Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

of the 18th International Conference on Process Control

Hotel Titris, Tatranská Lomnica, Slovakia, June 14 – 17, 2011

ISBN 978-80-227-3517-9

http://www.kirp.chtf.stuba.sk/pc11

Editors: M. Fikar and M. Kvasnica

Janík, Z., Žáková, K.: Online design of SciLab/Xcos block schemes, Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 18th International Conference on Process Control*, Tatranská Lomnica, Slovakia, 583–586, 2011.

Full paper online: http://www.kirp.chtf.stuba.sk/pc11/data/abstracts/016.html

Online design of SciLab/Xcos block schemes

Zoltán Janík and Katarína Žáková

Faculty of Electrical Engineering and Information Technology
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava
(Tel: +421 2 60 291 725; e-mail: zoltan.janik@stuba.sk)

Abstract: The paper presents a tool that supports design of simulation structures for online laboratories accomplished in SciLab/Xcos environment. For running such simulations, it is necessary to build a block scheme corresponding to a control of virtual or remote device. The presented tool offers a comfortable way of such solution. It's programmed in widely used technologies to ensure wide compatibility and platform independence. This application can be used as a supporting tool in virtual and remote laboratories.

1. INTRODUCTION

The importance of virtual and remote laboratories becomes more and more significant mainly in the area of technical education. They help students to understand basic problems because they illustrate and visualize controlled dynamics, and the necessity to exercise all design steps starting with the plant identification and ending with the evaluation of the control results achieved with a particular model. The advantage is that such laboratories are available to all interested users 24 hours a day and from any place. Students and interested users can access them via commonly used Web browser on any machine that has Internet connection.

Described web-based application supports block schemes created in SciLab/Xcos desktop application. SciLab is free software for numerical computation and simulation such as Matlab. It is available for all major operating systems: Linux, Mac OS and Windows. The Xcos is a free graphical editor that offers graphical user interface to design and run simulations of dynamical system models. Xcos is distributed together with SciLab.

2. TECHNOLOGY OVERVIEW

The created tool is based on XHTML, CSS and JavaScript language. The incorporated export and import is performed by PHP technology mainly through AJAX[1] requests. These technologies are selected in order to keep the widest availability of this tool. There are no special demands at client's side. The only thing that is required is compatible web browser (the application was tested mainly in FireFox 3.5 and Internet Explorer 8, but it should work with no limitations in IE 6.0+, FF 2+, Safari 3.0+, Opera 9.0+, Chrome).

JavaScript library jQuery is used to speed up the development and simplify the source code. The library is used for the object manipulation, changing CSS[2] properties, visual functions and AJAX requests.

The application internally operates with XML and JSON[3] formats. Therefore the support of XML and JSON handling functions are required. In fact, these functions are natively supported by PHP version 5.2 or higher, thus it is the only requirement to the web server.

3. FRONT-END

The front-end of the application is demonstrated in Fig. 1. Our aim was to prepare a design that would be familiar for users using Matlab/Simulink or SciLab/Xcos environment. The whole application consists of 3 parts: main menu, block toolbar and canvas.

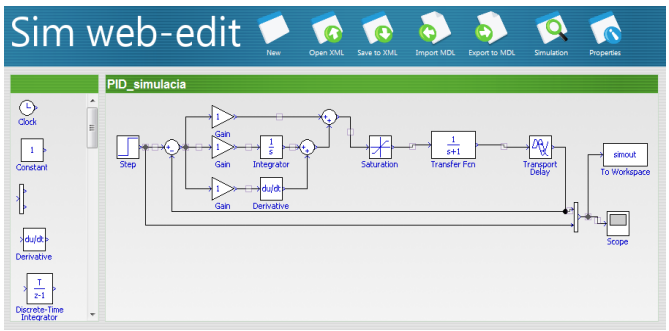

Fig. 1. Web editor interface

Block toolbar contains automatically generated list of all available blocks in the form as they appear in canvas. Each block is set to be draggable. After dragging starts, the clone of selected block is created in order it could be dropped into the canvas. The new block is inserted into canvas exactly in

[1] Asynchronous JavaScript and XML – it enables to exchange data with a server and update parts of a web page without reloading the whole page

[2] Cascading Style Sheets

[3] JavaScript Object Notation – set of key-value pairs

place where it has been dropped. In the case that the block is dropped outside canvas, it is not considered as valid and the user can see only visual feedback of moving dropped block back to its original position in toolbar.

In the application we considered 2 types of blocks:

- standard blocks are blocks from the toolbar panel that are used for modeling of dynamical behavior of systems;
- "pseudo" blocks are blocks that are generated automatically by the application for establishing an additional connection feature. In this way we recognize the *node block* that is used for multiple line connection and the *line block* that enables to modify the position of line segment placed between two blocks.

Block attributes can be changed in the pop-up window that is displayed after clicking on any standard block in the canvas. The content of the attributes window is fully dynamic. It is loaded via AJAX request from other script that builds the form (i.e. all window items) from attribute settings included in the configuration file whereby the form contains pre-filled values of current block attributes. The configuration file also defines the type of each attribute input that can be considered as text field, select box, checkbox or radio button group.

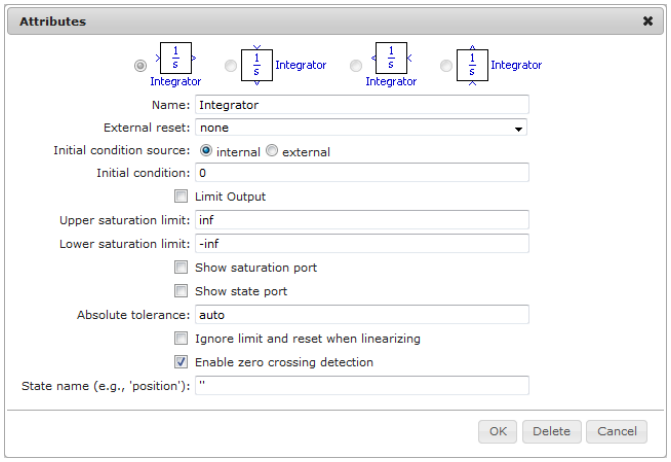

Fig. 2. Dialog window for changing block's attributes

Each block has at least one connection point – input or/and output port. These are marked by different color when the user moves the mouse cursor over the block. To connect two blocks, two connection points have to be selected by the clicking on them. The connection is possible only between input and output of two different blocks or between input and some other connection line. In this case the new node is created at selected spot on existing connection automatically. The used connection points (input and output ports) appear as inactive and they are not able to be marked and included to a new connection.

If a connection consists of more than two line segments, a line block is inserted automatically at the center of the connection line. This pseudo block can be used to adjust position of the middle line segment of the connection.

4. BACK-END

The back-end server is used for running scripts that perform tasks such as import or export. These scripts work with Xcos files. In fact, the Xcos file has the same structure as standard XML file (see Fig. 3). This format makes handling Xcos files very simple.

```
<XcosDiagram background="-1" modified="1" title="blocks">
  <Array as="context" scilabClass="String[]">
    <add value=""/>
  </Array>
  <mxUndoManager as="undoManager"/>
  <mxGraphModel as="model">
    <root>
      <mxCell id="-496982d5:12c53c4cc03:-7fff"/>
      <mxCell id="-496982d5:12c53c4cc03:-8000" parent="-496982d5:12c53c4cc03:-7fff"/>
      <ConstBlock blockType="d" id="-496982d5:12c53c4cc03:-7fdc" interfaceFunctionName
      simulationFunctionName="cstblk4" simulationFunctionType="C_OR_FORTRAN" style='
        <ScilabString as="exprs" height="1" width="1">
          <data column="0" line="0" value="1"/>
        </ScilabString>
        <ScilabDouble as="realParameters" height="1" width="1">
          <data column="0" line="0" realPart="1.0"/>
        </ScilabDouble>
        <ScilabDouble as="integerParameters" height="0" width="0"/>
        <Array as="objectsParameters" scilabClass="ScilabList"/>
```

Fig. 3. Example of Xcos file structure (part of the file)

4.1 Export

The export process is done in four basic steps. Firstly, the scheme in web editor is sent to back-end server. The editor and back-end communicate via AJAX requests using JSON as data exchange format.

In the second phase, the back-end export script transforms the scheme into Xcos format. This step include generation of standard Xcos file header, listing of each block used in scheme together with relevant group of attributes, input and output ports. Block parameters are defined in configuration file (see Section 5) and actual values are taken from the exported scheme. When the blocks are processed, the connections between blocks are specified and the scheme is finalized with Xcos file footer

In the next step, the transformed scheme is saved into file and it's offered to user to download afterwards.

4.2 Import

The import process realizes the same steps as the export but in reverse order. First, user has to upload file containing scheme that is going to be imported. The import script parses the whole file and stores it as XML object.

The loaded content is searched for list of blocks and their attributes. In a case of blocks or attributes that are not present in web editor's configuration file, the warning message is

shown. These unknown blocks will not appear in the imported scheme. Processing of unknown attributes is skipped but the parent block will still appear in the scheme. Blocks are followed by a list of connections among blocks that are also being processed.

5. CONFIGURATION

The key part of application is the common configuration file used by front-end part of the application as well as by the back-end scripts for export and import. The configuration is stored in XML format. The XML file contains settings for each block that is available in the web editor. These settings include information about attributes that are used for displaying blocks in the editor: image file name, dimensions, block class name (e.g. *BasicBlock*, *ConstBlock* or other – used as XML tag that encapsulates block attributes in Xcos document), input and output ports.

```
<block type="Constant">
 - <versions>
   - <version id="0">
       <fileName>constant_0.gif</fileName>
       <orientation>right</orientation>
       <dimensions width="30" height="30"/>
       <offset x="6" y="0"/>
       <class>ConstBlock</class>
     - <extraAttributes>
         blockType="d" interfaceFunctionName="CONST_m" simulationFunctionName="cst
         style="mirror=false;rotation=0;CONST_m;flip=false;" value="1"
       </extraAttributes>
       <inputs> </inputs>
     - <outputs>
         <output class="ExplicitOutputPort" id="E" x="40" y="15" portNumber="1" extraAt
         dataType="REAL_MATRIX" ordering="1" style="mirror=false;ExplicitOutputPort;rota
       </outputs>
     </version>
   + <version id="1"></version>
   + <version id="2"></version>
   + <version id="3"></version>
   </versions>
 + <attributeStructure></attributeStructure>
 - <attributes>
   - <attribute id="ConstantValue">
       <description>Constant value</description>
       <inputType>text</inputType>
     - <location>
         ScilabString[@as='exprs']/data[@column='0'][@line='0']/@value
       </location>
     - <location>
         Array[@as='objectsParameters']/ScilabDouble[1]/data[@column='0'][@line='0']/@
       </location>
       <default>1</default>
     </attribute>
   </attributes>
</block>
```

Fig. 4. Part of configuration file (Constant block)

Each block has a list of attributes (constant block in Fig. 4 has a single attribute – *ConstantValue*). The attribute definition contains its displayed name, input type (used in web editor's attributes window) and default value. Each attribute has its own specific location in Xcos file inside tags representing current block (e.g. *<ConstBlock> ... </ConstBlock>* as seen in Fig. 3). This location needs to be defined in configuration file as an XPath[4] address.

Data contained in the configuration file are used not only for visualization in web editor's interface, but also for correct function of import and export scripts.

The application is easily extendable by editing configuration file by the application administrator. It is not necessary to edit source codes of the application. New blocks can be easily added by modifying configuration XML file. Currently, the web editor includes the most common blocks used in Xcos block schemes. The set of included blocks is being extended simultaneously with further development of the web editor. Since the well-formed and error-free configuration file is very important for the correct function of the whole application, the DTD[5] document for configuration XML is also provided to prevent errors after modifications.

```
<!ELEMENT blocks (block*)>

<!ELEMENT block (versions, attributeStructure, attributes)>
<!ATTLIST block type ID #REQUIRED>

<!ELEMENT versions (version+)>

<!ELEMENT attributeStructure (#PCDATA)>

<!ELEMENT attributes (attribute*)>

<!ELEMENT version (fileName, orientation, dimensions, offset,
<!ATTLIST version id CDATA #REQUIRED>

<!ELEMENT fileName (#PCDATA)>

<!ELEMENT orientation (#PCDATA)>

<!ELEMENT dimensions EMPTY>
<!ATTLIST dimensions width CDATA #REQUIRED>
<!ATTLIST dimensions height CDATA #REQUIRED>

<!ELEMENT offset EMPTY>
<!ATTLIST offset x CDATA #REQUIRED>
<!ATTLIST offset y CDATA #REQUIRED>

<!ELEMENT class (#PCDATA)>
```

Fig. 5. Part of DTD for configuration file

6. CONCLUSION

The introduced application enables to prepare Xcos block scheme in online environment that in next step can be used for control of dynamical systems in frame of virtual and remote laboratories. Since it is developed by wide spread technologies it can be combined with the most of solutions that enable to use SciLab via Internet. It is to say that in the same time we are also developing similar tool for Matlab/Simulink environment (Janík and Žáková, 2009).

[4] XPath is used to navigate through elements and attributes in an XML document.
[5] Document Type Definition – defines legal document structure (list and structure of elements and attributes)

ACKNOWLEDGMENTS

The work has been partially supported by the Grant KEGA No. 3/7245/09.

It was was also supported by a grant (No. NIL-I-007-d) from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

Digiteo (1989-2011). SciLab website. http://www.scilab.org

Holzner, S. (2007). Mistrovství v AJAXu. Computer Press. Brno.

Huba, M., P. Bisták, M. Fikar, M. Kamenský (2006). Blended Learning Course "Constrained PID Control". *7th IFAC Symposium on Advances in Control Education ACE'06*. Madrid, Spain.

Janík Z., K. Žáková (2009). Online Design of Matlab/Simulink Block Schemes. *10th International Conference Virtual University*. Bratislava.

Liguš, J., I. Zolotová, P. Karch, J. Ligušová (2010). Information and control system of traverse and its integration into cybernetic centre. Electronics and electrical engineering. No. 6, p. 147-152. ISSN 1392-1215

The PHP Group (2001-2010). PHP Documentation. http://www.php.net/docs.php

Refsnes Data (1999-2010). JavaScript and HTML DOM Reference. http://www.w3schools.com/jsref/default.asp

Resig, J., the jQuery Team (2010). jQuery Documentation. http://docs.jquery.com

Restivo, M.T., J. Mendes, A.M. Lopes, C.M. Silva, F. Chouzal (2009). A Remote Lab in Engineering Measurement, IEEE Trans. on Industrial Electronics, vol. 56, nº12, pp 4436-4843.

Schauer, F., M. Ožvoldová, F. Lustig (2008). Real Remote Physics Experiments across Internet – Inherent Part of Integrated E-Learning. In: *Int. Journal of Online Engineering (iJOE)*, 4, No 2.

Schmid, Chr. (2003). Internet - basiertes Lernen, Automatisierungstechnik, 51, No. 11, p. 485-493.

Šimunek, M., P., Bisták, M. Huba (2005). Virtual Laboratory for Control of Real Systems, *Conference Proceedings ICETA*. Košice.

Žáková, K., M. Sedlák (2006). Remote Control of Experiments via Matlab. In: *Int. Journal of Online Engineering (iJOE)*, 2, No. 3.