Estudio sobre la implementabilidad de las normas WAI

ENFOQUES

A Survey on WAI Implementability

V. Luque, C. Delgado, L. Sánchez y N. Fernández

Resumen

La accesibilidad en las páginas Web supone el cumplimiento de unos puntos de chequeo que son costosos de evaluar. Empleando tecnologías adecuadas, basadas en estándares del W3C en lugar de los comúnmente utilizados algoritmos declarados sobre un lenguaje de programación, es posible, no sólo reducir ese coste sino también evitar la posibilidad de tener diferentes interpretaciones de la norma en cada posible herramienta de evaluación. Este artículo presenta una catalogación de varias formas de detección de barreras de accesibilidad y cuantifica la mejora que en la implementabilidad de las normas WAI [1] tiene el formato XHTML Basic [7].

Palabras clave: accesibilidad, WAI, estándares W3C.

Summary

Web accessibility consists on a set of checkpoints rather expensive to evaluate. Using proper technologies, based on well known W3C standards instead of the commonly used algorithms being declared on programming languages, it is possible, not only minimize this cost, but also to reduce the chance to have different interpretations of the guidelines on each evaluation tool. This article shows a classification of several ways to detect Web accessibility barriers and quantifies the positive impact that XHTML Basic [7] has on WAI [1] implementability.

Keywords: accessibility, WAI, W3C standards.

Las normas WAI han tenido un gran éxito de aceptación como guías de recomendación de mejoras en la accesibilidad de las páginas Web

1.- Accesibilidad

Las normas WAI (Web Accessibility Initiative) [1] del W3C han tenido un gran éxito de aceptación como guías de recomendación de mejoras en la accesibilidad de las páginas Web. La falta de esa accesibilidad afecta a un elevado colectivo de la población que se encuentra, por motivos muy diversos, con importantes barreras a la hora de navegar por sitios Web e interactuar con los documentos visitados. Al contrario de lo que mucha gente puede creer en un principio, no se trata sólo de personas con alguna discapacidad personal (dificultades en la visión, en su capacidad auditiva, en su capacidad motriz para manejar un teclado o un ratón, o en su capacidad intelectual). La accesibilidad se está demostrando como un objetivo primordial para que los documentos Web sean independientes, tanto de los dispositivos (hardware) como de navegadores o sistemas operativos (software) [14]. Es decir, la accesibilidad no sólo combate los problemas derivados de discapacidades personales (físicas o psíquicas), sino que también combate la denominada *discapacidad tecnológica* derivada del uso de tecnologías no contempladas por muchos de los diseñadores de las páginas Web que han sido publicadas a lo largo de los últimos años. De esta forma, los innumerables tipos de terminales (no sólo los basados en navegadores antiguos o alternativos, sino también los adaptados a discapacidades personales específicas de sus usuarios o los inalámbricos de reducidas dimensiones) que ofrecen también capacidad de navegación en el Web, pueden llegar a ser tan funcionales como los ordenadores de mesa con navegadores considerados como más *convencionales*. Teléfonos móviles, agendas personales, televisiones digitales, kioscos, sistemas de navegación empotrados en los salpicaderos de los coches,... son dispositivos que deberíamos poder usar también para navegar por el Web, pero para los que **muchos no están preparados**. Pese al hecho de que tener en cuenta estos principios de accesibilidad puede incrementar ligeramente el coste de desarrollo de los sitios Web, la accesibilidad proporciona menores costes de mantenimiento y será dentro de poco (para finales de 2005 según las leyes europeas y españolas) un requisito indispensable por ley [15] en los sitios Web de las administraciones públicas españolas y europeas, entre las que las Universidades, debiendo liderar el proceso, aún se encuentran con bastante camino por recorrer [16].

De entre las normas usadas para la accesibilidad en el Web destacan, entre otras, las WCAG (Web Content Accessibility Guidelines) [1] del propio W3C, un conjunto de 65 puntos de chequeo que los documentos considerados como accesibles deben cumplir. Las WCAG son un conjunto muy heterogéneo de condiciones que resultan muy costosos de evaluar y cuya evaluación automatizada sólo puede ser abordada de forma parcial, esto es, incompletamente. De estas condiciones, las hay de todo tipo: desde las que especifican que las combinaciones de colores de fondo y de texto de las distintas partes de una página Web deben tener un grado de contraste *suficiente* (para que resalte convenientemente el texto sobre el fondo) a las que especifican que las frases no deben ser *demasiado* largas o que debe existir contenido alternativo para aquellos elementos multimedia incrustados en la página que puedan ser inaccesibles para los que no los puedan ver u oír (no necesariamente ciegos o sordos, sino también personas que usen navegadores sin capacidad para visualizar imágenes, distinguir ciertos colores o reproducir sonidos). La especificación de estos 65 puntos de chequeo está redactada en muchas ocasiones en un nivel de abstracción alejado de los detalles técnicos del formato HTML de las páginas Web. Muchos de esos puntos de chequeo, incluso, están abiertos a interpretaciones subjetivas (no se define cuando dos colores presentan *suficiente* contraste, cuando un texto es *demasiado* largo,...) incluyen varias condiciones implícitas, o, simplemente, no son detectables automáticamente.

Las herramientas de evaluación clasifican los 65 puntos de chequeo en dos grupos: los automatizados y los no automatizados

Debido a esta incapacidad de los puntos de chequeo de la norma por ser completamente evaluados de forma automática, las herramientas actuales (Bobby [10], Tawdis [11], WebXACT [12], Torquemada [13]) de evaluación de accesibilidad sólo puedan realizar **evaluaciones incompletas**, identificando, pues, sólo algunas de las posibles barreras de accesibilidad que puede tener un conjunto de documentos. Ello conlleva que las numerosas herramientas de evaluación de la accesibilidad de las páginas Web sean actualmente herramientas **semiautomáticas** que guían al usuario a la hora de facilitarle esa evaluación, pero que no dejan de delegar en su juicio personal estas evaluaciones. Se necesita la supervisión de personas para completar la labor de las máquinas en muchos de estos puntos de chequeo. Muchas de las herramientas se limitan a enfocar la atención del usuario en los puntos considerados de mayor conflictividad en el marcado de las páginas.

Los resultados de una evaluación de accesibilidad (para una página dada) pueden ser fácilmente diferentes según la herramienta utilizada. Por ejemplo, hay un elevado número de páginas Web que pasan el *poco exigente* test automático de Taw (porque muchas condiciones quedan para ser evaluadas a mano) y sin embargo suspenden el *más exigente* test de Bobby o de WebXACT.

2.- Catalogación

Tradicionalmente, las herramientas de evaluación clasifican los 65 puntos de chequeo en dos grupos: los **automatizados** y los **no automatizados**. Éstos últimos necesitan la comprobación de un usuario supervisor y son muy costosas de evaluar, especialmente cuando el número de páginas es elevado, mientras que los automatizados no tienen ese coste de supervisión humana.

Analizando los detalles de implementación, se detecta que, mientras algunos criterios están claramente definidos y admiten una única interpretación (como la existencia o ausencia de determinados elementos o atributos en el marcado), muchos otros están basados en criterios subjetivos y definidos de forma ambigua en el estándar original (como el hecho de que un texto sea *demasiado* largo). Conviene, por lo tanto distinguir entre:

1.- Criterios **automatizables objetivos**: aquellos criterios fácilmente comprobables de forma automática que están objetivamente definidos y no admiten interpretaciones diferentes. Por ejemplo, que una imagen no venga acompañada de un atributo `alt`.

2.- Criterios **automatizables subjetivos**: los que, siendo igualmente fácilmente comprobables de forma automática, están definidos sobre condiciones *difusas* que quedan bajo el **criterio subjetivo** de un evaluador. No están establecidos de forma categórica, así que ciertos grupos de personas pueden estar de acuerdo con ciertos criterios y otros grupos no. Por ejemplo, la condición de que un determinado texto sea *demasiado largo* expresa una condición difusa donde no existe una frontera comúnmente aceptada para diferenciar lo que es *largo* de lo *corto*. Esta condición, que depende de varios condicionantes externos, a la hora de ser automatizada, suele adoptar la forma de una condición no difusa (como por ejemplo que la longitud del texto no supere unos 150 caracteres), sin que esté ampliamente aceptado que un texto con 149 caracteres deba ser considerado como corto y uno con 151 largo. El W3C ha definido un conjunto de heurísticas [2] que intentan ayudar en este asunto, pero cada herramienta tiene su propia interpretación de estas condiciones.

3.- Criterios **semiautomatizables**: aquellos para los que el ordenador no es capaz de discernir por sí mismo un cumplimiento o un incumplimiento con un grado aceptable de confianza y no existe posibilidad de automatizar un criterio para realizar esa evaluación. Por ello es necesaria la intervención de un operador humano que haga ese discernimiento. Los criterios semiautomatizables se distinguen de los siguientes (manuales) en que al menos el ordenador es capaz de **señalizar** los puntos concretos del marcado de una página que el evaluador humano debe revisar (enfocando mejor su atención en la revisión).

4.- Criterios **manuales**: aquellos que quedan totalmente al juicio del evaluador humano y que no son fácilmente automatizables, como el hecho de que la navegación y el vocabulario sean claros y consistentes. Los criterios manuales y semiautomáticos, al requerir la intervención de un operador humano, son muy costosos de evaluar, especialmente cuando se trata de grandes volúmenes de documentos.

Otro aspecto de los puntos de chequeo muy importante es el coste de evaluación o de automatización, el cual viene dado por la técnica usada para representar esa restricción

3.- Implementabilidad

Otro aspecto de los puntos de chequeo más importante aún que el grado de objetividad, es el coste de evaluación o de automatización, el cual viene dado por la técnica usada para representar esa restricción (o condición que refleje la ausencia de barreras). Normalmente, las restricciones que quedan reflejadas en un sistema de reglas declarativo tienen un coste de implementación mucho menor que las que quedan reflejadas en un algoritmo. Así, podemos agrupar los puntos de chequeo en las siguientes categorías, enumeradas en orden creciente de coste de implementación. Empezamos, por lo tanto, por las menos costosas:

3.1.- Declaradas en la gramática de XHTML

XHTML no es un lenguaje único. Desde su nacimiento ha tenido diferentes versiones, comenzando por la Transitional y la Strict [6], pasando por la XHTML Basic [7] y terminando por XHTML 1.1 [8] o incluso XHTML 2.0 [9] (en actual fase de borrador). Estos diferentes dialectos de XHTML tienen distintas restricciones de accesibilidad, algunas de las cuales fueron ya recogidas en la norma WCAG 1.0 [1] del año 1999. Por ejemplo, los atributos alt deben obligatoriamente estar presentes en todas las imágenes desde XHTML Transitional 1.0, al igual que los títulos de los documentos o de las descripciones textuales de las áreas de los mapas de cliente (reglas del WCAG 1.1a y 1.5 respectivamente). Elementos anquilosados como font o center fueron eliminados en XHTML Strict 1.0. XHTML Basic no permite marcos, plugins, scripts, mapas de imagen (tanto de cliente como de servidor) o tablas anidadas para maquetación, por lo que reglas como las que hacen referencia a la necesidad de incluir contenidos alternativos para estos elementos están garantizadas al estar estos

elementos prohibidos. Incluso a pesar de que estas reglas puedan ser expresadas también como reglas XPath [3] como aparece en el siguiente apartado, XHTML contempla reglas explícitas en su gramática que ya recogen estas restricciones. Además de esto, las reglas del WCAG 3.2 (Crear documentos válidos conforme a las gramáticas formales públicas), 3.3 (Usar hojas de estilo para controlar la maquetación y la presentación) y 11.2 (Evitar uso de tecnologías obsoletas) hacen referencia explícita a esta validación XHTML.

3.2.- Declarables en reglas XPath

Se trata de comprobaciones definibles en la versión de XPath 1.0 [3] (estándar) y que evalúan condiciones que no son detectables por la gramática del lenguaje de marcas utilizado. Con este tipo de reglas se pueden detectar los elementos que hacen uso de atributos *desaconsejados* y también aquellos con determinadas combinaciones no deseadas de atributos. Los siguientes puntos de chequeo (ver tabla 1) son declarables con este tipo de reglas:

Se trata de comprobaciones definibles en la versión de XPath 1.0 [3] y que evalúan condiciones que no son detectables por la gramática del lenguaje de marcas utilizado

- Las alternativas textuales deben estar presentes para todos los componentes multimedia y botones de imagen.
- La activación de código JavaScript no debe estar permitida fuera de los atributos específicos orientados a eventos.
- Los marcos deben tener descripciones en sus atributos `title` y contenido alternativo en sus secciones `noframes`.
- El atributo `longdesc` debe estar presente siempre que se requiera una descripción extensa en una imagen o en un marco.
- Los cambios de idioma (tanto dentro de un documento como al saltar de uno a otro) deben ser explícitos.
- El auto-refresco o el auto-redireccionamiento no está permitidos.
- Los eventos dependientes del dispositivo deben ser emparejados con sus equivalentes de otros dispositivos.
- Los campos de formularios visibles deben tener valores por defecto no nulos e imprimibles.
- Las opciones dentro de un `select` y los campos de un formulario deben aparecer convenientemente asociados en grupos manejables.

3.3.- Declarables en reglas XQuery 1.0

Se trata de comprobaciones definidas en la futura versión de XQuery (borrador) [5] (en ocasiones también en XPath 2.0 [4]) y que evalúan las relaciones que deben manifestarse entre varios elementos del documento y que son inexpresables en términos de XPath 1.0. Algunos ejemplos son:

- **Regla 3.5:** Las cabeceras (`h1, h2, ... h6`) de un documento deben estar correctamente jerarquizadas. Si aparece un `h2` antes que ningún `h1`, o si aparece un `h3` antes que un `h2` se puede deducir que la jerarquía de cabeceras está sufriendo irregularidades, lo cual puede desorientar a las personas que las utilizan para localizar las partes del documento que les interesa. La expresión XQuery del cuadro 1 puede servir para detectar las cabeceras `h3` que incumplen esta norma (similares reglas pueden escribirse para el resto de las cabeceras distintas a las `h3`).

```
//h3[let $h3:=self::h3 return
let $h2:=$h3/preceding::h2[last()] return
let $h1:=$h3/preceding::h1[last()] return
$h1=() or $h2=() or $h1>>$h2]
```

Cuadro 1

Tabla 1: Reglas XPath 1.0

WCAG #	Regla XPath 1.0
1.1b	//input[@type="image"][not(@alt)]
1.1c	//img[*toolong*(@alt)][not(@longdesc)]
1.1d	//object[not(*)][normalize-space(text())=""]
1.1e	//frameset[not(noframes)]
3.5a	//h2[not(preceding::h1)]
3.5f	(//h1//h2//h3//h4//h5//h6)[*toolong*_h(text())]
3.5g	(//p//div)[*tooshort*_p(text())][strong/text()=text()]
4.1a	//*[*lang*(.) != *lang*(..)][not(@xml:lang)]
4.3	//html[not(@xml:lang)]
5.6	//th[*toolong*(text())][not(@abbr)]
6.3b	(//a//area)[starts-with(@href,"javascript:")]
6.4a	//*[@onmouseover != @onfocus]
6.4b	//*[@onmouseout != @onblur]
7.4, 7.5	//meta[@http-equiv="refresh"]
9.2a	//*[@onmousedown != @onkeydown]
9.2b	//*[@onmouseup != @onkeyup]
9.2c	//*[@onclick != @onkeypress]
10.1a	//*[@target="_blank" or @target="_new"]
10.4a	//input[@type!="hidden"][not(@value)]
10.4a	//input[@type!="hidden"][@value=""]
10.4b	//textarea[normalize-space(text())=""]
10.4c	//select[not(@multiple)][not(.//option[@selected])]
12.1	//frame[not(@title)]
12.2	//frame[*toolong*(@title)][not(@longdesc)]
12.3a	//form[*toomany*_inputs(.//input)][not(.//fieldset)]
12.3b	//select[*toomany*_options(option)][not(optgroup)]
12.3c	//p[*toolong*(text())]

• **Regla 12.4:** Los campos de formularios deben aparecer convenientemente emparejados con sus textos descriptivos. La expresión XQuery del cuadro 2 puede servir para detectar las etiquetas `label` que no tienen asociado un campo de formulario o que tienen asociado más de uno.

```
//label[let $l:=self::label return
count((//select|//input|//textarea)[@id=$l/@for]) != 1]
```

Cuadro 2

A esta condición, hay que añadirle la inversa, reflejada en el cuadro 3, es decir, la que detecta campos de formulario para los cuales no hay una única etiqueta descriptiva (tienen más de una o no la tienen).

```
(//select|//textarea|//input[@type="text" or @type="password" or
@type="radio" or @type="checkbox"])[let $ff:=self::node() return
count(//label[@for=$ff/@id]) != 1]
```

Cuadro 3

Las abreviaturas y acrónimos de un documento deben tener definiciones consistentes

• **Regla 4.2a:** Las abreviaturas y acrónimos de un documento deben tener definiciones consistentes. Debemos evitar situaciones en las que un mismo texto pueda estar definido de varias formas distintas (lo cual produciría confusión al lector). La figura 4 muestra un claro ejemplo de definición múltiple (y por lo tanto ambigua) del término UN, para el que se le dan dos acepciones completamente distintas.

```
<acronym title="United Nations">UN</acronym>
<abbr title="Unified Notation">UN</abbr>
```

Cuadro 4

Para evitar situaciones como la que aparece en el cuadro 4, se puede usar la expresión XQuery del 5.

```
(//abbr | //acronym)[let $a:=self::node() return
count((//abbr | //acronym)[text() = $a/text()]) != 1]
```

Cuadro 5

3.4. Requieren un algoritmo complejo

Se trata de comprobaciones que, aunque automatizables, no son expresables de forma declarativa con estándares del W3C, por lo que se suelen comprobar con algoritmos especializados mediante un lenguaje de programación. Muchos de ellos no son considerados por las herramientas de validación por su elevada complejidad, con lo que acaban siendo evaluados manualmente. Por ejemplo, la detección de un *suficiente contraste* de colores o una *inadecuada* señalización de cambios de idioma son cuestiones que, aunque automatizables, no están al alcance de la mayoría de los programadores y requerirían un software muy específico y costoso (razón por la cual muchas de ellas acaban en la práctica cayendo dentro de la siguiente categoría).

3.5.- Manuales

Son las comprobaciones que debe realizar una persona porque no están automatizadas. Normalmente se pueden clasificar en tres subgrupos. Aquellas que deben ser comprobadas a nivel de **sitio Web** o a **nivel de página** (se pueden garantizar durante el proceso de construcción) y aquellas que deben ser comprobadas en todos **los trozos de las páginas** (requieren una mayor atención).

4. XHTML Basic y accesibilidad

XHTML Basic [7] fue concebido en el W3C como una versión de XHTML orientada a la accesibilidad, especialmente de terminales inalámbricos o con funcionalidades limitadas. Su utilización como formato de publicación, ciertamente mejora los costes de generación de contenidos accesibles. Con este estudio se ha pretendido cuantificar esas mejoras. En este sentido, hemos redefinido el conjunto de 65 puntos de chequeo en otro de 103 más extenso, desdoblando aquellas condiciones que convivían implícitamente en un mismo punto de chequeo y detallando con más especificidad aquellas que estaban definidas de forma difusa.

XHTML Basic fue concebido como una versión de XHTML orientada a la accesibilidad, especialmente de terminales inalámbricos o con funcionalidades limitadas. Su utilización ciertamente mejora los costes de generación de contenidos accesibles

La tabla 2 presenta cómo se distribuyen estos 103 puntos de chequeo según su coste de detección tanto en HTML (en general) como en XHTML Basic. Los números de tabla indican el número de puntos de chequeo para cada uno de estos formatos, que son clasificables dentro de las categorías mencionadas en la sección 3 (implementabilidad). Por ejemplo, en XHTML Basic hay 39 puntos de chequeo garantizados por el mero uso de la gramática. Ello indica que todo documento que siga esta gramática tiene garantizado de por sí el cumplimiento de 39 de los 103 puntos, mientras que las otras versiones de XHTML apenas tienen garantizados 6. Por el contrario, los 8 puntos semiautomatizables en XHTML Basic son un conjunto sensiblemente menor que los 21 de la misma categoría en HTML.

Si consideramos el conjunto de reglas formalizables como aquellas que no necesitan ser programadas en un algoritmo, sino que basta con que estén especificadas declarativamente en un DTD o XML Schema o una regla XPath o una consulta XQuery (las que requieren un algoritmo complejo, las semiautomatizables y las manuales no las consideramos formalizables), tenemos que, mientras en HTML el 46% (48/103) de los puntos de chequeo son formalizables, esta cifra asciende a casi el 68% (70/103) en XHTML Basic. Como resultado podemos afirmar que los autores que garantizan la validez de sus documentos en formato XHTML Basic, tendrán menos comprobaciones adicionales que efectuar para determinar la accesibilidad de sus documentos respecto a los autores que utilicen otros dialectos de HTML/XHTML. Desde el punto de vista de la objetividad, XHTML Basic aumenta el número de comprobaciones automatizables objetivas y disminuye el de las subjetivas o manuales.

Tabla 2: Distribución de las WCAG en HTML y en XHTML Basic

Tipo de detección	HTML	Basic
con DTD/Schema	6	39
con XPath 1.0	30	19
con XQuery 1.0	12	12
con algoritmo complejo	10	5
semiautomatizable	21	8
manual	24	20
Total formalizables	48	70
Total no formalizables	55	33
Total	**103**	**103**

5.- Conclusiones

La accesibilidad no es fácil de conseguir si no se incorpora como disciplina durante el diseño de las páginas Web. Intentar evaluar o reparar las barreras de las páginas una vez creadas es mucho más costoso que intentar incorporar unos principios básicos y de *sentido común* durante su diseño. La mayoría de las reglas de accesibilidad sólo indican que el marcado HTML debe ser adecuado.

Las herramientas de evaluación sólo producen resultados incompletos que deben ser completados con una supervisión manual. No existen herramientas que cubran todos los puntos de la evaluación. Por otro lado, las herramientas existentes pueden ofrecer distintos resultados para una misma muestra de páginas debido a las *interpretaciones particulares* que de las normas WCAG tuvieron los programadores de estas herramientas. Para minimizar esta subjetividad y reducir el esfuerzo de programación, se proponen mecanismos de reglas basados en técnicas declarativas formalizables del W3C (DTD/Schema, XPath y XQuery) en lugar de algoritmos sui-generis sobre lenguajes de programación.

XHTML Basic no es la panacea para conseguir la accesibilidad, pero sí es el formato de publicación Web que más la facilita en la actualidad. Su uso reduce sensiblemente el esfuerzo de evaluación y mejora la accesibilidad de los documentos en el Web, aunque no supone ninguna garantía.

El trabajo en el que se ha basado este artículo ha recibido el apoyo de los proyectos INFOFLEX TIC2003-07208 y SIEMPRE TIC2002-03635 financiados por el Programa Nacional Español de Tecnologías de Información y Comunicaciones.

Vicente Luque Centeno, Carlos Delgado Kloos,
(vlc@it.uc3m.es), (cdk@it.uc3m.es)
Luis Sánchez Fernández,Norberto Fernández García,
(luis.sanchez@uc3m.es), (berto@it.uc3m.es)
Dpto. de Ingeniería Telemática - UC3M

Las herramientas de evaluación sólo producen resultados incompletos que deben ser completados con una supervisión manual

Bibliografía

[1] W3C, "Web Content Accessibility Guidelines 1.0, 5 May 1999". www.w3.org/TR/WCAG10
[2] W3C, "Techniques for Accessibility Evaluation and Repair Tools W3C Working Draft, 26 Apr. 2000". www.w3.org/TR/AERT
[3] W3C, "XML Path Language (XPath) Version 1.0 W3C Recommendation 16 Nov. 1999". www.w3.org/TR/xpath
[4] W3C, "XML Path Language (XPath) 2.0 W3C Working Draft 23 Jul. 2004" www.w3.org/TR/xpath20
[5] W3C, "XQuery 1.0: An XML Query Language W3C Working Draft 23 Jul. 2004". www.w3.org/TR/xquery
[6] W3C, "XHTML 1.0 TM The Extensible HyperText Markup Language (2nd Edition), A Reformulation of HTML 4 in XML 1.0, W3C Recommendation 26 Jan.2000, revised 1 Aug. 2002". www.w3.org/TR/xhtml1
[7] W3C, "XHTML Basic W3C Recommendation 19 Dec. 2000". www.w3.org/TR/xhtml-basic
[8] W3C, "XHTML TM 1.1 - Module-based XHTML, W3C Recommendation 31 May 2001". www.w3.org/TR/xhtml11
[9] W3C, "XHTML TM 2.0, W3C Working Draft 22 July 2004". www.w3.org/TR/xhtml2
[10] Watchfire, "Bobby Accessibility tool". bobby.watchfire.com/bobby/html/en/index.jsp
[11] CEAPAT, Fundación CTIC, Mº de Trabajo y Asuntos Sociales (IMSERSO), "Test de accesibilidad Web" www.tawdis.net
[12] Watchfire, "WebXACT Accessibility tool". webxact.watchfire.com
[13] Fondazione Ugo Bordoni, "Torquemada, Web for all". www.webxtutti.it/testa_en.htm
[14] Luque Centeno, V.; Delgado Kloos, C.; Sánchez Fernández, L.; Fernández García, N. "Device independence for Web Wrapper Agents" Presentada en Workshop on Device Independent Web Engineering (DIWE'04) 26. July 2004, Munich
[15] BOE, LEY 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSICE). www.congreso.es/public_oficiales/L7/CONG/BOCG/A/A_068-13.PDF
[16] Fundosa Teleservicios; "La accesibilidad en los portales universitarios en España". Evaluación técnica de la accesibilidad y valoración de la experiencia de usuario en 15 portales de univ. españolas; Dic. 2004; www.discapnet.es/inc/infoaccesibilidad/html/Portales_universitarios_detallada.htm