

Achieving Universal Web Access through Specialized User Interfaces

Giorgio Brajnik¹

Dip. di Matematica e Informatica
Università di Udine, Italy

T: +39 0432 558445
www.dimi.uniud.it/giorgio

Abstract. The paper discusses how accessibility helps in extending the range of goals that users (disabled or not) can achieve, and how it fails to increase user bandwidth for achieving those goals. The paper then illustrates how transcoders that satisfy a set of requirements, can effectively deal with the problem of generating, on the fly, specialized user interfaces that would support a more universal web access leading to a greater bandwidth. An informal analysis of a commercially available text-transcoder is used as an example of what can be done.

1. INTRODUCTION

Text-only web pages are sometimes perceived as second-level web pages, “ghettos” for less able web visitors. In this paper I argue that appropriately built text-only pages are examples of specialized user interfaces that can improve usability of a web site and constitute a valid alternative to the original web site. These pages can be beneficial not only to individuals using screen readers (or other assistive technology), but also to people using small screen devices and/or mobile devices.

There exist at least three definitions of web accessibility: accessible web sites are such that *individuals with disabilities can access and use them as effectively as people who don't have disabilities* [Slatin and Rush 03], or such that *more people can use them effectively in more situations* [Thatcher et al. 02], or *perceivable, operable and understandable by the broadest possible range of users and compatible with their wide range of assistive technologies, now and in the future* [W3C/WAI 03].

Especially with the third definition, accessibility entails interoperability. Interoperability means that the web site is capable of interacting with various devices in such a way that information can be exchanged and used. Often interoperability refers to small and mobile devices, like Personal Digital Assistants or cellular phones. But interoperability is exploited also because an accessible web site is more easily indexable and searchable by search engines that benefit of absence of barriers like images without textual descriptions, navigation bars without textual labels, links implemented only in Javascript.

To understand why to date so few web sites are accessible, the benefits of accessible web sites have to be weighed against the difficulties that are faced by web developers. There are at least two scenarios to be considered: developers having to *retrofit* accessibility to a live web site and developers *creating new* web pages, free of legacy constraints.

¹ Scientific advisor for UsableNet Inc., manufacturer of LIFT Text Transcoder, a tool mentioned in the paper.

In the latter case the main problem is lack of skills (on technical issues), since *good design is accessible design* [Slatin and Rush, 03]: if appropriate care is taken at design time, accessibility comes mostly for free.

In the retrofitting case, I argue that the main difficulties in implementing accessible web sites are due to lack of skills, time, and money. Skills span technical issues and management practices, as accessibility is not a target but rather a process (see [Brajnik, 04] for an analysis of some of the issues related with quality models and tools that can be used for retrofitting web accessibility). Time and money are important factors because accessibility defects vary in terms of how easily they can be identified, solved and tested. Fixing a missing ALT for an image is much easier than fixing a page layout that is not liquid.

Besides these difficulties, a web developer has often to cope with environmental issues that include fast release cycles (changes are conceived, designed, implemented and tested in matters of hours), missing or incomplete specifications and guidelines to stick to, a rapidly changing environment (new technologies, languages, formats) and an increasingly tight coupling of web sites with other processes and systems (internal content management systems on one side and external agents on the other side).

Accessibility, since it entails interoperability, is a way for reducing the constraints due to rapidly evolving client-side technologies that use the web site: diverse graphical browser coupled with diverse screen readers and magnifiers, text-only browsers, reading browsers, proxies and transcoders/mediators, testing and analysis tools, micro-browsers on PDAs or cellular phones. Because an accessible web site sticks to standards and because it makes structural aspects of the information or interaction explicit, it automatically tolerates better these changes.

Accessibility, however, is not an optimal solution for web sites that have to be available to anybody, anytime, anywhere and even in a personalized manner. Accessibility requirements (given the current status of technology) may get in conflict with other requirements, making it even more difficult for the developer to produce high quality web sites. For example, accessibility conformance with respect to WCAG 1.0 level A [W3C/WAI 99] requires that Javascript is not essential for the usage of the web site (for example because some textual browser and micro-browser are not Javascript-capable). Yet careful addition of Javascript code on a web site can improve usability of its features.

In this paper I want to discuss when one should decide that an accessible web site is sufficient for its audience, and when instead one should go beyond accessibility and develop specific versions of the web site to suit different types of users in different situations. I claim that technologies based on transcoders (aka proxies or mediators) provide a strong support in developing these specialized interfaces.

2. LIMITS OF UNIVERSAL DESIGN

2.1 Universal Design and Accessibility

Universal design is a school of thought that suggests that the design of products and environments be usable by all people, to the greatest extent possible, without the need for adaptation or specialized design [CUD 97]. It encompasses 7 principles, some of them are directly relevant to web accessibility.²

² Namely:

- the design is useful and marketable to people with diverse abilities (n. 1)
- The design accommodates a wide range of individual preferences and abilities (n. 2)
- Use of the design is easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level (n. 3)
- The design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities (n. 4)

However there are limits to the extent to which these principles can be pushed without reducing the quality of use³ of the designed artifact.

A number of problems are faced by disabled persons when accessing web sites, even when those web sites are conforming to accessibility guidelines. In a study with disabled persons [NNG, 01] authors report that for users of screen readers or magnifiers the success rate is 6 times smaller than that of other people, that time on task is about twice as much, and that errors are 3 to 8 times more frequent.

Similar results were shown by a study on seniors (65+) [NNG, 02]: success rate for them is 30% smaller than for non seniors non disabled people, time on task is roughly twice as much, and errors are about 8 times more frequent.

2.2 INTERACTION OF DISABLED USERS WITH ACCESSIBLE WEB SITES

In the following, with the expression “disabled users” I will mean individuals with sensory or motor disabilities and individuals using technology that disables them, like using a gray-scale PDA with no Javascript capabilities.

Compare now two versions of a web site: a non-accessible one (let's call it “N”) and its accessible counterpart, called “A” (for example the versions of a web site before and after a retrofitting process).

Web site A fulfills the same visitors' goals as N, but it tolerates disabilities that they may be subject to. Web site A contains redundant information (for example textual descriptions of images and textual links redundant with Javascript-based navigation bars) that cater for alternative interaction channels and styles (for example via audio through a screen reader). Since web site A supports the same set of goals (as N), it offers the same content (information or functionalities) to all its visitors, with no exception. To disabled users, web site N supports only a subset of the goals that are supported to non-disabled visitors. Some information items are not reachable; others cannot be perceived; some functionalities cannot be operated.

If we now move from content to architectural aspects (i.e. pertaining to information architecture of the site and task models supported by the web site [Rosenfeld and Morville, 98; Hackos and Redish, 98]) and consider only disabled users, the set of tasks supported by A is extended when compared to the tasks supported by N, because more goals can be achieved. But tasks also change because of different interaction styles (for example, interaction with a screen reader is modal), because additional actions are needed, and because affordance of controls changes. For example, link text is spoken with a different voice, and the user has a limited time interval for pressing enter to activate the link. To a screen reader user, the task of exploring a web page includes new steps like skipping navigation bars, selecting the appropriate frame, navigating through cells of a table.

Web site N does not support adequately these tasks: they become more difficult to perform or even impossible for disabled users.

But inability to achieve certain goals is not the only disadvantage that disabled users have to face. Even with an accessible web site, their user bandwidth⁴ decreases when compared to non disabled users. Tasks become more difficult (for example because of additional navigation steps required to access some information item in a hierarchical navigation

-
- Appropriate size and space is provided for approach, reach, manipulation, and use regardless of user's body size, posture, or mobility (n. 7).

³ Quality of use is the effectiveness, efficiency, satisfaction and security with which specified users can achieve specified goals in specified situations (ISO 9126-1, 2001).

⁴ User bandwidth refers to the information processing rate for a human input or output channel [MacKenzie, 91; Mankoff et al., 02]. Bandwidth is measured in bits/seconds and depends on task difficulty (measured in bits) and time to complete the task. The higher the bandwidth is, and the more effective the interaction is.

between frames). It also decreases because presentation aspects may slow down the interaction (for example scanning a page visually is much faster than scanning it via a screen reader or magnifier).

Therefore with an accessible web site a disabled visitor has to cope with a lower bandwidth than a non disabled user (see figure 1). An accessible web site features the same utility for disabled and non disabled users, a smaller bandwidth for disabled users, and an increased utility for disabled users than the non-accessible web site.

Therefore, only if such a decrease in user bandwidth and associated lower quality of use are tolerated, then an accessible web site in the sense of Universal Design is a proper solution. Otherwise, in addition to making the site accessible, other solutions have to be deployed.

<p>Disabled users of N compared to disabled users of A</p>	<ul style="list-style-type: none"> - subset of goals can be achieved - subsets of tasks can be performed - tasks are more difficult - more errors - more time <p>less utility & smaller bandwidth</p>
<p>Disabled users of A compared to non disabled users of A</p>	<ul style="list-style-type: none"> - same goals can be achieved - tasks are more difficult - more errors - more time <p>same utility & smaller bandwidth</p>
<p>Non disabled users of A compared to non disabled users of N</p>	<ul style="list-style-type: none"> - superset of goals can be achieved - tasks are simpler - fewer errors - less time <p>same utility & greater bandwidth</p>

Fig. 1: changes in quality of use for disabled/non disabled users when using an accessible/non accessible web site

2.3 Specialized user interfaces

Imagine a situation where for a web site there are several different interfaces, each one catered towards users of different devices. For example a text-only version of the site, an i-mode version, and a PDA one, coupled to the graphic one. Assume also that some way for repurposing the content of the web site is in place so that the different interfaces can all be based on the same content and automatically updated when it changes.

Consider now a PDA user, browsing the site with the appropriate interface. Such a version of the web site, to be usable and avoid overloading the small PDA screen, will probably support only a subset of the goals of the graphic web site (such as directory access, access to the most important content, access to the things that are most relevant to mobile users). Even when the same goals are supported (and hence the same content is shown by the PDA oriented web site), they are achieved by different tasks, as the structure of the interaction will differ. Also the presentation aspect will change, due to the small screen of the device, to its inability to execute certain code, and to the different interaction style and modality.

For example a user of the Blazer browser available on Palm OS devices won't be able to detect links within image maps (i.e. there is no affordance for such commands), he/she won't be able to detect and follow hidden links (even when images are not downloaded, Blazer does

not show the ALT of spacers). In addition the Blazer user cannot use anything that is based on Javascript, let alone Java applets or Flash files, and no keyboard accelerators are available, even though pages contain `accesskey` attributes.

User bandwidth can decrease in this case too:

1. Task difficulty is in general increased (more commands), unless the specialized interface supports only a subset of the possible goals (i.e. unless only some of the content/functionality of the graphic site is offered by the specialized one). In fact the reason for simplifying the page and web site is precisely to reduce task difficulty.
2. Time to completion is in general increased, unless supported tasks and goals are a subset of the original ones, and the amount of data sent to the device (which might suffer from a poor communication bandwidth and poor processing power) is also reduced.
3. Chances of mistakes and slips are also increased, unless tasks are appropriately simplified and the presentation is such that proper affordances are provided.

Figure 2 shows the changes in quality of use for disabled users when using an accessible web site compared to a specialized user interface of it.

<p>Disabled users of specialized UI compared to disabled users of A</p>	<ul style="list-style-type: none"> - subset of goals can be achieved - same/lower task difficulty - same/fewer errors - same/less time <p>less utility & equal/greater bandwidth</p>
---	---

Fig. 2: changes in quality of use for disabled users when using an accessible web site or a specialized user interface of the same web site

In conclusion, a specialized interface is justified (compared to deploying a single accessible web site) when the web site content and corresponding tasks can be reduced, simplified and shaped in a way that is appropriate to the specific device. Only in such a case there will be no decrease in user bandwidth. In other terms, we are trading utility (achievable goals) with bandwidth. Accessibility leads to greater utility, but not necessarily to greater user bandwidth.

3. POSSIBLE FORMS OF SOLUTIONS

Appropriate technology has to be made available to web developers to support the development of specialized interfaces without incurring in the costs of re-developing the web site (or part of it). Ideally such a technology would take as input a model of the goal/task, a description of the target device, the original content, and would automatically produce the new interface.

At the moment it is not viable to locate this kind of software on board of client devices (PDAs, phones): limits in computing power, available memory and communication bandwidth hinder such a development.

Recent versions of CSS (Cascade Style Sheets), that can be used to control presentation and to filter content of the web page, should also be assimilated to client-side technology, and share its limits.

The world of web servers, application servers and content management systems can be the arena where such a technology could exist (for example see [Hori et al., 03]).

A similar solution, but independent from web servers, is based on transcoders [Takagi, 01;

Brown and Robinson ;, Hori et al., 03; Parente, 03]. A transcoder acts as a proxy (by sitting between a browser and a web server) and performs on-the-fly transformations on pages that are required by the browser and produced by the server.

Provided that the transcoders can be given a model of goals and tasks to be supported in the specialized interface they:

- are a modular solution, as they are independent from web servers, application servers or content management systems
- can be implemented on powerful servers offering all the computing resources that are needed to perform a quick transformation of the web pages
- can be adaptive to the target device and browser, and hence send only the data that is needed, and use the appropriate presentation.

For these reasons transcoders are a viable solutions for facing many of the issues that web developers have to cope with when they want to design specialized user interfaces (as discussed in the introduction).

3.1 Related work

Paternò and colleagues [Paternò and Santoro, 02; Mori et al., 02] describe an environment and notation for specifying abstract task models and for automatically generating one or more user interfaces supporting those tasks, each targeted towards a specific kind of device.

This is definitely an important research direction to pursue. And it is highly relevant to the problem discussed in this paper as it would allow the web developer to specify abstract models for the tasks and then let the system automatically generate the appropriate interface and coding suitable for the device.

However this approach requires that the task remain essentially the same in all the user interfaces being generated since only its presentation is allowed to change (for example what is normally presented as a set of radio buttons might become a drop-down menu; the task changes as the actual actions needed have to change correspondingly, but the deep structure of the task remains the same). When dealing with assistive technology or with small screen or mobile devices, tasks and goals are likely to change more significantly if the same user bandwidth is to be achieved.

In addition, given the current practice among web developers, it is unlikely that they are willing to delegate the entire generation of the user interface and of the look & feel aspects of the pages to an automatic tool.

VAQUITA [Bouillon and Vanderdonckt, 2002] is a system capable of *retargeting* a web page to different computing platforms. Retargeting (in this case) means to statically analyze a web page, to automatically derive an abstract user interface (for example by inferring the existence of an abstract object called “RadioButton”), to transform such an abstract interface into the abstract interface for another platform (for example on a WAP device the “RadioButton” object is mapped into a “Listbox”), and finally to generate an appropriate and running user interface on the selected platform.

This model-based approach for interface generation (the model is based on the abstraction, transformation, and generation rules) is another important way for dealing with the problem of generating appropriate specialized interfaces. Its potential stems in its ability to store in the abstract model a number of different mappings of interface components and possible constraints that otherwise need to be specified each single time. However there are still important practical issues that in the mentioned study appear to have been dealt only in part and that hinder practical utilization of tools like VAQUITA. The tool applies statical analysis on web

pages, and requires manual decisions to be taken by a web designer, and therefore it cannot be used to transform on-the-fly web pages or web pages that are dynamically produced. Secondly, for performance reasons the number and types of processing steps performed by the tool (assuming that it has to produce web pages on-the-fly) have to be rather limited in complexity.

Mankoff and colleagues [Mankoff et al., 02] discuss the problem of web accessibility for people with severe sensory or motor disabilities. They present a number of approaches that can be followed for making the web site accessible to these users (for example, by providing logical control through keyboard shortcuts, by providing automatically scanning interfaces, by wrapping in some modality part of the page so that limited human output signals can be overloaded with other meanings). They propose a prototype of a browser that is capable of being operated with two binary signals. They also propose a transcoder that adds large buttons for navigating within the page, that enlarges and highlights available links, that adds “Back” and “Forward” buttons to the page, and that adds the title attribute to the links in the page after pre-fetching the destination page and using its title.

The transcoder appears to be still a proof-of-concept prototype, as some of the functionalities are not yet completed. Feasibility and effectiveness of adopted techniques are also still to be empirically evaluated. For example, adding the title attribute with information automatically gathered from destination pages requires that the transcoder fetches those pages, dramatically reducing its efficiency. Other solutions, potentially useful also for mobile device users, are based on Javascript code. But Javascript might not be available on the micro-browsers installed on board of such devices.

Parente [Parente, 03] discusses the problem of maintaining annotations⁵ despite changes that the original pages may be subject to. The author claims that if original pages are dynamic then annotation robustness is an issue. We believe this is an issue only if annotations are associated to the pages they refer to in such a way that references to temporary features of the pages are made. If annotations are hooked to the pages by referring to features that do not change when the page content changes (for example layout features), then even though the page content is dynamic, no change is required to maintain robustness of the annotation.

The author explores the algorithmic problems related with the ability to automatically detect which changes in the original page do not make the annotation invalid. He frames this problem as a problem of matching tree-based data structures.

Access Gateway [Brown and Robinson, 01] is a transcoder intended for low vision people. The authors present a list of requirements of which section 3.1 is an extension. Access Gateway has a very large number of user preferences that determine the kind of transformations that the transcoder does on pages. However there is no easy way (other than apparently writing extension code in C++) to customize the way in which the transcoder processes the page content, which is a critical requirement in our view.

A framework of annotations authoring tools is discussed in [Hori et al., 03]. Annotations are classified as being formal or informal, tacit or explicit, embedded in the document or external ones. Depending on the type of annotations different requirements are discussed for authoring environments. Authors assume that annotations are hooked to relevant documents via XPath expressions and they discuss the problem of annotation robustness. They also present the user interface of a tool for authoring annotations that supports a develop-by-example strategy.

Asakawa and Takagi [Asakawa and Takagi, 01] discuss a transcoder that uses a database of

⁵ An annotation is a remark attached to a particular portion of a web page. Annotation-based transcoders use the information represented in annotations to determine how the resulting page should be produced [Hori et al., 03].

annotations developed by sighted users. The idea is that by providing a sufficiently easy system to use, sighted users can improve accessibility of a web site while navigating it by adding their own custom annotations. However only limited experimentation has been carried out with this idea and, at the moment, it is not clear if it really works.

The same authors in [Takagi and Asakawa, 01] discuss a transcoder that performs a number of automatic transformations (i.e. not determined by available annotations). The transcoder removes banners, images, and layout components; it adds ALT text by extracting relevant information from destination pages for links. It also simplifies the page by removing those elements that are also available in neighboring pages (authors don't provide a detailed description of what they call the differential of a page).

The work reported by the last two papers seems to require additional development and experimentation as little data exists at the moment for determining effectiveness and feasibility of these solutions.

3.2 Generic requirements for transcoders

Transcoders that generate specialized interfaces should satisfy at least the following general requirements:

Efficiency The transcoder should be very efficient, in order to avoid increasing too much the task completion time and reducing too much user satisfaction and effectiveness. Because a transcoder requires additional transfers of data through the net (the request comes from the browser, it is relayed to the server, the answer is acquired from the server, and the processed page is sent to the browser), it has to be carefully designed and installed so that these data transfers do not affect too much its response time.

Decoupling of servers The transcoder should be decoupled from the web servers it interacts with, in order to guarantee an appropriate level of modularity in the architecture of the web site.

Re-purpose of content The transcoder should be able to reuse all the content that is available in the original page, and such a content should be modifiable by the transcoder on the basis of some description of the goal/task model.

Appropriate output The transcoder has to produce a user interface for the web site that is appropriate for its users. This entails also plasticity [Thevenin and Coutaz, 99], i.e. the ability of the system to produce a user interface that is adapted to the device being used and possible context of use. This means for example that the transcoder has to detect the device being used and appropriate code has to be produced, for example by recoding the page in cHTML or WML.

Customizability The transcoder has to be customizable by the web developer who must specify which goals should be simplified, and how corresponding tasks should be shaped. Annotation-based transcoders are a viable solution for specifying these properties, provided that the following sub-requirements are met:

Decoupling of content Annotations should not be stored within web pages, but they should only refer to web page. Otherwise there would be no decoupling between server and transcoder.

Robustness Annotations have to be written in a way that they tolerate changes in the pages they refer to.

Modularity Annotations should be applicable to individual fragments of web pages and effects of more than one annotations should combine in producing a new version of a page.

Re-usability Annotations should be applicable to more than one web page.

Content preservation Annotations should be designed in such a way that they re-purpose already available content. They should not provide new content, unless it is missing in the original web page.

In addition, an appropriate development environment is needed. At minimum such an environment should:

1. support the developer in writing new annotations
2. support modification and debugging of annotations
3. support monitoring of application of annotations (to make sure that whenever new content or new pages are added to the web site, the annotations still work properly)
4. support testing and diagnosis of malfunctioning annotations.

Adaptability The user interface produced by the transcoder should be customizable by the end user, for example by supporting resizing of text, changing of colors, moving navbars to the end of the page, switching between a hierarchical and simultaneous navigation of frames.

Adaptivity The transcoder should consider the past behavior of the user, or of other users, and include appropriate features. For example, it might propose the three most frequently used outgoing links from the page (based on the web site traffic in, say, the previous two weeks).

Interoperability The transcoder should appropriately interact with other technologies that are normally used on web sites. For example cookie management, security management, form filling.

4. AN INFORMAL ANALYSIS OF LIFT TEXT TRANSCODER

LIFT Text Transcoder (LTT) is a commercial transcoder [Usablenet, 04c] that produces on-the-fly text-only pages suitable for disabled users. It can be customized by the web developer via formal annotations. When it's used to navigate within a web site, it automatically converts all the outgoing links so that the user keeps navigating through it.

LTT is adaptable by the end user, who can easily change the way in which text is shown, which colors are used, how big links are, whether frames should be shown simultaneously or hierarchically, whether navigation bars should be moved to the end of the page or not.

LTT is based on three main processing steps. The first one queries the web server and obtains the HTML document requested by the browser. A second step then takes as input the DOM (Document Object Model) of the page and applies to it all available and applicable annotations. Finally a rendering step takes care of omitting all those elements that should be excluded, rewriting the code of the page, laying out properly the remaining page content, and producing the final document for the browser that requested it.

LTT implements a number of built-in transformations of the web pages. What follows is a summary of [Usablenet, 04d].

- transcoded pages are stripped of all the original CSS and Javascript code, and simple CSS rules are added that yield a liquid design of the page and of the text;
- automatic redirects are replaced by server-side redirects;
- textual links that are too close each other are separated by white space;
- scripts and event handlers are removed; when appropriate they are replaced with content of NOSCRIPT clauses;
- frames and framesets are replaced by corresponding contents; frames can be viewed simultaneously;
- layout tables are linearized, whereas data-tables are preserved;

- images are replaced by their ALT or TITLE; imagemaps are replaced by lists of links;
- forms are linearized and labels are properly positioned (provided they are associated to their controls); form control titles are also used as labels if appropriate;
- objects and applets are replaced with their textual equivalent if any; they are stripped otherwise.

keep in touch

Become a [OneTouch Gold member](#) today!

AW 054-579B Text-Only

[Company](#) | [Products](#) | [Diabetes Care](#) | [Promotions](#) | [Professionals](#) | [Feedback](#) | [Around the World](#)

[Site Map](#) Search:

U.S. Flag Although LifeScan.com may be helpful to our worldwide customers, this site is governed solely by applicable governmental regulations in the United States.

By using this site, you agree to our [Legal Notice](#) and [Privacy Policy](#).

Contact us at CustomerService@LifeScan.com 800 227-8862

Web Site Questions? Contact our Webmaster@LifeScan.com

© [LifeScan, Inc.](#) 1996-2004.

Fig. 3: a fragment of a transcoded page (www.lifescan.com) showing how LTT moves navigation bars across the page and preserves their content. In fact the search box and the navigation bar are originally located at the top of the page. The original navigation bar is also represented as a TABLE containing graphical links that LTT transforms into a DIV

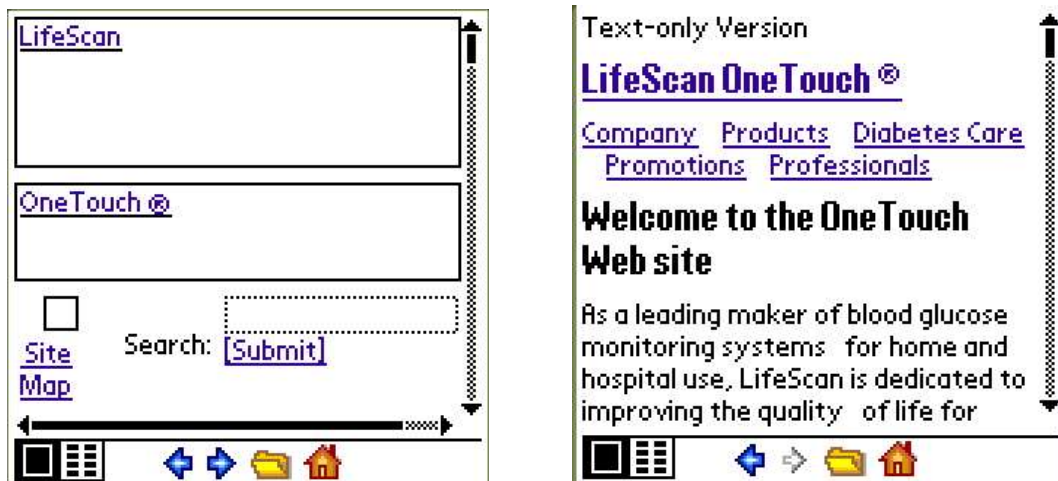


Fig 4: Two screenshots of a page (www.lifescan.com) viewed through Blazer on PALM OS devices; on the right LTT produces a more information-rich page than (left) the original page viewed directly with Blazer (with no images being loaded). No specific annotation for Blazer or PDA is used in this case and no plasticity is in effect.

Annotations are formal XML fragments, stored in files that are separate from the original web pages and owned by the administrator of LTT. Each annotation refers to an element type

(i.e. a tag name) of the DOM of the document (the target of the annotation), it can be page-specific (i.e. restricted to a single URL) or site-wide, it has a match condition specified through an XPath expression, and a transformation specification section that can include XSLT expressions. The transformation section says how to transform target elements matching the expression and belonging to documents on which the annotation applies.

Transformations include the ability to add, remove or replace elements attributes or elements content. Other transformations can be used to specify the linearization order of a layout table, or to move the element elsewhere in the page (before or after other elements). See the example shown in figure 5.

```
<annotation for="table">
  <description>Move side navbar to the bottom of the page</description>
  <match>../@background='/images/bg_sidenav_blue2.gif' and
  un:elt_equiv-length(1)!=0</match>
  <execute>
    <replace-element name="div" keep-content="no" />
    <set-content>
      <xsl:for-each select="descendant::a">
        <xsl:copy-of select="." />
        <xsl:if test="position()=last()"> | </xsl:if>
      </xsl:for-each>
    </set-content>
    <move-bottom pos="1" />
  </execute>
</annotation>
```

Fig 5: Example of annotation that applies to a navigation bar implemented through a TABLE that has a certain background image. The annotation reformats the navigation bar in order to render it as a DIV containing a horizontal list of links, and movable to the bottom of the page (copied from [Usablenet 04a]). This annotation is used to produce the output shown in figures 3 and 4 (right).

According to the model described in [Bouillon and Vanderdonckt, 2002] LTT performs transformations between “final interfaces” and “concrete interfaces”. No higher abstraction levels are considered. The main reason is to avoid penalizing its runtime performance.

LTT satisfies most, but not all, the requirements listed in section 3.1:

Efficiency LTT does not process (hence it does not even acquire) images, Javascript files, CSS files, applets, or Flash files. Therefore the pages it produces are very often much smaller than the original ones.

In the limited performance evaluation described in [Usablenet, 04b], the overall response time of LTT (in the worst case - considering network latency time) ranges between 82% and 138% of the response time of a direct connection between browser and server. Notice that values below 100% mean that the time required to get and render a page through LTT is smaller than the time required to get it directly from the web server. This is possible because of the reduced size of the text-only page that is sent to the browser.

Decoupling of servers LTT, being based on the HTTP protocol, is independent from web servers, web application servers or content management systems being used to deploy the web site.

Re-purpose of content LTT by default re-purposes all the text and HTML content of the page. The only content that cannot be re-purposed by LTT is what gets automatically removed (images, objects, CSS, Javascript).

Appropriate output If appropriately customized, LTT is capable of producing web pages that are potentially more accessible and usable by disabled persons than the original ones. Expected advantages of deploying LTT are:

- the liquid layout of the page, the re-sizable text and availability of different color modes mean that low vision persons and users of small screen devices can benefit;
- large buttons can be benefited by low vision and low mobility people;
- image and object filters can be benefited by no vision people, and by those using low communication bandwidth devices;
- preservation of layout of data tables can be benefited by low and no vision people;
- linearization of layout tables and forms can be benefited by low and no vision people, and by users of small screen devices;
- skippable navigation bars, movable navigation bars, hierarchical vs. simultaneous frame navigation and addition of new page headings can be benefited by low and no vision people, by mobility impaired users, by users of small devices and users of limited communication bandwidth devices.

See figures 3 and 4 for examples on how LTT renders pages with movable and reformatted navigation bar. It's worth noting that the more accessible the web site is and the more effective the transformed page produced by LTT is. Obviously, in no way LTT can automatically add appropriate content or fix possible defects. Unless it is properly customized with annotations, the increase in accessibility can only be achieved by removing part of the content of the pages.

LTT does not offer plasticity, in the sense that it is not able to detect the device being used and deliver appropriately coded pages, nor does it any retargeting.

Customizability Pages produced by LTT depend on the original ones, on applicable annotations and on user preferences. The web developer can define specialized interfaces only by writing annotations that specify how interface elements of web pages are to be transformed into other interface elements. Differently from the approach proposed in [Paternò and Santoro, 02] there is no description of abstract task or interaction models. Web developers thus are only required to state the differences in content and architecture that they want in the UI delivered by LTT. Most of the transformations are built-in, and the remaining ones have to be framed as changes of HTML (and coded in the XML-based annotation language).

Decoupling of content Annotations are external files, totally decoupled from the web pages they refer to.

Robustness Annotations are not robust, in the sense that if the original page changes so that the annotation target or match condition does not apply any more, or the representation of the content of the target element changes, the appropriate transformation is not carried out. This is a complex problem [Hori et al., 03; Parente, 03] whose solution requires sophisticated and still unreliable tree-matching algorithms. I believe that, for the sake of transcoder efficiency and quality of resulting pages, at the moment it is better to focus on supporting appropriate quality assurance processes during annotation development or maintenance. Appropriate tools and methods for supporting monitoring, diagnosis, fixing, regression testing of annotations have to be deployed to make sure that as the original pages changes, all the required annotations work properly. See [Brajnik, 04] for more details.

Modularity It is up to the developer to write annotations that are as modular as needed. Transformations specified by annotation can be combined in two ways: they can be carried out independently, if they refer to different sections of the page; and they can be carried out sequentially, when the input of an annotation depends on the output of a different one. This is achieved through a pipeline process within the annotation engine of LTT.

Re-usability Annotations can be reused several times within the same web page (for example to drop a useless ALT for a decorative image) or within the entire web site.

Content preservation Annotations can preserve the original content of the page. In general the annotation developer can use any XPath expression to extract any available information from the DOM of the page. In addition, he/she can extend these expressions by writing appropriate Java code to process other sort of information (for example to extract URLs from Javascript code).

Adaptability LTT can be customized by the end user.

Adaptivity LTT is not adaptive at all.

Interoperability LTT copes with some of the issues regarding cookie management, secure connections and form submissions. Limits are client-side form validation that is not performed, as LTT does not process Javascript, and correct management of complex cookie-based mechanisms.

5. EVALUATIONS

A number of experimental evaluations have been planned at the moment aimed at determining the quality of use of LTT with respect to disabled users. More specifically, comparative usability experiments will be performed, with respect to a control group using a web site through graphical browsers, by samples of users that:

- have low vision or no vision at all;
- have motor disabilities;
- use Blazer or similar browsers on PDA devices.

The experiments should determine if users under any of these conditions are more effective in using a web site through LTT than the same users in directly using the web site. Each experiment will report number of errors (use of “Back” button, visit of irrelevant pages, failed tasks), task completion time, n. of visited pages, n. of completed tasks, task completion rate and user satisfaction.

Preliminary analysis of some of the data gathered so far with visually impaired users indicates that LTT does not worsen the quality of use of tested web sites. In some of the cases LTT actually improves quality of use; in other cases its expected benefits are obfuscated by the large variability that exists in the ability of experimental subjects in using assistive technology and browsers.

6. CONCLUSION

The paper discusses why accessible web sites may not be the optimal solution for universal web access by individuals with sensory, motor or technological disabilities.

It argues that task difficulty, number of errors, task completion time and user bandwidth worsen considerably for these users. Only the development of specialized user interfaces for the same content of the web site can improve user bandwidth.

Transcoders are a viable solution, and this is argued by analyzing the capabilities of LIFT Text Transcoder, a commercially available tool.

REFERENCES

- [Asakawa and Takagi 01] C. Asakawa and H. Takagi. Transcoding system for non-visual web access (2): annotation-based transcoding. In *16th Int. Conf. on Technologies and Persons with Disabilities (CSUN2001)*, 2001
www.csun.edu/cod/conf/2001/proceedings/0086asakawa.htm.
- [Brajnik 04] G. Brajnik.
Using automatic tools in accessibility and usability assurance processes.
8th ERCIM Workshop "User Interfaces For All", June 2004, Vienna, Austria. To appear.
www.dimi.uniud.it/giorgio/publications.html#qapro
- [Bouillon and Vanderdonck, 2002] L. Bouillon and J. Vanderdonck. Retargeting web pages o other computing platforms with VAQUITA. Proc. of IEEE Working Conf. on Reverse Engineering WCRE' 2002, Richmond, Oct 2002; van Deursen and A. Burd eds., IEEE Computer Society Press, Los Alamitos, pp. 339-348.
- [Brown and Robinson 01] S. Brown and P. Robinson. A world wide web mediator for users with low vision. In *CHI 2001 Workshop n. 14*, 2001.
www.ics.forth.gr/proj/at-hci/chi2001/files/brown.pdf.
- [Hackos and Redish 98] J. Hackos and J. Redish. *User and task analysis for interface design*. Wiley Computer Publishing, 1998.
- [Hori et al. 03] M. Hori, M. Abe, and K. Ono. Extensible framework of authoring tools for web document annotation. In *Proceedings of International Workshop on Semantic Web Foundations and Application Technologies*, Nara, Japan, March 2003. National Institute for Informatics. www.kasm.nii.ac.jp/SWFAT/PAPERS/SWFAT20R.PDF.
- [MacKenzie 91] S. MacKenzie. *Fitts' law as a performance model in human-computer interaction*. PhD thesis, University of Toronto, Ontario, Canada, 1991.
www.york.ca/mack.
- [Mankoff et al. 02] J. Mankoff, A. Dey, M. Moore, and U. Batra. Web accessibility for low bandwidth input. In *Proc. of the Fifth ACM SIGCAPH Conference on Assistive Technologies*, July 2002.
- [Mori et al. 02] G. Mori, F. Paternò, and C. Santoro. CTTE: Support for Developing and Analysing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28(8):797-813, August 2002.
- [NNG 01] Nielsen Norman Group. Beyond ALT Text: Making the Web Easy to Use for Users with Disabilities. www.nngroup.com/reports/accessibility, Oct 2001.
- [NNG 02] Nielsen Norman Group. Web usability for senior citizens. www.nngroup.com/reports, April 2002.
- [Parente 03] P. Parente. Agile annotations. www.cs.unc.edu/parente/ip/AgileAnnotationsdraft.pdf, Oct 2003.
- [Paternò and Santoro 02] F. Paternò and C. Santoro. One model, many interfaces. In *Proceedings Fourth International Conference on Computer-Aided Design of User Interfaces*, pages 143-154. Kluwer Academics Publishers, May 2002.

- [Rosenfeld and Morville 98] L. Rosenfeld and P. Morville. *Information Architecture for the World Wide Web*. O'Reilly, 1998.
- [Slatin and Rush 03] J. Slatin and S. Rush. *Maximum Accessibility: Making Your Web Site More Usable for Everyone*. Addison-Wesley, 2003.
- [Takagi and Asakawa 01] H. Takagi and C. Asakawa. Transcoding system for non-visual web access (2): automatic transcoding. In *16th Int. Conf. on Technologies and Persons with Disabilities (CSUN2001)*, 2001. www.csun.edu/cod/conf/2001/proceedings/0196takagi.htm.
- [Thatcher et al. 02] Jim Thatcher, Cynthia Waddell, Shawn Henry, Sarah Swierenga, Mark Urban, Michael Burks, Bob Regan, and Paul Bohman. *Constructing Accessible Web Sites*. Glasshouse, 2002.
- [CUD 97] The Center for Universal Design. Principles of universal design. www.design.ncsu.edu:8120/cud/univ_design/princ_overview.htm, Feb 1997.
- [Thevenin and Coutaz 99] D. Thevenin and J. Coutaz. Plasticity of user interfaces: framework and research agenda. In Sasse A. and Johnson C., editors, *Proceedings of Interact '99*, pages 110-117, Edinburgh, UK, 1999.
- [Usablenet 04a] UsableNet Inc. Annotations XML code for Lifescan. www.usablenet.com/products_services/text_transcoder/annotations/lifescan/lifescan_annotations.html, Apr 2004.
- [Usablenet 04b] UsableNet Inc. Benchmarking LIFT Text Transcoder. www.usablenet.com/products_services/text_transcoder/benchmark.html, Apr 2004.
- [Usablenet 04c] UsableNet Inc. LIFT Text Transcoder. www.usablenet.com/products_services/text_transcoder/text_transcoder.html, Apr 2004. Demo version available at demott.usablenet.com/tt.
- [Usablenet 04d] UsableNet Inc. LIFT Text Transcoder User's Manual. Apr 2004. Demo version available at demott.usablenet.com/tt.
- [W3C/WAI 99] World Wide Web Consortium - Web Accessibility Initiative. Web content accessibility guidelines 1.0. www.w3.org/TR/WCAG10, May 1999.
- [W3C/WAI 03] World Wide Web Consortium - Web Accessibility Initiative. Web content accessibility guidelines 2.0 - internal draft. www.w3.org/WAI/GL/WCAG20/, Nov 2003.