increased broadband access.

These developments have transformed creating Mashups from being technically challenging to nearly mainstream. It is not that difficult to get going, but you need to know a bit about a fair number of things, and you need to be playful and somewhat adventurous.As we can see, Gartner rated Mashups as:

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Will Mashups remain cutting-edge forever? Undoubtedly, no, but not because they will prove to be an irrelevant fad but because the functionality we see in mashups will eventually be subsumed into the ordinary “what-we-expect-and-think-has-always-been-there” functionality of our electronic society.

Moreover, Mashups reflect deeper trends, even the deepest trends of human desire. As the quality, quantity, and diversity of information grow, users long for tools to access and manage this bewildering array of information. Many users will ultimately be satisfied by nothing less than an information environment that gives them seamless access to any digital content source, handles any

content type, and applies any software service to this content. Consider, for example, what a collection of bloggers expressed as their desires for next-generation blogging tools.[15]

"Bloggers want tools that are utterly simple and allow them to blog everything that they can think, in any format, from any tool, from anywhere.Text is just the beginning: Bloggers want to branch out to multiple media types including rich and intelligent use of audio,photos, and video.With input, having a dialog box is also seen as just a starting place for some bloggers: everything from a visual tool to easy capture of things a blogger sees, hears, or reads point to desirable future user interfaces for new generations of blogging tools."

So we can say Mashups are starting to forge this sought-after access and integration of data and tools - not only in the context of blogging but also to any point of interaction between users and content.

1.6 The Mashup Interoperability

Interoperability specifically in the Mashup context must be defined broadly enough so as to be useful in discussing all relevant applications of the term, but not so broadly that it encompasses so much as to lose meaning and value as a limiting force. Because all Mashups inherently take advantage of interoperability, we take the view here that Mashup interoperability is the set of conditions, including compatible technologies and willing participation by Web services and data providers that permit developers to create Mashups. Our definition is thus not limited to just one protocol or set of protocols. Mashup interoperability, therefore, includes the quality that allows a mashup developer easily to convert his or her mashup from using one data source (say, a map) to another. It also includes some kind of parity in messaging technology, and further encompasses the needs for interoperability at the content level, such as data portability.

The two active ingredients of Web mashups are the data and application programming interfaces (APIs), which provide an interface with which nonprogrammers can gain access to a malleable form of the data. Both data and APIs can be public or private.

Mashups have recently gained attention because of the creativity involved in their development and the functionality they afford users. If the Internet is thought of in superseding layers – physical (the wires), logical (the protocols), content, and social – mashups fit between the content and social layers, changing the ways in which individuals relate to content.

Mashups fit into the traditional stack as follows:

[15] http://www.cadence90.com/blogs/2004_03_01_nixon_archives.html#107902918872392913

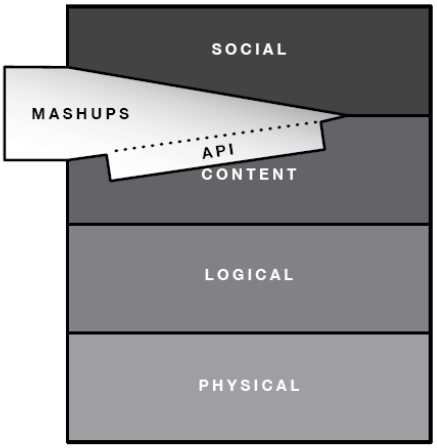
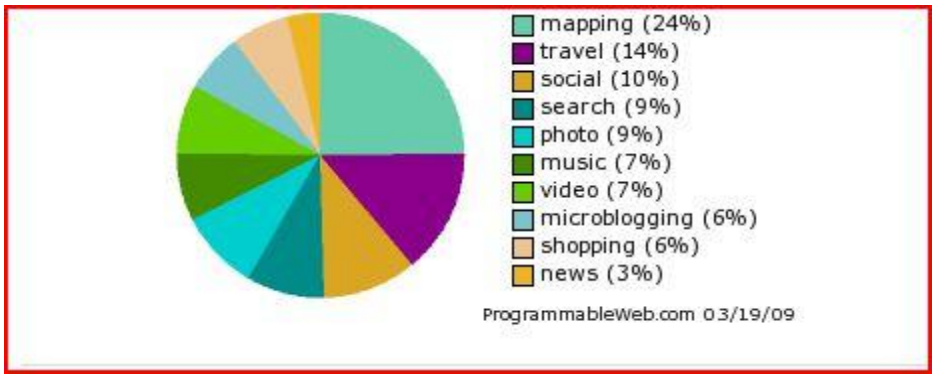


Figure 3: Mashup Layer

Mashups rely on the logical layer to support point-to-point messaging. This messaging is by definition a form of interoperability. Often, it is XML-based. Data is pulled from the content layer, greatly enabled by the API, and the final product affects the social interactions on the Internet.

Data sources come from a range of Web content, including posted APIs, statistics, maps, RSS feeds, and advertisements. Mashup content is also sourced by 'screen scraping,' a process where, in the absence of an API, a computer program 'scrapes' a site for data, using code that crawls the site and collects the information in a format the programmer can use for his or her mashup. Many people are experimenting with mashups using Microsoft, Google, eBay, Amazon, Flicker, Serena, Facebook, and Yahoo APIs; companies often post their own API so that developers can utilize it in new mashups. The result is a value-added representation of data that makes it easier for a user to synthesize information. The following chart shows the distribution of popular APIs that are used in this way:

Figure 4. Top APIs Used for Mashups[16]



The mere existence of these APIs does not lead inexorably to the development of any mashups, let alone mashups that are themselves interoperable with other APIs or mashups. The level of interoperability of a mashup depends on the endpoints into which the Web service plugs – the data on one end and the mashup on the other. The places at which data departs one point and arrives at another can be locked in such a way that they lessen the system's interoperability overall. For example,

[16] Programmable Web, Top APIs for Mashups, http://www.programmableweb.com/apis (last visited 22 March 2009)

a data provider could allow access to their data only through the use of their API, which may not allow for all uses, or a mashup programmer could represent his information in such a way that it is hard for someone who wants to use it as a data source to build on top of it.

1.7 Mashups Innovation

Because Web services began as a way to link large, non-interoperable systems, I observe that there is value in maintaining or enhancing the current level of interoperability from the perspective of promoting innovation in this space. Interoperability among Web services has facilitated the development of innovative mashups that are productive for society. The job of ensuring continued innovation in the mashups context becomes our focus. Challenges arise in developing an understanding of further ways to interoperate, what the stakeholders might demand, and how to achieve it if necessary. When it comes to mashup innovation, there are three primary stakeholders:

- Individual users of mashups
- Programmers
- Data providers

As players in the system, these stakeholders bring a variety of use cases, motivations and needs to the table. The availability of data sources via open APIs increases the innovation potential of building on top of Web services technology. There are several types of innovation occurring above the technology layer:

- Adapting existing business models or testing new ones
- Combining existing data in novel ways
- Creating new content by analysing existing data

The connection between these types of innovation and the interoperability we have observed currently crosses boundaries between layers in the stack – interoperability at the technical level enables innovation at the content level. However, the relationship may work for that step only, since that innovation may not in turn be interoperable or foster further interoperability at the logical or content levels.

2. Learning about Mashups

2.1 Types of Mashups

As we know Mashups have several different colloquial interpretations, which has resulted in some confusion regarding the term and its use. The word originated in the music industry, where a

mashup was a combination of two or more songs to create a new experience. Typically, the vocal track of one song was combined with the instrumental background of another in this process. The technology industry extended this definition to encompass a new application genus that described the combination of two or more sources into an integrated site. This technique of development hybridization can be roughly split into two separate categories: **Consumer Mashups** and **Enterprise Mashups**.

Consumer mashups are generally associated with Web 2.0. They require a lesser amount of programming expertise because they rely on public Web sites that expose well-defined APIs and feeds (see Figure 4). The output is usually created by one of the sites participating in the mashup. In the classic "show craigslist listings on a Google map,"[17]the API of Google Maps is used to plot and present the feed obtained from craigslist.com. The limitation of this approach was that resources had to be "mashup ready."

Enterprise 2.0 Mashups (sometimes referred to as data mashups) are more complex. Depending on which solution a firm deploys enterprise mashups can emerge in several ways:

- Mashups are used solely by IT to rapidly deliver products. Application developers use both internal and external sources to create data mashups and employ traditional coding techniques to create the user interface around them. Users aren't directly involved in the construction process but they benefit from IT's ability to provide solutions more quickly.

- IT creates a set of "mashable" components and gives end users a sand-box environment where they can freely mix and match the pieces together themselves. If users need new components, they have to solicit IT help to create them.

- An organization deploys an environment that lets anyone create and combine his or her own mashups. This approach is the most difficult implementation to manage, but probably has the greatest impact. To understand the challenge of this approach, consider the use of Microsoft Excel in many firms. Users can create spreadsheet-based applications and pass them around without any central oversight of what exists, how it is used, or if it was tested. This friction-free creation and distribution model spreads good solutions as quickly as bad ones.

Whether mashups are used by IT, business associates, or both, their agile nature makes them a key enabler of Enterprise 2.0. Unfortunately, they are not without potential downsides. In an attempt to "deconstruct" the success of Google, the *Harvard Business Review* points out several pitfalls[18] that can hinder success in a culture of open development:

- As people spend more time experimenting, productivity in other areas can suffer.
- Poor coordination across groups can lead to duplication of efforts and repeated mistakes.

[17] http://housingmaps.com

[18] Iyer, Bala, and Thomas H. Davenport. "Reverse Engineering Google's Innovation Machine." Harvard Business Review, April 2008.

- A constant stream of new products may confuse the organization and its employees

Despite these potential hazards, the authors indirectly identify the virtuous circle of Enterprise 2.0 (Figure 5). As diverse products are combined to create useful new resources, they themselves become fodder for the next generation of useful products. In principle, this process isn't very different from the longstanding goal of reusability that firms have strived for in their applications and architecture. Three important differences arise this time around, however:

1. In the age of mashups "reuse" is no longer an ivory-tower concept restricted to the purview of application architects. Because end users and developers alike will be creating solutions, *everyone* will engage in the practice of reuse.

Figure 5: Virtuous Cycle of Mashups

2. The existing approach to reuse front-loads development efforts with additional planning and coding to create open APIs and extra documentation that may never be used. Because mashups impose reusability "after the fact," their creators will build their own APIs and include only the minimum functionality needed.

3. Traditional reuse practices don't require that a system that leverages existing code or libraries is itself reusable. This leads to implementations that are essentially "dead ends." Mashups are implicitly reusable, which creates a never-ending cycle of potential associations and recombination.

2.2 Acquiring Data from the Web

As we saw in the last section, the majority of Consumer Mashups use the public APIs of a handful of Web sites. In the enterprise model, the best potential sources for mashup data may not be as forthcoming. In these situations, it becomes necessary to employ creative techniques to extract information. One of the most common and controversial techniques is often referred to as "screen scraping." This derogatory phrase carries a long sullied history and is thrown around by detractors seeking to undermine this approach.

Traditional "screen scraping" owes its origins to the early days of desktop computing, when IT departments developed various techniques to migrate "dumb terminal" mainframe applications to end-user computers. Rather than tackle the costly and time-consuming task of rewriting or replacing existing applications, many IT departments used special PC-based applications that emulated the original terminals.[19]These applications could receive the data from the mainframe and extract the contents of the forms presented on the old green-screen systems. User keystrokes were likewise emulated to send input back to the original application. This technique relied on developer-created templates and was both highly position-sensitive and extremely unforgiving. The smallest alteration in the mainframe display would break the predefined template and break the new application.

Among the various uses of "Mashup" we can perhaps identify gradations of the term, based on the techniques used and the data sources involved (although the source and format of the data source tends to drive the selection of technique):

1. Mashups that screen-scrape data in violation of Terms of Service agreements or that otherwise hack undocumented java-script code to accomplish mapping. Perhaps closest to the original sense of mashups in music, this type of mashup challenges the ownership (or, at least, control) of data and, in some cases, expresses a "hacktivist" stance, wherein the use of hacking and "liberating" data is employed with the goal of challenging corporate or government authority. Examples of this type of mashup are scarce for two reasons: the original websites may take legal action to shut down the mashup, or they may eventually agree that the information should be made public, providing it in more easily mashed APIs.

2. Mashups that screen-scrape data that is publicly available (in a legal sense), but which is not provided in a machine-readable format. As in the first case, hacking is employed to create a service that didn't exist before, using a site not how it was expected to be used. These mashups may screen-scrape or hack other sites in order to make them more functional or remove bothersome advertisements, or to create a searchable database of results for a particular query when the original website only provides results one at a time, as is sometimes the case with data that governments are legally required to published, but not necessarily in a convenient format. The original HousingMaps and Chicago Crime perhaps best fit into this category.

[19] Such as an IBM 3270 or VT220.

3. Mashups that use RSS feeds and APIs of third party sites to acquire data. This includes most of the mashups on ProgrammableWeb. Unlike the original sense of mashup, these sites use online APIs exactly as they were intended, but the juxtaposition of the data may still producing surprising results.

4. Mashups that only use a mapping API, but include general-interest user-uploaded data gathered from some other source. This second source is not from an API, and perhaps not even in a digital format, such as a user typing in locations from printed documents. This increasingly drifts away from the original sense of a mashup, for these mashups use the map API just as it was intended. While these mashups still give their creators the new and powerful ability to explore spatial patterns in the surrounding datascape, the creator's role is more passive, no longer participating in the engineering of the web, with no sense of "re-wiring" the network. The user participates on the level of content only.

5. Mashups that only use a mapping API, and map only personal interest data. The ability for the user to create, share, and explore data that they create themselves is, again, a significant change from the past, and one that will be addressed in later chapters. However, despite this ability for spatial introspection and publishing, these mashups are passive when it comes to other data-sources. They are read-only consumers of the rest of the data that surrounds them.

2.3 Looking for Patterns in Mashups

Mashups combine content from more than one source into a new integrated whole. You can understand a specific mashup by answering a number of basic questions:

- What is being combined?
- Why are these elements being combined?
- Where is the remixing or recombination happening?
- How are various elements being combined (that is, first in the interface but also behind the scenes in the technical machinery)?
- How can the Mashup be extended?

In this section, let us analyze following couple of examples using the previous questions loosely as a framework. The websites are:

- Housingmaps.com
- The Google Maps in Flickr Greasemonkey script

One pattern you will see repeated among Mashups that link two web sites is the combination of three actions:

1. Data is extracted from a source web site.
2. This data is translated into a form meaningful to the destination web site.

3. The repackaged data is sent to the destination site.

Of course, the details differ among the mashups, but this general pattern holds true, as you will see in the three mashups presented in detail in this section. *Where* the remixing actually happens differs in the three mashups you'll see in this section: in a separate application as in Housingmaps.com, in Flickr for the Google Maps in Flickr script.

2.3.1 Housingmaps.com

When we explain mashups to others, we typically use the example of the web site Housingmaps.com, a mashup of Craigslist and Google Maps. Housingmaps.com is useful in ways that are quick and easy to understand, which invites repeated usage. It also requires no software beyond a modern web browser. Moreover, Housingmaps.com takes two already well-known web applications to create something new.

Figure 6 shows Housingmaps.com displaying a specific rental listing. Note the photos of the apartment and the links to Craigslist. All the data is drawn from Craigslist and then displayed in a Google map.

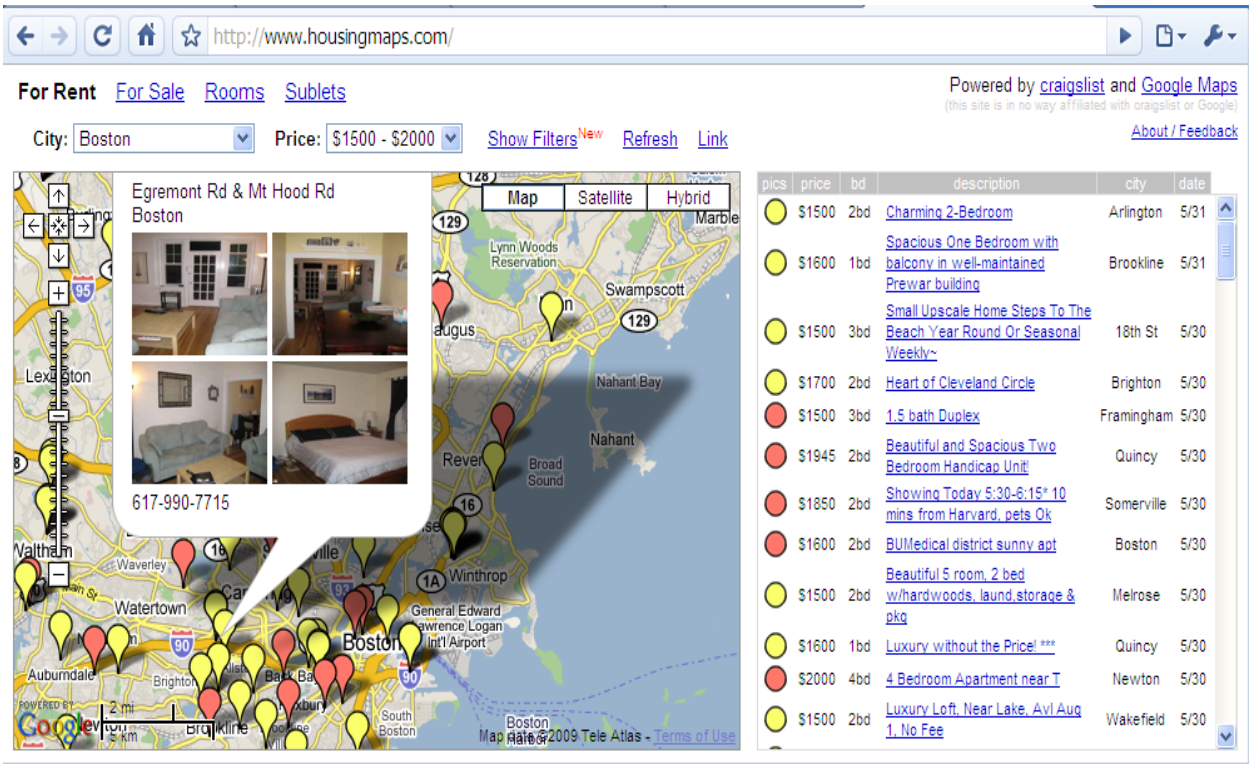


Figure 6: Housingmaps.com

2.3.1.1 What is Being Combined?

Housingmaps.com takes the list of houses, apartments, and rooms that are for sale or rent from Craigslist and displays them on a Google map. Note that it was invented by neither Google nor Craigslist but by an individual programmer, Paul Rademacher, who, at the time of its invention, was working for neither Google nor Craigslist but who was later hired by Google.

2.3.1.2 Why Are the Constituent Elements Being Combined? What's the Problem Being Solved?

Craigslist provides links to Google Maps and Yahoo! Maps for any individual real estate listing, but it does not map the listings collectively. The single listing per map on the Craigslist interface makes it a challenge to mentally track the location of all the properties. Moreover, when looking for real estate, you often want to look at a narrowly defined neighborhood or find houses with good access to transit. With Craigslist, you have to click many links and manually piece together a lot of maps to focus your search geographically.

Housingmaps.com addresses these challenge by letting you see on a Google map all the Craigslist apartments or houses in a specific area, not just an individual item. At Housingmaps.com, geographical location becomes the primary lens for looking for real estate, with a map as the central element of the user interface

2.3.1.3 Where Is the Remixing Happening?

The remixing occurs on the server side on a web site (Housingmaps.com) that is distinct from both the source web site (Craigslist) and the destination application (Google Maps). Data is drawn from the source and transformed into a Google map, which is embedded in web pages at Housingmaps.com.

2.3.1.4 How are these Elements Being Combined?

This question really breaks down into two questions:

- How does Housingmaps.com obtain the housing and rental data from Craigslist?
- How does Housingmaps.com create a Google map of that data?

A desirable, and increasingly common, method for Mashups to obtain data from a web site is through a web site's publicly available application programming interface (API). An API is designed specifically to facilitate communication between programs, often including the exchange of data.

At this time, Craigslist does not provide a public API but does provide RSS feeds. As we know by now, RSS feeds are used to *syndicate*, or transport, information from a web site to a program that *consumes* this information. The RSS feeds, however, do not provide enough detail to precisely position the listings on a map.

Consequently, Housingmaps.com *screen-scrapes* (or *crawls*) Craigslist; that is Housingmaps.com retrieves and parses the HTML pages of Craigslist to obtain detailed information about each listing. The crawling is performed carefully so as to minimize the use of bandwidth. When you access Housingmaps.com, you are accessing not real-time data from Craigslist but rather the data that has been screen-scraped by Housingmaps.com. To display the real estate information on a Google map, the current version of Housingmaps.com uses the Google Maps API,[20] which is the official Google-sanctioned way of embedding Google maps in a non-Google-owned web page.

It's interesting to go into a bit of history here to understand the emergence of the mashup phenomenon. When Housingmaps.com first showed up in April 2005, Rademacher was using Google Maps before it had any real API. He deciphered the original JavaScript of Google Maps and figured out how to incorporate Google Maps into Housingmaps.com. During the period between the release of Google Maps on February 8, 2005, and the publication of version 1 of the Google Maps API (on approximately June 29, 2005[21]), there was a period of intense "hacking" of Google Maps, described in the following way by members of the Google Maps team:[22]

For this and other reasons we were thrilled to see "hackers" have a go at Google Maps almost immediately after we launched the site back in early February. Literally within days, their blogs described the inner workings of our maps more accurately than our own design documents did, and soon the most amazing "hacks" started to appear: Philip Lindsay's Google Maps "stand-alone" mode, Paul Rademacher's [Housingmaps.com], and Chris Smoak's Busmonster, to mention a few.

2.3.1.5 Comparable Mashups

Since the debut of Housingmaps.com, many other mashups—in fact, tens of thousands—have followed this pattern set of recasting data to make geographical location the organizing principle. These mashups cover an incredible range of topics and interests.[23] Many other mashups involve extracting geo-coded data (location information, often latitude and longitude) from one source to then place it on an online map (such as a Google map or Yahoo! map). We name two prominent examples here:

- Adrian Holovaty's Chicago crime map (http://chicagocrime.org), which is a database of crimes reported in Chicago fronted by a Google Map interface.

- Weather Bonk, which is a mashup of weather data on a Google map (http://www.weatherbonk.com/weather/about.jsp)

[20] http://www.google.com/apis/maps/

[21] http://benmetcalfe.com/blog/index.php/2005/06/29/google-make-map-api-available-finally/

[22] *Google Maps Hacks* by Rich Gibson and Erle Schuyler (O'Reilly Media, 2006)

[23] See http://googlemapsmania.blogspot.com/ for many new mashups based on Google Maps that appear every day.

2.3.2 Google Maps in Flickr

In the earlier days of Flickr (before August 2006), there was no built-in feature that allowed a user to show pictures on a map. The Google Maps in Flickr (GMiF) script was created to fill in that gap by letting you see a Flickr photo on a Google map. Today, even with Flickr's built-in map of geotagged photos, which uses Yahoo! Maps technology, GMiF remains a valuable mashup. GMiF allows users to use a Google map, which many prefer over Yahoo! Maps, to display their photos. Moreover, GMiF also integrates Google Earth, a feature not currently built into Flickr. GMiF provides an excellent case study of how you can extend an application such as Flickr to fit user preferences.

2.3.2.1 What is Being Combined?

GMiF (http://webdev.yuan.cc/gmif/) brings together Flickr pictures, Google Maps, and GoogleEarth within the Firefox browser via a Greasemonkey script. We'll break this down for you:

- Flickr (http://flickr.com) is a popular photo-sharing site.
- Google Maps (http://maps.google.com/) is an online mapping system.
- Google Earth (http://earth.google.com/) is a desktop "magic-carpet" interface that lets you pan and zoom around the globe.
- The Firefox web browser (http://www.mozilla.com/firefox/) is an open source web browser. Notable among its features is its extension/add-on architecture, which allows developers to add functionality to the browser.
- The Greasemonkey extension (http://www.greasespot.net/) is a Firefox extension that "allows users to install scripts that make on-the-fly changes to specific web pages. As the Greasemonkey scripts are persistent, the changes made to the web pages are executed every time the page is opened, making them effectively permanent for the user running the script."[24] Greasemonkey scripts allow you—as the user of that web site and not as the author of the web site—to make customizations, all within the web browser.

2.3.2.2 Why Are the Constituent Elements Being Combined? What's the Problem Being Solved?

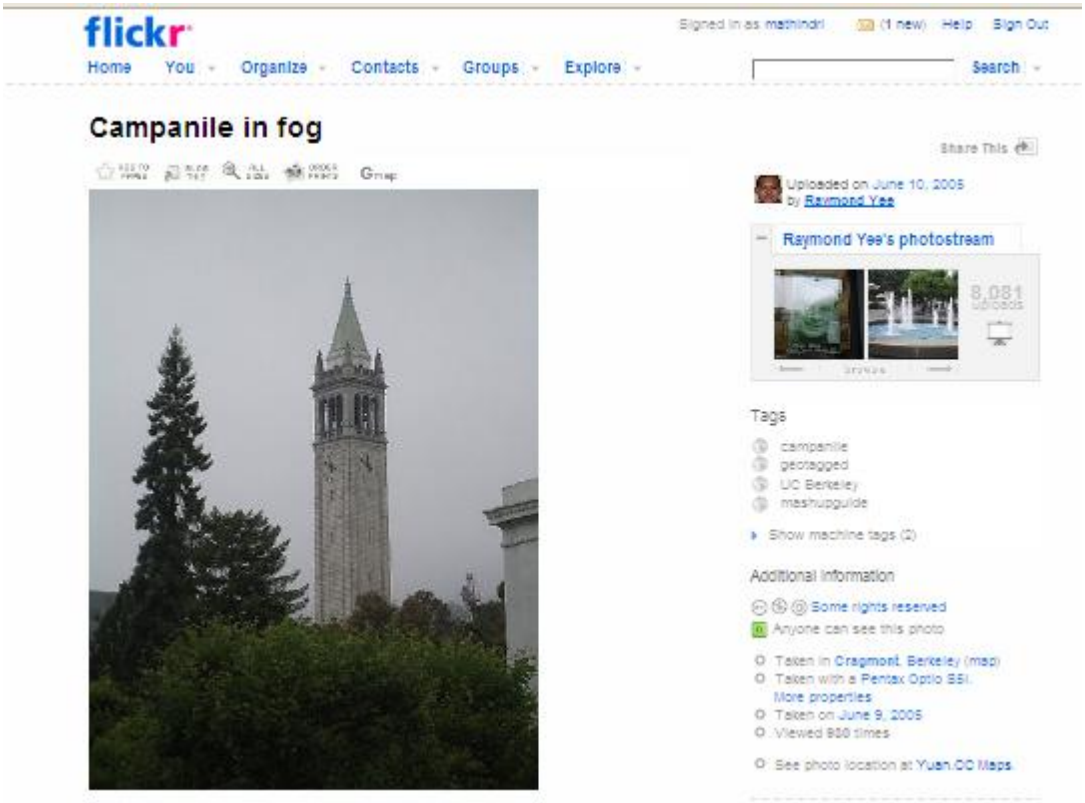
GMiF is a Greasemonkey script that allows you as a user to display a Flickr picture on a Google map or in Google Earth at the geographic location associated with that picture. GMiF was written to support geotagging in Flickr. Geotagging, in the context of Flickr, is the process of associating a location with a given photo, which is typically but not necessarily the location where the photo was taken.

[24] http://en.wikipedia.org/wiki/Greasemonkey, accessed on 03/05/2009, as http://en.wikipedia.org/w/index.php?title=Greasemonkey&oldid=97588087

Until geo-tagging was officially integrated into Flickr with the use of Yahoo! Maps in August 2006,[25] there was no direct way to associate geo-coding (location information) with any given picture. Rev Dan Catt catalysed the mass-geotagging phenomenon by suggesting that Flickr users shoehorn the latitude and longitude information into the tags associated with a photo. Many people took up the practice. The GMiF Greasemonkey script uses that geocoding for a photo.

Let's take a look at how GMiF works. Consider one of the photos, shown in Figure 7 (also available at http://flickr.com/photos/raymondyee/18389540/). Notice two things:

- This photo has associated geo-tagging information (for example, geo:lat=37.8721,geo:lon=-122.257704, and the tag geo-tagged).
- Note the presence of the rightmost GMap button above the photo. This button is the result of the GMiF script, which inserts the GMap button. In other words, if you do not have the GMiF Greasemonkey script installed, you won't see this button.



[25] http://blog.flickr.com/flickrblog/2006/08/great_shot_wher.htm and http://blog.flickr.com/flickrblog/2006/08/geotagging_one_.html

Figure 7: The Flickr photo "Campanile in fog" (http://flickr.com/photos/raymondyee/18389540/) with associated geocoding embedded in the tags.

Clicking the ***GMap*** button opens a Google map embedded in the Flickr web page, with a pin indicating the location of the picture in question (as shown in Figure 8). Note the red pin indicating the location of the photo. The blue pins correlate to other geotagged photos. The map also has a thumbnail of the photo in the upper-right corner.

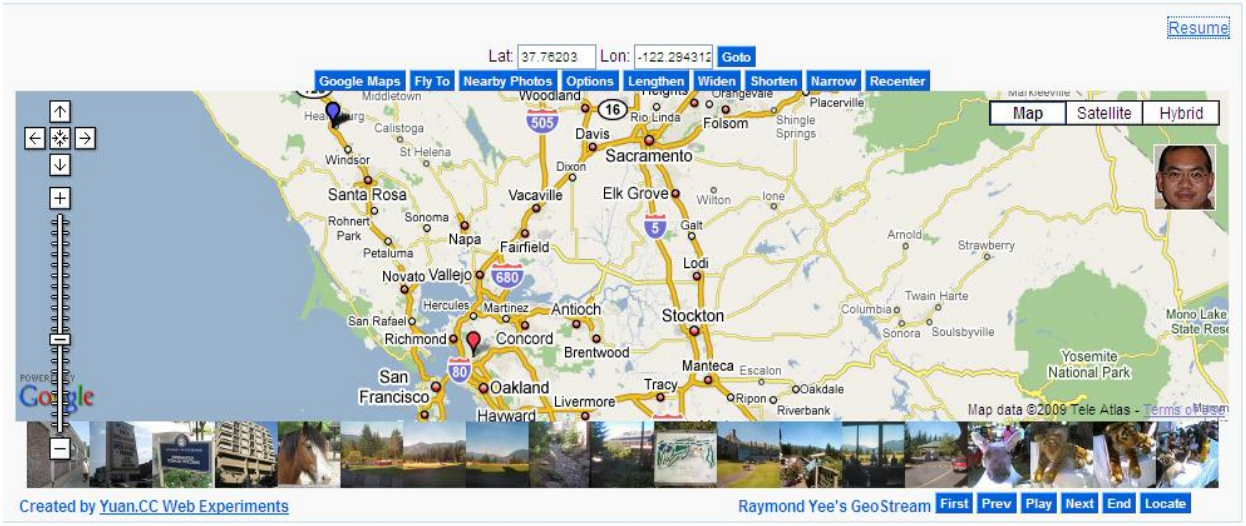


Figure 8: Clicking the GMap button opens a Google map in the browser

Clicking the pin opens a call out with a picture and options of other things to do with the picture. Note how the latitude and longitude listed correspond to the information in the geo:lat and geo:lon tags, respectively (as shown in Figure 9).

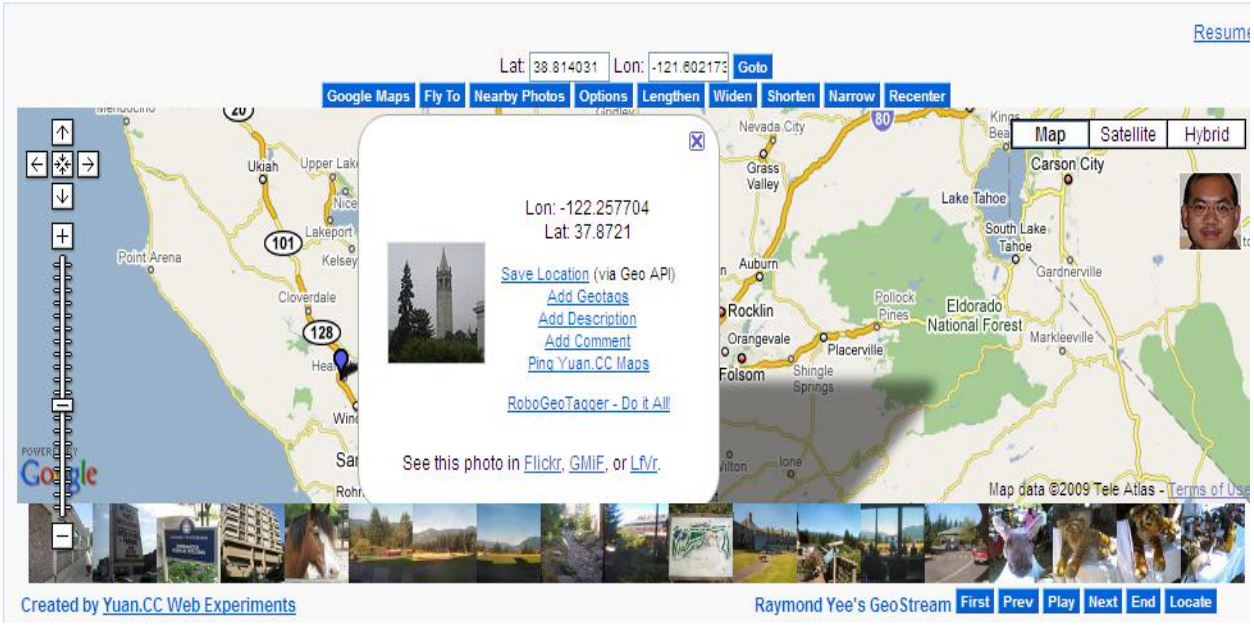


Figure 9*:* Clicking the red pin opens a balloon containing the photo and further geotagging functionality offered by GMiF.

Among the GMiF functions is integration with Google Earth. If you hit the Fly To button, you will be presented with a file to download. If you have Google Earth installed and it is configured in the default fashion, downloading the file launches Google Earth, and you will be "flown" to the location of the Flickr photo (as shown in Figure 10).

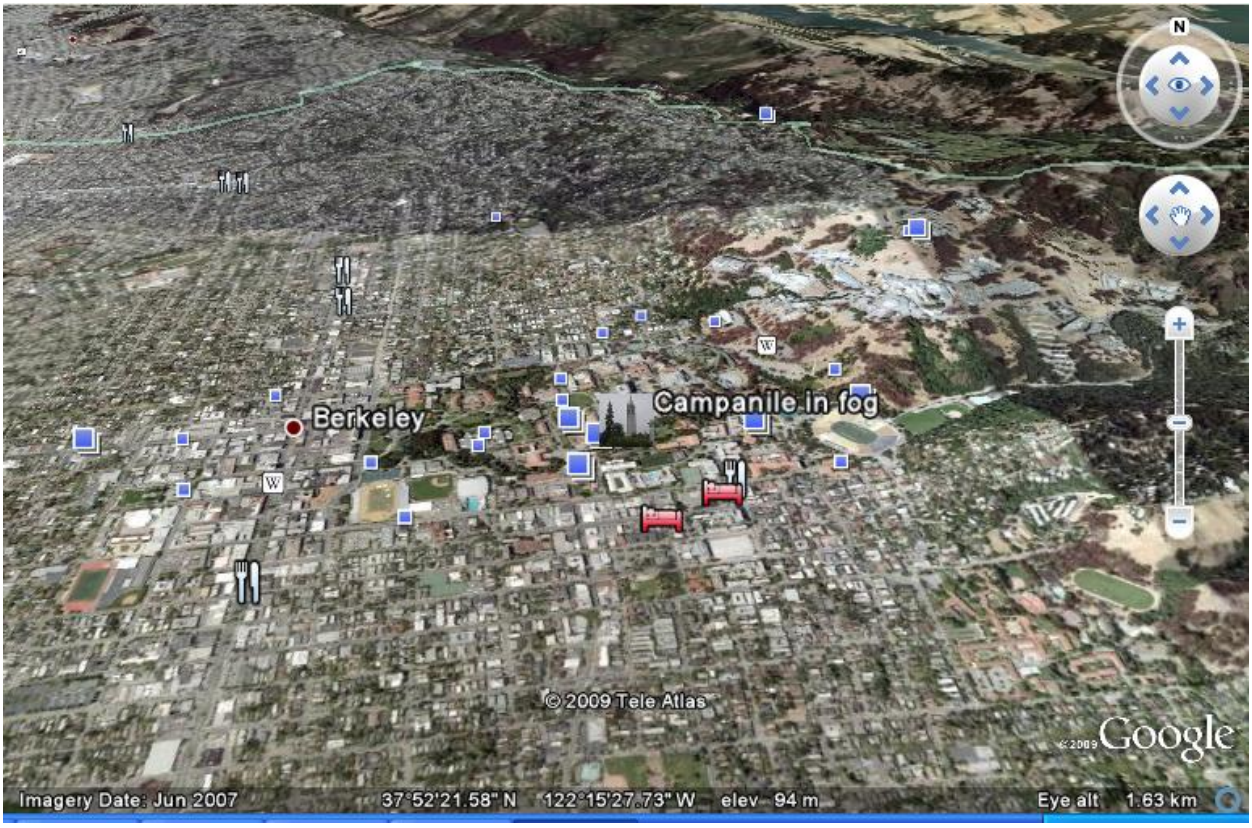


Figure 10: Clicking the GMiF Fly To button launches Google Earth, which then displays the photo at the latitude and longitude associated with the photo.

2.3.2.3 How are these Elements Being Combined?

The GMiF Greasemonkey script rewrites the HTML of the Flickr page to insert the GMap button (this rewriting of the HTML DOM is akin to looking in the HTML source for where the Flickr buttons are and inserting HTML code for the button). Furthermore, JavaScript code is added to embed a Google map in the Flickr page, when you (as the user) click the GMap button.

Integration happens in the context of Flickr web page, loaded in the user's browser. Note how powerful this is: you don't have to go to another application to see the picture on a Google map because you get to use a slightly modified version of Flickr. These modifications do not require the intervention of Flickr itself. Hence, there is room for a lot of customization.

How is the integration of GMiF with Google Earth created? The downloaded file is a KML file. KML is a dialect of XML, which is the closest thing we have to a *lingua franca* for exchanging data. The KML file contains the latitude and longitude associated with the picture and a URL of the picture. KML is used to exchange geographic type data that is understood by Google Earth. In other words, GMiF takes information from one source (the URL of the picture and the latitude and longitude of the picture embedded in tags from Flickr) and translates that information into a form that is understood by the destination application, namely, KML for Google Earth. Once you translate that information, you still need to get the information to the destination; in this case, the transport happens through the formation and downloading of the KML file.

Admittedly, GMiF is a bit "hackerish," requiring the installation of the Firefox web browser (which does not come by default on Windows or Mac OS X), the Greasemonkey extension, and the GMiF script. But we bring this up here to talk about the lengths to which people are willing to go to experiment with their tools to combine technologies.

2.3.2.4 Comparable Mashups

Mappr (http://www.mappr.com/), "an interactive environment for exploring place based on the photos people take," is a mashup of Flickr and a Flash-based map.

2.4 Tracking Other Mashups

Of course, many Mashups exist other than the ones we have highlighted in this section. You can always learn more by studying other examples of mashups. In studying mashups, you will find one web site that is a particularly useful resource: http://programmableweb.com. This site is created and managed by John Musser.

- The Programmableweb.com blog is a narrative of the latest developments in the world of mashups and APIs.[26]
- The Mashup Dashboard provides an overview of the mashups in the Programmableweb.com database, which as of August 2007, covers more than 2,220 mashups.[27]

In this section we studied two major examples of Mashups: Housingmaps.com and Google Maps in Flickr, We chose these examples to illustrate some commonalities and differences you will find among mashups. By posing a number of analytical questions (What is being combined? Why are these elements

[26] http://blog.programmableweb.com/

[27] http://www.programmableweb.com/mashups

being combined? Where is the remixing or recombination happening? How are they being combined, in terms of the interface and behind the scenes in the technical machinery? How can the mashup be extended?), we saw a repeated pattern:

1. Data is extracted from a source web site.
2. This data is translated into a form meaningful to the destination web site.
3. The repackaged data is sent to the destination site.

There are important differences among the various mashups, specifically in where the integration happens and what is being integrated. For instance, Housingmaps.com is a server side application, whereas the mashing up of GMiF and LibraryLookup occurs within the browser. (Appendix: Table 2) illustrates all the mashups we studied and labled for better classification. Note that Programmable Web contents thousands of Mashups. Some are not at upto any quality. So we explored by taking into account the Top 10 Mashups for the day in the site.

2.5 Classification of Mashups

After an exploratory study of various mashup patterns available in ProgrammableWeb, it was clear that there is no formal standard classification pattern available to categorized mashups. We investigated several models for Mashup categorization [1, 2, 3, 4]. To summarize from our point of view Mashups can be classified based on the following four questions:

- What to mash up?
- Where to mash up?
- How to mash up?
- For whom to mashup?

In the following subsections I will separately regard each of these four perspectives on mashups

2.5.1 What?

Depending on the sort of assets being combined or integrated mashups are assigned to one of the following three categories: *presentation*, *data* or application *functionality*.

A P*resentation Mashup* focuses on retrieving information and layout of different Web sources without regarding the underlying data and application functionality. For this type of mashup prebuilt widgets are simply drags and drops into a common user interface. Usually, the creation of a presentation mashup requires little or no knowledge of programming languages.

A D*ata Mashup* merges data provided by different sources (e.g.,Web services, feeds or plain HTML) into one content page (i.e.for a given city combining different services to obtain its weather forecast,

upcoming events and photos). The user mixes data from multiply sources and customizes the data flow of for example the Web page containing data from different sources.

A F*unctionality Mashup* combines data and application functionality provided by different sources to a new service. The functionalities are accessible via APIs.

Based on the assets being combined one can find another classification of mashups such as ***Mapping-Mashups*** (combination of information into maps, e.g., Google maps), ***Photo/Video-Mashups*** (combination of information into foto/video files e.g., from flickr), ***Search/Shopping-Mashups*** (integration of mechanisms for the comparison of product prices into Web pages) or ***News-Mashups*** (integration of news into personal Web pages). We regard this kind of mashup types as a refinement of the Mashups introduced above where for example a Presentation Mashup can consist of resources being mashed out of news.

2.5.2 Where?

Mashups can be distinguished depending on the location where they are mashed up. *Server-side* mashups integrate resources (e.g., services and data) on the server. ***Client-side M*ashups** integrate resources on the client, often a browser. Usually a mixture of client-side and server-side applications is used for the creation of mashups. An elaborated explanation of server-side and client-side mashups can be found in [5].

2.5.3 How?

Mashups can be further categorized depending on the modality the resources are integrated or combined to one representation.

The E*xtraction* Mashup can be considered as a data wrapper collecting and analyzing resources from different sources and merging the resources to one content page.

In a ***Flow M*ashup** the user customizes the resource flow of the Web page combining resources from different sources. The resources are transformed and integrated within the Mashup application.

2.5.4 For Whom?

Different mashup tools can be used to build mashups that combine content from different sources but distinguishing the target group being addressed. In this context, mashups can be categorized in C*onsumer Mashups* and E*nterprise Mashups*, also referred to as *business* mashups.

A Consumer mashup is intended for public use and combines resources (e.g., layout or data) from different public or private sources in the browser and organizes it through a simple browse based user interface.

An Enterprise mashup merges multiple resources (e.g., data and application functionality) of systems in an enterprise environment. These mashups combine data and application functionalities of different systems e.g., ERP, CRM or SCM in order to respond to their objectives. The creation of enterprise mashups requires considering security, governance or enterprise policies. Enterprise mashups provide a fast way for merging and representing internal and external enterprise resources from different sources without a middleman.

3.0 Making of Mashups

3.0.1 Mashup Developers

The goal of the Web services community is to encourage data mixing that will spark creativity and facilitate problem solving. We see that happening. A key advantage of the advent of open APIs is that many people can simultaneously tackle a particular problem by working on their own version of a mashup. Mashup contests are one way to get many people thinking about "cool," useful new combinations of Web-based information.

Various companies are making it possible for individuals to create mashups without having to do any of the programming themselves. ZeeMaps.com, for example, will transpose customers' data, submitted to ZeeMaps in an Excel spreadsheet, onto a Google map. Yahoo Pipes provides a community space where programmers can mix, match and share mashup code.[28] On the more complex end of the spectrum, enterprise-level technologists enable mashup creation throughout organizations by centralizing APIs and making them available to members of the organizations. For instance, JackBe is a company whose main product, Presto, facilitates the creation and exchange of customized mashups within an enterprise.[29]

In a business sense, mashups have tremendous potential for companies of all sizes as a mechanism that could help pull together disparate information and make it available in a form that is most useful

[28] Yahoo! Inc., Pipes, http://pipes.yahoo.com (last visited 04/04/2009).

[29] JackBe Corporation, Products, http://www.jackbe.com/products/index.php (last visited 04/04/2009).

and relevant to the specific user base.[30] While widespread and widely useful mashups are available and popular, the more easily a programmer can custom-tailor data to a certain use case, the more we observe the emergence of small, niche mashups. When individual developers can easily tweak or combine existing Web services to suit their own needs and then make the resulting mashup available to others at no additional cost, many more mashups will be made than if a significant investment of capital were necessary. And because the capital requirements are so low, a relatively modest amount of advertising revenue can suffice to make a mashup profitable even if it appeals to a limited audience. Mashup Servers, software packages designed to combine commonly used Web services technologies and make mashup creation easier for non-programmers in particular, are entering the market as a means of providing users and developers with secure and consolidated access to disparate data from a number of sources. In July 2007, the WSO2 Mashup Server team announced the release of the WSO2 Mashup Server v0.1. The WSO2 Mashup Server is a way to tailor Web-based information to the personal needs of individuals and organizations. It offers a platform for consuming data from a variety of sources including Web services, HTML pages, and feeds, and processing and combining it with other data using JavaScript with E4X XML extensions.[31] Its developers envisioned the WSO2 Mashup Server becoming central to a budding ecosystem of community-developed services broadening the range of capabilities for mashups and distributed applications. Indeed, Mashup Servers are emerging as a new Web service with rich metadata and artifacts that can be used to quickly build bespoke user interfaces.

Mashups are created to support a range of business models and practices. First, an existing profit-seeking enterprise can incorporate mashups to facilitate or complement another business model. Second, a start-up can try to make the mashup itself the primary revenue driver of a new business model. And third, various parties have produced public service mashups to support various nonprofit goals.

Existing businesses can incorporate mashups externally (i.e., for their customers), or for their own internal purposes. For example, stock trading company E*TRADE has used mashup technology to create a value-added service, an "intelligent cash optimizer," which shows the investor where they could put their money. It relies on E*TRADE's own, proprietary data, but is represented in a dynamic and easy-to-understand way. Mashups are also being used in the back-end of businesses, in companies that already have applications that simply need to talk to each other. This can allow for much quicker time to market, and leverages work already done.[32] Another example of an add-on mashup is a store locator with a map, which can be useful both to customers and to employees. News outlets have also turned to mashups as useful ways to show the information they are reporting, through dynamic maps and visuals. In this way, these uses of mashups represent an improvement upon the classic notion in computing of middleware.

Increasingly, the mashup is being used as the driver of the business. They work off pay-per-use, subscription, or ad-funded models. For example, sites like mapmyrun.com and povo.com are start-ups, providing value-added services to users. The use of mashups as the basis for a start-up entails greater

[30] In the words of JackBe's promotional material: "Imagine an enterprise with many highvalue actors who need easy access to information from different data sources as well as ways to manipulate and mashup the data to reach important decisions." JackBe Corporation, Products, http://www.jackbe.com/products/index.php (last visited 04/04/2009).

[31] E-mail from Jonathan Marsh to mashup-user mailing list, July 6, 2007, http://wso2.org/mailarchive/mashup-user/2007-July/000001.html.

[32] Interview with John Musser, May 18, 2007.

risk than simply adding it on to an existing business because it inherently depends on one or more APIs or data sources from third parties who could change terms or cut off access entirely. This phenomenon gives rise to a series of concerns that we will explore later in this paper.

In public service mashups, non-profits, governments, or private citizens use mashups to serve the public interest without a profit motive. Some are funded through grants and other non-profit funding sources. ChicagoCrime.org is a popular example of such a mashup. It pulls data from Google Maps and the Chicago Police Department, placing crimes and information about them onto a street-by-street map. This type of mashup is commonly opportunistic:[33] the programmer finds data sources, but sees them as inaccessible to those who would use them. She remixes and represents the data so that individuals get value out of it. Experimentation with mashups by members of the nonprofit and government sectors is an important reason to seek sustainability in mashup interoperability.

3.1 Mashup Developers

Data providers fall into two main camps: companies whose data is normally gathered and held for their own purposes, and public or government organizations that collect data in the course of their service to the public. Companies have increasingly made their data available via API, and opening data to mashup developers is quickly becoming part of the Web 2.0 movement. One proponent stated, "It's going to be almost like a decade ago. Do you have a Web site? Check. That's how it will be with APIs."[34]

The main motivation to make data more readily available via API appears to be exposure. For a private company such as Google, allowing others to use their map functionality increases the branding of the Google name. Some say that companies create APIs in recognition of the innovative and generative possibilities afforded by allowing others to mix and mash the data. A company's reliance on the uptake of its API varies. Some companies tie the use of their data to payment in a pay-per-use model. For example, salesforce.com provides customers access to its data on an as-needed basis. Others provide the API (and access to the data) free, sometimes in conjunction with placing advertisements.

While public-sector information sources will always be available, it is important to note that without the proper incentives to obtain data, corporations will not collect it in the first place, let alone make it available to mashup developers. Data providers usually have to invest resources in gathering or creating data, and they expect to be compensated for that in some way. Intellectual property laws, advertising revenue, and intangible rewards in terms of visibility, goodwill, and market position currently provide incentives to provide data, but if these cease to be compelling, the Web services ecosystem will have to adjust to ensure data providers continue to get a return from their investments.

[33] Interview with Adrian Holovaty, May 30, 2007.

[34] John Musser, quoted in Jason Snyder, Digg floats API, phishing mashups to come, April 20, 2007, http://www.infoworld.com/article/07/04/20/HNdiggapi_1.html.

3.2 Mashup Development Tools

Several mashup tools have been published that provide functionalities for building, storing and publishing mashups. These tools were conceived as Web 2.0 applications allowing users sharing their created mashups and providing them with very intuitive drag and drop facilities. The range of these mashup tools spans from open-source tools to highly-cost license tools. Some of the vendors offer a coding editor while others focus on users with no programming skills and thus provide easy-to-use access and application to their tool suites.

Based on our classification schema of mashups presented in Section 2.5 we discuss in this section tools supporting the creation of the corresponding mashup types. Usually resource owners facilitate the access to their data by offering application programming interfaces (API). These APIs follow standard protocols and can easily be used to combine resources by a mashup tool from multiple sources. The whole range of APIs for mashups is listed on the Web page `programmableweb.com`. The APIs can be browsed by the preferred programming languages (e.g., PHP, JAVA or .NET) or predefined categories where the Google Maps API seems to be the most popular one. The combined resources can quickly be displayed to users in a Web browser using, resulting transparent to users the techniques (SOAP [24], REST [25], Screen scraping or languages such as JSON[35] used to access and combine these resources.
Table 1 shows an overview of mashups tools categorized according to our classification model in Section 2.5 [36]. An analysis of mashup tools regarding its suitability for data analysis can be found in [8]. Table 1 does not consider server-side and client-side mashup styles because the location where to mashup differs with system configurations. In the following we will introduce four mashups tools covering different mashup categories and also a language for the mashup creation will be sketched.

Presentation, Extraction and Consumer Mashup Tool: Example for such a mashup tool is dapper [3]. The term dapper results from data and mapper, which exactly describes the functionality of the tool namely to simply drags and drops (map) pre-built widgets into a common user interface and subsequently to reuse and share the output.

The usage of dapper is very simple and does not require any knowledge of programming languages. Initially the user has to search for the Web pages out of them she would like to extract content. Then she highlights the area that should be extracted and finally dapper composes the extracted content to one representation. The user can make the output available for others who can reuse the representation in their mashup environment. Figure 11 shows a screenshot of the dapper interface. The user is scrolling a specific Web site and aims at extracting a logo of that side. The content to be extracted is highlighted in orange.

Data, Flow and Consumer Mashup tool: Example for such a mashup tool that mixes data flow from multiply sources is *DERI Pipes* [7]. The implementation of this tool was inspired by Yahoo! Pipes but the advantage to this tool (in contrast to Yahoo's tool) is that DERI pipes can handle the RDF format and thus enables to build semantically enhanced mashups. DERI Pipes does not need any knowledge of programming languages but requires an understanding of data formats such as RDF. The final output respectively mashup is defined in XML or RDF and can be published in order to share with other users.

[35] http://www.json.org/

[36] In 2009 Google has closed the Google Mashup Editor. Therefore we are not considering this.

Figure 12 shows the user interface of DERI pipes. In this example the data of Tim Berners-Lee is mixed from three different sources.

	Presentation	Data	Functionality	Extraction	Flow	Consumer	Enterprise
Apatar		×			×		×
Data Mashups	×	×			×		×
Dapper	×			×		×	
DERI pipes		×			×	×	
Grazr	×			×		×	
IBM InfoSphere MashupHub		×		×			×
Intel MashMaker	×				×	×	
JackBe Presto		×	×		×		×
Microsoft Popfly	×	×			×	×	
Openkapow	×			×		×	
Procession		×	×		×		×
Rssbus	×			×	×		×
Serena Mashup Suite		×	×		×		×
Snap Logic		×			×		×
TIBCO PageBus		×			×		×
Yahoo! Pipes		×			×	×	

Table 1: Classification of Mashup Tools

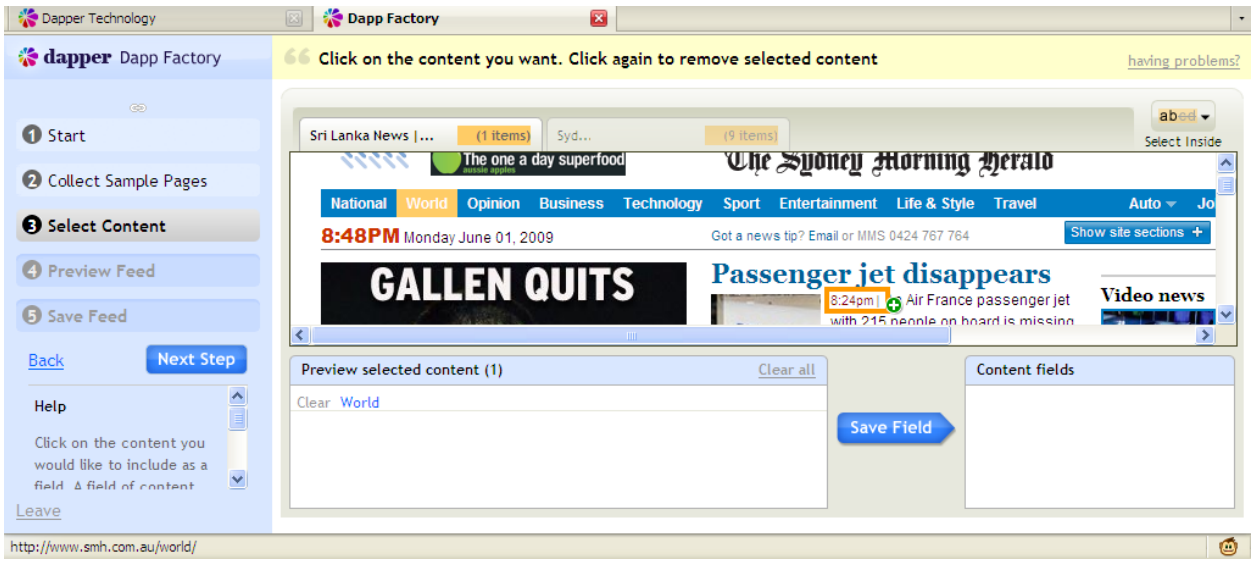


Figure 11: dapper Dapp Factory

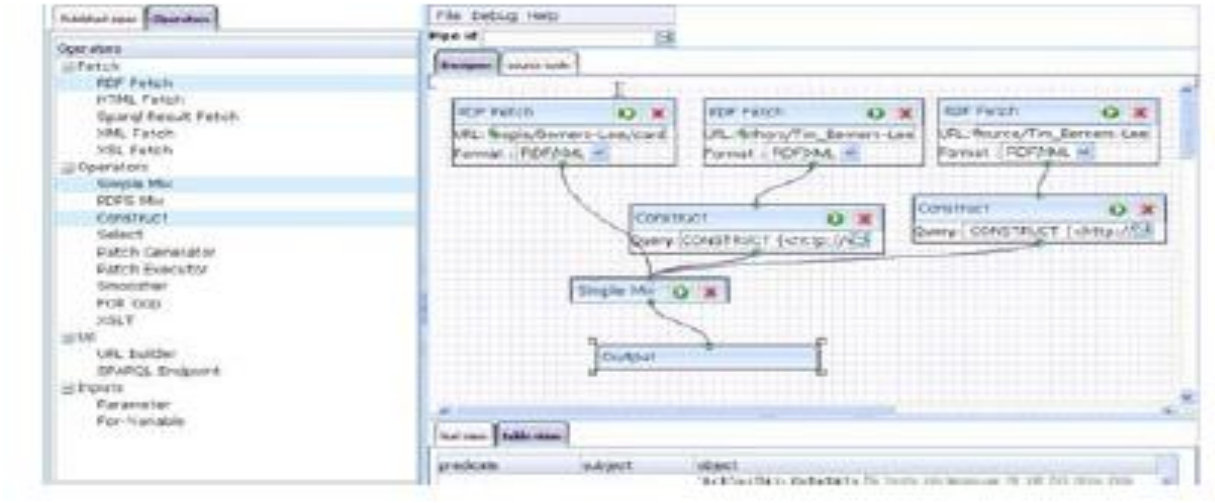


Figure 12: DERI Pipes

Data, Functionality, Flow and Enterprise mashup tool: Example for a tool providing functionalities to create enterprise mashups is Serena Mashup Composer included in the Serena Mashup Suite. According to the tool vendors Serena considers the integration of information, business processes and data to one common representation. Figure 13 shows an example for the creation of a vacation request mashup. The idea of that mashup is that users can submit their vacancy request with several devices such as laptops or blackberries. The used syntax for the mashup creation resembles a service language such as BPEL [1]. But the difference is that Serena Mashup Composer can consume and mix any kind of widgets (feeds, plain HTML, services).

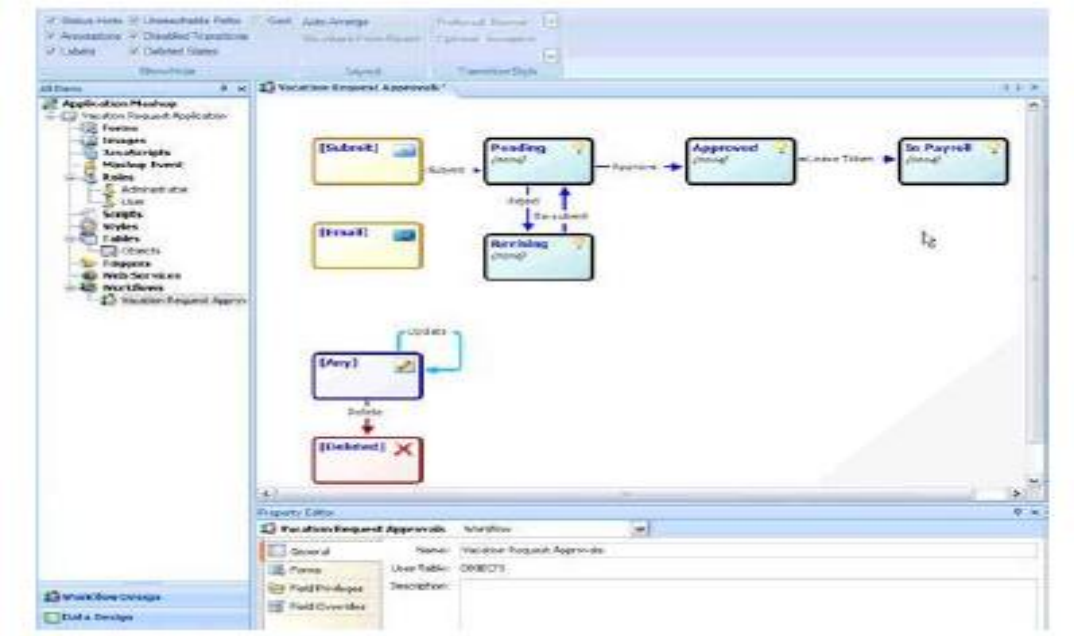


Figure 13: Serena Mashup composer

Presentation, Data, Flow and Consumer Mashup Tool: Example for such as mashup is Microsoft Popfly[37]. Microsoft Popfly uses Microsoft Silverlight[38], which is necessary since Popfly uses a great amount of optical effects of Silverlight. Popfly is not restricted to the generation of XML files but also offers possibilities to show the data (e.g., by using modules such as Microsoft Virtual Earth). To use Microsoft Popfly also does not require any skills of programming languages but the user should come with an understanding of data formats. Figure 14 shows an example for creating a mashup with Popfly. In this example the Facebook API is used to visualize pictures of Facebook in a carrousel.

[37] Microsoft Popfly. http://www.popfly.com/

[38] Microsoft Silverlight. http://www.microsoft.com/SILVERLIGHT/

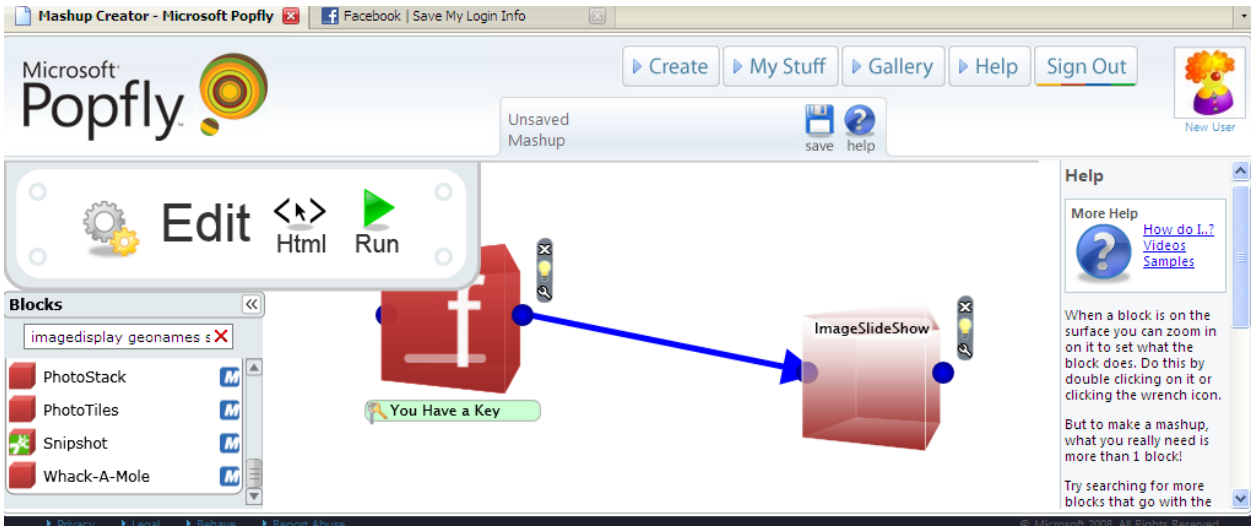


Figure 14: Microsoft Popfly

Language to create mashups; Several languages have been proposed for the construction of mashups [20, 22, 26]. Among all these languages, Orc[39] seems to be best documented and developed. The Web page of Orc offers an editor for the code programming. The code can be directly executed because the editor is connected to a server. Orc is also available as a standalone JAR file or Java application.To write a mashup application in Orc the user should come with knowledge in functional programming languages since Orc is a concurrent functional programming language. The authors of Orc define three appropriate application areas.

Orc can be used as a general purpose programming language for concise encoding of concurrent and distributed applications, as a Web scripting language to create a Web service mashup and as an executable specification language for workflow applications and process coordination problems.

To be used as a language in the workflow field Orc implements several workflow patterns [27]. Thus, Orc is suitable to build process-oriented mashups. Figure 15 shows an example of Orc syntax. The user is simultaneously searching in Yahoo and Google for a term that can be posed by the user after running the application. The ability to pose a query argument is given by the pre-defined method *Prompt*. The search field will appear when the user pushes the run button (already performed in this example). The classes *Yahoo* and *Google* are pre-defined in the "search.inc" library. The authors of Orc have already defined several libraries but the missing documentation of these libraries hamper the coding with Orc.

[39] Orc Language. http://orc.csres.utexas.edu/

Figure 15: Orc coding

3.4 Discussion

The reviewed mashup makers have many features that improve usability and learnability. Their support for online communities is especially good. However, there are several potential areas of improvement and future research required. Although the mashup makers provide many notations for different tasks and different complexity levels, as well as a variety of learning support features, high learning barriers still remain, especially between different levels of abstraction and different notations.

Evaluating mashup makers according to the classification models will help to identify identify the areas that need most improvement, as well as typical problems users encounter in developing mashups. This form of evaluation can be done via user studies.

From this research we think overall support for software engineering techniques such as testing and debugging in mashup makers is quite limited. Given the concerns regarding security and correctness of applications developed by non-programmers [19] and the growing popularity of mashups, these shortcomings are dangerous, especially in the context of enterprise mashups. Non-programmer oriented debugging and testing facilities such as debuggers, test frameworks and assertions are researched in the area of end user software engineering (e.g. [21, 12]). Applying those concepts to mashup makers could alleviate the dangers of incorrect and unsafe mashups.

4. Challenges for Mashup Construction

To establish mashups as an efficient technology for resource integration we think the following challenges should be solved, where some of these challenge is not only unique to mashups but also to the World Wide Web.

1. **Cataloguing**. Some Web pages are already available that list mashups and provide an interface for searching of mashups such as `programmableweb.com`. Mashup creators can insert their mashups in the list and thus share their mashups with others. But what is missing is a directory that stores and catalogues the mashups in a consistent way.

2. **Data integrity**. Mashups are a quick way to create new applications but they can raise data integrity problems when changes of end-users are not valid against the underlying commitment. Another concern may raise integrity problems if e.g., an end-user finds a service that brings some value to the data or functionality included in the mashup. Then mashups need to be modified at runtime. Thus, when running a mashup control mechanisms should be considered that ensure the integrity of the mashup against end-user changes.

3. **Making data Web-enabled**. Mashups are constructed of different resources that are available on the Web. However, currently a lot of data and functionalities are not set up on the Web and thus are not accessible via feeds, HTML or Web services. To make more resources "Web-enabled" require formats and tools that facilitate an efficient access and the connection of resources to the Web. Additionally, some data that is available on the Web cannot be reused to "mashing" because the data is capsulated with the presentation layer. Thus, mechanisms are needed that support the creation of mashups out of data and also tools that offer functionalities to decouple data from multiple sources from their presentation.

4. **Security and identity**. While security challenges have been identified for mashups [18, 5] only few approaches exist that try to handle security lacks and identity of mashups [6, 17, 11]. Lawton [18] sketches that security challenges emerge when end-user connect dynamically to Web sites and not necessarily under the provider's control. Additional security challenges arise if the mashup contains confidential data or security log-ins is required to enter some data. This requires mechanisms to control the user connection and the data security.

5. **Sharing and reusing**. The next concern for mashup construction is that vendors of mashup tools should provide mechanisms to allow end-users sharing their built mashups with others and thus facilitating the reuse of pre-built mashups. This means also that mashup owners need to give their permission before making the mashup available for the community. Otherwise end-users have to face legal implications of using this technology and have to expect consequences [6]. To facilitate a (legal) resource sharing the mashup should be defined in a format that is readable by different machines and consider accountability. Challenges that have to be met in this context are an easy-to-use access to mashups, efficient mashup search functionalities and lightweight formats that enable even for non-programmers a smooth mashup reuse.

6. **Trust certificates**. The owner of such a directory service can issue a license that certificates the mashup. Because so far, no certification mechanisms exist that guarantee end-users the trustworthiness of the mashup. Similar to trust certificates for online shopping it is imaginable

that mashup owners grant a licence at the owner of the directory service. In case of a positive certification the mashup owner can assure end-users the trustworthiness of the content and also the integrity of the mapping application.

Version control mechanisms. Mashups consist of different resources collected from various sources. Resource owners are responsible for their content and can change and update its content or respectively its software whenever they regard it as necessary. To keep the mashup content up-to-date a version control mechanism is required that automatically informs the mashup owner about updates of the integrated underlying software (imagine the mashup is build upon several APIs)

5. Conclusion

Mashups are suitable to build novel Web applications and to create new forms of visualization without little knowledge of programming languages. The lists of mashups created so far cover the whole range of Web applications (e.g., finance, government, sports or security). However, little attention has been paid to classification models.

The aim of this paper was to define a consistent understanding of mashups starting with a definition and a classification model for mashups. We presented several tools that support building mashups. From our point of view further research is especially needed in the fields of version control mechanisms, mashup certification, mashup quality and data integrity.

6. Reference

[1] Alves, A., and Arkin, A., and Askary, S., and Barreto, C., and Bloch, B., and Curbera, F., and Ford, M., and Goland, Y., and Guzar, A., and Kartha, N., and Liu, C.K., and Khalaf, R., and Knig, D., and Marin, M., and Mehta, V., and Thatte, S., and van der Rijn, D., and Yendluri, P., and Yiu, A.: Web Services Business Process Execution Language, Version 2.0, Specification (2007).

[2] Dangermond, Jack. 2008. GIS and the GeoWeb. *ArcNews*, Summer.

[3] dapper: The Data Mapper. http://www.dapper.net/

[4] David Gootzit, Gene Phifer, Ray Valdes, Nikos Drakos, Anthony Bradley, Kathy Harris, Daniel Sholler, Massimo Pezzini, Yefim V. Natis, Bill Gassman, David Mitchell Smith, David W. Cearley, Roy W. Schulte, Stephen Prentice, Nicholas Gall, William Clark, Anne Lapkin: Hype Cycle for Web and User Interaction Technologies, (2008). Gartner Publications id: G00159447

[5] Davidson, M. A., and Yoran, E.: Enterprise Security for Web 2.0, Enterprise Security for Web 2.0, 40(11), 2007, pp. 117-119

[6] De Keukelaere, F., and Bhola, S., and Steiner, M., and Chari, S., and Yoshihama, S.: SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers, In Proceeding of the 17th International Conference on World Wide Web, Beijing, China, ACM Press, 2008, pp. 535-544.

[7] DERI Pipes: Open Source, Extendable, Embeddable Web Data Mashups. http://pipes.deri.org/

[8] Di Lorenzo, G., and Hacid, H., and Paik, H., and Benatallah, B.: Mashups for Data Integration: An Analysis, School of Computer Science and Engineering, University of New South Wales, Technical Report 0810 (2008),

[9] Dion Hinchcliffe: Is IBM making enterprise mashups respectable?ZDNet Blog, 2006. http://blogs.zdnet.com/Hinchcliffe/?p=49&tag=nl.e622

[10] Erle, Schuyler. 2006. Web Map API Roundup. *Mapping Hacks*. April 7.

[11] Fielding, R. T.: Architectural styles and the design of network-based software architectures, University of California, Irvine, 2000, Dissertation.

[12] Gibson, R., Erle, S., *Google Maps Hacks*. O'Reilly Media, 2006.

[13] Goodchild, Michael F. 2007a. Citizens as sensors: the world of volunteered geography. *GeoJournal* 69, no. 4: 211-221.

[14] Goodchild, Michael F. 2007b. Citizens as Voluntary Sensors: Spatial Data Infrastructure in the World of Web 2.0. *International Journal of Spatial Data Infrastructures Research* 2: 24-32.

[15] Goodchild, Michael F. 2008. Spatial accuracy 2.0. In *Spatial Uncertainty*, *Proceedings of the Eighth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, 1:1-7. Shanghai: Liverpool, World Academic Union.

[16] Hoyer, V., and Fischer, M.: Market Overview of Enterprise Mashup Tools. In International Conference on Service oriented Computing, Volume 5364 of Lecture Notes in Computer Science, Springer-Verlag, 2008, pp.708-721.

[17] Jackson, C. and Wang, H. J.: Subspace: secure cross domain communication for web mashups, Proceedings of the 16th International Conference on World Wide Web, Banff, Canada, ACM Press, 2007, pp. 611-620.

[18] Lawton, G.: Web 2.0 Creates Security Challenges, In: Computer, 40(10), 2007, pp.13-16.

[19] Li, S., and Gong, J.: Mashup: a New Way of Providing Web Mapping and GIS Services. In ISPRS Congress Beijing 2008, Proceedings of Commission IV, 2008, pp. 639-649.

[20]Maximilien, E. M., and Wilkinson, H., and Desai, N. and Tai, S.: A Domain-Specific Language for Web APIs and Services Mashups. In Proceedings of the 5th International Conference on Service-Oriented Computing, Volume 4749 of Lecture Notes in Computer Science, Springer-Verlag, 2007, pp. 13-26.

[21] O'Reilly, Tim. 2005. What Is Web 2.0. September 30.
http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
(accessed March 16, 2009)

[22] Sabbouh, M., and Higginson, J., and Semy, S., and Gagne, D.: Web mashup scripting language. In Proceedings of the 16th International Conference on World Wide Web, Banff, Canada, ACM Press, 2007, pp.1305-1306

[23] Singel, Ryan. 2005. Map Hacks on Crack. *Wired*, July 2.

[24]SOAP Version 1.2 Part 0: Primer, W3C Recommendation,
http://www.w3.org/TR/2007/RECsoap12-part0-20070427/ , 27 April, 2007.

[25] Vikram, K., and Steiner, M.: Mashup component isolation via server-side analysis and instrumentation. In Web 2.0 Security & Privacy Workshop. IEEE Computer Society, Technical Committee on Security and Privacy, 2007.

7. Appendix