EcosimPro: VIRTUAL INTERFACE FOR DEVELOPING SCADA TOOLS FOR CHEMICAL PROCESSES

Luis Taboada Antelo, Marcos Villafín Martínez, Julio Rodríguez Banga and Antonio Álvarez Alonso
Process Engineering Team, **IIM-CSIC**
C/ Eduardo Cabello 6, 36208- Vigo (Spain)

Abstract

With this work we aim to demonstrate the possibilities that make EcosimPro a suitable basis for the development of SCADA applications which can subsequently be used in different chemical processes.

Key Words: EcosimPro, SCADA, model, application, server, client.

1   INTRODUCTION

Most industrial plants have a Control Centre containing a Supervisory Control and Data Acquisition (SCADA) system which, as its name suggests, gathers data for the supervision and control of industrial processes.

The current market offers a wide selection of data acquisition, supervision and control software packages of which the evolutionary trend is threefold:

- Intercommunication between applications
- Standardisation of communications with field devices
- Adoption of the Internet communications interface

Prominent among the elements that characterise the current supervision and control software market is programming in Visual Basic for Applications (VBA) which, together with ActiveX technology, constitutes a practical, quick and efficient method of developing a customised interface. The incorporation of an ActiveX control adds to the screen an object with associated code which carries out a determined function in a completely integrated manner within the application we are using.

In this work we develop two types of applications, either for installation in the *server* (supplier of data) or in a remote device called *client* (receiver of data):

a.   The SCADA system installed in the plant *server* has been developed on the basis of a model and simulation carried out with EcosimPro

b.   With regard to the *client* application which facilitates remote monitoring and control, it should be borne in mind that a series of requirements has to be met:

- The graphics interface has to be very similar to the SCADA graphics interface
- It must facilitate display of the SCADA system status and be able to interact with it through the transmission of control instructions
- And, finally, is must be as reliable and efficient as possible

A database is created which acts as an intermediary between the SCADA application and the client application. All the values of the SCADA parameters are then recorded in this database and in this way the remote application can obtain these values in real time and send control instructions which the server application monitors, reading from the database and correcting any operating variations.

Figure 1: General System Outline

This complete interface is generated thanks to the possibilities of connection with other applications offered by EcosimPro, because it incorporates *ActiveX* (EcoViewer.dll) which provides the programmer with a series of tools which makes it possible to create any interface where the results of the experiments run in EcosimPro can be displayed and the variables which intervene in them can be manipulated.

2   MODELLING WITH EcosimPro

Since EcosimPro is a mathematical tool for modelling and simulating dynamic systems, it can be used to study real processes on both a small and a large scale. Taking this as a starting point it is evidently essential to have a mathematical model which adjusts as much as possible to the real situation. In this way we can generate an exact reproduction of any plant using EL, the language of EcosimPro.

The following are the steps to be followed:

1. Implementation of the mathematical model in EcosimPro

2. Generation of experiments in accordance with the needs of the SCADA application to be developed, taking into account both the initial values and the limits of the variables involved

The main necessity in the model lies in synchronising the simulation time and the real time because the fact that the SCADA application carries out its function in real time constitutes the objective of this work. To this end, when it comes to configuring the experiments, the following sentence must be included in the BODY section:

BODY
FOLLOW_RT= TRUE
...
END EXPERIMENT

This facilitates interactivity between the server and client applications because if the calculation process is run in computational time and it is decided to take any control action, such action would be very difficult to locate at a specific moment in time.

Likewise, the setup process of the main variables (controlled, manipulated and those associated with disturbances) used in the model is essential, since they will be the starting point to generate the SCADA application.

2.1   TWO TANKS IN SERIES WITH RECIRCULATION

In order to demonstrate that described above, we developed a simple system comprising two thorough-mix tanks connected in series and a recirculation system from tank 2 to tank 1. The aim is to establish level control in both tanks (by means of height control h_1 and h_2) and temperature control at the tank outlets (T_{1s} and T_{2s}) which, since the mix is considered to be thorough, will coincide with the temperature at any point. This type of control is representative of product quality control in many chemical processes.

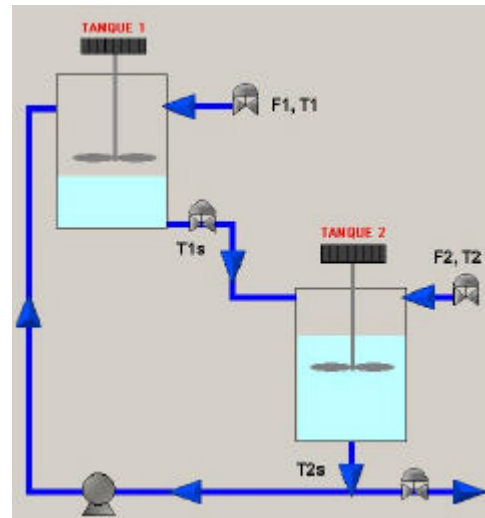The problem is illustrated in the following figure:

Figure 2: Outline of the System of Tanks in Series

This system considers four (4) *controlled variables* (h_1, h_2, T_{1s}, T_{2s}) and there are five (5) *manipulated variables* represented by the extent of opening of the different valves and the backflow [reflux???] pump illustrated in Figure 2.

With regard to possible disturbances that might affect the controlled variables we are only going to consider the temperatures of fresh feed to the two tanks (T_1 and T_2) and possible changes in the setpoints of the variables to be controlled (h_{1r}, h_{2r}, T_{1sr} and T_{2sr}).

The control loops to be implemented in this multivariable system were selected on the basis of a study using the **relative gain array (RGA)** or matrix of relative gains, defined in [1].

The RGA is defined as **a matrix of numbers**. The element *ij* in the matrix is called ?$_{ij}$. It is the ratio of the steady state gain between the controlled variable *i-ésima* and the manipulated variable *j-ésima* when all the other manipulated variables are constant, divided by the steady state gain between the same two variables when all the other controlled variables are constant:

$$?_{ij} ? \frac{\left| x_i / m_j \right|_{m_k}}{\left| x_i / m_j \right|_{x_k}} \qquad (1)$$

The RGA elements can be calculated for any system by means of the following equation:

$?_{ij} = (elemento\ ij\text{-}ésimo\ de\ \underline{\underline{K}}_p)(elemento\ ij\text{-}ésimo$

$$de\ ?\begin{Bmatrix}\underline{\underline{K}}_p^{?1}\end{Bmatrix}?^T\ ) \qquad (2)$$

where K_p is the steady state gain matrix in this system.

The nearer the $?_i$ values are to 1, the less interaction there will be (less difference between the open loop and closed loop system), so the pairs of variables are selected on the basis of those whose elements in the RGA are closest to 1.

The following are the control loops obtained for the process under study:

Table 1: Pairs of Variables Selected

Loop No.	Controlled Variable	Manipulated Variable
Loop 1	h_1	Opening of tank 1 outlet valve
Loop 2	h_2	Opening of fresh feed valve to tank 2 (F_2)
Loop 3	T_{1s}	Opening of fresh feed valve to tank 1 (F_1)
Loop 4	T_{2s}	Pumping of recirculating flow

As a result of this selection, the resulting configuration of the process is as shown in Figure 3:
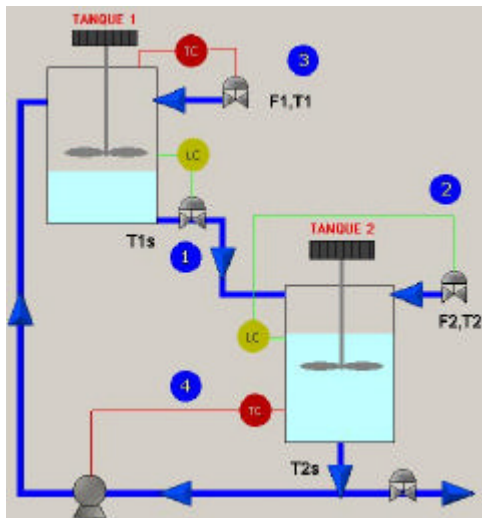

Figure 3: Sketch of the System considering Control Loops

It is assumed that all the controllers used are of the PI (proportional-integral) type, adjusted by the *frequency response* method. More information on the different alternatives for determining the parameters of different types of controllers is included in [5].

Once the complete mathematical model has been built and the loops and control parameters have been selected, the next step is to implement it in EcosimPro.

To do this we create a component called *tanques_en_serie*, which reflects the mass and energy balances in the two tanks together with the control setpoints so that the process is accurately reflected. The complete code for this simple component is included in Appendix A.

By means of a series of predefined gates in the controllers which incorporate the CONTROL library of EcosimPro, the four controllers interconnect with the *tanques_en_serie* component and generate a new component denominated *control_tanques _en_serie*. This component is then used to carry out the experiments which will serve as a basis for the SCADA application.

The following is the code for this component:

```
----------------------------------------------------------------
USE CONTROL

COMPONENT control_tanques_en_serie

TOPOLOGY
      tanques_en_serie TS
      Cntrl_pi PI (k=-0.175, Ti=12.64, u_min=-1)
      Cntrl_pi PI2(k=-0.109, Ti=11.375, u_min=-1)
      Cntrl_pi PI3(k=-0.10, Ti=10.554, u_min=-1)
      Cntrl_pi PI4(k=-0.12, Ti=10.04, u_min=-1)

      CONNECT TS.nivel1 TO PI.s_var
      CONNECT TS.href1 TO PI.s_set
      CONNECT PI.s_out TO TS.he
      CONNECT TS.nivel2 TO PI2.s_var
      CONNECT TS.href2 TO PI2.s_set
      CONNECT PI2.s_out TO TS.F2e
      CONNECT TS.T1sal TO PI3.s_var
      CONNECT TS.T1sref TO PI3.s_set
      CONNECT PI3.s_out TO TS.F1e
      CONNECT TS.T2sal TO PI4.s_var
      CONNECT TS.T2sref TO PI4.s_set
      CONNECT PI4.s_out TO TS.be

END COMPONENT
```

The values of the gain and time integers defined in the component will be defined by the application by default.

The following is the base experiment for the development of the SCADA applications:

```
------------------------------------------------------------------
EXPERIMENT exp2 ON control_tanques_serie.default

DECLS

INIT    -- set initial values for variables
    -- Dynamic variables
  TS.h1 = 1.361
  TS.h2 = 2.777
  TS.x = 110.747
  TS.r = 42.682
  PI.vi = 0
  PI2.vi = 0
  PI3.vi = 0
  PI4.vi = 0


BOUNDS    -- set expressions for boundary variables: v =
         f(t,...)
  TS.T1 = 0
  TS.T1sr1 = 0
  TS.T2 = 0
  TS.T2sr1 = 0
  TS.hr1v = 0
  TS.hr2v = 0

BODY
  FOLLOW_RT= TRUE
  TIME = 0
  TSTOP = 360000
  CINT = 0.5
  INTEG()


END EXPERIMENT
------------------------------------------------------------------
```

The following are the main characteristics of this experiment:

1. The aforementioned sentence of correspondence between the simulation time and the real time

2. The absence of the order to generate a results report because it is not necessary. The data will be automatically saved in the database which will serve as the centre of the SCADA system

3. The duration of the simulation is considered sufficient from a practical point of view

4. A short sampling time is selected in view of the speed with which the system responds to the disturbances; this means that the complete dynamic response cannot be observed for longer times and monitoring loses quality

When the experiment is being compiled, an executable file is generated (exp2.exe) which performs the function of the process plant considered by transmitting and/or reading the values of the variables which intervene in the simulation.

This executable file and the tools available in EcosimPro make it easy to develop the monitoring,

supervision and control system necessary for the process in question.

3   DESIGN AND DEVELOPMENT OF THE SCADA APPLICATION

Microsoft Office applications incorporate *Visual Basic for Applications* and this has several advantages:

- It is very widely used
- It is accepted by many manufacturers and is becoming a benchmark
- It has a good power—user friendly ratio

Microsoft Visual Basic 6.0 is used to develop the process.

3.1   COMMUNICATION BETWEEN EcosimPro AND VISUAL BASIC

As illustrated in the introduction, *ActiveX* technology is a simple solution to generating an *interface* with which to execute the EcosimPro experiments, as it provides the necessary tools to enable interaction with the experiments from the programming interface.

In this particular case, *ActiveX* is supplied as a Dynamic Link Library (DLL) called EcoViewer.dll. This library contains the objects necessary to program the application based on the experiment generated from a model implemented in EcosimPro.

As soon as EcosimPro has been installed the EcoViewer can be used because it can then access the dynamic and other libraries it requires. Simulations are run with the models built in EcosimPro. To run them we need the model executable file and its associated files *(.txt)* which are directly obtained from the compilation of the experiment. These .txt files are essential as they contain all the variables and equations which define the process model.

This work also contemplates the possibility that the application does not have direct dependency on EcosimPro; in other words, that the software is not installed in the machine. To solve the matter, EA International supplies a *setup.exe* to install everything necessary so that the interfaces developed in VB based on EcosimPro models work correctly.

3.2   DESIGN OF THE SCADA APPLICATION GRAPHICS INTERFACE

The design of the SCADA application graphics interface is based on the real design of the process into which it is integrated, as well as on the real supervision and control requirements of determined variables.

In the example of tanks in series we opted for a dynamic graphics interface in *Visual Basic 6.0*, both for the server and for the client.

The *server* application is shown in Figure 6 along with the main actions the user can carry out with the graphics interface.

This application has two basic operating modes -without control and with control- each of which is developed on the basis of a previously generated experiment:

a. **Without control:** By selecting the option *Without control* from the *Control Options* in the graphics window, the user can study the dynamic response of the system vis-à-vis disturbances in the flows and temperatures of the fresh feeds to the tanks, observing the speed with which the system illustrates and propagates the disturbances. This is a fundamental step for the subsequent implementation of control systems.
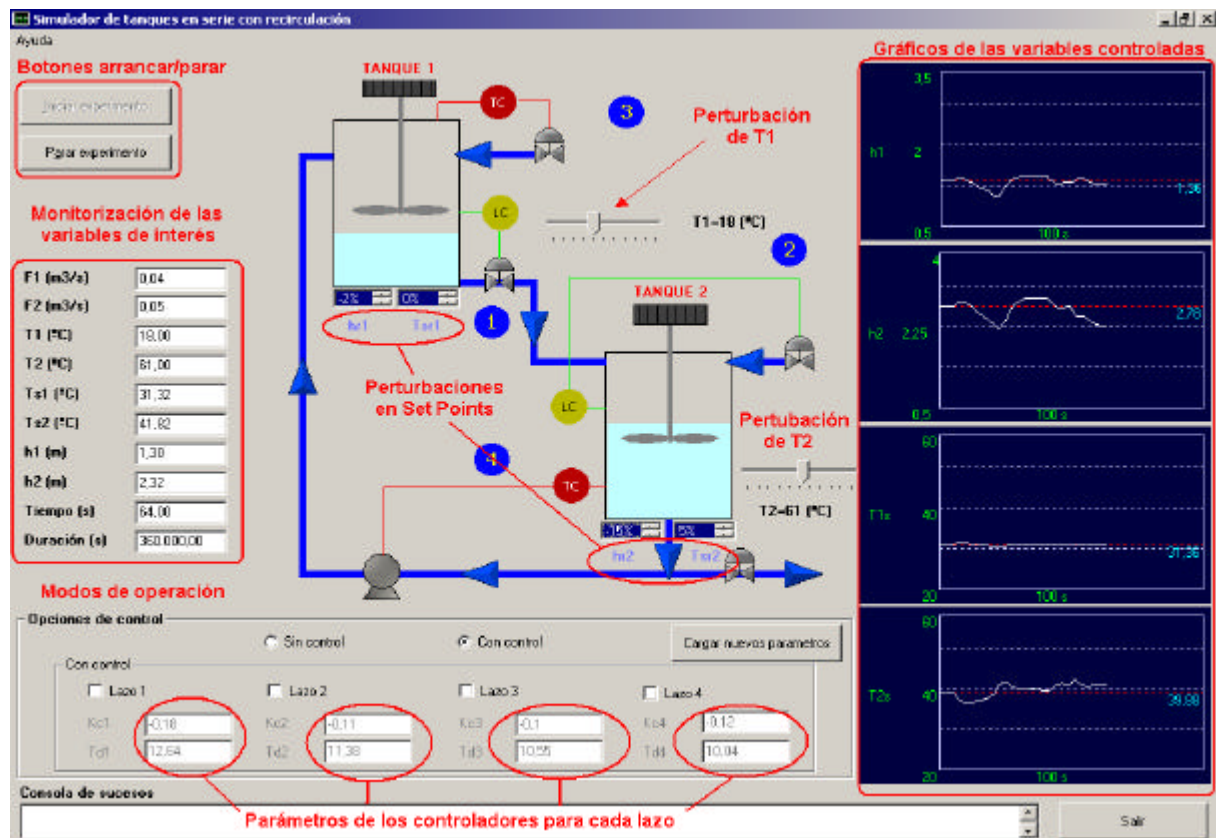

Figure 6: Graphics Window of the Server Application

b. **With control:** When *With Control* is selected from the *Control Options* the graphics window displays the different control loops that are implemented. In addition, the control parameters for each of the controllers can be edited and modified.

With regard to the disturbances, the fresh feed flows cannot now be manipulated as they are converted to manipulated variables of loops 2 and 3. However, modifications can be made to the setpoints of each of the controlled variables.

The program has a *Help* menu which gives more information on the operating modes.

Of vital importance in both operating modes is the monitoring function offered in the interface in the

form of dynamic graphics and tables which are updated over time, since possible system control decisions are made on the basis of such information.

In respect of the *remote* interface (based on the server application), it was decided to simplify the graphics display by eliminating the operating mode WITHOUT CONTROL. This level is basically of interest for process monitoring and control, not for its dynamic study which is supposedly a step prior to the implementation of the SCADA application.

3.3 THE BASICS OF PROGRAMMING THE INTERFACE

To create the interface it is first necessary to make reference to the EcoViewer.dll library which, as indicated previously, is the ActiveX component

C215-5

responsible for establishing Visual Basic—EcosimPro communication. In order to achieve this the DLL must be recorded, and this will already have been done if EcosimPro is installed or if the aforementioned *setup.exe* has been run.

On the premise that this condition has been met, from the programming display we access the VB menu *Project > References*. In the *References* dialogue box we select the option *EcoViewer Monitor EcosimPro 3.2* as shown in Figure 7. We click on *Accept* and the object is ready for use.
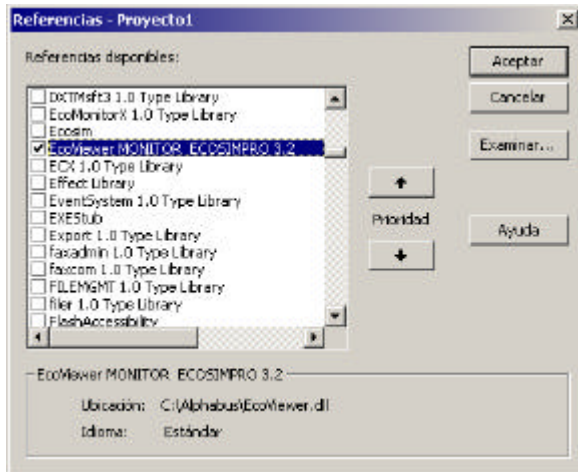

Figure 7: Dialogue box to Select the DLL

The following is a brief description of how to load an EcosimPro experiment in an object variable. We will use the simple system of tanks in series to explain the procedure:

1. *Declaration of the object variable* (Experiment):

```
Public    WithEvents   Experiment   As
EcoViewer.EcoSimViewer
```

2. *Creation of an instance of the object variable* Experiment *(in* Form_Load, an event which unfolds when the form is loaded):

```
Set     Experimento    =    New
EcoViewer.EcoSimViewer
```

3. *Assignment of a text box* (Textbox) *to follow the experiments:*

```
Set Experiment.WindowOuput = Text1
```

4. *Start experiment*:

```
ArchivoExp = App.Path & "\exp" & Expe &
".exe"
Experimento.LaunchSimulationFile ExpeFile
```

In the **ArchivoExp** variable in the first sentence, we introduce the path of the experiment executable file. *App.Path* returns to the path of the application file so the experiment should therefore be kept in that file.

The auxiliary variable **Expe** is used to determine the two operating modes considered in the executable file so that if the option WITHOUT CONTROL is selected, **Expe** will be equal to 1 (loading file exp1.exe) and if WITH CONTROL is selected, **Expe** will be equal to 2 (exp2.exe).

The second sentence links the experiment with the object variable.

If the experiment has been satisfactorily loaded it will go on to the next stage.

5. *The event* Experimento_Connected:

```
v.Vars.Add Vars.Item(32).Name
LoadValores v
End If

If OptControl(2).Value = True Then
v.Vars.Add Vars.Item(61).Name
v.Vars.Add Vars.Item(64).Name
v.Vars.Add Vars.Item(69).Name
v.Vars.Add Vars.Item(73).Name
v.Vars.Add Vars.Item(37).Name
v.Vars.Add Vars.Item(52).Name
v.Vars.Add Vars.Item(82).Name
v.Vars.Add Vars.Item(84).Name
v.Vars.Add Vars.Item(132).Name
v.Vars.Add Vars.Item(133).Name
.Vars.Add Vars.Item(89).Name
v.Vars.Add Vars.Item(77).Name
v.Vars.Add Vars.Item(8).Name
v.Vars.Add Vars.Item(22).Name
v.Vars.Add Vars.Item(36).Name
v.Vars.Add Vars.Item(51).Name
LoadValores v
v

Set v = Nothing

Set v = Views.Add("CONTROLADORES")
 v.Vars.Add Vars.Item(7).Name
v.Vars.Add Vars.Item(1).Name
v.Vars.Add Vars.Item(21).Name
v.Vars.Add Vars.Item(15).Name
v.Vars.Add Vars.Item(35).Name
v.Vars.Add Vars.Item(29).Name
v.Vars.Add Vars.Item(50).Name
v.

Vars.Add Vars.Item(44).Name
LoadParametros v

End If

End Sub
```

This event unfolds if the experiment—object connection has been successful. If so, the **StateConnect** variable takes the value 4 with which it can generate views and then create object variables which support the content of these **Views** as well as each of their associated variables (**Vars**).

In the particular case of the example of the tanks, as it is possible to choose between two different operating modes a sentence must be added to distinguish between them. Of the two *Control Options* on the graphics interface (see Figure 6), the one that is selected will take on the true value. For example, OptControl(1).Value = True will be true if the option WITHOUT CONTROL is selected.

Based on the selection, the program assigns a view where it loads the names of variables which are of interest from the point of view of the SCADA application. For the option WITH CONTROL, a second view is necessary (CONTROLLERS) where the variables associated with the controller parameters are loaded. This will facilitate manipulation of the variables because the monitoring actions are separate from possible actuations at controller level. The second view (VISTA2) is created taking into account the variables which are going to be required for the option selected (with control or without control). The following is the instruction which allows the name of a variable to be added to the view:

v.Vars.Add Vars.Item(n).Name

where n is the position which EcosimPro assigns to the variable in the model. These positions are reflected in the text file required by the executable file (with the extension .default).

After adding the required variables to the views, a procedure is called to assign the value of each variable in the view –passed as a parameter- to a numerical matrix. The following is the code for this operation, exemplifying VISTA2:

```
Private Sub LoadValores(v As ecmView)

Dim vVars As ecmViewVars
Dim vVar As ecmViewVar
Set vVars = v.Vars
Dim i As Integer
i = 1
  For Each vVar In vVars
    NombresVal(i) = vVar.Name
  ValoresIni(i) = vVar.ExpVar.Value
  i = i + 1
  Next

End Sub
```

where ValoresIni(i) is the numerical matrix which stores the values of the variables required and NombresVal(i) is a matrix where the names of the selected variables are stored and which will be used to assign the values.

Setup values have to be determined for certain variables to establish a starting point for the simulation. This can be done either in the experiment or with a code in the program. The second alternative offers the advantage of being able to modify the initial conditions without having to rerun the experiment, a task which would be impossible in a machine where EcosimPro is not installed. The next function assigns new values so it will suffice to call the function, passing as a parameter (i) the position which the variable occupies in the matrix of values (ValoresIni). Before the values can be assigned the names of the variables to be modified have to be known, which is why a matrix was created to store the names.

```
Dim NuevoValorVariable As Variant
Dim mVarType As Variant

Dim v As ecmExpVar
On Error GoTo MiError
NuevoValorVariable = ValoresIni(i)
Set v = Nothing

mVarNombre = NombresVal(i)
AsignarValor = 0
If mVarNombre <> vbNullString Then
Set v  = Experimento.ecmExpVars
(mVarNombre)
```

```
  mVarType = v.TypeDef
 Select Case mVarType
     Case ecmString
AsignarValor = CStr(NuevoValorVariable)
     Case ecmReal
AsignarValor = CDbl(NuevoValorVariable)
     Case ecmInteger
AsignarValor = CLng(NuevoValorVariable)
     Case ecmBoolean
AsignarValor = CBool(NuevoValorVariable)
     Case Else
AsignarValor = 0
   End Select
     End If
     v.Value = AsignarValor
     On Error GoTo 0
     Exit Function

   MiError:
     If v Is Nothing Then
      MsgBox "La variable no pertenece al
experimento", vbInformation, "ALERTA"
     End If
     AsignarValor = 0
```
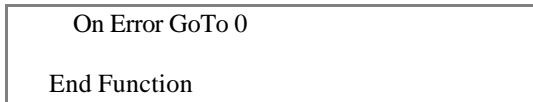
```
On Error GoTo 0

End Function
```

The development of this function is simple: The value and the name of the variable in question is stored in two temporary variables, so that through its name the datum type can be defined by an EcosimPro function (*TypeDef*) and from here, a value can be assigned in accordance with the type.

6. *Start, Stop and Pause Experiment:*

At this point in the development of the application the experiment can be started with the following instruction:

That described earlier for **VISTA2** is also applicable to the other views generated, whether in the example shown **(CONTROLLERS)** or in other systems which have been studied.

```
Experimento.Run
```

The same sentence restarts the experiment if it has been paused.

In the same way, the simulation can be easily paused and/or stopped by means of the following sentences:

```
Experimento.Break
Experimento.EndSimulate
```
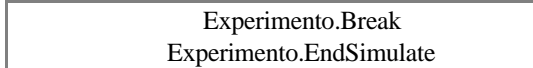
A simple way of interacting with these sentences is by associating them with buttons, as in the example shown in this work. All the sliders and windows for selecting disturbances (see Figure 6) have a common operating pattern based on the following sequence:

Figure 8: Logical Sequence of the Operation of Controls on the Graphics Display
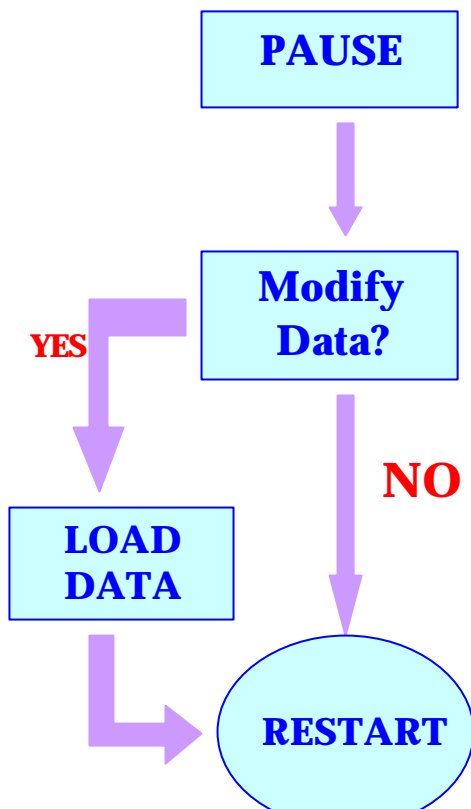
The new aspect here is the process of **loading new data** into the experiment. It is a simple operation, very similar to the setup process described earlier for variables. The data are loaded when the user introduces the new value in the graphics display (using the sliders, selection windows, etc) and it is assigned to the corresponding position in the previously defined matrix of values. Since the value has been updated in the matrix, the position of this value is taken and the assignment function is called to introduce the new value into the model.

3.4   CONNECTIVITY   THROUGH   THE   DATABASE

Once we have come this far, it seems logical to consider the possibility of remote monitoring, supervision and control of the process. It is this need that gives rise to the development of the client application.

The client application must be completely independent of EcosimPro because its data source lies in the server application. This is why a database has to be used which carries out the functions of intermediary between the server and the client.

The database has been created in Microsoft Access 2000. It was created with two tables to continuously record the values of the variables belonging to each of the previously defined views, VISTA2 and CONTROLLERS.

Parameters TABLE

Nombre del campo	Tipo de datos
Kc1	Numérico
Td1	Numérico
Kc2	Numérico
Td2	Numérico
Kc3	Numérico
Td3	Numérico
Kc4	Numérico
Td4	Numérico

ValoresIni TABLE

C215-8

Nombre del campo	Tipo de datos
F1	Numérico
F2	Numérico
T1	Numérico
T2	Numérico
T1s	Numérico
T2s	Numérico
h1	Numérico
h2	Numérico
Tiempo	Numérico
Duracion	Numérico
b	Numérico
k1	Numérico
h1r	Numérico
h2r	Numérico
T1sr	Numérico
T2sr	Numérico

Figure 9: Tables Defined in the Database

These tables will be used to store values from the server application based on the values generated by the experiment. The client application will gain access by consulting these values and using them to monitor the process. Likewise, when a control instruction is implemented in the client application the fields involved will immediately be updated in the database and therefore the server application must also monitor the database to detect changes at any instant.

3.4.1 Server—Database Interaction

The server has to take a reference to save the simulation results and must therefore take into account the refresh rate.

The event Experimento_RefreshView unfolds each time the integration time previously defined in the body of the experiment in EcosimPro elapses (CINT) and the values of the variables are updated. Taking advantage of this situation, it is easy to understand that this is the right moment to check whether the last record in each of the database tables has been updated. This can be done by comparing the last two records field by field. If they are different, then logically a change will have taken place that will have to be taken into account, assigning the values of the last record.

The following is the general code for this procedure:

```
Private Sub Experimento_RefreshView(sViewName
As String, dTime As Double, VarValues As Variant)
…
 AdoRs.MoveLast
 ValorX= AdoRs.Fields(i)
 AdoRs.MovePrevious
If AdoRs.Fields(i).Value <> ValorX Then
  ValoresIni(i) = ValorX
  AsignarValor (i)
End If
…
```

which executes the following tasks:

1. Shift to the last database record

2. Assignment of the value from the field **i** to the variable **ValorX** (time variable)

3. Shift to the penultimate database record

4. Comparison of the value in field **i** of the penultimate record with the time variable. If *they are different* (there has been a change), it is assigned to position **i** of the matrix of values of the selected variables and the assignment function is called

The read/write devices on the graphics display (monitoring windows, sliders, etc) are then modified in accordance with these new values.

The process of writing to the database also takes place when this event unfolds. The following briefly illustrates how this operation is carried out in Visual Basic:

```
Private Sub Experimento_RefreshView(sViewName
As String, dTime As Double, VarValues As Variant)
 …

 AdoRs.MoveLast
 AdoRs.MoveNext

       AdoRs.AddNew
       AdoRs.Fields(i) = VarValues(i)
       AdoRs.Update
…
```

This series of commands brings about the following actions:

1. Shift to the last database record

2. Establishing the conditions to be able to insert a new record

3. Creation of a new record

4. Introduction of the new values of the variables in the associated fields

5. Close updating

To answer or clarify any query regarding the programming of databases in VB, please see [2], [3] and [4].

C215-9

3.4.2 Client—Database Interaction

To simplify the task of developing the client application based on the server, we considered the option of using an *ADO* object (see [2] and [3]). This makes client follow-up of the database exceptionally easy because it is merely a question of linking the text tables with the *ADO* object fields to be viewed and/or controlled. This is achieved with a *Timer* control which is simply an event to which we assign a time interval after which the event unfolds and the values are updated. Updating is obtained by means of the following code:

```
Adodc1.Refresh
Adodc1.Recordset.MoveLast
```

For the case under study, the refresh interval value is set to 250 milliseconds. This value will be in accordance with the integration time considered in the base experiment, as it has to be fast enough to detect possible changes in the model.

New values are directly assigned to the variables of the desired objects, programming what is considered the most opportune event.

For example, the syntax of a slider code is as follows:

```
Private Sub Slider_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

Adodc1.Recordset.Fields(i) = Slider.Value
Adodc1.Recordset.Update
Adodc1.Refresh
Adodc1.Recordset.MoveLast

End Sub
```

where:

1. The slider value is assigned to the corresponding field in the database table

2. The update command is applied

3. The values of the object are refreshed so that they are active

4. There is a shift to the last record

4   CONCLUSIONS

With the connectivity functions offered by EcosimPro, it is simple and easy to use this software as a basis for working in programming environments aimed at the development of SCADA applications, using as the nexus between the server and the different clients a database and the possibilities of handling such applications that Visual Basic offers.

In this work we have exemplified the complete development using two tanks in series with recirculation. This type of example could serve as a starting point for:

a. The creation of a series of small-scale applications which represent simple units or processes which can subsequently be easily integrated to form more complex processes or plants

b. The development of web applications which would constitute a means both for learning the processing represented and for carrying out the real functions of a SCADA application (monitoring, control and supervision) via the Internet

References

[1] Bristol, E.H. (1966) *On a new measure of interactions for multivariable process control*, IEEE Transactions on Automatic Control, volume CA-11, pp. 133-134.

[2] Havolrson, M. (1999) *Learn Visual Basic 6.0 Now*, McGraw-Hill, Madrid.

[3] McManus, J.P. (1999) *Databases with Visual Basic 6*, Prentice-Hall, Madrid.

[4] Microsoft Press (1998) *Microsoft Visual Basic 6.0. User Manual*, McGraw-Hill, Madrid.

[5] Ogunnaike, B.A., Harmon, R. (1994) *Process Dynamics, Modelling and Control,* Oxford University Press, New York.

APPENDIX A   CODE FOR THE COMPONENT tanques_en_serie

USE CONTROL

COMPONENT tanques_en_serie

PORTS

        IN analog_signal he
        IN analog_signal F2e
        IN analog_signal F1e
        IN analog_signal be
        OUT analog_signal href1
        OUT analog_signal nivel1
        OUT analog_signal href2
        OUT analog_signal nivel2
        OUT analog_signal T1sref
        OUT analog_signal T1sal
        OUT analog_signal T2sref
        OUT analog_signal T2sal

DATA

--Datos de caudales

REAL F1r=0.05   --Caudal de referencia de la corriente entrada al tanque 1 (m3/s)
REAL F2r=0.05   --Caudal de referencia de la corriente entrada al tanque 2 (m3/s)
REAL k1r=0.1   --Constante de proporcionalidad (en e.e.) entre el Fsalida y h en el tanque 1 (m2/s)
REAL k2r=0.1   --Constante de proporcionalidad (en e.e.) entre el Fsalida y h en el tanque 2 (m2/s)
REAL T1r=20   --Temperatura de referencia de F1 (ºC)
REAL T2r=60   --Temperatura de referencia de F2 (ºC)
REAL T1sr=31.36 --Temperatura de referencia para la salida del tanque 1 (ºC)
REAL T2sr=39.88 --Temperatura de referencia para la salida del tanque 2 (ºC)
REAL br=0.4   --Valores de referencia para la definición del reciclo desde la salida del Tanque 2 al Tanque 1 (m2/s)
REAL gr=0.6

--Datos geométricos de los tanques

REAL At1=0.5   --Área transversal del tanque 1 (m2)
                REAL At2=0.2   --Área transversal del tanque 2 (m2)
REAL hr1=1.361 --Altura del tanque 1 en el estado estacionario (m)
REAL hr2=2.777 --Altura del tanque 2 en el estado estacionario (m)

DECLS

REAL F1   --Caudal de la corriente de entrada al tanque 1 (m3/s)
REAL F2   --Caudal de la corriente de entrada al tanque 2 (m3/s)
REAL T1   --Temperatura de F1 (ºC)
REAL T2   --Temperatura de F2 (ºC)
REAL T1s   --Temperatura de salida del tanque 1 (ºC)
REAL T2s   --Temperatura de salida del tanque 2 (ºC)
REAL h1   --Altura de líquido en el tanque 1 (m)
REAL h2   --Altura de líquido en el tanque 2 (m)
REAL k1   --Grado de apertura de la válvula de salida del tanque 1
REAL b   --Valores de definición del reciclo
REAL g
REAL z   --Variable auxiliar para el control de la altura en el tanque 1
REAL y   --Variable auxiliar para el control de la altura en el tanque 2
REAL x   --Variable auxiliar para el cálculo de T2s
REAL r   --Variable auxiliar para el cálculo de T1s
REAL p   --Variable auxiliar para el control de T1s
REAL m   --Variable auxiliar para el cálculo de T2s
REAL hr1v   --Perturbación en el nivel de referencia del tanque 1 (m)
REAL hr2v   --Perturbación en el nivel de referencia del tanque 2 (m)
REAL T1sr1 --Perturbación en la temperatura de referencia del tanque 1 (m)
REAL T2sr1 --Perturbación en la temperatura de referencia del tanque 2 (m)

INIT

CONTINUOUS

$h1'=(F1+(b*k2r*sqrt(h2))-(k1*sqrt(h1)))/At1$

$h2'=(F2+(k1*sqrt(h1))-(b*k2r*sqrt(h2))-(g*k2r*sqrt(h2)))/At2$

x'=((T1s*k1*sqrt(h1))+(T2*F2)-
(T2s*k2r*sqrt(h2)*b)-
(g*k2r*sqrt(h2)*T2s))/(At2*hr2)

T2s=x/h2

r'=((F1*T1)+(b*k2r*sqrt(h2)*T2s)-
(k1*sqrt(h1)*T1s))/(At1*hr1)

T1s=r/h1

--Control de la altura en el tanque 1

nivel1.signal=h1
href1.signal=hr1v
z=he.signal
k1=k1r+z

--Control de la altura en el tanque 2

href2.signal=hr2v
nivel2.signal=h2
y=F2e.signal
F2=F2r-y

--Control de la temperatura de salida del tanque 1
(T1s)

T1sref.signal=T1sr1
T1sal.signal=T1s
p=F1e.signal
F1=F1r+p

--Control de la temperatura de salida del tanque 2
(T2s)

T2sref.signal=T2sr1
T2sal.signal=T2s
m=be.signal
b=br+m
g=1-b

END COMPONENT