Logging and Analyzing User's Interactions in Web Portals

Gennaro Costagliola, Filomena Ferrucci, Vittorio Fuccella, Luigi Zurolo

Dipartimento di Matematica e Informatica, Università di Salerno, Via Ponte Don Melillo, I-84084 Fisciano (SA)
{gcostagliola, fferrucci, vfuccella}@unisa.it, zurololuigi@gmail.com

Abstract. Content Management Systems and Web Portal Frameworks are more and more widely adopted in Web development. Those kinds of software often produce Web pages whose layout is divided in sections called, in the case of Web Portals, "portlets". Portlets can be produced by different sources and then aggregated in the same page by the portal. For Web portals, traditional Web metrics based on page visits can be inadequate for fully understanding user's interest, due to the heterogeneity of content and the variety of sources. This paper proposes a system for evaluating the Web traffic at a deeper level than the page visit one: the level of the sections, or of the portlets. The interest of the user in the sections of the page is gauged through implicit interest indicators, such as, section visibility, mouse movements and other client-side interactions. Our system is composed of two different products: a framework that, opportunely instantiated in a Web portal, allows the production of a log, and a log analyzer. The possible uses and benefits gained by research in the fields of Web traffic analysis, portal design and usability are investigated in depth.

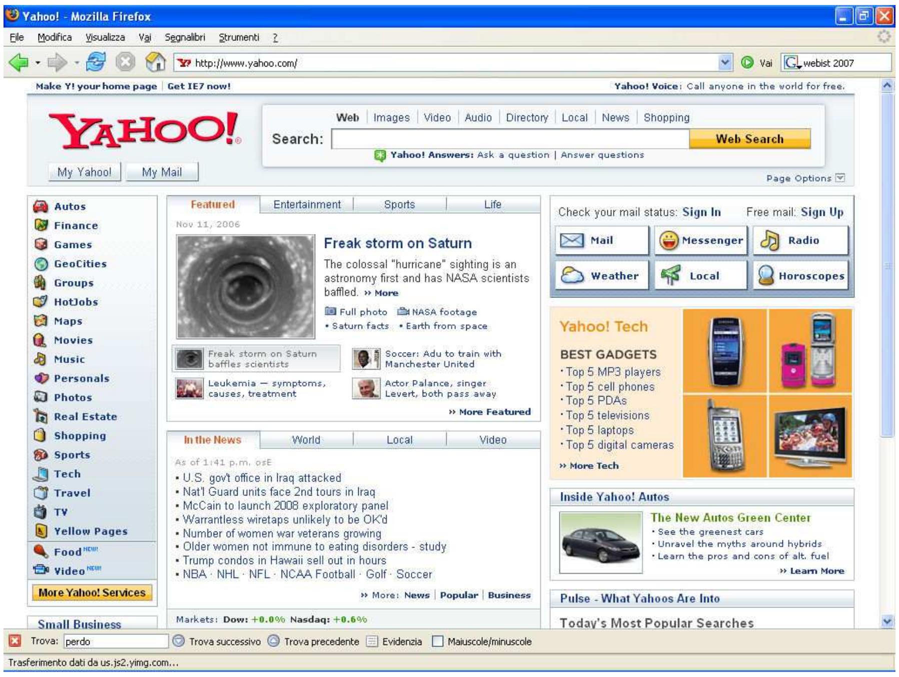
Keywords: Web, www, metrics, portal, portlet, JSR 168, WSRP, logging, log, query, log4p

1 Introduction

Content Management Systems (*CMS*, in the sequel) and *Web Portal Frameworks* are more and more widely adopted in the development of Web sites, mainly due to their characteristic of allowing the Web designers to rapidly develop a Web site and the portal administrators to rapidly update its contents.

CMS and portal frameworks produce Web pages whose layout is divided in sections called *portlets*. This division is not only a layout concern, but it occurs in all the steps of the generation of the pages: in the case of many portals, the *portlets* can be produced by different, eventually remote, sources and then aggregated in the same page. Thus, we are on the way towards the creation of a *portlet* market, where content, or part of it, is produced by third parties and then shown on the publisher's Web site.

The technical solution to achieve this organization, is based on the production of *markup fragments* (a concern of the *portlet*), and their aggregation in a single page (a concern of the portal). The development of standards and specifications, such as *JSR 168* [17] and *WSRP* [24], has helped to this extent. A noteworthy example of a Web site, whose layout demonstrates the use of a portal framework in its development, is that of *Yahoo*. A screenshot of its home page is shown in figure 1.


Yahoo! - Mozilla Firefox
File Modifica Visualizza Vai Segnalibri Strumenti ?
http://www.yahoo.com/
Vai
webist 2007
Make Y! your home page | Get IE7 now!
Yahoo! Voice: Call anyone in the world for free.
YAHOO!
Web | Images | Video | Audio | Directory | Local | News | Shopping
Search:
Web Search
Yahoo! Answers: Ask a question | Answer questions
My Yahoo! | My Mail
Page Options
Autos
Finance
Games
GeoCities
Groups
HotJobs
Maps
Movies
Music
Personals
Photos
Real Estate
Shopping
Sports
Tech
Travel
TV
Yellow Pages
Food NEW!
Video NEW!
More Yahoo! Services
Small Business
Featured | Entertainment | Sports | Life
Nov 11, 2006
Freak storm on Saturn
Full photo | NASA footage
• Saturn facts • Earth from space
» More Featured
In the News | World | Local | Video
As of 1:41 p.m. PST
» More: News | Popular | Business
Markets: Dow: +0.0% Nasdaq: +0.6%
Check your mail status: Sign In
Free mail: Sign Up
Mail | Messenger | Radio
Weather | Local | Horoscopes
Yahoo! Tech
BEST GADGETS
• Top 5 MP3 players
• Top 5 cell phones
• Top 5 PDAs
• Top 5 televisions
• Top 5 laptops
• Top 5 digital cameras
» More Tech
Inside Yahoo! Autos
The New Autos Green Center
• See the greenest cars
• Unravel the myths around hybrids
• Learn the pros and cons of alt. fuel
» Learn More
Pulse - What Yahoos Are Into
Today's Most Popular Searches
Trova: perdo
Trova successivo | Trova precedente | Evidenzia | Maiuscole/minuscole
Trasferimento dati da us.js2.yimg.com...

The colossal "hurricane" sighting is an astronomy first and has NASA scientists baffled. » More

Freak storm on Saturn baffles scientists

Soccer: Adu to train with Manchester United

Leukemia — symptoms, causes, treatment

Actor Palance, singer Levert, both pass away

• U.S. gov't office in Iraq attacked
• Nat'l Guard units face 2nd tours in Iraq
• McCain to launch 2008 exploratory panel
• Warrantless wiretaps unlikely to be OK'd
• Number of women war veterans growing
• Older women not immune to eating disorders - study
• Trump condos in Hawaii sell out in hours
• NBA · NHL · NFL · NCAA Football · Golf · Soccer

Fig. 1. The Yahoo home page and its sections.

As a result of the aggregation, a portal page can contain a highly heterogeneous content, taken from various portlet producers. For Web sites developed with CMS and portal framework technology, traditional metrics based on page visits can be inadequate to fully understand user's interest: new forms of metrics are needed. What we need are metrics which can give information at a deeper level than that of page visits: the level of the sections, or of the *portlets*. Unfortunately, at present there is a lack of these metrics.

This paper presents some tools which make use of new metrics suitable for describing the behaviour of the visitors on the portal pages. The gathering of such information is carried out through the use of a framework directly instantiated in the portal. The framework produces an XML log, which includes raw data, such as the implicit interest indicators (i. e. any interaction of the user with the *portlets*) and the visibility of the *portlets* in the pages, captured when the users browse the portal pages.

The logs are analyzed through a suitable log analyzer that obtains some higher level information by performing several queries on the logs, such as:

- An estimation of the visibility of the *portlets* in the page.
- The interactivity level of the *portlets*.

From this information we can obtain an estimation of the interest shown by the users in the *portlets*. The analysis can be made available per single user visit and session, across multiple visits of the same user or for all the users.

Once obtained, these data can be used for multiple purposes. The possible uses and benefits gained by research in fields of Web traffic analysis, portal design and Web usability are investigated in depth. Furthermore, the same principle of logging user's interactions with a Web-based interface and analyzing the gathered data in order to obtain interesting information about user's behavior is applicable in other circumstances: our approach has been exploited for analyzing the strategies used by the learners for executing on-line educational tests in an e-learning system.

The rest of the paper is organized as follows: section 2 gives the knowledge background necessary to understand some concepts on which the system is based. The system is presented in section 3: the section is composed in two sub-sections, the first to describe the logging framework and the second for the log analyzer. Lastly, in section 4, we briefly discuss possible uses and benefits of our system. Several final remarks and a discussion on future work conclude the paper.

2 Background

A portal is a Web site which constitutes a starting point, a gate to a consistent group of resources and services on the Internet or in an intranet. Most of the portals were born as Internet directories (as *Yahoo*) and/or as search engines (as *Excite*, *Lycos*, etc.). The offer of services has spread in order to increment the number of users and the time they spend browsing the site. These services, which often require user registration, include free email, chat rooms, and personalization procedures. In the history of portals, many authors identify two generations. Second generation Web portals distinguish themselves from first generation ones for their architecture, which is component-oriented. In particular, the basic component constituting them, is often referred to as *portlet*. The portal is responsible for aggregating information coming from different sources, local or remote, available in the form of *mark-up fragments*. Each of these *fragments* is produced by a *portlet*. In the context of Web portals, the possibility of deploying a *portlet* in any portal is particularly significant. To this extent, that is, to achieve interoperability among portals, it has been necessary to define a standard way to develop and deploy *portlets*. Two main standards have been defined and widely adopted by producers: the *Web Services for Remote Portlets (WSRP)* and the *Java Portlet Specification and API (JSR 168)*. The former is more oriented to the definition of rules about the use of remote *portlets*, the latter is focused on the definition of interfaces for the development of *portlets* which can run in Java-based portals.

WSRP defines a Web service interface through which portals can interact with the remote producer's *portlets*. The *WSRP 1.0* specification was approved as an *OASIS* standard in August, 2003. Being based on Web services, several interfaces to adopt the standard have been developed for the most used technologies (e.g. *J2EE*, *.NET*, and so on).

Most of the Java technologies, part of the *Java 2 Enterprise Edition*, the platform for the development and deployment of distributed enterprise applications, follow a consolidated architectural model, called *container/component* architecture. This model offers the chance to develop components and deploy them on different containers. Both component and containers compliant to specifications can be developed independently and commercialized by different software vendors, thus creating a market economy on Java software. Furthermore, several good-quality Open Source products compete with them. The *JSR 168* follows the *container/component* model and its adoption has grown until it has become an important reference-point which cannot be excluded from the projects aimed at the development of Web portals. Among other things, it defines the architectural model of conformant portals. Its main constituents are the *portlet*, which produces content mark-up, the *portal*, which aggregates the mark-up, and the *portlet container*, which manages the portlet lifecycle and provides an *API* to the *portlets* for accessing to a set of services. The typical architecture for a *JSR 168* conformant Web portal is shown in figure 2.

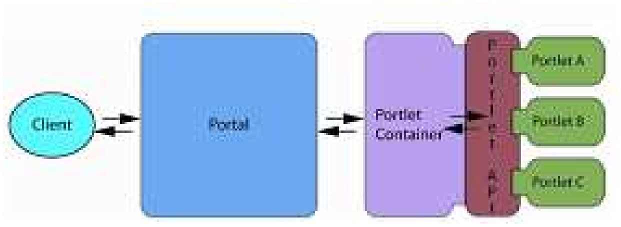


Fig. 2. JSR 168 compliant portal architecture.

3 The System

Our system is aimed at obtaining detailed statistics in order to have a deep analysis of the user's interest in the Web portal and, in particular, in the sections which compose its pages.

Our system is composed of two different pieces of software:

1. A *logging framework*, called *Log4p*, to be used by Web portal developers, which, instantiated in the Web portal, is in charge of capturing information about user's behavior during the navigation of the portal and storing it in an XML-formatted log.
2. A *log analyzer*, to be used by Web portal administrators, developed as a stand-alone application, which is responsible for analyzing the data gathered by the logger.

3.1 Log4p: the Logging Framework

The purpose of the *Logging Framework* is to gather all of the user actions during the browsing of the portal and to store raw information in a set of log files in an XML format.

The framework is composed of a server-side and a client-side module. The client-side module is responsible for "being aware" of the behavior of the user while he/she is browsing the portal pages. The server-side module receives the data from the client and creates and stores log files on the disk.

Despite the required interactivity level, due to the availability of *AJAX* (Asynchronous JavaScript and XML), the new technology for increasing the interactivity of Web content, it has been possible to implement the client-side module of our framework without developing plug-in or external modules for Web browsers. Javascript has been used on the client to capture user interactions and the text-based communication between the client and the server has been implemented through *AJAX* method calls. The client-side scripts can be added to the portal pages with a light effort by the programmer.

The events captured by the framework are the following:

- Actions undertaken on the browser window (opening, closing, resizing)
- Actions undertaken in the browser client area (key pressed, scrolling, mouse movements)

The event data is gathered on the browser and sent to the server at regular intervals. It is worth noting that the event capturing does not prevent other scripts present in the page to run properly.

The server-side module has been implemented as a Java servlet which receives the data from the client and prepares an XML document in memory. At the end of the user session the XML document is written to the disk. To reduce the size of log files, a new file is used every day.

The information model used for the log data is shown in figure 3. The model is organized per user session. At this level, an identifier (if available) and the IP of the user are logged as well as agent information (browser type, version and operating system). A *session* is composed of page visits data. For every page, the referrer is logged and a list of events is present. The data about the user interactions are the following:

- Event type
- HTML source object involved in the event
- Portlet containing the HTML object and its position in the page (coordinates of the corners)
- Mouse coordinates
- Timing information (timestamp of the event)
- More information specific of the event

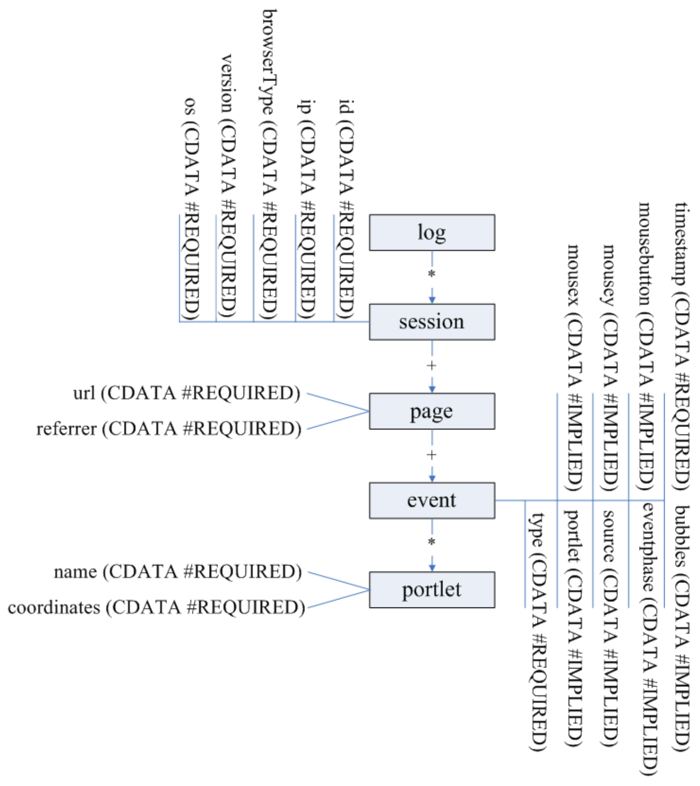


Fig. 3. The information model for log data.

An important concern in Web metrics is the log size. In very crowded Web sites, even simple HTTP request Web logs can reach big sizes. A configuration system, including the following configuration settings has been conceived above all to reduce log sizes:

- List of events to capture;
- List of *portlets* to monitor;
- Time interval between two data transmissions from the client to the server;
- Sensitivity for mouse movements;
- Sampling factor (logging is activated only for a random user out of n users).

The configuration is read by the server-side module but affects the generation of the javascript modules run on the client-side. The architecture of the framework is graphically represented in figure 4.

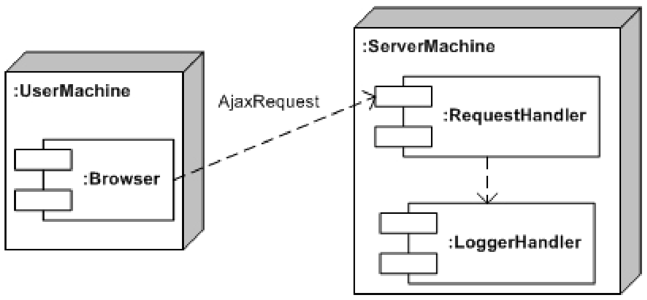


Fig. 4. Logging framework architecture.

On the client machine, everything can be done in the Web Browser. The Javascript modules for event capturing, dynamically generated on the server, are downloaded and run in the browser interpreter. Data is sent to the server through an *AJAX* request. On the server-side, a module called *RequestHandler* receives it. Once received, a module called *LoggerHandler* organizes the *XML* document in memory and flushes it to the disk every time a user session finishes.

3.2 The Log Analyzer

The next phase of the data gathering is the data analysis. In our system this is done through a Web-based stand-alone application, optionally hosted on the same server of the logging framework, which takes in input the log files.

The analysis phase consists of a series of analysis on the behavior of the user, starting from the data stored in the log. The analysis can have several aims. Among them we can cite:

- Giving a better organization to the portal;
- The choice of the contents more suitable to the user or to group of users;
- Usability analysis of the portal.

A deeper analysis on the uses that our system can offer is contained in the next section.

The analyzer performs queries on the logs to obtain the desired data and then calculates statistical indicators, shown in the form of charts and tables. Since the log files are in XML, the query engine has been developed to understand *XQuery* [25] language, and has been carried out by using an implementation of the *JSR 225* [16]. In the next sub-sections we will show some useful statistics we can obtain using the system. The module for drawing charts has been developed using a free Java library, named *jCharts* [13].

3.2.1 PageCounts

A generic analysis is given by the simple count of page visits. Even though such a task is easily performed by a lot of already existing tools, page visits count is

an important statistic for our system, since it allows us to understand on which page the interest of the user is mostly concentrated. Starting from this data, we could focalize our attention on a subset of the portal pages and perform a deeper, *portlet*-based, analysis on them. This statistic is easily obtainable with our analyzer through a *count* query on the *page* elements of our logs.
A sample of page visits chart, drawn using our analyzer, is shown in figure 5.

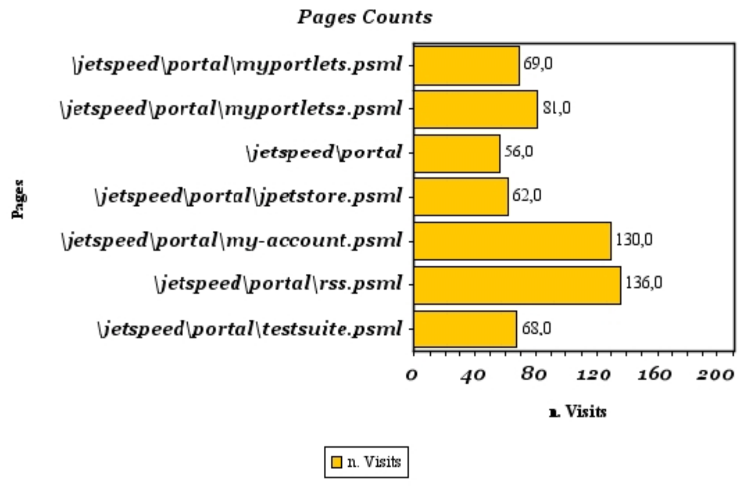


Fig. 5. Page count chart sample.

3.2.2 Portlet Visibility Time

A more specific analysis can be obtained by calculating the visibility time for each level of *portlet* visibility (total, partial, invisible). Portal layouts are usually organized per columns. A commonly used layout organizes *portlets* in two columns of the same size (50%, 50%). Another common layout is composed of three columns (i.e. 25%, 50%, 25% in size). The portal page can contain some other elements, such as, a header, a footer and an horizontal or a vertical menu or both of them. If the number and size of *portlets* is such that the portal page exceeds the size of the browser window, only a part of the page is shown, while some other parts are hidden and can be shown through scrolling. Thus, at any time some *portlets* can have full visibility, some others partial visibility (only a percentage of the portlet area is visible), while the remaining are completely hidden to the user.

Every time a scrolling event occurs, our logger records its timestamp and the position of the portlet in the page. The availability of this data allows us to precisely calculate the amount of time the *portlets* were fully visible, partially visible or completely invisible during the visit.

This information, in the context of a portal, is very useful, since, after knowing which page has attracted the user more, it let us know his/her interest in the single content units of the page. Figure 6 contains a chart showing the visibility percentage of the portlet.

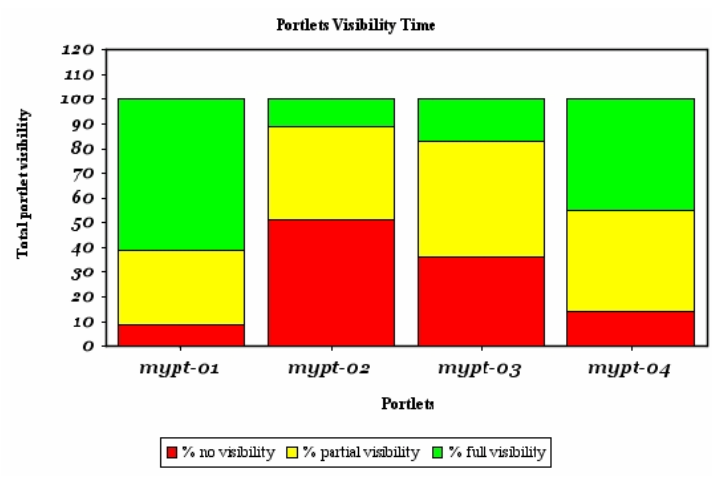


Fig. 6. Portlet visibility chart sample.

When a portlet is partially visible, the chart of figure 6 does not exactly tell us the extent of the visible and hidden areas of the portlet. Thus, in order to complete the visibility analysis, we considered it opportune to show another chart summarizing the visibility percentage of the portlet across users' page views. This indicator is calculated as the weighted mean of all the visibility times, using the following formula:

$$V = \frac{\sum_i (t_i * v_i)}{T} \tag{1}$$

V is the total visibility indicator for the portlet, t_i and v_i are, respectively, its time and percentage of visibility in the i-th interval, T is the total time of the page visit.
A sample of the chart is shown in figure 7. For the sake of readability, the bars are shown in green for high visibility, in yellow for low visibility and in red for scarce visibility. The elaboration and the queries performed on the log for obtaining the parameters in (1) , have been reported in appendix.

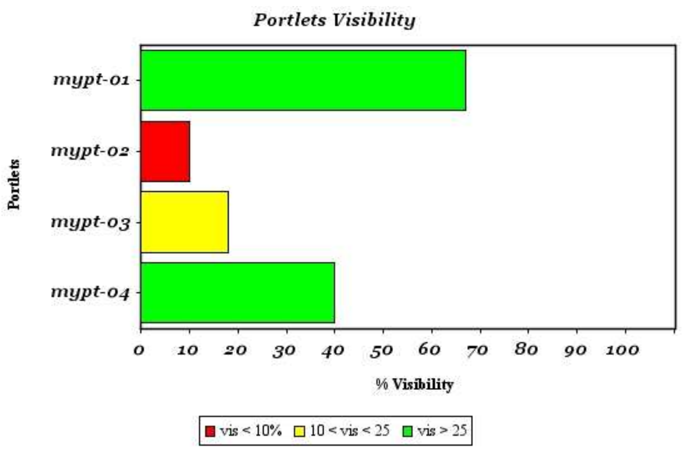


Fig. 7. Portlet visibility percentage chart sample.

3.2.3 Portlet Interactions

Some *portlets* can be more informative while others can be more interactive. For example, an article of an on-line news magazine is supposed to be informative, while a section containing a form should be more interactive, that is, it should receive more user interactions than the former. Many people use the mouse as a pointer while reading on-line news.

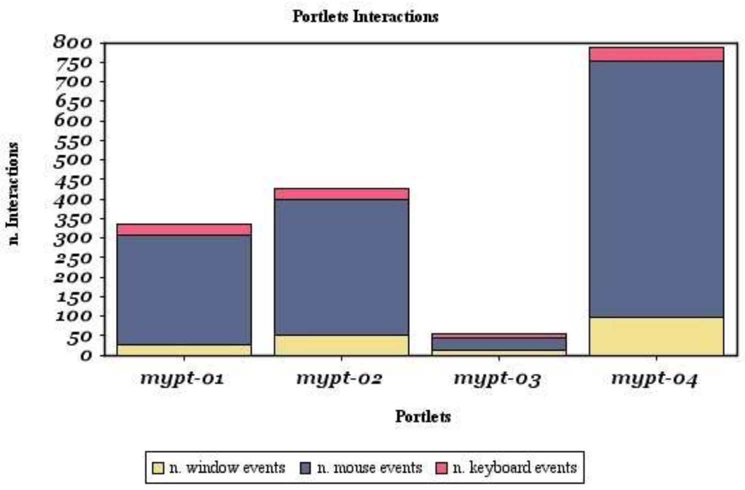


Fig. 8. Portlet interactivity level chart sample.

With our tool, we can obtain an information about portlet interaction from a bar chart. An example is shown in figure 8. In the chart, each bar is composed of three sections of different color. They represent three different types of interactions: window, mouse and keyboard events.
Another interest indicator to be considered is the total time a portlet has the mouse pointer in it. An eye tracking study [7] shows that there is a significant correlation between the eye movements and the mouse movements: tracking the trajectory drawn by the mouse pointer could be useful for obtaining the probable trajectory of user's eyes, that is, what the user is interested in.

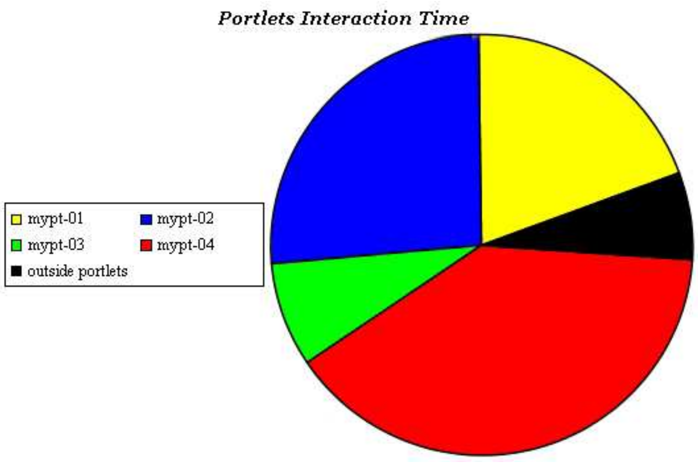


Fig. 9. Portlet mouse pointer focus.

While eye tracking cannot be performed, if not in ad hoc equipped laboratories, mouse tracking can be easily performed by our tool. All of the mouse movements can be reconstructed from the log analysis.
With a great number of users, the reconstruction of all mouse movements can be too onerous. A similar interest indicator can be obtained by just calculating the amount of mouse movements and the amount of time spent by the user with the mouse pointer inside a portlet. The number of movements has already been described and charted in figure 8. As for the amount of time the portlet has the mouse pointer in it, a pie chart, showing the times of presence of the mouse pointer inside the *portlets* of the same page, appears as the most appropriate choice. A sample is shown in figure 9.

4 Related Work, Uses and Applications

The effectiveness of implicit interest indicators is witnessed by several studies in literature [5; 20]. These works demonstrate the correlation between the implicit indicators and the actual interest of the user. Furthermore, such studies produce a list of the most used interest indicators. Some works propose the development of tools for determining user's interests. For example, in [1] a proxy server based system is used to capture client-side interactions. Other portal framework, such as *Websphere Portal Server* [22], also analyze the user behavior, but their analysis is based on classical clickstream metrics.

Some works are aimed at understanding the structure of Web pages in order to determine page sections. Many algorithms have been presented to this extent. The purposes of determining page sections include the detection of similarities among pages, the adaptation of the pages to small screens, the detection of the most significant content of the page, etc.

Wenyn et al. [23] divide pages in sections to detect similarities between two pages, in order to prevent phishing. Chen et al. [6] do the same thing in order to better view Web pages on small screen devices. Blocks in the pages are detected for identifying the informative sections of the page to reduce storing sizes for search engines [10] or to eliminate redundant information for Web mining [21]. It is clear that, if efforts have been made to divide pages in sections, where the pages are explicitly divided in sections, we can pursue the above discussed objectives and do much more. Once determined, the indication of user interest, calculated from the log analysis, can be used for several purposes. The next sub-sections analyze the possible practical uses of our system in several Web research fields.

4.1 Integrating Web Metrics

Web metrics tell us how the users are using a Web site. E-commerce sites need to know this information in order to improve their selling capacity. Some of the most commonly used Web metrics are: the number of page visits, the number of banner or link clicks, the percentage of users who complete an action, etc.

Unfortunately, clickstream analysis metrics have the following limitations, as remarked by Weischedel and Huizingh [22]:

- They report activity on the server and not user activity;
- They can overestimate the actual use of Web sites due to spiders activity;
- They do not include the real time spent on the page by the user.

Our system overcomes these problems, in fact, it captures client side activity, can easily recognize spiders from the absence of mouse movements and records times, in such a way that it is easy to detect inactivity time due to user absence from the screen.

4.2 Portal Design

The location of the *portlets* in the portal page has a great importance, since some *portlets* can have more visibility than others. An eye tracking study [12], analyzing

the behaviour of users in some browsing tasks, has shown that user's interest is more concentrated, at least in the initial phases of page browsing, in the *portlets* placed on the top of the first column on the left. In the same study, a complete classification of the places which are candidates to gain more user interest has been performed. It is advisable that, if the portal holder wants to emphasize the content of a portlet more than another, he/she should put these *portlets* in those places.
Our tool can help in determining the *portlets* which attract user's interest more and, on the basis of this data, it can help portal administrator in placing the *portlets* in the pages.

4.3 Personalization of the Portal

Web portal customization is often used to tailor the services of the portal to a single user or to a group of users. In some cases, the user has the freedom of choosing his/her favourite *portlets* to place in his/her home page. In other cases, the interest of the user can be inferred from the logs, and the pages of the portal constructed in order to give more visibility to content which matches user's interests. Our system can help in the latter case. Moreover, in the case of groups of users, groups can be obtained through clustering procedures. Our system can be useful for gathering data to obtain cluster of users.

4.4 Portal and Portlet Usability

Since *portlets* can be considered small Web applications, the definition of usability can be extended for *portlet usability*. Diaz et al. [11] define it as the capability of a *portlet* to be understood, learned or used under specified conditions. The implicit interest indicators can be used to facilitate the task of usability evaluators. Atterer et al. [1] show many situations in which this is true, for example by using true users instead of volunteers in the lab. Furthermore, a study [19] states the possibility of performing a choice among different *portlets* with similar features, choosing on the basis of their usability. Our tool can be useful to this extent, in order to isolate the interactions relative to a given portlet thus evaluating its usability.

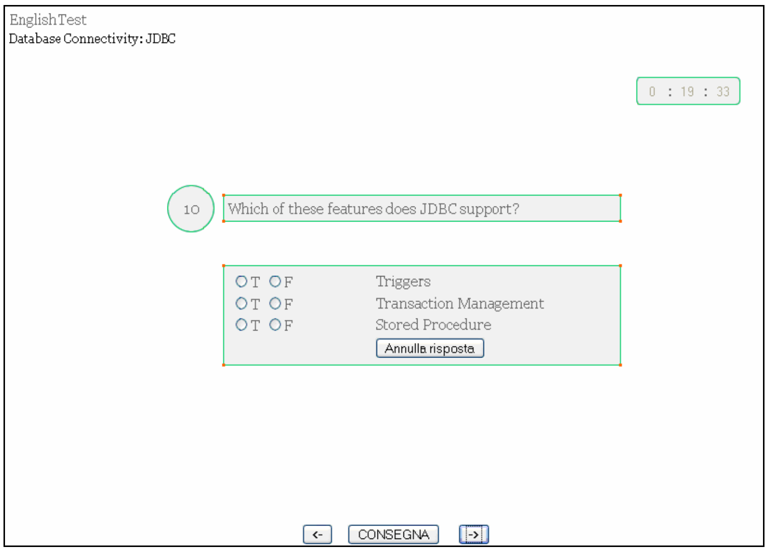
The position of the *portlets* affects the usability of the portal. For example, let us suppose that a task can be performed by interacting with more than one *portlet*. The position of the *portlets* involved in the task can affect the amount of time necessary to perform the task itself. A usability study can be aimed at finding the best location for each of these *portlets*. Our system allows the usability evaluators to record and evaluate the action of the users in all of the different portlet arrangements.

Lastly, our tool captures key press events. In the case of *portlets* with forms, the data can be used to understand if the user had problems in filling the form. This is valid for any kind of Web site and not only for portals.

4.5 Logging and Analyzing Learner Behavior in On-Line Testing

An approach similar to the one described so far has been used in on-line testing in order to log and analyze learner behavior in on-line testing. *On-Line Testing*, also known as *Computer Assisted Assessment* (*CAA*), is a sector of e-learning aimed at assessing learner's knowledge through e-learning means. Tracking learner's interactions during the execution of a test can be useful for understanding the strategy used by the learner to complete the test and for giving him/her advise on how to perform better in future tests. Several experiments have been performed to this aim [2; 15; 18] in the past. Our approach for on-line testing is rather similar to the one used for Web portals: learner's interactions during tests based on *multiple choice* questions have been logged and stored in XML files, then the information gathered is analyzed and visualized in a suitable chart.

The interactions have been logged by instantiating a slightly modified version of *Log4p* in an on-line testing system, called *eWorkbook* [8]. This system presents the questions, one at a time, in a Web page. A screenshot of the test interface is shown in figure10. The learner can browse the test questions by clicking on the *next* and *previous* buttons in the bottom of the page. The page is composed of different sections (in analogy to *portlets*) showing, respectively, the *stem* and the *options* of the question. The framework records the interactions of the user with the above described interface, including response and browsing events and mouse events.


EnglishTest
Database Connectivity: JDBC

Which of these features does JDBC support?

○T ○F	Triggers
○T ○F	Transaction Management
○T ○F	Stored Procedure

Fig. 10. A Screenshot of the Test Execution.

A chronological review of the test has been made available through a chart, obtained by showing the salient points of a test execution, synthesized in the interactions recorded in the log file. This chart shows, at any time, the item browsed by the learner, the mouse position (intended as the presence of the mouse pointer on the stem or on one of the options) and the presence of response type interactions, correct or incorrect. The chart is two-dimensional: the horizontal axis reports a continuous measure, the time, while the vertical axis displays categories, the progressive number of the item currently viewed by the learner. The test execution is represented through a broken line. The view of an item for a determined duration, is shown through a segment drawn from the point corresponding to the start time of the view to the one corresponding to its end. Consequently, the length of the segment is proportional to the duration of the visualization of the corresponding item. A vertical segment represents a browsing event. A segment oriented towards the bottom of the chart represents a backward event, that is, the learner has pressed the button to view the previous item. A segment oriented towards the top is a forward event.

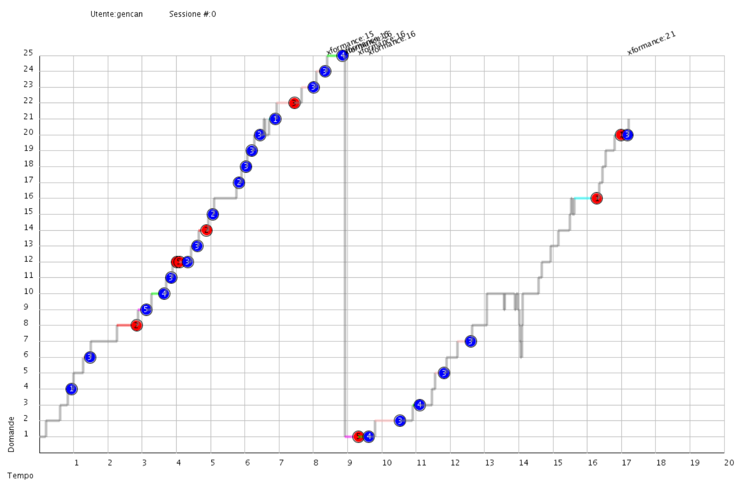


Fig. 11. Graphical Chronological Review of a Sample Test.

The responses given by a learner on an item are represented through circles. The progressive number of the chosen option is printed inside the circle. The indication of correct/incorrect response is given by the filling color of the circle: a blue circle represents a correct response, while an incorrect response is represented through a red circle. The color is also used for representing the position of the mouse pointer during the item view. The presence of the mouse pointer in the *stem* area is represented through a black color for the line. As for the options areas, the red, yellow, green, blue and purple colors have been used, respectively, for 1 to 5 numbered options. More than 5 options are not supported at present. Lastly, grey is used to report the presence of the mouse pointer in a neutral zone. The graphical chronological review of a sample test is shown in figure 11.

By analyzing the charts obtained in an experiment carried out during a laboratory exam involving approximately 80 learners, we can conclude that learners often follow common strategies for completing on-line tests. In our experiment we have identified the following three different strategies:

- **Single Phase**. This strategy is composed of just one *phase* (a part of the test execution needed by the learner for sequentially browsing all of the questions). The time available to complete the test is organized by the learner in order to browse all the questions just once. The learner tries to reason upon a question for an adequate time and then gives a response in almost all cases, since he/she knows that there will not be a revision for the questions. Eventual *phases* subsequent to the first one have a negligible duration and no responses.
- **Active Revising**. This strategy is composed of two or more *phases*. The learner intentionally browses all the questions in a shorter time than the time available, in order to leave some time for revising *phases*. The questions whose answer is uncertain are skipped and the response is left to subsequent *phases*. As a general rule, the first *phase* lasts a longer time and the subsequent *phases* have decreasing durations.
- **Passive Revising**. This strategy is composed of two or more *phases*. The learner browses and answers all the questions as fast as possible. The remaining time is used for one or more revising *phases*. As a general rule, the first *phase* lasts a longer time and the subsequent *phases* have decreasing durations.

For both the definition of the strategies and the classification of test instances, the charts have been visually analyzed by a human operator. The above tasks are rather difficult to perform automatically, while a trained human operator can establish the strategy used by the learner from a visual inspection of the charts of the test instances and giving advice to the learners on how to perform better next time.

Other uses of the above described method are the detection of correlation among questions and the detection of cheating during tests. The reader can refer to a separate paper [9] for obtaining a more detailed description on the educational results of our experiment.

5 Conclusions

In this paper we have presented a system aimed at obtaining and analyzing the data about the behaviour of the users of Web portals. The system overcomes the limitation of the simple page visit-based metrics, giving more valuable information related to the *portlets*, such as their visibility and interactivity and, consequently, the interest of the user in them.

The system is composed of two components: a framework for obtaining XML-based log files and an application for log analysis. Several charts, drawn using the analyzed data have been shown.

Referencing some recent work in literature, we have argued that our system can be useful for numerous purposes, such as, integrating Web metrics, optimizing portlet layout both for all users and for personalization, studying usability of portals and, in a slightly modified version, analyzing learner behavior in on-line testing. Future work is aimed at further demonstrating the use of our system in some of these fields.

Finally, an aspect that has been considered, but not yet put into practice, is the availability of the log data both to the portal and to the portlet producer. At present some architecture and secure schemas have been taken into account, as the one proposed by Blundo and Cimato [4], applied for determining banner clicks in advertising campaigns.

The system has been tested on a portal developed with Apache Jetspeed II [14] Portal framework.

References

1. Atterer, R., Wnuk, M., Schmidt, A.,: Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In Proceedings of the 15th international conference on World Wide Web WWW '06. ACM Press (2006)
2. Bath, J.A.: Answer-changing Behaviour on objective examinations. The Journal of Educational Research, 61, pp. 105-107, (1967).
3. Bellas, F.,: Standards for Second-Generation Portals. IEEE Internet Computing. 8(2): pp. 54-60 (2004)
4. Blundo, C. and Cimato, S.,: A Software Infrastructure for Authenticated Web Metering. IEEE Computer (2004)
5. Claypool, M., Le, P., Wased, M., Brown, D.,: Implicit interest indicators. In Proceedings of the 6th international conference on Intelligent user interfaces. ACM Press (2001)
6. Chen, Y., Xie, X., Ma, W. Y., Zhang, H. J.,: Adapting Web pages for small-screen devices. IEEE Internet Computing (2005)
7. Chen, M. C., Anderson, J. R., Sohn Moore, M. H.,: What can a mouse cursor tell us more?: correlation of eye/mouse movements on Web browsing. In CHI '01 extended abstracts on Human factors in comp. syst. (2001)
8. Costagliola G., Ferrucci F., Fuccella V., Oliveto R.: eWorkbook: a Computer Aided Assessment System. International Journal of Distance Education Technology. 5 (3). Pp 24-41. (2007).
9. Costagliola G., Fuccella V., Giordano M., Polese G.: A Web-Based E-Testing System Supporting Test Quality Improvement. In: Proceedings of The 6th International Conference on Web-based Learning. Pp 272 – 279. (2007).
10. Debnath, S., Mitra, P., Pal, N., Giles, C.L.,: Automatic identification of informative sections of Web pages. In IEEE Transactions on Knowledge and Data Engineering (2005)
11. Diaz, O., Calero C., Piattini M., Irastorza A.,: Portlet usability model. IBM Research Report. RA221(W0411-084). ICSOC 2004.pp. 11-15 (2004)
12. Goldberg, J. H., Stimson M. J., Lewenstein, M., Scott, N., Wichansky, A. M.,: Eye tracking in Web search tasks: design implications. In Proceedings of the 2002 symposium on Eye tracking research & applications (2002)
13. jCharts, Krysalis Community Project – jCharts. http://jcharts.sourceforge.net/ (2006)
14. Jetspeed 2, Apache Group. Jetspeed 2 Enterprise Portal. http://portals.apache.org/jetspeed-2/ (2006)
15. Johnston, J.J:: Exam Taking speed and grades. Teaching of Psychology, 4, pp. 148--149 (1977)
16. JSR 225, JSR 225: XQuery API for JavaTM (XQJ). http://jcp.org/en/jsr/detail?id=225 (2006)
17. JSR 168, JSR-000168 Portlet Specification. http://jcp.org/aboutJava/communityprocess/review/jsr168/ (2003)

18. McClain, L.: Behavior during examinations: A comparison of “A”, “C” and “F” students. Teaching of Psychology, 10 (2). (1983)
19. Moraga, M. A., Calero, C., Piattini, M.,: Ontology driven definition of a usability model for second generation portals. In Workshop proceedings of the sixth int. conference on Web engineering, ICWE’06 (2006)
20. Shapira, B., Taieb-Maimon, M., Moskowitz, A.,: Study of the usefulness of known and new implicit indicators and their optimal combination for accurate inference of users interests. In Proceedings of the 2006 ACM symposium on Applied computing SAC '06 (2006)
21. Taib, S.M., Yeom, S. J., Kang, B. H.,: Elimination of Redundant Information for Web Data Mining. In Proceedings of ITCC 2005, Int. Conf. on Information Technology: Coding and Computing, Vol 1 (2005)
22. Websphere. IBM WebSphere Portal Server. http://www-306.ibm.com/software/genservers/ portal/server/index.html?S_TACT=103BEW01&S_CMP=campaign
23. Weischedel, B., Huizingh, E. K. R. E.,: Website Optimization with Web Metrics: A Case Study. In Proceedings of ICEC '06, the 8th international conference on Electronic commerce. ACM Press (2006)
24. Wenyin, L., Huang, G., Xiaoyue, L., Deng, X., Min Z.,: Phishing Web page detection, In Proceedings of Eighth International Conference on Document Analysis and Recognition (2005)
25. WSRP, OASIS Web Services for Remote Portlets. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp (2003)
26. XQuery, XQuery 1.0: An XML Query Language W3C Candidate Recommendation. http://www.w3.org/TR/xquery/ (2006)

Appendix

As an example, the pseudo-code procedure used by the log analyzer for obtaining the portlet visibility percentage chart, follows. Each bar in the chart represents the percentage of visibility of a portlet across all page views. Those values are calculated using (1). The procedure assume, simplistically, that all of the analyzed pages contain the same *portlets*. Their number is passed as a parameter to the procedure (line 1), and is used to associate the correct event timestamp (lines 16-18) to portlet data (name and coordinates).

Every time a user scrolls the page, the percentage of visibility of a portlet changes, since part or all of its area can fall inside/outside the browser’s client area. On the initial page *load* event, and on every *scroll* event, our log records the coordinates of each *portlet* (through the *portlet* element of the information model, see figure 3). Through our sample code, for each *event* element and for each *portlet* element, *portlet* names, coordinates and event timestamps are obtained by querying the log and by storing the results in the *portletNames*, *coordinates* and *timestamps* vectors, rispectively (lines 3-7).

```
1  procedure showVisibilityChart(portletNum)
2
3     portletNames = let $x := //portlet return $x/@name;
4     coordinates = let $x := //portlet return
5           $x/@coordinates;
6     timestamps = for $x in //event where $x/@type="load"
```

```
7           or $x/@type="scroll" return $x/@timestamp;
8
9     old_time = timestamps[0];
10    T = 0;
11
12    for(j=0; j < portletNames.length; j++)
13
14       v = calculateVisibilityPercentage(coordinates[j]);
15
16       if (j % portletNum == 0)
17          i = j/portletNum + 1;
18          time = timestamps[i];
19       t = time - old_time;
20       T += t;
21       old_time = time;
22
23       sum{portletNames[j]} += v * t;
24
25    for each (portlet in sum)
26       visibility{portlet} = sum{portlet} / T;
27
28    createChart(visibility);
```

Once obtained event timestamps and *portlet* coordinates, the numerator in (1) is calculated through the iteration of lines 12-23. The partial sum is kept by the *sum* associative array (line 23), whose keys are portlet names. The calculation of the visibility percentage in the i-th time interval v_i is delegated, as shown in line 13, to the *calculateVisibilityPercentage* sub-routine. The time intervals t_i can be easily calculated by subtracting the (i+1)-th and the i-th timestamps (line 19). Those time intervals are summed in line 20 to obtain the total time *T*.

The final results are put in the *visibility* associative array, as shown in line 25. Those results are obtained by dividing the partial sums by *T*.

Lastly, visibility is passed to the *createChart* sub-routine, which is responsible for drawing the bar chart (line 27).