Developing Hera-FFX for WCAG 2.0

José L. Fuertes
Dept. LSIIS
Technical University of Madrid
Campus de Montegancedo
28660 Boadilla del Monte
Madrid. Spain
+34 91 352 25 46
jfuertes@fi.upm.es

Emmanuelle Gutiérrez
Sidar Foundation
Marqués de Mondejar, 34, 5D
28028 Madrid. Spain
+34 91 725 71 47
emmanuelle@sidar.org

Loïc Martínez
Dept. LSIIS
Technical University of Madrid
Campus de Montegancedo
28660 Boadilla del Monte
Madrid. Spain
+34 91 352 25 46
loic@fi.upm.es

ABSTRACT
WCAG 2.0 was published in December 2008. It has many differences to WCAG 1.0 as to rationale, structure and content. Two years later there are still few tools supporting WCAG 2.0, and none of them fully mirrors the WCAG 2.0 approach organized around principles, guidelines, success criteria, situations and techniques. This paper describes the on-going development of an update to the Hera-FFX Firefox extension to support WCAG 2.0. The description is focused on the challenges that we have found and our resulting decisions.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—Evaluation/methodology, input devices and strategies, user-centred design, interaction styles; K.4.2 [**Computers and Society**]: Social Issues—Handicapped persons/ special needs, assistive technologies for persons with disabilities

General Terms
Human Factors

Keywords
Web accessibility, accessibility evaluation, evaluation tools

1. INTRODUCTION
The Web Content Accessibility Guidelines (WCAG) have been the reference document on web accessibility since their publication in 1999 [8]. One key aspect of WCAG application is the evaluation of web accessibility. On this issue, the World Wide Web Consortium (W3C) has highlighted the need for human-based evaluation and tools to support the process [16].

WCAG were updated and published as WCAG 2.0 in December 2008 [7]. The new set of guidelines was developed with two main goals: technology-independence (for application to both current and future web technologies) and testability (where evaluators of web accessibility should agree on the results of evaluating the same web content). Based on these two goals, WCAG 2.0 are different from WCAG 1.0 as to rationale, structure and content.

Given these differences, evaluation tools have to be updated to WCAG 2.0 and, although this work has started, there are still few tools supporting WCAG 2.0 two years on.

This paper describes the on-going development of an update to the Hera-FFX Firefox extension [12] to support WCAG 2.0. This tool has a strong focus on fully supporting the new structure of WCAG 2.0 (organized around principles, guidelines, success criteria, situations, techniques and failures) and on fully supporting an expert-based manual evaluation of web accessibility.

The contents of the paper are as follows. Section 2 outlines related work on tools for WCAG 2.0 as a justification of the expediency of a new version of Hera-FFX. In Section 3, we explain why there was a need for tool redesign, and Section 4 outlines the resulting design decisions. Section 5 follows on with a description of an initial evaluation of the impact of the tool's use. Finally, Section 6 ends with some concluding remarks and future work.

2. RELATED WORK
Hera-FFX was designed based on most of the desirable features of web accessibility tools taken from several sources and summarized in [12]:

- *Automatic preliminary evaluation* (AE). Any tool should be able to automatically assess all (or parts of) the checkpoints that can be automated.
- *Support for manual filling* of success criteria results (MF). Once the automatic evaluation is complete, the tool should provide automated support for the evaluator to fill in the values of all the checkpoints and add comments about each checkpoint that could be used for later report generation.
- *Page presentation modification* for assisting checkpoint evaluation (PM). This modified presentation should highlight the elements that have to be inspected for a given checkpoint, and should display the key attributes of those elements.
- *Annotated code view* for assisting checkpoint evaluation (CV). The elements specified in the checkpoint should be highlighted in the HTML code.
- *Local pages evaluation* (LP). This feature is essential for web developers, as they should be able to assess the accessibility of unpublished web pages under development without having to send the code to a remote server.

- *Restricted-access pages evaluation* (RA). The tool should be able to evaluate restricted access web pages and secure pages.
- *Rendered-page evaluation* (RP). The tool should be able to evaluate the rendered version of the page, which implies that it can evaluate locally displayed styles and dynamically-generated content from scripts.
- *Report generation* (RG). The evaluators should be able to save reports based on the automatic and manual inspections in a handy format.
- *Support for training* (ST). The tool should provide detailed information about each checkpoint, including normative text and techniques to be applied for assessment. This information is very useful for novice evaluators, as well as for persons that do not regularly perform accessibility evaluations.
- *Multi-session capacity* (MS). An evaluation tool should provide some multi-session capacity, enabling the user to store current work and to load this work later to resume the assessment.
- *Flexibility* to integrate other accessibility guidelines (FL). There are other web accessibility requirements, apart from WCAG 1.0, such as the US Section 508 Standards, the Spanish UNE 139803:2004 Standard or the German "*Barrierefreie Informationstechnik-Verordnung*". Users should be able to change the guidelines and checkpoints, as well as the evaluation code to adapt the tool to other accessibility requirements.

Based on our experience of dealing with WCAG 2.0, we consider a new feature on top of the above eleven in this paper:

- *Fidelity to WCAG 2.0 structure* (WS). An evaluation tool should follow the WCAG 2.0 structure to ease the understanding of the good and bad practices in a web page according to the techniques, failures, situations and success criteria. One of the problems with evaluating web accessibility according to WCAG 1.0 was the lack of objectivity, as different experts could output different results. We believe that the W3C's "Techniques for WCAG 2.0" document could greatly improve this point if expert evaluators are able to provide detailed information about which techniques have been successfully applied and which failures have been found using an international common vocabulary. And the fact that this is a living document means that there is always an internationally-agreed and up-to-date checklist of techniques and failures available that evaluators can use to evaluate accessibility and compare the evaluation outcomes provided by others. For this reason, we believe that any tool should enable the evaluators to check both techniques and failures as defined by the W3C

An additional feature would be to analyse the quality of the results that each tool outputs automatically, that is, how many false positives and false negatives are produced. The focus of our paper, though, is on support for the manual evaluation process, and this is not such a relevant feature in this case, because a human evaluator should always be able to change or override the result output by the tool. For this reason we will omit the quality of the results from this description of related work.

We have found only four tools providing support for WCAG 2.0: AChecker [13], TAW [10], Total Validator [15] and Worldspace Fire Eyes [11]. Table 1 lists the features of these four tools.

AChecker (abbreviated as ACH in Table 1) is an open source Web accessibility evaluation tool [13]. It supports a variety of international accessibility guidelines like Section 508, Ley Stanca (Italy), WCAG 1.0 and 2.0, and BITV (Germany). There is a public web version, a PHP version for installation on your own server, and a plugin for TinyMCE.

TAW (abbreviated as TAW in Table 1) is a three-member family [10]: TAW3 WebStart, which is the online version of TAW; TAW3 with a click, which is a browser plug-in for Mozilla Firefox; and TAW3 standalone, a multiplatform software application that complements and extends the functionality of TAW WebStart. For the time being, only the online version works with WCAG 2.0, although is at the beta stage.

Total Validator (abbreviated as TVA in Table 1) is a free one-stop all-in-one validator comprising a HTML validator, an accessibility validator, a spelling validator, a broken links validator, and the ability to take screenshots with different browsers to see what your web pages really look like [15]. There is a web version, a Firefox extension, and a desktop version for purchase. The user can choose to check compliance with WCAG 1.0 or 2.0 or Section 508.

Worldspace FireEyes (abbreviated as WFE in Table 1) is a web accessibility tool that ensures that both static and dynamic content within a web portfolio are compliant with standards such as WCAG 1 (Priorities 1, 2 and 3), WCAG 2 (levels A and AA), Section 508 and contains some dynamic rules that test for WAI-ARIA compliance [11]. This tool is fully JavaScript aware and handles event-based page content. It works as a complement of the Firebug Firefox extension.

Table 1: Features of tools supporting WCAG 2.0

	ACH	**TAW**	**TVA**	**WFE**
Type	mixed	online	mixed	extension
AE	✓	✓	✓	✓
MF	✓	×	×	✓
PM	×	✓	✓	✓
CV	✓	✓	✓	✓
LP	×	×	✓	✓
RA	×	×	✓	✓
RP	×	×	×	✓
RG	✓	✓	✓	✓
ST	✓	✓	×	✓
MS	×	×	×	✓
FL	✓	×	×	×
WS	×	✓	×	×

The feature coverage of the tools listed in Table 1 is summarized below:

- *Type of tool*. There are multiple versions of Achecker and Total Validator; TAW is an online tool and FireEyes is an extension for Firefox.

- *Automatic preliminary evaluation* (AE). All of the tools incorporate automatic accessibility evaluation.
- *Support for manual filling* of success criteria results (MF). Registered users can manually fill in the results in AChecker. The status of an issue can be edited by the users in FireEyes. TAW online and Total Validator do not make provision for manual validation.
- *Page presentation modification* for assisting checkpoint evaluation (PM). All the tools except AChecker show a modified view of the page to help identify issues.
- *Annotated code view* for assisting checkpoint evaluation (CV). All the tools display the page source code for the user, annotated with marks associated with key issues.
- *Local pages evaluation* (LP). Local pages can be assessed using the Firefox plugin and desktop versions of Total Validator and FireEyes.
- *Restricted-access pages evaluation* (RA). The desktop version of Total Validator and FireEyes enable the user to evaluate pages with restricted access.
- *Rendered-page evaluation* (RP). Only FireEyes can evaluate dynamic pages that use JavaScript to update page content.
- *Report generation* (RG). All the tools generate some form of HTML accessibility report, although the quality of such reports is variable.
- *Support for training* (ST). All the tools, except Total Validator, offer some information about success criteria and the related techniques and failures. Total Validator includes links to the WCAG techniques document only.
- *Multi-session capacity* (MS). Only FireEyes offers support for saving and reloading current evaluation projects (on a server) to continue the evaluation.
- *Flexibility* to integrate other accessibility guidelines (FL). The only tool that enables the user to add and redefine the tests (both automatic and manual) to be run is AChecker (if installed on a server). The other tools just enable the user to select one guideline from a fixed set.
- *Fidelity to WCAG 2.0 structure* (WS). TAW is the only tool to use and refer to the techniques and failures as defined by the W3C, albeit in a limited manner. For instance, it does not enable the user to assign a situation to each element of a web page.

The conclusion of our analysis of related work is that none of the existing tools supporting WCAG 2.0 provide full coverage of what we consider to be the desirable features and that Hera-FFX intends to cover.

3. THE NEED FOR REDESIGN

Two years ago we presented Hera-FFX [12], an add-on for the Firefox web browser that supports semi-automatic web accessibility evaluation. This tool, based on Hera on-line [4], was able to provide guidance and help to human evaluators trying to assess the accessibility of a web page based on WCAG 1.0. Both tools have been successfully used by partners of the Sidar Foundation [14] and the Technical University of Madrid to evaluate web sites accessibility and also as a supporting technology for teaching web accessibility [3].

Hera on-line had some limitations, which led to the development of Hera-FFX. The first weak point was that Hera was unable to analyse local web pages. The second drawback was related to the evaluation of web pages that require some sort of user authentication. Like almost all other comparable tools, Hera often could not analyse these restricted access pages. The third snag was that Hera was unable to evaluate the rendered version of a web page, including locally displayed styles and dynamically-generated content from scripts.

Hera-FFX was developed to overcome the above difficulties by running an automatic preliminary evaluation of the web pages as they are browsed, as well as enabling the user to manually evaluate the accessibility of any of the pages.

One of the main features of Hera-FFX was its flexibility, as it was designed with the option of changing the guidelines used. This should have enabled Hera-FFX to adapt to the new WCAG 2.0, but Hera-FFX's flexibility was not enough to cope with the huge modification of WCAG 2.0 structure, leading us to develop a completely new version of the tool.

More specifically, the main issues that we found can be summarized as follows:

- WCAG 1.0 has only two de facto levels: guidelines and checkpoints. The WCAG 2.0 structure has a greater number of more complicated levels: principles, guidelines, success criteria and techniques and failures.
- WCAG 1.0 checkpoints were quite simple. WCAG 2.0 success criteria are rather complex. They include sufficient techniques (providing guidance and examples for meeting the guidelines using specific technologies), advisory techniques (potentially enhancing accessibility) and common failures (examples of bad practices that cause web pages to fail to meet the success criteria). In addition, techniques are grouped around situations. This way, only the techniques listed under one specific situation should be applied when that situation has been identified as applying to each element under assessment.
- WCAG 2.0's openness means that a live document is kept by the W3C on techniques and failures. Anyone can provide their own technique or failure, without this having to be approved or published by the W3C. The W3C can also update this document whenever they find new techniques or failures of general interest. In fact, this document was recently first updated by the W3C to include techniques and failures to cope with Adobe Flash among others [9]. But this openness means that any tool should be implemented to be open enough to easily integrate the new techniques and failures and keep it updated.
- New formulas are needed to cope with the way the partial results of the evaluation of the techniques and failures have to be aggregated to output the results for the success criteria. A detailed study has to be undertaken to account for the fact that the compliance of one technique means nothing in success criteria terms, whereas just one failure means that the success criteria are not met, even if some techniques are compliant.

- The human evaluator should be allowed to assign manual results to a success criterion without having to evaluate each failure and each technique on each web page element. Sometimes, an evaluator will be experienced enough to rate a success criterion without further analysis.

Accordingly, the development of the new Hera-FFX should deal with all these issues, providing all the desirable features for a web accessibility evaluation tool.

4. THE NEW HERA-FXX

WCAG 2.0 contains more detailed information: the number of levels of the structure has been increased and new concepts have been added. Figure 1 shows a comparison of the organizational levels of Hera-FFX 1 and WCAG 2 (dotted lines show possible concept equivalence).

Hera-FFX 1 did not consider the definition of principles. The first level was the guidelines (present in both WCAG 1.0 and 2.0). Checkpoints could be equivalent to success criteria, as they both refer to the baseline accessibility recommendations. Situations are another completely new concept.

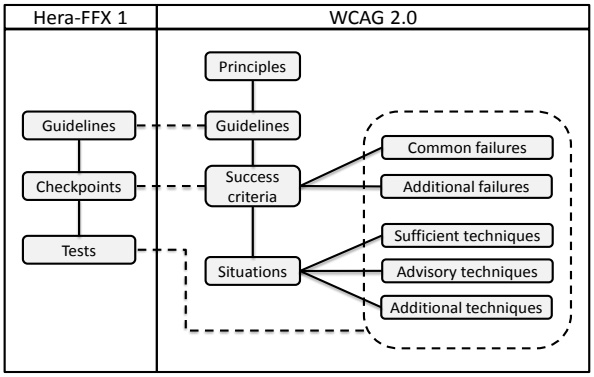

Figure 1: Concept equivalence between Hera-FFX 1 and WCAG 2.0

The tests used by Hera-FFX to automatically evaluate the checkpoints could be equivalent to WCAG 2.0's sufficient techniques and common failures, as they are able to specify algorithmic tests on certain web page elements and automatically output results. Even so, rather than providing for the independent evaluation of these tests, Hera-FFX displayed information about the tests to support decision making on the value of the respective checkpoint. This is another obstacle to the adaptation to WCAG 2.0, as the independent evaluation of techniques and failures is one of the key improvements over WCAG 1.0. In WCAG 2.0 one technique or failure can be reused for several success criteria. Taking this into account, the new version of Hera-FFX should evaluate each individual technique and failure only once, either automatically or manually. This result could then be propagated to all of the success criteria related to each technique or failure.

The other set of techniques, the advisory techniques, do not really influence web content accessibility evaluation. They are intended merely as accessibility improvements, which, on their own, do not guarantee success criteria fulfilment. For this reason we decided not to take them into account in the new tool.

The new Hera-FFX 2 solves all these problems. The core of the tool had to be redesigned to cope with the new structure and characteristics of WCAG 2.0.

In addition, the user interface also needed to be modified to be able to guide the human evaluator through the new WCAG 2.0 structure. The changes empowered users to identify situations (whenever necessary), and assign values to the failures or techniques, but also to the success criteria as a whole.

But the user interface retains the freedom of navigation that it inherited from the previous version. This freedom allows the user to evaluate the success criteria in the desired order, offering support for evaluation by priority level, by principle and guideline, or in any other order preferred by the user. Figures 2 and 3 show two different web page evaluation screens.

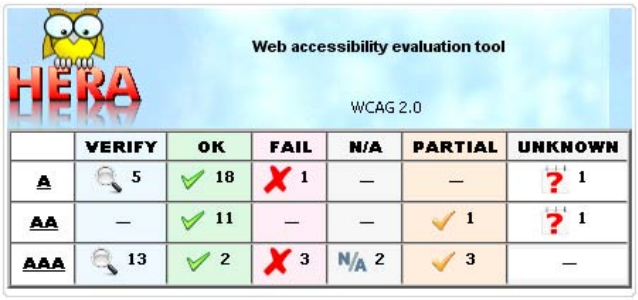

Figure 2: Summary table of the evaluation process

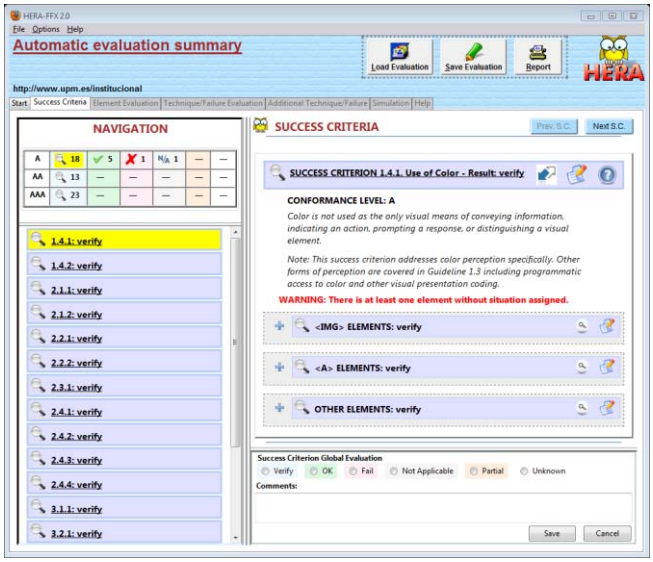

Figure 3: Evaluating a success criterion

One of the biggest issues was related to the aggregation of the individual evaluation results to output the global evaluation value.

Hera-FFX 2 allows the user to evaluate the accessibility of a web page using the following structure: success criteria, element categories, elements, situations, techniques and failures as defined by WCAG 2.0, plus additional techniques and failures defined by users. In addition, it can automatically evaluate some techniques and failures.

For any item that is evaluated (technique, failure, element, element category, success criterion, guideline, principle), either directly or by aggregation, Hera-FFX has one six-element array to represent its evaluation. This array takes the form [fail, NA, verify, ok, unknown, partial]. Each position lists the number of instances found for that particular evaluation result.

If the array is applied to a simple element (i.e., one HTML element when evaluating one success criterion), then only one position can have a value, and this value must be 1. If the array is applied to a complex item (i.e., one success criterion that is applied to several categories of elements), then each position contains the total number of values aggregated from the child elements. Some examples follow:

- Single item, result = [0, 1, 0, 0, 0, 0]. This result means not applicable: the current success criterion cannot be applied to that item.
- Single item, result = [0, 0, 0, 1, 0, 0]. This means pass (OK). The current item successfully conforms to the current success criterion.
- Complex item, result = [1, 0, 2, 3, 5, 4]. In this case, the current item (i.e., a category of HTML elements) has 15 child items with the following results: 1 item fails, 0 items are not applicable, 2 items need manual evaluation, 3 items conform to the success criterion, 5 items could not be evaluated and 4 items only partially conform to the success criterion.

In addition, the internal representation changes when the current item contains child items belonging to several conformance levels. For instance, one guideline contains several success criteria, and each success criteria has a different conformance level. In that case, the evaluation result consists of an array of three six-element arrays:

Result = [[fail, NA, verify, OK, unknown, partial],
[fail, NA, verify, OK, unknown, partial],
[fail, NA, verify, OK, unknown, partial]]

The first element of this complex array is the aggregated result for conformance level A, the second element corresponds to level AA and the third to level AAA.

This array structure is quite flexible as it provides for:

- Individual evaluation of an object assigning a "1" to just one position in the array. This is used for techniques, failures and global evaluations.
- Evaluation of a parent object depending only on the evaluation values of its child objects. The parent object value is output using an aggregation algorithm. This is used for groups of techniques, elements, situations, element categories and success criteria.
- Evaluation of an object that depends on both the evaluation values and the conformance level of its sub-objects. In these cases, it is useful to store quantitative information about the number of elements with each value and conformance level, creating one array per conformance level. This is used for guidelines, principles and final web page evaluation.

The terms "parent" and "child" in the above description refer to the internal conceptual model of Hera-FFX, where one principle contains several guidelines, one guideline contains several success criteria, one success criterion contains several situations and several failures, one situation contains technique groups, and one technique group contains other groups or techniques. Thus, for instance, the parent of a situation is its corresponding success criterion and the child of a situation is a group of techniques.

A web page evaluation involves different elements, each with characteristics requiring different analyses. For this reason, three different aggregation algorithms are needed: permissive, restrictive and semi-permissive.

A description of the algorithms follows, including a formalization that uses the following notation:

- The 'i' and 'j' variables represent six-element arrays corresponding to child items of the current item.
- The "eval" function is applied to simple items and returns the position of the '1' value. For instance, eval([0, 1, 0, 0, 0, 0]) is NOT APPLICABLE and eval([0, 0, 0, 1, 0, 0]) is PASS.
- In some cases auxiliary subsets of items are used: "S1", "S2"… Each subset represents a set of child items with the same value.
- The "card" function returns the cardinality of one set (that is, the number of items).

The aggregation algorithms are:

1. *Permissive algorithm.* It is applied to elements having different means of passing the evaluation, where just one success is enough. One example is the set of sufficient techniques applicable in one situation. The permissive algorithm can be represented by the following values hierarchy and is detailed in Table 2:

   PASS >> VERIFY >> UNKNOWN >> PARTIAL >> FAIL >> N/A

Table 2: Permissive algorithm

Condition	Result
$\exists i \mid (eval(i) = \textbf{PASS})$	PASS
$\exists i \mid (eval(i) = \textbf{VERIFY}) \wedge (\forall j \neq i\ eval(j) \neq \textbf{PASS})$	VERIFY
$\exists i \mid (eval(i) = \textbf{UNKNOWN}) \wedge (\forall j \neq i\ eval(j) \neq \textbf{PASS} \wedge eval(j) \neq \textbf{VERIFY})$	UNKNOWN
$\exists i \mid (eval(i) = \textbf{PARTIAL}) \wedge (\forall j \neq i\ eval(j) \neq \textbf{PASS} \wedge eval(j) \neq \textbf{VERIFY} \wedge eval(j) \neq \textbf{UNKNOWN})$	PARTIAL
$\exists i \mid (eval(i) = \textbf{FAIL}) \wedge (\forall j \neq i\ eval(j) \neq \textbf{PASS} \wedge eval(j) \neq \textbf{VERIFY} \wedge eval(j) \neq \textbf{UNKNOWN} \wedge eval(j) \neq \textbf{PARTIAL})$	FAIL
$\forall i\ eval(i) = \textbf{NOT APPLICABLE}$	NOT APPLICABLE

2. *Restrictive algorithm*. It is used for elements that have several means of failing the evaluation, all of which have to be successful to prevent failure. An example is the set of common failures associated with a single success criterion. This algorithm can be represented with the following values hierarchy and is detailed in Table 3:

   FAIL >> PARTIAL >> VERIFY >> UNKNOWN >> PASS >> N/A

Table 3: Restrictive algorithm

Condition	Result
$\exists i \mid (eval(i) = FAIL)$	FAIL
$\exists i \mid (eval(i) = PARTIAL) \wedge (\forall j \neq i\ eval(j) \neq FAIL)$	PARTIAL
$\exists i \mid (eval(i) = VERIFY) \wedge (\forall j \neq i\ eval(j) \neq FAIL \wedge eval(j) \neq PARTIAL)$	VERIFY
$\exists i \mid (eval(i) = UNKNOWN) \wedge (\forall j \neq i\ eval(j) \neq FAIL \wedge eval(j) \neq PARTIAL \wedge eval(j) \neq VERIFY)$	UNKNOWN
$\exists i \mid (eval(i) = PASS) \wedge (\forall j \neq i\ eval(j) \neq FAIL \wedge eval(j) \neq PARTIAL \wedge eval(j) \neq VERIFY \wedge eval(j) \neq UNKNOWN)$	PASS
$\forall i\ eval(i) =$ NOT APPLICABLE	NOT APPLICABLE

3. *Semi-permissive algorithm*. It is a generalization of the permissive algorithm. It is applied to elements that have different means of passing the evaluation, where a minimum number of successes are required to pass. This is the case of some sets of sufficient techniques applied to some success criteria. It uses the same hierarchy as the permissive algorithm, plus an *n* parameter to specify the minimum number of PASSES required (Table 4).

The aggregation method also depends on the type of elements considered, as explained below. Note that it is the tool that chooses which algorithm to apply for a given item, depending on its meaning. For instance, if the current item is a group of sufficient techniques linked with an OR operator and with no requirement on a minimum of positive results, then the permissive algorithm is used. If the current item is a group of failures, then the restrictive algorithm is applied.

Technique evaluation

One technique or failure (tf) can be evaluated on each page element using an automatic test or manually by the human evaluator. If both values exist, the manual score (HumanResult) takes precedence over the automatic score (AutomaticResult).

```
IF (ThereIsHumanResult(tf)) THEN
  RETURN HumanResult(tf)
ELSE
  RETURN AutomaticResult(tf)
```

Aggregate evaluation of groups of techniques

A group of techniques (tg) can contain other groups or individual techniques. Therefore, the evaluation value of a group of techniques can be obtained from its techniques or subgroups. First, an accumulated array is built by adding the arrays of the subgroups or child techniques. Then one of the three algorithms is applied to output the final result. If the current group has an AND operator, then the restrictive algorithm is used because the result for the group will be the least favourable value in the array. If the operator is OR, then the permissive algorithm is used because the group will have the most favourable value. A much less common possibility is the establishment of a minimum number of successful results. In these cases, the operator is called ORN, because it is a generalization of the OR operator, restricted to *tg.N* successful values to pass. In this case, the semi-permissive algorithm is used.

```
FORALL child IN tg
{
  IF IsTechGroup(child) THEN
     result = result + EvalTechGroup(child)
  ELSE
     result = result + EvalTechnique(child)
}
IF (OperatorType(tg) = AND) THEN
  RETURN RunRestrictiveAlgorithm(result)
ELSE IF (OperatorType(tg) = OR) THEN
  RETURN RunPermissiveAlgorithm(result)
ELSE
  RETURN RunSemiPermissiveAlgorithm(result, tg.N)
```

Table 4: Semi-permissive algorithm

Condition	Result
$card(S1) \geq n$, where $S1 = \{i \mid (eval(i) = PASS)\}$	PASS
$card(S1) \geq n \wedge card(S2) < n$, where $S1 = \{i \mid (eval(i) = VERIFY)\}$, $S2 = \{i \mid (eval(i) = PASS)\}$	VERIFY
$card(S1) \geq n \wedge card(S2) < n \wedge card(S3) < n$, where $S1 = \{i \mid (eval(i) = UNKNOWN)\}$, $S2 = \{i \mid (eval(i) = PASS)\}$, $S3 = \{i \mid (eval(i) = VERIFY)\}$	UNKNOWN
$card(S1) \geq n \wedge card(S2) < n \wedge card(S3) < n \wedge card(S4)\ n$, where $S1 = \{i \mid (eval(i) = PARTIAL)\}$, $S2 = \{i \mid (eval(i) = PASS)\}$, $S3 = \{i \mid (eval(i) = VERIFY)\}$, $S4 = \{i \mid (eval(i) = UNKNOWN)\}$	PARTIAL
$card(C1) \geq n \wedge card(C2) < n \wedge card(C3) < n \wedge card(C4) < n \wedge card(C5) < n$, where $S1 = \{i \mid (eval(i) = FAIL)\}$, $S2 = \{i \mid (eval(i) = PASS)\}$, $S3 = \{i \mid (eval(i) = VERIFY)\}$, $S4 = \{i \mid (eval(i) = UNKNOWN)\}$, $S5 = \{i \mid (eval(i) = PARTIAL)\}$	FAIL
$\forall i\ eval(i) =$ NOT APPLICABLE	NOT APPLICABLE

Aggregate evaluation of situations

Each situation (sit) contains only one group of techniques (TechGroup(sit)). Thus, the result of one situation is the result for this group applying the above process.

```
tg = TechGroup(sit)
RETURN EvalTechGroup(tg)
```

Aggregate evaluation of elements
The value of the evaluation of one element (el) for one success criteria (sc) is calculated from the results for the group of techniques in its situation (el.sit) and from its common failures. First the failures are aggregated using the restrictive algorithm (one failure is enough to decide that the element has failed). And the situation is only evaluated if there are no failures or they are declared not applicable.

```
FORALL fail IN sc
   failresult = failresult +
                EvalFailure(fail, el)
res = RunRestrictiveAlgorithm(failresult)
IF (res = FAIL) THEN
  RETURN FAIL
ELSE IF (res = PARTIAL) THEN
  RETURN PARTIAL
ELSE IF (res = VERIFY) THEN
  RETURN VERIFY
ELSE IF (res = UNKNOWN) THEN
  RETURN UNKNOWN
ELSE
  RETURN EvalSituation(el.sit)
```

Aggregate evaluation of elements category
In general, the value of the evaluation of an elements category (elemcat) is calculated by applying the restrictive algorithm to the sum of the results of its child elements, because if a problem is detected in one of the elements there will be a problem with the category. But if a global evaluation has been manually established by the user, then this global value is used.

```
IF (GlobalValue(elemcat) ≠ VERIFY) THEN
  RETURN GlobalValue(elemcat)
ELSE
{
  FORALL el IN elemcat
    result = result + EvalElement(el)
  RETURN RunRestrictiveAlgorithm(result)
}
```

Aggregate evaluation of success criteria
The value of the evaluation of a success criterion (sc) is calculated by applying the restrictive algorithm to the sum of the results of its elements categories, because if one of them fails then the whole criterion will fail. But if a global evaluation has been manually established by the user, then this global value is used.

```
IF (GlobalValue(sc) ≠ VERIFY) THEN
  RETURN GlobalValue(sc)
ELSE
{
  FORALL elemcat IN sc
    result = result + EvalElementCat(elemcat)
  RETURN RunRestrictiveAlgorithm(result)
}
```

Aggregate evaluation of guidelines
Guidelines must reflect the results of their success criteria, taking into account that each success criterion belongs to a certain conformance level, and results are aggregated among success criteria at the same level. This means that three values are output per guideline, one for conformance level A, one for AA and one for AAA.

```
FORALL sc IN guideline
{
  IF (ConformanceLevel(sc) = A) THEN
    resultA = resultA + EvalSC(sc)
  ELSE IF (ConformanceLevel(sc) = AA) THEN
    resultAA = resultAA + EvalSC(sc)
  ELSE
    resultAAA = resultAAA + EvalSC(sc)
}
RETURN [ resultA, resultAA, resultAAA ]
```

Aggregate evaluation of principles
Principles also must reflect the results of their success criteria. The evaluation is similar to the aggregation of guidelines, grouped by level.

```
FORALL guideline IN principle
  resultarray = resultarray +
                EvalGuideline(guideline)
RETURN resultarray
```

Final evaluation
The value of a web page evaluation is calculated from the results of the success criteria taking into account the conformance levels.

```
FORALL principle
  resultarray = resultarray +
                EvalPrinciple(principle)
RETURN resultarray
```

5. EVALUATION

Hera-FFX 2 is under active development but a working prototype exists and has been evaluated to assess the impact on the use of the tool in relation to the accuracy of the results provided by novice evaluators.

Previous work has researched the testability of WCAG 2, both by novices [1][5] and by experts [6]. In all cases researchers agree that some success criteria are not testable, that is, 80% of the evaluators did not agree on the correct result.

In our case we already ran an experiment with novice evaluators [1] and decided to determine the impact of Hera-FFX on this type of evaluators. We had data from several courses on web accessibility where no tool was used, and we compared them with new students.

The current version of Hera-FFX 2 was used by four students attending the “challenges of ICT accessibility for people with functional diversity” module as part of a Master in Software and Systems taught at the Technical University of Madrid’s Computer Science School. Part of the module focused on web accessibility, and we used a collaborative learning approach as described in [2]. These four students were somewhat less familiar with web accessibility than the students from previous courses, because the module included learning goals for a broader concept of accessibility beyond just the web domain.

Of course, this is a very low number of evaluators, and our conclusions will not be statistically significant. Our goal, however, was to find out if the use of an early version of Hera-FFX 2 had any impact at all.

The students of the module were set the exercise of evaluating one web page. The web page was our University’s home page, which we had also used in previous experiments. Although the contents have changed since our last experiment, the structure is the same. For this reason, we believe the results of the evaluation exercise will be comparable.

Figure 4 shows the percentage of students that provided the correct result in our previous experiments (web accessibility courses in the ATHENS 2009 and 2010 programme) and when using Hera-FFX 2 (labelled as MUSS).

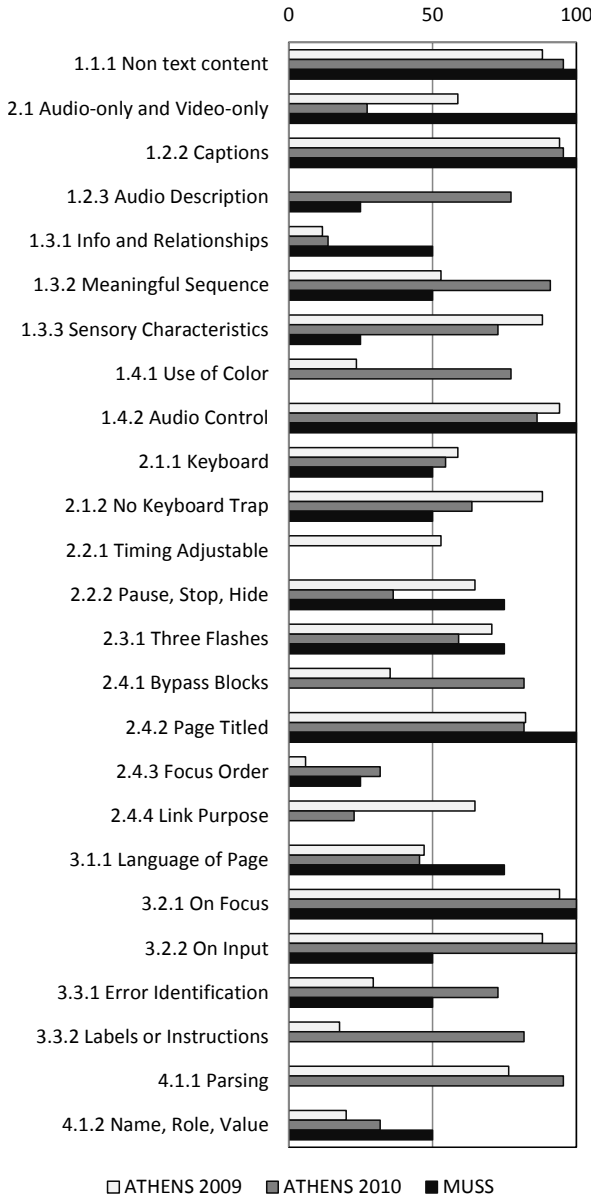

Figure 4: Comparing results when using no tool (ATHENS 2009 and 2010) and when using Hera-FFX 2 (MUSS)

Our experiment shows that roughly 50% of the success criteria yielded better results using the tool, even though the students were less well trained than the students from other courses. This leads us to believe that the use of the tool will be beneficial in future modules.

On the one hand, some key examples of the benefits of using Hera-FFX were success criteria 1.1.1, 1.2.1, 1.2.2, 1.4.2, 2.4.2, and 3.2.1, where 100% of the four students agreed on the correct results.

On the other hand, there are also some cases where the four students clearly performed worse than in other courses, especially on success criteria 1.3.3, 1.4.1, 2.4.1, 2.4.4, 3.3.2 and 4.1.1. We were concerned about these results, and we held a session in our module to discuss the findings. We reached the conclusion that the main reason for these mistakes was a knowledge gap due to the limited amount of time that students had had to work on WCAG 2.0, where we found that tool use had been unable to offset this missing knowledge.

In addition, although we did not perform any detailed usability evaluation, we were interested in the user experience of Hera FFX 2. We asked our four students to give their opinion about the tool that they had used. All the students had a very positive opinion of the tool, and they especially highlighted the intuitiveness of the user interface. They also stated that the tool was effective and useful for evaluating the accessibility of web pages.

Of course, there were some exceptions, all of which, however, were related to tool instability, which was understandable as it was an unfinished product.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the on-going development of Hera-FFX 2, a Mozilla Firefox plug-in supporting the manual evaluation of web accessibility based on WCAG 2.0.

Hera-FFX 2 has two main contributions. First, it is a tool that enables a full and detailed manual evaluation of the accessibility of a web page, with the additional support of a few automated tests. Second, it is a tool that mimics the complete structure of WCAG 2.0: principles, guidelines, success criteria, situations, techniques and failures.

Although the tool is under active development, the current prototype has been evaluated by a small number of students with positive results, leading us to think that we are working in the right direction.

There is still quite a lot of future work to be done. The user interface of the tool is being improved based on the feedback from the first evaluation. One specific complex issue is the generation of an HTML report. In the current version, the report is too big to be useful. In addition, we plan to add the function of generating semantic EARL-based reports that could be used by other tools. Other future work is related to adding more automated tests (which are easy to incorporate as each test is programmed as a JavaScript function) and updating the tool as soon as new techniques and failures are published by the W3C.

ACKNOWLEDGMENTS

We would like to acknowledge work by Carlos Núñez on implementing the current prototype of Hera FFX for WCAG 2.0.

REFERENCES

[1] Alonso, F., Fuertes, J. L., González, Á. L., and Martínez, L. 2010. On the testability of WCAG 2.0 for beginners. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility* (W4A2010). (Raleigh, USA, April 2010). DOI= http://dx.doi.org/10.1145/1805986.1806000.

[2] Alonso, F., Fuertes, J. L., González, Á. L., and Martínez, L. 2010. Using collaborative learning to teach WCAG 2.0.

Lecture Notes in Computer Science 6179 (July 2010), 400-403.

[3] Benavídez, C., Fuertes, J. L., Gutiérrez, E., and Martínez, L. 2006. Teaching Web Accessibility with "Contramano" and Hera. *Lecture Notes in Computer Science* 4061 (July 2006), 341-348.

[4] Benavídez, C., Fuertes, J. L., Gutiérrez, E., and Martínez, L. 2006. Semi-Automatic Evaluation of Web Accessibility with HERA 2.0. *Lecture Notes in Computer Science* 4061 (July 2006), 199-206.

[5] Brajnik, G. 2009. Validity and Reliability of Web Accessibility Guidelines. In *Proceedings of ASSETS'09* (Pittsburgh, Pennsylvania, USA, October 25 - 28, 2009). ACM Press. 131-138. DOI= http://dx.doi.org/10.1145/1639642.1639666.

[6] Brajnik, G., Yesilada, Y., and Harper, S. 2010. Testability and Validity of WCAG 2.0: The Expertise Effect. *Proceedings of ASSETS'10* (Orlando, Florida, USA, October 25 - 27, 2010). ACM Press 2010. 43-50. DOI= http://dx.doi.org/10.1145/1878803.1878813.

[7] Caldwell, B., Cooper, M., Reid, L. G., and Vanderheiden, G. (eds). 2008. Web Content Accessibility Guidelines 2.0. World Wide Web Consortium Recommendation (December 2008). http://www.w3.org/TR/WCAG20/.

[8] Chisholm, W., Vanderheiden, G., and Jacobs, I. (eds.) 1999. Web Content Accessibility Guidelines 1.0. World Wide Web Consortium Recommendation (May 1999). http://www.w3.org/TR/WCAG10.

[9] Cooper, M.; Reid, L.G.; Vanderheiden, G. and Caldwell, B. (eds.) 2010. Techniques for WCAG 2.0. Techniques and Failures for Web Content Accessibility Guidelines 2.0. W3C Working Group Note (October 2010) http://www.w3.org/TR/WCAG20-TECHS/.

[10] CTIC Foundation. 2011. TAW 3. http://tawdis.net/index.html?lang=en.

[11] Deque Systems. 2011. Worldspace FireEyes. http://www.deque.com/products/worldspace-fireeyes

[12] Fuertes, J. L., González, R., Gutiérrez, E., and Martínez, L. 2009. Hera-FFX: a Firefox add-on for semi-automatic web accessibility evaluation. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility* (W4A2009) (Madrid, Spain, April 2009). 26-35. DOI= http://dx.doi.org/10.1145/1535654.1535661.

[13] Gay, G., Qi Li, C. 2010. AChecker: open, interactive, customizable, web accessibility checking. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility* (W4A2010) . (Raleigh, USA, April 2010). DOI= http://dx.doi.org/10.1145/1805986.1806019.

[14] Sidar Foundation. 2011. Fundación Sidar - Acceso Universal, Seminario SIDAR (in Spanish). http://www.sidar.org/.

[15] Total Validator. 2011. http://totalvalidator.com/.

[16] World Wide Web Consortium. 2010. Evaluating Web Sites for Accessibility: Overview. http://www.w3.org/WAI/eval/Overview.html.