FAKULTÄT FÜR !INFORMATIK

Tool-Supported Web Accessibility Evaluation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

eingereicht von

Shadi Abou-Zahra
Matrikelnummer 9426418

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ. Prof. Dr. Harald Gall
Mitwirkung: Univ. Prof. Dr. Schahram Dustdar

Wien, 17.10.2008

(Unterschrift Verfasser)

(Unterschrift Betreuer)

Master's Thesis

Tool-Supported Web Accessibility Evaluation

Carried out at the

Information Systems Institute
Distributed Systems Group
Technical University of Vienna

Under the guidance of
Univ. Prof. Dr. Schahram Dustdar
and
Univ. Prof. Dr. Harald Gall
as the contributing advisor responsible

By

Shadi Abou-Zahra
shadi@abou-zahra.net
Matr.Nr. 9426418

Vienna, 17 October 2008

Acknowledgements

Many thanks to Wendy Chisholm for the precious discussions that lead to the initial idea for this study back in 2002, to Shawn Lawton Henry from who I've learned much about the user side of Web accessibility evaluation, and to Andrew Arch for the invaluable feedback during many iterations of this work. This work is also the result of numerous exchanges with people in the field, especially participants of the W3C Evaluation and Repair Tools Working Group and other members of the W3C Web Accessibility Initiative (WAI).

Disclaimer: the views in this study do not necessarily represent the views of the W3C Web Accessibility Initiative (WAI) or the W3C Evaluation and Repair Tools Working Group. The study does not endorse any of the mentioned tools, vendors, or references. However, it does encourage the development of tools and solutions for Web accessibility and evaluation.

Credits: Figures 1 and 1a – images by Michael Duffy, from: Essential Components of Web Accessibility, Copyright W3C (MIT, ERCIM, Keio) [WAI, 2]. Figures 2a and 11 – images from the W3C Web Accessibility Initiative (WAI) and Semantic Web Activity respectively, Copyright W3C (MIT, ERCIM, Keio). Parts of this study have been published in modified form in other publications by Shadi Abou-Zahra [Abou-Zahra 2005a - Abou-Zahra 2005d, Abou-Zahra 2006b, Harper and Yesilada 2008].

Abstract

While there is currently a broad selection of Web accessibility evaluation tools, most of them are designed around post-development quality control principles rather than on-going quality assurance measures. It seems that many of the Web accessibility evaluation tools encourage separated evaluation processes rather than to integrate evaluation tasks into the natural Web development process. It also seems that Web accessibility evaluation tools are designed for a specific type of developers rather than to address the broad diversity of Web content creators.

A more effective approach would be to ensure tool support throughout the entire development process, including the design, implementation, and maintenance stages. Ideally the evaluation functionality would merge into the existing development and management systems rather than to be presented through separated Web accessibility evaluation tools. For instance, that design utilities, code editors, content management systems and monitoring tools provide accessibility evaluation functionality that is executed by integrated evaluation tools, plug-ins, or extensions.

In order to facilitate this integration of Web accessibility evaluation tools into other types of applications, these tools must be able to exchange accessibility information. Host applications need to delegate and trigger evaluation functionality, and evaluation tools need to support the common protocol for data exchange. These common data formats and protocols are ideally platform-independent, vendor-neutral, and royalty-free to ensure the widest possible support. Furthermore, these formats need to be practical and simple to implement by tool developers.

This study proposes a distributed model for tool supported Web accessibility evaluation based on Semantic Web technologies. It explores the requirements and identifies existing promising techniques that enable the construction of collaborative systems that support different types of developers who may be using a variety of tools to develop and maintain accessible Web sites. Although there are challenges in realizing such systems, the approach is principally feasible. There are reasonably low risks and high return on investments for the application developers.

Zusammenfassung

Obwohl es derzeit eine breite Auswahl von Evaluierungswerkzeuge für Web-Barrierefreiheit gibt, wurden die meisten für Zwecke der Qualitätskontrolle entwickelt eher als für andauernde Qualitätssicherung. Es scheint das viele der Evaluierungswerkzeuge für Web-Barrierefreiheit getrennte Evaluierungsprozesse förden anstatt diese in den natürlichen Entwicklungsprozess zu integrieren. Es scheint auch das die Evaluierungswerkzeuge nur für bestimmte Entwickler konzipiert wurden anstatt die breite Vielfalt der Web-Entwickler zu beachten.

Ein wirksamerer Ansatz wäre die Sicherstellung von Werkzeug-Unterstützung während den gesamten Entwicklungsprozess, einschließlich der Konzipierung, Implementierung, und der Wartung. Idealerweise wäre die Evaluierungsfunktionalität in den existierenden Entwicklungs und Managementsysteme eingebunden anstatt diese durch getrennte Evaluierungswerkzeuge anzubieten. Zum Beispiel, dass Gestaltungswerkzeuge, Code-Editoren, Content Management Systeme und andere Werkzeuge Evaluierungsfunktionalität für Web-Barrierefreiheit anbieten, die durch integrierte Evaluierungswerkzeuge, Plug-ins, oder Extensions ausgeführt werden.

Um diese Integration der Evaluierungswerkzeuge für Web-Barrierefreiheit in Anwendungen zu erleichtern, müssen diese Werkzeuge Informationen austauschen. Anwendungen müssen die Evaluierung delegieren und auslösen, und Evaluierungswerkzeuge müssen einheitliche Protokolle für den Datenaustausch einhalten. Diese einheitliche Datenformate und Protokolle sind idealerweise Plattform-unabhängig, Hersteller-neutral, und Gebührenfrei um die größte mögliche Unterstützung zu erzielen. Weiters müssen diese Formate praktikabel und einfach zu implementieren für die Werkzeughersteller sein.

Diese Studie untersucht ein verteiltes Modell für Werkzeugunterstützte Evaluierung der Web-Barrierefreiheit das auf Technologien des Semantic Web beruht. Es erforscht Anforderungen und identifiziert viel versprechende Techniken um die Konstruktion solcher Kollaborativen Systeme zu ermöglichen, die von unterschiedlichen Entwicklern die verschiedene Werkzeuge verwenden benutzt wird um barrierefreie Web-Auftritte zu entwickeln und warten. Obwohl es Herausforderungen gibt um solche Systeme zu realisieren, ist der Ansatz prinzipiell erfüllbar. Es geht verhältnismäßig wenige Risiken und hohe Nutzen für die Anwendungshersteller.

Table of Contents

1. Problem Description

Web accessibility evaluation is an assessment of how well Web content can be used by people with disabilities. It is a quality assurance measure that is ideally carried out throughout all the development stages of Web content, including its design, implementation and maintenance. It is the responsibility of all parties involved in the development of the Web content to ensure its accessibility. This includes visual designers, code programmers, content authors, as well as Webmasters. Each individual involved in the development of the Web content, for instance a single Web page or an entire Web site, can contribute towards ensuring that the accessibility requirements are met. In fact, each individual involved in the development of Web content is an asset for the production of high quality Web content that better serves a broader audience.

However, it is not realistic to expect that each individual involved in the development of Web content has the same level of awareness and technical understanding of the requirements. It is important to understand the different roles of individuals involved in the production chain and to design an appropriate development process that takes advantage of their skills and expertise. Software tools play an essential part in enabling efficient and effective development processes that consider quality assurance. Even though evaluating Web content for accessibility can not be fully automated, tools can significantly reduce the effort required for evaluation. They can examine the Web content and assist developers in carrying out the different evaluation tasks, while they cache away much of the technical detail that overwhelms many of the developers.

Unfortunately Web accessibility is often not considered from the start but rather after the Web content has been developed, and sometimes even after it has been published. It is a common misconception that accessibility can be built into the Web content after it has been developed. In many cases it is however not economically feasible to retrofit Web content for accessibility because it may require significant alterations to the underlying design concepts. At the same time, the overhead for accommodating most of the accessibility requirements is negligible if they are considered from the start of a project. Also many Web accessibility evaluation tools focus primarily on evaluating existing Web content, rather than on supporting developers in creating accessible Web content throughout the different stages of the development processes.

Many Web accessibility evaluation tools also seem to focus on a specific group of users who are more technical. They generally also require some level of knowledge and expertise in the domain of Web accessibility, so that they can be often cumbersome and confusing for novice evaluators. In fact, in extreme cases Web accessibility evaluation tools could even reduce the efficiency of novice evaluators rather than help them become more productive. It is inherent to nature that Web accessibility is not automatable and needs human judgment in many cases. However, there is a lot of potential for Web accessibility evaluation tools to further assist the developers and evaluators who may have different backgrounds and expertise. The tools need to be more flexible and fit into the environment of the users, rather than the other way around.

There are promising techniques, especially in the domain of the semantic Web, that provide methods for expressing structured data. Machine-readable and semantically rich information could allow Web accessibility evaluation tools to exchange information with authoring tools or with other types of quality assurance tools. It could provide a new approach for small and focused tools that could be connected together into a collaborative system that supports the development and the evaluation of Web content for accessibility. This study explores some of these techniques and proposes a model in which different types of Web accessibility tools can better support Web developers in creating and maintaining accessible Web sites.

1.1. Introduction

Web accessibility is the extent to which Web content can be used by people with disabilities. A widely accepted definition is provided by [Slatin and Rush 2002]: “Web sites are accessible when individuals with disabilities can access and use them as effectively as people who don’t have disabilities”. The definition includes two important aspects of Web accessibility, namely the technical aspect of gaining access to the content as well as the functional aspect of being able to use the content in an effective manner. Also several other prominent definitions for Web accessibility, such as those provided by [Henry 2007], [Paciello 2000], or [Thatcher et al. 2006], underline these two closely related and fundamental principles of Web accessibility.

In order to be able to provide equal access to the Web content for users with disabilities, one must first better understand how users with disabilities use the Web [WAI, 3]. In some cases, users with disabilities make use of adaptive strategies such as enlarging the default font-size or customizing the default text colors. In other cases, users with disabilities may use assistive technologies which are specialized software or hardware that assist them in performing tasks. An Example assistive technology is screen reader software which reads the information on a screen out-loud to the users, or displays it on a refreshable Braille-display for blind computer users. Other examples include screen magnifiers which enlarge the screen for user with low vision. Also specialized keyboards, single-key switches, head-sticks, eye-tracking systems, voice commands, and many other software and hardware are used by people with disabilities.

One aspect in providing equal access for people with disabilities is therefore to ensure that the Web content is compatible with the assistive technologies and adaptive strategies. This means providing adequate structure and semantics so that the content can be processed and rendered in different modes as needed. For instance, structures such as lists, headings, and links need to be identified within the document structure, such as HTML markup code, so that these can be utilized by software. Another aspect of providing equal access is to implement accessibility features within the Web content to support the usage by people with disabilities. For instance, to improve the user experience and the quality in which the Web content is delivered.

While accessibility evaluation is part of the Web development process, there may be varying motivations for carrying out evaluations. Some of the common motivations for evaluating the accessibility of Web content include the following situations:

- A programmer wants to test the efficacy of an accessibility feature in the prototype for a new Web application that is currently under development.
- A content author wants to ensure that the newly developed text meets the requirements and conventions of the organization before it is published.
- A project manager wants to learn about potential accessibility barriers on the Web site to estimate the current situation and plan for improvements.
- A Webmaster wants to ensure that all Web content published on a Web site meets the minimum technical (possibly also legal) requirements.

Some of the dimensions for Web accessibility evaluations are therefore the thoroughness and the scope. In the first scenario highlighted above, the Web developer is carrying out a focused evaluation on a specific feature. While the scope of such an evaluation is limited, it is ideally carried out thoroughly before the accessibility feature is deployed on the actual Web site. This is the opposite situation to the project manager who may carry out a very coarse evaluation on a number of different Web pages to get an overview of the overall accessibility of a Web site, or to the Webmaster who needs to carry out a sufficiently broad and thorough evaluation.

1.1.1. Context

The accessibility of the Web for people with disabilities is dependent on several components [WAI, 2]. The most elementary component of Web accessibility is the accessibility support provided by the underlying Web technologies. If Web technologies such as HTML, CSS, or others do not support the accessibility requirements of people with disabilities, then any Web content that is developed using these technologies will also not meet these requirements. An example of an accessibility feature that has been built directly into HTML is the `alt` attribute, which can be used to provide equivalent alternatives for images in the form of text. This and many more accessibility features have been built into the Web technologies provided by the W3C [WAI, 10]. Also many of the non-W3C Web technologies such as Flash, PDF, or Word are undergoing continual improvements with regard to their support for accessibility.

Web browsers and media players are equally critical in making the Web accessible for people with disabilities. If browsers and media players (called user agents) do not provide support for the accessibility features provided by the Web technologies, then there is no point in adding these accessibility features in the first place. For instance, if Web browsers do not support the HTML `alt` attribute, for example by displaying the text equivalents to the users or by making them available to the assistive technologies, then these accessibility features become useless. Web browsers and media players are the points of contact between the users and the Web, and must therefore observe the accessibility requirements of the users. This means providing an accessible user interface for the user agent software, rendering the accessibility features in the Web content, and providing APIs or other means of interacting with assistive technologies.

Similarly, the authoring tools and evaluation tools are the points of contact between the Web developers and the Web content being developed. They are vital for supporting the production of accessible content. For instance, authoring tools could prompt the users to provide the text equivalents for the images that they want to publish. In the case of HTML the authoring tool could insert the text provided by the author into to the `alt` attribute without requiring that the author knows HTML. This is an important aspect because the majority of the Web content is developed by non-technical authors who need to be able to rely on the authoring tools to help them meet the accessibility requirements. In fact, they may often not even know the specifics of the accessibility requirements, but can still contribute effectively to the evaluation process.

Since Web accessibility encompasses these components, the evaluation of Web accessibility is in the formal sense an assessment of the underlying Web technologies, the authoring tools, the browsers, as well as the media players. However, the more common implication of Web accessibility is the development of Web content that is accessible for people with disabilities. It is the perspective of Web developers who are creating Web content but have no influence on the development of the other components. In turn, the more common understanding for the context of Web accessibility evaluation is the assessment Web content for accessibility. It also contains an implicit assumption about the current state of the remaining Web components.

It is important to remember that these definitions for Web accessibility and Web accessibility evaluation are bound to the assumptions made about the remaining components outside the Web content. As these components evolve and change, the assumptions made may need to be revised in order to maintain an accurate understanding for Web accessibility and evaluation. Also, by neglecting these factors in an evaluation process there is a risk of missing potential accessibility barriers or of failing to identify the causality. For instance, if a developer relies on specific accessibility features that are not supported by user agents then the Web content may be effectively inaccessible. Conversely, if users can not access the Web content then it could be an issue of the user agent that they are using rather than of the Web content itself.

1.1.2. Background

People with disabilities encounter different kinds of barriers on the Web even though, like all users, they may not be able to articulate what is stopping them from being able to use the Web content effectively. Technical barriers are directly related to the way in which the content has been programmed and coded. For instance, sometimes section headings are not coded as such within the HTML or PDF documents but are only presented visually by using bold and larger font-size. These types of headings will not be recognized as such by browsers and assistive technologies, and will therefore not appear in the headings outline view of the documents. In turn, people who are using software to navigate the content by headings, such as most screen-reader users, will not be able to use the content effectively by scanning the headings.

In other cases people with disabilities can encounter functional barriers that are inherent to the way in which the accessibility features are implemented. For instance, providing navigational cues in the document structure can help users to identify their current position within the Web content and to orient themselves. However, providing too many cues can be distracting to the users and could actually disorient them in some cases. These types of issues are usually less technical but they require some basic knowledge about how people with disabilities use their computers and how they interact with the Web. They are generally also directly related to the design concepts of the Web content, and are therefore especially important to consider during the early requirements analysis and design stages of the development processes.

Another facet of accessibility barriers is a differentiation between objectively measurable and qualitative requirements. Examples of objectively measurable requirements are providing `alt` attributes for images or headings for rows and columns of data tables. Each of these tests can be carried out objectively, regardless of the context of the Web content or the knowledge of the evaluators. The outcomes of such tests are pretty much consistent among different persons or tools evaluating the same Web content because the requirements are not subject to any bias. Qualitative requirements however usually need some level of knowledge of how they relate to the use of the Web content by people with disabilities, and are usually strongly dependent on the context of the Web content. An example of a qualitative requirement is ensuring that the `alt` attributes provide equivalent alternatives that adequately describe the respective images.

Only few accessibility requirements can be fully automated and most of them require human judgment to evaluate. A challenge for many organizations in ensuring that their Web sites are accessible for people with disabilities is to optimize the distribution of the tasks and processes between the different individuals involved in the development and maintenance of the Web content. For instance, to make sure that Web designers adhere to the requirements for reading order, color combinations, and text styles when they are developing the ‘look and feel’; that programmers adhere to the requirements for content structure and interaction when they are developing Web applications; or that content authors adhere to the requirements of reading level, information design, and equivalent alternatives when they are publishing Web content.

Web accessibility evaluation is ideally carried out during the development of the Web content to ensure that it is published accessibly. It is also essential to carry out periodic evaluations as a quality assurance measure, to monitor and continually improve the level of accessibility. In some cases evaluations may be more exploratory to learn about potential accessibility barriers or how the accessibility features are being used (to further improve the user experience), and in other cases evaluations may be only intended to confirm that the Web content meets the set requirements for accessibility. Web accessibility evaluations may be carried out by in-house developers or by acquiring expertise, depending on the skills and knowledge of the developers.

1.2. Motivation

Equal access to the information society is a human right. It has been recognized by the United Nations (UN) in its Convention on the Rights of Persons with Disabilities [UN-CRPD], which entered into force on 3 May 2008. Also the European Commission (EC) as well as many other governments around the world is addressing the discrimination of people with disabilities in their policies or legislations [WAI, 13]. Providing accessible Web sites is an obligation and a social responsibility rather than a voluntary option for motivated individuals.

However, studies show that an opposite situation is the reality and that only few Web sites are accessible for people with disabilities. In 2006 the UN commissioned a study that surveys the accessibility of 100 Web sites from around the world, to identify that only approximately 3% comply with existing standards[1]. In 2008 the EC commissioned a comprehensive study that surveys the broader scope of eAccessibility within the European Union (EU), and identifies similarly low values for the compliance of Web sites to existing accessibility standards[2].

The primary reason for this severe discrepancy between the ideal goal of an inclusive society and the reality of inaccessible Web sites seems to origin from:

- **Lack of awareness** – many developers and Web site owners are unaware of the issues that people with disabilities encounter while using the Web. They are often unaware of the availability of standards for Web accessibility and of the technical solutions.
- **Lack of education** – many developers and Web sites owners lack sufficient education and training in the domain of Web accessibility, in order to be able to implement the accessibility requirements for people with disabilities effectively and efficiently.
- **Lack of tool support** – many software tools that are used to create Web content do not generate code that is sufficiently accessibility, and do not provide adequate support to assist Web developers in improving the accessibility of the Web content.

Web accessibility evaluation tools also play a considerable role in reducing the effort required to meet the accessibility needs of people with disabilities. They can help explore Web content to learn about potential accessibility barriers. They can also help learn about the requirements for accessibility and how to meet them in a practical way. Web accessibility evaluation tools are also essential for determining and managing the level of accessibility in larger Web sites, and are therefore crucial for meeting the paramount goal of providing equal access to the Web.

At the same time, there seems to be a need for research and development activities to further improve the performance and functionality of Web accessibility evaluation tools. There is a demand to develop new approaches for tackling Web content, especially in today's interactive and application-driven Web. For instance, heuristics and fuzzy algorithms are rarely used in many tools, even though they could provide valuable feedback to the developers. Moreover, there is a considerable potential in semantic Web technologies to facilitate a collaborative environment for development and evaluation tools to work together more effectively.

Also outside the domain of Web accessibility there seems to be an opportunity for advanced quality assurance tools that help mange basic compliance with the Web standards, privacy and security requirements, as well as generic quality assurance. Pursuing the enhancement of tool support in Web accessibility evaluation is potentially beneficial for Web quality assurance.

---

[1] Nomensa: http://www.nomensa.com/resources/research/united-nations-global-audit-of-accessibility.html

[2] MeAC: http://ec.europa.eu/information_society/activities/einclusion/library/studies/meac_study/index_en.htm

1.2.1. Auxiliary Benefits

Besides the ultimate goal of providing equal access for people with disabilities, there are also numerous business opportunities for evaluating and implementing Web accessibility [WAI, 4]. Accessibility requirements for people with disabilities enlarge the market reach and provide benefits for the majority of the Web site users. In 2003 Microsoft commissioned a study to survey the market for accessible technologies[3], to identify that while only approximately 15% of the population is considered as having some form of disability, approximately 25% of the population is very likely to benefit from the use of accessible technologies. Moreover, another approximately 37% is likely to benefit from the use of accessible technologies, so that one can roughly estimate that over 60% of the working-age adults benefit from accessible solutions.

1.2.1.1. Mobile Users

Nokia forecasts that the usage of the Web through mobile devices will soon outnumber the usage of the Web through desktop computers[4]. It is therefore essential that the Web can work equally well on desktop computers, as well as on compact cell phones with typically minimal screens and minimal set of keys. There are many overlapping requirements between designing content for mobile devices and making it accessible for people with disabilities [WAI, 11]. For instance, making the Web content usable by keyboard enables people who can not use the mouse, such as users with physical disabilities or blind users, to operate the Web content. This also benefits many users with mobile phones that do not have pointing devices such as rocker switches, and can therefore not operate Web sites that rely on input through a mouse.

1.2.1.2. Older Users

Statistics from around the world have shown that as we grow older we develop ageing-related functional limitations [W3C, 4]. This is typically reflected in the gradual deterioration of the vision, hearing, memory, or dexterity capabilities; in some cases several of these symptoms can occur at the same time. It is clear that the needs of older users on the Web correlate very strongly with the needs of people with disabilities. At the same time, the population of older people who are using the Web is growing very rapidly in many parts of the world. In many of these countries older users are considered to have a substantial purchasing power. Developing accessible content has a critical impact on increasing the audience-reach to older users, and is therefore a direct business benefit. It increases the return of investment (ROI) substantially.

1.2.1.3. General Usability

While usability and accessibility are separate disciplines, conceptually they are related and share some fundamental approaches. In general, usability has an impact on the experienced satisfaction of the users, while accessibility has a more focused impact on the efficiency of people with disabilities. Some of the requirements that address the accessibility of the Web content by people with disabilities also benefit other users. For example, the requirement for clear and consistent navigation mechanisms in essential for people who rely on the navigation cues to orient themselves within the Web content. This includes blind users but also users with cognitive disabilities. At the same time, clear and consistent navigation mechanisms also benefit other users, especially novice computer users or who do not know the Web site.

---

[3] Forrster Research: http://www.microsoft.com/enable/research/phase1.aspx

[4] Source: Nokia forecast for 2006, and forecast report by Morgan Stanley Communications Equipment Research

1.3. Problem Definition

The objective of this study is to explore potential opportunities for increasing the tool support in Web accessibility evaluation. Evaluation is hereby defined as a quality assurance measure to ensure that the accessibility needs of people with disabilities are met throughout the entire development process for Web content. In particular it includes the design, the implementation and the maintenance stages. Evaluation in the general sense is not confined to quality control which takes place after the production but is a continuous quality assurance procedure that is an intrinsic part of the Web development process.

Currently Web accessibility evaluation tools focus primarily on the evaluation of Web content after its design and development stages. For instance, many evaluation tools are designed to crawl through Web pages in order to help assess the accessibility of collections of Web pages such as entire Web sites. While other types of evaluation tools focus on evaluating individual Web pages, they also seem to follow the post-production quality control paradigm. Measuring the accessibility of existing Web content is clearly an essential use-case. However, there is an untapped potential to broaden the coverage of Web accessibility evaluation tools to the other stages of the Web development process. For instance, other use-cases for evaluation include:

- **Requirements Analysis** – do the requirements for the development or the re-design of Web content include adequate considerations for accessibility?
- **Prototyping and Design** – are the layouts, visual designs, and templates functionally and technically compliant with the accessibility requirements?
- **Code Programming** – does the source code including markup and application scripts comply with the accessibility requirements?
- **Content Production** – does the main content including text, images, and multimedia comply with the accessibility requirements?
- **Conformance Audits** – does the Web content (in this case usually entire Web pages or Web sites) as a whole conform to formal standards for Web accessibility?
- **Enhancement Audits** – are there improvements that can be made to further enhance the experience of people with disabilities using the Web content?
- **Change Management** – does the addition, modification, or removal of Web content lead to accessibility barriers, possibly in other parts of the Web content?
- **On-Going Monitoring** – has the Web content (in this case usually entire Web sites) undergone changes that impact its overall level of accessibility?

Moreover, Web accessibility evaluation tools seem to focus on a specific group of users who are more technical or who have knowledge in the domain of Web accessibility evaluation. It is a common practice that Web accessibility evaluation tools, especially the automated tools, generate reports of their findings. Despite the fact that some of these tools can also produce summaries and diagrams to provide higher-level overviews on the evaluation findings, these types of reports are generally more apt for technical users who are familiar with bug reports and similar formats. In other cases, evaluation tools require that their users have knowledge about how to carry out evaluation tasks, or how to interpret the results or feedback provided by the tools. Web accessibility evaluation tools needs to better address the heterogeneity and diversity of the developers in order to further assist them in carrying out different evaluation tasks that are part of a comprehensive quality assurance measure. The evaluation tools need to better fit into the production chain in order for them to be optimally effective for developers.

This study aims to analyze the parameters that influence the dimensions of Web accessibility evaluation processes. For instance, which parameters change between focused evaluations of specific features and coarse evaluations to learn about the overall performance (as discussed in section 1.1. Introduction)? When during the development process are the different types of evaluation tasks ideally carried out and which types of developers are involved in these tasks?

It is also equally important to analyze the characteristics of the accessibility requirements for people with disabilities in order to better understand how these influence the evaluation tasks. For instance, can the evaluation of these requirements be partially automated and could tools provide assistance for non-automatable requirements? What skills and experience do different types of accessibility requirements need in order to be evaluated effectively and efficiently?

Finally, it is also important to explore currently available Web accessibility evaluation tools and learn about how they generally work. Which types of situations do they address and what type of functionality do they typically provide? How do current evaluation tools fit into the development process and how do they integrate with other types of tools such as authoring or generic quality assurance tools? What are the benefits and drawbacks of using such tools?

These questions are all part of the review on the current state of the art in Web accessibility evaluation. The answers to these questions are expected to give a solid foundation that helps identify some of the opportunities for additional tool support in Web accessibility evaluation. In particular this study aims to pursue the use of structured data formats to describe evaluation procedures, evaluation results, and Web resources in machine-readable forms. These types of descriptions enable the exchange of semantically rich information between different types of tools such as authoring tools, evaluation tools, and generic quality assurance tools. They are therefore promising techniques that could enable ad-hoc integration of different types of tools.

Based upon these findings the study is set out to propose and discuss potential approaches for a distributed model of tools that can connect ad-hoc depending on the overall evaluation and development process for the Web content. It should investigate the potential interoperability of evaluation tools that have been created to help carry out different evaluation tasks. These evaluation tasks are possibly carried out by different developers who typically have varying backgrounds in technical skills or expertise in Web accessibility and evaluation.

While this approach is significantly different than what seems to be commonly employed by Web accessibility evaluation tools, there is reason to believe that it is principally feasible. The domain of software engineering provides mature concepts and strategies for quality assurance that could be reused in the domain of Web engineering. There are numerous software quality assurance tools that help users model their software, examine their code, and debug the issues. There are considerable differences between code development and user interface design that could lead to challenges in adopting many of these concepts and strategies, but many of the basic approaches and models should be promising for the domain of Web quality assurance.

2. Current State-of-the-Art in Web Accessibility Evaluation

As introduced in section 1.1.1. Context, there are essential components of Web accessibility that need to work together in order to ensure that Web content is accessible for people with disabilities. The underlying Web technologies such as HTML, CSS, SMIL, SVG; the user agents such as browsers and media players; the authoring tools and evaluation tools; as well as the Web content itself need to support accessibility. The W3C Web Accessibility Initiative (WAI) [WAI] developed a suite of accessibility guidelines that complement each other and address each of the components described above [WAI, 5-7]:

- **Web Content Accessibility Guidelines (WCAG)** – defines criteria for the creation of accessible Web content for people with a variety of disabilities, including people with different forms of visual-, hearing-, physical-, neurological-, or cognitive disabilities.
- **Authoring Tool Accessibility Guidelines (ATAG)** – defines criteria for software that is used to generate Web content, so that the interface of these tools is accessible for people with disabilities and so that the content generated by these tools is accessible.
- **User Agent Accessibility Guidelines (UAAG)** – defines criteria for software, such as browsers and media players, that are used to render Web content, so that they provide accessibility features and so that they can be compatible with assistive technologies.

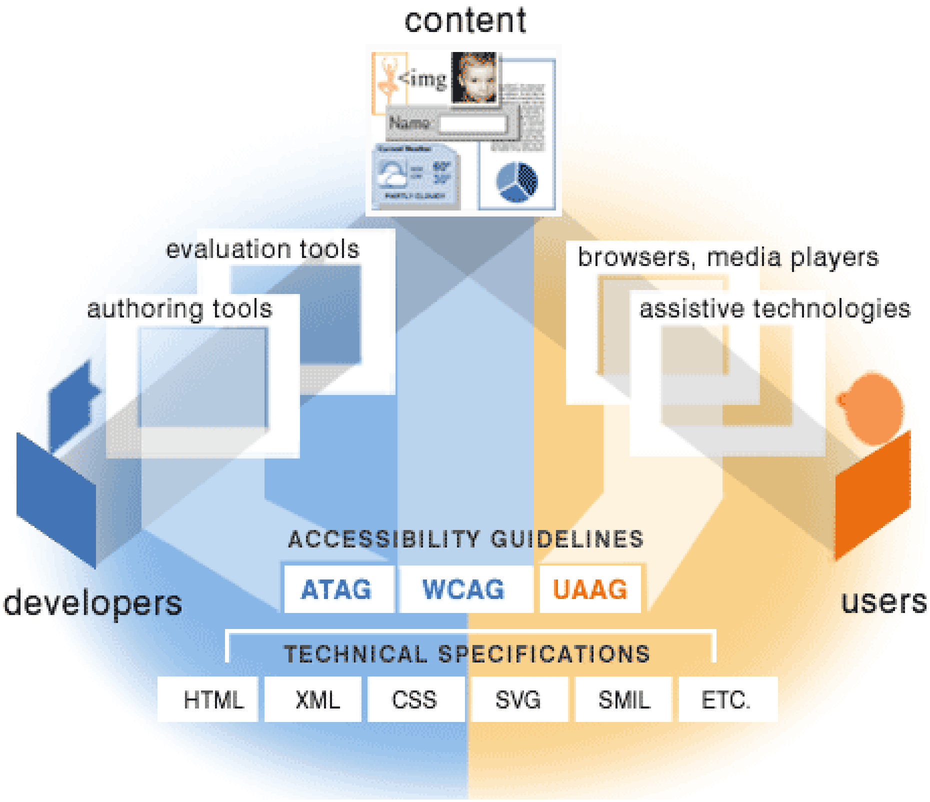


Figure 1: Essential Components of Web Accessibility

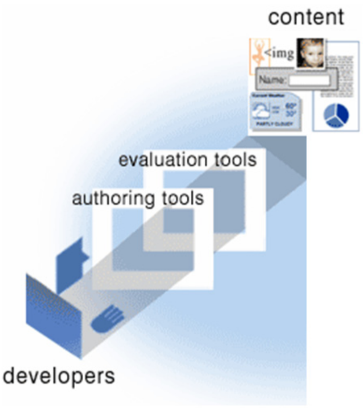
This section explores the current state-of-the-art in Web accessibility evaluation. It examines the current standards for Web accessibility and the processes for Web accessibility evaluation. This section also outlines the relationship between the processes for Web development and accessibility evaluation processes, and how these can be assisted by software tools.

2.1. Technical Standards for Web Accessibility

The accessibility features built into the technical specifications of the core Web technologies play a vital role in Web accessibility. The support for these accessibility features within the Web browsers, media players, and assistive technologies plays an equally decisive role in the evaluation of the accessibility of Web content. For instance, the HTML specification provides the `longdesc` attribute which can, in addition to the `alt` attribute that can convey the textual equivalent for the purpose of the images, link the images to extended descriptions. While this feature seems like a useful accessibility solution, the `longdesc` attribute is actually not widely supported by browsers and assistive technologies. This means that in practice it is not a viable accessibility solution on its own due to the broad lack of support for the feature.

We are also observing an increased deployment of formats such as Adobe PDF, and those of Microsoft Office, Open Office, or other office applications on the Web. While these formats were traditionally used as formats for desktop documents, they are being increasingly used on the Web for publishing information. Some of these formats provide varying levels of support for accessibility, such as capabilities for identifying the document structures, for providing the captions for images, or for defining the reading flow for text. Because many of these formats were initially proprietary, it was difficult to develop tools such as accessible browser plug-ins, assistive technologies, or evaluation tools which could support them directly. Some of these formats, such as ODF [ISO 26300] and PDF/X [ISO 32000], have been recently standardized by ISO [ISO]. This allows more insight into the accessibility provided by these formats, and enhances the interoperability and the availability of accessibility solutions.

Also multimedia formats are being increasingly deployed and used on the Web. Formats for video and audio also need to provide accessibility features for people with disabilities. For instance, these formats need to facilitate subtitling[5], transcription[6], or audio descriptions[7] for multimedia presentations. Only few of the widely used multimedia formats such as [MPEG] have been formally standardized, the majority of the formats such as Microsoft Windows Media, Real Media, Quick Time, or Flash Video are proprietary. Unfortunately, the current situation of licensing for these formats has created severe interoperability issues amongst the media players. This has a significant impact on the development of accessible multimedia content, as developers need to be aware of the capabilities of the individual media players and to create tailored content rather than to be able to rely on a common behavior of the tools.



Web accessibility is therefore a moving target that strongly depends on the current state of the Web technologies and the support provided by the user agents for the accessibility features. The accessibility standards must therefore account for the current state of the Web with the multitude of Web technologies, increasing number of browsers and assistive technologies, and a wide variety of Web applications. This section examines some of the accessibility standards that are related to the production of Web content. It explores how authoring and evaluation tools work together to help developers create and publish accessible Web content.

---

[5] Subtitles: captions that are presented with a video presentation and describe the audio for the hard of hearing.

[6] Transcripts: textual files that provide equivalent alternatives for the visual tracks of multimedia presentations.

[7] Audio Descriptions: audio narration that provides descriptions of the visual tracks of multimedia presentations.

2.1.1. International Standards

The three W3C Web accessibility Initiative (WAI) [WAI] guidelines for Web accessibility introduced earlier have been internationally recognized as the standard for Web accessibility. Especially the W3C Web Content Accessibility Guidelines (WCAG) [WAI, 5] has become widely adopted by many organizations and governments from around the world. Currently the following versions of these WAI Guidelines are operational and generally regarded as stable:

- W3C Web Content Accessibility Guidelines (WCAG) 1.0 [W3C, 5-10]
- W3C Authoring Tool Accessibility Guidelines (ATAG) 1.0 [W3C, 12-14]
- W3C User Agent Accessibility Guidelines (UAAG) 1.0 [W3C, 15-17]

W3C is currently working on a new generation of these guidelines to meet the current state of the Web and address the current challenges that people with disabilities face. In this study the focus is this new generation of the guidelines, in particular on WCAG 2.0 which is expect to become a final Web standard during 2008[8]. This section examines the following standards:

- W3C Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18-21]
- W3C Authoring Tool Accessibility Guidelines (ATAG) 2.0 [W3C, 22-23]

These standards are developed in the collaborative environment of the W3C with stakeholder representatives from research, disability organizations, industry, policy, as well as experts and individuals in the field. W3C standards are developed through the W3C Process that ensures a transparent and consensus-oriented approach. The development process is iterative to provide opportunities for contribution by the public [WAI, 15]. As with all W3C standards, the WAI Guidelines are published with royalty-free licensing. This is an important aspect as it can be made available to all users including developers, researchers, and accessibility advocates with out licensing fees. This promotes the adoption and usage within the Web community.

Unfortunately diverging derivatives of WCAG are creating fragmentation in the field of Web accessibility. Diverging standards have a negative impact on Web developers as well as on authoring tools, browsers, media players, assistive technologies, and evaluation tools. It is often not economically feasible to support all the different standards so that the solutions and tools are not easily reusable. In the worst case, diverging standard may be incompatible and the competing requirements can not be achieved by Web developers or software producers.

There are different motivations and drivers for standards fragmentation [WAI, 14], many of which are organizational and are of non-technical nature. One significant driver is however inherent to the technical design of WCAG 1.0 as it was in some areas open to interpretation, and in other areas it made assumptions about the capabilities of the user agents and assistive technologies. As the user agents and assistive technologies evolved, many requirements that were built around assumptions of their capabilities became outdated very quickly.

Many of the diverging standards were therefore initially developed as an interpretation for WCAG 1.0, also to help developers evaluate if they have met the requirements or not. They were not specifically designed to remove, add, or modify the overall requirements but rather to define more granular steps for meeting these requirements or for evaluating them. Other standards were developed with less attention for harmonization of standards, and now pose some additional challenges for defining a generic approach to Web accessibility evaluation.

---

[8] As announced by W3C on 30 April 2008 at http://www.w3.org/WAI/WCAG20/wcag2faq.html#done.

2.1.1.1. W3C Web Content Accessibility Guidelines (WCAG) 2.0

The W3C Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18] is the successor of the WCAG 1.0 standard. WCAG 2.0 is, at the time of writing this study, in the final stages of development and currently published as a W3C Candidate Recommendation. This is the pre-final development stage according to the W3C Process [WAI, 15], and is the stage in which the document is regarded as stable and is published to gather feedback about its maturity. If the document proves to be implementable in practice, in other words that its requirements are achievable on real and live Web sites, the document will proceed to the final development stage before publication as a W3C Recommendation (the term used to denote W3C standards).

WCAG 2.0 is designed to overcome many of the design issues that were intrinsic to the prior version 1.0. It is specifically designed to meet the following objectives:

- **Objectively Testable** – requirements must not be open to subjective interpretations.
- **Technology-independent** – requirements do not relate to specific Web technologies.
- **Future-proof** – requirements must apply to the current and future Web technologies.

From these objectives it was clear that the guidelines now needed to primarily address the functional requirements of people with disabilities interacting with Web content, rather than to prescribe technical approaches for accessible Web design. Four Principles are defined:

- **Principle 1: Perceivable** – Information and user interface components must be presentable to users in ways they can perceive.
- **Principle 2: Operable** – User interface components and navigation must be operable.
- **Principle 3: Understandable** – Information and the operation of user interface must be understandable.
- **Principle 4: Robust** – Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

WCAG 2.0 Principles – http://www.w3.org/TR/WCAG20/

Each Principle is a category that has several Guidelines. Guidelines in WCAG 2.0 are similar in intent to those of WCAG 1.0. They are however clearer, especially when they are read in the context of their respective Principle. For instance, WCAG 2.0 Guideline 2.1 says:

> “Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard” – http://www.w3.org/TR/WCAG20/#keyboard-operation

The WCAG 2.0 guidelines are still fairly abstract even though they are more detailed than the principles. Additional precaution has been taken to ensure that the guidelines in WCAG 2.0 remain agnostic to any specific Web technologies. For instance, the quoted guideline above applies to HTML and XHTML equally to PDF, Flash, or Word. It is a requirement for people with disabilities to be able to (functionally) operate the Web content effectively, regardless of how it has been implemented or realized technically.

The Working Group developing WCAG also moved away from the concept of Checkpoint in version 2.0, and adopted the concept of Success Criteria. This may be somewhat subject to a rather philosophical discussion but it reflects that the requirements are criteria that need to be fulfilled functionally. In practical terms this means that these Success Criteria are usually not evaluated directly, but rather using Techniques. One evaluates if Techniques that meet the Success Criteria were adequately used. For instance, WCAG 2.0 Success Criteria 2.4.1 says:

> “A mechanism is available to bypass blocks of content that are repeated on multiple Web pages” – http://www.w3.org/TR/WCAG20/#navigation-mechanisms-skip

This Success Criteria is provided under the WCAG 2.0 Guideline 2.4 “Provide ways to help users navigate, find content and determine where they are”. It is a however still a functional requirement that needs to be manifested into the specific Web technologies. The appropriate implementation of such a requirement will depend on the accessibility features built into the specific Web technologies being used, and the availability of user agents the provide support for this feature. While these may change with time as the technologies evolve, it is expected that the functional requirement of this success criteria will have a longer lifetime.

Similarly to WCAG 1.0, this second version of the standard has separated the Techniques into its own document called Techniques for WCAG 2.0 [W3C, 21]. This is published as a W3C Working Group Note, which has a less rigorous development process than the official WCAG guidelines themselves. While the Techniques for WCAG 2.0 document can be viewed as a single file, it is actually an extensive collection of approaches for meeting the Success Criteria, and is not intended to be read from top to bottom. The Techniques are categorized by the primary Web technology that they address. Currently there are General, HTML, CSS, and Scripting techniques. Additional techniques for W3C technologies such as SMIL, SVG, and XForms as well as for non-W3C technologies such as PDF, Flash, or Silverlight are expected.

There is a sub-category of techniques called Sufficient Techniques. These provide means for adequately satisfying the respective Success Criteria. These techniques are not exclusive and it is principally possible to use other means to satisfy the requirement, ideally techniques that are well documented and widely accepted as solutions. In addition to the group of Sufficient Techniques there are Advisory Techniques, which are recommended best practices that go beyond the minimum requirement. There are also Failure Techniques which are common mistakes that fail the criteria. These could be seen as the ‘worst practices’ that should not be followed. The overall structure of WCAG 2.0 therefore looks as follows:

- **Principles** – Perceivable, Operable, Understandable, Robust – “POUR”
    - **Guidelines** – functional requirements for accessible Web design
        - **Success Criteria** – testable statements for each guideline
            - **Techniques** – General, HTML, CSS, Scripting, …
                - **Sufficient Techniques** – bare minimum
                - **Advisory Techniques** – best practices
                - **Failure Techniques** – common mistakes
                - **(External Techniques** – not from W3C**)**

Along with the official WCAG 2.0 standard and the accompanying techniques document, two support documents are published as part of WCAG 2.0. The documents match the diverging needs of the different audiences that range from developers, evaluators, and authors to project managers, policy makers, and executives. Any of these readers may be more or less technical, and more or less experienced in Web accessibility. These additional documents are provided:

- **How to Meet WCAG 2.0** [W3C, 19] – provides a customizable and quick reference to the WCAG 2.0 guidelines and success criteria, and provides links to the techniques.
- **Understanding WCAG 2.0** [W3C, 20] – provides detailed background on the intent, benefits, and examples for each of the WCAG 2.0 guidelines and success criteria.

It is expected that most of the developers and evaluators will use the How to Meet WCAG 2.0 document as their main starting point. It provides a quick overview of all the Guidelines and Success Criteria, and pulls the currently available and relevant Techniques dynamically from the underlying database. For developers and evaluators who are new to Web accessibility or who occasionally need to lookup background information on the intent or purpose, there are links referring to the Understanding WCAG 2.0 document. This document is like a reference manual that contains a lot of valuable background information about the specific requirements.

The How to Meet WCAG 2.0 document is also customizable. There are toggles to display or to hide Success Criteria and Techniques that relate to multimedia, scripting, CSS, or others if they are not applicable to the respective Web content being developed or evaluated. It is also possible to adjust the target accessibility level, which will also result in displaying or hiding the respective Success Criteria. The How to Meet WCAG 2.0 document is thus a dynamic checklist that displays only the minimum information needed by the Web developers.

The following figure illustrates the WCAG 2.0 documents described above. It highlights the WCAG 2.0 document as the central Web standard, with the supplementary three documents derived from it. The How to Meet WCAG 2.0 document is a customizable quick reference that links to the more detailed Understanding WCAG 2.0 and Techniques for WCAG 2.0:

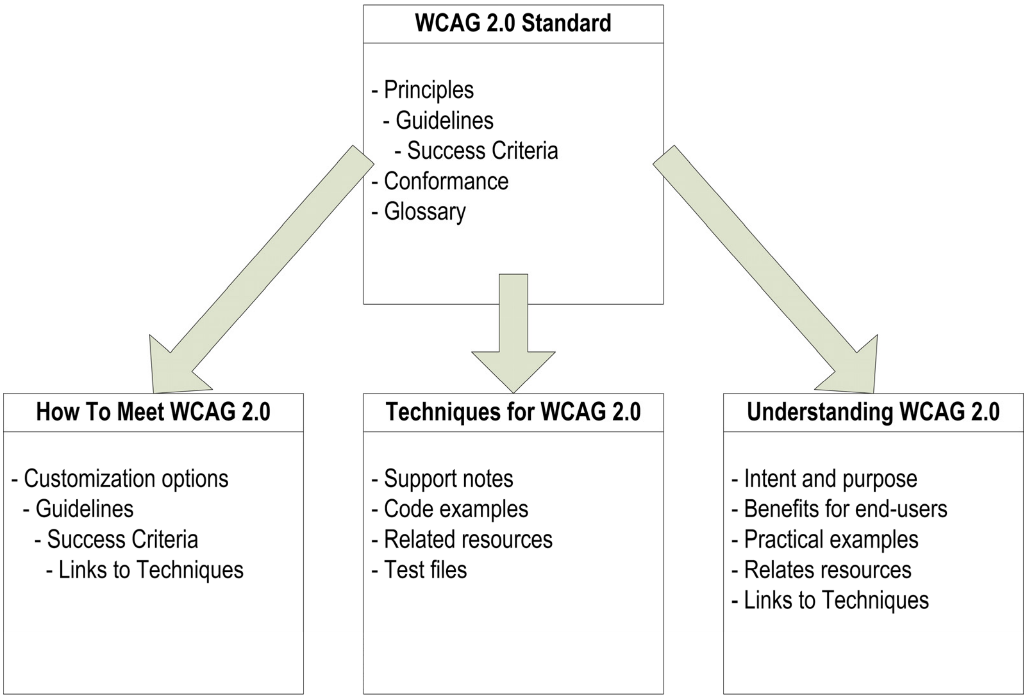


Figure 2: Overview of WCAG 2.0 Documents

To help readers transition from WCAG 1.0 to WCAG 2.0, W3C provides a comparison of the checkpoints and success criteria [WAI, 16]. While there is no pure one-to-one mapping of the two, there is a high degree of compatibility. This is because functional requirements of people with disabilities are the underlying intents in both versions. Developers who implemented the first version thoroughly will not be impacted significantly. Also evaluators who have learned to assess Web accessibility according to WCAG 1.0 will be able to carry over most of their skills and knowledge when they are using the new version. WCAG 2.0 does however provide many more features to address the current state of the Web. It also provides flexibility to be able to address different situations such as evolving Web technologies. It is however a larger suite of documents that needs to be first understood so that it can be used efficiently.

2.1.1.1.1. Evaluation Aspects

The Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18] provides far more support for evaluation than the previous version. As noted in the previous introduction, WCAG 2.0 is specifically designed to be more testable. Each Success Criteria is formulated to be a testable statement. For instance, WCAG 1.0 Checkpoint 14.1 "Use the clearest and simplest language appropriate for a site's content" is an example of a requirement that is open to interpretation. In order to make this requirement objectively testable, WCAG 2.0 Success Criteria 3.1.5 uses the following formulation that provides more clarity and testability:

> "When text requires reading ability more advanced than the lower secondary education level, supplemental content, or a version that does not require reading ability more than lower secondary education level, is available"
> – http://www.w3.org/TR/WCAG20/#meaning-supplements

The threshold 'lower secondary education level' is well defined by the International Standard Classification of Education of the UNESCO. It is therefore a standardized reading level, even though it may vary according to the specific region of the world, culture, or other factors[9].

Despite all this effort to make the requirements objectively testable, evaluators must be clear that for some requirements there is no solid yes/no boundary. It is in the nature of some of the requirements that there is a grey zone between correct and incorrect implementation. To take the Success Criteria cited above as an example, there are situations in which individual words or phrases require 'borderline' decisions (decisions that can go either way depending on the interpretation and context of the situation). This is especially when such words or phrases are used in unconventional or irregular ways[10]. Another example that highlights such situations is the judgment of the appropriateness of 'equivalent text alternatives'. Since by virtue this is an interpretation from one form of the information to another (for example from audio content to text or from visual content to text), there may be differences between such 'transformations'.

It is however the role of technical standards to minimize the spectrum for the 'grey zone', and to minimize the potential inaccuracies that can occur. WCAG 2.0 has taken significant steps to manage the 'space of interpretation' in its requirements. For instance, WCAG 2.0 Success Criteria 1.1.1 enumerates specific situations for 'text alternatives' such as alternatives for tests or for decorations [11]. Text equivalents for decorative images are therefore handled differently than the equivalents for informative images. This means that it is less likely that one evaluator reaches a contradicting conclusion to another, because many of the key situations are outlined in the requirements. Without separating some of the key situations, evaluators would be left with more decisions to make and therefore more chances of reaching diverging results.

A lot of valuable information such as scenarios and examples that describe the background and intent for the requirements is part of the Understanding WCAG 2.0 document [W3C, 20] that accompanies the actual standard. While the information provided in this document is not normative, it provides useful clarifications to help educate developers and evaluators. Also, less experienced developers and evaluators may find additional guidance on specific context and situations and therefore further reduce the discrepancy in interpretation. The document is however lengthy and it is therefore intended to be used as a reference manual.

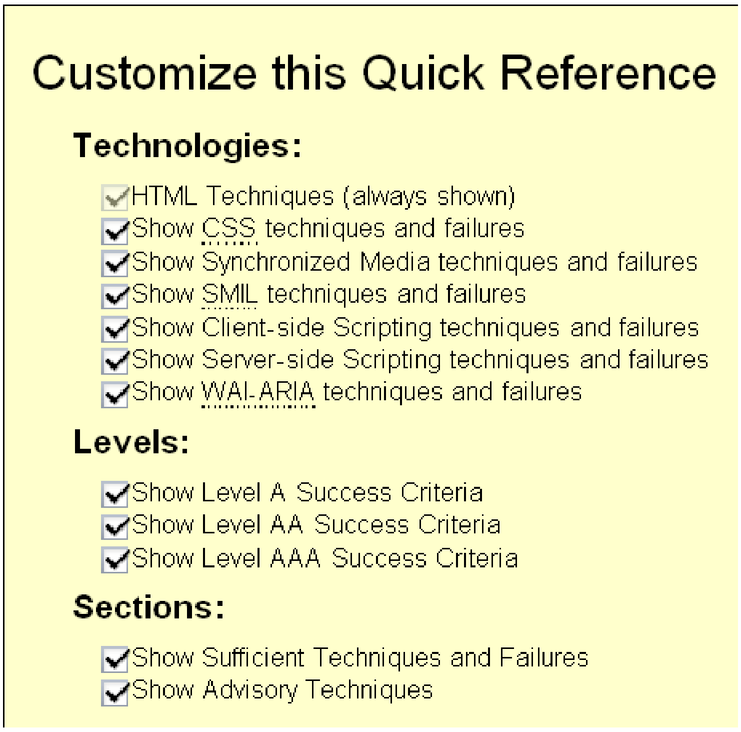
---

[9] At the time of writing this study, Success Criteria 3.1.5 is under discussion by the responsible Working Group.

[10] WCAG 2.0 tries to address this through Success Criteria 3.1.3 as a counterpart to 3.1.5, but the issue of having 'borderline' situations for individual words or phrases is still applicable to both of these Success Criteria.

[11] The term 'Test' in WCAG 2.0 Success Criteria 1.1.1 refers to exams for performance in educational settings.

In practice, the main starting point for most evaluators will be the How to Meet WCAG 2.0 document [W3C, 19]. As introduced in the previous sub-section, it provides a customizable interface to toggle the display of the relevant Success Criteria and Techniques. Evaluators can select the technologies that are used in the Web content or for which they want to evaluate for. The HTML Techniques are always selected as it is considered to be the basic technology for most Web sites. As more techniques for the different Web technologies are added to the techniques database, more options will be provided for evaluators to select from. Also the priority level can be selected by evaluators, depending on the target level of accessibility that they are evaluating for. Finally, evaluators can select to show the techniques, and especially the Advisory ones. This is an important aspect to help manage the amount of information that is displayed. Many developers and evaluators may only want to know about the least amount of information to fulfill a specific task that has been assigned to them.



Customize this Quick Reference

Technologies:

☑ HTML Techniques (always shown)
☑ Show CSS techniques and failures
☑ Show Synchronized Media techniques and failures
☑ Show SMIL techniques and failures
☑ Show Client-side Scripting techniques and failures
☑ Show Server-side Scripting techniques and failures
☑ Show WAI-ARIA techniques and failures

Levels:

☑ Show Level A Success Criteria
☑ Show Level AA Success Criteria
☑ Show Level AAA Success Criteria

Sections:

☑ Show Sufficient Techniques and Failures
☑ Show Advisory Techniques

After adjusting the dynamic checklist through the Quick Reference interface, evaluators will likely refer to the Techniques listed for the Success Criteria. Each Success Criteria may have several Techniques that may be organized in specific groups or combinations. For instance, for WCAG 2.0 Success Criteria 1.1.1 that was discussed on the previous page, the Techniques are organized into groups that reflect the specific situations that are elaborated in the wording of the requirement. Within "Situation A: If a short description can serve the same purpose and present the same information as the non-text content" for instance, there is a combination of two Techniques that have to be used to meet the Success Criteria according to the current set of Techniques in WCAG 2.0. The first is a General Technique describing how to author 'short text alternatives', and the second needs to be one from a selection of HTML Techniques.

Each Technique includes a dedicated section describing a specific test procedure to evaluate if the technique was correctly implemented. For instance, WCAG 2.0 Technique H37 "Using `alt` attributes on `img` elements" defines the following evaluation steps:

1. Examine each `img` element in the content.
2. Check that each `img` element which conveys meaning contains an `alt` attribute.
3. If the image contains words that are important to understanding the content, the words are included in the text alternative.

List 1: WCAG 2.0 Technique H37 Test Procedure – http://www.w3.org/TR/WCAG20-TECHS/H37

Despite WCAG 2.0 starting at the very high-level of the abstract Principles, it drills-down to very specific evaluation guidance and testing procedures. Evaluators can customize the view on this rich resource of information by displaying or hiding the information relevant to them.

2.1.1.1.2. Specifics and Caveats

The Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18] is, at the time of writing this study, still a draft work in progress so there is little experience with it in practice. There are some opportunities for improving issues encountered in its current testing phase of the development process, and so it is expected to be a robust and stable Web standard when it is finalized. However, there are some observations that can be made at this stage regarding its conceptual and architectural design. There are important differences between the first and second version of the guidelines. These differences have an impact on some of the approaches for Web accessibility evaluation, and on the tools that support evaluation.

In WCAG 2.0 more attention has been given to making the requirements as pure as possible, so that they cover only the designated needs of people with disabilities. They are also pure in the sense that they only relate to the Web content, and try to decouple and relationship to the underlying Web technologies and user agents as much as possible. One of the implications of this approach was a seemingly minor yet significant change to the requirement that was first defined by WCAG 1.0 Checkpoint 3.2 “Create documents that validate to published formal grammars”. The intent of this requirement was to provide valid HTML and CSS content, on which user agents and assistive technologies could rely. Besides the fact that this formulation is only applicable to markup languages such as SGML and XML variants (it does not apply to PDF, Flash, or Word documents because they have no formal grammars), it is argued that user agents and assistive technologies accept and sometimes even prefer code that is not valid according to the HTML or CSS specifications. This is of course a less optimal situation from an engineering perspective but it is a fact and relates to the implementation and support for HTML and CSS in the mainstream browsers, rather than to the specific needs of people with disabilities. WCAG 2.0 Success Criteria 4.1.1 has therefore chosen a different formulation to address these issues highlighted above; the requirement says:

> “In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features” – http://www.w3.org/TR/WCAG20/#ensure-compat-parses

This Success Criteria first limits itself to the family of markup languages then it enumerates properties that the content needs to have in order to be parsed and processed appropriately by browsers and assistive technologies. It is theoretically possible to develop code that does not validate to the specification, such as HTML, but still meets the requirements of WCAG 2.0.

Also the formulation of the requirements so that they are objectively testable has an impact on their coverage and applicability. When requirements need to provide a concrete threshold and an approach for measurement, their scope and reach needs to be understood and well defined. In some cases this means that ‘blanket requirements’ that had a broad coverage in WCAG 1.0, had to be broken down into individual and focused requirements. For instance, the WCAG 1.0 Checkpoint 3.2 “Use style sheets to control layout and presentation” is very broad and needs several individual steps to evaluate. In WCAG 2.0 there is no directly corresponding Success Criteria but there are several criteria that relate to the individual user needs that are part of this WCAG 1.0 requirement. In WCAG 2.0, Success Criteria 1.3.1 “Info and Relationships”, 1.3.2 “Meaningful Sequence”, 1.4.1 “Use of Color”, 1.4.4 “Resize Text”, 1.4.5 “Images of Text”, 1.4.8 “Visual Presentation”, 1.4.9 “Images of Text (No Exception)”, and 2.4.7 “Focus Visible” all relate to the different aspects presented by the WCAG 1.0 requirement. They are however more focused in scope and more specific in their formulation. This makes them easier to implement and to evaluate for in practice.

One of the ground-breaking aspects of WCAG 2.0 is that, unlike its predecessor, it allows the use of client-side scripts such as JavaScript[12]. For instance, the WCAG 1.0 Checkpoint 6.3 requires that the Web content works when scripts are turned off. This was mainly due to the fact that back in 1999 scripting was very inconsistently supported by the mainstream browsers such as Netscape Navigator and Internet Explorer. Also assistive technologies such as major screen readers had little or no support for JavaScript. Scripting was also most commonly used for simple tasks such as for checking form submissions or for creating effects for navigation menus (for example 'drop-down or 'fly-out' effects). There were all features that could often be built onto an accessible HTML base using a 'layering' approach. For instance, the form checking script is only triggered if scripting is available in the user agent but it is not required for the form to be submitted. A similar approach can be used for constructing navigation menus that used overlay scripts and that have fallback mode.

In today's Web however, scripting has become a central feature and whole Web applications are built using scripts. More specifically 'Ajax' – Asynchronous JavaScript and XML – set a buzz in the Web developer scene. It is a technique that allows content to be fetched from the server dynamically and be added to the Web pages[13], for instance to load database entries or to retrieve other types of data dynamically. Meanwhile client-side scripting is standardized and is largely supported in major browsers and assistive technologies. The issue for people with disabilities is however that HTML provides too few semantics so that user agents are unaware of what is happening while they are executing the scripts. A 'Tree View' component is often coded as a complex nesting of `div`, `span`, and `a` elements. Visually these elements are shown or hidden depending on the user interaction but browsers and assistive technologies have no means of identifying the actual 'Role' and 'State' of each of these items. In turn they can not render this information for the users who can not see the screens, and who therefore need the information to be presented in a different mode (for example through audio).

The failure to provide sufficient semantics is however a shortcoming of the underlying Web technology, in this case it is the HTML 4 family. It is expected that HTML 5 which is under heavy development may remedy many of these issues. People with disabilities however need solutions to be available today. For this reason, W3C developed a suite of documents called Accessibility Rich Internet Applications (WAI-ARIA) [W3C, 26]. WAI-ARIA provides the taxonomy of semantics that can be used in XML-based languages and HTML. It can be used to identify the type of object and the functionality that it provides. For instance, to say that a `div` element is actually a button that is currently pressed, the following code can be used:

```
<div role="button" aria-pressed="true">Pressed!<div>
```

The values in the `role` and the `aria-…` attributes are defined by the WAI-ARIA specification. They are currently being implemented by most of the mainstream Web browsers, including Firefox, Opera, Safari, and Internet Explorer. They also map to the accessibility API of the operating systems and can therefore be accessed by assistive technologies. WAI-ARIA also provides 'Live Regions' which can be used by Web browsers to notify assistive technologies about changes in the Web content, for instance after an update to the content using an Ajax function call. WAI-ARIA is expected to be an interim solution that will be built into the Web technologies directly. For client-side scripting to be accessible it is essential that the Web content provides sufficient semantics about the different objects and about the functionality that they provide. This ways the can be operated independently of the interaction modality.

---

[12] Formally JavaScript is a dialect of the ECMA-262 standard [ECMAScript], but it is the common term used.

[13] The term 'Web page' in the context of scripting refers to the Web content in a browser observed by the user.

WCAG 2.0 has been designed with the assumption that Web technologies can provide the necessary semantics to make scripting accessible for people with disabilities. For the moment WAI-ARIA will need to be used for scripted Web content developed with HTML 4. Ideally future Web technologies such as HTML 5, Flash, and Silverlight will provide the semantics natively. WCAG 2.0 has a requirement for providing these semantics in the Web content. It is agnostic to whether these semantics are provided in the Web content by using WAI-ARIA or the native properties of a Web technology. WCAG 2.0 Success Criteria 4.1.2 says:

> “For all user interface components (…), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies”
> – http://www.w3.org/TR/WCAG20/#ensure-compat-rsv

As observed in many other examples throughout this passage, WCAG 2.0 only describes how the Web content must behave for it to be conformant with the requirements. It decoupled the requirements from the current state of Web technologies and from the support provided by the user agents and assistive technologies. The Techniques for WCAG 2.0 document [W3C, 21] provides an extensive collection of approaches to manifest these requirements into some of the currently available Web technologies. It is expected that this knowledge base will be kept up to date as the best practices for implementing the requirements evolve (due to changes in the key Web technologies or in their support by user agents and assistive technologies). This will probably satisfy the needs for developing the majority of the Web sites effectively.

There are however specific situations in which the developers may need to develop alternative techniques for meeting the requirements. For instance, if they are using new technologies for which there are no documented techniques. A more common situation may however be a closed environment such as a corporate or otherwise private network. In some of these cases the network operators are aware of the computer hardware, operating systems, and installed software. In fact, in some cases they may develop customized browsers or media players for the network users. This is a completely different situation to developing a public Web site in which the operators can not guarantee the functionality of the clients, or rely on a specific user profile. In turn, some of the commonly accepted techniques may not be suitable for a specific environment. For instance, because the browser or assistive technology relied upon does not provide support for it (it may however be more widely supported by other user agents and is therefore an accepted technique). Conversely, some techniques that are not widely supported and thus not generally regarded as valid may be supported by the user agents relied upon in the closed environment, and may therefore be a valid solution for the particular situation.

It is however not a trivial task to decide whether a technique is acceptable or not in a specific situation. It is very dependant on what purpose the Web content has, what technologies have been used to realize it, and what level of support there is for the particular feature of the Web technology in the assumed user agents. To take a concrete example, an intranet login may be developed using an embedded plug-in for security purposes. The intranet operators assume that the users have a Web browser on their machine that is compatible with the plug-in. They need to be able to assess if the plug-in framework (the API) provides sufficient accessibility support so that the users with assistive technology can also use the login functionality. This is different than assessing the actual coding of the plug-in (which is also a necessary step). It is an assessment if such login functionality can be made accessible using the specific browser and assistive technologies. The W3C Web Accessibility Initiative (WAI) considers providing a repository with information about the accessibility support in different Web technologies to aid developers make decisions about the techniques that they can use in specific situations.

2.1.1.2. W3C Authoring Tool Accessibility Guidelines (ATAG) 2.0

The W3C Authoring Tool Accessibility Guidelines (ATAG) 2.0 [W3C, 22] complements the other WAI Guidelines introduced earlier in this section. It is currently a working draft but it is generally considered to be fairly mature. It follows the ATAG 1.0 and focuses on two primary objectives relating to the accessibility of authoring tools:

- **Production of accessible Web content** – the Web content produced by the authoring tools should support accessibility by adhering to requirements such as WCAG 2.0.
- **Providing an accessible user interface** – the interface of the authoring tools should itself be accessible to people with disabilities by adhering to relevant requirements.

While the first objective is pretty obvious, the second one is commonly forgotten. Many tool developers are not aware that contributing to the production of Web content is equally part of participating in the information society. Especially if we consider applications such as blogs, wikis, or other publicly available channels for user-generated content it becomes apparent that users, including people with disabilities, are sometimes authors. But also the traditional code editors, content management systems (CMS), or integrated development environments (IDE) are used by people with disabilities to create and publish Web content, and therefore also need to be accessible. ATAG 2.0 defines the term Authoring Tools as:

> "… any software, or collection of software components, that authors can use to create or modify Web content for use by other people."
> – http://www.w3.org/TR/ATAG20/#intro-def-au

This definition is intentionally very broad and potentially includes tools that were not initially designed for the production of Web content. For instance, if the production chain for the Web content includes using a word processor at some stage, then this word processor must also be accessible for people with disabilities. More importantly however, this definition includes the Web accessibility evaluation tools, generic quality assurance tools, as well as other types of tools that help improve or manage the accessibility of the Web content.

ATAG 2.0 considers the different types of editing views that are often provided by authoring tools, such as WYSIWYG or source code views. For Web-based authoring tools such as many CMS, the accessibility of the user interfaces is defined by conformance to WCAG 2.0 or other similar standards. However, authoring tools are commonly not Web-based but rather desktop applications. This includes but is not limited to code editors or IDE tools. For these situations ATAG 2.0 defines some of the functional requirements such as providing compatibility with assistive technologies or observing accessibility settings of the operating systems. The actual implementation of these requirements may often depend on the underlying operating system, accessibility API, and on software accessibility standards such as [ISO 9241-171].

While ATAG 2.0 does not define a specific development process or production chain, it has been specifically designed to be flexible and to fit into different process structures (hence the broad definition of the term). However, it explicitly relates so called 'accessibility checks' to the authoring process, which is where evaluation tools typically fit in. Specifically ATAG 2.0 Guideline B.2.2 says "Assist authors in checking for accessibility problems", and is essential for defining the interplay between development and evaluation tasks. It acknowledges that the checks sometimes need to be carried out manually by the developer rather than automatically by the software tool. However, it recognizes the importance of guiding the developers through any evaluation processes, and promotes the use of software tools to support the developers. It also recognizes the potential of metadata to describe the accessibility of the Web content in a machine readable form. This is an important aspect for later discussions in this study.

2.1.2. National Standards

As discussed in section 1.2 Motivation, accessibility of the Web for people with disabilities is an essential aspect of our modern information society. Many countries around the world have recognized this need, and have therefore developed policies or legislations that address the discrimination of people with disabilities on the Web [WAI, 13]. For instance, the Americans with Disabilities Act (ADA) or the Disability Discrimination Act (DDA) in the UK recognize access to information technology as a right of people with disabilities. In many cases these laws or policies apply to public sector as well as commercial Web sites equally. However, it is the strength and enforcement of these laws and policies that determines how effective they are.

In order to drive implementation and deployment of accessible solutions, some governments have additional and more stringent policies that relate to public Web sites. For instance, the US Section 508 law includes regulation of accessibility in public procurement that requires governmental institutions to acquire accessible Web sites. Since the US Federal Government has a substantial purchasing power, this mechanism has shown to be an effective vehicle for the promotion of accessible Web sites. Also individual European countries and the European Commission (EC) as a whole have or are in the process of adopting such public procurement approaches. Overall however, legal frameworks differ strongly from one country to another.

Regardless of the legal framework and jurisdictional implication, the technical definition for Web accessibility needs to be provided. In general there are three main approaches for this:

- **Adoption of existing standards** – some countries such as Austria, Australia, Ireland, Switzerland, or the UK have adopted the W3C Web Content Accessibility Guidelines (WCAG) 1.0 [W3C, 5] by referencing it as a technical standard for Web accessibility.
- **Development of formal standards** – some countries such as Japan, the Netherlands, or Spain have developed competing standards through their national standardization organizations. This was primarily done where the legal framework does not permit to reference standards developed by international standards consortia such as the W3C.
- **Development of legal provisions** – some countries such as Germany, Italy, France, or the United States have developed provisions that are directly part of the legislation or the policies, rather than to reference any existing international or national standards.

While most of these formal standards or legal provisions are derivatives of the WCAG 1.0, in many cases they include some modifications that sometimes lead to substantial fragmentation. For instance, in several cases requirements have been added, removed, or their meaning was changed so that the different standards have become incompatible. This fragmentation of the accessibility standards has a severely negative impact on the implementation and deployment of accessible solutions, including Web accessibility evaluation tools [WAI, 14].

2.1.3. Other Standards

Besides the formal international or national standards there are also less formal standards that were primarily developed by organizations that provide evaluation or certification services. In many cases these standards were initially developed as interpretation for WCAG 1.0 but have since been commonly perceived as separate standards, especially when they are coupled with certification programs. Examples include AnySurfer in Belgium, AccessiWeb in France, or the See It Right label in the UK. This progression also applies to criteria for awards such as the BIENE award in Germany which focuses exclusively on the accessibility of Web sites, or on corporate guidelines such as those by HP or IBM. It has become a requirement for Web accessibility evaluation tools to be able to address also these standards for Web accessibility.

2.2. Processes for Web Accessibility Evaluation

The process for Web accessibility evaluation is closely linked to the development process. At each stage of the Web development process, different types of evaluation tasks are carried out to assess the different aspects of the overall accessibility. In order to better understand when and how the evaluation tasks are carried out, it is important to separate out and to analyze the different stages of the Web development process. This section examines the process for Web accessibility evaluation including its parameters and dimensions. It also discusses how these relate to the overall development process, and how they can be supported by software tools.

There are currently no standards that define the Web development process. However, the ISO Software Life Cycle Processes [ISO 12207] defines the development process for software. It includes considerations for the different modes of acquiring, commissioning, or supplying the software, and outlines the different stages and activities that take place. It is a comprehensive standard, although it can be simplified to a circular process that consists of a 'Requirements Analysis', 'Design', 'Implementation', and 'Operation' stages. The assumption is that any software is created by first analyzing its requirements, then creating some initial designs, then implementing the logic and code, and finally released as an operational application. A new version of the same software starts with analyzing the new requirements, preparing updated designs, and so forth. This is generally referred to as a waterfall development model, although this concept is also imminent for other development models such as in agile programming.

While this life cycle refers to software development processes, it can be assumed that Web development processes follow a similar flow. There is however a considerable difference in that software is generally unchangeable after its release, with the exception of configuration or adjustment options. Web content is on the other hand commonly designed to change with time. Especially when observing Web sites in their entirety it becomes apparent that most of the development (such as publication of new content and modification of old content) takes place after they are launched. Also, the development of software is primarily focused on the underlying logic and programming. Web content is however much more dependent on the information that it provides, and on the user interface design aspects. These are the primary difference between software and Web development.

The following image illustrates these development stages that are further explained below:

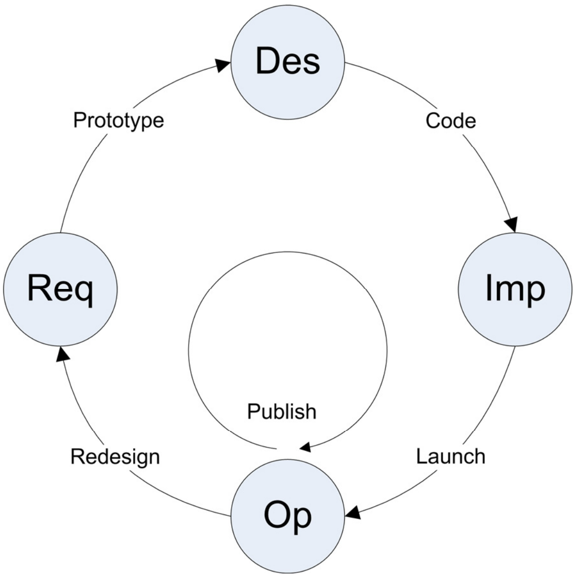


Figure 3: Illustration of the Web development life cycle

- **Requirements Analysis** – during the requirements analysis stage of development, the vision, purpose, and objectives are identified. Typically, common techniques such as sketches, storyboards, and personas are used to develop the requirements and to ensure that they address the needs of the end-users. The accessibility requirements are ideally also considered during this early stage of the development as it will save valuable time and effort in addressing these requirements later. For instance, a common accessibility requirement is to grant the user sufficient time to complete tasks, such as by alerting the user before a timeout is reached. These types of requirements tend to influence the overall behavior and characteristics of the Web content. Also the formulation and the presentation of text, images, and multimedia are subject to accessibility requirements.
- **Design Stage** – after the objectives and requirements are matured, first prototypes and mock-ups to validate the requirements are created. During this stage of development, formative and functional evaluations are carried out to asses the design and interaction concepts. This includes the document structure, visual layout, navigational features, color schemes, and other presentational aspects. Often the templates that later control the overall presentation and key interaction concepts (the ‘look and feel’) of the Web content are developed during this stage. These are critical assets for evaluation as it is usually much more difficult to change these underlying concepts during later stages of development. Also, templates and snippets are often used throughout the entire Web content and can quickly improve or degrade the level of accessibility.
- **Implementation Stage** – once the overall design has been matured, the realization of the actual Web content starts during this stage. This primarily involves developing the markup code that controls the content structure, as well as the server- and client-side scripts that control the functional behavior of the content. However, this development stage also involves the creation of the actual content such as the text, video, or sound that constitutes the Web resources. These are equally important aspects to evaluate but unfortunately they often tend to be neglected during many evaluation processes.
- **Operation Stage** – if the Web content has been initially developed with consideration for accessibility and meets certain standards for accessibility, then the evaluations that are carried out during the operation phase are primarily intended to maintain that level of accessibility or to identify additional optimizations that can be made to improve the level of accessibility. However, many Web sites continue to be developed with little or no consideration for accessibility so that broader-scoped evaluations are necessary to determine the overall level of accessibility, and to help plan for improvements.

2.2.1. Evaluation Parameters

The effort required to carry out Web accessibility evaluation varies strongly depending on several different parameters. These parameters depend mainly on the following variables:

- **Purpose of the evaluation** – the more ambitious, the more effort is required.
- **Realization of the content** – the more intelligible, the less effort is required.
- **Capacity of the evaluators** – the more advanced, the less effort is required.

Each evaluation may have significantly different dimensions due to changes of these variables according to the specific situation. For instance, if an organization wants to evaluate its own Web site for accessibility then the realization of the Web content can be assumed as constant. Depending on the purpose of the evaluation and on the capacity of the evaluators, the level of effort will vary from one organization to the other. If the scope of the purpose exceeds the capacity of the evaluators then the organization may consider acquiring external expertise to manage the required effort. It may also optimize the evaluation process and tools used to help improve the capacity of the evaluators and therefore reduced the overall effort required.

In a different situation, and organization may offer an accessibility evaluation service. In this case, the capacity of the evaluators can be assumed as constant since it is the capability of the service provider. The effort required for the evaluation will therefore depend on the purpose of the evaluation and on the realization of the content. Usually accessibility evaluation service providers will evaluate according to a specific methodology which governs the dimensions of the purpose so that this variable can also be regarded as constant (or at least discrete). In other words, the effort in this scenario maps directly to the realization of the Web content, possibly also to the type of service that was acquired. Such service providers therefore usually measure their effort based on the realization of the Web content, often by the number of Web pages.

Each of the variables described above can be further broken-down into individual parameters, each of which will be described in more detail in the following sub-sections:

- **Purpose of the evaluation**
    - **Scope** – more broad, more effort required
    - **Thoroughness** – more detailed, more effort required
- **Realization of the content**
    - **Complexity** – more complex, more effort required
    - **Consistency** – more homogenous, less effort required
- **Capacity of the evaluators**
    - **Expertise** – more skills, less effort required
    - **Tools** – more support, less effort required

To help picture how these parameters fit together in calculating the effort required for a Web accessibility evaluation, the following equation can be used as rough guidance for estimation:

$$Effort = \frac{Scope + Thoroughness + Complexity}{Consistency + Expertise + Tools}$$

Equation 1: Estimating the effort required for evaluation

2.2.1.1. Scope of the Evaluation

While the terms ‘scope’ or ‘coverage’ are sometimes used to refer to the overall purpose of the evaluation, in this context it is used to refer to the realm. For instance, an evaluation may be targeted to evaluate only a small component of the Web content such as a navigation menu or even just a single button. Such a narrow-focused evaluation can be useful in several cases:

- A developer is testing a prototype before it is deployed on the Web site.
- An evaluator is examining an existing component that will be re-designed.
- A customer is exploring why they are not able to use the component.

For similar reasons, an evaluation might also only target a specific portion of a Web site such as an application or sub-site. For instance, the customer online shop, the employee intranet, or the ‘press releases’ section of a Web site could be candidates of a targeted evaluation. In some cases organizations may also narrow the scope of the evaluation by the ‘depth’ of the Web pages. For instance, they may carry out an evaluation of all the pages that can be reached by a certain number of links from a starting point such as the homepage as these are regarded to be the more important ones. In other cases legacy content may be excluded from an evaluation.

In many of the cases however, the scope of the evaluation is the entire Web site. It is usually not economically feasible to evaluate the accessibility of all the features on all the Web pages provided on a Web site. It is usually also not necessary to evaluate all of the Web content in order to achieve the purpose of the evaluation. In many cases sampling can be used to reduce the scope of the evaluation and therefore also to reduce the effort needed for carrying out the evaluation. The tradeoff in selecting only a sample from the Web content is that accessibility problems can be potentially missed so that the evaluation result will be inaccurate. However, using carefully selected sampling strategies can achieve adequately results [Brajnik 2006].

2.2.1.2. Thoroughness of the Evaluation

The level of detail that is pursued by the evaluation is an important metric and probably has the most significant impact on the required effort for the majority of Web sites. An evaluation could be very coarse to get an initial sense for the level of accessibility on the Web site before deciding subsequent measure to take. While these types of evaluations do not find all of the accessibility barriers, they are fairly easy to carry out and could also be conducted by non-technical evaluators. It is therefore an effective indicator for the level of accessibility.

Sometimes an evaluation may also only focus on a single type of issue or a group of related ones. For instance, to evaluate the navigation and orientation features on a Web site, or to check how the layout works with screen magnification. Also these types of evaluations could be regarded as narrow since they do not identify all the accessibility issues but rather focus on only specific aspects. These approaches are however very useful to learn about the individual features and how they perform in practice. For instance, it could be used to collect input for a redesign project, or to confirm that the changes to the content were actually an improvement.

Conversely, an evaluation may be very detailed and intense. For instance it may include high end usability testing experiments in laboratories with specialized software. This can be useful to get the full picture of the accessibility performance of the Web site in order to plan for the improvement of any issues. In many of the cases however, Web accessibility evaluations are targeted to assess the conformance to the technical standards such as the W3C Web Content Accessibility Guidelines (WCAG) [WAI, 5]. Commonly these standards have different levels of conformance, such as the WCAG Priority Levels, which are used to determine the target and therefore also the thoroughness of the Web accessibility evaluations.

2.2.1.3. Complexity of the Web Content

The complexity of Web content is mainly determined by the amount of different types of Web technologies used in conjunction with how these technologies are used. For instance, a Web site that only used HTML and CSS is likely to be less demanding to evaluate than a similar Web site that uses scripting, multimedia, or other advanced technologies. The usage of these technologies is not necessarily a challenge for Web accessibility evaluation but it requires more work to evaluate the additional requirements that are applicable to such technologies. It may also be more demanding to evaluate specific types of Web technologies such as Flash as it is fairly complex (it is a programming language rather than a markup language). Evaluating multimedia is typically more time consuming as it requires the files to be examined in real time, for instance to compare the captions, transcripts, or other alternative formats provided.

Also the variety of features that are part of the Web content has an impact on the effort. For instance, a simple layout with few page elements is less demanding to evaluate than pages with navigation structures, forms, and tables. The more sophisticated page elements there are, the more time it will take for them to be analyzed and evaluated appropriately. Especially when the Web pages become larger more attention has to be paid to the usability aspect of the accessibility features. For instance, as Web pages become larger the in-page navigation and the orientation cues become increasingly important. Finally, also dynamically generated Web content may be more demanding to evaluate, especially if it generated by client-side scripting. Dynamically generated content requires some level of knowledge about how the script works in order to be able to evaluate it effectively. For instance to understand the transaction process of an application or to anticipate potential accessibility issues that may occur for certain users.

2.2.1.4. Consistency of the Web Content

Web content that is developed within confined boundaries tends to show steady performance, also with regard to accessibility. For instance, if Web pages are created consistently using a set of templates in a closed content management system environment then all the pages will tend to have a similar coding structure. If the templates contain accessibility problems then these will quickly propagate into all the Web pages that were created using these templates. Also, if the content management system generates certain code for table, forms, or other page elements then these will likely be repeated on all the Web pages that contain these elements.

The benefit of evaluating Web pages with a high level of consistency in their structure is that they become predictable. For instance, if it is clear how a script generates lists then evaluating one instance of a generated list will likely be a good representative for all the lists throughout the entire Web site. It will also be easier to anticipate potential issues, for example that the script may fail to produce accessible content for nested lists because it fails to nest the HTML elements appropriately. One could therefore search explicitly for such instances in order to confirm or dismiss the assumption. The more structured and logically the content is developed the less content needs to be evaluated in order to achieve acceptable evaluation results. There is therefore a direct correlation between the consistency and a sampling strategy that may be used to reduce the scope of the evaluation (see also section 2.2.1.1. Scope of the Evaluation).

Also content that is not generated by database scripts or content management systems could be more or less consistent. For instance, if there are guidelines for the Web developers and authors (often called ‘coding conventions’ or ‘style guides’) that are being implemented than it is more likely that there will be a higher level of similarity between the different Web pages than without such common rules. Unfortunately however many Web sites are developed over many years and using different technologies, development approaches, and have usually been developed by different people. Such Web sites require a high degree of effort to evaluate.

2.2.1.5. Expertise of the Evaluators

The skills and knowledge of the evaluators to carry out a specific evaluation task is a critical aspect of the efficiency of the evaluation. This does not mean that evaluators must be experts in the field of Web accessibility but they must posses the capabilities to carry out evaluation tasks that meet the purpose of the evaluation. For instance, non-technical authors who want to evaluate the publication text need to be able to evaluate the relevant requirements such as the reading level or headings structure. They could also evaluate color contrast combinations, depending if the system allows them to modify the text colors in an uncontrolled manner.

The expertise of the evaluators plays a significant role. In the field of usability which uses similar evaluation approaches to accessibility, expert evaluators are ~1.8 times more effective than novice evaluators. Furthermore, usability experts with expertise in the respective domain of application can be ~2.7 times more effective than novices [Nielsen 1993]. The experience is an aspect of the expertise, and contributes similarly significantly to the efficacy of the Web accessibility evaluations. Even a basic initial training, for example for authors and developers to learn about the principles of Web accessibility, could have relevant implications. It has been observed that such training for novice Web accessibility evaluators could make them up to ~31% more effective than their untrained novice evaluators [Chevalier and Ivory 2003].

2.2.1.6. Tool Support for the Evaluators

Web accessibility evaluation tools will be discussed in more details in later sections of this study. In the context of this section however, it is important to note that tools are an equally important parameters that significantly influences the effort required for an evaluation. While Web accessibility evaluation can not be carried out fully automatically, tools can assist the evaluators in many different ways. These range from scanning entire Web sites to help find potential accessibility barriers to helping evaluate for specific requirements. The tools used by the evaluators need to match the evaluation tasks that will be conducted [Brajnik 2004b]. For instance, some tools may provide better support for evaluating color contrast, table structures, or form controls in specific situations (such as Web technologies) than others.

2.2.2. Testing Approaches

In section 2.1 Technical Standards for Web Accessibility the accessibility requirements for people with disabilities were discussed. Each of these requirements can be evaluated through individual evaluation checks. These evaluation checks typically consist of series of individual atomic testing steps. In the W3C Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18] and its accompanying Techniques for WCAG 2.0 document [W3C, 21], this corresponds to the Success Criteria as the requirements and the Test Procedure sections of the Techniques as the evaluation checks. Consider evaluating the accessibility of an image in HTML:

	WCAG 2.0 Structure	**Practical Implication**
Requirement	Success Criteria 1.1.1	Images need equivalent alternatives
Evaluation Check	Technique H37	Equivalent alternatives need to be provided by the `alt` attribute of the `img` element
Testing Steps	Test Procedure of H37	1. Examine each `img` element in the content 2. Check that each `img` element which conveys meaning contains an `alt` attribute 3. If the image contains words that are important to understanding the content, the words are included in the text alternative

Table 1: Example of the practical implication of the WCAG 2.0 structure on the evaluation approach

Each of the individual testing steps for evaluation may be carried out in different modes. In general, there are three basic types of testing modes:

- **Automated Testing** – carried out by software without human intervention
- **Manual Testing** – carried out by evaluators, possibly using some software
- **User Testing** – carried out by end-users following usability testing methods

These testing approaches are however not mutually exclusive. In fact, each testing approach provides better performance for testing specific types of accessibility barriers. Each testing approach can also be used to go beyond what the accessibility standards address, which is a subset of all possible accessibility barriers. Ideally the different testing approaches are used together to maximize the effectiveness and efficiency of an evaluation.

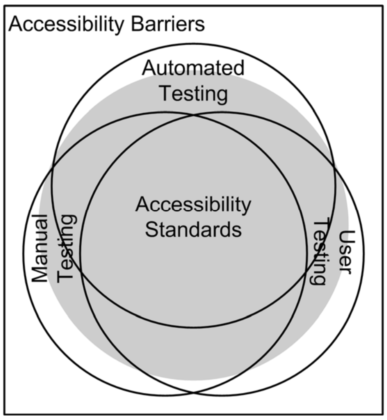


Figure 4: Testing approaches

2.2.2.1. Automated Testing

Automated testing is carried out without the need for human intervention. It is cost effective and can be efficiently executed periodically over large amounts of Web pages. At the same time, automated testing only addresses a subset of the accessibility requirements set out by Web accessibility standards. This is inherent to the nature of the requirements as they tend to be qualitative rather than measurable. For instance, they address user interface, interaction, as well as natural language aspects. For example, a common requirement for Web accessibility is to ensure that the document markup reflects the semantic structure that is conveyed through its visual presentation, yet it is difficult to develop algorithms that analyze such semantics.

Another difficulty of automated testing is simply computational limitations. For example, an accessibility requirement is to ensure sufficient color contrast between the foreground and the background of the Web content. For text in HTML and other document formats, this can be calculated using the Red-Green-Blue (RGB) values. However, for bitmapped content such as images it is generally difficult to differentiate between foreground and background pixels automatically. Also using Optical Character Recognition (OCR) approaches to help evaluate images of text are resource intensive and not sufficiently reliable in many cases. In general, one could differentiate between the following types of automated testing:

- **Syntactic Checks** – analyze the syntactic structure of the Web content such as check for the existence of `alt` attributes in `img` elements, or `lang` attributes in the HTML root elements. While these types of syntax checks are reliable and often quite simple to realize, they only address the minor subset of the accessibility requirements.
- **Heuristic Checks** – fuzzy algorithms that examine some of the semantics in the Web content such as the layout and markup, or the natural language of information. While these types of checks cover a broader range of requirements than syntactic checks, they are considered less reliable and usually only serve as warnings for evaluators.
- **Indicative Checks** – use statistical metrics and profiling techniques to estimate performance of whole Web sites or large collections of Web content. While these checks are too imprecise for detailed assessments of the Web content, they are useful for large-scale surveys. For example as indicators to monitor the overall progress.

2.2.2.2. Manual Testing

In practice, the majority of the tests need to be carried out by human evaluators, even if they are sometimes guided or supported by software tools. For instance, while software tools can quickly determine the existence `alt` attributes of `img` elements, human evaluators need to judge the adequacy of the text in these attributes. In many cases automated heuristic checks can provide additional assistance for the evaluators, for example by triggering warnings if the `alt` attribute contains typical default texts such as ‘image’, or ‘spacer’, and so on. However, the primary responsibility for making the final decisions is held by the human evaluators.

Because manual tests cover such a broad range of accessibility requirements and have varying degrees of software tool support, they have varying requirements with regard to the skills and knowledge of the evaluators. Some tests can be carried out by non-technical evaluators while others may need more technical knowledge. Some tests can be carried out by evaluators who only know basic principles of Web accessibility while others may require significantly more domain knowledge. While the required skills are mainly determined by the nature of the tests to be carried out, the software tool support provided to the evaluator can also be an important factor. In general, one can differentiate between the following types of manual testing:

- **Non-Technical Checks** – can be carried out by non-technical evaluators such as Web content authors. For instance to determine if the `alt` attributes describes the purpose of the images appropriately, or if the captioning (or transcriptions) for the multimedia provide adequately equivalent alternatives.
- **Technical Checks** – are usually carried out by Web developers who have technical expertise with Web technologies. However, they may often only need to understand the basics of Web accessibility to check for potential barriers. Such checks typically address markup code and document structure as well as compatibility with assistive technology and other programming aspects.
- **Expert Checks** – are carried out by evaluators who have knowledge of how people with disabilities use the Web, and who can identify issues that relate to the user interaction. This is comparable to 'walkthroughs' and 'heuristic evaluations' in the field of usability engineering, as the experts anticipate the issues that end-users may encounter in the Web content.

2.2.2.3. User Testing

User testing is carried out by real end-users rather than by human evaluators or by software tools. It is a broad field of study and closely relates to usability. In fact, it is often referred to as 'Usability Testing with People with Disabilities' [Henry 2002]. User testing is an important testing approach that complements the other ones highlighted so far. It focuses on the end-users and how well the technical solutions match their requirements in a specific context. For instance, it is generally good practice to provide orientation cues and landmarks to help users to navigate through the Web content. At the same time, too many cues can be irritating or even become a barrier in itself. While it is the goal of accessibility standards to capture such conflicts and define provisions to avoid them, studies show that even experienced usability evaluators find only about 35% of the usability problems on average [Nielsen 1993]. It is essential to involve people with disabilities to improve the efficacy of evaluation processes.

Probably the biggest caveat with user testing is the difficulty to filter out personal bias and preferences, and identify the actual issues. For instance, that a user was not able to complete a task does not automatically mean that there is a valid accessibility barrier in the Web content but it could equally be an issue with the browser, or the assistive technology, or even that the user is not able to use these tools effectively; for example if the user is a novice computer or assistive technology user. Conversely, that users were able to successfully complete the tasks does not automatically imply accessibility since the completion of the task often relates to the experience of the user (for example to find workarounds) and likely also to their specific type of disability. While these issues are usually related, it is important to separate them in order to identify the underlying causes and address them accordingly. Although the methods for user testing cover a broad spectrum, in general there are two main directions:

- **Informal Checks** – do not strictly follow the usability procedures, for example by asking individuals such as friends or colleagues for their opinions. While these types of quick checks can be effective and useful, they are coarse and thus prone to personal characteristics, bias, and preferences. Some informal checks can also be simple do that they can be carried out by non-experts, if they understand the cautions and caveats.
- **Formal Checks** – are usually carried out by professionals who follow well established usability procedures. It is important that the evaluators can identify an appropriate user population and establish appropriate tasks. They also need to understand how people with disabilities use the Web and how to interpret the observations.

2.2.3. Roles and Responsibilities

Regardless of the experience and skills provided by evaluators (as described in section 2.2.1.5 Expertise of the Evaluators) there are different roles and responsibilities throughout the entire Web accessibility evaluation process. Depending on the structure of the development process, an individual can take one or more of these roles. In fact, typically only larger enterprises tend to have distinct employees or groups that assume specific roles. For instance, in many projects the application developers also author or publish the actual content as part of their tasks. On larger Web sites there may typically be separate groups of application developers and content publishers. In fact, enterprises may sometimes even have sub-groups such as the testers who share the application development responsibility with the programmers.

Essential to evaluation is however the role of the accessibility expert. It is a central entity that provides guidance and support to the individuals in the Web accessibility evaluation process, and is therefore a crucial advisor [Thatcher et al 2006]. The following image illustrates this central role of the accessibility experts that actively provides support to the remaining actors (each of the roles illustrated in this image is described in the following sections):

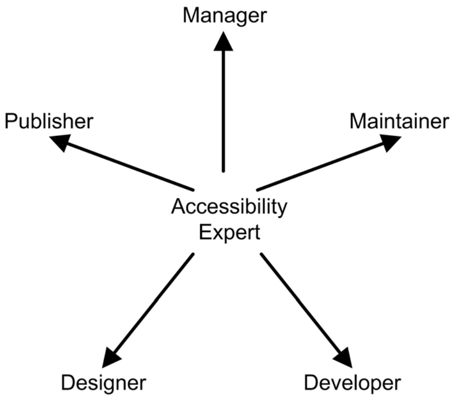


Figure 5: Roles in Web accessibility evaluation

The roles and responsibilities are not necessarily assumed by a single entity such as an agency or an organization that owns the Web content. They are commonly outsourced depending on the structure of the development process. For instance, often the visual design may be done with the help of an external agency or expert. In some cases the entire development process is contracted to a Web agency. For instance, if the Web site owner does not have in-house Web developers. This practice also applies to expertise in accessibility – especially the accessibility expert could be an external entity that provides consulting or support services, and that works with the contracting organization throughout the entire development process.

2.2.3.1. Aesthetics Designer

The aesthetics designers are responsible for the visual design and overall ‘look and feel’ of the Web content. This includes layout, branding, corporate design, as well as the navigation. Also multimedia or scripts that enhance the user experience are being increasingly used. For instance to provide background sounds, interactive dialogs, or special effects such as fly-out menus and similar. The relevant accessibility issues usually relate to orientation, navigation, use of colors, and other functional aspects. In several cases they could also relate to technical aspects of the implementation, such as the CSS presentation or the HTML document structure.

2.2.3.2. Application Developer

Application developers are commonly technical such as programmers, testers, and application architects. They are responsible for developing the underlying scripts, applications, as well as markup code that is often dynamically generated by the scripts and programs. The application developers are often also involved in the acquisition or development of content management systems, and are primarily responsible for installing and configuring these. It is essential that the application developers are aware of the accessibility requirements that apply to CMS and other development tools as these are likely to be used by others such as the content publishers.

2.2.3.3. Content Publisher

Content publishers are typically non-technical users who are responsible for the authoring and publishing of information through the Web. For instance, journalists writing articles for news Web sites, public relations personnel updating the information about their organizations, or teaching staff publishing educational resources are common situations. Also User Generated Content such as blogs, wikis, and other applications are becoming an increasingly popular mean of interacting with the public. For instance, employees could use such communication channels to provide job-related information, just as private individuals for social interaction.

2.2.3.4. Content Maintainer

Content maintainers are responsible for monitoring and maintaining the proper functionality of the Web content. This includes ensuring that the Web content meets the desired level of quality such as conformance with Web standards. In many organizations the responsibility for maintaining the content is delegated to the Webmasters. Depending on the nature of the Web site, new Web content may be generated continually or it may be modified rapidly by content publishers and editors. Content maintainers are often also confronted with legacy content, and may sometimes be responsible for migrating it to new applications or to new Web standards.

2.2.3.5. Project Manager

Project managers are instrumental in delegating responsibilities to individual actors, and in assigning the respective tasks. They oversee the Web development process and are primarily responsible for integrating the accessibility requirements throughout the entire process. They need to be able to monitor the overall situation and progress on accessibility in order to ensure that the Web content meets the requirements at all times. Project managers also need to assess the feasibility of meeting certain targets and what impact such decisions may have on the rest of the project. For instance, there are often key applications that are easy to retrofit or that are crucial to the functionality of the Web sites; these should be prioritized in the project plans.

2.2.3.6. Accessibility Expert

As discussed in the introduction, accessibility experts are essential for any Web development process that considers accessibility. The expert provides guidance to the remaining actors in the development process. In particular, the expert should be in the position to inform and advise the project manager on optimizing the development process to better address the needs of people with disabilities. Accessibility experts are not necessarily technical though they need to have founded knowledge about the technical standards and how to implement these in practice, so that they can train developers or help identify the cause of accessibility barriers.

2.2.4. Evaluation Methodologies

The term 'evaluation methodology' typically refers to a documented quality control process that is carried out on Web sites after they have been developed. Evaluation methodologies are typically designed for the following purposes [WAI, 17-21]:

- **Preliminary evaluation** – coarse assessment to explore some of the potential barriers.
- **Conformance evaluation** – rigorous assessment of conformity to technical standards.
- **Comprehensive evaluation** – assessment of the user experience beyond the standards.

The objective of evaluation methodologies is therefore to provide processes that are:

- **Precise** – identify all accessibility barriers without false negatives or false positives.
- **Repeatable** – produces repeatable outcomes regardless of the evaluators involved.
- **Scalable** – can be used for larger or smaller Web sites, without losing effectiveness.

Evaluation methodologies therefore describe the specific procedures and approaches that are used in an organization or for specific types of Web sites. For instance, they often define:

- **User needs** – documented user requirements, such as WCAG 2.0 [W3C, 18], against which the Web content is to be tested for accessibility.
- **Testing procedures** – documented steps for carrying out the testing, such as these provided by the Techniques for WCAG 2.0 [W3C, 21].
- **Sampling procedures** – documented steps for selecting the representative samples of Web pages from the Web site to be tested (see section 2.2.4.1. Sampling Strategies).
- **Tools** – documented list of browsers, media players, assistive technologies, evaluation tools, and operating systems that are to be used for testing.

Often evaluation methodologies may also define other aspects of the processes. For instance, they commonly define the reporting format or methods for aggregating the results into some form of an overall score or indicator. There are not many publicly documented methodologies since many are regarded as corporate or organization-internal intellectual property. However, most of the well known evaluation methodologies are designed for the use-case of evaluating the conformance of Web sites against a set of standards such as WCAG. For instance, the Unified Web Evaluation Methodology (UWEM) [UWEM] is such a methodology that was developed through an EC-funded project involving 24 project partners from Europe.

There is a close relationship between the underlying user needs and how well these have been formulated and the evaluation methodologies. For instance, in WCAG 1.0 some requirements were open to interpretation and different testing procedures emerged as a consequence. This has an immediate impact on the precision and repeatability of evaluation methodologies built around WCAG 1.0. Because WCAG 2.0 also defines the exact testing procedures it is less prone to these issues, so that more effective evaluation methodologies can be designed for it.

As mentioned earlier, evaluation methodologies are typically used to determine the current state of existing Web sites. For instance, Web site owners may commonly commission such an evaluation as an initial step to learn about the accessibility barriers and to repair them in later versions of the Web site. These types of evaluations can also be carried out periodically by the content maintainer or accessibility experts of an organization (see section 2.2.3. Roles and Responsibilities) to monitor the level of accessibility on a Web site.

2.2.4.1. Sampling Strategies

As discussed in section 2.2.1. Evaluation Parameters, there are inter-dependent relationships between different evaluation parameters. Since the evaluation methodologies are typically designed to evaluate larger collections of Web pages such as entire Web sites, the intended scope can quickly become very significant. It is however usually not economically feasible to evaluate all the Web pages with the same thoroughness to identify the issues on the Web site. Instead, sampling strategies can be used to take advantage of the consistency and similarity between the Web pages on a Web site. It is often sufficient to evaluate an adequately diverse selection of Web pages to identify the majority of the types of barriers that occur in the Web content. The optimal sample size depends primarily on the diversity of features, technologies, and development practices used to create the Web pages, but also on how representative the selected pages are. For instance, consider the following situations:

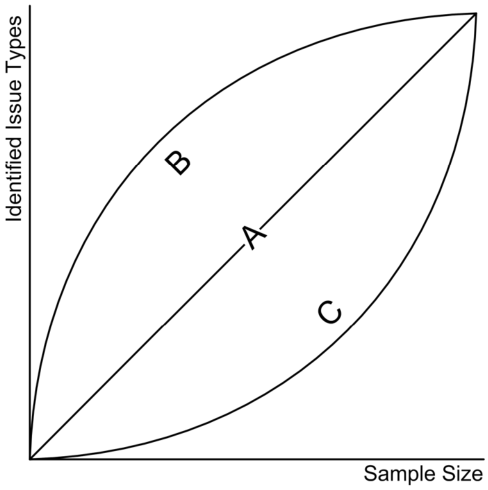


Figure 6: Sample size versus identified issue types

Figure 6 illustrates the relationship between the sample size and the identified types of issues. In the first situation, 'A', the increase of the sampling size is directly proportional to the types of issues identified. This would mean that all the Web pages are unique in terms of the profile for accessibility, and that each page provides new types of issues. This is however an unlikely scenario, since Web sites usually demonstrate clustering behavior with respect to accessibility characteristics of Web pages [Vigo et al. 2007]. The second situation, 'B', makes use of this behavior. It creates profiles of the pages and selects representatives from each to identify the types of issues more quickly. The third situation, 'C', illustrates inefficient sampling methods that only gradually identify the different types of issues. This occurs when strictly sequential approaches are used, such as selecting Web pages from the same logical group. For instance, selecting all the Web pages that contain forms, or that belong to a department, or that are linked from the home page are strategies that confine the evaluation findings to similar types of accessibility barriers. Using appropriate sampling strategies can significantly reduce the amount of effort needed to identify the existing barriers [Brajnik et al. 2007].

There are currently only few Web accessibility evaluation tools that consider these factors to help evaluators identify representative samples of Web pages from a given collection such as an entire Web site. At the same time, several studies have demonstrated that Web pages can be profiled according to their accessibility characteristics based on automatically measurable metrics. For instance, it is likely that Web pages with similar structure and elements will demonstrate similar accessibility characteristics. Such sampling approaches can significantly improve the precision, repeatability, and scalability of evaluation methodologies.

2.2.4.2. Informed Methodologies

When Web accessibility evaluation methodologies are executed in complete isolation from the development processes, then the evaluators have no knowledge about the internals of the Web site and how the code is generated. This resembles the black-box testing approaches in traditional software quality assurance. Most publicly documented evaluation methodologies seem to be designed with this assumption. However, in many cases it is possible to provide the evaluators with background information to accelerate the evaluation. For instance:

- What were the outcomes from previous evaluation runs?
- Which pages have been added, removed, or changed since?
- How are the Web pages created, and what scripts are used?

Some Web accessibility evaluation tools can actually store the history of previous evaluation runs or keep track of the changes made to the Web pages. This can significantly improve the scalability and efficiency of such broad evaluations. However, it seems that more research is required to help identify the relationships between the internals of the Web sites (‘backend’) and the accessibility characteristics observed on the user interface (‘frontend’). This would resemble white-box testing approaches in traditional software quality assurance.

Another approach to support informed methodologies is by maintaining information about the accessibility of the Web content as it is being developed. For instance, if Web developers add information about the accessibility testing they have done to the Web content, then evaluators could use this information during conformance evaluations. This concept is generally referred to as ‘incremental evaluation’, in which each actor in the production chain of the Web content provides additional segments of information about the accessibility of the Web content being developed. An example scenario could be evaluation information provided incrementally by:

- **Designers:** templates, code snippets, and ‘look and feel’ aspects.
- **Developers:** scripts, markup, and other programmatic components.
- **Publisher:** text, images, multimedia, and other informative content.
- **Maintainer:** composition of all the above in the actual Web pages.

This means that the level of accessibility is not only measured on discrete intervals when the conformance evaluations are executed, instead there is continually reliable information about level of accessibility available. Informed methodologies establish a relationship between the evaluation carried out during development and the evaluation carried out after the publication. Section 2.1.1.2. W3C Authoring Tool Accessibility Guidelines (ATAG) 2.0 describes the role of authoring tools for supporting evaluation during the development process. In particular, the ATAG 2.0 Success Criterion B.2.2.8 “Metadata for Discovery” provides an opportunity for tools to support the concept of incremental evaluation. The requirement says:

> “If the authoring tool records accessibility status, then authors have the option to associate this status with the content as metadata to facilitate resource discovery by end users” – http://www.w3.org/TR/ATAG20/#checking-scAA

While this may not be a very strong requirement, it is an important starting point to informing post-development evaluation methodologies, and to help manage the overall accessibility of a Web site. However, hardly any Web accessibility evaluation tools provide support for these types of quality management approaches. Later sections of this study will look at potentially viable methods to promote frameworks that fit into this Web development concept.

3. Analysis of Current Web Accessibility Evaluation Tools

W3C defines Web accessibility evaluation tools as “software programs or online services that help determine if a Web site is accessible” [WAI, 22]. This is a very broad definition and also includes software that may not have been explicitly designed for the purpose of evaluating the accessibility of Web content. For instance, in many cases standard functions in Web browsers or assistive technologies can be used to evaluate how the Web content renders using different settings. In fact, the W3C Preliminary Evaluation approach [WAI, 17] uses such techniques to evaluate the accessibility of Web content. However, in a narrower sense the term ‘evaluation tools’ refers to software that provides specific functionality to help evaluate the accessibility of Web content. This includes stand-alone or plug-in tools that can carry out automatic, semi-automatic, or manual tests for accessibility. The W3C List of Web Accessibility Evaluation Tools [WAI, 23] currently contains over 120 such software tools.

While Web accessibility evaluation tools can significantly reduce the time and effort required to carry out evaluations, no single software tool can automatically determine the accessibility of Web content without human judgment. As described in section 2.2.2. Testing Approaches, only a minority of the accessibility requirements can be tested automatically. Moreover, only the syntactic checks of the automatic tests can be considered as reliable, while the other types of automated tests need to be verified by human evaluators. Evaluation tools can be compared to dictionaries in word processors as they generally only identify potential issues, rather than reliably determine the level of accessibility [UI Access, 1]. Despite the limitations, evaluation tools can assist evaluators in many different ways, and are therefore essential assets.

There are many types of Web accessibility evaluation tools that provide different functionality. Some focus on evaluating only one or two specific requirements such as the color contrast or the table markup, while others aim to address the broader scope of requirements. Also some pages evaluate only single pages while other can crawl across large collections of Web pages such as entire Web sites. In general however evaluation tools tend to provide these following modes to interact with the evaluators, some tools provide more than one of these [WAI, 22]:

- **Reports** – listings that typically contain line numbers, error messages, or an indication whether certain requirements were or were not met. Sometimes it may contain lists of tests that were not carried out but still need to be executed based on the Web content.
- **User Dialogs** – guide the users through accessibility checks step-by-step, or otherwise prompt the users for input. For instance, a tool may prompt the evaluators to select the foreground and background colors, or may query if a table is used for layout purposes.
- **In-Page Feedback** – insert temporary icons and markup to display evaluation results or to highlight areas of interest directly within the Web content. For instance, to show the location of errors, outline forms and tables, or highlight the text equivalents.
- **Transformations** – modify the display of the pages to help highlight potential issues. This includes displaying the content with altered color schemes, without style sheets, or as it would be presented by assistive technologies such as screen readers.

Besides the coverage of accessibility requirements, the coverage of multiple Web pages, and the user interface characteristics above, there are other aspects that differentiate between the Web accessibility evaluation tools. Some of these include:

- **Application Type** – while some Web accessibility evaluation tools are plug-ins for authoring tools or Web browsers, they are more commonly stand-alone desktop applications or remote online services and do not integrate well into other tools.

- **Technology Support** – the majority of the Web accessibility evaluation tools focus on evaluating HTML and often also CSS content. However, only few focus on evaluating other formats such as SVG, SMIL, PDF, or other formats and Web technologies.
- **Platform Support** – many of the Web accessibility evaluation tools are designed for the Windows operating system and only few support other systems. This also applies to evaluation tools that are installed as remote services such as on an intranet.
- **Reliability** – there is a strong discrepancy between the outcomes of different types of Web accessibility evaluation tools, so that it can be assumed that some tools are more accurate than others with respect to false negative and false positive results.

All these parameters contribute to vast differences between Web accessibility evaluation tools. Different evaluation tools seem to demonstrate varying benefits for specific contexts, such as the Web content to be evaluated or the background of the evaluators. For instance, advanced evaluators may find wizard-based tools as verbose and annoying while novice evaluators may find them informative and helpful. Also, an evaluation tool that only focuses on color contrast may be exactly what a Web designer needs during the early design and color selection phase, especially if it can be plugged into the software that the designer uses to create the designs (in many cases Photoshop, Gimp, or similar products). On the other hand, such an evaluation tool may be too focused for a content maintainer who is primarily interested in other requirements.

Often a selection of different Web accessibility evaluation tools may provide optimal support throughout the development life cycle. Commonly one of these is a primary tool and others are used to provide additional support for specific situations. For instance, often an enterprise tool may be used as the central repository in a university, corporate, or other organization. In addition to this, the content maintainer may be using an evaluation tool that plugs directly into the Web browser, and provides in-page feedback for page-by-page evaluations of a selected sample of Web pages. In a smaller or medium-sized organization such as a Web agency, the central evaluation tool may be one that can execute automated tests rather than the enterprise solutions. However, since individual developers tend to assume multiple roles in these types of settings, the continuous switching between different types of tools can become problematic. In these cases the integration functionality of the evaluation tools has increased importance.

3.1. Current Approach for Web Accessibility Evaluation Tools

Throughout the previous sections it was identified that Web accessibility evaluation is ideally an intrinsic part of the Web development process from the early requirements analysis phases to the maintenance phases. However, in general evaluation tools are not specifically designed to be part of this development process. Evaluation is often designed to be a separate process that is often carried out after the actual development of the Web content.

This section explores the current approach that many Web accessibility evaluation tools seem to employ. It is however important to note that these evaluation tools have been developed to address the demand by developers. As long as the developers continue to address accessibility during later stages in the development process rather than during earlier stages, tool vendors are not likely to change their approach. At the same time, if the evaluation tool vendors do not change their approach then the developers will be less encouraged to evaluate earlier on in the development process. It is a cycle that needs to be broken by raising awareness for the issue.

3.1.1. Post-Development Evaluation Paradigm

Many of the currently available Web accessibility evaluation tools introduced at the beginning of this section are designed for post-development evaluation. In other words, the evaluation is carried out on already existing Web content. This is generally known as summative evaluation, and is a useful approach for identifying and documenting the status of the given Web content. For instance, it is an effective method for developers who have not implemented accessibility consistently and want to determine the current level of accessibility. After an initial evaluation, post-development evaluation is ideally carried out periodically in order to monitor and ensure the intended level of accessibility for a Web site.

Unfortunately it is a common situation that Web content is not developed with awareness for accessibility requirements from the start, and therefore needs to be evaluated thoroughly after it has been already implemented. Web accessibility evaluation tools – especially tools that can crawl across Web pages and carry out automated testing – are essential to help evaluate large volumes of Web content effectively. This is possibly a primary use-case for developing many of the currently available Web accessibility evaluation tools. Especially the commercial tools seem to focus strongly on crawling and automation features, even though many of them seem to have started to shift their focus towards monitoring and managing the level of accessibility.

It is however not always economically feasible to retrofit Web content for accessibility after it has been implemented. For instance, it is often too expensive to rewrite database scripts after they have been deployed, even though the changes that need to be made could be minute. The overhead for re-opening the development process, making the changes, and then carrying out regression testing to ensure that the change did not break anything else is often too expensive. Often the changes made, if any, are therefore only compromise and work-around solutions. On the other hand, incorporating accessibility requirements from the start is often a negligible effort, especially for technical requirements such as improving the output of database scripts.

While most of the Web accessibility evaluation tools can also be used during the development of Web content, only some are specifically designed to help evaluate Web content during its development. For instance, some of the Web accessibility evaluation tools are designed to be extensions or plug-ins for authoring tools such as editors and content management systems. Some evaluation tools extend the functionality of authoring tools by providing direct feedback on the Web content as it is being developed. For instance, they highlight code segments that contain or that may contain accessibility barriers. Other evaluation tools can be launched from the menu options of the authoring tools, and provide their feedback directly in the workspace.

It is essential to continue promoting the availability of Web accessibility evaluation tools that can help evaluate Web content during its development. These types of tools can considerably reduce the overall development costs on the long run since they help ensure the production of accessible Web content. This also applies to tools that can help design accessible Web content, even before any development. For instance, the selection of aesthetic and accessible fonts and color combinations is often only a matter of awareness. It is far easier to address these design issues before any development rather than after the content or applications are created. There are hardly tools that have been specifically designed to help design accessible Web content.

On the other hand, there are software quality assurance concepts and tools for modeling user interfaces. These early models, such as wireframes or storyboards, are used to help assess the overall requirements after which the user interfaces are created. While many of these concepts for software quality assurance have found their way into Web development, unfortunately it seems that they have not yet found their way into many Web accessibility evaluation tools.

3.1.2. Developer-Oriented Application Design

The user interface design of Web accessibility evaluation tools has been subject of discussion in various literature and relevant forums. While Web accessibility evaluation tools provide a broad selection of user interface designs, it seems that they primarily target a similar audience of more experienced and technical developers. A considerable number of Web accessibility evaluation tools is developed using a report-based approach. This is especially applicable to automated Web accessibility evaluation tools. They are designed to launch accessibility tests on either a single Web page or on an entire collection of Web pages, and report their findings.

The reports generated by Web accessibility evaluation tools differ in many ways. For instance, some of the text-based report formats provide the following options [WebAIM, 1]:

- Errors listed by the location of the errors.
- Errors listed by the type of error or priority.
- Errors listed according to the source code.
- Errors listed according to the visual display.

Some Web accessibility evaluation tools also provide features to summarize the reports and to generate different views on the data. For instance, barometer-type displays that only indicate the current level of accessibility are very popular among executives and project managers. In some cases it is possible to generate graphs and charts, for example to visualize the progress.

Despite the views and representations for the evaluation findings, reporting-based approaches are in general more apt for experienced Web developers. Often the evaluation reports will cite the requirements set out by the technical standards. For evaluators who are new to the field of Web accessibility, these types of error messages may often be too difficult to understand and may therefore not provide sufficient guidance. In fact, in some cases the usage of evaluation tools can reduce rather than improve the performance of novice evaluators [Ivory 2003a].

The stronger focus on more experienced Web developers seems to be also present in the case of manual evaluation tools. For instance, in order to effectively use toolbar-based evaluation tools, developers must know how the different functions of the application can be used. They must know the requirements and how to carry out the test procedure in order to be able to use these types of tools effectively. While it is certainly desirable that developers understand the technical requirements, it is not always the case. Especially non-technical developers such as content authors and publishers are often not included in Web accessibility training programs.

In response to this, some Web accessibility evaluation tools have employed wizard-based user interfaces. While these types of evaluation tools are generally considered as verbose by more experienced Web developers, they provide better assistance for novice developers. Still, many of the wizard-based tools seem to be targeting novice technical developers rather than content authors or designers for example. For instance, only a handful of Web accessibility evaluation tools are specifically designed to help visual designers in selecting color combinations that provide sufficient contrast, without requiring them to know about the technical requirements.

While many argue that it is not the responsibility of evaluation tools to educate the developers in the topic of accessibility, tools could substantially contribute to the implementation of Web accessibility. By creating tools that target more diverse audiences than experienced technical Web developers, tools can serve considerably larger audiences. They can also lower the entry level for learning about Web accessibility and therefore promote the implementation.

3.1.3. Disjoined and Monolithic Architectures

In general, Web accessibility evaluation tools tend to be disjoined from the normal process of Web development. Many tools are designed as desktop applications that need to be installed and run separately. When for instance content authors are publishing articles using many of the currently available content management systems, they must use additional evaluation tools outside the content management system in order to evaluate the content. They must repeat the following steps as long as the evaluation tool identifies accessibility barriers:

1. create a preview of the article
2. launch an evaluation tool
3. evaluate the article preview
4. retrofit any found barriers

Potentially they may need to carry out these steps with other evaluations tools, for example to evaluate other aspects of the Web content that one tool could not evaluate adequately. With all the other priorities that the Web developers have, it is likely that some of these evaluation steps may be skipped or may not be carried out thoroughly. As a result, inaccessible content will be published or Web developers will spend additional effort to compensate for the tools. It is essential that Web accessibility evaluation tools are integrated into normal processes of design, development and maintenance for them to be effective [Brajnik 2004b].

Enterprise Web accessibility evaluation tools aim to improve the coverage of the tests so that developers do not need additional tools for evaluation. They also tend to be customizable so that they can be tailored to the specific needs of an organization. For instance, different Web developers can have their own accounts to manage the accessibility of the Web content that they are responsible for, such as Web pages that they published or that they are maintaining. Webmaster may have a different view in their account, for instance to get an impression of the overall situation across the Web site or to generate specific types of reports. Some tools offer complete workflow support, similar to these presented by content management systems.

While these types of evaluation tools aim to address the different developer roles and support their specific needs for Web accessibility evaluation, many of these tools follow a monolithic approach where all the functionality is centralized in a single tool or sometimes system. It still forces Web developer to go out of their normal development environment such as the content management system or the integrated development environment (IDE) and to carry out their evaluations using a different tool. This continuous switching between different types of tools is a significant constraint to the efficacy of the Web developers [Englefield et al. 2005].

3.1.4. Centralized Responsibility and Expertise

As introduced in section 2.2.3 Roles and Responsibilities, the responsibility for evaluating the accessibility of the Web content needs to be distributed among the different developers that are involved in the production. However, the role of the 'content maintainer' is commonly delegated to the Webmasters. The Webmasters are tasked with regularly evaluating all the Web content being developed or modified, and to contact the responsible developers with issues. In turn, Webmasters often have to respond to the queries of the developers in order to resolve issues. Webmasters therefore often also assumes the role of the 'accessibility expert'.

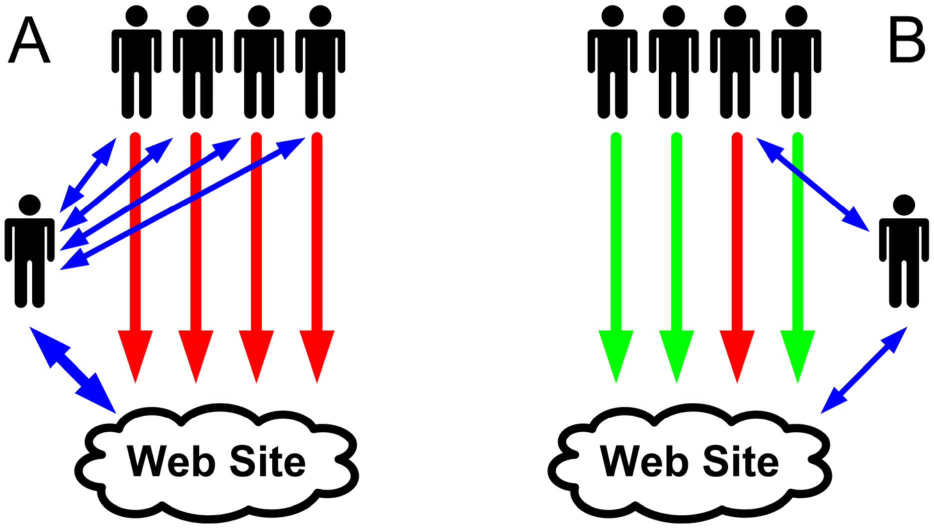


Figure 7: Illustration of the Webmaster bottleneck

This phenomenon is known as the 'Webmaster bottleneck' and is illustrated above. In the first situation 'A', the developers are publishing content that has not been evaluated and therefore potentially contains accessibility barriers. The Webmaster has to evaluate all the content in order to maintain the level of accessibility. The Webmaster has to establish a dialog with each of the developers in order to resolve the issues and respond to additional queries that they may have. These are all tasks in addition to the other responsibilities that a Webmaster usually has.

This model is not scalable and quickly results in an overload of queries to the Webmaster and a low response time in return. Usually this model can not be maintained over longer periods, especially if there are several Web content authors. However, in the second situation 'B', the developers are publishing content that is evaluated by them. It may occasionally still contain some uncaught accessibility barriers but it is generally reliable and good quality content. The Webmaster can focus on verifying the accessibility of the Web content and retrofit occasional uncaught accessibility barriers. It is a scalable and economically feasible model for evaluation.

While the Webmaster bottleneck is primarily a managerial issue, Web accessibility evaluation tools could play a significant part in diminishing it. When evaluation tools help developers in evaluating the content throughout the different development stages, when they do not require technical expertise for non-technical requirements, and when they can better integrate into the normal development process of the developers, then they can help distribute the responsibility and expertise within the organization. Moreover, such a distribution of expertise could further contribute to the development of more specialized evaluation tools. However, currently many Web accessibility evaluation tools seem to be designed with the understanding that the main expertise and responsibility for evaluation is centralized; an understanding that propagates the Webmaster bottleneck as well as other organizational gridlocks.

3.2. A Need for Integrating Web Accessibility Evaluation Tools

A new approach for Web accessibility evaluation tools could be taken by following a more distributed model where different tools, potentially developed by different vendors, could dynamically plug into a collaborative framework. The tools in this framework have varying functionality and user interfaces, to address the needs and preferences of the evaluators. For instance, one tool could focus solely on improving its heuristics for evaluating tables while it relies on the user interface being provided by a hosting authoring tool. Another tool could focus solely on processing and reporting evaluation results in different ways while it relies on other tools to carry out the actual evaluations and generate the initial results. There are many different combinations that could be realized, if the tools could communicate with each other.

In the paper A Proposed Architecture for Integrating Accessibility Test Tools [Englefiled et al. 2005], the authors outline similar motivations for integrating the Web accessibility evaluation tools into a common framework. The proposal is based on the following assumptions:

1. The coverage of tests carried out by different Web accessibility evaluation tools varies strongly between different types of tools so that developers need to use several tools in order to compensate for the lack of performance in each
2. The development of the crawling, parsing, reporting, and user interface infrastructure “consumes 40-50% of the effort” for each tool, and therefore has a significant impact on the costs for the production of the individual tools
3. Many of the Web accessibility evaluation tools are developed by academic, research, or charity organizations who may not have the expertise and competence to develop such a comprehensive infrastructure effectively
4. Learning the different user interfaces and specifics of the different Web accessibility evaluation tools, as well as having to use them sequentially in order to carry out the evaluation procedures is a significant burden

The conclusion drawn from the analysis in the paper is that an integration of different Web accessibility evaluation tools into a common framework would significantly reduce the costs of production. The framework is designed to provide the I/O and user interface infrastructure that is otherwise reinvented in each tool, if at all. Web developers and evaluators would only be confronted with a single user interface that allowed them to launch several evaluation tools in parallel, and manage the results in a uniform reporting functionality.

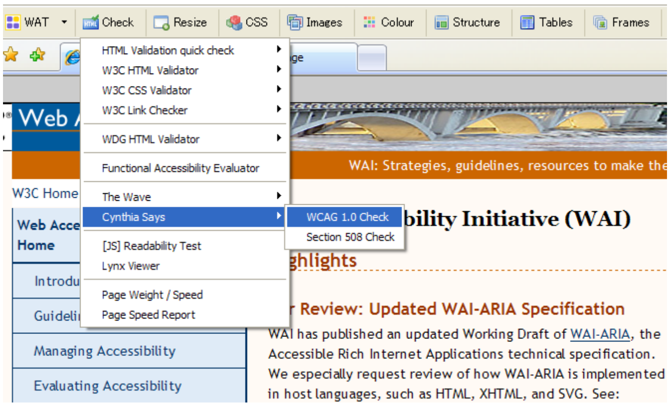
However, the paper focuses almost exclusively on automated testing in the analysis and in the approach. The framework is therefore limited to evaluation tools that do not need to provide a user interface and can be launched in the background. Many testing approaches and therefore also the evaluation tools require an interaction with the evaluators, especially if these tools provide an explanatory interface such as a wizard dialog. There is also an implicit assumption about the context for application of this framework, namely evaluation contexts that set out to assess the accessibility of complete Web sites. In other words, the framework is not suitable for a context in which an evaluator, such as a Web content author, wants to evaluate a single Web page for accessibility. It is also a centralized and monolithic approach that primarily serves the needs of Webmasters or other quality assurer roles in Web accessibility evaluation rather than the needs of the Web developers and evaluators during earlier development stages.

It is however imaginable to achieve a distributed model where different types of tools could connect in an ad-hoc manner, without any presumptions by the tool developers. There already exists such ad-hoc integration of different tools that were not specifically designed to work with each other to evaluate Web accessibility. Some of these examples are described below.

3.2.1. Example 1: Integration of Manual and Automated Evaluation Tools

Web accessibility evaluation tools that can be launched directly from within the Web browser have become increasingly popular. Initially there were several evaluation tool developers that offered a Web-based service. Developers could go to a Web page and type in the Web address for another Web page that they wanted to evaluate. This Web-based application launched an evaluation tool on the server that displayed its findings directly within the Web interface of the tool. This was a simple and comfortable approach for many developers. They only had to keep a list of the favorite services for evaluating Web pages in order to be able to launch them.

As Web browsers became increasingly customizable and allowed plug-in applications to be developed for them, Web accessibility evaluation tool developers started using this possibility to create a new form of toolbar tools. These tools provided functionality such as resizing the window and font sizes to observe how the Web content behaves, outline inline tables, forms, and other page structures, or reveal the text alternatives to compare the images and more. An interesting new feature was however the possibility of launching some of the Web-based tools directly from the toolbar and receiving the results directly within the browser. Especially the more experienced tool evaluators who know what they are looking for in a specific Web page seem to prefer such toolbar applications.



WAI has published an updated Working Draft of WAI-ARIA, the Accessible Rich Internet Applications technical specification. We especially request review of how WAI-ARIA is implemented in host languages, such as HTML, XHTML, and SVG. See:

Figure 8: Toolbar that can launch external evaluation tools

While some toolbar tools can launch many different external evaluation tools, they are usually not well integrated into the toolbars. Often initiating evaluation functions just takes the user to the external Web-based service, rather than to actually launch the external evaluation tool in the background and present the user with the results. This is due to the fact that the evaluation tools do not usually provide a uniform interface such as a Web-service[14] to interact with other applications such as the toolbars. Only in a few examples where the tool developers explicitly coded their plug-ins for specific services could they reach more transparent integration. These solutions are however not reusable for other services. In fact, most of the toolbars can not be easily customized to use different sets of external services because they need to hard-code the functionality around the services they want to provide. A uniform interface between the types of evaluation tools would improve their collaborative capabilities for the end-users.

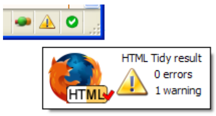


Figure 9: Browser plug-ins

---

[14] 'Web-service' refers to the technical standard as opposed to providing a service through a Web-based interface

3.2.2. Example 2: Integration of Authoring Tools and Evaluation Tools

The importance of providing accessibility support within authoring tools has been highlighted throughout many areas of this study. It is an essential aspect of Web accessibility as it allows Web content to be directly accessible at publication time, and it lowers the burden for both the individual evaluators as well as for the organization as a whole [WAI, 2]. Therefore the W3C Authoring Tool Accessibility Guidelines (ATAG) [WAI, 6] contains several requirements for authoring tools to carry out different types of evaluation checks to support the developers in creating accessible. Some of these checks are automated, for example to check the nesting of the elements in the markup code (this only applies to situations were the developers can enter the source code by hand, in WYSIWYG interfaces however the authoring tool should ensure the production of accessible code). Other checks are manual, for example to prompt the user to provide equivalent text alternatives for images that they want to publish.

While there is currently no tool vendor that claims full support for ATAG, many of them have been investing gradual work in improving the accessibility functionality. Many authoring tool manufactures now support at least some basic level of accessibility ‘out-of-the-box’. Often it is also possible to improve the initial level of support, for instance by setting specific types of preferences or by otherwise customizing the applications. In general there are four categories of authoring tools, each of which is typically used slightly differently:

1. Content management system – is typically template driven and limits the types of accessibility barriers that the often non-technical users can create.
2. HTML Editor – is typically used by slightly more technical developers, some of who may use the source-code views to manipulate the content directly.
3. Programming IDE – is typically used by programmers and application developers, especially to create Web applications with a run-time code.
4. Save-as HTML functionality – is typically used by non-technical users who prefer to work with the word-processing tools that they are acquainted with.

In each of these categories there are individual examples of integration between authoring and evaluation tools. However, most of these implementations are in the commercial or enterprise arena, especially to support commercial content management systems such as the Microsoft Content Management Server (MCMS). Other evaluation tools support market-leading HTML Editors such as FrontPage and Dreamweaver, or office applications such as Microsoft Word, Excel, and PowerPoint. Only few examples of evaluation tools support open source authoring tools, for instance some tools are already working towards the open Eclipse platform [ACTF].

This situation is probably due to the fact that it is not economically feasible for evaluation tool developers to support multiple frameworks and APIs. They need to select a minimal number of the frameworks that they can realistically support, and work towards these. However, since the overall economical market of Web accessibility is not immense, this situation is unlikely to change unless a new open and vendor-neutral approach is taken. A single framework that is common among several authoring tools and accessibility evaluation tools, and that facilitates a cost-effective mechanism for integration between different types of tools.

This assumption seems to be supported by several examples for the Eclipse or for the FireFox platforms. These open source products provide a framework for arbitrary plug-ins that allows the core functionality of these products to be extended. As these products became increasingly popular among the users, more evaluation tool developers started to work towards providing support for them. This concludes a market demand for integrated Web accessibility evaluation tools. However, the initial cost for supporting the framework needs to be lucrative.

3.2.3. Example 3: Integration of Data Analysis Tools and Evaluation Tools

There are not many examples of tools that specifically focus on analyzing Web accessibility evaluation data. However, several evaluation tools, especially enterprise commercial tools, have sophisticated data analysis and reporting functionality. Based upon evaluation results collected from automated testing as well as from additional testing carried out by the users, these tools monitor the progress on the Web site and analyze the results. For instance, they could highlight specific areas of the Web site that need more attention or Web pages that are more likely to contain accessibility barriers based upon the evaluation history. It is important to be able to provide the different developers with specifically tailored reports that contain the information that they need most. For instance, application developers may prefer the detailed and technical error reports in a bug-report format while project managers may prefer overall statistics and tables showing the progress. This reporting functionality in the evaluation tools is also part of the infrastructure that must be repeatedly reinvented in each of the tools.

One example of a tool that was specifically developed to analyze the evaluation results that are generated by Web accessibility evaluation tools is the Simple Tool for Error Prioritization (STEP) [Gieber and Caloggero 2006]. STEP has some built-in information about the relative ‘severity’ for some of the accessibility barriers so that it could prioritize the issues that it recognizes from the results in the evaluation reports. It also provides functionality to visualize the data and to highlight some of the key areas of focus. The STEP tool can therefore provide additional information about the data than the raw evaluation reports, and is a useful tool to help manage the accessibility of Web content. However, STEP relies on the evaluation tools to provide machine-readable and structured formats for the evaluation results so that it can process them. Moreover, because Web accessibility evaluation tools provide different formats for their reports, STEP was only designed to support a limited number of evaluation tools. It had to build a pre-processor for each of the formats in order to convert them to an internal form that it can process uniformly. It is maybe the reason why STEP is only one of the few examples of such tools available.

A slightly different example of a tool that included data analysis functionality that is based on the evaluation results generated by Web accessibility evaluation tools is the European Internet Accessibility Observatory (EIAO) [EIAO]. The EIAO framework is significantly larger than the data analysis module, as it is primarily intended to monitor the accessibility of Web sites all around Europe. It contains crawlers that fetch selections of pages from candidate Web sites, a sampling module to identify representative Web pages, and a data warehouse that contains pre-parsed information for the reporting module. The interesting part in the context of this study is however the testing module which launches different kinds of quality assurance tools to measure the required ‘Web Accessibility Metrics’. It is important to note that EIAO is a fully automated approach so that the framework resembles more the proposal in [Englefield et al. 2005] than the one suggested by this study. It does however underline the demand for a uniform framework in which Web accessibility evaluation tools can be integrated into a data analysis system, in this specific case to carry out large-scale Web accessibility monitoring.

4. Promising Techniques for Web Accessibility Metadata

In the previous sections the current state of the art in Web accessibility evaluation was under investigation. It was identified that there are:

- Several phases of Web development in which accessibility evaluation occurs
- Different types of accessibility testing that is carried out on the Web content
- Variety of actors that carry out accessibility testing for different purposes

There is a similarly large variety of actors that need to access or process the information that is generated about the accessibility of the Web content. For instance, Web developers may be interest in the problems found on specific Web pages or collection of Web pages so that they can retrofit these. Web project managers may be interested in getting overviews about the overall level of accessibility for entire Web sites. All the involved parties, regardless if they are contributing or consuming accessibility information for Web content need to be able to share data. They need to be able to exchange the information in a collaborative environment.

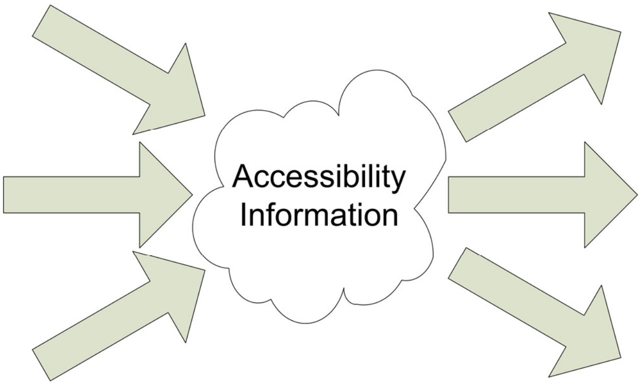


Figure 10: Collaborative environment

Often enterprise Web accessibility evaluation tools provide functionality to manage a central repository of accessibility information. They support different types of roles that may want to carry out evaluation or query the data. However, each evaluation tool has its own proprietary system and controls the data. The information is often locked into the specific evaluation tool and can not be easily shared with other types of tools such as authoring tools.

The central question in this section is if there are promising technologies that enable this basic collaborative system to be extended beyond Web accessibility evaluation tools alone. In other words, the evaluation tools should be a part of the suite of applications that interact with the accessibility information but the system should not be confined to such tools. It is imaginable to have other types of tools interact with this information. For instance, drawing utilities such as those often used by visual designers may be able to output accessibility information about the color contrast or about the font selection. At the same time, a reporting tool that generates statistics and overviews (not necessarily about accessibility only) may need to access this data repository to include the information in the surveys.

This concept of ‘opening the data’ or sharing knowledge across different types of applications is the underlying concept of the Semantic Web. It is an already existing framework with many readily available standards, tools, and resources so that it lends itself to closer inspection. The domain of Web accessibility evaluation can potentially benefit from reusing or building upon this comprehensive set of technologies that constitute the Semantic Web.

4.1. The W3C Evaluation and Report Language (EARL)

The W3C Evaluation and Report Language (EARL) [WAI, 9] is a machine-readable format to express test results, such as those generated by Web accessibility evaluation tools. EARL is designed to be flexible and accommodate different types of Web accessibility evaluation tools that currently exist, and different types of contexts in which they may be used. For instance, EARL does not make assumptions about the tests that are carried out or how they fit into any evaluation procedures or into evaluation methodologies. Instead, EARL captures information about a test execution. EARL provides syntax to capture the following type of information:

- What or who carried out the test?
- What was the subject under test?
- Which criterion was tested for?
- What was the result of this test?

This basic set of information is applicable to any quality assurance test run for any purpose. For instance, any bug report should contain exactly this type of information so that the error can be reproduced and followed-up on. In fact, EARL is designed to be reusable for generic quality assurance purposes outside the field of Web accessibility evaluation. It was however specifically optimized to address Web accessibility evaluation, especially through extensions to the core EARL vocabulary that allow EARL reports to record the specifics of the content tested. These extensions are described in later parts of this section.

While EARL was primarily designed to provide a uniform format for exporting test results in Web accessibility evaluation tools, there are several additional scenarios for which it can be used. EARL was designed to address the following use-cases:

- **Combining Results** – a standardized format for expressing test results allows the combination of output from different Web accessibility evaluation tools. This enables different reviewers to carry out separate evaluation tasks using different tools, and for reviewers to employ different tools at different stages evaluation of the process, yet still be able to combine the results effectively in a uniform manner.
- **Comparing Results** – calibrated results, such as results from pre-defined test suites, could be recorded in the same standardized format that Web accessibility evaluation tools use to express test results. This allows evaluation tool developers to compare the output from their tools to the test suites, and for end-user to have better benchmarks about the performance of the different evaluation tools.
- **Processing Results** – quality assurance and monitoring tools for Web accessibility can rely on a standardized format to analyze, sort, and prioritize issues. For instance, tools could specialize in analyzing evaluation data that is generated by evaluation tools without actually executing any tests. These tools could monitor the overall status and highlight specific areas of Web sites that need more attention.
- **Reporting Results** – tools that specialize in visualizing data and on reporting could rely on a standardized input format for the data. These reports can be used to provide customized views such as more verbose bug reports with line numbers and error messages for the programmers, or statistical and higher-level management reports for the project managers and executives.

- **Authoring Tool Integration** – Web authoring tools could use the standardized format to integrate the test results provided by Web accessibility evaluation tools directly into the development environment of the developers. It enables evaluation tool developers to focus on the testing techniques and algorithms, and rely on the authoring tools to provide the user interface for the developers.
- **Web Browser Integration** – Web accessibility evaluation results in a standardized format could be used by Web browsers to improve the experience of the end-users. For instance, Web browsers could use the results from Web accessibility evaluation tools to indicate Web pages that do not match the requirements of the end-users, for example because they content moving or distracting content.
- **Search Engine Integration** – similar to Web browsers, search engines could enhance the experience of the end-users by considering their preferences for accessibility. For instance, users may want to limit their search results to Web pages that they can use effectively, for example to prefer Web pages that can be operated through a keyboard or Web pages that do not cause them to have seizures.
- **Accessibility Metadata** – Web site owners may choose to publish information about the accessibility tests that were carried out on their Web pages, in order to promote transparency and confidence. For instance, the test results may be used to supplement a conformance claim or a quality mark, and provide the information about the quality assurance testing in a standardized format.

While some of these use-cases have been realized in different forms, many of them remain to be realized. EARL is currently still under development so that only few implementations are available. However, the motivations for EARL fit well with the current move towards a more semantically rich Web in which different types of tools can share and exchange information. More discussion about the potential impact of EARL in improving the overall availability and performance of Web accessibility evaluation tools can be found in later sections of this study.

Inline with the move towards more semantics on the Web, EARL is designed using the W3C Resource Description Format (RDF) [W3C, 32]. RDF provides a semantic Web framework to develop semantically rich vocabularies. For instance, RDF provides a framework to define the terms in EARL such as `Assertor`, which represents a person or a tool that carried out the test. The formalization and processing rules for this term is governed by the RDF framework and does not need to be reinvented for EARL. The EARL specification only serves to define the meaning for each of the terms that it introduces.

In a way this is similar to defining the terms in a framework like XML, however RDF is more semantically rich and can provide more information about the relationship between different entities. Also similarly to XML, RDF is supported by a large community so that EARL can benefit from the availability of processors, libraries, and other resources. In fact, EARL reuses existing RDF vocabularies to describe people or specific types of relationships rather than to redefine such widely deployed terms.

Another important aspect of RDF is its flexibility for extension through inheritance features. For instance, the more specific EARL term `Software` can be used instead of the abstract term `Assertor` from above. Even if a processor does not know the exact definition for this term, it understands that it is a type of assertor and can continue working with the information. EARL therefore provides a minimal set of terms to construct semantically rich assertions about test results. It is a common language for different tools to exchange specific type of information.

4.1.1. W3C Evaluation and Report Language (EARL) 1.0 Schema

The W3C Evaluation and Report Language (EARL) 1.0 Schema [W3C, 27] provides the core EARL vocabulary. It defines a set of terms in RDF to provide a common format for quality assurance tools, such as Web accessibility evaluation tools, to describe test results. The basic structure of an EARL report is a collection of assertions about test results. An `Assertion` in EARL is defined to be "a statement about the results of performing a test" [W3C, 27]. Each assertion creates a relationship between entities of the following:

- **`Assertor`** – An *Assertor* determines the results of a test.
- **`Subject`** – The *Test Subject* is the class of things that have been tested.
- **`TestCriterion`** – A *Test Criterion* is a testable statement.
- **`TestResult`** – The actual *result* of the test.

EARL 1.0 Schema, Assertion – http://www.w3.org/TR/EARL10-Schema/#assertion

Each of these basic building blocks provides semantics to provide more detailed information, for example about the specific type of assertor or about the subject that has been tested. The assertor can be one of two of the following types:

- **Single Assertor** – a single entity such as person, organization, or software.
- **Compound Assertor** – a group of one or more entities, with at least one of them being the primary entity responsible for the claim in the test result.

This definition for the Assertor accommodates many different combinations to describe many types of situation, especially in collaborative environments. For instance, if an evaluator was using a tool to conclude a test result, this could be represented through a Compound Assertor with the person as the primary assertor (`mainAssertor`) and the tool as the secondary assertor (`helpAssertor`). Conversely, it is imaginable to have the opposite situation in which a tool uses human judgment to conclude a test result. In this case the software tool does not require that a human evaluator carries out the test procedure but rather to provide input. For instance, if a tool can not automatically determine if a table is used for layout or for providing data, it may ask a human evaluator to provide a judgment. Based upon this input the tool can now resume testing the syntax of the table to serve the specific purpose (layout tables should not have header or summary elements, data tables on the other hand should provide these).

While the Assertor class in EARL is quite sophisticated and well established, the Test Subject is quite abstract and generic. It only provides some basic properties to provide rudimentary information and describe simple relationships. For instance, to describe a Web page that has been tested, the following EARL code could be provided:

```
<earl:TestSubject rdf:about="http://www.example.org/index.html"/>
```

In this case EARL is taking advantage of the fact that RDF is built on the Universal Resource Identifier (URI) schema so that the address of the Web page is a sufficient identifier for the Test Subject. While one can add some context information, such as the date for fetching the subject or in-line parameters passed on to the Web page, in general this approach has several limitations to addressing Web content. Especially dynamically generated content, such as that created by client-side scripts, is difficult to represent using only the URI for the Web page. A more comprehensive approach to represent this type of Web content is explained in the next sub-section. It is however important to note that this class has been kept intentionally generic so that it can be applied to other contexts, for example for non Web-based content.

A Test Criterion in EARL is a testable statement against which the Test Subject is tested. It is difficult to define this term any closer as it strongly depends on the type of test and the type of subject that is being tested. Some claims that quality assurance tools may want to make could be very broad (for instance 'subject X conforms to standard Y'), while others may be specific (for instance 'element X has property Y'). While both assertions essentially have the same syntactic and semantic structure, they have very different implications in relationship to their meaning. To better address specific situations, EARL provides two further refinements:

- **`TestRequirement`** – A higher-level requirement that is tested by executing one or more sub-tests. For instance, WCAG 1.0 Checkpoint 1.1 which is tested by executing several sub-tests and combining the results.
- **`TestCase`** – An atomic test, usually one that is a partial test for a requirement. For instance, checking if an image has an `alt` attribute which could be part of testing WCAG 1.0 Checkpoint 1.1.

EARL 1.0 Schema, Test Criterion – http://www.w3.org/TR/EARL10-Schema/#testcriterion

The actual payload of an EARL assertion is provided through the Test Result. It contains both structured as well as unstructured information. For instance, the Test Results provides a value for the Outcome of the test which could be one of the following predefined values:

- **`pass`** – An assertor claims a test passed successfully.
- **`fail`** – The *Test Subject* did not meet the *Test Criterion*.
- **`cannotTell`** – An *Assertor* can not tell for sure what the outcome of the test is. Usually this happens when an automated test requires human judgment to make a definite decision.
- **`notApplicable`** – The *Test Criterion* is not applicable to the given *Test Subject*.
- **`notTested`** – Test has not been carried out. This is useful for reporting as well as for other uses of progress monitoring.

EARL 1.0 Schema, Outcome Value – http://www.w3.org/TR/EARL10-Schema/#outcomevalue

Finally, EARL also provides an optional property to record information about the mode in which the test was carried out in. The Test Mode can be one of the following values:

- **`manual`** – Where the test was performed based on a person's judgment. This includes the case where that judgment was aided through the use of a software tool (…)
- **`automatic`** – Where a software tool has carried out the test automatically without any human intervention.
- **`semiAutomatic`** – Where a software tool was primarily responsible for generating a result, even if with some human assistance.
- **`notAvailable`** – Where a combination of persons and/or software tools was used to carry out the test, but there is no detailed information about the primary responsibility for determining the outcome of the test. This includes when testing is carried out by organizations or groups of assertors, and the exact testing process is not disclosed.
- **`heuristic`** – This property was designed to cover assertions which are made by inference, for example based on several existing test results.

EARL 1.0 Schema, Test Mode – http://www.w3.org/TR/EARL10-Schema/#testmode

4.1.2. W3C HTTP Vocabulary in RDF

The W3C HTTP Vocabulary in RDF [W3C, 29] is an extension to the W3C Evaluation and Report Language (EARL) 1.0 Schema [W3C, 27] vocabulary, and provides terms to record HTTP exchanges between clients and servers. These terms can be used for several purposes, for instance by quality assurance tools that want to evaluate the compliance of HTTP servers to the specification. However, it has been specifically created with the motivation that Web accessibility evaluation tools could use it to record information about the Web content that they are testing. For instance, an evaluation tool may want to keep information about the HTTP headers that were sent to the server, for example during a content negotiation exchange. Without this information, Web developers may not be able to identify the versions of the Web pages that were evaluated, and may therefore not be able to reproduce the problem.

This scenario is applicable to both automated Web accessibility evaluation tools, as well as to manual evaluation tools. For instance, a toolbar application could capture the HTTP exchange and provide it for the evaluators to include in their evaluation reports. Recording these HTTP exchanges is vital for two primary reasons:

- Web sites are commonly driven by cookies, which store information about the user's preferences or behavior. This means that two different users may essentially receive different content depending on the parameter values stored in their cookies. In order to reproduce the content, these same parameters must be known to the developers.
- Web applications commonly fetch additional information from the server using AJAX technology. The Web content as observed by the evaluator or evaluation tool is built dynamically through a series of HTTP exchanges with the server. These exchanges depend largely on the interaction of the users with the Web content and can therefore not be anticipated. Recording them is one approach to reconstruct the situation.

The vocabulary to record such HTTP exchange is designed from the perspective of an entity that is located at the client and is recording the communication coming from both directions. The basic class is the `Connection`, which is used to record information about the connection; for instance, which server host was contacted and at which port. The Connection Class also records the series of Request and Response messages that were sent by the client and received from the server. Each of these classes can contain properties that represent the HTTP headers that were exchanged within these messages. For instance, these headers are typically at least the Request URI to fetch a specific Web page from the server, and a Response Code sent by the server to confirm each request.

The W3C HTTP Vocabulary in RDF contains terms to record the HTTP headers as received from the server (usually this means as a literal value), as well as representations for most of the commonly used headers. For instance, to facilitate efficient processing and querying of recordings provided in HTTP Vocabulary in RDF format, the vocabulary contains dedicated terms to represent common HTTP headers such as the `accept-language`, which is commonly used to negotiate the language preference for the Web content. At the same time, the actual sequence of characters '`accept-language`' received from the server can also be recorded in addition, for example to debug faulty HTTP exchanges or to preserve the exchange in its original form to the largest extent possible.

HTTP requests and response message have body elements which contain the actual content of the messages. However, since these could be byte sequences in any format, for instance to transmit non-text content such as images, it is not trivial to record this as RDF because it is text based. A further extension to this vocabulary has been developed to tackle this issue.

4.1.3. W3C Representing Content in RDF

The W3C Representing Content in RDF [W3C, 30] vocabulary is designed to represent any type of content in RDF format. For instance, it could be used to represent text-based, XML, or binary content that is stored in a local media or on the Web. It is an extension to the W3C HTTP Vocabulary in RDF [W3C, 29], which was described in the previous sub-section but it can also be used separately. For instance, SMTP messages have a very similar structure to HTTP messages so that this vocabulary could be potentially reused in another context which was not specifically addressed by EARL. This work could also be used to directly extend the core W3C Evaluation and Report Language (EARL) 1.0 Schema [W3C, 27] vocabulary, for instance to represent the subject that was evaluated.

However, the primary use-case for which this vocabulary was developed is for it to be used in conjunction with the W3C HTTP Vocabulary in RDF, and as part of an EARL 1.0 report. It is designed to provide different representations for content, potentially for the same content. For instance, an XHTML Web page returned by a server in response to an HTTP `GET` request could be recorded in an XML representation as well as in a plain-text representation for tools (including browsers) that are unable to parse XML effectively. It could also be represented in binary (byte64) encoding to overcome any potential character encoding issues and to preserve the original byte sequence returned from the server to the largest extent possible.

Besides a generic and abstract Content class the W3C Representing Content in RDF provides the following three main classes as instance representations for the content:

- **Base64Content** – Base64 encoded binary content as defined by RFC 2045
- **TextContent** – textual content using a specific character encoding
- **XMLContent** – XML-based content using XML declarations

While recording the full content downloaded from the server may seem to be excessive, there are different situations when it is useful. For instance if only a small sample of Web pages is selected to represent entire Web sites, it may be necessary to precisely document what has been subject to testing. Also when heuristics are involved or if further testing needs to take place, it may be worth capturing the current content being evaluated. The following example shows how Web content from a specific URI could be recorded, along with the relevant HTTP headers, in an EARL 1.0 report using different representation for the content:

```
<earl:TestSubject rdf:about="http://www.example.org/index.html">
      …
      <http:Connection …>
            …
            <http:Response …>
                  …
                  <http:body>
                        …
                        <content:XMLContent …>
                              …
                        </content:XMLContent>
                  </http:body>
            </http:Response>
      </http:Connection>
      <content:TextContent …>
            …
      </content:TextContent>
</earl:TestSubject>
```

Code Listing 1: Representations of content in EARL 1.0

4.1.4. W3C Pointer Methods in RDF

The W3C Pointer Methods in RDF [W3C, 31] is also an extension for the W3C Evaluation and Report Language (EARL) 1.0 Schema [W3C, 27] vocabulary. However, it is intended to be used as part of the Test Result, and to identify the key areas within the Test Subject that lead to the Test Result. For instance, if a test to check the existence of `alt` attributes for `img` elements in HTML failed because one or more instances in a Web page did not provide such attributes, then the pointer vocabulary could be used to identify these instances that did not meet the test requirement. In a trivial approach, these identifiers could be line and character counts or another offset value. However, depending on the type of content being tested there could be other means to provide more indicative identifiers.

Traditionally software debugging has been using line and character counts to identify areas in the source code that caused an error. While this approach is simple and effective for syntax checks, such as during compiling or pre-processing software code, it has many considerable limitations. For instance, it may be more precise to describe an HTML element using XPath or XPointer identifiers than using the line and character counts. It is more semantically robust to identify such complete elements, attributes, or areas of the Web content rather than to point to the beginning or the end of these because it could lead to ambiguity. It may also be more resilient to changes that can occur in the source code which may lead to destruction of all the offset-type identifiers that rely on a specific version of the source.

Moreover, it would be useful to provide identifiers for non text-based content. For instance, it would be important to identify the relevant part of a multimedia file, such as an audio or video track, that does not provide adequate captioning. In many cases one may also want to cut out a piece of the content, a snippet, and provide it as an exemplary instance for the error. This is the case for text-based or for byte-based content. However, in both cases a line and character count approach will not suffice to provide such information.

The W3C Pointer Methods in RDF vocabulary therefore describes a variety of different types of pointers that could be used to identify the key pieces of the content. Different pointers can be used in an `EquivalentPointer` class, each of which is assumed to be equivalent. In other words, each is a different pointer format to identify the same piece of the content. The tool that is processing the information can select any of the formats provided, depending on its preferences and capabilities. Some of the varieties provided by the vocabulary include XPath, XPointer, CSS Selector, Offset and Line/Character counts. In addition, the vocabulary also provides pointers that can identify entire ranges, such as those that provide a start and an end or those that record a snippet. For instance, the pointer type `StartEndPointer` contains a start pointer and an end pointer so that it could point to the beginning and to the end of an item list.

A different use-case for combining pointers is to identify causal effects that lead to an error. This is a situation that is less common in traditional software programming. If for instance the background and foreground color combinations do not provide sufficient contrast, then it would be important to be able to point the developers to both color definitions rather than to only one. That means that it would need a mechanism for pointing to two different locations within the content that are together responsible for the result of the test. It is different than a range pointer as there may be no sequence between the two areas identified. In fact, in some cases they may be located on different Web pages (for instance to test consistent navigation).

The W3C Pointer Methods in RDF vocabulary is currently an early draft that is undergoing a lot of development and refinement. It is however a useful extension to the core EARL 1.0 vocabulary as it significantly contributes to the machine-readability of test reports.

4.2. Other Relevant Semantic Web Technologies

W3C defines the Semantic Web as a Web of data. It envisions that data in daily applications is provided in a common language that can be understood and used by browsers. The typical example provided is linking transaction statements from a banking application to a calendar application, rather than having these two sets of information be disjoined. It is connections or 'links' that human can make between different sets of related information, but is however not systematic enough for computers to be able to process effectively.

The basic building block of the Semantic Web is provided by the W3C Resource Description Format (RDF) [W3C, 32]. It is a simple format that allows any resource that is identified by a URI to be described with properties. For instance, statements such as `"John" isA Person` or `Person has Parent` can be expressed using RDF. Ontologies, such as the W3C Evaluation and Report Language (EARL) [WAI, 9], can be created using RDF to describe specific type of information. In the case of EARL, it is to describe test results such as those created by Web accessibility evaluation tools (see section 4.2. Evaluation and Report Language).

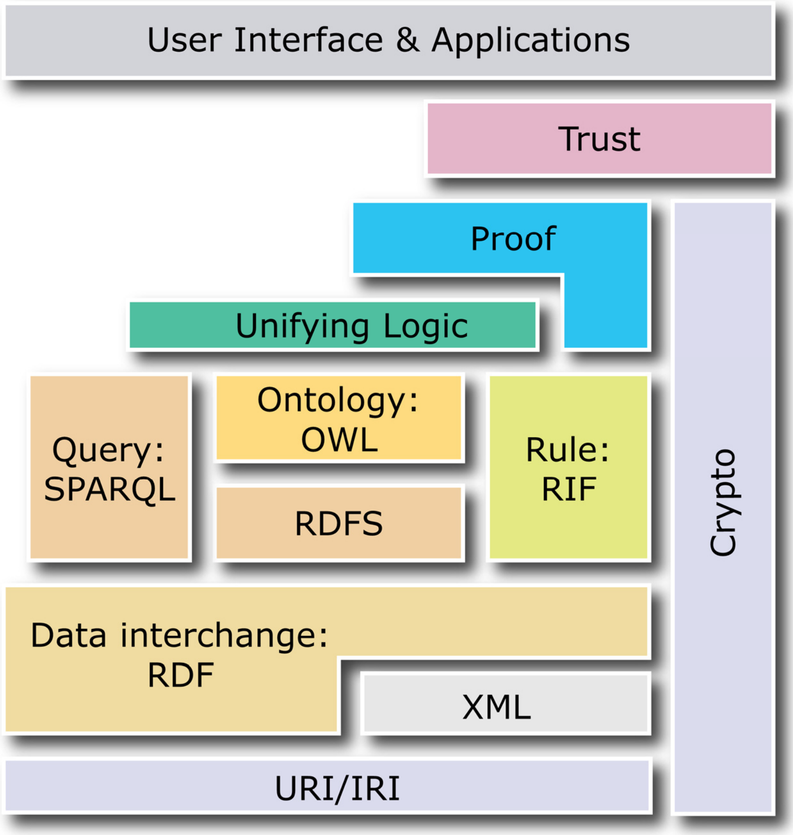


Figure 11: Semantic Web layer-model

While the Semantic Web has not yet reached wide-spread deployment in daily applications, there are several attempts to make use of the currently existing data and formats to enable the exchange of information between different types of applications. There are also ontologies emerging in different domains that could prove to be useful in the context of this study. This section looks at some of the currently existing standards in the domain of the Semantic Web and how these could be useful for advancing the concept of tool-supported Web accessibility evaluation. In particular, this section looks at how accessibility metadata could be exchanged between different types of tools that aim to improve or manage the accessibility of Web sites.

4.2.1 W3C Protocol for Web Description Resources (POWDER)

The W3C Protocol for Web Description Resources (POWDER) [W3C, 42-44] is a format to help identify the content of Web resources. For instance, it could be used to label Web sites according to child protection ratings to identify content that contains violence or potentially offending material. Similarly, POWDER descriptions could be used to identify the level of accessibility provided by Web sites or individual Web pages. This type of information could be useful to end-users in many ways:

- Identify content that particularly matches their needs and preferences;
- Assess the expectations for the level of quality provided by the content.

An example use-case is for search engines to consider such labels and provide the end-users with more directly matching results. POWDER has been specifically designed to work with HTML through the usage of the `Link` element. For instance:

```
<link rel="powder" href="powder.xml" type="application/xml">
```

This directive can be processed by applications that understand the `powder` protocol, and can relate the referenced XML data with the HTML document. Similarly to the W3C Evaluation and Report Language (EARL) [WAI, 9], POWDER makes use of the RDF/XML notation to support XML-based applications. POWDER is however an RDF language and can be reused or extended in other vocabularies. In other words, data in POWDER and EARL format could be used together to comprehensively describe the accessibility of resources. For instance, the CMS used to create the Web content could provide a machine-readable claim about the level of accessibility of the resource using POWDER, and provide an EARL report that contains detailed information about the accessibility testing that has been carried out in order to justify the claim. The EARL data would contribute to the trust and perceived validity of the claim while the POWDER claim provides simple information that could be searched for more easily.

4.2.2. W3C RDFa: Bridging the Human and Data Webs

The W3C RDFa [W3C, 38-39] is a notation to embed RDF data into XHTML documents. It makes use of the existing XHTML framework to add metadata. For instance, the following code could be used to identify the author of the Web content:

```
<meta property="dc:creator" content="John Doe" />
```

This is similar to the currently widely used approach of using the `meta` element to identify such metadata, however, RDFa uses ‘compact URIs’ to refer to RDF ontologies. In this case it refers to the property `creator` that is defined by the Dublin Core Metadata Initiative [DCMI]. RDFa also makes use of many other hooks within the XHTML framework including the `rel` and `rev` attributes which can be used within the anchor elements. This is particularly useful for linking datasets or otherwise relating them to each others.

There are two interesting use-cases for RDFa in the context of Web accessibility evaluation:

- **Identifying accessibility** – for Web applications that aggregate content from different sources or that include user generated content, RDFa could provide a mechanism to add accessibility metadata directly into the Web content. It could used to identify parts of the Web site that conform or that do not conform to the accessibility requirements.
- **Providing single reports** – for Web accessibility evaluation tools that can generate reports, RDFa could provide a mechanism to insert machine-readable data directly into the XHTML reports that are intended to be read by human evaluators. It is a mean of providing a single report with rich information that could be repurposed as needed.

4.2.3. W3C Gleaning Resource Descriptions from Dialects of Languages

The W3C Gleaning Resource Descriptions from Dialects of Languages (GRDDL) [W3C, 40] is a format for declaring that XML-based documents contain data that could be represented in RDF, and links to transformations (such as XSLT) that can be used to extract this data from the source document. In a way this is similar to RDFa although the RDF data is not as directly embedded into the source documents in a uniform manner but it relies on a transformation to extract the data. The transformation is therefore application specific and 'knows' how to parse the source document for relevant information. This is then exported into uniform RDF data.

GRDDL provides a promising approach for many of the current Web accessibility evaluation tools that already export information in XML-based formats. Rather than updating the core of the application to generate EARL reports, tools could use external transformations (such as simple XSLT scripts) that convert the source data into the EARL format. This approach could also be used by authoring tools such as CMS or quality assurance tools such as issue and bug tracking systems to export any relevant accessibility information in a uniform format.

4.2.4. W3C SPARQL Protocol and RDF Query Language (SPARQL)

The W3C SPARQL Protocol and RDF Query Language (SPARQL) [W3C, 33-35] is both a protocol as well as an RDF query language in a single framework. The query language was specifically designed to be similar to the widely deployed Structured Query Language (SQL) that is used for relational databases. However, the SPARQL query language is designed to traverse RDF data and to perform similar types of functionality as SQL. The protocol part allows applications to query HTTP servers directly. The query and response messages are wrapped into HTTP envelopes. From the application perspective it is irrelevant if the data is stored in separate databases or if it is directly part of the Web content.

SPARQL provides a heart piece for the Semantic Web and enables powerful interactions with the underlying data. There are already several implementations for SPARQL that are provided in different programming languages and frameworks. Web accessibility evaluation tools can make use of these readily available libraries and modules to access the data provided by other tools. For instance, Web accessibility evaluation tools could query their own reports generated through previous evaluation runs in order to sort, filter, or summarize the findings. At the same time, these tools are not confined to managing reports generated by themselves but can essentially make use of any reports available in a uniform format such as EARL.

4.2.5. W3C Rule Interchange Format (RIF)

The W3C Rule Interchange Format (RIF) [W3C, 41] is in it infancy but promises to become a central component of the Semantic Web. It enables basic Horn-type rules and other types of constraints to facilitate the construction of simple logical statements. For instance, basic '`if … then … else …`' type conditions can be declared for processing RDF data.

RIF is particularly interesting for formalizing the logic that is part of test descriptions, as it is described in the next section of this study. It is unclear how powerful RIF logics will become and which exact functionality it will provide as it develops. However, it is likely that RIF will provide sufficient logic to express the structure of basic requirements, such as the W3C Web Content Accessibility Guidelines (WCAG) [WAI, 5], in a machine-readable format. This could be used by authoring tools, quality assurance tools, and Web accessibility evaluation tools to process EARL reports and to identify which requirements the individual accessibility tests map to. It enables a basic framework to manage accessibility requirements set out by technical standards and the accessibility data generated different types of tools.

5. Excursion: Proposed Test Description Notation Format

Formalized test descriptions express testing procedures in a uniform format. For instance to express the testing procedures provided by the Techniques for WCAG 2.0 [W3C, 21] using formal notations rather than natural language. Currently there is no widely accepted format for expressing testing procedures for Web content. This is partially due to the fact that testing procedures tend to be application specific, since they are translations of the requirements from natural language into computer logic. Moreover, they are very domain specific with regard to the type of requirements that need to be formalized as well as the type of subjects that need to be tested. For instance, testing the syntax of markup code or formal grammar is significantly different to testing the accessibility behavior of navigation mechanisms.

In traditional software testing, testing procedures primarily describe parameters for executing tests and the expected outcome value. These are used by test harness systems to compare the output of the application or component to the normalized input and output pairs. While this approach can be reused to create test cases that assess the syntax of markup or the validity of programming logic that is part of Web content, it can not be easily reused to assess the user interface behavior. The W3C Test Metadata [W3C, 37] describes properties that are needed to describe test cases for Web quality assurance. It does not however formalize the actual testing procedure, such as the testing steps or the sequences in which they need to be carried out.

Testing user interface behavior is significantly different than testing programming logic or formal grammar. There is no output value as such but rather an output behavior. For instance, what is expected to happen when a button is selected? Some software quality assurance tools can model user interface structures. For instance, Rational[15], TestComplete[16], or WebKing[17] use the Universal Modeling Language (UML) or variants of it to constrain how elements can be nested or where properties may be used. Such models could potentially address some of the accessibility checks that are based on user actions such as the situation described above.

However, testing user interfaces, such as for accessibility, often includes qualitative tests. The main interest is to evaluate the behavioral and interaction characteristics of the Web content rather than to compare any output to test cases. UML-type modeling languages may also not suffice to express the actual steps that need to be taken in order to evaluate the Web content.

Formalizing the accessibility requirements for Web content into a uniform notation such as pseudo-code has several benefits:

1. **Clarity** – facilitates a common understanding for the requirements.
2. **Coverage** – ensures that tests address all aspects of the requirements.
3. **Validity** – helps validate and possibly optimize the requirements.

While there is currently no single and widely used standard for formalizing test descriptions for Web content, there are individual projects and initiatives in the field of Web accessibility, usability, and Web quality assurance with interest in such formalizations. However, currently there does not seem to be as strong interest from industry vendors as there seems to be from researchers and other practitioners. On the other hand, quality assurance is generally less well attained by many organizations and developers, especially in the Web arena.

---

[15] http://www.ibm.com/software/rational/

[16] http://www.automatedqa.com/

[17] http://www.parasoft.com/

5.1. Example Formalizations in Web Accessibility Evaluation Tools

Unfortunately there is not much documentation about the specific approaches that are adopted by Web accessibility evaluation tool developers. However, it seems that there is a significant trend for evaluation tools to move away from hard-coding the evaluation logic and implement more flexible approaches using rule-sets. One motivation for evaluation tool developers is so that they can more easily adapt their tools to different technical standards. For instance to be able to accommodate WCAG 1.0, Section 508, as well as the new WCAG 2.0 requirements without needing to redesign the entire application. Another growing motivation for evaluation tool developers is to be able to accommodate other types of requirements such as the W3C Mobile Web Best Practices (MWBP) [W3C, 33]. The evaluation tools are morphing into rule execution engines that execute arbitrary rule-sets that are plugged into them.

From analyzing a small selection of freely available Web accessibility evaluation tools it can be concluded that many of the rule formalizations share a similar approach. This approach is generally referred to as Keyword-Driven testing[18]. It is an approach that creates a mapping between the input data that initiates triggers and the actions that are then executed. While the studied implementations use different programming languages, technologies, and formalities to express these relationships, it is the basic structure that seems to be common among many of the tools observed. Also many of the commercially available tools are designed to have interchangeable rule-sets so that they can be easily customized. It is unclear how these are realized internally but it is an indication that they must also have some form of formalization.

The evaluation tools selected for this study show varying degrees of automation. They were also noticeably developed for different types of target audiences. For instance, some tools are more verbose and are designed to assist the evaluators through the evaluation process, while others target more experienced evaluators. One tool is specifically targeted to Web application developers and has likely therefore adopted a sophisticated framework for testing. It was also designed to be used as a plug-in for an IDE rather to provide its own user interface. In other words, there are many ways in which the same requirement could be translated into a set of rules. These differences depend largely on the capabilities of the tool, but also on the method that the tool employs for interacting with the evaluators.

There were also considerable differences with regard to declarative versus procedural variants of the rule-sets. While one tool chose a strictly declarative approach, another chose a strictly procedural approach. However, the other two examined in this case study seem to use both of these approaches combined. For instance, they use declarative approaches for identifying the key parts of the (markup-based) Web content, but then revert to procedural approaches or for executing the actual tests. In general there is a strong dependency between the rule-sets and the rule execution engines. However, there seem to be significant commonalities between the different approaches so that a common framework may be possible to construct.

---

[18] Also referred to as Table-Driven or Action-Word testing http://en.wikipedia.org/wiki/Keyword-driven_testing

5.1.1. ATRC Accessibility Checker

- Reference: http://checker.atrc.utoronto.ca/

The ATRC Accessibility Checker is a Web-based evaluation tool. It was developed to execute automated tests for WCAG 1.0, but it was designed to be flexible and adaptable to WCAG 2.0 or other technical standards. Rule-sets are separated out from the rule execution engine into an XML-based file. The rule-set file contains all the information that is necessary for the tests to be carried out. This includes textual descriptions of the test procedures, the dialogs and error messages, as well as additional information and advice about the purpose of the test. Much of the information in the rule-sets is similar to that provided by the W3C Techniques for WCAG 2.0 [W3C, 21] or the Understanding WCAG 2.0 [W3C, 20] documents that were introduced in section 2.1.1.1. W3C Web Content Accessibility Guidelines (WCAG) 2.0.

While the rule-sets also contain XQuery expressions to formalize some of the syntax-based tests, they do not seem to be required. The necessary directives for the test executions are the `trigger` which specifies the trigger, and the `function` which specifies a Boolean function to be executed. For instance, consider the following code that checks for HTML images that do not have an `alt` attribute:

```
<machine>
      <trigger element="img"/>
      <function call="attribute-missing" node="." value="alt"/>
      <xquery>
            //img[not(@alt)]
      </xquery>
</machine>
```

Each `img` element that appears in the input Web content is therefore a trigger for the function `attribute-missing()`, which checks if the current node (the image itself) has an attribute that is called ‘alt’. The rule-sets for the ATRC Accessibility Checker can also contain simple logics, such as ‘AND’ or ‘OR’ combinations of the Boolean functions that are executed in by the `function` directive. This functionality is very important to keep the Boolean functions atomic and simple, and allows the overall logic to be described in the rule-sets.

The rule-sets also provide simple mechanisms to define the sequence for the tests the needs to be followed in order to evaluate for a requirement. For instance, the following part of the test description contains the error message that should be displayed to the evaluator when the test fails, and identifiers for subsequent tests that should be executed if the test succeeds:

```
<fail>
      <step>Add an <code>alt</code> attribute to each
      <code>img</code> element.</step>
</fail>
<pass>
      <next-check number="3">
      …
</pass>
```

Finally, the test descriptions also identify the confidence provided by each test. For instance, automatic tests could identify errors that are definitely existent (called ‘Known Problem’ in the tool), errors that are ‘Likely Problems’ (such as suspicious `alt` attributes), or ones that are ‘Potential Problems’ (such as a mouse-based event handler in the JavaScript). Current the tool supports ~267 tests with these varying degrees of confidence, which are expected to be used in a manual context. That is, the tests are executed automatically but are intended to highlight key areas of the Web content for the evaluators to further examine manually.

5.1.2. IBM Rule-Based Accessibility Validation Environment (RAVEn)

- Reference: http://www.alphaworks.ibm.com/tech/raven

The IBM Rule-Based Accessibility Validation Environment (RAVEn) is a plug-in tool for the open source Eclipse[19] development platform. RAVEn is a user interface checker that supports Java AWT, Java Swing, and Eclipse Standard Widget Toolkit (SWT) programming languages as well as Web applications. It is therefore very developer-oriented, and primarily intended to be used by developers while they are creating new applications using the Eclipse IDE. While it is not an ideal tool for Web accessibility evaluators who are evaluating Web content after it has been developed, it is one of the few tools that evaluate the accessibility of client-side Web applications. It uses the Mozilla Document Object Model (DOM)[20] to render the Web content, and analyze the different elements and attributes as they are manipulated through the scripting and the interaction with the end-user. In other words, it does not only assess the static HTML markup, it can also execute the scripts and continually assess the HTML as it changes its state.

RAVEn also distinguishes between the rule-sets and the rules execution engine. The engine is actually a suite of Java classes that extend the core functionality of the Eclipse IDE using the Aspect-Oriented Programming (AOP) approach. It observes the execution of the source code and carries out the tests as indicated by the rule-sets. The rule-set files themselves are written in Jython[21], which is a scripting language based on Java and Python. While the rule-sets for Web content use the XML hierarchy to reflect the structure of the markup to be evaluated, the actual execution seems to be more procedural rather than declarative. Especially the logic and the test procedures are coded in the Jython script code. For instance, the following code is the famous check that images have `alt` attributes:

```
<Img>
      <alt raven:severity="ERROR" raven:message="Missing ALT text for
      non-textual element">
            isPlaceHolderImage(thisComponent) or
            isValidAltText(propertyValue)
      </alt>
</Img>
```

Unlike the rule-sets used by the ATRC Accessibility Checker, the rules used by RAVEn can not be easily related to each other, such as to describe a sequence for executing the tests. The rules are strictly based on the trigger and action principle, and are not aware of an evaluation process. This makes sense since RAVEn is not intended to provide a user interface and help an evaluator carry out an evaluation process, but rather to generate the error messages and warnings that are transmitted to the debugging functionality of the Eclipse IDE.

However, through the programming functionality of the Jython scripts and its interface with the Eclipse framework, the RAVEn rule-sets provide a powerful mechanism to define highly complex rules. It is essentially possible to write rule-sets that not only check the Web content but also interact with it, for instance to submit forms or to imitate other user interactions. One could record a sequence of steps, for example to carry out a transaction in a Web application, and replay this sequence for debugging purposes as a rule. The downside is however that such scripting is very application-specific and is not easily portable to other frameworks.

---

[19] http://www.eclipse.org/

[20] http://developer.mozilla.org/

[21] http://www.jython.org/

5.1.3. UC3M Web Accessibility Evaluator in a single XSLT file (WAEX)

- Reference: http://www.it.uc3m.es/vlc/waex.html

The UC3M Web Accessibility Evaluator in a single XSLT file (WAEX) is, as the name says, an XSLT file. It is therefore a strictly declarative approach based on the well-known XSLT standard. While this has significant limitations, for instance XSLT does not work with invalid HTML, there is a lot of functionality in XSLT and XQuery that this tool makes use of. For instance, it makes use of external Web-based services such as the W3C HTML Validator[22] to check the validity of the content before it is further processed. It also uses advanced XQuery expressions that identify patterns in the content that match the test requirement. For instance, the following code matches HTML links and area elements that use `javascript:` function calls as the target rather the designated `onClick` attribute:

```
<xsl:for-each select="(//xhtml:a|//xhtml:area)[starts-
with(@href,'javascript:') and not(@onclick)]">
```

Since there is no dedicated rules execution engine other than the XSLT processor, WAEX can not rely on any additional function calls other than what is provided by XSLT. For instance, it can not initiate a dialog with the evaluator to get additional input on a specific decision before it continues to carry out further tests. It is therefore strictly limited to tests that can be realized in XQuery expressions. However, it is highly portable as it makes use of the widely deployed XSLT standard. Since XSLT is supported by many different kinds of processors for different platforms, WAEX can be easily plugged into different kinds of applications.

5.1.4. WebAIM Logical Rapid Accessibility Evaluation (LRAE)

- Reference: http://eval.webaim.org/lrae/

The WebAIM Logical Rapid Accessibility Evaluation (LRAE) was specifically designed to be a counterpart for the W3C Evaluation and Report Language (EARL)[23]. The idea was to create a standardized format for rules as input for Web accessibility evaluation tools, and to receive the output of the tools in a standardized format as well. It was primarily designed for the WebAIM WAVE[24] Web accessibility evaluation tool which is available as a Web-based service, a browser toolbar, and previously also desktop application.

While developed independently, LRAE is astonishingly similar to the approach used for the ATRC Accessibility Checker. It is based on an XML format and uses XQuery expressions to match for triggers in the Web content. It also relies on function calls that are provided by the rule execution engine. Most of the logic and sequence for carrying out the individual tests is however represented directly in the rule-sets. This enables a flexibility to adapt to different types of requirements and to optimize the tool without needing to recode the application.

The framework conceived for LRAE was to evaluate formats beyond HTML such as Linux Glade, Mozilla XUL, or the Open Document Format (ODF). It therefore aims to be agnostic to any specific technology as much as possible. It is left to the rule execution engine to parse and to execute the tests on the respective technologies that are supported by the application. The rules focus on the test descriptions and procedures while the application handles the I/O, including any interaction with the human evaluator.

---

[22] http://validator.w3.org

[23] Hence the word-play with the reversing order for the letters in the acronyms 'LRAE' and 'EARL'.

[24] http://wave.webaim.org

5.2. Example Formalizations in Software Quality Assurance Tools

- Reference: Software Automation Framework Support[25]
- Reference: Abbot Java GUI Test Framework[26]
- Reference: Marathon Integrated Testing Environment[27]

Also outside the field of Web accessibility evaluation there is significant interest in testing the user interfaces. As noted at the beginning of this section, there are several tools to help model software user interfaces. For instance, to describe where the components should appear, what properties they should have, as well as other characteristics. However, there do not seem to be as many tools that help evaluate the behavior of such user interfaces. For instance, do actions such as clicking specific types of buttons generate the expected type of behavior? These types of checks do not observe the static properties of the user interface but rather how the interface reacts to simulated user interactions with the software.

From studying a variety of freely available tools for software user interface quality assurance, it seems that their primary focus is on automating the testing steps and measuring deviances rather than measuring qualitative aspects, such as the usability design. The testing scripts for many these tools therefore resemble macros and recordings rather than rule-sets that describe the behavior. In other words, the testing script is strongly dependent on the sequence in which the individual steps are executed. Each step may include directives that change the state of the application or of the user interface, which in turn changes the assumptions about the interface for the remaining steps to be executed.

This approach is generally also a variant of Keyword-driven testing. However, as opposed to the approach adopted by most of the Web accessibility evaluation tools, it does not only query the content but also interacts with it (simulates keyboard or mouse events, changes settings, or carries out other events). It is likely that the following arguments are reason for this difference between software user interface and Web quality assurance approaches:

- **Generic Requirements versus Application Requirements** – evaluating accessibility is based on a set of functional requirements that need to be met throughout the entire application. However, quality assurance at large measures whether applications have met specific requirements, such as the behavior set out by the product specification.
- **Static User Interfaces versus Dynamic User Interfaces** – initially Web pages were static HTML. Even if they were generated by server-side scripting, they were often based on fairly simple parameters and could be reconstructed. Software applications that need automated testing of the interface are however often dynamically generated.

With the advancement of Web applications and dynamically generated Web content, testing frameworks that interact with the content are becoming increasingly important for evaluating the accessibility. Sub-section 5.1.2 IBM Rule-Based Accessibility Validation Environment (RAVEn) explores one of the currently available Web accessibility evaluation tools that have such capabilities. It is therefore not surprising to see many of the currently available software user interface testing frameworks use a similar approach to RAVEn. They provide an API and use programming languages such as Java or Jython for writing the procedural rule-sets.

---

[25] http://safsdev.sourceforge.net/

[26] http://abbot.sourceforge.net/

[27] http://marathonman.sourceforge.net/

5.3. Possible Approach for a Unified Test Description Notation

From studying the different approaches for formalizing test descriptions in Web accessibility evaluation tools and in software user interface quality assurance tools it can be concluded that there are significant commonalities. In fact, the two disciplines provide complementary means for addressing different aspects of user interface testing. While Web accessibility evaluation tools traditionally approach the content from the perspective of testing qualitative properties of static components, software quality assurance tools traditionally approach the content from the perspective of finite state machines (FSM). In principle however, both approaches seem to be based on Keyword-driven testing concepts with rule-sets that are based on a collection of atomic tests, each of which consists of the following parts:

1. **Trigger** – an entity such as an element, pattern, or parameter that triggers an action.
2. **Action** – a condition that is evaluated for in the context of the specific action trigger.

For instance, consider the following examples of atomic tests and how they may be realized:

Natural Language	**Pseudo Code**
Each `img` element must have an `alt` attribute	Trigger(`img`) := Action(*check_alt_exists*(`img`))
The `alt` attribute must describe its `img` element appropriately	Trigger(`img,alt`) := Action(*check_alt_good*(`img,alt`))

Table 2: Requirement translation into computer logic

Beyond these basic tests there are also testing procedures that define the rules for combining the individual testing outcomes. For instance, the result from the first test does not constitute a result for the overall requirement. Only after carrying out the second test and combining the two individual results can claims be made. In some cases there could be a complex sequence of (branching) logic that needs to be followed in order to determine the compliance with the requirements. The above requirement would therefore be likely to be realized as follows:

```
Test^1(Trigger(img) := Action(check_alt_exists(img)))
Test^2(Trigger(img,alt) := Action(check_alt_good(img,alt)))
Requirement := True(Test^1) & True(Test^2)
```

While this is a simple example, it highlights some important aspects of formalizing the testing procedures and test descriptions for Web accessibility. While the function *check_alt_exists*() can be easily realized by rule execution engines, the function *check_alt_good*() needs human intervention. Where evaluation tools can not interact with the human evaluators, for instance because they are fully automated, the result ‘`cannotTell`’ should be assumed. This is a first hurdle already since the logic for combining the atomic tests now needs to consider multiple outputs rather than simple Boolean ‘`True`’ or ‘`False`’ type results. However, the output can be a set of discrete possible results so that they remain computable with a reasonable amount of effort (potentially using logical truth-tables).

From the previous examples of quality assurance tools that test dynamic user interfaces it can be concluded that not only query functions are required but also functions that perform events on the content. For instance, functions such as *click_button*() or *submit_form* () are useful to help evaluate dynamically generated Web content and Web applications. These functions can be used by rule-sets that are designed to test the behavior of generic Web content (they map to WCAG requirements), or that test the behavior of specific Web applications (custom rules).

Finally, from studying the previous examples for formalizing test descriptions, it seems that most of the rules employ some form of an indicator for the level of severity for an error. For instance, while some functions such as *check_alt_exists*() identify existing errors (they have high confidence values), other functions such as *check_suspicous_alt*() are not regarded as reliably. These types of functions are however very useful and provide additional information to evaluators. Part of the test description therefore needs to declare properties of the test such as the severity or the confidence. Different Web accessibility evaluation tools could use this information differently depending on their approach for interacting with the evaluators.

In conclusion it can be suggested that the formal descriptions for Web accessibility evaluation tests consist of the following components:

- **Context** – each test needs to provide context information such as the requirement that it tests for (for instance, a WCAG 2.0 Success Criteria), the Web technologies that it addresses, its relationships to other tests, the severity of the outcome, etc.
- **Logic** – the formal grammar and the operators that can be used to express the testing procedures or the individual tests. The more operators are provided by the logic (for example ‘`if … then … else`’, ‘`XOR`’, or ‘`length()`’ etc. type operators), the more tests and test procedures can be adequately expressed using the formalization.
- **Functions** – the built-in or API functions provided by the test execution engines that support the formalization. This should be a minimal set of atomic functions that can not be realized using the logic describe above. For instance, the functions should not provide string operations if these are already provided by the formal logic.

Ideally, the main difference between the execution engines would be in the realization of the set of basic functions. For instance, which types of functions do they support and for which Web technologies? For example, a technology-independent function *image_has_equivalent*() as opposed to the HTML-centric *check_alt_exists*() could be applicable to PDF, Flash or other formats. Different execution engines would likely provide varying support for the different functions, also with regard to the performance (such as speed) for executing these functions.

Also the user interfaces (if any) provided by the rule execution engines will likely help to differentiate between the implementations. While one may be more automatic, another may specialize on providing dialogs to interact with the evaluators during evaluation. However, these tools will be flexible to interpret different sets of rules that evaluate different requirements. The tools can be easily reused in different domains such as, mobile Web, privacy, security, or usability. At the same time, they can be easily customized by developers or evaluators, for instance to debug sophisticated Web applications. Using a common format for test descriptions promotes the exchange of knowledge and adoption of the format among different types of tools, such as existing quality assurance tools.

6. Proposed Model for Web Accessibility Support Tools

The previous sections outlined a broad variety of promising technologies that can be used to aid Web accessibility evaluation tools, as well as other types of tools that are part of the Web development process, to share accessibility information. While some of these technologies are currently still under development, they are potentially essential building blocks for creating a collaborative environment for Web accessibility evaluation. In particular the W3C Evaluation and Report Language (EARL) [WAI, 9] has been specifically designed for exchanging data about evaluation between different types of tools. The question for this section is what is the added benefit in exchanging this type of information and how should this exchange look like?

In particular this section explores the concept of 'incremental evaluation' that was outlined in section 2.2.4.2. Informed Methodologies of this study. The following illustrates this concept:

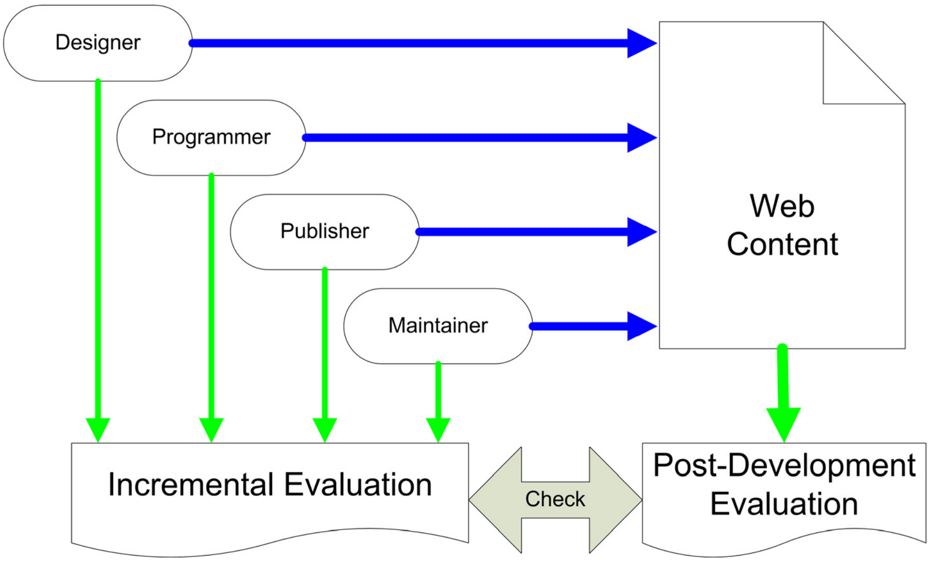


Figure 12: Incremental evaluation

During the development of Web content, the different types of developers are responsible for creating different parts of the Web content. Ideally each developer carries out the accessibility evaluation that relates to their respective roles (see section 2.2.3. Roles and Responsibilities). However, without some level of system support this information about what has been checked and by who is easily lost and not anymore retrievable at a later point in time. Conversely, if an authoring system supports accessibility information to supplement the Web content then each developer in the production chain can incrementally add small pieces of information about the accessibility of the resource. By the time the resource is operational, basic information about the level of accessibility is available. Even if this information is incomplete, the alternative is not having any information about the accessibility of resources. In fact, this is the usual case for the majority of Web sites, the information about the level of accessibility is only known at discrete intervals when comprehensive evaluations are carried out to assess the Web sites.

This does not mean that recurring post-development accessibility evaluation is not important. In fact, it is an essential quality control mechanism to help check the accuracy of the internal quality assurance mechanism. The incremental evaluation complements the post-development evaluation and provides continuous information about the level of accessibility for any given point in time. It encourages developers to think about accessibility and to declare which of the checks they have carried out. At the same time, it facilitates accurate monitoring of the status and helps track the origin of accessibility barriers.

6.1. Framework for Distributed Web Accessibility Evaluation Tools

While previous examples demonstrated the demand and feasibility of integrating different Web accessibility evaluation tools, they also underline the fundamental problem that hinders more availability and deployment of such solutions. Without a common, open, and vendor-neutral framework it is not economically feasible to realize such consistent integration, and only occasional examples will emerge. The open interface that underlies this framework can be based on currently existing technologies. The open interface can be used to connect the following types of components that integrate into a system:

- **User Applications** – are the programs that are launched by the end-users and that provide the user interface. Usually these are Web accessibility evaluation tools but they could also be authoring tools, toolbars, or data analysis tools. These tools can communicate with any of the components listed below, or they can access the Web content directly as needed.
- **Remote Services** – are programs that run remotely on a server to carry out a specific task. They may access the Web content directly and/or process information in a data repository, depending on the service that they provide. They can launch other remote services or plug-in components as needed.
- **Plug-in Components** – are extensions to any of the two components above. They do not necessary provide a user interface although they may initiate system dialogs. They may access the Web content directly and/or process information in a data repository. They may also launch other remote services or plug-in components as needed.
- **Data Repositories** – are usually not active components, for example they could be data files located in a shared space so that they can be accessed by the components. They can be located and used by the different components, for example they could be generated by a server-side script and processed by a data-analysis application.

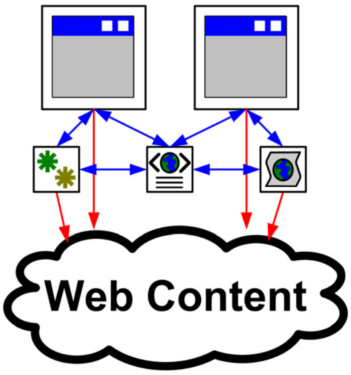


Figure 13: Framework components

Assuming there is a uniform method for these building blocks to exchange information, any of the combinations of tool integration that were discussed in the previous section could be realized. For instance, a toolbar application could use a remote service to carry out some of the automated evaluations as outlined in example 1 of the previous section. At the same time, it could export the evaluation results into a common data repository, for instance to record the status of the Web content at a specific point in time. A different tool, such as a data analysis tool, could then use this information to monitor the overall status of the Web site, including the Web content evaluated manually through the toolbar. The following sub-sections describe the different parts that provide a uniform method for tool interaction.

6.2. Common Format for Test Descriptions – Request

As discussed in section 5. Excursion: Proposed Test Description Notation Format, there to date no widely used standard for test descriptions. While such a standard would promote the exchange of rule-sets between different types of quality assurance tools, it is unclear if such a standard will become available in the foreseeable future. However, a simple approach that uses existing resources could be a starting point for immediate implementation. The increased granularity of the W3C Web Content Accessibility Guidelines (WCAG) 2.0 [W3C, 18] lends itself as a common basis for a reference point. As described in previous sections, each WCAG 2.0 Technique has a unique Universal Resource Identifier (URI). Also each WCAG 2.0 Principle, Guideline, and Success Criteria can be uniquely identified in the same manner. This means that while tools are currently not able to exchange information about the context, logic, and execution of the test procedures, they can exchange information through the identifiers of the requirements. For instance, larger quality assurance frameworks could delegate specific requirements, such as individual Techniques or entire Success Criteria, to integrated Web accessibility evaluation tools. Any of the examples for ad-hoc evaluation tool integration that were highlighted in the previous sections could be realized this way.

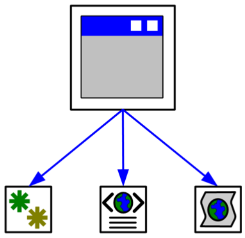


Figure 14: Delegation of test requests

The image above illustrates this interaction in which overarching applications delegate testing to plug-in tools. The applications need to identify two entities in their request to the delegates:

1. **Test Procedure** – an identifier for the test requirement or test procedure.
2. **Test Subject** – an identifier or the actual content that needs to be tested.

Both of these entities are however already defined by the vocabulary of the W3C Evaluation and Report Language (EARL) Schema 1.0 [W3C, 27], and could therefore be reused until a more comprehensive test description vocabulary becomes available. Specifically the EARL terms `TestCriterion` and `TestSubject` could be used by evaluation tools to delegate testing requests. They could also be used to query repositories for information about the test results. For instance, the following code segment could be used by applications to request that plug-in tools execute the WCAG 2.0 Technique 'G55' on a local file called 'index.html', or to query data repositories for information about previous executions of this test.

```
<earl:TestSubject rdf:about="file://index.html"/>

<earl:TestCase rdf:about="http://www.w3.org/TR/WCAG20-TECHS/G55">
      <dc:title xml:lang="en">G55: Linking to definitions</dc:title>
      <dc:description xml:lang="en">For each word, phrase, or
      abbreviation to be defined:
      1. Check that at least the first instance of the item is a link.
      2. Check that each link navigates to the definition of the item.
      </dc:description>
      <dct:isPartOf rdf:resource=
      "http://www.w3.org/TR/WCAG20/#meaning-idioms"/>
</earl:TestCase>
```

Code Listing 2: Request arguments

6.3. Common Format for Test Results – Response

As described in section 4.1 The W3C Evaluation and Report Language (EARL), a common format for exchanging test results is currently under development. It is the counterpart for the test description and could be used to collect the test results from the delegates in an integrated system of tools. Consider the example from the previous sub-section in which an overarching tool, potentially an authoring tool, delegated specific testing requirements to plug-in tools. In reaction to this, the individual plug-in tools need to communicate their findings back to the main application. The following illustration highlights the direction of communication flow:

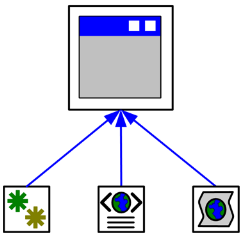


Figure 15: Collection of test results

As opposed to the scenario previous sub-section, the response flow from the individual plug-in delegates to the main application. The responses consist of complete EARL assertions that satisfy the requests made. For instance, the following code segment could be the response to the query described in code listing 2 in the previous sub-section:

```
<earl:Assertion>
      <earl:assertedBy>
            <foaf:Person rdf:about="jd">
                  <foaf:name>Jon Doe</foaf:name>
            </foaf:Person>
      </earl:assertedBy>
      <earl:subject rdf:resource="file://index.html"/>
      <earl:test>
            <earl:TestCase rdf:about=
            "http://www.w3.org/TR/WCAG20-TECHS/G55"/>
      </earl:test>
      <earl:result>
            <earl:TestResult>
                  <earl:outcome rdf:resource=
                  "http://www.w3.org/ns/earl#fail"/>
                  <dc:title xml:lang="en">Unusual phrase not
                  defined</dc:title>
                  <dc:date>2008-08-31</dc:date>
            <earl:TestResult>
      </earl:result>
      <earl:mode rdf:resource="http://www.w3.org/ns/earl#manual"/>
</earl:Assertion>
```

Code Listing 3: Response output

In this specific case, the response from the delegate is that a person called ‘Jon Doe’ asserted on the ‘31 August, 2008’ that the content in file ‘index.html’ fails the test procedure of ‘G55’, because there was an ‘unusual phrase that was not defined’. This information could have been generated upon request, or it may have been previously recorded in a data repository.

6.4. Common Format for Communication – Protocol

The W3C SPARQL Protocol and RDF Query Language (SPARQL) [W3C, 33-35] provides a protocol for exchanging queries between different entities as well as defines a query language for RDF. The SPARQL protocol uses the W3C Web Services Description Language (WSDL) Version 2.0 [W3C, 36]. It defines HTTP bindings and so allows different entities to exchange RDF queries using the well-known HTTP Protocol. For instance, it allows the following type of requests between different types of tools:

```
GET /sparql/?query=[SPARQL Query Expression] HTTP/1.1
```

Code Listing 4: SPARQL Request

The response is also based on the HTTP Protocol and reuses the status codes, error messages, as well as the other parts of the specification. For instance, the request from above could have the following response:

```
HTTP/1.1 200 OK
Date: Fri, 06 May 2005 20:55:12 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 DAV/1.0.3
Connection: close
Content-Type: application/sparql-results+xml

<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
      [SPARQL Query Result]
</sparql>
```

Code Listing 5: SPARQL Response

This is a useful mechanism for interacting with remote services. However, SPARQL queries can also be exchanged directly without the HTTP/WSDL wrapper. SPARQL query language is similar to existing database query languages such as SQL. It is therefore easy to learn and understand by many developers, even though they may not have prior experience with RDF. The following code is an example SPARQL query in which one tool requests to receive all the EARL assertions about the file called ‘index.html’:

```
PREFIX earl: <http://www.w3.org/ns/earl>
SELECT ?assertion
WHERE { ?assertion earl:subject "file://C:\index.html" }
```

Code Listing 6: SPARQL Query

If the request is sent to a data repository then it will return all the information it has about this resource. Otherwise, if the request is sent to an evaluation tool, then the tool may carry out the tests that it is capable of executing, and return the results. The output of such a SPRQL query could be in the RDF/XML notation, such as that of code listing 3 in the previous sub-section. However, SPARQL also provides an XML format for the query results [W3C, 35] so that it can be more easily processed by XML-based applications.

In summary, SPARQL provides a complete protocol and query language that can be reused by Web accessibility evaluation tools to exchange EARL assertions. SPARQL also cascades the details of RDF from developers who may not have experience with it. It provides a query language that is similar to SQL and provides the results in XML format. Both of these formats are well-known and widely used by many developers and implementations.

7. Evaluation of the Proposed Tools-Supported Model

The previous section provides and in-depth discussion about the current approach employed by Web accessibility evaluation tools, the need for integrating different types of tools, as well as a potential approach for a distributed model that supports a collaborative environment. The discussion proposes a vendor-neutral format and protocol for different types of tools that are part of the development process, to exchange accessibility information about resources. This can be used to support incremental evaluation that is paired with post-development evaluation to provide comprehensive quality management with respect to accessibility evaluation.

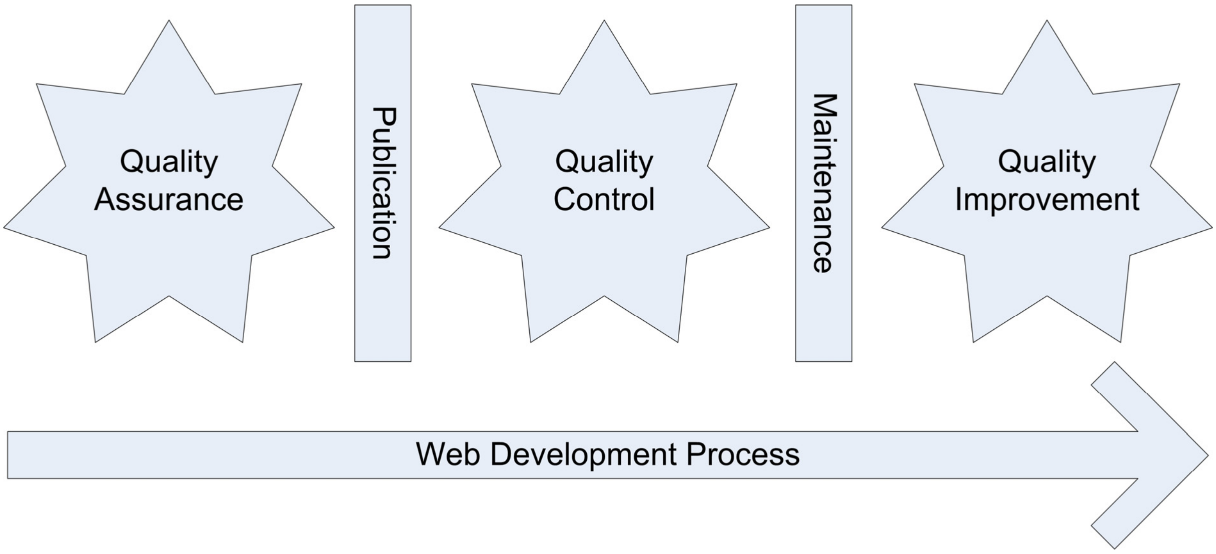


Figure 16: Quality management in the Web development process

The image above depicts some of the quality management practices that take place during the development of Web content. It includes:

- **Quality Assurance** – organization of the internal production and processes to ensure that the Web content being developed meets certain criteria such as accessibility.
- **Quality Control** – testing that takes place after the production to validate the level of quality, such as the accessibility, of the published Web content.
- **Quality Improvement** – testing that takes place after the production to assess possible opportunities to further improve the quality of the published Web content.

Note however that the term ‘quality assurance’ is commonly used to refer to all these aspects of quality management. This is likely due to the fact that quality control and improvement is not frequently practiced in Web development. Once the Web content is published it quickly becomes legacy and less interesting to developers in many Web development situations.

Web accessibility evaluation is however slightly different and currently seems to focus much more on the quality control aspect than on the other areas of quality management. This may be due to the introduction of policies and legislation, and the reaction of Web site owners and developers. They were confronted with the need to assess the compliance of the existing Web content to legal requirements and created a market for such solutions. However, the previous sections showed that there are technologies available to help broaden this focus towards the other areas of quality management. This section analyzes the proposed distributed model and discusses some of the benefits as well as challenges for this approach.

7.1. Potential Impact on Web Accessibility Evaluation

Employing a structured format for describing accessibility checks, evaluation results, as well as a protocol to exchange this information provides many opportunities for Web accessibility evaluation and tools that support such evaluation. The foremost impact is on the integration of different types of tools, such as the scenarios discussed in section 4.2. A Need for Integrating Web Accessibility Evaluation Tools. It includes the integration of evaluation tools within:

- **Authoring tools** – different types of Web accessibility evaluation tools, including ones with broad or narrow evaluation focus and ones that could be automatic or that need manual interaction, could integrate seamlessly with different types of authoring tools including content management systems, code editors, or WYSIWYG editors.
- **Evaluation tools** – primarily manual Web accessibility evaluation tools or other evaluation tools that already provide mature interfaces (such as enterprise evaluation tools) could integrate with primarily automated evaluation tools that work in the background, or evaluation tools that provide specific functionality or heuristics.
- **Data-Analysis tools** – new bug-tracking and performance monitoring tools could be specifically developed on the basis that the evaluation data could be provided by the evaluation tools in a uniform structure.

While this has a direct impact on some of the currently existing tools, it is equally an indirect opportunity for the development of new types of tools. For instance, the smaller and focused evaluation tools could save significant development costs for the interface and other types of I/O-infrastructure as outlined in [Englefield et al. 2005]. This could encourage development of a larger variety of plug-in tools. Since many of these types of tools are typically developed in the research sector and therefore often provide sophisticated algorithms and heuristics, the increase of such tools could quickly stimulate the overall production of evaluation tools.

Especially now that Web accessibility evaluation has entered into a new generation of content that is dynamic and highly interactive new approaches need to be developed. For instance, the evaluation of Web applications requires consideration for the runtime and the code-execution which are new concepts of many Web accessibility evaluation tools. Only few of these tools currently support the W3C Accessibility Rich Internet Applications (WAI-ARIA) [W3C, 26], and probably need considerable amount of effort to be re-engineered to support applications. The possibility of integrating focused tools that address dynamic content and other types of Web applications could therefore be a more viable approach under some circumstances.

Besides the opportunities for tools and tool vendors there is potentially also significant impact on the overall practice of Web accessibility evaluation. Especially the direct integration of the authoring and evaluation tools brings numerous benefits to the developers involved during the production of the Web content. When developers can evaluate 'in-place' rather than needing to switch back and forth between different tools, there is a direct increase of their efficiency as well as the efficiency of the overall development process. The developers can also spend the time and effort in implementing optimal solutions from the early development stages because their authoring tools support them in doing so, rather than spending this same time and effort in finding less optimal workaround solutions after the content has been published.

Finally it should also be noted that common formats for exchanging accessibility information can also be used by tool vendors to calibrate their tools against test suites such as the WCAG 2.0 Test Samples [WAI, 24]. The output of the tools could be compared to a normalized set of results to ensure correct implementation of the tools. Similarly, evaluation tools could also be directly compared to each other using such a uniform data exchange mechanism.

7.2. Relationship to Generic Web Quality Assurance

Web accessibility evaluation has had strong spotlight which induced the production of many types of tools and solutions. However, the evaluation practices and tools could easily be used to address many other aspects of Web quality assurance that are quickly gaining traction:

- **Mobile Web** – there is a substantial overlap between the accessibility needs of people with disabilities and designing Web sites for mobile devices [WAI, 11]. In fact, many of the checks can be directly re-used to help evaluate the criteria for both requirements. Many evaluation tool developers have therefore already started to support the criteria of the W3C Mobile Web Best Practices [W3C, 45].
- **Privacy and Security** – while privacy and security were predominantly aspects of the Web infrastructure, with the rapid advancement of Web applications and scripting the issue has moved more directly into the Web content arena. Web developers need to be much more aware of the potential issues and related solutions than they needed to be for the creation of more static Web sites and content.
- **General Usability** – also general usability seems to be increasingly an important part of the Web development processes. This may be due to the rapid uptake of the Web for commercial purposes, and the strong competition that has evolved in a relatively short period. Web sites therefore need to provide a better experience for the end-users in order to convey an impression of high quality and build trust.

The existing solutions and practices in Web accessibility evaluation, including the proposed distributed model for evaluation tools, could be reused to advance generic quality assurance. Conversely, the deployment of such practices and solutions into other areas of development could significantly benefit the mainstreaming of Web accessibility.

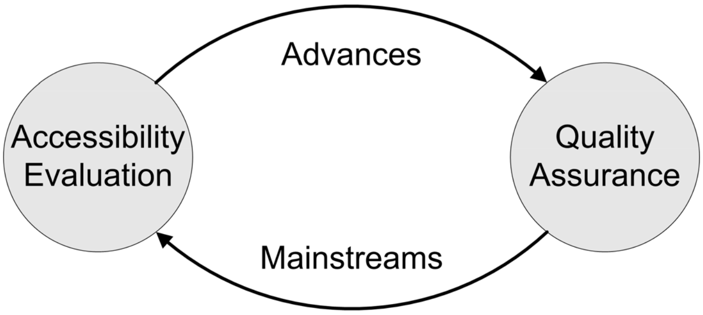


Figure 17: Mutual benefits of accessibility evaluation and quality assurance

Web accessibility evaluation encompasses several disciplines and has over the years bridged between the different types of testing approaches (see section 2.2.2. Testing Approaches) that are involved in comprehensive evaluation processes. Each of the other areas mentioned above seem to however focus on specific types of testing approaches and could benefit from reusing the existing models in Web accessibility. For instance:

- Mobile Web testing has been, so far, focusing primarily on the automatically testable criteria and could benefit from approaches that include manual and user testing.
- Privacy and security testing is primarily addressed by experts who carry out manual inspections of the source code, some of which could be formalized for non-experts.
- Usability testing is also primarily address by experts, possibly involving end-users; some basic usability tests could however be addressed by non-experts or by tools.

7.3. Practical Considerations for the Implementation

While the adoption of a distributed model of Web accessibility evaluation tools brings many advantages to Web accessibility evaluation and generic Web quality assurance, there are also many challenges that may prevent the rapid deployment of such solutions in the near future. Most importantly is that Web quality assurance, including accessibility evaluation, seems to be generally less funded. For instance, the majority of the Web accessibility evaluation tools have been developed by academia and research, charity organizations, or small and medium-sized enterprises (SME). Besides this reflection in the evaluation tools market, the numerous surveys on the conformance with Web accessibility standards or even basic HTML validation lead to believe the Web quality assurance is currently not the priority for many organizations and Web site owners. It is unlikely that this situation will change rapidly in the near future.

Due to this smaller market share of Web accessibility evaluation, it can be assumed that there is less margin and opportunities for evaluation tool vendors to venture into new approaches. There is little time and resources to experiment with new technologies and or to substantially change existing models. Also, despite the rapid maturing of Semantic Web technologies and the increasing availability of applications, modules, and education resources, in general they are not as well supported by mainstream development environments as other technologies are. For instance, it seems much easier to learn about XML and deploy it in existing applications than it is to understand and correctly implement RDF. The architecture needs to change from hierarchical to graph concept models, which sometimes brings substantial changes. Especially when the true benefits of the Semantic Web technologies are used, for example reification, the gaps and issues related to the infancy of the Semantic Web start to consume vital resources.

Fortunately the transition to the distributed model for Web accessibility evaluation could take place in a series of small steps rather than in a single large switch-over, and therefore lower the risk and potential drawbacks for tool vendors. Many of the automated Web accessibility evaluation already provide functionality to export the evaluation results in structured formats, most commonly in XML format. Several evaluation tool developers are also directly or less directly involved in the development of the W3C Evaluation and Report Language (EARL) [WAI, 9]. While EARL is an RDF syntax, it explicitly supports an XML serialization to help facilitate the adoption by developers of automated Web accessibility evaluation tools. This is a first step that should be feasible without substantial changes to existing tool functionality.

Once automated evaluation tools start to provide a consistent output format, it is likely that manual evaluation tools that already interact with automated tools may make use of this new format in order to also interact with other tools. For instance, toolbar applications that have been designed to work with specific online evaluation services could equally work with any other online service. A uniform output format may also be a crucial incentive for authoring tools to adopt it, and to include the evaluation output of automated evaluation tools into the testing or debugging functionality. It would be an important proof of concept that integration of different types of tools can take place without significant adaptation of current approaches. Finally, it would also set the stage for quality management tools, such as data analysis and reporting tools, and would encourage the development of many new types of tools.

The difficulty will however be the integration of manual tools as plug-in components in the distributed model. Without a broadly recognized and widely adopted standards to formalize test descriptions and to formalize a protocol for exchanging evaluation data, it will remain difficult to integrate such tools. It is essential to develop these standards as well as a vendor-neutral and platform-independent API that complement EARL in order to realize the model.

7.4. Opportunities for Research and Development

While tool-supported Web accessibility evaluation has evolved and matured over the previous years, there are still significant opportunities for research and development to further improve the level of tools support in many areas. For instance, similar to the software development life cycle [ISO 12207], also the stages, processes, and actors in the Web development life cycle need to be well-defined and understood. As was described in the introduction for section 2.2. Processes for Accessibility Evaluation, it can be expected that there exist several similarities between the two disciplines but that there are severe differences in some areas. For instance in the evolution and maintenance of Web content after it has been produced, and how this differs from software production that happens in more discrete releases of static products.

Also on the testing level it seems that there are many similarities and approaches that can be reused from traditional software testing, yet more research is required to better understand how these approaches can be adapted for the context of Web content [Mendes and Mosely 2004]. Especially with the advancement of Web applications and the growing complexity of Web content, the traditional ‘black-box’, ‘white-box’, or ‘grey-box’ testing approaches need to be repurposed. Also the, for software testing well defined, terms of ‘component’, ‘module’, or ‘regression’ testing seem to be useful approaches that could support incremental evaluation. However, these terms are not commonly used in the context of Web quality assurance and do not seem to have a broadly accepted definition in that context [Ash 2003].

Besides the opportunities for research in the field of Web quality assurance, the proposed use of Semantic Web technologies to exchange accessibility information between different types of tools also relates to the concepts of content adaptation. Different approaches have been undertaken to realize systems that allow content, including Web content, to be transformed according to users preferences and accessibility needs. For instance, the IMS Global Learning Consortium [IMS] is pursuing the development of learning environments in which learning objects can be matched to user profiles, so that learners can receive modules that meet their accessibility needs. Also the Dublin Core Metadata Initiative [DCMI] is pursuing this notion of content adaptation although on a broader level. However, one of the main issues remains the actual production of the accessibility metadata about the resources with which the proxies or content adaptation systems can work. Tool-supported Web accessibility evaluation could help provide this information in Semantic Web formats which are especially apt for reusing for other purposes or in other contexts than was anticipated by the initial author.

The potential approaches discussed in this study also provide numerous opportunities for the development of prototype applications that demonstrate the feasibility. For instance, it should be fairly simple to build an application that collects and summarizes data from any evaluation tool that provides the results in EARL format. This would be an immediate benefit for the tool developers and an incentive for them to provide support for EARL. Other examples include support for EARL in toolbar applications, development environments, or in online evaluation services. In many cases these developments would be simple extensions that may not have a significant impact on the core functionality of any of these applications. In fact, for some of the open source applications such as Eclipse[28], Hera[29], or the W3C Validator[30], these types of extensions could be provided by the community or research projects.

---

[28] http://www.eclipse.org/

[29] http://www.sidar.org/hera/

[30] http://validator.w3.org/

8. Summary and Conclusions

Web accessibility evaluation encompasses automatic, manual, and user testing approaches that complement each other. Technical standards for Web accessibility define the criteria for many of these checks, especially the automatic and manually testable requirements. In turn, several Web accessibility evaluation tools have been developed that address these criteria and help developers in evaluating them. Over the years a broad variety of evaluation tools have been developed. Each of these tools demonstrates different evaluation functionality and user interface characteristics without clear categories between them. The evaluation tools have been developed based on slightly different use-cases that reflect the preferences of evaluators and others users of the tools. For instance, some have been based on wizard approaches that are helpful for some developers but equally described as too verbose by other developers. Some have a Web-based interface, others are desktop applications, and others are plug-ins for browsers or content management systems.

Despite this broad variety of Web accessibility evaluation tools, it seems that most of them focus on post-development evaluation of the Web content by more experienced developers rather than on evaluation throughout the entire development process. Especially the support during early design stages and during the actually implementation stages of the development life cycle seem to be largely unaddressed with the exception of few undertakings. While it is possible to use some of the existing tools during these early stages, it seems that accessibility evaluation would be better served with tools that are specifically designed to address these purposes. For instance, toolbar evaluation tools that plug into browsers have become popular among many evaluators. However, in order to use these during the development process, the developers have to preview the Web content in a browser and evaluate it separately from the normal development environment, rather than have this evaluation functionality directly from within the WSYIWYG interface of the authoring tools.

Moreover, Web content is often created by non-technical developers who do not know much about Web technologies and the accessibility requirements. They are end-users of the content management systems and are often unaware of the possibilities of installing additional tools that help them to evaluate Web accessibility, and how to best switch back and forth between these different types of applications in order to develop accessible Web content. Evaluation is an integral part of the development process and therefore also the evaluation tools must be a natural part of the development environment, and must effectively assist the heterogeneous developer community in evaluating the accessibility of the Web content. Currently it seems that the majority of Web accessibility evaluation tools focus on more experienced evaluators rather than to reach out to many of the novice and often also non-technical evaluators.

There are several reasons why Web accessibility evaluation tools currently only address this confined group of users and seem to prefer the post-development paradigm. These include the awareness and training of the developers, which are necessary factors in creating the demand for other types of solutions. As long as developers prefer to carry out accessibility evaluation towards the end of the development process rather than during the earlier stages, the demand for other types of evaluation tools will remain minimal. However, several individual attempts seem to show that there may be sufficient demand by the developers. For instance, there exist evaluation tools that are specifically designed to work with certain authoring tools, and even one or two examples of evaluation tools that are specifically designed to work with drawing utilities that are often used by Web designers. It seems however to be uneconomical and not well scalable for evaluation tools to be specifically designed for only certain types of host applications, rather than to use open standards and APIs to plug into any application.

Providing this vendor-neutral and platform-independent interface between different types of tools that are involved in Web accessibility evaluation could open up new approaches and a new era in evaluation tools. Especially the direct integration of authoring tools and evaluation tools could significantly change the Web development processes and promote the production of accessible Web content. It could also contribute to the development of new types of tools that help monitor, report, or otherwise manage the accessibility of the Web content based on the evaluation results provided by Web accessibility evaluation tools. It would provide a new approach for tool-supported Web accessibility evaluation that takes place during the entire development process, and that addresses the needs of many different types of Web developers.

Semantic Web technologies seem particularly apt for this purpose and a promising approach for 'opening the data' and allowing the exchange of information between different types of tools. Especially the W3C Evaluation and Report Language (EARL) [WAI, 9] is an important milestone and a first step in providing a uniform format for representing evaluation results. It can be easily adopted by evaluation tools that already generate structured reports, yet enable many use-cases based on the availability of such an open format. However, EARL needs to be complemented by a test description format as well as an API to manage the actual interaction between the different types of tools. This study examined potential approaches based on many of the currently existing standards, conventions, and practices employed by Web accessibility evaluation tools. In conclusion it can be said that the technologies for realizing a distributed model of Web accessibility evaluation tools that connect and act together as a collaborative system already exist or are currently under development. These need to be deployed into the domain of Web accessibility evaluation and into the actual evaluation tools.

There are significant benefits of evaluation tool developers to pick up on these technologies and implement them early. Especially open source solutions and smaller exemplary solutions may pave the way for mainstream adoption of these technologies in accessibility evaluation. Also, while these solutions are specifically beneficial for Web accessibility evaluation, they are also beneficial for generic Web quality assurance. Especially in the domains of Mobile Web, privacy and security, and general usability many of the models and approaches that are developed in the field of Web accessibility evaluation can be reused. There are also several opportunities for research and development to help better understand the Web development life cycle and the testing practices, and to further improve the current approaches for Web quality assurance. Especially in the light of rich internet applications and the rapidly growing complexity of the Web content there needs to be a better understanding of the core processes and activities that constitute Web development.

9. References

9.1. Scientific Papers

[Abou-Zahra 2005a]
Automated Web Site Accessibility Evaluation
Abou-Zahra S.; Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites (WWV) 2005

[Abou-Zahra 2005b]
A Vendor Neutral Reporting Language for Web Accessibility Evaluation
Abou-Zahra S.; Proceedings of 20th CSUN Conference 2005

[Abou-Zahra 2005c]
Semantic Web Enabled Web Accessibility Evaluation Tools
Abou-Zahra S.; Proceedings of 2nd W4A Conference 2005

[Abou-Zahra 2005d]
A Common Vocabulary to Facilitate the Exchange of Web Accessibility Evaluation Results
Abou-Zahra S.; Proceedings of 10th Dublin Core Conference 2005

[Abou-Zahra 2006a]
Selecting and Using Web Accessibility Evaluation Tools
Abou-Zahra S.; Proceedings of 21st CSUN Conference 2006

[Abou-Zahra 2006b]
Managing and Monitoring Web Site Accessibility
Abou-Zahra S.; Proceedings of 10th ICCHP Conference 2006

[Abou-Zahra 2007]
Introducing the WCAG 2.0 Test Samples Repository
Abou-Zahra S.; Proceedings of 22nd CSUN Conference 2007

[Archer 2005]
Quatro – A Metadata Platform for Trustmarks
Archer P.; Proceedings of 10th Dublin Core Conference 2005

[Brajnik 2004a]
Comparing Accessibility Evaluation Tools: A Method for Tool Effectiveness
Brajnik G.; Universal Access in the Information Society, 2004, 3(3-4):252-263, http://sole.dimi.uniud.it/~giorgio.brajnik/papers/eval-method.pdf

[Brajnik 2004b]
Using Automatic Tools in Accessibility and Usability Assurance Processes
Brjanik G.; Proceedings of 8th ERCIM User Interfaces for All 2004, http://sole.dimi.uniud.it/~giorgio.brajnik/papers/qa_processes.pdf

[Brajnik 2006]
Web Accessibility Testing: When the Method is the Culprit
Brajnik G.; Proceedings of 10th ICCHP Conference 2006, http://sole.dimi.uniud.it/~giorgio.brajnik/papers/bw06.pdf

[Brajnik 2007]
Ranking websites through prioritized web accessibility barriers
Brajnik G.; Proceedings of 22st CSUN Conference 2007

[Brajnik et al. 2007]
Effects of Sampling Methods on Web Accessibility Evaluations
Brajnik G., Mulas A., Pitton C.; Proceedings of 9th ASSETS Conference 2007,
http://sole.dimi.uniud.it/~giorgio.brajnik/papers/sampling-assets.pdf

[Bühler et al. 2005]
A Framework for Automated Web Accessibility Assessment
Bühler C., Heck H., Perlick O., Snaprud M.; Proceedings of AAATE Conference 2005

[Bühler et al. 2007]
Combining empirical and theoretical methods
to enhance large scale web accessibility monitoring results
Bühler C., Heck H., Nietzio A., Craven J., Snaprud M.H.;
Proceedings of AAATE Conference 2007

[Bühler et al. 2008]
Monitoring Accessibility of Governmental Web Sites in Europe
Bühler C., Heck H., Nietzio A., Olsen M.G., Snaprud M. H.; Proceedings of 11th ICCHP Conference 2008

[Chevalier and Ivory 2003a]
Web Site Designs: Influences of Designer's Experience and Design Constraints
Chevalier A., Ivory M.Y.; International Journal of Human-Computer Studies, 2003

[Chevalier and Ivory 2003b]
Can novice designers apply usability criteria and recommendations to make web sites easier to use?
Chevalier A., Ivory M.Y.; International Journal of Human-Computer Studies, 2003

[Craven 2005]
Whose web is it anyway?
Craven J.; 9th Institutional Web Management Workshop (IWMW) 2005

[Craven and Nietzio 2007]
A taks-based approach to assessing the accessibility of web sites
Craven J., Nietzio A.; Performance Measurement and Metrics, 8(2):98-109

[Englefiled et al. 2005]
A Proposed Architecture for Integrating Accessibility Test Tools
Englefiled P., Paddison C., Tibbits M., Damani I.; IBM Systems Journal, 2005

[Gieber and Caloggero 2006]
Prioritizing Web Site Errors with STEP
Giber K., Caloggero R.; Proceedings of 21st CSUN Conference 2006,
http://www.csun.edu/cod/conf/2006/proceedings/2925.htm

[Gjøsæter et al. 2006]
Modelling Accessibility Constraints
Gjøsæter T., Nytun J.P., Prinz A., Snaprud M.H., Skjelten T. M.;
Proceedings of 10th ICCHP Conference 2006

[Henry et al. 2001]
Adapting the Design Process to Address More Customers in More Situations
UI Access – Henry S.L., Law C., Barnicle K.; Proceedings of UPA Conference 2001,
http://www.uiaccess.com/upa2001a.html

[Henry 2002]
Another -ability: Accessibility Primer for Usability Specialists
UI Access – Henry S.L.; Proceedings of UPA Conference 2002,
http://www.uiaccess.com/upa2002a.html

[Ivory 2001]
An Empirical Foundation for Automated Web Interface Evaluation
Ivory M.Y.; PhD Dissertation, University of California at Berkley, 2001

[Ivory et al. 2001]
Empirically Validated Web Page Design Metrics
Ivory M.Y., Sinha R.R., Hearst M.A.; Proceedings of SIGCHI Conference 2001

[Ivory and Hearst 2002]
Statistical Profiles of Highly-Rated Web Sites
Ivory M.Y., Hearst M.A.; Proceedings of SIGCHI Conference 2002

[Ivory and Chevalier 2002]
A study of automated web site evaluation tools
Ivory M.Y., Chevalier A.; University of Washington, 2002
http://ubit.ischool.washington.edu/pubs/tr02/toolstudy.pdf

[Ivory 2003b]
Characteristics of Web Site Designs: Reality vs. Recommendations
Ivory M.Y.; Proceedings of 10th HCI Conference, 2002

[Ivory et al. 2003]
Using Automated Tools to Improve Web Site Usage by Users w. Diverse Abilities
Ivory M.Y., Mankoff J., Le A.; IT & Society Journal, 2003

[Krishnamurthi 2005]
Web Verification: Perspective and Challenges
Krishnamurthi S.; Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites 2005

[Law et al. 2000]
Usability screening techniques:
evaluating for a wider range of environments, circumstances and abilities
UI Access – Law C., Branicle K., Henry S.L.; Proceedings of UPA Conference 2000,
http://www.uiaccess.com/upa2000a.html

[Nevile 2005]
Anonymous Dublin Core Profiles
for Accessible User Relationships with Resources and Services
Nevile L.; Proceedings of 10th Dublin Core Conference 2005

[Nietzio 2006]
Barrieres in (Vector-)Space:
A Clustering Approach to Web Accessibility Evaluation
Nietzio A.; Workshop on Web Accessibility and Modeling 2006

[Nietzio et al. 2007]
A framework for quality assurance
for data from large scale accessibility evaluations
Nietzio A., Ulltveit-Moe N., Gjøsæter T., Olsen M.G., Snaprud M.H.;
Proceedings of HCI Conference 2007

[Pickin et al. 2001]
A UML-integrated test description language for component testing
Pickin S., Jard C., Heuillard T., Jezequel J.M., Desfray P.; UML Workshop, 2001

[Rioux 2005]
Improving the Quality of Web-based Enterprise Application with Extended Static Checking: A Case Study
Rioux F., Chalin P.; Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites 2005

[Robles et al. 2005]
Promoting Accessibility by Using Metadata in the Framework of a Semantic-Web Driven CMS
Robles R.G., Diaz del Rio F., Civit A., Prieto J.A.;
Proceedings of 10th Dublin Core Conference 2005

[Rodriguez et al. 2005]
Testing web applications in practice
Rodriguez J.J.G., Cuaresma M.J.E., Risoto M.M., Valderrama J.T.; Proceedings of 1st Int'l Workshop on Automated Specification and Verification of Web Sites 2005

[Sinha et al. 2001]
An Empirical Analysis of Criteria for Award-Winning Websites
Sinha R., Hearst M., Ivory M.Y.; Proceedings of Human-Factors for the Web, 2001

[Stone 2005]
Validating Scripted Web-Pages
Stone R.G.; Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites 2005

[Vigo et al. 2007]
Quantitative Metrics for Measuring Web Accessibility
Vigo M., Arrue M., Brajnik G., Lomuscio R., Abascal J.;
Proceedings of 4th W4A Conference 2007

9.2. Printed Publications

[Alpuente et al. 2005]
Proceedings of 1st International Workshop on
Automated Specification and Verification of Web Sites 2005
Alpuente M., Escobar S., Falaschi M. (eds.); Politechnic University of Valencia

[Ash 2003]
The Web Testing Companion
Ash L.; ISBN 0471430218, Wiley Publishing 2003

[Berners-Lee 1999]
Weaving the Web
Berners-Lee T.; ISBN 0062515861, Harper-Collins 1999

[Black 1999]
Managing the Testing Process
Black R.; ISBN 073560584, Microsoft Press 1999

[Harper and Yesilada 2008]
Web Accessibility: A Foundation for Research
Harper S., Yesilada Y. (eds.); ISBN 9781848000490, Springer 2008

[Henry 2007]
Just Ask: Integrating Accessibility Throughout Design
Henry S.L.; ISBN 9781430319528, Friends of Ed (Apress) 2007
http://www.uiaccess.com/accessucd/

[Ivory 2003a]
Automated Web Site Evaluation
Ivory M. Y.; ISBN 1402016727, Kluwer Acadamic Publishers 2003

[Kan 2004]
Metrics and Models in Software Quality Assurance
Kan S.H.; ISBN 0201729156, Addison-Wesley 2004

[Mendes and Mosely 2004]
Web Engineering
Mendes E., Mosely N. (eds.); ISBN 9783540281962, Springer-Verlag

[Mendez 2005]
Proceedings of 10th International Conference on Dublin Core
Mendez E.R. (ed.); ISBN 8489315442, Universidad Carlos III De Madrid

[Miesenberger et al. 2004]
Proceedings of 9th International Conference on
Computers Helping People with Special Needs 2004
Miesenberger K., Klaus J., Zagler W., Burger D. (eds.); ISBN 3540223347,
Springer Lecture Notes in Computer Science (LNCS) 2004

[Miesenberger et al. 2006]
Proceedings of 10th International Conference on
Computers Helping People with Special Needs 2006
Miesenberger K., Klaus J., Zagler W., Karshmer A. (eds.); ISBN 3540360204,
Springer Lecture Notes in Computer Science (LNCS) 2006

[Miesenberger et al. 2008]
Proceedings of 11th International Conference on
Computers Helping People with Special Needs 2008
Miesenberger K., Klaus J., Zagler W., Karshmer A. (eds.); ISBN 9783540705390,
Springer Lecture Notes in Computer Science (LNCS) 2008

[Mueller 2003]
Accessibility for Everybody:
Understanding the Section 508 Accessibility Requirements
Mueller J.P.; ISBN 1590590864, Apress 2003

[Nielsen 1993]
Usability Engineering
Nielsen J.; ISBN 9780125184069, Academic Press, 1993

[Paciello 2000]
Web Accessibility for People with Disabilities
Paciello M.G.; ISBN 1929629087, CMP Books, 2000

[Slatin and Rush 2002]
Maximum Accessibility: Making Your Web Site More Usable for Everyone
Slatin J., Rush S.; ISBN 0201774224, Addison-Wesley, 2002

[Stary and Stephanidis 2004]
Proceedings of 8th International Workshop of User Interfaces for All: User-Centered Interaction Paradigms for Universal Access in the Information Society
Stary C., Stephanidis C. (eds.); ISBN 354023375X, Springer LNCS 2004

[Thatcher et al. 2002]
Constructing Accessible Web Sites
Thatcher J., Bohman P., Burks M., Henry S.L., Regan B., Swierenga S., Urban M., Waddell C.D.; ISBN 1590591488, glasshaus (Apress) 2002

[Thatcher et al. 2006]
Web Accessibility: Web Standards and Regulatory Compliance
Thatcher J., Burks M.R., Heilmann C., Henry S.L., Kirkpatrick A., Lauke P.H., Lawson B., Regan B., Rutter R., Urban M., Waddell C.D.;
ISBN 9781590596388, Friends of Ed (Apress) 2006

9.3. Online Publications

[UI Access, 1]
Web Accessibility Evaluation Tools Need People
UI Access – Henry S.L.; http://www.uiaccess.com/evaltools.html, August 2002

[UI Access, 2]
About Web Accessibility
UI Access – Henry S.L.; http://www.uiaccess.com/accessibility.html

[UWEM]
Unified Web Evaluation Methodology (UWEM)
WAB Cluster; http://www.wabcluster.org/uwem1_2/

[WAI, 1]
Introduction to Web Accessibility
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/intro/accessibility.php

[WAI, 2]
Essential Components of Web Accessibility
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/intro/components.php

[WAI, 3]
How People with Disabilities Use the Web – Draft, 5 May 2005
W3C Education and Outreach Working Group (EOWG) – Brewer J. (ed.);
http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/

[WAI, 4]
Developing a Web Accessibility Business Case for Your Organization
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/bcase/

[WAI, 5]
Web Content Accessibility Guidelines (WCAG) Overview
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed);
http://www.w3.org/WAI/intro/wcag.php

[WAI, 6]
Authoring Tool Accessibility Guidelines (ATAG) Overview
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/intro/atag.php

[WAI, 7]
User Agent Accessibility Guidelines (UAAG) Overview
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/intro/uaag.php

[WAI, 8]
Accessible Rich Internet Applications (WAI-ARIA) Overview
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/intro/aria.php

[WAI, 9]
Evaluation and Report Language (EARL) Overview
W3C Education and Outreach Working Group (EOWG) – Abou-Zahra S., Henry S.L. (eds.); http://www.w3.org/WAI/intro/earl.php

[WAI, 10]
Accessibility Improvements in HTML 4.0
W3C Education and Outreach Working Group (EOWG) – Jacobs I., Brewer J., Dardailler D. (eds.); http://www.w3.org/WAI/References/HTML4-access

[WAI, 11]
Web Content Accessibility and Mobile Web: Making a Web Site Accessible Both for People with Disabilities and for Mobile Devices
W3C Education and Outreach Working Group (EOWG) and W3C Mobile Web Best Practices Working Group (BPWG) – Thorp J., Henry S.L. et al (eds.);
http://www.w3.org/WAI/mobile/

[WAI, 12]
WAI: Ageing Education and Harmonization (IST 035015)
W3C Web Accessibility Initiative (WAI); http://www.w3.org/WAI/WAI-AGE/

[WAI, 13]
Policies Relating to Web Accessibility
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed.);
http://www.w3.org/WAI/Policy/

[WAI, 14]
Why Standards Harmonization is Essential to Web Accessibility
W3C Education and Outreach Working Group (EOWG) – Brewer J. (ed.);
http://www.w3.org/WAI/Policy/harmon.html

[WAI, 15]
How WAI Develops Accessibility Guidelines through the W3C Process: Milestones and Opportunities to Contribute
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed);
http://www.w3.org/WAI/intro/w3c-process.php

[WAI, 16]
Comparison of WCAG 1.0 Checkpoints to WCAG 2.0, in Numerical Order
W3C Education and Outreach Working Group (EOWG) and W3C Web Content Accessibility Guidelines Working Group (WCAG WG);
http://www.w3.org/WAI/WCAG20/from10/comparison/

[WAI, 17]
Preliminary Review of Web Sites for Accessibility
W3C Education and Outreach Working Group (EOWG) – Abou-Zahra S. (ed);
http://www.w3.org/WAI/eval/preliminary.html

[WAI, 18]
Conformance Evaluation of Web Sites for Accessibility
W3C Education and Outreach Working Group (EOWG) – Abou-Zahra S. (ed);
http://www.w3.org/WAI/eval/conformance.html

[WAI, 19]
Involving Users in Web Accessibility Evaluation
W3C Education and Outreach Working Group (EOWG) – Henry S.L. (ed);
http://www.w3.org/WAI/eval/users.html

[WAI, 20]
Evaluation Approaches for Specific Contexts
W3C Education and Outreach Working Group (EOWG) – Abou-Zahra S. (ed);
http://www.w3.org/WAI/eval/considerations.html

[WAI, 21]
Using Combined Expertise to Evaluate Web Accessibility
W3C Education and Outreach Working Group (EOWG) – Brewer J. (ed);
http://www.w3.org/WAI/eval/reviewteams.html

[WAI, 22]
Selecting Web Accessibility Evaluation Tools
W3C Education and Outreach Working Group (EOWG) – Abou-Zahra S. (ed);
http://www.w3.org/WAI/eval/selectingtools.html

[WAI, 23]
Web Accessibility Evaluation Tools
W3C Education and Outreach Working Group (EOWG) and W3C Evaluation and Repair Tools Working Group (ERT WG) – Abou-Zahra S. (ed);
http://www.w3.org/WAI/ER/tools/

[WAI, 24]
WCAG 2.0 Test Samples
W3C Evaluation and Repair Tools Working Group (ERT WG) and W3C Web Content Accessibility Guidelines Working Group (WCAG WG);
http://www.w3.org/WAI/ER/tests/

[WebAIM, 1]
Accessibility Evaluation Tools
Article by WebAIM; http://www.webaim.org/articles/tools/

[WebAIM, 2]
A Review of Free, Online Accessibility Tools
Article by WebAIM; http://www.webaim.org/articles/freetools/

[WebAIM, 3]
Using the AIS Web Accessibility Toolbar
Article by WebAIM, Steve Faulkner, August 2005;
http://www.webaim.org/articles/ais/

[WebAIM, 4]
Evaluating Web Sites for Accessibility with the Firefox Web Developer Toolbar
Article by WebAIM, Patrick H. Lauke, September 2005;
http://www.webaim.org/articles/evaluatingwithfirefox/

[WebAIM, 5]
Toward User-Centered, Scenario-Based Planning and Evaluation Tools
Article by WebAIM, July 2004; http://www.webaim.org/articles/scenarios/

[WebAIM, 6]
Planning, Evaluation, Repair, and Maintenance
Article by WebAIM; http://www.webaim.org/articles/process/

9.4. Standards

[ECMAScript]
ECMAScript Language Specification
Standard ECMA-262 – 3rd Edition, December 1999;
http://www.ecma-international.org/publications/standards/Ecma-262.htm

[ISO 9241-171]
Ergonomics of Human-System Interaction – Part 171:
Guidance of Software Accessibility
TC 159/SC 4; ISO 9241-171:2008

[ISO 12207]
Software Life Cycle Processes
JTC 1/SC 7; ISO 12207:2008

[ISO 26300]
Information Technology – Open Document Format for Office Applications
JTC 1/SC 34; ISO 26300:2006

[ISO 32000]
Document Management – Portable Document Format – Part 1 : PDF 1.7
ISO TC 171/SC 2; ISO 32000-1:2008

[W3C, 1]
Accessibility Features of CSS
W3C Protocols and Formats Working Group (PFWG) – Jacobs I., Brewer J. (eds.);
http://www.w3.org/TR/CSS-access

[W3C, 2]
Accessibility Features of SMIL
W3C Protocols and Formats Working Group (PFWG) – Koivunen M.R., Jacobs I., (eds.); http://www.w3.org/TR/SMIL-access/

[W3C, 3]
Accessibility Features of SVG
W3C Protocols and Formats Working Group (PFWG) – McCathieNevile C., Koivunen M.R. (eds.); http://www.w3.org/TR/SVG-access/

[W3C, 4]
Web Accessibility for Older Users: A Literature Review
W3C Education and Outreach Working Group (EOWG) – Arch A.; http://www.w3.org/TR/wai-age-literature/

[W3C, 5]
Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10/

[W3C, 6]
Techniques for Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10-TECHS/

[W3C, 7]
Core Techniques for Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10-CORE-TECHS/

[W3C, 8]
HTML Techniques for Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10-HTML-TECHS/

[W3C, 9]
CSS Techniques for Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10-CSS-TECHS/

[W3C, 10]
Checklist of Checkpoints for Web Content Accessibility Guidelines 1.0
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Chisholm W., Vanderheiden G., Jacobs I. (eds.); http://www.w3.org/TR/WCAG10/full-checklist

[W3C, 11]
Techniques for Accessibility Evaluation and Repair Tools – Draft, 26 April 2000
W3C Evaluation and Repair Tools Working Group (ERT WG) – Ridpath C., Chisholm W. (eds.); http://www.w3.org/TR/AERT

[W3C, 12]
Authoring Tool Accessibility Guidelines 1.0
W3C Authoring Tool Accessibility Guidelines Working Group (AUWG) – Treviranus J., McCathieNevile C., Jacobs I., Richards J. (eds.); http://www.w3.org/TR/ATAG10/

[W3C, 13]
Techniques for Authoring Tool Accessibility Guidelines 1.0
W3C Authoring Tool Accessibility Guidelines Working Group (AUWG) – Treviranus J., McCathieNevile C., Richards J., Rosmaita G. (eds.);
http://www.w3.org/TR/ATAG10-TECHS/

[W3C, 14]
Checklist of Checkpoints for Authoring Tool Accessibility Guidelines 1.0
Treviranus J., McCathieNevile C., Jacobs I., Richards J. (eds.) ;
http://www.w3.org/TR/ATAG10/atag10-chktable

[W3C, 15]
User Agent Accessibility Guidelines (UAAG) 1.0
W3C User Agent Accessibility Guidelines Working Group (UAWG) – Jacobs I., Gunderson J., Hansen E. (eds.); http://www.w3.org/TR/UAAG10/

[W3C, 16]
Techniques for User Agent Accessibility Guidelines 1.0
W3C User Agent Accessibility Guidelines Working Group (UAWG) – Jacobs I., Gunderson J., Hansen E. (eds.); http://www.w3.org/TR/UAAG10-TECHS/

[W3C, 17]
Table of Checkpoints for User Agent Accessibility Guidelines 1.0
W3C User Agent Accessibility Guidelines Working Group (UAWG) – Jacobs I., Gunderson J., Hansen E. (eds.); http://www.w3.org/TR/UAAG10/uaag10-chktable

[W3C, 18]
Web Content Accessibility Guidelines 2.0 – Draft, 30 April 2008
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Caldwell B., Cooper M., Guarino Reid L., Vanderheiden G. (eds.);
http://www.w3.org/TR/WCAG20/

[W3C, 19]
How to Meet WCAG 2.0 – Draft, 30 April 2008
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Vanderheiden G., Guarino Reid L., Caldwell B., Henry S.L., (eds.);
http://www.w3.org/WAI/WCAG20/quickref/

[W3C, 20]
Understanding WCAG 2.0 – Draft, 30 April 2008
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Caldwell B., Cooper M., Guarino Reid L., Vanderheiden G. (eds.);
http://www.w3.org/TR/UNDERSTANDING-WCAG20/

[W3C, 21]
Techniques for WCAG 2.0 – Draft, 30 April 2008
W3C Web Content Accessibility Guidelines Working Group (WCAG WG) – Caldwell B., Cooper M., Guarino Reid L., Vanderheiden G. (eds.);
http://www.w3.org/TR/WCAG20-TECHS/

[W3C, 22]
Authoring Accessibility Guidelines 2.0 – Draft, 10 March 2008
W3C Authoring Tool Accessibility Guidelines Working Group (AUWG) – Treviranus J., Richards J. (eds.); http://www.w3.org/TR/ATAG20/

[W3C, 23]
Implementation Techniques for Authoring Accessibility Guidelines 2.0 – Draft, 10 March 2008
W3C Authoring Tool Accessibility Guidelines Working Group (AUWG) – Treviranus J., Richards J., Boland T. (eds.); http://www.w3.org/TR/ATAG20-TECHS/

[W3C, 24]
User Agent Accessibility Guidelines 2.0 – Draft, 12 March 2008
W3C User Agent Accessibility Guidelines Working Group (UAWG) – Allan J., Richards J. (eds.); http://www.w3.org/TR/UAAG20/

[W3C, 25]
XML Accessibility Guidelines – Draft, 3 October 2002
W3C Protocols and Formats Working Group (PFWG) – Dardailler D., Palmer S.B., McCathieNevile C. (eds.); http://www.w3.org/TR/xag

[W3C, 26]
Accessible Rich Internet Applications (WAI-ARIA) 1.0 – Draft, 4 February 2008
W3C Protocols and Formats Working Group (PFWG) – Seeman L., Coopers M., Schwerdtfeger S., Pappas L. (eds.); http://www.w3.org/TR/wai-aria/

[W3C, 27]
Evaluation and Report Language (EARL) Schema 1.0 – Draft, 23 March 2007
W3C Evaluation and Repair Tools Working Group (ERT WG) – Abou-Zahra S. (ed.); http://www.w3.org/TR/EARL10-Schema/

[W3C, 28]
Evaluation and Report Language (EARL) Guide 1.0 – Draft, 2 August 2007
W3C Evaluation and Repair Tools Working Group (ERT WG) – Velasco C.A., Koch J., Abou-Zahra S. (eds.); http://www.w3.org/TR/EARL10-Guide/

[W3C, 29]
HTTP Vocabulary in RDF – Draft, 23 March 2007
W3C Evaluation and Repair Tools Working Group (ERT WG) – Velasco C.A., Koch J., Abou-Zahra S. (eds.); http://www.w3.org/TR/HTTP-in-RDF/

[W3C, 30]
Representing Content in RDF – Draft, 27 June 2008
W3C Evaluation and Repair Tools Working Group (ERT WG) – Koch J., Velasco C.A. (eds.); http://www.w3.org/TR/Content-in-RDF/

[W3C, 31]
Pointer Methods in RDF – Draft, 23 June 2008
W3C Evaluation and Repair Tools Working Group (ERT WG) – Iglesias C. (ed.); http://www.w3.org/TR/Pointers/

[W3C, 32]
W3C Resource Description Framework (RDF)
http://www.w3.org/RDF/

[W3C, 33]
SPARQL Protocol for RDF
W3C RDF Data Access Working Group (DAWG) – Clark K.G., Feigenbaum L., Torres E. (eds.); http://www.w3.org/TR/rdf-sparql-protocol/

[W3C, 34]
SPARQL Query Language for RDF
W3C RDF Data Access Working Group (DAWG) – Prud'hommeaux E., Seaborne A. (eds.); http://www.w3.org/TR/rdf-sparql-query/

[W3C, 35]
SPARQL Query Results XML Format
W3C RDF Data Access Working Group (DAWG) – Beckett D., Broekstra J. (eds.); http://www.w3.org/TR/rdf-sparql-XMLres/

[W3C, 36]
Web Services Description Language (WSDL) Version 2.0
Chinnici R., Moreau J-J., Ryman A., Weerawarana S. (eds.); http://www.w3.org/TR/wsdl20/

[W3C, 37]
Test Metadata
W3C Quality Assurance Working Group (QA WG) – Curran P., Dubost K. (eds.); http://www.w3.org/TR/test-metadata/

[W3C, 38]
RDFa Primer: Bridging the Human and Data Webs
W3C Semantic Web Deployment Working Group (SWD WG) – Adida B., Birbeck M. (eds.); http://www.w3.org/TR/xhtml-rdfa-primer/

[W3C, 39]
RDFa in XHTML: Syntax and Processing
W3C Semantic Web Deployment Working Group (SWD WG) – Adida B., Birbeck M., McCarron S., Pemberton S. (eds.); http://www.w3.org/TR/rdfa-syntax/

[W3C, 40]
Gleaning Resource Descriptions from Dialects of Languages (GRDDL)
W3C GRDDL Working Group – Connolly D. (ed.); http://www.w3.org/TR/grddl/

[W3C, 41]
Rules Interchange Format (RIF)
W3C Rules Interchange Format Working Group (RIF); http://www.w3.org/2005/rules/wg/

[W3C, 42]
Protocol for Web Description Resources (POWDER): Description Resources – Draft, 15 August 2008
W3C POWDER Working Group – Archer P., Smith K., Perego A. (eds); http://www.w3.org/TR/powder-dr/

[W3C, 43]
Protocol for Web Description Resources (POWDER): Grouping of Resources – Draft, 15 August 2008
W3C POWDER Working Group – Archer P., Perego A., Smith K. (eds); http://www.w3.org/TR/powder-grouping/

[W3C, 44]
Protocol for Web Description Resources (POWDER): Formal Semantics – Draft, 15 August 2008
W3C POWDER Working Group – Konstantopoulos S., Archer P. (eds); http://www.w3.org/TR/powder-formal/

[W3C, 45]
Mobile Web Best Practices 1.0: Basic Guidelines
W3C Mobile Web Best Practices Working Group (BPWG) – Rabin J., McCathieNevile C. (eds.); http://www.w3.org/TR/mobile-bp/

9.5. Organizations

[ACTF]
Accessibility Tools Framework (ACTF)
Eclipse Project; http://www.eclipse.org/actf/

[BenToWeb]
Benchmarking Tools and Methods for the Web (BenToWeb)
EC-Funded Project; http://www.bentoweb.org

[DCMI]
Dublin Core Metadata Initiative
http://www.dublincore.org

[EIAO]
European Internet Accessibility Observatory
EC-Funded Project; http://www.eiao.net

[IMS]
IMS Global Learning Consortium
http://www.imsglobal.org

[ISO]
International Organization for Standardization (ISO)
http://www.iso.org

[MPEG]
Moving Picture Experts Group (MPEG)
JTC 1/SC 29/WG 11; http://www.chiariglione.org/mpeg/

[UN-CRPD]
United Nations (UN) Convention on the Rights of Persons with Disabilities
http://www.un.org/disabilities/

[W3C]
World Wide Web Consortium (W3C)
http://www.w3.org

[WAI]
W3C Web Accessibility Initiative (WAI)
http://www.w3.org/WAI/

[WAT-C]
Web Accessibility Tools Consortium (WAT-C)
http://www.wat-c.org