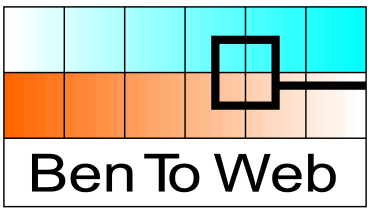
Benchmarking Tools and Methods for the Web (FP6—004275)



Sixth Framework Programme
Information Society Technologies Priority

D4.5 SVG & XForms Accessibility Issues

Contractual Date of Delivery to the EC:	30 September 2007 + 45 days
Actual Date of Delivery to the EC:	29 November 2007
Editor:	Christophe Strobbe (K.U. Leuven)
Contributors:	Christophe Strobbe (K.U. Leuven), Gerhard Weber (TU Dresden), Carlos A Velasco, Johannes Koch (FIT)
Workpackage:	WP4
Security:	Public
Nature:	Report
Version:	H
Total number of pages:	24

Keywords: Scalable Vector Graphics, SVG, XForms, accessibility, Web Content Accessibility Guidelines, WCAG, Web Accessibility Initiative, WAI.

DOCUMENT HISTORY			
Version	**Version date**	**Responsible**	**Description**
A	30 August 2007	K.U.Leuven	First TOC and draft
B	15 September 2007	K.U.Leuven	SVG-related content
C	27 September 2007	K.U.Leuven	Editorial changes and more SVG issues
D	28 October 2007	K.U.Leuven	XForms section added
E	15 November 2007	K.U.Leuven	Executive summary added.
F	17 January 2008	K.U.Leuven	More details on accessibility in SVG 1.2.
G	28 January 2008	K.U.Leuven	New section on ARIA and SVG
H	29 January 2008	K.U.Leuven	Various additions (References, Outlook)

Table of Contents

1 Executive Summary

This report discusses the Scalable Vector Graphics (SVG) and XForms formats from an accessibility perspective. For each specification, we describe the current support in user agents and assistive technologies, and the accessibility features and issues. For SVG, it appears that more work is necessary on the specification to enhance accessibility. For both formats, more work needs to be done to make the user agents (often plug-ins) accessible to assistive technologies.

2 Introduction

The HyperText Markup Language (HTML) is a very successful language with many documented guidelines and techniques for making Web content accessible [W3C 1999a, W3C 2007, BITV,...]. It also has severe limitations in a number of areas, for which more powerful formats were developed later:

- images: for a very long time, browsers only supported bitmap-based image formats (GIF, JPEG and – later – PNG); Macromedia Flash and Scalable Vector Graphics are two formats that were meant to fill this gap;
- scientific notation in areas such as mathematics, physics and chemistry: authors often rely on images to represent scientific notation, but text alternatives are hard to provide or hard to understand (for example, Wikipedia uses LaTeX/TeX as textual alternative for images that contain formulas): Mathematical Markup Language (MathML) and Chemical Markup Language (CML)[1] have been developed to address this, but browser support is limited for MathML and non-existing for CML (CML is supported through a Java applet);
- forms: HTML forms currently lack features to mark required fields, data formats for fields, conditional fields etcetera; XForms was developed to address these issues and to build accessibility into forms markup.

Even though the major browsers added generic XML support to their feature set some time ago, the promise of accessible vector graphics, scientific notations and forms has not been fulfilled. This report discusses accessibility issues for two of the formats mentioned above: Scalable Vector Graphics and XForms.

[1]Peter Murray-Rust, Henry S. Rzepa and Christopher Leach: "CML - Chemical Markup Language" Presented as a poster at the 210th ACS Meeting in Chicago on 21 August, 1995. The open-source site for CML is at http://cml.sourceforge.net/.

3 Accessibility of Scalable Vector Graphics

3.1 Introduction

The SVG format has its roots in Precision Graphics Markup Language (PGML), submitted to the World Web Web Consortium in 1998[2] by Adobe, IBM, Netscape and Sun, and Microsoft's Vector Markup Language (VML)[3], submitted by Microsoft, Macromedia, Hewlett-Packard, Autodesk and Visio. The W3C Graphics working group released SVG 1.0 in 2001. Below is the development history of SVG:

- Jon Ferraiolo, ed: "Scalable Vector Graphics (SVG) 1.0 Specification - W3C Recommendation 04 September 2001"[4]
- Jon Ferraiolo et al, eds: "Scalable Vector Graphics (SVG) 1.1 Specification - W3C Recommendation 14 January 2003"[5]
- Tolga Capin, ed: "Mobile SVG Profiles: SVG Tiny and SVG Basic - W3C Recommendation 14 January 2003"[6]
- Ola Andersson et al, eds: "Scalable Vector Graphics (SVG) Tiny 1.2 Specification - W3C Candidate Recommendation 10 August 2006"[7]
- Alex Danilo et al, eds: "SVG Print 1.2, Part 2: Language - W3C Working Draft 01 May 2007"[8] (Part 1 is a primer[9].)

SVG 1.1 Tiny and SVG 1.1 Basic are profiles aimed at mobile devices with limited capabilities compared to desktop computers. SVG Tiny is defined to be suitable for cell phones; SVG Basic is defined to be suitable for Personal Digital Assistants (PDAs). For example, SVG Basic removed some of the filter effects, and removed support for the 'cursor' and 'marker' elements. SVG Tiny also removed filter effects, 'cursor', 'marker', 'linearGradient', 'mask', 'style', all markup related to client-side scripting, and some text-related features.

[2]Nabeel Al-Shamma et al: "Precision Graphics Markup Language (PGML) - World Wide Web Consortium Note 10-April-1998" http://www.w3.org/TR/1998/NOTE-PGML.
[3]Brian Mathews et al: "Vector Markup Language (VML) - World Wide Web Consortium Note 13-May-1998" http://www.w3.org/TR/1998/NOTE-VML-19980513.
[4] http://www.w3.org/TR/2001/REC-SVG-20010904/
[5] http://www.w3.org/TR/2003/REC-SVG11-20030114/
[6] http://www.w3.org/TR/2003/REC-SVGMobile-20030114/
[7] http://www.w3.org/TR/2006/CR-SVGMobile12-20060810/
[8] http://www.w3.org/TR/2007/WD-SVGPrint12-20070501/
[9] http://www.w3.org/TR/2007/WD-SVGPrintPrimer12-20070501/

Like HTML, SVG is a language intended to work in conjunction with other languages, including:

- Synchronized Multimedia Integration Language (SMIL) for animations,
- Cascading Style Sheets (CSS),
- JavaScript,
- Document Object Model (DOM),
- XML Linking Language (XLink)[10].

This means that accessibility issues for images in SVG go beyond the features of the format.

In addition to the specifications listed above, there is also a W3C working draft titled "An XHTML + MathML + SVG Profile" [W3C 2002d], which has not evolved since August 2002 and which is generally ignored by the SVG developer community[11].

3.2 User Agent and Assistive Technology Support

3.2.1 Browsers and Plug-ins

SVG support in web browsers has progressed slowly. For several years, the Adobe SVG Viewer[12] was the only real option to make SVG work in a web browser. Microsoft Internet Explorer and Netscape supported SVG through this plug-in. The Adobe SVG Viewer, supports most of SVG 1.0 and SVG 1.1. The plug-in can also be made to work with Opera[13] and with Mozilla Firefox and SeaMonkey[14]. However, Adobe stopped developing the viewer several years ago (except for the release of a beta for version 6) and eventually announced that it will discontinue support for the Adobe SVG Viewer on 1 January 2009[15].

[10] http://www.w3.org/TR/2001/REC-xlink-20010627/
[11] With a few exceptions, for example, http://internet-apps.blogspot.com/2005/06/xhtmlmathmlsvg-document-in-sidewinder.html, http://www.echoofeden.com/digest/mit/2007/03/07/ie-understands-xhtml/ and http://internet-apps.blogspot.com/2005/06/xhtmlmathmlsvg-document-in-sidewinder.html. There are examples that render in Gecko-based browsers (for example, Firefox 1.5 and higher) and Opera.
[12] http://www.adobe.com/svg/viewer/install/
[13] Installing the Adobe SVG Viewer plug-in: http://www.opera.com/support/search/view/466/
[14] http://plugindoc.mozdev.org/windows-all.html#AdobeSVG
[15] Adobe to Discontinue Adobe SVG Viewer: http://www.adobe.com/svg/eol.html

The Corel SVG Viewer[16] could not compete with Adobe's and is no longer available as a separate download.

In February 2007, the Renesis SVG Player[17] became available; it can be used as a standalone viewer on many platforms (including the mobile platforms Windows CE and Symbian) and as a browser plug-in. It aims to support for SVG 1.2 Full and Tiny[18].

In September 2007, the Japanese SVG Map Consortium released "SVG Map Toolkit", an SVG viewer (beta) for Microsoft Internet Explorer[19]; the goal is to support SVG 1.2 Tiny.

In the last few years, a number of browsers have been adding native support for SVG. Opera Software added SVG support in Opera 8; Opera 9.24 supports SVG 1.1 Basic[20].

The Mozilla SVG Project[21] has been working on SVG support in the Gecko layout engine, which is used by Mozilla Firefox, SeaMonkey, Camino, Netscape and others. The goal is to enable complete support for SVG 1.1 Full. Mozilla Firefox has native support for SVG since version 1.5, but some modules of the specification have not been implemented[22] yet, including the important Animation Module. Work on Firefox 3 has improved SVG support, for example by adding support for the 'foreignObject' element (which is important for rendering HTML content within SVG), 'em' and 'ex' units to indicate lenghts, a number of SVG DOM methods, and SVG 1.1 filters[23].

Konqueror, a web browser and file manager for the K-Desktop Environment (KDE) supports SVG through the KSVG[24] plug-in (since Konqueror 3.2). The goal is to support SVG 1.1.

KSVG32 was ported into Apple's WebCore framework (part of the browser engine WebKit[25]) in order to enable SVG support in Safari. The goal is also to support SVG 1.1[26].

[16] Corel® SVG Viewer - Technical Specification:
http://apps.corel.com/svgviewer/Viewer_tech_spec/viewer_tech_spec.htm
[17] http://www.examotion.com/pages/renesis/player.php
[18] Reneis Player: Feature Matrix:
http://www.examotion.com/pages/renesis/player.php?view=matrix
[19] SVG Map Toolkit Download: http://blog.svg-map.com/2007/09/svg_map_toolkit.html
[20] http://www.opera.com/products/desktop/svg/
[21] http://www.mozilla.org/projects/svg/
[22] SVG in Firefox: http://developer.mozilla.org/en/docs/SVG_in_Firefox_1.5
[23] SVG improvements in Firefox 3:
http://developer.mozilla.org/en/docs/SVG_improvements_in_Firefox_3
[24] http://svg.kde.org/
[25] WebKit: http://developer.apple.com/opensource/internet/webkit.html
[26] WebKit Open Source Project: SVG: http://webkit.org/projects/svg/

The SVG Viewer Matrix[27] provides a more complete overview.

3.2.2 SVG in (X)HTML

Even though there is some level of SVG support in the major browsers (natively or through plug-ins), it is still difficult to make SVG work across browsers. For example, some SMIL animations work in the Adobe SVG Viewer and Opera, but Firefox does not support animations (except through scripting). There are several way of embedding SVG in (X)HTML (the 'iframe', 'object' and 'embed' elements), but each has different consequences for browser support, especially when there needs to be JavaScript interaction between the HTML page and the embedded SVG. For example, Adobe SVG Viewer 3.01 and its successors disable scripting for security reasons when an SVG file is embedded using the 'object' element in Microsoft Internet Explorer, and Adobe recommends that developers use the 'embed' element (which was never part of the HTML and XHTML standards) instead of the 'object' element when embedding SVG in HTML[28]. Some developers resort to "browser sniffing" and dynamically write either an 'object' or an 'embed' element into the HTML code depending on the browser[29]. Some other frequent errors are addressed by Jonathan Watt's SVG Authoring Guidelines[30].

As a consequence of the above cross-browser issues, some developers have resorted to alternative approaches, for example, displaying SVG through a Flash applet[31]. Other have worked on inline SVG in Microsoft Internet Explorer through Silverlight[32].

3.2.3 SVG and Assistive Technology

It is clear from the issues described above that supporting SVG is not a straightforward task. Developers of assistive technologies have spent considerable time and resources on building support for the many (often non-standard) uses of HTML, CSS and JavaScript, and the addition of SVG is yet another source of complicated issues, even though SVG has a number of accessibility features, which were described in a W3C Note

[27] http://wiki.svg.org/Viewer_Matrix
[28] http://www.adobe.com/svg/viewer/install/main.html
[29] Jeff Schiller: "Inlaying SVG with HTML": http://blog.codedread.com/archives/2006/01/13/inlaying-svg-with-html/
[30] http://jwatt.org/svg/authoring/
[31] Claus Wahlers: "Display SVG in 97% of all Web Browsers": http://wahlers.com.br/claus/blog/display-svg-in-97-of-all-web-browsers/
[32] Sam Ruby: "Inline SVG in MSIE": http://www.intertwingly.net/blog/2007/05/04/Inline-SVG-in-MSIE

[W3C 2000]. In spite of the accessibility features, SVG has even been criticised as being less accessible than Macromedia (now Adobe) Flash[33].

One of the main benefits of SVG is that images can be resized without loss of precision, which makes it an attractive image format for persons who otherwise rely on magnification solutions. However, this feature is not available in every implementation (for example, for SVG files that are directly loaded into Firefox 2), and its usefulness is limited in some implementations by the fact that it is impossible to pan over the enlarged image (for example, Internet Explorer 6 with Adobe SVG Viewer 3.0)[34]. In some implementations, zooming causes text elements to overlap (see [Kirkpatrick 2006] for an example in Firefox 1.5). There are also examples where zooming works well: for example, "peepo with svg"[35] uses SVG files embedded in an XHTML 1.0 Strict file by means of 'object' elements; the buttons created by means of these 'object' elements scale very well when viewed in Gecko-based browsers such as Firefox and SeaMonkey.

Martin Rotard and others [Rotard 2004a, 2004b] have shown that access to SVG can be provided to blind people using tactile displays. They developed several types of filters which users with a visual impairment can interactively apply and combine. One example is a filter that removes gradients and patterns in fillings (for example in charts). Another example is the contour filter that removes fillings from shapes and shows just the outlines. They also developed functionality to assign RGB colours and SVG colour keywords to the nearest colour in the (L*a*b*) uniform colour space developed by McLaren and others [McLaren et al]. The described approach has some limitations. For example, the SVG file needs to be separated from the HTML file in which it is embedded (unless it is a standalone file), animations and scripting are not supported, and the tactile devices that are used for rendering the image are very expensive.

The IVEO software by ViewPlus, which was presented at the CSUN 2006 conference [Gardner 2006], was designed to enable access to non-textual information such as charts, graphs, maps and diagrams. One of its components, the IVEO Viewer, enables blind users to open SVG files and read them on-screen or with the IVEO Touchpad. The software allows the user to create a tactile copy of the image with an embosser and to place it on the IVEO Touchpad, where it can be explored: when the user presses a component in the image, the software speaks its associated label and

[33] http://tech.groups.yahoo.com/group/svg-developers/message/19323

[34] SVG 1.0 (http://www.w3.org/TR/2001/REC-SVG-20010904/interact.html#ZoomAndPanAttribute), SVG 1.1 (http://www.w3.org/TR/SVG11/interact.html#ZoomAndPanAttribute) and the SVG 1.2 draft (http://www.w3.org/TR/SVGMobile12/interact.html#MagnificationAndPanning) make the following requirement reqarding panning: "SVG user agents that operate in interaction-capable user environments are required to support the ability to magnify and pan" (the text is identical in the three specifications).

[35] http://www.peepo.com/index.html

descriptions[36]. It uses only the SVG elements 'title', 'desc' and 'path'. There is also an IBM ViaVoice plug-in for IVEO Viewer[37].

Research by Zaineb ben Fredj and David Duce at Oxford Brookes University [ben Fredj & Duce, 2003, 2006] took a different approach: in their view, working on SVG directly is too limiting because it is a presentation-oriented language. Instead, they envisage a system in which SVG is only one of the possible outputs of a system with three layers: a semantic layer (a domain-specific language), a structure layer (ZineML, which describes common graphical objects such as boxes, circles and arrows but without defining how these should be rendered), and a presentation layer, which may use SVG, textual or other representations. The researchers consider a process that generates the presentation layer from a semantic layer as more effective with regard to accessibility than creating the presentation layer – including the addition of 'title' and 'description' elements to the drawing primitives – directly.

Mozilla Firefox added support for Microsoft Active Accessibility (MSAA) in version 2 in order to enable support by screen readers. GW Micro's Window-Eyes (since version 5.5) and Freedom Scientific's JAWS (since version 7.0) have been reported to work with Firefox 2. In addition, several plug-ins have appeared that can turn Firefox into a talking browser or a screen reader emulator: Fire Vox[38], Fangs[39] and Foxy Voice[40]. This combination of speech access and native SVG support may look promising, but it has not yet resulted in SVG being accessible through speech in a web browser. For example, if an SVG file uses 'title' and 'desc' elements to describe the overall image and some of it components (see example code below), and the file is loaded in Firefox 2, JAWS 8.0 will speak the content of the topmost 'title' element (the content is displayed in Firefox' title bar), but none of the other elements. (For comparison, the effect is the same with Internet Explorer 6, Adobe SVG Viewer 3 and JAWS 8.0 on Windows 2000. In both browsers, JAWS also ignores language attributes on the 'svg' root element, just as it ignores the 'xml:lang' in XHTML.) If the same file is loaded into Fire Vox, the speech synthesizer only says: "End of document"[41].

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
```

[36] See also http://www.viewplus.com/company/news/2051/.
[37] http://www.viewplus.com/products/touch-audio-learning/IVEO/
[38] http://www.firevox.clcworld.net/
[39] http://sourceforge.net/projects/fangs
[40] http://foxyvoice.kenche.info/ (not available at time of writing)
[41] For a few examples of simple SVG files with 'title' elements, see the weather symbols submitted to Wikimedia Commons: http://commons.wikimedia.org/wiki/Category:Weather_symbols. For a web comic using SVG, see http://pixelpalaces.com/dk/.

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 150 150"
xml:lang="en">
  <title>Light rain</title>
  <desc>The picture is a weather symbol for light rain (a cloud with
one rain drop).</desc>
...
</svg>
```

3.3 Accessibility Features and Issues in SVG

Each of W3C's SVG specifications contains a section on accessibility support. Both in SVG 1.0[42] and in SVG 1.1[43], these sections reference the WAI's accessibility guidelines (Web Content Accessibility Guidelines 1.0, User Agent Accessibility Guidelines 1.0 and Authoring Tool Accessibility Guidelines 1.0) and provide a summary of the W3C Note "Accessibility Features of SVG" [W3C 2000]. SVG 1.1 became a recommendation two and a half years after the publication of "Accessibility Features of SVG", but since the goals of SVG 1.1 were modularization and the creation of mobile profiles, and not the introduction of new features, the accessibility note did not need to be updated. However, when SVG Tiny 1.2 reaches Recommendation stage, it will be possible to expand the W3C Note on SVG accessibility (see below).

One of the guidelines in the Web Content Accessibility Guidelines 1.0 [W3C 1999a] is the separation of content and presentation by using style sheets. It is not clear how this can be achieved in SVG, especially in the light of the fact that due to cross-browser issues, developers recommend the use of specific attributes like fill and stroke[44] (instead of the style attribute, which does not separate content from presentation, either).

SVG supports declarative animations (as opposed to animations in client-side script) through elements borrowed from SMIL[45]: 'animate', 'set', 'animateMotion', 'animateColor' and 'animateTransform'. It is possible to define the start time and duration of an animation in a declarative way (for example, with the 'begin' and 'dur' attributes of the 'animate' element), but it is hardly possible to give the user control over the start and end of an animation without scripting. The 'accessKey()' event handler suffers from the same problems as the 'accesskey' attribute in HTML[46] [Foliot 2004, 2006] and is probably rarely used in practice to start and stop animations. The absence of keyboard control over animations is serious because an animation may be distracting or even contain flickering or flashing content.

[42] http://www.w3.org/TR/2001/REC-SVG-20010904/access.html
[43] http://www.w3.org/TR/2003/REC-SVG11-20030114/access.html
[44] http://jwatt.org/svg/authoring/#the-style-attribute
[45] http://www.w3.org/TR/2003/REC-SVG11-20030114/animate.html
[46] http://www.w3.org/TR/1999/REC-html401-19991224/interact/forms.html#adef-accesskey

SVG developers make extensive use of client-side script when creating interactive applications[47]. They use DOM methods[48] to create and manipulate SVG content. Content can be added or changed anywhere in the SVG DOM, and this creates a problem similar to that of DOM manipulations in (X)HTML: how do assistive technologies get notified of changes in the DOM, and how can these changes be presented to the user (especially blind users) in a user-friendly fashion?

An important issue that is not addressed in the accessibility support sections of SVG 1.0 and 1.1 is keyboard accessibility. The W3C Note on "Accessibility Features of SVG" assumes that this can be handled by implementing the User Agent Accessibility Guidelines:

> An accessible user agent will allow the triggering events to be generated from a mouse or other pointer device (where available) as well as from a keyboard. According to the User Agent Accessibility Guidelines [UAAG10], Guideline 1 and especially Checkpoint 1.1, user agents must provide device-independent ways of activating all application functions and indicate how those functions are activated (for example a text-based system could provide a "context menu" listing available actions to the user).

Examples of SVG with keyboard control are usually limited to applications that require the use of a keyboard for all users, for example games that require keyboard control[49].

Unlike HTML 4[50] [W3C 1999b], the SVG 1.0 and 1.1 specifications don't explain how elements may receive focus, except for the definition of the events focusin and focusout (with the corresponding attributes 'onfocusin' and 'onfocusout', respectively). The SVG 1.1 specification mentions examples like "when a 'text' [element] gets selected", but in the absence of support for keyboard navigation, these events are only useful for persons who can handle a pointing device. (The section "Text selection and clipboard operations"[51] in the SVG 1.1 specification completely focuses on pointing devices, without any mention of keyboard selection.)

Fortunately, the drafts for SVG Tiny 1.2 have incorporated a section on navigation[52] and have expanded the "Accessibility Support" chapter[53].The

[47] A recent example is Lily, a visual programming environment for Mozilla based on SVG and JavaScript, by Bill Orcutt (beta release announced on 28 January 2008): http://www.lilyapp.org/.
[48] http://wiki.svg.org/index.php?title=DOM_Methods
[49] See for example Jeff Schiller's Tetris demo at http://www.codedread.com/yastgame.php
[50] http://www.w3.org/TR/1999/REC-html401-19991224/interact/forms.html#focus
[51] http://www.w3.org/TR/SVG11/text.html#TextSelection
[52] http://www.w3.org/TR/SVGMobile12/interact.html#navigation
[53] http://www.w3.org/TR/SVGMobile12/access.html

navigation section defines new attributes such as 'nav-next', 'nav-prev', 'nav-up' and 'nav-right', which can be used to specify navigation order, and the 'focusHighlight' attribute, which indicates whether a user agent should highlight an element on focus. The navigation section also defines navigation behaviour, which should allow users to tab through the focusable elements in an SVG document. The order in which elements can receive focus is called the "focus ring". The default focus ring is the set of focusable elements in document order. Authors can use the focus attributes[54] ('nav-next', 'nav-prev', etcetera) to modify the default focus ring. In addition, the 'focusable' attribute[55] can be used to specify whether an element can receive focus or not. Some elements are focusable by default: the 'a' element, text content block elements when they are editable, elements that are the target of a specify category of animations, and certain elements that respond to specific 'listener' elements.

The expanded "Accessibility Support" chapter now contains a new section on "SVG User Agent Accessibility Guidelines", with guidelines regarding colour information, navigation, interaction and a text-only view.

3.4 SVG and WAI-ARIA

Recent years have seen growing popularity of so-called "rich internet applications" (RIAs): Web-based applications that have features and functionality of traditional desktop applications and that often rely on remote scripting (for example, AJAX) to update the user interface with data from the application server. However, neither (X)HTML nor SVG have sufficient elements and attributes to implement many of the user interface elements that we are familiar with on desktop applications: collapsible menus, tree views, required form fields, sortable rows and columns in tables, spreadsheet functionality, date pickers, colour pickers, drag-and-drop functionality, etcetera. Creative developers started emulating these features by means of clever combinations of (X)HTML, JavaScript and CSS. However, when a certain HTML construct is turned into a user interface control, for example, when a list with sublists is turned into a menubar, there is no programmatic way to notify a screen reader of the presence of a menubar. As a consequence, many HTML elements become overloaded with semantics for which the markup language was not originally designed. The Protocols and Formats Working Group (PF WG) of the W3C's Web Accessibility Initiative (WAI) started work on a "Role Taxonomy for Accessible Adaptable Applications"[56] to define roles (e.g. tabpanel, liveregion, menubar, breadcrumbs, ...) and role properties in order to support a richer set of user interface controls in

[54] http://www.w3.org/TR/SVGMobile12/interact.html#specifyingnavigation
[55] http://www.w3.org/TR/SVGMobile12/interact.html#focusable-attr
[56] http://www.w3.org/WAI/PF/GUI/

web pages. This was followed up with more recent work: "Roles for Accessible Rich Internet Applications (WAI-ARIA Roles)"[57], "States and Properties Module for Accessible Rich Internet Applications (WAI-ARIA States and Properties)"[58] and "Embedding Accessibility Role and State Metadata in HTML documents"[59], and ARIA is being implemented in Firefox[60]. GW Micro's Window-Eyes was the first screen reader that added support for Firefox and ARIA; JAWS 7 added partial support for ARIA[61]. The Protocols and Formats and the HTML Working Groups are also discussing how to integrate ARIA into HTML5[62].

SVG developers often try to mimic HTML form elements[63], create progress bars[64], charts and diagrams[65], etcetera. Of course, custom interface elements like these are not recognized as such by assistive technologies. The ARIA specification is not tied to (X)HTML, so the ARIA attributes seem good candidates for telling assistive technologies that a certain set of elements is meant to represent a particular type of user interface control and to describe its current state [Schepers 2006, 2007], or how a certain "live region" should be treated by assistive technology[66].

[57] http://www.w3.org/TR/aria-role/

[58] http://www.w3.org/TR/aria-state/

[59] http://www.w3.org/WAI/PF/adaptable/HTML4/embedding-20061212.html

[60] http://developer.mozilla.org/en/docs/Accessible_DHTML and http://developer.mozilla.org/en/docs/ARIA:_Accessible_Rich_Internet_Applications/Relationship_to_HTML_FAQ

[61] http://developer.mozilla.org/en/docs/ARIA:_Accessible_Rich_Internet_Applications#Testing_with_screen_readers

[62] http://esw.w3.org/topic/HTML/ARIAIntegration

[63] See for example http://tech.groups.yahoo.com/group/svg-developers/message/55930

[64] See for examle http://tech.groups.yahoo.com/group/svg-developers/message/57215

[65] See for example http://blog.codedread.com/archives/2006/01/12/pretty-web-statistics-in-svg

[66] WAI-ARIA States and Properties: Live Regions: http://www.w3.org/TR/aria-state/#liveregions. For an implementation example see Firefox support in Orca: http://live.gnome.org/Orca/Firefox/LiveRegions (announcement 24 January 2008).

4 Accessibility of XForms

4.1 Introduction

HTML forms have a number of well-known limitations because HTML provides no features to

- mark fields as required,
- define data types for input fields,
- validate form input in the browser,
- perform calculations on form input in the browser,
- display error messages and help messages,
- add or remove form fields dynamically based the input in other fields.

Form developers have relied on other HTML features (for example, plain text to indicate required form fields and data formats) and especially on client-side script to address these limitations. Unfortunately, client-side script has many browser-compatibility issues, is brittle and is hard to maintain. For these reasons, the World Wide Web Consortium started work on a new forms specification. The resulting XForms format is part of XHTML 2[67] (which is still under development) but also generic enough to be combined with other XML-based languages, for example with SVG[68]. Below is the development history of XForms:

- Micah Dubinko et al, eds.: "XForms 1.0 - W3C Recommendation 14 October 2003"[69]
- John M. Boyer et al, eds.: "XForms 1.0 (Second Edition) - W3C Recommendation 14 March 2006"[70]
- John M. Boyer, ed.: "XForms 1.0 (Third Edition) - W3C Recommendation 29 October 2007"[71]

In addition to these editions of XForms 1.0, the XForms working group is also preparing two other specifications:

[67] XHTML 2.0 – W3C Working Draft 26 July 2006: http://www.w3.org/TR/2006/WD-xhtml2-20060726/
[68] http://www.w3.org/TR/2007/REC-xforms-20071029/#N1408D
[69] http://www.w3.org/TR/2003/REC-xforms-20031014/
[70] http://www.w3.org/TR/2006/REC-xforms-20060314/
[71] http://www.w3.org/TR/2007/REC-xforms-20071029/

- Micah Dubinko, T. V. Raman, eds.: "XForms 1.0 Basic Profile - W3C Candidate Recommendation 14 October 2003"[72]
- John M. Boyer, ed.: "XForms 1.1 - W3C Working Draft 22 Februrary 2007"[73] ("Last Call Working Draft").

4.2 User Agent and Assistive Technology Support

XForms can be implemented on the client side or on the server side. In the latter case, XForms are converted to traditional XHTML forms (with JavaScript to emulate XForms features) before they are sent to the browser. This discussion only considers client-side XForms.

Browsers based on the Mozilla platform, such as Firefox and SeaMonkey, support XForms (second edition) through the Mozilla XForms extension[74]. The work is still ongoing; at the time of writing an evaluation version of the extension is available for testers[75].

Microsoft Internet Explorer 6.0 supports XForms through the formsPlayer plug-in[76].

The Lotus (Workplace) Forms Viewer[77] (or "IBM Lotus Forms" since version 3.0) is a Microsoft Windows application that allows users to view XForms standalone application or in a supported browser. Forms Viewer 2.7 supports Microsoft Internet Explorer 5.5 or higher and Firefox 1.5 or higher[78].

Novell XForms Explorer is a plug-in for Internet Explorer 6 or higher[79]. Support for Mozilla browsers was planned but apparently never came acout. As mentioned above, Mozilla now has its own XForms project.The XForms Explorer is part of the Novell exteNd suite, which also contains an XForms Java Servlet, and a standalone WebStart version of the XForms processor.

[72] http://www.w3.org/TR/2003/CR-xforms-basic-20031014/
[73] http://www.w3.org/TR/2007/WD-xforms11-20070222/
[74] Mozilla XForms Project: http://www.mozilla.org/projects/xforms/
[75] http://www.mozilla.org/projects/xforms/download.html
[76] http://www.formsplayer.com/
[77] http://www-306.ibm.com/software/lotus/forms/trials.html
[78] http://www-1.ibm.com/support/docview.wss?rs=2357&uid=swg27008965
[79] http://developer.novell.com/wiki/index.php/Novell_XForms_Explorer and http://forge.novell.com/modules/xfmod/project/?xforms-explorer

X-Smiles[80] is an open-source browser in Java that supports XHTML Basic, SVG (through the open-source CSIRO SVG Toolkit), SMIL 2.0 Basic Profile, CSS Mobile Profile 1.0, XForms[81] and a few other technologies.

Other client-side implementations rely on JavaScript, Java applets or Flash applets[82].

None of the above implementations (with the possible exception of IBM Lotus Forms Viewer[83]) have been reported to make XForms accessible with a screen reader.

4.3 Accessibility Features and Issues in XForms

XForms contains many features that are intended to increase accessibility of web-based forms, for example:

- content is separated from presentation (XForms Model versus XForms User Interface),
- generic user interface controls that are suited for device independence,
- form controls always have labels directly associated with them as (required!) child elements,
- XML events that allow declarative coding of functionality that would otherwise require client-side scripting.

The two XForms books [Dubinko; Raman] that have appeared in print both devote a chapter to accessibility.

In spite of its accessibility features, it is still possible for developers to create accessibility issues (no technical specification can prevent that), for example

[80] http://www.xsmiles.org/index.html
[81] http://www.xsmiles.org/features_xforms.html
[82] More data are available on http://www.w3.org/MarkUp/Forms/wiki/XForms_Implementations and http://en.wikipedia.org/wiki/XForms#Software_support
[83] See "IBM Lotus Forms V3.0 and IBM Lotus Forms Express V3.0 help businesses of all sizes build integrated applications faster by providing a security-rich, intelligent front end to business process automation" (IBM United States Software Announcement 207-218, 25 September 2007) at http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS207-218: it is not entirely clear if the MSAA support applies only to the server software or also to the viewer. However, Appendix B ("Screen reader announcements" - http://publib.boulder.ibm.com/infocenter/forms/v3r0m0/index.jsp?topic=/com.ibm.help.forms.doc/Best_Practices_Form_Design/i_bpfd_g_screen_reader_announcements.html) of the Lotus Forms 3 Information Center contains the following note: "Webform Server can only use JAWS while the Viewer can use JAWS, Microsoft Narrator, or Microsoft Eyes."

- empty form labels,
- labels that are not sufficiently descriptive,
- time limits,
- interruptions (for example for updated content),
- insufficient colour contrast,
- language that is hard to understand (jargon, abbreviations, required reading ability, ...).

The most important issue, however, is probably the dynamic nature of some forms: even with XML Events, it is still a challenge to present error messages, additional form fields, form validation messages etcetera in a user-friendly to users of assistive technologies.

5 Outlook

SVG and XForms are two different types of languages: SVG is presentation oriented, whereas XForms describes semantics. It is hardly surprising then that the accessibility issues related to these specifications are different. For XForms, users need more accessible implementations, so that issues such as event handling and DOM manipulation can become the subject of research supported by end-user tests.

It is clear from the above that much more needs to be done to make "SVG accessibility" a reality. The SVG Tiny 1.2 specification is the first SVG specification that clearly addresses keyboard navigation; hopefully, user agents will implement also this new part of the specification. User agents should pass more than just the title element to assistive technologies.

SVG and XForms can also be used together. There is an example of this in the XForms specification ("Survey Using XForms and SVG"[84]), and the formsPlayer and Mozilla XForms extensions also show examples[85].

It is also interesting that XForms has been integrated in OpenOffice.org 2.0 Writer[86]. Future work is needed to evaluate it with screen readers as the GUI elements rely on the host environment and thus may be inaccessible.

From the above, it appears that the accessibility features in the specifications are either available (XForms) or well under way (SVG 1.2 Tiny, WAI ARIA) but that user agents and assistive technologies need to catch up. Some research is needed into best practices for presenting dynamic changes in the DOM to end users with a disability, but the same is also true in the context of (X)HTML (for ongoing work regarding WAI ARIA see [PFWG]).

[84] http://www.w3.org/TR/2007/REC-xforms-20071029/#N1408D
[85] See "Introducing SVG-rich XForms controls" (http://www.formsplayer.com/node/402) and the XForms calculator with the SVG skin (http://www.mozilla.org/projects/xforms/samples.html).
[86] Choose File – New – XML Form Document. See "Using Forms in Writer": http://wiki.services.openoffice.org/wiki/Documentation/OOoAuthors_User_Manual/Writer_Guide/Using_Forms_in_Writer

6 References

[Altmanninger 2006] Altmanninger, K.; Wöß, W. 2006. "Dynamically Generated Scalable Vector Graphics (SVG) for Barrier-Free Web Applications". In: Miesenberger, K.; Klaus, J.; Zagler, W.; Karshmer, A. (Eds.). Computers Helping People with Special Needs. 10th International Conference, ICCHP 2006, Linz, Austria, July 11-13, 2006: Proceedings. Lecture Notes in Computer Science 4061. Berlin: Springer. Pages 128-135. (DOI: 10.1007/11788713_20)

[Altmanninger 2007] Altmanninger, K.; Wöß, W. 2007. "Scalable Vector Graphics Pattern P2P-Service for Enhanced Barrier-Free Web Applications". In: Cham, T.-J. et al (Eds.). Advances in Multimedia Modeling, 13th International Multimedia Modeling Conference, MMM 2007, Singapore, January 9-12, 2007. Proceedings, Part II. Lecture Notes in Computer Science 4352. Berlin: Springer. Pages 730-737. (DOI: 10.1007/978-3-540-69429-8_80)

[ben Fredj & Duce 2003] ben Fredj, Z.; Duce, D. A. 2003. "Schematic Diagrams, XML and Accessibility". In: Proceedings of the theory and Practice of Computer Graphics 2003 (June 03 - 05, 2003). TPCG. IEEE Computer Society, Washington, DC. Pages 49-55. (DOI: 10.1109/TPCG.2003.1206930)

[ben Fredj & Duce 2006] ben Fredj, Z.; Duce, D. A. 2006. "GraSSML: accessible smart schematic diagrams for all". In: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A): Building the mobile web: rediscovering accessibility? (Edinburgh, U.K., May 22 - 22, 2006). W4A, vol. 134. ACM, New York, NY. Pages 57-60.
(DOI: http://doi.acm.org/10.1145/1133219.1133229 (10.1109/TPCG.2003.1206930) (See also http://www.w4a.info/2006/prog/8-benfredj.pdf)

[BITV] Bundesministerium des Innern [Ministry of the Interior]. 2002. Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie Informationstechnik-Verordnung – BITV). 17 July 2002. http://www.bmi.bund.de/Annex/de_22681/Barrierefreie_Informationstechnik-Verordnung_BITV_als_PDF-Download.pdf.

[Campin] Campin, B.; McCurdy, W.; Brunet, L.; Siekierska, E. 2003. "SVG Maps for People with Visual Impairment". In Proceedings of SVG Open Conference 2003.

http://www.svgopen.org/2003/papers/svgmappingforpeoplewithvisualimpairments/

[Dubinko] Dubinko, M. 2003. XForms Essentials. Sebastopol, CA: O'Reilly & Associates. (Online version: http://xformsinstitute.com/essentials/browse/.)

[Foliot 2004] Foliot, J. 2004. "Using Accesskeys – Is it worth it?". (Last updated October 2006) http://www.wats.ca/show.php?contentid=32

[Foliot 2006] Foliot, J. 2006. "Access + Key Still Equals Accesskey". http://www.wats.ca/show.php?contentid=47

[Gardner 2001] Gardner, J.; Bulatov, V. 2001. "Smart Figures, SVG, and Accessible Web Graphics". CSUN Conference, 2001. http://www.csun.edu/cod/conf/2001/proceedings/0103gardner.htm

[Gardner 2006] Gardner, J. 2006. "Tactile-Audio Diagrams - IVEO". CSUN Conference, Los Angeles, USA, 20-25 March 2006.

[Herman] Herman, I.; Dardailler, D. 2002. "SVG Linearization and Accessibility". In Proceedings of SVG Open Conference 2002. http://www.svgopen.org/2002/papers/herman_dardailler__svg_linearization_and_accessibility/

[Herman b] Herman, I.; Dardailler, D. 2002. "SVG Linearization and Accessibility". Computer Graphics forum. Volume 21 (2002), number 4, Pages 777-186.

[Jackson] Jackson, D. 2004. "Making Accessible SVG". (Abstract) In Proceedings of SVG Open Conference 2003. http://www.svgopen.org/2003/paperAbstracts/making_accessible_svg.html

[Kirkpatrick 2006] Kirkpatrick, A. 2006. "Overview of Accessible Technologies" In: Web Accessibility: Web Standards and Regulatory Compliance. Berkeley, CA: friends of ED (Apress). Pages 85-100. http://www.friendsofed.com/web-accessibility/chapter04.html

[Lemon 2007] Lemon, Gez. 2007. "WAI-ARIA Live Regions" http://juicystudio.com/article/wai-aria-live-regions.php

[McLaren] McLaren, K. et al. 1976. The development of the CIE 1976 (L*a*b*) uniform colour space and colour-difference formula, Journal of the Society of Dyers and Colourists 92, 1976.

[Mertz] Mertz, D. 2005. "Program with SVG" In: IBM developerWorks. http://www.ibm.com/developerworks/xml/library/x-matters40/

[Murphy] Murphy, S. 2005. "Accessibility of Graphics in Technical Documentation for the Cognitive and Visually Impaired". SIGDOC '05: Proceedings of the 23rd annual international conference on Design of communication. Pages 12-17.

[PFWG] W3C Protocols and Formats Working Group. (no date). "ARIA Best Practices". W3C ESW Wiki. http://esw.w3.org/topic/PF/ARIA/BestPractices

[Quint 2003] Quint, Antoine. 2003. "SVG and XForms: Rendering Custom Content". IBM developerWorks. http://www-128.ibm.com/developerworks/xml/library/x-svgxf2/

[Raman] Raman, T.V. 2004. XForms: XML Powered Web Forms. Boston, MA: Addison-Wesley.

[Rotard 2004a] Rotard, M.; Ertl, T. 2004. "Tactile Access to Scalable Vector Graphics for People with Visual Impairment". In Proceedings of SVG Open Conference 2004. http://www.svgopen.org/2004/papers/TactileAccessToSVG/

[Rotard 2004b] Rotard, M.; Otte, K.; Ertl, T. 2004. "Exploring Scalable Vector Graphics for Visually Impaired Users". In: Miesenberger, K.; Klaus, J.; Zagler, W.; Burger, D. (Eds.). Computers Helping People with Special Needs. 9th International Conference, ICCHP 2004, Paris, France, July 2004: Proceedings. Lecture Notes in Computer Science 3118. Berlin: Springer. Pages 725-730.

[Schepers 2006] Schepers, Doug. 2006. "SVG Text, Semantics, and Accessibility". http://www.schepers.cc/?p=11

[Schepers 2007] Schepers, Doug. 2007. "Underscoring Accessibility". http://www.schepers.cc/?p=46

[Takagi 2007] Takagi, H.; Ishihara, T. 2007. "Technology Advances and Standardization Toward Accessible Business Graphics". In: Stephanidis, C. (Ed.). Universal Access in Human-Computer Interaction, Part III, Applications and Services. HCII 2007, Beijing, China, July 2007: Proceedings. Lecture Notes in Computer Science 4556. Berlin: Springer. Pages 426-435.

[Venn] Venn, B. 2003. "Add interactivity to your SVG: Create SVG documents that respond to user input". In IBM developerWorks. http://www.ibm.com/developerworks/xml/library/x-svgint/

[WAI 2004] Web Accessibility Initiative. 2004. WAI comments 27 October 2004 Working Draft of SVG 1.2. (Edited by Jon Gunderson). http://www.w3.org/WAI/UA/2004-11-wai-comments-svg12.html.

[W3C 1999a] World Wide Web Consortium. 1999. Web Content Accessibility Guidelines 1.0. (Edited by Wendy Chisholm, Gregg Vanderheiden and Ian Jacobs). W3C Recommendation 5-May-1999. http://www.w3.org/TR/WAI-WEBCONTENT.

[W3C 1999b] World Wide Web Consortium. 1999. HTML 4.01 Specification. (Edited by Dave Raggett, Arnaud Le Hors, Ian Jacobs). W3C Recommendation 24 December 1999. http://www.w3.org/TR/1999/REC-html401-19991224.

[W3C 2000] World Wide Web Consortium. 2000. Accessibility Features of SVG. (Edited by Charles McCathieNevile and Marja-Riitta Koivunen). W3C Note 7 August 2000. http://www.w3.org/TR/SVG-access/.

[W3C 2002a] World Wide Web Consortium. 2002. Conformance Testing and Certification Model for W3C Specifications. W3C Working Draft 2 January 2002. http://www.w3.org/QA/2002/01/Note-qa-certif-20020102.html.

[W3C 2002b] World Wide Web Consortium. 2002. Requirements for WCAG 2.0. (Edited by Gregg Vanderheiden and Wendy Chisholm). W3C Working Draft 26 April 2002. http://www.w3.org/TR/wcag2-req/.

[W3C 2002c] World Wide Web Consortium. 2002. Authoring Tool Conformance Evaluations. (Last updated by Ian Richards, 28 May 2002.) http://www.w3.org/WAI/AU/2002/tools.

[W3C 2002d] World Wide Web Consortium. 2002. An XHTML + MathML + SVG Profile. (Edited by 石川 雅康 (ISHIKAWA Masayasu)). W3C Working Draft 9 August 2002. http://www.w3.org/TR/XHTMLplusMathMLplusSVG/.

[W3C 2004a] World Wide Web Consortium. 2004. World Wide Web Consortium Process Document. (Edited by Ian Jacobs.) 5 February 2004. http://www.w3.org/2004/02/Process-20040205/.

[W3C 2004b] World Wide Web Consortium. 2004. The QA Handbook. (Edited by Lofton Henderson.) W3C Working Group Note 22 November 2004. http://www.w3.org/TR/qa-handbook/.

[W3C 2007] World Wide Web Consortium. 2007. Web Content Accessibility Guidelines 2.0. (Edited by Ben Caldwell, Michael Cooper, Loretta Guarino Reed and Gregg Vanderheiden). W3C Working Draft 17 May 2007. http://www.w3.org/TR/2007/WD-WCAG20-20070517/.

[W3C DOM] World Wide Web Consortium. Document Object Model (DOM). [Homepage of the W3C DOM Activity]. http://www.w3.org/DOM/.