Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA

Migda Rosalba Peralta Herrera
Asesorado por el Ing. Edwin Estuardo Zapeta Gómez

Guatemala, septiembre de 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

MIGDA ROSALBA PERALTA HERRERA
ASESORADO POR EL ING. EDWIN ESTUARDO ZAPETA GÓMEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA EN CIENCIAS Y SISTEMAS

GUATEMALA, SEPTIEMBRE DE 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Caceres
EXAMINADOR	Ing. Roberto Estuardo Ruíz Cruz
EXAMINADOR	Ing. Luis Fernando Quiñonez López
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, en el mes de enero del 2014.

Migda Rosalba Peralta Herrera

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 08 de agosto de 2014

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador del Área de Trabajos de Graduación

Respetable Ingeniero Azurdia:

Por este medio informo que he revisado y aprobado el trabajo de investigación titulado: **"TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA",** desarrollado por la estudiante **Migda Rosalba Peralta Herrera**, quien se identifica con el número de **carné 200610987**, ya que considero que cumple con los requisitos establecidos, por lo que el autor y mi persona somos responsables del contenido y conclusiones del mismo.

Agradeciendo su atención a la presente.

Atentamente.

Ing. Estuardo Zapeta
Ingeniería en Ciencias y Sistemas
Colegiado 12767

Ing. Edwin Estuardo Zapeta Gómez
Escuela de Ciencias y Sistemas
Asesor de trabajo de graduación
Colegiado: 12767

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 3 de Septiembre de 2014

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de la estudiante **MIGDA ROSALBA PERALTA HERRERA** con carné **2006-10987,** titulado: **"TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA"****, realizado por la estudiante MIGDA ROSALBA PERALTA HERRERA, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pérez Türk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de septiembre 2014

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.511-2014

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **TECNOLOGÍAS DE DESARROLLO DE INTERFAZ GRÁFICA PARA APLICACIONES WEB MULTIPLATAFORMA,** presentado por la estudiante universitaria: **Migda Rosalba Peralta Herrera,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERÍA

Guatemala, septiembre de 2014

/cc

ACTO QUE DEDICO A:

Dios	Por darme la sabiduría necesaria, por ayudarme cuando los problemas estaban fuera de mí alcance, por poner a la gente indicada en mi camino y por estar siempre a mi lado bendiciéndome.
Mis padres	Por aconsejarme siempre a salir adelante, por ser un ejemplo a seguir, por apoyarme tanto financiera como moralmente, por todo su amor y cuidado durante mi vida.
Mi mamá Yoly	Por todo su amor y cariño, por sus comidas, por todos sus detalles y por siempre alentarme a seguir adelante.
Mis hermanos	Por su apoyo, por estar conmigo en los momentos difíciles y porque estoy segura que siempre estarán a mi lado ayudándome en lo que puedan.
Mis amigos de la universidad	Porque con ellos viví momentos inolvidables, por todos los desvelos que compartimos, por todos los proyectos que hicimos juntos, por todos los momentos divertidos tanto dentro como fuera de la universidad, pero sobre todo, por su amistad incondicional.

Mis amigos en general	Por apoyarme siempre, escuchar mis angustias, mis deseos, mis sueños, por siempre tener palabras de aliento para conmigo y porque sin ustedes muchos de mis anhelos no serían realidad.
Mi novio	Por creer en mí, por estar a mi lado, por nunca dejar que me rindiera, por hacerme reír, por quererme y por ayudarme siempre que lo necesitaba.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	La casa de estudio que me dio la oportunidad de formarme como profesional y por proveerme de los medios necesarios para consolidar mis estudios.
Facultad de Ingeniería	Por brindarme las herramientas, instalaciones y personas adecuadas para enriquecer mi conocimiento y poder crecer profesionalmente.
Mi asesor	Por apoyarme en el desarrollo de mi trabajo de graduación, por evaluar y analizar cada aspecto de este trabajo, colaborando a que se obtuviera un trabajo de calidad.
Mis catedráticos	Por transmitirme sus conocimientos, por sus consejos durante la carrera y por toda la sabiduría que me transmitieron.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES V
GLOSARIO VII
RESUMEN XI
OBJETIVOS XIII
INTRODUCCIÓN XV

1. INTERFACES GRÁFICAS 1
   1.1. Satisfacción del usuario 1
   1.2. Inicios de la interfaz 1
   1.3. Definición de interfaz gráfica 2
      1.3.1. Interfaz gráfica web 3
      1.3.2. Interfaz gráfica web: nuevos desafíos 3
   1.4. Ingeniería de la interfaz 4
   1.5. Ciclo de vida de un sistema interactivo 5
   1.6. Fases 6
   1.7. Diseño incremental centrado en el usuario 8
   1.8. Utilidad del diseño centrado en el usuario 9
      1.8.1. Ventajas sobre la competencia 9
      1.8.2. Ser directo y específico con el usuario 10
      1.8.3. Reducción de costos 11
   1.9. Aplicación del diseño centrado en el usuario 12
      1.9.1. Recolección de requerimientos 12
      1.9.2. Normas 12
      1.9.3. Diseño 13

1.9.4. Testeo 13
1.10. Otros enfoques de usabilidad 13
1.10.1. Diseño participativo 13
1.10.2. Diseño centrado en la actividad 14
1.10.3. Diseño centrado en el proceso 14
1.10.4. Diseño centrado en el uso 14
1.11. Hacia nuevos horizontes en la creación de interfaces multiplataforma 15
1.11.1. Propuestas metodológicas: Teresa 15
1.11.2. Propuestas metodológicas: OO-H 16
1.11.3. Propuestas metodológicas: Cameleon 17

2. LENGUAJES Y NOTACIONES 19
2.1. Lenguaje de marcado 19
2.1.1. Características de los lenguajes de marcas 19
2.1.2. Un nuevo panorama de la estructuración de documentos 20
2.1.3. Partes de un esquema 21
2.1.3.1. Elementos 21
2.2. UIML 21
2.2.1. Modelo de UIML 23
2.2.2. Características 23
2.2.3. Elementos de UIML 25
2.2.4. Estructura de un documento UIML 27
2.3. XIML 27
2.3.1. Relación 28
2.3.2. Atributos 28
2.3.3. Interfaces multiplataforma 28
2.4. XUL 28

2.5. HTML 30
2.6. Dinamyc HTML 31
2.7. WML 32
2.8. Java 33
2.8.1. Características 33
2.8.2. Ventajas 34
2.8.3. Desventajas 34
2.9. Análisis comparativo 34

3. IMPLEMENTACIÓN 37
3.1. Implementación de una interfaz gráfica utilizando lenguajes de marcado 37
3.1.1. Inicios de UIML 37
3.1.2. Estructura 38
3.1.3. Componentes 39
3.1.4. Estilo 40
3.1.5. Contenido 41
3.1.6. Comportamiento 42
3.1.7. Lógica 43
3.1.8. Presentación 44
3.1.9. Interfaces multiplataforma 45
3.1.10. Múltiples documentos UIML 46
3.1.11. Vocabulario genérico 47
3.1.12. Mapeando un vocabulario genérico a Java y HTML 48
3.1.13. Describiendo las familias de interfaces de usuario 49
3.1.14. Múltiples elementos <structure> 50
3.1.15. Diferentes diseños 50
3.1.16. Renderizador de UIML 51

3.1.17. UIML No es 53
3.1.18. Implementación de un diccionario utilizando XIML. 53
3.1.18.1. Especificaciones de XIML 54
3.1.18.2. Comentarios 62
3.1.18.3. Especificaciones en UIML 64
3.1.19. Diferencias entre XIML y UIML 69

CONCLUSIONES 73
RECOMENDACIONES 75
BIBLIOGRAFÍA 77

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Ciclo de vida – sistema interactivo 5
2. Esquema de un sistema interactivo 7
3. Modelo de UIML 23
4. Uso de vocabulario especifico contra el uso de vocabulario genérico. 24
5. Encabezado del documento 27
6. Integración de XUL con Java 29
7. Estructura general de un documento UIML 39
8. Componentes 40
9. Estilo 41
10. Contenido 42
11. Comportamiento 43
12. Lógica 44
13. Presentación 45
14. Interfaces multiplataforma 46
15. Mapeo a Java y HTML 48
16. Describiendo las familias de interfaces de usuario 49
17. Múltiples elementos <structure> 50
18. Renderizador de UIML 52
19. Interfaz del diccionario 54
20. Especificaciones de XIML 55
21. Especificariones en la interfaz del XIML 56
22. Definiciones 57
23. Componentes del modelo 58

24. Presentación del modelo 59
25. Elementos de presentación 59
26. AIOs propios 60
27. Mas AIOs 61
28. Modelo de diálogo 62
29. Dictionary.ui 64
30. Asignaciones 65
31. Definición de un modelo widget 65
32. Definición de un modelo de presentación 66
33. Definiciones para el estilo 67
34. Normas de comportamiento 68
35. Otras normas de comportamiento 69

TABLAS

I. Fases del ciclo de vida 6
II. Análisis comparativo 35
III. Vocabulario 47
IV. Propiedades y eventos 48

GLOSARIO

Comando	Es una instrucción que se le da a un sistema informático.
Dispositivo	Es un aparato que realiza acciones específicas con el fin de satisfacer una necesidad, como por ejemplo una computadora, celular, televisión, PDA's, tablets, etc.
Extranet	Es parte de la intranet, pero puede ser accedida afuera de la intranet pero solo por ciertas personas con los permisos necesarios para poder acceder a la información.
Hoja de estilo XSL	Representa a una familia de lenguajes basados en el XML, este permite transformar el contenido de un documento XML a otros formatos que no son XML.
Interfaz gráfica	Es el medio de comunicación entre el usuario y un programa informático, de una manera visual que facilite el manejo del mismo.
Intranet	Es una red que se utiliza internamente para compartir información dentro de una compañía y se utiliza de forma privada

JavaScript	Es un lenguaje de programación utilizado en las páginas web que permite crear acciones específicas y estas son interpretadas por el navegador.
Lenguaje de bajo nivel	En programación, hace referencia al lenguaje que actúa directamente sobre el hardware. Estos están orientados a procesos.
Link	Enlace o hipervínculo.
Paper Prototyping	Prototipos de papel, se trata de establecer un prototipo desechable y de fácil crear, como los bosquejos a mano o dibujos del diseño de una interfaz con el fin de crear mejores productos.
Plug-in	Es un complemento de una aplicación que aporta una funcionalidad nueva.
Renderizado	La renderización es el proceso de interpretación de los códigos y propiedades que tiene un documento con el fin de poder ser mostrado en la pantalla de un dispositivo.
Sistema Interactivo	Es un sistema informático en el que las personas interaccionan con diversos dispositivos para realizar una o varias tareas.

Teoría cognitiva	Es una teoría de aprendizaje que describe los procesos mediante los cuales los seres humanos aprenden.
Teorías cognitivas	Se enfocan en el aprendizaje tratando de explicar los procesos internos del pensamiento que conducen a comprender o aprender un tema específico.
UNICODE	Es un estándar para la codificación, que asigna un número a las letras y otros caracteres para que estos puedan ser interpretados por la computador
Usabilidad	Es una medida de calidad que se refiere a la facilidad con la que un usuario interactúa con un sistema informático.

RESUMEN

Las interfaces gráficas surgen de la necesidad de elevar el nivel de manipulación de un ordenador o computada, es por ello que esta limitación tuvo que vencerse con el desarrollo de entornos visuales, que permitieran acceder a un ordenador sin tener que saber comandos específicos con un lenguaje a más bajo nivel para enviar instrucciones.

Con el tiempo han surgido más dispositivos que necesitan de estas interfaces y por lo tanto surgen lenguajes específicos para cada dispositivo.

A partir de esto surgen los lenguajes de marca, los cuales a través de especificar un perfil, pueden ajustar el formato de manera automática de acuerdo al tipo de dispositivo. En el lenguaje de marca se definen códigos que indican a un programa la forma de tratar su contenido, si se desea que un texto aparezca con un formato determinado, dicho texto debe ir delimitado por la correspondiente marca que indique como debe ser mostrado en pantalla o impreso. Conociendo este sistema y conociendo a la perfección el sistema de marcas de cada aplicación sería posible pasar información de un sistema a otro sin necesidad de perder el formato indicado.

La estructura que compone el lenguaje de marca es de fácil interpretación y traducción debido a que utiliza etiquetas que recursivamente son transformadas por el intérprete que los traduce en gráficos o interfaces.

OBJETIVOS

General

Desarrollar una investigación que presente nuevos paradigmas orientados al desarrollo de interfaces gráficas web multiplataforma, es decir, independientes del dispositivo o el sistema operativo en el que son desplegadas.

Específicos

1. Desarrollar un análisis comparativo de los lenguajes tradicionales para el desarrollo de aplicaciones web contra los lenguajes de marcado.

2. Exponer metodologías de desarrollo de aplicaciones web, que promuevan la optimización del tiempo de desarrollo, a través del diseño de interfaces gráficas independientes del lenguaje y del sistema operativo.

3. Realizar caso práctico que permita identificar los componentes y la estructura general utilizando el lenguaje UIML y XIML.

INTRODUCCIÓN

Ante la necesidad de comunicación directa con el computador sin ser un experto en el tema, nace la interfaz gráfica, antes de ella los programadores trabajaban en tarjetas perforadas y la salida al usuario era por medio de una impresora.

Xerox fue el creador de la primera interfaz gráfica, pero tras el poco éxito en su lanzamiento de su Xerox Star, desarrolladores como Apple y Microsoft marcaron el comienzo de lo que ahora se conoce como la interfaz gráfica de usuario.

En los años 90 nace la WEB (Word Wide Web) y con ella los sitios web, que a través de internet hacen posible la comunicación sin importar la ubicación geográfica. En una aplicación o sitio web, la interfaz de usuario es cualquier elemento gráfico que le permita interactuar con la aplicación para obtener los resultados deseados

Una interfaz no solo debe contener elementos gráficos, sino a la vez debe ser amigable, intuitiva al usuario y agradable a la vista, de otra manera el usuario puede sentirse frustrado y no desee utilizar el sistema debido a esto. Al usuario no le interesa la estructura interna de la aplicación, sino los resultados que obtiene al utilizarla. El diseño de la interfaz debe estar centrado en el usuario, existen aplicaciones que no llenan las expectativas de los usuarios por el hecho de no conocer sus necesidades.

Para poder aplicar un diseño centrado en el usuario se requieren de ciertas técnicas tales como la recolección de requerimientos al observar como el usuario se comporta con interfaces similares, identificación de oportunidades para hacer su trabajo más fácil, definición de normas que estén orientadas a la usabilidad, basándose siempre en el criterio del especialista que mide la conveniencia de las mismas.

Las pruebas son otra técnica fundamental de todo sistema y deben estar durante todo el proceso de creación, de esta manera se identifican problemas y se arreglan por anticipado a un menor costo.

Existen otros enfoques de usabilidad, que al implementarse de manera correcta pueden ser la clave para un diseño exitoso. Enfoques como el diseño participativo que consta de involucrar a los usuarios, los clientes y los empleados de otros departamentos en el proceso, para que estos cooperen en la toma de decisiones de diseño. El diseño centrado en la actividad que crea productos orientados a la actividad a la que servirán, el diseño centrado en el proceso y el diseño centrado en el uso.

La aceptación del usuario ante una nueva aplicación está directamente relacionada al formato visual y la facilidad de uso de la aplicación. Las nuevas metodologías y tecnologías para la concepción de una interfaz gráfica tratan de optimizar el tiempo que el desarrollador invierte en conseguir estos dos objetivos al diseñar aplicaciones web multiplataforma. Entre ellas se pueden mencionar tres principales: Teresa, OO-H y Cameleon.

Para poder desarrollar interfaces web genéricas se expondrán los lenguajes de marca UIML y XIML principalmente.

Los lenguajes de marca son utilizados para manipular texto, las marcas o etiquetas son códigos que se encuentran delimitadas por los signos “<” y “>”, cualquier texto que aparezca dentro de estos signos es parte del lenguaje de marcado. A través de la marca correspondiente un texto puede tener un formato específico y este podrá ser visualizado de esa forma en la pantalla del ordenador o al imprimirse físicamente.

Las principales características de los lenguajes de marcado son: texto plano, facilidad de procesamiento y su flexibilidad.

UIML (User Interface Markup Language) sirve para describir los elementos de una interfaz por lo tanto el contenido web puede ser creado sin saber específicamente en que dispositivo será visto, es un lenguaje abierto y estandarizado que puede ser implementado sin ningún costo de licenciamiento. Su principio fundamental es el Modelo Vista Controlador.

XIML (eXtensible Interface Markup Languaje) basado en XML, utiliza etiquetas para definir la gramática de lenguajes específicos. Tiene la capacidad de proveer un mecanismo estándar para las aplicaciones y herramientas con el objetivo de intercambiar datos de iteración e interactuar con la interfaz de usuario desde el diseño hasta la operación.

1. INTERFACES GRÁFICAS

Antes de presentar en detalle los nuevos métodos y tecnologías de generación de interfaces gráficas para aplicaciones web multiplataforma, es necesario revisar algunas ideas acerca del origen de este componente del software y la importancia que este ha adquirido en los sistemas de software convencionales.

1.1. Satisfacción del usuario

Satisfacer significa complacer un deseo, atender a las necesidades del usuario en un inicio demanda un mayor esfuerzo pero devuelve más de una recompensa. Es indispensable medir si los recursos o herramientas que se presentan al usuario son adecuados para cubrir sus necesidades. Un sitio web debe facilitar la forma en la que el usuario encuentra la información y los productos ofrecidos.

1.2. Inicios de la interfaz

Las interfaces gráficas surgen de la necesidad de elevar el nivel de manipulación de un ordenador o computadora, es por ello que esta limitación tuvo que vencerse con el desarrollo de entornos visuales, que permitieran acceder a un ordenador sin tener que saber comandos específicos con un lenguaje a más bajo nivel para enviar instrucciones.

En 1945 el ingeniero Vannevar Bush diseño un dispositivo llamado Memex, el cual tendría el aspecto de un escritorio con dos pantallas traslucidas, un teclado y unas palancas para el manejo de las instrucciones. La idea era crear un artefacto en el que cualquier persona pudiera guardar cualquier tipo de información y pudiera ser accedida fácilmente en otro momento. Era un accesorio para ampliar la memoria de un individuo.

La primera GUI (interfaz gráfica de usuario) como tal fue creada por Xerox en su centro de investigaciones en Palo Alto llamado PARC, pero al no poder explotar su creación, Steve Jobs, quien fue invitado a visitar Xerox PARC, fue sorprendido con la interfaz de usuario e inmediatamente supo que ese sería el futuro de las computadoras.

1.3. Definición de interfaz gráfica

Una interfaz gráfica es el medio de comunicación ante el usuario final y el sistema, un ejemplo sencillo y entendible de esto es el conjunto de botones de un televisor los cuales son manipulados por el usuario para él envió de instrucciones.

Dentro de las principales funciones que maneja una interfaz gráfica se encuentran la comunicación con otros sistemas, la manipulación de la información y el control del acceso tanto al sistema como a los datos manejados.

En una aplicación web la interfaz de usuario son los componentes con los que el usuario interactúa como los botones, las cajas de textos, los menús, etc. En general cualquier elemento gráfico que le permita interactuar con la aplicación para obtener los resultados deseados.

1.3.1. Interfaz gráfica web

Desde el momento en que aparece la web, las personas interactuaban en base a enlaces que les permitían saltar de página en página alrededor de todo el mundo web. Las páginas web hacen posible la aparición de las interfaces web.

Estas interfaces sirven de medio de comunicación entre la parte visual de la página web y el funcionamiento que hay detrás de ella, como el acceso a la información, procesamiento de datos, comportamiento de los botones, etc.

Las interfaces web presentan beneficios notorios sobre las interfaces de aplicaciones de escritorio, al ser vistas sobre un navegador son independientes del sistema operativo y cuentan con la facilidad de actualizarse y dar mantenimiento sin la necesidad de distribuir e instalar software.

1.3.2. Interfaz gráfica web: nuevos desafíos

Según un artículo realizado por Luciano Moreno (2005) indica que “El diseño de las interfaces web ha evolucionado con el tiempo hacia un esquema general perfectamente definido, ofreciendo unas interfaces bien definidas, con un conjunto de componentes gráficos y funcionales similares, que hacen posible que sea cual sea el usuario que accede a un sitio web, la comunicación entre ellos sea posible y efectiva”.

Ahora bien, el surgimiento de los múltiples dispositivos en los últimos años que acceden al internet, hace que los desarrolladores busquen alternativas para lograr que las interfaces puedan ser vistas y comprendidas de manera correcta por los usuarios y desde diferentes plataformas.

Las nuevas tecnologías crean retos en cuanto a la usabilidad y las formas de interacción con las interfaces, debido a que la interacción es mucho mayor, pero los usuarios no comprenden claramente su funcionamiento. Se introducen novedades que les pueden generar complicaciones. No son intuitivas y no siguen las convenciones actuales, por lo que no está claro qué deben hacer los usuarios para interactuar, lo que les exige un proceso de aprendizaje, que puede ser difícil para algunos.

Por ello entra en juego la parte de la automatización de las interfaces por medio de herramientas que colaboren en la construcción de páginas web enfocadas a diferentes dispositivos y que sean menos complejas que las que se manejan hasta el momento, esto ayudaría a que los diseñadores tengan más opciones en cuanto a los estándares tecnológicos y visuales así como también de puntos de usabilidad básica como contraste de texto, tamaños de fuentes y accesibilidad.

1.4. Ingeniería de la interfaz

Una característica de los sistemas interactivos es la comunicación con el usuario, la interfaz de usuario determina como un usuario percibe la aplicación, ya que al usuario no le interesa la estructura interna de la aplicación, sino que se enfoca más en cómo usarla.

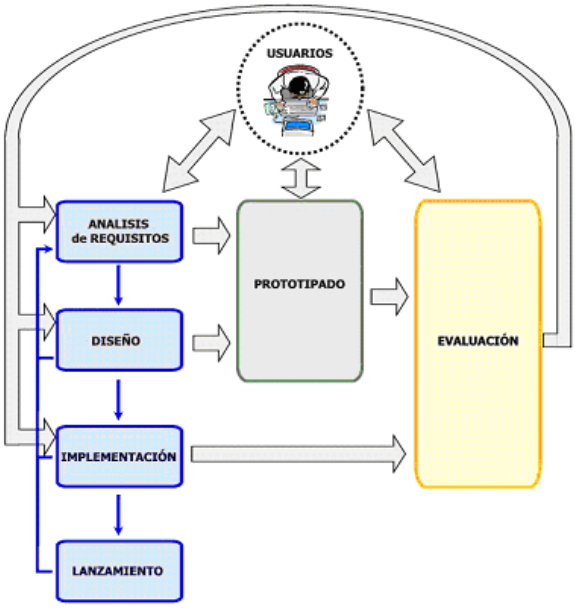
No hay que dejar de lado el aspecto visual y únicamente interesarse por la especificación, diseño y estructuración de funciones, se debe trabajar todo con la misma prioridad y tener en cuenta al usuario que va a utilizar la aplicación.

Al momento de desarrollar sistemas interactivos se pueden aplicar técnicas de ingeniería del software al modificar algunos aspectos de los métodos de diseño clásicos para adaptarlos a estos sistemas. Como por ejemplo la captura de requisitos de interacción, el análisis de las tareas, la realización de prototipos y la evaluación de los elementos creados.

1.5. Ciclo de vida de un sistema interactivo

El sistema interactivo depende de aquellas acciones realizadas por un usuario. Estas fueron desarrolladas para que cada usuario opere con una terminal.

Figura 1. **Ciclo de vida – sistema interactivo**



Fuente: CARRERAS, Olga. Metodología DCU. http://olgacarreras.blogspot.com/2008/03/metodologa-dcu-mplua.html. Consulta: marzo de 2012.

1.6. Fases

Con un sistema interactivo se logra mayor comprensión de la incidencia de los aspectos humanos en el desarrollo y aplicación de los diferentes sistemas, creando nuevos ambientes de trabajo, aprendizaje y entretenimiento basados en paradigmas de interacción emergentes.

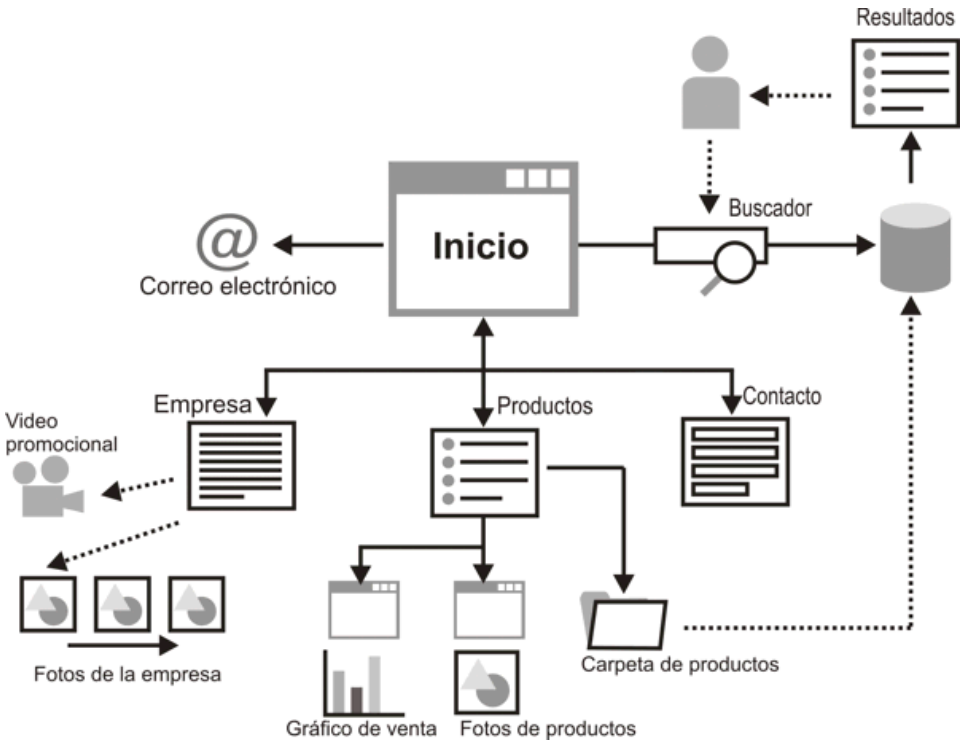
Tabla I. **Fases del ciclo de vida**

<table>
<tr><td>Análisis de requerimientos
Son definidas como las limitaciones que cubrirá la interfaz, dependiendo de qué tipo de diseño sea realizado.</td><td rowspan="2">Prototipado
Son los diferentes esquemas realizados antes de llegar a una versión final que llene con los requerimientos y expectativas del usuario.</td><td rowspan="3">Evaluación
Constantemente se evalúa para determinar los alcances que se han obtenido dependiendo de la respuesta y la visión que el usuario haya abstraído.</td></tr>
<tr><td>Diseño
Se empieza la construcción de la interfaz que será manipulada por el usuario con las necesidades específicas de este.</td></tr>
<tr><td>Implementación
Se emplea el diseño realizado, este sistema debe manejar amigabilidad para llegar a la usabilidad.</td><td></td></tr>
<tr><td>Lanzamiento
Puesta en producción para ser monitoreado y obtener la respuesta ante los usuarios de la automatización y facilidad de la interfaz.</td><td colspan="2"></td></tr>
</table>

Fuente: elaboración propia.

Según un artículo publicado por Yusef Montero y Sergio Santamaría (s.f.) indica que “Un sistema interactivo debe cumplir con metodologías y técnicas de desarrollo de interfaces centradas en los usuarios donde estén integradas teorías cognitivas que colaboren en el desarrollo del diseño de la interfaz. Debe englobar un conjunto heterogéneo de metodologías y técnicas que comparten un objetivo común: conocer y comprender las necesidades, limitaciones, comportamiento y características del usuario, involucrando en muchos casos a usuarios potenciales o reales en el proceso”.

Figura 2. **Esquema de un sistema interactivo**



Fuente: RONDA, Rodrigo. La diagramación en la arquitectura de información. 2007. http://www.nosolousabilidad.com/articulos/diagramacion.htm. Consulta: diciembre de 2013.

Se recomienda que aunque el desarrollador tenga conocimiento sobre usabilidad o en la creación de un diseño, evaluar el mismo con usuarios. Debido a que, conforme más tiempo está involucrado en el proyecto, menor es su perspectiva y será más difícil que pueda detectar posibles problemas.

El diseño y desarrollo de interfaces web de estos sistemas interactivos es algo que cualquier programador debería conocer y explotar, ya que la implementación de herramientas de estilo y estructura, permiten alcanzar un mejor rendimiento de la interfaz web y una mejorar visualización de la misma en los diferentes dispositivos.

1.7. Diseño incremental centrado en el usuario

Al momento de diseñar se debe estar enfocado directamente en el usuario para tomar sus necesidades y mejorar su experiencia, con el fin de que el sistema permita a dicho usuario conseguir su objetivo concreto, en el dominio de la aplicación.

Las métricas y metodologías clásicas de evaluación en el diseño enfocado al usuario, permiten facilitar la detección de problemas de usabilidad, pero por su carácter cualitativo difícilmente pueden servir para obtener una aproximación fiel sobre la usabilidad global de la interfaz.

Un diseño puede medirse a través de:

- Facilidad de uso
- Facilidad de aprendizaje
- Satisfacción de un sistema.

El objetivo del diseño visual no es solo impactar sino ayudar al navegante a tener orientación en todo momento sin perder tiempo saltando de página en página.

1.8. Utilidad del diseño centrado en el usuario

La aplicación del enfoque de diseño centrado en el usuario ha demostrado que existen un conjunto de ventajas importantes que deben tomarse en cuenta al momento de escoger una metodología de diseño de interfaz gráfica. Estas ventajas son revisadas a continuación.

1.8.1. Ventajas sobre la competencia

Al hacer los productos basados en técnicas de diseño centrado en el usuario, se asegura el estar un paso adelante de la competencia. Cuando un sitio web no satisface las necesidades del usuario, el usuario simplemente descartará el sitio y buscará sitios que si lo hagan.

Realizar un diseño enfocado en satisfacer al usuario aumentará la calidad percibida, y reducirá los rediseños tras las pruebas con usuarios finales, tanto en preproducción como en posproducción.

Si el usuario se siente a gusto con el sitio web, volverá a visitarlo y podría recomendarlo a otras personas, lo cual es catalogado como una publicidad que no tiene un costo extra para la organización.

1.8.2. Ser directo y específico con el usuario

Se debe crear un sitio que sea intuitivo y amigable para el usuario, reduciendo al mínimo la complejidad de la interfaz y así evitar que el usuario prefiera ir a otra parte. Hoy en día se encuentran muchos sitios web que ahuyentan a los usuarios por el hecho de no conocer sus necesidades debido a que no se obtuvo una comunicación directa de lo que se requiere.

Según el artículo publicado por Meher, Jessica (2012), llamado *12 Critical Elements Every Homepage Must Have* se habla de los 12 elementos que la página de inicio de un sitio web debe tener para poder generar clientes potenciales, los cuales se describen a continuación:

1. El título: al ingresar a una página en 3 segundos el usuario debe de saber qué es lo el sitio web ofrece.
2. El subtítulo: es una breve descripción de que el sitio web ofrece.
3. Los beneficios: se deben listar las ventajas y los beneficios del producto o servicio que se ofrece en el sitio web. No mostrar solo que se hace sino también que le da valor.
4. *Call to action*: colocar botones o *links* llamativos que le indiquen al usuario que debe hacer para conseguir el producto o servicio de una manera rápida y concisa.
5. Características: listar las características del producto. Esto complementa los beneficios listados y el usuario comprende de mejor manera que se ofrece en el sitio web.
6. Testimonios de los clientes: los testimonios y experiencias de clientes satisfechos es una herramienta poderosa para generar confianza sobre el sitio.

7. Casos de éxito: también se deben incluir los reconocimientos de la empresa obtenidos para generar una buena primera impresión.
8. Navegación: es necesario que el usuario tenga una estructura clara del sitio para que pueda navegar fácil y rápidamente.
9. Soporte de imágenes: esto ayuda a crear un impacto visual. Las imágenes o videos deben estar acorde al servicio que se ofrece. Tomar en cuenta no estar sobrecargar el sitio.
10. Ofrecer contenidos: si es posible ofrecer la descarga de catálogos, libros, guías, etc. Para darle un valor agregado al sitio.
11. recursos: la mayoría de los visitantes no están listos para comprar. Es necesario ofrecer un enlace a un centro de recursos en donde puedan aprender más acerca del servicio o producto.
12. *Call-to-action* secundarias: al momento de que el visitante llega al pie de la página debe existir un botón que incentive la adquisición del producto o servicio nuevamente.

1.8.3. Reducción de costos

Se reducirán los costos con esta estrategia debido a que se disminuyen los módulos o tareas, ya que se minimiza el número de operaciones que el usuario debe realizar, lo cual implica menos código escrito por los desarrolladores y hará al sitio más accesible y manejable.

1.9. Aplicación del diseño centrado en el usuario

Para la aplicación del diseño centrado en el usuario como primer punto se debe escoger el ambiente en donde se va a realizar la página web, luego se deben implementar diferente técnicas que tendrán como objetivo colocar al usuario como el centro y el rol más importante dentro del diseño de la página web.

1.9.1. Recolección de requerimientos

En esta parte se trata de investigar y analizar las necesidades de los usuarios, puede ser a través de una entrevista con el usuario u observando el comportamiento e interacción que tiene con interfaces similares, de esta manera se asegura que los requerimientos son los requisitos importantes para el usuario, y se identifican oportunidades al enfocarse facilitar el trabajo del usuario.

1.9.2. Normas

Existen diferentes heurísticas y resultados empíricos para crear un diseño centrado en el usuario que cumpla con los requerimientos y especificaciones deseadas. En cada método utilizado se debe analizar el impacto que tiene sobre el proyecto y con esto se decidirá si se utilizara o no.

1.9.3. Diseño

Técnicas como el *paper prototyping* ayudan a obtener una retroalimentación por parte de los usuarios sobre la interfaz, antes que se empiece a desarrollar la funcionalidad, de esta manera los cambios se manejan de una manera más flexible, ya que no es necesario volver a desarrollar toda la funcionalidad ante un cambio en la interfaz.

1.9.4. Testeo

La realización de pruebas se realiza desde el inicio de la creación de la interfaz para que continuamente se identifiquen los problemas por anticipado y sea arreglado de manera fácil y con menor costo.

1.10. Otros enfoques de usabilidad

Además de los enfoques tradicionales que son las más comúnmente utilizados, hay un conjunto de enfoques más flexibles que siendo bien implementados pueden ser la clave de un diseño exitoso.

1.10.1. Diseño participativo

El diseño participativo no solamente puede ser utilizado para un proyecto de software, es un proceso que se enfoca en involucrar a todos los interesados como los usuarios, empleados, socios, etc. En todo el proceso del diseño donde los usuarios no solo son fuentes de información, sino que participan activamente en cuanto a las decisiones acerca del diseño en cuestión.

1.10.2. Diseño centrado en la actividad

Una actividad es un conjunto de tareas, acciones y operaciones que se realizan con un fin en común. Este tipo de diseño plantea tomar a la actividad para la cual se plasma el producto, como el centro en la toma de decisiones del diseño y no tanto a los usuarios finales.

1.10.3. Diseño centrado en el proceso

Se basa en seguir una serie de técnicas y modelos que reducen las iteraciones y por lo tanto las pruebas con los usuarios. La utilización de una metodología RUP es un ejemplo de esto ya que recurre a técnicas como casos de uso o diagramas para la creación del proyecto. Tiene una estructura definida y la interacción con el usuario es prácticamente solo al inicio del proyecto.

1.10.4. Diseño centrado en el uso

Propone que el diseño se encuentre dirigido al uso que el usuario hace con el sistema y los objetivos sobre el sistema. El propósito es mejorar las herramientas para el cumplimiento de las tareas, los usuarios son solo actores del sistema.

1.11. Hacia nuevos horizontes en la creación de interfaces multiplataforma

El nivel de aceptación que un usuario presenta ante una nueva aplicación está directamente relacionado al formato visual y la facilidad de uso de la aplicación. Ambos aspectos son relativos al diseño de la interfaz gráfica. Las nuevas metodologías y tecnologías de generación de interfaz gráfica buscan optimizar el tiempo que el desarrollador invierte en conseguir estos dos objetivos al diseñar aplicaciones web multiplataforma.

1.11.1. Propuestas metodológicas: Teresa

Se basa en tres niveles de abstracción que permiten al diseñador enfocarse en aspectos lógicos de importancia, dejando a un lado los pequeños detalles. Es capaz de obtener interfaces desde abstracciones, considerando las plataformas disponibles y su interacción mientras sea posible. Para apoyar las transformaciones se aporta la herramienta.

El proceso de esta metodología está estructurado en tres fases principales:

En la fase de inicio los diseñadores desarrollan un solo modelo, que trata los contextos posibles del uso y varias plataformas implicadas, incluyendo un modelo del dominio para identificar todos los objetos que tienen que ser manipulados para realizar tareas y las relaciones entre tales objetos.

Lo que se logra es proporcionar una descripción de las tareas apoyadas por el uso multiplataforma. Que cada tarea, indique qué plataformas pueden soportarlo y qué dependencias entre las tareas que se pueden realizar a través de diversas plataformas.

En la siguiente fase se realiza una transformación de modelo anterior, aquí los diseñadores deben adaptarse al modelo según la plataforma objetivo y, en caso de necesidad, refinar el modelo para adaptarlo a la especificidad de la plataforma elegida. Después de esto, los diseñadores transformarán el modelo original en un interfaz de usuario abstracto. En la última fase, se procede con la generación de código.

1.11.2. Propuestas metodológicas: OO-H

Método que se diseñó con el objetivo de poder crear aplicaciones web basándose en tres modelos principales: el Diagrama de Acceso Navegacional (DAN), el Diagrama de Presentación Abstracta (DPA) y el Diagrama de Diseño Visual (DDV).

El diagrama de acceso navegacional se crea usando cuatro constructores básicos:

- Agrupa constructores facilitando el diseño, de forma análoga a cómo funcionan los paquetes en UML.
- Clase navegacional: son vistas de las clases identificadas en el modelo de dominio. A través de estas vistas se puede especificar la visibilidad de los distintos atributos de las clases.

- Enlace navegacional: permiten describir los caminos que puede seguir el usuario para navegar a través de las vistas creadas con las clases navegacionales.
- Colección: es un agrupamiento de enlaces navegacionales (por ejemplo para crear menús).

1.11.3. Propuestas metodológicas: Cameleon

Construye métodos y entornos de apoyo para el diseño y desarrollo de sistemas informáticos interactivos multicontexto.

Usa un nivel alto de abstracción que permite que la definición de la interfaz de usuario abstracta no dependa tanto de la plataforma final, en comparación a la modalidad usada para interaccionar con la interfaz de usuario.

La interfaz de usuario como resultado final da a conocer el código que será ejecutado en la plataforma destino para mostrar la interfaz de usuario. Por lo tanto, la interfaz de usuario depende tanto de la plataforma que estará desplegada como la interfaz de la modalidad que será usada para interactuar con ella.

2. LENGUAJES Y NOTACIONES

A continuación se presentan los lenguajes y notaciones utilizados para la creación de interfaces web genéricas.

2.1. Lenguaje de marcado

Un lenguaje de marca es un lenguaje utilizado para manipular texto, las marcas o etiquetas son códigos que se encuentran delimitadas por los signos “<” y “>”, cualquier texto que aparezca dentro de estos signos es parte del lenguaje de marcado. A través de la marca correspondiente un texto pude tener un formato específico y este podrá ser visualizado de esa forma en la pantalla del ordenador o al imprimirse físicamente.

El lenguaje de marcas generalizado GML (Generalized Markup Language), fue desarrollado principalmente para definir la estructura de un documento con el objetivo de ser visualizado en diferentes plataformas sin necesidad de cambiar el documento por cada plataforma, al definir un perfil para el dispositivo específico, el documento se puede ajustar automáticamente.

2.1.1. Características de los lenguajes de marcas

- Texto plano, una de las principales ventajas del lenguaje de marcas es que se interpreta de forma directa al tratarse de archivos de texto plano. Esto es una ventaja evidente respecto a los sistemas de archivos binarios, que hace uso de un programa intermediario para trabajar con ellos que lo interpreta.

- Facilidad de procesamiento, la estructura que compone el lenguaje de marca esta fácil de interpretar y traducir debido a que utiliza etiquetas que recursivamente son transformadas por el intérprete que los traduce en gráficos o interfaces.

- Flexibilidad, aunque en un principio los lenguajes de marcas se crearon para documentos de texto, se han empezado a utilizar en áreas como gráficos vectoriales, interfaces y servicios web.

Históricamente, el marcado se usaba y se usa en la industria editorial y de la comunicación, así como entre autores, editores e impresores.

2.1.2. Un nuevo panorama de la estructuración de documentos

Nos encontramos ante un lenguaje de naturaleza artificial cuya meta es la descripción de información para facilitar su proceso; de donde surge el tema de qué es lo que se describe y para qué, produciéndose así un efecto de retroalimentación sobre el lenguaje.

Las diferencias más significativas entre esquemas normales y el lenguaje de marcas son las siguientes:

- Los esquemas son documentos XML.
- Los esquemas son más potentes. Es posible describir con mucho más detalle un documento XML.
- No es posible describir entidades utilizando esquemas.

2.1.3. Partes de un esquema

Además de seguir las normas de los documentos XML para estar correctamente formados, hay varios elementos en un esquema que se pueden diferenciar:

- La declaración del esquema
- Los elementos
- Los tipos intrínsecos
- Los tipos definidos por los usuarios
- La lista de atributos

2.1.3.1. Elementos

El uso de esquemas permite estructurar perfectamente la definición del documento. En primer lugar se pueden definir los elementos que componen nuestro documento.

2.2. UIML

UIML (User Interface Markup Language) fue creado por en 1997 por Harmonia Inc. Es tipo de lenguaje de marcado derivado de XML que sirve para describir los elementos de una interfaz por lo tanto el contenido web puede ser creado sin saber específicamente en que dispositivo será visto. La idea es crear un archivo que represente toda la interfaz de usuario por medio de etiquetas.

Básicamente UIML trata de responder a las siguientes preguntas:

- ¿Cuáles son las partes que componen a la interfaz de usuario?
- ¿Cuál es la presentación que se utiliza para las partes?
- ¿Cuál es el contenido (imágenes, textos, sonidos) que se utiliza en la interfaz de usuario?
- ¿Cuál es el comportamiento de la interfaz?
- ¿Cuáles son las partes que se mapearan?

El objetivo principal de UIML es poder crear y diseñar interfaces de usuario para cualquier tipo de dispositivo sin necesidad de aprender un lenguaje específico para cada dispositivo, pero con la opción de poder mapearlo a diferentes lenguajes existentes.

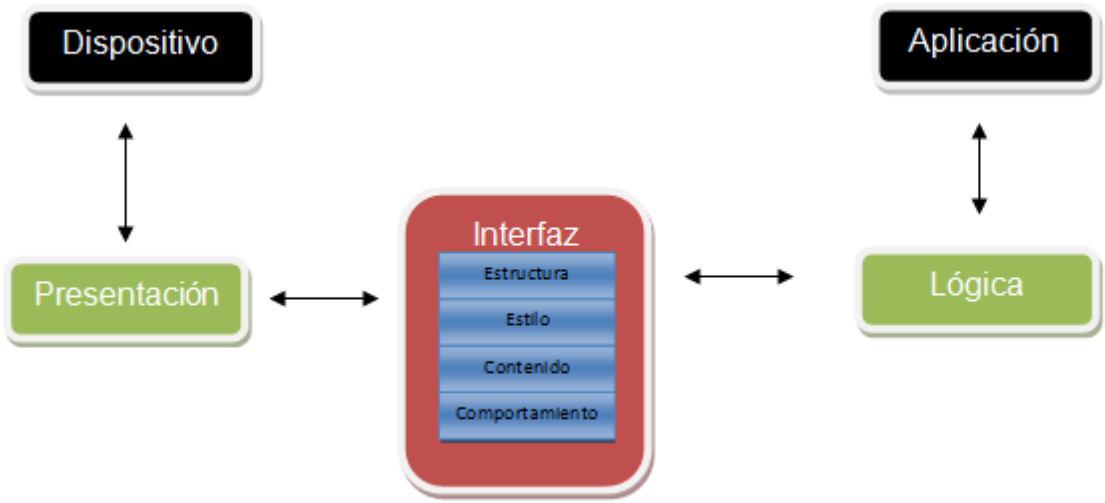
Debido a que muchos programadores saben poco sobre Interacción Humano Computadora y ante el éxito de HTML con su sistema de etiquetas sencillo y fácil de aprender, nace la idea de crear un lenguaje que pueda generar cualquier interfaz que un lenguaje de programación también.

UIML es un lenguaje abierto y estandarizado que puede ser implementado sin ningún costo de licenciamiento.

2.2.1. Modelo de UIML

Su principio fundamental es el modelo MVC (Modelo Vista Controlador) el cual separa los datos, la interfaz de usuario y la lógica.

Figura 3. **Modelo de UIML**



Fuente: elaboración propia.

En el modelo UIML existen seis maneras de separar la descripción de la interfaz de usuario, mientras que en el modelo vista controlador solo hay tres.

2.2.2. Características

Las características principales de UIML son:

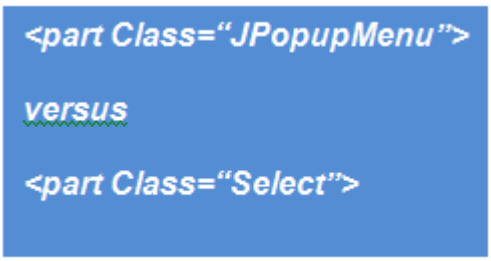
- Los programados pueden describir de forma separada la estructura, contenido y comportamiento de la interfaz de usuario.

- Describe el comportamiento de ejecución en términos abstractos. (eventos dependientes de la plataforma y tareas genéricas)
- Permite un diseño basado en componentes dividiendo la descripción de la interfaz en una o más plantillas reusables.
- No define un kit de etiquetas específicas. (<Menú>,...)
- Utiliza unas pocas etiquetas de gran alcance. (<part>, <property>, ...)
- Se debe añadir el vocabulario de herramientas para que sea útil.
- No es necesario cambiar UIML con los nuevos dispositivos inventados.

El vocabulario se complementa con un vocabulario de las partes de la interfaz de usuario, propiedades y eventos definidos fuera de esta especificación. UIML permite el desarrollo continuo.

Por un lado ofrece una mejor experiencia para el usuario ya que usa el vocabulario especifico del dispositivo, y por otro ayuda a los programadores a reducir el tiempo de desarrollo.

Figura 4. **Uso de vocabulario específico contra el uso de vocabulario genérico**



Fuente: elaboración propia.

2.2.3. Elementos de UIML

Un elemento <behavior> en UIML es el que describe las acciones que ocurren cuando un usuario se comunica con la interfaz gráfica.

El elemento <behavior> se basa en reglas específicas de los lenguajes. Cada regla contiene una condición y una secuencia de acciones. Siempre que una acción es verdadera las acciones asociadas son ejecutadas.

En la última versión de UIML (4.0) cada condición se evalúa solo cuando existe un evento asociado con la condición. Esto simplifica el mapeo de UIML a otros lenguajes.

Cada acción puede realizar una de las siguientes opciones:

1. Cambiar una propiedad o variable de una parte de la interfaz de usuario.
2. Llamar a una función de un lenguaje de script.
3. Llamar a una función o método de un Software
4. Ignorar a un evento.

En las opciones (2) y (3) UIML brinda una sintaxis para describir cada llamada, pero no especifica una aplicación de cómo se realiza la llamada.

UIML viene siendo un metalenguaje utilizado para hablar acerca de otro lenguaje, al lenguaje acerca del cual se está hablando se le denomina el lenguaje objeto.

El propio XML no contiene etiquetas específicas para un propósito en particular como en HTML (<Table> o <H1>). La ventaja que brindan los lenguajes de marcado es que puede ser estandarizado una vez y extendido a través de las estructuras periféricas.

UIML no contiene etiquetas especificas relacionadas con un determinado conjunto de herramientas de una interfaz de usuario como por ejemplo <BUTTON>, <WINDOW> o <MENU>, en cambio captura los elementos que son comunes a cualquier interfaz por medio de un conjunto de etiquetas genéricas.

La sintaxis de UIML también define etiquetas de atributos que mapean estos elementos a un conjunto de herramientas en particular. Sin embargo, las abstracciones definidas en un conjunto de herramientas en particular, como por ejemplo una ventana, no son parte de UIML. De esta manera UIML solo necesita ser estandarizada una vez, y diferentes grupos de usuarios finales pueden definir vocabularios que son adecuados para diversas herramientas de forma independiente de UIML.

Por lo tanto, un creador de una interfaz en UIML no solo necesita conocer el lenguaje UIML, sino también debe conocer cada conjunto de herramientas de interfaz de usuario de los lenguajes a los que desea mapear UIML por ejemplo, Java Swing, Microsoft Foundation Classes, WML.

2.2.4. Estructura de un documento UIML

Está compuesto de 2 partes:

- La primera parte indica la versión del lenguaje XML y la ubicación del documento de definición de UIML.
- La segunda parte es la etiqueta <uiml> la cual es la raíz en los documentos UIML.

Figura 5. **Encabezado del documento**

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC
  "-//OASIS//DTD UIML 4.0 Draft//EN" http://uiml.org/dtds/UIML4_0a.dtd">
```

Fuente: elaboración propia.

2.3. XIML

XIML (eXtensible Interface Markup Language) es una propuesta para representar la interacción de los datos, soporta el diseño, operación, organización y funciones de evaluación, también es capaz de relacionar lo abstracto y no abstracto de los elementos de datos de una interfaz.

La industria del software está haciendo un esfuerzo importante para establecer las bases para un nuevo modelo de computación que permita una forma estándar interoperen e intercambien información. Este modelo principalmente está basado para aplicaciones de internet pero sin embargo, será capaz de integrarse a entornos futuros de trabajo.

2.3.1. Relación

Una relación en XIML es una definición o declaración que une dos o más elementos XIML ya sea dentro de un componente o entre componentes. Mediante la captura de las relaciones de una manera explícita, XIML crea un conjunto de conocimientos que pueden ayudar en el diseño, operación y evaluación de interfaces de usuario.

2.3.2. Atributos

Los atributos son las características o propiedades de los elementos a los cuales se les puede asignar un valor. El valor del atributo puede ser un valor de los tipos de datos básicos o una instancia de otro elemento existente.

2.3.3. Interfaces multiplataforma

Una de las características principales de XIML es que puede ser utilizado para desarrollar interfaces de usuario que puedan ser visualizadas en una variedad de dispositivos. En XIML se puede utilizar una definición de interfaz única en cualquier número de dispositivos destino. Esto es posible gracias a la separación que hace XIML entre la definición de una interfaz de usuario y la renderizado que se hace sobre la interfaz ya que la definición de la interfaz es en si la especificación de XIML y el dispositivo es el encargado del renderizado.

2.4. XUL

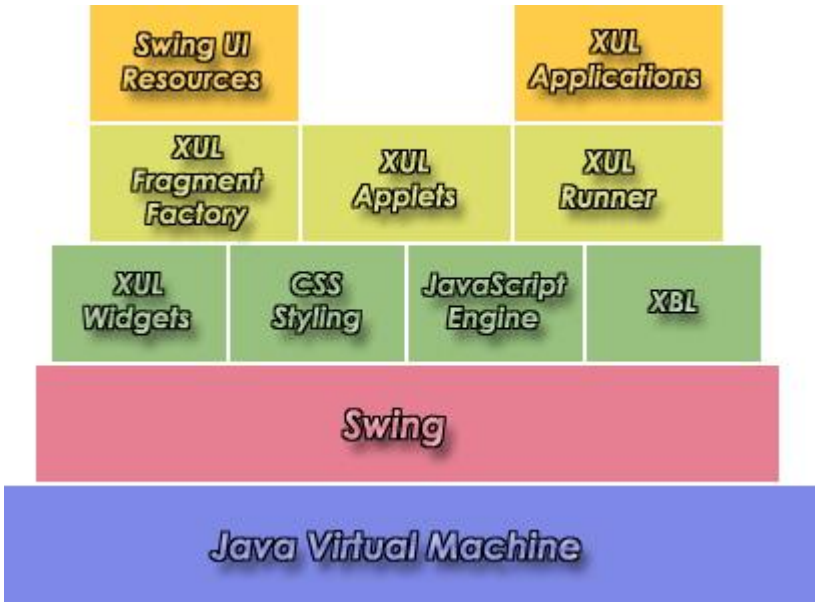
Basado en XML para la interfaz de usuario. Dentro de las ventajas que aporta se encuentra la definición de interfaces GUI simple y portable. Esto reduce el esfuerzo empleado en el desarrollo de software.

Se utiliza para la definición de interfaces de usuario, su diseño se enfoca en la portabilidad y el desarrollo de aplicaciones multiplataforma sofisticadas o complejas sin necesidad de herramientas especiales.

Existe un proyecto en plataforma java para integrar el lenguaje XUL llamado jXUL. Hay varias formas en que esta integración se llevó a cabo. De hecho, este proyecto se amplió debido a que los miembros encontraran nuevas formas de integrar estas dos tecnologías.

El siguiente diagrama muestra el objetivo inicial de la arquitectura jXul. La capa superior es el resultado final.

Figura 6. **Integración de XUL con Java**



Fuente: BULLARD, Vaughn, SMITH, Kevin y C, Michael. jXLU. [en línea] 2008. http://jxul.sourceforge.net. Consulta: marzo de 2012.

2.5. HTML

Hyper Text Markup Language. Utilizado como un lenguaje de marca para la elaboración de páginas web. Su traducción se define como “Lenguaje de Marcas de Hipertexto”, esta definición es debido a que está conformado por etiquetas que definen la estructura y el formato del documento que observará el usuario como producto final.

Las etiquetas que son utilizadas por el lenguaje para la generación de las páginas Web son leídas e interpretadas por el navegador o visualizador, es decir el programa que es utilizado para navegar, y que es el que ejecuta las funciones definidas con el lenguaje HTML permitiendo que puedan ser visibles en los dispositivos que se estén utilizando.

En cuanto a las características del lenguaje cabe mencionar que se está trabajando con un archivo de texto plano (que es el documento en sí), permitiendo crear enlaces entre distintas secciones del documento actual o bien entre diferentes fuentes de información a través de hiperenlaces o más común mente llamados hipervínculos e incluso insertar otra clase de objetos como lo son las imágenes y sonidos.

Es muy importante mencionar que las páginas Web elaboradas completamente en HTML son aplicaciones de texto, las cuales poseen la ventaja de ocupar poco espacio, ser rápidas y la mayoría tienen mucho desarrollo.

En contraparte se puede decir que la ventaja que poseen puede llegar a ser una desventaja en el sentido de que las páginas por ser completamente texto son poco amigables por lo que tienen interfaces muy restringidas, son ideales para tareas administrativas, terminales con enlaces lentos y software en general para computadoras con poca capacidad.

En la actualidad existen una gran cantidad de navegadores y asimismo una gran cantidad de versiones de los mismos, por lo que unos de los inconvenientes que se presentan es que no todos pueden interpretar el código HTML de la misma manera, es por esta razón que las personas que se dedican al desarrollo de páginas Web, deben verificar que las páginas desarrolladas puedan ser leídas por lo menos por los navegadores más comúnmente utilizados.

2.6. Dinamyc HTML

DHTML o HTML dinámico se le denomina a las páginas Web que hacen uso del lenguaje HTML para el diseño de páginas Web, pero poseen la característica de incorporar junto al lenguaje estático HTML lenguajes tales como los interpretados del lado del cliente como lo es JavaScript o bien lenguajes de hojas de estilo en cascada mejor conocidos como CSS y la jerarquía de objetos de un DOM.

La utilidad que se le da a este tipo de lenguaje va enfocado a la creación de menús desplegables, objetos dinámicos, botones con desplazamiento de texto, textos explicativos que aparecen al situar el cursor sobre ciertas palabras, entre otros.

Dentro de las ventajas que esta tecnología poseen es continuar con el tamaño de los archivos, es aceptado por los fabricantes de navegadores más importantes, algo muy importante es que no es necesario instalar los famosos *plug-ins*, no requiere una Java Virtual Machine (JVM), se pueden hacer contenidos dinámicos, posicionamiento de elementos y animaciones y tipos de letras dinámicos.

Dentro de sus desventajas se puede mencionar que Netscape y Microsoft tienen implementaciones de DHTML diferentes, la creación de DHTML es mucho más compleja que el HTML y el código fuente es desprotegido y visible.

2.7. WML

Wireless Markup Language o WML es un lenguaje de programación soportado en XML. Posee aspectos reducidos del lenguaje HTML que facilita la conexión a Internet de dichos dispositivos y que logra que las páginas web en dispositivos inalámbricos puedan utilizar la tecnología WAP.

La vista que resulte de la construcción y el diseño de una página depende del dispositivo que se utilice y de la forma en que este interprete el código. WML también visto como metalenguaje, establece que además de usar etiquetas predefinidas se puedan crear componentes propios y tiene ciertas similitudes con otro lenguaje de etiquetas bastante conocido, como lo es HTML, el cual es utilizado para la creación de páginas web convencionales.

Acá también se hace uso de un lenguaje de script como JavaScript para obtener dinamismo en los documentos, WML dispone del WMLS que es un lenguaje similar al JavaScript, pero con alguna diferencia fundamental.

Si existe un error en la sintaxis de las etiquetas devolverá un error en vez de mostrar la página; al provenir del XML requiere que las etiquetas como
 (que sirven para empezar una nueva línea) finalicen con />.

También permite el uso de variables en sus etiquetas, algo que no es posible en HTML, esta función es útil ya que el valor de las variables se puede mantener entre tarjeta, utilizado para anchos de banda reducidos, cada página creada está compuesta de una serie de tarjetas y cada una de estas se muestra como una página individual.

2.8. Java

Utilizado para la creación de sitios web dinámicos, su orientación se encuentra dirigida a páginas web en Java como JSP (Java Server Pages). JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET. Posee un motor de páginas basado en los servlets de Java.

2.8.1. Características

A continuación se presentas las características principales del lenguaje Java

- Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas web.
- Los archivos se encuentran con la extensión (jsp).
- El código JSP puede ser incrustado en código HTML.

2.8.2. Ventajas

Entre las principales ventajas que Java posee se pueden mencionar las siguientes:

- Ejecución rápida del servlets.
- Crear páginas del lado del servidor.
- Multiplataforma.
- Código bien estructurado.
- Integridad con los módulos de Java.
- La parte dinámica está escrita en Java.
- Permite la utilización se servlets.

2.8.3. Desventajas

Como todo lenguaje Java posee desventajas, las cuales no son tan importantes como para desvalorar este lenguaje tan poderoso. A continuación se listan las desventajas principales:

- Complejidad de aprendizaje.
- Evoluciona muy lentamente
- Sintaxis engorrosa

2.9. Análisis comparativo

En la siguiente tabla se muestra un análisis comparativo entre los lenguajes de marco o de etiqueta y los lenguajes convencionales.

Tabla II. **Análisis comparativo**

LENGUAJE DE ETIQUETA	**LENGUAJE NORMAL**
Sigue un estándar.	No existe declaración de variables.
Es un lenguaje independiente de la plataforma sobre la que se trabaje, lo que garantiza un grado de reutilización muy alto haciendo los desarrollos más sencillos, rápidos y baratos. Reduce gastos, ya que las operaciones que permite hacer sobre la Web serían muy costosas en otros lenguajes.	No hay funciones o llamadas a procedimientos para la realización de alguna operación.
Posibilita la interacción con otras aplicaciones y bases de datos de manera sencilla, por lo que hace más eficiente el manejo de datos e información.	No es necesaria alguna instancia para las llamadas a otras clases o la creación de objetos.

Continuación de la tabla II.

Al ser UNICODE puede ser utilizado en múltiples lenguajes.	No todos los lenguajes son multiplataforma.
Avanzadas funcionalidades en las aplicaciones Internet, Intranets y Extranets.	Se requiere automatización para la implementación o realización de funcionalidades internas al desarrollo de la aplicación
Permite un nivel muy elevado de automatización de aplicaciones.	
El elevado grado de reutilización de sus componentes los hace más baratos que los tradicionales.	

Fuente: elaboración propia.

3. IMPLEMENTACIÓN

3.1. Implementación de una interfaz gráfica utilizando lenguajes de marcado

En el siguiente capítulo se presenta la implementación de una interfaz de un diccionario utilizando los lenguajes de marca UIML y XIML y las diferencias entre estos lenguajes de marca.

3.1.1. Inicios de UIML

Es un lenguaje de desarrollo de interfaces:

- Independientes del dispositivo
- Que puede ser mapeadas a cualquier lenguaje
- Que se ejecutan sistema operativo

UIML es un lenguaje declarativo, compatible con los metalenguajes para describir interfaces de usuarios.

UIML abarca múltiples mundos, tales como:

- Interfaces Web como HTML, WML
- Interfaces de escritorio como JAVA, Visual basic, C++
- Interfaces para dispositivos portátiles como WAP Phones
- Interfaces de usuario de voz

UIML provee un lenguaje genérico para las interfaces de usuario como por ejemplo java un botón se declara: JButton boton = new JButton(“Aceptar”);

En HTML la sintaxis para declarar este mismo botón se realizaría de la siguiente manera: <INPUT TYPE= “BUTTON” NAME=boton” VALUE=“Aceptar”>

Mientras que en UIML solo se utiliza una sintaxis para cada lenguaje de interfaz: <part name=“boton” class=“Button”/> <property name=“content”>Aceptar</property>

3.1.2. Estructura

Utilizando componentes de AWT y SWING de Java se describirá una pequeña interfaz usando UIML para Java.

Figura 7. **Estructura general de un documento UIML**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>...</structure>
<style>...</style>
<content>...</content>
<behavior>...</behavior>
</interface>
<peers>
<logic>...</logic>
<presentation>...</presentation>
</peers>
</uiml>
```

Fuente: elaboración propia.

3.1.3. Componentes

Las estructura de la interfaz de usuario se define dentro de las etiquetas “<structure>...</structure>” y las partes de la estructura dentro de las etiquetas “<part>...</part>” como se muestra en la siguiente imagen:

Figura 8. **Componentes**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>
<part id="Frame" class="JFrame">
<part id="Label" class="JLabel"/>
<part id="Button" class="JButton"/>
</part>
</structure>
<style>...</style>
<content>...</content>
<behavior>...</behavior>
</interface>
<peers>
<logic>...</logic>
<presentation>...</presentation>
</peers>
</uiml>
```

Fuente: elaboración propia.

3.1.4. Estilo

Las propiedades como el color, el nombre, tamaño de fuente entre otros, de cada parte de la estructura se especifican dentro de las etiquetas“<style>...</style>” como se muestra en la siguiente imagen:

Figura 9. **Estilo**

```
<?xml version="1.0">
    <uiml>
    <interface>
    <structure>
    <part id="MyFrame" class="JFrame">...</part>
    </structure>
    <style>
    <property part-name="MyFrame
    name=”title">UIML Example
    </property>
    </style>
    <content>...</content>
    <behavior>...</behavior>
    </interface>
    <peers>
    <logic>...</logic>
    <presentation>...</presentation>
    </peers>
    </uiml>
```

Fuente: elaboración propia.

3.1.5. Contenido

El contenido para cada parte se detalla dentro de las etiquetas “<content>...</content>” y con el elemento <reference/> se especifica el contenido que pertenece a esa parte. Como se muestra a continuación:

Figura 10. **Contenido**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>...</structure>
<style>
<property part-name="MyFrame" name="title">
<reference constant-name="titleText"/>
</property>
</style>
<content id="Spanish">
<constant id="titleText">Ejemplo de UIML</constant>
</content>
<content id="English">
<constant id="titleText">UIML Example</constant>
</content>
<behavior>...</behavior>
</interface>
<peers>...</peers>
</uiml>
```

Fuente: elaboración propia.

3.1.6. Comportamiento

El comportamiento de las partes se coloca dentro de las etiquetas “<behavior>...</behavior> “ aquí se describira toda la interaccion logica de la interfaz.

Figura 11. **Comportamiento**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>
<part id="MyFrame" class="JFrame">
<part id="MyLabel" class="JLabel"/>
<part id="MyButton" class="JButton"/>
</part>
</structure>
<behavior>
<rule>
<condition>
<event class="actionPerformed" part-name="Button"/>
</condition>
<action>
<property part-name="MyLabel" name="text">Button pressed
</property>
</action>
</rule>
</behavior>
</interface>
<peers>...</peers>
</uiml>
```

Fuente: elaboración propia.

3.1.7. Lógica

La manera en la que se realizara la comunicación con el mundo exterior (lógica del negocio, las fuentes de datos, interfaz de usuario) se detalla dentro de las etiquetas “<logic>...</logic>”

Figura 12. **Lógica**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>...</structure>
<style>...</style>
<content>...</content>
<behavior>...</behavior>
</interface>
<peers>
<logic>
<d-component id="Counter" maps-to="org.something.SimpleCounter">
<d-method id="increment" return-type="int" maps-to="count"/>
</d-component>
</logic>
<presentation>...</presentation>
</peers>
</uiml>
```

Fuente: elaboración propia.

3.1.8. Presentación

El código será mapeado a un lenguaje en específico. Para este caso es Java, por lo que se deberá especificar.

Figura 13. **Presentación**

```
<?xml version="1.0">
<uiml>
<interface>
<structure>
<part id="MyFrame" class="JFrame">
<part id="MyLabel" class="JLabel"/>
<part id="MyButton" class="JButton"/>
</part>
</structure>
<style>...</style>
<content>...</content>
<behavior>...</behavior>
</interface>
<peers>
<logic>...</logic>
<presentation base="Java_1.3_Harmonia_1.0"/>
</peers>
</uiml>
```

Fuente: elaboración propia.

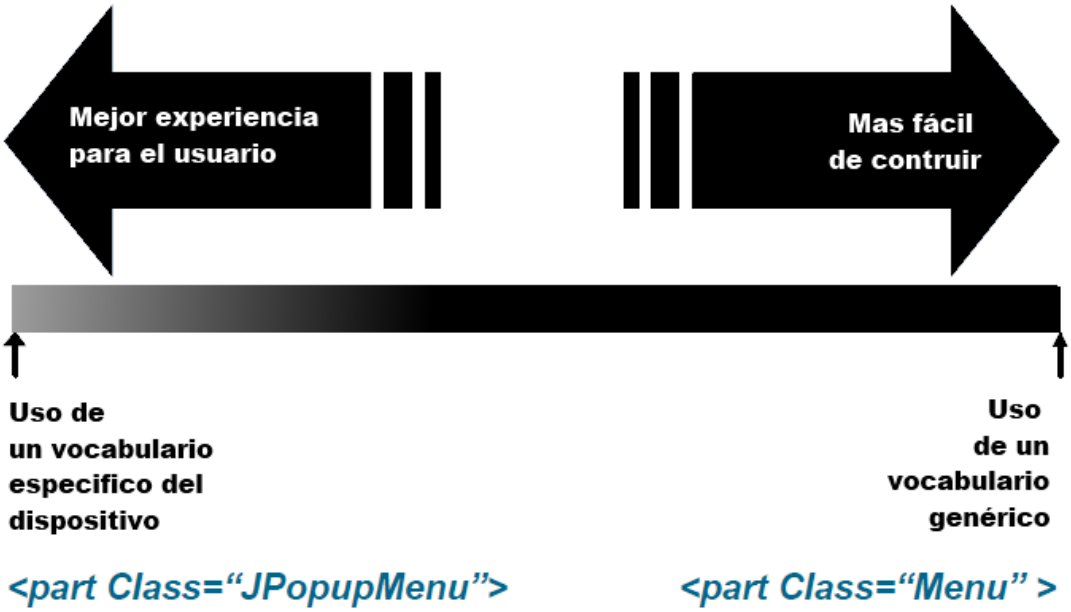
3.1.9. Interfaces multiplataforma

Existen tres maneras de lograr interfaces multiplataforma:

- Múltiples documentos UIML
- Un lenguaje genérico
- Vocabularios para las familias de dispositivos

En la siguiente imagen se muestra la línea de esfuerzo que amerita.

Figura 14. **Interfaces multiplataforma**



Fuente: VÖGLER, Gabriel.User Interface Markup Language. [en línea] 2003. http://www.jeckle.de/files/UIML_Voegler.pdf. Consulta: marzo de 2012.

3.1.10. Múltiples documentos UIML

Se debe crear un documento para cada plataforma usando el vocabulario específico para cada una.

Claramente esto tiene sus ventajas en cuanto a los métodos tradicionales, una de ellas es que se utiliza un lenguaje para todas las plataformas y el otro es que solo se necesita una herramienta de auditoría para cada plataforma.

3.1.11. Vocabulario genérico

Se trata de tener un vocabulario genérico para todas las plataformas, cumpliendo dos objetivos:

- Lo suficientemente potente como para incluir a una familia de dispositivos
- Lo suficientemente genérico como para que alguien que no es experto en todas las plataformas pueda utilizarlo sin problema.

Todo esto teniendo en cuenta que solo un subconjunto de todos los elementos de la interfaz gráfica del usuario, pueden ser apoyadas.

Tabla III. **Vocabulario**

Elementos Genéricos	Java Swing	PalmOS	WML	HTML
GButton	JButton	Command Button	<input type= “Button”>	<input type= “Button”>
GArea	JFrame | JWindow | JPanel	Window	<card>	<form> | <table>
GTop-container	JFrame	Form	<wml>	<html> and <body>

Fuente: elaboración propia.

Donde cada clase genérica tiene sus propiedades y eventos.

Tabla IV. **Propiedades y eventos**

Generic Name	GButton
Properties	Name, title, size, location, foreground, background, layout, font, border
Events	actionPerformed

Fuente: elaboración propia.

3.1.12. Mapeando un vocabulario genérico a Java y HTML

Esto puede ser escrito solamente una vez para cada evento y llamada al mundo exterior que se maneja de manera similar.

Figura 15. **Mapeo a Java y HTML**

```
...
<part id="myButton" class="GButton">
...
<presentation name="Java">
<d-class name="GButton" … maps-to="javax.swing.JButton">
…
</d-class >
</presentation>
<presentation name="HTML">
<d-class name="GButton" … maps-to="html:input">
…
</d-class >
</presentation>
```

Fuente: elaboración propia.

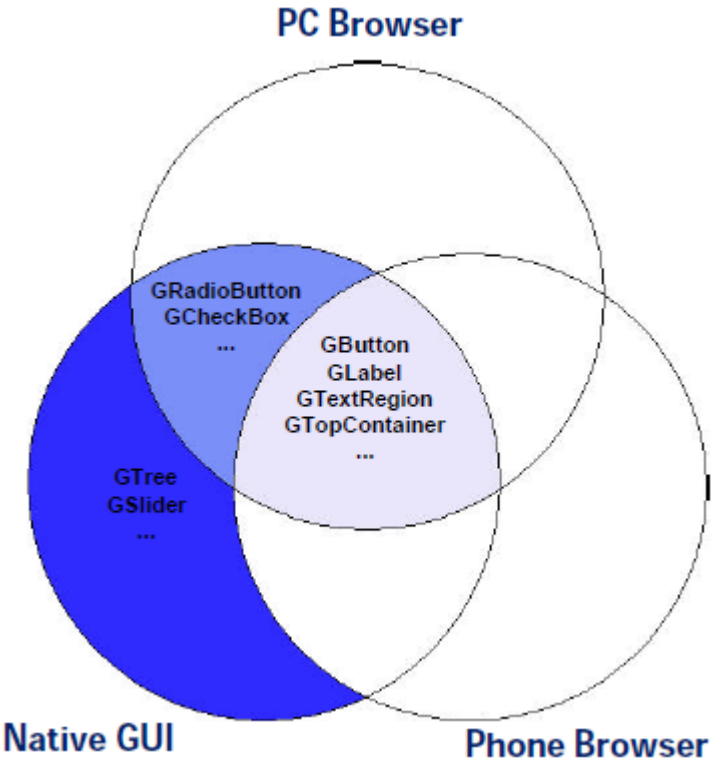
3.1.13. Describiendo las familias de interfaces de usuario

Existen dos problemas que con el uso del vocabulario genérico

- Muy pocos elementos que puedan ser soportados para cada dispositivo
- Diseño muy diferente

La solución para este problema es utilizar vocabularios para cada familia de dispositivos.

Figura 16. **Describiendo las familias de interfaces de usuario**

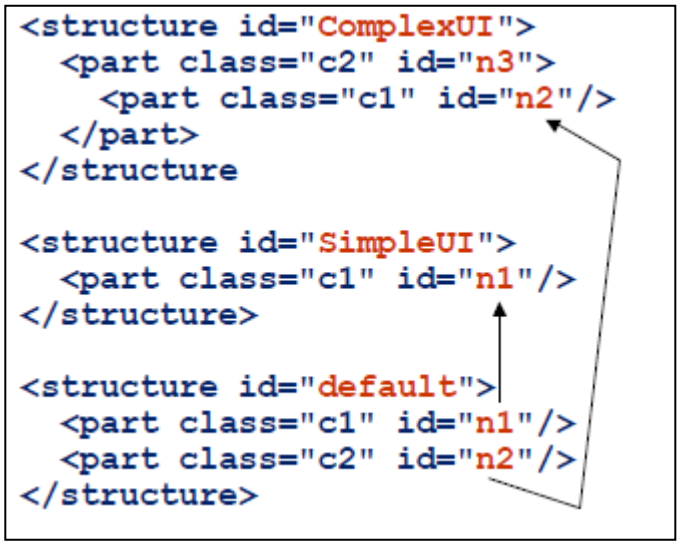


Fuente: VÖGLER, Gabriel.User Interface Markup Language. [en línea] 2003. http://www.jeckle.de/files/UIML_Voegler.pdf. Consulta: marzo de 2012.

3.1.14. Múltiples elementos <structure>

Descripción de los múltiples elementos estructurales contenidos en un solo documento.

Figura 17. **Múltiples elementos <structure>**



Fuente: VÖGLER, Gabriel.User Interface Markup Language. [en línea] 2003. http://www.jeckle.de/files/UIML_Voegler.pdf. Consulta: marzo de 2012.

3.1.15. Diferentes diseños

Para los diferentes diseños se debe cumplir con los siguientes aspectos

- Un proceso de diseño
- Describir primero
  - La metáfora de interfaz de usuario (GUI, los diálogos de voz, ...)
  - Los widgets a ser utilizados en la interfaz.

- Particiones en las pantallas o los diálogos.

- Redefinir la interfaz de usuario en grupos, donde cada grupo es un conjunto de plataformas con un diseño en común y diferentes widgets.
- Ajustar los miembros del grupo al elegir los widgets (solo si el vocabulario genérico permite múltiples opciones).
- Se necesitan buenas metodologías de diseño multiplataforma.

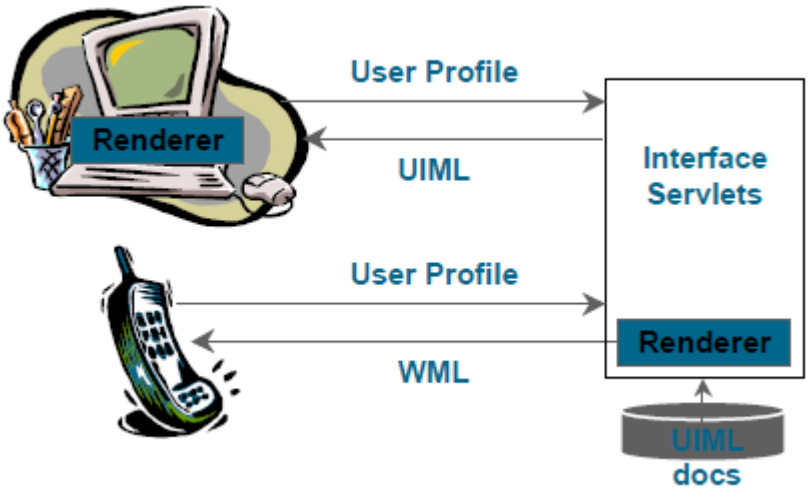
3.1.16. Renderizador de UIML

El renderizador de UIML puede:

- Interpretar UIML en el dispositivo cliente(como el navegador lo hace con el lenguaje HTML).
- Compilar UIML a otro idioma (WML, C++,…)

En la imagen siguiente se muestra como la interfaz de usuario se comunica con el exterior.

Figura 18. **Renderizador de UIML**



Fuente: VÖGLER, Gabriel.User Interface Markup Language. [en línea] 2003. http://www.jeckle.de/files/UIML_Voegler.pdf. Consulta: marzo de 2012.

Harmonia está creando una serie de renderizadores de UIML llamados en conjunto LiquidUI, los cuales se detallan a continuación:

- JAVA Renderer: intérprete que mapea interfaces de usuario en UIML a Java AWT / Swing.
- HTML Renderer: compilador que mapea interfaces de usuario en UIML a HTML para su visualización a través de navegadores web.
- WML Renderer: compilador que mapea interfaces de usuario en UIML a WML para su visualización con dispositivos compatibles.
- VoiceXML Renderer: compilador que mapea interfaces de usuario en UIML a VoiceXML para la implementación en los dispositivos de voz.
- InterfaceServer: personaliza UIML basado en un perfil y compila UIML a la plataforma destino.

3.1.17. UIML No es

Existen varias ideas erróneas de lo que verdaderamente se puede lograr con UIML.

- UIML no es para diseñar una sola interfaz y que esta funcione en cualquier dispositivo, sino más bien crear interfaces para distintos dispositivos con un solo lenguaje.
- Si se diseñan varias interfaces de usuario, estas pueden expresarse en un solo lenguaje, UIML.

Si se usa una metodología de diseño adecuada y un vocabulario multiplataforma, el conjunto de interfaces pueden ir descritas en un solo documento.

3.1.18. Implementación de un diccionario utilizando XIML

A continuación se detalla la implementación de la interfaz de un diccionario utilizando el lenguaje XIML, para explorar como los modelos se pueden especificar con varios niveles de detalle, y comparar estas especificaciones producidas por UIML.

En la imagen siguiente se muestra la interfaz del diccionario que consta de una caja de texto en donde se selecciona una palabra y se muestra su definición en la siguiente caja de texto.

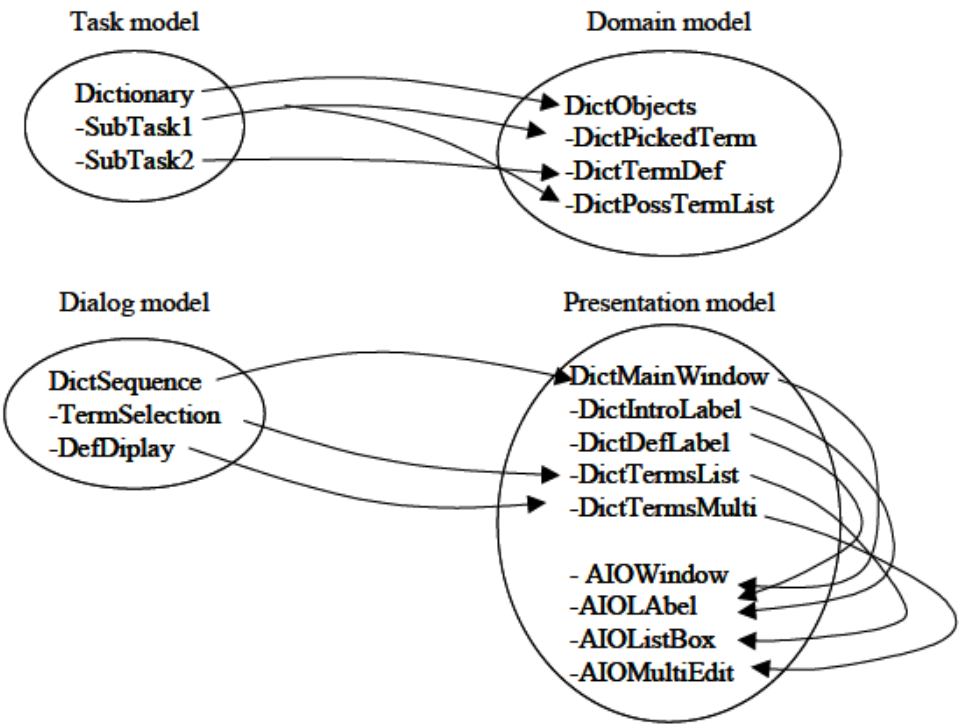
Figura 19. **Interfaz del diccionario**

Fuente: VANDERDONCKT, Jean. XIML Specification of a simple dictionary. [en línea] 2000. http://www.ximl.org/XIMLStarterKit.zip/SimpleDictionarySpec.pdf. Consulta: marzo de 2012.

3.1.18.1. Especificaciones de XIML

Las especificaciones de XIML relacionadas con el ejemplo y los comentarios respectivos para su entendimiento se demuestran en el siguiente prototipo. La figura representa gráficamente las relaciones entre los modelos y dentro de los modelos.

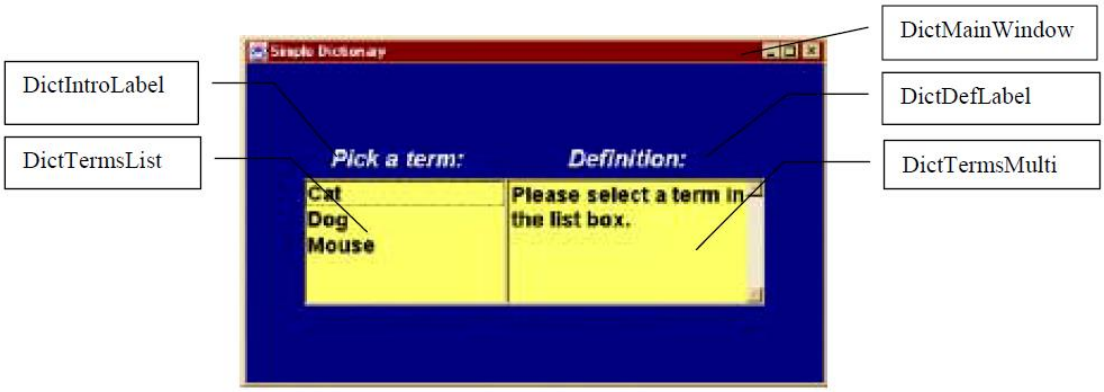
Figura 20. **Especificaciones de XIML**



Fuente: VANDERDONCKT, Jean. XIML Specification of a simple dictionary. [en línea] 2000. http://www.ximl.org/XIMLStarterKit.zip/SimpleDictionarySpec.pdf. Consulta: marzo de 2012.

La siguiente figura muestra gráficamente la composición física de la "DictMainWindow".

Figura 21. **Especificaciones en la interfaz del XIML**



Fuente: VANDERDONCKT, Jean. XIML Specification of a simple dictionary. [en línea] 2000. http://www.ximl.org/XIMLStarterKit.zip/SimpleDictionarySpec.pdf. Consulta: marzo de 2012.

A continuación se presenta y explica brevemente el XML del diccionario, la imagen siguiente representa las definiciones de las relaciones entre cada modelo de la interfaz.

Figura 22. **Definiciones**

```
<INTERFACE ID="_Ex1">
  <DEFINITIONS>
    <RELATION_DEFINITION NAME="Task_USES_Dom">
    <!--Definición de la relación entre el modelo de tarea y el modelo de dominio del diccionario
      afirmando que el diccionario de tarea usa un objeto de tipo diccionario-->
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="Dictionary" SLOT="left" />
        <CLASS REFERENCE="DictObjects" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
    <RELATION_DEFINITION NAME="Dial_IS_MAPPED_ONTO_Pres">
    <!--Definición de la relación entre el modelo de diálogo y el modelo de presentación
      afirmando que la secuencia del diálogo se proyecta dobre una ventana-->
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="DictSequence" SLOT="left" />
        <CLASS REFERENCE="DictMainWindow" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
    <!--Definición de las relaciones entre los elementos de diálogo y los elementos correspondientes-->
    <RELATION_DEFINITION NAME="TermSelection_IS_MAPPED_ONTO_DictTermsList">
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="TermSelection" SLOT="left" />
        <CLASS REFERENCE="DictTermsList" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
    <RELATION_DEFINITION NAME="DefDisplay_IS_MAPPED_ONTO_DictTermsMulti">
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="TermSelection" SLOT="left" />
        <CLASS REFERENCE="DictTermsList" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
  </DEFINITIONS>
```

Fuente: elaboración propia.

En la imagen siguiente se detallan los componentes del modelo. No existe un modelo de usuario, ya que se asume que la población de usuarios sigue siendo homogénea.

Figura 23. **Componentes del modelo**

```
<MODEL_COMPONENTS>

  <TASK_MODEL ID="Dictionary">
    <NAME> Simple Dictionary </NAME>
    <TASK_ELEMENT ID="DictSubTask1" EXECUTION_ORDER="optional">
      <NAME> Pick a term from the list of terms</NAME>
      <GOAL> DictPickedTerm !={} </GOAL>
      <CONDITION CONDITION_TYPE="pre"> DictPickedTerm={} </CONDITION>
    </TASK_ELEMENT>
    <TASK_ELEMENT ID="DictSubTask2" EXECUTION_ORDER="sequential">
      <NAME> Display definition of the picked term </NAME>
      <GOAL> Displayed(DictPickedTerm) = DictTermDef </GOAL>
      <CONDITION CONDITION_TYPE="pre" > DictPickedTerm!={} </CONDITION>
      <CONDITION CONDITION_TYPE="post"> Displayed(DictPickedTerm)=DictTermDef </CONDITION>
    </TASK_ELEMENT>
  </TASK_MODEL>

  <DOMAIN_MODEL ID="DictObjects">
    <NAME> Information viewed and manipulated by the user in the dictionary </NAME>
    <DOMAIN_ELEMENT ID="DictPickedTerm">
      <NAME> Term picked by the user </NAME>
    </DOMAIN_ELEMENT>
    <DOMAIN_ELEMENT ID="DictTermDef">
      <NAME> Definition of the term picked by the user </NAME>
    </DOMAIN_ELEMENT>
    <DOMAIN_ELEMENT ID="DictTermList">
      <NAME> List of all possible terms the user the user can pick up</NAME>
    </DOMAIN_ELEMENT>
  </DOMAIN_MODEL>

  <!--No existe un modelo de usuario, ya que se asume que la población de usuarios sigue siendo homogénea-->
```

Fuente: elaboración propia.

Las siguientes relaciones mapean un elemento de la presentación lógica a sus correspondientes elementos de la presentación física de las diferentes plataformas. Este mecanismo logra la transformación de los objetos de interacción abstractos (AIOs) en objetos de interacción concreta (CIOs o widgets) de las diferentes plataformas.

Figura 24. **Presentación del modelo**

```
<PRESENTATION_MODEL ID="DictBasicPres">
  <NAME> Basic presentation model for the dictionary </NAME>
  <DEFINITIONS>
    <!--Las siguientes relaciones mapean un elemento de la presentacion logica a sus correspondientes elementos
        de la presentacion fisica de las diferentes plataformas informaticas.
        Este mecanismo logra la transformación de los objetos de interacción abstractos (AIOs) en
        objetos de interacción concreta (CIOs or widgets) de las diferentes plataformas informaticas-->
    <RELATION_DEFINITION NAME="DictMainWindow_IS_a_AIOWindow">
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="DictMainWindow" SLOT="left" />
        <CLASS REFERENCE="AIOWindow" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
    <RELATION_DEFINITION NAME="DictIntroLabel_IS_a_AIOLabel">
      <ALLOWED_CLASSES>
        <CLASS REFERENCE="DictIntroLabel" SLOT="left" />
        <CLASS REFERENCE="AIOWindow" SLOT="right" />
      </ALLOWED_CLASSES>
    </RELATION_DEFINITION>
```

Fuente: elaboración propia.

En esta parte se muestra la jerarquía de los elementos de presentación. Las relaciones anteriores, establecen la conexión con los Abstract Interaction Objects o AIOs.

Figura 25. **Elementos de presentación**

```
<PRESENTATION_ELEMENT ID="DictMainWindow" LOCATION="optional">
  <!-- En esta parte se muestra la jerarquía de los elementos de presentación-->
  <!-- Las relaciones amteriores establecen la conexión con los Abstract Interaction Objects (AIOs) -->
  <NAME> Dictionary main window </NAME>
  <PRESENTATION_ELEMENT ID="DictIntroLabel" LOCATION="optional">
    <NAME> Dictionary introduction label </NAME>
  </PRESENTATION_ELEMENT>
  <PRESENTATION_ELEMENT ID="DictDefLabel" LOCATION="optional">
    <NAME> Dictionary definition label </NAME>
  </PRESENTATION_ELEMENT>
  <PRESENTATION_ELEMENT ID="DictTermsList" LOCATION="optional">
    <NAME> List of terms in the dictionary </NAME>
  </PRESENTATION_ELEMENT>
  <PRESENTATION_ELEMENT ID="DictTermsMulti" LOCATION="optional">
    <NAME> Definition of the term picked by user </NAME>
  </PRESENTATION_ELEMENT>
</PRESENTATION_ELEMENT>
```

Fuente: elaboración propia.

A continuación se detallan las definiciones de los AIOs propios.

Figura 26. **AIOs propios**

```
<!-- Las definiciones de los AIOs propios -->
<PRESENTATION_ELEMENT ID="AIOWindow" LOCATION="optional">
  <NAME> Definition of the abstract window </NAME>
  <PRESENTATION_ELEMENT ID="JavaWindow" LOCATION="http://www.ximl.org/ui/JavaAWTRenderings.ui#AWT">
    <NAME> Window rendered for the Java toolkit </NAME>
  </PRESENTATION_ELEMENT>
  <PRESENTATION_ELEMENT ID="WinNTWindow"
    LOCATION="http://www.ximl.org/ui/WinNT/window.vbx">
    <NAME> Window rendered for the Windows NT environment </NAME>
  </PRESENTATION_ELEMENT>
</PRESENTATION_ELEMENT>
<PRESENTATION_ELEMENT ID="AIOLabel" LOCATION="optional">
  <NAME> Definition of the abstract label </NAME>
  <PRESENTATION_ELEMENT ID="JavaLabel"
    LOCATION="http://www.ximl.org/ui/JavaAWTRenderings.ui#AWT">
    <NAME> Label rendered for the Java toolkit </NAME>
  </PRESENTATION_ELEMENT>
  <PRESENTATION_ELEMENT ID="WinNTLabel"
    LOCATION="http://www.ximl.org/ui/WinNT/label.vbx">
    <NAME> Label rendered for the Windows NT environment </NAME>
  </PRESENTATION_ELEMENT>
</PRESENTATION_ELEMENT>
<PRESENTATION_ELEMENT ID="AIOListBox" LOCATION="optional">
  <NAME> Definition of the abstract list box </NAME>
  <PRESENTATION_ELEMENT ID="JavaListBox"
    LOCATION="http://www.ximl.org/ui/JavaAWTRenderings.ui#AWT">
    <NAME> List box rendered for the Java toolkit </NAME>
  </PRESENTATION_ELEMENT>
```

Fuente: elaboración propia.

Se puede observar en las imágenes que las definiciones de los AIOs deben tener un nombre del elemento, una dirección y una descripción del mismo como mínimo.

Figura 27. **Mas AIOs**

```
    <PRESENTATION_ELEMENT ID="AIOMultiEdit" LOCATION="optional">
        <NAME> Definition of the abstract multi edit box </NAME>
        <PRESENTATION_ELEMENT ID="JavaMultiEdit"
            LOCATION="http://www.ximl.org/ui/JavaAWTRenderings.ui#AWT">
            <NAME> Multi-line edit box rendered for the Java toolkit </NAME>
        </PRESENTATION_ELEMENT>
        <PRESENTATION_ELEMENT ID="WinNTMultiEdit"
            LOCATION="http://www.ximl.org/ui/WinNT/multi.vbx">
            <NAME> Multi-line edit box rendered for the Windows NT environment </NAME>
        </PRESENTATION_ELEMENT>
    </PRESENTATION_ELEMENT>
</PRESENTATION_MODEL>
```

Fuente: elaboración propia.

En esta parte se muestra la estructura del modelo de diálogo con cada uno de sus elementos.

Figura 28. **Modelo de diálogo**

```
<DIALOG_MODEL ID="DictDialog">
  <NAME> Main dialog model for the dictionary example </NAME>
  <DIALOG_ELEMENT ID="DictSequence" EXECUTION_ORDER="sequential">
    <NAME> Sequencing of a selection followed by a display </NAME>
    <DIALOG_ELEMENT ID="TermSelection">
      <NAME> Selection of a term by the user </NAME>
      <GOAL> DictPickedTerm!={} </GOAL>
      <INTERACTION_TECHNIQUE>
        <RELATION_STATEMENT DEFINITION="TermSelection_IS_MAPPED_ONTO_DictTermsList"
          REFERENCE="DictTermsList"/>
      </INTERACTION_TECHNIQUE>
      <RESPONSE RESPONSE_TYPE="final">
        <RELATION_STATEMENT DEFINITION="TermSelection_IS_MAPPED_ONTO_DictTermsList"
          REFERENCE="DictTermsList"/>
      </RESPONSE>
    </DIALOG_ELEMENT>
    <DIALOG_ELEMENT ID="DefDisplay">
      <NAME> Display the definition of the picked term </NAME>
      <GOAL> Displayed(DictPickedTerm) = DictTermDef </GOAL>
      <INTERACTION_TECHNIQUE>
        <RELATION_STATEMENT DEFINITION="DefDisplay_IS_MAPPED_ONTO_DictTermsMulti"
          REFERENCE="DictTermsMulti"/>
      </INTERACTION_TECHNIQUE>
      <RESPONSE RESPONSE_TYPE="final">
        <RELATION_STATEMENT DEFINITION="DefDisplay_IS_MAPPED_ONTO_DictTermsMulti"
          REFERENCE="DictTermsMulti"/>
      </RESPONSE>
    </DIALOG_ELEMENT>
  </DIALOG_ELEMENT>
</DIALOG_MODEL>
</MODEL_COMPONENTS>
</INTERFACE>
```

Fuente: elaboración propia.

3.1.18.2. Comentarios

Las etiquetas ID y NAME a veces parecen muy similares, pero la etiqueta ID está destinada solo para propósitos internos, mientras que la etiqueta NAME es para propósitos externos (comunicación), aunque parece ser más una descripción que un nombre en sí. EL ID debe ser único, mientras que el NAME puede no ser único.

Una plantilla o Template puede ser una definición lógica de un conjunto de especificaciones XIML que pueden servir como una sentencia Include. Ya que en lugar de volver a escribir todo nuevamente, se escribe una plantilla la cual puede ser utilizada en cualquier parte del documento XIML. Esto a su vez nos daría la oportunidad de introducir patrones de diseño. Tener un patrón tendría sentido en el nivel de lenguaje de definición, para identificar patrones en comunes a través de una definición única o una familia de definiciones diferentes. La plantilla sería especialmente útil para el modelo de presentación (Presentation_model), donde a menudo se comparten varios elementos de presentación (Presentation_elements) que cuentas con las mismas propiedades.

La etiqueta Execution_order solo acepta valores, paralelos, secuenciales y opcionales como valores permitidos. Se podrían introducir algunos operadores más como el de elección, repetición, intercalación, habilitar/deshabilitar. Aunque se pueden definir estos operadores con un Atribute_statement, puede ser interesante incorporarlos directamente.

Algunos de los operadores alternativos y la notación utilizada en el ConcurTaskTree, se detallan a continuación:

- T* : Set/unset, es una tarea iterativa T
- [T] : Set/unset, es una tarea opcional T
- [] : Elección
- |||: Concurrente; ejemplo: T1 ||| T2 significa que los elementos de las dos tareas se pueden realizar en cualquier orden.
- |[ ]| : Concurrente con el intercambio de información
- [>: Disabling; ejemplo: T1 [> T2 significa que cuando un elemento de la segunda tarea ocurre entonces la primera tarea se desactivara.

- >>: Enabling; ejemplo: T1 >> T2 significa que cuando la primera tarea se termina se activara la segunda.
- []>>: Habilita intercambio de información; ejemplo: T1 []>> T2 significa que cuando termina la primera tarea le proporciona un valor de tarea a la segunda tarea, además de activarla.
- |>: Suspender o reanudar; ejemplo: T1 |> T2 significa que t1 puede suspender la ejecución de T2 o reanudarlo en caso de ya haya sido suspendida.

3.1.18.3. Especificaciones en UIML

Estas son las especificaciones relacionadas con la implementación anterior con la diferencia de utilizar el lenguaje UIML.

Figura 29. **Dictionary.ui**

```
<!--Root element = UIML tag-->
<?xml version="1.0"?>
<!--Este es un Dictionary.ui.
Consta de una caja de texto en donde se selecciona una palabra y
se muestra su definición en la siguiente caja de texto -->
<uiml>
```

Fuente: elaboración propia.

Esta sección describe la asignación de cada propiedad y el nombre del evento utilizado en otras partes del documento UIML para la interfaz de usuario y la lógica de la aplicación.

Figura 30. **Asignaciones**

```
<!--Definición de la ubicación de la libreria
que contiene las herramientas-->
<peers>
<presentation name="java"
source="http://www.uiml.org/toolkits/Java20AWT.ui"
how="replace"/>
</peers>
```

Fuente: elaboración propia.

Definición de un modelo de widget en términos de una jerarquía de clases con propiedades. Esta definición no es una definición de las clases abstractas de interacción de objetos en general, pues UIML soporta 28 tipos de controles (widgets) estandarizados. En lugar de eso esta es una descripción de la jerarquía de los widgets utilizados.

Figura 31. **Definición de un modelo widget**

```
<interface>
<structure>
<part class="Frame" name="ListBoxes">
<part class="Label" name="IntroLabel"/>
<part class="List" name="Terms"/>
<part class="Label" name="DefnLabel"/>
<part class="TextArea" name="Defn"/>
</part>
</structure>
```

Fuente: elaboración propia.

Definición del modelo de presentación directamente en términos de las propiedades de los controles.

Figura 32. **Definición de un modelo de presentación**

```
<style source="http://www.uiml.org/ui/JavaAWTRenderings.ui#AWT" how="cascade">
<property part-class="Frame" name="layout" >gridBagLayout</property>
<property part-class="Frame" name="xplace" >relative</property>
<property part-class="Frame" name="yplace" >relative</property>
<property part-class="Frame" name="background" >blue</property>
<property part-class="Frame" name="font-style" >bold</property>
<property part-class="Frame" name="location" >100,100</property>
<property part-class="Frame" name="size" >500,300</property>
<property part-class="Label" name="font-size" >20</property>
<property part-class="Label" name="foreground" >white</property>
<property part-class="Label" name="font-style" >boldItalic</property>
<property part-class="List" name="background" >gray</property>
<property part-class="TextArea" name="background" >gray</property>
<!--Titulo de la Ventana principal-->
<property part-name="ListBoxes" name="content" >Simple Dictionary</property>
<property part-name="IntroLabel" name="content" >Pick a term:</property>
<!--Definición del contenido del listbox-->
<property part-name="Terms" name="content">
<constant name="Cat" >Cat</constant>
<constant name="Dog" >Dog</constant>
<constant name="Mouse">Mouse</constant>
</property>
```

Fuente: elaboración propia.

En esta sección se muestran las definiciones de las propiedades globales para el estilo.

Figura 33. **Definiciones para el estilo**

```
<property part-name="DefnLabel" name="content">Defn:</property>
<property part-name="Defn" name="content">Please select a term in the list box</property>
<!--Definición de la posición lógica de cada widget-->
<property part-name="Terms" name="xplace" >0</property>
<property part-name="Terms" name="alignment" >north</property>
<property part-name="Terms" name="fill" >both</property>
<property part-name="DefnLabel" name="alignment" >center</property>
<property part-name="DefnLabel" name="xplace" >1</property>
<property part-name="DefnLabel" name="yplace" >0</property>
<property part-name="Defn" name="xplace" >1</property>
<property part-name="Defn" name="columns" >20</property>
<property part-name="Defn" name="rows" >4</property>
<property part-name="Defn" name="scrollbars">vertical-only</property>
<property part-name="Defn" name="editable" >false</property>
<property event-class="LSelected" name="rendering" >ItemEvent</property>
</style>
```

Fuente: elaboración propia.

A continuación se detallan las normas de comportamiento que tendrá la interfaz. Como por ejemplo en la siguiente regla se indica que si el valor seleccionado en el listado de términos es “Cat” entonces que aparezca en la caja de texto de definiciones (Defn) "Carnívoro, mamífero domesticado que es aficionado a las ratas y a los ratones".

Figura 34. **Normas de comportamiento**

```
<behavior>
<rule>
<condition>
<equal>
<event part-name="Terms" class="LSelected" name="item-selected"/>
<reference constant-name="Cat"/>
</equal>
</condition>
<action>
<property part-name="Defn" name="content"
>Carnivourous, domesticated mammal that's fond of rats and mice</property>
</action>
</rule>
```

Fuente: elaboración propia.

Las demás reglas del comportamiento y el resto del documento UIML se muestra a continuación:

Figura 35. **Otras normas de comportamiento**

```
<rule>
<condition>
<equal>
<event part-name="Terms" class="LSelected" name="item-selected"/>
<reference constant-name="Dog"/>
</equal>
</condition>
<action>
<property part-name="Defn" name="content"
>Domestic animal related to a wolf that's fond of chasing cats</property>
</action>
</rule>
<rule>
<condition>
<equal>
<event part-name="Terms" class="LSelected" name="item-selected"/>
<reference constant-name="Mouse"/>
</equal>
</condition>
<action>
- 14 -
<property part-name="Defn" name="content"
>Small rodent often seen running away from a cat</property>
</action>
</rule>
</behavior>
</interface>
</uiml>
```

Fuente: elaboración propia.

3.1.19. Diferencias entre XIML y UIML

Después de haber analizado la implementación del diccionario en ambos lenguajes, se pueden observar las siguientes diferencias:

- UIML se limita solo a 3 modelos: el modelo de widget (la jerarquía de elementos con sus atributos mostrados en la sección <structure>), el modelo de la presentación (mostrado en la sección <style> y el modelo de diálogo (mostrado en la sección <behavior>). En cambio XIML soporta el modelo de tarea, el de usuario, el de dominio y el modelo general, aunque este último puede ser representado por el modelo de widget de UIML.

- UIML soporta 28 widgets predefinidos como list, label, window, etc. Esto nos da la ventaja de reutilizar las definiciones existentes hasta el momento, pero también nos lleva al inconveniente de que no está claro como agregar o personalizar un widget. XIML no lo proporciona directamente, sino indirectamente ya que se supone que el desarrollador expresa estos widgets en el modelo de presentación. El inconveniente de esto es que el autor debe definir todos. Existe el riesgo de tener múltiples autores definiendo los mismos widgets, pero en maneras distintas. La solución de UIML de esto, es a través de una definición previa, lo que permite su reutilización, pero disminuye la flexibilidad. Por otro lado la solución de XIML es a través de la autodefinición, reduciendo así su reutilización, pero aumentando la flexibilidad. Aunque esto se puede minimizar al recomendar la utilización de algunas definiciones ya existentes.

- La etiqueta de estilo <style> parece estar limitada a los atributos de estilo únicos, mientras que XIML puede prácticamente soportar cualquier tipo de atributo. Sin embargo, UIML soporta la definición de las propiedades globales que son comunes a varios elementos de presentación. Las propiedades locales se heredan de las propiedades globales, a excepción de que se especifique una redefinición. Tal vez el concepto de TEMPLATE pueda ser utilizado por igual.

- El modelo de diálogo de UIML consiste a primera vista basada en reglas: Si una condición se cumple, entonces hacer tal acción. O bien: si se produce un evento, activar esta función.

CONCLUSIONES

1. En el análisis comparativo de los lenguajes se identificó características que permitieron evaluar las ventajas de un lenguaje de marca sobre uno tradicional, tales como la disminución en el uso de etiquetas y el diseño basado en componentes.

2. Se lograron evaluar diferentes metodologías de desarrollo, permitiendo evaluar aspectos relevantes en la construcción de interfaces, funcionalidad en los enlaces de navegación y los niveles de abstracción en el diseño y desarrollo de sistemas interactivos.

3. En el desarrollo práctico se lograron evaluar aspectos de compatibilidad con diferentes plataformas, uso de componentes, mapeo, adaptabilidad de dispositivos y ejecución en diferentes sistemas operativos del lenguaje UIML.

RECOMENDACIONES

1. Dar a conocer este tipo de metodologías, que logren colaborar en el desarrollo de servicios de sistemas de información.

2. Realizar propuestas de desarrollo con este tipo de tecnologías, que permita ofrecer herramientas de servicio para los procesos de negocios.

3. Desarrollar sistemas informáticos interactivos multicontexto que eleven la experiencia del usuario y aumenten la flexibilidad del servicio.

4. Diseñar interfaces de usuario con soporte para la fácil adaptación a cualquier dispositivo a través de lenguajes de marca.

BIBLIOGRAFÍA

1. AGRAWAL, Manoj. *Java Tip 123: Dial into the wireless world.* [en línea]. <http://www.javaworld.com/article/2077488/mobile-java/java-tip-123--dial-into-the-wireless-world.html>. [Consulta: 11 noviembre 2013].

2. ARAGÓN, Héctor. *Lineamientos de Diseño para Interfaces en UIML.* [en línea] [ref. de diciembre 2002]. Disponible en Web: <http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/aragon_c_h/indice.html>.

3. BURRIEL, Daniel. *La metodología de diseño centrado en el usuario.* [en línea] [ref. de julio 2011]. Disponible en Web: < http://www.congresointernetdelmediterraneo.com/presentaciones/2011/MK/torres-burriel.pdf>.

4. CASTELLANOS, Nohema. *Espacios personales genéricos en bibliotecas digitales.* [en línea] [ref. de enero 2004]. Disponible en Web: < http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/castellanos_r_na/indice.html>.

5. DE GRACIA, Avinguda. *Web 2.0: los nuevos desafíos de la interfaz de usuarios.* [en línea] [ref. de noviembre 2005] Disponible en Web: <http://www.usolab.com/articulos/desafios_interfaz_web_2.php>.

6. DEHAES, Viv. *Breve reseña del diseño centrado en el usuario (dcu)*. [en línea] [ref. de 11 de marzo de 2001] Disponible en Web: <http://www.interacciones.com.ar/breve-resena-del-diseno-centrado-en-el-usuario-dcu/>.

7. GRAHAM, Ian. *The HTML SourceBook.* [en línea]. [ref. enero de 2010]. Disponible en Web: <http://www.wiley.com/legacy/compbooks/graham/book.html#support>.

8. HUSSMANN, H., Meixner, G., Zuehlke, D*. Model-Driven Development of Advanced User Interfaces.* [en línea]. Disponible en Web: <http://is.uni-paderborn.de/uploads/tx_sibibtex/Pleuss_-_Model_driven_Development_of_Advanced_User_Interfaces.pdf>. [Consulta: 24 marzo 2014].

9. LÓPEZ, Víctor Manuel. *Interfaces de usuario adaptativas basadas en modelos y agentes de software.* [en línea] [ref. julio de 2005] Disponible en Web: <http://www.dsi.uclm.es/personal/VictorManuelLopez/mipagina/archivos/thesis.pdf>.

10. LUZARDO, Ana. *Diseño de la interfaz gráfica web en función de los dispositivos móviles*. [en línea] [ref. agosto 2009]. Disponible en Web: <http://www.palermo.edu/dyc/maestria_diseno/pdf/tesis.completas/43.luzardo.pdf>.

11. MARTÍNEZ, Chema. *Diseño centrado en el usuario.* [en línea] [ref. de 4 de mayo de 2007]. Disponible en Web: <http://www.maestrosdelweb.com/editorial/diseno-centrado-en-el-usuario/>.

12. MEHER, Jessica. *12 Critical elements every homepage must have*. [en línea] [ref. julio 2012]. Disponible en Web: < http://blog.hubspot.com/blog/tabid/6307/bid/31097/12-Critical-Elements-Every-Homepage-Must-Have-Infographic.aspx>.

13. MONTERO, Yusef y SANTAMARÍA, Sergio. *Informe APEI sobre Usabilidad.* [en línea] [ref. julio 2009]. Disponible en Web: <http://www.nosolousabilidad.com/manual/3_2.htm>.

14. MORENO, Luciano. *Componentes de una interfaz web*. [en línea]. [ref. de 22 de septiembre de 2005] Disponible en Web: <http://www.desarrolloweb.com/articulos/2171.php>.

15. PAVAN, Bárbara. *Historia del Software: GUI.* [en línea]. [ref. julio de 2012]. Disponible en Web: <http://bitelia.com/2012/02/historia-del-software-gui-graphical-user-interface>.

16. PUERTA, A., EISENSTEIN, J. *XIML: A Universal Language for User Interfaces*. [en línea]. [ref. de diciembre de 2010]. Disponible en Web: http://www.ximl.org/documents/XimlWhitePaper.pdf.

17. SÁNCHEZ, Pablo. *El futuro del diseño web.* [en línea] [ref. 6 de enero 2010] Disponible en Web: <http://www.aveiroperoni.com.ar/el-futuro-del-diseno-web>.

18. Universidad de Tarapacá. *Ingeniería del software.* [en línea]. [ref. febrero de 2011] Disponible en Web: <http://chitita.uta.cl/cursos/2011-2/0001007/recursos/r-8.pdf>.

19. VANDERDONCKT, Jean. *XIML Specification of a simple dictionary*. [en línea] [ref. marzo 2000]. Disponible en Web: <http:// http://www.ximl.org/XIMLStarterKit.zip/SimpleDictionarySpec.pdf>.

20. VÖGLER, Gabriel. *User Interface Markup Language.* [en línea] [ref. febrero 2003]. Disponible en Web: <http://www.jeckle.de/files/UIML_Voegler.pdf>.