LNCS Proceedings of 8th ERCIM UI4ALL workshop, Vienna, June 2004.

Using automatic tools in accessibility and usability assurance processes

Giorgio Brajnik[1]

Dip. di Matematica e Informatica
Università di Udine, Italy

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

Abstract. The paper claims that processes for monitoring, assessing and ensuring appropriate levels of accessibility and usability have to be adopted by web development and maintenance teams. Secondarily it argues that automatic tools for accessibility and usability are a necessary component of these processes.

The paper presents first an analysis of web development and maintenance activities that highlights the reasons why accessibility and usability are so poorly achieved. It then suggests which processes, borrowed from the domain of software quality assurance, should be established to improve production and maintenance of web sites. The paper finally shows how automatic tools could fit in those processes and actually improve them, while being cost-effective.

1. INTRODUCTION

There are many books and other published material that present a wealth of information on what to do, and what to avoid, when designing, developing or maintaining a web site, like for example [Lynch and Horton, 02]. They discuss typography on the web, dealing with issues like alignment, capitalization, typefaces, etc. Although extremely educational and useful, this written knowledge is obviously not sufficient, considering the low quality level of existing web sites. In order to improve the quality of sites a web developer has to study this material, has to extract useful guidelines from it and has to decide which principles to apply to the specific case, how to apply them, and when.

A better start is by looking at compiled lists of web usability guidelines [Nielsen, 04; Nielsen, 99; NCI, 02]. But also in this case the web developer has to study this material to figure out which guideline is relevant for the specific situation at hand, and to apply it.

A crucial decision is which principles to apply, as different situations and contexts call for different choices. In order to determine which principles are relevant to a specific situation a web developer has to (i) detect failures of the site, (ii) diagnose them and identify their causes, (iii) prioritize them in terms of importance, (iv) determine how to repair them, and (v) estimate benefits and costs of these changes.

The rapid pace and tight deadlines at which development and maintenance of web sites proceed severely hinder the ability to perform processes (i) to (v) in an effective, reliable and cost-effective manner.

On the other hand, processes (i) to (v) are necessary if the site is bound to match a certain level of quality since accessibility and usability are playing an increasingly important role in the game of achieving and maintaining a successful web site.

The claim of the paper is twofold. First, in order to improve accessibility and usability lev-

[1] Scientific advisor for UsableNet Inc., manufacturer of LIFT tools that are mentioned in the paper.

els of web sites developers and maintainers have to establish assurance processes that are similar to those currently adopted for ensuring software quality. Second, due to the nature of accessibility and usability failures, the adoption of automatic tools supporting these assurance processes appears to be inevitable.

2. LOW ACCESSIBILITY AND USABILITY LEVELS

The current demand for accessible and usable web sites (as witnessed also by the large number of books published on the subject in the last years: [Nielsen, 93; Mayhew, 99; Nielsen, 99; Nielsen and Tahir, 01; Thatcher et al., 02; Holzschlag and Lawson, 02; Rosson and Carrol, 02; Slatin and Rush, 03] to name a few) stems from the fact that the vast majority of existing web sites suffer from chronically low accessibility and/or usability levels. The trend is improving, thanks to legislation [USDOJ, 01], public awareness and market pressure [Ramasastry, 02].

2.1 Reasons

Even simple changes to web sites like adding a couple of new pages may lead to reduced accessibility and/or usability (A&U). The most relevant reasons are:

- Lack of resources. Lack of time, money or skilled persons hinder any effort aiming at improving quality of the web site. Of the three, by far the biggest problem is lack of skills because if A&U are thought of during design phases (rather than being retrofitted once the web site has already been produced) then their cost becomes much smaller.
- Release cycles of web sites are extremely fast. Changes to a web site may be conceived, designed, implemented and released in a matter of hours. This is due to the informative nature common to many web sites. It is also supported by the flexibility of the medium (changing something in HTML is in general much easier than in any programming language) and by the low distribution barrier (once the modified pages are published then their message is automatically delivered to web visitors).
- Detailed, accurate and complete specifications are missing. When a web site has to be changed, unless the change is a significant redefinition or revamp of the site, often no analysis and specification steps have been carried out. At least not in full detail and accuracy. This implies that the change is implemented sooner than it should, with a number of uninformed decisions being taken.
- Web browsing technologies are quickly evolving. The change to the web site may occur well after the web site has been designed for the first time, and since then the technologies and standards may have evolved. For example, assistive technologies may have evolved in such a way that the choices which were made when the web site had been designed are now obsolete. This means that changing the web site without considering this evolution is likely to worsen its A&U.
- Not all web architects, web designers, web masters and web programmers (in short the persons that are required to design and implement the change) are A&U savvy. Certain apparently trivial design or implementation decisions (f.e. typographic choices like font face and size and their implementation in HTML) may lead to solutions that violate accessibility standards or usability guidelines.

3. USABILITY AND ACCESSIBILITY ASSURANCE PROCESSES

3.1 Web site development processes

In general a site development and maintenance team includes at least the following roles (potentially carried out by the same persons): web architects, web programmers, web designers, web content producers, accessibility and usability engineers, web masters, project managers.

These job roles are involved in a number of processes, covering the following activities: elicitation of clients' and users' goals, definition of sites' purposes, analysis of users jobs and work flows, definition of task models, choice of technologies, definition of the information architecture and of the look and feel of the site, definition of web site style guides, of page layout, of the site launching activities, and of the site maintenance processes; implementation of the site, graphical design, page debugging, multimedia production, site launch, bug fixing [Hackos and Redish, 98; Lynch and Horton, 02; Goto and Cotler, 02; Holzschlag and Lawson, 02].

3.2 Developing web sites vs. packaged software

Developing a web site is, in many ways, like developing an interactive software system, as in the latter case the processes being involved are very similar to the one listed above.

However, the specific nature of web sites leads to the following major differences , each one requiring more specific and demanding A&U assurance processes.

- Broader user audience: thus there are more user goals, user tasks and user contexts to cope with, which makes the design of effective sites more difficult.
- Less control on technologies used by web visitors and potential conflicts between standards, browsers, plug-ins, assistive technologies, web site implementations. Which further complicate the user interface of the web site.
- Less experienced users: in few cases a web designer can assume that the users of the site are already experienced in using it. In most of the cases visitors have to learn how to use a web site while trying to get a job done.
- More emphasis on user interface issues, as most of the decisions concern the content, information architecture, interaction structure or look and feel aspects, that are all components of the user interface of the web site.

For these reasons it is important that every design and implementation decision that may affect the user interface (i.e. anything that manifests in DHTML coding of the pages) is validated as soon as possible. *Validation* means to make sure that the adopted design or implementation is indeed something that solves the right problem.

3.3 Difficult decisions

Achievement of appropriate A&U levels should not be seen as a target, but as a process. A&U assurance is just like cleaning a house: one has to decide which rooms should be cleaned at which level of hygiene (which depends on who will be using the rooms, and why), how to determine if they are clean, how frequently they have to be cleaned, who and with what kind of tools will be cleaning them.

When carrying out these processes, and especially assurance ones, a number of difficult managerial and technical decisions have to be taken (see the glossary in the appendix for some of the more technical terms).

1. Determining the goal of the assurance process.
2. Determining which guidelines to adopt.
3. Determining how to resolve conflicts between competing guidelines.

   Discussion Determining the goal of the assurance process requires that somebody decides which part of the site has to be assessed, and against which level of A&U. One has to decide if this level should be based on existing guidelines (official ones [USDOJ, 01; W3C/WAI, 99] or unofficial but well known ones [Nielsen, 04; NCI, 02; NNG, 01; Nielsen, 99] or if in-house guidelines should be developed for use within the organization. The advantage of adopting existing guidelines stems from the fact that guidelines embody the results of investigations carried out by others which may be relevant also to the specific case. However it may happen that guidelines need to be adapted to fit to specific situations.
   For example, putting navigation bars to the right of the page improves accessibility (even though they are repeated in many pages, a screen reader user does not have to hear them over and over *before* getting the page content). However, sighted visitors might be required to horizontally scroll the page to see the navigation bar, which is a known usability hindrance.

4. Choosing the assessment methods.
5. Determining how to detect failures.

   Discussion Choosing the methods affects the effectiveness of the process and its efficiency. One has to decide when to run user testing sessions, comparative usability investigations, expert reviews or walkthroughs for accessibility or usability, or when to use automatic tools. And in any case general principles stated in the guidelines have to be made operational so that it is possible to determine if a web site satisfies the guideline or not.
   For example even an apparently simple principle like “provide equivalent alternative textual descriptions for images” requires an ad-hoc standard to be defined regarding different kinds of images (spacers, decorative images, buttons, technical diagrams, etc.), regarding the actual phrasing of alternative texts (are they to be the same as captions, when figure captions exists? are they to be used solely for detecting existence of an image, or should they attempt to provide a concise description of the image?), regarding the way in which the image is embedded in the page (what should the alternative text be for images that are appropriately described by the text already shown in the page?) and regarding the implementation of the solution (should the IMG/ALT tag and attribute be used or would it be better to use the OBJECT tag, perhaps because it supports multilingual alternative text to be attached to the image?).

6. Deciding which guidelines to apply at which stage of the development process.
7. Determining how to prevent errors.

   Discussion Certain methods and guidelines should be applied to early deliverables of the development process. For example, some accessibility failures are due to bad design decisions: the use of frames, putting navigation bars to the left of the page, laying out the page content with fixed-size tables, or implementing menus with rollovers. These decisions are taken when the page layout is designed, at a stage that

occurs earlier than production of content. But, unless a careful assessment of the page template has been carried out *when the template is produced*, the consequences of these choices show up only when content is produced. Leading, for example, to pages where skip-to-contents links have to be added, navigation bars have to be moved to the right, a new implementation of the table layout has to be done in order to yield the correct reading order, and navigation options implemented through rollovers have to be made redundant with other mechanisms. The problem is that fixing a seemingly simple defect requires a reassessment of page design, and it is likely that a change in many of the pages is needed. The later this is done, the more costly it is.

On the other hand, violation of other guidelines can be spotted at later times (say, after content is produced) without incurring in significantly larger costs. For example, figuring out that images are not labeled (with the ALT attribute) is a problem which requires a localized fix that does not impact on other parts of the site. However, doing it afterward still requires a systematic, tedious activity of locating all image occurrences, labeling them and then making sure that labels are all consistent.

8. Determining how to estimate the cost, and the *Return on Investment* of the entire assurance process.

Discussion A quality cost model used for software quality [Black, 02] could be applied to web development as well.

According to the model the cost of quality (C_q) is the sum of the *cost of conformance* (C_c) and the *cost of non-conformance* (C_{nc}). C_q includes any expenditure for preventing defects (e.g. training, tools, process establishment and tuning) (C_p) and the cost of appraising the current quality level through planning and running verifications (personnel, setting the test environment, running the verifications, reporting results) (C_a). C_p is basically a fixed cost incurred initially, while C_a is a cost that depends on the number of defects that are found (and fixed, which requires repeated verifications). C_{nc} includes the economic effects of bugs discovered by the development team or by users after release. At the very least, C_{nc} includes the costs for the infrastructure, time and effort needed for reporting, replicating, isolating, fixing, verifying each of the bugs reported by web visitors (C_{direct}). In addition it includes also less tangible costs like delays, slowdowns, customer alienation, uncompleted transactions, reduced traffic, reduced sales, reduced customers acquisition, conversion, retention ($C_{indirect}$).

This leads to

$$C_q = C_c + C_{nc} = C_p + C_a(N) + C_{direct}(M) + C_{indirect}(M+K)$$

where N and M are the number of defects found and fixed prior to (and respectively after the) release; K is the number of defects experienced by web visitors but not reported to the technical support team.

This cost model, imported from software engineering, could be applied also to web site development and maintenance. However, for web sites the proportion of defects that are reported by visitors is smaller than for software. Most visitors will not complain and simply will move away from the site after facing one or more A&U failures. Therefore C_{direct} will be small, because of the small number of reported bugs. However $C_{indirect}$ is likely to soar. Only when it reaches certain levels, management might be willing to launch some investigations to determine why the web site is not

yielding the expected business results (in terms of visitor reach, acquisition, conversion, retention). These investigations increase C_{direct}. When a number of defects have been found (usability and accessibility ones, perhaps dealing with a poor information architecture, poor task support or poor presentation) they have to be fixed, yielding another increase in C_{direct}.

Little expenditures on C_c will apparently lower the overall cost, but C_{direct} and especially $C_{indirect}$ will increase. The latter will happen without explicit warnings since no monitoring is in place. Only effective assurance processes can avoid this hidden increase in non conformance costs.

The return on investment has to be viewed as a reduction of non-conformance costs, and especially for the intangible costs. Since these costs are affected by many other factors (like usefulness of the web site, popularity of the web site and the organization owning it, their credibility), this explains why it is difficult to accurately characterize the ROI of A&U efforts and why they are often perceived mainly as a cost source.

3.4 GQM: a general framework for assessments

The *Goal, Questions and Metrics* (GQM) approach [Basili and Weiss, 84] can be helpful to frame many of the decisions discussed above. It is very likely that each organization has to develop and establish its own means for answering those questions. Generality of guidelines and principles, and specificity of the organization and its work processes are the primary reasons for this state of affairs. Therefore appropriate methods have to be deployed to investigate those questions. The GQM approach supports this. It is based on the following steps, described here in the context of web site analysis:

Establish the goals of the analysis. Possible goals include: to detect and remove usability obstacles that hinder an online sale procedure; to learn whether the site conveys trust to the intended user population; to compare two designs of a site to determine which one is more usable; to determine performance bottlenecks in the web site and its back-end implementation.
Goals can be normally defined by understanding the situation of concern, which is the reason for performing the analysis. Which actions will be taken as a result of the quality assessment? What do we need to know in order to take these actions?

Develop questions of interest whose answers would be sufficient to decide on the appropriate course of actions. For example, a question related to the online sale obstacles goal mentioned above could be “how many users are able to buy product X in less than 3 minutes and with no mistakes?”. Another question related to the same goal might be “are we using too many font faces on those forms?”. Questions of interest constitute a bridge between goals of the analysis and measurable properties that are used for data collection. Questions also lead to sharper definition of the goals. They can be used to filter out inappropriate goals: goals for which questions cannot be formulated and goals for which there are no measurable properties. Questions can also be evaluated to determine if they completely cover their goal and if they refer to measurable properties. In addition certain questions will be more important than others in fulfilling a goal.

Published guidelines are useful since they suggest relevant questions. For example a guideline requiring to use white space to separate links in navbars suggests questions like “are links too close to each other?”, “how will the space between links change if font size is increased?”, “how many visitors will be affected by violations of this guideline?”, “what are the possible consequences on visitors' tasks if this violation occurs?”.

Establish measurement methods (i.e. metrics) to be used to collect and organize data to answer the questions of interest. A measurement method (see [Fenton and Lawrence Pfleeger, 97] for additional details) should include at least:

1. a description of how the data have to be elicited. For example via user testing with a given task description; or by automatically inspecting HTML sources of online pages to learn how links are organized.
2. a description of how data are identified. For example, what does “successfully completed task” mean in a user testing session?; how is “time to complete the task” going to be measured?; what constitutes a link in a web page (i.e. which HTML tags should be considered: A, AREA, IMG, BLOCKQUOTE, ...)?; what constitutes a local link to a web site (e.g. same server, same sub-domain, or same path)?
3. a description of how data are going to be categorized. For example, what are the different kinds of mistakes that users might get involved in, how can the “completion time” be broken down into different phases, if there are different categories of local links: links to images, videos, HTML files, etc. Measurement methods sharpen and refine the questions of interest.
   Even though some A&U guideline is sufficiently precise and specific to suggest the appropriate measurement method, it is often the case that in order to be operational guidelines need to be specialized yielding an internally defined standard.

Accessibility metrics may include the number of violations of a certain guideline/checkpoint, once the guideline or checkpoint have been made sufficiently specific and operational.

3.5 Defect flow model

A fundamental assumption of any quality assurance activity is that the artifact being evaluated is rarely free of defects. The purpose of the quality assurance activity is to understand which defects and how many of them are included in the artifact.

Now consider how defects flow in and out of the web site. Defects are inserted into and removed from the web site as an effect of the overall development and maintenance process (see figure 1). More specifically, the following subprocesses can be identified:

Defect insertion: this occurs during web site development and maintenance, at any stage. Defects may be due to errors in implementation decisions (like forgetting to label correctly controls of a form) or in design decisions (like relying on colors to convey information) or during analysis (like not providing the appropriate navigation path to support a given information task). In the majority of the cases this happens implicitly, without notice. Sometimes errors simply occur because somebody has thought of doing something, but then it's been forgotten. Other times they occur because of misjudgment

(f.e. deciding that a certain accessibility requirement is not so important) or because of ignorance. Finally there are situations where defects are known to be included in the design, because alternative correct solutions are too expensive to be developed.

Defect removal: this happens after defects have been discovered, have been analyzed and their removal has been planned. Web developers design one or more solutions, rank them, plan their execution, and finally implement them. Usually, after executing a solution, a subsequent verification step takes place making sure that the defect has been removed, and that applied changes do not perturb the rest of the site (*regression testing*).

Defect removal is a process informed by *defect triage*, since the latter provides priorities and schedules for defect removals.

Defect elicitation: this occurs when A&U failures are detected and traced back to their causes: the defects. This may happen when web site visitors complain about the site or when explicit verification activities are carried out and they identify the defects.

During this process several elicitation techniques may be used: user testing, heuristic evaluations, accessibility inspections, accessibility "sniff tests", automated analysis, analysis of web server logs, code inspections.

Defect elicitation is guided by *A&U monitoring* and by *A&U assurance plan definition*: the former triggers elicitation, the latter determines which part of the site has to be tested, why, how and when.

Defect triage: is the process of determining if observed defects have been already dealt with in the past, which defects should be removed, to schedule their removal and to track their evolution in the site. *Defect triage* evaluates reported defects so that most critical ones are treated first [McCarthy, 95]. A complex step, especially with highly changing web sites, is to determine if a defect just being reported has been already managed in the past (for example, it was decided that it was not a violation of some guideline, or that its solution should be deferred).

"Failure Mode and Effects Analysis" [Black, 02] is a way for computing criticality of a defect within a formal approach. In any case, criticality of a defect depends on the following factors, whose importance depends on the specific organization running the assurance processes:

Violation on required standards: for example a failure by a U.S. federal web site that does not satisfy *Section 508* accessibility requirements is a good candidate for a high level of criticality.

Individual impact: (or *severity*) the impact of the failure with respect to an individual user who has to face it. This estimation may be formulated in terms of user inability to successfully complete the task that s/he intended to do, in terms of the importance of the task and importance of that kind of user.

Affected audience: (or *likelihood*) the fraction of the intended user population of the web site that is affected by the failure. The likelihood of a failure in turns depends on (i) how frequently the pages causing it are shown to visitors and (ii) which proportion of those visitors will actually experience the failure.

A defect located on the home page will be more critical than the same defect located on a secondary page, which will be visited by a smaller proportion of visitors. Web server logs can also be used to determine the frequency of display of given pages. On the other hand, determining the proportion of visitors that will experience the failure depends on assumptions, like knowing how many visitors use the page in the contexts leading to the failure. For example, only visually impaired visitors using JAWS on Windows 98 clicking on a certain link might experience the failure.

Symbolic impact: the failure might be perceived by the audience as a symbolic quality problem; it could be an embarrassing bug that requires a prompt treatment, even though its impact and likelihood are low.

Removal costs: in order to remove a bug a number of activities have to performed: (i) to diagnose the failures and determining their causes (i.e. the defect), (ii) to determine one or more possible solutions (i.e. alternative treatments), (iii) to estimate their requirements in terms of persons, skills, and infrastructure, (iv) to implement one solution, (v) to verify that the defect has disappeared, and finally (vi) to estimate time and money needed for all these activities. Fixing a defect like a missing ALT is a relatively easy task (easy to spot, to plan, to implement, and to verify), whereas changing a page layout that is based on fixed size tables so that it becomes CSS based with floating elements is much more complex. More complex to plan and to devise a solution, to implement, and especially to verify on all the pages with all the possible dynamic content that is available on the web site.

Instability of the site: fixing a defect requires changing some part of the web site, and these changes may propagate to other pages across the site. For example, fixing a usability defect by replacing a label of a link appears to be an easy and localized change of a page. But a deeper analysis may indicate that the same label was used in a global navigation bar, in the heading of an entire sub site and also in the URLs of the pages. What appeared to be an easy change has become a very destabilizing activity on a large fraction of the site.

A&U assurance plan definition: the plan should describe:

- what parts of the web site should be evaluated. For example, all the pages at a depth up to 2 from the home page; or all the pages that have a request rate, as gathered from web server logs, higher than 10% of the total.
- which criteria should be used to evaluate them and what levels of A&U should be reached. For example aiming at AA conformance level for accessibility as specified by the Web Content Accessibility Guidelines [W3C/WAI, 99].
- when these evaluations should take place. For example, a global accessibility evaluation should take place whenever any page template is changed. Or a link checking evaluation should be run every day to identify dead links pointing to web resources that have disappeared.

A&U monitoring: it is the process of managing methods, tools and roles for alerting when a failure has occurred and therefore triggering the *defect elicitation* process.

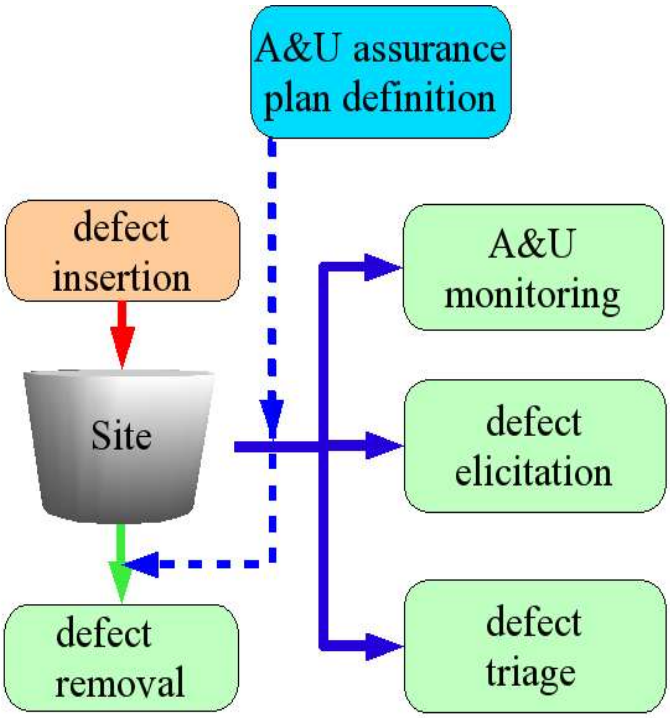

Figure 1: The defect flow model: the web site is seen as a container of defects. Monitoring, elicitation, triage processes, guided by an assurance plan definition, can lead to effective defect removal.

4. IMPACT OF AUTOMATIC TOOLS ON ASSURANCE PROCESSES

In software engineering the deployment of several kinds of tools has been functional to the advance of the discipline. Tools are used to develop code and documentation, to define test cases, to run tests, to categorize and manage defects. Without these kinds of tools, software quality processes would be more demanding in resources than they are now, and would yield lower levels of quality. The same thing is to be expected for web development.

4.1 Automatic tools

There are several flavors of web testing tools [Brink and Hofer, 02; Brajnik, 00]:

- tools for accessibility testing and repair, like Bobby, A-Prompt, 508Accessibility Suite, Site Valet, AccVerify, LIFT, etc. The metrics implemented by these tools correspond (more or less accurately) to official (W3C or Section 508) accessibility criteria.
- tools for usability testing (e.g. WebXM, LIFT, WebCriteria) that are based on usability guidelines.
- tools for performance testing (e.g. TOPAZ)
- tools for security testing (e.g. WebCPO)
- tools for analyzing web servers logs;
- tools for classifying a web site after learning the classification criteria from other

web sites (e.g. WebTango [Sinha et al., 01]);

- tools for analyzing the content structure in a site (e.g. *Information Foraging* theories and models [Chi et al., 00]).

These tools cover a large set of tests for A&U. For example, usability tests can be grouped under the following factors (for more details see [Brajnik, 00]):

- consistency of presentation and controls (e.g. link labels, email labels, color consistency, navbar consistency, link colors)
- adequate feedback (e.g. freshness of information, authorship of pages)
- contextual navigation (e.g. NOFRAMES with equivalent content, links to home, breadcrumbs, skip-links, self-referential pages, frame titles, table summary)
- efficient navigation (e.g. table cells sizes, image sizes, download time, hidden elements, recycled graphics)
- clear and meaningful labels (e.g. page titles, link labels and titles, form labels)
- robustness (e.g. browser compatibility, safe colors, appropriate color contrast, standard font faces, no color-coding of information)
- flexibility (e.g. image ALT, auto refresh and redirect, font and page resizing, optional form fields)
- maintainability (e.g. relative URLs, use of CSS).

Obviously, automatic tools cannot assess all sorts of properties. In particular, anything that requires interpretation (e.g. usage of natural and concise language) or that requires assessment of relevance (e.g. ALT text of an image is equivalent to the image itself) is out of reach. Nevertheless these tools can highlight a number of issues that have to be later inspected by humans and can avoid highlighting them when there is reasonably certainty that the issue is not a problem. For example, a non-empty ALT can contain placeholder text (like the filename of the image); it is possible to write heuristic programs that can detect such a pattern and flag such an issue only when appropriate - and be able not to flag it if the ALT contains other text.

4.2 How tools support assurance processes

No tool can be expected to directly provide answers to the questions discussed in section 3.1. However tools can support those decisions, by better informing the decision maker and by providing functionalities that can foster certain processes and decisions.

Similarly no tool can be expected to do all the work by itself. To effectively deploy a tool, persons have to be trained on A&U methods and issues, on how to use effectively the tools, on how to establish appropriate processes that fit or improve the existing work flows.

A quick analysis of tools like LIFT (produced by UsableNet Inc., see `www.usablenet.com`) can highlight its impact on the A&U assurance processes. LIFT is in fact a family of tools including:

- *LIFT for Dreamweaver - Nielsen Norman edition* (LFD): a plug-in for the Macromedia Dreamweaver authoring environment that does evaluation and repairs with respect to accessibility guidelines (W3C/WCAG priority 1 and 2 [W3C/WAI, 99], and Section 508 [USDOJ, 01]) and with respect to usability guidelines for disabled persons developed by Nielsen Norman Group [NNG, 01];
- *LIFT Machine* (LM): a multi-user server-based tool for quality assurance that does evaluation of public web pages with respect to standard accessibility guidelines and

usability guidelines related to some of the usability factors mentioned above.

Impact on Defect insertion

Automatic tools can affect the *Defect insertion* process.

LFD and LM, for example, come with a rich description of what an accessibility problem is (i.e. the failure and the possible defect), why it is important to fix it (i.e. the consequences of the failure), where you can learn more about it, and how you could fix it (i.e. how to remove the defect). This description can be used to recall a specific guideline to an experienced developer or to train novices, an important requirement as many U.S. federal agencies hire external contractors to retrofit accessibility on their web sites, as required by Section 508; these contractors are not always selected on the basis of their experience in accessibility.

In addition, being LFD embedded in the authoring environment, this information is readily available while developing the site. And actually through the familiar user interface of the development environment.

Thirdly, LFD has a monitoring function that continuously evaluates the page as it is developed, highlighting the potential defects. This can therefore alert the developer as soon as the defect is inserted in the site so that it can be removed as early as possible.

Finally, both LFD and LM offer page previewers that help to debug a page, right when it is being developed or when it is being assessed. Previewers display the page with colors turned off (to discover if there is information that is color coded) or with linearized content (to check reading order) or in text mode.

Impact on Defect elicitation

Automatic tools can improve the effectiveness of the defect elicitation process. They are systematic, do not get tired or bored, and are fast.

LFD and especially LM can scan a large number of local or live pages (up to several thousands) and apply a large (140+) set of tests on them.

Obviously, only certain kinds of properties can be tested in an automatic way. Usability and accessibility are *external* properties [Fenton and Lawrence Pfleeger, 97], since they can be evaluated only when a web site is being used by somebody. On the other hand, tools mainly look at static properties of the web site (their coding) to see if certain violation patterns arise. In general everything that has to do with human interpretation and context of use is likely to be poorly machine testable.

If heuristic tests are adopted (which often is the case since accessibility and usability are properties that seldom can be reduced to easy clear-cut decisions), the tools may produce false positives (i.e. they report potential failures in cases where there is no defect). LFD and LM reduce the number of false positives by guessing the role that images and tables play in a page. On the basis of image size and type, and its location in the page, LFD and LM are able to guess with good accuracy if the image is a spacer, a decoration, a banner, a button, or something else. And therefore they can offer a more specific diagnosis of the problem and suggestions for a solution. See [Brajnik, 04] for a description of an assessment method of tools and some preliminary results about effectiveness of LM with respect to accessibility.

[Lang, 03] discusses criteria to be used to compare different A&U evaluation methods. The same criteria can be used to compare the ability of different tools to elicit defects: ability to detect problems (called *completeness* in [Brajnik, 04]); accuracy and quality of results (called *correctness* in [Brajnik, 04]); time, effort and cost effectiveness; usefulness of results and who can use the results; generality of results.

Impact of Defect triage

Tools like defect tracking systems are fundamental in this area. These tools have to support the development team in (i) entering defect descriptions, making sure that there are no duplicates, (ii) document possible interactions among defects (for example one defect may mask a different one; a defect may show up only when a different one occurs), (iii) track their evolution over time, and (iv) support triage processes. Bugzilla [Bugzilla.org, 04] is an example of such a tool, used for tracking software bugs.

Tracking defects on web sites is different, and more complex, than for software. In fact it is not easy, in general, to determine if a defect that has just shown up is exactly the same as the one that was seen one week ago. Consider for example that a single web page may show several hundreds of accessibility guidelines violations. And if the content of the page has changed, in principle no two violations are the same.

Secondly, especially for web pages that are dynamic, it is not easy to determine which files or components have to be fixed in order to remove a defect. The wrong code may be part of a template, of an included piece of code processed at compile-time, of content that is pulled out of a database or back-end application. If it is part of a template, or of content that is reused elsewhere, then the same defect will cause several failures.

LM offers several different views of the list of failures that it detected on a web site, but it does not support defect triage.

Impact on Defect removal

Tools can improve also the removal process, at least for certain kinds of defects.

LFD and LM offer, whenever possible, examples of good solutions (like fragments of Javascript code for correctly handling events causing new browser windows to be opened) that can be copied into the page being developed.

In addition, LFD offers *Fix Wizards* that allow a user to fix a problem without having to manually edit DHTML code. Which again is convenient for users who are not experienced or not used to work with HTML or Javascript. For example, fix wizards for data tables (i.e. TABLE tags used to display tabular information) that allow to markup correctly table cells so that they refer to table headers.

Finally, the *ALT Editor* of LFD supports a global analysis and fix of the ALT attributes of all the images found in the site. This is useful to ensure consistency among these image labels.

Impact on Monitoring

Tools can support the monitoring process.

For example, LM can schedule evaluations on a live site over time; they are run in an unsupervised mode and when ready an email is sent to the person that requested them.

While LM provides means for getting an overview of the status of a web site, this is limited to pages that have been tested within a single run of the LM. It does not directly support integration of results regarding different runs. For example, users cannot practically see the status of large web sites (10000+ pages) unless they first attempt to run a test on all of them.

[NCAM, 03] describes STEP508, a free tool that uses data produced by accessibility evaluation tools and identifies the defects that are most critical to fix and tracks the progress of the accessibility repair processes. STEP508 support the monitoring process since it can be used to integrate and compare different evaluations. It also supports other processes: by prioritizing defects it supports *Defect triage*; by tracking the progress it supports monitoring *the assurance processes themselves* rather than monitoring the A&U of the web site.

Impact on Plan definition

Tools can support the definition of the test plan.

For example, LFD and especially LM allow the user to enable/disable individual tests, groups of tests and to define and use named guidelines profiles. In this way the user has a fine control on which tests are applied.

Secondly, the behavior of many tests is affected by parameters whose value can be changed by the user. For example, to determine if a site has a "text only" version, LFD and LM look for links containing words like "text only", "text version", etc. By changing the value of this parameter the user can customize the behavior of the test.

Impact on ROI

Tools affect the cost of A&U assurance in two ways. They contribute to increase fixed costs for the infrastructure and training: cost of the tool itself, of training and of tuning of work processes (C_p). However, if appropriately deployed they are likely to reduce the running costs of appraisal (C_a), to reduce the number of defects that are uncaught by the development team and the direct costs ($C_{direct}(M)$), and to reduce the indirect costs because fewer defects will be experienced by users.

5. CONCLUSION

While there are differences between development and maintenance of web sites and of software, it is likely that the same assurance processes adopted for software can be deployed for web sites. And, as a consequence, also automated tools for supporting A&U assurance can play a significant and positive role.

They can address the issues raised in section 2.1:

- lack of resources: appropriately deployed tools lead to a positive ROI and save time. Tools are likely to improve effectiveness and productivity of the team and, *at the same time*, increase the quality level that can be achieved.
- Fast release cycles: tools are systematic, fast and reliable. They can be routinely used also within short cycles. In certain cases only automatic testing is viable.
- Lack of specifications: a tool may be used to define in-house guidelines, to make them practical and be easily enforced. Tools can also help in the prevention of errors.
- Evolving technologies: tools can be updated and can reflect the state of the art of the

available technology.
- Ignorance: tools are powerful training and reminding vehicles. They can deliver guidelines at the fingertips of those needing them.

At the moment adoption of tools is still limited. In my view this is due to several factors, including a limited awareness of the benefits of a high-quality (high accessibility and usability, to be more precise) web site and absence of established methods for comparing tools and their results. [Brajnik, 2004] is a preliminary step towards such a direction.

GLOSSARY

The following terminology has been used in the paper:

bug: a generic term for referring to a misbehavior of the system and its causes.
failure: the manifestation of a misbehavior of the system. For example, when a web visitor using a screen reader gets the content of the page read in an incorrect order. Notice that in case of a usability and accessibility failure of a web site, misbehavior has to encompass the behavior of: the site, the web server, the browser, browser's plug-ins, any assistive technology, and the operating system used by the visitor.
failure mode: the category of failures that share the same kind of misbehavior. A failure mode is the set of symptoms; these symptoms may show up during a specific failure, and are caused by one or more defects (i.e. the disease).
defect: or *fault*, the reason why a failure may show up. Typically, for web usability and accessibility, a defect is rooted in some fragment of code implementing the site (HTML, Javascript, CSS). In the previous example, the defect associated with the incorrect reading order might be a bad use of the TABLE tag to layout the page.
error: is the misbehavior of the developer causing a defect that is inserted into the site.

REFERENCES

[Basili and Weiss, 84] V. Basili and D. Weiss. A methodology for collecting valid software engineering data. *IEEE Trans. on Software Engineering*, 10(6):728-738, 1984.

[Black, 02] R. Black. *Managing the testing process.* Wiley Publishing Inc., 2002.

[Brajnik, 00] G. Brajnik. Automatic web usability evaluation: what needs to be done? In *Proc. Human Factors and the Web, 6th Conference*, Austin TX, June 2000. www.dimi.uniud.it/giorgio/papers/hfweb00.html.

[Brajnik, 04] G. Brajnik. Comparing accessibility evaluation tools: a method for tool effectiveness. *Universal Access in the Information Society*, 2004. To appear.

[Brink and Hofer, 02] T. Brink and E. Hofer. Automatically evaluating web usability. CHI 2002 Workshop, April 2002.

[Bugzilla.org, 04] Bugzilla.org. Bugzilla - bug tracking system. www.bugzilla.org/about.html, Feb 2004.

[Chi et al., 00] E. Chi, P. Pirolli, and J. Pitkow. The scent of a site: a system for analyzing and predicting information scent, usage and usability of a web site. In ACM, editor, *Proceedings of CHI 2000*, 2000.

[Fenton and Lawrence Pfleeger, 97] N. Fenton and S. Lawrence Pfleeger. *Software metrics*. International Thompson Publishing Company, 2nd edition, 1997.

[Goto and Cotler, 02] K. Goto and E. Cotler. *Web redesign: workflow that works*. New Riders Publishing, 2002.

[Hackos and Redish, 98] J. Hackos and J. Redish. *User and task analysis for interface design*. Wiley Computer Publishing, 1998.

[Holzschlag and Lawson, 02] M. Holzschlag and B. Lawson, editors. *Usability: the site speaks for itself*. Glasshouse Ltd., 2002.

[Lynch and Horton, 02] P. Lynch and S. Horton. *Web Style Guide*. Yale University, 2nd edition, 2002.

[Lang, 03] Lang T. *Comparing website accessibility evaluation methods and learnings from usability evaluation methods.* Peak Usability. www.peakusability.com.au/articles. 2003.

[Mayhew, 99] D. Mayhew. *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*. Morgan Kaufmann, 1999.

[McCarthy, 95] J. McCarthy. *Dynamics of software development*. Microsoft Press, 1995.

[NCI, 02] National Cancer Institute. Usability.gov. www.usability.gov, 2002.

[NCAM, 03] National Center for Accessible Media. *STEP508*. www.section508.gov/step. 2003.

[Nielsen, 93] J. Nielsen. *Usability engineering*. Academic Press, 1993.

[Nielsen, 99] J. Nielsen. *Designing Web Usability: the practice of simplicity*. New Riders Publishing, 1999.

[Nielsen, 04] J. Nielsen. Alertbox. www.useit.com/alertbox/, 2004.

[Nielsen and Tahir, 01] J. Nielsen and M. Tahir. *Homepage usability: 50 websites deconstructed*. New Riders, 2001.

[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.

[Ramasastry, 02] A. Ramasastry. Should web-only businesses be required to be disabled-accessible? www.cnn.com/2002/LAW/11/07/findlaw.analysis.ramasastry.disabled/index.html, Nov 7, 2002.

[Rosson and Carrol, 02] M. Rosson and J. Carrol. *Usability Engineering*. Morgan Kaufmann, 2002.

[Sinha et al., 01] R. Sinha, M. Hearst, M. Ivory, and M. Draisin.
Content or graphics? an empirical analysis of criteria for award-winning websites. In *Proc. Human Factors and the Web, 7th Conference,*, Madison, WI, June 2001.

[Slatin and Rush, 03] J. Slatin and S. Rush. *Maximum Accessibility: Making Your Web Site More Usable for Everyone*. Addison-Wesley, 2003.

[Thatcher et al., 02] J. Thatcher, C. Waddell, S. Henry, S. Swierenga, M. Urban, M. Burks, B. Regan, and P. Bohman. *Constructing Accessible Web Sites*. Glasshouse Ltd., 2002.

[USDOJ, 01] U.S. Dept. of Justice. Section 508 of the rehabilitation act. www.access-board.gov/sec508/guide/1194.22.htm, 2001.

[W3C/WAI, 99] World Wide Web Consortium - Web Accessibility Initiative. Web content accessibility guidelines 1.0. www.w3.org/TR/WCAG10, May 1999.