TECHNICAL ARTICLE

SIMULATION 73:4, 206-212

ISSN 0037-5497/99
Printed in the United States of America

A Configurable ACSL-Based Interface Generator for Simulated Systems

Miguel A. García
Department of Systems Engineering and Automatic Control
University of Valladolid, Spain
E-mail: miguel@autom.uva.es

Diego R. Llanos
Department of Computer Science
University of Valladolid, Spain
E-mail: diego@infor.uva.es

César de Prada
Department of Systems Engineering
and Automatic Control
University of Valladolid, Spain
E-mail: prada@autom.uva.es

This paper presents a software tool for generating graphical interfaces for general-purpose ACSL simulation models. The tool is aimed to construct simulators for education and training in fields such as process control. Final users can manipulate the simulation from an upper level, avoiding the details of simulation, names of variables and experiments, etc., and focusing attention on the work with each particular simulated system. The tool is based on graphical operation and transparent communication with the supporting simulation. The steps that must be followed by the programmer to prepare the graphical interface for users of the simulation are outlined. An application example, including the interface configuration as well as its use with the simulation, is provided.

Keywords: Graphical interfaces, ACSL, simulation tool development environments, simulation languages

1. Introduction

One of the aspects that define the quality of a simulation language is the key point of user interaction. Two important problems have received continuous attention in this field: how to describe the model, and how to perform the experiments on the simulation. Earliest simulation languages, such as CSMP (Continuous Systems Modeling Program), tried to mimic the way analog simulators described the set of differential equations that make up the model. Besides, there was no real separation between model and experiments, these being performed in alphanumeric form and accessing the simulation program. The arrival of the CSSL'67 (Continuous Systems Simulation Languages) standard introduced many improvements, among which we could mention a model description closer to the mathematical format, and the detachment between model and experiments. A command language was an integral part of the simulation language, allowing changes in the model variables, plotting, etc., without the need for recompiling the simulation program. A typical example of this kind of language is ACSL (Advanced Continuous Simulation Language).

These simulation languages are well established and have thousands of users all over the world. Ideally, the separation between model and experiments

should be total, in such a way that once the simulation is finished, its use in different experiments made by different users could be carried out as easily as possible, taking into account the differences between users because of their aims, level of expertise, etc. The way in which ACSL handles the separation between models and experiments is by using different files and syntax for each one. The models are implemented in a Fortran-based language and compiled separately, while the experiments are performed by typing commands that involve the names of the variables and parameters of the model, as well as procedures composed of groups of predefined commands with a given syntax. This way of performing experiments using procedures and commands requires a certain degree of expertise and knowledge of the implementation of the simulation model, which can be a real barrier for certain classes of users.

Nowadays everyone recognizes the advantages of the description of simulation models at an upper level of abstraction. This has been approached using object-oriented languages, and libraries of pre-written models, such as Dymola [1], or with expert systems, such as SIMPD [2]. These kinds of modeling languages are generators of source code of a simulation language, as with ACSL. This means that from the point of view of the final user, he is in the same position as with the traditional simulation languages, with the added disadvantage that the naming of the variables of the generated model is quite often very obscure due to their automatic generation.

At the same time, with the arrival of graphical interfaces, certain languages, such as Simulink or Graphics Modeller [3], have given a new life to the old approach of describing a model in terms of interconnecting blocks, but this time using all the graphical facilities of modern operating systems and computers. At the same time, recognizing the importance of the user interface, they utilize the same graphical interface for performing experiments in the model, with the user having access to all the model parameters, and with the same problems as before.

Our point of view is that the requirements of an interface for describing a model are quite different from those of an interface oriented to performing experiments, and that the latter requires adaptation to the needs of specific users and aims. This is particularly true when the aim of the simulation is to train people, or when the users are non-experts in simulation.

In this paper we present a tool for developing graphical interfaces for general-purpose simulation models, oriented to performing experiments by the final users. The implementation has been made with ACSL, but the ideas behind it are general, and the tool could be adapted to other environments. The current implementation is made in C language for the Windows environment and communicates with ACSL for Windows by using Dynamic Data Exchange (DDE)[1], but with the same policy and different communication methods, versions for other operating systems could be developed.

The developed system allows the user to avoid the need for knowledge of the details of the simulation and the use of the ACSL command tool. The work with the experiments can be carried out graphically, using controls such as scrollbars, buttons, bitmaps, etc., in a different application that communicates with the ACSL command tool in order to load the commands remotely.

The capability of dealing with the models transparently with regard to the background simulation enhances its power and accessibility by users, without the need for knowing the simulation details. The advantage of this approach is that attention can be focused on the experiments and the evolution and operation of the simulated system, which is the proper object of interest, reducing the simulation to its condition of supporting tool.

Of course, this advantage requires extra work for the programmer of the simulation. Simulation programming will consist not only of implementing the algorithms of the model and preparing the battery of experiments for using this model, but also of designing a graphical user interface that corresponds both with the variables in the model and with the commands in the experiments. This last task must be achieved by the programmer of the simulation himself, who is the person with the necessary deep knowledge to prepare a user interface for users not familiarized with the simulation aspects.

2. Interface Generator Architecture

The interface generator consists mainly of two applications: PConfig, which allows the programmer to design a user interface, and SimuModu, which allows the user to run the simulation through the graphical interface, avoiding the need to know the details of the simulation language. This set of programs interacts with ACSL, as shown in Figure 1, where we can see the group of different applications, files and communications between them and with the simulation project [4].

The programmer of the ACSL model uses PConfig to implement a graphical user interface that is stored as a file with a dedicated .pcf extension. The non-expert user of the simulation uses SimuModu to load the .pcf configuration file and automatically generate the user interface and establish a communication with ACSL. Then, manipulating the buttons, bars, etc. of the interface, he is able to obtain results from the ACSL program.

One related goal in the development of this application was to achieve resolution independence, in

[1] DDE is an established protocol for exchanging data through active links between applications that run under Microsoft Windows.

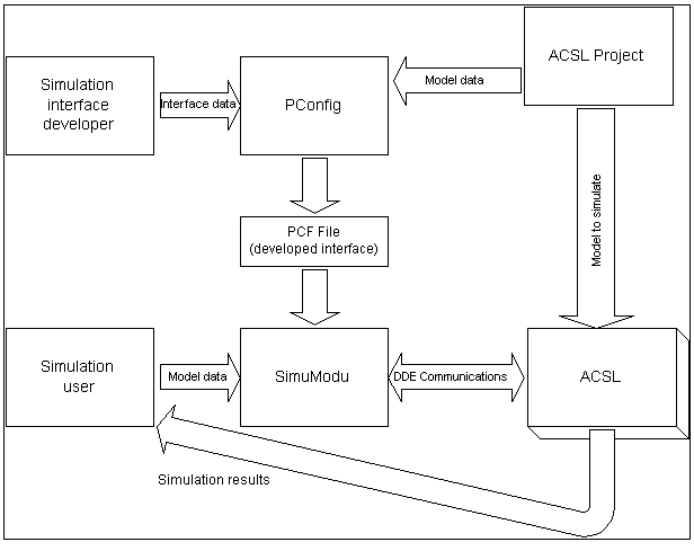

Figure 1. Block diagram of the set of applications

order to use it in any graphical mode with any kind of computer configuration

2.1 PConfig: Interface Generation

The objective of this application is to facilitate the design of an interface in such a way that a user not familiar with the simulation details can understand at a glance the way to manipulate the simulated system. This is done by combining a schematic of the process in which the programmer can place a set of graphical controls (sliding bars, buttons, etc.) once he has a design of the desired interface, such as, for instance, the one in Figure 4. PConfig will provide the tools he needs for implementing his design. The way in which this is done is described next.

First, PConfig asks for an ACSL project to communicate with. Once this is done, PConfig starts ACSL and establishes a DDE channel with it. Then it tries to find those variables of the ACSL simulation involved in the experiments. Typically, during the simulation, a user changes values of the variables defined as Constants (real valued parameters, flags, etc.), fires predefined procedures of the command file with a .cmd extension, and visualizes results of the experiments in numerical or graphical format. Because of this, PConfig is able to look for this kind of variable in the source code file (with a .csl extension) of an ACSL project, as well as for the names of the procedures of the experiments file (.cmd). Both of them are able to be associated with graphical controls or items on the experiment menu.

PConfig offers four main classes of elements for building an interface: bitmaps, scrollbars, buttons, and a specialized association of these.

The first step in the interface configuration is the introduction of a bitmap representing a schematic of the simulated system. The configuration application allows the introduction of any kind of bitmap previously created and composed with general-purpose editors (Paintbrush, Paint Shop Pro or others), and to load them in the interface at execution time. The bitmap can represent a plan or scheme of the simulated element, and constitutes the backbone of the graphical environment. Around the bitmap or over some of the elements in it, the controls associated to each variable are placed in a logical manner, in such a way that they would be found close to the zone in which they act or are expected to appear in the representation. The bitmap will play a central role and must be carefully designed to give visual meanings to the different variables, procedures, etc., that are in relation with the simulated system, making access to them easier.

The other elements, scrollbars and buttons, can be placed freely on the screen, in addition to the schematic. A value within a given range can be easily assigned with scrollbars to an associated variable. The buttons, when pressed, allow a set of associated actions to start:

- Open windows with a variable number of scrollbars of related variables.
- Open windows with a variable number of options (radio buttons) to be selected, each one giving a different value to an associated variable.
- Open a window with a text message.
- Open a file with a related document.
- Start a procedure defined in the command file.
- Introduce a reset to their original values of the variables of the simulation.

The programmer can use the messages and document files to warn about some specific behavior of the simulation, or to explain the simulated system. These kinds of dialog boxes work as pre-built help windows at the disposal of the designer of the simulation to insert comments he considers necessary for the users.

The way in which the association of a variable to one of those elements is made is through an "Examine" button in the configuration dialog of each control added to the interface. So when the designer chooses a control (e.g., a scrollbar), the application establishes a communication with the selected project and presents a table with all the variables in the project that can be associated to the given control.

When a variable in the simulation project is associated to a control in the graphical interface, the initial—normally stationary—value for that variable in the simulation is automatically loaded, giving the designer help in establishing the allowed values around the initial one. In the case of scrollbars, minimum and maximum allowed values are requested, in such a way that values out of the range of the imposed ones are substituted for the limit values in the same trend.

In the same way, all the user interfaces have a menu item called "Experiments," and the different procedures in the command file of the simulation project can be associated to sub-items of "Experiments" with

a proper experiment name. Likewise, the experiments can be associated to buttons inside the mainframe workspace, which is of interest for procedures that execute a plot of the evolution of the variable associated to a transmitter and are loaded from a button in the graphical representation of the transmitter.

The fourth graphical utility is a specialized dialog box. Sometimes it is of interest to join a group of variables with a strong link between them in a separate representation. For instance, a PID (Proportional, Integral and Derivative) regulator has three parameters (Kp, Ti and Td) that can be put together in order to manipulate them simultaneously when the tuning operation of the controller is broached. Of course, such a dialog box is normally thrown from a controller button placed on the controller representation at the graphical scheme of the regulation system.

2.2 Configuration File

Once the graphical interface with all the bitmaps, menu items, controls and secondary dialog boxes is finished, the option of saving it as a text-only file can be chosen. Indeed, the file itself can be edited and modified outside of the design application just by changing the coordinates of the position of the different controls in the graphical interfaces that are registered in the configuration file next to each type of control. Likewise, the configuration file includes the names and paths of the ACSL executable, the ACSL project associated to the graphical interface, and the bitmap file used as the background of the interface.

2.3 SimuModu: Application Execution

SimuModu provides the user with the capability of using the simulation execution environment designed by the programmer. When invoked, it is able to load the different configuration files generated by the interface designer. When this execution application loads a configuration file, its first action is to run ACSL and to execute the corresponding ACSL project read. With the information in the configuration file, this application can reproduce the graphical interface and add the functionality of the controls in it. The way in which this capability is implemented is by communicating the changes in the values of the variables made with the scrollbars, radio buttons, etc., from the interface to the ACSL project that is being executed. In the same way, when an experiment is chosen, the name of its procedure is passed to the executing project as those in the ACSL command line.

Both types of changes are related in such a way that new values for the variables are sent to the simulation, together with the chosen command, so when the experiment is executed, it takes the values selected for the variables in that moment. For instance, an experiment can be based on a jump in an input variable at a given time from an initial value to another one. The concrete values for both the input variable and the instant of the jump are taken from the user interface, and they will be the current values when the experiment is selected in the graphical interface and is sent to the simulation. If only the value of the input variable is changed at a fixed time instant, the selection of the experiment executes the simulation and when the simulation time reaches the given instant, the jump to the new value in the input variable is imposed.

Because of the arbitrary changes in the values of the variables, effects of different experiments can be hidden or interfere with them. Thus a procedure (in the ACSL command file) linked to an "initialization" experiment (in the graphical interface) can be useful in order to fix all the variables at their stationary values before executing any other kind of experiment.

3. Application Example

As an application example, let's consider a process composed of a heat exchanger and its associated control system: a PID regulator. The aim of the process is to heat a flow of a cool liquid, a juice in this example, using saturated steam as the heating element. A schematic can be seen in Figure 2. Here the cool juice enters the heat exchanger by the left-hand-side pipe and, after being heated, leaves through the pipe on the right-hand side. The steam comes from the top pipe, and its flow can be changed by the opening of the valve placed on it.

The temperature of the juice at the output of the chamber is measured with the temperature transmitter TT, which uses this signal as input (controlled variable) of the temperature controller TC. According to the set point chosen by the user, the controller computes the value of the manipulated variable, that is, the closing or opening of the valve that regulates the input of steam into the heating chamber. Inside the chamber, the steam condenses on the pipes that the juices pass through. After condensing, the steam leaves the heater as condensate by the lower pipe. There are

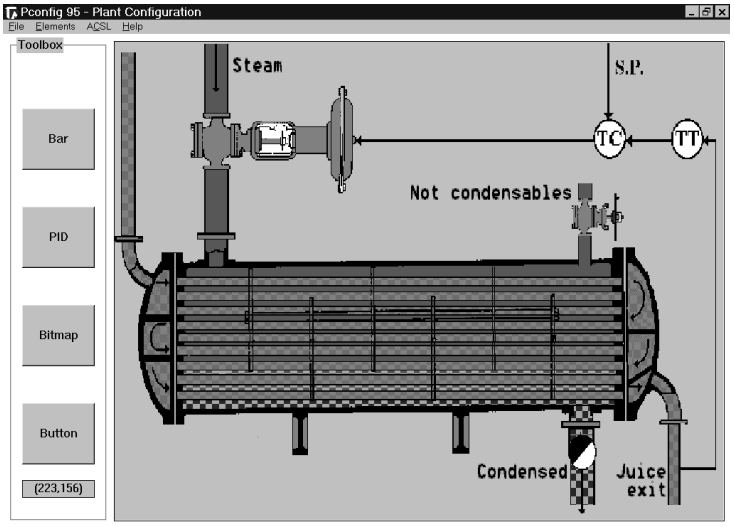

Figure 2. Development of a graphical interface using PConfig

three main disturbances that affect the interchange of heat between the juice and the heating steam: the flow and the temperature of the juice at the input of the chamber, and the pressure of the feeding steam. Another effect to be considered is the possibility of having non-condensable gases mixed with the steam. The consequence is to reduce the heat exchange between vapor and juice. This is a real effect that decreases the efficiency of the heat exchanger, and normally is avoided by using a manual valve to get the non-condensables out of the heating chamber.

Let's assume that we are interested in implementing a graphical simulation environment that allows the user to change the operating conditions (for example, set point, disturbances or tuning parameters of the controller) and to observe the results of the changes as a function of time.

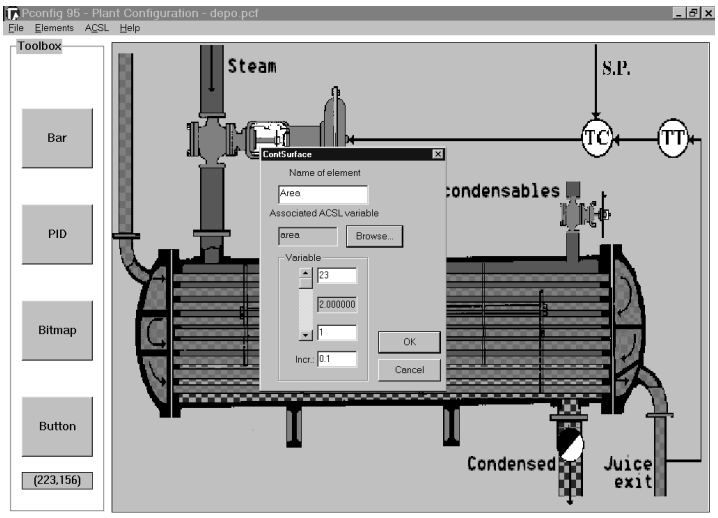

Figure 3. Scrollbar dialog window

3.1 Interface Generation

Once a simulation program (source code in a file with a .csl extension) of the given system has been written, the following steps must be followed in order to design the graphical interface of PConfig:

1. First, after starting PConfig, the user, using the option ACSL of the main bar, should introduce a path to indicate where ACSL is stored, and load the ACSL project that will support the interface functionality. From this point, the application analyzes the command file of the ACSL project, and the name of the procedures and variables in the simulation are available for linking to menu items and controls of the interface.
2. Then the user must load a previously drawn bitmap of the heat exchanger and its temperature control loop, using the button "Schematic" of the PConfig window. The bitmap appears in the interface generator tool with its final aspect, and can be used as reference for the location of the remaining controls. Figure 2 shows the appearance of the graphical interface at this point. Note the menu bar and toolbar out of the frame of this window, with the rest occupied by the bitmap with its basic aspect.
3. The next step, having the design in mind, is to place elements that allow the user to modify the set point, disturbances, etc., and to observe the results of the simulation. For instance, the user can select a scrollbar control that will open a related window and associate it to the variable of input flow of juice. For this purpose the "Examine" button will provide a list of the variables in the simulation. Once this is done, PConfig will read the initial value of the variable representing the input flow of juice in the ACSL project that will be the default value in the scrollbar. Next, the user must introduce minimum and maximum acceptable values for this variable, as well as a representative name, such as "Flow of juice, m3/h." Then the user must choose a location in the window for the configured scrollbar with the mouse, normally beside the representation of the input pipe of juice in the bitmap.

   The programmer can also place scrollbars for the temperature of the input juice, and set point of the output temperature controller. These scrollbars will allow the user to modify the values of the corresponding variables in the simulation within the selected range, using the mouse with the scrollbar in the usual way.

   Similarly, it is possible to define a button in order to manipulate the tuning parameters of the PID controller. In this case, selecting the PID option of the PConfig window will open a window like the one shown in Figure 3. In that window, the variables representing the gain, integral and derivative times can be defined, with the same procedure described above, as well as other details of this particular element such as initial state (auto/man), name, controller type (continuous/ discrete), etc. This button, given an appropriate name and placed on the controller of the schematic, can be identified easily by the user.

   There are other types of parameters that the user could modify, for example, the number of pipes inside the heat exchanger. As was said above, the interface generator tool offers the possibility of associating this parameter to a button in the graphical environment; when the user presses the button, a window appears that shows the actual value of it, and the range of possible values that it can receive. The possibility of "hiding" certain parameters adds modularity to the simulation environment. The designer of the interface can do this by selecting the "button" option in the toolbar, and choosing the kind of element he wants to associate to it. Another example of this is the valve type: the

user can select between different types of valves, checking the corresponding radio button associated to it.

4. Besides changing some numerical values, the designer might be interested in giving information about some particular aspects of the simulation using a text window. This can be achieved by first selecting a "button" in the toolbar at the left-hand side of the PConfig window, and then the option "Text" in the associated window. This will open another window where the desired text can be typed. The button will be given a name and each time the button is activated, the text will display.
5. The last step is to select the experiments that the user can use later. These experiments correspond with procedures defined in the ACSL command file of the project. There are two ways of implementing them: either associating them to a button using the corresponding option, or including them in the main bar, using the "Select experiments" option. An example of a procedure often associated to a button is one dedicated to plotting the results of an experiment. A "browse" button allows the user to select the experiments that will be included in the graphical interface. Finally, the design must be stored in a file with a .pcf extension, using the "Save" option of the main bar.

The final result in the configuration of the interface is shown in Figure 4. Notice the "M" buttons beside the graphical controllers. Pressing them makes it possible to move the controls to different positions during the design session, but they will not appear in the user interface at execution time.

The same simulation code can be used for a different purpose, for instance, for studying the effect of changing the design parameters (as sizes or materials). In this case, a different user interface should be designed and generated, showing only what is relevant for this particular use.

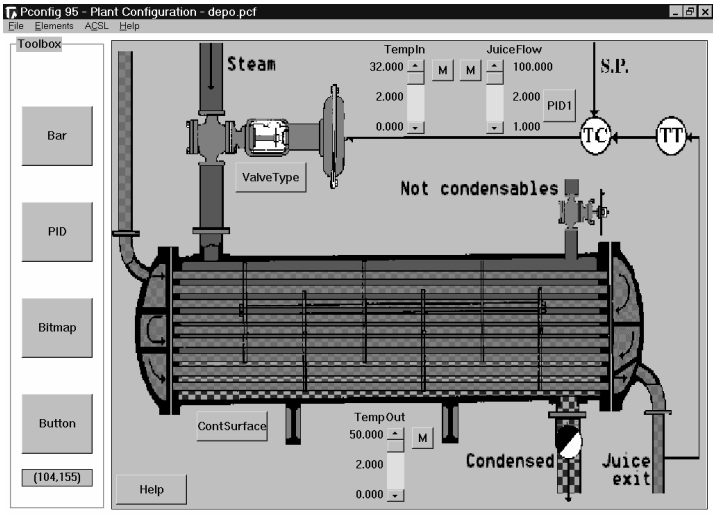

Figure 4. Final aspect of the interface

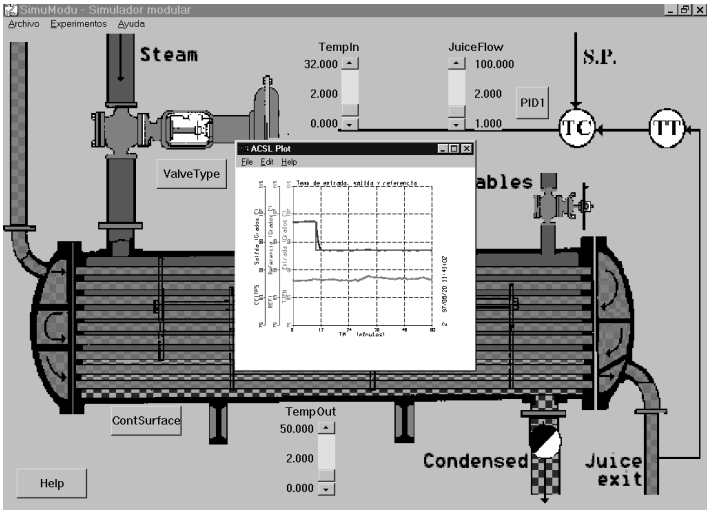

Figure 5. Running the simulation: note the graphic plot provided by ACSL

3.2 Running the Simulation

Once the simulation designer ends the graphical interface, he must save it on disk. The interface description file is a text file that describes it in terms of elements, their names and main properties, the position on the screen of the graphical controls, and so on, and can be edited separately in order to make minor changes if necessary.

To run the simulation, the user can start SimuModu and load the interface description file that contains the interface developed above (see Figure 4). Then the ACSL project will be run, a window with the same interface will appear, and a communication channel with ACSL will be opened.

Then the user can start to play with the simulation, using the scrollbars, buttons, etc., to make changes in the working conditions (always into the range allowed by the interface designer), and seeing the results provided by ACSL. For instance, the user can change the value of the temperature set point with the corresponding scrollbar, start the simulation, and see in graphical form the evolution of the main variables involved (for example, output temperature, set point, or valve opening) just by pressing a button (see Figure 5). The trends of these variables can be seen beside the schematic of the process, with no need to know the ACSL experiment language. In this concrete case, a step from 104.6ºC to 96.5ºC is imposed in the set point for the temperature of the juice at the output of the heat exchanger, and the values of the controlled and manipulated variables offered by the simulation are represented.

4. Conclusion

This paper introduces a software tool for using the ACSL-based simulation models in a convenient way, unbinding the model itself and its use, and complementing the module of experiments in the simulation

language. The proposed tool is suitable for those interested in producing simulation programs oriented to end-users who are not familiar with the simulation language or the details of the project implementation. In particular, it allows the generation of training simulators for educational and other fields. It consists of two main software tools: PConfig, oriented to the generation of the interfaces by the programmer of the simulation, and SimuModu, which implements the user simulation environment.

5. References

[1] Elmquist, H., Brück, D. and Otter, M. *Dymola, Dynamic Modeling Laboratory User's Manual, Version 3.0*, 1996.

[2] Acebes, L.F. "SIMPD, Sistema Inteligente de Modelado de Procesos Dinámicos." PhD Thesis, Universidad de Valladolid, 1996.

[3] *ACSL, Graphic Modeler Users Guide for Windows 3.1, 95 and NT, Version 4.2*, MGA Software, 1996.

[4] Petzold, C. *Programming Windows*, Fifth Edition, Microsoft Press, 1999.

6. Additional Reading

Cellier, F.E. *Continuous Systems Modeling*, Springer Verlag, Berlin, 1991.

de Prada, C. *Modelado y Simulación en Control de Procesos*, Dpto. Informática, Universidad Autónoma de Barcelona, Spain, 1990.

ACSL, Reference Manual, Version 11, MGA Software, 1995.

Pressman, R.S. *Software Engineering: A Practitioner's Approach*, 4th Edition, McGraw-Hill, New York, 1997.

Miguel A. García is a Teaching Assistant of Robotics and Control Technology at the University of Valladolid, Spain. His areas of research are distributed control, integration, real-time systems and simulation environments. He was a Visiting Researcher at the Fraunhofer Institute in Erlangen, Germany. Dr. García received a degree in Physics (Electronics) and a PhD from the University of Valladolid with highest honors. He is a Researcher at the Sugar Technology Center (CTA), where he has developed SICODI, a complete configurable distributed control system implemented in C++ under Windows NT. As his main project at the Center, he participates in the development of the Training Simulator for operators of whole beet-sugar plants.

Diego R. Llanos is currently a Teaching Assistant of Computer Architecture in the Department of Computer Science, University of Valladolid, Spain. He obtained his Bachelor's of Engineering and his Master's of Engineering degrees in Computer Science from the University of Valladolid, Spain, in 1994 and 1996, respectively. He was a Visiting Researcher at the Computer Science Department at the University of Illinois at Urbana-Champaign, USA. His research interests include heterogeneous process communications, parallel and distributed simulation, distributed virtual shared memory systems, cache-only memory architectures, and coherence maintenance protocols in distributed cache environments. He is a member of the Institute of Electrical and Electronics Engineers and the Association for Computing Machinery.

César de Prada is a Professor at the University of Valladolid, Spain, with the Department of Systems Engineering and Automatic Control. He obtained his degree in Physics (Electronics) in 1972 and, later, his PhD at the University of Valladolid. In 1987 he was appointed Full Professor in the Autonomous University of Barcelona, Spain, Department of Computer Science, where he directed the System Engineering Division. His research interests are centered on modeling and simulation, identification, and model predictive control. At present he is a member of the Spanish Commitee of IFAC, where he chairs its Simulation Technical Group.