EJECUCIÓN REMOTA DE MODELOS SIMULINK UTILIZANDO RELATED

R. Pastor
Dpto. de Informática y Automática, UNED, Avda. Senda del rey nº9, 28040 Madrid,
rpastor@dia.uned.es

J. Sánchez, S. Dormido
Dpto. de Informática y Automática, UNED, Avda. Senda del rey nº9, 28040 Madrid,
{jsanchez,sdormido}@dia.uned.es

Resumen

RELATED es un marco de desarrollo y definición basado en XML, creado desde la necesidad de unificar esfuerzos y crear un nuevo paradigma dentro de los laboratorios basados en Internet. Un dispositivo con soporte Java (por ejemplo, un computador con un navegador) debe ser la única herramienta necesaria para poder realizar prácticas y/o experiencias de forma remota, ya sea laboratorios reales o simulaciones. En este articulo se presenta un ejemplo completo de la facilidad de integración de tecnologías de simulación con RELATED.

Palabras Clave: Teleoperación, Simulación, laboratorios virtuales.

1. INTRODUCCIÓN

Actualmente, la realización de prácticas dentro de las asignaturas implicadas en la enseñanza de la ingeniería de control adolecen de varios defectos que se pueden resumir básicamente en dos: falta de espacio para la instalación de equipos didácticos, es decir, plantas de laboratorio y falta de recursos financieros. Para resolver estas deficiencias, se han desarrollado muchos trabajos enfocados al desarrollo de entornos virtuales (basados en simulaciones) y remotos (plantas reales) [1], [2], y [5].

Pero todos estos laboratorios remotos y virtuales son esfuerzos puntuales de grupos de investigación diferentes. El uso del software y hardware de universidades diferentes no se contempla, lo que implica no aprovechar el trabajo llevado a cabo por otros previamente. Es decir, hasta ahora, no existía una metodología o una norma para la construcción de redes de laboratorios virtuales/remotos basada en desarrollos ya realizados.

Para crear estas redes de laboratorios, se necesitan nuevas herramientas e idiomas para la definición e integración de los diferentes elementos que pueden aparecer en los laboratorios (plantas didácticas, controladores, interfaces gráficas, experimentos, permisos de acceso, etc.). Una vez que los componentes se declaran por medio de estos nuevos idiomas de definición, las herramientas llevan a cabo la integración y permiten definir en la WWW un nuevo laboratorio virtual/remoto, con independencia de la situación del componente. Estos componentes (plantas, código de control, modelos, etc.) pueden pertenecer a otros desarrollos anteriores pero la interfaz de experimentación asociada al usuario debe esconder este hecho, y hacer que la situación del mismo sea irrelevante. Existen acercamientos similares en otros campos son los **IML** (Instrument Markup Language), que define una especificación para el control de instrumentos astronómicos (http://pioneer.gsfc.nasa.gov/public/aiml/) o JBeans y CORBA para el uso de componentes distribuidos sin tener en cuenta su situación u sistema operativo en el cual se ejecutan.

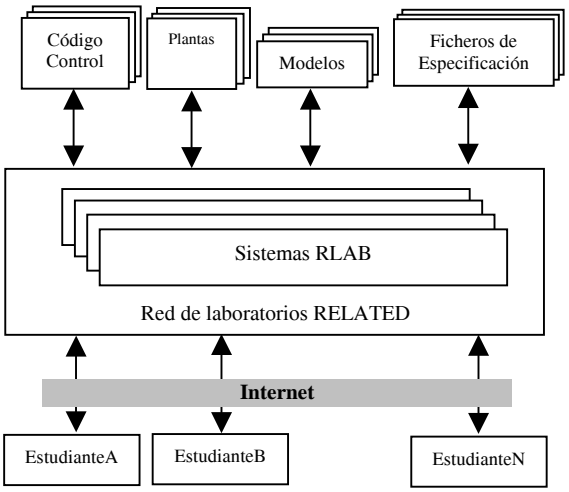

Figura 1: Arquitectura de la red de laboratorios.

2. DEFINICIÓN DE RELATED

RELATED (**RE**mote **Lab**oratory Ex**t**end**ed**) [4] es un entorno de desarrollo/definición que usa XML para el desarrollo de nuevos paradigmas en los laboratorios remotos. RELATED se ha definido para la teleoperación de sistemas de control académicos como problemas de simulación o plantas en tiempo real (modelos o equipos didácticos). La idea principal es definir una entidad abstracta llamada sistema **RLAB** por medio de un **DTD XML** y publicarlo en un servidor de sistemas remotos, permitiendo de esta forma a los estudiantes el acceso remoto al mismo. Por consiguiente, será posible trabajar con cualquier sistema RLAB conectado al servidor de sistemas RLAB y compartir recursos tan diferentes como son plantas, simulaciones, o, simplemente, código de control ya implementado (ver Figura 1). Las características principales de RELATED son:

1. Acceso a través de Internet a los servicios ofrecidos por los sistemas RLAB por medio del servidor de sistemas RLAB.

2. Teleoperación de cualquier sistema RLAB definido en la red de sistemas. Los certificados y firmas digitales proporcionarán la seguridad necesaria para el uso de un sistema RLAB.

3. La especificación del sistema remoto usando XML. Archivos XML diferentes definen tipos diferentes de sistemas en computadoras o plataformas diferentes.

4. Independencia de la plataforma y portabilidad. El marco de trabajo RELATED se está desarrollando totalmente en Java. Según la famosa frase de los diseñadores de Java "escriba una vez y ejecute en todas partes", un sistema RLAB podría ejecutarse en cualquier plataforma con soporte Java como Windows, sistemas Unix/Linux, o tarjetas Java.

5. Reusabilidad de código. Cualquier código desarrollado para el funcionamiento local de un sistema de RLAB puede adaptarse para la aplicación a otros nuevos sistemas RLAB.

6. Repositorio de código de control. El código de otros sistemas RLAB remotos puede usarse para construir otro sistema RLAB remoto.

7. La arquitectura es abierta. Se pueden hacer cambios fácilmente en el sistema. Por ejemplo, añadir nuevas características al sistema simplemente consiste en "agregar líneas XML a los archivos de definición."

Como se muestra en Figura 2, la arquitectura del sistema RELATED es bastante simple. Hay varios sistemas RLAB (formalmente se conocen como Componentes Servidor RLAB) conectados al servidor de sistemas RLAB, que actúa como un sistema de información. Permite a los usuarios conectar a un sistema RLAB particular, es decir, es decir funciona como un servidor de referencias.

3. DEFINICIONES XML DE UN SISTEMA RLAB

XML se usa en la definición de cada sistema RLAB. Todos los objetos incluidos en un sistema RLAB están "escritos" usando un archivo XML en términos de etiquetas y atributos. Estos objetos son: el sistema en sí, módulos, variables, parámetros, vistas, referencias, y experimentos.

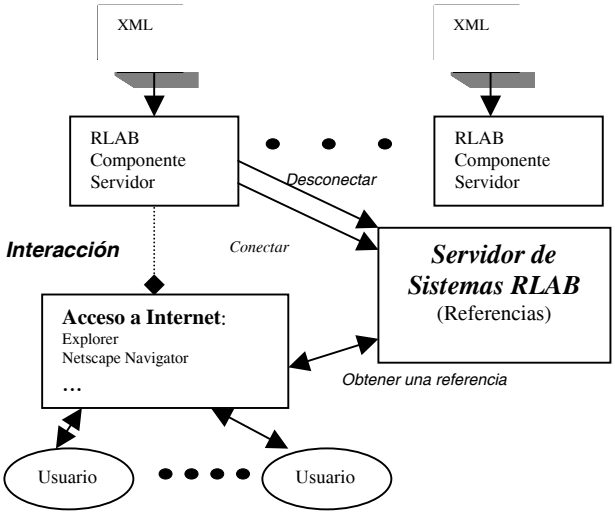

Figura 2: Arquitectura de RELATED

Un archivo de definición XML está compuesto de dos secciones:

- *Sección de comportamiento estático.* Incluye el sistema, los módulos, las referencias y las vistas. Aunque las variables y parámetros se modifican dinámicamente forman parte del sistema y/o los módulos y por consiguiente se incluyen en esta sección.
- *Sección de comportamiento dinámico.* Aquí sólo se definen los experimentos.

Un sistema RLAB es una caja negra que puede manipularse a través de sus variables como ocurre en aplicaciones como por ejemplo SIMULINK o LabView. La etiqueta *<system>* se usa para definir un sistema y sólo se permite una etiqueta *<system>* en un archivo de definición.

Un módulo es una entidad compuesta variables y parámetros. Esta entidad ejecuta el código dentro de un hilo (Thread) local mientras el usuario está conduciendo la ejecución de un experimento remoto. En la actualidad RELATED soporta dos tipos de implementaciones: código nativo (llamadas JNI desde Java) o código Java.

Desarrollar la implementación de un módulo es el esfuerzo más grande que el administrador de un sistema RLAB debe hacer, pero son muchas las ventajas que se obtienen. La etiqueta XML *<module>* se usa para definir un módulo y la etiqueta *<implementation>* se usa para indicar el tipo de código a ejecutar. En una implementación de código nativa se necesita indicar una biblioteca dinámica (extensión de DLL para Windows o SO para UNIX), mientras que en una implementación Java es necesario indicar el fichero con extensión JAR (mediante una dirección URL) además del nombre de la clase principal.

Una variable es un objeto que guarda un valor (double, float, entero, entero largo, boolean, o string) que puede modificarse mientras se está ejecutando un experimento en el sistema. Una variable siempre está asociada a un módulo, puesto que dicho módulo implementa las funciones para leer y escribir estas variables. Una variable se define usando la etiqueta *<var>*.

Un parámetro es parecido a una variable pero no puede modificarse mientras el sistema está ejecutando un experimento. Los parámetros son útiles para inicializar las variables internas de la implementación de un módulo. Hay parámetros predefinidos para los módulos y vistas (por ejemplo, el parámetro **ExecutionTime** define una estimación del máximo tiempo que debe tardar en ejecutarse el código del módulo). Un parámetro se define usando la etiqueta XML <param>.

Una vista define una interfaz gráfica o GUI, compuesta de componentes gráficos y capacidades multimedia (video, animaciones, sonido). Estos componentes permiten a los usuarios ver y manipular los datos remotos definidos en un sistema RLAB de una forma gráfica.

La etiqueta XML *<view>* define una vista y la sintaxis es muy parecida a la definición de un módulo.

Una referencia es una dirección URL que permite publicar información usando páginas HTML. Cualquier información sobre los sistemas RLAB locales puede ser agregada usando referencias. La etiqueta XML *<reference>* define este objeto.

Un experimento describe una posible conducta dinámica del sistema RLAB, por ejemplo, un sistema de control, un funcionamiento manual de una planta, una simulación con determinadas características, etc., Un experimento se construye usando las definiciones estáticas de módulos (y variables que los componen). Un sistema RLAB podría incluir tantos experimentos como el administrador del sistema agregue al archivo de definición. Otra posibilidad consiste en permitir a los usuarios agregar sus propios experimentos al archivo de definición.

4. CÓMO PUBLICAR UN SISTEMA RLAB

Se ha definido un procedimiento para publicar un sistema RLAB. Las fases del procedimiento son:

1. Definir la conducta estática del sistema en términos de variables y parámetros. En este momento se debe realizar un trabajo de categorización para estas variables, quedando de esta forma clasificadas.

2. Para cada de grupo de variables y parámetros se desarrolla el módulo adecuado. El administrador del sistema puede seleccionar una implementación nativa o Java para cada módulo para posteriormente programar el módulo, posiblemente reusando código ya desarrollado.

3. Desarrollo de las vistas. Varios formularios GUI pueden ayudar a describir y entender la conducta dinámica del sistema visualmente.

4. Tarea de documentación. Los resultados de la misma son documentos HTML (o cualquier URL válida) que se incluyen en el sistema como referencias.

5. Ahora se define la conducta dinámica del sistema. Primero, el administrador del sistema decide qué propiedades del sistema deben ser manipuladas por los usuarios. Segundo, se deben definir los experimentos usando los módulos disponibles en el sistema y aquellos que otros sistemas permitan ser usados. Se define el tiempo del experimento, que puede ser de dos tipos: tiempo de usuario y tiempo puro.

6. El resultado de los pasos 1-5 es el **Fichero de definición XML**. Este archivo debe ser cargado por la aplicación de publicación, de tal forma que se publica el sistema a los usuarios del sistema. Esta aplicación está totalmente escrita en Java (ver Figura 3). El archivo de definición se analiza para verificar posibles errores usando la opción ***Open XML File***. Si no ha habido ningún error, entonces el sistema RLAB se publica en el servidor de referencias de RELATED. A partir de este momento, el sistema está activo y los usuarios pueden acceder a él a través de Internet.

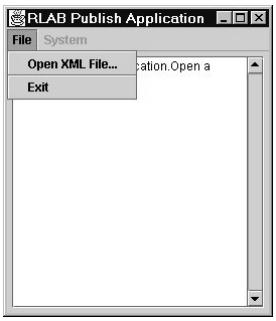

Figura 3: Aplicación de publicación de un sistema RLAB.

Un applet o una aplicación cliente se usa para acceder al sistema publicado anteriormente (ver Figura 4 y 5). Este applet permite a los usuarios arrancar y detener experimentos, mostrar y esconder las vistas, o abrir las referencias en ventanas de documentación o usar un navegador de Internet. Además, este applet dispone de capacidades gráficas para mostrar la evolución de las variables definidas en los módulos.

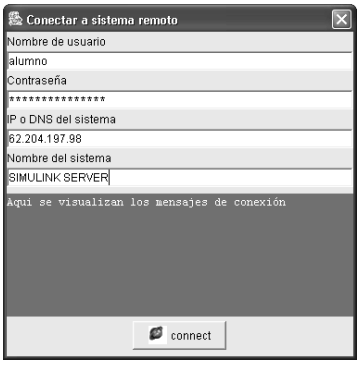

Figura 4: Conexión a un sistema remoto.

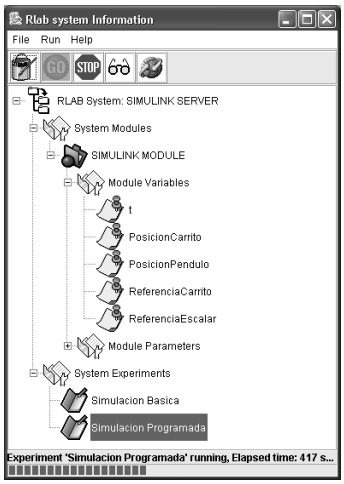

Figura 5: Aplicación cliente para acceder a un sistema RLAB.

5. EJEMPLO DE SIMULACIÓN EN RELATED: IMPLEMENTACIÓN DEL MODULO SIMULINK

Se ha desarrollado un modulo RELATED implementado en JAVA que usa la librería JMATLINK [3], que permite usar la tecnología Engine Server de MATLAB© a través de código Java, lo que permite su integración como módulo Java de un sistema RELATED. El módulo puede ser usado para ejecutar modelos SIMULINK que soporten ciertas características:

- El acceso a las variables del modelo debe hacerse a través de bloques **ToWorkspace** de SIMULINK, de tal forma que se defina un buffer de datos suficientemente grande que soporte los posibles retardos de conexión del appet/aplicación cliente con el sistema RELATED.
- Debe existir una variable **t** que almacene el tiempo de simulación. Se puede usar un bloque **Clock**.
- Las variables que se deseen modificar durante la ejecución del modelo deben definirse en bloques **Constant** de SIMULINK.

Además se han incorporado dos parámetros mas al módulo:

1. **SimulationTime**, que define el tiempo de simulación.
2. **ColletDataTime**, que define cada cuanto tiempo se comprueba si existen datos disponibles nuevos en la simulación.

5.1 CONVERSION DEL MODELO SIMULINK DEL PENDULO CARRITO A UNA VERSIÓN "EJECUTABLE" POR RELATED

En la Figura 6 se presenta el modelo **penddemo** que viene con la distribución de MATLAB. Para convertir dicho modelo a una versión ejecutable RELATED es necesario:

- Eliminar el bloque **Animation** , puesto que en el sistema cliente no se va a visualizar y lo único que hace es retardar la simulación.
- Definir las variables que se desea visualizar del modelo. En este caso son la posición del carrito (**PosicionCarrito**) , la posición del péndulo (**PosicionPendulo**) y la referencia de posición del carrito (**ReferenciaCarrito**).
- Eliminar la variable **y** del modelo y añadir tres bloques **ToWorkspace** que tengan los nombres de las variables definidas anteriormente.

Con estos sencillos cambios, el modelo (renombrado como **pend**, ver Figura 7) ya puede ser usado como un sistema RELATED. Para ello se debe definir los parámetros necesarios para configurar el módulo SIMULINK MODULE, que permite la ejecución del modelo **pend**.

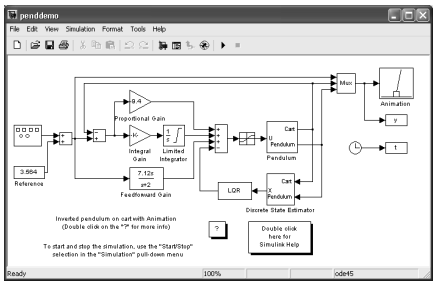

Figura 6: Modelo SIMULINK original.

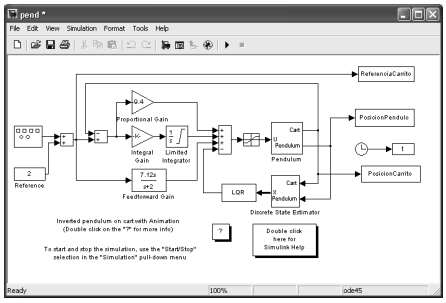

Figura 7: Modelo SIMULINK modificado para ejecutarse en RELATED.

5.2 ARCHIVO DE DEFINICIÓN XML PARA EL MODELO SIMULINK

Siguiendo la metodología descrita anteriormente el sistema se puede describir con un solo módulo SIMULINK MODULE (véase Figura 8), que tiene:

Tres parámetros:

- *MODELNAME*, usado para cargar el modelo SIMULINK© a ejecutar.
- *COLLETDATATIME*.
- *SIMULATIONTIME*.

```
<module name="SIMULINK MODULE">
Modulo java de ejecucion de modelos Simulink
        <param name="ExecutionTime" type="long"
value="2000">Tiempo de ejecucion estimado</param>
        < param name="MODELNAME" type="String"
value="pend" >Nombre del modelo a ejecutar</param>
        < param name="COLLETDATATIME" type="long"
value="2000" >Tiempo usado por la tarea Simulink</param>
        < param name="SimulationTime" type="double"
value="1000.0" >Tiempo de simulacion</param>
        <var name="t" type="double" vector="yes" initial="0"
max="1000" min="0" units="seconds">Tiempo</var>
        <var name="PosicionCarrito" type="double"
vector="yes" initial="0" max="5.0" min="-5.0"
units="cms">Carrito</var>
        < var name="PosicionPendulo" type="double"
vector="yes" initial="0" max="360" min="0"
units="grados">Pendulo</var>
        < var name="ReferenciaCarrito" type="double"
vector="yes" initial="1.0" max="5.0" min="-5.0"
units="cms">posicion de referencia (vector)</var>
        <var  name="ReferenciaEscalar" type="double"
initial="1.0" max="5.0" min="-5.0" units="cms">posicion de
referencia</var>

<implementation type="JAVA"
jarfile="file://c:/users/rafa/rlab_xml/rlab/SIMULINK_JAVAImplement
ation/SimulinkModule.jar" classname="SimulinkModule">Modelo de
ejecucion de modelos Simulink</implementation>
</module>
```

Figura 8: Definición del modulo SIMULINK MODULE.

Cinco variables:

- **t**, tiempo de la simulación.
- PosicionCarrito.
- PosicionPendulo.
- ReferenciaCarrito.
- ReferenciaEscalar, usado para modificar el valor de la referencia de posición del carrito.

Finalmente se han definido dos experimentos:

a) Simulación básica, realiza la simulación del modelo SIMULINK **pend** durante 100 segundos. En la Figura 9 se muestra el código XML necesario para definir dicho modelo.

b) Simulación programada, donde se han realizado cambios programados a la variable ReferenciaEscalar para analizar el comportamiento del modelo. En la Figura 10 se muestra el código XML necesario.

```
<experiment name="Simulacion Basica"
sampleTime="1000">Realiza la simulacion basica del modelo pend
        <duration type="Time" time="100"/>
        <run module="SIMULINK MODULE">Esta es la
simulacion basica
        <setparam name="SimulationTime" value="100.0"/>
        <interactives names="ReferenciaEscalar" show="false"/>
        </run>
</experiment>
```

Figura 9: Definición del experimento Simulación Básica

```
<experiment name="Simulacion Programada"
sampleTime="1000">Se cambia la referencia
<duration type="Time" time="1000"/>
<run module="SIMULINK MODULE">
        <setparam name="SimulationTime" value="1000.0"/>
        <set name="ReferenciaEscalar" time="0" value="1.0"/>
        <set name="ReferenciaEscalar" time="100" value="-
1.0"/>
        <set name="ReferenciaEscalar" time="350" value="2.0"/>
        <set name="ReferenciaEscalar" time="500" value="-
2.0"/>
        <set name="ReferenciaEscalar" time="700" value="3.0"/>
        <set name="ReferenciaEscalar" time="850" value="-
3.0"/>
        <interactives names="ReferenciaEscalar" show="true"/>
</run>
</ experiment>
```

Figura 10: Definición del experimento Simulación Programada.

5.3 EJECUCIÓN DEL MODELO SIMULINK

Para realizar la simulación del modelo simplemente se debe usar la aplicación cliente de RELATED y lanzar la ejecución de uno de los experimentos. En la Figuras 11 y 12 se muestra la ejecución del experimento Simulación Básica dentro de RELATED, mostrando las variables obtenidas desde RELATED y comparando los resultados con los

obtenidos del modelo SIMULINK. Como se puede apreciar los resultados son idénticos, como se esperaba. Los beneficios de esta aproximación se basan en la portabilidad del modelo SIMULINK a un entorno abierto, donde cualquiera, aunque no tenga instalado MATLAB/SIMULINK puede acceder a complejas simulaciones usando el acceso básico a Internet.

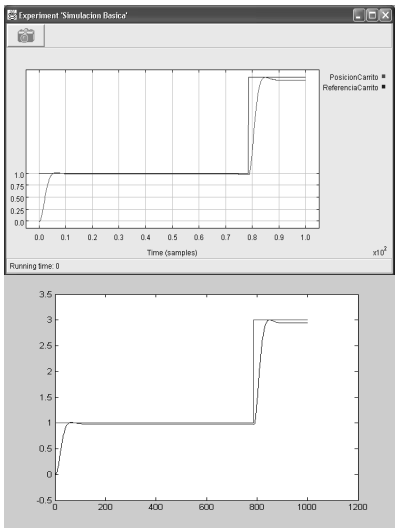

Figura 11: Posición del carrito y referencia para el módulo RELATED y la simulación desde SIMULINK.

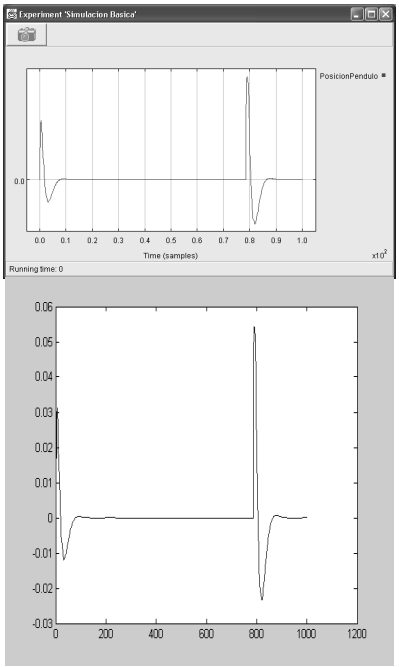

Figura 12: Posición del péndulo para el módulo RELATED y la simulación desde SIMULINK.

6. CONCLUSIONES

Se ha presentado como el uso de una metodología específica para el desarrollo de módulos dentro de RELATED puede facilitar el acceso a tecnologías de simulación, como SIMULINK, permitiendo a usuarios que no disponen de MATLAB poder ejecutar modelos SIMULINK.

Se ha presentado una herramienta muy útil para la ejecución generalizada de modelos SIMULINK, que es reusable dentro del marco RELATED, en forma de módulo de ejecución. Además se ha visto la facilidad de integración de código ya desarrollado dentro de RELATED.

Referencias

[1] Gillet, D., C. Salzmann and P. Huguenin (2000). A distributed architecture for teleoperation over the Internet with application to the remote control of an inverted pendulum. In: *Second Nonlinear Control Network (NCN) Workshop*, Paris, France

[2] Kheir, N.A., K.J. Ämstrom, D. Auslander, K.C. Cheok, G.F. Franklin, M. Masten, and M. Rabins (1996). Control system engineering education. *Automatica*, **32**, 147-166.

[3] Müller S. and Waller H. (1999). Efficient Integration Of Real-Time Hardware And Web Based Services Into MATLAB. In: *11th European Simulation Symposium*, Erlangen, Germany.

[4] Pastor R., Sánchez J. and S. Dormido (2001). Related: a framework to publish web-based laboratory control systems. In: *Internet Based Control Education 2001*, Madrid, Spain.

[5] Schmid, C. (2000), Remote Experimentation Techniques for Teaching Control Engineering. In: *4th International Scientific - Technical Conference PROCESS CONTROL 2000*, Kouty nad Desnou, Czech.