Aplicando Estrategias de Mapeo OOHDM–WCML para el desarrollo de Aplicaciones Web

Germán Terrazas[(1)], Gustavo Rossi[(1)], Fernando Lyardet[(1)], Christian Segor[(2)]

(1)
7.1 LIFIA - Facultad de Informática
Universidad Nacional de La Plata
50 y 115, 1er piso
1900 La Plata
Argentina

EMail: {german,gustavo,fer}@sol.info.unlp.edu.ar

(2)
Telecooperation Office (TecO)
Universität Karlsruhe
Vincenz-Prießnitz-Str. 1
76131 Karlsruhe
Alemania

EMail: segor@teco.uni-karlsruhe.de

Desde hace bastante tiempo es aceptado que para el diseño e implementación de aplicaciones web es necesario el empleo de algún método de ingeniería de software. Los métodos empleados en este tipo de desarrollo no proporcionan, con mucha frecuencia, una manera eficiente de implementar el diseño de la aplicación y además el mantenimiento del código resultante no es tarea fácil. En este trabajo se muestran los resultados obtenidos al usar Object Oriented Hypermedia Design Method (OOHDM) para el diseño de una aplicación web y el uso de WebComposition Markup Languaje (WCML) para su implementación mediante una estrategia de mapeo.

1 Introducción

El desarrollo a gran escala de aplicaciones web, del estilo hypermedia, se ve dificultada por las diferentes maneras "ad-hoc" de desarrollo existentes y por la complejidad de su mantenimiento. Por este motivo, desde hace bastante tiempo en la comunidad de desarrollo de tales aplicaciones, es aceptado que para su diseño e implementación es necesario el empleo de algún método de ingeniería de software. Desafortunadamente, los métodos empleados no proporcionan una manera eficiente de implementar el diseño obtenido, el mantenimiento del código resultante no es tarea fácil y la mayoría de ellos se complican cuando la aplicación debe proveer diversas maneras de navegación.
Uno de los métodos usados para el diseño de aplicaciones hypermedia es OOHDM. Este método nos provee un modelo orientado a objetos del dominio de la aplicación que puede ser implementado en diferentes lenguajes de aplicaciones web como XML, HTML, etc.. Sin embargo la implementación de un modelo orientado a objetos en un lenguaje que no lo es produce la pérdida de algunas propiedades del diseño como el rehuso y mantenimiento de código, herencia, etc.
Dado que es posible obtener el diseño de un modelo orientado a objetos para aplicaciones web, necesitamos un lenguaje para su implementación que nos permita mapear fácilmente el modelo obtenido sin perder las propiedades del paradigma. WCML es un lenguaje para el desarrollo de aplicaciones web orientado a objetos y se basa en la definición de componentes, sus propiedades y la composición de ellos.
Este trabajo tiene como objetivo mostrar las experiencias y resultados obtenidos al usar OOHDM y WCML para el diseño e implementación de una aplicación web mediante una estrategia de mapeo entre ambos. En la siguiente sección introduciremos conceptos básicos de la metodología OOHDM; seguido a esto damos a conocer conceptos, características y algunos ejemplos de uso de WCML. Luego, en la cuarta sección presentaremos una estrategia de mapeo de modelos OOHDM a WCML para finalmente mostrar, a modo de ejemplo, las diferentes etapas de desarrollo de una pequeña parte del proyecto Portinari [Portinari].

2 Introducción a OOHDM

En esta sección introduciremos brevemente la metodología OOHDM (Object Oriented Hypermedia Design Method) pudiendo contar con información más específica en [Rossi99, Schwabe95, Schwabe98].
OOHDM considera al desarrollo de una aplicación hypermedia como un proceso compuesto por cuatro actividades: la definición del esquema conceptual del dominio de la aplicación, el diseño del esquema navegacional, la especificación de la interfaz del usuario y la implementación.
La definición del esquema conceptual del dominio de la aplicación, se realiza utilizando los principios de diseño de la programación orientada a objetos y otras primitivas tales como la definición de atributos y de subsistemas.
Una de las características principales de las aplicaciones web es la noción de navegación. En OOHDM, se considera a una aplicación web como una vista navegacional del modelo conceptual. El diseño de esta vista es lo que se obtiene en la etapa de diseño del esquema navegacional teniendo en cuenta los diferentes tipos de usuarios que tendrá la aplicación.
El esquema navegacional puede ser recorrido mediante diferentes contextos. Un contexto navegacional es un conjunto de nodos relacionados que poseen una determinada estrategia de recorrido.
En la especificación de la interfaz del usuario se definen los objetos que serán vistos por los usuarios, la representación que tendrán los objetos navegacionales, los objetos que permitirán la navegación, la sincronización entre los objetos que brindan funcionalidad multimedia y las transformaciones que sucederán en la interfaz mientras el usuario navega.
Finalmente, en la etapa de implementación, se realiza el mapeo de los objetos del esquema navegacional y los de la interfaz a un lenguaje de programación. Para que en el mapeo no perdamos las propiedades que brinda la programación orientada a objetos que posee nuestro modelo, necesitamos un lenguaje con tales características.
En la siguiente sección se dan a conocer conceptos generales, características y ejemplos de uso de un lenguaje orientado a objetos para la implementación de un modelo hypermedia obtenido con OOHDM.

3 Elementos básicos de WCML

En esta sección describiremos brevemente conceptos básicos de WCML [Gaedke99] (WebComposition Markup Language), características y algunos ejemplos de uso.
WCML es un lenguaje que implementa los conceptos de WebComposition definidos en [Gellersen97]. Una especificación en este lenguaje se obtiene mediante la definición de uno o más componentes. Cada componente posee un identificador de referencia, esta definido por un conjunto de propiedades y cada propiedad se define como un par nombre-valor.

Una especificación en WCML posee las siguientes propiedades:

- Las decisiones generales de diseño pueden ser capturadas en componentes abstractas
- Las componentes abstractas pueden ser usadas como prototipos de otras componentes y de esta manera manejar el concepto de herencia
- Las componentes definidas pueden tener diferentes niveles de refinamiento
- La composición de componentes permite mapear de alguna forma el concepto de colaboración
- Nuevas componentes pueden ser agregadas fácilmente rehusando o modificando código de componentes previamente existentes en la definición de nuestra aplicación

Dado que WCML es una implementación de XML (eXtensible Markup Language) [W3CX98], la definición de las componentes, propiedades y relaciones entre componentes se basan en el concepto de tag. De las características que este lenguaje hereda por ser una implementación de XML, como por ejemplo ser de plataforma independiente, y de las propiedades que posee una especificación en este lenguaje se tiene que WCML es orientado a objetos.

Definición de componentes WCML	Valor de *content*	Vista final
<component uuid='Cversion'> <property name='content'> version 2.23 </property> </component>	Version 2.23	Version 2.23
<component uuid='CversionNice'> <property name='fontstyle' value='b'/> <property name='content'> <<refprop name='fontstyle'/>> <refprop name='content' from='Cversion'/> </<refprop name='fontstyle'>> </property> </component>	 Version 2.23 	**Version 2.23**
<component uuid='CversionNice2'> <prototype is='CversionNice'/> <property name='fontstyle' value='i'/> </component>	 Version 2.23 	*Version 2.23*

Figura 1: Ejemplo de uso de WCML usando generalización y composición entre componentes

En la figura 1 podemos ver un pequeño ejemplo de definición, herencia y composición entre componentes. En este ejemplo se define una componente (CVersion) que informa su número de versión. Esta componente será usada para proveer la versión a una segunda componente (CVersionNice) mostrando de esta manera el concepto de agregación entre componentes. Finalmente se define una tercer componente (CversionNice2) que hereda de CVersionNice. Esta última componente usa el concepto de herencia cambiando el formato de la vista definido en su superclase mediante la redefinición de una propiedad llamada *fontstyle* que especifica el tipo de letra con que se muestra la versión.

4 Estrategia de mapeo OOHDM-WCML

Esta sección describe de manera resumida una estrategia que sirve para mapear modelos obtenidos con OOHDM en WCML. El proceso de especificación de componentes descripto en este framework se realiza mediante dos operaciones de mapeo totalmente independientes que pueden llevarse a cabo simultáneamente. Estas operaciones se centran en la especificación del "*Esquema de Clases Navegacionales*" y el "*Esquema de Contextos Navegacionales*" obtenidos mediante la metodología OOHDM. Una vez finalizada la especificación de cada esquema, ambos se componen dando origen a una estructura de documento que implementa la aplicación.

4.1 Mapeo del Esquema de Clases Navegacionales

En un modelo OOHDM, el esquema navegacional se obtiene a partir del esquema conceptual mediante la definición de distintos tipos de vistas especificadas por el usuario. Las clases navegacionales definidas en esta fase se especifican como especializaciones de un conjunto de clases básicas que definen la semántica de los objetos navegacionales. Luego, una vez que han sido definidas todas las clases navegacionales, cada clase deberá ser mapeada a una componente de WCML mediante las siguientes reglas.
El mapeo comienza con la definición de una componente denominada FR_INFO, que especifica de manera standard cómo serán mostradas las propiedades contenidas en una componente. Esta funcionalidad queda definida para alguna tarea relacionada con el desarrollo de la aplicación.

El paso siguiente es mapear cada una de las clases navegacionales. Cada clase será mapeada como una componente INFO_x (donde x es el nombre de la clase mapeada) que hereda de FR_INFO y sus atributos estarán representados por las propiedades de dicha componente.
Finalmente, las componentes abstractas de datos (ccd) que heredan de INFO_x son utilizadas para instanciar diferentes componentes concretas de datos correspondientes al dominio de la aplicación.

4.2 Mapeo del Esquema de Contextos Navegacionales

En un modelo OOHDM, un nodo de información puede ser accedido en diferentes tipos de contextos navegacionales [Schwabe96, Schwabe98]. Los contextos navegacionales generalmente son inducidos por las clases navegacionales.
El mapeo del esquema de contextos navegacionales comienza por la definición de una componente denominada FR_STYLE, que contiene información general y común a todos los documentos (color, fondo, tipo de letra, etc.). La descendiente directa de esta componente es FR_PG, que define el contenido y un formato general para todos los documentos.
Los ítems navegacionales se definen en una componente llamada FR_NAV, que hereda de FR_PG y posee una propiedad en particular llamada NAV_Navigation donde se definen todas las relaciones navegacionales entre documentos.

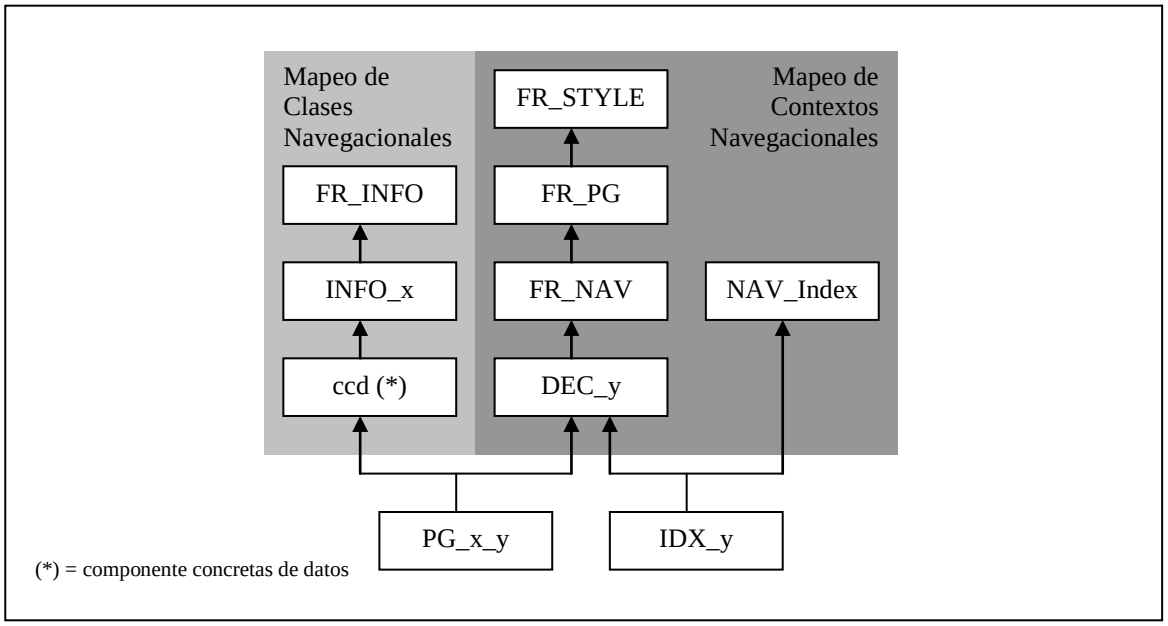

Figura 2: Estructura jerárquica del mapeo del esquema de clases navegacionales y del esquema de contextos navegacionales en WCML

Finalmente, la última componente de esta jerarquía son los decoradores de contexto DEC_y (donde y es el nombre del decorador) que decoran la información de acuerdo al contexto navegacional. Estas componentes poseen una propiedad denominada NAV_Layout donde se especifica la forma en que se muestra la información al usuario.
Una vez que ambos esquemas han sido mapeados en WCML, estos se unen para dar origen a una componente con herencia múltiple. Esta nueva componente se denomina PG_x_y (donde x es el nombre de la componente de datos e y el nombre del decorador) y define un documento HTML [W3CH98]. Cada uno de estos documentos representará diferentes nodos de información que podrán ser accedidos de acuerdo a un contexto en particular elegido por el usuario. En la figura 2 se muestran las estructuras jerárquicas de ambos esquemas definidas por el framework.

5 Ejemplo de aplicación: Proyecto Portinari

La metodología de diseño hypermedia orientada a objetos (OOHDM) no especifica ninguna forma o tipo de implementación del modelo obtenido, quedando dicha responsabilidad por parte el programador. Por otro lado, WCML provee un enfoque de implementación orientado a objetos para aplicaciones web sin necesidad de que el modelo a implementar sea producto de alguna metodología de diseño en especial. En esta sección proponemos la implementación, en WebComposition Markup Language, de un modelo obtenido con OOHDM. Por razones de simplicidad, usaremos un ejemplo basado en la generación de documentos HTML.

5.1 Especificación y Diseño

El ejemplo de aplicación seleccionado, permitirá al usuario navegar por una colección de obras del artista Candido Portinari. Las obras podrán recorrerse por tema o por fecha de realización permitiéndole al usuario cambiar entre estos dos tipos de contexto en cualquier instante de la navegación. El esquema conceptual completo de la aplicación puede verse en [Schwabe98] al igual que el esquema navegacional. Respecto a este último solo hemos elegido dos tipos de contextos navegacionales con el propósito de simplificar y realizar una rápida implementación con WebComposition Markup Languaje. En la figura 3 se muestran los dos contextos navegacionales elegidos para la realización de este ejemplo.

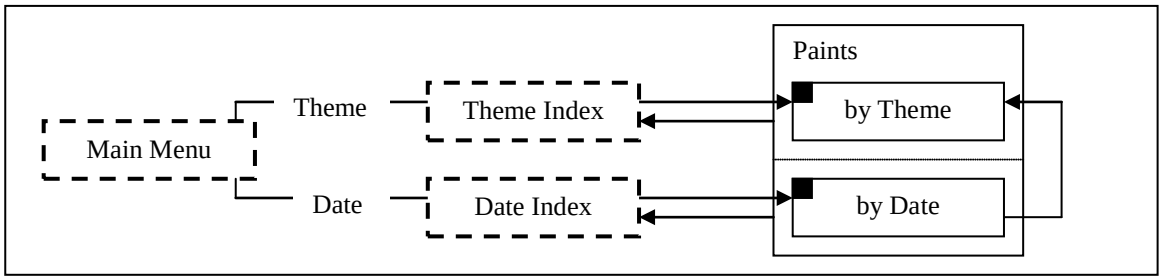

Figura 3: Contextos navegacionales de la aplicación

Para la especificación de las diferentes componentes que componen la aplicación nos basamos en la estrategia de diseño descripta en la sección anterior. El proceso de especificación de componentes descripto en dicho framework se realiza mediante dos operaciones de mapeo totalmente independientes que pueden ser llevadas a cabo simultáneamente. Estas operaciones se centran en la especificación del esquema de clases navegacionales y del esquema de contextos navegacionales obtenidos mediante la metodología OOHDM.

5.2 Implementación

Una vez elaborado con OOHDM el modelo del esquema conceptual y del esquema navegacional de la aplicación, realizamos la especificación en WCML de los distintos componentes mediante el uso del framework anteriormente descripto.
En el esquema de clases navegacionales hemos definido un prototipo (INFO_Artwork) que hereda de FR_INFO. Este prototipo tiene como objetivo especificar componentes concretas para cada obra de la colección. Las propiedades que se definen en este prototipo hacen referencia al título de la obra, la fecha de realización, una descripción textual de la misma, las dimensiones del cuadro, la firma del artista, una imagen de la obra y la técnica de pintado con la que se realizó. La figura 4 muestra la especificación de una componente concreta de datos de esta clase de componentes.

```
<component uuid='Artwork3'>
    <prototype is='INFO_Artwork'/>

    <property name='IdArtwork' value='3'/>
    <property name='Title' value='Menino com Estilingue'/>
    <property name='Date' value='1958'/>
    <property name='Description'
              value='Assinada e datada no canto superior direito PORTINARI 58'/>
    <property name='Dimentions' value='100 x 81 cm'/>
    <property name='Signature' value='No signature'/>
    <property name='LowResImg' value='meninoe.gif'/>
    <property name='Technique' value='Oleo/tela'/>
</component>
```

Figura 4: Definición de una componente concreta del prototipo INFO_Artwork

En el esquema de contextos navegacionales hemos definido dos componentes DEC_ByDate y DEC_ByTheme que heredan de FR_NAV. Estas componentes denotan los diferentes contextos navegacionales y actúan como decoradores de los mismos. Cada decorador provee la estrategia de navegación entre las diferentes obras que integran la colección, el formato de presentación de la información y la manera en que el usuario puede cambiar de contexto.
Dado que la aplicación se basa en la generación de documentos HTML que representan la información de un nodo accedido en un cierto contexto navegacional, se define una nueva componente usando herencia múltiple. Esta nueva componente (PG_x_y) hereda de cada uno de los diferentes decoradores y de las componentes concretas de INFO_Artwork. La figura 5 muestra la definición de una componente llamada PG_art3_byTheme que implementa un documento HTML, mostrando una obra accedida dentro del contexto “obras por tema”. La misma obra podrá ser accedida dentro del contexto “obras por fecha de realización” y la presentación de su información se realizará de diferente manera.

```
<component uuid='PG_art3_byTheme'>
    <prototype is='DEC_ByTheme'/>
    <prototype is='Artwork3'/>

    <property name='NAV_INTER_Somelink' value='PG_art4_byDate'/>
    <property name='NAV_INTRA_Prev' value='PG_art2_byTheme'/>
    <property name='NAV_INTRA_PrevTxt'
              <refprop name='Title' from='PG_art2_byTheme'/>
    </property>
    <property name='NAV_INTRA_Home' value='IDX_AnimalsByTheme'/>
    <property name='wcml.filename' value='PG_art3_byTheme.html'/>
</component>
```

Figura 5: Definición de un nodo del contexto navegacional “obras por tema”

El acceso a la aplicación se realiza a través de un menú principal que permite elegir el tipo de recorrido o contexto de navegación. Luego un índice correspondiente al contexto elegido, permite seleccionar las obras a visitar de acuerdo al tema, o intervalos de fechas en que fueron realizadas, y dentro del mismo tema, o intervalo de fechas, se puede acceder a una obra en particular. Tanto el menú principal como los diferentes índices fueron modelados con componentes distintas.
El menú principal se define con una componente (IDX_main) usando herencia múltiple; una de las componentes de la que hereda es un decorador (DEC_main) que se encarga de definir los dos contextos opcionales de navegación. Los menúes secundarios de cada contexto de navegación también fueron definidos como componentes, usando la misma metodología que para el menú principal. A estos últimos se le agregan opciones de navegación que son definidas en sus respectivos decoradores. La figura 6 muestra la definición de la componente del menú principal e índices del contexto “obras por tema”.

```
<component uuid='IDX_AnimalsByTheme'>
    <prototype is='DEC_ByTheme'/>
    <prototype is='IDX_NAV_Body_Animals'/>
    <prototype is='NAV_Index'/>

    <property name='Title' value='Animals'/>
    <property name='specImage' value='themes.gif'/>
    <property name='wcml.filename' value='IDX_AnimalsByTheme.html'/>
</component>

<component uuid='IDX_ByTheme'>
    <prototype is='DEC_ByTheme'/>
    <prototype is='IDX_NAV_Body_Themes'/>
    <prototype is='NAV_Index1'/>

    <property name='specImage' value='themes.gif'/>
    <property name='wcml.filename' value='IDX_ByTheme.html'/>
</component>

<component uuid='IDX_main'>
    <prototype is='DEC_main'/>
    <prototype is='IDX_NAV_Body_main'/>

    <property name='specImage' value='hp_head.gif'/>
    <property name='wcml.filename' value='IDX_main.html'/>
</component>
```

Figura 6: Definición de las componentes del menú principal y del índice del contexto “obras por tema”

Menú de acceso principal

Indice del contexto “obras por tema”

Indice de obras relacionadas con animales

Obra titulada “Menino com Estilingue”

Figura 7: Ejemplo de navegación de las obras por tema

Luego de finalizadas las definiciones de las componentes que implementan nuestro modelo, procedemos a la compilación. Dado que el ejemplo presentado en este trabajo se basa en la generación de documentos HTML, del resultado de la compilación obtenemos el script correspondiente para cada documento. La figura 7 muestra un ejemplo de navegación dentro del contexto "obras por tema", los documentos fueron obtenidos a partir de las especificaciones que muestran las figuras 4 a 6.

6 Conclusiones

En este trabajo hemos presentado las experiencias y los resultados obtenidos con el uso de la metodología OOHDM y el lenguaje WCML para el diseño e implementación de una aplicación web del estilo hypermedia.

Hemos demostrado que a partir de un modelo orientado a objetos, obtenido con la metodología OOHDM, se puede preservar su estructura de diseño con el uso de WCML en la etapa de implementación. Esta característica esta establecida por la similitud existente entre el modelo de diseño que provee OOHDM y el uso de componentes en WCML. La ventaja que nos provee este tipo de implementación es que cualquier cambio realizado en el modelo de nuestra aplicación podrá ser fácilmente implementado sin necesidad de realizar grandes cambios en el código. Además, dado que las componentes que forman la aplicación son totalmente rehusables, la definición de nuevos contextos navegacionales es una tarea bastante fácil al igual que su mantenimiento.

Para la implementación de nuestro pequeño ejemplo hemos usado una estrategia de mapeo de un modelo OOHDM a una implementación en WCML. Esta estrategia comienza con la definición de diferentes esquemas navegacionales y finaliza con la especificación de herencia y composición de componentes definidas en WCML. De la misma manera, esta idea podría ser utilizada para otros tipos de diseño y diferentes niveles de implementación dando origen a una serie de herramientas CASE para el desarrollo de aplicaciones hypermedia.

Por otro lado, si vemos a la estrategia de mapeo como una metodología de implementación para un modelo orientado a objetos, estamos frente a un patrón de diseño. De la misma forma podríamos definir diferentes estrategias similares para implementar diversos tipos de aplicaciones orientadas a objetos en WCML y así dar origen a una serie de patrones de diseño.

7 Referencias

[Gaedke99] M. Gaedke, D. Schempf, H. –W. Gellersen; *WCML: An enabling technology for the reuse in object-oriented Web Engineering;* Poster-Proceedings of the 8th International World-Wide Web Conference (WWW8), Toronto, Ontario, Canada, 1999

[Gellersen97] Hans-Werner Gellersen, Robert Wicke, Martin Gaedke; *WebComposition: an object-oriented support system for the Web Engineering Lifecycle;* Computer Networks and ISDN Systems 29, Special Issue on the 6th International World-Wide-Web Conference, Santa Clara, USA, 1997

[Portinari] *Portinari Project;* http://www.portinari.org.br

[Rossi99] Gustavo Rossi, Fernando Lyardet, Daniel Schwabe; *Web application models are views on conceptual models;* submitted to WWWCM99; 1999

[Schwabe95] Daniel Schwabe, Gustavo Rossi; *The OOHDM;*Comm ACM, August 1995

[Schwabe96] Daniel Schwabe, Gustavo Rossi, S. *Barbosa; Systematic Hypermedia Design with OODHM*; Proceedings of the ACM International Conference on Hypertext (HT'96), Washington, USA, 1996

[Schwabe98] Daniel Schwabe, Gustavo Rossi; *An Object Oriented Approach to Web-Based Application Design*; Theory and Practice of Object Systems (TAPOS), Wiley and Sons, October 1998

[W3CH98] World-Wide-Web Consortium; *HTML: Hypertext Markup Language;* http://www.w3c.org/MarkUp/, 1998

[W3CX98] World-Wide-Web Consortium; *XML: eXtensible Markup Language;* http://www.w3c.org/XML/, 1998