OJAX: A Case Study in Agile Web 2.0 Open Source Development

Dr Judith Wusteman
School of Information and Library Studies
University College Dublin
Belfield, Dublin 4
Ireland
Tel: +353 1 2070 718 or +353 1 716 7612
Fax: +353 1 716 1161
judith.wusteman@ucd.ie
URL: http://www.ucd.ie/wusteman/

Biographical note:

Judith Wusteman gained a PhD in Artificial Intelligence from Exeter University and then spent seven years as a lecturer in Computer Science at the University of Kent. She has been lecturing at the UCD School of Information and Library Studies since 1997. Her research interests include Web 2.0, digital libraries, virtual research environments and XML.

OJAX: A Case Study in Agile Web 2.0 Open Source Development

Abstract

Purpose: This paper describes a case study of the development, features and evaluation of a Rich Internet Application for libraries. It attempts to demonstrate best practice in the use of software standards, development processes and evaluation.
Methodology/approach: Web 2.0, open source design methods and usability testing were used within an Agile framework.
Findings: The adoption of Agile software development methods, coupled with usability testing, would enable the library community to take full advantage of the techniques and principles inherent in Web 2.0 open source software.
Research limitations: A major component of the evaluation of OJAX was a series of usability tests. As is the nature of most usability studies, the results are not generalisable.
Originality/value of paper: Aspects of Agile software development methods, such as usability testing and iterative design, are recognised in the literature as contributing to the usability of library software. However, exploration of the use of a full Agile framework plus usability testing to facilitate Web 2.0 open source software is rare in library-related literature.

Keywords: Agile, Open source software, Web 2.0, Usability testing, User Centered Design, Rich Internet Application
Type of paper: Case study

Introduction

OJAX provides an open source federated-search service for metadata from OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) compatible repositories (Wusteman and O'hIceadha, 2006). It is being developed using Web 2.0 open source design methods and usability testing within an Agile software development framework.

This article provides a case study of OJAX's development to date and illustrates how the methodologies and principles chosen have helped to ensure usability. It first briefly explains the technology behind Rich Internet Applications. The various methodologies and principles involved in the design, development and evaluation of OJAX are then introduced. These are Agile methodology, the principles of open source and Web 2.0 software, and usability testing. How these facets can complement one another to produce a usable system is explained. The OJAX system is then described, concentrating on its architecture and its use of Ajax to support the iterative nature of search. This is followed by a discussion of the methods used to evaluate OJAX, in particular, usability testing.

Rich Internet Applications

The OJAX federated search service [1] is an example of a Rich Internet Application (RIA). That is, it is a web application that offer users the features, functionality and dynamic interactivity of a traditional desktop application. One of the key techniques used in many RIAs, including OJAX, is Ajax (Garrett, 2005; Wusteman and

O'hIceadha, 2006), as popularised by Google applications such as Gmail and Google Maps.

All RIAs enable asynchronous communication between browser clients and server-side systems. In an Ajax Web application, the Web page incorporates an Ajax engine written in JavaScript. Users interact with this engine in the same way that they would with a standard Web page except that, instead of every action resulting in a request for an entire new page from the server, user actions generate JavaScript calls to the Ajax engine. If the engine needs data from the server, it requests this asynchronously in the background. So the user is not left waiting for the server to respond to their input. In addition, the entire page need not be refreshed at every user interaction; instead, individual page elements can be rapidly updated as required.

The adoption of Ajax facilitates the movement of network applications away from the traditional monolithic client/server approach and towards Service-Oriented Architecture, epitomised by smaller reusable services with discrete functionality. Such services can be combined and recombined to deliver different applications to users (Wusteman, 2006). In the case of OJAX, for example, multiple front ends could be built using different user interface technologies, but each using the same Ajax technology at the server, thus decoupling the user interface technology from the service provision technology.

Open Source

As the name suggests, software labelled as *open source* provides free access to its source code. But the term implies much more than this: the relevant software must be freely available to any party for any purpose, including all forms of modification or extension. And any such derivations must be distributed under the same open source terms [2]. Open Source Software (OSS) includes Web servers such as Apache [3], the OpenOffice suite [4], the Firefox browser [5], the digital repository systems Fedora [6] and DSpace [7], as well as the topic of this article, the federated search system OJAX.

FLOSS (Free/Libre Open Source Software) [8] is a closely related concept but differs slightly to OSS in it ethos. Whereas open source software is available for incorporation in commercial products, FLOSS software is only available to developers who agree to make the code of any resulting products freely available. FLOSS projects, for example, Wikipedia and the digital repository system EPrints [9], are usually released under a Gnu General Public Licence (GPL) [10]. OSS software is made available under a number of different licences [11], for example, the Apache licence [12] used for OJAX.

OSS development is epitomised by collaborative development by a group of volunteers. As OSS guru Eric Raymond (2000) explains, this may proceed "in a carefully coordinated way by a small, tightly-knit group of people"; alternatively, the software may be "rather casually hacked on by huge numbers of volunteers coordinating over the Internet". Whichever approach is taken, the result is often software of higher quality and higher stability than commercial rivals (Valloppillil, 1998).

Not surprisingly, the development and use of OSS in libraries is growing rapidly (Wusteman, 2004). The benevolent nature of the open source ideal fits particularly well with librarianship culture.

*Open source software development and librarianship have a number of similarities - both are examples of gift cultures …and gain reputation by the amount of "stuff" they give away (*Morgan, 2000).

Agile Development

Agile software development methods encompass many variants but share a common philosophy, as detailed in the Manifesto for Agile Software Development [13]. This philosophy distinguishes it from traditional software development. In the latter, sometimes referred to as the Waterfall lifecycle, the analysis, design, coding and testing phases occur sequentially, with little if any feedback from users, until a software release near the end of the project, possibly years after the requirements were defined.

Whereas Waterfall methods attempt to deny the possibility of change of requirements or opinions within a project, Agile methods attempt to respond to change. Agile development (Sy, 2007) involves a rapid succession of incremental software releases, or iterations. Coding begins early in the project and is preceded by only as much documentation as is actually necessary to move the project forward. Each iteration, generally produced at regular intervals of between two and four weeks, must produce a "working version". The latter is a stable, testable piece of software and implements a subset of the product's final functionality.

As well as frequent releases, Agile philosophy focuses on close collaboration and rapid feedback between developers, users and customers. (In Agile terminology, customers are members of the product team who represent the users' needs within the development process.) Each working version is delivered to users or customers for evaluation, and their feedback is fed into planning for the next iteration.

Open Source and Agile

OSS is not a software development methodology. However, common development practices may be identified within the OSS community. These practices tend to reflect the collaborative nature of OSS projects.

Community is central to the OSS vision. When an active community does not develop around an OSS project, its chances of success diminish. For example, the development of an open source version of the X Window System ultimately failed because of the reluctance of the core developers to allow meaningful collaboration by the rest of the community (Barr, 2004).

Other common aspects of OSS development are the iterative and incremental development of software. Given these practices, it is not surprising that Agile methods, or components of the Agile methodology, are commonly used in OSS development.

The same Agile processes can work for commercial and OSS projects. However, a common difference between such projects is the relative importance placed on features and release dates. Commercial Agile projects tend to trade off features for dates. In OSS Agile projects, on the other hand, there is often less concern about the exact release date. Thus, dates are more likely to be traded off for features.

User-centered design

User-centered design (UCD) (Long et al., 2005), also known as interaction design, identifies end users as central to the design process. As with Agile development, users are directly and iteratively involved at key points within the project (Sy, 2007; Churchville, 2007). Thus, UCD fits Agile's core design principles. There are various user-centred design methods [14], the most popular of which are focus groups, card sorting, questionnaires, interviews, participatory design and usability testing. The first four have been widely used in library-related software development for many years. The latter two have become common more recently. In participatory design, for example, (Roda *et al*., 2005), users do not just provide advice on design issues, but are actively involved in design and decision-making; it is generally used as one of several techniques within a project. Arguably the most important and successful UCD method in the design of software is usability testing.

Usability Testing

Usability refers to "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [15].

Usability testing, or user testing, is now a common component of library-related software design; for example (Jung *et al*., 2008; George, 2008, Long *et al*., 2005; Norberg *et al*., 2005). It involves representative users performing a series of representative tasks (Notess, 2005) in the presence of a tester. Asking users to think out loud about the tasks they are performing is generally found to be the most productive approach. Alternatively, users may be asked to discuss the process after the tests are finished.

When usability testing first emerged in the 1980s, it was a major undertaking that required a fully fitted "usability lab", including a one way mirror and two video cameras at a minimum (Krug, 2005). It was assumed that results were only useful if they were statistically significant. As Krug explains, "It cost $20,000 to $50,000 a shot. It didn't happen very often."

This began to change in the early 1990s with the recognition that useful results could be achieved without expensive usability labs (Nielsen, 1994) and a statistically significant number of testers (Nielsen, 2000). Hence, the emergence of "discount usability testing", with its emphasis on testing early in the design phase, testing regularly and responding to feedback from tests. Thus, usability testing makes most sense as part of an iterative design process (as explored, for example, in (Norberg, 2005)). Testing early and regularly and responding to feedback are all key Agile design principles.

Krug (2005) states that three testers, or at most four, are likely to experience most of the major problems with any system. At this point, the most useful thing a developer can do is to fix those problems and then test again, rather than to search for the less significant problems in the same version.

A simple test early—while you still have time to use what you learn from it is almost always more valuable than a sophisticated test later (Krug, 2005).

Web 2.0 and OSS

Many Web 2.0 services - for example Drupal [16], WordPress [17] and OJAX - are OSS. Others are FLOSS, for example, Wikipedia. However most of the highest profile Web 2.0 services - for example, Google, Flickr, YouTube and Facebook, – are not OSS *per se*. But it is increasingly common for such high profile services to provide open source application programming interfaces (APIs). For instance, Facebook announced plans for an open API in May 2008 (Arrington, 2008). A significant development in 2007 was the emergence of OpenSocial [18], a common API for social applications across multiple websites, with the support of Yahoo!, MySpace, and Google. Another illustration of the interconnectivity of Web 2.0 and OSS is the fact that Google runs on Linux, as does Amazon, and Yahoo runs on BSD, an open source version of Unix.

Central aspects of both OSS and Web 2.0 are community, openness and user control. In OSS, this refers primarily to the community of developers, along with openness and user control of code. User behaviour is continuously monitored and, in response, applications are modified on an on-going basis; such software is sometimes referred to as being in "permanent beta" release [19].

In Web 2.0, on the other hand, these references are primarily to the community of users and openness and user control of content. According to O'Reilly (2005), one of the defining characteristics of Web 2.0 applications is that users are treated as "co-developers". But this generally refers to co-development of content, not code. Users may share data streams, as illustrated in the many mashups created using Google Maps. Users may share comments and feedback, as evinced by the book reviews in Amazon and the folksonomies created in Flickr. And users may share content, as in Wikipedia, LibraryThing [20] and eBay. In the latter examples, the actual product is "the collective activity of all its users" (O'Reilly, 2005).

When Web 2.0 "network effects from user contributions" (O'Reilly, 2005) are combined with aspects of the OSS feedback cycle, powerful development methods are enabled for open source Web 2.0 services.

The Open Source feedback cycle

In addition to directly solicited feedback, for example via user centred design methods, an important dynamic in the evolution of OSS is the informal feedback from members of the open source community. OJAX has benefited from such feedback.

OSS and FLOSS are generally developed by distributed teams, communicating solely via electronic means; most such developers never meet in person. As Crowston and Scozzi (2008) point out, the literature identifies major problems with distributed software development. However, FLOSS development and by extension, OSS development, present an "intriguing counter-example", resulting in many extremely successful projects. The teams in such projects are generally self-organizing, without hierarchical control. "Coordination seems… to spontaneously emerge." Developers move between roles as the project evolves. Active users often become co-developers by submitting a bug report or fix, providing code updates or contributing results of user trials.

SourceForge

One of the main conduits for feedback in open source software is the software foundry. The best known example of the latter is SourceForge.net which is the world's largest open source software development website. It offers free hosting for project code and documentation and provides tools and resources for managing and enabling community input to projects, and for developing, distributing and supporting software. The OJAX project is hosted on SourceForge [21].

In a typical SourceForge-hosted project, a developer checks source code changes into the foundry. These changes are then available to anyone who is interested in the project. They can download the code and try it out, provide feedback or ask questions, raise bug reports or comment on reports raised by others. All input is recorded so that it is simple to track changes to a version, or discover the entire history of a bug, for example.

Central to an open source project's success or otherwise is its *traction*, measures of which include how actively the project is being advanced and the proportion of the relevant community that is contributing to development or use. SourceForge provides detailed statistics on all project activity and all input from developers and users. If more than one project meets a proportion of a potential user's requirements, they are likely to make pragmatic decisions based on traction. Given the choice between a product that meets all the user's requirements but has not had a file change for three years and a product that does not meet all requirements but is highly active, users will tend to choose the latter. The assumption is that the missing features may well be added within a reasonable timeframe. In fact, the user in question may well become actively involved in suggesting the advances they need.

Agile Web 2.0 OSS Development with Usability testing

As demonstrated in the previous sections, and illustrated in Figure 1, there is considerable overlap between Agile software development methodology, usability testing and the principles commonly used to develop OSS and Web 2.0 services. Combining all of these methods and principles maximises the probability that the resulting system will satisfy the users' needs and facilitate their productivity. There is discussion in library-related literature concerning the pairing of some of these systems in the development of software. But, until now, there has been no discussion concerning the combination of all of these approaches in one library-related project. The OJAX project attempts to explore this and to illustrate best practice in the use of these methods and principles in the development and evaluation of library-relevant RIAs.

Figure 1:
The overlap between methodologies and principles in Agile, UCD, OSS and Web 2.0

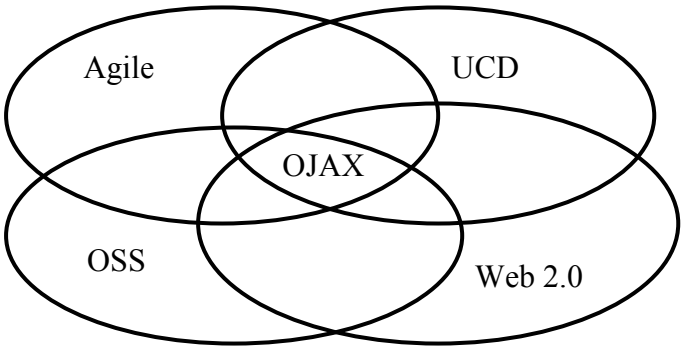


Figure 2: The OJAX user interface

OJAX

The OJAX GUI is illustrated in Figure 2. One of the goals of OJAX is to reflect the "simple search" experience of Google while providing the power of an advanced search.

Google Scholar has taught us, quite powerfully, that the user just wants a search box. Arguments as to whether or not this is "best" for the user are moot – it doesn't matter if it's the best if nobody uses it (Miller, 2004).

In most cases, search is an iterative process:

Users do not know the right questions to ask until they begin to see some of the results and learn about the subject….. The interface should support the iterative nature of the search task. In particular, it should invite refinement and exploration (Rose, 2006).

OJAX is designed to support this iteration and refinement. As users enter data, OJAX responds in real time to accommodate it via a range of Ajax-enabled features, some of which are summarised below. A full description of features available in Version 0.1 is available in (Wusteman and O'hIceadha, 2006).

One such features is the immediate search that is triggered whenever an entire option has been selected (for example, when a suggested auto-completion is selected in the search terms field) or when the system deems that a user is likely to have finished entering a search term (for example, five seconds of user inactivity for a modified field). The auto-search feature effectively renders the Search button redundant but, as confirmed by usability testing, the latter is retained to avoid confounding user expectations.

OJAX's dynamism means that it is unnecessary to provide distinct options for simple and advanced search and for refining a completed search. The user need never navigate between pages because all options, both simple and advanced, are available from the same page. And, as illustrated in Figure 3, all results are made available on that same page in the form of a scrollable list. The only point at which a new page is presented is when the user chooses to view the full details of a search result.

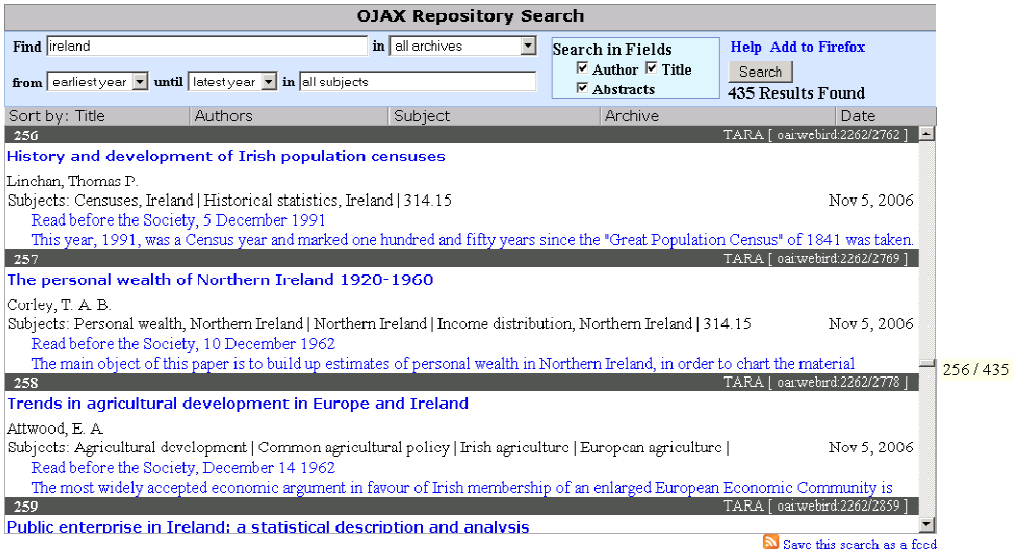
To the user, it appears that the scrollable list is seamless and that all 435 search results are already downloaded. In fact, only 259 have been downloaded. The rest are available at the server. As the user scrolls further down, say to item 270, an Ajax request is made for the next eleven items. Any item downloaded is cached by the Ajax engine and need not be requested again if, for example, the user scrolls back up the list.

The dynamic scroll bar to the right of the results indicates that there are 435 results for this particular search and that the current scroll position is at result number 256. This number updates instantaneously during scrolling, preserving the illusion that all results have been downloaded and providing the user with dynamic feedback on their progress through the results set.

In the initial results display, only one line of the title, authors, subject, abstract and (optional) comment fields are shown for each item. As the cursor is hovered over the relevant field, the display expands to show any hidden detail in that field. When the cursor is hovered over the bar containing the resource identifier, all display fields for that item are expanded.

Because results are available so quickly and because they can be re-sorted so rapidly, it is not necessary to offer pre-search selection of sort options. Rather, users iterate towards the search results they require by manipulating the results in real time. To further facilitate rapid presentation of results, after a re-sort, only the first screenful must be downloaded before they can be presented to the user.

Figure 3: Display of search results in OJAX

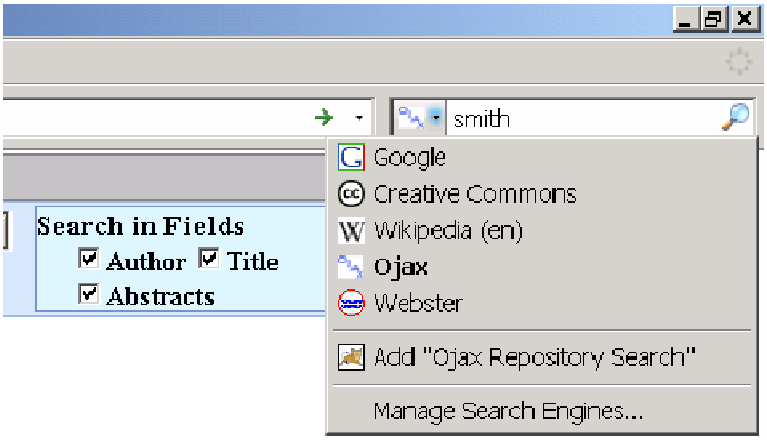


Use of OpenSearch in OJAX

As of Version 0.7, an OJAX-powered repository can be searched directly from the browser search bar, without the need to download or configure extensions or plugins and without having first to navigate to the repository search page. This is enabled by use of the OpenSearch standard [22] and is illustrated in Figure 4. In addition, OJAX was the first open source search engine to incorporate OpenSearch auto-completion in the browser search bar [23].

OJAX also supports OpenSearch Discovery [24]. This means that, when users visit an OJAX-powered repository search page, Firefox 2 or IE 7 will automatically detect that the repository can be searched via OpenSearch and will offer to add that repository to the set of search plugins installed in their browser. In addition, when using Firefox 2, the phrase “Add to Firefox” will appear above the search button, as in Figure 4. Similarly, the phrase “Add to IE” will appear when using IE 7, and “Add to browser” when using any other OpenSearch compatible browser. This feature is OJAX-specific and enabled via a JavaScript call rather than via OpenSearch.

Figure 4: Searching OJAX directly from the browser search bar



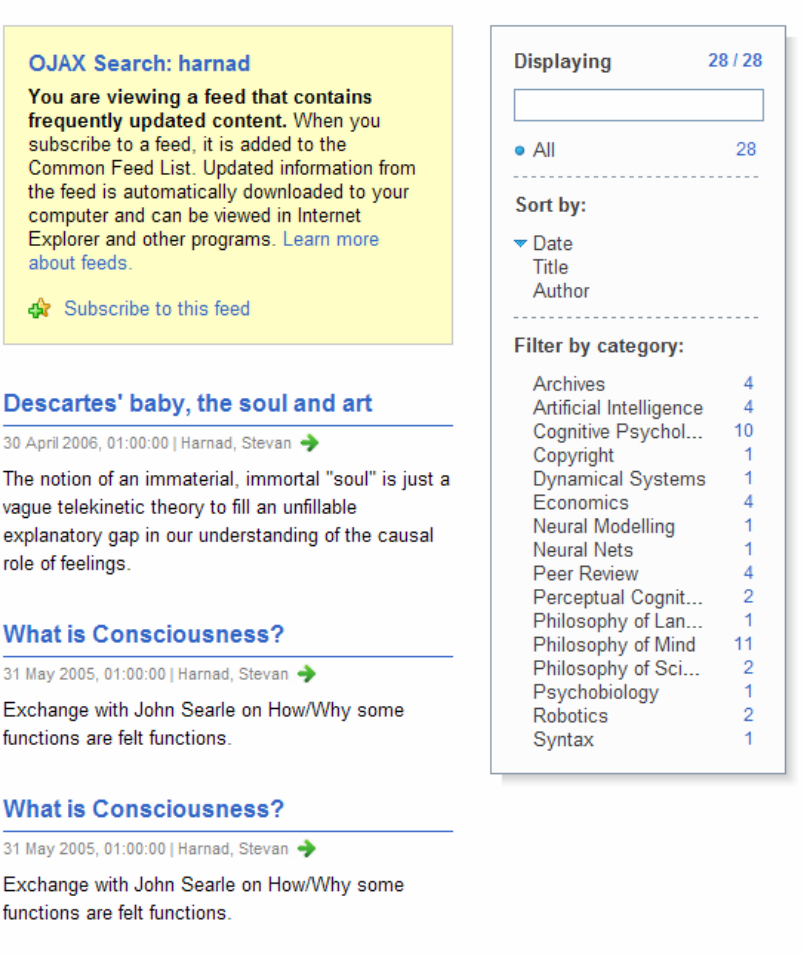
Storing searches as Atom feeds

If users frequently perform the same searches on an OJAX-powered repository, they can click on the "Save this search as a feed" link, at the bottom right of the page, to add an OpenSearch Atom feed for that search to their feed reader. As content matching their query is added to the repository in question, the feed reader will detect the new additions. Some OpenSearch-aware feed readers, such as IE 7.0, will take advantage of the extra structuring of metadata in Atom to sort and filter the results data and to display the different fields appropriately to their function. This is illustrated in Figure 5, which shows the results of an OJAX search on “harnad”. Firefox, on the other hand, does not interpret the extra Atom metadata but treats the feed as it would an RSS file.

Most sites, even those using OpenSearch to describe their search syntax to browsers, publish their search results in HTML. Unfortunately, there is no standardisation in the HTML formats used for search results. One of the great advantages of returning search results in Atom is that, unlike HTML, it is machine-readable. Thus, any OpenSearch-aware browser or library application can initiate an OpenSearch query against the OJAX

repository and receive a machine-readable response. Hence, the results from a search via OJAX could be fed directly into another application, thus facilitating mash-ups.

Figure 5: An Atom feed displayed in Internet Explorer

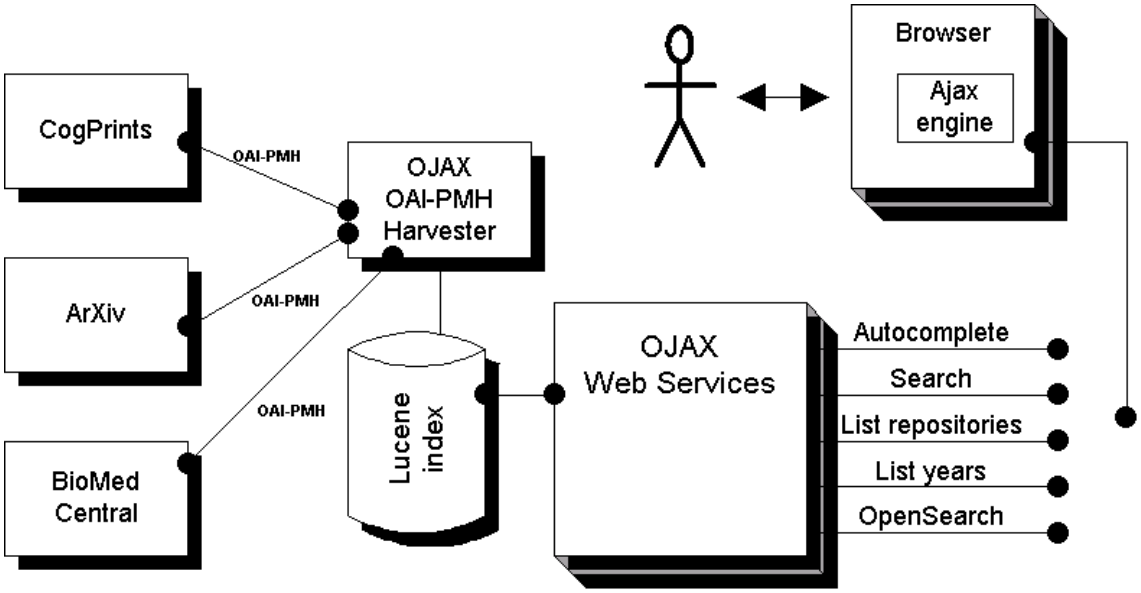


7.0

OJAX Architecture

As illustrated in Figure 6, the OJAX architecture comprises four main components: at the server end are the Harvester, repository search index and Web Services and, at the client end, is the Ajax engine that drives the GUI. As of Version 0.7, the OJAX software is in Beta release. It is available for download [1] with or without a sample repository index; the latter allows users to try out the software without having to harvest a repository. A live demo is also available [1].

Figure 6: OJAX architecture



OJAX GUI

The OJAX GUI comprises a single HTML file plus a CSS (Cascading Style Sheet) file. The JavaScript Ajax engine is included within the HTML file. By default, searching is on author, title and abstract. These fields map to the creator, title and description Dublin Core metadata fields [25] harvested from the original repositories. The search can be restricted by deselecting unwanted fields.

Harvester and Search Index

Apache Lucene [26] is an open source information retrieval API (Application Programming Interface). In effect, it is the technology to build a search engine. The OJAX Harvester uses Lucene to index the records harvested from the required OAI-PMH-compatible repositories and to maintain the resulting inverted index of terms. The use of Lucene also provides a query language, for advanced OJAX users, that includes Boolean operators, support for fielded data and wildcard searches.

Web Services

The term Web Service refers to the use of XML and related technologies to enable the seamless interoperability of Web-based applications. Web Services go beyond the functionality of simple Web pages; they provide dynamic application-to-application functionality that can be remotely invoked (Wusteman, 2005). OJAX Version 0.7 includes five Web Services, each of which is a small reusable service with discrete functionality. For example, the Autocomplete Web Service is invoked by the Ajax engine whenever the user presses a key while in the Search term field. The Web Service simply searches the Lucene index for suggested completions for the search term entered so far. The Ajax Engine will then display this list to the user. Because of its discrete functionality, this Web Service can be reused in other aspects of the system, for example, in providing suggested completions for user entries in the Subject field. In addition, the Service-Oriented Architecture used by OJAX's Web Services allows OJAX to be reconfigured relatively simply, thus enabling mash-ups.

User Feedback

There have been multiple avenues for user feedback concerning OJAX. These include directly solicited feedback via usability testing, with both proxy and representative users, and involvement in the open source feedback cycle via SourceForge.

In relation to the latter, several members of the OJAX development community became temporary co-developers: reporting and fixing bugs, suggesting enhancements and proposing new features. OJAX benefited particularly from the contributions of one co-developer who performed extensive testing and made considerable contributions to the analysis of issues he uncovered.

Usability Testing

Usability testing has informed the evolution of OJAX, providing justification for the modification of existing features and the inclusion of new features. To date, three forms of usability testing have been applied to OJAX. The first was usability testing of similar systems by a proxy user. The results of this testing, along with some discussions with potential users, were used to inform the development of Agile use cases. The latter are simple descriptions of user requirements and goals. At this point, the initial prototype was developed. Once the first working version was produced, iterative proxy usability testing began. It was repeated at the end of each iteration, the length of which varied between a week and a month.

The third form of usability testing involved representative users and was set in motion once the system was feature-complete and all of the proxy users's feedback had been taken into account. These usability tests took place between February and March 2007. Seven test users took part, three evaluating one iteration and four evaluating the following iteration. As OJAX will be used chiefly in an academic setting, all users were university-related. They included postgraduate and undergraduate students, lecturers and library professionals. The testers for both iterations represented a range of ages and expertise in online searching and were diverse in gender. None were familiar with OJAX before the tests and so were able to provide the facilitator with feedback on their initial impressions of the software.

Following the guidelines in (Krug, 2005), four main methods of data collection were employed: pre- and post-evaluation questionnaires, "get it" testing and "key task" testing. The scripts for all of these components may be found in the Appendix. Each session, including pre and post evaluation questionnaires, lasted approximately one hour (Nielsen, 2005).

Pre-evaluation

Users were asked to answer a brief pre-evaluation questionnaire, covering their occupation and questions concerning their use of the internet, search tools and related software. Users were requested to provide signed permission for the session to be videotaped.

The facilitator's introductory comments were adapted from a test script by Steve Krug [27] and included the crucial reassurance to the tester that "I'm testing the software, not you. You can't do anything wrong here." Testers were asked to think out loud and to give honest opinions. They were invited to ask questions during the testing with the proviso that the facilitator might answer some questions at the end of the session.

"Get it" testing

Users were then shown OJAX for the first time. They were asked to comment on their immediate response to it and then to discuss how they might carry out a search. This gave the facilitator some initial pointers as to which aspects of OJAX were intuitive and which were not.

Key task testing

Users were asked to use OJAX to perform searches on a series of pre-defined queries, again thinking aloud as much as possible. In general, the facilitator avoided giving hints, at least initially. However, if several users were having problems with the same feature, the facilitator helped to move them on.

Post-evaluation

The key task tests were followed by a brief post-evaluation questionnaire in which users were given the opportunity to summarise qualitatively their experience of using OJAX.

Results of Usability testing

Substantial pretesting by the proxy user eliminated many of the more major usability issues before the commencement of usability testing by representative users. The issues that remained represent three types of scenario:

- Those in which users went astray momentarily but were able to right themselves almost immediately: so-called "kayak" situations (Krug, 2005).
- Those in which users were confronted with a metaphor that was sufficiently unfamiliar that it was necessary to spend some time learning it before successful use could proceed.
- Those in which users confronted a metaphor that was either unintuitive, or that involved an unintuitive component, resulting in confusion.

Representative examples of each of these scenarios follow.

"Kayak" issues

Among the new or unfamiliar metaphors to which there was generally a positive response were:

- Auto-search
- Displaying results in a dynamic scrollable list
- The ability to search OJAX directly from the browser search bar
- The Atom feed feature

There were occasional "kayak" moments when users were initially taken by surprise by OJAX's behaviour. But, in the case of the features listed above, the users were more often pleasantly surprised than disgruntled. For example, the immediate feedback provided by auto-search was generally welcomed and did not cause much confusion, largely because the search button was still available. Users felt that the system was providing an extra service for them without expecting them to change their behaviour to any great extent. Having said that, the most experienced searchers, that is, the librarians,

tended to prefer to do things for themselves and were not quite as keen for the system to take the initiative.

Learning a new metaphor

Features that required some change in behaviour but resulted in an immediate advantage to the user were also relatively popular. For example, the auto expansion of results and the dynamic scroll bar both required some experimentation by the user before they "got it", but then were welcomed as potentially useful features. However, in both of these cases, feedback from the users resulted in changes to the interface. These were made immediately and tested in the next iteration.

Another aspect that required some learning on the part of the users was the sorting feature. OJAX results may be sorted by title, authors, subject, archive and date. These options are listed on the grey bar immediately above the results list, as illustrated in Figure 3. Clicking one of these options sorts the results in ascending order; an upward pointing arrow appears to the right of the sort option chosen. Clicking on the option again sorts in descending order and reverses the direction of the arrow. Clicking on the arrow removes the sort; the results revert to their original order.

Most aspects of this sorting method have been available to users of Microsoft Windows for the last ten years. A similar sort bar is visible in Windows Explorer. But, when they saw it out of its usual context, the test users did not immediately "get it". However, once it was explained to them, they were pleased with this feature. It appears that the context of familiarity of metaphors is important. This type of result demonstrates the usefulness of usability studies with representative users. The unfamiliarity of this form of search had not been predicted by the proxy user and was a surprise to the developers.

In the post-evaluation questionnaire, users were asked how long it would take them to feel as competent using OJAX as they are using their favourite search system. Most were confident they could master it reasonably quickly; this was particularly true of the four testers in the second round of testing.

Unintuitive features

All test users appreciated the clear, "unfussy" nature of the interface. However, some simplifications were confusing. For example, the use of the "Back" button as the only method of clearing a previous search, was perceived as unintuitive. Users suggested the addition of a "New Search" button.

Similarly, most users found the Lucene Boolean search syntax unintuitive. The use of this search syntax is not central to the functioning of OJAX and may be altered in the future.

In the development iterations between usability testing, the focus was on fixing those issues that were actually affecting the users' ability to use the system successfully; for example, the redesign of the dynamic scroll bar. Minor irritations were eradicated if this could be done quickly, as this would contribute to a positive user approach to the system, ensuring that users would be more likely to persist in their attempt to learn how to use it successfully. For example, after the first set of usability tests, dates listed in the **From** and **Until** date fields were reversed so that the most recent date was listed first.

Usability Study Conclusions

Studies (Adkisson, 2002; Nielsen, 2004) demonstrate that users have a mental model of how the majority of simple web design elements should work. Nielsen (2004) suggests that "a design becomes the expectation when users see it more than half the time". He proposes that "we should design standards for every important website task". In practice, such "ideal" design standards will always be influenced by the potential of technology. Thus, if they are to be realistic, they must be able to evolve as technology enables new features and designs.

According to a recent report commissioned by the British Library and JISC (2008), the traits commonly associated with the Google Generation - "impatience in search and navigation, and zero tolerance for any delay in satisfying their information needs - are becoming the norm for all age groups" [28]. Thus, interfaces need to be familiar enough so that users are prepared to give them a chance. What is needed is to build on user familiarity, determining whether each potential change is acceptable to the user or pushes them beyond their tolerance limit. Evolution is the key here; revolution in user interfaces is unlikely to be successful.

A central aim of OJAX is to test novel or unfamiliar metaphors that have been enabled via Ajax technology. Results show that users are prepared to accept some new or seemingly unfamiliar features, on occasion with enthusiasm, but that other changes are a step too far. However, the level of user acceptance is not static; as specific metaphors emerge and become more familiar, users may be able to attune themselves to some designs that are currently unmeaningful.

In an ideal scenario, usability testing with representative users would have been introduced earlier in the process, rather than relying on usability testing by a proxy user for a considerable number of development iterations. This would probably have identified some usability issues earlier in the design process. However, in practice, the use of a proxy user worked well and was a realistic compromise for a project with limited resources.

As is common with most usability studies, the results are not generalisable but they have been extremely useful in the development of OJAX.

Future Work

Science Foundation Ireland is funding the next stage of development of OJAX [1]. The aim of this new project, which started in September 2007, is to investigate how concepts from the Social Web and recommender technology can be applied to the research environment in order to facilitate dynamic collaboration and the sharing of ideas among researchers. It will be illustrated via the creation of OJAX++, a next-generation collaborative research tool, using Web 2.0 technologies such as Web Services, Ajax, collaborative tagging and recommender functionality. OJAX will be a component of OJAX++. Like OJAX, the OJAX++ project is being carried out using OSS Web 2.0 Agile development methods and usability testing.

In 2008, the majority of widely-used web sites incorporate RIA technology. However, the current generation of RIAs are not entirely accessible to users with special needs. The World Wide Web Consortium now recognises that it is unrealistic to advise web developers to avoid the use of RIAs; instead it is fostering the evolution of "Accessible

Rich Internet Applications" (W3C, 2008). This standard is currently at Working Draft stage and support for it has already been built into Firefox Version 3 [29]. As this standard matures, it will be applied to OJAX++.

Conclusions

In the development of OJAX, an attempt has been made to demonstrate best practice in the application of Service-Oriented Architecture, open standards, Agile development practices, open source and Web 2.0 principles and usability testing. The OJAX project aims to demonstrate how these facets can be used in the development, refinement and evaluation of a library-related open source Rich Internet Application. The literature demonstrates how usability testing combined with iterative development can have a constructive impact on library-related software design. There is also considerable interest in how open source software and Web 2.0 services can benefit the library. The natural next step is for the library community to adopt Agile development methods, thus gaining the full advantages of all of these techniques.

Notes

[1] OJAX homepage: http://ojax.sourceforge.net
[2] The Open Source Definition: www.opensource.org/docs/definition.php
[3] Apache: http://httpd.apache.org
[4] OpenOffice.org: www.openoffice.org
[5] Firefox: www.mozilla.org/firefox
[6] Fedora: www.fedora-commons.org
[7] DSpace: www.dspace.org
[8] Free Software Foundation: www.fsf.org
[9] EPrints: www.eprints.org
[10] GNU General Public License: www.gnu.org/licenses/gpl.html
[11] Open Source Licenses: www.opensource.org/licenses
[12] Apache License: www.apache.org/licenses
[13] Manifesto for Agile Software Development: http://agilemanifesto.org/
[14] User-centered design (UCD) - 6 methods: www.webcredible.co.uk/user-friendly-resources/web-usability/user-centered-design.shtml
[15] Usability Net: International standards for HCI and usability: www.usabilitynet.org/tools/r_international.htm
[16] Drupal open source content management platform: http://drupal.org
[17] WordPress publishing platform: http://wordpress.org
[18] OpenSocial: http://sites.google.com/a/opensocial.org/opensocial
[19] The developers of the Wine application released Version 1.0 of Wine, on June 17th 2008, after 15 years of development: www.winehq.org
[20] LibraryThing: www.librarything.com
[21] OJAX download page, SourceForge: http://sourceforge.net/projects/ojax
[22] OpenSearch: www.opensearch.org
[23] OpenSearch Suggestions extension: http://www.opensearch.org/Specifications/OpenSearch/Extensions/Suggestions/1.0
[24] OpenSearch 1.1 Draft 3: www.opensearch.org/Specifications/OpenSearch/1.1/Draft_3
[25] Dublin Core Metadata Initiative: http://dublincore.org
[26] Apache Lucene: http://lucene.apache.org
[27] Steve Krug's Sample usability test script: www.sensible.com
[28] "'Google Generation' is a myth, says new research":

www.jisc.ac.uk/news/stories/2008/01/googlegen.aspx
[29] New Accessibility features in Firefox 3:
http://support.mozilla.com/en-US/kb/New+Accessibility+features+in+Firefox+3

References

Adkisson, H. (2002), "Identifying De-Facto Standards for E-Commerce Web Sites", MSc thesis, University of Washington. Available at: www.hpadkisson.com/papers/hpa_thesis_final2.pdf (accessed 1 July 2008)

Arrington, M. (2008), "Facebook To Open Source Facebook Platform", *TechCrunch*, 26 May. Available at: www.techcrunch.com/2008/05/26/facebook-to-open-source-facebook-platform/ (accessed 1 July 2008)

Barr, J. (2004), "XFree86 core developers disband - so what?", *Linux.com*, 2 January. Available at: www.linux.com/feature/33494 (accessed 1 July 2008)

British Library and JISC (2008), "Information behaviour of the researcher of the future", 11 January. Available at: www.ucl.ac.uk/slais/research/ciber/downloads/ (accessed 1 July 2008)

Churchville, D. (2007), "Agile User Interface Development", *InfoQ.com*, 19 Feb. Available at: www.infoq.com/articles/agile-useability-churchville (accessed 1 July 2008)

Crowston, K. and Scozzi, B. (2008), "Bug fixing practices within free/libre open source software development teams", *Journal of Database Management*, Vol. 19, No. 2, pp. 1-30.

Garrett, J. J. (2005), "Ajax: A New Approach to Web Applications", *Adaptive Path Inc*, 18 Feb. Available at: www.adaptivepath.com/publications/essays/archives/000385.php (accessed 1 July 2008)

George, C.A. (2008), "Lessons learned: usability testing a federated search product", *The Electronic Library*, Volume 26 Issue 1 pp 5 – 20.

Jung, S., Herlocker, J.L., Webster, J., Mellinger, M. and Frumkin, J. (2008), "LibraryFind: System design and usability testing of academic metasearch system" *Journal of the American Society for Information Science & Technology*, Feb, Vol. 59 Issue 3, pp 375-389.

Krug, S. (2005), "*Don't make me think: A Common Sense Approach to Web Usability*", 2nd ed., New Riders Press, Pearson Education.

Long, H., Lage, K. and Cronin, C. (2005), "The flight plan of a digital initiatives project, part 2: Usability testing in the context of user-centered design", *OCLC Systems and Services,* Vol. 21, No. 4, pp. 324-345.

Miller, T. (2004), “Federated Searching: Put It in Its Place”, *NetConnect Library Journal*, 15 April. Available at: www.libraryjournal.com/article/CA406012.html (accessed 1 July 2008)

Morgan, E.L. (2000), “Gift Cultures, Librarianship, and Open Source Software Development”, *Infomotions*, 28 Dec. Available at: www.infomotions.com/musings/gift-cultures.shtml (accessed 1 July 2008)

Nielsen, J. (2005), “Time budgets for usability sessions”, *Useit.com: Jakob Nielsen’s Website*, 12 Sept. Available at: www.useit.com/alertbox/usability_sessions.html (accessed 1 July 2008)

Nielsen, J. (2004), “The Need for Web Design Standards”, *Useit.com: Jakob Nielsen’s Website*, 13 Sept. Available at: www.useit.com/alertbox/20040913.html (accessed 1 July 2008)

Nielsen, J. (2000), “Why you only need to test with 5 users”, *Useit.com: Jakob Nielsen’s Website*, 19 March. Available at: www.useit.com/alertbox/20000319.html (accessed 1 July 2008)

Nielsen, J. (1994), “Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier”, *Useit.com: Jakob Nielsen’s Website*. Available at: www.useit.com/papers/guerrilla_hci.html (accessed 1 July 2008)

Norberg, L.R., Vassiliadis, K., Ferguson, J. and Smith, N. (2005), "Sustainable design for multiple audiences: The usability study and iterative redesign of the Documenting the American South digital library", *OCLC Systems and Services,* Vol. 21, No. 4, pp. 285-299.

Notess, M (2005), “Designing effective tasks for digital library user tests: lessons learned”, *OCLC Systems & Services*, Vol. 21 No. 4, pp. 300-310.

O’Reilly, T. (2005), “What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software”, *O’Reilly Media*, 30 September. Available at: www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (accessed 1 July 2008)

Raymond, E.S. (2000), “A Brief History of Hackerdom”, 5 May. Available at: www.hackemate.com.ar/hacking/eng/part_00.htm (accessed 1 July 2008)

Roda, C., Borel, A. M., Gentchev, E. and Thomas, J. (2005), “Digital image library development in academic environment: designing and testing usability”, *OCLC Systems & Services*, Vol. 21 No. 4, pp. 264-284.

Rose, D.E. (2006), **“**Reconciling information-seeking behavior with search user interfaces for the Web”, *Journal of the American Society for Information Science & Technology*, Apr, Volume 57 Issue 6 , pp 727 – 846.

Sy, D. (2007), "Adapting Usability Investigations for Agile User-Centered Design", *Journal of Usability Studies*, May, Volume 2, Issue 3, pp. 112-132. Available at: www.upassoc.org/upa_publications/jus/2007may/agile-ucd.html (accessed 1 July 2008)

Valloppillil, V. (1998), "Open Source Software: A (New?) Development Methodology", *Microsoft internal report*, August 11. Available at: www.scripting.com/misc/halloweenMemo.html (accessed 1 July 2008)

W3C (2008), "Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap)", *W3C Working Draft*, 4 February.
Available at: www.w3.org/TR/wai-aria-roadmap/ (accessed 1 July 2008)

Wusteman, J. and O'hIceadha, P. (2006), "Using Ajax to Empower Dynamic Searching", *Information Technology and Libraries (ITAL)*, Vol. 25, No. 2, June 2006, pp 57-64. Available at: www.ucd.ie/wusteman/articles/wusteman-ojax.pdf (accessed 1 July 2008)

Wusteman, J. (2005), "Realising the Potential of Web Services", *OCLC Systems & Services*, Vol 22, Issue 1. Available at: www.ucd.ie/wusteman/articles/wusteman-web-services.doc (accessed 1 July 2008)

Wusteman, J. (2004), "Patently ridiculous", *Library Hi Tech,* Vol 22, No 2, pp. 231 – 237. Available at: www.ucd.ie/wusteman/lht/wusteman-patents.doc (accessed 1 July 2008)

Appendix

OJAX pre-evaluation questionnaire

1. Occupation
    - Undergrad
    - Postgrad
    - Postdoc
    - Lecturer
    - Library professional
    - Other. Specify:

2. On average, how many days a week do you access the Internet?
    - 7 days a week
    - 6 days a week
    - 5 days a week
    - 4 days a week
    - 3 days a week
    - 2 days a week
    - 1 days a week
    - less than once a week

3. If you access the Internet, what Web browsers do you use?
    - Firefox

- Microsoft Internet Explorer
- Mozilla (other than Firefox)
- Opera
- Safari
- Konqueror
- Netscape
- Other. Specify:

4. On average, how many days a week do you **search** the Internet?
    - 7 days a week
    - 6 days a week
    - 5 days a week
    - 4 days a week
    - 3 days a week
    - 2 days a week
    - 1 days a week
    - less than once a week

5. If you search the Internet, what search engines do you use?
    - Google
    - Yahoo
    - MSN
    - Ask
    - AOL
    - Other. Specify:

6. On average, how many days a week do you use the Internet to search for academic material? (That is, journals, journal articles, theses, reports, grey literature, monographs etc)

    - 7 days a week
    - 6 days a week
    - 5 days a week
    - 4 days a week
    - 3 days a week
    - 2 days a week
    - 1 days a week
    - less than once a week

7. If you search online for academic material, what search engine/ databases / other systems do you use?
    - Google
    - Yahoo
    - Ask
    - Google Scholar
    - SwetsWise
    - Science Direct
    - Cambridge Scientific Abstracts

- LISAnet
- Web of Knowledge
- Library catalogue
- Other. Specify:

8. Which, if any, of the above systems is your preferred method of searching for academic material and why?

9. Is there anything that you think could be improved in you favourite system(s) for searching for academic material?

10. Have you heard of / used "search feeds" or "Atom feeds" or "RSS feeds"?
    - Never heard of them
    - Heard of but not used them
    - Have used them

11. Have you heard of / used Google Scholar?
    - Never heard of it
    - Heard of but not used it
    - Have used it

12. If you have used Google Scholar, what do you think of it?

OJAX Evaluation

(Adapted from a test script by Steve Krug [27].)

Introduction

You probably already know, but let me explain why I've asked you to come here today: I'm testing a federated search system that we're working on. It's called OJAX. I want to see how people use it and what they think of it.

Introducing federated search

A federated search system is a search system that searches across more than one database – or repository - at the same time. The implementation that you're going to be looking at is set up to search for academic articles, and the like, across Cogprints. Cogprints is a database of articles on Cognitive Science.

Reassurance

- I want to make it clear right away that I'm testing the *software*, not you. You can't do anything wrong here.
- I want to hear exactly what you think, so please don't worry that you're going to hurt my feelings. I want to improve it, so I need to know honestly what you think.
- As we go along, I'm going to ask you to think out loud, to tell me what's going through your mind. This will help me.
- If you have questions, just ask. I may not be able to answer them right away, since I'm interested in how people do when they don't have someone sitting next to them, but I will try to answer any questions you still have when we're done.

- With your permission, I'm going to videotape the computer screen and what you have to say. The video will be used only to help me figure out how to improve the site, and it won't be seen by anyone except the people working on the project. It also helps me, because I don't have to take as many notes. I will destroy all personal data after the study.
- Would you mind signing to say you don't mind me videotaping you and using the video as I just explained?
- Do you have any questions before we begin?

Reactions to OJAX

First, I'm just going to ask you to look at this page and tell me

- What strikes you about it
- What you think you would do first if you wanted to carry out a search

Again, as much as possible, it will help me if you can try to think out loud so I know what you're thinking about.

Key task testing

Now I'm going to ask you to perform some searches. Please read the details of each search and ask me any questions you have about the particular search before you performed it. Again, please think aloud as much as possible.

Search Query #1: Imagine you want to find out what articles are available on the subject of dreaming. Do a search and look at the results.

Search Query #2: Imagine you want to find out what articles have been written on neurology since 1996. Which of them are written by Robin Allott?

Search Query #3: In article "Self-Awareness, self-esteem, and alcohol use in famous and relatively well-known individuals." by Alain Morin, who's short stories indicated "increased use of first-person voice after he attained celebrity"?

Search Query #4: Search for all articles involving "epistemology" published after 2005 under the subject "cognition". If you can't find any results for this search, expand the search to cover all articles covering "mind" as well.

Search Query #5: OJAX can be used directly from the browser search bar (top right). Try any search you like using the browser search field.

Query #6: What do you think the "Save this search as a feed" button does?

Post-evaluation questionnaire

1. How would you describe your experience of using OJAX?
2. What features did you particularly like/dislike
3. How long would it take you to feel as competent using OJAX as you are using your favourite search system?
4. What new features would you like in OJAX?

5. Are there any features from other systems you would like to see incorporated in OJAX?
6. Overall, how do you rate OJAX?