

POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Gestionale



**DEFINITION AND VALIDATION OF A QUALITY
MODEL FOR MASHUPS**

Relatore: Cinzia Cappiello

Tesina di laurea di:

Alejandro Acuña
Juan José Cuéllar

Matr. 735365
735382

Academic Year 2009/2010

ACKNOWLEDGEMENTS

RESUME

This Thesis has its foundations on the Information Technology field, focusing specially on the part related to the development of Web Technologies, and on the use of Application Programming Interface for the creation of customized Web tools to better satisfy customers needs and expectations, i.e. Mashup APIs (Application Programmable Interface) into a single interface that complains with the user needs.

The so called mashups are tools created by programmers, represent a new paradigm according to which it is possible to get a greater benefit from the APIs available on the web; its importance lies on the fact that its use, besides being a growing trend, represents significant savings in terms of time and resources, whether its use is industrial or public.

As in the use of any other tool the results and benefits that can be obtain from it depend mostly on the quality of the tool itself; therefore an analysis to assess the quality of Mashups has to be run in order to define wheatear or not the tool will be able to meet user expectations. In the particular case of mashups measuring this quality requires a deep analysis of the structure and characteristics of the mashups; this analysis is carry out through an understanding of its components and the interactions that take place among them in order to provide the final user with quality applications.

The objective of this thesis is to describe and validate our quality model for mashups, which privileges properties of the component APIs. It states that in order to assess the quality of a mashup, especially of the information it provides to its users, it is required to understanding how the mashup has been developed, how its components look like, and how quality propagates from basic components to the final mashup application. An analysis has been perform to get to the core elements that

influence quality in each area. In this process other valuable data arise like the most common elements chosen by the mashups authors and the interactions among them.

INDEX

1 INTRODUCTION.....	3
2 MASHUP, THE PHENOMENON	7
2.1 DEVELOPMENT OF WEB TECHNOLOGIES	7
2.2 THE ORIGINS	11
2.3 THE RISE OF WEB MASHUPS.....	13
2.4 MASHUPS	14
2.5 MASHUPS WITHIN THE ENTERPRISE	17
2.6 QUALITY IN THE MASHUPS	19
3 RESEARCH & ANALYSIS.....	23
3.1 QUALITY MODEL PRESENTATION	24
3.1.1 <i>Component quality</i>	25
3.1.2 <i>Composition quality</i>	31
4 QUALITY MODEL VALIDATION.....	43
4.1 SURVEY.....	43
4.1.1 <i>Validation of the Quality model</i>	43
4.1.2 <i>General Information Analysis</i>	45
4.1.3 <i>Component Quality Results Analysis</i>	48
4.2 COMPOSITION QUALITY.....	67
4.2.1 <i>Mashup List Analysis</i>	68
4.2.2 <i>Association Rules</i>	78
5 CONCLUSION	85
ANNEX.....	90
5.1 ANNEX “A” LIST OF MASHUPS	90
5.2 ANNEX “B” API / SUBSTITUTES	93
5.3 ANNEX “C” MASHUPS PATTERN LIST	95
5.4 ANNEX “D” MASHUP QUALITY SURVEY	97
5.5 ANNEX “E” MASHUP QUALITY SURVEY, RESULTS	102

1

INTRODUCTION

For many years the construction of web pages and the elaboration of programs and software, has been an exclusive activity of people with high programming skills; those with the knowledge on programming language, capable of translating their ideas and/or the needs of the user into code, in order to create a tools with the characteristics necessary to perform certain useful tasks and operations that will cover a certain need or want coming from the clients. Over the past decade some changing trends have emerge on relation to this matter; now, computer applications have become more comprehensive, i.e. have become modular on its use, and also the Internet has turn into a more accessible and friendly environment to users; also with this changes the opportunity to innovate on the creation of new application has growth, not just for high skilled programmers but also for the amateur ones, whom with little or absent programming knowledge, but whit a lot of creativity and ideas to put in motion, can aspire to create a new tool on the Web. More and more, the Web is being used not only as a portal for information but also for application-to-application communication.

This change towards a more accessible Web, has modified the way in which users behave and interact among themselves and with the Web, also has promoted the development of a variety of new technologies (e.g. Blogs, Wiki's and Web Services). Web Mashups have emerged as the newest Web technology and have gained lots of momentum and attention from both academic and industry communities. Mashups are part of an ongoing shift towards a more interactive and participatory Web (Web 2.0). People with a little skill can create new applications using common elements found lying around the Web in almost no time at all. As the skill requirements for building these applications are decreasing, we think this opens a whole new world of possibilities (*Spool J. 2007*).

Mashups represent a wide area of opportunity for innovation, which could manifest in many different ways, such as new technologies, new business models, and imaginative new ways to share and take advantage of the available data. In order for this to happen, people have to be able to share information, using same language, same code, that will allow them to interact with each others; it is this interoperability¹ the one that has exploded through Mashups development. This interoperability of the Web services² has enabled the creation of numerous Mashups, of many different types, useful for important sectors of the society, from individuals to enterprises.

Taking into consideration its increasing importance, the growing use of mashups and the opportunity that they represent as tools, it becomes essential to have the knowledge, tools and elements that allow the qualification of mashups in a proper way. It is therefore the aim of this work to introduce our quality model for the qualification of the components and the mashups itself, in terms of different but all important variables for this end. Also this work has the objective

1 Interoperability. Set of conditions, including compatible technologies and willing participation by Web services and data providers, that permit developers to create mashups (*Palfrey, Gasser. 2007*)

2 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. (*Palfrey, Gasser. 2007*)

of validate the previously mention model, through a series of researches and analysis, that will lead to a deeper comprehension on what respects to the most common patterns use by developers, its preference and needs with respect to the components use (APIs) and the quality and frequency of use of the components themselves, information that will be use for the validation of our quality model.

In order to offer a wide perspective and the whole information necessary with respect to main interest of this thesis, we also introduce some necessary elements to comprehend the development of web technologies, its evolution to what we know as Web 2.0, to the birth of the mashups on the Web. After the definition of important things that are necessary to know and comprehend, like Web 2.0 and the APIs (both cornerstones in development of Mashups) we can submerge ourselves in the subject center of this paper validate the quality model for mashup components, “Quality Mashups”.

In chapter 2 we take a look at the Mashups as a phenomenon; as such we talk about the beginning of this phenomenon, its growth and evolution; the difficulties and oppositions that had to be faced before becoming accepted by all the parts involve in the creation process; they are described the implications of this specific practice of remixing and recombination. Then we center ourselves in the description of the Mashup, the tool; the current environment and all the players around the phenomenon. Finally it is introduced the Mashup phenomenon from a different perspective, the business one; it is describe the use and the potential advantages hidden behind the use of mashups on such context. And we also look at the previous quality works made that propose quality models for Web applications and to modern Web 2.0 applications and the current one regarding the mashups.

Chapter 3 is the core of this thesis, in it we introduce the Quality Mashup Model and its validation proving that the stages, components, dimensions and attributes specified on it are in fact real well-place elements to measure the quality in a mashup, and this is proved by applying a deep wide survey among programmers and by an

analysis made to all the categories of mashups that are available and deployed in the web and by doing this analysis it is proved that the composition of the components in a mashup and their roles as explain in the model are the ones in fact used in all the mashups available in the web so far.

In chapter 4, we present the results of the research, the conclusions of the investigation and the analysis of the information gathered.

Finally in the Appendix we include the tables we built in order to analyze the mashup patterns and also the table that contains the results of the survey.

.

MASHUP, THE PHENOMENON

2.1 Development of Web Technologies

Web 2.0, is the terminology used to define a certain set of different but interconnected trends occurring on the Web since the beginning of year 2000. Web 2.0 is a paradigm of user participation driven by modular and customizable online services (*O'Reilly 2005*).

The tools that conform Web 2.0 empower the consumer to become a publisher online, just as a newspaper publishes content, now there is no difference, any consumer can publish this content, available freely to anyone who chooses to find it. Web 2.0 also allows a two way conversation, so instead of being just one who speaks at the consumer, the consumer now has the opportunity to speak back at one, in a public way for everybody to see.

With the development of the Web 2.0, creating a personal web presence was easier than ever thanks to the appearance of blogs and social networking sites, such as Facebook and MySpace, but also facilitates the easy exchange of information and

media with friends and acquaintances. Through the user-generated content sites, such as YouTube, it became possible the sharing of content with everyone, not just within social networks. The social bookmarking sites (e.g., del.icio.us, Digg and StumbleUpon) enable the sharing of links with friends and like-minded strangers, forming a rich layer of interconnected annotations and tags on top of the “content” layer of web pages and multimedia. Finally, user-editable websites known as wikis have facilitated the collaborative development of large, complicated projects, of which the most notable is Wikipedia³.

As we can see, Web 2.0 is a collection of approaches that give application developers a new way to address problems, difficult to crack with outstanding effective results. (*O'Reilly 2005*). All this in contrast with the traditional Web of monolithic sites providing information to an audience of passive observers (exemplified by web portals such as Yahoo and e-commerce sites like Amazon.com).

With traditional Web there were communication tunnels, to get e-mail, this goes from one person to another, one of them could have the answer or the information that the other one is needing, and so the question and the answer travels across this tunnel that is the e-mail; but what if there is also the possibility that there is somebody else that has the same question? Or someone else that could provide some additional valuable information? then another tunnel has to be created, but at the end this valuable information stays within two people that are connected by e-mail; it is here where Web 2.0 has modified the experience of exchanging information, the conversation is not any more between two people, it happens in public, it could be in a blog, in a forum or on a FaceBook wall; people who has a question can publish it and it will be saved there, and it would be like asking something to a group of people at the same time, anything that is published will remain there for years, and it can be information that at a certain point would be

³ Wikipedia. A free encyclopedia created and maintained completely by volunteers.

valuable for someone else, and this person might use it even if it has been a long time since its publication.

Speaking about the number of persons that are reachable, traditional Web was more the type of close end communication. While in Web 2.0 our question, our opinion, what we have to say will remain there, whether somebody answer it or not it still will be there, the communication will remain open.

There are some authors that have characterized the shift to Web 2.0 as one from a “read-only web” to a “read-write web” (Lowe, 2008).

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

Figure 1 Application approach, Web 1.0 & Web 2.0 (O'Reilly 2005)

While a core part of Web 2.0 is the idea of interconnected services, perhaps the more significant change has been the rise of user-created content and social networking. Modern Web 2.0 applications are characterized by a high user involvement: users are supported in the creation of content and annotations, but also in the “composition” of applications starting from content and functions that are provided by third parties (Cappiello et al, 2010).

As part of the wave of Web 2.0 technologies, mashups represent a shift toward distributed authoring and sharing of Internet content.

There are two main components required for the successful creation of a Mashup, first, the data, and second, the application programming interfaces (APIs), these last will provide us with an interfaces that will concede, even for those with low programming skills, the opportunity to work with a flexible form of the data.

APIs are part of the operating system on which relay the responsibility of perform such basic functions accessing the a system file. The API will have a compilation of predefined requests that will be use by one software to request another one to do some things for it. In other words an API will be the way in which a developer will request the program to perform a certain task or action, is the way to information exchange.

The diversification of Web services, the creation of social networks and Web sites that allowing the sharing of knowledge, opinions, media and of course the construction of Mashups, is possible due to the increasing availability of a single tool: APIs.

APIs have been around for a while. What is interesting to us right now is that we have arrived to a certain point, where our understanding of them and the tools available to implement them make it possible to create powerful applications very fast and relatively inexpensively.

A great example of how APIs can open up new opportunities for developers to expand the user experience in the Web, is the Google Maps API; since it provides a rich interface, developers of these new applications don't have to dedicate resources to building a mapping system and populating it with geographic data. It already exist and is available. Instead, the developers can focus on their data source and how they want to overlay it.

A newspaper article interviewing Google Local product manager Bret Taylor about the significance of Google's "open code" puts it this way: "Google recognized while developing the mapping feature that it would not have the time or the desire to

create a host of special interest maps. Yet having numerous Mashups would serve Google's strategy of becoming the ubiquitous organizer of the world's information⁴ - hence its openness.[...] (In fact, in exchange for allowing use of the maps, Google reserves the right to run ads on the sites in the future.) 'It's great for the developer and it's great for Google,' Mr. Taylor said" (*Darlin 2005*).

The availability of data sources via open APIs increase the innovation potential of building on top of Web services technology.

2.2 The Origins

When the term "Mashup" first appear, it was originally used to make reference to a certain type of songs that were born as the result of mashing two different styles of music. This happened when the technology capable of editing and recording digital music emerge, now everyone was able to create their own songs, and also to share them and thanks to certain web sites even distribute it. The music mashup, as a phenomenon, has a peculiarity that differs from what before was going to be known as the Web Mashups, the creativity of these was limited to the task of taking parts from already existing songs and putting them together without adding anything new; in other words it consist exclusively in the manipulation of the exiting data (or in this case digital media).

A few years ago, the same concept of mashup was applied to the World Wide Web, which in the general idea was the same principle (users could take data form different sources and put it together into a single interface) but also, unlike music mashups, user could integrate their own data into existing web pages (*Lowe, 2008*).

⁴ The Google Maps API grants anybody the power to overlay any data onto any place Google Maps can show.

"They're taking little bits and pieces from a number of companies and stitching them together in some clever way," Amazon Chief Executive Jeffrey P. Bezos noted recently. "You'll start to see the real power of Web services."(Hof,2005).

In the beginning, the relation between the Web pages owners of the APIs and the mashup developers was not the best, just as it happened with the mashups in music, the owners of the API were trying to protect their ideas and to avoid its use by people from outside their own Web page or company. But there was pressure from programmers to the owners of the Web applications, these bottom-up efforts present strong challenges for the Web sites; some of them decided to encourage the development and growth of the mashup phenomenon, while some others try to prevent the use of their data and code. Mashups developer often use the code from big Web companies without asking first and then present it in unintended ways, as a consequence some of these Web companies reacted by blocking the information exchange from their sources to protect themselves. Another reason why companies reacted that way, is because they did not see any profit on letting others use their software, and even now, some years later, mashup business models don't extend beyond running a few Google ads and collecting fees for sending buyers to e-commerce sites. One reason is that most Web sites don't allow for-profit use of their data by outsiders. But in any case, none of this conflicts between companies and developers have been able to stop what was already in motion, the mashup phenomenon. On the other hand as traffic to mashups grows, companies may cut deals, especially if mashup sites spur new markets.(Hof, 2005)

There are some big companies on the Web, that now are granting easier access their codes, data and services, accepting this way the mashup wave. With mashups this big companies have the opportunity to take advantage of the creativity and work of thousands of people who develop mashups on the Web on a daily basis, and not just the creativity of those who work within their company. Encouraging the participation of the people on the development of new mashups is a way in which

companies can also make publicity for themselves. Furthermore, most Web companies are now programming their services so that more computing tasks, such as displaying maps onscreen, get done on the users' PCs rather than on their far old servers, other way to take advantage of the user involvement.

2.3 The rise of Web Mashups

It was Google, in July 2005, one of the first players on release its data to encourage the development of mashups using as a platform its well now Google Maps service. Its API (Application Programming Interface) present empty geographic picture of the world, in which developers could deploy its own data. Now the developers could have access to the software behind the service, they had the opportunity to manipulate all the elements within the Google Maps API (data, interface and iconography), along with all the functionalities proper of the API (zooming and scrolling); all these elements were now at the hand of developers ready to create their own maps, with the main advantage of not having to write their own software in order to support the map service.

After Google's decision of making available its Google Maps API, the response from its Yahoo similar was immediate; just a day before the release of Google's API, Yahoo putted out its own Mapping service API. Thanks to the APIs that now are at everybody's disposition on the internet we can say that everyone can customized its own applications and also it is the beginning of a new era on Web applications development, the rise of mashups.

On the contrary to what we all might think, this public release of APIs from Google and Yahoo, did not mean that until then there wouldn't had been any attempts for creating mashups, in fact the first mashups already existed by that time; they were created by skill programmers that were able to decode parts of the APIs

that were not yet public domain, but still available for those skilled enough to get them.

It was in response to these initial mashup activities, that Google had ultimately decided to publish the API, explained Bret Taylor, Google Maps product manager, “because they were already doing it” (Singel, 2005).

So far we have defined the “environment” that allows the user to interact in a more dynamic way with the systems and with other users, where sharing data and communication is a simple task that everyone can perform it; also we now know that the APIs are an indispensable element for the creation of mashups, its accessibility has given the opportunity to anyone with access to internet, to innovate and manipulate the information in order to better satisfied their own needs or the needs of others. We know the factors that detonated the phenomenon of Web Mashups, and how the feeling towards it has been evolving, from the different perspectives (code owners and developers). Now we will take a look at the Mashups as a whole; we will understand the implications of this specific practice of remixing and recombination.

2.4 Mashups

Strictly speaking, mashups online are hybrid web applications, combining disparate data sources and web services in ways that are not how they were intended (Lowe, 2008). For example, a Web forum may contain a mashup that uses Google Maps to display what parts of the world the users are posting from. Yahoo offers a mashup called Yahoo! Pipes that aggregates RSS feeds into a single page that can be navigated using a graphical interface.

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 (*Palfrey, 2007*)

With the origin of mashups has come to change the way in which users interact among themselves, with Web pages and with companies. The Web is now something more than just Web sites to be visited, is changing into a system in which everyone participates and interacts; now the control of the Web has move from companies to users in general. Currently mashups are being developed by the people who day by day navigates the web for other people to be used, they are not anymore the result of big companies programmers.

The primary purpose of most Web mashups is to consolidate information with an easy-to-use interface, tools that help make information easier to find. Emerging technologies, such as Web services, user interface (UI) widget⁵ libraries, and tool-specific mashup (meta-)⁶ models have significantly simplified the access to and the reuse of such kind of building blocks, leading to a component-oriented paradigm that is shared by many of the current mashup platforms. This paradigm especially facilitates the development of so called situational applications⁷, which aim at answering a precise query over a limited but heterogeneous data space. (*Cappiello et al, 2010*). Mashups are part of an ongoing shift towards more a more interactive and participatory Web 2.0. Because the combinations of Web applications are limitless, so are the possibilities of mashups. (*TechTerms.com, 2005-2010*). "The Web was originally designed to be mashed up," says Google Web developer Aaron Boodman, the 27-year-old creator of a program called Greasemonkey that makes it easy to

5 Widget. portable chunk of code that can be installed and executed within any separate HTML-based web page by an end user without requiring additional compilation (*Wikipedia, 2010*).

6Meta-modeling, in software engineering and systems engineering among other disciplines, is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for modeling a predefined class of problems (*Wikipedia, 2010*)

7 Situational Applications. Applications that: serve a highly focused purpose (e.g., visualize apartment offers onto a map), are intended to be used for a limited time horizon (e.g., until a suitable apartment has been found), and where the developer is also the final user (*Jhingran, Enterprise information mashups: integrating information*)

create and use mash-ups. "The technology is finally growing up and making it possible." (Hof, 2005)

An operating system is a collection of API (application programming interfaces) that developers use to build their applications and is also user interface. This APIs make it easier for a developer to build their applications, in the old days developers had to say where every dot had to be placed on the display, today they only request a window from a specific coordinate to another, and suddenly the window appears, this is what APIs do they do all the heavy lifting. Therefore, to mash-up, is to take APIs from multiple websites and merging them into a new innovating application.

Mashups are applications developed by integrating content and functionality sourced from the Web. While in most cases mashups are still applications that are hand written by enthusiastic programmers, especially the recent emergence of so called mashup tools or mashup platforms, such as Yahoo! Pipes, Dapper, or Intel Mash Maker has significantly lowered the barriers to the development of mashups, enabling also unskilled Web users to easily assemble their own applications on the Web (Cappiello et al, 2010).

Challenges arise in developing an understanding of further ways to interoperate, what the stakeholders⁸ 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

⁸ Person, group, or organization that has direct or indirect stake in an organization because it can affect or be affected by the organization's actions, objectives, and policies(*businessdictionary.com*)

As players in the system, these stakeholders bring a variety of use cases, motivations and needs to the table.

2.5 Mashups within the Enterprise

One of the reasons for this great develop and rapid increase is that you don't have to be a C programmer to tap your creativity and bring something innovative. It is this simplicity on the creation process that it is making mashups to overcome any other IT⁹ system growth rate, you don't need to go through anybody like a director of product manager to approve a new API. The advantage is that in the internet you don't have to go through anybody to approve what you done, your API can be post and be available to all the developers that want to use it, and could be related to a large number of mashups in the internet.

Creating a Mashup is a creative process in witch one see some data on the application and you take it and combine it into something new, this enable business users to take pre-configured software components or APIs and chain them together to create applications. In the context of enterprises there is a large amount of custom built applications that enterprise users want but IT is unable to deliver. IT is been focus on infrastructure and putting the service components in the play, the idea would be to provide an environment trough witch end user can actually create the applications from service components that IT is providing.

The data that people need when building mashups is one of the most important aspects that need to be taken care of, in an enterprise scenario, the challenges from a data perspective are very similar to the challenges the developer face in the Web 2.0 world, so the idea within the enterprise is to reduce complexity, making it easier the integration of data, to the extent that this happen, we are going

⁹ Information technology (IT). Is a general term that describes any technology that helps to produce, manipulate, store, communicate, and/or disseminate information. (*Wikipedia,2010*)

to have more mashups within enterprises, and could start taking advantage of the benefits. The real value of mashups comes when we enable and empower our end users to perform these mashups the way they want to do it without having to call and rely on IT needs, mashups must be user driven and user focused. It is about giving a business, self service IT to solve their problems, something similar to what they currently do with excel sheets which is hard; consider how one uses a spreadsheet which is a great example of a canvas where mashups happen in the enterprise when a user pulls in data from different sources, combines, sorts, computes, filters and visualizes by iteratively processing it until he gets the desired results. If you really think about it, what the user is doing is really orchestrating getting data from different sources and processing using one or more operations to yield an integrated view of the data, making all these easier for them is what mashups is all about.

The heart of the problem is that the economic cost of production for applications in traditional IT is been so high that they have limited the scope and their involvement to only those applications that have big impact across different functions of the enterprise; with mashups the enterprises can reduce in a successful way the cost and time it takes to deliver an application from concept to actual user. This application development could happen anywhere inside the enterprise.

Another benefit of mashups is being able to quickly see something in a way that makes sense to you as business users, things that IT might have already miss, you have now the user as a part of the creative process, prototypes having a larger field for an application and also at the same time creating a feedback mechanism.

Taking a look at the extension capability of mashups we can understand its power; we can take something that we already have and add something else to it which gives a greater context and value to me as user and we can do that instantly, translating it into business value. But Just because mashups are user driven and user

focused does not mean that we no longer need IT. IT is still a critical part of the whole mashup infrastructure in an enterprise. (Alur, 2007)

The data source used to create the mashups is located inside and outside the firewall of the enterprise, so security can become a concern, providing the adequate tools would help users create the mashups they need in a control and measured way; an enterprise can monitor what kind of data people are using, and it gives us a tremendous insight into what the business users actually want. It is important to recognize that the security risks from somebody accessing a customer data base is not that they are accessing, but that they are downloading thousands of information into an excel work sheet because the functionality is not there for them to do what they want to do on the online side. So with mashups we are not opening up any of these sources any more than what they are already open, people create their mashups from data that they can already access.

One of the dominant trends in IT right now is the service oriented architecture, the idea is to publish the data sources that the enterprise wants people to use as services, within a service oriented architecture, and this creates a library of services that are available to people just to do mashups with.

If you look at enterprise mashups, many of the individual projects are so small that there is no way that a single particular project can pay for the product, so the way to get the investment paid is like in the head of IT, with the big IT projects, that will pay for the product and then spread it into the organization with all the small projects.

2.6 Quality in the Mashups

The Quality in the mashups is basic in order to have a proper output that fulfills the requirements and expectations for all the parts involved programmers and users, therefore the quality of the elements to be used and the quality of the mixing will be

related to the quality of the output in other words assessing the quality of a mashup requires understanding both the quality of components and the effect that the composition has on the overall quality of the final mashup, quality is highly relevant in mashup development, the quality of a mashup is sensitive to both the quality of its components and the way components are integrated. Whereas components with low quality cannot lead to a high-quality mashup, it is possible that the composition logic introduces additional quality issues (e.g., inconsistencies). Development of high quality mashups turns out to be non-trivial, and mashup composers should be assisted in their task (*Cappiello et al, 2010*).

There has been previous work about quality in the web application stating that Quality is an essential characteristic for web success. Several authors have described different methodologies, guidelines, techniques and tools in order to assure the quality of web sites. Recently, a wide ranging set of metrics has been proposed for quantifying web quality attributes. However, there is little consensus among them. These metrics are sometimes not well defined, nor empirically or theoretically validated. Moreover, these metrics focus on different aspects of web sites or different quality characteristics, confusing, rather than helping, the practitioners interested in using them. With the aim of making their use easier, it has been developed the WQM model (Web Quality Model), which distinguishes three dimensions related to web features, lifecycle processes and quality characteristics. This classification obtain the metrics that are classified into the “usability / exploitation / presentation” cell. Another conclusion obtained from our study is that, in general, metrics are automated but not validated formally nor empirically which is not a good way of doing things (*Cappiello et al, 2010*).

Also it has been state that the concept of quality is not simple and atomic, but a multidimensional and relative one. Common practice assesses quality by means of the quantification of lower abstraction concepts, such as attributes of entities. The attribute can be briefly defined as a measurable property of an entity category.

Therefore, quality – and its sub dimensions, called characteristics and sub-characteristics in the ISO 9126-1 standard – is an abstract relationship between attributes of an entity and a specific information need, with regard to its purpose, context, and user's viewpoint . On account of such multidimensionality, a quality model, which specifies the relationships between characteristics, sub characteristics and associated attributes, is usually necessary. Further, an instantiated quality model can in the end be calculated and evaluated in order to determine the level of satisfaction achieved.

The ISO 9126-1 standard (and the ongoing square project) distinguishes among three different approaches to ICWE 2008 Workshops, 7th Int. Workshop on Web-Oriented Software Technologies – IWWOST 2008 software product quality, viz. internal quality, external quality, and quality in use.

The quality mashup model being presented on this work is based on the assumption that quality in the mashups is strongly depends on the information that the integration of different components is able to provide. Quality aspects like maintainability, reliability, or scalability play a minor role, as the final mashup is needed only for a short timeframe. Information quality, instead, is crucial for both components and composition. Assessing the quality of a mashup therefore requires understanding both the quality of components and the effect that the composition has on the overall quality of the final mashup.

So a quality model is introduce for mashup components, that analyze how typical composition operations affect quality (with special attention to the mashup composers' perspective), and defines a quality model for mashups (as seen from the perspective of the users of the mashup) – everything with a special eye on information quality (*Cappiello et al, 2010*).

Conclusions

The rise of the so called Web 2.0 has open the door to the creativity and active participation of the user on the enhancement of available content on the Web. The old days of passive, observing users have benn left behind to make way to a new era of user-generated, user-shared and user-editing content, now the user can leave a print of his presence on the Web.

Although it is true that the main idea of the Web 2.0 is the offering of interconnected services we can still say that its most significant change from traditional Web is the introduction of user-created content, category where mashups are positioned and represent this shift towards a wider distribution of internet content. On this matter APIs play an important role, as they are the means trough which developers work with code and data in a flexible way, required for the successful creation of mashups.

To put ir in simple words, mashup developers are making use of small parts of already existing applications and web sites, to put them together in such a clever way that the result is a completely different tool; this is the real potential of the web services.

Nowadays it is possible to access the data and code of big web companies to manipulate it, developers do not have to write their own programs from scratch, now they just have to modify the existing ones in order to customize its own applications.

Even though quality in mashups is really important to get the adequate results that could meet the expectations from all the parties involve, there was no proper model that consider the right variables and validate them in the proper way.

In Chapter 3 we will introduce our Quality Model for mashups, describing the all the dimensions to be consider for a proper evaluations of the quality of a mashup and its components.

3

RESEARCH & ANALYSIS

In any tool we use in our daily life, the quality with which it performs the function it was designed for, depends in a big measure on the quality of the parts that conform to the tool itself; also when receiving a service, the quality of this and our satisfaction as clients depends highly on how the elements of the service are orchestrated; something similar happens with “Mashups”, in order to get a good Web service capable of satisfying our needs as clients or the needs of the final user, whoever this is, we need to know the elements that we count on, how they work and if we can rely on their capability to work together in order to accomplish the objective pursued. Quality aspects like maintainability, reliability, or scalability play a minor role, as the final mashup is needed only for a short timeframe. Information quality, instead, is crucial for both components and composition. Assessing the quality of a mashup therefore requires understanding both the quality of components and the effect that the composition has on the overall quality of the final mashup (*Cappiello et al, 2010*).

3.1 Quality Model Presentation

The result of integrating components into a mashup is typically a Web application. Several works have proposed quality models for Web applications (see for example). Few proposals are specific to modern Web 2.0 applications. Quality of content, i.e., information quality, is commonly recognized as a major factor. Yet, specific studies on mashup quality and on the role of information quality in mashups are still missing.

We identify three stages in the mashup process in which information quality comes into play. Each stage has its own actor.

The component developer creates components for mashups. We assume that developers correctly implement the component functionality, taking into account well-known principles, best practices and methodologies for guaranteeing the internal quality of the code. From an external perspective, building a component implies taking decisions, e.g., on the architectural style (e.g., SOAP services vs. RESTful services vs. widget APIs), the programming language (e.g., client side such as JavaScript vs. server side such as Ruby), the data representation (e.g., XML vs. JSON), the component operability and interoperability (e.g., the multiplicity of APIs targeting different technologies). Such external aspects affect the “appeal” of the component from the mashup composer perspective.

The mashup composer integrates components to create a new mashup. S/he discovers components, directly from the Web or from component repositories accessed from the mashup tool. The selection of components takes into account the fitness of each component for its purpose within the mashup, the complexity of its technological properties (e.g., a simple programming API, languages and data formats enhancing operability and interoperability), as well as the richness and completeness of the provided data. The mashup composer then implements the integration logic that is necessary to orchestrate the components. This requires a good

understanding of the components, to make the most of the components' value and to implement a high-quality mashup.

Finally, the mashup user is not interested in how the mashup is built. S/he simply wants the mashup application to perform as expected, without missing data, badly aligned data, or similar information quality problems. In other words, s/he is interested in the perceived external quality (*Cappiello et al, 2010*).

3.1.1 Component quality

Publishing mashup components through APIs or services hides their internal details and gives more importance to their external properties. It is proposed a quality model for mashup components that privileges properties of the component APIs – this is indeed the perspective that is most relevant to the mashup composer or the mashup user. The model is based on both our own experience with the development of components and “Mashups”, and on experimental evidence gathered by analyzing data from *programmableweb.com*. We organize the model along three main dimensions recalling the traditional organization of Web applications into data, application logic and presentation layer:

- Data quality focuses on the suitability of the data provided by the component in terms of accuracy, completeness, timeliness, and availability.
- API quality refers to software characteristics that can be evaluated directly on the component API. We split API quality into functionality, reliability, and API usability.
- Presentation quality addresses the user experience, with attributes such as presentation usability, accessibility, and reputation.

Now we will detail the Quality dimensions, Quality attributes and Attribute sub-characteristics that model presents that were proof to be exact and precise to measure the quality elements valuation for the “Mashups” components by the programmers.

Quality dimensions

➤ Data Quality

Quality attributes and Attribute sub-characteristics

Accuracy: It refers to data correctness and to the consistency between the data provided by a component and the real world context they represent. It can be measured as the proximity of the component data with correct data.

Completeness: The capability of a component to produce all the expected data values. It can be assessed by estimating the ratio between the amount of data produced by a component and the amount of expected data.

$$Completeness = 1 - \left(\frac{Number\ of\ Missing\ values}{Total\ number\ of\ values} \right)$$

Timeliness: The “freshness” of the component output: how up-to-date the produced data are for their users. Its assessment is centered on the concept of validity, expressed as the ratio between currency (the “age” of data from the time of the component creation or last update) and volatility (the average period of data validity in a specific context).

$$Timeliness = \max \left(0, 1 - \frac{currency}{volatility} \right)^s$$

where the exponent s controls the sensitivity of timeliness to the currency-volatility ratio.

Availability: It refers to possible access limitations, such as the ones defined by component licenses. Depending on the usage context, such limitations can be considered as restrictions decreasing the component quality or as necessary actions to prevent abuses that can decrease the component availability.

Quality dimensions

➤ API Quality

Quality attributes and Attribute sub-characteristics

Functionality: The aggregation of API interoperability (the set of covered protocols, languages and data formats), compliance (with respect to standard formats and technologies) and security (the provision of authentication mechanisms). Functionality can be refined by considering the interoperability, the compliance, and the security level of a component.

Interoperability is one of the most important attributes that affect the quality of a mashup component. In fact, the diffusion of a component depends on its capability to be used in different and heterogeneous environments. Interoperability is affected by the number of data formats accepted for information exchange. Thus, the interoperability of a mashup component can be defined as:

$$\text{Interoperability}_{comp} = |P_{comp}| + |L_{comp}| + |DF_{comp}|$$

where $P_{comp} \subset \mathcal{P}$, $L_{comp} \subset \mathcal{L}$, and are the subsets of protocols, languages, and data formats used by the specific component. \mathcal{P} , \mathcal{L} , and DF are the sets of possible protocols, languages, and data formats that can be used for the development of mashup components.

Some data formats are also standard (e.g., Atom, RSS, GData) and this increases the interoperability level and gives also the possibility to assess the compliance dimension as follows:

- Protocols Rest, SOAP
- Languages Javascript, PHP
- Data Formats Atom, RSS, Gdata, JSON, XML, Parameter-Value

$$Compliance_{comp} = std(DF_{comp}) : DF_{comp} \rightarrow [0; 1]$$

where $\text{std}(DF_{comp})$ produces 1 when at least one of the data formats supported by the component is a standard data format, and 0 if none of the supported data formats is standard.

The security of a component is related to the protection mechanism that is used to rule the access to the offered functionalities. We distinguish between two aspects: *SSL support* and *authentication mechanisms*. That is, the component might allow for encrypted communications, which improves security, or not. As for the authentication mechanism, we distinguish between no authentication, API key, developer key, and user account.

Formally, it is possible to define the security metric as

$$SEC_{comp} = SSL_{comp} + AUT_{comp}$$

where SSL_{comp} is a boolean value that indicates the use of SSL inside the component, while AUT_{comp} is a number between 1 and 4 that indicates the type of authentication method according to some complexity values.

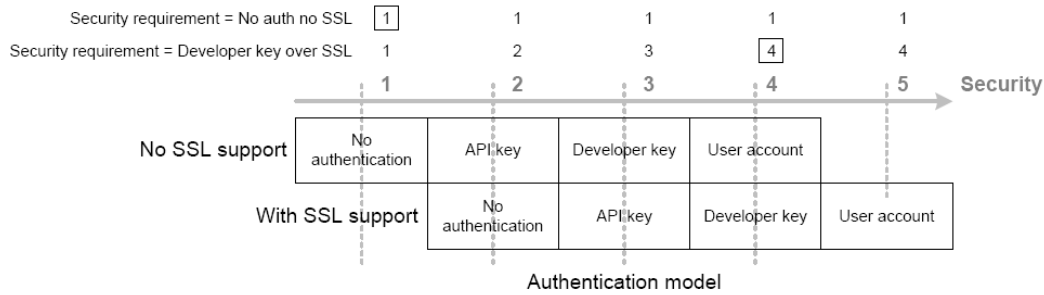


Figure 2

Reliability: In the API black-box approach, reliability corresponds to the component maturity, which can be assessed in terms of frequency of a component’s usage and updates. Reliability can be evaluated in terms of maturity, by considering the available statistics of usage of the component together with the frequency of its changes and updates :

$$Maturity_{comp} = \max\left(1 - \frac{CurrentDate_{comp} - LastUseDate_{comp}}{\frac{CurrentDate_{comp} - CreationDate_{comp}}{|V_{comp}|}}; 0\right)$$

where V_{comp} is the set of versions available for a specific mashup component.

API Usability: The ease of use of the component API, which can be measured in terms of learnability and understandability (e.g., the availability of documentation, examples, blogs, forums, etc), and operability (the complexity of the protocols, languages, data formats and security mechanisms).

Operability also affects the ease of use of a component. It depends on the complexity of the technologies used at the application and data layers, and of the adopted security mechanisms. The operability of technologies at the application level can be evaluated by considering the diffusion and the interaction overhead of both protocols and languages used in the API development. In *Figure 3(a)* we show a method to estimate the operability of the most common technologies generally adopted at the application level.

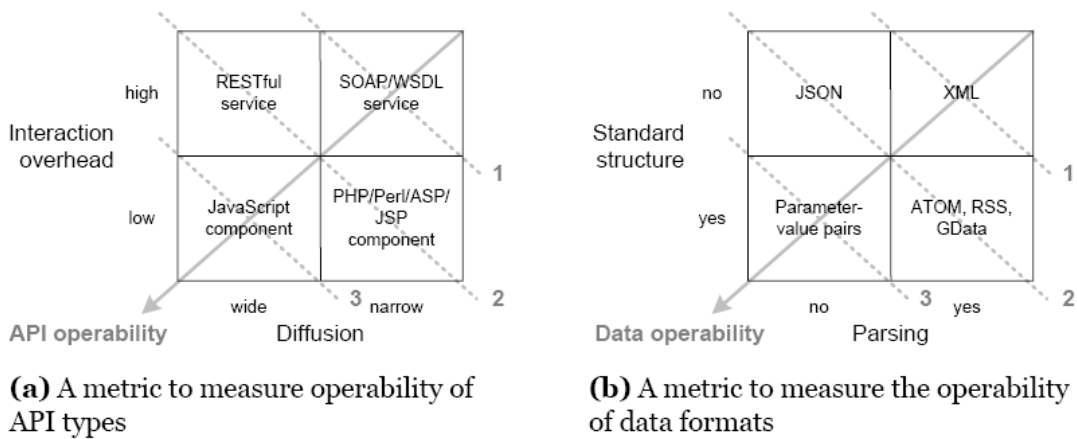


Figure 3 Operability of the technologies used at the application and data level.

Similarly, operability at the data layer can be evaluated by analyzing the data formats offered by the component along two aspects: the need for a parsing, meaning that further transformations are needed before the component can be integrated in the

final mashup, and the use of a standard format. *Figure 3(b)* describes a method to assess the operability of the most common data formats.

The security operability and the actual level of security are instead inversely proportional. The higher the level of security, the lower the security operability. *Figure 5* represents the different degrees of security operability that can be identified by considering the security mechanisms typically adoptable in a mashup component.

In general, once the above technologies have been classified using the described criteria, it is possible to define clusters and characterize them with an operability level. As shown in *Figure 4*, technologies in the same cluster are associated with the same operability value.

For example, in our analysis described in *Figure 3*, we use the following function family: $OP(T_{comp}): T_{comp} \rightarrow OPV$, where $T_{comp} = \{P_{comp}, L_{comp}, DF_{comp}, SEC_{comp}\}$ includes the technologies used by a mashup component at the application and data layers and the adopted security mechanisms, and $OPV \subset \mathbb{N}$ is the set of operability values defined for each technology. Since a component can be offered by using different APIs and thus more application and data technologies have to be evaluated, the overall operability measure can be defined as:

$$OP_{comp} = \max(OP(P_{comp} \cup L_{comp})) + \max(OP(DF_{comp})) + \max(OP(SEC_{comp}))$$

The first term considers the technologies characterizing the application layer of the component; the second refers to the data layer; and the last term refers to the security mechanism implemented by the component. For each addend, we only consider the maximum operability value, as we think this characterizes best the overall operability of the component.

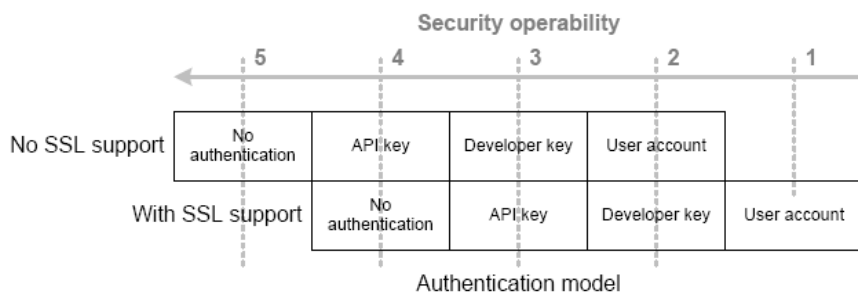


Figure 4 Operability of the security mechanisms

Quality dimensions

➤ Presentation Quality

Quality attributes and Attribute sub-characteristics

Presentation Usability: The usability of the presentation mechanisms adopted for the interaction with UI components. Given the situational nature of mashups, learnability and understandability of presentation, and compliance with presentation standards should be maximized to improve efficiency.

Accessibility: The capability of the component presentation to be “read” by any class of users and Web client. It is increased if a component offers a multiplicity of APIs supporting different presentation modalities for different devices, and also through textual annotations of multimedia contents enabling alternative browsing technologies (such as screen readers assisting impaired users).

Reputation: The perceived trustworthiness of the component. It is particularly affected by the brand of the component provider, the availability of documentation, especially if available in different formats and through different channels, and by the compliance of the component UI with common presentation standards.

3.1.2 Composition quality

Assessing the quality of each mashup component is not enough; the quality of the final mashup application indeed also depends on how these components are interconnected. There is no literature that has an approach that focuses on information quality and “Mashups”. Mashup quality is not simply an aggregation of the quality of the individual components. Instead it depends on the particular combination of components into a composite logic, layout and, hence, user experience.

Mashup components can be UI widgets, data sources and computational services. Some of them are visible in the mashup, some are hidden:

- Hidden components (e.g., data sources like RSS feeds) require another component for data rendering. For example, we can use an RSS reader to display the RSS feed items, so that the user can inspect them and navigate through them.
- The visible components may play different roles that affect the user's perception of the quality of the final integration, and must therefore be carefully taken into account.

We refer our self to Programmable.com - an specialized Web Page created with the aim of being a forum for developers, where information on APIs, Mashups and Web s as platform can be found, and also it is use as a showcase for developers since it has the largest compendium of Mashups and APIs on the Web, as of today amounts to over 5000 Mashups and over 2000 APIs, and still growing day by day – to carry on an analysis on Mashups in order to identify and validate the existence of the most typical elements on the structure of a “Mashup” according to the proposed on the Quality Mashups mode. A significant sample was token from each one the ten top categories of Mashups on the site, in proportion to their percentage with respect to the total of the Mashups on the top 10 categories. The following typical roles were identified:

– Master: Even if a Mashup integrates multiple components in one page, in most cases one component is more important than the others. Such a master component is the component the user interacts with the most. It usually is the starting point of the user interaction that causes the other components to react and synchronize accordingly.

– Slave: The behavior of a slave component depends on another component: its state is mainly modified by events originating in another (master) component. Many mashups also allow the user to interact with slave components. However, the content items displayed by slave components are selected via the user’s interaction with the master component, and by automatically propagating synchronization information from the master to the slaves.

– Filter: Filter components allow the users to specify conditions over the content shown by the other components. They provide (possibly hierarchical) access mechanisms that allow the users to incrementally select the contents they want to see. They also reduce the size of the dataset shown by the other components, thus improving the “Mashup” understandability and ease of use. In most cases, filter conditions are specified over the data set of master components, while slaves are synchronized and, hence, their content is automatically filtered by the integration logic.

In short, a filter allows the user to select groups or sets of data items, while the master component is used to select individual items that will be then complemented by detailed data provided by slaves. While master and slave components are usually sourced from the Web, it is the mashup composer who develops suitable filter components.

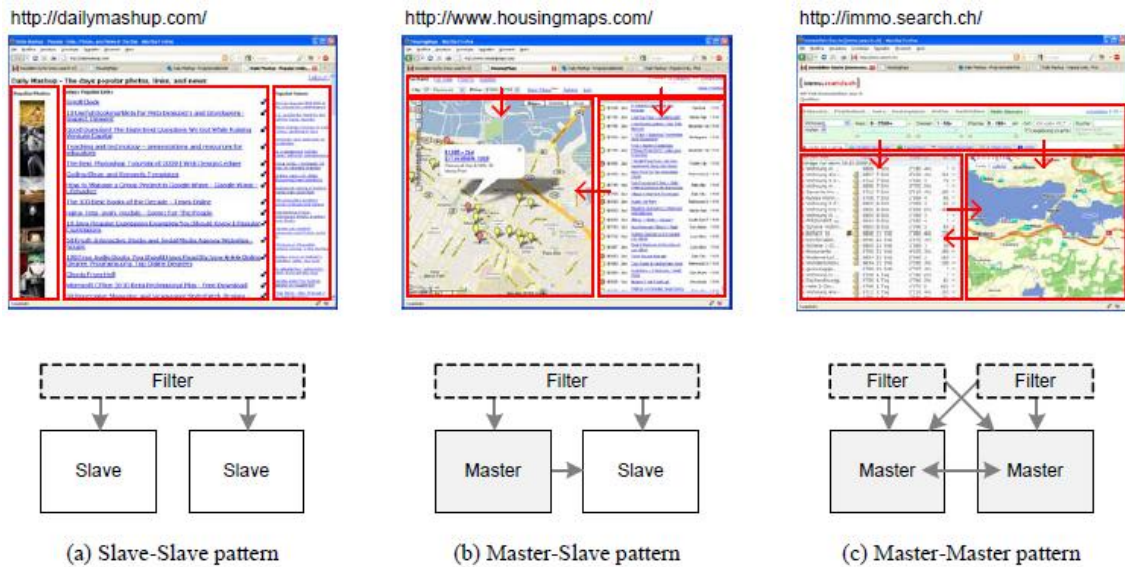


Figure 5 Basic mashup development patterns. Solid lines represent components; dashed lines (Cappiello et al, 2010).

Based on these three roles, the analysis of the mashups on programmableweb.com further allowed us to identify three basic patterns that characterize most of today’s mashup applications and to highlight some mutual dependencies among the identified roles that impact mashup quality:

– Slave-Slave (a): The mashup integrates several slave components with which the user interacts in an isolated fashion, without any propagation of data/events from one component to another. At startup or during runtime, users define filter conditions that steer all of the slave components. The effect is that of a rather static application with very simple interaction facilities, allowing users to “query” the data set of the slave components. An example is dailymashup.com, which integrates data from Flickr, Del.icio.us, furl, and Yahoo News.

Regarding the information quality of the resulting mashup, we assume that the filter does not degrade the perceived quality of the components, e.g., by issuing queries to them that cannot be satisfied and that would reveal data incompleteness problems.

This assumption is reasonable, as the filter conditions are specified by the mashup developer, who is aware of the coverage of the selected components.

– Master-Slave (b): This is the most widely used pattern of today’s mashup applications. It features all three component roles. A filter component allows users to restrict the data shown simultaneously by all the other components. The master component is used to perform the main interactions with the application, such as selecting interesting data items. The slave component is automatically synchronized according to the selections performed on the master component, thereby visualizing the details of the selected elements. The housingmaps.com application is a good example of master-slave mashup: a header bar acts as filter, allowing users to specify some conditions for apartment search (e.g., city and price), the Craigslist table acts as master, showing the list of the retrieved apartments with a link to a page showing major details, and the Google map then acts as slave, showing the locations of selected apartments. With the master-slave pattern, the information quality of the final application may depend on the composition logic of the application. Provided that master and slave are compatible in terms of data to be visualized, their integration might degrade the quality of the slave. If the master provides access only to a subset of the slave data, it may prevent the user from accessing the full data provided by the slave. If, instead, the master contains a superset of the slave data, it allows the user to ask for data items that cannot be served by the slave, thus revealing the incompleteness of the slave.

– Master-Master, Figure (c): This is the most complete pattern, where – in addition to suitable filter components – all integrated components are masters. All components provide interaction facilities that allow users to perform selections or to provide inputs that are propagated to all the other components that synchronize accordingly. The master components therefore also act as slaves. An example of master-master

pattern is the immo.search.ch application, where – in addition to locating a housing offer on the map – moving the map allows filtering the housing offers.

From an information quality perspective, the master-master pattern is similar to the master-slave pattern. If the components have different underlying data sets, there could be situations in which one component is able to satisfy the user request while another component is not, lowering the overall perceived quality of the mashup. The master-master pattern is however more “problematic” than the master-slave pattern, as it supports all directions of communication and therefore increases the likelihood of revealing incompleteness problems in any of the components.

The three Mashup patterns raise integration issues at data level, process level and presentation level. Integration at the process level requires setting up the necessary synchronization/orchestration logic among components using the operations and events they expose. Integration at the presentation level requires designing a composite layout, in which components are visually effective, and the different presentation styles are aligned. For the purpose of this article, we assume that integration at the process and the presentation level is performed correctly. To characterize information quality in the context of mashups, we instead focus our attention on the data level (*Cappiello et al, 2010*).

Mashup Information Quality

Integration at the data level concerns data mediation and data integration. The main challenge is the integration of data extracted from heterogeneous sources, whose exact characteristics are not known a-priori. Data integration in mashups corresponds to a Global As View (GAV) problem, in which the global schema is expressed in terms of views over the integrated data sources. During mashup development it is possible to inspect the attributes exposed by the components (the local schemas), as specified in the component APIs, and to infer join attributes on which to base data integration. The unpredictability of the underlying data instances,

however, raises new issues, which cannot be exhaustively managed through the traditional rules for the integration of structured and unstructured data.

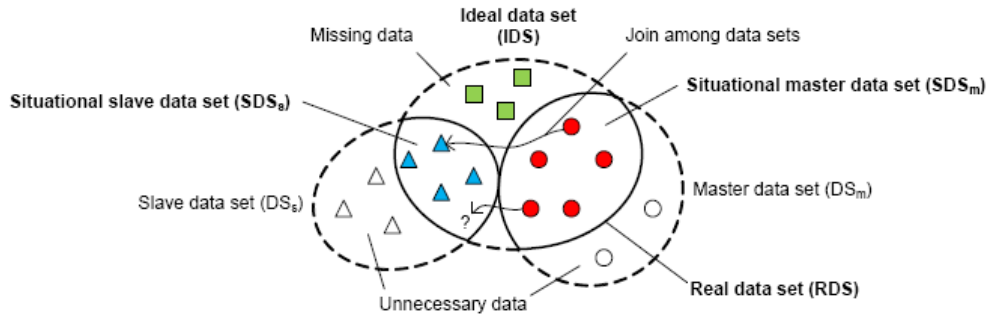


Figure 6. Data sets involved in mashup development (Cappiello et al, 2010).

Data integration for mashups can be characterized as follows (Figure 6 exemplifies the situation for the master-slave pattern):

- Mashup applications are developed in order to retrieve and give access to a set of data that we call the *Ideal Data Set (IDS)*.
- Each component k has its own data set DS_k . To fulfill the mashup requirements a smaller portion $SDS_k \subseteq DS_k$ could be sufficient. SDS_k is the *Situational Data Set* of the corresponding component.
- The integration of all the Situational Data Sets SDS_k gives the *Real Data Set* $RDS \subseteq IDS$ provided by the mashup. The information quality of RDS therefore depends on the quality of the data provided by the individual components.
- The information quality of the mashup can be determined by comparing its real data set (RDS) with the corresponding ideal one (IDS).

Evaluating information quality in mashups requires looking at both components and composition patterns. Analogously to the data quality attributes already defined for components in Section 3, we characterize information quality of mashups by means of *accuracy*, *completeness*, *timeliness*, and *availability*. Additionally, since integrating different data sets may lead to inconsistencies, we propose *consistency* as a new quality attribute.

Next, we discuss each of these dimensions for the master-slave and master-master patterns; we omit the slave-slave pattern, since its simple integration logic allows us to express mashup quality only as aggregation (minimum, average, maximum, or similar) of its component qualities. We further omit the filter component, since the filter can be considered an auxiliary element in the composition logic whose content stems from the master component it filters. It therefore suffices to take into account all master components to also include the respective filter components in the quality assessment. Finally, as we assume components are sourced from the Web, we also assume components are independent of each other. (Cappiello *et al*, 2010).

Accuracy

The accuracy of a component can be expressed as the probability that its data are correct: $p(\text{corr}_k) = 1 - p(e_k)$, where $p(e_k)$ is the probability that an error occurs. Data incorrectness arises each time a data value produced by the component is different from its real-world counterpart. This can happen for different reasons, such as typos, wrong representation, or missing updates. $p(e_k)$ considers all types of errors and can, for instance, be defined on the basis of the usage history of a component.

In the master-slave pattern, an error might occur in both the master and the slave component. Given the dependency between the master and the slave, the

probability of error in the slave is conditioned by the selection performed in the master. Therefore: $Acc_{ms} = 1 - (p(e_m) + p(e_s|corr_m))$.

Master-master compositions can be considered as the combination of two master-slave patterns: a selection in one master causes the other master to acts as a slave and vice versa. Therefore: $Acc_{mm} = 1 - [\alpha(p(e_{m1}) + p(e_{m2}|corr_{m1})) + (1 - \alpha)(p(e_{m2}) + p(e_{m1}|corr_{m2}))]$, where α is the probability for a component to act as master in the user selection. (Cappiello et al, 2010).

Completeness

The *situational completeness* SC evaluates how the components' data sets are able to provide the desired information and can be defined as the degree with which the RDS covers the IDS: $SC = \frac{|RDS|}{|IDS|}$.

In the master-slave pattern, the cardinality of the RDS is the sum of the cardinalities of the situational master data set and the cardinality of the joined situational data sets of master and slave. Therefore:

$$SC_{ms} = \frac{|SDS_m| + |SDS_s \text{ semi join } SDS_m|}{|IDS|}$$

Since the master-master pattern can be modeled as the combination of two master-slave patterns, the cardinality of the RDS results from the sum of the cardinalities of the two situational master data sets (we assume that there is no overlapping of the master data sets, which is reasonable in that two components typically serve two different needs). Therefore: $SC_{mm} = \frac{|SDS_{m1}| + |SDS_{m2}|}{|IDS|}$

The situational completeness does not cover the case where in both the master-slave and the master-master patterns data in the slave component are not accessible due to missing linkages to any of the master data items. For this reason we

define the *compositional completeness* CC as the degree with which the mashup integration effectively covers the situational data sets.

In the master-slave pattern, the compositional completeness is the ratio of the cardinality of the join among the situational data sets of master and slave to the cardinality of the situational data set of the master:

$$CC_{ms} = \frac{|SDS_m \text{ join } SDS_s|}{|SDS_m|}.$$

In the master-master pattern, we again use a linear combination of the two corresponding master-slave patterns, with α being the probability that the first component acts as master:

$$CC_{mm} = \alpha \frac{|SDS_{m_1} \text{ join } SDS_{m_2}|}{|SDS_{m_1}|} + (1 - \alpha) \frac{|SDS_{m_1} \text{ join } SDS_{m_2}|}{|SDS_{m_2}|}.$$

(Cappiello et al, 2010).

Timeliness

Timeliness provides information about the freshness of the available data sets. The timeliness of the mashup can be computed as aggregation of the timeliness values of the individual situational data sets: $Time = f_{agg}(time_1, \dots, time_k)$ where f_{agg} can be *minimum*, *average*, or *maximum*.

The evaluation of the timeliness is independent of the mashup patterns; the chosen aggregation function may depend on the role of time in the application domain. For instance, considering a mashup that shows news from different newspapers, the *maximum* may be appropriate, as it reflects the latest up-date. For a mashup that provides stock values for online trading, the *minimum* may be suitable to describe the freshness of the overall data published. If instead time is not a major concern, for instance if the mashup shows pictures on a map, the *average* could be a good choice (Cappiello et al, 2010).

Availability

The availability is the likelihood that the mashup is able to provide any data, that is, in order for a mashup to be available it suffices that one of its components is

available. Therefore, the availability of a mashup can be expressed as $Avail = 1 - \prod_k(1 - Avail_k)$, where $Avail_k$ is the availability of the situational data set of the component k .

Also availability is independent of the mashup patterns. However, especially in the master-slave pattern the unavailability of the master may impact on the overall functionality of the mashup (e.g., the user might not be able to access data in the slave), while the other two patterns do not present this dependency. (Cappiello et al, 2010).

Consistency

In the component model, we assume that each component provides consistent data, i.e., components are not contradicting themselves. If mashed up, however, situational data sets may conflict with each other, leading to inconsistency in the data shown in the mashup. For instance, when plotting university locations on a map, it could happen that the address of a university cannot be parsed correctly and that it is placed wrongly (e.g., the MIT might be mapped to Cambridge in the UK). Traditionally, consistency is assessed and enforced with business rules expressing domain knowledge. In mashups, the composer does not have sufficient knowledge about the data provided by the components, and is therefore unable to write such rules in advance; thus inconsistencies only emerge during the execution of the mashup.

When mashups are developed as a comparison tool of multiple data sources from different providers (e.g., news feeds, like in *slashdigg.com*, or *doggdot.us*), inconsistency may not be problematic, as it is up to the users to compare through the mashup the results of querying different data sources and infer which one should be trusted as the one providing the most correct and timely data. (Cappiello et al, 2010).

Conclusion

Through the quality model it has been well established that the quality of a mashup is intrinsically related with the components that conform it and the way in which these components interact among each others. However it is worth to say that the existence of quality components does not necessary lead to the obtainance of a quality mashup, the way how the components are integrated can lead to different results.

In Chapter number 4 we will focus on the validation of the presented quality model through a series of research and analysis that cover all dimensions required to evaluate the quality of the mashup and its components.

4

QUALITY MODEL VALIDATION

4.1 Survey

In order to validate the component quality stated on our Quality Model we have developed a survey that allows us to gather the required data to run an analysis. This survey has been applied to developers with different level of programming skills and working in different fields, either private or business sector.

The survey is made of ten main questions regarding the different elements making up the component quality as described on the model.

4.1.1 Validation of the Quality model

Model Validation

Previously we have introduced the Quality Mashup Model, now we will proceed with the validation of it. Through this analysis we will gather the opinion of the mashups developers with respect to the component quality, and then match it with the proposal of the model, this way we can find out how different or similar our proposal is from current developers uses and practices.

4.1.1.1 Component Quality

In order to validate the component quality proposed by the model a deep survey was developed, and apply to several and diverse programmers in order to validated if the dimensions and attributes stipulated on the quality model were in fact accurate to measure the quality of the components of a mashup, and to prove the criteria define at chapter 3 including functionality with all the data formats, languages and protocols and usability.

First the survey was develop with multiple choice questions regarding the points of the dimensions and attributes like “Which are the factors that most influence your choice of an API besides the actual functional Documentation, Technology, Data Quality and Perceived Quality” and surveyed were supposed to rate the following items with a value between 1 and 5; 1=poor and 5=high; on the Annex you can find the completed survey.

With this types of questions and answer we were capable of evaluate the degree of precision the attributes and dimension had with the programmers at the time they selected the components to built mashup, or when they were using one.

So the Survey was deploy in different connection channels so it could reach a more varied population of programmers that would gave us a more realistic and heterogenic sample. The channels we used were: Programmers communities on facebook, Programmers communities on LinkedIn, Programmers communities on prgrammableweb, Industries related professionals contacts (such as IT managers at Nissan, Autoliv Inc, etc..) and Programmers contacts of IT department Politecnico di Milano among others.

There were more than 50 survey answers that completed our analysis and were collected on an electronic way and all the results displayed in a file that was then studied and breakdown on a detail work that showed the data on different graphic ways that proofed the type of programmers, their context, general information and

according to them what factors, formats and component / service types they preferred and pondered so we could measure which dimensions and attributes proposed were ideal to select the API or components of the mashup.

4.1.2 General Information Analysis

The general information of the sample involved in the survey is graphed and detail so it can picture the wide range of programmers taken into account for this model validation.

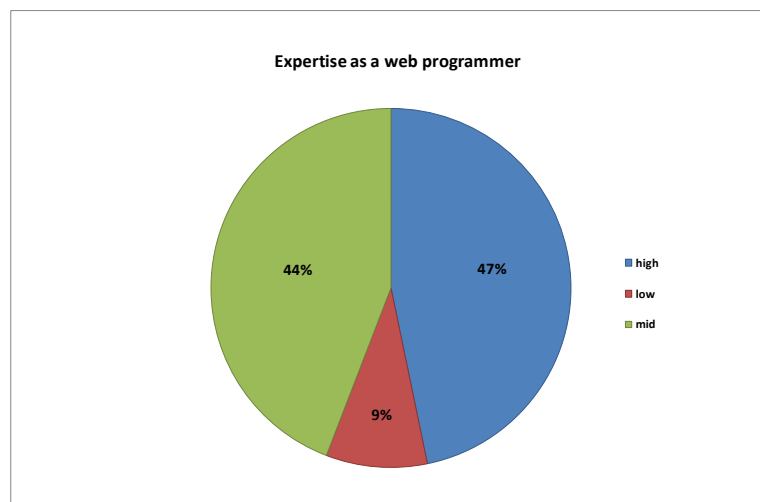


Figure.7

In *Figure 7* we can see that 47 % of the people that answer the survey are in a high level of programmer expertise, leaving behind with 44 % the programmers with a medium level and just a 9% has a low level of expertise in programming. This information give us a certain level of security about the trueness of the answers in the survey.

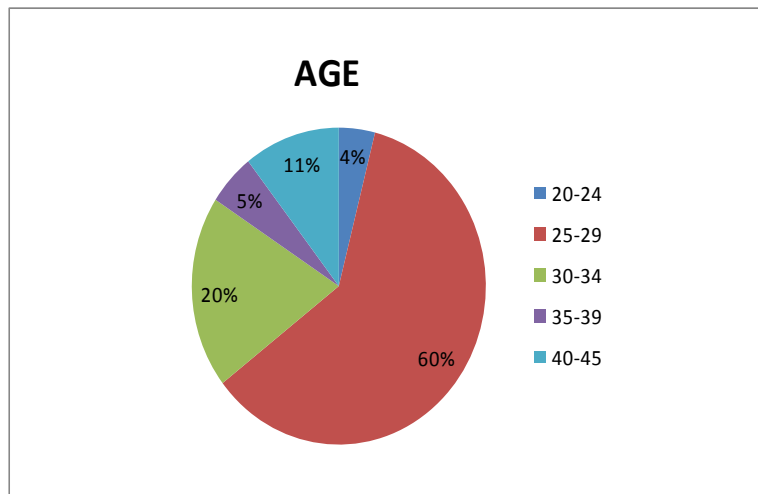


Figure. 8

Figure 8 shows us the detail of the ranges of age of the people whom answered the survey, also in which proportion they are located within the different age ranges. As show on the graphic Figure 8 we can see that the age of these programmers goes mainly from 24 to 34, age that indicates that most of them are already graduated and very possible already within a job position.

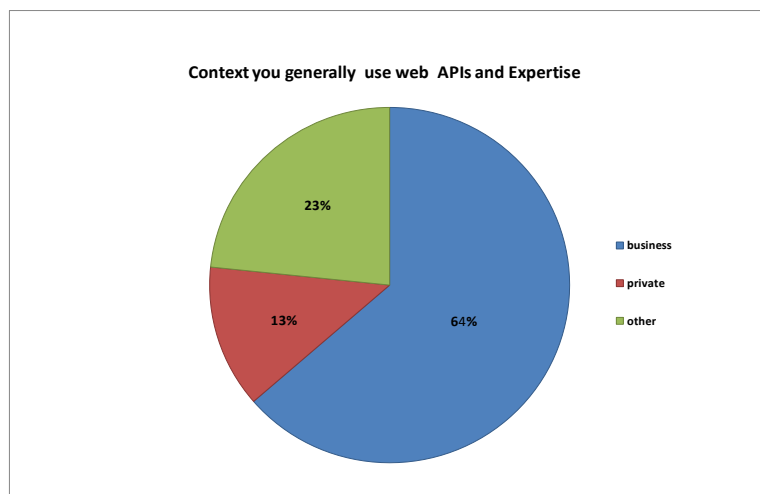


Figure.9

The respondents may apply their programming skills in different context therefore also the use they give to APIs falls in different sectors; *Figure 9* represent the concentration of the use of API and expertise in three different categories (Business, Private and Others). The main use of APIs is on a Business area, leaving in last place the use of them within a Private sector.

In the last graph of general information *Figure.10* The main context of use for APIs is the “Business” sector where programmers are high and medium expertise, second is “Other” uses apart from business and private where the expertise level is also first high and decreases to low gradually and at the end the “Private” sector where the expertise level is medium.

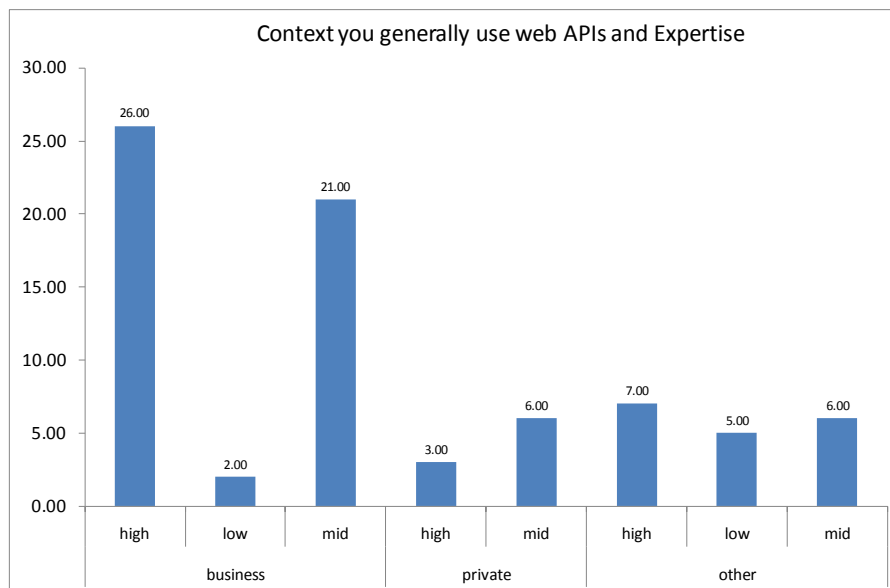


Figure.10

4.1.3 Component Quality Results Analysis

The Model is organized in three main dimensions Data, Application Logic and Presentation Layer. And for each one of this there are the following quality attributes to be validated:

- Data Quality (Data provided) .- Accuracy, Completeness, Timeliness, Availability
- API Quality (Software).- Functionality, reliability, API usability
- Presentation Quality.- Presentation usability, Accessibility and Reputation

The following *Figure 11* is one of the most representative figures because it globalized the vision and importance of the quality in the mashup components and shows the order of priority among the three main dimensions. On the lower axe are the four factors which are related to the three main dimensions : Documentation and Technology are related to API quality, Data Quality to Data Quality and Perceived Quality to Presentation Quality.

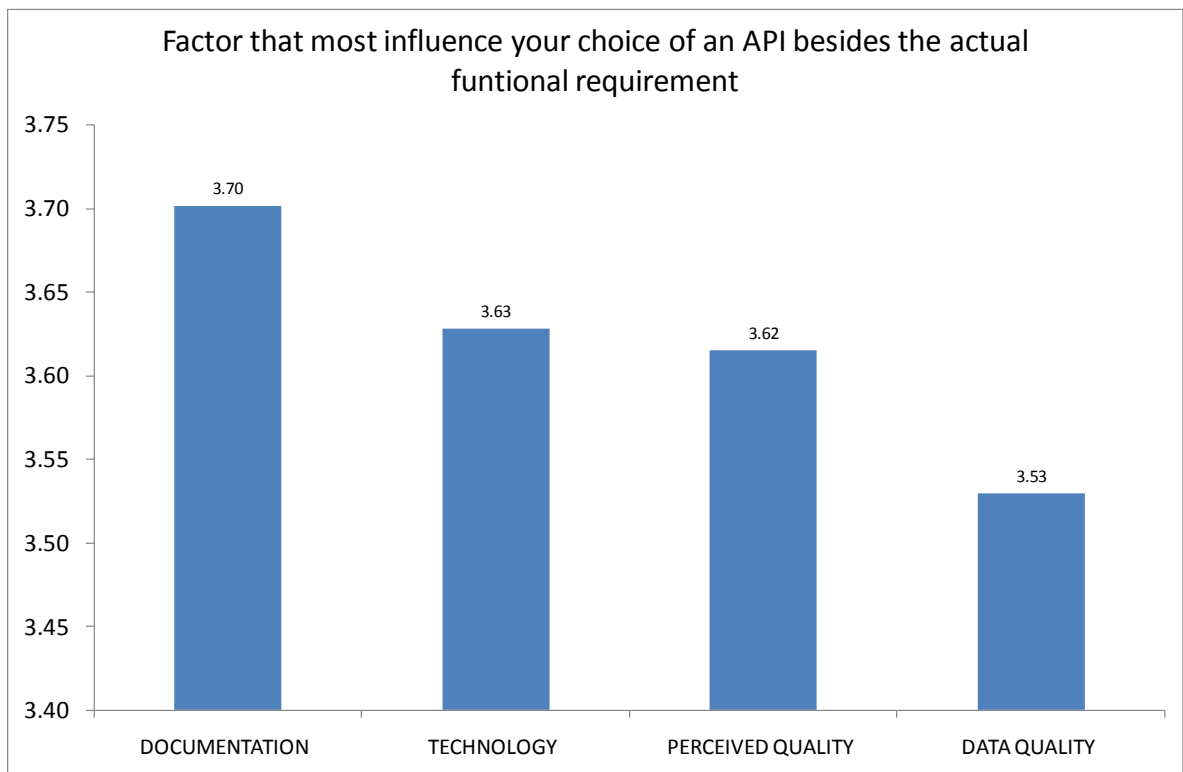


Figure. 11

As Documentation ranks in first place this confirmed what our model states that documentation is the most relevant factor that influence programmers beside the actual functional requirement. The availability of documentation has the strongest correlation with the diffusion of the component this means that developers usually prefer easy to combined components over complex components. Documentation is part of the attribute API Usability that belongs to the main dimension API Quality.

The interoperability of a component can be assessed by inspecting its API, since it particularly depends on the technologies used at the application and data layers. According to our analysis based on the survey results, Technology is on the second place of importance for developers at the moment of choosing an API; Technology belongs to the Functionality attribute which as well as Documentation, occupying first place, is part of the dimension API Quality.

In third place is Perceived Quality factor that is part of the main dimension Presentation Quality. Data Quality factor that belongs to the main dimension Data

Quality is placed fourth regarding the factors that influence the choosing of an API besides the actual functional requirement.

As shown on *Figure 11* the results of our survey proves that Documentation and Technology are the first two most relevant factors for developers, and they both belong to the same Quality dimension (API Quality). Therefore API Quality can be consider the most important one, followed by Perceived Quality and at the end Data Quality.

Our Quality Model offers a series of assessment metrics and fine-grained attributes mainly on the dimension API Quality, more specifically in Functionality and Usability, attributes that according to the model and supported by the results of the survey are the most important and relevant ones to the developers.

Figure 12 shows that depending on the level of expertise the answers were less or more variable. On the low expertise level Perceived Quality was the top and Technology was the last, on the medium level Documentation made the top followed very close by Perceived Quality and Data Quality was left at the end, but with the High expertise level programmers, we see they almost evaluate equally every factor being Data Quality the first one followed very close by Technology and then by Documentation and Perceived Quality. As a trend in this graph we can conclude that the higher the level of expertise the programmer has the factors weight the same whereas when the level of expertise decreased the variation on the weight of each factor increased making the overall graph *Figure 11* look not equal and leaving Data quality relegated to the last place and placing Documentation as the most relevant factor besides the actual function for an API.

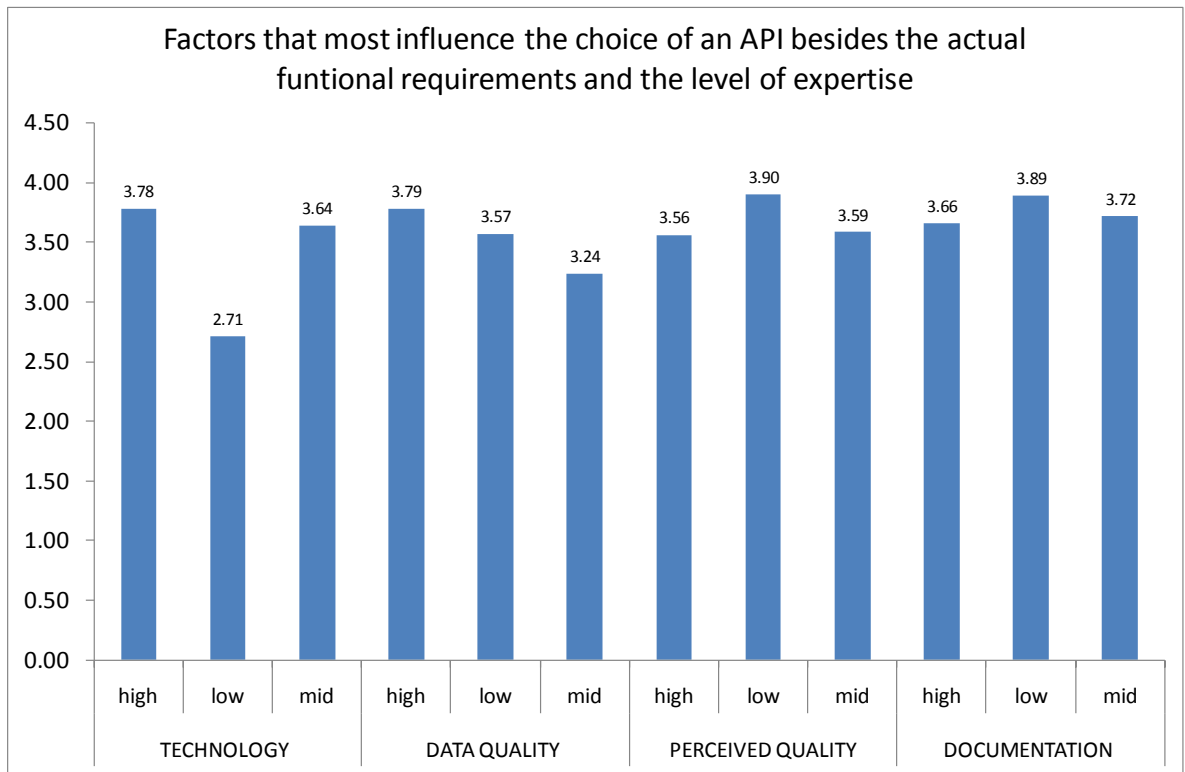


Figure 12

With this we can conclude that these factors are the ones that the programmers appreciate, search and look for to select their components API because they reflect the privileges properties of a component API Quality that the Quality Model presents. Also the attributes: Accuracy, Completeness, Timeliness, Availability, Functionality, Reliability, API Usability, Presentation Usability, Accessibility, and Reputation, collects all the different angles and characteristics needed and measurable for an API component. To prove this we have the scores given by the programmers all above 3 points out of 5 which shows the relevance of the selected factors and dimensions so none of them can be considered not important.

To take this analysis more into detail the following figures will explain how this factor are divided and according to their characteristic which ones are more pondered depending on the area of development and also of the degree of expertise the programmer has.

Before analyzing each factor (Documentation, Technology, Perceived Data and Data quality) we will see by context (Business, Private and Others) the levels they got for then go into their subdivisions and analyzed each factor independently by context.

In the following three *Figures 13, 14 and 15* we can get a similarity of order of importance in the factors between the Business context and the Private context where Documentation is the first and Data Quality is the fourth. For Technology in the Business sector is the second being more important or relevant than in the Private sector where it was third, Perceived Quality is the opposite of Technology being third in the Business context and second in the Private one. But for the Other context is completely different valuation the first place is for Data Quality, the second is for Documentation, the third is for Technology and the forth is for Perceived Quality.

This shows us that this context evaluates the factors in a different way also this effect happens due to the level of expertise that programmers have when they use API to create mashups in the Others context, this is low or medium rarely it is an expert level programmer so this also affect the way of evaluation of factors as it will be explain in detail in the specific explanation of the others context analysis.

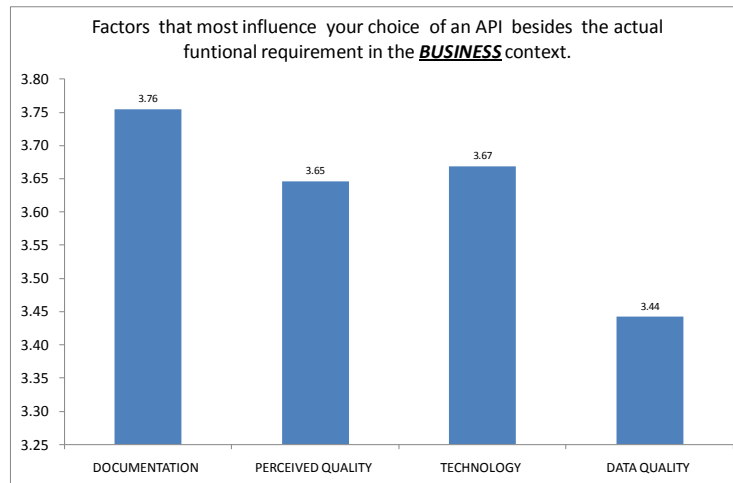


Figure 13

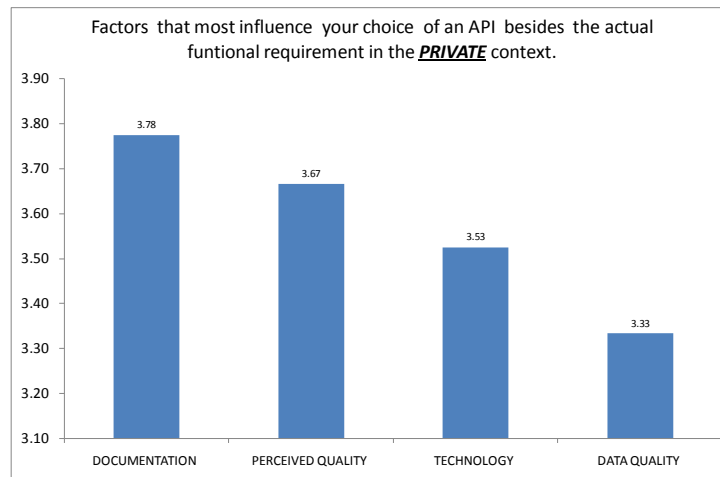


Figure 14

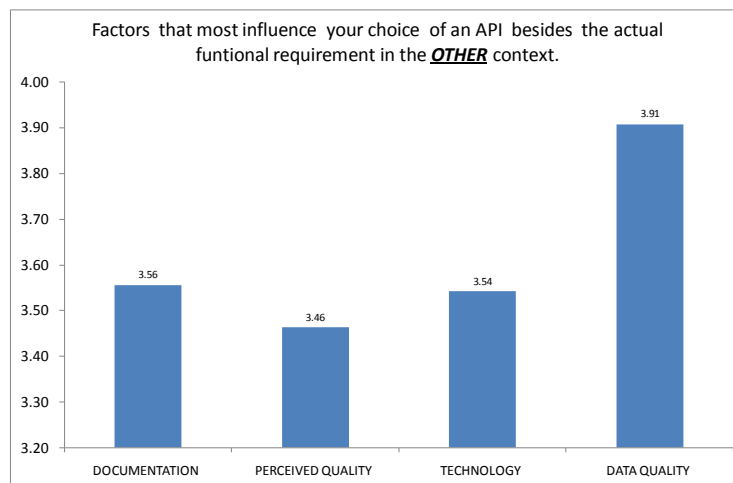


Figure 15

The Factor of Documentation that has being rated as the number one in importance subdivide in the following factors that were evaluating as it follows, *Figure 16* being all this options part of attribute API usability of the API Quality main dimension.

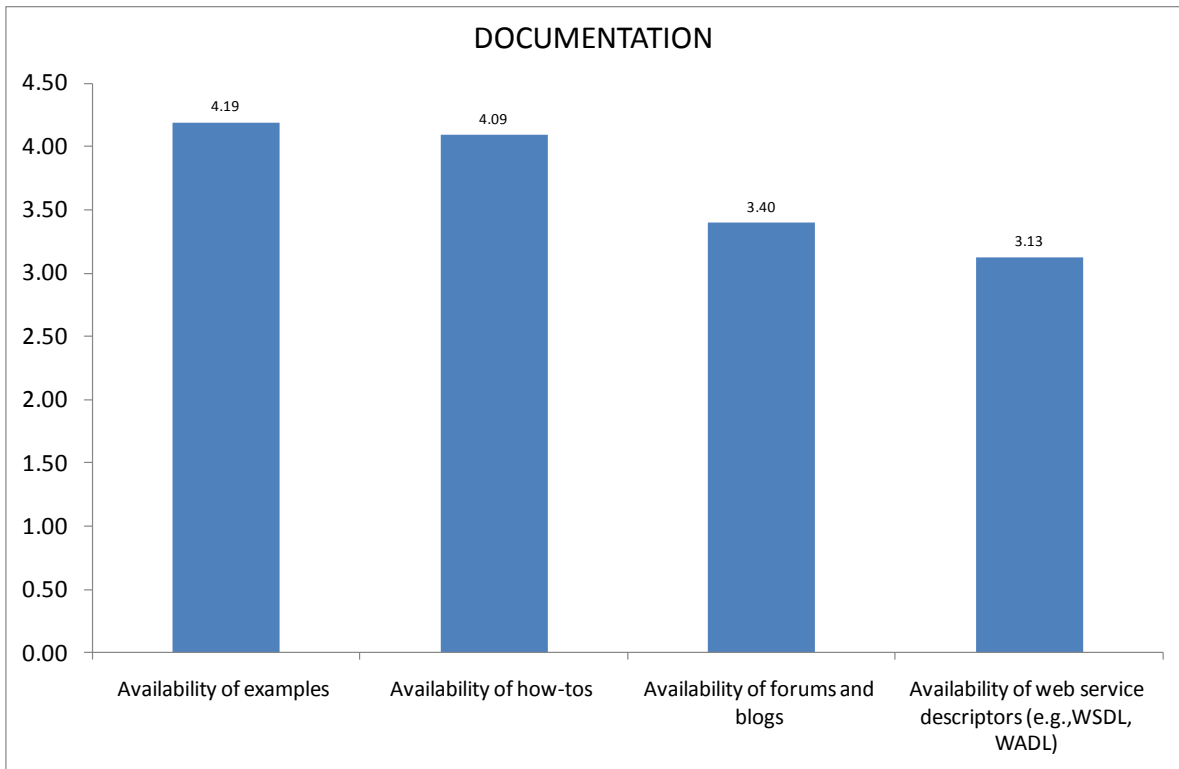


Figure 16

Regarding the factor of Documentation *Figure 16* we can see the weighted percentage that was given by the programmers according to the options given being first Availability of examples then the availability of how to on third the availability of forums and blogs and at the end availability of web service descriptors.

On the following figures *Figures 17, 18 and 19* the three graphs will show the percentage given according to the context: Business, Private or Others and we will see that Business and Others have the same order and that Private has another order.

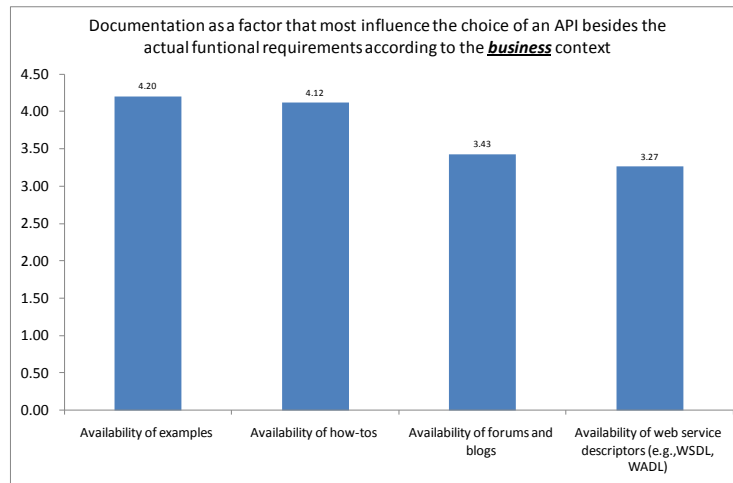


Figure 17

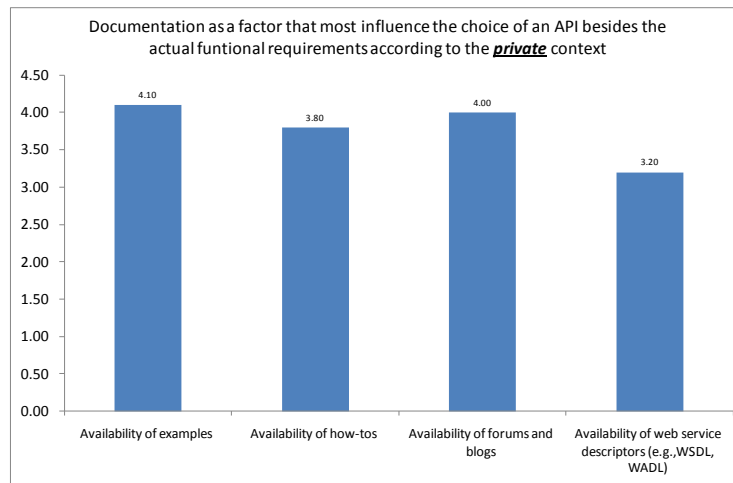


Figure 18

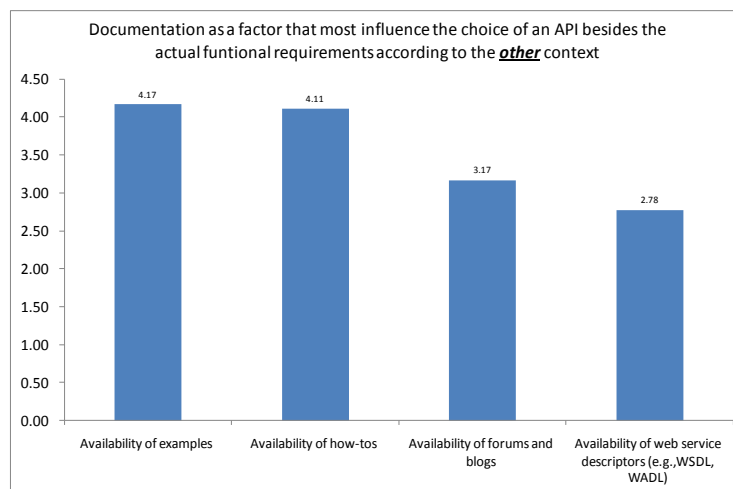


Figure 19

Regarding the factor of Technology on the *Figure 20* we can see the weighted percentage that was given by the programmers according to the options given; being first Support for your preferred programming language then Use of standard data on third Support for your preferred programming protocols and at the end Adopted authentication model. Being all this part of API usability attribute that belong to the API Quality main dimension.

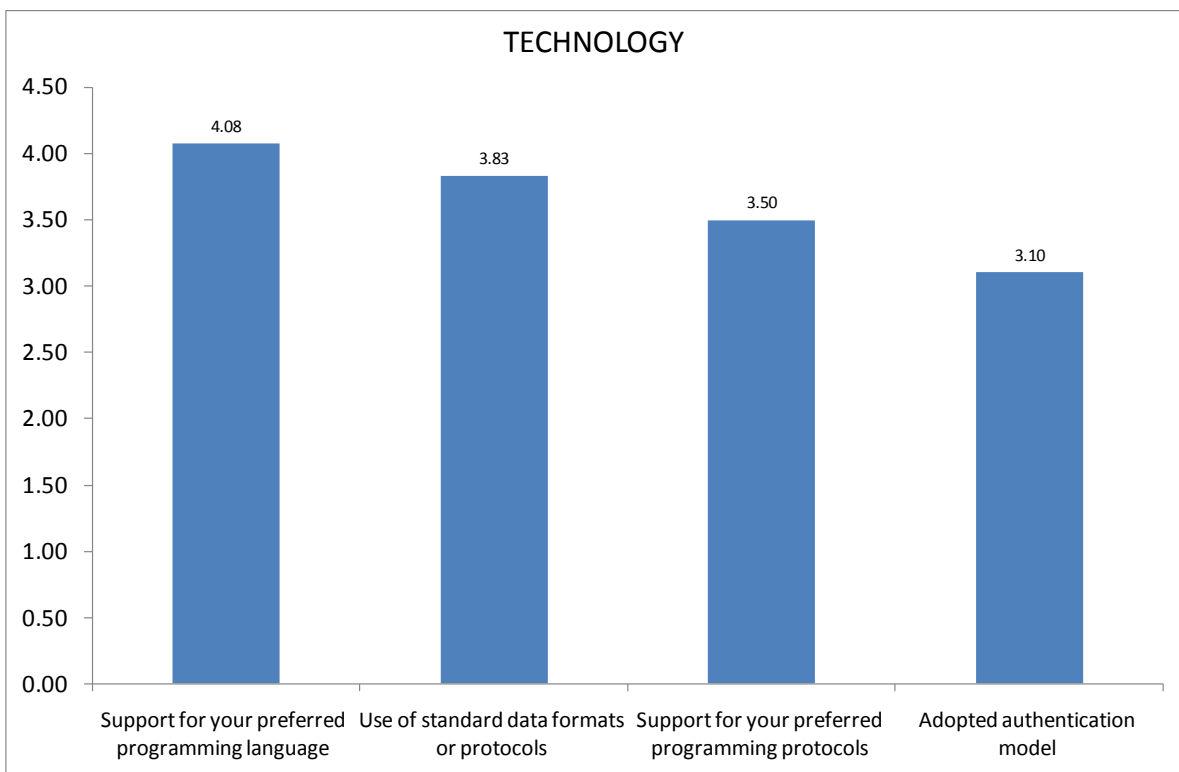


Figure 20

The *Figure 21, 22 & 23* will show the percentage given according to the context: Business, Private or Others and we will see that Business and Others contexts are similar leaving Private different from the rest in this particular factor.

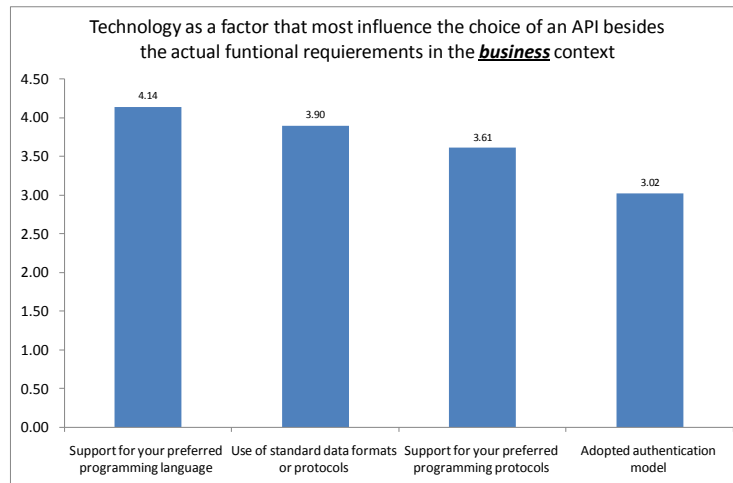


Figure 21

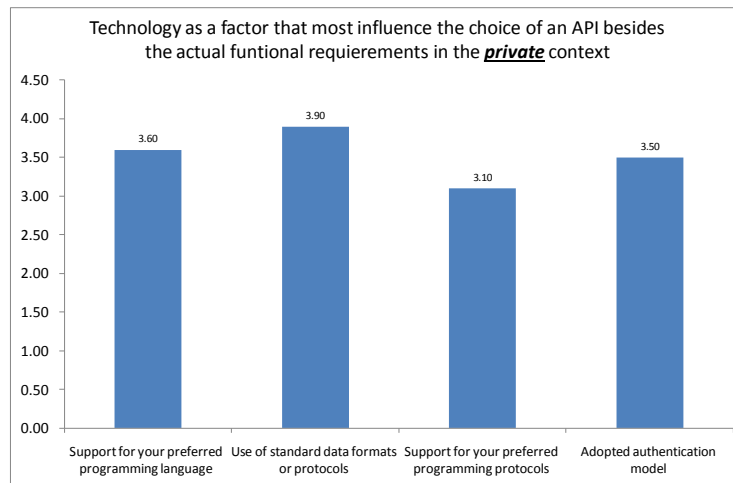


Figure 22

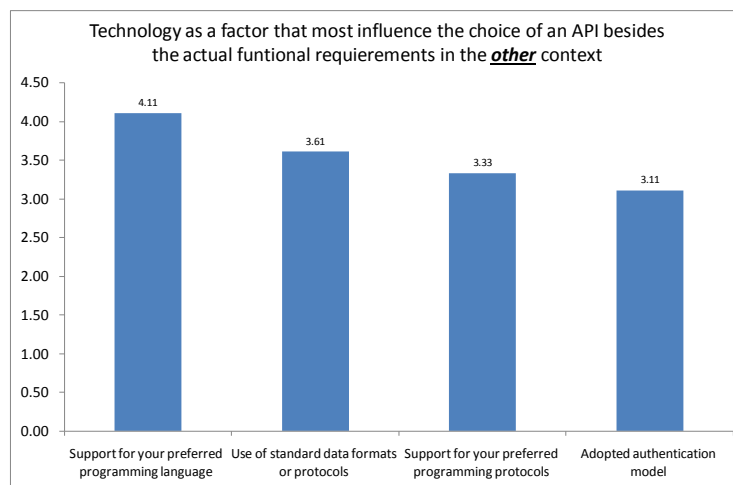


Figure 23

Regarding the factor of Data Quality on *Figure 24* we can see the weighted percentage that was given by the programmers according to the options given; being first Accuracy of the provided data set then Coverage of the provided data set and third Freshness of the provided data set all this are part of the attributes Accuracy, Completeness, Timeliness (freshness) that belong to the Data Quality main dimension.

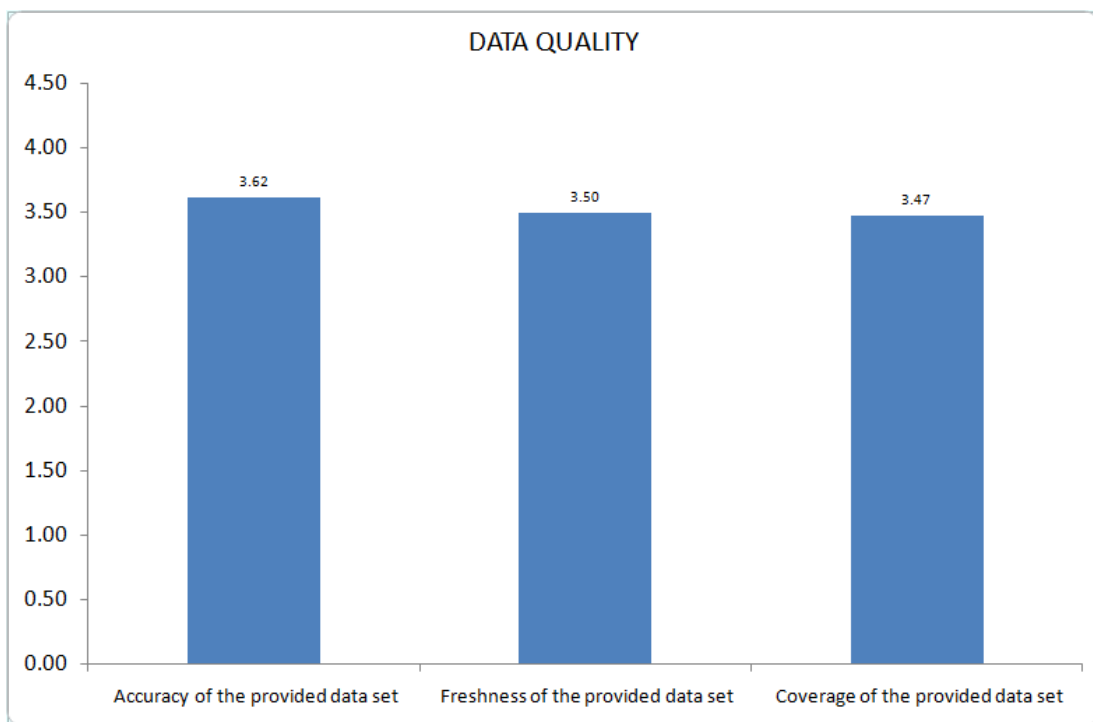


Figure 24

On the following *Figures 25,26 & 27* is shown the percentage given according to the context: Business, Private or Others and we will see that each context evaluated different from each other what tells us that each context is different regarding Data Quality.

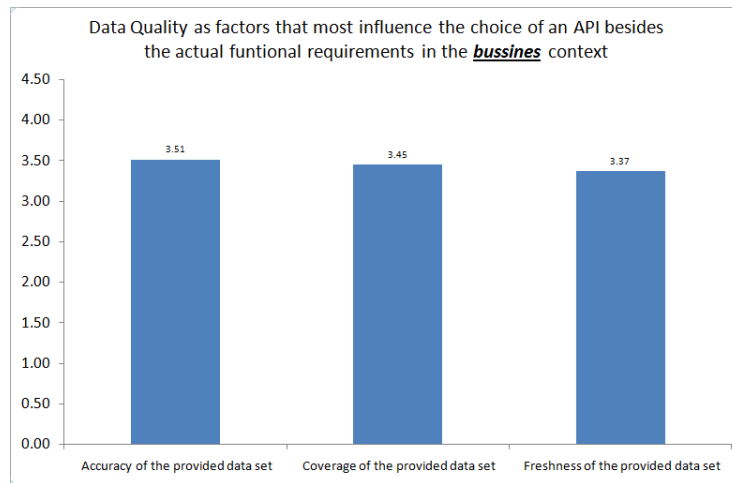


Figure 25

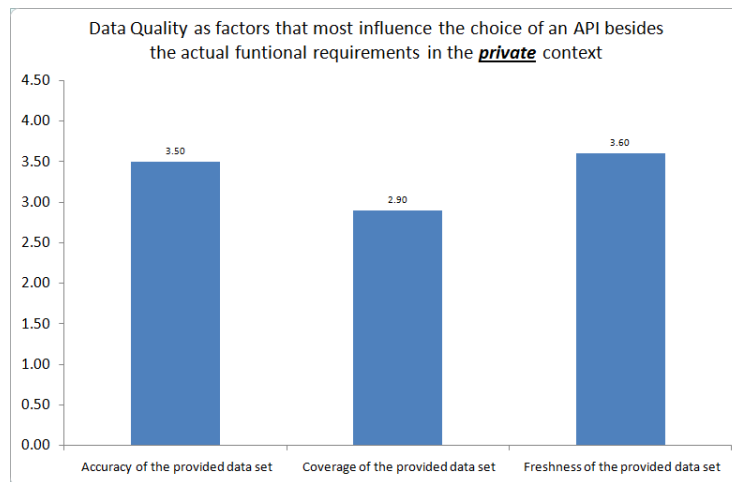


Figure 26

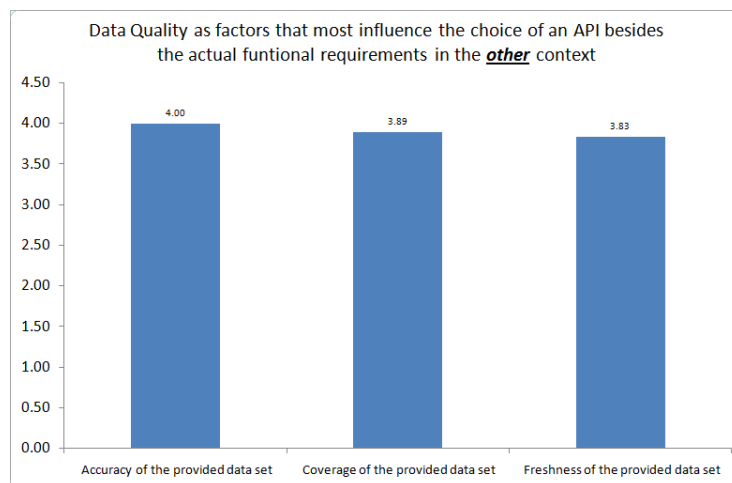


Figure 27

Regarding the factor of Perceived Quality on *Figure 28* we can see the weighted percentage that was given by the programmers according to the options given; being first Reputation of the API provider then Usability and accessibility of the user interface and third Diffusion of the API all this belonging to the attributes Presentation usability, Accessibility, Reputation of the main dimension Perceived Quality.

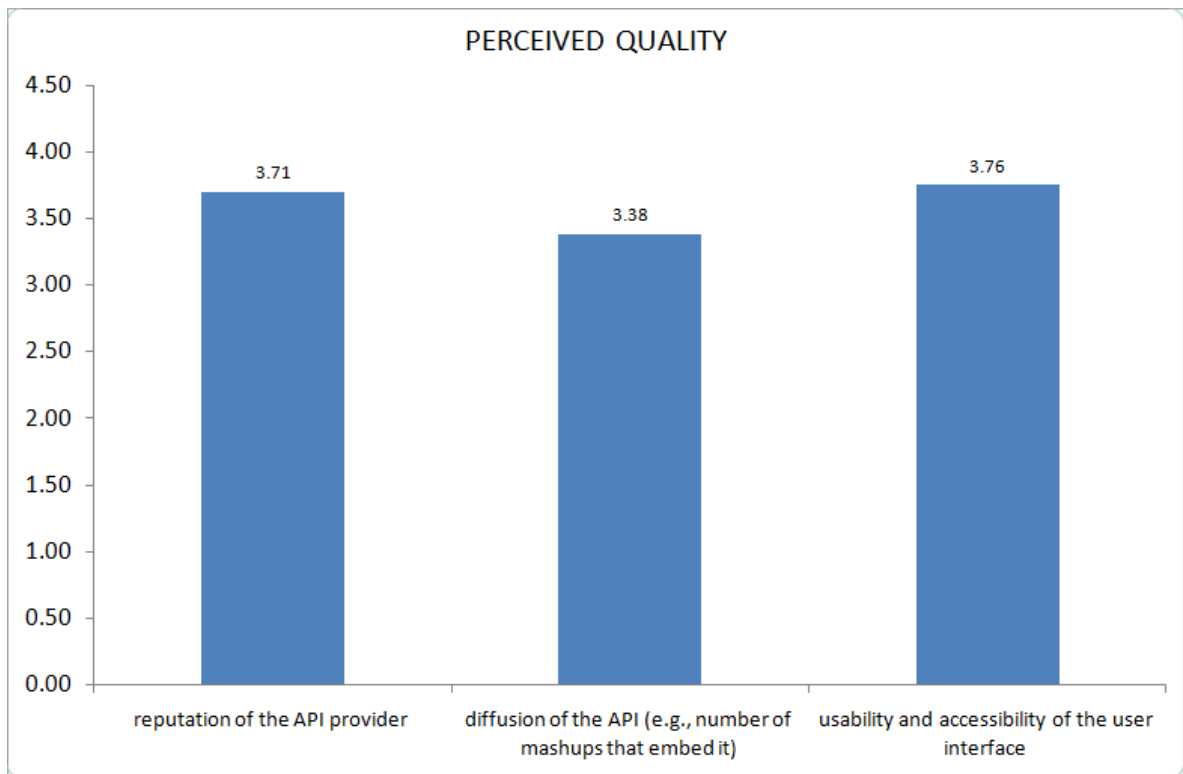


Figure 28

On *Figure 29, 30 & 31* it is shown the percentage given according to the context: Business, Private and Others and we see that Private and Others have the same order but different values and that in the Business context is totally different regarding Perceived Quality, but they have the first place in common Reputation of API provider is always the first.

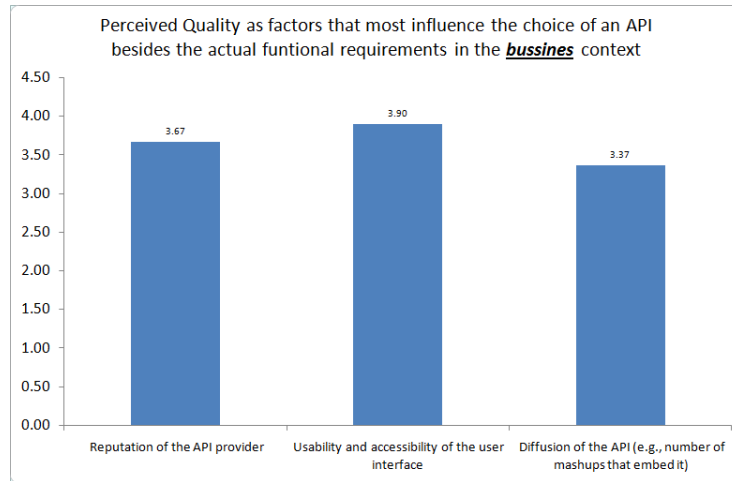


Figure 29

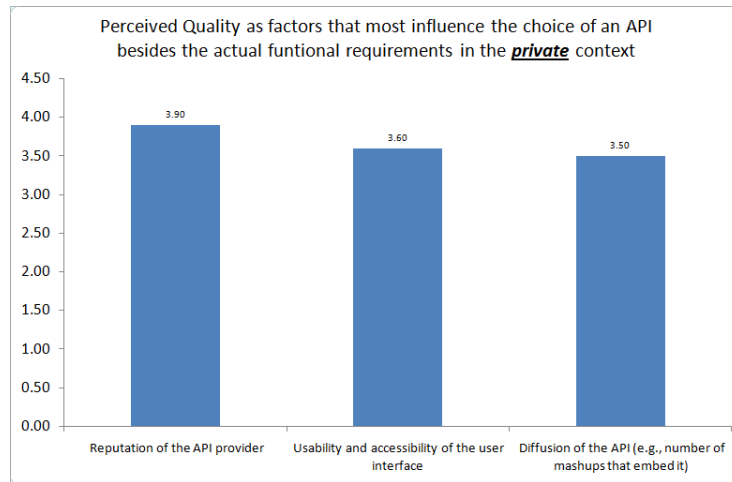


Figure 30

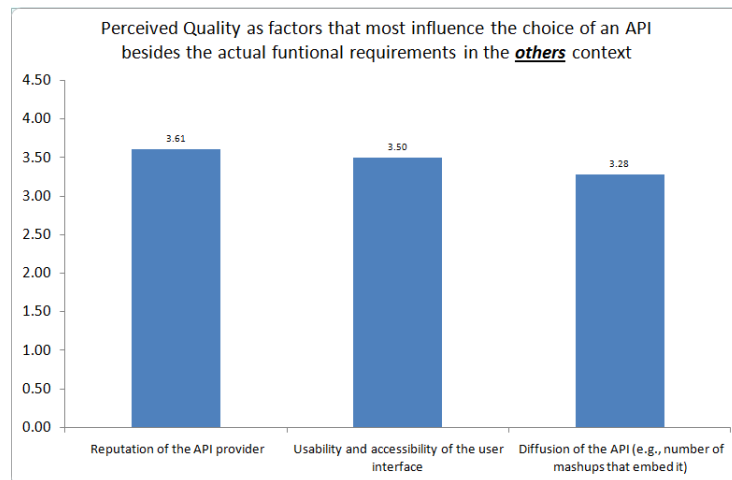


Figure 31

To finish this analysis the following figures show the result given by programmers towards factors related to attribute Timeliness of the main dimension Data Quality and of the attribute Functionality of the main dimension API Quality.

The figure below *Figure 32* show that for programmers the API age and the Frequency of update are factor that have more impact on the reliability than the API number of versions

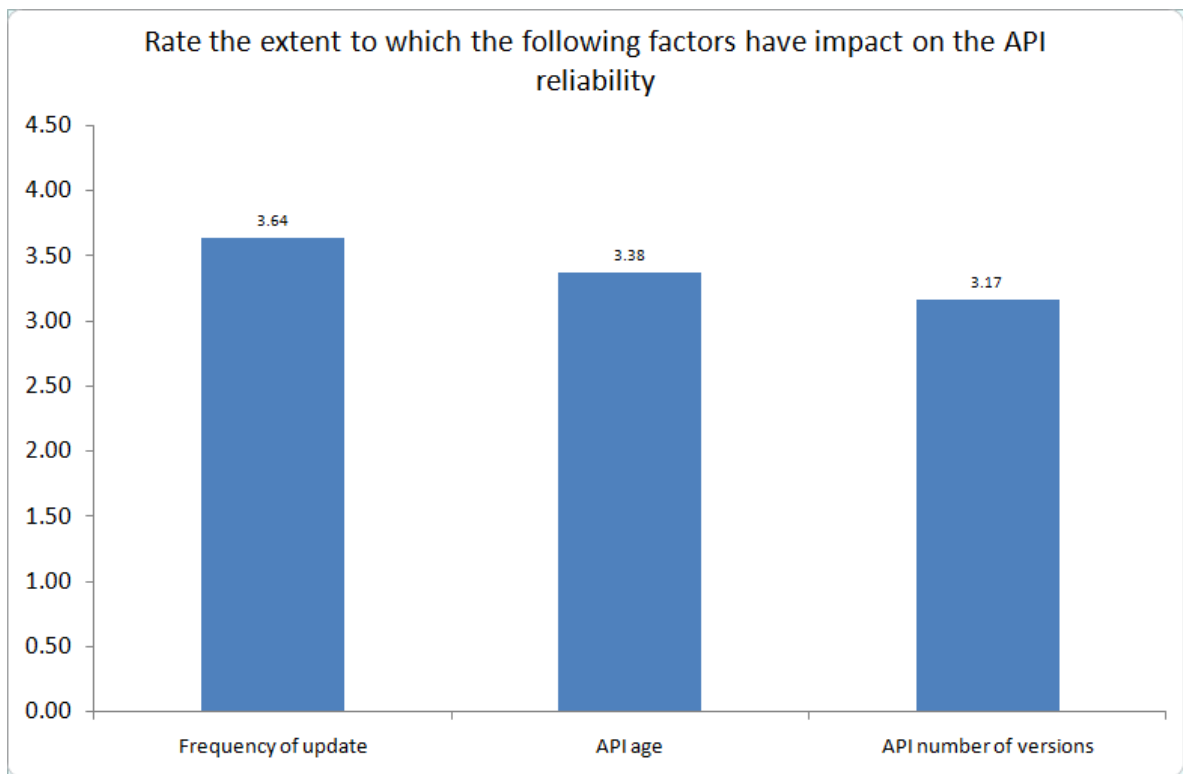


Figure 32

The programmers were also asked to evaluate the different formats in order to know which ones are more common, preferred and rated on the following two figures *Figure 33 & 34* we will see this in detail.

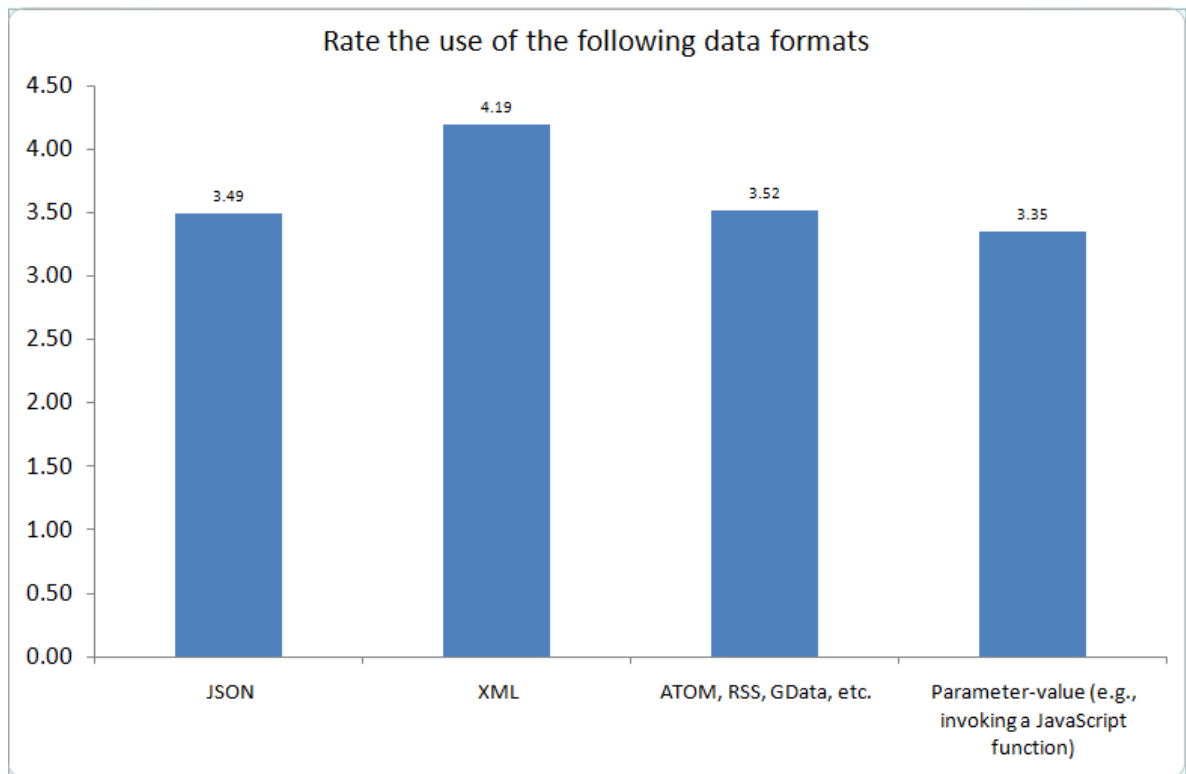


Figure 33

From a non functional perspective, choosing an API to compose a mashup implies taking decisions regarding the architectural style of the component, its programming language and the data formatting logic. All these aspects affect the appeal of the API from the point of view of the mashup composer and therefore influence his decision (Cappiello et al, 2010).

In the Quality Model, Data Operability is evaluated using two aspects: parsing that means that further transformation are needed before the component can be integrated in the final mashup, and the use of a standard structure. According to this, in the model it is stated that the data formats classify according to their operability are: Parameter-value pairs with high operability, follow by ATOM¹⁰,

¹⁰ ATOM defines a feed format for representing and a protocol for editing Web resources such as Weblogs, online journals, Wikis, and similar content. The feed format enables syndication; that is, provision of a channel of information by representing multiple resources in a single document. The editing protocol enables agents to interact with resources by nominating a way of using existing Web standards in a pattern. (datatracker.ietf.org)

RSS, GData¹¹ and JSON¹² all of them with medium operability, and in last place with low operability is XML.

In this case the results of the surveys differ from the model (*Figure 33*), since it was obtained that the most common format is XML then in second and third place ATOM, JSON with medium operability and at the end Parameter-value.

On the model the operability of technologies at the application level is evaluated by considering the diffusion and the interaction overhead of both protocols and languages used in the API development. According to this, in the model it is stated that the operability of the APIs are: JavaScript¹³ components with high operability, PHP/Perl/ASP/JSP¹⁴ component and RESTful service both with medium operability, and in last place SOAP¹⁵/WSDL¹⁶ service with low operability.

Also in this case the results of the surveys differ from the model (*Figure 34*), since it was obtained that the most common protocol and languages in order of usage are: SOAP – WSDL, followed by RESTful, and JAVASCRIPT, at the end PHP-PERL-ASP.

11 Google Data Protocol (GData) provides a simple standard protocol for reading and writing data on the Internet, designed by Google (*Wikipedia, 2010*)

12 JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate (*json.org, 2010*)

13 JavaScript is an implementation of the ECMAScript (scripting language standardized by Ecma International in the ECMA-262 specification and ISO/IEC 16262. The language is widely used for client-side scripting on the web, in the form of two well-known dialects, JavaScript and JScript) language standard and is typically used to enable programmatic access to computational objects within a host environment (*Wikipedia, 2010*)

14 These are all programming languages. PHP: Hypertext Preprocessor. ASP: Active Server Pages. PERL is not an Acronym (*Webhostingchoice, 2010*)

15 SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks (*Wikipedia, 2010*)

16 WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information (*W3, 2001*)

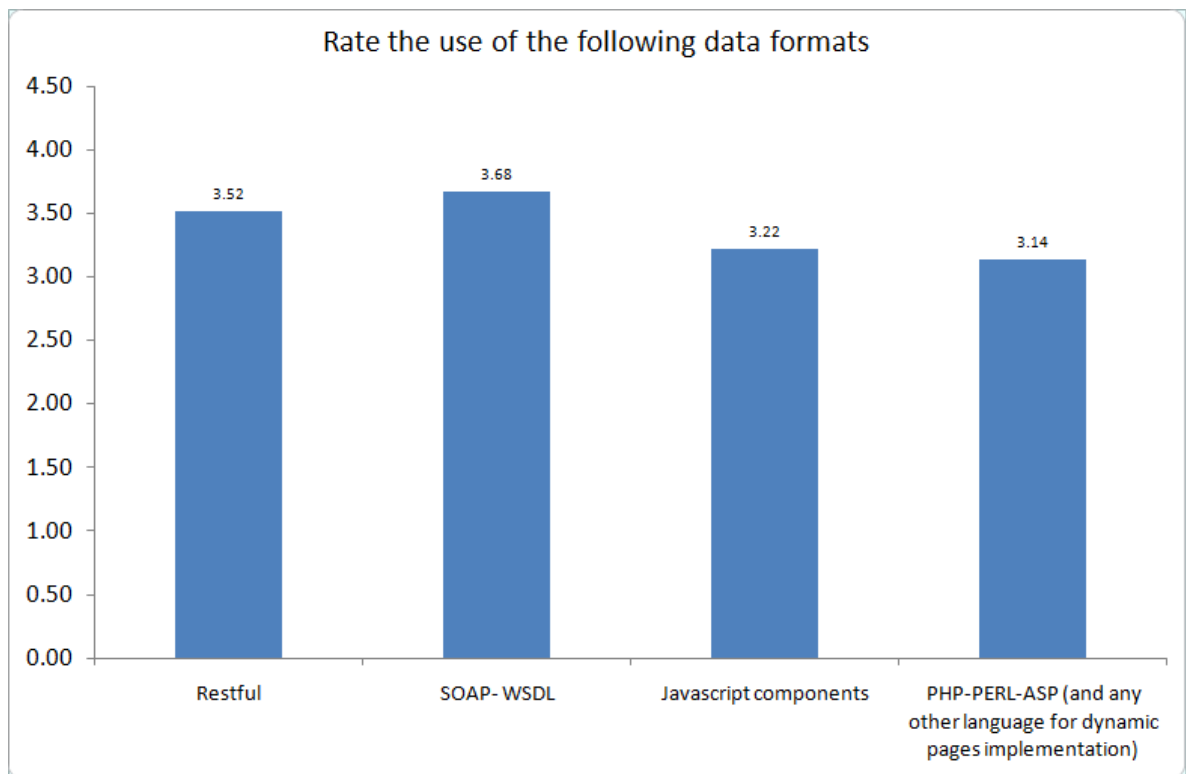


Figure 34

The security of a component is related to the protection mechanism that is used to rule the access to the offered functionalities. We distinguish between SSL support no authentication, API key, developer key, and user account (Cappiello et al, 2010).

Figure 35 shows that the variation on the security mechanisms is minimum, being these: SSL¹⁷, API key and SSL plus, the ones that prevents more from using an API. In the model the security mechanism are classified from the less effective to the most effective one in terms of protection that is use to rule the access to the offered functionalities are : No Authentication, API Key, Developer Key, User Account & SSL Support. This is being validated by the answer given by the developers shown on the graphic below were the final ranking was in first SSL plus,

¹⁷ Secure Socket Layer (SSL) is a cryptographic protocols that provide security for communications over networks such as the Internet(Wikipedia, 2010)

then second user account on third SSL then Developer key then API key and last No security.

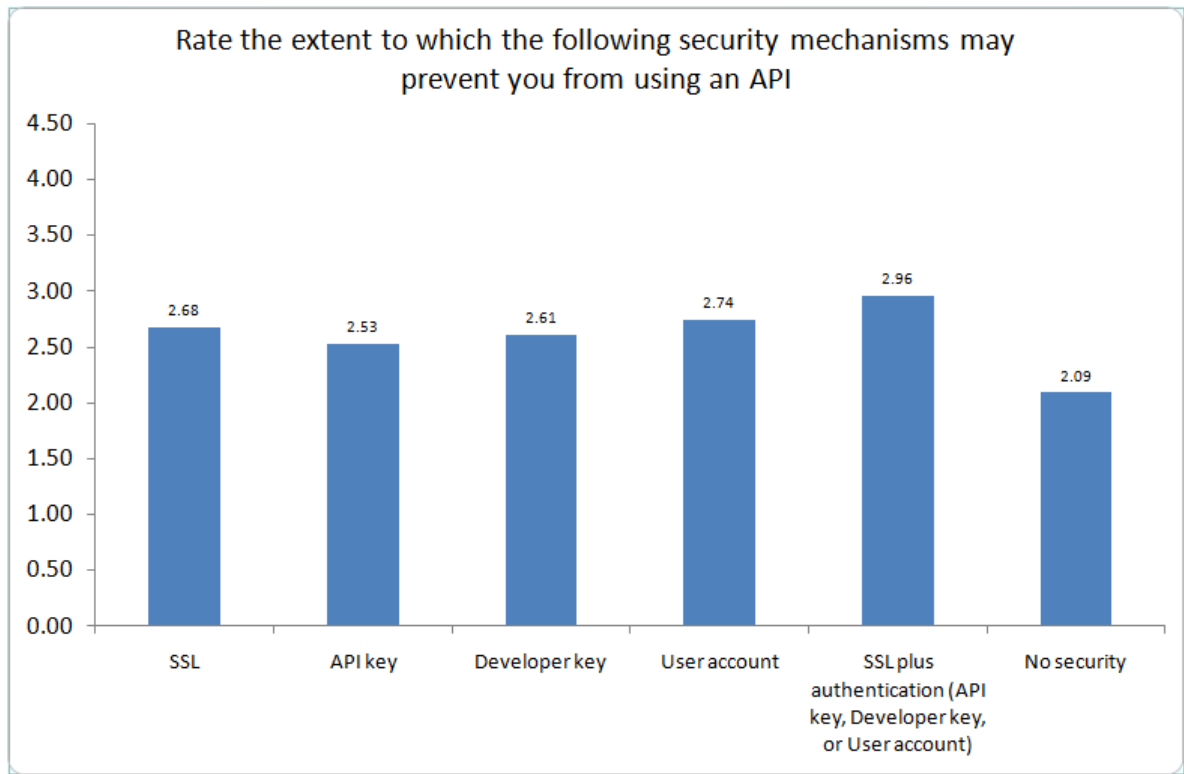


Figure 35

4.2 Composition Quality

In order to prove the composition quality that are the roles : Master, Slave and filter. The Patterns Master-Master, Master-Slave and Slave-Slave a detail work of analysis was made over the existing mashups in the web in order to verify that the proposal of the model are real. For this we select 200 mashups out of the 4000 that were active in the web at April 2010 from all the categories based on programmableweb.com.

The total universe of mashups is segment in different categories and each one has a different percentage amount of mashups according to its popularity and to if it is common or not, so we divided our sample with this same percentage distribution that the whole mashup universe had, so the analysis could keep the same representation per category than that of the reality and any variations were possible to be find in it.

These are the categories analyzed:

- “Mapping Mashups” 52 mashups analyzed. This is the biggest category of the ones analyzed and in it are used the majority of the most popular elements, like: Google maps, Facebook, Youtube, Amazon, Flickr, eBay etc...
- “Social Mashups” with 26 mashups analyzed
- “Photo Mashups” with 20 mashups analyzed
- “Shopping Mashups” with 18 mashups analyzed
- “Video Mashup” with 19 mashups analyzed
- “Search Mashups” with 17 mashups analyzed
- “Travel Mashups” with 13 mashups analyzed
- “Music Mashup” with 13 Mashups analyzed
- “News Mashup” with 13 mashups analyzed
- and “Messaging Mashups” with 9 mashups analyzed.

Each one of the mashups on the sample was individually analyzed to understand the way it works, the pattern it has and the elements involved. From this analysis we got the following data for each one of the mashups:

- Name
- Description and date of addition
- The presence of a “Filter”
- The existence of “ad hoc data” and the role this data has
- All the APIs involved in the Mashups and the role they play
- and at the end the pattern of the mashups (Master-Master, Master-Slave, Slave-Slave).

All the data gathered was inserted into a matrix that was use as support to run all the data analysis, and graphical results proving that the proposed patters and roles described in the model are indeed the once that depict the mashups, are deployed on next chapter.

4.2.1 Mashup List Analysis

The mashup have typical roles: Master, Slave and Filter, based on these three roles it can be identify three basic patterns that characterize most of today’s mashup applications :

- Master - Master
- Slave - Slave
- Master – Slave

Finally mashup contains some proprietary components. For example, usually when there is the Google maps component, the mashup creator provides a component that contain data to put on the map, this component we call it “Ad-Hoc data”

To prove this roles and patters we analyze 200 mashups and the following are the results that prove right this model that states the roles and patters mentioned

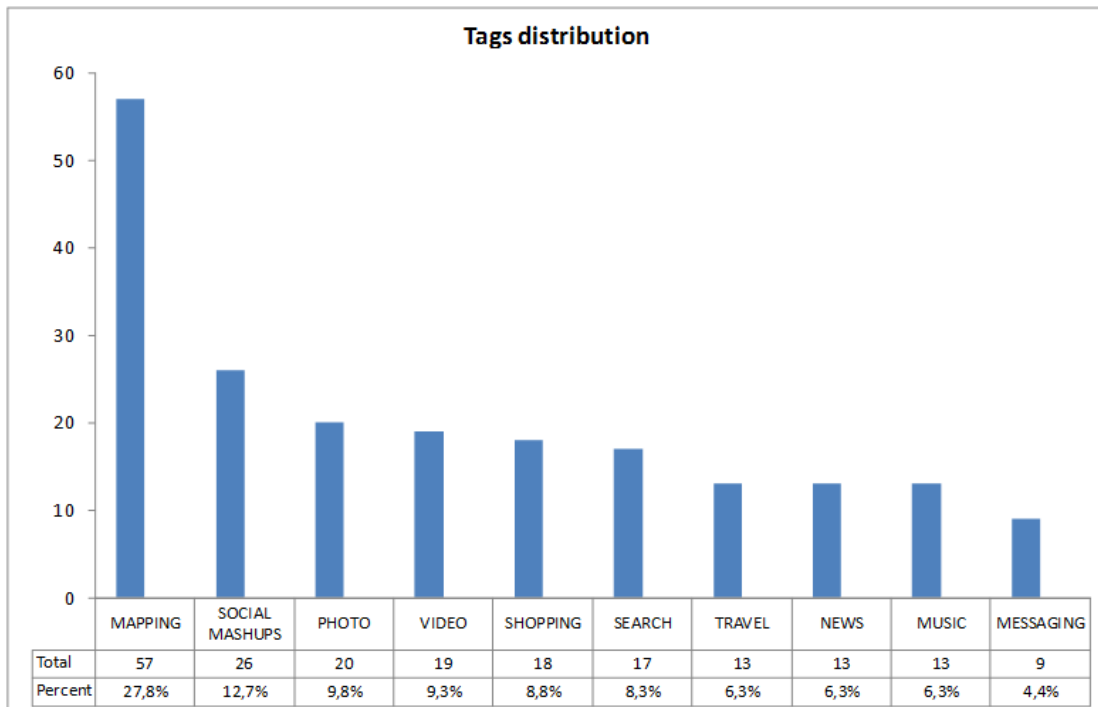


Figure 36

On the *Figure 36* it is shown the distribution of the sample taken into analysis. All mashups on programmableweb.com are divided into tags or categories, depending on their area of application, the APIs they use or the need they solve. It is worth to say that one single mashup could lay under more than one tag, this due to the fact that different functions could be perform by the same mashup, thanks to the properties of their different component APIs.

According to our sample most of the mashups being currently develop belong to the Mapping category (almost 30 percent of the total mashups being develop), and Messaging (4.4 percent) , Travel, News and Music (all three with 6.3 percent each) are the less popular categories ones according to the percentage they have on the total existence mashup world. (*Programmableweb.com, March 2010*).

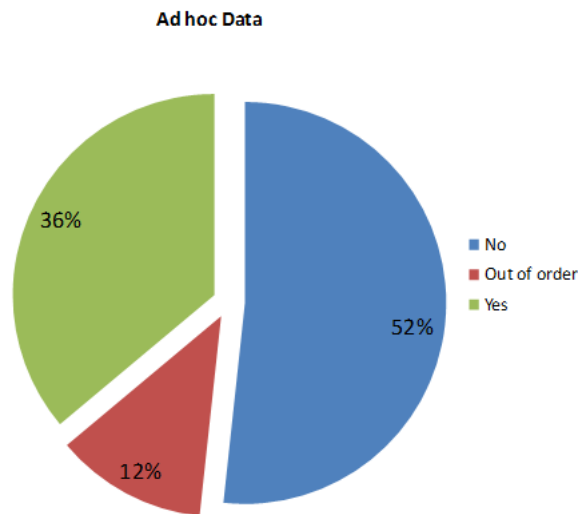


Figure 37

In *Figure 37* it is shown that a 52% of the mashups contain Ad hoc Data and a 36% of them do not. The red part correspond to the 12 % of mashups that out of order.

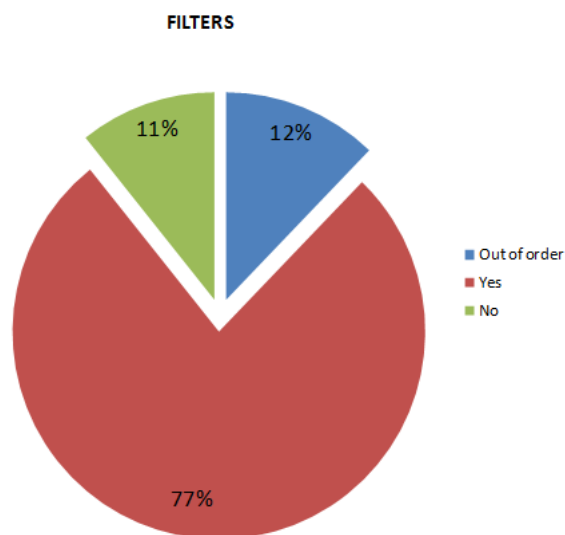


Figure 38

Figure 38 shows the percentage of mashups that have Filters and the ones that do not, the blue part with 12 % is the one of mashups out of order

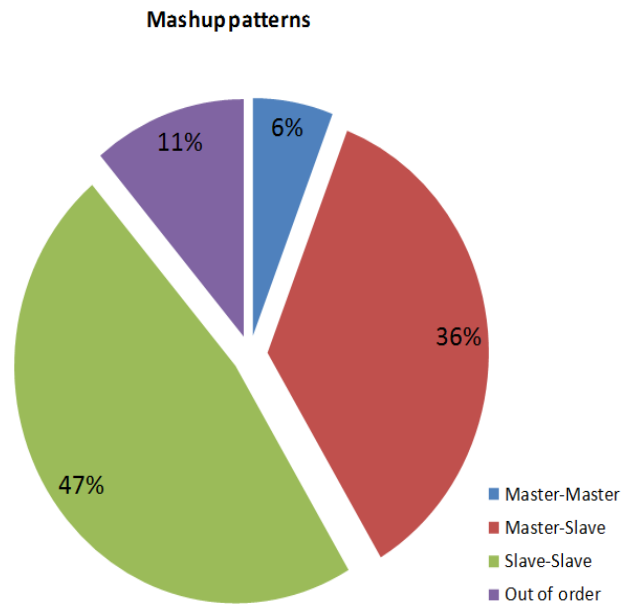


Figure 39

According to the analysis applied to the sample see *Figure 39*, most of the mashups present a Slave-Slave pattern (47%), Master-Slave (36%) and the Master-Master pattern the lower one, present in the (6%) of the mashups from the sample. After analyzing the 200 mashup selected as a sample we conclude that they always follow the patters proposed and that there are no other patters stricter of mashups available, this three patters include all.

The patterns present in the analyzed tags are shown in *Figure 40*. As we can see in the mapping category the most common pattern is Master-Slave, while in the rest of the categories is the pattern Slave-Slave the one that is present in most of the mashups.

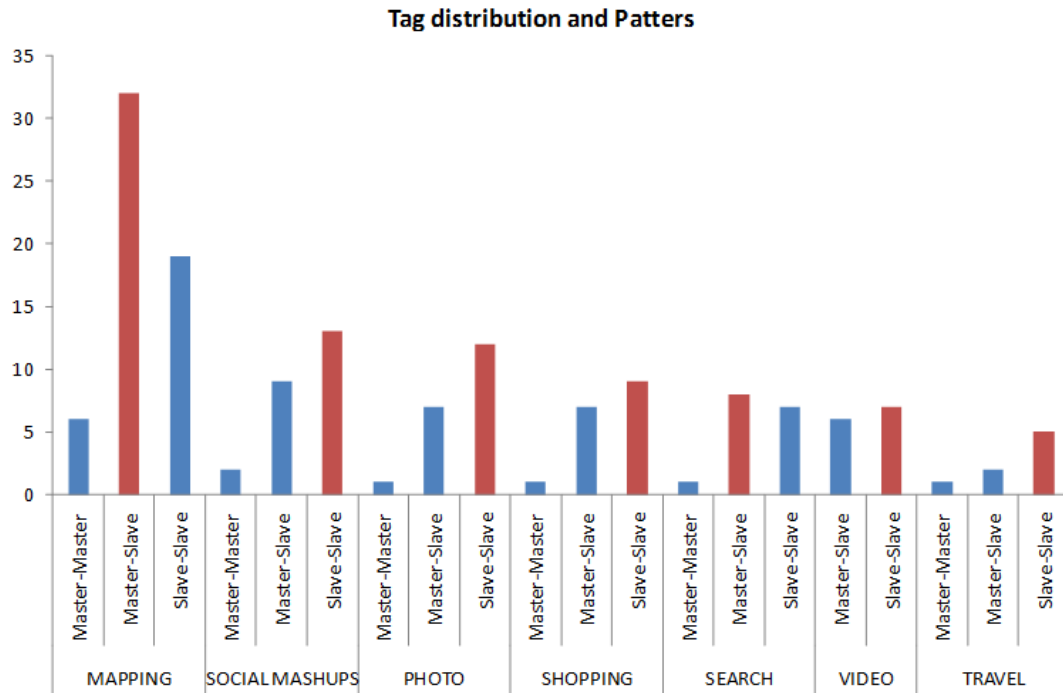


Figure 40

When developing a new mashup, selecting the right APIs is an important part since these will be performing the functions that we want the new application service to have. According to our analysis there are some APIs that have consolidated a primary position among developers at the time of developing a mashup. The election of this APIs depend on different factor that interest the programmers, further ahead we will go deeper into this matter.

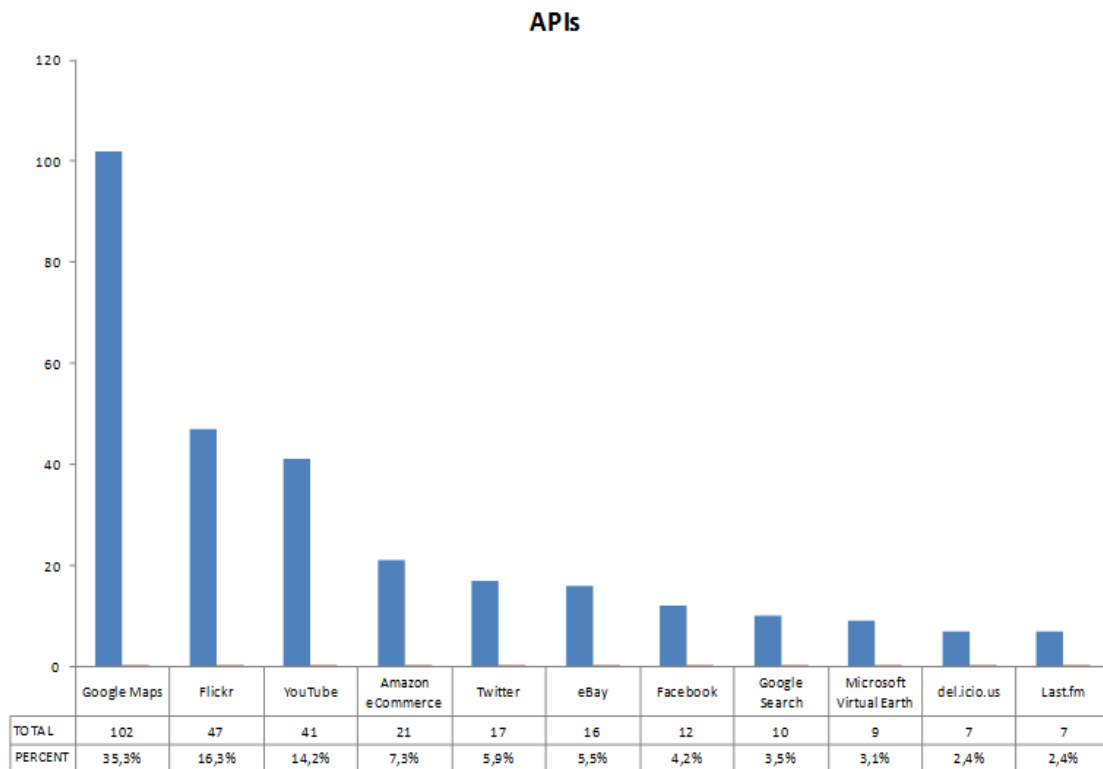


Figure 41

Figure 41 shows us the top 10 APIs that appear in a more frequent way along the analyzed sample, we can see a clear dominant presence from Google Maps, followed by Flickr and YouTube. Most of this APIs belong to well know companies or Web pages, that have putted their data out there on the web so it can be combined and used as convenient for final client. This is how they work:

- *Google Maps API.* The Google Maps API allow for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript¹⁸ interface or a Flash interface. It is designed to work on both mobile devices as well as traditional desktop browser applications. The API includes language localization for over 50 languages, region localization

¹⁸ A scripting language developed to design interactive sites(*webopedia.com, 2010*)

and geocoding, and has mechanisms for enterprise developers who want to utilize the Google Maps API within an intranet (*programmableweb.com, 2010*)

- *Flickr*. The Flickr API can be used to retrieve photos from the Flickr photo sharing service using a variety of feeds - public photos and videos, favorites, friends, group pools, discussions, and more. The API can also be used to upload photos and video (*programmableweb.com, 2010*).

- *YouTube*. The Data API allows users to integrate their program with YouTube and allow it to perform many of the operations available on the website. It provides the capability to search for videos, retrieve standard feeds, and see related content. A program can also authenticate as a user to upload videos, modify user playlists, and more. This integration can be used for a variety of uses such as developing a web application allowing users to upload video to YouTube, or a device or desktop application that brings the YouTube experience to a new platform. The Data API gives users programmatic access to the video and user information stored on YouTube. This can be used to personalize a web site or application with the user's existing information as well as perform actions like commenting on and rating videos(*programmableweb.com, 2010*).

- *Amazon eCommerce API*. What was formerly the ECS - eCommerce Service -has been renamed the Product Advertising API. Through this API developers can retrieve product information. The API exposes Amazon's product data and e-commerce functionality. This allows developers, web site publishers and others to leverage the Amazon Product Discovery features that Amazon uses to power its own business, and potentially make money as an Amazon affiliate. Additionally, the API has features allowing developers to advertise products, let users search for Amazon products and help users discover Amazon products(*programmableweb.com, 2010*).

- *Twitter API.* The Twitter micro-blogging service includes two RESTful APIs¹⁹. The Twitter REST²⁰ API methods allow developers to access core Twitter data. This includes update timelines, status data, and user information. The Search API methods give developers methods to interact with Twitter Search and trends data(*programmableweb.com, 2010*).
- *eBay API.* World's largest online auction service. API allows for both searching of products and upload of new listings(*programmableweb.com, 2010*).
- *Facebook API.* The Facebook API is a platform for building applications that are available to the members of the social network of Facebook. The API allows applications to use the social connections and profile information to make applications more involving, and to publish activities to the news feed and profile pages of Facebook, subject to individual users privacy settings. With the API, users can add social context to their applications by utilizing profile, friend, Page, group, photo, and event data(*programmableweb.com, 2010*).
- *Google Search API.* As of December 2006 this API is no longer actively supported by Google. They suggest that developers use the Google Ajax Search API(*programmableweb.com, 2010*).
- *The Google AJAX Search API* allows you to embed Google Search in your web pages and other web applications. You can embed a simple, dynamic search box and display search results in your own web pages or use the results in innovative, programmatic ways. The API allows developers to integrate Web Search, News Search, and Blog Search into their web site. Add Local Search results to their web site, or integrate them with their Google Maps API mashup. Add YouTube Videos and

¹⁹ A RESTful web service (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of REST.(*Wikipedia, 2010*)

Google Image Search results to their web site or [blog\(programmableweb.com, 2010\)](#).

- *Microsoft Virtual Earth API*. Microsoft Virtual Earth has been renamed Bing Maps. Use BM to build maps which can include routes and traffic info. Gives developers the ability to code the controls, shapes, and layers of the maps, and can summon the birds-eye, 3D, and aerial imagery. For commercial applications Bing Maps Web Services allow users to integrate maps and imagery, driving directions, and other location features into a Web application([programmableweb.com, 2010](#)).
- *del.icio.us API*. From their site: del.icio.us is a social bookmarking website -- the primary use of del.icio.us is to store your bookmarks online, which allows you to access the same bookmarks from any computer and add bookmarks from anywhere, too. On del.icio.us, you can use tags to organize and remember your bookmarks, which is a much more flexible system than folders([programmableweb.com, 2010](#)).
- *Last.fm API*. The Last.fm API gives users the ability to build programs using Last.fm data, whether on the web, the desktop or mobile devices. It allows for read and write access to the full slate of last.fm music data resources - albums, artists, playlists, events, users, and more([programmableweb.com, 2010](#)).

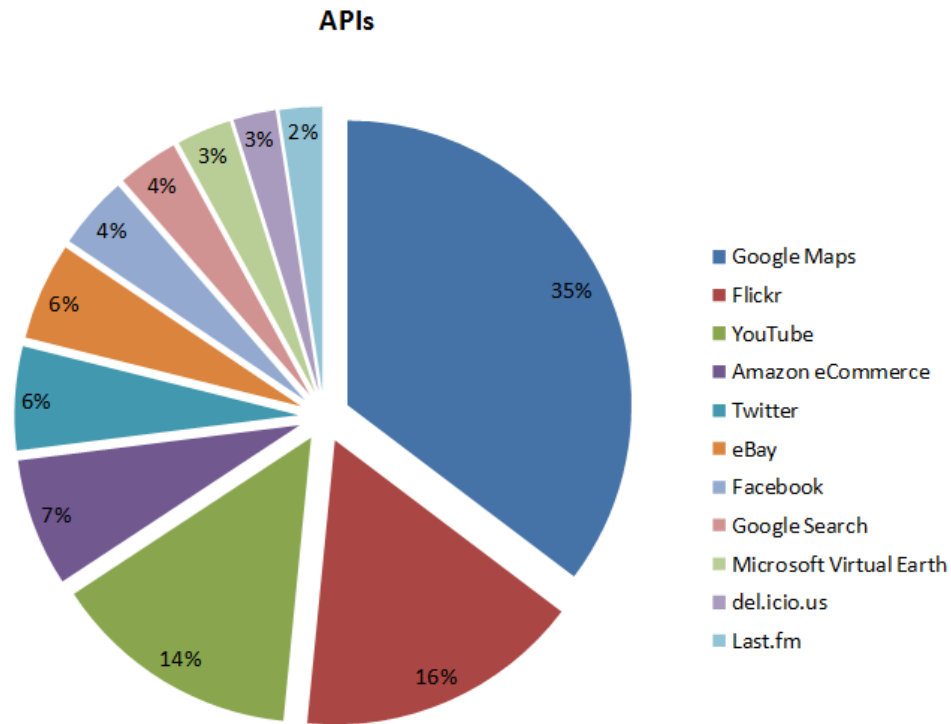


Figure 42

The number of mashups developed by year has been changing since the creation of the first mashups; this might be due to the fact the being creative and innovative becomes more and more difficult as time goes by since most of the ideas would be already executed and coming up with something new result into a time consuming task.

Figure 43 shows the growth the mashup development has had during the past five years based on the date of release of the mashups from the sample taken for analysis. Is evident the pick of innovation and creation of new mashups that happened in year 2008 and it has decreased abruptly in 2010.

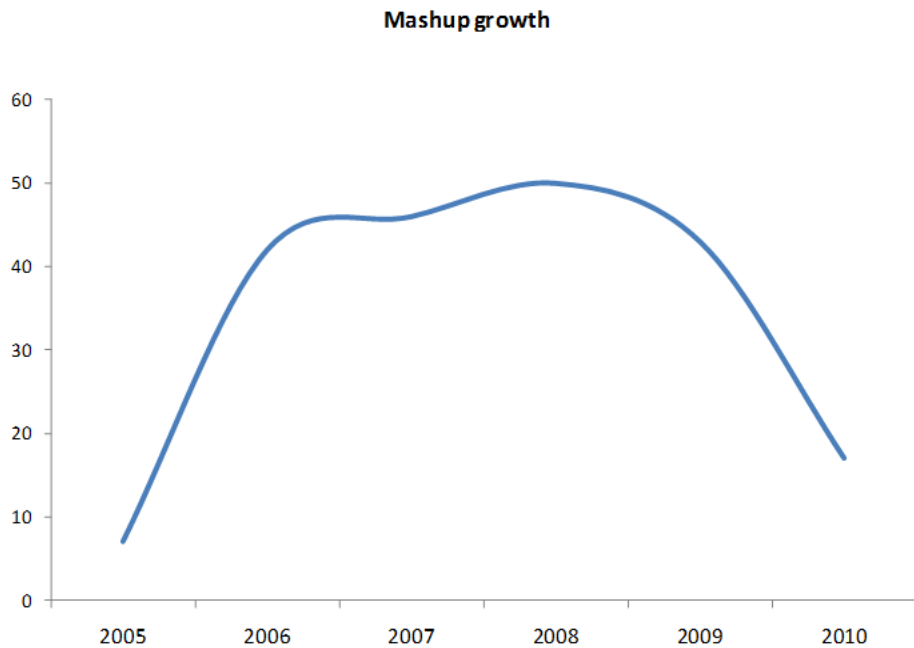


Figure 43

With this information analysis we validated that the roles and patters (Master, Slave and Filter) & (Master – Master, Master – Slave, Salve – Slave) are sufficient to involve all the types of mashups, and that all the variety of mashups relay on the same patters and roles proposed by the Quality Model.

4.2.2 Association Rules

Association rule mining finds interesting associations and/or correlation relationships among large set of data items. Association rules show attribute value conditions that occur frequently together in a given dataset.

Association rules provide information of this type in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

In addition to the antecedent (the "if" part) and the consequent (the "then" part), an association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called item sets) that are disjoint (do not have any items in common).

The first number is called the **support** for the rule. The support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

The other number is known as the confidence of the rule. **Confidence** is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

In this work we used the association rules in order to identify those elements that constantly appear in the architecture of the analyzed mashups together. The main idea is to uncover the frequency with which these APIs appear together on the analyzed mashups, and in this way discover the most important and relevant correlations hidden within the mashups structures.

4.2.2.1 Association Analysis

From our mashup analysis of 200 mashups we identified the top 7 most commonly used APIs. The 80% of the total quantity of APIs used in the 200 mashup analysis is contained within these top 7 APIs.

In order to identify the association rules we eliminated the mashups in which just one API was been used or there were no significant APIs involved (Top 7). After this first analysis the original sample of 200 mashups was reduced to 116 mashups on which we ran the association rules analysis.

These top 7 APIs and the number of times they appear in the 116 mashup list are:

- Google Maps 80 times
- YouTube 67 times
- Flickr 62 times
- Amazon eCommerce 25 times
- Twitter 25times
- eBay 15 times
- Facebook 12 times

On the following figures under the X column it is shown one of the top 7 APIs and on the Y column the API with which it is been associated. Under the XuY column it is indicated the number of times that these APIs appeared together in the 116 mashup list. Under the Support column is the rate between the number of mashups containing “XuY” and the 116 mashups. Finally under the confidence column it is indicated the rated between the number of mashups containing “XuY” and the number of mashups containing “X”.

X	Y	X u Y	Support	Confidence
Google Maps	YouTube	48	41%	60%
Google Maps	Flickr	42	36%	53%
Google Maps	Amazon eCommerce	6	5%	8%
Google Maps	Twitter	13	11%	16%
Google Maps	eBay	6	5%	8%
Google Maps	Facebook	6	5%	8%

Figure 44

On *Figure 44* we analyzed the relation between Google maps and the rest of the 7 APIs, we can see that the strongest association is between Google Maps and YouTube which appeared together 48 times in the structure of the analyzed mashups.

X	Y	X u Y	Support	Confidence
YouTube	Flickr	31	27%	46%
YouTube	Amazon eCommerce	16	14%	24%
YouTube	Twitter	11	9%	16%
YouTube	eBay	8	7%	12%
YouTube	Facebook	4	3%	6%

Figure 45

Figure 45 shows the association founded for YouTube with the remaining top 7 APIs being the strongest one the one with Flickr (31 times among the 116).

X	Y	X u Y	Support	Confidence
Flickr	Amazon eCommerce	11	9%	18%
Flickr	Twitter	14	12%	23%
Flickr	eBay	5	4%	8%
Flickr	Facebook	7	6%	11%

Figure 46

Figure 46 shows the association between Flickr and the remaining top 7 APIs being the strongest one the one with Twitter (14 times among the 116).

X	Y	X u Y	Support	Confidence
Amazon eCommerce	Twitter	3	3%	12%
Amazon eCommerce	eBay	10	9%	40%
Amazon eCommerce	Facebook	2	2%	8%

Figure 47

Figure 47 shows the association between Amazon eCommerce and the remaining top 7 APIs being the strongest one the one with eBay (10 times among the 116).

X	Y	X u Y	Support	Confidence
Twitter	eBay	1	1%	4%
Twitter	Facebook	4	3%	16%

Figure 48

Figure 48 shows the association between Twitter and the remaining top 7 APIs being the strongest one the one with Facebook (4 times among the 116).

X	Y	X u Y	Support	Confidence
eBay	Facebook	1	1%	7%

Figure 49

Figure 49 shows the relation between eBay and Facebook comes down to 1 time only.

To complete the analysis we ran the association rules among tree and more APIs at the same time, the information obtain is shown in the *Figures 50, 51, 52 & 53*.

X	Y	Z	XY u Z	Support	Confidence
Google Maps	YouTube	Flickr	17	15%	27%

Figure 50

X	Y	Z	W	XYZ u W	Support	Confidence
Google Maps	YouTube	Flickr	Amazon eCommerce	3	3%	12%

Figure 51

X	Y	Z	W	V	XYZW u V	Support	Confidence
Google Maps	YouTube	Flickr	Amazon eCommerce	Twitter	0	0%	0%

Figure 52

X	Y	Z	W	R	XYZW u R	Support	Confidence
Google Maps	YouTube	Flickr	Amazon eCommerce	eBay	1	1%	7%

Figure 53

Conclusions

After chapter 4 each dimension present on the model has been validate, demonstrating its importance on the development of quality mashups.

Thanks to analysis of the preferences of developers through the survey we have been able to identify the preferences and concerns that developers shown towards each of the dimensions, the importance assigned to each one of them and to validate what was stated on the model.

Also after the analysis of the sample of mashups token from programmableweb we have been able to identify certain patterns of use, correlations and preference towards certain APIs when building up mashups, allowing us to have wide vision of the trends occurring on the development of mashups field.

5

CONCLUSION

Mashups are a mean for people to manipulate the information around them trough and active engagement with different sources. The mashup, through its mixing and synthesizing of existing data, suggests a means for these individual expressions to build on one another. Thinking of mashups as this powerful way to create service applications, we also know that the real value that it might have for one person will be completely different from others, and would highly depend on the use, expertise or area of application in which they are located. At the same time the requirements for quality will not be the same, since a developer working within an enterprise would be more interest in the quality of the documentation while a private developer would care more about the technology aspects. However, mashups may soon (or already have) become so integrated into the fabric of our experience of the internet and information sharing, that the term may no longer be meaningful.

In this work we have proof that quality in the mashup area is substantially relevant for all the involved parts and that quality mushups are not unimportant, for the contrary they are essential for the reach of the desire information or out-put. It has been proof also what factors are more relevant for the quality mashups and their percentage so we can in future works based our analysis in a more detail way.

It has been also shown the way mashups are built right now and the patterns they follow so far and their components and API mainly used. All this has led towards the approval of the Quality Mashup Model that states that the quality of the mashups can be divided in terms of the composer, developer and user and that the quality should be on the components and on the integration being these two the main and basic elements to gain a quality mashup offering attributes that measure the privileges of the component to determine its quality that is indeed the perspective that is most relevant to the mashup composer or the mashup user

REFERENCES

- Alur, Deepak. 2007. Defining Mashups. The Enterprise Web 2.0 Blog. JackBe. July 22 <http://blogs.jackbe.com/2007/07/defining-mashups.html>
- von Busch, Otto, and Karl Palmås. 2006. Abstract Hacktivism: The making of a hacker culture. London: OpenMute.
- Cappiello C., Daniel F., Matera M., Pautasso C., 2010 Information Quality in Mashups. Politecnico di Milano, Italy, University of Trento, Italy, University of Lugano, Switzerland.
- Cappiello C., Daniel F., Matera M., 2010 A Quality Model for Mashups Components. Politecnico di Milano, Italy, University of Trento, Italy.
- Darlin, Damon. 2005. A Journey to a Thousand Maps Begins With an Open Code. The New York Times, October 20.
- Fielding, Roy. 2000. Architectural Styles and the Design of Network-based Software Architectures <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- Hof, Robert D. 2005. Mix, Match, And Mutate. Bloomberg Businessweek, July 25. http://www.businessweek.com/magazine/content/05_30/b3944108_mz063.htm
- Jhingran A., Enterprise information mashups: integrating information, simply. VLDB'06, pp.3-4.
- IETF. Atom Publishing Format and Protocol. <http://datatracker.ietf.org/wg/atompub/charter/>
- JSON. *Introducing JSON*. <http://www.json.org/>.
- 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>.
- Palfrey J., Gasser U. 2007, Mashups Interoperability and Innovation, Berkman Publication Series, November 2007. <http://cyber.law.harvard.edu/interop>
- PressFeed, 2009. RSS feed – A tutorial http://www.press-feed.com/howitworks/rss_tutorial.php

©ProgrammableWeb.com, 2010. All rights reserved.
<http://blog.programmableweb.com/2010/04/14/twitter-mashup-growth-2009-was-big-now-what/>
<http://www.programmableweb.com/api/google-maps>
<http://www.programmableweb.com/api/flickr>
<http://www.programmableweb.com/api/youtube>
<http://www.programmableweb.com/api/amazon-ecommerce>
<http://www.programmableweb.com/api/twitter>
<http://www.programmableweb.com/api/ebay>
<http://www.programmableweb.com/api/facebook>
<http://www.programmableweb.com/api/google-search>
<http://www.programmableweb.com/api/google-ajax-search>
<http://www.programmableweb.com/api/microsoft-virtual-earth>
<http://www.programmableweb.com/api/del.icio.us>
<http://www.programmableweb.com/api/last.fm>

Sharpened Glossary Definition of Computer Terms,
<http://www.sharpened.net/glossary/definition.php?mashup>

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

Spool, Jared. 2007. Web 2.0: The Power Behind the Hype, User Interface Engineering. http://www.uie.com/articles/web_2_power/

©TechTerms.com, 2005-2010. All rights reserved.
<http://www.techterms.com/definition/mashup>

W3C, 2001. Web Services Description Language (WSDL) 1.1. March 15
<http://www.w3.org/TR/wsdl>

WebHostingChoice, 2010. <http://www.webhostingchoice.com/faq/what-are-php-asp-perl.shtml>

Webopedia, 2010, JavaScript. <http://www.webopedia.com/TERM/J/JavaScript.html>

Wikipedia, the free encyclopedia, 2010, ECMAScript
<http://en.wikipedia.org/wiki/ECMAScript>

Wikipedia, the free encyclopedia, 2010, GData
http://en.wikipedia.org/wiki/Representational_State_Transfer

Wikipedia, the free encyclopedia, 2010, JavaScript
<http://en.wikipedia.org/wiki/JavaScript>

Wikipedia, the free encyclopedia, 2010, Information Technology
http://en.wikipedia.org/wiki/Information_technology

Wikipedia, the free encyclopedia, 2010, Metamodeling
<http://en.wikipedia.org/wiki/Metamodeling>

Wikipedia, the free encyclopedia, 2010, SOAP <http://en.wikipedia.org/wiki/SOAP>

Wikipedia, the free encyclopedia, 2010, Representational State Transfer
http://en.wikipedia.org/wiki/Representational_State_Transfer

Wikipedia, the free encyclopedia, 2010, Transport Layer Security
http://en.wikipedia.org/wiki/Transport_Layer_Security

ANNEX

5.1 Annex “A” List of Mashups

The list of mashup analyzed to prove this Quality model are:

- | | | | |
|-----|--|----|---|
| 1 | 08 celebrity pictures, videos and news | 23 | Connecting Consumers and Businesses in Cities, Worldwide! |
| 2 | 10 Fascinating Googlers | 24 | GMaps Flight Tracker |
| 3 | 1001 Secret Fishing Holes | 25 | Realtime Satellite Tracking Map |
| 4 | 2008 US Electoral Map | 26 | CityRanks US Populations |
| 4.1 | 2009 US Electoral Map | 27 | 100 Most Powerful Celebrities |
| 4.2 | 2010 US Electoral Map | 28 | 2008 Basketball Tourneys |
| 4.3 | 2011 US Electoral Map | 29 | 2009 Formula One Map |
| 4.4 | 2012 US Electoral Map | 30 | 25 Best Companies to Work For |
| 4.5 | 2013 US Electoral Map | 31 | 29 Travels |
| 5 | 250 Wedding Tent Venues | 32 | 2Spaghi |
| 6 | 2itch - Open 24 hours | 33 | 360 degree Sardinia |
| 7 | 2RealEstate Auctions | 34 | 4Hotels.us Hotel Maps |
| 8 | 360 Cities | 35 | 5 TVs |
| 9 | 43things GeoSearch | 36 | Access Denied Map |
| 10 | Geolover | 37 | Zillow.com |
| 11 | A World of Nirvana | 38 | Zimride |
| 12 | Acid | 39 | Yoga Yoga Yoga |
| 13 | ActiveTrails | 40 | Your Mapper |
| 14 | Africa Bespoke | 41 | YouTube Slideshow on Google Maps |
| 15 | Ajax Map Comparison | 42 | Zeeqa |
| 16 | All of Ibiza from one Google Map | 43 | Zillion Events |
| 17 | alkemis local nyc | 44 | Worlds Fastest Elevators |
| 18 | Top 99 Women on Google Maps | 45 | Yamusica |
| 19 | Geowalk | 46 | World and Regional Earthquakes |
| 20 | Facebook Friend Mapper | 47 | World Port Source |
| 21 | Run London | 48 | World Time Engine |
| 22 | WeatherMole | | |

49	Windows Live Contacts Map	90	Daily Mashup
50	Wolpy	91	depictr
51	What's happening, London?	92	Earthplacemarks
52	Where Can I Live?	93	Eventsites
53	1click2destiny	94	Flickr inSuggest
54	22books	95	Flickr Related Tag Browser
55	a.placebetween.us	96	Flickr Slide Show
56	AgeAnalyzer	97	flickr wrappr
57	Ask 500 People	98	Follow the Music
58	Autopendium	99	Woya Shopping
59	Beergeeks - Beer Finder	100	XY Shop
60	Blip.fm	101	Yard Sale Maps
61	Blog on a Map	102	Yelp Search
62	Checkin Mania	103	Yosle yardsales
63	Chitter.TV	104	Zeeqa
64	CitySounds.fm - The sounds of cities	105	Used Cars
65	Contoso University	106	Valentines Day Map
66	DormItem Free College Classifieds	107	Velyoo Local Marketplaces
67	favreel.com	108	Verkoops.com Search
68	FillUs.in	109	Weekend Treasure Garage Sales
69	Flittr	110	Tour de Sound
70	glancemap	111	Trading Vans
71	GoMojo	112	Tupalo
72	Got Search: Google and Twitter	113	Twitter Meets Amazon Wishlist
73	Gridpop	114	Spoiler 4 Movie
74	Hand Picked Twitter	115	Superhighstreet.com
75	I am here	116	ResultR
76	Ipoki	117	2lingual Bilingual Search
77	Lyrics Muse	118	A World of Nirvana
78	Tinkrbox	119	YouTube Vision
79	What's on Twitter	120	World Cup Soccer - Latest GeoTagged YouTube V
80	Zoogle IN	121	TopicTrends
81	Yimmiy	122	Toronto Buddhism
82	viewAt.org - The World in panoramic	123	TotalVideos
83	TruDat	124	TravelMapia
84	TuneChimp	125	Trendite
85	4 in 1 search	126	TrendyNewz
86	Blog on a Map	127	ChizMax Lyrics and Video Search Engine
87	Blogabond World Map	128	Chromomulator
88	Congress SpaceBook	129	Cleepr
89	CoolFlick	130	Customize Travel Mashup

131	dbrec	172	Album art search
132	Doodle Source	173	Albumart.org
133	DoubleTab Video	174	AllOrNone.org Pearl Jam
134	Dvdz Review	175	Album Cover Art
135	idiomag	176	Alertunes
136	Wikicinema	177	Arcane Pillage
137	Yahoo Buzz	178	Artist Explorer
138	Yahoo! APIs	179	4 in 1 search
139	YouTube Made Simple and Visual	180	AlertNet
140	Twitter Top News Trends	181	All Around China - Live China 360 Search
141	Twitter-Trending Local Restaurants	182	AP National News + Google Maps
142	UK Job Search	183	BBC News Map
143	Ultrasearchula	184	BBC News on Mobile
144	Unofficial Tupalo Widget	185	Best Cities For Singles
145	vdiddy	186	Bester News
146	googlUpon	187	Breaking New Map Flash
147	Googlecloud	188	Daily Mashup
148	Google vs Yahoo Visualized	189	LastNews
149	Got Search: Google and Twitter	190	Search The Web
150	GYbrowse	191	Oilaholic
151	Nextoid Shopping Search Engine	192	10 camera
152	oSkope	193	411Sync + Yahoo Traffic
153	10 Top US Cities Travel Guides	194	Agnostic Platform Aggregator
154	29 Travels	195	Airplane Booking System
155	360 Tuscany	196	Anyvite
156	411Sync Travel	197	Baebo
157	4Hotels.us Hotel Maps	198	buddyPing
158	7 Wonders of the World Map	199	Cool Bars, Restaurants and Clubs
159	71Miles	200	DeliciousMona
160	a.placebetween.us		
161	Africa in Style by Trek Holidays		
162	Africa Tourism Information Portal		
163	Air Travel Emissions Calculator		
164	Aircraft Flight Tracking Demo		
165	Bed and Breakfast Italy		
166	1000 songs		
167	44tips - your visual start page		
168	ain't just soul		
169	Air Veejay		
170	Akama Music		
171	Album Art Cloud		

5.2 Annex “B” API / Substitutes

The colors in the table below are used to classified the APIs in three main categories:

Dark blue.- Common API

Light blue.- Substitute API (API that works like the common one)

White.- API without substitute.

The last column shows the frequency of API which were used within the 200 mashup sample analyzed.

API / Substitute	API	Definition	QTY
API	Last.fm	Online radio service	7
Substitute	Blip.pm	Social music service	1
API	eBay	Online auction marketplace	16
Substitute	Amazon eCommerce	Online retailer	21
Substitute	SNOCAP	Digital music marketplace	1
API	Facebook	Social networking service	12
Substitute	Twitter	Microblogging service	17
Substitute	LinkedIn	Business social networking platform	1
Substitute	MySpace	Social networking service	1
API	Flickr	Photo sharing service	47
Substitute	Yahoo Image Search	Image search services	3
Substitute	Panoramio	photo upload site with organizer and geolocation	2
Substitute	Tumblr	Web scrapbook post and view service	1
API	Gigablast	Search service	1
Substitute	Google Ajax Search	Web search components	6
Substitute	Microsoft Bing	Internet search	1
Substitute	Yahoo BOSS	Customizable search service	2
API	Yahoo Search	Search services	3
Substitute	Google Search	Search services	10
API	Google Maps	Mapping services	102
Substitute	Microsoft Virtual Earth	Mapping services	9
Substitute	Yahoo Maps	Mapping services	5
Substitute	Google Earth	Mapping and 3D geo visualization	3
Substitute	OpenStreetMap	The free wiki world map	1
API	LyricWiki	Song lyrics search engine	2
Substitute	Lyricsfly	Song lyrics search engine	1

API	YouTube	Video sharing and search	41
Substitute	Vimeo	Video sharing service	1
API	Amazon A9 OpenSearch	Search services	1
API	del.icio.us	Social bookmarking	7
API	Digg	Community driven news links and ratings	1
API	Findory	Personalized news aggregation	1
API	FriendFeed	Activity stream aggregator	1
API	FUTEF Wikipedia API	Third party Wikipedia web service	1
API	Google AdSense	Advertising management	1
API	Google Ajax Feeds	Access RSS and Atom feeds with JavaScript	1
API	Google AJAX Libraries	Content distribution network for AJAX libraries	1
API	Google Analytics	Web analytics service	1
API	Google Base	Platform for structure and semi-structured data	1
API	Google Maps Data	Service to store, update and view geodata	1
API	Google Mashup Editor	Mashup creation tool extensions API	1
API	Google Spreadsheets	Online spreadsheets	3
API	Google Static Maps	Simple online mapping service	1
API	hostip.info	IP lookup	1
API	IntelePeer	Telephony From the Cloud Service	1
API	PriceRunner	Shopping comparison engine	2
API	SlideShare	Presentation sharing community	1
API	Upcoming.org	Collaborative event calendar	1
API	Wikipedia	Online collaborative encyclopedia	3
API	Windows Live Data	Service for users to control data access	1
API	Windows Live Expo	Online classifieds service	1
API	Yahoo Geocoding	Geocoding services	3
API	Yahoo Map Image	Map image creation service	1
API	Yahoo Term Extraction	Contextual search service	2
API	Yahoo Traffic	Traffic data and routing	1

5.3 Annex “C” Mashups Pattern list

Sample of the Mashup list data table.

No.	Name	Added	Filtered	Hoc Da	Type	APIs 1	Type 1	APIs 2	Type 2	APIs 3	Type 3	APIs 4	Type 4	APIs 5	Type 5	APIs 6	Patterns	
1	08 celebrity pictures	02-ago-08	Yes	No	N/A	Google-Bas	Slave	-Yahoo-In	Slave	YouTube	Slave	?	?	?	?	?	?	Slave-Slave
2	10 Fascinating Goog	02-feb-08	No	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
3	1001 Secret Fishing	28-nov-05	Yes	Yes	Slave	Google-Ma	Master	?	?	?	?	?	?	?	?	?	?	Master-Slave
4	2008 US Electoral M	04-jul-08	Yes	No	N/A	Google-Ma	Master	Google-S	Slave	?	?	?	?	?	?	?	?	Master-Slave
4.1	2009 US Electoral M	05-jul-08	Yes	No	N/A	Google-Ma	Master	Google-S	Slave	?	?	?	?	?	?	?	?	Master-Slave
4.2	2010 US Electoral M	06-jul-08	Yes	No	N/A	Google-Ma	Master	Google-S	Slave	?	?	?	?	?	?	?	?	Master-Slave
4.3	2011 US Electoral M	07-jul-08	No	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
4.4	2012 US Electoral M	08-jul-08	No	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
4.5	2013 US Electoral M	09-jul-08	Yes	Yes	Master	Google-Ma	Slave	YouTube	Slave	?	?	?	?	?	?	?	?	Master-Slave
5	250 Wedding Tent V	10-dic-09	Yes	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
6	2itc - Open 24 hour	17-sep-08	Yes	No	N/A	Google-Ad	Slave	Google-M	Master	?	?	?	?	?	?	?	?	Master-Slave
7	2RealEstate Auction	02-nov-05	Yes	No	N/A	eBay	Master	Google-Se	Slave	?	?	?	?	?	?	?	?	Master-Slave
8	360 Cities	11-may-08	No	Yes	Slave	Google-Ma	Master	?	?	?	?	?	?	?	?	?	?	Master-Slave
9	43things GeoSearch	05-sep-08	Yes	No	N/A	YahooGeoc	Slave	43Thingsy	Slave	Yahoo-Ma	Slave	?	?	?	?	?	?	Slave-Slave
10	Geolover	23-apr-08	Yes	No	N/A	Flickr	Slave	Foursquar	Slave	Google-M	Master	Google-M	Slave	Google-St	Slave	Twitter	Master-Slave	
11	A World of Nirvana	07-abr-07	Yes	Yes	Master	YouTube	Slave	Google-M	Slave	?	?	?	?	?	?	?	?	Master-Slave
12	Acid	17-mar-07	Yes	No	N/A	Yahoo-Geoc	Slave	Yahoo-Loc	Slave	Yahoo-Ma	Slave	?	?	?	?	?	?	Slave-Slave
13	ActiveTrails	15-ago-06	No	Yes	Master	Google-Ma	Master	?	?	?	?	?	?	?	?	?	?	Master-Master
14	Africa Bespoke	04-dic-09	No	Yes	Master	Google-Ear	Slave	Google-M	Slave	Google-M	Slave	Flickr	Slave	YouTube	Slave	?	?	Master-Slave
15	Ajax Map Comparis	31-oct-06	Yes	No	N/A	Google-Ma	Master	Microsoft	Master	Yahoo-Ma	Master	?	?	?	?	?	?	Master-Master
16	All of Ibiza from one	08-sep-07	Yes	No	N/A	BBC	Slave	Flickr	Slave	Google-M	Slave	?	?	?	?	?	?	Slave-Slave
17	alkemis local nyc	19-dic-05	Yes	No	N/A	Amazon-A9	Slave	del.icio.us	Slave	Flickr	Slave	Yahoo-Tra	Slave	Google-M	Master	?	?	Master-Slave
18	Top 99 Women on G	20-jul-06	Yes	Yes	Master	Google-Ma	Slave	Yahoo-Ge	Slave	YouTube	Slave	?	?	?	?	?	?	Master-Slave
19	Geowalk	06-jul-06	Yes	No	N/A	Flickr	Slave	GeoName	Slave	Google-M	Master	Wikipedia	Slave	?	?	?	?	Master-Slave
20	Facebook Friend M	16-ago-06	No	No	N/A	Facebook	Master	Google-M	Slave	?	?	?	?	?	?	?	?	Master-Master
21	Run London	03-mar-06	Yes	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
22	WeatherMole	05-may-06	Yes	No	N/A	Google-Ma	Master	NOAA-We	Slave	?	?	?	?	?	?	?	?	Master-Slave
23	Connecting Consum	02-dic-06	Yes	No	N/A	City-and-St	Slave	geocoder	Slave	Google-A	Slave	Google-A	Slave	Google-A	Slave	YouTube	Slave-Slave	
24	GMaps Flight Tracke	26-dic-06	Yes	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
25	Realtime Satellite T	22-may-06	Yes	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
26	CityRanks US Popul	03-ene-06	No	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
27	100 Most Powerful C	23-jul-07	Yes	Yes	Slave	Google-Ma	Slave	Yahoo-Ge	Slave	YouTube	Slave	?	?	?	?	?	?	Slave-Slave
28	2008 Basketball Tou	26-mar-08	Yes	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
29	2009 Formula One M	21-ene-09	Yes	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
30	25 Best Companies t	15-ene-07	Yes	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
31	29 Travels	24-nov-08	Yes	No	N/A	Google-Ma	Slave	Google-Ea	Slave	?	?	?	?	?	?	?	?	Slave-Slave
32	2Spaghi	26-mar-07	Yes	Yes	Slave	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
33	360 degree Sardinia	08-ago-07	Yes	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
34	4Hotels.us Hotel Ma	26-sep-06	Yes	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
35	5 TVs	13-mar-07	No	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
36	Access Denied Map	19-nov-07	No	Yes	Master	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Master-Slave
37	Zillow.com	30-abr-06	Yes	Yes	Slave	Microsoft-V	Master	?	?	?	?	?	?	?	?	?	?	Master-Slave
38	Zimride	15-abr-07	Yes	Yes	Master	Facebook	Master	Google-M	Slave	?	?	?	?	?	?	?	?	Master-Slave
39	Yoga Yoga Yoga	18-mar-08	Yes	No	N/A	eBay	Slave	Google-M	Slave	YouTube	Slave	?	?	?	?	?	?	Slave-Slave
40	Your Mapper	12-sep-09	Yes	Yes	Master	Bing-Maps	Slave	Caf-Press	Slave	Google-St	Slave	?	?	?	?	?	?	Master-Slave
41	YouTube Slideshow	06-jun-07	Yes	No	N/A	Google-Ma	Master	YouTube	Slave	?	?	?	?	?	?	?	?	Master-Slave
42	Zeeqa	22-oct-09	Yes	No	N/A	eBay	Master	Google-M	Slave	PriceRunn	Master	?	?	?	?	?	?	Master-Slave
43	Zillion Events	20-oct-08	Yes	No	N/A	Eventful	Master	Google-M	Slave	?	?	?	?	?	?	?	?	Master-Slave
44	Worlds Fastest Elev	24-jun-08	Yes	Yes	Slave	Google-Ma	Slave	YouTube	Slave	?	?	?	?	?	?	?	?	Slave-Slave
45	Yamusic	14-mar-10	Yes	No	N/A	Last.fm	Slave	Bandsintc	Slave	GeoName	Slave	?	?	?	?	?	?	Slave-Slave
46	World and Regional	13-nov-06	No	No	N/A	Google-Ma	Slave	?	?	?	?	?	?	?	?	?	?	Slave-Slave
47	World Port Source	29-mar-06	Yes	Yes	Slave	Google-Ad	Master	Google-M	Master	?	?	?	?	?	?	?	?	Master-Master
48	World Time Engine	29-feb-08	Yes	No	N/A	Google-Ma	Master	World-Tin	Master	?	?	?	?	?	?	?	?	Master-Master

5.4 Annex “D” Mashup quality survey

Please insert below your information. Data will only be used for the analysis of the use of mashup technology by practitioners and will not be shared with third parties.

For info:

Cinzia Cappiello, cappiell [at] elet.polimi.it



Your Age:

Please, rate your expertise as Web programmer:

- Low
- Mid
- High

Which is the most important information you lookup in ProgrammableWeb.com?

In which context do you generally use web APIs?

- Private/Fun
- Business

Others (please specify)

Which is the intended usage of the components you select?

- Composition of different components for a UI-provided mashup application
- Composition with other web services
- Use of a single component within a web page (e.g., your web page)
- Others (please specify)

Which are the factors that most influence your choice of an API besides the actual functional requirements (please, rate the following items with a value between 1 and 5; 1=poor and 5=high).

DOCUMENTATION	1	2	3	4	5
The availability of how-tos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The availability of web service descriptors (e.g.,WSDL, WADL)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The availability of examples	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The availability of forums and blogs	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

TECHNOLOGY	1	2	3	4	5
The support for your preferred programming language	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The support for your preferred programming protocols	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The use of standard data formats or protocols	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The adopted authentication model	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>


DATA QUALITY	1	2	3	4	5
The accuracy of the provided data set	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The freshness of the provided data set	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
The coverage of the provided data set	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

PERCEIVED QUALITY	1	2	3	4	5
The reputation of the API provider	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The diffusion of the API (e.g., number of mashups that embed it)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The usability and accessibility of the user interface	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please, rate the extent to which the following factors have impact on the API reliability?

	1	2	3	4	5
API age	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
API number of versions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Frequency of update	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please, rate the ease of use of the following data formats

	1	2	3	4	5
JSON	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
XML	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
ATOM, RSS, GData, etc.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Parameter-value (e.g., invoking a JavaScript function)		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please, indicate the data format that you usually prefer

Please, rate the ease of use of the following component/service types

	1	2	3	4	5
Restful	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SOAP- WSDL	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Javascript components	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PHP-PERL-ASP (and any other language for dynamic pages implementation)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please, rate the extent to which the following security mechanisms may prevent you from using an API

	1	2	3	4	5
SSL	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
API key	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Developer key	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
User account	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SSL plus authentication (API key, Developer key, or User account)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
No security	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5.5 Annex “E” Mashup quality survey, results

Sample of the survey results

AGE	Expertise as web	Most important information you lookup in	context you generally use web APIs	Intended usage of the components	Factors that most influence your choice of an API besides the actual functional requirements													Rate the extent to which the following factors have impact on the API reliability			Rate the ease following d			
					DOCUMENTATION				TECHNOLOGY				DATA QUALITY			PERCEIVED QUALITY		API age	API number of versions	Frequency of update	JSON	XML	AF	
					availability of how-tos	availability of web	availability of example	availability of forums	support for your preference	support for your preference	use of standard data	adopted authentication	accuracy of the provided	freshness of the provided	coverage of the provided	reputation of the API	diffusion of the API							usability and access
1	34	mid	the ranking of use of t	other	4	5	3	2	5	4	4	4	4	3	3	3	4	3	4	3	3	3	5	
2	27	high	Experience	business	compositionUI	5	1	3	3	5	3	3	4	5	5	4	5	3	4	5	3	4	4	5
3	29	high		business	single	3	4	5	4	4	4	4	3	3	3	4	3	4	3	4	3	3	3	4
4	26	low	API listing and descrip	other	compositionUI	5	2	5	4	4	4	2	2	4	5	5	5	5	5	4	2	3	3	3
5	27	mid		private	single	4	4	4	5	5	4	3	5	3	3	0	5	3	5	3	1	1	5	5
6	26	high		business	compositionUI	5	4	5	4	5	4	4	4	4	4	4	4	5	5	5	4	5	4	5
7	31	high		other	compositionW!	3	2	4	5	5	2	4	3	4	4	4	5	3	4	1	2	5	3	5
8	28	mid	DLL, example code	business	single	4	3	4	4	5	4	4	3	4	3	5	4	4	4	4	4	3	3	5
9	25	mid		private	single	2	1	3	4	1	2	4	3	1	3	3	3	3	1	3	2	2	4	3
10	29	mid	s	business	compositionW!	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	26	mid	API documents and sc	business	compositionW!	5	4	3	4	3	5	5	4	5	4	4	5	4	3	0	4	4	3	5
12	44	high	data and integration /	business	compositionW!	3	1	3	3	3	4	5	5	5	5	4	4	2	3	3	3	4	5	5
13	32	high	I dont use it per se, I c	other	other	5	1	4	1	3	3	5	5	5	5	5	1	1	1	5	1	5	5	3
14	29	mid		business	single	3	3	5	4	3	3	4	4	4	3	4	4	4	5	3	5	4	4	4
15	43	high	Mashups & Federate	business	compositionW!	3	4	4	2	4	3	5	3	4	3	3	4	3	3	3	3	4	5	5
16	45	high	news about mashups	business	compositionW!	4	4	4	4	5	4	4	3	3	3	3	2	4	4	2	3	4	4	5
17	28	mid	availability of differer	business	compositionUI	5	5	5	5	5	5	5	0	0	0	5	5	5	3	4	5	2	4	4
18	0	high		business	compositionUI	5	4	5	3	5	2	2	1	4	4	3	5	1	3	2	2	3	4	4
19	24	high	Templates, Patterns,	other	single	4	2	4	4	5	4	4	5	4	3	4	5	4	3	5	4	3	3	5
20	25	mid	Effectiveness	business	compositionW!	4	5	5	5	5	4	4	5	4	5	5	5	5	5	5	4	5	4	4
21	35	high	mashups	business	compositionW!	4	5	4	3	3	4	5	3	4	3	5	3	5	4	4	3	3	5	4
22	27			private	compositionUI	4	4	4	4	4	3	3	3	4	4	4	4	4	3	4	4	3	2	4
23	27	mid	the easiness in the AF	business	compositionW!	5	5	5	3	4	5	5	3	5	4	4	5	3	5	3	5	5	3	5
24	26	mid		other	other	3	4	4	3	4	4	3	3	3	3	3	4	3	4	3	3	4	1	4
25	25	high		business		5	2	5	5	5	3	4	2	4	4	4	4	2	5	2	3	4	4	4
26	31	high		business	compositionW!	5	4	5	4	5	4	3	3	4	4	5	3	5	5	3	5	4	3	5
27	30	high				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	23	mid	API	business	compositionW!	3	3	3	3	5	4	4	3	3	3	3	3	4	3	3	3	3	4	4

