# INTEGRACIÓN DE COMPONENTES COM DE MATLAB/SIMULINK EN EL ENTORNO CASE XBDK, PARA EL MODELADO DE SISTEMAS DE CONFORMACIÓN DE HAZ

# MATLAB/SIMULINK COM COMPONENT INTEGRATION FOR XBDK CASE ENVIRONMENT, ORIENTED TO BEAMFORMING APPLICATIONS

**Mariano Raboso Mateos[1] Alberto Izquierdo Fuente[2] Juan J. Villacorta Calvo[2]**
**Lara Del Val Puente[2] Mª Isabel Jiménez Gómez[2]**

Recibido 11 de septiembre de 2007, aceptado 5 de diciembre de 2008
*Received: September 11, 2007 Accepted: December 5, 2008*

**RESUMEN**

En este artículo se describe la interfaz de acceso a Matlab desde la plataforma XBDK (XML-Based Beamforming Development Kit). La contribución más novedosa es la utilización del lenguaje de script Tcl/Tk para el acceso al entorno Matlab utilizando las interfaces COM, ofrecidas por el servicio Matlab Automation Server.

La utilización de lenguajes de script tiene innumerables ventajas a la hora de diseñar, construir y depurar prototipos o automatizar procesos. Muchas de las herramientas que se utilizan hoy en día para procesado de señal de una u otra manera permiten la utilización de lenguajes de script.

La combinación de un lenguaje de script, con la posibilidad de acceder de forma detallada a los servicios de Matlab, proporciona una manera flexible, rápida y potente, de integrar servicios en una herramienta CASE integrada como XBDK.

Palabras clave: Conformación de haz, procesado digital de señal, XML, reutilización de componentes, modelado software.

***ABSTRACT***

*This paper describes Matlab access within the XBDK (XML-Based Beamforming Development Kit) platform. A well-known script language named Tcl/Tk has been used to perform Matlab COM interfacing, a powerful and little known mechanism, provided by Matlab Automation Server.*

*Automation processing or prototype development, take advantage from script languages such Tcl/Tk. Most recent digital signal processing tools, provide mechanisms to be invoked by script languages.*

*A language script plus COM integration, performs a detail, flexible, quick and powerful mechanism to provide services for a CASE integrated development environment such XBDK.*

*Keywords: Beamforming, digital signal processing, XML, component reuse, software modeling.*

## INTRODUCCIÓN

Los lenguajes de script (hoy llamados lenguajes dinámicos) se han venido utilizando con frecuencia para automatizar tareas y desarrollar programas de una forma sencilla y rápida. Por el contrario, la eficiencia del código generado no es comparable al código compilado que se ejecuta directamente sobre la máquina, puesto que éstos necesitan de un intérprete.

La gran aceptación de estos lenguajes ha tenido como consecuencia el desarrollo de versiones muy potentes, con capacidad de ser ampliadas mediante extensiones. La gran flexibilidad con que cuentan dichos lenguajes les dota de la posibilidad de interactuar con muchos otros lenguajes compilados o no como son C, C++, C#, Java, Fortran, Perl, Python, JScript, Visual Basic, etc.

[1] Facultad de Informática. Universidad Pontificia de Salamanca. Compañía 5. Salamanca 37008. España. E-mail: mrabosoma@upsa.es
[2] Grupo de Procesado en array. Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática. Universidad de Valladolid. Campus Miguel Delibes. Valladolid, España. E-mail: alberto.izquierdo@tel.uva.es

El entorno case integrado XBDK se ha diseñado para modelar sistemas de conformación de haz mediante XML [1-2]. Para la simulación de los modelos se necesita interactuar con Matlab, de tal forma que se pueda acceder al *workspace* y poder transferir las variables necesarias. Utilizar un acceso tradicional mediante el uso de la línea de comandos del sistema operativo es una solución poco eficiente y tosca.

En este artículo se describe cómo interactuar de manera eficiente con Matlab y aprovechar estas características mediante la utilización del lenguaje Tcl/Tk, utilizado en XBDK para el desarrollo del prototipo principal.

**La plataforma XBDK**

XBDK es una plataforma para el desarrollo de sistemas de conformación de haz utilizando el lenguaje XML [1-4]. Es una herramienta CASE integrada que, además de permitir describir modelos de sistemas de conformación de haz en este lenguaje, permite generar el código Matlab asociado a dichos modelos, e importar modelos previamente diseñados en Simulink.

La arquitectura de esta plataforma se organiza jerárquicamente en una serie de niveles, según el nivel de abstracción.

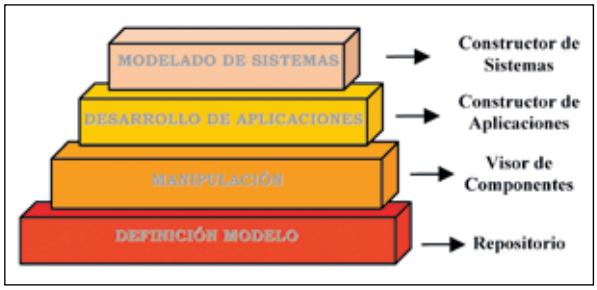


Figura 1. Arquitectura de niveles para XBDK.

Relacionadas con cada uno de los niveles, se dispone de herramientas que permiten crear y manipular todos los ficheros asociados, así como los traductores y herramientas de simulación.

1. *Modelado de Sistemas.* Nivel que permite modelar sistemas de conformación de haz con el mayor nivel de abstracción.
2. *Desarrollo de aplicaciones.* Nivel de construcción de aplicaciones, que ofrece la capacidad para crear aplicaciones genéricas en base a la definición de los sistemas a través de sus bloques funcionales.
3. *Manipulación*. Este nivel permite manejar los ficheros que contienen la información necesaria para describir los modelos.
4. *Definición del modelo*. Se trata del nivel que permite definir la información que contendrá el modelo.

En la figura 1 se observa cómo cada uno de los niveles están relacionados con las herramientas que se muestran a la derecha, salvo el nivel inferior, en el que se encuentra el repositorio o almacén de datos. Estas herramientas son las encargadas de dotar a la plataforma de las funcionalidades necesarias para poder ofrecer los servicios.

**El lenguaje Tcl/Tk**

Tcl (Tool Command Language) es un lenguaje de script muy potente y fácil de aprender [5]. Es muy útil para aplicaciones de automatización, test, en sistemas embebidos y en aplicaciones web y de acceso a base de datos.

Fue desarrollado en 1988 por John K. Ousterhout en la Universidad de California y posteriormente mantenido por Sun Microsystems Laboratories, por el grupo SunScript. Una de sus grandes ventajas es que es multiplataforma. Existen versiones para Windows, Mac OS X y la mayor parte de las versiones de Unix: Linux, Solaris, IRIX, AIX, BSD, etc.

Es un lenguaje en continua evolución, por lo que de forma constante aparecen nuevas versiones de intérpretes, junto con extensiones que amplían la funcionalidad del lenguaje.

Tcl es flexible y abierto, de tipo *open source*, por lo que cualquier desarrollador puede investigar sobre los detalles del lenguaje, incluir nuevas funcionalidades, modificar las existentes, y si se quiere, desarrollar nuevas versiones comerciales. Las extensiones suelen tener el mismo tipo de licencia, aunque depende de cada desarrollador.

Asociado a Tcl, existe un kit de herramientas para el desarrollo de interfaces gráficas de usuario llamado Tk (Tool Kit), que es la extensión más popular de Tcl. Tk también fue desarrollado por John Ousterhout y ofrece un intérprete que añade a los comandos de Tcl, otros capaces de crear interfaces gráficas de usuario típicas tales como botones, paneles, listas desplegables, ventanas de diálogo, etc. Normalmente se distribuyen ambos en un paquete llamado Tcl/Tk.

En las últimas versiones se incluye un paquete para acceder a objetos COM, mediante una API muy extensa, y se ha sustituido el intérprete tradicional por un compilador, que traduce el código fuente a un código de bytes, que posteriormente otro intérprete ejecuta. Esta mejora consigue un aumento sustancial en la velocidad de ejecución, que aún así no puede compararse con otros lenguajes compilados.

### Tecnología de componentes COM de Microsoft

COM (Component Object Model) es una plataforma desarrollada por Microsoft para utilizarla, principalmente, en las distintas versiones del sistema operativo Windows. Existen implementaciones para otras plataformas.

Se basa en crear componentes reutilizables y crear aplicaciones con esos componentes. Para ello proporciona servicios de comunicación entre procesos y creación dinámica de objetos desde cualquier lenguaje de programación que soporte esta tecnología.

Una ventaja fundamental que ofrece esta tecnología es que los objetos están ya implementados en código binario, por lo que no es necesario disponer del código fuente original. Esta característica permite que se puedan integrar componentes escritos en distintos lenguajes, modificar las aplicaciones sin necesidad de recompilar los objetos, e incluso reubicarlos en otra máquina distinta.

Esta tecnología ofrece la posibilidad de obtener componentes de tipo COTS (Commercial off-the-Shelf) y crear aplicaciones en base a esos componentes, diseñados por distintos fabricantes. Esta característica es muy útil a la hora de evaluar el comportamiento de componentes de igual funcionalidad, pero de distinto fabricante. También es muy útil para desarrollar metodologías de prototipado rápido de aplicaciones.

Alrededor de COM existen una serie de tecnologías relacionadas que forman una familia. Estas son COM+, Distributed COM (DCOM) y ActiveX.

OLE (Object Linking an Embedding) es un sistema que utiliza objetos distribuidos, bajo un protocolo desarrollado por Microsoft. Se utiliza principalmente para manejar documentos complejos mediante enlaces y para transferir datos entre aplicaciones mediante las operaciones de copiar y pegar.

Originariamente fue una evolución de DDE (Dynamic Data Exchange). A diferencia de DDE, OLE permitía no sólo transferir datos entre aplicaciones, sino mantener enlaces activos entre documentos.

La versión 1.0 de OLE evolucionó hacia la 2.0, que fue implementada utilizando tecnología COM.

En 1996 Microsoft cambió el nombre de OLE 2.0 por ActiveX, como consecuencia de la fusión de la tecnología OLE con Internet. Esta tecnología introdujo los controles ActiveX, los Active Documents y la tecnología Active Scripting, esta última basada en OLE Automation.

COM+ hace referencia al conjunto de servicios basados en COM que se desarrollaron por primera vez en Windows 2000. Introduce servicios avanzados para manejar tareas de difícil programación como sondeo de recursos, aplicaciones desconectadas, publicación de eventos, etc. Para ello se aprovechan los servicios del Microsoft Transaction Server.

DCOM (Distributed Component Object Model) es una tecnología de Microsoft para proveer servicios de comunicación entre componentes distribuidos en distintas máquinas conectadas a través de una red. Extiende la funcionalidad de COM y utiliza los nuevos servicios de COM+. Se basa en los servicios DCE/RPC (Distributed Computing Environment/Remote Procedure Call). DCOM es un fuerte competidor de CORBA.

Con la aparición de las tecnologías .NET, Microsoft aconseja utilizar éstas en vez de COM ya que .NET puede integrar de forma transparente y bidireccional los objetos COM. Esto significa que se puede utilizar COM desde .NET y viceversa [6]. De esta forma se pueden reutilizar todos los componentes COM que ya se hubieran diseñado.

## MATLAB Y COMPONENTES COM

Matlab ofrece la posibilidad de interactuar con otras aplicaciones a través de la utilización de la tecnología COM sobre Windows [7]. Permite acceder desde Matlab a controles y procesos externos, o comportarse como un servidor para otras aplicaciones cliente.

Matlab también ofrece un servicio básico DDE para el intercambio de información entre aplicaciones que permite compartir comandos e intercambiar información de tipo copiar y pegar bajo Windows.

Este artículo se centra en el acceso a los servicios que ofrece el Matlab Automation Server mediante aplicaciones externas basadas en el lenguaje de script Tcl/Tk.

### Los objetos COM

Un objeto COM es una instancia de una clase objeto componente, o sencillamente componente. Se almacena en una aplicación de tipo servidor, siendo accesible por una o más aplicaciones cliente.

Los objetos COM están fuertemente encapsulados, por lo que únicamente pueden ser accesibles a través de las interfaces que exportan. Estas interfaces permiten acceder a aquellos métodos y propiedades que el objeto hace públicos y que por tanto es necesario conocer.

El fabricante del componente debe proporcionar en detalle la descripción de éstos.

Hay cuatro tipos básicos de interfaces COM:

- **IUnknown**. Es una interfaz básica estándar, obligatoria para todos los objetos COM.
- **IDispatch**. Interfaz estándar que sirve para obtener información general sobre el objeto y particular sobre los métodos y propiedades que son accesibles.
- **Custom**. Se corresponde con una interfaz que puede ser definida por el usuario.
- **Dual**. Es una combinación de IDispatch y Custom.

Matlab sólo soporta las implementaciones de objetos COM que sean compatibles con la API estándar: *Microsoft Active Template Library (ATL)*.

Para poder usar un componente, éste se debe instanciar utilizando un identificador llamado *programmatic identifier*. Cada desarrollador debe utilizar un identificador único para cada servicio. Matlab tiene asociados tres:

- **Matlab.Application**. Se utiliza para arrancar un servidor de Matlab (automation server) en una ventana independiente, con la versión más reciente de Matlab que se haya asociado con este servidor.
- **Matlab.Autoserver**. Se utiliza para arrancar un servidor de Matlab (automation server) en una ventana independiente, con la versión más reciente de Matlab que esté instalada.
- **Matlab.Desktop.Application**. Arranca el entorno completo de la versión de Matlab más reciente que se tenga instalada.

**Arquitectura cliente/servidor de Matlab**

El modelo cliente/servidor de Matlab está diseñado en base a cuatro arquitecturas:

- **Cliente Matlab (In-Process Server)**. En esta arquitectura, el cliente Matlab accede a los servicios mediante un componente implementado como un control ActiveX o un DLL. Ambos, cliente y servidor, se ejecutan en el mismo proceso y, por tanto, comparten el mismo contexto.

  La comunicación entre ellos es muy rápida y eficaz. A diferencia del control ActiveX, el componente asociado a una DLL se ejecuta en una ventana independiente a la del proceso cliente.

- **Cliente Matlab (Out-Process Server)**. En esta configuración, la aplicación cliente interactúa con un componente que ha sido implementado como un fichero ejecutable (.exe). El componente se instancia en un servidor que se ejecuta en un proceso independiente. Esto hace que la comunicación ya no sea tan rápida como en el caso anterior.

- **Aplicación cliente y servidor Matlab (Automation Server)**. En esta configuración, una aplicación cliente (*automation controller*) puede acceder a los servicios proporcionados por el *automation server*. Éste proporciona los servicios para poder ejecutar comandos y transferir variables con el *workspace* de Matlab.

  El servidor se puede arrancar en la misma máquina del cliente o en otra conectada a la misma red. Esta configuración sólo se puede utilizar si el sistema soporta el modelo de objetos distribuido DCOM.

  La comunicación se ve afecta por el rendimiento de la red de comunicaciones a la que se conectan las máquinas. En esta configuración, se debe crear un identificador que haga referencia a una ubicación de red.

  Como ejemplo, la figura 2 muestra un identificador disponible en \\hiseuibd01\home$\MATLAB2\bin, que es un directorio compartido donde se encuentra instalada la versión 5.2 de esta herramienta. La ventana se corresponde con la utilidad para agregar referencias en Visual Basic .NET.

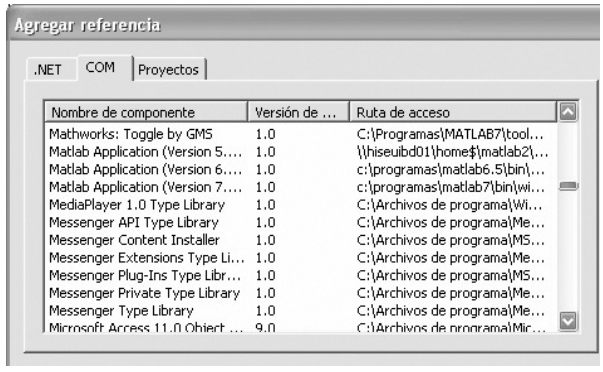

Figura 2. Agregar referencias en VB .NET.

  También existe la posibilidad de arrancar en modo exclusivo o compartido, en función de que pueda ser accedido por una o más aplicaciones cliente, respectivamente.

- **Aplicación cliente y servidor Matlab (Engine Server)**. Para aplicaciones escritas en C, C++ y Fortran, Matlab ofrece una interfaz mucho más rápida que se denomina IEngine.

**Acceso a través del Matlab Automation Server**

En la instalación habitual de Matlab queda registrado el servidor de componentes. No obstante, se puede comprobar mediante la edición del registro de Windows, o utilizando herramientas como OLEViewer del producto Visual Studio .NET.

La figura 3 muestra el contenido del registro, donde se aprecian tres versiones de Matlab instaladas (5, 6 y 7).


Figura 3. Componentes COM en el registro de Windows.

La figura 3 muestra los distintos identificadores para acceder a los componentes de Matlab instalados:

- Matlab.Application
- Matlab.Application.Single
- Matlab.AutoServer
- Matlab.AutoServer.Single

La diferencia marcada por el sufijo *Single*, determina que el modo de arranque del servidor se realice en modo exclusivo, en vez de compartido.

Con el explorador de objetos de Visual Basic .NET, se pueden ver los métodos que están accesibles. La figura 4 muestra los métodos accesibles en MLApp:

El proceso para acceder a los objetos COM es el siguiente:

1. Crear un objeto COM especificando el *programmatic id* correspondiente. Esta operación devuelve un manejador que se podrá utilizar para acceder al componente.
2. Invocar los métodos que se deseen a través del manejador.

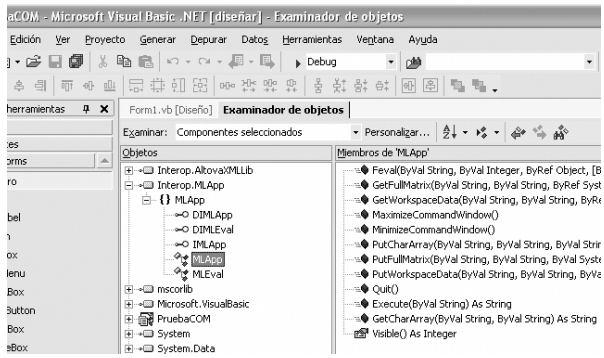


Figura 4. Explorador de objetos en Visual Studio .NET.

Las siguientes líneas muestran el proceso para crear un objeto COM de Matlab, desde lenguaje Tcl/Tk:

```
package require tcom
set application [::tcom::ref createobject "Matlab.
Application.7"]
```

Una vez creada la referencia, ya se pueden activar los métodos. La siguiente sentencia muestra cómo arrancar un fichero de Matlab y obtener la variable (*diagram*) en el workspace "*base*". La nueva variable Tcl *diagrama* contiene ahora los valores devueltos:

```
$application Execute "run('C:/smi.m')"
$application GetWorkspaceData "diagram" "base"
"diagrama"
```

Una de las posibilidades más interesantes es la oportunidad de interactuar con otras aplicaciones, por ejemplo MS Excel, para construir un entorno de prueba en el que los resultados de los ensayos puedan ser analizados por una herramienta de cálculo.

## LA PLATAFORMA XBDK

XBDK es una herramienta CASE integrada que permite crear, modificar y evaluar sistemas de conformación de haz mediante modelos descritos en XML. Tiene gran aplicación no sólo en el ámbito profesional, sino también en educación. XBDK ha sido diseñada por el Grupo de Procesado en Array (GPA) de la Universidad de Valladolid.

Al tratarse de un entorno integrado, en el diseño de la herramienta se analizaron las distintas posibilidades de integración, tanto con herramientas del entorno del procesado de señal (Matlab/Simulink), como con gestores de bases de datos, y herramientas para manipular ficheros XML.

La solución más inmediata a un problema de integración es la de arrancar procesos y ejecutar los programas como si se tratara de la línea de comandos. La mayor parte de los lenguajes poseen una función (tipo *exec*) para poder hacer esta tarea. En unos casos esta función es bloqueante (detiene el proceso padre hasta que termine el nuevo proceso arrancado), y en otros se utiliza un proceso independiente. El lenguaje Tcl/Tk posee las dos alternativas.

La técnica anterior es interesante sólo en su segunda versión, puesto que con la llamada bloqueante no se tiene control de la herramienta hasta que finaliza el nuevo proceso arrancado. Aun así, la única forma de interactuar con la aplicación arrancada es mediante sus parámetros de la línea de comandos. Esta aproximación es muy poco flexible, y aunque se ha utilizado en las primeras versiones de XBDK, se ha desechado por su poca capacidad de integración con la herramienta CASE.

La versión actual de XBDK está escrita en Tcl/Tk y se integra con otras aplicaciones mediante el uso de objetos COM [8]. Ha sido necesario utilizar los servicios COM de Matlab/Simulink, de Microsoft Excel, Microsoft Access y Altova XML, un paquete de servicios para la manipulación de documentos XML.

Si el programador lo desea, los objetos COM arrancados se pueden hacer invisibles, de tal forma que el usuario no percibe el uso externo de estas herramientas. Esta simple característica refuerza las bases de un entorno fuertemente integrado.

A continuación se muestra un ejemplo de aplicación en el que se realizan las pruebas de evaluación de las prestaciones de un conformador de haz escrito en Matlab, y que forma parte de las herramientas disponibles en XBDK. El módulo lee datos de configuración de un fichero y genera el diagrama de radiación. Desde un programa en Tcl/Tk se cambian los valores de origen y se realiza un análisis de los resultados de cada simulación, transfiriendo éstos a la hoja de cálculo. Este procedimiento permite evaluar las prestaciones sin necesidad de realizar modificaciones en el programa original de Matlab, y sin salirse del entorno de XBDK.

El siguiente código muestra un ejemplo para interactuar con el objeto COM de MS Excel, desde Tcl/Tk:

```
set Excel [::tcom::ref createobject “Excel.Application”]
$Excel Visible 1
set libros [$Excel Workbooks]
set libro [$libros Add]
set hojas [$libro Worksheets]
set hoja [$hojas Item [expr 2]]
foreach j $tmp3 {
 $celdas Item $i B [lindex $tmp3 $i]
 $celdas Item $i A [lindex $tmp2 $i]
 incr i
}

set rango [$hoja Range “A1” “B181”]
set dibujos [$libro Charts]
set dibujo [$dibujos Add]
$dibujo ChartWizard $rango 4 [::tcom::na] 2 1 1 0 \
“Diagrama Radiación Conformador orientado a 60º\n8-sensores
d=0.5 lambda” “Ángulo de llegada” “Nivel de salida (dB)”
```

El resultado de este código se muestra en la figura 5 y se corresponde con la gráfica de los datos almacenados en una hoja de MS Excel que han sido obtenidos previamente desde Matlab a través del objeto COM correspondiente:

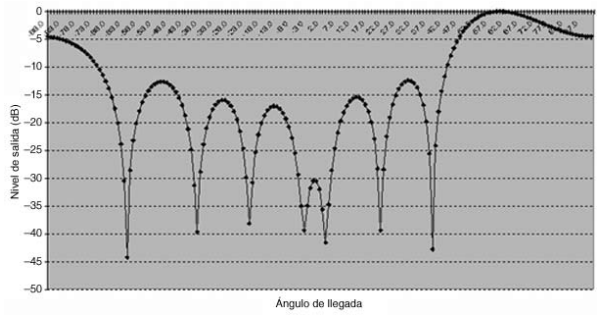


Figura 5. Diagrama de radiación en MS Excel.

## CONCLUSIONES

En este artículo se ha presentado la aplicación de una técnica poco conocida y no suficientemente aprovechada, para la integración de aplicaciones en el ámbito del procesado de señal.

Muchos programadores, aun conociendo la existencia de los lenguajes de script, sacan poco partido a su uso para tareas de test, prueba y sobre todo, elaboración de prototipos. Para estas tareas es necesaria una fuerte capacidad de integración con otras herramientas, que no se consigue utilizando la línea de comandos para pasar argumentos.

La tecnología COM de Microsoft proporciona una forma rápida, sencilla y potente de integrar objetos distribuidos en otras aplicaciones, y así construir entornos de trabajo fuertemente integrados.

Por último, para construir una herramienta CASE integrada, es necesario aprovechar al máximo todos los recursos de integración que estén disponibles, no sólo para obtener un aspecto visual integrado, sino también integración en el plano funcional. El lenguaje Tcl/Tk es una herramienta excepcional para conseguir este nivel de integración.

## REFERENCIAS

[1] M. Raboso, A. Izquierdo y J.J. Villacorta. "Beamforming Systems Modeling using Component Reusability with XML Language". International Signal Processing Conference. Texas (Dallas), USA. 2003.

[2] M. Raboso. "Modelado de Sistemas de Conformación de Haz Mediante Lenguaje XML, Basado en Reutilización de Componentes". Tesis para optar al grado de doctor. Universidad de Valladolid. Valladolid, España. 2007.

[3] M. Raboso, A. Izquierdo, J.J. Villacorta y L. del Val. "Traductor de modelos Simulink a XML, para la Plataforma XBDK". XII Congreso Internacional de Telecomunicaciones Senacitel. Valdivia, Chile. 2004.

[4] M. Raboso, A. Izquierdo, J.J. Villacorta, L. del Val y M. I. Jiménez. "Integración de Componentes COM de Matlab/Simulink en el entorno case XBDK (XML-Based Beamforming Development Kit), para el modelado de Sistemas de Conformación de Haz". XII Congreso Internacional de Telecomunicaciones Senacitel. Valdivia, Chile. 2006.

[5] Tcl Community. "Tcl Documentation". Tcl/Tk Developer Xchange. Fecha de consulta: 20 de septiembre de 2008. URLs: www.tcl.tk

[6] M. Gunderloy. "Calling COM Components from NET clients". 2001. Fecha de consulta: 20 de septiembre de 2008. URLs: http://msdn2. microsoft.com/en-us/library/ms973800.aspx

[7] The Mathworks. "Matlab Function Reference". Fecha de consulta: 20 de septiembre de 2008. URLs: www.mathworks.com/access/helpdesk/ help/techdoc/matlab.html

[8] C. Huang. "Access and implement Windows COM objects with Tcl". 2006. Fecha de consulta: 20 de septiembre de 2008. URLs: www.vex.net/~cthuang/tcom/