Facultad de Ciencias de la Computación

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema de Evaluación Automática de Usabilidad de Interfaces de Usuario

Tesis

Para obtener el título de

Licenciatura en Ingeniería en Ciencias de la Computación.

Presenta

Cesar Manuel Rodríguez Mendiola.

Asesores

Dr. Juan Manuel González Calleros.
Dra. Josefina Guerrero García.

Heroica Puebla de Zaragoza Septiembre 2018.

Copyright © 2018 por Cesar Manuel Rodríguez Mendiola. Todos los derechos reservados.

Dedicatoria

A DIOS.
Dedico este trabajo de tesis a Dios por colmarme de perseverancia y permitirme llegar hasta este día con plena salud.

A MIS PADRES.
Dedico esta tesis a mi madre Graciela Mendiola Escobar y a mi padre Rafael Cesar Rodríguez Sánchez, quienes siempre me han brindado su apoyo incondicional y me han motivado a superar cada obstáculo en mi corta trayectoria.

A MIS ABUELOS.
Entre las personas más importantes están mis abuelos Cecilia Sánchez y Manuel Rodríguez quienes incluso en momentos complicados me han dado fuerzas para retomar el buen camino del conocimiento.

A MIS HERMANAS.
No por dejarlas al último son menos importantes, este gran logro también conlleva esfuerzo y sacrificio de mis tres hermanas, quienes me apoyaron y aportaron su granito de arena para que pueda culminar mi carrera profesional.

Gracias por tu apoyo,
compañía y amor
R.R.C.

Agradecimientos

A MIS ASESORES DE TESIS.

Agradezco de antemano al Dr. Juan Manuel González Calleros y a la Dra. Josefina Guerrero García, quienes me apoyaron, condujeron e impartieron conocimiento para poder realizar esta tesis, sobre todo quiero agradecer su tiempo y disposición incondicional durante estos meses de trabajo.

A MIS PROFESORES.

Agradezco a todos y cada uno de los profesores que voluntaria e involuntariamente participaron en mi formación profesional e incluso algunos participan en mi formación personal, quisiera nombrar a cada uno de ellos pero me reservo de hacerlo por temor a dejar fuera a alguno, sin embargo, sé que quienes más me han compartido su conocimiento y experiencia sin escatimar… están leyendo estas líneas.

Gracias profes…

Resumen

La usabilidad es fundamental para el éxito de sistemas informáticos. Las pruebas y evaluaciones de usabilidad durante el desarrollo de software han ganado amplia aceptación como estrategia para mejorar la calidad del producto. La introducción temprana de las perspectivas de usabilidad en un producto es muy importante para brindar una clara visibilidad de aspectos de calidad, tanto para desarrolladores, como los analistas de pruebas. Sin embargo, en la actualidad, someter el producto a rigurosos análisis de usabilidad es una tarea engorrosa. En esta tesis se desarrolla una herramienta para reducir el estrés y tiempo invertido en la realización de pruebas de usabilidad durante el desarrollo formal de software.

Abstract

The usability is essential for success of computer systems. The assay and evaluation of usability during software development have gained wide acceptance as strategy to improve product quality. The early introduction of usability perspectives in product is very important to provide a clear visibility of quality aspects, both for developers and test analyst. However, at present, submit the product to strict usability analysis is a cumbersome task. In this thesis a tool is developed to reduce the stress and time invested in the realization of usability tests during the formal development of software.

Tabla de Contenidos

Capítulo 1 Introducción 1
1.1. Motivación 2
1.2. Problemática 2
1.3. Objetivos 3
1.3.1. Desarrollo del sistema (Usability Adviser) 3
1.3.2. Registro de derechos de autor (INDAUTOR) 3

Capítulo 2 Estudio del estado del arte 4
2.1. Clasificación de los métodos de evaluación de usabilidad 4
2.1.1. Lugar 5
2.1.2. Técnica 5
2.1.3. Participantes 6
2.2. CAMELEON 6
2.2.1. Tareas y Conceptos (T&C) 7
2.2.2. Interfaz de usuario abstracta (AUI). 7
2.2.3. Interfaz de usuario concreta (CUI) 8
2.2.4. Interfaz de usuario final (FUI) 8
2.3. USIXML 8
2.4. GrafiXML 9
2.4.1. FlowBox 10
2.4.2. GridBox 10
2.4.3. BorderBox 10
2.4.4. GridBagBox 10
2.4.5. Componentes 10
2.5. Análisis sintáctico 12
2.6. HTML 12
2.7. PHP 12
2.8. JavaScript 13
2.9. JQuery 14
2.10. AJAX 14
2.11. Google analytics 15
2.11.1. Usuarios 15
2.11.2. Sesiones 15
2.11.3. Porcentaje de rebote 15
2.11.4. Duración de la sesión 15
2.11.5. En este momento 16
2.11.6. Páginas vistas 16
2.11.7. Ubicaciones 16
2.12. Aplicaciones similares 16
2.12.1. PROKUS 16
2.12.2. SIRIUS 17
2.12.3. WebSAT 19
2.12.4. Metricspot 20

Capítulo 3 Arquitectura del Sistema 23
3.1. Estructura de reglas de usabilidad 25
3.1.1. Cabecera (Header) 26
3.1.2. Regla (Rule) 26
3.1.3. Caracteres reservados 29
3.2. Analizador sintáctico de reglas de usabilidad 30
3.3. Estructura interna de reglas de usabilidad 36
3.4. Estructura de archivos UsiXML (.usi) 36
3.4.1. prólogo 37
3.4.2. uiModel 37
3.4.3. head 38
3.4.4. cuiModel 39
3.4.5. contextModel 39
3.5. Compilador de UsiXML 40
3.6. Estructura de archivos de Texto Plano (.txt) 44
3.7. Análisis de datos (UsiXML vs Guidelines) 45
3.7.1. RunGuideline 46
3.7.2. ApplyGuideline 47
3.7.3. subconjunto 49
3.7.4. cuentaErrores 50
3.7.5. validaCuantificador 51
3.6.7. validaRestriccion 52
3.6.8. validaLink 57
3.6.9. validaValor 59
3.6.10. getPosicion 60
3.7. Inclusión de Google Analytics 60

Capítulo 4 Sistema Automático de Evaluación. 62
4.1. Reglas de usabilidad 62
4.2. Cargar archivos 63
4.2.1. Chooser file 64
4.2.2. Drag and drop 64
4.3. Tabla de archivos a evaluar 65
4.4. Selección de reglas a verificar 66
4.4.1. Guideline to evaluate 66
4.4.2. Guideline without evaluate 66
4.5. Reporte para certificador (Diseñador gráfico) 66
4.6. Reporte para desarrollador 69
4.7. Reporte de alertas 71

Capítulo 5 Validación de la Propuesta. 73
5.1. Caso de estudio 1 – Calculadora 73
5.2. Caso de estudio 2 – Calculadora mal diseñada 80
5.3. Caso de estudio 3 – Usability Adviser 85

Capítulo 6 Conclusiones y trabajos futuros. 92
6.1. Conclusiones 92
6.2. Trabajos futuros 93

Referencias 95

Lista de tablas

Tabla 2.1. Clasificación de los métodos de evaluación de la usabilidad. 4
Tabla 2.2. Componentes CUI de UsiXML. 11
Tabla 3.1. Tabla de valores (Links). 57
Tabla 4.1. Reglas de usabilidad en Usability Adviser. 63

Lista de figuras

Figura 2.1. Marco de referencia de interfaz de usuario [19]. 7
Figura 2.2. Sistema de evaluación SIRIUS [23]. 17
Figura 3.1. Proceso global para evaluación automática. 24
Figura 3.2. Estructura formal para reglas de usabilidad, propuesta en [4]. 25
Figura 3.3. Ejemplo de Guideline válida. 30
Figura 3.4. Diagrama de flujo-Analizador sintáctico de reglas de usabilidad. 31
Figura 3.5. Diagrama de flujo – Compilador de archivos UsiXML. 41
Figura 3.6. Estructura de archivos de texto plano (.txt) válida. 45
Figura 3.7. Relación entre submódulos para análisis de datos (UsiXML vs Guideline). 46
Figura 3.8. Distribución de vértices (is_correctly_balanced). 54
Figura 4.1. Carga de archivos (Chooser file). 64
Figura 4.2. Carga de archivos (Drag and drop). 65
Figura 4.3. Lista de archivos a evaluar (.usi, .txt o .java). 65
Figura 4.4. Listas de guidelines. 66
Figura 4.5. Reporte para certificador. 67
Figura 4.6. Exportación a Excel del reporte para certificador. 68
Figura 4.7. Exportación a PDF del reporte para certificador. 68
Figura 4.8. Reporte para desarrollador. 69
Figura 4.9. Exportación a Excel del reporte para desarrollador. 70
Figura 4.10. Exportación a PDF del reporte para desarrollador. 71
Figura 4.11. Reporte de alertas. 72
Figura 5.1. Interfaces de calculadoras. 74
Figura 5.2. Maquetado de calculadora en GrafiXML, Caso de estudio 1. 75
Figura 5.3. Vista previa de calculadora generada por GrafiXML, Caso de estudio 1. 75
Figura 5.4. Parámetros de entrada para el primer caso de estudio. 79
Figura 5.5. Reporte de errores vista Certificador, Caso de estudio 1. 80
Figura 5.6. Reporte de errores vista Desarrollador, Caso de estudio 1. 80
Figura 5.7. Reporte de alertas, Caso de estudio 1. 80
Figura 5.8. Maquetado incorrecto de calculadora en GrafiXML, Caso de estudio 2. 81
Figura 5.9. Vista previa de calculadora mal diseñada, Caso de estudio 2. 82
Figura 5.10. Parámetros de entrada para el segundo caso de estudio. 82
Figura 5.11. Reporte de errores vista Certificador, Caso de estudio 2. 83
Figura 5.12. Reporte de errores vista Desarrollador, Caso de estudio 2. 83
Figura 5.13. Reporte alertas, Caso de estudio 2. 83
Figura 5.14. Exportación a Excel del reporte de errores vista Certificador, Caso de estudio 2. 84
Figura 5.15. Exportación a PDF del reporte de errores vista Desarrollador, Caso de estudio 2. 84
Figura 5.16. Maquetado de Usability Adviser en GrafiXML, Caso de estudio 3. 85
Figura 5.17. Vista previa de Usability Adviser, Caso de estudio 3. 89
Figura 5.18. Parámetros de entrada para el tercer caso de estudio. 89
Figura 5.19. Reporte de errores vista Certificador, Caso de estudio 3. 90
Figura 5.20. Reporte de errores vista Desarrollador, Caso de estudio 3. 91
Figura 5.21. Reporte de alertas, Caso de estudio 3. 91

Capítulo 1

Introducción

En la actualidad, el desarrollo de interfaces para aplicaciones interactivas resulta complejo debido a los múltiples contextos de uso, perfiles de usuario, sistemas operativos y sobre todo las múltiples plataformas de ejecución. Por lo anterior, frecuentemente se integran más equipos de desarrollo multidisciplinarios principalmente conformados por administradores, desarrolladores informáticos y diseñadores gráficos quienes deben de compartir información y características específicas del proyecto. Así mismo, la evaluación de la usabilidad es una de las tareas más importantes que deben tomarse en cuenta cuando se desarrolla una interfaz de usuario.

En este sentido, los trabajos de Ferré Grau refieren que "Sin llevar a cabo algún tipo de evaluación es imposible saber si el sistema satisface las necesidades de los usuario y encaja adecuadamente en el contexto físico, social y organizacional en el que va a ser usado" [9].

Centremos la atención en el diseño y desarrollo de interfaces de usuario, los diseñadores gráficos son los encargados del diseño de la interfaz y certificación de la misma una vez implementada por los desarrolladores. Este último paso genera problemática entre el diseñador y el desarrollador dado que cada uno de los implicados interpreta de diferente manera la misma directriz de usabilidad, peor aún, los sistemas de evaluación automática existentes también tienen su propia interpretación.

En este trabajo de tesis, se plantea el desarrollo de un sistema informático para la evaluación automática de interfaces de usuario, tomando como base la metodología

propuesta y probada en el artículo Advance human-machine interface automatic evaluation [3]. Esta metodología consiste en: Cargar normas ergonométricas en la estructura de datos, realizar un análisis de la interfaz de usuario codificada en UsiXML, buscar violaciones a estas reglas y generar un reporte con información específica para el desarrollador, otro para el diseñador y uno más de alertas.

1.1. Motivación

La aplicación resultante de esta tesis, contribuye en el proceso de ingeniería de software para la realización de productos de calidad, específicamente en aquellos productos que contemplan el desarrollo de interfaces de usuario de forma sistemática para una amplia aceptación en el usuario final o cliente.

1.2. Problemática

El diseño e implementación de interfaces de usuario para aplicaciones interactivas requiere de una comunicación precisa entre diseñadores gráficos y desarrolladores informáticos además de una ardua evaluación, en ambos casos, la inadecuada interpretación de reglas ergonométricas y/o de usabilidad generan confusión en las partes involucradas, para la evaluación de las interfaces, frecuentemente se apoyan en herramientas que deben producir el mismo resultado cuando se evalúan automáticamente las directrices, lo cual no siempre se cumple.

1.3. Objetivos

El objetivo principal de esta tesis es desarrollar una herramienta para la evaluación automática de interfaces de usuario descritas en UsiXML, cuya aplicación es ampliamente aceptada en el proceso de ingeniería de software.

1.3.1. Desarrollo del sistema (Usability Adviser)

A lo largo de esta tesis se plantea el desarrollo de una aplicación web capaz de analizar e interpretar guías de desarrollo de interfaces de usuario, por otro lado analiza las interfaces de usuario descritas en UsiXML, una vez creadas ambas estructuras de datos (guidelines e interfaces) se realiza una búsqueda de guidelines violadas. Las violaciones a guidelines se presentan en dos formatos uno para el certificador y otro para el desarrollador.

1.3.2. Registro de derechos de autor (INDAUTOR)

Los resultados de esta tesis se presentarán ante el Instituto Nacional del Derecho del Autor (INDAUTOR) para el trámite correspondiente.

Capítulo 2

Estudio del estado del arte

En el presente capítulo se realiza una descripción del estado actual de trabajos, relacionados con este proyecto, tales como: PROKUS, SIRIUS, WebSAT y Metricspot. Así mismo se destaca la importancia de las interfaces de usuario en una aplicación de software, métodos formales que permiten la inclusión del diseño de la interfaz de usuario dentro de un proceso de desarrollo basado en modelos, describiendo algunas de estas metodologías a detalle y su clasificación.

Se introducen algunas de las técnicas, modelos y herramientas utilizadas durante el desarrollo de la herramienta Usability Adviser.

2.1. Clasificación de los métodos de evaluación de usabilidad

Los métodos de evaluación pueden clasificarse de varias formas [14], en la Tabla 2.1 se propone una clasificación.

Clasificación	*Tipo*
Lugar	Laboratorio
	Entorno real
Técnica	Inspección
	Indagación
	Test
Participantes	Con usuario
	Sin usuario

Tabla 2.1. Clasificación de los métodos de evaluación de la usabilidad.

2.1.1. Lugar

Considera el lugar de realización, se distinguen dos categorías generales:

Laboratorio: Las pruebas se realizan en un laboratorio especializado para pruebas de usabilidad e implica la participación de usuarios para un estudio particular.

Entorno real: Las pruebas se realizan en un entorno natural o habitual del escenario donde se realizan las tareas a evaluar.

2.1.2. Técnica

En este tipo se distinguen tres categorías:

Inspección: Expertos examinan aspectos de la interfaz del sistema relacionados con la usabilidad y la accesibilidad que la misma ofrece a sus usuarios. Estos métodos tienen en cuenta las opiniones, juicios e informes de los inspectores sobre elementos específicos de la interfaz como factor fundamental de la evaluación de la usabilidad. Son populares en el ámbito de las empresas de producción de servicios, pues permiten identificar, clasificar y contabilizar un gran número de errores potenciales de usabilidad a precio relativamente bajo.

Indagación: La información acerca del agrado del usuario, quejas, necesidades e identificación de requisitos es información indispensable, sobre todo en etapas tempranas del proceso de desarrollo para que proporcionen detalles del uso y las posibilidades de acceso de un producto. Este tipo se realiza hablando con los usuarios, observándolos y/o usando el sistema en el entorno real.

Test: En estos métodos, una muestra de usuarios trabajan en tareas concretas utilizando el sistema (o prototipo), y los evaluadores utilizan los resultados para ver cómo la interfaz de usuario da soporte a los usuarios con sus tareas.

2.1.3. Participantes

Con usuarios: La evaluación del sistema requiere de intervención directa de usuarios representativos, pudiendo también participar en las sesiones personas que sin ser usuarios finales tienen la condición de implicados en el sistema.

Sin usuarios: La evaluación es realizada por expertos con la ayuda de guiones, pautas o documentos que permitan hacer un seguimiento de lo que han hecho los usuarios mientras utilizaban el sistema.

2.2. CAMELEON

CAMELEON (Context Aware Modelling for Enabling and Leveraging Effective interactiON) [5] es una metodología enfocada en construir métodos y entornos para respaldar el diseño y desarrollo de sistemas informáticos interactivos multi-contexto.

Para fines de esta tesis, del proyecto Europeo CAMELEON destacamos el modelo de interfaz constituido por cuatro capas de abstracción ilustradas en la figura 2.1.

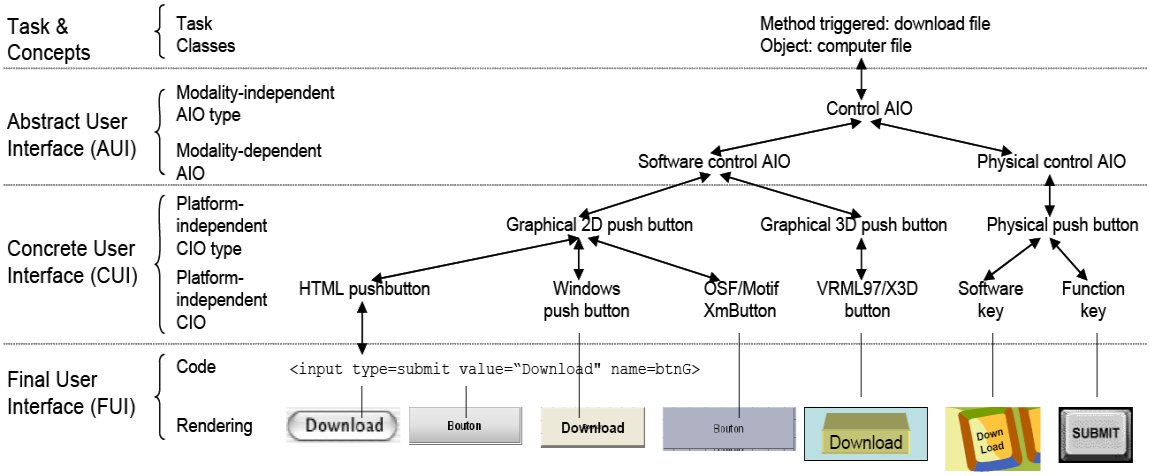


Figura 2.1. Marco de referencia de interfaz de usuario [19].

2.2.1. Tareas y Conceptos (T&C).

En esta etapa se definen las diferentes tareas que podrá realizar el usuario y el dominio de conceptos orientados como sean requeridos por estas tareas a realizar.

2.2.2. Interfaz de usuario abstracta (AUI).

En esta etapa se definen contenedores abstractos (AC) y componentes individuales abstractos (AIC), dos formas de abstraer objetos de interacción (AIO), agrupando las subtareas según el criterio (patrón estructural de modelo de tareas, análisis de carga de trabajo cognitivo, identificación de relaciones semánticas), un esquema de navegación entre el contenedor y la selección del componente individual abstracto para cada concepto de modo que sean independientes de cualquier modalidad.

2.2.3. Interfaz de usuario concreta (CUI)

La interfaz abstracta para un determinado contexto de uso es aterrizada en objetos de interacción concreta (CIO), a fin de definir widgets de interfaz y diseño de navegación. Un CIO se define como una entidad que los usuarios pueden percibir y/o manipular (por ejemplo, un botón, un listado, una casilla de verificación, un sonido, etc.). La especificación actual realiza una abstracción de conjuntos de widgets encontrados en kits de herramientas populares: gráficos 2D (Swing de Java, Flash, HTML 4.01) y auditivos (auriculares y voz XML 2.0).

En esta capa se centra el desarrollo de esta tesis.

2.2.4. Interfaz de usuario final (FUI)

En esta última etapa, se define la UI operacional, es decir, cualquier interfaz de usuario se ejecuta en una plataforma de computación en particular, ya sea por interpretación (a través de un navegador web) o por ejecución (después de la compilación de código en un entorno de desarrollo interactivo).

2.3. USIXML

UsiXML por sus siglas en inglés (User Interface eXtensible Markup Language) [18] es un lenguaje de descripción de interfaces de usuario (User Interface Description Language, UIDL), compatible con XML, que permite trabajar a diferentes niveles de abstracción. Describe interfaces de usuario para múltiples contextos de uso, tales como interfaz de usuario de caracteres (CUI), interfaz gráfica de usuario (GUI), interfaz

auditiva, interfaz multimodal, y sus elementos constitutivos como widgets, controles y contenedores, modalidades, técnicas de interacción, etc.

Utilizando UsiXML, tanto un desarrollador de interfaces de usuario como un analista, especificador, diseñador, líder de proyecto, etc., es capaz de modelar una interfaz de usuario para una nueva aplicación interactiva, sin requerir habilidades de programación como Java, C++, etc. Dado que el modelado es independiente del instrumento, plataforma y modalidad.

UsiXML proporciona un enfoque MDA para la especificación de interfaces de usuario y se basa en la arquitectura CAMELEON.

2.4. GrafiXML

GrafiXML [21] es una herramienta gráfica para diseñar interfaces de usuario independientemente de la plataforma, contexto o uso. Mediante plugins, las interfaces de usuario pueden ser guardadas en diversos formatos tales como: Java, XHTML, HTML, XUL aunque el formato nativo y por el cual nos interesa esta aplicación es UsiXML.

El entorno de desarrollo de GrafiXML, es muy similar al de cualquier otro editor, sin embargo, antes de añadir componentes a la interfaz generada por el usuario, la herramienta solicita que el usuario elija un layout para la distribución de los componentes en la nueva ventana.

Los layouts que ofrece son:

2.4.1. FlowBox

Posicionar un componente a continuación de otro. El usuario puede seleccionar la alineación del primer componente.

2.4.2. GridBox

Permite al usuario establecer el número de columnas y filas en las cuales se dividirá la venta, admitiendo un único componente por celda.

2.4.3. BorderBox

Divide la ventana en cinco secciones colocadas arriba, abajo, derecha, izquierda y al centro de la ventana. Permite al usuario habilitar cada una de las secciones y en su defecto elegir un layout para cada sección.

2.4.4. GridBagBox

Divide la ventana en casillas de un mismo tamaño y permite al usuario colocar componentes libremente por la ventana.

2.4.5. Componentes

A continuación se muestran diversos componentes individuales incorporados en GrafiXML y utilizados para dar forma a la especificación de la interfaz de usuario (Tabla 2.2), así mismo, servirán de referencia para el presente trabajo.

Componente	*Código UsiXML*	*Descripción*
window	<window id="w0" name="w_0" width="400" height="350"> “contenido” </window>	Típicamente este componente contendrá otros contenedores como ‘box’, ‘tables’, etc.
box	<box id="b1" name="b_1" type="vertical"> ... </box>	Hace la de función de contenedor de componentes gráficos.
button	<button id="b1" name="b_1" isVisible="true" isEnabled="true"/>	Los botones tendrán, además de las propiedades propias de un elemento gráfico, propiedades como si están activos o no.
radioButton	<radioButton id="r1" name="r_1" isVisible="true" isEnabled="true"/>	Botón de radio, características similares al anterior.
comboBox	<comboBox id="c1" name="c1" isVisible="true" isEnabled="true"/>	Múltiples opciones desplegables, características similares al anterior.
checkBox	<checkBox id="c3" name="c_3" isVisible="true" isEnabled="true"/>	Casilla de marcado, características similares al anterior.
listBox	<listBox id="l1" name="l_1" isVisible="true" isEnabled="true" multiple_selection="false"/>	Lista con diversas opciones a seleccionar, con la propiedad ‘multiple_selection’ se permite seleccionar varias opciones.
inputText	<inputText id="i1" name="i_1" isVisible="true" isEnabled="true" isPassword="false" maxLength="50" numberOfColumns="15" numberOfLines="2" isEditable="true"/>	Componente de introducción de texto, propiedades interesantes son la de la máxima longitud para introducir los datos, número de filas y columnas, si es posible editarlo y si es campo contraseña.
table	<table id="t3" name="t_3" isVisible="true" isEnabled="true" xSize="3" ySize="5"/>	Componente que representa a las tablas.
outputText	<outputText id="o1" name="o_1" isVisible="true" isEnabled="true" isBold="true"/>	Representa a la etiqueta de texto que comúnmente acompaña a otros componentes gráficos.

Tabla 2.2. Componentes CUI de UsiXML.

2.5.Análisis sintáctico

El analizador sintáctico o parser [1], es la segunda fase de un compilador, utiliza los primeros componentes de los tokens producidos por el analizador léxico (fase 1 del compilador) para generar una representación intermedia en forma de árbol que describa la estructura gramatical del flujo de tokens.

Una representación típica es el árbol sintáctico, en el cual cada nodo interior representa una operación y los hijos del nodo representan los argumentos de la operación, la construcción de éste árbol garantiza una traducción correcta.

2.6.HTML

Lenguaje de marcado para hipertexto (Hyper Text Markup Language, HTML) [16]. Basado en Standard Generalized Markup Language (SGML), un sistema mucho más completo y complicado de procesamiento de documentos que indica cómo organizar y etiquetar un documento. HTML define e interpreta las etiquetas de acuerdo a SGL.

Las páginas HTML se pueden diseñar usando texto con distintos tipos de fuente, colores, imágenes, tablas, etc. Su modo de empleo es muy sencillo: se basa en el uso de etiquetas que indican que elementos contiene cada página, el formato que hay que aplicar a cada uno de ellos y como se tienen que distribuir por la página.

2.7. PHP

PHP (Hipertext Procesor, inicialmente llamado Personal Home Page) [7]. Es un lenguaje interpretado del lado del servidor que se caracteriza por su potencia, robustez y modularidad, es directamente embebido en código HTML y ejecutado por el servidor

web a través de un intérprete antes de transferir al cliente que lo ha solicitado, devolviendo un resultado en HTML puro.

PHP es un lenguaje fácil de aprender; especialmente para programadores familiarizados con C, Perl o Java, debido a la similitud de sintaxis. Además, es multiplataforma.

2.8. JavaScript

JavaScript es un lenguaje interpretado, basado en objetos (no es un lenguaje orientado a objetos "puro") y multiplataforma [10], inventado por NETSCAPE COMUNICATIONS CORPORATION. El primer nombre oficial de este lenguaje fue LiveScript en septiembre de 1995, poco después fue rebautizado JavaScript en un comunicado en conjunto con SUN MICROSYSTEMS el 4 de diciembre de 1995.

El núcleo de JavaScript (Core JavaScript) contiene una serie de objetos, como Array, Date, Math, Number y String, y un conjunto de elementos del lenguaje como operadores, estructuras de control y sentencias. JavaScript puede ser implementado de las siguientes maneras:

- JavaScript en el cliente (Client-side JavaScript) amplía el lenguaje añadiendo objetos que permiten controlar un navegador y su DOM.
- JavaScript en el servidor (Server-side JavaScript) amplía el lenguaje añadiendo objetos que son útiles cuando JavaScript se ejecuta en un servidor (lectura de ficheros, acceso a bases de datos, etc.).

Mediante la característica de JavaScript's LiveConnect, se pueden intercomunicar el código JavaScript con Java. Desde JavaScript, se pueden instanciar objetos Java y

acceder a sus propiedades y métodos públicos. A su vez, desde Java se puede acceder a los objetos de JavaScript, a sus propiedades y a sus métodos.

2.9. JQuery

JQuery es un framework de Javascript [17], es decir, es una biblioteca multiplataforma que sirve como base para la programación avanzada de aplicaciones, aporta una serie de funciones o códigos para realizar tareas habituales. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

2.10. AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML) [12], es un conjunto de tecnologías existentes tales como: HTML o XHTML, CSS, JavaScript, DOM, XML, XSLT y el objeto XMLHttpRequest. Trabajando de forma cooperativa, para lograr el desarrollo de aplicaciones web capaces de actualizarse continuamente sin tener que volver a cargar la página, es decir, los datos se solicitan al servidor, se reciben y se cargan en segundo plano sin interferir con la visualización del comportamiento de la página, aunque también se pueden configurar las peticiones de manera síncrona pudiendo detener la interactividad de la página hasta obtener una respuesta del servidor.

Ajax es multiplataforma dado que está basado en estándares como JavaScript y DOM (Document Object Model).

2.11. Google analytics

Google Analytics es una herramienta web multiplataforma y gratuita para monitorizar el flujo hacia una aplicación WEB, Android, iOS [15]. Así mismo, ofrece múltiples reportes de gran importancia para tomar decisiones del rumbo de la aplicación, por ejemplo, gracias a las estadísticas se podrían determinar los idiomas a los cuales traducir la aplicación.

Los datos procesados por Google Analytics, se actualizan constante y automáticamente, cada intervalo de tiempo, ofreciendo al administrador una vista en tiempo real del funcionamiento de la aplicación en monitoreo.

A continuación se presentan algunos datos relevantes que ofrece Google Analytics.

2.11.1. Usuarios

Total de usuarios que han utilizado la aplicación en un rango de tiempo (días).

2.11.2. Sesiones

Total de veces que han accedido a la aplicación en un rango de tiempo (días).

2.11.3. Porcentaje de rebote

Cantidad de usuarios que entran a la aplicación y salen sin hacer uso de la misma en un rango de tiempo (días).

2.11.4. Duración de la sesión

Estimación de tiempo que permanece cada usuario haciendo uso de la aplicación.

2.11.5. En este momento

Número de usuarios activos ahora mismo.

2.11.6. Páginas vistas

Gráfica representante del total de páginas visitadas en intervalos de tiempo; minuto y/o segundo.

2.11.7. Ubicaciones

Google Analytics presenta un mapa mundial en el cual pinta el o los países donde están utilizando la aplicación.

2.12. Aplicaciones similares

Existen otras aplicaciones similares a este proyecto, la mayoría son consideradas "secreto industrial", algunas otras son privadas y pocas son gratuitas. A continuación se citan algunos ejemplos:

2.12.1. PROKUS

PROKUS, PROgram system zur Kommunikations ergonomischen UterSchung rechnerunterstützte verfahren. (Sistema de comunicación de procedimientos ergonómicos asistidos por computadora). Software desarrollado por el laboratorio del Instituto de Ingeniera Humana e Industrial de la Universidad de Karlsruhe (Alemania) que mide la usabilidad de un sistema basándose en la ergonomía como criterio de calidad. Se basa en

el estándar ISO 9241-10, que especifica los principios de dialogo de las terminales visuales en términos ergonómicos.

2.12.2. SIRIUS

Sistema de evaluación de la usabilidad web orientado al usuario y basado en la determinación de tareas críticas (SIRIUS) [24]. Es un sistema de evaluación de la usabilidad web que parte de la evaluación heurística que ofrece:

- Aplicación en cualquier tipo de sitio web.
- Aplicación durante todo el ciclo de vida del sitio.
- Valor porcentual del nivel de usabilidad del sitio evaluado.

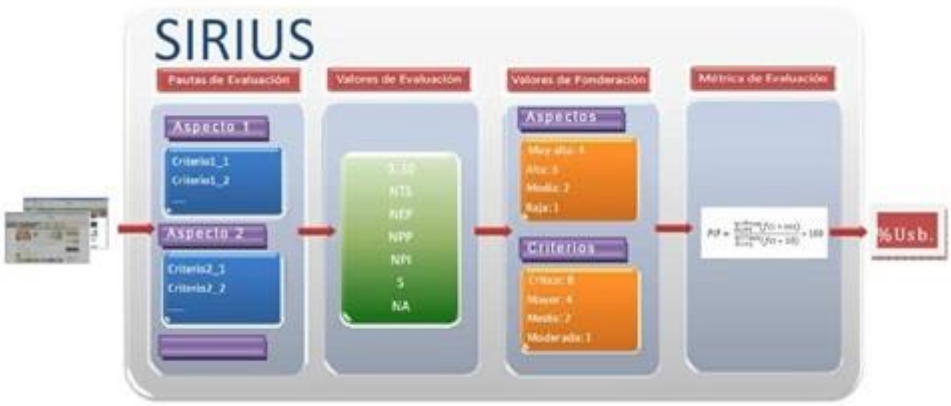


Figura 2.2. Sistema de evaluación SIRIUS [23].

A continuación se presentan las heurísticas a evaluar con este sistema:

Aspectos generales: Elementos relacionados con los objetivos del sitio, el look & feel, coherencia y nivel actualización de contenidos.

Identidad e Información: Elementos relacionados con la identidad del sitio, información proporcionada y la autoría de los contenidos.

Estructura y navegación: Elementos relacionados con la idoneidad de la arquitectura de la información y la navegación.

Rotulado: Elementos relacionados con la significación, corrección y familiaridad del rotulado de los contenidos.

Layout de la página: Elementos relacionados con la distribución y el aspecto de los elementos de navegación e información en la interfaz.

Facilidad en la interacción: Elementos relacionados con la adecuación y calidad de los contenidos textuales, iconos y controles de la interfaz.

Control y retroalimentación: Elementos relacionados con la libertad del usuario en la navegación y la información proporcionada al mismo en el proceso de interacción con el sitio.

Elementos multimedia: Elementos relacionados con el grado de adecuación de los contenidos multimedia del sitio web.

Búsqueda: Elementos relacionados con el buscador implementado en el sitio web.

Ayuda: Elementos relacionados con la ayuda ofrecida al usuario durante la navegación por el sitio.

2.12.3. WebSAT

Web Statics Analyzer Tool (WebSAT) [26]. Es una herramienta desarrollada por el Instituto Nacional de Estándares y Tecnología (NIST). Este sistema mediante pautas de usabilidad, detecta errores en el HTML de páginas web. Realiza inspecciones utilizando su propio conjunto de reglas de usabilidad o las IEEE Std 2001-1999.

WebSAT espera como entrada una URL. El resultado para una sola página web es casi inmediato (dependiendo del tamaño de la página), si se trata de un sito completo, el tiempo está en función de la cantidad de páginas que componen el sitio, por lo cual, deberá proporcionar un correo electrónico al cual se enviaran los resultados.

A continuación se presentan seis categorías en las cuales WebSAT organiza las pautas de usabilidad:

Accesibilidad: Asegura que la página hace un uso adecuado de etiquetas para usuarios con discapacidad visual.

Formulario: Comprueba la presencia de los botones Enviar y Restablecer formulario. Implementar un botón Restablecer podría ser imprudente porque los usuarios podrían borrar su entrada.

Desempeño: Analiza el tamaño y la codificación de los gráficos en relación con las velocidades de descarga de la página.

Mantenibilidad: Comprueba las etiquetas y la información de codificación que facilitaría el acceso a esta página desde otro servidor.

Navegación: Solo hace una comprobación básica de cómo se codifican los enlaces y no proporciona ningún comentario sobre la efectividad del etiquetado y la navegación del sitio.

Legibilidad: Proporciona una evaluación sobre la densidad de enlaces versus contenido.

2.12.4. Metricspot

Metricspot [20] es una herramienta de análisis web gratuita aunque también contiene secciones premium que requieren de algún pago, sin embargo, no afectan a su uso gratuito. Además, tenemos la posibilidad de hacer el seguimiento a múltiples páginas web.

La sección gratuita nos ofrece un informe detallado de todos los puntos que influyen en el posicionamiento de cualquier sitio web. El informe se compone de seis categorías Autoridad SEO, SEO básico, Contenido, Usabilidad, Aspectos técnicos, Redes sociales. A fin de esta tesis, revisaremos los aspectos evaluados dentro de la categoría Usabilidad.

Usabilidad; Recalca las principales cosas que benefician o perjudican al sitio web, por ejemplo:

URL: El nombre del dominio debería tener menos de 15 caracteres.

Favicon: Para que Google lo reconozca correctamente, el icono tiene que estar en el directorio de la raíz en formato .ico y debe llamarse “favicon.ico”.

Página de error 404: El manejo de una página de error 404 personalizada permite una mejor navegación en caso de que el usuario llegue a una página inexistente.

CSS para impresión: Estas hojas de estilo aseguran que la versión impresa de una página es legible y evitan que un usuario tenga que imprimir imágenes de gran tamaño.

Formulario de conversación: Los formularios de conversación o contacto, deberían de ser lo más sencillos posibles y que no pidan demasiados datos al usuario.

Idioma: Es necesario declarar el idioma del contenido en el HTML de cada página. Además, para webs en múltiples idiomas, también es importante especificar el código de idioma en la URL y utilizar HREFLANG en el sitemap.xml para indicar las versiones alternativas de una misma página en múltiples idiomas.

Tiempo de carga: La velocidad de carga de una web es considerado por Google para el ranking de webs. Además, una carga lenta conlleva que Googlebot tarde más en rastrear un sitio y por supuesto también afecta a la experiencia del usuario. Las web que tardan más tiempo en cargar tienden a tener un porcentaje de rebote más alto.

Optimización móvil: Valida que el sitio web sea responsivo para dispositivos móviles, así mismo, es considerado por Google para su ranking de búsqueda.

Es importante destacar que las herramientas automáticas no sustituyen la evaluación manual, sino que, después de haber validado el código a través de la herramienta automática es necesario completarlo con una evaluación manual.

Capítulo 3

Arquitectura del Sistema

En el capítulo anterior se han introducido las bases, de manera teórica, que serán utilizadas para desarrollar una propuesta de solución al problema planteado en esta tesis. El objetivo del presente capítulo es presentar la forma en que se ha puesto en práctica. Concretamente y en esencia, el problema al cual se le ha hecho frente es al de validar que la especificación de una interfaz de usuario a nivel concreto (CUI) cumpla las reglas de usabilidad, guías ergonómicas, estructuras (Smith y Mosier 1984) o varias recomendaciones (WCAG 2.0, AccessiWeb), codificadas en un formato formal.

El punto de partida, la interfaz de usuario a nivel concreto en UsiXML, se asumirá que está facilitada por la herramienta GrafiXML (GrafiXML, 2007), a partir de ella se ha desarrollado un marco de trabajo soportado por una herramienta, Usability Adviser, la cual generará reportes con diversas características para desarrollador de interfaz, certificador de interfaz y alertas. Por lo tanto, en este capítulo se analizarán las particularidades e ideas de los códigos finales.

A continuación en la Figura 3.1, se ilustran los pasos correspondientes a la metodología a seguir.

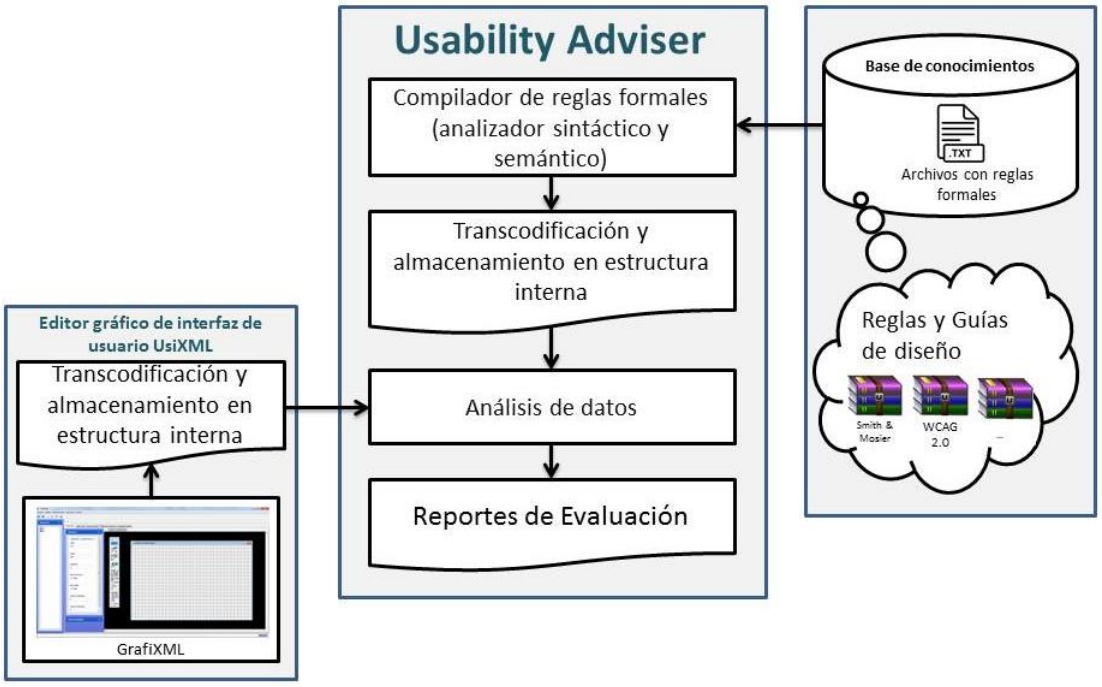


Figura 3.1. Proceso global para evaluación automática.

Las reglas se comprueban con un analizador sintáctico para determinar errores en la codificación de las reglas. Una vez que se completa el análisis sintáctico, las reglas formales se considerarán cumplidas y listas para ser utilizadas en la evaluación de la interfaz de usuario. La evaluación de la interfaz de usuario se realiza sobre la interpretación directa de un archivo que contiene la descripción de la interfaz de usuario.

Los resultados obtenidos en este capítulo, finalmente cristalizaran en un sistema web, Usability Adviser.

3.1. Estructura de reglas de usabilidad

Se toma como punto de referencia la estructura formal para reglas de usabilidad propuesta en articulo Automated UI Evaluation based on a Cognitive Architecture and UsiXML [4], ver Figura 3.2.

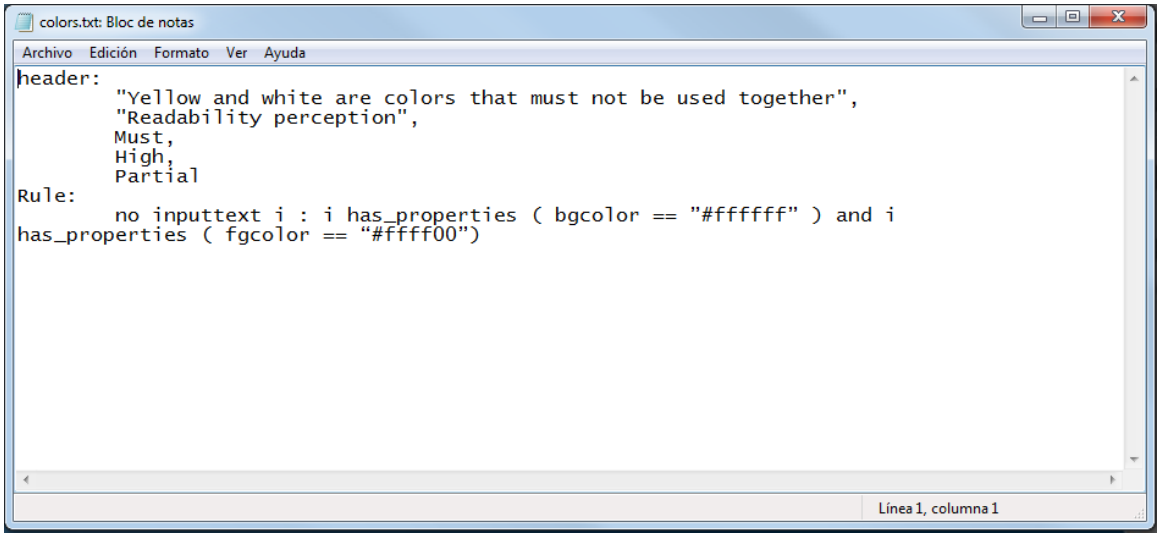


Figura 3.2. Estructura formal para reglas de usabilidad, propuesta en [4].

Para efectos prácticos y de legibilidad, se realizaron algunas adaptaciones por ejemplo, al declarar tanto el header como la rule se sustituye ":" por "$" evitando confusión al declarar variables, para indicar fin de línea se sustituye "," por ";" evitando confusión al declarar parámetros (20,80). Sin embargo, se mantiene al margen de la estructura en cuestión.

Cada regla de usabilidad (guideline) se compone de dos partes 1) Header y 2) Rule, a continuación se describe cada una de ellas:

3.1.1. Cabecera (Header)

La cabecera deberá ser denotada con la palabra reservada “Header” seguida de símbolo “$”. Cada línea del contenido deberá finalizar con “;” exceptuando la última de estas.

Header$ Contiene información general de la guideline como son:

Example; Descripción de la regla en lenguaje natural.

Ergonomic Criteria; Grupo que reúne una característica en común de las guidelines, tales como: Visibilidad, Densidad de información, consistencia, etc.

Priority Level; Nivel de importancia de cada directriz. La lista de MoSCoW a menudo se utiliza para indicar la importancia; MUST: Cuando varias fuentes mencionan la regla; SHOULD: Cuando algunas fuentes mencionan la regla; COULD: Pocas fuentes mencionan la regla; WOULD: Una sola fuente menciona la regla pero la fuente es fuerte.

Certainty; Evidencia que denota la fuente de la guideline y su validez. LOW: Existe al menos una referencia; MEDIUM: Existe un estudio empírico que demuestra la guideline; HIGH: Dos o más estudios arrojan los mismos resultados.

Degree of Coverage; Partial ó Total;

3.1.2. Regla (Rule)

La Regla deberá ser denotada con la palabra reservada “Rule” seguida de símbolo “$”. Cada línea del contenido deberá finalizar con “;” exceptuando la última de estas.

Rule$ Contiene información puntual para la validación de la guideline separada en dos partes, Subheader y Rule.

Estructura general:

Subheader + “ : ” + Rule + “ ; ”

Subheader; Define el subconjuto de elementos a evaluar.

Cuantificador (C); Define el total de ítems afectados por la guideline.

all: Referencia a la regla universal, es decir, “Para todo”.

one or more: Referencia a la regla existencial, es decir, “Existe”.

at least n: Referencia a reglas como “Al menos n widgets”.

at most: Referencia a reglas como “A lo sumo n widgets”.

exactly n: Referencia a reglas como “Exactamente n widgets”.

not n: Referencia a reglas como “No debería tener n widgets”.

no: Referencia a reglas como “No widgets”.

Número cuantificador (Nc); Se emplea para especificar cuantos widgets deben cumplir con la regla.

Widget (W); Nombre del elemento a evaluar, combobox, inputtext, etc.

Identificador (I); Letra representativa del conjunto de widgets.

Link; Existen tres tipos diferentes de enlaces para asociar restricciones:

and: Simbolizado por la palabra clave “y” en el lenguaje formal.

or: Simbolizado por la palabra clave “o” en el lenguaje formal.

Implicación (->): Una implicación se compone de una premisa y un consecuente. Esta premisa y el resultado están compuestos de al menos una restricción que puede vincularse opcionalmente mediante y/o. El principio de participación puede

reflejarse en la siguiente formula: “Si la premisa es verdadera, entonces el consecuente debe ser verdadero”

Rule; Definición formal de la(s) regla(s) que debe cumplir cada ítem.

Nuevo identificador: Se especifica el identificador sobre el cual se aplicara la guideline.

Restricción: Se define la acción a ejercer sobre el conjunto de items:

has properties: Esta restricción comprobará si el atributo de un widget tiene el valor esperado al compararlo con una constante o atributo de otro widget.

contains: Esta restricción permite expresar el hecho de que un widget debe contener otro widget en una cantidad específica.

is correctly balanced: Esta restricción garantiza que la distribución de widgets esté correctamente equilibrada.

has attribute value in file: Esta restricción comprobará si el atributo de un widget está contenido en un archivo que se especifica como parámetro. El contenido de este archivo se carga en la memoria al compilar reglas.

has external constraint: Esta restricción llamará a un método Java definido por el usuario. Los parámetros requeridos son el nombre de la clase definida por el usuario, seguido de un punto, luego el nombre del método (dentro de la clase del usuario) para llamarla con un paréntesis abierto y cierre.

Propiedad: Se especifica la propiedad a evaluar.

Comparador: Se retoman los operadores de comparación utilizados en leguajes de programación:

== Igualdad

!= Desigualdad

=== Estrictamente iguales

!== Estrictamente desiguales

> Mayor que

>= Mayor o igual que

< Menor que

<= Menor o igual que

Valor esperado: Resultado que se espera obtener.

3.1.3. Caracteres reservados

A continuación se listan los caracteres de uso específico en el sistema.

$ Fin de cabecera

; Fin de línea

, Separación de sub headers

: Separación entre subheader y rule

() Se utilizan para denotar parámetros o argumentos

- Separador para denotar rango

Así mismo, cada guideline deberá ser guardada en un archivo de texto plano (.txt)

Ejemplo

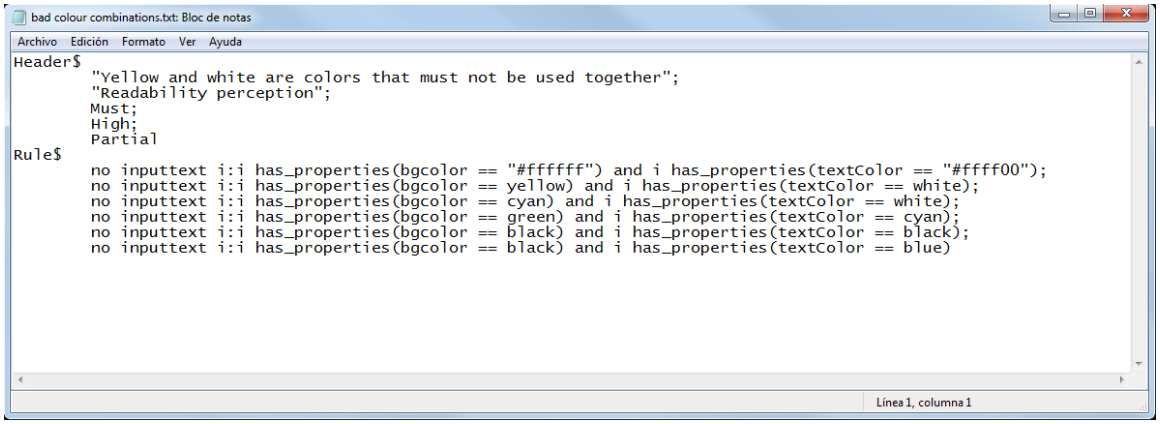


Figura 3.3. Ejemplo de Guideline válida.

3.2. Analizador sintáctico de reglas de usabilidad

Este módulo se encarga de revisar que la guideline (.txt) contenga la información mínima requerida, así mismo, valida la estructura de la regla.

Cualquier guideline que no cumpla con la estructura, será desechada. En la figura 3.4, se presenta el diagrama de flujo correspondiente.

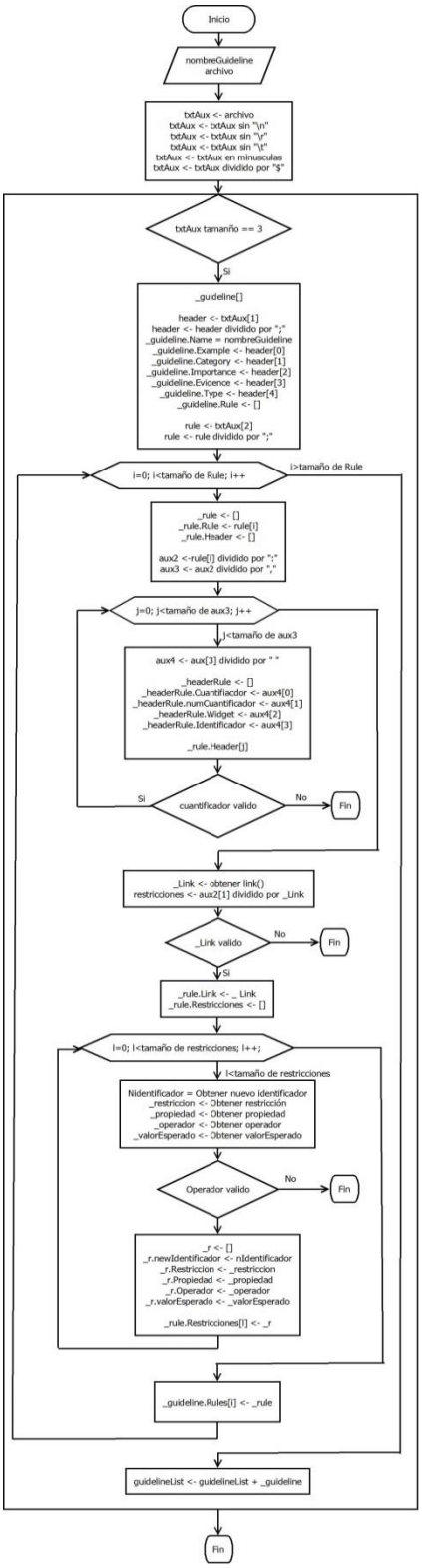


Figura 3.4. Diagrama de flujo-Analizador sintáctico de reglas de usabilidad.

A continuación se presenta el código relacionado a este módulo.

```
function compilador(nombreGuideline){
  // Creamos la url hacia el archivo
  var TXT_URL = "../Formal Rules/"+nombreGuideline;
  // Leemos el archivo
  $.ajax({
   url : TXT_URL,
   dataType: "text",
   success : function (data) {
    // Eliminamos los saltos de lineas
    var txtAux = data.replace(/\n/gi, " ");
    txtAux = txtAux.replace(/\r/gi, " ");
    // Eliminamos las tabulaciones
    txtAux = txtAux.replace(/\t/gi, " ");
    // Convertimos a minusculas
    txtAux = txtAux.toLowerCase();
    // Separamos la rule
    txtAux = txtAux.split("$");
    // Validamos la longitud del array
    if(txtAux.length == 3){
     // Inicializamos la estructura
     var _guideline = {};
     // Separamos el header y la rule
     var header = txtAux[1].replace(/rule/gi, " ");
     // Separamos las restricciones
     header = header.split(";");
     // Limpiamos
     for (var i = 0; i < header.length; i++) {
      // Eliminamos los espacios en blanco y en los extremos
      header[i] = header[i].trim();
     }
     _guideline.Name = nombreGuideline;
     _guideline.Example = header[0];
     _guideline.Category = header[1];
     _guideline.Importance = header[2];
     _guideline.Evidence = header[3];
     _guideline.Type = header[4];
     _guideline.Rules = [];
     var rule = txtAux[2];
     // Separamos las restricciones
     rule = rule.split(";");
     for (var i = 0; i < rule.length; i++) {
      var _rule = [];
      t+="<br>->"+rule[i];
      _rule.Rule = rule[i].trim();
      _rule.Header = [];
      // dividimos el headerRule y la restriccionRule
      var aux2 = rule[i].split(":");
      // Obtenemos todos los headerRule
      var aux3 = aux2[0].trim();
      aux3 = aux3.split(",");
```

```
for(var j=0; j<aux3.length; j++){
 // Subdividimos el headerRule
 var aux4 = aux3[j].trim();
 aux4 = aux4.split(" ");
 // Recuperamos
 var cuantificador = "";
 var Ncuantificador = 0;
 var widget = aux4[aux4.length-2];
 var identificador = aux4[aux4.length-1];
 for(k=0; k<aux4.length-2; k++){
  if(aux4[k]>0){
   Ncuantificador = aux4[k];
  }
  else{
   if(aux4[k] != ""){
    cuantificador += " " + aux4[k];
   }
  }
 }
 // Eliminamos los espacios en blanco y en los extremos
 cuantificador = cuantificador.trim();
 var _headerRule = [];
 _headerRule.Cuantificador = cuantificador;
 _headerRule.numCuantificador = Ncuantificador;
 _headerRule.Widget = widget;
 _headerRule.Identificador = identificador;
 _rule.Header[j] = _headerRule;
 // Validamos que sea correcto el cuantificador
 switch(cuantificador){
  case "at least":
  case "at most":
  case "exactly":
  case "not":
   // Validamos tener un numero correcto
   if(Ncuantificador < 1){
    alert("Error en el Ncuantificador:")
    return false;
   }
  case "all":
  case "one or more":
  case "no":
   break;
  default:
   alert("Error en el cuantificador:")
   return false;
   break;
 }
}
// Separamos por links
var _Link ="and";
var restricciones = aux2[1].split(" and ");
if(restricciones.length == 1){
 _Link ="or";
 restricciones = aux2[1].split(" or ");
}
```

```
if(restricciones.length == 1){
 _Link ="->";
 restricciones = aux2[1].split(" -> ");
}
if(restricciones.length == 1){
 _Link ="SIN LINKS";
 // Veridicamos que no existan más de 2 "(,)"
 var parentesisA = 0;
 var parentesisC = 0;
 var oper = 0;
 for(var a=0; a<aux2[1].length; a++){
  switch(aux2[1][a]){
   case "(":
    parentesisA++;
    if(parentesisA==2){
      return false;
    }
    break;
   case ")":
    parentesisC++;
    if(parentesisC==2){
     return false;
    }
    break;
   case "=":
   case "!":
   case "<":
   case ">":
    oper++;
    if(oper==4){
     return false;
    }
    break;
  }
 }
}
_rule.Link = _Link;
_rule.Restricciones = [];
for(var l=0; l<restricciones.length; l++){
 var Nidentificador = "";
 var rest = "";
 var m =0;
 // Ignoramos espacios en blanco
 while(restricciones[l][m] == " "){
  m++;
 }
 // Separamos el nuevo identificador
 while(restricciones[l][m] != " "){
  Nidentificador += restricciones[l][m];
  m++;
 }
 for(m; m<restricciones[l].length; m++){
  // Eliminamos espacios en blanco
  if(restricciones[l][m] != " "){
   rest += restricciones[l][m];
```

```
        }
       }
       //
       var n=0;
       var _restriccion="";
       var _propiedad="";
       var _operador="";
       var _valorEsperado="";
       // Obtenemos la restriccion
       while(rest[n] != "(" && n<rest.length){
        _restriccion += rest[n];
        n++;
       }
       n++;
       if(_restriccion == "contains"){
        // Obtenemos el valor esperado
        while(rest[n] != ")" && n<rest.length){
         _valorEsperado += rest[n];
         n++;
        }
       }
       else{
        // Obtenemos
        while(rest[n]!="=" && rest[n]!="!" && rest[n]!="<"  &&
rest[n]!=">" && rest[n]!="-" && n<rest.length){
         _propiedad += rest[n];
         n++;
        }
        // Obntenemos el operador
        while((rest[n]=="=" || rest[n]=="!" || rest[n]=="<"  ||
rest[n]==">" || rest[n]=="-") && n<rest.length){
         _operador += rest[n];
         n++;
        }
        // Obtenemos el valor esperado
        while(rest[n] != ")" && n<rest.length){
         _valorEsperado += rest[n];
         n++;
        }
        // Validamos el operador de comparacion
        switch(_operador){
         case "==":  //Igualdad
         case "!=":  //Desigualdad
         case "===": //Estrictamente iguales
         case "!==": //Estrictamente desiguales
         case ">":   //Mayor que
         case ">=":  //Mayor o igual que
         case "<":   //Menor que
         case "<=":  //Menor o igual que
         case "-":  //Separador
          break;
         default:
          return false;
          break;
        }
```

```
        }
        // Validar Nidentificador
        var _r = [];
        _r.newIdentificador = Nidentificador;
        _r.Restriccion = _restriccion;
        _r.Propiedad = _propiedad;
        _r.Operador = _operador;
        _r.valorEsperado = _valorEsperado;
        _rule.Restricciones[l] = _r;
       }
       _guideline.Rules[i] = _rule;
      }
     }
     return false;
    }
   });
}
```

3.3. Estructura interna de reglas de usabilidad

Creamos una estructura interna con las guidelines que sí cumplieron la sintaxis descrita anteriormente.

```
guidelineList.push(_guideline);
```

3.4. Estructura de archivos UsiXML (.usi)

Existen múltiples herramientas para maquetar interfaces gráficas, algunas de ellas ofrecen la opción de exportar el maquetado, generando un archivo de descripción de interfaz con formato UsiXML (.usi), tal es el caso de GrafiXML, IdealXML, EgiuXML, etc.

Para fines de esta tesis, analizaremos la estructura del archivo de descripción de interfaz generado con la herramienta GráfiXML versión 1.2.0 [13]. A continuación se presentan las principales secciones que lo conforman.

3.4.1. prólogo

Aunque no es obligatorio, los documentos UsiXML pueden empezar con una línea que describen la versión de XML y otros atributos descritos a continuación.

```
<?xml version="…" encoding="…" standalone="…"?>
```

Version: Indica la versión del estándar XML a la cual se ajusta el documento. Actualmente, el único valor es "1.0".

Encoding: Indica el juego de caracteres del documento.

Standalone: Tiene dos posibles valores;

- **no**: Es necesario acceder al DTD (Definición de Tipo de Documento) del documento para obtener ciertos datos sobre este (por ejemplo, los valores por defecto de atributos)
- **yes**: El documento no depende del DTD.

El valor por defecto es "no".

3.4.2. uiModel

Indica el comienzo y el final de un documento de UsiXML. Ninguna etiqueta o contenido puede colocarse antes o después de la etiqueta <uiModel>. En el interior de este, se define la cabecera, la interfaz y el contexto. Así mismo, se definen algunos atributos tales como:

Id: Cadena de identificación de la interfaz.

Name: Nombre de la interfaz.

creationDate: Fecha y hora exacta del sistema en cual se crea la interfaz.

schemaVersion: Versión de UsiXML que se utiliza.

```
<uiModel xmlns="http://www.usixml.org"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.usixml.org/
      http://www.usixml.org/spec/UsiXML-ui_model.xsd"
   id="Adviser_13" name="Adviser"

    creationDate="2017-12-13T15:32:50.102-06:00" schemaVersion="1.6.4">
       .
       .
       .
</uiModel>
```

3.4.3. head

La cabecera de un documento UsiXML está delimitado por las etiquetas <head> y </head>. Contiene información sobre el propio documento UsiXML, por ejemplo el nombre del autor, fecha de la última modificación, etc. A continuación se describen los principales atributos contenidos en el header:

modifDate: Fecha y hora del sistema en cual se modifica por última vez.

authorName: Presenta el nombre del usuario del sistema que está desarrollando la interfaz.

comment: Se implementa para denotar comentarios y generalmente especifica la herramienta utilizada para la generación del modelo de interfaz de usuario.

```
<head>
   <version modifDate="2017-12-13T15:32:50.102-06:00">1</version>
   <authorName>Cesar M Rodriguez M</authorName>
   <comment>Generated by GrafiXML 1.2.0 build id :
      200612141408</comment>
   <comment>WARNING : AUI Model save is a work in progress. Use it at
      your own risk</comment>
   .
   .
   .
</head>
```

3.4.4. cuiModel

Delimita el cuerpo del documento UsiXML. El cuerpo contiene todas las interfaces que se presentan al usuario, estas a su vez contienen cientos de elementos con cuales tendrá que interactuar, ver Tabla 2.3.

```
<cuiModel id="Adviser-cui_21" name="Adviser-cui">
   <window id="window_component_1" name="window_component_1"
     width="400" height="350">
       .
       .
       .
   </window>
</cuiModel>
```

3.4.5. contextModel

Define todos los contextos de uso para las interfaces, cada contexto es definido con:

id: Cadena de identificación del contexto.

name: Nombre del contexto.

Así mismo, el contexto establece:

userStereotype: Estereotipo de usuario que interactuará con la interfaz.

platform: Plataforma sobre la cual se implementara la interfaz.

```
<contextModel id="estructura-contextModel_20" name="estructura-
contextModel">
   <context id="estructura-context-es_ES_20" name="estructura-context-
es_ES">
      <userStereotype id="estructura-stes_ES_20" language="es_ES"
stereotypeName="estructura-stes_ES"/>
      <platform id="estructura-platform_20" name="estructura-
platform"/>
      <environment id="estructura-env_20" name="estructura-env"/>
   </context>
   .
   .
   .
```

```
    .
</contextModel>
```

3.5. Compilador de UsiXML

Este módulo se encarga de procesar el contenido de cada archivo UsiXML y a su vez, crea una estructura interna con los widgets contenidos en el archivo (.usi), en la Figura 3.5 se ilustra el diagrama de flujo correspondiente.

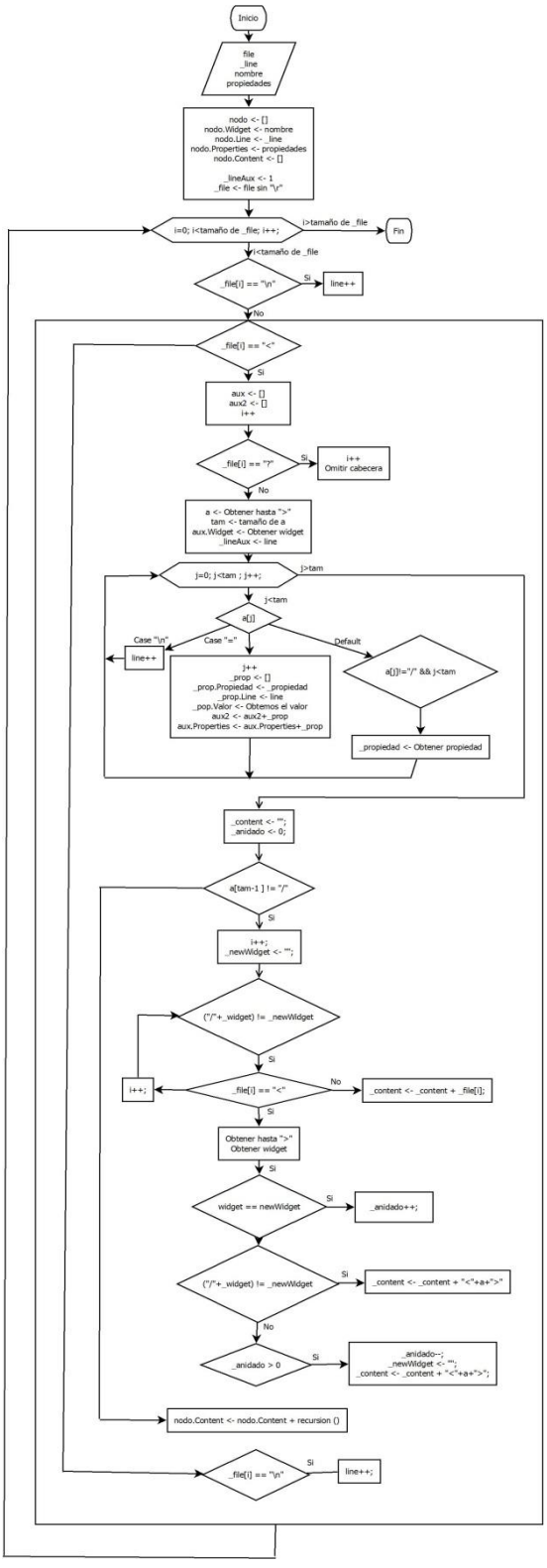


Figura 3.5. Diagrama de flujo – Compilador de archivos UsiXML.

El código resultante se presenta a continuación.

```
function compilaUsi(file, _line, nom, prop){
 var nodo = [];
 nodo.Widget = nom;
 nodo.Line = _line;
 nodo.Properties = prop;
 nodo.Content = [];
 var _lineAux = 1;
 var _file = file.replace(/\r/gi, " ");
 //
 for(var i=0; i<_file.length; i++){
  // Validamos si es un salto de line
  if(_file[i] == "\n"){
   line ++;
  }
  else{
   if(_file[i] == "<"){
    var aux = [];
    aux.Widget = "";
    aux.Line = "";
    aux.Properties = [];
    aux.Content = [];
    var aux2 = [];
    i++;
    // Omitir cabecera
    if(_file[i] == "?"){
     i++;
     while(_file[i]!="?" && _file[i+1]!=">"){
      if(_file[i] == "\n"){
       line ++;
      }
      i++;
     }
    }
    else{
     var a = "";
     while(_file[i] != ">"){
      a+=_file[i];
      i++;
     }
     var _tam = a.length;
     var j=0;
     var _widget="";
     // Obtener el widget
     while(a[j]!=" " && j<_tam){
      _widget += a[j];
      j++;
     }
     aux.Widget = _widget;
     _lineAux = line;
     var _del = "";
     var _propiedad = "";
```

```
var _valor = "";
for(j; j<=_tam; j++){
 switch(a[j]){
  case "\n":
   line ++;
   break;
  case "=":
   // Obtenemos el delimitador inicial
   j++;
   _del = a[j];
   j++;
   // Obtenemos el valor de la propiedad
   while(a[j]!=_del && j<_tam){
    _valor += a[j];
    j++;
   }
   //
   var _prop = [];
   _prop.Line = line;
   _prop.Propiedad = _propiedad.trim();
   _prop.Valor = _valor;
   aux2.push(_prop);
   aux.Properties.push(_prop);
   //
   _propiedad = "";
   _valor = "";
   break;
  default:
   if(a[j]!="/" && j<_tam){
    _propiedad += a[j];
   }
   break;
 }
}
var _content = "";
var _anidado = 0;
// Validamos
if(a[a.length-1] != "/"){
 i++;
 var _newWidget = "";
 while(("/"+_widget)!=_newWidget){
  if(_file[i] == "<"){
   i++;
   var a = "";
   while(_file[i] != ">"){
    a+=_file[i];
    i++;
   }
   var _tam = a.length;
   var j=0;
   _newWidget="";
   // Obtener el widget
   while(a[j]!=" " && j<_tam){
    _newWidget += a[j];
    j++;
```

```
        }
        // Validamos i existe un Widget del mismo tipo
        if(_widget == _newWidget){
         _anidado++;
        }
        // Validamos si es el cierre del widget
        if(("/"+_widget)!=_newWidget){
         _content += "<"+a+">";
        }
        else{
         // Validamos si ya se cerraron todos los widgets anidados
         if(_anidado > 0){
          _anidado--;
          _newWidget = "";
          _content += "<"+a+">";
         }
        }
       }
       else{
        _content += _file[i];
       }
       i++;
      }
     }
     nodo.Content.push(compilaUsi(_content,_lineAux,_widget,aux2));
    }
   }
   if(_file[i] == "\n"){
    line ++;
   }
  }
 }
 return nodo;
}
```

3.6. Estructura de archivos de Texto Plano (.txt)

Los archivos de texto plano son utilizados para establecer un valor por defecto a ciertas propiedades de widgets específicos, estos archivos son utilizados por las guidelines definidas con la restricción “has attribute value in file”.

Las declaraciones están agrupadas por bloques, en los que el conjunto de declaraciones se encuentra entre llaves, “{” indica inicio de bloque y “}” indica fin de bloque, precedido de un punto “.” y el nombre del widget afectado. Cada declaración

contenida en un bloque declarativo debe estar separada por un punto y coma “;”, la última declaración de un bloque no necesita llevarlo aunque se considera buena práctica añadirlo. Cada declaración se compone de dos partes 1) propiedad y 2) valor(es), conjuntadas mediante dos puntos “:”

- **Propiedad**: Identificador que denota la propiedad a inicializar.
- **Valor**: A cada propiedad se le asigna un valor, que indica cómo queremos inicializar esta propiedad, por ejemplo (tipo de fuente, ancho, color, texto predeterminado, etc.).

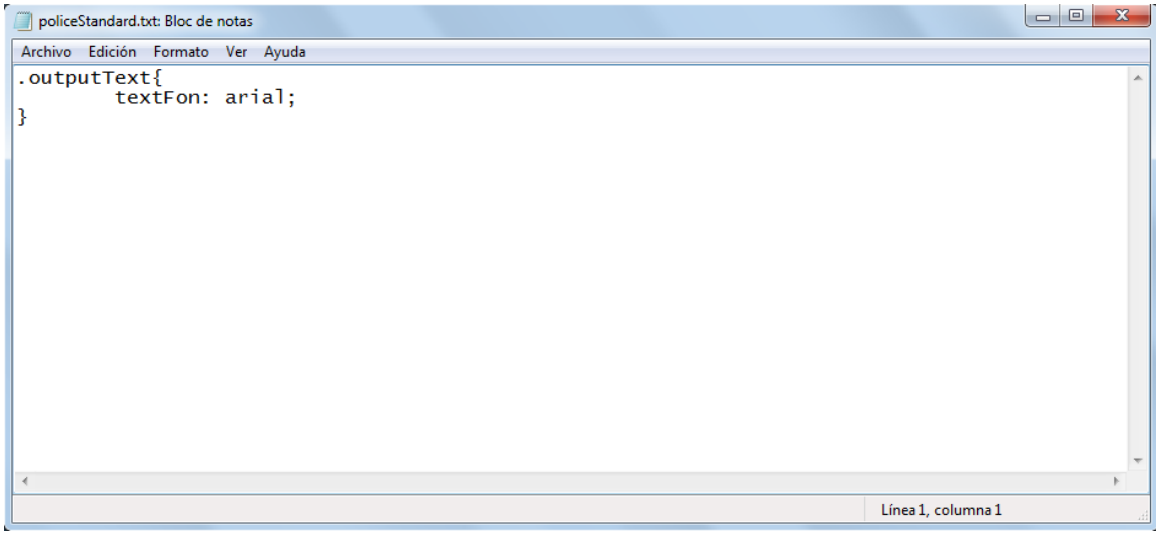


Figura 3.6. Estructura de archivos de texto plano (.txt) válida.

3.7. Análisis de datos (UsiXML vs Guidelines)

Este módulo se encarga de recorrer la lista de archivos UsiXML proporcionados por el usuario, por cada archivo, se recorre la lista de guideline a evaluar, así mismo, se

inicializa el “Interprete de reglas de usabilidad” (descrito anteriormente) para validar que el contenido del archivo UsiXML cumpla con dicha guideline.

El proceso se ha desarrollado en 13 submódulos (runGuideline, apply Guideline, subconjunto, cuentaErrores, validaCuantificador, validaRestriccion, validaLink, addFila_reportDesing, addFila_reportDevelop, addFila_reportWarnings, validaValor, getPosicion, validaCuantificador2), con el fin de hacerlo más legible, mejorable y reutilizable. En la Figura 3.7 se presenta la relación entre submódulos.

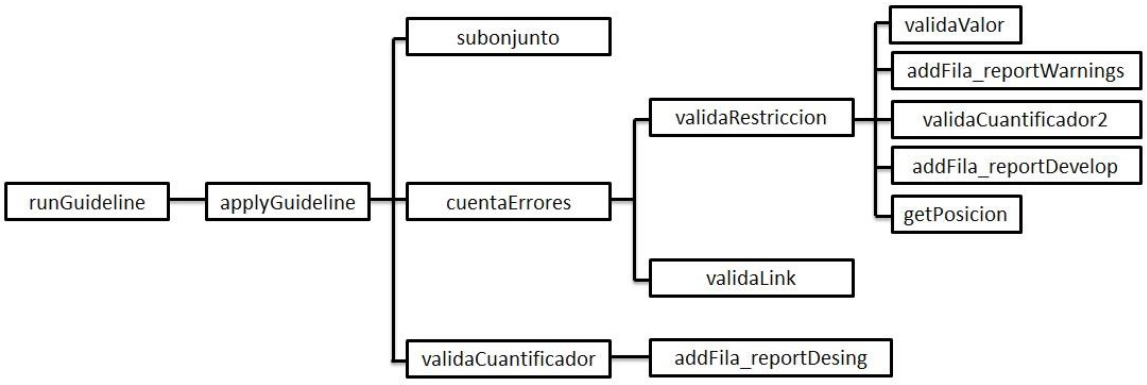


Figura 3.7. Relación entre submódulos para análisis de datos (UsiXML vs Guideline).

A continuación se describen cada uno de los submódulos, así mismo, se presenta el código resultante de la implementación.

3.7.1. RunGuideline

Este submódulo es el encargado de recorrer la lista de archivos proporcionados por el usuario, por cada archivo a evaluar, se recorre toda la lista de Guidelines a aplicar.

Así mismo, se inicializa el “Interprete de reglas de usabilidad” mandando como parámetros tanto el archivo que se evalúa como la regla que deberá cumplir.

```
function runGuidelines(){
  // Recorremos todos los archivos
  while(fileListOrg.length > 0){
   var _file = fileListOrg.pop();
   obj=document.getElementById("sel1");
   // Validamos que no sea el elemento vacio "-"
   if (obj.options[0].value != "-"){
    // Recorremos la lista de guidelines a evaluar
    for (i=0; opt=obj.options[i]; i++){
     // Recorremos la estructura interna en busqueda de la Guideline
     for(var j=0; j<guidelineList.length; j++){
      if(opt.text == guidelineList[j].Name){
       // Recorremos todas las rules
       for(var k=0; k<guidelineList[j].Rules.length; k++){
        // Verificar que los archivos cumplan la guideline
        guidelineCurrent = guidelineList[j];
        applyGuideline(_file,guidelineList[j].Rules[k]);
       }
      }
     }
    }
   }
  }
}
```

3.7.2. ApplyGuideline

En este submódulo, se crea el subconjunto de ítems sobre los cuales se aplicara la regla de usabilidad, se analizan y contabilizan los ítems que no cumplan con dicha regla. Posteriormente se analiza el cuantificador de la regla pasándole como parámetros el total de ítems que no cumplen la regla de usabilidad en evaluación.

```
function applyGuideline(file, guideline){
 switch(guideline.Link){
  case "or":
  case "and":
  case "->":
   // Validamos tener 2 restricciones
   if(guideline.Restricciones.length == 2){
    // Obtenemos el identificador de la restricción
    var id0 = guideline.Restricciones[0].newIdentificador;
```

```
    var id1 = guideline.Restricciones[1].newIdentificador;
    var head0;
    var head1;
    // Obtenemos el header para dicho id
    var i=0;
    while(i<guideline.Header.length){
     // Comparamos el idHeader con el idRestricción
     if(id0 == guideline.Header[i].Identificador){
      head0 = guideline.Header[i];
     }
     if(id1 == guideline.Header[i].Identificador){
      head1 = guideline.Header[i];
     }
     i++;
    }
    // Validamos que ambas restricciones sean sobre el mismo widget
    if(id0 == id1){
     // Obtenemos subconjunto
     var widgetList = [];
     widgetList = subconjunto(file, head0.Widget, widgetList);
     var _contaError = cuentaErrores(file.Widget, guideline,
head0.Cuantificador, widgetList);
     // Validar cuantificador
     validaCuantificador(file.Widget, head0.Cuantificador,
head0.numCuantificador, _contaError, guideline.Rule);
    }
   }
   else{
    alert("Número de restricciones no valido");
   }
   break;
  case "SIN LINKS":
   // Obtenemos el identificador de la restricción
   var id0 = guideline.Restricciones[0].newIdentificador;
   var head0;
   // Obtenemos el header para dicho id
   var i=0;
   while(i<guideline.Header.length){
    // Comparamos el idHeader con el idRestricción
    if(id0 == guideline.Header[i].Identificador){
     head0 = guideline.Header[i];
    }
    i++;
   }
   var widgetList = [];
   widgetList = subconjunto(file, head0.Widget, widgetList);
   var _contaError = cuentaErrores(file.Widget, guideline,
head0.Cuantificador, widgetList);
   // Validar cuantificador
   validaCuantificador(file.Widget, head0.Cuantificador,
head0.numCuantificador, _contaError, guideline.Rule);
   break;
  default:
   alert("Link no soportado por el sistema");
   break;
```

```
 }
}
```

3.7.3. subconjunto

Este subconjunto recibe como parámetros el contenido del archivo y el nombre del widget sobre el cual se aplicará la regla de usabilidad. Recorre el contenido del archivo UsiXML, crea una pila con los ítems afectados por la regla de usabilidad. La pila resultante de este proceso se retorna en una variable hasta donde fue llamado este submódulo.

```
function subconjunto(file, widget, widgetList, width, height){
 switch(file.Widget){
  case "window":
   for(var x=0; x<file.Properties.length; x++){
    switch(file.Properties[x].Propiedad){
     case "width":
      width = file.Properties[x].Valor;
      break;
     case "height":
      height = file.Properties[x].Valor;
      break;
    }
   }
   break;
  case "gridBagBox":
   var _width;
   var _height;
   for(var x=0; x<file.Properties.length; x++){
    switch(file.Properties[x].Propiedad){
     case "gridWidth":
      _width = file.Properties[x].Valor;
      width = width/_width;
      break;
     case "gridHeight":
      _height = file.Properties[x].Valor;
      height = height/_height;
      break;
    }
   }
   break;
 }
 // Validar widget
 if(file.Widget.toLowerCase() == widget.toLowerCase()){
  // Añadir campo con tamaño de pixeles
  file.PixelW = width;
  file.PixelH = height;
```

```
  widgetList.push(file);
 }
 for(var i=0; i<file.Content.length; i++){
  // Content
  subconjunto(file.Content[i],widget,widgetList,width,height);
 }
 return widgetList;
}
```

3.7.4. cuentaErrores

En este submódulo se realiza un llamado al submódulo validaRestriccion, se almacena el valor retornado por este último, se repite esto para cada una de las restricciones. Una vez validadas todas las restricciones, se invoca al submódulo validaLink pasándole como argumento tanto el valor de cada restricción como el link.

En caso de existir error alguno, se incrementa el contador de errores, finalmente, el contador de errores se retorna hasta el origen de la invocación.

```
function cuentaErrores(file, guideline, cuantificador, widgetList){
 var _contaError = 0;
 var res1 = false;
 var res2 = false;
 // Creamos una copia del wideList
 var widgetListAux = [];
 for(var x=0; x<widgetList.length; x++){
  widgetListAux.push(widgetList[x]);
 }
 // Recorremos todo el subconjunto
 while(widgetList.length > 0){
  var item = widgetList.pop();
  // Validamos restricciones
  res1 = validaRestriccion(file, item, guideline.Restricciones[0],
guideline, widgetListAux);
  if(guideline.Restricciones.length > 1){
   res2 = validaRestriccion(file, item, guideline.Restricciones[1],
guideline, widgetListAux);
  }
  // Validar link
  _contaError +=  validaLink(file, guideline, res1, guideline.Link,
res2, cuantificador);
 }
 return _contaError;
}
```

3.7.5. validaCuantificador

Este submódulo recibe como parámetros: cuantificador, número ideal de ítems afectados por la guideline, número real de ítems afectados por la guideline. Con base en el cuantificador, se compara el número ideal de ítems contra el número real de ítems afectados por la guideline para determinar si se cumple o no se cumple la guideline. Finalmente, el valor de la comparación se retorna hasta el origen del llamado.

```
function validaCuantificador(file,cuantificador,numCuantificador,
errores,rule){
// alert(file+" -> "+cuantificador+" -> "+errores+" -> "+rule);
 var _error = false;
 switch(cuantificador){
  case "all":
  case "no":
   if(errores > 0){
    _error = true;
   }
   break;
  case "one or more":
   if(errores >= 1){
    _error = true;
   }
   break;
  case "at least":
   if(errores < numCuantificador){
    _error = true;
   }
   break;
  case "at most":
   if(errores > numCuantificador){
    _error = true;
   }
   break;
  case "exactly":
   if(errores == numCuantificador){
    _error = true;
   }
   break;
  case "not":
   if(errores != numCuantificador){
    _error = true;
   }
   break;
 }
 if(_error == true){
  // Añadimos el informe al reportDesing
```

```
  addFila_reportErrorsDesing(file,guidelineCurrent.Name,
guidelineCurrent.Example,guidelineCurrent.Category,guidelineCurrent.Imp
ortance,guidelineCurrent.Evidence,guidelineCurrent.Type,rule);
 }
}
```

3.6.7. validaRestriccion

Este es el submódulo principal de Usability Adviser, aquí se evalúan las restricciones: has properties, contains, is correctly balanced, has attribute value in file y has external constraint. Al momento de redactar esta tesis, se encuentran implementados los primeros cuatro (has properties, contains, is correctly balanced y has attribute value in file), sin embargo, el desarrollo del sistema sigue en curso y constante evolución.

```
function validaRestriccion(file, content, rule, guideline, widgetList){
 var valida = false;
 // Se asume la línea de error
 line_Error = content.Line;
 switch(rule.Restriccion){
  case "has_properties":
   var _propiedad = rule.Propiedad.toLowerCase();
   // Buscamos la propiedad
   var i = 0;
   while((_propiedad != content.Properties[i].Propiedad.toLowerCase())
&& (i<content.Properties.length-1)){
    i++;
   };
   if(_propiedad==content.Properties[i].Propiedad.toLowerCase()){
    var valReal = content.Properties[i].Valor;
    var valEspe = rule.valorEsperado;
    // Convertimos a minusculas
    valReal = valReal.toLowerCase();
    valEspe = valEspe.toLowerCase();
    // Eliminamos el delimitador de valor
    valReal = valReal.replace(/['"]+/g, '');
    valEspe = valEspe.replace(/['"]+/g, '');
    // Validamos el valor
    valida = validaValor(rule.Operador,valReal,valEspe);
    line_Error = content.Properties[i].Line;
    if(valida == true){
     break;
    }
   }
   else{
    addFila_reportWarnings(file,rule.Propiedad,content.Line);
   }
   break;
```

```
  case "contains":
   // Recuperamos enunciado para el valor esperado
   var head0;
   // Obtenemos el header para dicho id
   var i=0;
   //alert("valorEsperado: "+rule.valorEsperado);
   while(i<guideline.Header.length){
    // Comparamos el idHeader con el idRestriccion
    if(rule.valorEsperado == guideline.Header[i].Identificador){
     head0 = guideline.Header[i];
    }
    i++;
   }
   var _contaError = 0;
   // Recorremos todo el contenido
   for(var i=0; i<content.Content.length; i++){
     if(content.Content[i].Widget.toLowerCase() ==
head0.Widget.toLowerCase()){
     _contaError++;
    }
   }
   var _r = head0.Cuantificador +" " +
      head0.numCuantificador+" " +
      head0.Widget+" " +
      head0.Identificador;
   // Validar cuantificador
   var _error = validaCuantificador2(file, head0.Cuantificador,
head0.numCuantificador, _contaError, _r);
   if(_error == true){
    // Añadimos el informe al reportDevelop
addFila_reportErrorsDevelop(file, guidelineCurrent.Example,
guidelineCurrent.Importance, guideline.Rule, _r);
   }
   valida = !_error;
   break;
```

Para validar si los ítems de la interfaz están correctamente balanceados, es decir, la distancia existente entre cada ítem respecto de otro se encuentra dentro del rango permitido, deberemos conocer las coordenadas exactas de la ubicación del ítem por lo anterior, denotaremos los vértices del ítem a evaluar con las letras A, B, C y D mientras que los vértices del ítem para comparación serán denotados por las letras A2, B2, C2 D2 como se ilustran en la Figura 3.8.

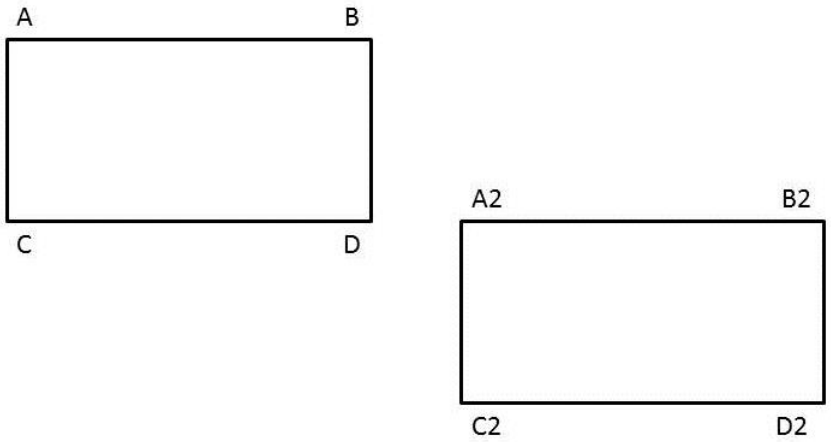


Figura 3.8. Distribución de vértices (is_correctly_balanced).

```
  case "is_correctly_balanced":
   var min = parseFloat(rule.Propiedad);
   var max = parseFloat(rule.valorEsperado);
   // Obtenemos la posicion del widget actual
   var aux = getPosicion(content);
   // Calculamos los vertices
   var pxA =
parseFloat(aux.gridX)*parseFloat(aux.weightX)*parseFloat(content.PixelW
);
   var pyA =
parseFloat(aux.gridY)*parseFloat(aux.weightY)*parseFloat(content.PixelH
);
   var pxB = parseFloat(pxA)+((parseFloat(aux.gridWidth)-
1)*parseFloat(content.PixelW));
   var pyB = pyA;
   var pxC = pxA;
   var pyC = parseFloat(pyA)+((parseFloat(aux.gridHeight)-
1)*parseFloat(content.PixelH));
   var pxD = pxB;
   var pyD = pyC;
   // Validamos tener elementos para comparar
   if(widgetList.length > 1){
    // Comparamos contra todos los elementos
    for(var x=0; x<widgetList.length; x++){
     // Obtenemos la posicion del widget a comparar
     var aux2 = getPosicion(widgetList[x]);
     // Calculamos los vertices
     var pxA2 =
parseFloat(aux2.gridX)*parseFloat(aux2.weightX)*parseFloat(widgetList[x
].PixelW);
     var pyA2 =
parseFloat(aux2.gridY)*parseFloat(aux2.weightY)*parseFloat(widgetList[x
].PixelH);
     var pxB2 = parseFloat(pxA2)+((parseFloat(aux2.gridWidth)-
1)*parseFloat(widgetList[x].PixelW));
     var pyB2 = pyA2;
```

```
     var pxC2 = pxA2;
     var pyC2 = parseFloat(pyA2)+((parseFloat(aux2.gridHeight)-
1)*parseFloat(widgetList[x].PixelH));
     var pxD2 = pxB2;
     var pyD2 = pyC2;
     // Validamos A-B Vs C2-D2
     if( ((pyA-max)<=pyC2) && (pyC2<=(pyA-min)) ){
      // El vector A-B > C2-D2
      if( ((pxA-max)<=pxC2) && (pxC2<=(pxB+max)) || ((pxA-max)<=pxD2)
&& (pxD2<=(pxB+max)) ){
       valida = true;
       break;
      }
      // El vector A-B < C2-D2
      if( (pxC2<(pxA-max)) && ((pxB+max)<=pxD2) ){
       valida = true;
       break;
      }
     }
     // Validamos B-D Vs A-C
     if( ((pxB+min)<=pxA2) && (pxA2<=(pxB+max)) ){
      // El vector B-D > A2-C2
      if( ((pyB-max)<=pyA2) && (pyA2<=(pyD+max)) || ((pyB-max)<=pyC2)
&& (pyC2<=(pyD+max)) ){
       valida = true;
       break;
      }
      // El vector B-D < A2-C2
      if( (pyA2<(pyB-max)) && ((pyD+max)<=pyC2) ){
       valida = true;
       break;
      }
     }
     // Validamos C-D Vs A2-B2
     if( ((pyC+min)<=pyA2) && (pyA2<=(pyC+max)) ){
      // El vector C-D > A2-B2
      if( ((pxC-max)<=pxA2) && (pxA2<=(pxD+max)) || ((pxC-max)<=pxB2)
&& (pxB2<=(pxD+max)) ){
       valida = true;
       break;
      }
      // El vector C-D < A2-B2
      if( (pxA2<(pxC-max)) && ((pxD+max)<=pxB2) ){
       valida = true;
       break;
      }
     }
     // Validamos A-C Vs B2-D2
     if( ((pxA-max)<=pxB2) && (pxB2<=(pxA-min)) ){
      // El vector A-C > B2-D2
      if( ((pyA-max)<=pyB2) && (pyB2<=(pyC+max)) || ((pyA-max)<=pyD2)
&& (pyD2<=(pyC+max)) ){
       valida = true;
       break;
      }
```

```
    // El vector B-D < A2-C2
    if( (pyB2<(pyA-max)) && ((pyC+max)<=pyD2) ){
     valida = true;
     break;
    }
   }
  }
 }
 else{
  // Sin elementos para comparar
  valida = true;
 }
 break;
case "has_attribute_value_in_file":
 // Obtenemos la propiedad a validar
 var _propiedad = rule.Propiedad.toLowerCase();
 // Obtenemos el nombre del archivo txt
 var _fileTXT = rule.valorEsperado;
 // Convertimos a minusculas
 _fileTXT = _fileTXT.toLowerCase();
 // Eliminamos el delimitador de valor
 _fileTXT = _fileTXT.replace(/['"]+/g, '');
 // Validamos tener almenos un archivo txt
 if(fileListTXT.length > 0){
  for(var i=0; i<fileListTXT.length; i++){
   // Recuperamos el nombre del archivo
   var _fl = fileListTXT[i].Name;
   // Convertimos a minusculas
   _fl = _fl.toLowerCase();
   // Eliminamos el delimitador de valor
   _fl = _fl.replace(/['"]+/g, '');
   // Validamos el archivo
   if(_fileTXT == _fl){
    // Recuperamos el contenido del TXT
    var _contentTXT = fileListTXT[i].Content;
    _contentTXT = _contentTXT.split("."+widgetList[0].Widget+"{");
    for(var j=1 ; j<_contentTXT.length; j++){
     var aux = _contentTXT[j].split("}");
     var aux2 = aux[0].toLowerCase();
     // Buscamnos que este definida la propiedad
     var aux2 = aux2.split(_propiedad+":");
     if(aux2.length == 2){
      valida = true;
      break;
     }
    }
    if(valida == false){
     addFila_reportWarnings(file,rule.Propiedad,_fileTXT);
    }
   }
  }
 }
 else{
  alert("Please, provide at least one file TXT");
 }
```

```
   break;
  default:
   alert("Restriccion: "+guideline.Restricciones[i].Restriccion+" no
soportada por el sistema");
   break;
 }
 return valida;
}
```

3.6.8. validaLink

En este submódulo, se evalúa el link con base a la siguiente tabla de valores.

A	*B*		*A*	*A and B*	*A or B*	*A -> B*
0	0		0	0	0	1
0	1		0	0	1	1
1	0		1	0	1	0
1	1		1	1	1	1

Tabla 3.1. Tabla de valores (Links).

```
function validaLink(file,guideline,_op1,_link,_op2, cuantificador){
 var _contaError = 0;
 var _error = "";
 switch(_link){
  case "and":
   // Validamos que NO se cumpla la Rule
   if(_op1==false || _op2==false){
    _contaError++;
   }
   // Asumimos que el error esta en _op2
   _error = guideline.Restricciones[1].Propiedad;
   break;
  case "or":
   // Validamos que NO se cumpla la Rule
   if(_op1==false && _op2==false){
    _contaError++;
   }
   // Asumimos que el error esta en _op2
```

```
   _error = guideline.Restricciones[1].Propiedad;
   break;
  case "->":
   // Validamos que NO se cumpla la Rule
   if(_op1==true && _op2==false){
    _contaError++;
   }
   // Asumimos que el error esta en _op2
   _error = guideline.Restricciones[1].Propiedad;
   break;
  case "SIN LINKS":
   // Validamos que NO se cumpla la Rule
   if(_op1==false){
    _contaError++;
   }
   break;
 }
 // Validamos si el error esta en _op1
 if(_op1==false){
  _error = guideline.Restricciones[0].Propiedad;
 }
 switch(cuantificador){
  case "no":
  case "at most":
   if(_contaError == 0){
    _contaError = 1;
    // Aqui se podria cambiar el campo de error
    _error = guideline.Restricciones[0].Propiedad;
   }
   else{
    _contaError = 0;
   }
   break;
 }
 // Validamos tener un texto de _error
 if(_error == ""){
  _error = "DOSN'T SAY";
 }
 // Validamos si existe error
 if(_contaError > 0){
  // Añadimos el informe al reportDevelop

addFila_reportErrorsDevelop(file,guidelineCurrent.Example,guidelineCurr
ent.Importance,guideline.Rule,_error);
 }
 return _contaError;
}
```

3.6.9. validaValor

Este submódulo se ha implementado específicamente para comparar valores, los operadores reconocido por Usability Adviser son: igualdad, desigualdad, estrictamente iguales, estrictamente desiguales, mayor que, mayor o igual que, menor que, menor o igual que.

```
function validaValor(_operador,_valReal,_valEspe){
 switch(_operador){
  case "==":  // Igualdad
   if(_valReal == _valEspe){
    return true;
   }
   break;
  case "!=":  //Desigualdad
   if(_valReal != _valEspe){
    return true;
   }
   break;
  case "===": //Estrictamente iguales
   if(_valReal === _valEspe){
    return true;
   }
   break;
  case "!==": //Estrictamente desiguales
   if(_valReal !== _valEspe){
    return true;
   }
   break;
  case ">":   //Mayor que
   if(_valReal > _valEspe){
    return true;
   }
   break;
  case ">=":  //Mayor o igual que
   if(_valReal >= _valEspe){
    return true;
   }
   break;
  case "<":   //Menor que
   if(_valReal < _valEspe){
    return true;
   }
   break;
  case "<=":  //Menor o igual que
   if(_valReal <= _valEspe){
    return true;
```

```
   }
   break;
  default:
   alert("Operador no reconocido por el sistema");
   break;
 }
 return false;
}
```

3.6.10. getPosicion

Recupera las coordenadas x,y del ítem recibido como parámetro.

```
function getPosicion(item){
 var aux = [];
 for(var x=0; x<item.Properties.length; x++){
  switch(item.Properties[x].Propiedad){
   case "gridx":
    aux.gridX = item.Properties[x].Valor;
    break;
   case "gridy":
    aux.gridY = item.Properties[x].Valor;
    break;
   case "gridwidth":
    aux.gridWidth = item.Properties[x].Valor;
    break;
   case "gridheight":
    aux.gridHeight = item.Properties[x].Valor;
    break;
   case "weightx":
    aux.weightX = item.Properties[x].Valor;
    break;
   case "weighty":
    aux.weightY = item.Properties[x].Valor;
    break;
  }
 }
 return aux;
}
```

3.7. Inclusión de Google Analytics

Una vez creada la cuenta en google analytics, se procede a incrustar el condigo de seguimiento en cada una de las páginas deseadas.

Código de seguimiento. A continuación se presenta el código de seguimiento proporcionado por google analytics.

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-120459522-1"></script>
<script>
	window.dataLayer = window.dataLayer || [];
	function gtag(){dataLayer.push(arguments);
	gtag('js', new Date());

	gtag('config', 'UA-120459522-1');
</script>
```

Capítulo 4

Sistema Automático de Evaluación.

A lo largo de este capítulo conoceremos la interfaz e interacción de la herramienta "Usability Adviser", desarrollada a lo largo de esta tesis. El sistema se encuentra disponible en www.usabilityadviser.com

4.1. Reglas de usabilidad

A continuación se presentan algunas de las reglas de usabilidad, las cuales, Usability Adviser verificará que se cumplan en el archivo de descripción de interfaz proporcionado por el usuario.

Regla ergonómica	*Criterios ergonómicos*	*Nivel de prioridad*	*Certeza*	*Grado de cobertura*	*Regla*
Amarillo y blanco son colores que no deben utilizarse en conjunto.	Percepción de facilidad de lectura	Debe	Alto	Parcial	no inputtext i : i has_properties (bgcolor == "#ffffff") and i has_properties (fgcolor == "#ffff00")
Evitar el uso de oraciones para etiquetar elementos del menú.	Percepción de contenido	Debería	Medio	Parcial	all item i : i has_properties (defaultContent !=" \w{1,}\s\w{1,}\s.* ")
Establecer un texto alternativo para cada imagen decorativa.	Percepción de contenido	Debería	Alto	Parcial	all imagecomponent i : i has_properties (ismandatory == false) -> i has_properties (tooltipdefaultcontent != "")

Establecer un texto alternativo para cada imagen importante.	Percepción de contenido	Debe	Alto	Parcial	all imagecomponent i : i has_properties (ismandatory == true) -> i has_properties (tooltipdefaultcontent != "")
Establecer un texto alternativo para cada vídeo.	Percepción de contenido	Debe	Alto	Parcial	all videoComponent v:v has_properties(toolTipDefaultContent != ‘’)
Limitar el número de imágenes decorativas a 4.	Sobrecarga	Debería	Alto	Total	at_most 4 imagecomponent i : i has_properties (ismandatory == false)
Limitar el número de iconos en una barra de herramientas para 20.	Sobrecarga	Debería	Alto	Total	no toolbar t, at_least 20 imagecomponent i : t contains (i)
Limitar el número de iconos en una barra de herramientas para 12.	Sobrecarga	Debería	Alto	Total	all toolbar t, at_most 19 imagecomponent i1, at_most 12 imagecomponent i2 : t contains(i1) -> t contains(i2)

Tabla 4.1. Reglas de usabilidad en Usability Adviser.

4.2. Cargar archivos

Usability Adviser trabaja sobre la descripción formal de interfaz gráfica 2D, 3D, auditiva, manipulable, etc. Existen múltiples lenguajes para la descripción de interfaces, sin embargo, Usability Adviser interpreta interfaces descritas en UsiXML (Detalla en el capítulo anterior), para validar que se cumplan ciertas reglas de usabilidad se requieren archivos de texto plano y/o archivos codificados en Java, por ende, Usability Adviser nos

ofrece dos formas para realizar la carga de archivos en formato UsiXML (.usi), texto plano (.txt) y Java (.java):

4.2.1. Chooser file

Permite seleccionar archivos mediante un cuadro de dialogo.

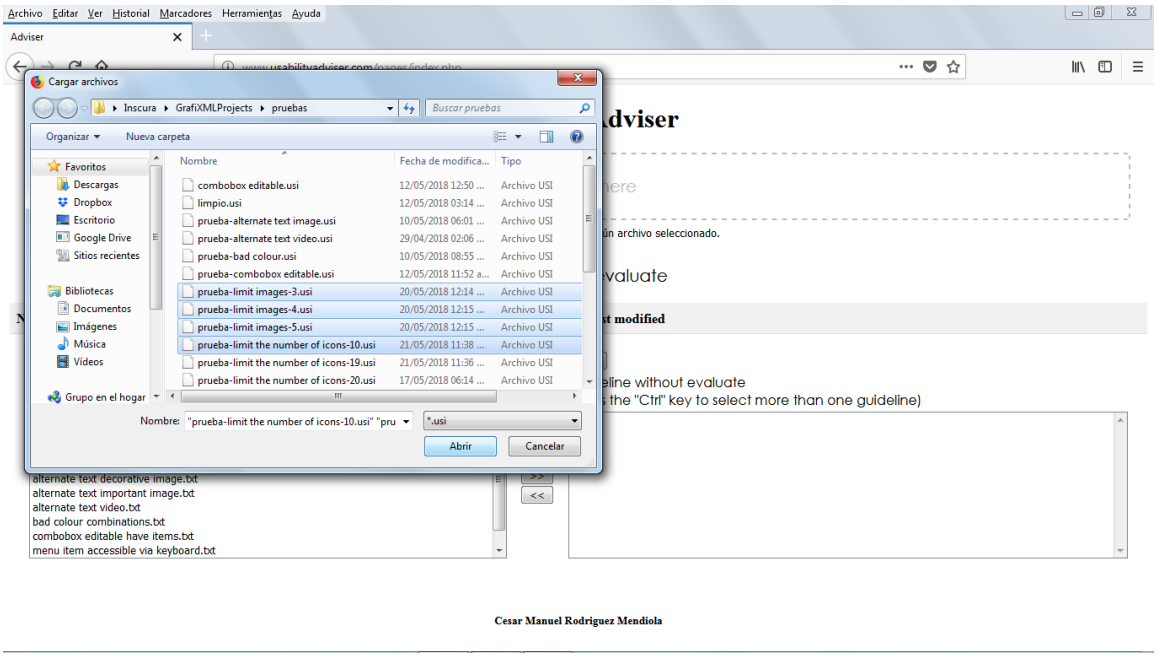


Figura 4.1. Carga de archivos (Chooser file).

- Dar clic en el botón “Choose Files”.
- Mediante el cuadro de dialogo, seleccionar el o los archivos (.usi, .txt, .java) necesario(s).
- Dar clic en el botón “Open”

4.2.2. Drag and drop

Permite cargar archivos mediante un arrastre hasta una zona específica.

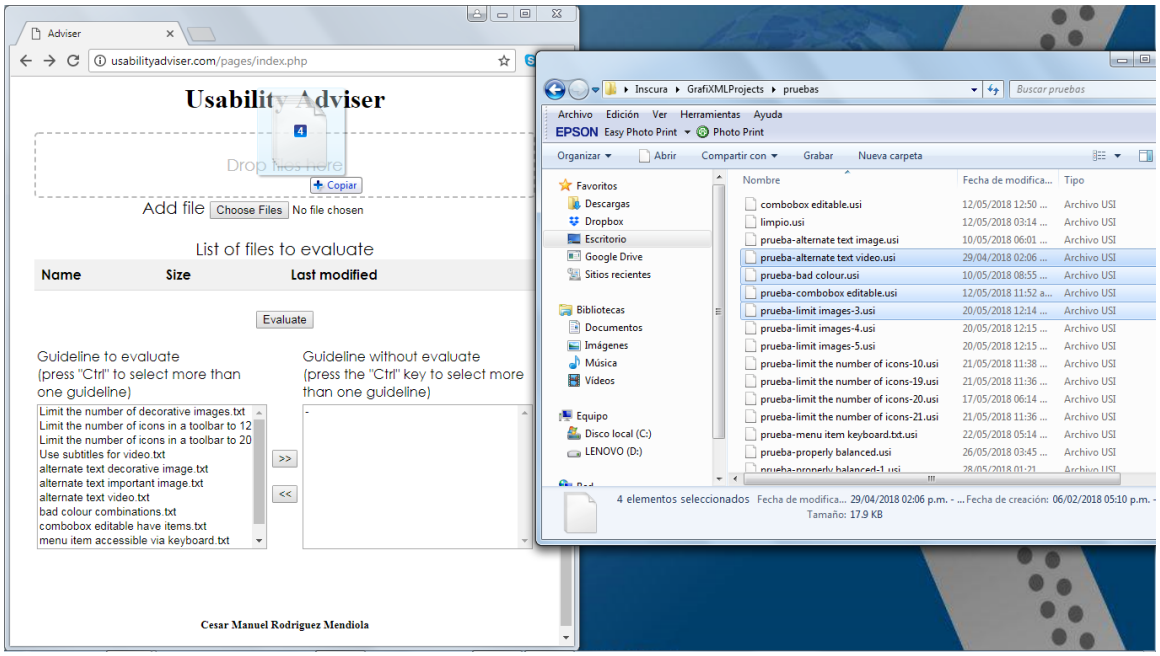


Figura 4.2. Carga de archivos (Drag and drop).

- Mediante un buscador de archivos, navegar hasta la carpeta deseada.
- Seleccionar los archivos (.usi, .txt, .java) a evaluar.
- Arrastrar hasta la zona punteada y con la etiqueta “Drop files here”

4.3. Tabla de archivos a evaluar

Los archivos (.usi, .txt, .java) que serán sometidos a evaluación, Usability Adviser los presenta en una tabla detallando Nombre, Tamaño en bytes, Fecha de última modificación y añade la opción para quitarlo de la tabla.

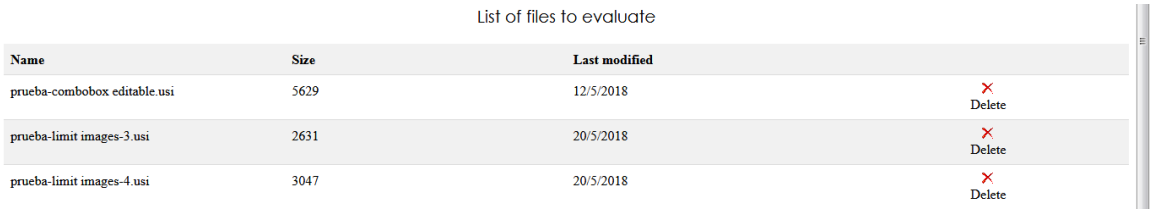


Figura 4.3. Lista de archivos a evaluar (.usi, .txt o .java).

- Dar clic en el botón “Delete” para excluir de la evaluación al archivo (.usi, .txt o .java).

4.4. Selección de reglas a verificar

Usability Adviser nos ofrece la posibilidad de seleccionar las guidelines que deseamos aplicar a nuestros archivos. Inicialmente, el sistema asume que se aplicarán todas las guidelines.

4.4.1. Guideline to evaluate

Se muestra el listado de guidelines a aplicar sobre los archivos UsiXML.

4.4.2. Guideline without evaluate

Se presenta el listado de guidelines excluidas para la evaluación de los archivos UsiXML.

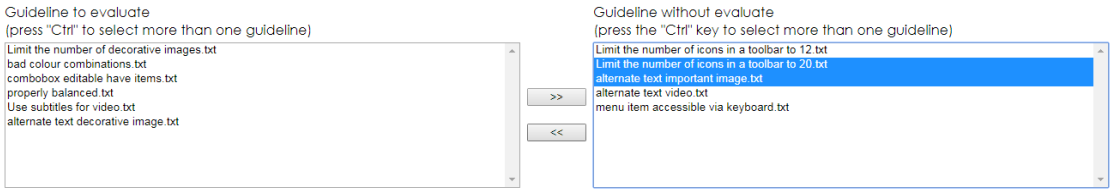


Figura 4.4. Listas de guidelines.

- Seleccionar la(s) guideline(s) a mover.
- Dar clic en el botón “>>” para excluir, de la evaluación, todas las guidelines seleccionadas.
- Dar clic en el botón “<<” para incluir, en la evaluación, todas las guidelines seleccionadas.

4.5. Reporte para certificador (Diseñador gráfico)

En la sección de resultados, dentro de la pestaña “Errores” encontramos el “Report errors for desiner” el cual presenta información relevante para el encargado de

certificar la interfaz de usuario, la información presentada es sobre interfaces que violan reglas de usabilidad, tal como: Nombre del archivo, Regla de usabilidad violada, Descripción de regla, Característica en común de las reglas de usabilidad, Nivel de prioridad, Nivel de evidencia, Nivel de cobertura y la propia Regla de usabilidad en lenguaje formal tal cual se ilustra en la Figura 4.5.

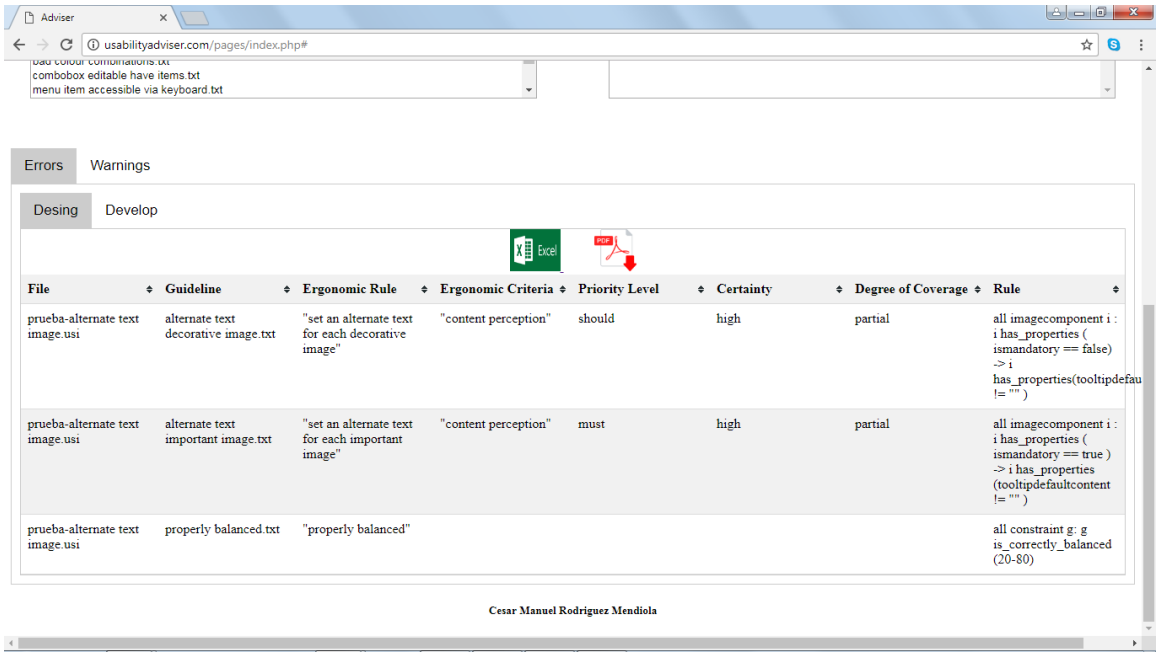
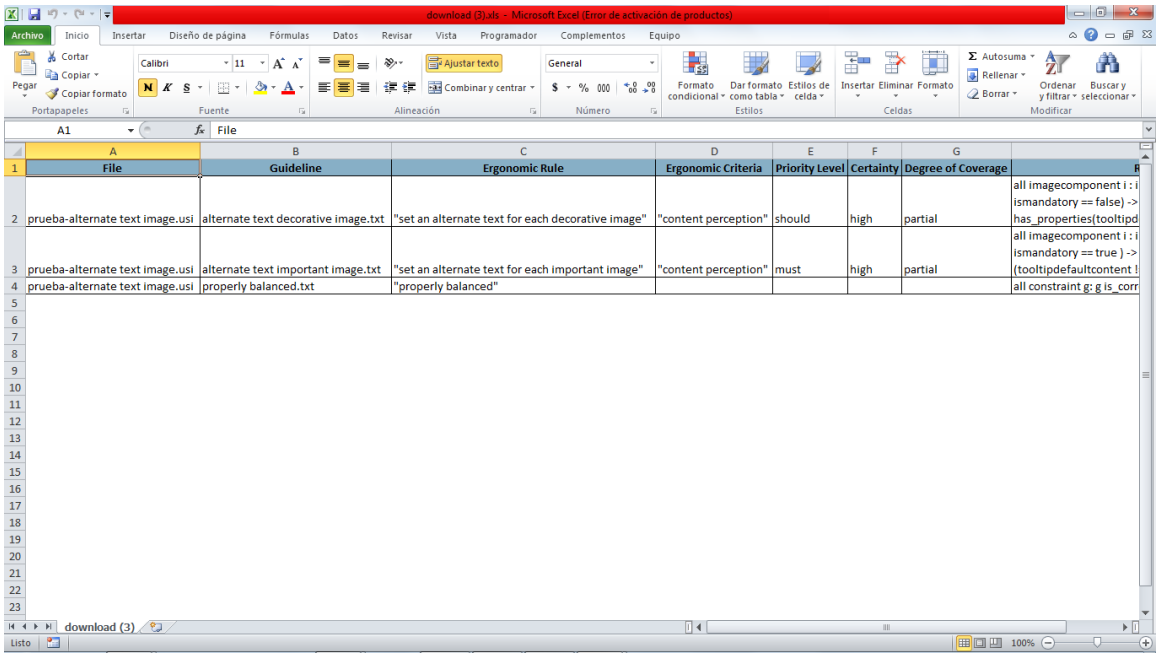


Figura 4.5. Reporte para certificador.

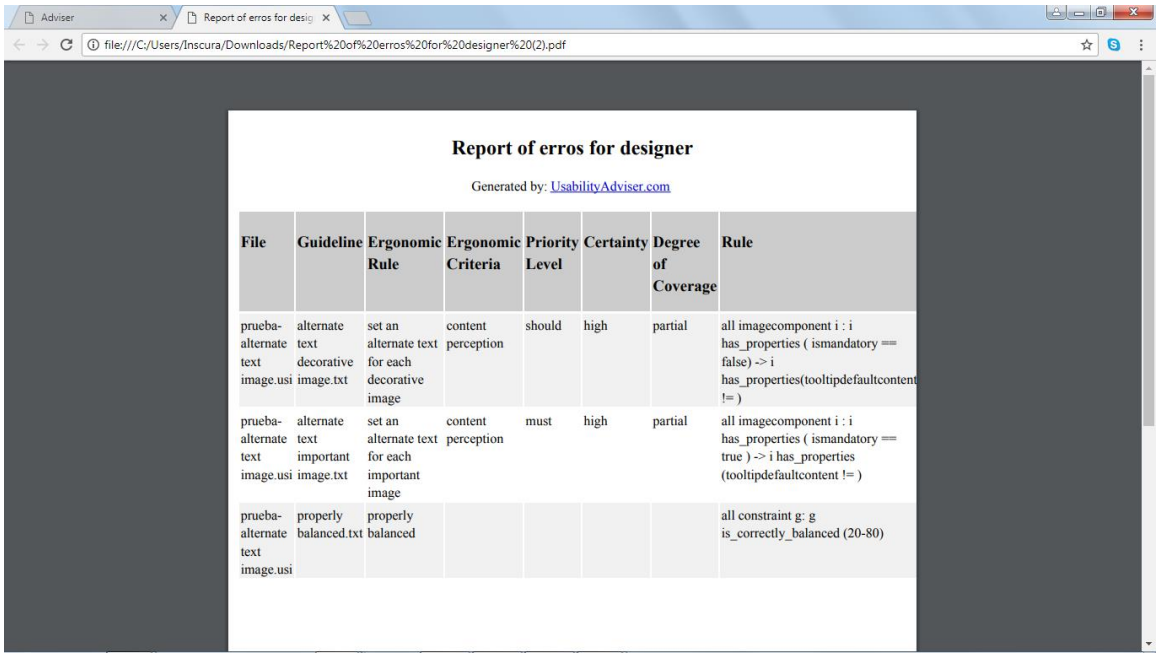
- Dar clic sobre el logo “Excel” para exportar a Excel.
- Dar clic sobre el logo “PDF” para exportar a PDF.

Mediante un clic, Usability Adviser nos permite exportar a Excel o exportar a PDF el reporte en cuestión. Las Figuras 4.6 y 4.7 ilustran el formato de exportación del reporte de errores vista certificador.



File	Guideline	Ergonomic Rule	Ergonomic Criteria	Priority Level	Certainty	Degree of Coverage	R
prueba-alternate text image.usi	alternate text decorative image.txt	"set an alternate text for each decorative image"	"content perception"	should	high	partial	all imagecomponent i : i ismandatory == false) -> has_properties(tooltipd
prueba-alternate text image.usi	alternate text important image.txt	"set an alternate text for each important image"	"content perception"	must	high	partial	all imagecomponent i : i ismandatory == true) -> (tooltipdefaultcontent !
prueba-alternate text image.usi	properly balanced.txt	"properly balanced"					all constraint g: g is_corr

Figura 4.6. Exportación a Excel del reporte para certificador.



Report of erros for designer

Generated by: UsabilityAdviser.com

File	Guideline	Ergonomic Rule	Ergonomic Criteria	Priority Level	Certainty	Degree of Coverage	Rule
prueba-alternate text image.usi	alternate text decorative image.txt	set an alternate text for each decorative image	content perception	should	high	partial	all imagecomponent i : i has_properties (ismandatory == false) -> i has_properties(tooltipdefaultcontent !=)
prueba-alternate text image.usi	alternate text important image.txt	set an alternate text for each important image	content perception	must	high	partial	all imagecomponent i : i has_properties (ismandatory == true) -> i has_properties (tooltipdefaultcontent !=)
prueba-alternate text image.usi	properly balanced.txt	properly balanced					all constraint g: g is_correctly_balanced (20-80)

Figura 4.7. Exportación a PDF del reporte para certificador.

4.6.Reporte para desarrollador

En la sección de resultados, dentro de la pestaña “Errores” encontramos el “Report errors for developer” el cual presenta información relevante para el desarrollador de la interfaz de usuario, la información presentada es sobre interfaces que violan reglas de usabilidad, tal como: Nombre del archivo UsiMXL, Línea de código UsiXML que viola la regla de usabilidad, Descripción de regla, Nivel de prioridad, Regla de usabilidad en lenguaje formal y la Propiedad que no cumple con la regla, tal cual se ilustra en la Figura 4.8.

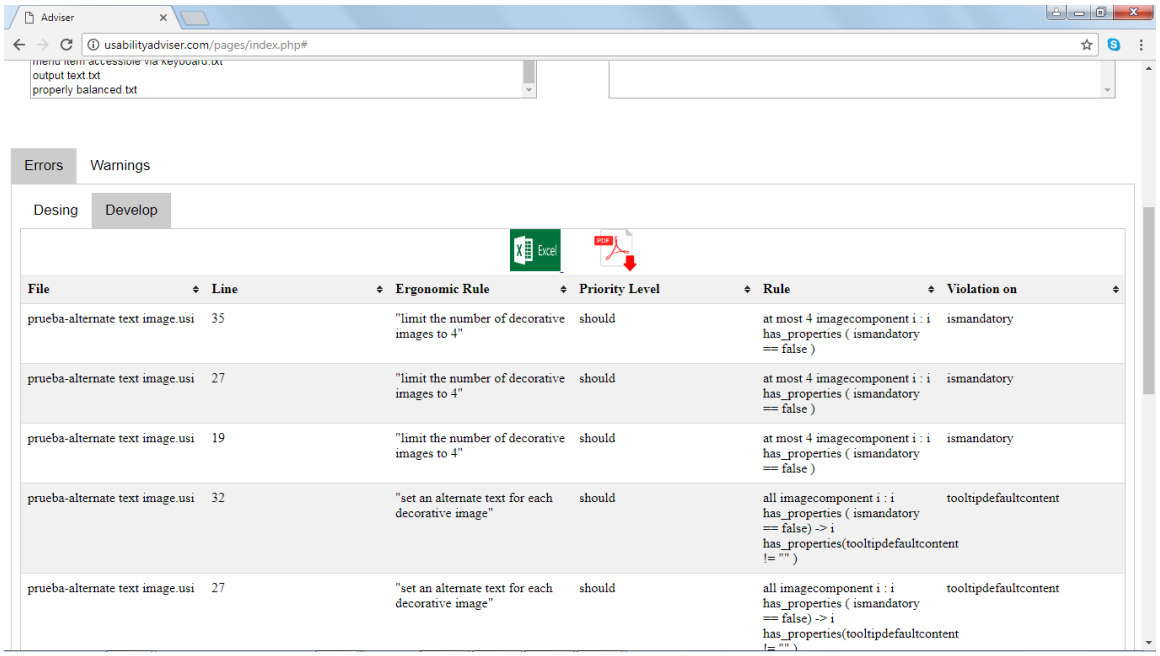
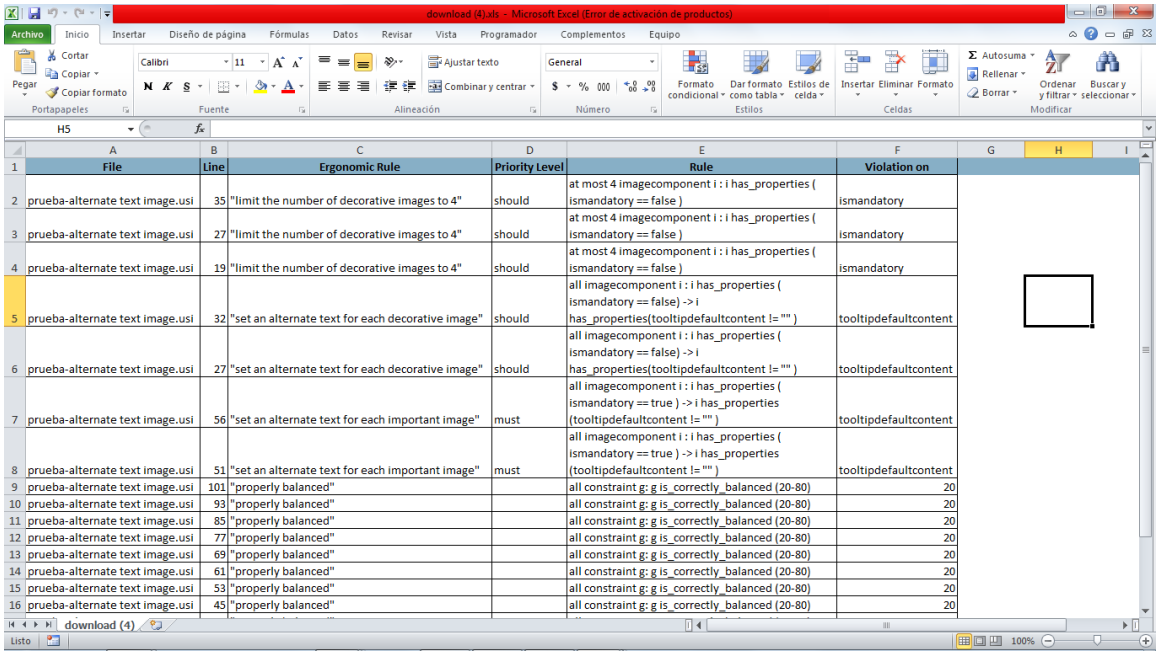


Figura 4.8. Reporte para desarrollador.

- Dar clic sobre el logo “Excel” para exportar a Excel.
- Dar clic sobre el logo “PDF” para exportar a PDF.

Mediante un clic, Usability Adviser nos permite exportar a Excel o exportar a PDF el reporte en cuestión. Las Figuras 4.9 y 4.10 ilustran el formato de exportación del reporte de errores vista desarrollador.



	File	Line	Ergonomic Rule	Priority Level	Rule	Violation on
2	prueba-alternate text image.usi	35	"limit the number of decorative images to 4"	should	at most 4 imagecomponent i : i has_properties (ismandatory == false)	ismandatory
3	prueba-alternate text image.usi	27	"limit the number of decorative images to 4"	should	at most 4 imagecomponent i : i has_properties (ismandatory == false)	ismandatory
4	prueba-alternate text image.usi	19	"limit the number of decorative images to 4"	should	at most 4 imagecomponent i : i has_properties (ismandatory == false)	ismandatory
5	prueba-alternate text image.usi	32	"set an alternate text for each decorative image"	should	all imagecomponent i : i has_properties (ismandatory == false) -> i has_properties(tooltipdefaultcontent != "")	tooltipdefaultcontent
6	prueba-alternate text image.usi	27	"set an alternate text for each decorative image"	should	all imagecomponent i : i has_properties (ismandatory == false) -> i has_properties(tooltipdefaultcontent != "")	tooltipdefaultcontent
7	prueba-alternate text image.usi	56	"set an alternate text for each important image"	must	all imagecomponent i : i has_properties (ismandatory == true) -> i has_properties (tooltipdefaultcontent != "")	tooltipdefaultcontent
8	prueba-alternate text image.usi	51	"set an alternate text for each important image"	must	all imagecomponent i : i has_properties (ismandatory == true) -> i has_properties (tooltipdefaultcontent != "")	tooltipdefaultcontent
9	prueba-alternate text image.usi	101	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
10	prueba-alternate text image.usi	93	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
11	prueba-alternate text image.usi	85	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
12	prueba-alternate text image.usi	77	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
13	prueba-alternate text image.usi	69	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
14	prueba-alternate text image.usi	61	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
15	prueba-alternate text image.usi	53	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20
16	prueba-alternate text image.usi	45	"properly balanced"		all constraint g: g is_correctly_balanced (20-80)	20

Figura 4.9. Exportación a Excel del reporte para desarrollador.

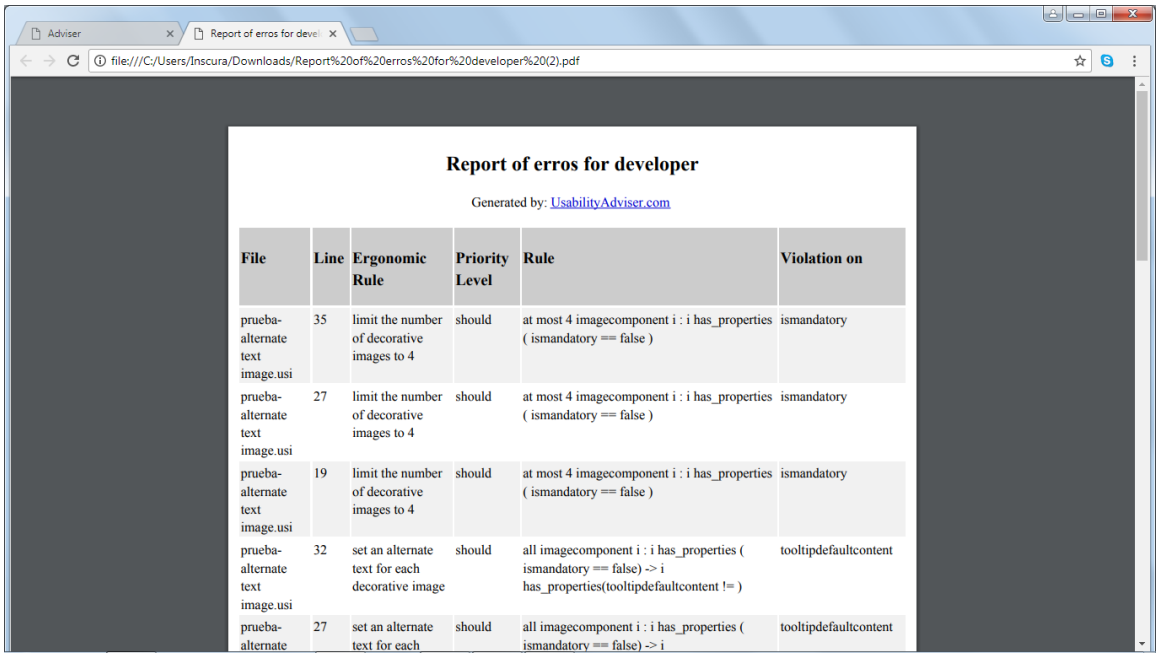
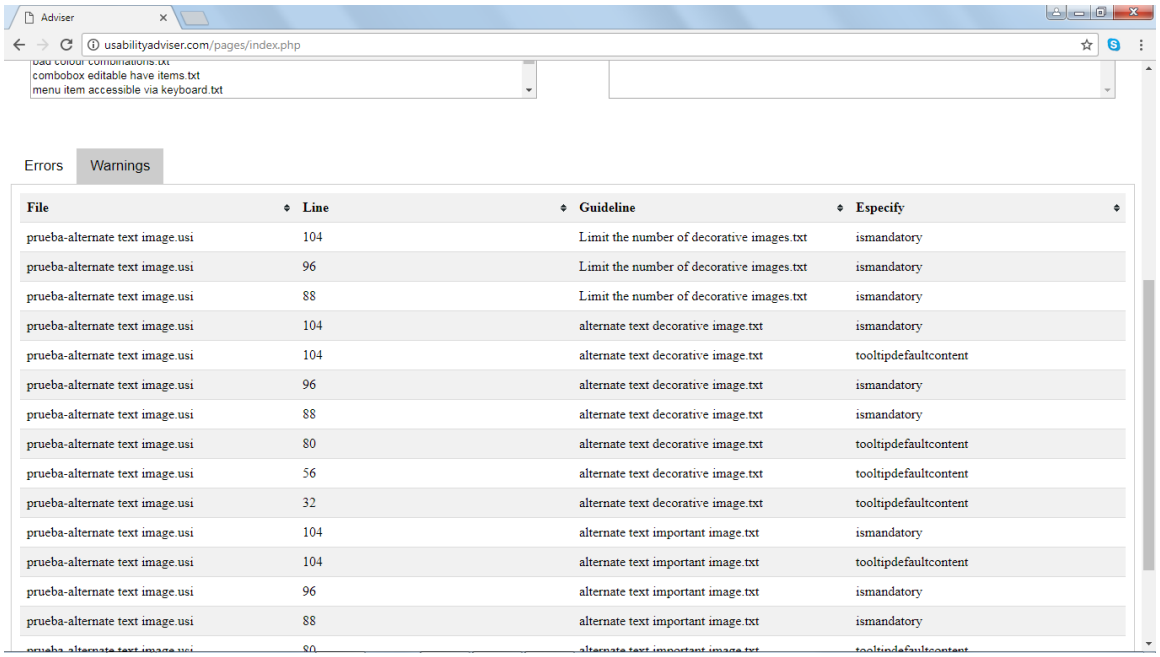


Figura 4.10. Exportación a PDF del reporte para desarrollador.

4.7.Reporte de alertas

Dentro de la sección de resultados, encontramos la pestaña “Alertas” la cual evidentemente presenta alertas, las alertas advierten al usuario de que algo está mal o debería mejorarse, generalmente indican la carencia de propiedad(es) para validar la regla de usabilidad. En este reporte complementario se incluye información básica como: nombre del archivo UsiMXL, línea de código UsiXML que carece de propiedad(es) para validar la regla de usabilidad, regla de usabilidad que no puede ser validada y la propiedad carecida, la Figura 4.11 ilustra este reporte de alertas.



File	Line	Guideline	Especify
prueba-alternate text image.usi	104	Limit the number of decorative images.txt	ismandatory
prueba-alternate text image.usi	96	Limit the number of decorative images.txt	ismandatory
prueba-alternate text image.usi	88	Limit the number of decorative images.txt	ismandatory
prueba-alternate text image.usi	104	alternate text decorative image.txt	ismandatory
prueba-alternate text image.usi	104	alternate text decorative image.txt	tooltipdefaultcontent
prueba-alternate text image.usi	96	alternate text decorative image.txt	ismandatory
prueba-alternate text image.usi	88	alternate text decorative image.txt	ismandatory
prueba-alternate text image.usi	80	alternate text decorative image.txt	tooltipdefaultcontent
prueba-alternate text image.usi	56	alternate text decorative image.txt	tooltipdefaultcontent
prueba-alternate text image.usi	32	alternate text decorative image.txt	tooltipdefaultcontent
prueba-alternate text image.usi	104	alternate text important image.txt	ismandatory
prueba-alternate text image.usi	104	alternate text important image.txt	tooltipdefaultcontent
prueba-alternate text image.usi	96	alternate text important image.txt	ismandatory
prueba-alternate text image.usi	88	alternate text important image.txt	ismandatory

Figura 4.11. Reporte de alertas.

Capítulo 5

Validación de la Propuesta.

En capítulos anteriores se presentaron a detalle los módulos de Usability Adviser, así mismo, se explicó el funcionamiento de la herramienta. En el presente capítulo se abordan tres ejemplos, presentados en dos casos de estudio diferentes y un complementario, con la finalidad de verificar que las interfaces de usuario cumplan las reglas de usabilidad.

Los casos de estudio girarán principalmente en torno a la herramienta Usability Adviser, la cual ofrece una forma rápida y sencilla de evaluar descripciones de interfaces gráficas con base en reglas de usabilidad bien definidas. Además, dando oportunidad a usuarios no familiarizados con la problemática presentada en capítulos anteriores obtener beneficios que puedan resultar interesantes y útiles.

5.1. Caso de estudio 1 – Calculadora

En este primer ejemplo, se realizará un recorrido por la metodología propuesta con un ejemplo sencillo. Se propone analizar la interfaz de una calculadora, es bien conocido que existen múltiples formatos de esta y múltiples plataformas; físico, web, móvil, etc. como se ilustra en la Figura 5.1.

Figura 5.1. Interfaces de calculadoras.

El primer paso es conseguir una especificación de la interfaz de usuario a nivel concreto que refleje estos aspectos en UsiXML. Básicamente esta interfaz constará de un outputText y diecisiete botones (diseño básico). Para la descripción de la interfaz hay dos posibilidades: escribir desde cero la especificación atendiendo la sintaxis UsiXML ó hacer uso de alguna herramienta como GrafiXML. La solución elegida va a ser una mezcla de ambas, si bien, arrastramos de forma sencilla los componentes, la especificación de los detalles se hará a mano en el archivo resultante.

En la Figura 5.2 se ilustra el maquetado de la interfaz de una calculadora básica.

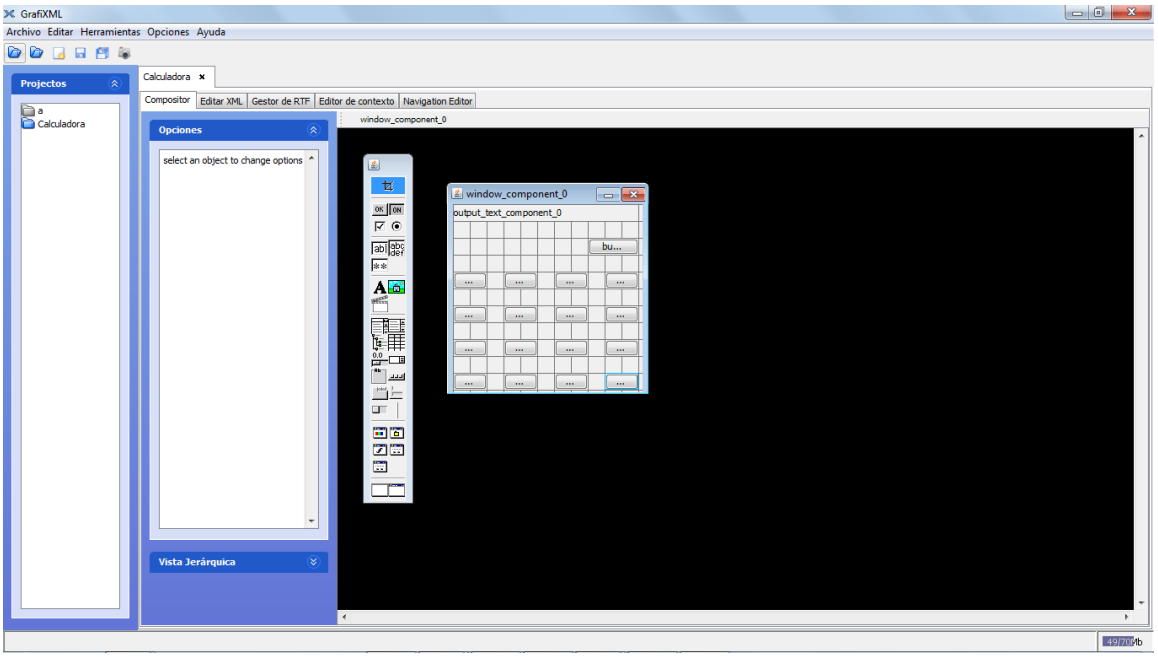


Figura 5.2. Maquetado de calculadora en GrafiXML, Caso de estudio 1.

Una vez maquetada la interfaz, se realizan ajustes manualmente, en este caso se agrega la propiedad `defaultContent="…"` para establecer un contenido predeterminado a cada botón. Al finalizar los ajustes, podremos visualizar el siguiente maquetado, ver Figura 5.3.

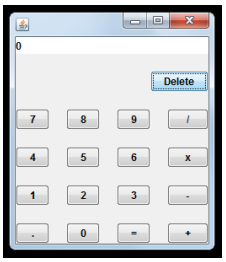


Figura 5.3. Vista previa de calculadora generada por GrafiXML, Caso de estudio 1.

Finalmente, se obtendrá el siguiente archivo UsiXML.

```
<?xml version="1.0" encoding="UTF-8"?>
<uiModel xmlns="http://www.usixml.org"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.usixml.org/
http://www.usixml.org/spec/UsiXML-ui_model.xsd"
 id="Calculadora_4" name="Calculadora"
 creationDate="2018-07-04T14:46:06.023-05:00" schemaVersion="1.6.4">
 <head>
  <version modifDate="2018-07-04T14:46:06.023-05:00">1</version>
  <authorName>Inscura</authorName>
  <comment>Generated by GrafiXML 1.2.0 build id :
200612141408</comment>
  <comment>WARNING : AUI Model save is a work in progress. Use it at
your own risk</comment>
 </head>
 <auiModel id="Calculadora-aui_16" name="Calculadora-aui"/>
 <cuiModel id="Calculadora-cui_16" name="Calculadora-cui">
  <window id="window_component_0" name="window_component_0"
   width="241" height="252">
   <gridBagBox id="grid_bag_box_1" name="grid_bag_box_1"
    gridHeight="12" gridWidth="12">
    <constraint gridx="0" gridy="0" gridwidth="11"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_0"
      name="output_text_component_0"
      defaultContent="0" isVisible="true"
      isEnabled="true" fgColor="#000000"
      bgColor="#ffffff" isBold="true"
      textColor="#000110" textHorizontalAlign="right"
numberOfColumns="43"/>
    </constraint>
    <constraint gridx="8" gridy="2" gridwidth="3"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_1"
      name="button_component_1" defaultContent="Delete"
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="0" gridy="4" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_3"
      name="button_component_3" defaultContent="7"
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="3" gridy="4" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_4"
      name="button_component_4" defaultContent="8"
      isVisible="true" isEnabled="true" textColor="#000000"/>
```

```
 </constraint>
 <constraint gridx="6" gridy="4" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_5"
   name="button_component_5" defaultContent="9"
isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="9" gridy="4" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_6"
   name="button_component_6" defaultContent="/"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="0" gridy="6" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_7"
   name="button_component_7" defaultContent="4"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="3" gridy="6" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_8"
   name="button_component_8" defaultContent="5"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="6" gridy="6" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_9"
   name="button_component_9" defaultContent="6"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="9" gridy="6" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_10"
   name="button_component_10" defaultContent="x"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="0" gridy="8" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_11"
   name="button_component_11" defaultContent="1"
   isVisible="true" isEnabled="true" textColor="#000000"/>
 </constraint>
 <constraint gridx="3" gridy="8" gridwidth="2"
  gridheight="1" weightx="1.0" weighty="1.0"
  fill="both" insets="0,0,0,0">
  <button id="button_component_12"
   name="button_component_12" defaultContent="2"
```

```
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="6" gridy="8" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_13"
      name="button_component_13" defaultContent="3"
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="9" gridy="8" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_14"
      name="button_component_14" defaultContent="-"
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="0" gridy="10" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_0"
      name="button_component_0" isVisible="true"
      isEnabled="true" textColor="#000000" defaultContent="."/>
    </constraint>
    <constraint gridx="3" gridy="10" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_15"
      name="button_component_15" isVisible="true"
      isEnabled="true" textColor="#000000" defaultContent="0"/>
    </constraint>
    <constraint gridx="6" gridy="10" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_16"
      name="button_component_16" isVisible="true"
      isEnabled="true" textColor="#000000" defaultContent="="/>
    </constraint>
    <constraint gridx="9" gridy="10" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_17"
      name="button_component_17" isVisible="true"
      isEnabled="true" textColor="#000000" defaultContent="+"/>
    </constraint>
   </gridBagBox>
  </window>
 </cuiModel>
 <contextModel id="Calculadora-contextModel_4" name="Calculadora-
contextModel">
  <context id="Calculadora-context-en_US_4" name="Calculadora-context-
en_US">
   <userStereotype id="Calculadora-sten_US_4" language="en_US"
stereotypeName="Calculadora-sten_US"/>
   <platform id="Calculadora-platform_4" name="Calculadora-platform"/>
   <environment id="Calculadora-env_4" name="Calculadora-env"/>
```

```
  </context>
 </contextModel>
 <resourceModel id="Calculadora-res_16" name="Calculadora-res"/>
</uiModel>
```

Hasta este punto tenemos completa la descripción de la interfaz gráfica con formato UsiXML, el siguiente paso será acceder a Usability Adviser e introducir esta descripción en la herramienta. Bastará con especificar como entrada el archivo que acabamos de crear, así como filtrar las guidelines que deseamos verificar, en este caso se ha escogido aplicar todas las guidelines, estos paso son ilustrados en la Figura 5.4.

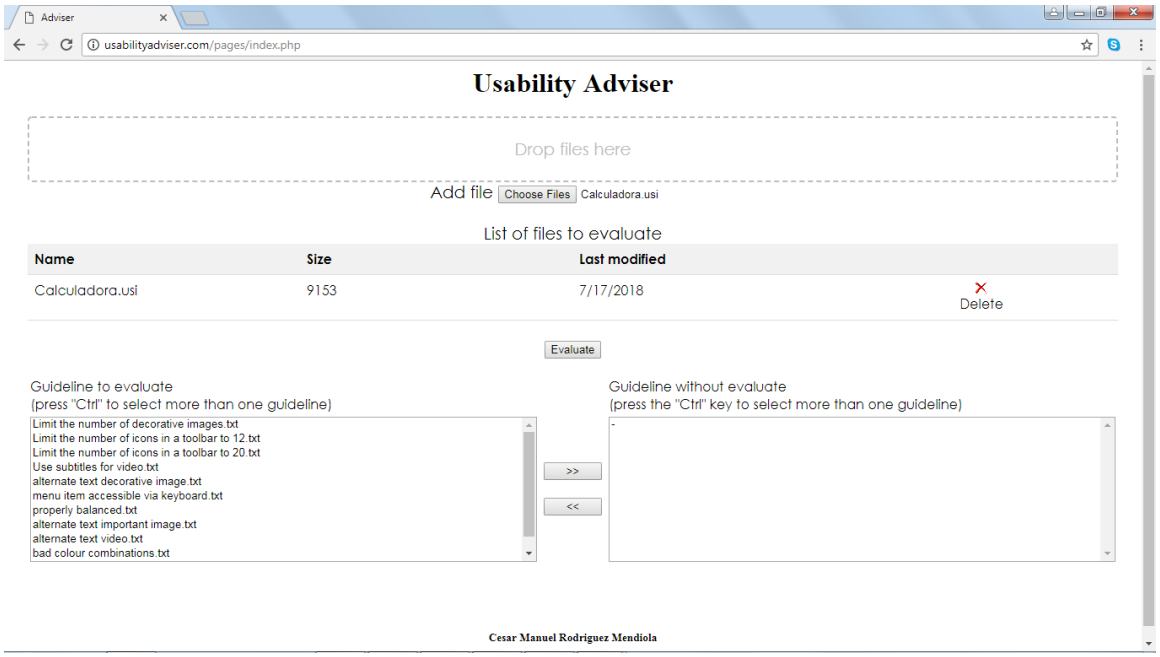


Figura 5.4. Parámetros de entrada para el primer caso de estudio.

El siguiente paso es dar clic en el botón “Evaluar”. Finalmente como resultado obtendremos tres reportes: Certificador, Desarrollador y Alertas. En este caso, gracias al

correcto diseño de la interfaz no se viola ninguna de las guidelines tal como se ilustra en la Figura 5.5, Figura 5.6 y Figura 5.7.

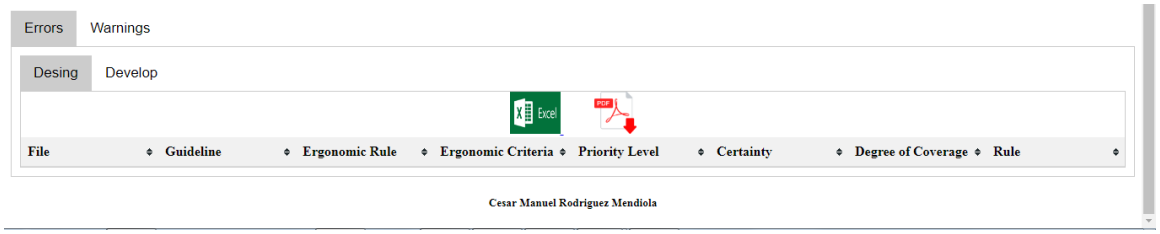


Figura 5.5. Reporte de errores vista Certificador, Caso de estudio 1.

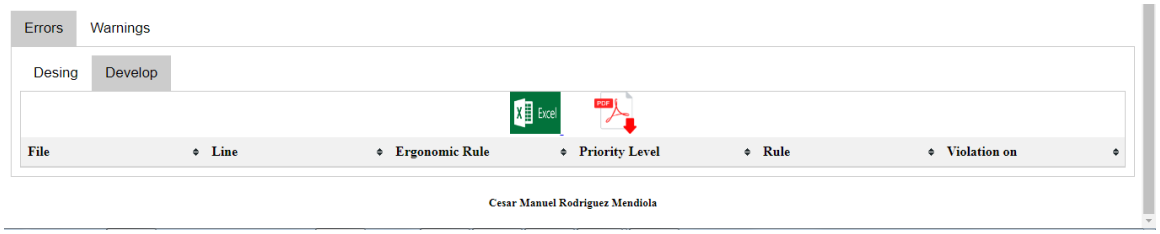


Figura 5.6. Reporte de errores vista Desarrollador, Caso de estudio 1.

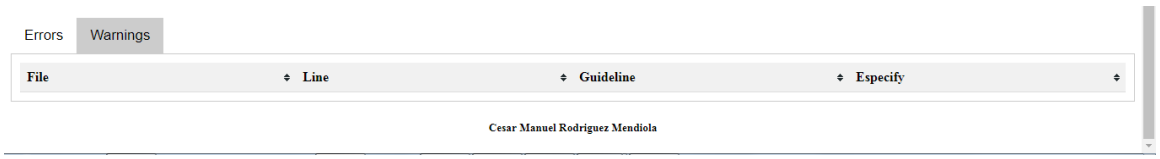


Figura 5.7. Reporte de alertas, Caso de estudio 1.

5.2. Caso de estudio 2 – Calculadora mal diseñada

Para este caso de estudio, se retoma el archivo UsiXML resultante del caso de estudio 1 y se modifica de tal forma que viole algunas de las guidelines disponibles en Usability Adviser, por ejemplo, se elimina la propiedad `defaultContent="0"` del

outputText con lo cual se violaría la regla que estipula “Todo outputText deberá tener un contenido por defecto”, el código resultante se muestra a continuación.

```
<constraint gridx="0" gridy="0" gridwidth="11" gridheight="1"
weightx="1.0" weighty="1.0" fill="both" insets="0,0,0,0">
 <outputText id="output_text_component_0"
name="output_text_component_0" isVisible="true" isEnabled="true"
fgColor="#000000" bgColor="#ffffff" isBold="true" textColor="#000110"
textHorizontalAlign="right" numberOfColumns="43"/>
</constraint>
```

Otra modificación consiste en desfasar uno de los botones con lo cual se violara la regla de distribución balanceada.

La Figura 5.8 ilustra el maquetado incorrecto de la calculadora.

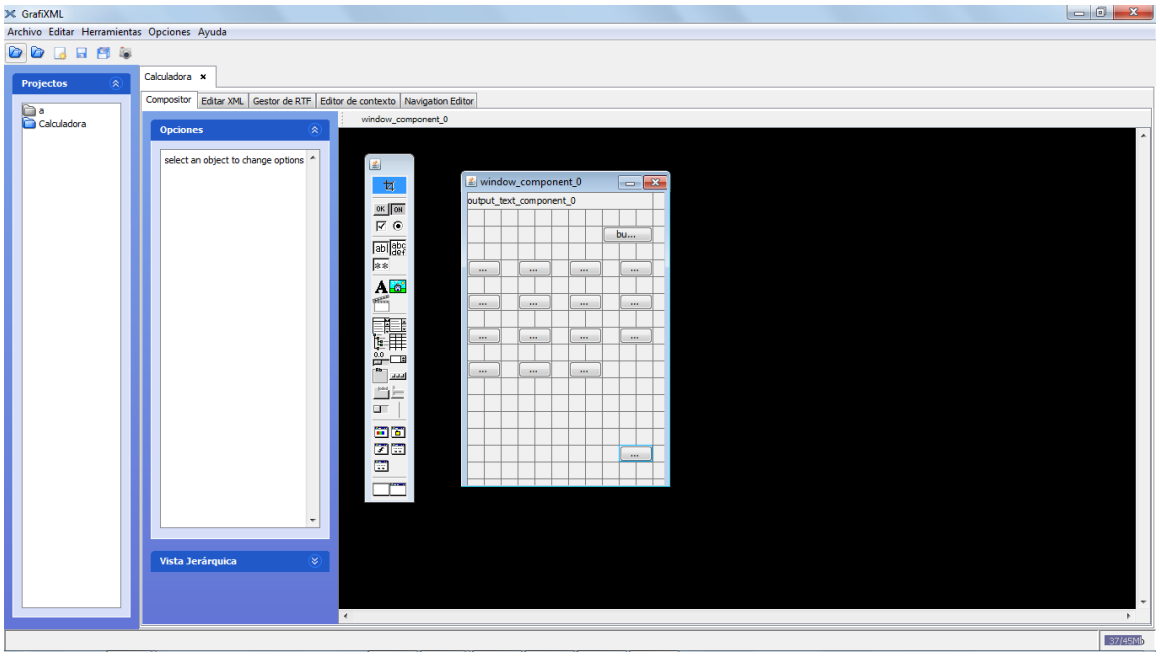


Figura 5.8. Maquetado incorrecto de calculadora en GrafiXML, Caso de estudio 2.

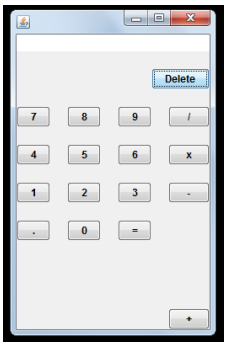


Figura 5.9. Vista previa de calculadora mal diseñada, Caso de estudio 2.

Nuevamente accedemos a Usability Adviser, introducimos el nuevo archivo y aplicamos todas las guidelines como se presenta en la Figura 5.10.

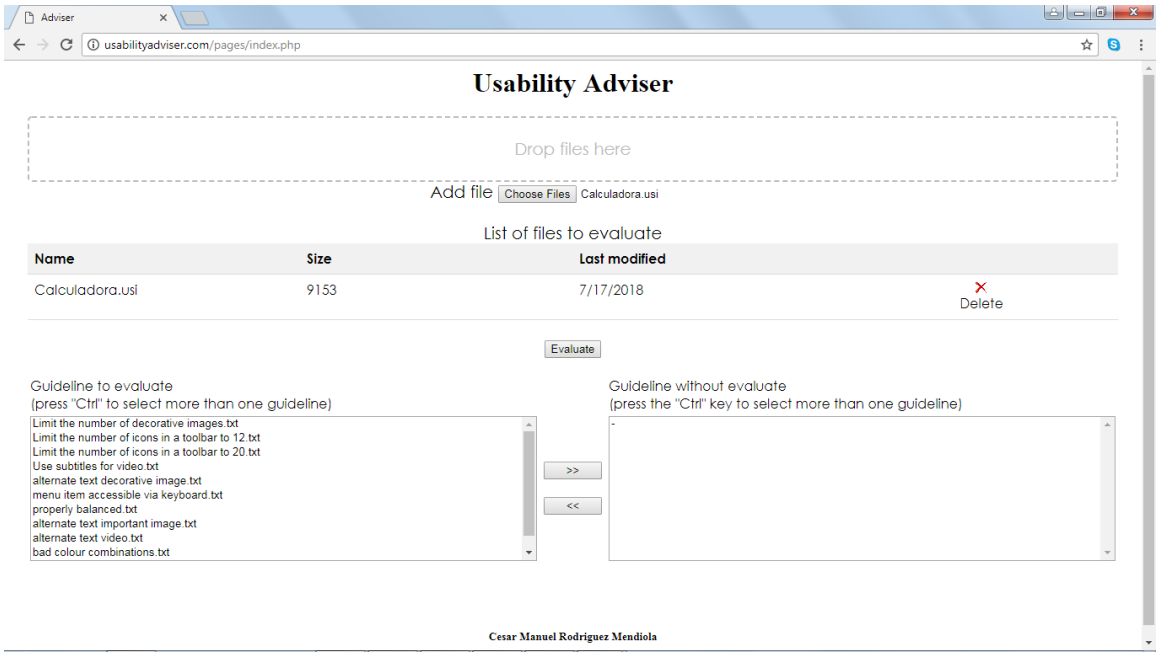


Figura 5.10. Parámetros de entrada para el segundo caso de estudio.

Con lo cual nos arrojara los siguientes resultados:

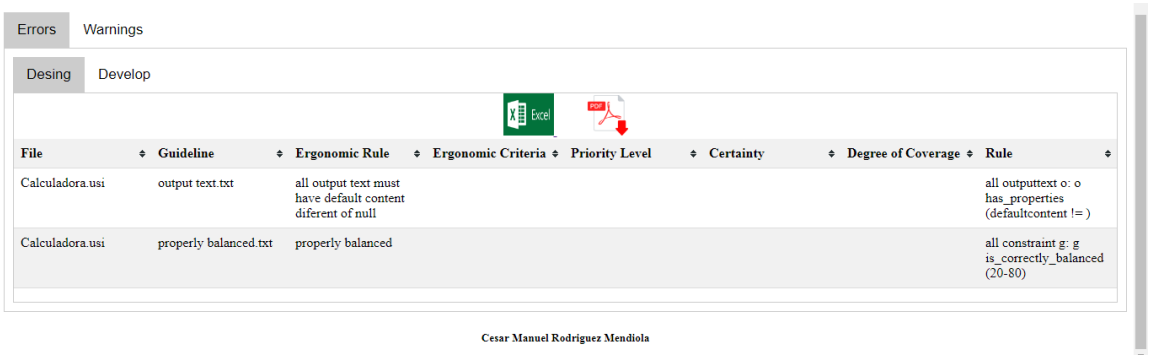


Figura 5.11. Reporte de errores vista Certificador, Caso de estudio 2.

Figura 5.12. Reporte de errores vista Desarrollador, Caso de estudio 2.

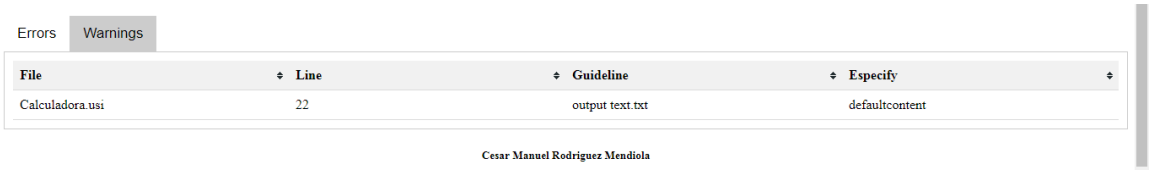
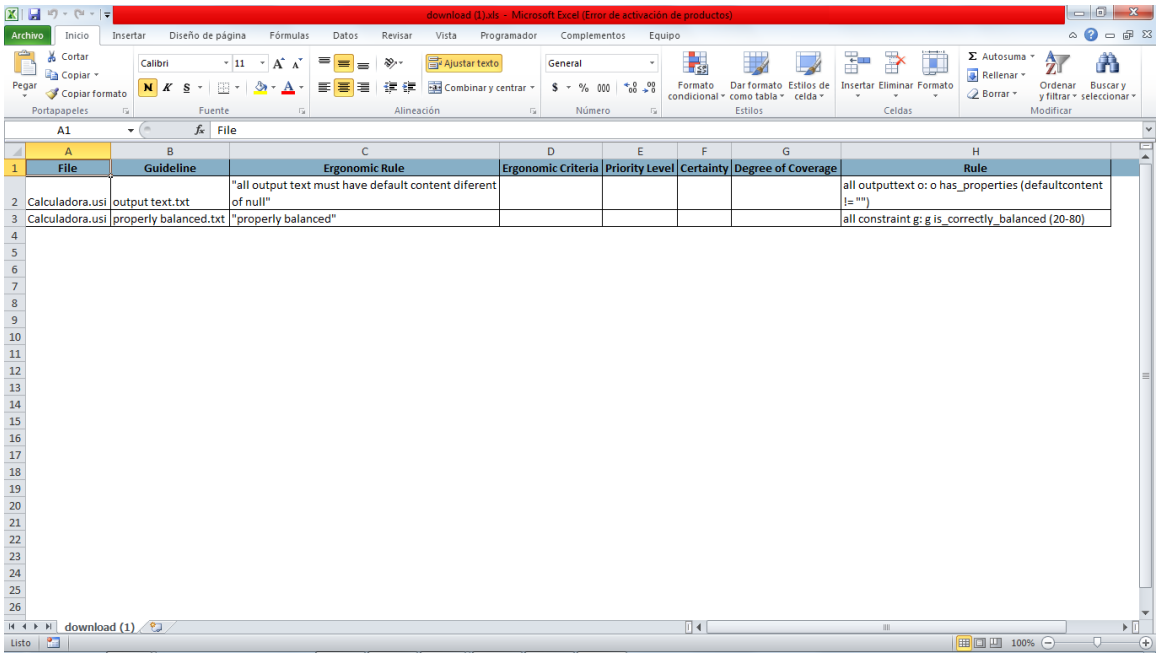


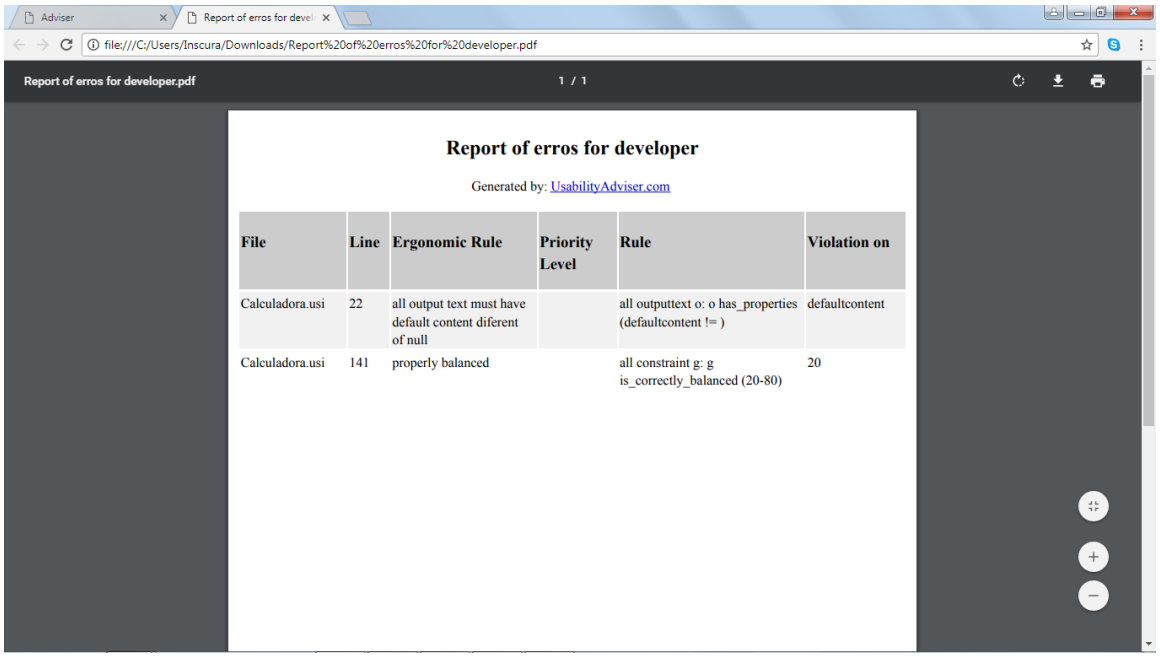
Figura 5.13. Reporte alertas, Caso de estudio 2.

En las Figuras 5.14 y 5.15 se ilustran los formatos descargables de cada reporte.



File	Guideline	Ergonomic Rule	Ergonomic Criteria	Priority Level	Certainty	Degree of Coverage	Rule
Calculadora.usi	output text.txt	"all output text must have default content diferent of null"					all outputtext o: o has_properties (defaultcontent != "")
Calculadora.usi	properly balanced.txt	"properly balanced"					all constraint g: g is_correctly_balanced (20-80)

Figura 5.14. Exportación a Excel del reporte de errores vista Certificador, Caso de estudio 2.



Report of erros for developer

Generated by: UsabilityAdviser.com

File	Line	Ergonomic Rule	Priority Level	Rule	Violation on
Calculadora.usi	22	all output text must have default content diferent of null		all outputtext o: o has_properties (defaultcontent !=)	defaultcontent
Calculadora.usi	141	properly balanced		all constraint g: g is_correctly_balanced (20-80)	20

Figura 5.15. Exportación a PDF del reporte de errores vista Desarrollador, Caso de estudio 2.

5.3. Caso de estudio 3 – Usability Adviser

Como tercer caso de estudio, analizaremos la propia interfaz de usuario de la herramienta Usability Adviser, desarrollada a lo largo de este trabajo de tesis, la cual se compone de etiquetas, botones, tablas, selector de archivos y listas. Procedemos con el maquetado de la interfaz de usuario mediante la herramienta GrafiXML, ver Figura 5.16.

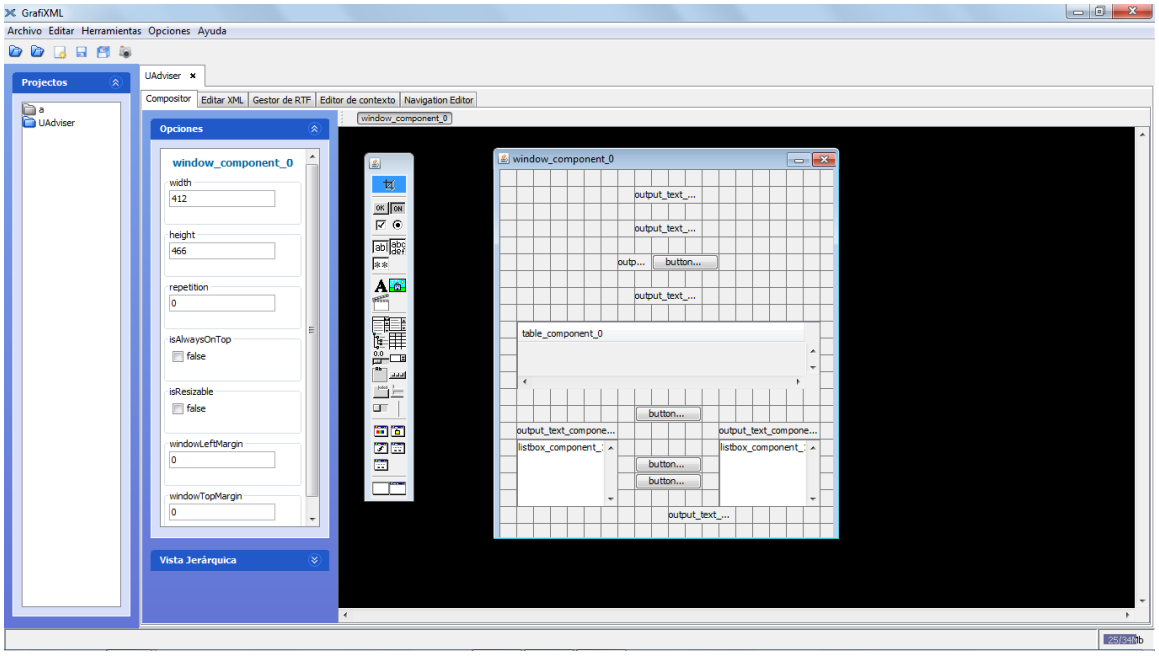


Figura 5.16. Maquetado de Usability Adviser en GrafiXML, Caso de estudio 3.

Mediante código, agregamos la propiedad `defaultContent="…"` e introducimos el texto correspondiente a cada etiqueta. A continuación se presenta el código UsiXML resultante.

```
<?xml version="1.0" encoding="UTF-8"?>
<uiModel xmlns="http://www.usixml.org"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xsi:schemaLocation="http://www.usixml.org/
http://www.usixml.org/spec/UsiXML-ui_model.xsd"
 id="UAdviser_4" name="UAdviser"
 creationDate="2018-07-04T19:41:21.592-05:00" schemaVersion="1.6.4">
 <head>
  <version modifDate="2018-07-04T19:41:21.592-05:00">1</version>
  <authorName>Inscura</authorName>
  <comment>Generated by GrafiXML 1.2.0 build id :
200612141408</comment>
  <comment>WARNING : AUI Model save is a work in progress. Use it at
your own risk</comment>
 </head>
 <auiModel id="UAdviser-aui_24" name="UAdviser-aui"/>
 <cuiModel id="UAdviser-cui_24" name="UAdviser-cui">
  <window id="window_component_0" name="window_component_0"
   width="412" height="466">
   <gridBagBox id="grid_bag_box_1" name="grid_bag_box_1"
    gridHeight="23" gridWidth="20">
    <constraint gridx="8" gridy="1" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_2"
      name="output_text_component_2"
      defaultContent="Usability Adviser"
      isVisible="true" isEnabled="false"
      fgColor="#000000" bgColor="#cccccc"
      isBold="false" textColor="#000000"
      textVerticalAlign="middle" textHorizontalAlign="middle"/>
    </constraint>
    <constraint gridx="8" gridy="3" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_1"
      name="output_text_component_1"
      defaultContent="Drop files here"
      isVisible="true" isEnabled="true"
      fgColor="#000000" bgColor="#cccccc"
      isBold="true" textColor="#000000" textHorizontalAlign="middle"/>
    </constraint>
    <constraint gridx="7" gridy="5" gridwidth="2"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_0"
      name="output_text_component_0"
      defaultContent="Add file" isVisible="true"
      isEnabled="true" fgColor="#000000"
      bgColor="#cccccc" isBold="true"
      textColor="#000000" textHorizontalAlign="right"/>
    </constraint>
    <constraint gridx="9" gridy="5" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_0"
      name="button_component_0"
      defaultContent="Choohe Files" isVisible="true"
```

```
      isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="8" gridy="7" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_3"
      name="output_text_component_3"
      defaultContent="List of files to evaluate"
      isVisible="true" isEnabled="true"
      fgColor="#000000" bgColor="#cccccc"
      isBold="true" textColor="#000000"
      textVerticalAlign="middle" textHorizontalAlign="middle"/>
    </constraint>
    <constraint gridx="8" gridy="14" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_1"
      name="button_component_1"
      defaultContent="Evaluate" isVisible="true"
      isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="1" gridy="15" gridwidth="6"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_4"
      name="output_text_component_4"
      defaultContent="Guideline to evaluate  (press 'Ctrl' to select
more than one guideline)"
      isVisible="true" isEnabled="true" isBold="true"
textColor="#000000"/>
    </constraint>
    <constraint gridx="13" gridy="15" gridwidth="6"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_5"
      name="output_text_component_5"
      defaultContent="Guideline without evaluate  (press the 'Ctrl' key
to select more than one guideline"
      isVisible="true" isEnabled="true" isBold="true"
textColor="#000000"/>
    </constraint>
    <constraint gridx="8" gridy="17" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_9"
      name="button_component_9" defaultContent="skip"
      isVisible="true" isEnabled="true" textColor="#000000"/>
    </constraint>
    <constraint gridx="8" gridy="18" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <button id="button_component_10"
      name="button_component_10"
      defaultContent="attach" isVisible="true"
      isEnabled="true" textColor="#000000"/>
```

```
    </constraint>
    <constraint gridx="10" gridy="20" gridwidth="4"
     gridheight="1" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <outputText id="output_text_component_12"
      name="output_text_component_12"
      defaultContent="CopyRight" isVisible="true"
      isEnabled="true" isBold="true"
      textColor="#000000" textHorizontalAlign="middle"/>
    </constraint>
    <constraint gridx="1" gridy="16" gridwidth="6"
     gridheight="4" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <listBox id="listbox_component_14"
      name="listbox_component_14" isVisible="true"
      isEnabled="true" textColor="#000000" multiple_selection="false"/>
    </constraint>
    <constraint gridx="13" gridy="16" gridwidth="6"
     gridheight="4" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <listBox id="listbox_component_15"
      name="listbox_component_15" isVisible="true"
      isEnabled="true" textColor="#000000" multiple_selection="false"/>
    </constraint>
    <constraint gridx="1" gridy="9" gridwidth="18"
     gridheight="4" weightx="1.0" weighty="1.0"
     fill="both" insets="0,0,0,0">
     <table id="table_component_0"
      name="table_component_0" isVisible="true"
      isEnabled="true" xSize="1" ySize="1"/>
    </constraint>
   </gridBagBox>
  </window>
 </cuiModel>
 <contextModel id="UAdviser-contextModel_4" name="UAdviser-
contextModel">
  <context id="UAdviser-context-en_US_4" name="UAdviser-context-en_US">
   <userStereotype id="UAdviser-sten_US_4" language="en_US"
stereotypeName="UAdviser-sten_US"/>
   <platform id="UAdviser-platform_4" name="UAdviser-platform"/>
   <environment id="UAdviser-env_4" name="UAdviser-env"/>
  </context>
 </contextModel>
 <resourceModel id="UAdviser-res_5" name="UAdviser-res"/>
</uiModel>
```

La versión “1.2.0” de GrafiXML, no cuenta con un modelo que permita la visualización previa de tablas, por lo cual excluiremos dicho modelo solamente para generar la vista previa del diseño de la interfaz ilustrada en la Figura 5.17.

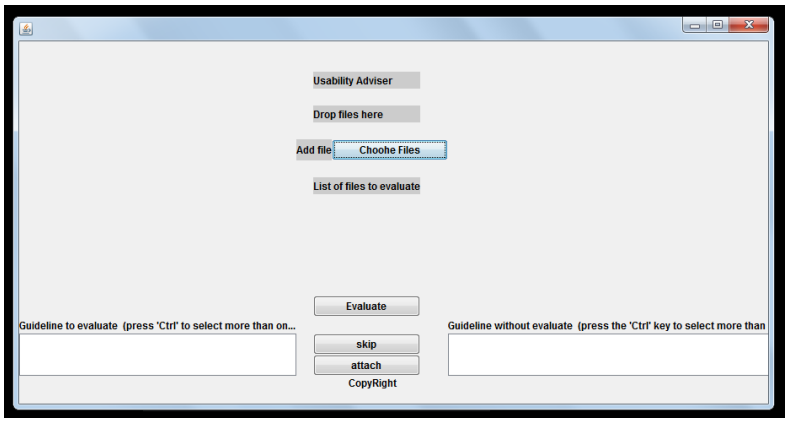


Figura 5.17. Vista previa de Usability Adviser, Caso de estudio 3.

Finalmente, accedemos a Usability Adviser y proporcionamos el archivo .usi descrito anteriormente. Aplicaremos todas las guidelines del sistema.

Figura 5.18. Parámetros de entrada para el tercer caso de estudio.

Damos clic en el botón “Evaluar” y esperamos los resultados. Debido a un correcto diseño de interfaz, como podemos ver en las Figuras 5.19, 5.20 y 5.21, no se viola ninguna de las guidelines evaluadas por la herramienta.

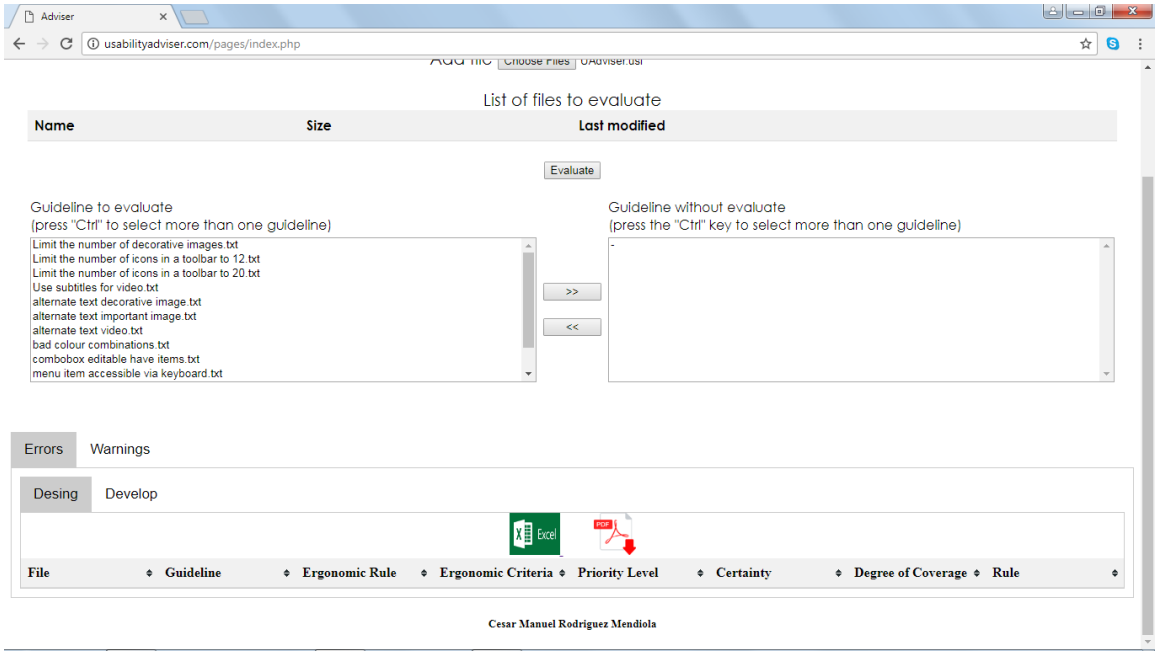


Figura 5.19. Reporte de errores vista Certificador, Caso de estudio 3.

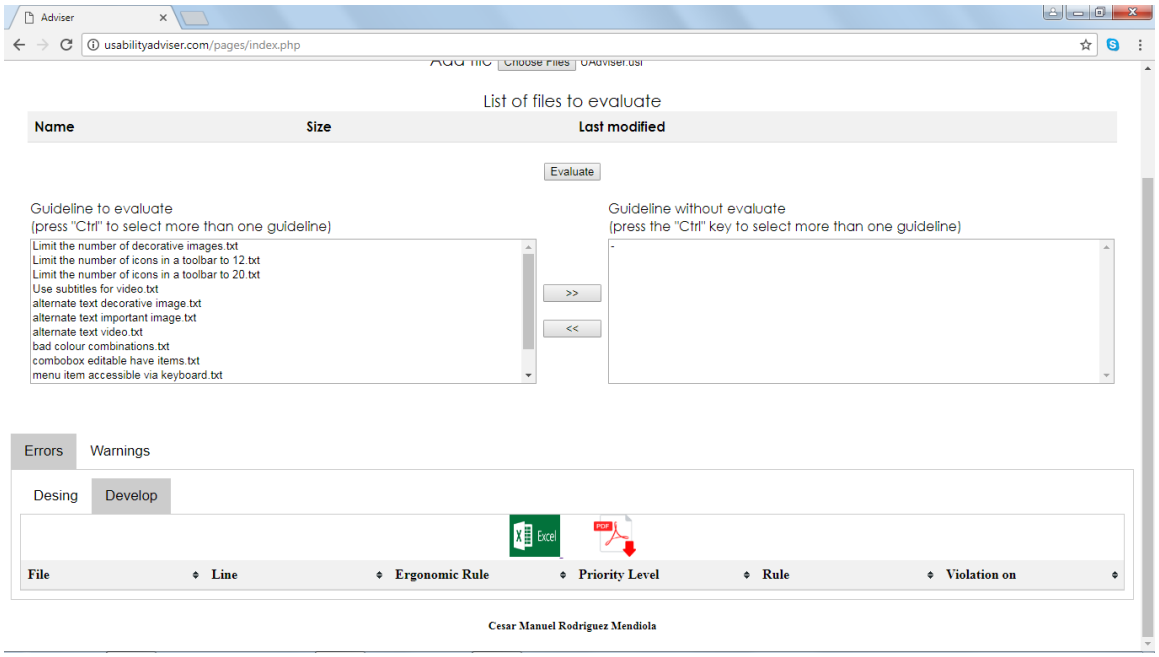


Figura 5.20. Reporte de errores vista Desarrollador, Caso de estudio 3.

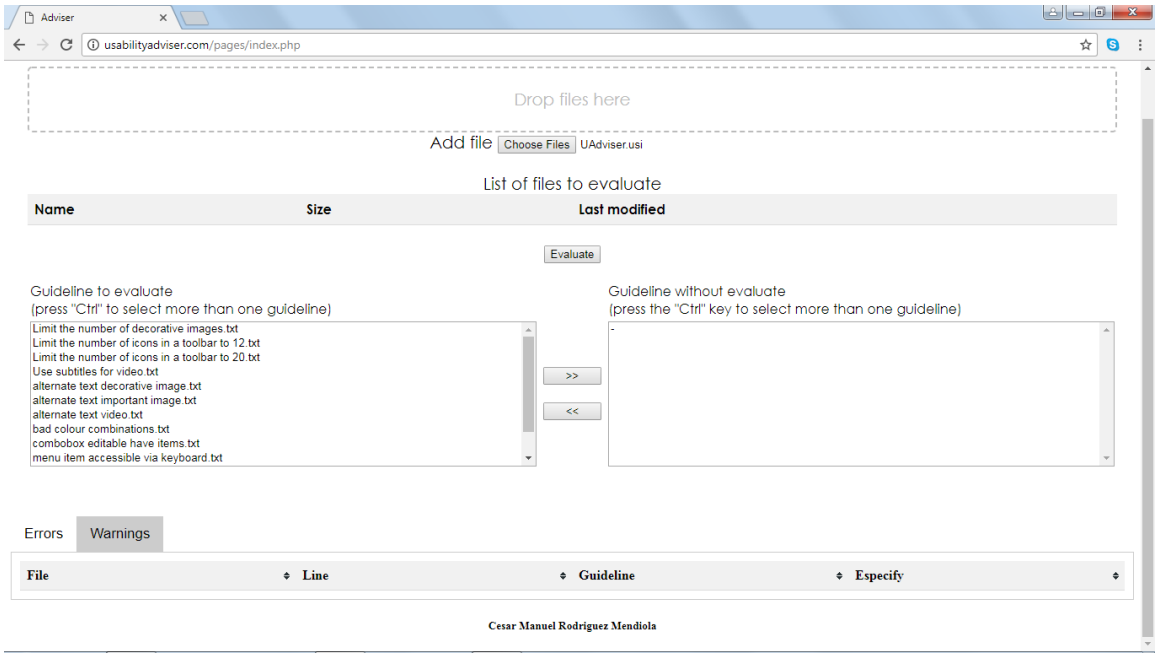


Figura 5.21. Reporte de alertas, Caso de estudio 3.

Capítulo 6

Conclusiones y trabajos futuros.

6.1. Conclusiones

La usabilidad es reconocida como un factor de calidad importante para sistemas interactivos, incluidas las interfaces de usuario de una gran variedad de dispositivos.

El diseño de aplicaciones, de tal manera que logre efectivamente sus propósitos en términos de facilidad de uso, no siempre resulta tarea fácil ante la existencia de varios métodos y técnicas para la evaluación de la usabilidad.

A lo largo de esta tesis hemos desarrollado una herramienta web para la evaluación automatizada de interfaz de usuario con una arquitectura cognitiva, que se puede utilizar en aplicaciones comerciales. Se implementó con base en un enfoque basado en modelos, y se puede conectar a herramientas existentes en el mercado como GrafiXML, lo cual permite la reutilización de modelos definidos por diseñadores de sistemas. La metodología desarrollada, hace parte de las técnicas automatizadas de acuerdo con la clasificación presentada en esta tesis. Sin embargo, recordemos que las pruebas automatizadas no desplazan las pruebas manuales o análisis de expertos por ende, se recomienda que el uso de Usability Adviser se acompañe de pruebas manuales.

El objetivo de la evaluación de usabilidad de interfaces, es utilizar los resultados para realizar una retroalimentación, con el fin de mejorar los diseños y por ende la interactividad con el usuario. La herramienta desarrollada le permite a un inexperto en usabilidad agilizar la evaluación para proponer mejoras que den lugar a la optimización de la interfaz.

6.2. Trabajos futuros

Durante el desarrollo de este proyecto, se identificaron posibles trabajos futuros relacionados con esta tesis. Se proponen dichos trabajos para que posteriormente puedan ser realizados, y así fortalecer esta línea de investigación:

- Desarrollar una aplicación web para gestionar una comunidad de expertos quienes podrán dotar de nuevas guidelines la base de datos existente de Usability Adviser, así mismo tendrán acceso y privilegios de edición sobre las guidelines actualmente disponibles.
- Dotar de nuevas funcionalidades a la herramienta, por ejemplo:
    - Validación de métodos codificados en lenguajes de programación como son: Java, C Sharp, Phyton, etc.
    - Calcular el grado de usabilidad que tiene la descripción de la interfaz. El grado de usabilidad seria en función de la colocación, la cantidad y el tipo de elementos que conforman la interfaz de usuario.
- Establecer lazos de comunicación con creadores de herramientas gráficas que generan descripción de interfaz en formato UsiXML, tal es el caso de GrafiXML, IdealXML, EgiuXML, etc. Realizar los ajustes oportunos para soportar los distintos formatos y permitirnos empotrar estas soluciones en sus herramientas. Así potenciaríamos el uso de ambas herramientas.
- Ampliar la gama de formatos de descripción de interfaz de usuario soportados por Usability Adviser, es decir, analizar otras metodologías que permitan la especificación de interfaz de usuario, tales como IDEAS, UWE y UMLi (basadas en UML), TERESA (basada en ConcurTaskTrees, CTT), etc.

- Quizás el más ambicioso de los trabajos propuestos sea éste, desarrollar un algoritmo basado en inteligencia artificial que permita traducir guías de desarrollo y/o reglas de usabilidad descritas en lenguaje natural en guías y/o reglas de usabilidad descritas en lenguaje formal. El lenguaje formal debería mantener la sintaxis definida en el apartado 3.1 de esta tesis, que pueda ser interpretada por algún sistema informático y por supuesto deberá ser interpretada y evaluada por Usability Adviser.

Referencias

1. Aho, A., Compiladores principios, técnicas y herramientas, Person/Addison Wesley (2008).
2. Bartek, V., Cheatham, D., Experience remote usability testing, Part1, examine study results on the benefits and downside of remote usability testing (2003).
3. Calleros, J., García, J., & Vanderdonckt, J. (n.d). Advance human-machine interface automatic evaluation. *Universal Access In The Information Society*, *12*(4), 387-401 (2013).
4. Calleros, J., Osterloh, J., Feil, R., Lüdtke, A., Automated UI Evaluation based on a Cognitive Architecture and UsiXML (2012).
5. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers. Vol.15, No. 3, pages 289-308 (2003).
6. Campbell, E., Maintaining accessible Websites with Microsoft Word and XML. In: Proceedings of XML Europe 2003 (London, 5–8 May 2003), XML Workshop Ltd., London (2002).
7. Cobo, A., Gómez, P., Rocha, R., PHP y MySQL Tecnologías para el desarrollo de aplicaciones web. España (2005).
8. Dray, S., Siegel, D., Remote possibilities, international usability testing at a distance. Interactions V. 11, issue 2, p. 10-17. ACM Press (2004).
9. Ferré, X., Marco de integración de la usabilidad en el proceso de desarrollo de software, Tesis Doctoral, Universidad Politécnica de Madrid (2005).
10. Flanagan, D., JavaScript: The Definitive Guide, O'Reilly, (2002)
11. Florián, B., Solarte, O. & Reyes, J. Propuesta para incorporar evaluación y pruebas de usabilidad dentro de un proceso de desarrollo de software, Revista EIA, núm. 13, pp. 123-141 (2010).
12. Garrett, J., Ajax: A new Approach to Web Applications, URL http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/ Consultado en 2018.
13. GrafiXML. GrafiXML graphical tool. URL http://www.usixml.org/en/michotte-b-vanderdonckt-j-grafixml-a-multi-target-user-interface-builder-based-on-usixml.html?IDC=465&IDD=1593 Consultado en 2018.
14. Granollers, T., MPIu+a. Una metodología que integra la ingeniería del software, la interacción persona ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares. Tesis Doctoral. Departament de Llenguatges i Sistemes Informàtics, Universitat de Lleida (2004).
15. Google Anaytics. URL https://marketingplatform.google.com/about/analytics Consultado en 2018.
16. HTML. URL https://www.w3schools.com Consultado en 2018.
17. JQuery. URL https://jquery.com Consultado en 2018.
18. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V.: UsiXML: a language supporting multi-path development of user interfaces. In: Proceedings of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th International Workshop on Design, Specification, and Verification

of Interactive Systems EHCIDSVIS'2004 (Hamburg July 11–13, 2004). Springer, Berlin (2004).

19. McCathieNevile, Ch., Koivunen, M.-R., Accessibility Features of SVG, W3C Note 7, W3C, August 2000, accessible at http://www.w3.org/TR/SVG-access (2000).
20. Metricspot. URL http://metricspot.com Consultado en 2018.
21. Michotte, B., Vanderdonckt, J., GrafiXML, A Multi-Target User Interface Builder based on UsiXML. IEEE DOI 10.1109/ICAS.2008.29
22. Nunes, N. J. Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach. Tesis Doctoral. Universidad de Madeira, Portugal (2001).
23. SIRIUS. URL https://olgacarreras.blogspot.com/2011/07/sirius-nueva-sistema-para-la-evaluacion.html Consultado en 2018.
24. Torrente M,. SIRIUS: Sistema de evaluación de la usabilidad web orientado al usuario y basado en la determinación de tareas críticas. Tesis doctoral. Universidad de Oviedo (2011).
25. User Interface eXtensible Markup Language. URL http://www.usixml.org/en/home.html?IDC=221 Consultado en 2018.
26. Web Static Analyzer Tool. URL http://zing.ncsl.nist.gov/WebTools/WebSAT/overview.html Consultado en 2018.