D4.1 Test Suites' State of the Art and Quality Assurance Methods for W3C Recommendations

Contractual Date of Delivery to the EC: 28th February 2005
Actual Date of Delivery to the EC: 31st March 2005
Editor: Christophe Strobbe (KULRD)
Contributors: Christophe Strobbe (KULRD), Carlos A Velasco (FIT)
Workpackage: 4
Security: Public
Nature: Report
Version: K
Total number of pages: 45

Keywords: test suite, benchmark, quality assurance, accessibility, Web Content Accessibility Guidelines (WCAG), evaluation & repair tool, HTML, XHTML, CSS, SVG, XForms, Flash, Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C), test suite architecture, unit testing.
# Table of Contents

1 Executive Summary ................................................................. 4  
2 Introduction.............................................................. 5  
3 W3C Quality Assurance ............................................................... 7  
4 Test Suites .............................................................................. 9  
   4.1 Accessibility-Related Test Suites ........................................... 11  
      4.1.1 UAAG 1.0 Test Suite for HTML 4.01 ............................ 11  
      4.1.2 UAAG 1.0 Test Suite for SVG 1.1 .............................. 12  
      4.1.3 W3C WAI Test Suites for User Agent Accessibility Guidelines Hosted by CITA ....................................................... 13  
         4.1.3.1 HTML 4.01 Test Suite for User Agent Accessibility Guidelines .... 13  
         4.1.3.2 UAAG 1.0 Test Suite for HTML 4.01 .......................... 14  
         4.1.3.3 UAAG 1.0/SVG 1.1 Test Suite .................................. 14  
         4.1.3.4 UAAG 1.0 and SMIL 2.0 ........................................ 14  
      4.1.4 HTML Test Suite for WCAG 2.0 .................................. 15  
      4.1.5 ATRC Web Checker – Guidelines ................................ 16  
      4.1.6 Web-based Intranet and Internet Information and Applications Checklist... 19  
      4.1.7 The HTML Challenge ................................................... 19  
      4.1.8 Test Plan to Test HTML Elements for Accessibility ............................... 21  
      4.1.9 Flash Test Suite ........................................................ 22  
      4.1.10 Screen Reader Visibility Testing .................................. 22  
      4.1.11 Conclusion ............................................................. 23  
   4.2 Test Suite Architecture ..................................................... 25  
      4.2.1 Introduction ............................................................ 25
4.2.2 Test Suites and the W3C Quality Assurance Framework.........................27
4.2.3 The DOM Test Suite Methodology..........................................................29
4.3 Test Suites as Unit Tests: Benefits and Drawbacks....................................32
5 Future Work.................................................................................................33
6 References..................................................................................................34
7 Appendix A: Test Suites Not Covered in This Report....................................36
  7.1 CSS........................................................................................................36
  7.2 HTML/XHTML....................................................................................37
  7.3 Other XML Vocabularies.......................................................................39
  7.4 Test Suites for Web Clients/Browsers.....................................................41
  7.5 Character Sets and Entities....................................................................43
  7.6 Other Types of Content or Applications................................................43
8 Appendix B: Tools.........................................................................................45
  8.1 JUnit......................................................................................................45
  8.2 XmlUnit................................................................................................45

**List of Tables**

Table 1. Summary of accessibility-related test suites.................................24

**List of Figures**

Figure 1. Test Execution Framework..........................................................28
1 Executive Summary

This document describes test suites and quality assurance methods for W3C recommendations. It briefly describes how test suites are defined by the W3C Quality Assurance activity. The next chapter analyses accessibility-related test suites (both drafts and finished versions) from which BenToWeb may use materials and/or gather ideas. The analysis addresses scope, coverage, size, automation aspects and copyright/licensing issues. The next chapter discusses the architecture of W3C test suites. The DOM test suite is used as an illustrative example of a suite with a high degree of automation. The final chapter discusses future work that is required to move from test suites to benchmarks.
2 Introduction

To improve the quality of an accessibility evaluation tool, different kinds of evaluations can be performed. Melody Ivory and others have reported on empirical studies of web site evaluation tools [Ivory 2002, 2003a, 2003b]. Giorgio Brajnik developed an evaluation method based on sampling issues from evaluation reports [Brajnik 2004]. The quality of the samples (e.g. when sampling evaluation reports) and the coverage of issues addressed by WCAG in the sampled web sites is crucial to the reliability of the results of such evaluations. Although it is important to see how web site evaluation tools perform on web pages found “in the wild”, it would be beneficial to have a test suite. A test suite has several advantages over real web sites:

- coverage of issues addressed by WCAG can be guaranteed to be complete;
- comparisons of subsequent versions of a tool would be more reliable, because a test suite is stable, whereas websites change;
- test suites can also be built for document types that are currently not common enough on real websites, e.g. SVG and XForms;
- only a fixed set of test cases (i.e. a test suite) can allow reliable benchmarking.

When web site evaluation tools support EARL, comparisons between tools can be more complete and easier to automate.

The World Wide Web Consortium (W3C) has constructed test suites for many of its technical recommendations. However, W3C does not consider the comparison of software programs implementing its specifications as one of its tasks. Quality Assurance primarily focuses on the quality of W3C reports, and the test suites support software developers in quality assurance efforts for their implementations of W3C specifications. This report will provide a short overview of the W3C Quality Assurance guidelines that are relevant to test suite development.

BenToWeb would like to build on previous efforts and reuse existing test material whenever possible. For this reason, this report also analyses existing test suites, their relevance to Web accessibility, and IPR issues.

Since the test cases that make up a test suite can often be thought of as unitary tests, and running a test suite automatically bears resemblance to running a set of unit tests, this report will also compare test suites with unit tests. An exhaustive analysis of traditional methods for software testing is not considered useful: using web site evaluation tools to test
how well web sites conform to WCAG is already at one remove from actual user experience, so evaluating how well web site evaluation tools perform these tests is something much more abstract than testing a browser's HTML support or measuring the performance of an XML parser's DOM support. Software engineering literature provides information on quality assurance for software, including testing frameworks that can be used for this, but the level that is relevant to BenToWeb is how to build and test testing frameworks. Fortunately, some test suite development efforts are well documented. The DOM Test Suite Methodology is an example with a high degree of automation, but the analysis will show that it is not a suitable methodology for BenToWeb.
3 W3C Quality Assurance

In 2001, W3C launched the Quality Assurance (QA) Activity\(^1\) to ensure that its deliverables – W3C Recommendations – are implemented correctly. The Quality Assurance Activity gathers and formalises quality assurance efforts of the W3C working groups. One of the activities of the Quality Assurance Working Group (QAWG) is the “development of a common framework and harness for developing and running tests, and a process for maintaining, adding and removing tests from test suites”\(^2\). W3C working groups can create several types of quality-related deliverables:

- test materials,
- systematic, thorough reviews of specification,
- sample code (if not considered test material),
- model specification in formal language. [W3C, 2004b]

Test materials include:

- test suites (tests, documentation and harness);
- validation tools;
- test assertions;
- sample code (in some cases);
- checklists;
- Implementation Conformance Statement (ICS) pro-forma. [W3C, 2004b]

The W3C QA Handbook also points out that not all test materials are conformance test materials: “**Test materials** provide for the evaluation of an implementation against the requirements of a specification and/or provide information about the implementation. **Conformance Test Materials** are test materials that are used to indicate conformance of an implementation to a specification” [W3C, 2004b]. Test materials are helpful for implementors of W3C specifications; in fact, one of the conditions for a call for review of a proposed recommendation is evidence that each feature of the specification has been implemented (preferably, two interoperable implementations of each feature should exist). However, the W3C Process Document does not require that implementations should use a Working Group's test materials (Section 7.4.4 of [W3C 2004a]). Moreover, Working Groups are not expected to perform evaluations of

---

\(^1\) [http://www.w3.org/QA/](http://www.w3.org/QA/).

\(^2\) QA Working Group Charter (QAWG) - [http://www.w3.org/QA/WG/charter](http://www.w3.org/QA/WG/charter) (dated 2003/07/17).
implementations, let alone benchmark implementations. Where conformance evaluations are available, they are for informative purposes only. For example, the disclaimer of the Authoring Tool Conformance Evaluations\(^3\) states that the evaluations are not meant to be used as definitive reviews of the products or to compare product accessibility [W3C 2002c].

The next chapter discusses accessibility-related test suites developed at the W3C and elsewhere.

---

4 Test Suites

Many, but not all, W3C specifications have a test suite. The Quality Assurance WG maintains a Matrix of W3C Specifications\(^4\), which lists specifications which are at least at Last Call stage or for which a test suite is being developed at Working Draft stage\(^5\). On 7 February 2005, the Matrix contained 78 Recommendations (with 42 test suites), 17 Candidate Recommendations (with 6 test suites), 1 Proposed Recommendation and 19 Last Call Working Drafts (with no test suites). The Matrix does not mention the HTML Test Suite for WCAG 2.0, the SVG 1.0 Test Suite and any test suites that were not contributed to W3C. Some other organisations have also produced test suites. A list of accessibility-related test suites is provided below.

- UAAG 1.0 Test Suite for HTML 4.01 (Draft; Jon Gunderson, Colin Koteles, Matt May; last modified 2004/03/09; URL: [http://www.w3.org/WAI/UA/TS/html401/](http://www.w3.org/WAI/UA/TS/html401/));
- HTML Test Suite for WCAG 2.0: [http://www.w3.org/WAI/GL/WCAG20/tests/](http://www.w3.org/WAI/GL/WCAG20/tests/) (test suite for testing conformance WCAG 2.0 guidelines, to be used by authors, manufacturers of authoring tools and manufacturers of user agents).
- HTML Test Suite for WCAG 2.0 - Sorted by Guideline (W3C Working Draft 2004-12-9; Chris Ridpath, Jenae Andershonis; URL: [http://www.w3.org/WAI/GL/WCAG20/tests/](http://www.w3.org/WAI/GL/WCAG20/tests/));
- ATRC Web Checker – Guidelines (maintained by Chris Ridpath and mostly contributed to the HTML Test Suite for WCAG 2.0; URL: [http://tile-cridpath.atrc.utoronto.ca/acheck/servlet/ShowGuide](http://tile-cridpath.atrc.utoronto.ca/acheck/servlet/ShowGuide));
- The HTML Challenge (example files produced by Freedom Scientific to demonstrate features of JAWS): [http://www.w3.org/QA/TheMatrix](http://www.w3.org/QA/TheMatrix).

\(^4\) For a definition of the terms “Last Call” and “Working Draft”, see [W3C 2004a].

- Test Plan To Test HTML Elements For Accessibility (Mozilla): http://www.mozilla.org/quality/embed/plans/accessibility/HTMLAccessibility.html.

- Flash Test Suite (Bob Regan & Andrew Kirkpatrick): http://www.markme.com/accessibility/archives/005564.cfm (a test suite is discussed at this location, but there is no pointer to a set of files).


The following test suites are relevant to the document types for which BenToWeb will produce accessibility test suites:

- HTML 4.01 test suite (general, not oriented towards accessibility; can be downloaded as a zipfile): http://www.w3.org/MarkUp/Test/HTML401/current/ (see W3C HTML/XHTML Test Suites);

- CSS Test Suites (http://www.w3.org/Style/CSS/Test/). Some “CSS Browser Test Results” by David Barron are available at http://dbaron.org/css/browsers/ (last updated in 1999).

- W3C Scalable Vector Graphics (SVG) 1.0 Test Suite (Historical) (3rd release 26 October 2001; can be downloaded a a zipfile; http://www.w3.org/Graphics/SVG/Test/Overview-10.html);

- W3C Scalable Vector Graphics (SVG) 1.1 Test Suite (2nd release 13 August 2003; http://www.w3.org/Graphics/SVG/Test/);

- XForms 1.0 Test Suite (created during CR phase; can be downloaded as a zipfile; http://www.w3.org/MarkUp/Forms/Test/); the document “How to maintain the XForms Test Suite” provides links to implementation reports;

- XML W3C Conformance Test Suite (10 December 2003; downloadable as zipfile or tarfile; http://www.w3.org/XML/Test/xmlconf-20031210.html) is a test suite for XML 1.1.

- Mozilla: Browser Standards Compliance QA: http://www.mozilla.org/quality/browser_sc.html. Test suites for HTML 4.0 Transitional, CSS1, DOM1 (HTML and Core), XML, XHTML 1.0 Transitional (HTML) and XHTML 1.0 Transitional (CSS). The last two sets of tests have been removed “due to numerous bugs, lack of a maintainer, and duplication of better-maintained test suites”

http://www.w3.org/MarkUp/Forms/Test/maintain.html.
but can still be downloaded as a tar.gz file.\footnote{See \url{http://www.mozilla.org/quality/browser/standards-removed.html}.}

Note that there are no test suites for any version of XHTML, except for the HTML Test Suite for WCAG 2.0, which is still under development.

## 4.1 Accessibility-Related Test Suites

The following sections analyse accessibility-related test suites. Each section discusses the purpose and status of the test suite, its size, technical aspects (DTD, XSLT, scripts) that may be relevant to test automation, whether the test suite could be useful for the benchmarking efforts of BenToWeb, and licensing issues.

### 4.1.1 UAAG 1.0 Test Suite for HTML 4.01

URL: \url{http://www.w3.org/WAI/UA/TS/html401/}.

This test suite is a draft edited by Jon Gunderson, Colin Koteles and Matt May. It was last modified on 9 March 2004. The User Agent Accessibility Guidelines 1.0 are a W3C Recommendation since 17 December 2002\footnote{\url{http://www.w3.org/TR/UAAG10/}.}. The HTML 4.01 Specification dates from 24 December 1999\footnote{\url{http://www.w3.org/TR/1999/REC-html401-19991224}.}.

According to the disclaimers in the documentation, the “test suite may be useful in conducting a conformance review of a user agent, but is not sufficient for determining if all requirements have been satisfied for making a conformance claim”. The test suite contains test cases for 36 of the 83 checkpoints of the UAAG; with no test cases for guidelines 6, 7, 8 and 12\footnote{Not all checkpoints of UAAG can be tested with a test suite of HTML files. Conformance to guidelines 6, 8 and 12 may be tested with other test suites, such as test suites for DOM and HTML (guidelines 6 and 8) or other types of testing or verification.}. A “Summary implementation report for UAAG 1.0” with results for 15 different user agents is also available\footnote{\url{http://www.w3.org/WAI/UA/impl-pr2/}.}. An example of a test report for Mozilla 1.6 is available in the mailing list archive of the User Agent Working Group\footnote{\url{http://lists.w3.org/Archives/Public/w3c-wai-ua/2004JanMar/0015.html}.}. Kynn Bartlett posted an evaluation of Safari Public Beta 2 (v74) on his weblog\footnote{\url{http://www.maccessibility.com/archive/000596.php}.}, but his article refers only to the UAAG 1.0 checklist, not to the test suite. Jon Gunderson and Jim Allan made a presentation on the test suite at CSUN 2003\footnote{\url{http://www.csun.edu/cod/conf/2003/proceedings/190.htm}.}.

The test suite “attempts to represent all the accessibility features of HTML 4.01, but should not be considered as a definitive list”. The coverage of all
accessibility features of HTML could make this set of files a good starting point for a test suite for benchmarking evaluation and repair tools. Since each of the test files are generated from an XML file, for which the DTD and the XSLT file are provided, it should be straightforward to generate a similar set of XHTML files. The test suite consists of 124 XML files (and their corresponding HTML files), which are all linked from the start page of the test suite (see URL above).

Since the test suite is intended for the evaluation of the accessibility of user agents, testing is by definition a manual process and cannot be automated (unlike testing a web site with an evaluation and repair tool).

W3C owns copyright for the test suite. When using and/or copying the documents and the test suite, the W3C Document and License applies\(^\text{15}\). However, the DTD is governed by the W3C Software Notice and License\(^\text{16}\). The documentation for the test suite leaves the licensing of the XSLT undefined; we assume that the W3C Software Notice and License applies.

### 4.1.2 UAAG 1.0 Test Suite for SVG 1.1

**URL:** [http://www.w3.org/WAI/UA/TS/svg/Overview.html](http://www.w3.org/WAI/UA/TS/svg/Overview.html).

This test suite is a draft edited by Jon Gunderson and Matt May. It was last modified on 23 April 2004. The Scalable Vector Graphics (SVG) 1.1 Specification was first published on 4 September 2001 and revised on 14 January 2003\(^\text{17}\).

According to the disclaimers in the documentation, the “test suite may be useful in conducting a conformance review of a user agent, but is not sufficient for determining if all requirements have been satisfied for making a conformance claim”. The test suite contains four test cases: 3 files for checkpoint 1.1 (“full keyboard access”) and one for checkpoint 4.1 (“configure text scale”). The test cases make use of two SVG files: one that conforms to the SVG 1.1 Tiny specification, and one that conforms to the SVG 1.0 specification. The first file was borrowed from the SVG 1.1 Test Suite (fonts-elem-01-t.svg, with <SVGTestCase> element removed); the second was borrowed from the SVG 1.0 Test Suite (style-selector-BE-02.svg). There is no indication that these files were originally produced to test or demonstrate accessibility aspects of SVG. (The SVG files contain <title> and <desc> elements that provide information on the purpose of the files, but there is no such information for each of the graphical elements they contain.)

\(^{15}\) [http://www.w3.org/Consortium/Legal/copyright-documents-19990405].  
\(^{16}\) [http://www.w3.org/Consortium/Legal/copyright-software-19980720].  
\(^{17}\) [http://www.w3.org/TR/2003/REC-SVG11-20030114/index.html].
As in the UAAG 1.0 Test Suite for HTML 4.01, test files are generated from XML files describing the tests. The DTD and XSLT files are the same as those used in the test suite for HTML. The test files that are generated are HTML files that explain and describe the tests, not the actual SVG files, which are actually borrowed from SVG test suites.

Since the test suite is intended for the evaluation of the accessibility of user agents supporting SVG, testing is by definition a manual process and cannot be automated.

Regarding copyright and licensing, the same terms apply as for the UAAG 1.0 Test Suite for HTML 4.01.

### 4.1.3 W3C WAI Test Suites for User Agent Accessibility Guidelines Hosted by CITA

The website Center for Instructional Technology Accessibility (CITA) of the University of Illinois at Urbana-Champaign (UIUC) hosts a number of test suites (mostly in draft status) by Jon Gunderson and others. Jon Gunderson is chair of the User Agent Working Group. The test suites are being used by WAI and are part of Jon Gunderson's work with WAI. The materials from the test suites can be reused for other test suites (such as those in BenToWeb), but CITA should be credited for any materials that are reused.  

### 4.1.3.1 HTML 4.01 Test Suite for User Agent Accessibility Guidelines


The HTML 4.01 Test Suite for User Agent Accessibility Guidelines, edited by Jon Gunderson, Dominique Kilman and Colin Koteles appears to be an older draft of the test suite hosted at the W3C web site. The test suite resulted from a class project (LIS350AR Spring 2002 Class Project) It contains only static HTML pages and does not provide links to a DTD, an XSLT file or XML files (like those that are available at W3C) although the production of these resources was among the goals of the project. Since the HTML 4.01 Test Suite hosted by CITA is now part of the test suite discussed in section 3.1.1, it will not be treated as a separate resource.

---

18 Personal e-mail with Jon Gunderson, 7 March 2005.
4.1.3.2 UAAG 1.0 Test Suite for HTML 4.01


The UAAG 1.0 Test Suite for HTML 4.01 at CITA is larger than the one at W3C: it contains 145 tests, covering the same guidelines as the W3C version (guideline 11 has 7 tests instead of 14: 7 of the 8 tests of checkpoint 11.4 were removed). A web interface for conducting a review and recording results for Internet Explorer 6 (on Microsoft Windows) is available on Jon Gunderson’s web site. Draft evaluations for seven different user agents (six browsers, one screen reader) are also available.

The home page of the test suite does not provide links to XML files, a DTD or an XSLT file for generating the HTML files. Copyright belongs to the University of Illinois at Urbana-Champaign.

4.1.3.3 UAAG 1.0/SVG 1.1 Test Suite


The UAAG 1.0/SVG 1.1 Test Suite at CITA contains only one test (for checkpoint 1.1), which is not available in the W3C test suite.

The home page of the test suite does not provide links to XML files, a DTD or an XSLT file for generating the HTML files. Copyright belongs to the University of Illinois at Urbana-Champaign.

4.1.3.4 UAAG 1.0 and SMIL 2.0


CITA hosts a test suite for UAAG 1.0 and SMIL 2.0. There is no similar test suite at the W3C, although it is on the list of potential future test suites of the UAAG Working Group. The Synchronized Multimedia Integration Language (SMIL 2.0) Specification was published on 7 August 2001.

The test suite contains 66 tests covering guidelines 1-5.

The home page of the test suite does not provide links to XML files, a DTD or an XSLT file for generating HTML or SMIL files. Copyright belongs to the University of Illinois at Urbana-Champaign.

22 [http://www.w3.org/WAI/UA/TS/](http://www.w3.org/WAI/UA/TS/).
4.1.4 HTML Test Suite for WCAG 2.0

URL: http://www.w3.org/WAI/GL/WCAG20/tests/.

This test suite is a draft edited by Chris Ridpath and Jenae Andershonis. It was last updated on 22 February 2005. The homepage of the test suite states that “this document provides information to Web content developers who wish to make their content comply with the 'Web Content Accessibility Guidelines 2.0' [WCAG20] (currently a Working Draft). It includes test cases, code examples, and references to help authors test for compliance.” In addition, the WCAG Working Group also “strongly encourages manufacturers of authoring tools to support the process of authoring content that conforms to this test suite, and encourages manufacturers or user agents, including assistive technologies, to implement the behaviors described by these tests”. Although the title of the test suite only mentions HTML, the examples used in the tests are actually XHTML, unless there is a specific reason to use HTML. The tests are intended for use both with HTML 4.01 and XHTML 1.0/1.1.

The test suite contains 28 accepted test cases and 141 other test cases (pending, rejected or holding). Unlike some of the UAAG test suites, this test suite does not use XML to generate and describe test files. All test files are linked from a file describing the test; these test descriptions are in turn linked from the homepage of the test suite. The list of tests can be viewed by guideline, by priority level or by HTML element. Each test file is an HTML or XHTML file with minimal code: apart from the code that is required in any (X)HTML file (DOCTYPE, <html>, <head>, <title>, <body>), the files only contain the code that demonstrates a fail or pass for a success criterion. A tester or reader uses the test suite by reading the test description (which contains, among other things, the relevant source code and states whether the test file passes or fails), following the link to the test file and checking it (visually or otherwise) in the interface of his user agent. There is no machine-readable description (e.g. in XML) of the test suite, and consequently there is no mechanism that allows the test suite to be used for automated testing of evaluation and repair tools.

Since the Web Content Accessibility Guidelines 2.0 are technology independent, the WCAG Working Group also defines technology-specific information in so-called techniques documents and checklists; the requirements for these documents were documented in 2003. These techniques documents may also link to tests. The charter of the WCAG

24 http://www.w3.org/WAI/GL/WCAG20/tests/.
25 See the “Conformance Test Process For WCAG 2.0” at http://www.w3.org/WAI/GL/WCAG20/tests/ctprocess.html for a description of the steps that must be performed so that a test case may be added to the test suite.
26http://www.w3.org/TR/wcag2-tech-req.
Working Group contains a list of milestones that includes the techniques documents for the following technologies:

- HTML,
- CSS,
- scripting,
- VoiceXML,
- RDF,
- SMIL,
- SVG,
- XForms.

The Techniques Task Force of the WCAG Working Group has developed a web application for submitting techniques for WCAG 2.0. The submission form can be accessed at [http://www.w3.org/WAI/GL/WCAG20/TECHS-SUBMIT/](http://www.w3.org/WAI/GL/WCAG20/TECHS-SUBMIT/); the list of submitted techniques is available at [http://lists.w3.org/Archives/Public/public-wcag2-techs/](http://lists.w3.org/Archives/Public/public-wcag2-techs/). Some techniques contain references to test files, e.g. the (X)HTML technique regarding links submitted by Chris Ridpath on 21 June 2004. The Techniques Task Force uses an XML DTD which is a extension of XMLSpec (the DTD for W3C specifications; see section 4.2.3) and uses XSLT transformations to generate an XHTML version of the techniques documents. There seems to be no DTD or XML Schema for test cases.

Regarding copyright and licensing, the same terms apply as for the UAAG 1.0 Test Suite for HTML 4.01.

### 4.1.5 ATRC Web Checker – Guidelines


Chris Ridpath of ATRC (the Adaptive Technology Resource Centre of the University of Toronto), who is a “participant in good standing” of the WCAG WG, maintains a site with guidelines in use at ATRC. Chris Ridpath refers interested parties to the test set at W3C instead of the above URL, so this test set should not be considered as a different set from the WCAG 2.0 test suite discussed in the previous section. The test set at ATRC is discussed separately because the guidelines allow different views of the

---

28 See Development of Techniques for WCAG 2.0, at [http://www.w3.org/WAI/GL/wcag20.html#techs](http://www.w3.org/WAI/GL/wcag20.html#techs).
29 This is an unstable link; Chris Ridpath recommends going to the test set at the W3C: [http://www.w3.org/WAI/GL/WCAG20/tests/](http://www.w3.org/WAI/GL/WCAG20/tests/).
test files. Visitors can select any guideline name to view the accessibility checks that are required by that guideline. The guidelines listed are:

- BITV 1.0 (Level 2) (Barrierefreie Informationstechnik-Verordnung, the German Decree on Barrier-Free Information Technology [BITV])
- Section 508
- WCAG 1.0 (Level A)
- WCAG 1.0 (Level AA)
- WCAG 1.0 (Level AAA)
- WCAG 2.0 Alt Text
- WCAG 2.0 L1
- WCAG 2.0 L2
- WCAG 2.0 L3

For each accessibility check, there is a table of properties (such as Description, Error Text, Status, Detection Confidence and Decision Text), two or more test files and a list of guidelines known to use the check. (Many accessibility checks and test files are reused for several guidelines.) For each accessibility check, there is one test file that passes and at least one that fails the check (if test files are available). There is one large set of test files, and each guideline is made up of a subset from this set. The large set is expandable and more accessibility checks, e.g. for BITV, may be added in the future. There is not one guideline that includes all of the accessibility tests. For WCAG 2.0 L3, which is presumably the most comprehensive set, there are 340 test files (16 accessibility checks in this set have no test files). For each guideline, a disclaimer states that the accessibility checks required by the guideline were assiged by the ATRC and that the governing bodies for BITV, Section 508 or WCAG (respectively) have not sanctioned the accessibility checks.

On 25 August 2004, Chris Ridpath sent a call for public comment to the WAI GL mailing list, requesting public comment on the WCAG 2.0 Conformance Test Suite at ATRC, and providing the following description:

All of these tests are in draft form and are open to change. This test suite has not been sanctioned by the WAI.

The suite is comprised of a series of atomic tests that check the accessibility of HTML content. It will clearly describe all the HTML accessibility problems that are covered by the WCAG2 and will be a clear interpretation of the these guidelines.
This test suite applies to HTML/XHTML content only but it is anticipated that other technologies will have test suites that test for conformance.  

Chris Ridpath also mentioned that

This development has grown out of a common goal between the software products Bobby (from Watchfire) and A-Prompt (from University Of Toronto) to harmonize a series of accessibility checks. The test files used in this test suite are based upon a series of tests files created by Josh Krieger and Chris Ridpath that were part of the Accessibility Tool Reviewer software project[3]. This work is also related to the Accessibility Evaluation and Repair Techniques (AERT) document that was produced for the WCAG1[4]. The test process format is based upon the UAAG test suite[5].


The Techniques for Accessibility Evaluation and Repair Tools (AERT) is a W3C Working Draft dating from 26 April 2000 that “describes techniques that Web accessibility validation tools may use to evaluate the conformance of HTML documents to the Web Content Accessibility Guidelines 1.0 (WCAG 1.0)”\(^{31}\). The Access Tool Reviewer (ATR)\(^{32}\), also by Chris Ridpath, is a tool to assist in the evaluation of accessibility validation tools. The ATR includes test files keyed to the techniques of AERT. A tester can run the test files through an E&R tool, compare the results of the tool to the information in the ATR, and judge whether the tool passes or fails with regard to the relevant techniques. When all test files have been checked, the ATR allows the tester to save the results.

The lists of accessibility checks are linked from dynamically generated web pages. Since the HTML/XHTML test files are meant for submission to WAI, the description from the previous section also applies here.

The WCAG 2.0 Conformance Test Suite is hosted at ATRC until a suitable place at the WAI is found\(^{33}\); Chris Ridpath also says that the tests are currently released under the W3C license so they may be distributed and used by others\(^{34}\).

\(^{31}\) http://www.w3.org/TR/AERT.  
\(^{32}\) http://www.aprompt.ca/ATR/ATR.html; the ATR can be downloaded at http://www.aprompt.ca/ATR/GetAtr.html.  
\(^{34}\) Personal e-mail with Chris Ridpath, 4 April 2005.
4.1.6  Web-based Intranet and Internet Information and Applications Checklist

URL:

This is a small test suite maintained by Jessie Li (Sun) for the Mozilla Accessibility project. There is no information on the date of the last modification. The test suite is part of a larger set of “Test Cases of Section 508 Standards for Electronic and Information Technology”\(^{35}\) and contains tests for technical standard 1194.22 (for “Web-based intranet and Internet Information and Applications”). The suite contains a set of test case descriptions, with the corresponding test files being located on various servers (including web servers of mozilla.org, the National Center for Accessible Media (NCAM), CNN and others).

The test descriptions state the initial conditions for the test (e.g. “Have a screen reader installed”), the steps to be taken, the expected result and sometimes also HTML source code (for the test files hosted at mozilla.org). Since the test suite is meant for evaluating accessibility features of user agents, testing is done manually. There are no mechanism for automating tests.

Copyright for the testing resources resides with the Mozilla Foundation (for the files hosted at their website) or with other organisations for resources hosted elsewhere.

4.1.7  The HTML Challenge

URL:

This is a set of files produced by Freedom Scientific, manufacturer of the screen reader JAWS, to show “how JAWS® 5.0, when used with Microsoft Internet Explorer 5.5 and 6.0, gives you access to the many features of Web pages”. The test files were created in 2003 and “are designed for use in Internet Explorer 5.5 and 6.0, and may not display properly in other Web browsers”. The acknowledgements page\(^{36}\) states that the test suite was updated to JAWS 5.0 by Brian J. Walker of the Iowa Department of the Blind\(^{37}\). (Older versions, e.g. for JAWS 4.5, have existed; for example,


\(^{37}\)http://www.blind.state.ia.us/index.htm.
on 2 October 2002, Lloyd G. Rasmussen posted his findings on how Lynx and IBM Home Page Reader coped with lists to the WAI IG mailing list.\(^\text{38}\)

There are 13 test cases, each of which is an HTML 4.01 file describing a “challenge”, which means that each page demonstrates a feature of HTML and describes how JAWS users can use their software to benefit from the software's support for that feature. The 13 “challenges” are:

- **Navigation** (general navigation in a page, moving by headings, skipping past links, jumping to lines, moving by element),
- **Structure and format** (announcing heading level, the heading list dialogue, appearance of headings, long quotations, font information, etc),
- **PlaceMarkers** (temporary and permanent placemarkers, placemarkers list dialogue, sharing placemarkers),
- **Personalized Settings** (personalizing web settings, e.g. turning on/off graphics or heading announcements; sharing personalized settings),
- **Information and Help** (hot key help, context-sensitive help, the JAWS help system, adding help to a page, multiple HTML attributes that provide information, HTML element information),
- **Links** (moving through links, link types, links with title, graphics and link),
- **Language** (automatic detection of language changes in HTML and language switching for speech synthesisers),
- **Images** (images with alt text, images with title, images with long description, images with more than one attribute, graphics with no alternative text),
- **Lists** (simple and complex ordered lists, unordered lists),
- **Forms** (moving through forms, forms mode and form controls, form control prompts, prompts that are not associated with controls, accessibility features with HTML forms; this page also refers to other pages on 'prompts associated with form controls', 'forms and tab order', 'forms with access keys' and 'poorly designed forms'),
- **Tables** (commands for reading and moving within tables, moving between tables, nested tables, complex tables),
- **Frames** (quick key navigation and frames list dialogue for navigating frames; this page links to examples of frames without titles, frames with titles and inline frames),
- **Dynamic HTML** (refreshing content, dynamically editing HTML text,

\[^{38}\text{http://lists.w3.org/Archives/Public/w3c-wai-ig/2002OctDec/0010.html}^\text{.}
onclick and onmouseover).

The setup of this test suite is different from other test suites because each "challenge" file combines the functions of test case and test description. As a consequence, these files are never atomic tests. The pages on accessibility features of forms and the page on poorly designed forms follow the same approach. The only exceptions are the two demonstrations of frames; the demonstration of inline frames also combines the test description with the test case proper. Although the test cases can provide ideas and examples for BenToWeb's HTML test suites, the files as a whole would not be suitable for reuse, except for complex test cases.

Each file was expressly made for testing with JAWS and Internet Explorer; neither the generation nor the testing of these files can be subject to automation.

Freedom Scientific owns copyright of this test suite; it is not clear if materials can be reused if Freedom Scientific is properly credited.

### 4.1.8 Test Plan to Test HTML Elements for Accessibility


This is a test suite created by Dharma Sirnapalli (Netscape) on 26 June 2002. It is part of Mozilla's quality assurance activities. The test suite is probably intended for testing embedded versions of the Gecko rendering engine (on which Mozilla and FireFox are based)\(^\text{39}\), but this can only be inferred from the location of the test suite on the web server.

There are 19 test cases. The documentation does not refer to any set of guidelines (WCAG, Section 508 or other) against which resources are to be tested. The test suite is part of the "Test Plan for nsIAccessible Interface Attributes and Methods"\(^\text{40}\). The nsIAccessible interface is similar to IAccessible interface of MSAA. The tests are written in HTML and JavaScript and required some specific change to the setup of Mozilla. The tests can be run automatically in Mozilla with the XUL Framework, however, the link provided to this framework is out of date.

There is no copyright statement, but it is assumed that the Mozilla Organization owns copyright of these resources.


4.1.9  Flash Test Suite


This is a test suite created by Bob Regan (Macromedia) and Andrew Kirkpatrick (National Center for Accessible Media). In mid 2003 Bob Regan and Andrew Kirkpatrick started documenting how various screen readers were handling Flash content. The test suite consists of several categories addressing specific topics. The suite may also help developers who want to explore specific issues of Flash accessibility. On 13 July 2004, a set of seven test cases for buttons and movie clips coded as buttons were released, together with test results for JAWS 4.50. Since that time, no new test cases or test results have been published. However, a blog post from 28 March 2005 lists publicly available examples of accessible Flash, and a blog post from 25 March links to a draft on “WCAG 1.0 Techniques for Flash” (unfortunately only available in Flash).

Testing screen reader support for Flash is a manual process, so there are no resources that allow an automated evaluation of the test cases.

The test cases are hosted on www.markme.com, a web site that hosts weblogs by Macromedia employees. The site contains no copyright statement; Bob Regan says that the test files are informal in nature and can be reused.

4.1.10  Screen Reader Visibility Testing


This set of test was created by Bob Easton and last modified on 13 September 2003. It consists of seven specific tests of screen reader behaviour. Each of the tests uses an HTML or CSS technique (display:none; visibility:hidden, etc); users of screen readers can check whether invisible content is spoken by their user agent. The conclusion is that content that is hidden visually is almost always hidden from screen readers too.

Testing screen reader support for various techniques to make content invisible is a manual process, so there are no resources that allow an automated evaluation of the test cases.

---

44 Personal e-mail with Bob Regan.
4.1.11 Conclusion

The table below summarizes the findings in this chapter. “Scope” refers to what the suite is meant to test (software or content).

<table>
<thead>
<tr>
<th>Test Suite</th>
<th>Scope</th>
<th>Status</th>
<th>Approximate size</th>
<th>Last modified</th>
<th>Automation</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>W3C UAAG 1.0/HTML 4.01</td>
<td>User agent</td>
<td>Draft</td>
<td>124</td>
<td>2004</td>
<td>Some</td>
<td>W3C</td>
</tr>
<tr>
<td>W3C UAAG 1.0/SVG 1.1</td>
<td>User agent</td>
<td>Draft 4 / 2 (SVG)</td>
<td>2004</td>
<td>Some</td>
<td>W3C</td>
<td></td>
</tr>
<tr>
<td>CITA UAAG 1.0/HTML 4.01</td>
<td>User agent</td>
<td>Draft?</td>
<td>145</td>
<td>?</td>
<td>No</td>
<td>UIUC</td>
</tr>
<tr>
<td>CITA UAAG 1.0/SVG 1.1</td>
<td>User agent</td>
<td>Draft?</td>
<td>0</td>
<td>?</td>
<td>No</td>
<td>UIUC</td>
</tr>
<tr>
<td>CITA UAAG 1.0/SMIL 2.0</td>
<td>User agent</td>
<td>Draft?</td>
<td>66</td>
<td>?</td>
<td>No</td>
<td>UIUC</td>
</tr>
<tr>
<td>W3C WCAG 2.0/HTML (X)HTML</td>
<td>Draft</td>
<td>24 (+141)</td>
<td>2005</td>
<td>No</td>
<td>W3C</td>
<td></td>
</tr>
<tr>
<td>ATRC Web Checker</td>
<td>(X)HTML</td>
<td>Draft?</td>
<td>340 (+)</td>
<td>2005?</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Mozilla Section 508 checklist</td>
<td>Web-based information and applications</td>
<td>Final? (12+)</td>
<td>?</td>
<td>No</td>
<td>Mozilla and others</td>
<td></td>
</tr>
<tr>
<td>Mozilla Embed HTML tests</td>
<td>?</td>
<td>Final?</td>
<td>19</td>
<td>2002</td>
<td>Yes</td>
<td>Mozilla</td>
</tr>
<tr>
<td>HTML Challenge</td>
<td>Assistive technology</td>
<td>Final</td>
<td>20</td>
<td>2003</td>
<td>No</td>
<td>Freedom Scientific</td>
</tr>
<tr>
<td>Flash Test Suite</td>
<td>User agent</td>
<td>Incomplete</td>
<td>7</td>
<td>2004</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Screen reader visibility</td>
<td>User agent</td>
<td>Final?</td>
<td>7</td>
<td>2003</td>
<td>No</td>
<td>?</td>
</tr>
</tbody>
</table>
Table 1. Summary of accessibility-related test suites

Most of the test suites described in this chapter were made to evaluate the accessibility of user agents, or user agent support for accessibility features of certain types of web content. Most of them are still in draft stage. Only two test suites have mechanisms for generating test files, but only one test suite has metadata or another mechanism that allows a tester to run the entire test suite at once. Only one test suite comes close to the goal of using a test suite for benchmarking E&R tools (Chris Ridpath’s Access Tool Reviewer; see section 4.1.5). Materials from some of the test suites would be suitable as a starting point for benchmarking purposes.

There is some concern in the Web community about the status that the test suites for WCAG 2 may have. On 23 August 2004, Matt May, one of the “participants in good standing” of the WAI GL Working Group and W3C team member, posted the following comment on his weblog:46

And while I’m at it, let me say this to the folks who are demanding bulletproof, objective, testable criteria from the next version of WCAG: this is not what you want. You do not want any authority to state that accessibility is as easy and clean as a mathematical formula. You do not want to be forced to test your writing against Fog indexes, or have entire technologies circumscribed, just for the sake of your own convenience. Any normative test suite for Web accessibility would stifle the creative process of every designer by its very nature, and that’s a worse fate than you could ever imagine. (And much worse for users, as well.) Web designers are much better off using tools that help them create accessible content, and learning the problems and solutions involved in accessibility. WCAG 2.0 is, in my opinion, a step in the right direction.

The perception of a test suite as something normative is justified by certain statements in the “HTML Test Suite for WCAG 2.0” document47. This document states that the test suite “is the only method for testing conformance to WCAG 2.0” and that “If HTML Web content fails one or more of these tests then it does not conform to the WCAG 2.0”. Since a test suite is usually thought of as a set of test cases (usually files) accompanied by metadata (test descriptions etc) instead of a checklist of tests accompanied by example files (which is what the current draft of the test suite looks like), readers might conclude that the set of test files is normative instead of the checklist.

---

47 [http://www.w3.org/WAI/GL/WCAG20/tests](http://www.w3.org/WAI/GL/WCAG20/tests).
4.2 Test Suite Architecture

4.2.1 Introduction

Before discussing test suite architecture, it is useful to introduce some definitions. According to the W3C Quality Assurance glossary\textsuperscript{48}, a test suite is

A set of documents and tools providing tool developers with an objective methodology to verify the level of conformance of an implementation for a given standard.

A test suite contains test cases, a test case being defined as

An individual test that corresponds to a test purpose, which in turn maps back to the assertion(s), and finally the spec

or

Documentation specifying inputs, predicted results, and a set of execution conditions for a test item. Syn: test case specification. [IEEE Std. 100-1992]

Test cases may be atomic or more complex. An atomic test is

A test case that tests a single rule from the specification and maps back to exactly one assertion. This is in contrast to some test cases that may test a combination of rules.

For each test in a test suite, there should be a test purpose:

An explanation of why the test was written, and must map directly to one or more test assertions.

A test assertion is

A statement of behavior, action, or condition that can be measured or tested. It is derived from the specification’s requirements and provides a normative foundation from which test cases can be built.

A test suite may also have a test harness or a test driver:

A software module used to invoke a module under test and, often, provide test inputs, control and monitor execution, and report test results. [IEEE Std. 100-1992]

\textsuperscript{48}http://www.w3.org/QA/glossary.
Not all W3C specifications can be objectively tested for conformance. If a specification cannot be objectively tested, then an alternate approach to conformance testing should be used to verify whether a product faithfully implements the specification (see [W3C 2002a], which is work in progress). In this regard, it is worthwhile to refer to the discussion on the mailing lists of the WAI Interest Group and the WAI Guidelines Working Group in August 2004. Randal Rust sent the following proposal:

WCAG should be divided into Guidelines, which can be measured and tested, and Suggested Best Practices, which can only be tested by a person.49

The rationale was the following:

The Guidelines should deal strictly with W3C Technologies, so that vendors can be left to ensuring the accessibility of proprietary technologies such as Shockwave and PDF. Vendor technologies can then be addressed in the Suggested Best Practices. Other items, such as clarity of content, should also move out of Guidelines.

I propose this because WCAG Guidelines must be measurable and quantifiable. There can be no gray areas, otherwise it makes it too difficult to make a business case for accessibility. The measurable Guidelines must work entirely in concert with other W3C publications, such as HTML, XHTML, CSS and DOM. Moving outside of the W3C realm only causes confusion, frustration and, ultimately, ignorance of Accessibility Guidelines. (...)

In order for WCAG to gain greater acceptance, its Guidelines must be quantifiable. Developers and designers must be able to validate their pages and get clear-cut results, just like with HTML validation.

Some WAI IG members found this a clever proposal, while others found the proposed division irrelevant, e.g. arguing that "most of accessibility is like usability can't be automatically 'measured' but needs to be manually tested and evaluated"50. However, Randal Rust argued that some WCAG 1.0 Guidelines were more about usability than accessibility. Although usability is dependent on accessibility, it may be possible to test these two characteristics separately51. On the mailing list of the WAI Guidelines Working Group, Jason White pointed out that the main problem with the suggested division is that there is so little that can be reliably verified automatically, so "If satisfaction of machine testable requirements were

50Jesper Tverskov: http://lists.w3.org/Archives/Public/w3c-wai-ig/2004JulSep/0404.html.
51Randal Rust: http://lists.w3.org/Archives/Public/w3c-wai-ig/2004JulSep/0425.html.
allowed as the basis of a conformance claim, then this would be meaningless as a statement about the accessibility of the content.” The WCAG 2.0 guidelines should not deal exclusively with W3C technologies: their intention is to be technology-independent [W3C 2002b]; only the Techniques for WCAG 2.0 are technology-specific, while the development for techniques for other technologies is the responsibility of external parties.

If the division proposed by Randal Rust were possible and useful, there could be a similar division in the test suites for WCAG, and benchmarks might make a distinction between how a web site evaluation tool performs on both types of accessibility problems. However, this division was not accepted by the WAI GL Working Group, so making this division in the BenToWeb test suites would only benefit benchmarking purposes but not the stated purpose of the current draft of the HTML Test Suite for WCAG 2.0.

4.2.2 Test Suites and the W3C Quality Assurance Framework

In 2002, W3C Quality Assurance published a first working draft on test suites: QA Framework: Test Guidelines (W3C Working Draft 20 December 2002)\(^{54}\), but since 2004, progression of this working draft is on hold\(^{55}\). W3C Quality Assurance started a discussion on test suite architecture on a Wiki\(^{56}\). This Wiki was started with the goal of building a community view on some QA-related topics; its content is not endorsed or maintained by the QA WG\(^{57}\). According to the current state of the Wiki, a test suite minimally comprises:

- a set of test cases
- a description of how they are supposed to be used
- a description of the objectives, scope of the test suite
- an identification of the Recommendation(s) or specification(s) being tested

In addition, a test suite may also comprise:

- a test execution framework (test harness?)

---

\(^{52}\)http://lists.w3.org/Archives/Public/w3c-wai-gl/2004JulSep/0429.html.

\(^{53}\)http://www.w3.org/WAI/GL/WCAG20/tests/.

\(^{54}\)http://www.w3.org/TR/2002/WD-qaframe-test-20021220/.


\(^{56}\)http://esw.w3.org/topic/TestSuiteArchitecture.

- a test reporting system
- a test case management system
- test suite design
- an adjudication process – how to handle inquiries and/or disputes
- disclaimers
- branding (icon) criteria

Dominique Hazaël-Massieux of the QA WG provided the following (provisional) description of a test execution framework.

Test cases are often accompanied with test case metadata. A test case management system should use these metadata to offer different views of the state of the test suite, and to bundle sets of test cases adapted e.g. to a given profile. Possible implementations could use or be based on Bugzilla Test Runner\(^{58}\) (Open Source) or MUTAT\(^{59}\) (for human-centred testing).

The adjudication process for W3C test suites may be described in a Quality Assurance Process Document (QAPD). Writing a QAPD is required by guideline 4 (“Define the QA process”) of the Operational Guidelines\(^{60}\) of the W3C QA Framework (which is, admittedly, only work in progress). Examples of QA Process Documents are the DOM Conformance Test Suites Process Document\(^{61}\), the W3C XML Schema Test Collection document\(^{62}\).


\(^{59}\)http://www.w3.org/QA/Tools/MUTAT/.

\(^{60}\)http://www.w3.org/TR/2003/CR-qaframe-ops-20030922/.


\(^{62}\)http://www.w3.org/2001/05/xmlschema-test-collection.html.
the XML Conformance Test Suites Document\(^{63}\), the document “W3C SOAP Version 1.2 test collection - How to contribute”\(^{64}\) and the Conformance Test Process For WCAG 2.0\(^{65}\). These documents include a description of procedural issues, such as submitting tests or test suites, receiving and reviewing tests and reevaluating tests.

Not mentioned in the Wiki, but worth having, is a set of test case authoring guidelines, such as the “CSS2.1 Test Case Authoring Guidelines” of the CSS Working Group\(^{66}\).

The CSS Test Suite Documentation also contains a section on writing a (CSS) test suite, with templates for navigation pages, section pages and a cover page.\(^{67}\)

### 4.2.3 The DOM Test Suite Methodology

The DOM test suite (DOM TS) deserves a separate discussion because it was one of the first W3C test suites to incorporate automation. Dimitris Dimitriadis, one of the members of the DOM Working Group and of the Quality Assurance Working Group, wrote a report that documents the DOM TS methodology [Dimitriadis 2004].

DOM (Document Object Model)

is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page. [W3C DOM]

The Document Object Model provides a structured, object-oriented presentation of the individual elements and content in a document with methods for retrieving and setting the properties of those objects. It also provides an interface for dealing with events, allowing developers to capture and respond to user actions. Currently, there are three “versions” of W3C DOM: DOM Level 1 (1998; 2nd edition 2000), DOM Level 2 (2000, 2003) and DOM Level 3 (2003, 2004)\(^{68}\). Each level adds more features to the previous one.


\(^{65}\) [http://www.w3.org/WAI/GL/WCAG20/tests/ctprocess.html](http://www.w3.org/WAI/GL/WCAG20/tests/ctprocess.html).

\(^{66}\) [http://www.w3.org/Style/CSS/Test/guidelines.html](http://www.w3.org/Style/CSS/Test/guidelines.html).

\(^{67}\) [http://www.w3.org/Style/CSS/Test/testsuitedocumentation.html](http://www.w3.org/Style/CSS/Test/testsuitedocumentation.html).

\(^{68}\) The DOM specifications consist of several sections, which don’t necessarily reach Recommendation status simultaneously. For an overview of DOM Technical Reports and their exact dates, see [http://www.w3.org/DOM/DOMTR](http://www.w3.org/DOM/DOMTR).
The DOM specification (i.e. the actual technical report) is written using the W3C XML specification DTD (“XMLspec”)\(^{69}\). Since DOM is an interface and not a (programming or scripting) language, its interfaces and methods are defined in IDL (Interface Definition Language)\(^{70}\). DOM is implemented in particular environments by means of bindings; W3C defines normative bindings only for Java and ECMAScript\(^{71}\) (the standard on which JavaScript is based). Other organisations are free to define bindings for other languages\(^{72}\).

One of the intentions of the DOM TS activity was to make it as easy as possible to write tests. Since the DOM specification defines two bindings, it was decided that the test framework should allow for writing one test case and testing many implementations, i.e. “to write a test case in a neutral language and generate the different tests for particular bindings, instead of writing multiple test cases, one for each binding” [Dimitriadis 2004, p. 7].

Generating tests was a four-step process that brought some limitations with it. The steps are described below.

1. Write a test case using a “smart” markup language. Such a language did not exist, so it was decided to use a language similar to that which the DOM specification itself uses. This language was extended for metadata (creation date, author, pointer to the relevant part of the specification, etc). Adding all specific information to the markup was a time-consuming and difficult process.

2. Validate the test case using the original XML version of the DOM specification.

3. Generate the relevant language versions of the test using XSLT transformations for limiting ambiguity. If the test was valid, the transformation would generate valid test code. The correctness of the XSLT stylesheets was crucial in this step.

4. Provide a framework to build the test suite, run the tests, report an implementation’s success or failure, generate an overview of results, provide links to the test cases themselves, provide links to the part of the specification being tested, etc.


\(^{70}\)IDL is a language defined by the Object Management Group (OMG), the organisation that also defined the Uniform Modeling Language (UML). For IDL, see http://www.omg.org/gettingstarted/omg_idl.htm. OMG IDL was published as an international standard: ISO/IEC 14750:1999.


\(^{72}\)See Document Object Model (DOM) Bindings: http://www.w3.org/DOM/Bindings.
The architecture of the test suite consists of a number of components, which will be described below.

1. The **Test Case Description Language** (TCDL) enables the description of test cases independently from the test instances (which would be in a particular language). The TCDL is generated from the DOM XML Specification using XSLT. **Test cases** are validated against the XML version of the DOM specification.

2. **Tests are transformed** into executable code (test instances in ECMAScript and Java) using XSLT. This code is run in two **frameworks**: JSUnit for ECMAScript, and JUnit for Java.

3. A **test case table** is generated using XSLT. This table contains a pointer to each test, the part of the specification tested by the test, the test description and a pointer to both the JavaScript and Java versions of the test.

4. The DOM TS Group used CVS (Concurrent Versions System) for **test case versioning and storage**. The tests are stored in a file hierarchy that reflects the structure of the DOM specification (levels, each consisting of modules).

The advantages of the DOM TS methodology are clear:

- the easy creation of tests for multiple languages in one test suite, and
- the ability to check the coverage of the specification.

Like DOM, the Web Content Accessibility Guidelines are a specification that can be “bound” to several languages (XHTML, SVG, ...), by means of the Techniques Documents. However, there are crucial differences between DOM and WCAG that make the DOM TS methodology unsuitable for the development of the WCAG test suites. Dimitriadis discusses three requirements or limitations of the DOM TS methodology.

- The requirement to use a rich markup language to describe tests.
- Specifications that have been written in a functionally rich markup language. Tests written in plain HTML are not suitable for automated processing.
- “The DOM TS and similar frameworks work best for interface or function style specifications (DOM, XSLT and the like) and not equally well for prose-based, explicitly user-oriented specifications (WAI, for instance), as these are not supported by an agreed-on markup language which would be used to describe the specification’s intended behaviour.” [Dimitriadis 2004, p. 19]
During a teleconference on the UAAG test suites in 2002\textsuperscript{74}, Lofton Henderson (former co-chair of the Quality Assurance WG) had already pointed out that the generation of test suites for specifications like SVG and UAAG might be hard. The same is true for WCAG test suites:

- There is no rich markup language for describing behaviour and writing tests. Since WCAG 2.0 (including the accompanying Techniques documents) does not specify an API or a programming language, this requirement can probably not be satisfied.
- WCAG 2.0 Techniques documents are written using an XML grammar, but it is not clear if this grammar would allow for test generation.

### 4.3 Test Suites as Unit Tests: Benefits and Drawbacks

Test cases in the current WCAG test suite each address one specific accessibility problem. For each checkpoint or success criterion there is (or will be) one test case that passes and at least one that fails. Since these test cases are atomic tests, running the complete test suite through an evaluation and repair (E&R) tool, is similar to running a suite of unit tests: each test functions as a small sanity check on the code that evaluates the compliance to the checkpoint or success criterion for which the test was written.

However, a set of atomic tests cannot cover all the Web Content Accessibility Guidelines, because some guidelines address consistency of certain features (style, navigation, language, ...) of a web site. In many web sites, these features are implemented with code that is only written once but imported into many pages: links to style sheets, server-side includes for navigation menus, etc. An error in an imported piece of code can corrupt many pages and make them inaccessible. In a situation like this, an E&R tool will probably report as many errors as the number of affected pages, even though the root cause is only one single error. The distinction between many different errors and one repeated error cannot be made if web pages are treated as “units”. JUnit, the first unit testing framework ever developed, also has this shortcoming, which Cédric Beust calls the “failure cascade trauma” [Beust 2004]. To address this problem, Beust developed a testing framework (TestNG\textsuperscript{75}) that features a mechanism to accurately report failures due to a dependency that is not satisfied. A similar mechanism would be useful in E&R tools, if it does not already exist.

\textsuperscript{74}http://lists.w3.org/Archives/Public/w3c-wai-ua/2002JulSep/0035.html.
\textsuperscript{75}http://www.beust.com/testng/.
5 Future Work

The previous sections have reviewed the state of the art in regard to test suites related to Web specifications and accessibility guidelines. As it can be seen, there is a big gap between the existing materials and the objectives of BenToWeb that will be filled under the scope of this WP and in collaboration with WPs 3 and 5.

We need to focus our efforts not only on the definition of the test unit files for the selected technologies from the accessibility standpoint, but also on the verification tools that will allow to automate to the biggest possible extent the validation process.

It is also critical in this process to refine the report format. In spite of the limitations of EARL\textsuperscript{76} (see D5.1 for further details), the selected format must be based on it, and try to accommodate the needs of our test suites.

For the test suites development process, it is important to define a method to store metadata for each test case and for keeping these metadata synchronised with the test cases. These metadata should also support the description of complex test cases, i.e. test cases that consist of multiple files.

Since BenToWeb will develop not one but several test suites, it is desirable to divide the work between project partners in a way that enables a learning process that takes “lessons learned” from one test suite to the next. This learning process is important in view of D4.7 (Guidelines for the development of test suites), which should be produced by the end of the project.

\textsuperscript{76} Evaluation and Report Language 1.0 (http://www.w3.org/TR/EARL10/).
6 References


Appendix A: Test Suites Not Covered in This Report

There are many collections of test files that are not hosted at W3C and/or that are too small to be considered as test suites. This appendix provides a non-exhaustive list, organised by technology. Other lists of test sites or test suites are available at http://www.robinlionheart.com/stds/html4/links (Robin Lionheart’s web site) and http://www.hixie.ch/tests/evil/mixed/home.html (Ian Hickson’s Evil Test Suite), but these are less up to date.

7.1 CSS


- Another non-W3C test suite for CSS 2 is available at http://dbaron.org/css/test/ (by David Barron, invited expert to the CSS WG, contributing to CSS3).

- RichInStyle.com test suite: http://www.richinstyle.com/test/. This test suite claims to test every aspect of CSS2. The same website also has documentation on bugs in the CSS support of Internet Explorer (versions 3-5), Netscape 4, Opera 3 and 4 and Mozilla 5: http://www.richinstyle.com/bugs/.


- Index of /css-test: http://www.kitchenquest.com/css-test/ (a set of HTML files with embedded CSS rules, without test assertions or any
other description)

- **Index of /tests/adhoc/css**: [http://www.hixie.ch/tests/adhoc/css/](http://www.hixie.ch/tests/adhoc/css/) (by Ian Hickson, who is an Invited Expert of the W3C CSS Working Group and co-editor of several CSS specifications; for some background see [http://www.damowmow.com/playground/css-testing.html](http://www.damowmow.com/playground/css-testing.html); see also Hickson's Evil Test Suite below).


- **CSS1 box/float/clear tests / The Box Acid Test**: [http://style.cleverchimp.com/boxacidtest/](http://style.cleverchimp.com/boxacidtest/). By Todd Fahrner; “a composite test of the CSS-1 box model as applied to HTML 4.

### 7.2 HTML/XHTML

- **Index of /tests/adhoc/html**: [http://www.hixie.ch/tests/adhoc/html/](http://www.hixie.ch/tests/adhoc/html/) (by Ian Hickson, who notes: “It's difficult to test HTML given how loose the spec is!”).


- **HTML Test Suite at**: [http://www.math.ucla.edu/~jimc/html40-test/](http://www.math.ucla.edu/~jimc/html40-test/), by James F. Carter, 1998-02-01. “The HTML Test Suite provides sample material exercising all the tags in HTML 4.0 and many of the attributes on those tags. It is intended to aid a developer in debugging a User Agent (web browser) which presents documents written in HTML 4.0.”

- **OBJECT test suite**: [http://www.student.oulu.fi/~sairwas/object-test/](http://www.student.oulu.fi/~sairwas/object-test/) (by Antti Näyhä): "This unofficial test suite is based on the section Objects, images and applets of the HTML 4.0 Specification. The specification itself is sometimes ambiguous, and the 'correct
renderings’ of the tests are presented as interpreted by the author." The test suite is work in progress and was last updated on 16 February 1999. The test suite also has a results page (http://www.student.oulu.fi/~sairwas/object-test/results/) that compares OBJECT implementations for Lynx 2.8, MSIE for Windows 3.x, 4.x and 5.0b2, MSIE for Mac 3.x and 4.x, Mozilla NGLayout (Nov '98, Feb '99) for Windows and Netscape Navigator 4.x (Unix, Mac, Windows).

- iCab Browser-Test: http://www.icab.de/test.html (a single web page containing a number of HTML constructs for testing a browser; in German).
- The “Break this page” experiment (by Joe Clark and Aaron Doust): http://www.joeclark.org/breakthispage.html. Describes and links to four test files that use various techniques to present alternative text for images. Subscribers to various mailing lists were asked to load the four test files in every conceivable combination of platform, browser, graphics mode and assistive technology. In his article “The Glorious People's Myth of Standards Compliance” (http://www.joeclark.org/glorious.html), Joe Clark had complained about false claims to standards compliance regarding Netscape 6 en Internet Explorer 5 for MacOS, because these browsers do not support longdesc and other accessibility-enhancing features of HTML 4.

Accessibility-related test files (most are referenced from http://www.w3.org/WAI/GL/wcag20.html):

- Tests for User Agent Support Page: http://www.w3.org/WAI/GL/tests/UA-support.html (last updated 8 November 2000; for testing combinations of user agent, assistive technology and operating system).
- Div, tabindex and accesskey (by Cynthia Shelly): http://www.w3.org/WAI/GL/tests/divs.html (a single page).
- Specifying font sizes (by Wendy Chisholm):
http://www.w3.org/WAI/GL/tests/font-sizes.html (last updated 8 November 2000; tries to determine whether certain methods prevent a browser from resizing the text).

- Tests for images (by Wendy Chisholm): http://www.w3.org/WAI/GL/tests/image.html: to test what happens if you link to an image if the browser has images turned off (last modified 8 November 2000 or February 2000?).

- Keyboard access to forms (by Wendy Chisholm): http://www.w3.org/WAI/GL/tests/keyboard-access.html (last modified 8 November 2000 or June 2000?).

- Broken or abused markup: http://www.w3.org/WAI/GL/tests/missing-id.html; test broken or abused markup to see how it will pass through validators (author and date unknown; last modified 8 November 2000).

- Test for navigation bars: http://www.w3.org/WAI/GL/tests/navbar.html: use of MAP to group links into a navigation bar (by Wendy Chisholm; last modified 8 November 2000).


- Style sheets and graceful transformation (one does, one doesn't): http://www.w3.org/WAI/GL/tests/ss-tg.html: (author unknown; last modified 8 November 2000).

- Creating Structured Tables (by a member of the WAI Interest Group who identifies himself as Mike [Isofarro]; see his post to the WAI IG mailing list 30 August 2003): http://www.isolani.co.uk/articles/structuredTables.html (article with four examples of code for tables, with links to HTML files that demonstrate these tables).

### 7.3 Other XML Vocabularies

- Mozilla SVG Samples: http://www.croczilla.com/svg/samples/.


- SMIL 2.0 Test Suite: http://www.w3.org/2001/SMIL20/testsuite/ (the test suite can be downloaded as a single ZIP file of 25 MB).

- SMIL 2.0 Testcase for SMIL Interoperability Test:

  [http://lists.w3.org/Archives/Public/w3c-wai-ig/2003JulSep/0374.html](http://lists.w3.org/Archives/Public/w3c-wai-ig/2003JulSep/0374.html).
http://smil.nist.gov/Testcase2.html (March 2001, W3C SYMM Working Group, by Wo Chang of NIST/ITL. The test suite can be downloaded as ZIP files.)

- SMIL 2 Interop Tests:
  http://www.oratrix.com/DemoSet/SMILTests?get=linking (Oratrix is the developer of GRiNS, a SMIL player.)

- SMIL 2.0 Testsuite:
  Test reports in French are available at

- XML Schema Test Collection:
  http://www.w3.org/2001/05/xmlschema-test-collection.html (the test collection can be downloaded as a compressed archive).

- MathML Test Suite 2.0: http://www.w3.org/Math/testsuite/ (can be downloaded as a zip-file).

- MathPlayer MathML Test Suite (superset of the official W3C MathML 2.0 Test Suite):

- VoiceXML Forum Platform Certification Program: test suite for W3C VoiceXML 2.0 (there is a public and a member-only version):
  http://www.voicexml.org/platform_certification/index.html. A list of certified platforms (for VoiceXML Forum Test Suite v1.0, 07 Sept. 2004) is available at
  http://www.voicexml.org/platform_certification/certified_platforms.html. This test suite is different form the W3C's VoiceXML 2.0 Test Suite at

- NIST (National Institute of Standards and Technology): Software Diagnostics and Conformance Testing Division: Web Technologies:
  http://xw2k.sdct.itl.nist.gov/brady/xml/index.asp. This page contains links to downloadable archives for the NIST XML Query Tests, the W3C/NIST XML Tests, the W3C/NIST XInclude Tests, the NIST Schema Tests, XSL-FO tests and XSLT/XPath Tests.

- Sarvega XSLT and XML Validation benchmarks:
  http://www.sarvega.com/all_benchmarks.php. (Originally developed for internal use and subsequently put in the public domain. The XML Validation benchmark uses a collection of pre-existing XML files and tests the performance of a number of XML parsers. The XSLT benchmark tests the performance of several XSLT engines with fifteen real-life stylesheets.)

- The Wireless FAQ – Demos:
http://www.thewirelessfaq.com/demos.asp. (These demos require a WML-capable browser. To use the demos from a WAP device, go to http://www.thewirelessfaq.com/demos.wml. The demos are the collective property of the WMLProgramming and OUI-support mailing lists.)

- DocBook test suite: available in the CVS repository of the DocBook project at SourceForge: http://cvs.sourceforge.net/viewcvs.py/docbook/ (under testdocs/tests). The test suite is used by the DocBook Technical Committee for testing the integrity of new versions of the DTD, but is not considered as a formal test suite. It is not rigorously maintained, and there is no formal process of certifying that a new DTD version passes the test suite. It is one tool used by the DocBook Technical Committee, among many tools used for testing. The test suite is also informally used for testing the DocBook stylesheets that are developed on the SourceForge site. Since the test suite is publicly available, other people may be using it for testing their tools or stylesheets, but the DocBook Technical Committee or the DocBook project do not try to track such usage.  

7.4 Test Suites for Web Clients/Browsers

Some test suites are not limited to one specification but test the capabilities of web browsers.

- The Evil Test Suite / Ian Hickson’s Evil Test Suite: http://www.hixie.ch/tests/evil/mixed/home.html (This test suite has two sections: evil tests and wet blanket tests: “The wet blanket tests are those that are so basic that everything should pass them (unfortunately, even some of these tests are failed by some browsers). In time, more and more of the evil tests will become wet blanket tests, as the standard of browsers improves.” The test suite does not limit itself to HTML, but also tests CSS and sometimes HTTP. Some tests examine the co-operation of all three. Ian Hickson warns that “the HTML tests do not test actual compliance with the letter of the spec, but rather examine whether UAs (web browsers) support the spirit of the spec.” The test suite was last updated in March 1999. The test results are no longer available.)

- Pete’s Web Browser Tests (by Peter K. Sheerin): http://petesguide.com/WebStandards/tests/ (Peter Sheerin notes that his test pages “are designed to be torture tests of browsers,

78 http://groups.yahoo.com/group/wmlprogramming.
79 http://groups.yahoo.com/group/oui-support.
80 Personal e-mail with Bob Stayton.
mainly to demonstrate advanced HTML, XHTML, and XML markup that is desperately needed to work well and consistently, but which breaks on many browsers, and is not adequately highlighted elsewhere. A few of the examples also show bugs caused by incorrect coding practices. “There are tests for entities, character sets, object tags, PNG, tables, tool tips, style sheets with Unicode Byte Order Mark, typesetting characters, XHTML MIME types and XHTML+MathML+SVG. The home page and test pages were last modified on 2 September 2004, but the test pages contain an error message: “[an error occurred while processing this directive]”.

- MauveCloud’s Browser Tests (by Brian McCloud): http://www.mauvecloud.net/tests.html (Character set tests, HTML tests for HTML 2.0, 3.2 and 4.01, image file format tests, sound file format tests, colour tests and several sets of JavaScript tests.)

- NGLayout Test Cases: http://www.mozilla.org/newlayout/testcases/. These are test cases for the Mozilla Project. There are stress tests, Cascading Style Sheet tests, Document Object Model tests, General Layout tests (HTML 4: phrasal elements, lists, quoting, table columns, table column proportional widths, horizontal rules, character entities), HTML 4.0 Forms tests, Printing tests, and a link to Antti Näyhä’s OBJECT test suite.

- i-Bench (by VeriTest and PC Magazine): http://www.etestinglabs.com/benchmarks/i-bench/default.asp (“i-Bench is a comprehensive, cross-platform benchmark that tests the performance and capability of Web clients as they take on the latest Web technology and features.” i-Bench 5.0 was released on 17 November 2003; VeriTest and PC Magazine no longer support the benchmark. i-Bench is not offered as an online application but anyone can run it over their own local area network. i-Bench 5.0 is a 70 MB self-extracting archive that can be downloaded from PC Magazine’s website. A license agreement is only available for version 4.0, however.)

- Network Computer Test Suite: http://www.opengroup.org/testing/testsuites/vsnetcom.htm (By The Open Group. The test suite assessed conformance to the rendition aspects of the Network Computer Product Standard and included tests for HTTP 1.0/1.1 and HTML 3.2. The test suite used to be located at http://www.opengroup.org/nc but is no longer available.)

81 http://www.etestinglabs.com/benchmarks/i-bench/license.asp?visitor=
7.5 Character Sets and Entities


- Unicode test material (by Alan J. Flavell): http://ppewww.ph.gla.ac.uk/~flavell/unicode/ (According to Alan Wood\(^{82}\), these are test materials for Unicode 3.0.0. Alan Flavell’s introductory pages was last changed in November 2004; test pages were last modified in April 2005.)

- Browser Tests of Entities in 2004 (by Nancy): http://www.santagata.us/characters/CharacterEntities.html (The tests were run on various browsers on Windows and MacOS.)

- Character entity references in HTML 4.0: http://www.mozilla.org/newlayout/testcases/layout/entities.html (by David Gardiner; contributed to the NGLayout Test Cases of the Mozilla Project at http://www.mozilla.org/newlayout/testcases/layout/; see previous section of this appendix).

- HTML 4.0 special characters – test files: http://web.yost.com/computers/htmlchars/ (by Dave Yost; last modified in October 1999).


7.6 Other Types of Content or Applications


- XHTML Test Page: http://www.robinlionheart.com/stds/xhtml/ (by Robin Lionheart; this page allows you to load the same page with different MIME types).


\(^{82}\)Alan Wood’s Unicode Resources: http://www.alanwood.net/unicode/.


• MauveCloud's Core JavaScript 1.0-1.5 Tests: [http://www.mauvecloud.net/jscore/index.html] and other JavaScript tests are linked from [http://www.mauvecloud.net/tests.html].

• Flash Lite Test Suite, included in the Flash Lite SDK: see [http://www.macromedia.com/devnet/devices/articles/msvg_t.html].

8 Appendix B: Tools

For the implementation and verification of the test suites (e.g. for comparison of test reports), several tools will be used and evaluated. As presented in previous sections, the most likely approach will be based on a set of unit tests. In particular, there are two tools that will likely be used: JUnit and XmlUnit, and most likely in combination.

8.1 JUnit

JUnit\(^{83}\) is a simple and Open Source framework to write repeatable tests (unit tests). A unit test is a piece of code written by a developer that exercises a very small, specific area of functionality of the code being tested.

8.2 XmlUnit

XmlUnit\(^{84}\) is Java library that enables JUnit-style assertions about the content and structure of XML documents or transformations of them. Comparing XML documents is not a trivial task as it is a complex structure with elements, attributes, comments, entities, etc. in which order and blanks and order might or might not play a role. It is also important to be able to compare content of elements and attributes.

XMLUnit extends JUnit and allows to make assertions about:

- The differences between two pieces of XML
- The validity of a piece of XML
- The outcome of transforming a piece of XML using XSLT
- The evaluation of an XPath expression on a piece of XML
- Individual nodes in a piece of XML that are exposed by DOM Traversal

XMLUnit can also treat HTML content as XML to allow these assertions to be made about this type of documents.

\(^{83}\) http://junit.org/.

\(^{84}\) http://xmlunit.sourceforge.net/.