Issues in WebLab development: security, accessibility, collaboration and multilinguality

Javier Oliver, Joseba Abaitua, JosuKa Díaz, Inés Jacob, David Buján, Pablo Garaizar, Luis Lázaro
Faculty of Engineering, University of Deusto. Bilbao, Spain
e-mail: oliver@eside.deusto.es – abaitua@fil.deusto.es
[josuka, ines, dbujan, garaizar, llazaro]@eside.deusto.es

Abstract

WebLabs are novel eLearning applications with a promising future. Once a WebLab reaches the state of stable prototype, there are still many issues that have to be taken care of before the system can be considered a first quality web based application. In the following pages some of these issues are addressed, and pointers are given so as to take the first steps in dealing with WebLab security, accessibility, collaborative development and multilinguality.

1. **WebLab security**

For years WebLabs have been developed and maintained by professionals working in laboratories. These professionals, mainly electronic and control engineers, are experts in hardware or microelectronics, but not necessarily in system administration and security issues.

Supported by the new European Higher Education Space influenced by the Bologna Declaration, WebLabs are growing in importance [1]. In a near future, they should be able to leave the restricted laboratory scope to be included in the infrastructure of the University and be considered as an important part of e-Learning for certain subjects. Of course, to reach this point the WebLabs should meet some requirements including the fulfilment of the security policies that already apply to the rest of the IT services offered by the University.

In most cases, the collaboration of the IT Department of the University will be a must to be able to apply the existing knowledge of system administration and security to the WebLab servers. Of course, lecturers and laboratory technicians will still be responsible for the contents and the hardware side of the system.

Experts agree: Absolute security does not exist. It is not difficult to find peculiar phrases supporting this affirmation like “The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts.” [2]

However, we should always keep in mind some good practices and recommendations related to network and both server and client-side security.

1.1. *Server-side security*

Once WebLab servers fulfil the security policies defined by the IT Department of the University, we should not have to face unpleasant situations like servers which are not up to date with missing security patches/hotfixes or old packages installed, servers with unnecessary services running or servers with unconfigured or badly configured services.

Of course, a secure operating system is an important and necessary piece of the total system security puzzle, but it is not the only one. A highly secure operating system would be insufficient without application-specific security built upon it [3]. To achieve this objective, security should be considered an essential matter during the design stage of the WebLab.

Another important aspect of security WebLab developers should work in is the integration of the user authentication system with existing Lightweight Directory Access Protocol (LDAP) or RADIUS servers. We can find good examples of this integration in the Virtual Internet and Telecommunications Laboratory of Switzerland (VITELS) [4].

This practice would help not only to avoid situations like the use of weak passwords as the complexity requirements established by the IT Service for the LDAP server apply, but also to prevent attacks against the usually less secure local database keeping usernames and passwords.

Finally, we should not forget about physical security of the servers supporting the WebLab. If possible, servers should always be physically located in an access-controlled environment to prevent locally performed attacks.

1.2. *Client-side security*

A large number of the analyzed WebLabs require the use of Java applets, ActiveX controls or a Flash player on the client side. This necessity could be considered a drawback for different reasons.

First of all, some of this plug-ins may not available on every platform. Secondly, any plug-in might require administrative privileges for its installation. In some scenarios a student may fail when trying to access a WebLab using a computer of the University because of a lack of privileges.

Even though Java Virtual Machine or ActiveX controls are claimed to be secure having their own security models like Java sandbox or digital signatures, malicious or flaw executable code may exploit security breaches [5].

It's not easy to find WebLabs working without any plug-in installation required. WebLab-Deusto, a pure HTML/JavaScript system programmed with AJAX [6], is a good example of these so called thin clients. In the thin client model, all the code is executed by the server side of the system while the client only performs simple display functions.

1.3. *Network security*

The TCP/IP protocol has been criticized as having been designed with no thought of security. There are a number of serious security flaws inherent in the protocol suite. In spite of this, we can always use mechanisms that ensure confidentiality and integrity of data transmitted over the network and allow us to establish secure and encrypted connections between the clients and servers.

We have some examples of the use of Virtual Private Networking (VPN) mechanisms to encrypt network communications between the client and the server in off-campus accesses [7] [8]. VPNs are used to create an encrypted tunnel for the traffic sent between the student's computer and the VPN server, allowing students to make secure connections to Campus Lab environments.

More extended is the use of secure HTTP (HTTPS) which includes the Secure Socket Layer (SSL). SSL is used to provide secure channels and its use is mainly recommended. This is specially important in pages that work with sensitive information like the login pages that perform the authentication of the username and password.

Firewall technology has become the most popular defense to prevent illegitimate traffic inside the corporate network. Configuration of firewalls is a formidable and error prone task. This emphasizes the need to restrict or reduce complexity at the firewalls [5]. The use of not well-known ports in addition to http (80/tcp) and http-ssl (443/tcp) ports is quite usual in existing WebLabs and could be considered a security flaw because it adds complexity to the management of the firewalls.

Taking all this into account, it seems clear that security is should be considered an essential matter during the design stage of the WebLab, and that the installation, administration and maintenance of the servers which support the WebLab should be left to the IT Service of the University. It is advisable to locate the servers supporting the WebLab in a restricted access area and to integrate the authentication of the web server with existing LDAP servers in the organization. One should minimize the number of not well-known open ports required to allow communication between the server and the clients, and try to use thin clients, leaving the code execution work to the server side of the system.

2. **WebLab accessibility and usability**

Web applications have changed the lives of many people nowadays. Communication, access to information, learning, commercial transactions and entertaining, among others, have changed dramatically with the introduction of the Internet and Web based applications. These technologies are specially promising for disabled people, who can now perform for the first time many of the above activities. For example, until now a blind person could not access the information in a daily newspaper, except by having someone reading it out loud. Today the combination of Web versions of newspapers, and screen reading software allow blind people to read the news, assuming that the newspaper has designed its Web version without accessibility barriers [9].

Accessibility barriers that WebLabs and other Web based applications should avoid include graphical information without a textual description, mouse only interaction, or multimedia content without adequate captioning. The Internet has the potential to increase the access of disabled people to all kinds of learning tools, but accessibility barriers reduce or completely eliminate this potential, leaving disabled people as helpless and discouraged as before.

Developers should take into account that disability is a relatively common condition. As many as 10% of the population suffer some kind of disability, and we could add to these certain temporary situations in which the ability to carry out certain tasks is decreased: medication, noisy work environment, use of old technology (slow devices, small screens), etc. It should also be mentioned that optimizing the access to Web applications for the disabled often improves the use of the system for all users.

Other reasons to consider this issue seriously are, in the case of commercial applications, the fact that a company can hardly ignore 10% of their potential customers, and for many other developers, legal issues that might make conformance to certain accessibility guidelines mandatory.

2.1. *Laws and regulations*

Accessibility of information technologies has not been regulated until recently. In the case of Spanish legislation, we can mention the "LEY 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico" (LSSICE), in which the issue of accessibility of the elderly and the disabled to information delivered through electronic media is addressed. The LSSICE states that Public Administrations should ensure that information they offer in their Web pages is accessible for disabled and elderly users by the end of the year 2005. Public Administrations should also require that all new Web pages they finance, fulfill that accessibility requirements mentioned above. This law also recommends to promote the adoption of accessibility guidelines in the case of service providers and hardware and software vendors [10].

More recently, the "LEY 51/2003, de 2 de diciembre, de Igualdad de Oportunidades, no Discriminación y Accesibilidad Universal de las personas con discapacidad" (LIONDAU), enforces universal accessibility and no discrimination. This law makes it compulsory to gradually make all products and services accessible for all, and gives deadlines to carry out all necessary adaptations. In the case of computer products and services -WebLabs included- there is a 4 to 6 year deadline for new developments (8 to 10 years for existing products) to fully apply all accessibility requirements. The National Government assumes the responsibility to study the accessibility of products and services of major impact in the population, and to develop a curriculum in Universal Design for the education of future professional in various fields, including information technology [11].

Many other countries around the word are also regulating accessibility of information technology products and services, and this could apply to WebLabs developed in those countries. New laws and regulations appear frequently, so the interested reader is advised to consult legislation of his home country. The following are just a few examples:

— Germany: "Gesetz zur Gleichstellung behinderter Menschen und zur Änderung anderer Gesetze vom 27. April 2002".

— Italy: “Legge Stanca - "Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici 2004”.
— Portugal: “Resolution of the Council of Ministers Concerning the Accessibility of Public Administration Web Sites for Citizens with Special Needs”.
— USA: “Section 508”. It is part of the Rehabilitation Act of 1973 and it has been one of the most influential legislations in developing awareness in accessibility issues. It states that all computer applications used by the US Federal Government should be accessible to disabled people “unless it would pose an undue burden to do so”.

2.2. *Web Accessibility Initiative*

The World Wide Web Consortium (W3C) stresses the importance of accessible Web content, and promotes the Web Accessibility Initiative (WAI) to develop resources to make Web applications, including WebLabs, accessible to people with different disabilities.

The WAI has developed guidelines which are considered the standard reference for Web content accessibility and can provide different materials and resources to help in the motivation of developers and the development of accessible sites. Guidelines and materials are developed by working groups formed by companies, governments, disability organizations, research institutions, etc. so that a high degree of independence is achieved [12].

Web accessibility includes not only Web sites and applications that people with disabilities can interact with but also Web clients that can be used with screen readers and other assistive technologies and Web authoring tools that promote production of accessible Web pages.

2.3. *A first evaluation of Web accessibility*

One of the resources that the Web Accessibility Initiative offers is a list of actions to take if one wants to verify for the first time the accessibility of a WebLab or any other Web based application. These steps are not intended to fully check the conformance to any set of accessibility guidelines, but to get a general picture of the accessibility status of the application. The steps combine manual checks with the use of semiautomatic tools, and can be easily performed:

1. Selection of a representative sample of pages of the site. One should select the most probable entry pages of the site together with some

pages rich in graphics and tables, or having complex functionality and interaction.

2. Check the selected pages with different browsers, carrying out the following actions:

   — Do not show graphical content and verify that a descriptive text for each image is available
   — Do not activate sound and verify that all audio information has a text equivalent.
   — Modify text size and verify that all changes are properly shown on the screen, and that the applications are still usable at larger font sizes.
   — Test the application with different screen resolutions and window sizes.
   — Change the video equipment to monochrome mode and verify that the contrast among different colors is acceptable.
   — Try to access all the functions in the application through keyboard interaction only.

3. Check the selected pages with special browsers [13] such as a voice browser or a text browser. Verify that the information available using this special browsers is equivalent to that obtained through a conventional graphical user interface. It is specially important that information scanned sequentially is meaningful, because that is the way many of the adaptive technology tools work.
4. Use two or three automatic accessibility evaluation tools [14] and check the results. Web accessibility evaluation tools can be a useful resource and reduce the effort of carrying out evaluations. They can help prevent and eliminate accessibility barriers in Web applications, but they cannot detect or eliminate all accessibility problems, and some checks and verifications should be done manually. These tools cannot judge by themselves the degree of accessibility of a Web application, but can greatly help a human operator in performing this task.

2.4. *WebLab usability*

The design of any interactive information system involves achieving a high degree of usability. This general term includes different aspects such as effectiveness, ease of use, efficiency, learnability and safety.

How can we increase the usability of our WebLabs? The only way is to develop the system with the final user in mind from the very beginning of the process, approach that has been termed *user centered design*. A good place to start getting used to the terminology and techniques of usability is the collection of papers available at [15]. This author puts forward an interesting method, from the practical point of view, which he calls *discount usability*: it is basically a set of low cost techniques that can be applied to any software project. He argues, not without reason, that if one wants to do things very thoroughly, one might not do them at all, and so, it is better to do some simple usability engineering than none.

One of these simple techniques is called heuristic evaluation, and it consists of performing a systematic inspection of a user interface design checking its conformance to recognized usability principles [16]. With just three or four evaluators, the benefits to cost ratio is maximum, and many of the areas of improvement of the user interface can be detected at an early design stage. Normally, evaluators do not communicate until the inspection has been carried out, and only afterwards they combine their findings.

Although the *think aloud* method has traditionally been executed by psychologists and human factors experts, a simplified version can be done by any of the members of the development team of a Web based application. Just get some representative real users, and select some typical tasks to be done with the system. Observe user interaction carefully and insist that the user speaks out all his thoughts. Any difficulty that a user encounters is a hint to a possible design flaw. Accessibility and usability of WebLabs are important factors that must be considered from the first stages in development. Solidarity, economical and legal issues should lead developers to take these factors into account in order to develop better WebLabs for an even wider audience.

3. **Collaborative development models and WebLabs**

There are different development methodologies in the so-called Collaborative Development Model such as Agile Methods, Pair Programming (Extreme Programming) or Community Based Development models. The following section will be focused in the relationship between WebLabs and Community Based Development model comparing it against the traditional Closed-Source Software model.

3.1. *Why should WebLab development model be collaborative?*

Several successful Open-Source projects are based on a Collaborative Development model that keeps them alive and prevents from reaching a level of complexity which could be difficult to manage. Many analyses can be found in literature, [17] [18] [19] [20], describing it and summarizing Community Based models' characteristics: growth speed, creativity, simplicity, fewer defects and extensibility/modularity. Nevertheless, there are also some limitations of Collaborative models [21] and tensions between Community Based model and Corporate Culture or Agile Methods could arise [22].

3.1.1. Community-based models' strengths

Faster system growth. Analyses of big Open-Source projects such as Mozilla or Apache [18] [19] (Mockus *et al*., 2000, 2002) confirm their strong growth ratio, corroborated later by [17] Paulson *et al*. (Paulson *et al*., 2004) by calculating a percentage of growth in each release.

Creativity. It is not easy to measure creativity but [23], [24] and [17] agree that is a typical characteristic of Community Based Development model and it can be estimated by examining the number of features or functions added by release.

Simplicity. Both modularity and simplicity help a project to be extensible and that is a main prerequisite for a successful Open-Source project [23]. Code complexity can be measured using well-known metrics, but as Paulson *et al*. noticed, the complexity of some systems may be found in its data structures rather than in its code.

Fewer defects. This characteristic is very controversial and there are several studies defending each point of view: [19] and [22] state that Open-Source community is more responsive in identifying and fixing defects; in the other hand, [25] argues that high quality only applies to some projects, those with good code review and those with good authors, Open-Source is not inherently better or safer. It may be related with software project's lifecycle and amount of developers. As commented by [17], Open-Source projects need to be in a fairly developed state to be successful [23] [26] [22].

Extensibility/Modularity. All successful Open-Source projects have been designed allowing to be extended easily without having to change core functionality. [23] shows successful examples such as Linus Torvalds' Linux and Larry Wall's Perl, where their feature-set have been evolved in response to the needs of their users.

These characteristics may be merged: by making the users of a product into code developers, you speed debugging, improve quality and gain specialized new features that may eventually turn out to be important to a wider audience [23].

3.1.2. Limitations of the Community Based Development model

Community Based Development model's characteristics are not present in every stage of a project's lifecycle. Both Open-Source and Closed-Source approaches begin in a similar manner: [23], [26] and [22] suggest that it must be something that can be run and tested by developers and contributors, otherwise commented characteristics will not arise.

Besides, Community Based Development process is usually radically different from Corporate Culture: volunteer and non-volunteer work mixed, work is not assigned, no explicit system-level design, no project plan, schedule, list of deliverables, etc. [19].

Other Collaborative Development Models can also conflict with Community Based ones mainly in communication issues [27]):

— Adapting to remote communication: instead of face-to-face verbal communication rigorously controlled, informal communication using e-mail, IRC, Instant Messaging, etc.
— Managing internal and external communication: many contributors may be in other countries speaking other languages.
— Relinquishing control: instead of controlling the direction and development style of the project, share the control with other members of the community.
— Delivery schedules: instead of using fixed time cycles, deliver when the deliverables are useful and stable.
— Good citizenship: there are underlying, sometimes unwritten, rules when participating in the Open-Source Community that must be understood.

Tensions not only come from one side, Open-Source developers are usually not comfortable with Corporate Culture [27]:

— Monitoring of developers: due to voluntary nature of the collaboration, Open-Source developers are not used to be monitored.
— Fixed time schedules: Return-On-Investment and cash flow cycles instead of typically ad-hoc cycles of Open-Source projects ("It is done when it is done").
— Quality Assurance Processes: in spite of the "Linus's Law" ("Given enough eyeballs, all bugs are shallow"), there are no formal Open-Source code review processes normally, and code reviews may be shown by Open-Source developers as an extension of the "monitoring of developers" requirement.

All these arguments may be analyzed and discussed in order to decide if it is a good moment to release a software project as Open-Source or

Free Software. According to this section, it is very reasonable to release as Open-Source a working WebLab that can be run and tested in order to speed up its growth, increase its creativity and extensibility and ease bug fixing, but not before having adapted WebLab's development into a Community Based Development model.

3.2. *Common tools used in Community Based Development models*

When the number of developers involved in a project increases, having a source code managing systems becomes mandatory. Big Open-Source projects like Apache or Mozilla have their own source *"forges"* typically implemented using a Concurrent Version System (CVS) or a SVN [19]. Not so big projects have their source code forges hosted in SourceForge.net or savannah.gnu.org. Designers of SourceForge.net [28] started the portal with the Community Based Development model in mind:

— Minimize administrative work.
— Maximize communications and collaboration.
— Preserve project knowledge.
— Make it easy to establish projects and recruit experts.
— Find and leverage existing code.
— Do all of this on a global scale.

Nearly as important as managing project's source code is managing its bugs. Big projects like Apache use Problem Reporting Databases and Mozilla developed its own problem tracking system called Bugzilla [19]. Smaller projects manage them using communication services of their source forges or by email.

TODO lists are widely used in Open-Source projects, sometimes related with bug tracking and fixing, but mostly to provide a rough idea about what to do next [29], serving as a "map" rather than a "script" which coercively determine actions in a fixed order.

Documentation and communication are the base of collaboration. Project's extensibility depends on a good developers' documentation and fast and easy communication channels. Community Based projects use wiki-pages for collaborative documentation and translation to many languages, blogs of developers (usually syndicated in a "planet", like planetkde.org), mailing lists where decisions are taken, IRC or IM (Gtalk, Jabber, etc.) meetings, or even MUD (Multi-User Dungeon) systems (Globus Toolkit developers use it, really) to decide which will be the next common step.

3.3. *Common features in a WebLab that can be reused*

When developing a WebLab it is obvious that *"one size doesn't fits all"*, but there are some common features which are difficult to develop, need much time to be fine-tuned, or can be developed easily by a distributed group (i.e. internationalization). Those ones are perfect candidates to be implemented collaboratively.

Analyzing several WebLabs [1] we have found some examples of typical features that could be considered:

- Authentication module: nearly every WebLab needs an Authentication module and It is not easy to develop a good one, there are security issues (i.e., SQL Injection attacks), interoperability problems, etc.
- Reservation module: usually a WebLab is a valuated resource that must be well assigned and scheduled.
- Traffic Shapping module: network access speed is often a problem for IT staff and professors (both sides).
- Web Controls module: in each WebLab there are similar hardware (i.e. webcams, oscilloscopes, etc.) that must be controlled by a web application. Browsers' compatibility issues can be fixed in a clean and centralized way, and AJAX can be used in order to minimize network traffic.
- Multilingual Support module: it may be difficult to develop a portable multilingual support module, but at least having a common glossary is a good idea in order to avoid confusing terms and misconceptions.

3.4. *Is my WebLab using a Community Based Development model?*

Developing a software project using a development model is not a binary question, there are many degrees between a community driven project and a totally closed one. These questions may help to analyze if your remote lab is using a Community Based Development model or not:

- Has my WebLab been developed by many people working together?
- Has my WebLab been developed using code of other projects?
- Is my WebLab's code managed using a CVS (Concurrent Versions System) such as subversion (SVN)?
- Do I use a Open Source/Free Software License in my WebLab?
- Is our WebLab's code downloadable freely from Internet by anyone (i.e: at sourceforge.net)?

— Is our WebLab's code is in other organizations / universities / companies and that brings us synergy?
— Do we use a mailing list to manage contributions, installation issues, problems about our WebLab?
— Do we use a wiki-page to manage internationalization ("i18n"), contributions, installation issues, problems about our WebLab?
— Does our WebLab's development team contribute with patches for projects used to develop the WebLab (i.e: contributions to Apache project if WebLab is based on Apache)?
— Every time I need a new feature in our WebLab, do I develop it on my own?

If you have answered most of these questions affirmatively, your WebLab may have been developed under a Community Based Development model, and some typical characteristics of the model can be assumed.

4. **Multilinguality in WebLabs**

Internationalization and localization [30] [36] are complementary processes aimed to adapt a product (mainly software, but also web content as a specific case) to users of any nation or culture. Internationalization, also known as "i18n" (due to the 18 letters omitted), may be seen as the first stage of the process, seeking to prepare the product to serve any user worldwide, or at least users of several nations and/or cultures, but in the way less dependent on these. Localization (known as "l10n" for a similar reason) is the second stage, in which the product is prepared specifically for users of concrete countries, regions and/or cultures [37].

Most of the efforts of internationalization and localization lay in the user interface, but their development is pervasive, that is, it has strong relations with other parts of a software application. This is dramatically true for the specific case of web content, due to its nature, purpose, and universality, so today it is mandatory to consider i18n and l10n as key factors from the very first stages of the overall design of a complex web site.

Internationalization and localization originates from the language point of view, i.e. the need to offer a product in a (widely) known language or in the user native language, but it is important not to think that internationalization is merely "translating a content to English" and that localization is only "translating that content to other languages as Dutch, French, Spanish or Japanese". Now, i18n and l10n involve, apart of the language, many other social, economic, and cultural aspects: date/time calendar and format (including time zone), currency and numbers formatting, measure units, names and titles of people and organizations, addresses and telephone numbers, government assigned identities (such as personal ID, passport, or Social Security numbers), legal issues, and so on.

The key question today is to understand that "language translation" (or better, "multilingualization" or "m17n") is only one of the efforts to be carried out in the process of making available a product to a wide range of users, probably the most expensive of the efforts, but one that must be embedded in the overall design of a product verifying i18n and l10n guidelines [31] [38].

4.1. *Multilingualization*

Thinking specifically of a web site, it is true that the multilingualization task has some particularities. The key issue of a web site (as distinctive of a classical software application, for example) is its typical dynamic behavior: a web site changes more or less continuously, indeed "very" continuously, with frequent addition of new contents/functionalities and/or modification of existing ones. So, it is necessary not only to make a good design of the site, but also to plan carefully the strategy that will cope with the site's dynamic behavior along its life-cycle.

Designing the web site with internationalization and localization issues in mind allows to fulfill most of the requirements at the development stage, and there should not exist significant efforts to be done during the life-cycle of the web site.

But content translation, as the main subtask of multilingualization, extends not only to the development stage, but also to the whole life-cycle stage, and indeed with a significant stress: every new content or either every content change have to be translated to the site's languages. For this reason, and given the expensiveness of the translation efforts, former assertions become almost a law: it is necessary an integrated good design, and it is necessary a site's administration tool supporting, among others, translation task.

4.2. *Standards, platforms, tools*

Given the requirements above mentioned, design and development of a web site should take into account several key technologies. First of all, the use of international standards is almost mandatory, not only for content mark-up, visualization, or dynamic behavior, but also for character sets (UNICODE, ISO 8859, etc.), content encoding (UTF, etc.), languages and locales (ISO 639, ISO 3166, RFC 3066) and so on.

Second, the selection of the web development platform is a key factor from the practical point of view. Obviously, this selection must take into account the main purpose of the web site to be developed, but the fact is that there are web platforms that integrate some or all of the internationalization, localization, and multilingualization issues, and that, in that case, they normally provide support to make easier the administration task to cope with the life-cycle evolution of the web site.

Third, specifically for the multilingualization objective, there are several tools, most notably "gettext" (GNU Project, Free Software Foundation) [32], oriented to the multilingual content management, that allow not only to store and classify texts in several languages, but also to reuse and to obtain little variations of existing ones.

4.3. *The translation task*

Content translation task has specific needs, as it has been said before, because it extends normally to the whole life-cycle of the web site, but also due to the fact that it is a complex task that frequently (if not always) involves the work of "different" people with "different" purposes. Obviously, the grade in which this imposes strong requirements on the web design depends on the nature and purpose of the web site, but it may be interesting to look at the most complex case (a big text-centered multilingual web site, for example, an enterprise or institution) to see the solution.

In a medium or big organization, (web) contents development follows a certain workflow from the stage of writing to the stage of final publication. This workflow is indeed more complex when the contents have to be multilingual. Managing this complexity involves three (almost) orthogonal aspects.

First, it is necessary a basic mechanism of "version control" that keeps track of the different steps of content development, and not only in the original language, but also in the other languages versions.

Second, there are different "roles", or functions, of the people involved: writers who develop original content, translators who do the translation, proofreaders who must validate it, managers who must approve publication.

Third, the contents themselves should be assigned several "properties" informing of their "state", that, on the first hand, take into account the stage of development/translation (with values as draft, translated, revised, approved), but on the other hand, that could reflect different kinds of relations, as publicability (confidential, shared, public) and others. It is important not to confuse state with version control, although there are obviously several similarities: version control should be seen as a low-level mechanism, whereas state is a high-level property.

Given that, the workflow should define the procedures to be carried out for the development of the (multilingual) contents, making the processes to be done explicit for the different agents involved (roles), the conditions and subsequent changes of the contents state, and the mechanisms of communication needed for the tasks.

For that reason, extending that said in former sections, depending on the textual nature of the web site, it may be convenient to select a web development platform that allows for text content development and translation management, or else, at least to have a minimal support tool and a set of procedures that could cope with the complexity of the translation task. As an example for the last, among others, wikis are means of collaborative work that implicitly have some of the functionalities explained, and that can be used with a little control in a structured way to implement the processes involved.

4.4. *The multilinguality test*

The Multilinguality test consists of a list of features that a software should fulfill to support services and resources in more than one language or locale on the Web. See also localizability testing [33] [34] [35] [39] [40].

Basic check list:

1. Was your website planned to be multilingual before it was designed and implemented? Y N
2. Did you analyzed and selected the developing platform bearing in mind localizability? Y N
3. Is it sensitive to user preferences set on the Language-Options menu? Y N
4. Do language symbols follow international standards (ISO 639, etc.)? Y N
5. Does it support input of texts in any language via UNICODE? Y N
6. Does it have a multilingual content management tool (eg. gettext or similar)? Y N
7. Can it cope with date, currency and other language or locale sensitive issues? Y N
8. Does it support version control? Y N
9. Does it allow for different content management roles (writer, translator, proofreader)? Y N

10. Does it account for publicability issues (draft, private, approved, revised, translated, etc.)? Y N

The last 3 requirements refer to workflow.

Conclusion

A WebLab, and any other web based application, has to deal with various issues in order to move up from the state of stable prototype to that of a high quality application. Among these issues we can mention security, accessibility, collaborative development and multilinguality

The security of the WebLab should be taken care of by the IT Service of the University, and it should be considered an essential matter during the design stage of the project. It is important to minimize the number of open ports required to allow communication between the server and the clients, and try to leave the code execution work to the server side of the system.

Accessibility and usability of WebLabs are basic factors that must be taken into account from the first stages in development. Legal, economical and solidarity reasons should lead developers to consider accessibility and usability in order to develop better WebLabs for an even wider audience.

WebLabs, can benefit from collaborative development by having many people working together in the project, and using code of other projects. The potential audience of the system can be increased by using Open Source/Free Software License and by making it downloadable freely from Internet by anyone (i.e: at sourceforge.net). Different technologies like mailing lists or wikis can help with the management of contributions, installation issues, etc.

In today's globalized world, multilinguality is increasingly important, and this issue should be considered from the very first stages of development. Developing platform should be selected bearing in mind localizability. Standard technologies should be used when possible (for example, ISO 639 for language symbols or text input via UNICODE). Other problems to be solved include date, currency and other language or locale sensitive issues.

References

[1] García-Zubia J., López de Ipiña D., Orduña P. "Evolving towards better architectures for remote laboratories: a practical case." *International Journal of Online Engineering*, 2005.

[2] Spafford E.H., Dewdney A.K. “Computer Recreations: Of Worms, Viruses and Core War” *Scientific American,* 1989.

[3] Loscocco P., Smalley S., Muckelbauer P., Taylor R., Turner J., Farrel J. *The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments*. National Security Agency, 1998.

[4] Zimmerli S., Steinemann M., Braun T. “Resource Management Portal for Laboratories Using Real Devices on the Internet.” *ACM SIGCOMM Computer Communication Review*, 2003.

[5] Joshi J., Aref W.G., Ghafoor A., Spafford E.H. *“Security models for Web-based Applications” Communications of the ACM*, 2001.

[6] García-Zubia J., López de Ipiña D., Orduña P., Hernández U., Trueba I. “Questions and Answers for Designing Useful WebLabs”. *International Journal of Online Engineering*, 2006.

[7] Toderick L., Mohammed T., Tabrizi M. “A Consortium of Secure Remote Access Labs for Information Technology Education” *Proceedings of the ACM SIGITE ‘05 Conference*, 2005.

[8] Rigby S., Dark M. “Designing a Flexible, Multipurpose Remote Lab for the IT Curriculum” *Proceedings of the ACM SIGITE ‘06 Conference*, 2006.

[9] Webaim, Web Accessibility in Mind, Utah State University, 1999-2007.

[10] BOCG - Boletín Oficial de las Cortes Generales, Num. 68-13, 3 de julio de 2002, http://www.congreso.es/public_oficiales/L7/CONG/BOCG/A/A_068-13.PDF, 2002.

[11] BOE - Boletín Oficial del Estado, Num. 289, 3 de diciembre de 2003, http://www.boe.es/boe/dias/2003/12/03/pdfs/A43187-43195.pdf, 2003.

[12] WAI - Web Accessibility Initiative (WAI), http://www.w3.org/WAI/, 1994-2006

[13] WAI - Web Accessibility Initiative (WAI), Alternative Web Browsing, http://www.w3.org/WAI/References/Browsing#4, 1994-2006a.

[14] WAI - Web Accessibility Initiative (WAI), *Complete List of Web Accessibility Evaluation Tools*, http://www.w3.org/WAI/ER/tools/complete.php, 1994-2006b.

[15] Nielsen, J., *Papers and essays by Jakob Nielsen*, http://www.useit.com/papers/, 1995-2007.

[16] Nielsen, J. *Heuristic evaluation*. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods. John Wiley & Sons, New York, NY, 1994.

[17] Paulson J.W., Succi G., Eberlein A., “An Empirical Study of Open-Source and Closed-Source Software Products” *IEEE Transactions on Software Engineering*, vol. 30, no. 4, 2004.

[18] Mockus A., Fielding R.T., Herbsleb J., “A Case Study of Open Source Software Development: The Apache Server”, *Proc. 22nd Int’l Conf. Software Eng*., 2000.

[19] Mockus A., Fielding R.T., Herbsleb J., “Two Case Studies of Open Source Software Development: Apache and Mozilla” *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, 2002.

[20] Samoladas I., Stamelos I., Angelis L., Oikonomou A., “Open Source Software Development should strive for even greater code maintainability”, *Communications of the ACM*, October 2004/Vol. 47, No. 10, 2004.

[21] Domino M.A., Webb Collins R., Hevner A.R., Cohen C.F., “Conflict in Collaborative Software Development”, *SIGMIS Conference ‘03*, Philadelphia, Pennsylvania, 2003.

[22] Wheeler S., Wheeler D., “Why Open Source Software/Free Software (OSS/FS)? Look at the Numbers!”, http://www.dwheeler.com/oss_fs_why.html, 2003.

[23] O’Reilly T., “Lessons from Open-Source Development Model”, April 1999/ Vol. 42, No. 4 *Communications of the ACM*, 1999.

[24] Dalle J., Jullien N., “Windows vs. Linux: Some Explorations into the Economics of Free Software”, *Proc. Acts of SSII Conf.*, 2000.

[25] Cox A., “*Afternoon Keynote - Software Quality and Open Source”*, *Linux-World Conference*, 2006.

[26] Raymond E.S., *The Cathedral and the Bazaar*, http://www.catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/, 2003.

[27] Morkel Theunissen W.H., Boake A., Kourie D.G., “In Search of the Sweet Spot: Agile Open Collaborative Corporate Software Development”, *Proceedings of SAICSIT 2005*, Pages 268-277, 2005.

[28] Augustin L., Bressler D., Smith G., “Accelerating Software Development Through Collaboration”, *ICSE’02*, Orlando, Florida, 2002.

[29] Yamauchi Y., Yokozawa M., Shinohara T., Ishida T., “Collaboration with Lean Media: How Open-Source Software Succeeds”, *CSCW’00*, Philadelphia, PA, 2000.

[30] Wikipedia. *Internationalization and localization*. http://en.wikipedia.org/wiki/Internationalization_and_localization (visited Jan. 2, 2007)

[31] LISA. *Global Information Management Metrics eXchange (GMX)*. 2004. http://www.lisa.org/archive/newsletters/2004/4.1/zydron.html

[32] Free Software Foundation. gettext. http://www.gnu.org/software/gettext/ (visited Jan. 2, 2007)

[33] Localisability testing - Microsoft test: http://www.microsoft.com/globaldev/getWR/steps/testing/test_loc_test.mspx

[34] Localisability testing - Mozila test: http://www.mozilla.org/docs/refList/i18n/l12yGuidelines.html

[35] Localisability testing - Plone test: http://plone.org/documentation/manual/plone-developer-reference/patterns/localisability

[36] Glossary on multilinguality http://www.wordforge.org/static/guide-glossary.html

[37] LRC. *Advanced course on Localization*. 2001. http://www.localization.ie/resources/courses/summerschools/2001/Advancecourse.htm

[38] Ishida, Richard. *W3C Internationalization Workshop Position Statement*. 2002. http://www.w3.org/2002/02/01-i18n-workshop/Ishida.html

[39] Saikali, Matta. *Internationalization & Localization Testing Successfully Testing Multilingual Software And Web Sites*. http://www.i18n.ca/workshops/ILT_overview.htm (visited Oct. 26, 2006)

[40] Vine, Andrea. *Internationalizing Software Testing*. Multilingual 15-5: 25-30. 2004. http://developers.sun.com/dev/gadc/des_dev/Intl_Testing.html