Capítulo VI. Incorporación de guías al proceso de conversión

El presente capítulo aborda el tema referente a la modificación del convertidor de interfaces, para la incorporación de las guías de usabilidad a este último.

6.1 Diseño conceptual del software

El modelo que se presenta en la Figura 6.1 se basa en la arquitectura del convertidor "EditMos". Se pretende que con este modelo sea más fácil para el usuario de este sistema, integrar convertidores que contengan guías de usabilidad y que éstas puedan ser aplicadas de manera automática a las interfaces de entrada al convertidor.

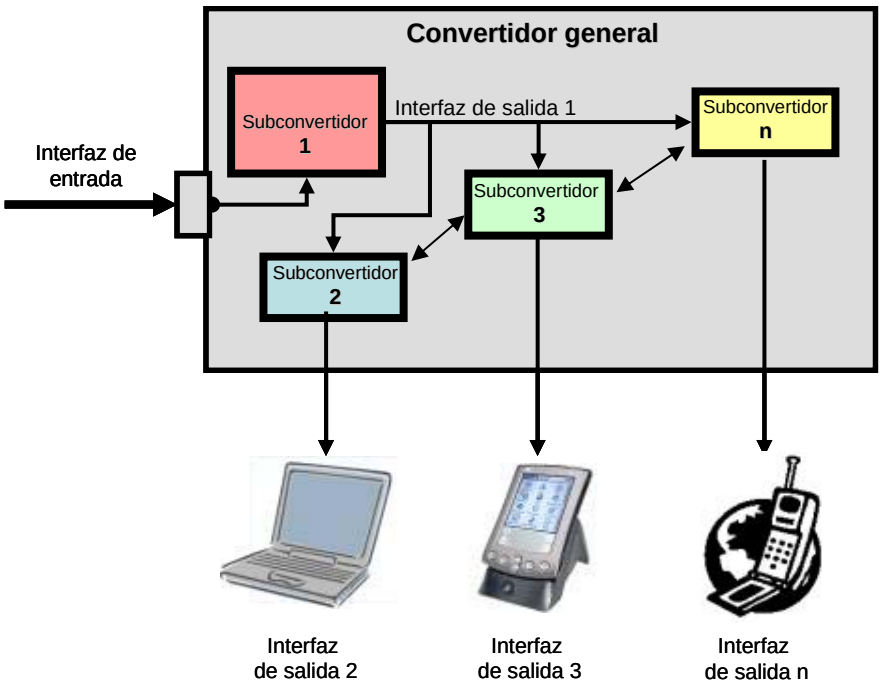

Figura 6.1 Modelo conceptual

A continuación se explica de manera detallada cada una de las partes del modelo conceptual. El modelo se divide en cuatro partes, las cuales son: convertidor general, interfaz de entrada, interfaces de salida y subconvertidores.

Una interfaz de entrada es aquella proporcionada por el usuario con el fin de transformarla a otro tipo de interfaz. Por otro lado las interfaces de salida son el producto de la conversión que se realiza en el convertidor general.

El convertidor general es el contenedor de todos los subconvetidores que están señalados en la Figura 6.1 como: Subconvertidor 1, Subconvertidor 2, Subconvertidor 3, Subconvertidor n. Este convertidor recibe como entrada una interfaz, dando como salida una interfaz con guías de usabilidad integradas.

Los subconvertidores forman parte del convertidor general, mismos que sirven como intermediarios entre la interfaz de entrada y la interfaz de salida. Cada uno de estos subconvertidores genera una interfaz de salida para diferentes tipos de dispositivos móviles.

Aplicando el modelo conceptual al convertidor “EditMos”, éste representaría al convertidor general, teniendo como subconvertidores a Tidy, Saxon y XSL.

Para poder resolver la problemática en la incorporación de guías al convertidor denominado “EditMos”, se estudió la estructura del mismo. Tomando en cuenta esta última, se modificaron las hojas de estilo correspondientes para generar una visualización a los documentos XML.

XHTML da la oportunidad de integrar “scripts” a sus páginas para hacerlas dinámicas, como lo es el mostrar letreros al apretar una liga o un botón, mostrar mensajes de confirmación, entre otros. Se sabe que existen dos grupos de “scripts”, uno llamado JavaScript y el otro VBScript, los cuales permiten hacer uso de este tipo de lenguajes de programación que se describen más adelante.

La decisión del uso de algún tipo de “scripts” se basa en diferentes circunstancias, como lo es el conocimiento previo del lenguaje, el trabajo que soporta y da estabilidad al lenguaje, además de la necesidad que se ha visto por el mayor número de navegadores.

El lenguaje que cumple con la mayoría de requerimientos para este proyecto es JavaScript, por contar con un estándar internacional denominado ECMAScript, el cual permite desplegar los scripts programados en este lenguaje en una gran variedad de navegadores como lo son Explorer, Netscape y otros. En la Tabla 6.1 se muestra un listado de los navegadores más usuales en PDAs, indicando en cuál de ellos se pueden visualizar los programas en JavaScript.

Tabla 6.1 Navegadores que soportan JavaScript

Nombre del navegador	¿Soporta JavaScript?
Netfront 3.1	SÍ
Blazer	SÍ
BTB browser	NO
Palm comm	SÍ
Pocket link	NO
Plucker	SÍ
Web Pro	SÍ
Opera	SÍ
Internet explorer	SÍ
Thunder howk	SÍ

6.2 Implementación de guías

En esta sección se explica la forma en la que se implementaron cada una de las guías al convertidor “EditMos”, con el fin de brindar información a detalle de cada una de ellas. Cada punto contiene el nombre de la guía, código en el que incrustó y por último una descripción de la forma en la que se implementó *.

Cabe mencionar que la incorporación de guías de usabilidad sólo se realizó al protocolo XHTML, dejando como trabajo a futuro la integración de dichas guías al protocolo WML.

Guía 1: Proveer al usuario una opción para regresar a la página anterior.

Código:

```
<xsl:template match="body" name="body">
<xsl:element name="body">
        <xsl:attribute name="bgcolor">white</xsl:attribute>
        <xsl:attribute name="text">black</xsl:attribute>
                <xsl:apply-templates/>
                <xsl:call-template name="ligaregreso"/>
    </xsl:element>
</xsl:template>
```

Este es la plantilla de la etiqueta “body” el cual llama a la plantilla de la etiqueta “ligaregreso”, mediante la función call-template ubicada en la línea 7 del “template”.

```
<xsl:template name="ligaregreso">
<center>
<font face="Courier New">   <a href="javascript:history.go(-1)">REGRESO</a></font> 
</center>
</xsl:template>
```

* Nota: Para ver el código completo de la hoja de estilo véase apéndice A.

La plantilla hace uso de una función de JavaScript que permite volver a la página anterior mediante la función “history.go (-1)”, el -1 indica cuantas páginas regresará, en este caso específico es sólo una. Cabe mencionar que la opción de regreso está definida tanto en liga como en botón, para que el usuario final tome la decisión del uso de una de ellas según sus necesidades.

Guía 2: Proveer al usuario una opción de salida.

Código:

```
<xsl:template match="body" name="body">
<xsl:element name="body">
        <xsl:attribute name="bgcolor">white</xsl:attribute>
        <xsl:attribute name="text">black</xsl:attribute>
                <xsl:apply-templates/>
                <xsl:call-template name="ligaregreso"/>
    </xsl:element>
</xsl:template>
```

La plantilla de la etiqueta “body” llama a la plantilla de la etiqueta “ligaregreso”, mediante la función call-template que está en la línea 7 del “template”.

```
<xsl:template name="ligaregreso">
<center>
<font face="Courier New">  <a href="javascript:askData();">SALIDA</a></font>
</center>
</xsl:template>
```

Al igual que la guía anterior, esta guía está implementada dentro de la plantilla “ligaregreso”, que a su vez hace un llamado a una función de JavaScript que se definirá a continuación. Esta opción se implementa tanto en liga como en botón, además de hacer el llamado de la función de confirmación.

Guía 3: Proveer un diálogo de alerta al salir del sistema.

Código:

```
<font face="Courier New">  <a href="javascript:askData();">SALIDA</a></font>
```

Una vez presionado el botón o liga de salida, se invoca la función “askData()” para que el XHTML sepa que va a utilizar una función que está programada en JavaScript, misma que debe definirse en la etiqueta “head” de la siguiente manera:

```
<xsl:template match="head" name="head">
<xsl:element name="head">
<script  language="JavaScript1.1" type="text/javascript" src="ficheros.js"></script>
<font face="arial"> <u>  <xsl:apply-templates/></u></font>
</xsl:element>
</xsl:template>
```

Esto se define por una etiqueta “script” que indica el lenguaje de éste, su tipo y la localización del código de la función.

Dentro del archivo ficheros.js se encuentra el siguiente código.

```
function askData() {
var inputedData =  confirm("Quieres salir?")
if (inputedData) {
window.location = "http://www.google.com/";
}
else {
window.location = "#";
    }
}
```

En esta función se define el letrero que llevará el mensaje de confirmación, además de incluir la liga a la que se desee llegar al momento de hacer la confirmación de salida o en su caso el regresar a la página de la cual no se confirmó salir.

Guía 4: Evitar dos blancos consecutivos entre líneas.

Para la eliminación de blancos entre líneas no fue necesario codificar, si no identificar los lugares donde se estaban colocando etiquetas de espacios que están identificados por el siguiente símbolo
 y evitar que aparecieran un sin número de veces. Esto se

logró evitar mediante la eliminación de la plantilla “br” y colocando esta etiqueta sólo en los lugares necesarios, como lo es el espacio después del título principal.

Guía 5: Diferenciar los encabezados del contenido.

Código:

Una vez más la identificación de plantillas fue fundamental en la implementación de la guía, en este caso en particular se identificó la etiqueta <p/> ya que ésta marca un párrafo y por lo regular se utiliza para colocar encabezados. Otra de las etiquetas identificadas importantes es <h1/> - <h6/> que permite indicar el tamaño del encabezado, donde <h1/> define el tamaño más grande del mismo. El conflicto dentro de este punto se encontró en la aplicación de esta guía, debido que al utilizar negritas la información que está dentro de la etiqueta <p/> corre el riesgo de tomar un párrafo muy grande y no sólo el título. Como se menciona, la diferenciación entre los encabezados y el contenido se realiza utilizando negritas en éstos.

Guía 6: Utilizar sólo dos tamaños y tipos de letra a lo largo de la página**.**

Código:

Esta es otra de las guías que partieron de una identificación de aquellas plantillas que daban como salida una letra, un párrafo, una liga, una palabra, etc. Para después integrar a ellos etiquetas que permiten definir el tamaño y el tipo de letra. Un ejemplo de estas etiquetas integradas es:

```
<font face="Courier New">
    <xsl:call-template name="ultimo"></xsl:call-template>
        </font>
```

Esta etiqueta “font” define como tipo de letra “Courier New”, para todo elemento en la plantilla llamada “último”. Para este trabajo en específico se utilizó tipografía de tipo

“Courier New” y “Tahoma” que son dos de los tipos de letra disponibles para navegadores de PDAs.

Guía 7: Ajustar el texto de la liga a una sola línea.

Código:

```
<xsl:template match="a">
<xsl:element name="a">
<xsl:attribute name="href" >
  <xsl:value-of select="@href" /><br/>
 </xsl:attribute> <br/>
          <font face="Courier New">
        <xsl:call-template name="ultimo"></xsl:call-template>
              </font>
</xsl:element>
</xsl:template>
```

En la línea ocho tomando como referencia el comentario se llama a la plantilla “ultimo”.

```
<xsl:template name="ultimo">
              <xsl:variable name="cadena" select='substring(.,0,20)'/>
              <xsl:call-template name="substring-after-last">
               <xsl:with-param name="input" select="$cadena" />
               <xsl:with-param name="marker" select="' '" />
               </xsl:call-template>
</xsl:template>
```

Esta plantilla utiliza la función “substring”, misma que permite regresar una subcadena dependiendo de los valores que recibe esta función.

El primero de los valores es la cadena que recibe, el segundo es la posición en la que comenzará el “substring”, finalizando con la posición en la que termina.

Una vez que se determina el “substring” se asigna a una variable que servirá como parámetro de la plantilla a la que se llama; "substring-after-last".

Otro de los parámetros de entrada es el símbolo que determina la palabra final de la función “substring”.

```
<xsl:template name="substring-after-last">
<xsl:param name="input" />
<xsl:param name="marker" />
```

```
<xsl:choose>
 <xsl:when test="contains($input,$marker)">
  <xsl:call-template name="substring-after-last">
   <xsl:with-param name="input"
     select="substring-after($input,$marker)" />
   <xsl:with-param name="marker" select="$marker" />
  </xsl:call-template>
 </xsl:when>
 <xsl:otherwise>
<xsl:if test="string-length($input) &gt; 4">
            <xsl:value-of select='substring(.,0,20)'/>
 </xsl:if>
   <xsl:if test="string-length($input) &lt; 4">
            <xsl:value-of select='concat(substring(.,0,20-string-length($input)))'/>
    </xsl:if>
 </xsl:otherwise>
</xsl:choose>
</xsl:template>
```

Una vez que se define la entrada de la plantilla "substring-after-last" y el símbolo que separa la palabra final, se pregunta si ésta última es mayor de cuatro caracteres para que la subcadena se imprima completa, de no ser así se elimina la última palabra.

Guía 8: Optimizar el tamaño de los gráficos

Código:

```
<xsl:template match="img" name="img">
<xsl:element name="img">
        <xsl:attribute name="src">
        <xsl:value-of select="@src"/>
        </xsl:attribute>
        <xsl:attribute name="alt">
        <xsl:value-of select="@alt"/>
        </xsl:attribute>
        <xsl:attribute name="width">30
        </xsl:attribute>
        <xsl:attribute name="height">30
        </xsl:attribute>
        <xsl:attribute name="border">0
        </xsl:attribute>
</xsl:element><br/>
</xsl:template>
```

Para resolver este problema se redujo los gráficos a tamaño 30 x 30 pixeles. Esto se señala en el atributo “width” y el atributo “height”.

Guía 9: Hacer uso de colores contrastantes

Código:

```
<xsl:template match="body" name="body">
<xsl:element name="body">
        <xsl:attribute name="bgcolor">white</xsl:attribute>
        <xsl:attribute name="text">black</xsl:attribute>
                <xsl:apply-templates/>
                <xsl:call-template name="ligaregreso"/>
    </xsl:element>
</xsl:template>
```

El código anterior se refiere a la forma de implementación de la guía número nueve, misma que contiene el atributo “bgcolor” el cual da color al fondo de la página, para hacer contraste con éste se utiliza el atributo “text” que asigna el color al texto de la página. Los colores utilizados en el sistema desarrollado en este proyecto es el blanco como fondo y negro para el texto, debido al contraste que generan ambos colores.

Guía 10: Optimizar el uso de tablas

Código:

```
<xsl:template match="table" name="table">
<xsl:param name="border"><xsl:value-of select="@border"/></xsl:param>
<xsl:param name="cellspacing"><xsl:value-of select="0"/>
</xsl:param>
<xsl:param name="cellpadding"><xsl:value-of select="0"/></xsl:param>
<xsl:param name="width"><xsl:value-of select="@width"/></xsl:param>
<xsl:param name="cols"><xsl:value-of select="@cols"/></xsl:param>
<xsl:param name="bgcolor"><xsl:value-of select="@bgcolor"/></xsl:param>
<table border="0" cellpadding="3" cellspacing="3" width="20%" cols="{$cols}" bgcolor="white">
    <xsl:for-each select="tr"  >
    <tr bgcolor="white">
        <xsl:call-template name="imprime_renglon"/>
    </tr>
    </xsl:for-each>
</table>
</xsl:template>
```

Para poder hacer válida esta guía en el convertidor, se estudió la etiqueta “table” para de esta forma predefinir el ancho de la tabla, el espacio entre bordes, el espacio entre el contenido de la celda y la celda misma y el ancho del borde de toda la tabla. Los

tamaños predefinidos son los siguientes: border="0" cellpadding="3" cellspacing="3" width="20%". Con esto se optimiza el tamaño de la tabla.

En el siguiente capitulo se muestran los resultados obtenidos al incorporar las guías de usabilidad al convertidor de interfaces, explicando las diferencias que existen entre las interfaces usables y las interfaces sin implementación de guías.