# *Nuevas características del lenguaje OPScript para la creación de aplicaciones educativas*

**Vicente Arturo Romero Zaldivar, Jon Ander Elorriaga Arandia, Mateo Gerónimo Lezcano Brito**

Universidad del País Vasco
Apdo. 649 P.K. - 20080 Donostia - San Sebastián, España
vicenterz@yahoo.es, jon.elorriaga@ehu.es

Universidad Central de Las Villas
Carretera a Camajuaní Km 5, Santa Clara, Cuba
mlezcano@uclv.edu.cu

**Resumen**: Se presenta una breve introducción al proyecto YADBrowser, el cual incluye el navegador educativo YADBrowser y su lenguaje OPScript. Se exponen algunas características recientemente añadidas al lenguaje; con ellas el autor de una aplicación educativa puede lograr más funcionalidad y adaptabilidad con menos código. Además el navegador YADBrowser reduce la interacción con el servidor, disminuyendo de esta forma el tiempo de respuesta de la aplicación. También puede mantener un registro de las acciones y preferencias del usuario durante la sesión. Las nuevas características fueron añadidas para facilitar la creación de aplicaciones dinámicas, adaptables a las habilidades del estudiante. Estas incluyen comunicación "verbal" entre objetos, modelos de objetos basados en XML y métodos reutilizables. También se exponen algunas características que pueden ser de importancia en el desarrollo de aplicaciones educativas con un componente matemático.
**Palabras clave**: aplicaciones educativas, lenguajes de programación, adaptación, modelos de objetos, comunicación verbal.

**Abstract:** In this paper the authors make a brief introduction to the YADBrowser project which includes the educational browser YADBrowser and its language, OPScript. The authors show some features added to it recently. Using them the author of an educational application can achieve more functionality and adaptation with less code. Also, the YADBrowser reduces the interaction with the server, decreasing this way the application response time. In addition it can keep a record of the actions and preferences of the user during the session. These new features were added to facilitate the creation of dynamic applications, adaptable to student skills. They include "verbal" communication between objects, XML object models and reusable methods. Some features valuable in the development of educational applications with a mathematical component are also presented.
**Key words:** educational applications, programming languages, adaptation, object models, verbal communication.

## 1. Introducción

Internet, se ha convertido en uno de los medios de difusión más importantes creados por el hombre La Web, surgida más tarde, sentó las bases para la proliferación de una gran cantidad de aplicaciones basadas en varias de las tecnologías existentes hasta el momento para Internet. Hoy en día, se considera la Web como un campo natural para la publicación de aplicaciones educativas [Montessoro et al. 03].

 

A pesar de las bondades de las tecnologías actuales, existen problemas inherentes que dificultan la creación de aplicaciones educativas, por ejemplo: el protocolo HTTP no mantiene el estado, es necesario usar sesiones, es incomodo tener que generar todo el contenido cada vez que se envía una página al usuario, falta información semántica en la Web, etc. Actualmente, se han realizado y se realizan muchos trabajos con el propósito de resolver estos problemas; en particular, en el área educativa, existe gran interés en la creación de aplicaciones que se adapten a las habilidades y conocimientos de los estudiantes [Atif et al. 03] y [Manouselis et al. 02].

Algunos autores han abordado la solución de algunos de estos problemas creando nuevos navegadores y/o modificando la forma de navegación. El navegador Grendel [Dennis et al. 97] resulta un ejemplo interesante de un navegador adaptable, ya que permite, entre otras cosas, modificar (por programación) la interfaz del navegador, las interacciones con comandos HTTP, la forma en que se muestran los documentos, etc. Existen además informes relacionados con otros aspectos, tales como la navegación colaborativa, que se incluyen dentro del paradigma de aprendizaje colaborativo. Muchos autores han concluido que este tipo de navegación es más provechosa que la navegación en solitario, por tanto se han reportado herramientas para este tipo de navegación [Hoyos-Rivera et al. 03]. Otra tendencia en cuanto al desarrollo de navegadores es la creación de aplicaciones que se adapten a los recursos de la máquina donde estén instaladas, ancho de banda, etc. [Henricksen et al. 01].

Es importante tener en cuenta que la solución de cada problema relativo a la creación, distribución, etc., de aplicaciones educativas implica la creación de nuevos lenguajes de programación y/o nuevas tecnologías. En el caso particular de los lenguajes de programación se puede plantear que todo lenguaje dirigido a realizar aplicaciones educativas debe ser poderoso y al mismo tiempo fácil de emplear. La generación automática y la reutilización de código son también características deseables para un lenguaje de este tipo. De esta forma se podrían agregar nuevas funcionalidades a una aplicación con menos trabajo. No es posible, en el caso de las aplicaciones Web, separar el lenguaje en que éstas se crean, de la aplicación en la cual se visualizan, en la mayoría de los casos un navegador Web. En este sentido existen trabajos relacionados con el desarrollo de nuevos navegadores y lenguajes con objetivos educativos, [Huang et al. 01].

Existe otra tendencia de importancia para este trabajo: el desarrollo de frameworks que faciliten la creación de aplicaciones educativas empleando las tecnologías actuales. En este campo se encuentra por ejemplo la arquitectura Avante [Theoktisto et al. 03] y el framework MTeach, [Montessoro et al. 03]. Algunos de los objetivos de los creadores de estos frameworks pueden incluirse entre los objetivos que persiguen los autores con el navegador YADBrowser y el lenguaje OPScript. Generalmente estos frameworks son creados para facilitar el trabajo de los autores de aplicaciones y para suplir funcionalidades que no existen en las tecnologías actuales o, si existen, no son fáciles de emplear. Podríamos citar como otro ejemplo el framework FACT [Mora et al. 02]. Una de las funcionalidades que incluye es la posibilidad de modificar, de forma dinámica, contenidos ya vistos por el estudiante en caso de respuestas incorrectas. Estas modificaciones podrían incluir: añadir más explicaciones, proponer ejercicios más simples sobre un tema complejo, etc.

La principal diferencia que tiene desarrollar una aplicación educativa con OPScript en lugar de con uno de los frameworks mencionados es que en el primer caso se obtiene mucha más flexibilidad y generalidad. En el caso del framework MTeach, éste fue creado para un tipo de aplicaciones educativas específico: aplicaciones que hagan un uso intensivo del video interactivo como medio de comunicación fundamental. Este framework permite crear este tipo de aplicaciones con poco esfuerzo y cuenta además con un lenguaje de programación, pero con características muy limitadas y específicas para este tipo de aplicaciones. En el caso de FACT posee características de importancia para el desarrollo de aplicaciones educativas pero no posee un lenguaje con el cual el desarrollador de una aplicación pueda agregar nuevas funcionalidades a las ya existentes. Luego no se quiere decir que se esté proponiendo una variante mejor a todos estos frameworks, cada uno tiene sus virtudes en su campo de aplicación. Simplemente se está resaltando la posibilidad que ofrece el lenguaje OPScript, como se verá más delante, de poder ser modificado y aumentado para

adaptarlo mejor a las necesidades de una aplicación determinada.

Además de las tendencias mencionadas, también hay gran interés actualmente en la reutilización de componentes de lecciones y courseware, [Boyle et al. 01] y [Boyle 03]. Debido a esto, el lenguaje OPScript posee varias construcciones que facilitan la reutilización de componentes de lecciones, de objetos, etc. En este sentido, con OPScript además de la reutilización de componentes de lecciones, se permite la reutilización de métodos, elevando las posibilidades a niveles superiores a lo que ofrecen los trabajos mencionados.

El navegador YADBrowser y su lenguaje OPScript fueron creados con el objetivo de solucionar la mayor parte de los problemas inherentes a las aplicaciones Web mencionados hasta el momento. Así, el propósito consiste en facilitar el desarrollo de aplicaciones educativas para Internet que tengan una serie de prestaciones que son deseables tanto por los autores de éstas como por los estudiantes. Además, los autores tienen la intención de crear con este proyecto un ambiente de desarrollo que permita elevar a un nivel superior el proceso de creación de aplicaciones Web y en particular las aplicaciones educativas. En este sentido se está trabajando en la actualidad en la creación de un entorno de desarrollo que permita la compilación del código, la depuración de los errores y la generación automatizada de código, entre otras facilidades.

Otros desarrollos en este proyecto se dirigen a lograr que el navegador y el lenguaje se adapten a los requerimientos de los autores. En este sentido se está trabajando en aspectos que hagan el lenguaje flexible, que se pueda ampliar con relaciones y funcionalidades nuevas. En este sentido se está trabajando, entre otras características, en la sobrecarga de operadores, la definición de nuevos tipos de datos primitivos, etc. Sobre esto se habla en la sección 6. Este aspecto es muy novedoso en lenguajes para la Web. Además la implementación que se ha hecho en OPScript tiene características novedosas cuando se compara con las hechas en lenguajes que tradicionalmente permiten estas facilidades como C++.

Esta característica no se debe menospreciar ya que implica que el lenguaje sea aumentado con tipos de datos específicos de una aplicación dada, facilitando el posterior desarrollo de ésta, reduciendo la cantidad de líneas de código a generar, etc. En este artículo se recalca la importancia de la definición de nuevos tipos de datos primitivos en aplicaciones dedicadas a la enseñanza de las matemáticas pero podría haber otros muchos campos de aplicación.

Actualmente uno de los lenguajes más utilizados del lado del cliente en aplicaciones Web es JavaScript. Pero este lenguaje tiene limitaciones para la reutilización y la persistencia del lado del cliente de objetos, código, etc.

En el artículo se hará referencia a una aplicación educativa denominada CARDIODEF que ha sido desarrollada para ser visualizada en el YADBrowser. Esta permite enseñar a estudiantes de medicina y doctores, a trabajar con un desfibrilador creado por el Instituto Central de Investigaciones Digitales (ICID), centro de investigaciones cubano que cuenta con una larga experiencia en la fabricación de equipos médicos. Este equipo se emplea para dar descargas en el corazón a personas con cardiopatías, ataques cardíacos, etc.

El artículo está organizado en 9 secciones. En la sección 2 se hace una breve introducción al navegador educativo YADBrowser y a su lenguaje OPScript. En la sección 3 se comentan las funcionalidades incluidas en el lenguaje OPScript para el trabajo con metadatos y se introduce el concepto de métodos reutilizables. En la sección 4 se introduce la comunicación verbal entre objetos. En la sección 5 se expone la posibilidad de cargar modelos de objetos incluidos en ficheros XML. En la sección 6 se comenta sobre la definición de nuevos tipos de datos. En la sección 7 se exponen las relaciones existentes entre las funcionalidades expuestas y en la 8 se resalta la importancia de las nuevas características del lenguaje desde el punto de vista educativo. Finalmente las conclusiones generales aparecen en la sección 9.

## 2. Introducción al Lenguaje OPScript y al YADBrowser

OPScript, [Romero et al. 04 a y b] y [Romero et al. 05], es el lenguaje del navegador educativo YADBrowser que ha sido desarrollado con el objetivo de crear una aplicación adecuada para visualizar software educativo, con un lenguaje que permita desarrollar rápidamente las aplicaciones, además de otras facilidades.

Inicialmente el YADBrowser fue provisto con características orientadas al ahorro de ancho de banda, aspecto muy importante para algunos autores [Uehara et al. 01] y [Xiao et al. 02, 04]. Esto ha permitido además la reutilización de componentes de lecciones, código, etc. Actualmente este aspecto sigue siendo de mucho interés para los autores de aplicaciones educativas, el motivo principal de esto es el ahorro de tiempo y esfuerzo. Otro aspecto de interés en la literatura es el desarrollo de modelos de objetos para aplicaciones Web; con el objetivo de proveer funcionalidades inexistentes en los navegadores actuales [Hennen et al. 00]. Por esto en el YADBrowser se ha incluido un modelo de objetos orientado a la creación de aplicaciones educativas.

Además se ha observado en la literatura consultada gran interés en herramientas que permitan la creación de aplicaciones que se adapten a las habilidades y preferencias del usuario [Montessoro et al. 03]. Debido a esto se han incluido nuevas etiquetas en el lenguaje de marcado del navegador que faciliten la creación de aplicaciones adaptables además de nuevas construcciones en el lenguaje OPScript con el mismo objetivo.

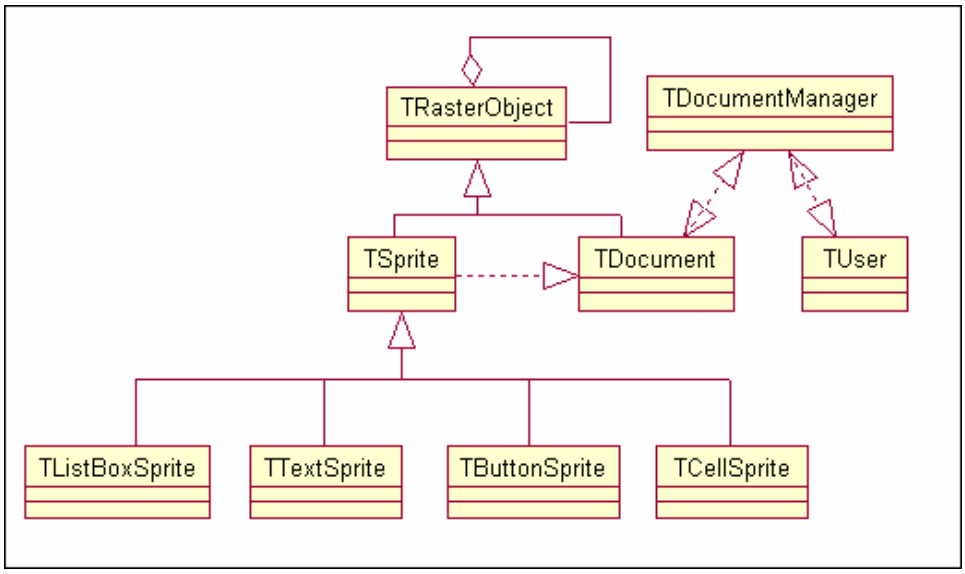


**Figura 1.** Diagrama con algunas de las clases del modelo de objetos del YADBrowser

En la Figura 1 se muestra un diagrama con algunas de las clases que integran el modelo de objetos del YADBrowser. En este modelo existen clases que tienen uso en aplicaciones generales pero que su principal objetivo es ser empleadas en aplicaciones educativas. Por ejemplo la clase *TCellSprite* se emplea en la creación de ejercicios de respuesta múltiple, crucigramas, etc. Un miembro importante del modelo de objetos del YADBrowser es el objeto *User*, que está disponible en cualquier página. Este objeto contiene información relacionada con el usuario, tal como páginas visitadas, evaluación obtenida en la solución de ejercicios, etc. Además posee métodos para agregar información que es específica de una aplicación y otros que permiten su posterior acceso. Estos datos pueden ser útiles para decidir qué mostrar y cómo hacerlo, teniendo en cuenta las preferencias del usuario. Por ejemplo, se podrían modificar enlaces entre documentos dependiendo del conocimiento de un usuario sobre un tema, etc. Además se puede reducir el número de interacciones con el servidor porque esta información está siempre presente en el navegador, ofreciéndose de esta forma una solución al problema del no

mantenimiento del estado por parte del protocolo HTTP. En la Figura 2 se muestra una página de la aplicación CARDIODEF que se mencionó en la sección 1. Un detalle interesante en esta figura es el objeto que permite visualizar en tiempo real los latidos del corazón de un paciente virtual, este es un objeto muy útil dentro del modelo de objetos del YADBrowser.

Con el objetivo de lograr la creación de aplicaciones más adaptables y de ahorrar interacciones con el servidor se emplean adiciones persistentes de código, objetos, etc. Una adición persistente de código permite agregar nuevos miembros, es decir campos y métodos a clases del lenguaje. Esto permite obtener de forma muy simple toda una serie de ventajas, además de las ya mencionadas, las adiciones persistentes permiten que la creación de una aplicación educativa con el YADBrowser y el lenguaje OPScript no se diferencie mucho de la creación de una aplicación de escritorio. Esto se logra mediante una abstracción sobre las complejidades de una aplicación Web, tales como el manejo de sesiones, etc. Por ejemplo, lo que se necesita mantener como un dato asociado al usuario se puede definir como uno o varios campos añadidos a una clase del lenguaje mediante una adición persistente. La posibilidad de almacenar información en la memoria del navegador tiene también otras ventajas: permiten mantener un estado de la aplicación del lado del cliente, facilitan la programación porque el código

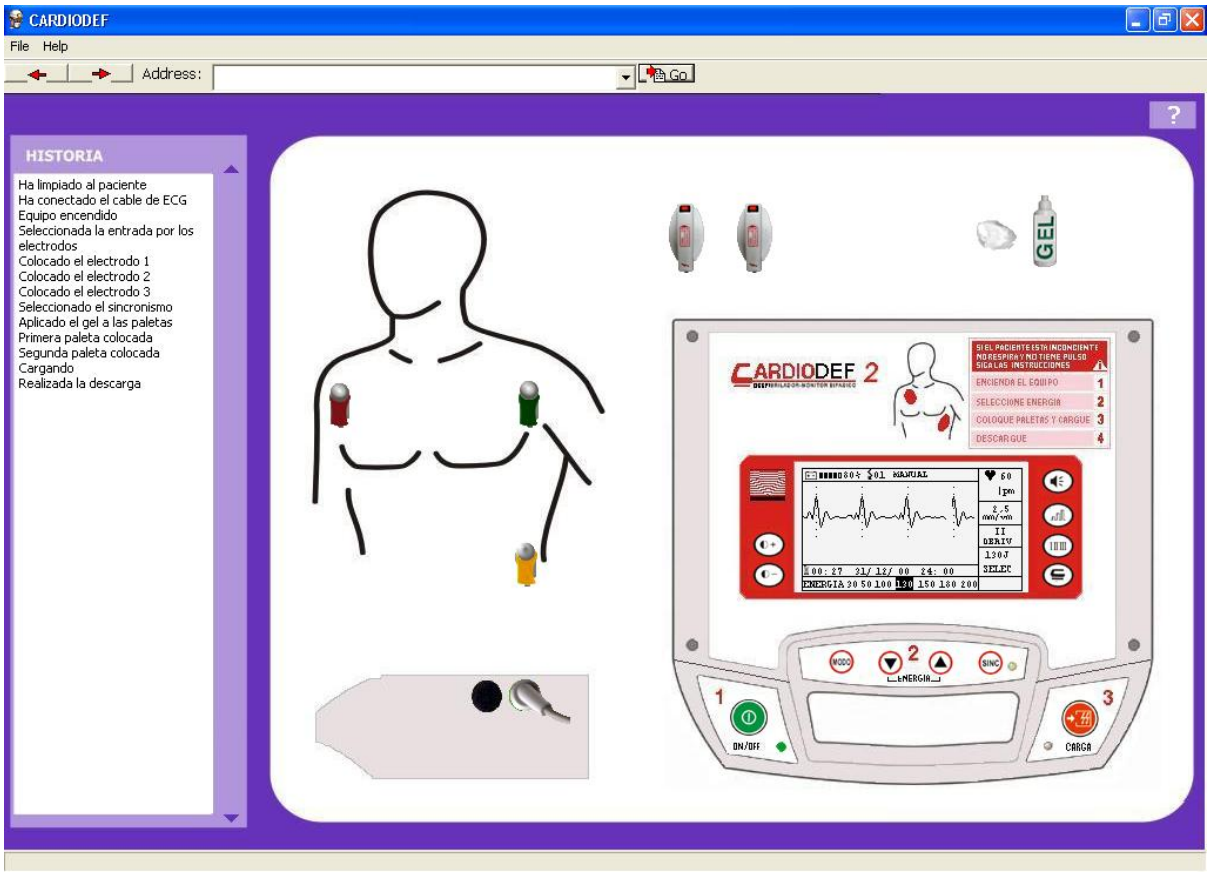


**Figura 2.** Pantalla de una aplicación creada con OPScript para el YADBrowser.

común a varias páginas se puede ubicar en un solo lugar, disminuyendo de esta forma el esfuerzo de los desarrolladores y evitando inconsistencias dentro de la aplicación, entre otras. Estas adiciones permiten modificar no sólo clases definidas como parte de una aplicación específica sino también clases básicas del lenguaje. Esta última modificación implica cambios en la aplicación que las produce además de modificaciones en los procesos básicos que vienen definidos como parte del lenguaje; permitiendo la

adaptación del OPScript a los requerimientos de una aplicación específica. Estas modificaciones no son permanentes y cuando termina la aplicación todo vuelve a su estado inicial, pero esta posibilidad ofrece muchas ventajas para los creadores de aplicaciones y para la capacidad de adaptación que se puede ofrecer a los estudiantes. En la Figura 3 se muestra un ejemplo de cómo se puede definir una adición a una clase en OPScript. En este caso se hace una adición a la clase *Object*, de la cual heredan todos los objetos del lenguaje, luego a partir del momento en que se encuentre todos los objetos tendrán los métodos añadidos.

```
TObjectAdd=addition(Object)
methods
 procedure ExecuteVerb(string aVerb);
 begin
  VerbManager.ExecuteVerb(Self,aVerb);
 end;
 function GetIntegerFieldVal(string MetaData):integer;
 begin
  result:=MetaDataManager.GetIntegerField(Self,MetaData);
 end;
end;
```

**Figura 3.** Una adición persistente de código en OPScript

## 3. Metadatos y Métodos Reutilizables

Los metadatos han probado ser muy útiles en los lenguajes de programación, en el caso particular de lenguajes orientados a aspectos han probado ser de gran utilidad [Lieberherr, 04], [Shonle, et al. 03]. Los metadatos permiten añadir metainformación a campos, clases, métodos, etc. Un campo modificado por algún metadato específico se puede obtener mediante funciones del lenguaje, y un objeto, dato, etc., admite un tratamiento diferente a dependiendo de los metadatos que se le hayan aplicado. Un metadato en OPScript, se puede definir entre corchetes delante del elemento al cual se va a aplicar, según se observa en el siguiente fragmento de código:

```
TStudent =
  class
    fields
      string [level] fKnowledgeLevel;
    ...
  end;
```

Si más tarde se necesita acceder a un campo al cual se ha aplicado un metadato, se puede usar un conjunto de funciones presentes en cada objeto. En OPScript los metadatos son el soporte principal del concepto de métodos reutilizables. Un método reutilizable es aquel que puede emplearse por objetos de diferentes clases sin importar en qué clase se definió originalmente. Teniendo en cuenta el segmento de código mostrado anteriormente el campo modificado por el metadato *level* podría ser accedido dentro de un método reutilizable de la siguiente forma:

```
var
  string s;
begin
  ...
  s := GetStringFieldVal('level');
  ...
end;
```

La función empleada arriba para acceder al campo se aplica al objeto para el cual se está ejecutando el método sin importar su clase y permite extraer el valor del campo al cual se haya aplicado el metadato *level*. Cuando se invoca un método reutilizable, se ejecuta como si fuera un método incluido en la clase del objeto con cuyos datos se está ejecutando el método en un momento dado. Para hacer que un método sea reutilizable, éste no puede acceder a los campos de los objetos de forma directa, como se hace en todos los lenguajes de programación, sino usando metadatos. Los métodos reutilizables disminuyen la cantidad de código a generar para la aplicación,

haciéndola más simple y son un importante componente del nuevo tipo de relación entre objetos que será expuesta más abajo.

## 4. Comunicación Verbal entre Objetos

La comunicación verbal entre objetos es una relación desconectada en la cual no se especifica de forma explícita cuáles son sus extremos, ni se define qué interfaz deben implementar las clases de los objetos para su establecimiento. Se dice que la relación es desconectada porque en un momento dado un objeto que denominaremos O1 puede interactuar con otro objeto que llamaremos O2, pero esto no implica una modificación en la definición de ninguno de los dos objetos ni una dependencia entre ellos. En otro instante de tiempo O1 puede interactuar con cualquier otro objeto sin importar que antes lo hubiera hecho con O2. En este sentido, también se podría denominar relación dinámica entre objetos porque esta relación permite que la estructura topológica de una aplicación cambie dinámicamente. En adelante en este artículo se referirá también a esta relación como relación verbal.

La relación se crea automáticamente cada vez que es necesario. Esto permite que la creación de una aplicación Web pueda reducirse, al menos una parte importante del proceso, a la inclusión de un grupo de objetos en varias páginas para luego dejar que el navegador establezca las relaciones entre ellos de forma automática. Esto puede reducir gran parte del trabajo, permitiendo que personas con pocos conocimientos técnicos puedan desarrollar aplicaciones sencillas o prototipos de aplicaciones complejas. Con la comunicación verbal, dos objetos pueden seguir interactuando aunque las interfaces de uno o de ambos cambien o, incluso, en el caso de que cualquiera de ellos cambie de forma casi total. La comunicación verbal es importante porque acelera la programación, al no tener que especificar con detalle todas las posibles interacciones entre objetos y facilita el desarrollo de aplicaciones útiles y reales. El siguiente fragmento de código muestra cómo funciona la comunicación verbal en OPScript:

```
TExercise = class
needs Play;//verbo que necesita
fields
  string [media] fPath;
methods
procedure PlayMedia();
begin
  ExecuteVerb('Play');
end;
  ...
end;

TPlayer = class
methods
procedure DoPlay();
var
  string Path;
begin
Path:=GetStringFieldVal('media');
...
end;
  offers Play(DoPlay);//verbo ofrecido
end;

TAnyClass = class
fields
  TExercise Ex1;
  ...
methods
procedure PlaySound();
begin
  Ex1.PlayMedia();
end;
end;
```

Esta sección de código presenta, mediante un ejemplo hipotético, cómo pueden comunicarse dos objetos usando comunicación verbal. Véase el empleo de las palabras claves *needs* y *offers*, la clase *TExercise* necesita interactuar con una clase que ofrezca el verbo *Play*, puesto que la clase *TPlayer* lo ofrece, si en un momento dado existen dos objetos, uno de cada clase, la relación se establecerá de forma automática cuando se ejecute el método *PlayMedia* de la clase *TExercise*. En la sección *offers* después del verbo y entre paréntesis debe aparecer el nombre de un método, que se ejecutará cuando se produzca la interacción, como es de suponer este método debe ser reutilizable.

Se ha mencionado varias veces el término verbo que se usa tanto en la sección *needs* como en la *offers* y como se podrá suponer es lo que ha dado nombre a este tipo de relación. Los verbos no son más que identificadores que permiten definir o denotar los extremos de una relación verbal en un momento dado. Se debe observar, además, que el hecho de que exista uno de los extremos de la relación no implica

que exista el otro y, por tanto, que la relación sea completa; así, es importante no sólo la existencia de los extremos de la relación sino también el instante de tiempo en que están disponibles.

El método ofrecido por la clase *TPlayer* accede a la información contenida en el objeto con el cual se llama, en el ejemplo *Ex1*, mediante metadatos. Esto facilita la comunicación con objetos de clases distintas, lo que significa que el método ofrecido podría usarse por cualquier objeto que necesite reproducir un sonido. Finalmente, nótese que el método *ExecuteVerb*, presente en cada objeto, es quien hace que se establezca la relación con un objeto que ofrece un verbo dado. Es importante resaltar que en el modelo de objetos del YADBrowser podría ser incluida cualquier clase dinámicamente, en respuesta a un evento generado por el usuario, a la descarga de una página, etc. Cada vez que se añade una nueva clase al modelo de objetos, el navegador actualiza, de forma automática, sus listas de clases que ofrecen o que necesitan verbos. De esta forma los extremos de una relación dada pueden cambiar dinámicamente.

Debido a la comunicación verbal un objeto dado puede interactuar fácilmente con más de un objeto durante el tiempo de vida de la aplicación, sin siquiera notarlo, dependiendo de los cambios que surjan en la aplicación. Esto puede facilitar la adaptación de una aplicación educativa a las habilidades y conocimientos de un estudiante dado, porque un objeto determinado puede interactuar sin ningún cambio con objetos que poseen diferente información o comportamiento.

Además de las ventajas antes mencionadas, se puede agregar que la comunicación verbal reduce el tiempo de desarrollo y la complejidad de una aplicación. Es conocido que el empleo de componentes de software puede reducir considerablemente el tiempo de desarrollo de cualquier aplicación y su complejidad. Entonces, si los componentes emplean además comunicación verbal, los objetos pueden predefinir cómo se comunican entre ellos. Por lo tanto, estos objetos podrían emplearse con mayor facilidad sin que sus usuarios tengan que conocer los detalles de la forma en que se comunican, los parámetros que hay que pasar para llamar a un método dado o qué hacer con los datos resultantes. Este tipo de comunicación debe seguir evolucionando pero en su estado actual ofrece ya muchas facilidades que la hacen provechosa en el desarrollo de aplicaciones educativas. Supóngase, por ejemplo, que un desarrollador posee un amplio conjunto de objetos, desarrollado por programadores experimentados y que los objetos se puedan comunicar entre ellos empleando verbos; el trabajo se optimiza de esta forma porque la comunicación verbal reduce el tiempo necesario para crear las relaciones entre los objetos del modelo y la cantidad de errores que puedan aparecer. Como un efecto colateral, los objetos que se comunican con otros objetos mediante comunicación verbal son más simples y pueden ser reutilizados más fácilmente que los que no lo hacen.

## 5. Modelos de Objetos y XML

El lenguaje OPScript posee objetos que permiten extraer modelos de objetos contenidos en archivos XML. El lenguaje XML tiene muchas propiedades que lo hacen adecuado para la representación de modelos de objetos: es un lenguaje bastante conocido, posee además una representación jerárquica intrínseca, entre otras características. La estructura jerárquica de XML permite reducir la cantidad de declaraciones necesarias para expresar las interrelaciones y las propiedades de un modelo de objetos dado. Al extraer un modelo de objetos de un archivo XML el navegador YADBrowser sigue las siguientes convenciones:

- Cada nodo XML debe contener un atributo nombrado “id” que será el nombre del objeto correspondiente cuando se cargue el modelo. Si algún nodo XML incluye una propiedad cuyo valor es el “id” de otro nodo, se interpretará como una relación existente entre ambos, la cual se representará en el modelo de objetos resultante. Esto significa que la estructura topológica del modelo es construida de forma automática.
- Cada nodo XML se convierte en un objeto del lenguaje, cada atributo del nodo será un campo del objeto con el valor y el tipo del atributo, de esta forma es prácticamente innecesario implementar de forma explícita los constructores.
- En caso de ser necesario, un nodo puede incluir una etiqueta denominado *methods* la cual puede contener código OPScript que se agregará a la clase correspondiente a ese nodo.

- Para todos los nodos con el mismo nombre se crea una clase del lenguaje que los representa. Cada una de estas clases tiene, si fuera necesario, una lista para contener otros objetos. Todas las listas poseen métodos para agregar, eliminar, insertar, etc., cualquier objeto. Como el proceso se hace de forma automática en cada clase, se reduce la cantidad de código a ser incluido en el archivo XML.
- Una característica muy importante de estos modelos es que pueden ofrecer verbos, de esta forma; la aplicación se puede comunicar con los objetos del modelo sin tener un conocimiento previo de su estructura.

Los modelos de objetos contenidos en archivos XML pueden agregarse a la aplicación en cualquier momento de acuerdo a las necesidades de ésta. La representación de modelos de objetos en archivos XML tiene muchas ventajas, entre estas tenemos:

- La adición dinámica de modelos de objetos al modelo actual de la aplicación.
- Es preciso generar menos código gracias a las ventajas del lenguaje XML mencionadas anteriormente.
- Permiten el desarrollo de modelos complicados de manera conjunta. La persona que posee el conocimiento sobre el área específica que representa el modelo de objetos puede definir su estructura topológica, las dependencias etc., y por su parte los programadores pueden agregarle luego comportamiento.

En la Figura 4 se muestra un modelo de objetos definido en un fichero XML que fue creado para la aplicación CARDIODEF. Define un grafo que representa los pasos que se deben seguir para usar el desfibrilador, en el caso en que se deba tratar a un paciente en estado grave.

Para cargar un modelo desde un archivo XML dinámicamente se puede usar el código siguiente:

```
var
  Object Graph;
begin
Graph=XMLModel.LoadModel('Graph.xml');
...;
end;
```

El modelo se añade a la aplicación dinámicamente y se emplea para determinar si el usuario está siguiendo los pasos correctos para un funcionamiento apropiado del equipo. En la figura se puede observar la variable *fItemsList* que es una lista añadida automáticamente por el navegador. Existen otras, que no aparecen en el ejemplo, como *fParent*; éste es un campo presente en cada objeto para acceder al objeto que lo contiene. Estas variables son creadas automáticamente por el navegador YADBrowser cuando carga un modelo desde un archivo XML. El objetivo del proceso descrito es reducir la cantidad de código necesario para definir el modelo, logrando que el desarrollador necesite menos esfuerzo para su definición, pudiendo referirse a estas variables que estarán disponibles cuando se cargue el modelo. El modelo mostrado se emplea para evaluar el desarrollo de ejercicios prácticos por parte del estudiante. Se puede determinar cuándo este se equivoca en algún paso y se puede tomar la decisión de corregirlo en el momento o indicarle al final qué hizo incorrectamente. En lugar de este modelo se pueden agregar otros diferentes según la cardiopatía que se quiera tratar, el nivel de conocimientos del estudiante, etc. Esta funcionalidad del leguaje permite que se pueda crear una aplicación en la cual, en un momento dado, coexistan en una misma página varios modelos cargados dinámicamente desde archivos XML. Estos modelos podrían interactuar por medio de comunicación verbal: aún cuando estuvieran desarrollados para aplicaciones diferentes podrían coexistir e interactuar sin problemas en una aplicación dada.

Este tipo de modelos pudieran representar sin ningún problema objetos de aprendizaje que además de la información que naturalmente poseen, ofrecieran verbos para comunicarse con cualquier aplicación, sin necesidad de establecer una interfaz rígida para lograr la interacción.

```
<graph id='Root' CurrentNode='Start'>
  <methods>
      procedure ProcessEvent();
      var
        integer i, event, terminate;
        Node Current; Link tmpLink;
      begin
        event := GetIntegerFieldVal('eventdata');
        Current := this.CurrentNode;
        i := 0; terminate := 0;
        while (i < Current.fItemsList.Count()) and (terminate = 0) do begin
            tmpLink := Current.fItemsList.Objects(i);
            if tmpLink.Event = event
              then begin
                  terminate := 1; this.CurrentNode := tmpLink.Node;
              end
              else i := i + 1;
        end;
        if i = Current.fItemsList.Count()
          then SetIntegerFieldVal('eventvalid', 0)
          else SetIntegerFieldVal('eventvalid', 1);
      end;
  </methods>
  <verb name='ChangeEvent' offers='true' method='ProcessEvent' />
  <node id='Start'>
    <link id='LStartOn' node='On' event='1' />
  </node>
  <node id='On'>
    <link id='LOnPulsoPaletas' node='PulsoPaletas' event='10' />
  </node>
  <node id='PulsoPaletas'>
    <link id='LPulsoPaletasEnergia' node='Energia' event='3' />
  </node>
  <node id='Energia'>
    <link id='LEnergiaPaleta1' node='paleta1' event='6' />
  </node>
  <node id='Paleta1'>
    <link id='LPaleta1Paleta2' node='paleta2' event='7' />
  </node>
  <node id='Paleta2'>
    <link id='LPaleta2Carga' node='Carga' event='8' />
  </node>
  <node id='Carga'>
    <link id='LCargaDescarga' node='Descarga' event='13' />
  </node>
  <node id='Descarga'>
    <link id='LDescargaEnergia' node='Energia' event='3' />
  </node>
  <node id='Off' />
</graph>
```

**Figura 4.** Ejemplo de modelo de objetos representado en un fichero XML

## 6. Definición de Tipos de Datos Básicos en OPScript

Las características que se exponen a continuación pueden simplificar el desarrollo de cierta gama de aplicaciones educativas. Podríamos mencionar entre estas las que tengan un componente matemático, ya sea porque deban hacer cálculos con ciertos conjuntos numéricos o porque su principal objetivo sea la enseñanza de alguna rama de las matemáticas. La base de esta funcionalidad radica en extensiones al lenguaje. Uno de los primeros lenguajes que permitió ser extendido por el programador fue el C++ [Stroustrup 97], que permite la sobrecarga de operadores. Se entiende por sobrecarga de operadores a dar otro significado semántico a los operadores ya definidos por el lenguaje. Esto da la posibilidad de definir, por ejemplo, una clase *fraction* y sobrecargar los operadores de suma, resta, etc., para poder sumar, restar, multiplicar, etc., fracciones de la misma forma que se haría con los tipos de datos predefinidos por el lenguaje como el tipo entero.

La importancia de este hecho es muy grande porque una vez que se haya definido una clase como *fraction* entonces muchas personas podrían hacer su trabajo con fracciones de forma mucho más sencilla y natural. Luego desde el punto de vista educativo, esto permitiría crear aplicaciones de forma más rápida, sencilla, con menos errores y con menos conocimientos de programación por parte del autor de la aplicación. Para desarrollar gran parte de la aplicación sólo se necesitarían conocimientos de matemáticas y no de programación. Sin embargo, no tiene mucho sentido que se puedan sobrecargar los operadores si al definir una nueva fracción, ésta no se puede representar de la manera habitual en ningún lenguaje de programación conocido, como 2/3 por ejemplo. Cualquier lenguaje de programación entendería que en lugar de una fracción se quiere dividir el número 2 entre el 3 y el resultado de esto es un número en coma flotante. Luego la solución para esto es que los lenguajes se puedan extender en un mayor grado según las necesidades de los autores de las aplicaciones.

Ahora se explicará mediante ejemplos cómo se puede hacer esto en OPScript. Se tomarán como caso de estudio las fracciones aunque se podrían hacer desarrollos similares para número complejos, números telefónicos, etc. Lo primero es determinar qué sintaxis va a tener el nuevo tipo con el que vamos a trabajar, en el caso de las fracciones la sintaxis pudiera ser: un número entero, el símbolo "/" y otro número entero. Luego serían fracciones 2/3, -4/5, etc. Sería ideal poder trabajar directamente así pero hay un pequeño problema: se debe colocar delante algún carácter o caracteres que indique al lenguaje que se trata de una fracción y no de la división de un número por otro. Se podría escoger por ejemplo el carácter "f". Ahora una fracción en OPScript se puede escribir como f2/3, f-4/5, etc. Ahora en OPScript para que se pueda entender cualquier fracción con esta forma se debe indicar una expresión regular que las defina de manera general. Veamos cómo puede quedar ahora una clase fracción en OPScript:

```
fraction = class
fields
  integer Numerator, Denominator;
methods
  procedure Register();
  begin
    RegularExp.Pattern('f(#)/(#)');
  end;
  constructor  RegularExpCreate(integer
aNum,integer aDen);
  begin
    Numerator := aNum;
    Denominator := aDen;
  end;
  ...
end;
```

En este segmento de código *Register* es el método que se emplea para indicar al OPScript cuál va a ser la sintaxis de un nuevo tipo de dato. El constructor que aparece en la clase es llamado automáticamente cuando se escriba algo como esto:

```
var
  fraction f1;
begin
  f1 := f2/3;
end;
```

A pesar de lo complejo que pueda parecer llegar a este punto no se deben obviar las ventajas que tiene esta nueva funcionalidad del lenguaje. La definición de nuevas clases como la clase *fraction* podría recaer en programadores expertos, incluso el lenguaje podría incluir una amplia colección de estas clases como *complex* para números complejos, *matrix* para matrices, etc. Pero un usuario del lenguaje podría ahora hacer cosas como estas:

```
var
  fraction f1, f2, f3, f4;
begin
  f1 := f2/3;
  f2 := f4/5;
  f3 := f3/7;
  f4 := f1/3 + f6/11 + f1;
  ...
end;
```

Es decir el lenguaje mediante la definición de construcciones complejas puede ofrecer una mayor sencillez a personas con menos conocimientos técnicos.

La sobrecarga de operadores en OPScript se hace de forma similar a como se hace en C++. Es importante mencionar que de la misma forma que se definió un tipo de datos para trabajar con fracciones se podría haber hecho otro para trabajar directamente con números telefónicos, o con cualquier cosa que se

pueda definir mediante una expresión regular. En este artículo se ha hecho más énfasis en tipos con un significado matemático por considerarlos de mayor interés para las aplicaciones educativas, pero las posibilidades son muy amplias.

## 7. Relación entre las nuevas funcionalidades

Las nuevas funcionalidades del lenguaje OPScript expuestas en este artículo pueden ayudar a los autores a crear aplicaciones educativas adaptables. Ahora se explicará cómo estas funcionalidades interactúan entre ellas.

En el ejemplo del modelo que se muestra en la Figura 4 se puede notar la presencia de una etiqueta *verb*. Este es un verbo ofrecido por la clase *Graph* definida en el archivo XML. Este verbo expone un método *ProcessEvent* el cual busca un nodo válido en el grafo después de la generación de un evento por parte del usuario. Este método es reutilizable y puede ser *prestado* por un objeto de la clase *Graph* al objeto que en un momento determinado interactúe con este. Para esto su contraparte no tiene que conocer qué métodos son ofrecidos por la clase *Graph*; de hecho, cuando se produce la interacción, ni siquiera conoce que está interactuando con un objeto de una clase denominada *Graph*. En un momento determinado, el objeto que en un principio interactuaba con un objeto de la clase *Graph*, podría interactuar con cualquier otro objeto que represente las respuestas válidas a un ejercicio en cualquier forma. De hecho, en la aplicación CARDIODEF es posible interactuar con diferentes modelos parecidos al mostrado que cambian de acuerdo al nivel del estudiante y/o a una cardiopatía previamente seleccionada. Para la comunicación con el modelo se emplea en la aplicación la clase siguiente:

```
TGraphCaller = class
needs ChangeEvent;
fields
  integer [eventdata] Event;
  integer [eventvalid] Valid;
  string [message] HistoryMessage;
methods
procedure DoChangeEvent();
begin
  ExecuteVerb('ChangeEvent');
 end;
end;
```

Obsérvese el empleo de metadatos para la comunicación con el modelo contenido en el fichero XML. Un caso típico de comunicación con el modelo usando un objeto de esta clase se presenta en la Figura 5.

```
if(GraphCaller.event=8)
 then
  begin
   GraphCaller.event := 13;
   GraphCaller.DoChangeEvent();
   if GraphCaller.Valid = 1
    then HistoryMemo.AddItem(GraphCaller.HistoryMessage)
    else Window.Document.HistoryMemo.AddItem('Incorrecto');
  end;
```

**Figura 5.** Comunicación con un modelo definido en un fichero XML

También se podría presentar un ejemplo de modelo que incluyera en su código la definición de alguna clase que, a partir de ese momento, fuera un nuevo tipo de dato, incluyendo de esta forma en un solo cuadro las novedades expuestas en este artículo. La inclusión de un nuevo tipo de datos básico en un modelo de objetos definido en un archivo XML no dista del ejemplo que se ha mostrado, lo importante es la presencia del método *Register* en la etiqueta *methods* de la clase del modelo que define el tipo de datos.

Luego la relación entre las características del lenguaje OPScript expuestas en este artículo se puede ver de

forma completa en un modelo representado en un archivo XML, el cual ofrezca verbos que expongan métodos reutilizables que puedan ser empleados por cualquier clase que necesite de los verbos ofrecidos por el modelo.

## 8. Importancia Educativa

Considerando que los archivos XML pueden bajarse en cualquier momento es posible, empleando las facilidades expuestas, obtener un alto grado de adaptación en aplicaciones educativas. La idea tras esto es que, empleando estas funcionalidades, en una aplicación Web es posible enviar al estudiante una página con información seleccionada y luego, de acuerdo a sus preferencias o a otras características, descargar un modelo de objetos contenido en un archivo XML. Estos modelos interactuarán con el resto de la página de forma transparente.

El modelo añadido puede obtenerse empleando el modelo del estudiante representado en la aplicación del lado del servidor. El ejemplo de modelo de la Figura 4 se emplea para un estudiante principiante. La aplicación es capaz de interactuar con este modelo XML o con un modelo creado para un estudiante experto; su programación, definición, etc., puede ser la misma. Luego la diferencia radicaría en el modelo XML. La técnica que hace posible las interacciones de un grupo de objetos con modelos diferentes es la comunicación verbal entre objetos y los métodos reutilizables. Los resultados serán diferentes de acuerdo al modelo que se añada a la aplicación; por ejemplo en una aplicación real se podría encontrar el código siguiente:

```
if
User.SelectedDisease='Desfibrilacion'
 then Graph=XMLModel.LoadModel
                    ('Asincronic.xml')
 else  Graph=XMLModel.LoadModel
                    ('Sincronic.xml');
```

En este ejemplo se puede ver que el mismo objeto sirve para establecer la comunicación con dos modelos de objetos diferentes. Entre las ventajas de estas funcionalidades para las aplicaciones educativas, además del grado de adaptación y el dinamismo que puede obtenerse, se puede mencionar la separación que se puede establecer en la aplicación entre lo que es común a varios usuarios y lo que puede cambiar. Esto permite que el desarrollo se pueda hacer de forma paralela, facilita la reutilización de componentes, etc. Por ejemplo, en una aplicación educativa se podrían incluir en ficheros XML contenidos opcionales o contenidos más complejos para estudiantes avanzados, y estos se podrían acoplar al resto de la aplicación a petición del estudiante. Por otra parte, en caso de respuestas erróneas se podrían agregar ficheros XML con más ejercicios o explicaciones sobre temas complejos o difíciles para el estudiante. Los modelos de objetos pueden ser generados dinámicamente sin importar prácticamente qué objetos han sido ya enviados al navegador a través de Internet porque la comunicación empleando verbos es muy flexible y ofrece un amplio margen de soluciones para cualquier escenario.

## 9. Conclusiones

Las nuevas características del lenguaje introducidas en este artículo se insertan dentro de los esfuerzos de los autores por crear una plataforma para la creación de aplicaciones educativas. Este proyecto se inició con el objetivo de dar respuestas a varias carencias observadas en las tecnologías existentes para la creación de aplicaciones Web, y en particular de las aplicaciones educativas. Entre estas se puede mencionar el no mantenimiento del estado en el protocolo HTTP lo cual obliga a los autores de aplicaciones a lidiar con las variables de sesión y otras soluciones. En OPScript este problema se resuelve con las adiciones persistentes de código. En relación con esto es necesario mencionar algunas tecnologías actuales como Asynchronous JavaScript And XML (AJAX) en español JavaScript y HTML Asíncronos [Garrett 05]. La importancia de esta tecnología, o más bien grupo de tecnologías, es que permite la interacción de una aplicación Web con el servidor sin que el usuario observe esta interacción, es decir no se ve refrescar la página. La relación con este trabajo es que AJAX permite resolver el problema del mantenimiento del estado del lado del cliente además de que disminuye las latencias en las interacciones Web.

Este esfuerzo indica la importancia de este aspecto al cual los autores han dado particular importancia. De

hecho la nueva característica aparecida en este artículo que permite la carga asíncrona de modelos de objetos desde ficheros XML puede emplearse para lograr el mismo efecto. Pero incluso se puede ir más allá de lo que la tecnología AJAX permite. En primer lugar el lenguaje OPScript permite cargar automáticamente modelos de objetos proceso que se debe hacer manualmente con AJAX. Además hay otro problema con AJAX y es que todo pasa en una misma página que se modifica constantemente cada vez que se hace una interacción con el servidor, luego la complejidad de esta página puede ser grande. Además, si el usuario usa el botón para ir atrás regresa al principio de la aplicación lo cual es un problema. En este sentido los autores consideran que el lenguaje OPScript y el YADBrowser permiten soluciones más generales.

El esfuerzo de AJAX es muy valioso y demuestra la validez de uno de los propósitos de los autores: desarrollar potencialidades en los navegadores, aspecto este un tanto olvidado por los creadores de nuevas tecnologías que casi siempre desarrollan nuevas potencialidades en las aplicaciones que se ejecutan en los servidores. Es claro que existen razones para esto, en particular es más fácil generar cambios en un solo lugar que en todas las máquinas de los usuarios. Pero las aplicaciones educativas necesitan algo más que lo que los navegadores actuales pueden ofrecer. Por ejemplo se necesitan modelos de objetos en los navegadores actuales que sean más apropiados para sus propósitos. Una de las fortalezas del YADBrowser es su modelo de objetos creado para aplicaciones educativas que es un intento por resolver esta carencia.

Las nuevas características añadidas al lenguaje OPScript permiten además un mayor grado de adaptación empleando técnicas del lado del cliente de una aplicación Web. Este es un tema de gran importancia para los autores de aplicaciones educativas dados los beneficios que tienen para los estudiantes ya que permiten acercarse más a un modelo personalizado de enseñanza. Este proyecto podría beneficiarse si se uniera a modelos que puedan lograr la adaptación desde el lado del servidor, como el modelo LAOS [Cristea 03].

Empleando comunicación verbal un objeto dado puede interactuar con diferentes objetos en diferentes momentos durante el tiempo de vida de la aplicación sin necesidad de crear código complejo. Mediante la descarga de modelos de objetos completos, representados en archivos XML, la aplicación puede adaptarse fácilmente a las necesidades, respuestas, etc., del estudiante. El código añadido de forma automática a los modelos de objetos incluidos en archivos XML reduce el esfuerzo necesario para desarrollar la aplicación final. La base sobre la cual funciona todo lo expuesto son los métodos reutilizables y los metadatos, los cuales hacen posible que se puedan compartir métodos entre clases diferentes.
La aplicación educativa CARDIODEF, actualmente en desarrollo, emplea todas las características del lenguaje OPScript mencionadas en este artículo.

## Agradecimientos

Este trabajo ha sido parcialmente financiado por la Universidad del País Vasco (UPV00141.226-T-15948/2004), el CICYT (TIC2002-03141) y la Diputación Foral de Gipuzkoa en un programa de la Unión Europea. Los autores quisieran además agradecer a los revisores anónimos por sus comentarios y sugerencias que han ayudado grandemente a mejorar la calidad y presentación del artículo.

## Referencias

[Atif et al. 03] Atif, Y., Benlamri, R., Berri, J. “Learning Objects Based Framework for Self-Adaptive Learning”. Education and Information Technologies 8(4), pp. 345–368. (2003).

[Boyle et al. 01] Boyle, T., Cook, J.: “Towards a pedagogically sound basis for learning object portability and re-use”. In: G. Kennedy, M. Keppell, C. McNaught, T. Petrovic (eds.): Meeting at the Crossroads. Proceedings of the 18th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education. The University of Melbourne. pp. 101-109. (2001).

[Boyle 03] Boyle, T. “Design principles for authoring dynamic, reutilizable learning objects”. Australian

Journal of Educational Technology, 19(1). pp. 46-58. (2003).

[Cristea 03] Cristea A. I. "Automatic Authoring in the LAOS AHS Authoring Model". [Online] Available: http://wwwis.win.tue.nl/ah2003/proceedings/ht-2/. (2003).

[Dennis et al. 97] Dennis, B., M., Harrison, M., A. "Grendel: A Web Browser with End User Extensibility" [Online]. Available: http://csdl.computer.org/comp/proceedings/compcon/1997/7804/00/78040074.pdf. (1997).

[Garrett 05] Garrett, J., J. "Ajax: A New Approach to Web Applications". [Online]. Available: http://adaptivepath.com/publications/essays/archives/000385.php.

[Hennen et al. 00] Hennen, D., S., Ramachandran, S., Mamrak, S., A. "The Object-JavaScript Language". [Online]. Available: http://acuity.cis.ohio-state.edu/Nois/Tguide/ojs.ps. (2000).

[Henricksen et al. 01] Henricksen, K., Indulska, J. "Adapting the Web Interface: An Adaptive Web Browser". [Online]. Available: http://www.dstc.edu.au/m3/papers/AUIC2001.pdf. (2001).

[Hoyos-Rivera et al. 03] Hoyos-Rivera, G., J., Lima-Gomes R., Courtiat, J., P., Benabbou R. "The Web as a Tool for Collaborative *e*-Learning: the Case of CoLab". [Online]. Available: http://csdl.computer.org/comp/proceedings/icalt/2003/1967/00/19670312.pdf. (2003).

[HTML 4.01] W3C HTML 4.01 Specification (n. d.). [Onbline]. Available: http://www.w3.org/TR/html4/.

[Huang et al. 01] Huang, R., Ma, J. "A Java Technology Based Shared Browser for Tele-Lecturing in University 21". [Online]. Available: http://csdl.computer.org/comp/proceedings/iccima/2001/1312/00/13120298.pdf. (2001).

[JavaScript] Netscape Corporation. JavaScript Central (n. d.). [Online] Available: http://devedge.netscape.com/central/javascript/.

[Lieberherr, 04] Lieberherr, K. J. "Connections between Demeter/Adaptive Programming and Aspect-Oriented Programming (AOP)". [Online]. Available: http://www.ccs.neu.edu/home/lieber/connection-to-aop.html. (2004).

[Manouselis, et al. 02] Manouselis, N., Sampson, D. "Dynamic knowledge route selection for personalised learning environments using multiple criteria". In Proceedings of the IASTED International Conference on Applied Informatics, pp. 351–605. (2002).

[Montessoro et al. 03] Montessoro, P., Pierattoni, D., Cicuttini, R. "MTeach: A Simple Production Framework for Context-Based Educational Hypermedia". Journal of Educational Multimedia and Hypermedia 12(4) 335-359. (2003).

[Mora et al. 02] Mora, M. A., Saiz, F., Moriyón, R. "Aprendizaje colaborativo guiado: Fundamentos y aplicaciones". Revista de Enseñanza y Tecnología, Septiembre-Diciembre. pp. 18-27. (2002).

[Romero et al. 04a] Romero, V. A., Mateo, L., Elorriaga, J. A. "The educative browser YADBrowser and its script language OPScript". In: Actas del II Congreso Internacional de Tecnologías y Contenido Multimedia. Tecnología y aplicaciones Web. La Habana. (2004).

[Romero et al. 04b] Romero, V. A., Elorriaga, J. A., Mateo, L. "The Language for the Educational Browser YADBrowser". In: Actas del 6º Simposio Internacional de Informática Educativa. Cáceres. (2004).

[Romero et al. 05] Romero, V. A., Elorriaga, J. A., Mateo, L. "YADBrowser: A browser for Web based educational applications". Journal of Educational Multimedia and Hypermedia. 14(2), 129-149. (2005).

[Shonle, et al. 03] Shonle, M., Lieberherr, K., Shah, A. "XAspects: An Extensible System for Domain-Specific Aspect Languages". [Online]. Available: http://www.cs.ucsd.edu/users/mshonle/p28-shonle.pdf. (2003).

[Stroustrup 97] Stroustrup, B. "The C++ Programming Language" Third Edition. Addison-Wesley. (1997).

[Theoktisto et al. 03] Theoktisto, V., Bianchini, A., Ruckhaus, E., Lima, L. "AVANTE: A Web Based Instruction Architecture based on XML/XSL Standards, Free Software and Distributed CORBA

Components". UPGRADE 4(5), pp. 29-38. (2003).

[Uehara et al. 01] Uehara, S., Mizuno, O., Kikuno, T. "An Implementation of Electronic Shopping Cart on the Web System using Component-Object Technology". [Online]. Available: http://www-kiku.ics.es.osaka-u.ac.jp/paper/data/pdf/10.pdf. (2001).

[Wang et al. 03] Wang, P., S., Kajler, N., Zhou, Y., Zou, X. "WME: Towards a Web for Mathematics Education". [Online]. Available: http://ox.cs.kent.edu/~pwang/47wang.pdf (2003).

[Xiao et al. 02] Xiao, L., Zhang, X., Xu, Z. "On Reliable and Scalable Peer-to-Peer Web Document Sharing". [Online]. Available: http://csdl.computer.org/comp/proceedings/ipdps/2002/1573/00/15730023b.pdf. (2002).

[Xiao et al. 04] Xiao, L., Zhang, X., Andrzejak, A., Chen, S. "Building a Large and Efficient Hybrid Peer-to-Peer Internet Caching System". [Online]. Available: http://csdl.computer.org/comp/trans/tk/2004/06/k0754.pdf. (2004).