Hacia una Herramienta de Soporte para el Modelado Web con Accesibilidad

Rafaela Mazalu[1], Adriana Martín[1] Alejandra Cechich[1]

[1]Grupo de Investigación en Ingeniería de Software del Comahue (GIISCo)
Departamento de Ciencias de la Computación, Universidad Nacional del Comahue
(8300)Neuquén, Argentina
{rafaelamazalu@gmail.com, adrianaelba.martin@gmail.com, acechich@uncoma.edu.ar}

Abstract Web Accessibility is a basic attribute of quality in use and a player to a successful Web application. The principles of design called for all or universal design, are aimed at product design and user-friendly environments for as many people as possible, without the need to adapt or redesign it so specially. There are different tools and approaches to assist to the Accessibility evaluation of existing Web applications. In contrast, there are no similar efforts for the early design with Accessibility principles in mind. Designing Web applications to improve Accessibility implies the analysis of different concerns that can be linked through the use of techniques from Aspect-Oriented Software Development (AOSD). In this work we propose a tool's architecture based on AOSD design concepts and the Web Content Accessibility Guidelines 1.0 (WCAG 1.0) to support building *accessible* user interfaces.

Keywords: Web Accessibility, WCAG 1.0, User Interface Design, AOSD.

Resumen La Accesibilidad Web es un atributo básico de la calidad en uso y una protagonista esencial para el éxito de una aplicación Web. Los principios del denominado diseño para todos o diseño universal, tienen como objetivo el diseño de productos y entornos de fácil uso para el mayor número posible de personas, sin la necesidad de adaptarlos o rediseñarlos de forma especial. Existen distintos enfoques y herramientas para asistir a la evaluación de la Accesibilidad de aplicaciones Web existentes. En contraposición, no existen esfuerzos similares para el diseño temprano con los principios de Accesibilidad en mente. Diseñar aplicaciones Web para mejorar la Accesibilidad implica el análisis de distintos intereses que pueden asociarse mediante la aplicación de técnicas provenientes del Desarrollo de Software Orientado a Aspectos (AOSD). En este trabajo proponemos una arquitectura de herramienta basada en los principios de diseño AOSD y las Web Content Accessibility Guidelines 1.0 (WCAG 1.0) para soportar la construcción de interfaces de usuario *accesibles*.

Palabras claves: Accesibilidad Web, WCAG 1.0, Diseño de Interfaz de Usuario, AOSD.

1 Introduction

La Accesibilidad aplicada al contenido de la Web se denomina Accesibilidad Web y básicamente se refiere a que las personas con discapacidad puedan acceder y usar la Web. Específicamente, significa que todas las personas (aún aquellas con limitaciones propias o derivadas del contexto de uso) puedan percibir, entender, navegar e interactuar con la Web, aportando a su vez contenidos [4]. El Word Wide Web Consortium (W3C)[1] es el principal ponente en Accesibilidad Web y trabaja incansablemente desde hace más de diez años en el desarrollo de guías de Accesibilidad denominadas Web Content Accessibility Guidelines 1.0 (WCAG 1.0) [11] que son consideradas referentes para la mayoría de las legislaciones sobre Tecnología de la Información y Comunicación en todo el mundo. Basados en estas recomendaciones, un gran número de herramientas y enfoques han surgido en los últimos años y están disponibles para asistir a los desarrolladores Web en la generación de contenido *accesible*.

Debido a que la interfaz de usuario de una aplicación Web es: (i) El principal punto de contacto entre el usuario y la computadora, (ii) la parte de la aplicación Web con la que el usuario interactúa, y (iii) la encargada de informar las posibles acciones a realizar, los cambios producidos y estado actual de la aplicación, es justamente a nivel de interfaz de usuario donde finalmente se manifiestan las barreras de Accesibilidad. En la arena de la Accesibilidad Web, existen numerosos trabajos [1][2][5][7] que, desde diferentes perspectivas buscan concientizar en la incorporación de los principios de Accesibilidad. A pesar de ello, es un hecho que a la hora de diseñar interfaces *accesibles* sólo se evalúa la conformidad a los principios de Accesibilidad que dictaminan las guías y estándares vigentes [6][8][11] sobre la aplicación corriendo. En el mejor de los casos, esta evaluación se efectúa durante la etapa de implementación de la aplicación Web, ya que generalmente la misma tiene lugar post implementación, vía una herramienta de evaluación libre o comercial (tales como Bobby[2], Lift[3], TAW[4]). Esta práctica tardía de evaluación de la Accesibilidad demanda un esfuerzo extra de re-desarrollo (en diseño e implementación), muy costoso en tiempo y dinero. Como contraparte, una práctica temprana que reconozca los requerimientos de Accesibilidad propiciaría mantener vigentes los mismos desde la concepción hasta la implementación de la aplicación Web. Sobre esta idea y a los efectos de lograr buenos diseños de interfaces de usuario, existen líneas de investigación que aplican los principios del Desarrollo de Software Orientado a Aspectos (AOSD) para tratar a la Accesibilidad como un requerimiento generador de *intereses* ("concerns" en la terminología AOSD) esenciales al desarrollo de aplicaciones Web con interfaces *accesibles*. En los hechos, una buena separación en *intereses* permite identificar y manejar en forma individual los distintos componentes de una aplicación, para atender a los principios de Accesibilidad en las distintas etapas del desarrollo (requerimientos, diseño, implementación).

---

[1] Ver http://www.w3.org/

[2] Ver http://webxact.watchfire.com/

[3] Ver http://www.usablenet.com/

[4] Ver http://www.tawdis.net/taw3/cms/es

En este trabajo proponemos una arquitectura de herramienta basada en los principios de diseño AOSD y las Web Content Accessibility Guidelines 1.0 (WCAG 1.0) para soportar la construcción de interfaces de usuario *accesibles*.

El trabajo se organiza de la siguiente manera: la Sección 2 introduce los antecedentes que constituyen la base conceptual de la propuesta; la Sección 3 presenta y detalla la arquitectura de nuestra herramienta; la Sección 4 valida dicha arquitectura a través de un caso de estudio. Finalmente la Sección 5 presenta las conclusiones y el trabajo futuro.

2 Diseño de Interfaz de Usuario y Accesibilidad

Para modelar los requerimientos no funcionales, el Softgoal Interdependency Graph (SIG) ha sido ampliamente utilizado por la ingeniería del software [3][9]. Ya en el terreno específico de la ingeniería Web y extendiendo la perspectiva tradicional de estos trabajos, en [5] se propone un conjunto de herramientas conceptuales para permitir la representación de los requerimientos de Accesibilidad derivados de la interacción usuario-aplicación. Esta representación permite capturar de manera temprana los *intereses* de Accesibilidad y que los mismos acompañen todo el proceso de desarrollo de una interfaz de usuario *accesible*. La captura de estos *intereses* de Accesibilidad se lleva a mediante la extensión de dos tipos de diagramas: el User Interaction Diagram (UID) [10] con *puntos de integración* de Accesibilidad [5] y la *plantilla* SIG de Accesibilidad que refleja directamente los *intereses* de Accesibilidad según las guías de la WCAG 1.0 [5]. A continuación describiremos brevemente ambas herramientas conceptuales.

2.1 UIDs con puntos de integración de Accesibilidad

Para identificar los requerimientos de Accesibilidad en etapas tempranas del proceso de desarrollo, en [5] se propone un diagrama UID cuyo propósito modelar los *intereses* de Accesibilidad presentes en la interacción usuario-aplicación, se proponen dos tipos de *puntos de integración*:

Punto de integración para una interacción Usuario-UID (U-UI). Es un *punto de integración* para representar la Accesibilidad requerida a nivel de interacción del UID, es decir para propiciar un intercambio de información *accesible* entre el usuario y una interacción específica del UID. Por ejemplo, un *punto de integración* a nivel de interacción (U-UI), puede representar la necesidad de que el contenido que constituyen la interfaz concreta esté dispuesto de izquierda a derecha y de arriba hacia abajo, para que la presentación de la página respectiva pueda ser leída por un "screen reader"[5].

Punto de integración para un componente de una interacción Usuario-UID (U-UIc). Es un *punto de integración* para representar la Accesibilidad requerida a nivel de un componente de una interacción del UID, es decir para propiciar un intercambio de información *accesible* entre el usuario y un componente en particular de una interacción del diagrama UID. Por ejemplo, un *punto de integración* a nivel de

[5] Aplicaciones y dispositivos especiales que vocalizan los datos de la pantalla. Las páginas se leen desde el extremo superior izquierdo al extremo inferior derecho, línea por línea y de a una palabra a la vez.

componente de interacción (U-UIc), puede representar la necesidad de que un campo destinado al ingreso de un dato tenga una etiqueta en la interfaz concreta, para que en la presentación provea al usuario de una leyenda que explique qué dato se debe ingresar y cómo se debe ingresar dicho dato, en el campo.

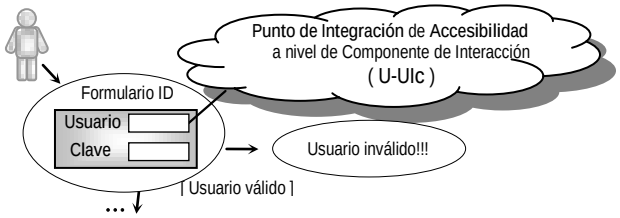

Figura 1. UID con punto de integración a nivel de componente de interacción [5]

La Figura 1 muestra un ejemplo de UID con *puntos de integración* [5] correspondiente al proceso típico de identificación de usuario en un sistema. Como se puede apreciar se ha definido un U-UIc para el componente de interacción correspondiente al formulario ID de usuario, para modelar tempranamente sus requerimientos de Accesibilidad. Para más detalles sobre el diagrama UID con *puntos de integración* ver [5].

2.2 Accesibilidad con SIGs

Para completar la captura de los intereses de Accesibilidad, una vez diseñado el UID con *puntos de integración*, en [5] se propone desarrollar un diagrama SIG que permite mapear estos *intereses* a *checkpoints*[6] de Accesibilidad concretos correspondientes a la WCAG 1.0. Mientras que el UID refleja básicamente el diálogo, es decir ayuda a modelar el contenido y la secuencia de intercambio de información usuario-aplicación, el SIG además modela las cuestiones de presentación y pragmática[7]. La Figura 2 muestra la *plantilla* SIG [5] donde el "softgoal"[8] de Accesibilidad, que representa al respectivo *punto de integración* del UID, constituye la raíz del árbol. El tipo de *punto de integración*, U-UIc, se destaca en la nube de la raíz. En este nodo raíz se originan dos ramas iniciales: (i) Soporte de Tecnología para el Usuario y (ii) Soporte de Presentación para el Usuario. El soporte tecnológico representa la influencia de los conceptos pragmática y tecnología en los "softgoals" de Accesibilidad, mientras que el soporte de presentación representa la influencia de la elección de contenidos y objetos específicos para la interacción con el usuario. Cabe volver a señalar que el diagrama SIG garantiza la Accesibilidad a través de las pautas de las WCAG 1.0. El proceso de instanciación de la *plantilla* SIG es conducido como un proceso de refinamiento sobre el árbol SIG el cual finalmente refleja los *checkpoints* que se deben satisfacer para lograr un diseño de interfaces de usuario *accesible*. Para más detalles sobre la *plantilla* SIG de Accesibilidad ver [5].

---

[6] Ver http://www.w3.org/TR/WCAG10/full-checklist.html

[7] En el campo del diseño de interfaz de usuario, se denominan tópicos de "layout" y "pragmatic" respectivamente.

[8] Un "softgoal" representa un requerimiento de calidad que puede contribuir positiva o negativamente a alcanzar otro "softgoal" y que puede o no ser satisfecho.

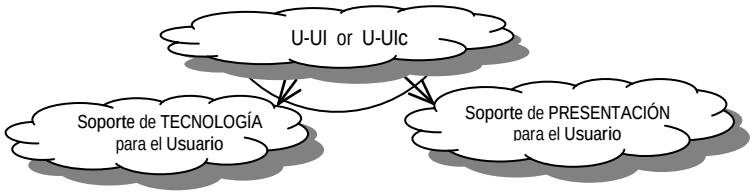

Figure 2. Plantilla SIG de Accesibilidad [5]

3 Una Arquitectura para Diseñar con Accesibilidad

En el contexto descripto por la secciones anteriores, nuestro objetivo es proveer soporte a los desarrolladores de aplicaciones Web, para la transformación de una interfaz de usuario abstracta *accesible* en una interfaz de usuario concreta *accesible.* Hemos concebido este soporte como una herramienta que dentro de un proceso de desarrollo prioriza la Accesibilidad de manera temprana y automatiza la tarea de desarrollar una interfaz concreta *accesible*, asistiendo para ello al desarrollador a través de técnicas de diseño AOSD e interfaces gráficas. Dichas técnicas de diseño están basadas en transformar el modelo conceptual de entrada (representado en los modelos de interfaz abstracta y SIG de Accesibilidad) en su respectiva interfaz concreta (representada por un archivo HTML bien formado[9] y *accesible*[10]).

En primer término y a los efectos de definir cuáles son los componentes y cómo se van a relacionar en el ambiente de desarrollo, la Figura 3 muestra la arquitectura simplificada para nuestra herramienta. Dicha arquitectura está compuesta por módulos que interactúan brindando los servicios necesarios para satisfacer los requerimientos de otro módulo, o bien del desarrollador de interfaces. Tal como ilustra la Figura 3 los módulos que componen la arquitectura propuesta son: (1) la Capa de *Presentación*, (2) la Capa de *Dominio* y (3) la Capa de *Depósito de Objetos*. A simple vista puede notarse que la arquitectura propuesta responde al patrón arquitectural Model View Controller (MVC) pues separa en tres componentes los datos, la interfaz de usuario y la lógica que efectúa el control. La Figura 3 nos permite observar la entrada E/ y la salida /S de nuestra herramienta. La E/ consta de un diagrama SIG de Accesibilidad y el modelo de la interfaz abstracta que provee el diseñador de interfaces de usuario; mientras que la /S es un HTML bien formado y *accesible*. A continuación, en las secciones sucesivas, describiremos cómo son y qué función cumple cada uno de estos módulos.

---

[9] Un HTML bien formado "well formed" tiene una estructura que está de acuerdo con las reglas definidas por los documentos de la W3C. Ver http://www.w3.org/TR/REC-html40/intro/intro.html#h-2.3.2

[10] Es conveniente aclarar que un HTML bien formado no asegura la Accesibilidad. Por ejemplo, un formulario puede tener bien definido el campo para el ingreso de una clave pero no tener una etiqueta descriptiva que facilite al usuario la manipulación de dicho campo. De ahí que se pretende como salida a nuestra herramienta un HTML bien formado y *accesible*.

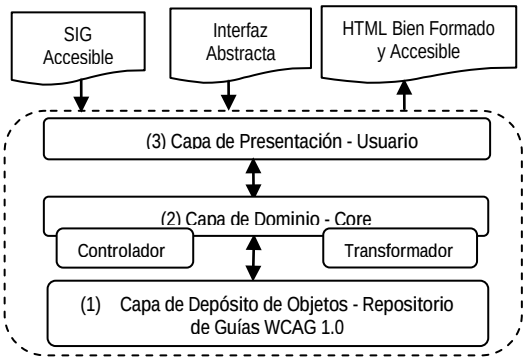

Figura 3. Arquitectura de nuestra herramienta

3.1 Capa de Presentación - Interfaz Gráfica de Usuario

La capa de *Presentación* es la encargada de capturar el modelo de interfaz abstracta (provisto por el diseñador) que será transformado en un modelo de interfaz concreta para la aplicación Web. Además, con el fin de obtener una interfaz concreta *accesible*, la herramienta también captura los principios de las WCAG 1.0 a través del modelo SIG de Accesibilidad (también provisto por el diseñador) que guiará luego el proceso de transformación. La capa de *Presentación* realiza el tratamiento considerando los *aspectos*[11] de Accesibilidad que dan soporte de Presentación y de Tecnología para el usuario (ramas del SIG de Accesibilidad), de manera que el desarrollador pueda visualizar cómo los *checkpoints* de WCAG 1.0 se aplican en la interfaz concreta resultante. Es importante aclarar que la capa de *Presentación* se comunica únicamente con la capa de *Dominio*, de manera que la primera se desacopla de los controles y procesos realizados en capas inferiores. Así, la capa de *Presentación* administra la calidad y naturaleza de la comunicación con el desarrollador y con la capa inmediatamente inferior que genera información luego de realizar el procesamiento solicitado por el desarrollador. Por ello, al momento de determinar la arquitectura de nuestra herramienta, se ha considerado que el desarrollador podrá realizar dos tareas al momento de interactuar con la herramienta: (i) Seleccionar los servicios disponibles en la herramienta con los que desea interactuar (se proporcionarán mecanismos de estructuración, acceso y búsqueda), (ii) Interactuar con un servicio concreto de la herramienta (se proporcionará información de las funcionalidades, por ejemplo para trabajar en el diseño de una nueva interfaz concreta considerando los *checkpoints* de un cierto *aspecto* de Accesibilidad).

3.2 Capa de Dominio – Core

La capa de *Dominio* se divide en dos módulos: el controlador y el transformador. El controlador gestiona los pedidos realizados por la capa de *Presentación* y los manipula considerando los *aspectos* de Accesibilidad, es decir, el controlador es el que decide que vista se muestra al desarrollador y que información es la que se envía según el *aspecto* sobre el cual se esté trabajando. Además establece sobre que objetos

[11] Un aspecto “aspect” es una manera de realizar un interés “concern” en software.

de entrada trabajará el transformador. El módulo transformador tiene por objeto generar una interfaz concreta implementada en un HTML bien formado que además responde a los *checkpoints* partes de cada uno de los *aspectos* modelados por el SIG de Accesibilidad. De esta manera, los "widgets"[12] abstractos que componen una interfaz abstracta son mapeados en "widgets" HTML concretos bien formados y *accesibles*. Para poder efectuar ésta transformación, la capa de *Dominio* debe solicitar al *Depósito de Objetos*, cómo componer la interfaz concreta para que exhiba en dichos "widgets" los *checkpoints* de los *aspectos* modelados por el SIG de Accesibilidad. Como veremos a continuación, esta información está estructurada y almacenada en el repositorio de guías de la WCAG 1.0.

3.3 Capa de Depósito de Objetos

Para que nuestra herramienta sea capaz de entregar una interfaz concreta *accesible*, el proceso de transformación guiado por el SIG de Accesibilidad, debe disponer de una representación almacenada de elementos de datos concretos y *accesibles* para cada uno de los "widgets" requeridos en la interfaz concreta por la interfaz abstracta. Para dar soporte a esta necesidad, hemos concebido la capa de *Depósitos de Objetos*. La misma almacena los elementos de datos *accesibles* que incumben al manejo de los "widgets" de interfaz y recibe las solicitudes de recuperación de estos elementos por *aspectos* de Accesibilidad que realiza la capa de *Dominio*. Este repositorio almacena los elementos de datos especificados según las WCAG 1.0 y podrá ser utilizada por la propia herramienta para recuperar dichos elementos con el fin de importarlos por *aspectos* en el modelo de interfaz concreta y de esta manera concebirla *accesible*.

Dado que un lenguaje de especificación es requerido, nuestra propuesta es que dicho lenguaje sea el eXtensible Markup Language (XML)13. Esta elección se debe a que XML es un lenguaje sencillo y potente, que permite expresar información compleja, en forma clara y estructurada, generando documentos bien formados y altamente transportables14. El propósito del uso de este lenguaje es especificar en una estructura admisible y universal los *checkpoints* de Accesibilidad correspondientes a los "widgets" de interfaz brindando la mayor cantidad de detalles relevantes para la manipulación de dichos *checkpoints* según lo indique la capa de Dominio en aspectos de Accesibilidad. Éste módulo también contiene los procedimientos y esquemas necesarios para la correcta transformación del modelo de interfaz abstracta a uno concreto expresado en un documento HTML bien formado y *accesible*.

---

[12] Un "widget" es un elemento en la interfaz gráfica de usuario (GUI). Existen dos tipos de "widgets": (i) "widget" abstracto que determina un tipo de funcionalidad para al usuario (ii) "widget" concreto que es la implementación actual del "widget" abstracto en un leguaje de marcado "mark-up language" o entorno de ejecución dado.

[13] Ver http://www.w3.org/XML/s

[14] XML ha respondiendo no solo al desafío de publicación electrónica a gran escala sino que también desempeña un papel cada vez más importante en el intercambio de una amplia variedad de datos y modelos en la Web.

4 Caso de Estudio

Un caso muy común de página Web es la que permite la identificación de un usuario por cierta aplicación o sitio. Tal como muestra la Figura 1 el proceso de identificación de usuario implica normalmente el ingreso del usuario y clave creando una interacción usuario-aplicación. Para mostrar la funcionalidad de nuestra herramienta, a continuación proveemos un detalle paso-a-paso de su utilización tomando como ejemplo este caso de estudio.

Paso 1. Tal como señalamos la E/ a nuestra herramienta es el modelo de interfaz abstracto y el SIG de Accesibilidad donde el diseñador ha capturado los requerimientos de funcionalidad y Accesibilidad tal como los trasmitió el UID con *puntos de integración* respectivo. La Figura 4 muestra parte del SIG de Accesibilidad correspondiente a la página Web de identificación de usuario. Este modelo fija los requerimientos de las WCAG 1.0 (expresados en "softgoals" de Accesibilidad) para los dos campos de entradas de texto (usuario y clave del formulario ID) desde la perspectiva de la Tecnología y de la Presentación para el usuario (ramas del SIG).

Paso 2. El desarrollador busca en la capa de *Presentación* la funcionalidad con la que desea operar para obtener un modelo interfaz concreta y *accesible*. La Figura 4 muestra parte del SIG de Accesibilidad correspondiente a la página de identificación de usuario que junto con el modelo de interfaz abstracta exhibirá la capa de *Presentación*. La capa de *Presentación* transfiere los elementos capturados a la capa de *Dominio* para su procesamiento.

Paso 3. El desarrollador se concentra en las especificaciones de diseño para la página de identificación del usuario, por ejemplo observando los requerimientos de Accesibilidad para las etiquetas "labels" que acompañan a los campos de entrada de texto "textfields" del formulario ID. Para la creación de la interfase concreta, el controlador de la capa de *Dominio* toma el conocimiento de Accesibilidad capturado por el SIG en *checkpoints* de la WCAG 1.0 que abarcan y atraviesan los "widgets" de interfaz necesarios para el formulario ID. La Figura 4 ilustra qué *checkpoints* deben verificarse para un formulario ID *accesible*. El controlador identifica en el SIG los *aspectos* de soporte de Tecnología y de Presentación para el usuario, que atraviesan ("crosscut") los dos campos para la captura de usuario y clave en el formulario ID. Luego, el controlador identifica qué *checkpoints* son encapsulados en cada *aspecto*. Por ejemplo, el *aspecto* tecnológico atraviesa a las etiquetas de los campos usuario y clave con los *checkpoints 9.5* y *10.2*, mientras que el *aspecto* de presentación atraviesa dichos campos con el *checkpoint 12.4*. Una vez realizado el proceso de reconocimiento del SIG de Accesibilidad, el controlador transfiere el control al transformador.

Paso 4. El transformador interactúa con la cada de Depósito de Objetos, de la cual obtendrá la información para generar la estructura de la interfaz concreta. Por cada *checkpoint* identificado por el controlador, el transformador captura los requerimientos de Accesibilidad y recupera los elementos de datos desde las estructuras basadas en XML del *Depósito de Objetos* que son necesarios para la generación del código HTML bien formado y *accesible*. En primer término, el transformador localiza en la capa de *Depósito de Objetos* el *checkpoint 9.5*, el cual requiere que se proporcionen atajos de teclado para los campos de entrada de texto del formulario ID, asegurando la independencia de dispositivo. La información

recuperada desde la capa de *Depósito de Objetos* explicita el uso del atributo "accesskey", que el transformador aplica a cada uno de los campos de entrada de texto del formulario ID en el HTML resultante.

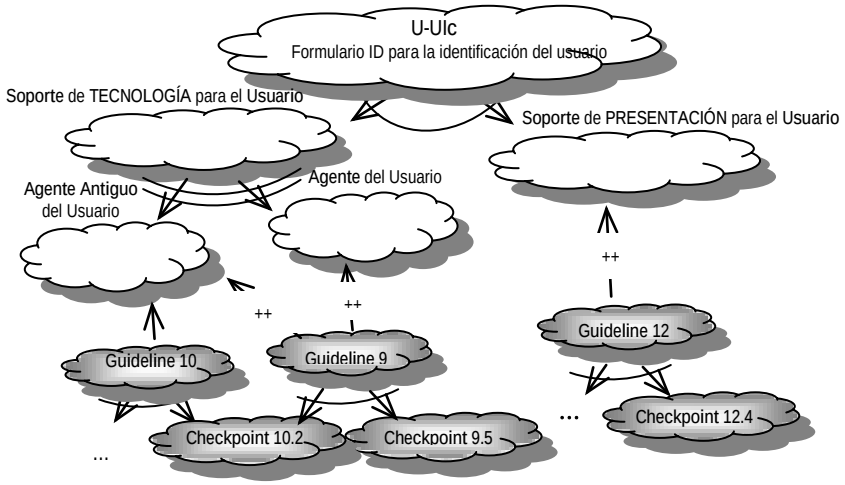

Figura 4. Parte del SIG de Accesibilidad para la página Web de identificación del usuario

Seguidamente, el transformador busca el *checkpoint* 10.2 el cual requiere que mientras existan agentes de tecnología antiguos que no soporten manejar apropiadamente la asociación entre un campo y su respectiva etiqueta, se asegure al dispositivo el posicionamiento adecuado de dicha etiqueta. Entonces, el transformador aplica a la interfaz concreta HTML del formulario ID, los posicionamientos (o marcas) necesarios para vincular adecuadamente las etiquetas a sus respectivos campos de entrada de texto. Posteriormente el transformador busca el *checkpoint* 12.4, el cual requiere que se asegure en la presentación la asociación explícita de las etiquetas con sus respectivos campos de entrada de texto. Entonces, el transformador aplica a la interfaz concreta HTML asociaciones explícitas entre las etiquetas y sus respectivos campos de entrada de texto mediante el atributo "for". Finalmente, el proceso de diseño llevado a cabo en la capa de *Dominio*, provee a la capa de *Presentación* (y a través de ella al desarrollador) del diseño de interfaz de usuario *accesible* para la página Web de identificación de usuario. La porción de HTML bien formado y *accesible* resultante del Paso 4 es la siguiente:

```
<LABEL for="idNumber" accesskey ="I"> Identificación:</LABEL>
   <INPUT type="text" id="idNumber" size="8" tabindex="1">
<LABEL for="idPassword" accesskey = "C"> Clave:</LABEL>
   <INPUT type="text" id="idPassword" size="6" tabindex="2">
```

5 Conclusiones y Trabajo Futuro

El desarrollo de aplicaciones Web *accesibles* es un factor fundamental para la concreción del principio básico de acceso universal. Sin embargo, los diseñadores de aplicaciones Web han desarrollado y siguen desarrollando desconociendo a los

potenciales beneficiarios de un desarrollo *accesible*. Motivados por esta realidad, desarrollamos nuestra propuesta que asiste con las directivas de Accesibilidad de la WCAG 1.0 al desarrollo de interfaces de usuario *accesibles.* Nuestra herramienta de soporte hace posible que se concreten los principios de diseño *accesible* especificados por el diseñador de interfaces abstractas en interfaces concretas de usuario para aplicaciones Web *accesibles*. En este trabajo proponemos la arquitectura de una herramienta que permite enfrentar la problemática de automatizar el proceso de construcción de interfaces concretas *accesibles* a partir de sus respectivos modelos de interfaces abstractas y SIGs de Accesibilidad provistos por el diseñador de interfaces de usuario. En lo inmediato esta pendiente enriquecer la capacidad de la herramienta, incorporando extensiones que permitan también automatizar el modelado del SIG de Accesibilidad a partir del respectivo UID con *puntos de integración*.

Agradecimientos

Este trabajo es resultado del proyecto de investigación UNComa 04/E072 (Identificación, Evaluación y Uso de Composiciones Software).

Referencias

1. Bustos Torres, B., Martín, A. and Cechich, A.: Extendiendo MVC para Diseñar Interfaces de Usuario Accesibles. In XIV Congreso Argentino en Ciencias de la Computación, pp. 1163-1174. CACIC (2008)
2. Centeno, V., Kloos, C., Gaedke, M., Nussbaumer, M.: Web composition with WCAG in mind. In International Cross-Disciplinary Workshop on Web Accessibility (W4A), pp. 38-45. ACM Press (2005)
3. Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000)
4. ISO/TS 16071 International Organization for Standardization/Technical Specification. Ergonomics of Human-System Interaction - Guidance on Accessibility for Human-Computer Interfaces (2002)
5. Martín, A. Cechich, A. and Rossi, G.: A Three-Layered Approach to Model Web Accessibility for Blind Users. In 5th Latin American Web Congress (LA-WEB), pp. 76-83. IEEE Computer Science Press (2007)
6. PAS 78 Publicly Available Specification: A Guide to Good Practice in Commissioning Accessible Websites, ICS 35.240.30. Disability Rights Commission (DRC) http://www.drc.org.uk/library/website_accessibility_guidance/pas_78.aspx. (2006)
7. Plessers P. , Casteleyn S. , Yesilada Y. , De Troyer O. , Stevens R. , Harper S. & Goble C.: Accessibility: A Web Engineering Approach. In 14th International World Wide Web Conference (WWW), pp. 353-362. ACM press (2005)
8. Section 508 US Federal Government: A Quick Reference Guide to Section 508 Resource Documents from http://www.accessibilityforum.org/paper_tool.html (2003)
9. Supakkul, S., Chung, L.: Integrating FRs and NFRs: A Use Case and Global Driven Approach. In 2nd International Conference on Software Engineering Research and Applications, pp. 29-41. SERA (2004)
10. Vilain, P., Schwabe, D., de Souza, C.: A Diagrammatic Tool for Representing User Interaction in UML, In 3rd International Conference on the Unified Modelling Language, UK (2000)
11. WCAG 1.0 Web Content Accessibility Guidelines 1.0. World Wide Web Consortium (W3C) Recommendations http://www.w3.org/TR/WAI-WEBCONTENT/ (1999)