AIRES: A STANDARD FOR WEB-BASED REMOTE EXPERIMENTS

Marco Casini [*] **Alberto Leva** [**] **Francesco Schiavo** [**]

[*] *Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy*
Email: casini@ing.unisi.it
[**] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy*
Email: {leva,schiavo}@elet.polimi.it

Abstract: This paper proposes a standard for the communication between a remote laboratory and the clients willing to use that laboratory. The ultimate goal of the research is to allow anyone to use any experiment, without the need to use the specific interface provided by the laboratory site that offers each experiment. The authors strongly believe that adopting such a standard would greatly enhance the effectiveness of remote experiments. The study presented herein is (of course) preliminary, and another purpose of this paper is to promote and stimulate further research on the matter.

Keywords: Control education, distance learning, web-based learning, remote laboratory.

1. INTRODUCTION AND PROBLEM STATEMENT

Dozens of remote experiments are nowadays available on the web, see e.g. (Poindexter and Heck, 1999; Dormido Bencomo, 2004; Gasperini *et al.*, 2004) and many other works. To use them, however, one has to employ (and therefore to learn) the operator interface provided by the site that hosts the experiment (termed from now on the 'Experiment Server' or ES). In addition, in the great majority of cases the remote user is only allowed to change some parameters in control schemes that are already implemented in the ES, and there is no possibility to specify one's own schemes. Finally, there is no uniform way to bring the experiment data from the ES to the location whence it is employed by the user (termed herein the 'Experiment Client' or EC). All these facts, with various degree of relative importance depending on the particular situation and user type, diminish the usefulness of most remote experiments.

This paper firstly aims at raising the problem sketched above, in the hope that the scientific community will undertake the challenge to solve it. In addition, a preliminary proposal for the definition of a suitable standard (AIRES, standing for Architecture Independent Remote Experiments Standard) is formulated, some solutions designed in accordance to that proposal are described, and finally some application examples are briefly reported.

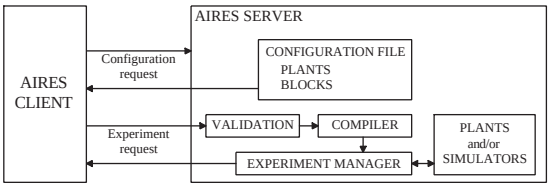

Fig. 1. Scheme of the AIRES architecture.

To state the problem addressed by AIRES, assume that several ESs exist in the net, each of them composed of a computer accessible via the web and some simulated or physical experiments. Details on the ES architecture are inessential at this stage. The purpose of the AIRES project is to allow any potential EC (that is, any computer connected to the Internet) to query an ES to know which experiments are available, design a control scheme for one of those experiments, perform an experiment with that control scheme, and retrieve the generated data, without the need to learn any specific user interface and without the necessity of any software other than a browser.

The EC must not be constrained to use any specific editor (in the broadest sense of the term) to build control schemes, and any detail concerning both the ES and the EC architecture, operating system, and so forth must be absolutely transparent for the user.

2. ARCHITECTURAL OVERVIEW OF AIRES

Any communication in AIRES is textual and file-based, and is centered on two formats. The first one is the AIRES Server Configuration file (ASC), an ASCII file that any ES publishes on the web, which describes the available experiments and the blocks that can be used to compose control schemes. The second one is the AIRES Client Experiment Request file (ACER), that is uploaded by an EC to an ES to require that an experiment be run. In this paper we only refer to batch experiments, though AIRES can be extended to interactive ones without excessive difficulties. Some remarks in this respect are given in Section 4). The AIRES architecture is summarized in Figure 1.

An AIRES ES is defined, from the standpoint of the EC, by its configuration file. Space reasons oblige in this paper to illustrate the matter *per exemplum*. Therefore, a typical ASC file is reported and briefly commented in the following. The example defines some blocks and one plant, namely the simple temperature control presented in (Leva, 2002) and shown in Figure 2, in which the two transistors heat a small metal plate while the fan provides cooling. The outputs are the measurements of the temperatures of the transistors and of the plate, while the inputs are the commands to the transistors and to the fan.

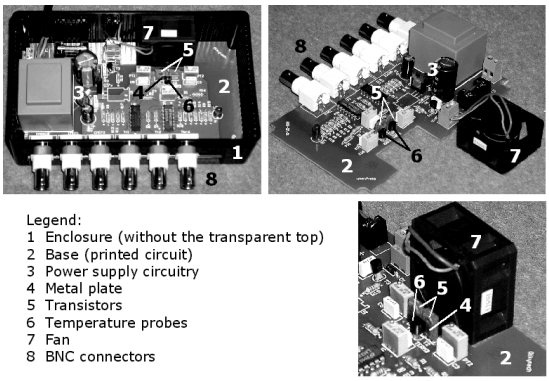

Fig. 2. The thermal apparatus referred to in the ESSC file example.

```
{GENERAL
 (SITE AIRESexample)
 (URL http://...)
 (DESCRIPTION brief example of AIRES ES
  configuration)
}
{BLOCKS
 (ISAPID2dof
  (DESCRIPTION 2-d.o.f. ISA PID)
  (INPUTS
   (SP  , real, req, Set point)
   (PV  , real, req, Process Variable)
   (TS  , bool, def, F, Track Switch)
   (TR  , real, def, 0, Track Reference)
   (NoUp, bool, def, F, CS increment lock)
   (NoDn, bool, def, F, CS decrement lock)
  )
  (OUTPUTS
   (CS   , real, Set point)
   (HIsat, bool, High Sat. Signal)
   (LOsat, bool, Low Sat. Signal)
  )
  (PARAMETERS
   (K , real Gain)
   (Ti, real Integral time)
   (Td, real Derivative time)
   (N , real ratio between Td and the 2nd
        pole time constant)
   (b , real SP weight in the P action)
   (c , real SP weight in the D action)
  )
 ) // end ISAPID2dof
 (STEPGEN
  (DESCRIPTION Step generator
  (OUTPUTS
   (Out, real, Output signal)
  )
  (PARAMETERS
   (Vini, real, Initial value)
   (Vfin, real, Final value)
   (Tstep, real, Step time)
  )
 ) // end STEPGEN
 // other blocks (e.g. transfer function, sum,
 // product, logic elements, nonlinear
 // functions, and so on)
} // end BLOCKS section
{PLANTS
```

```
  (MultiVarTC
  (DESCRIPTION Multivariable thermal apparatus
   described in A. Leva, "A Hands-on Experimental
   Laboratory for Undergraduate Courses in
   Automatic Control", IEEE Transactions on
   Education, 46(2), 2003, 263-272)
  (MINTIMESTEP 0.05 s)
  (MAXTIMESTEP 2 s)
  (MAXDURATION 1200 s)
  (INPUTS
    (Q1, real,Unit,0,100,Command to transistor 1)
    (Q2, real,Unit,0,100,Command to transistor 2)
    (Qf, bool, , , , On-off command to the fan)
   )
  (OUTPUTS
    (T1, real, °C, Temperature of transistor 1)
    (T2, real, °C, Temperature of transistor 2)
    (Tp, real, °C, Temperature of metal plate)
   )
 ) // end MultiVarTC
} // end PLANTS section
```

The GENERAL section contains information about the laboratory name and web address. The BLOCKS section contains all data regarding the implemented blocks. For each block, inputs, outputs and parameters are described, along with their names, descriptions and types. Moreover, an input flagged as `req` *must* be connected and has no default, while other inputs may stay unconnected and assume in that case the value given after the `def` keyword (similar consideration can be done regarding a parameter). The PLANTS section describes the processes available for on-line control; the I/O signals are described along with their physical meaning, and the upper and lower bounds are reported for the plant inputs. In addition to inform a user about which blocks can be used for designing an experiment, in the near future this file should be used as input for a graphical interface (GUI) which helps the user in designing an experiment by connecting some graphical blocks. In fact, it is the authors' intention to develop a GUI which allows a user to design an experiment by using the blocks provided by the server. It is worthwhile to note that such a GUI, which will be implemented as a Java applet, could be used by any laboratory which joins the AIRES standard. Moreover, anyone could implement his own interface and share it with other remote labs.

To require an experiment, an EC uploads to the ES an ACER file. Also in this case, for space reasons, we show the structure of that file by means of an example. The example refers to the control scheme of Figure 3, concerning the already quoted multivariable thermal apparatus.

```
{GENERAL
 (EXPERIMENT AnExample)
 (DESCRIPTION cascade control experiment in which
  transistor 1 is used to control the temperature
  of transistor 2, having that or transistor 1 as
  the controlled variable of the inner loop)
```

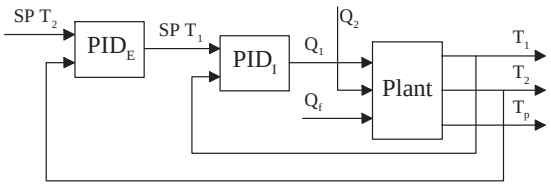

Fig. 3. Cascade control system.

```
 (PLANT MultiVarTC)
 (TIMESTEP 0.5)
 (DURATION 360)
}
{DEFINE
 (Kext=2) (Kint=20)
}
{BLOCKS
 (PIDext ISAPID2dof
  (K, Kext) (Ti, 30) (Td, 0) (N, 1)(b, 1) (c, 0)
 )
 (PIDint ISAPID2dof
  (K, Kint) (Ti, 20) (Td, 0) (N, 1)(b, 1) (c, 0)
 )
 (SetPointT2 STEPGEN
  (Vini, 30) (Vfin, 35) (Tstep, 120)
 )
 (LoadDistQ2 STEPGEN
  (Vini, 0) (Vfin, 25) (Tstep, 240)
 )
} // end BLOCKS section
{CONNECTIONS
 (SetPointT2.Out, PIDext.SP) // Outer PID
 (Plant.T2, PIDext.PV)
 (PIDext.CS, PIDint.SP)      // Inner PID
 (SubI.Out, Plant.Q1)
 (LoadDistQ2.Out, Plant.Q2)  // Load dist.
 (PIDint.HIsat, PIDext.NoUp) // Interloop
 (PIDint.LOsat, PIDext.NoDn) // antiwindup
} // end CONNECTIONS section
{RECORD
 (SetPointT2.Out, Set point for T2)
 (LoadDistQ2.Out, Load disturbance (Q2))
 (Plant.T1, temperature of transistor 1)
 (Plant.T2, temperature of transistor 2)
 (Plant.Tp, temperature of the plate)
 (Plant.Q1, Command to transistor 1)
 (Plant.Q2, Command to transistor 2)
} // end RECORD section
```

The GENERAL section contains some general information such as the name, the duration and the time step of the experiment. The DEFINE section allows a user to define some variables and to use them as values of a block parameter. The BLOCKS section contains all the used block; for each block one has to specify the type of the block and the parameters values. The CONNECTIONS section reports all the signal connections between a source block and one or more destination blocks. In particular it is needed to indicate which I/O ports are involved in the connection. Such ports can be referred to by the port number or the port name. Finally, the RECORD section contains all the signals that have to be stored for downloading along with the name they will be referred to. This data will

be stored in a text file with fields delimited by tabulations. Time values are automatically stored by default.

3. OPERATION OF AIRES

The typical complete cycle of an AIRES experiment can be summarized as follows. First, an EC finds an ES hosting the desired experiment (possible ways to aid such a search are outside the scope of this paper) and downloads its ASC file. Then, the user generates the ACER file for the desired experiment. This can be done manually, i.e., writing the file with any ASCII editor, or by means of specialized software allowing to do the job with a graphical interface. A very interesting side product of AIRES would be the possibility for everyone to write his/her own graphical application to build control schemes: as long as that application adheres to the standard, importing an ASC file could easily populate the necessary palettes with the blocks that can be used to specify a control scheme suitable for the ES providing the ASC file. This would allow anybody to use any remote experiment with his/her own control scheme editor, and the advantages are apparent.

Once the ACER file is prepared, the EC uploads it to the ES (possibly after an authentication process that may be subject to an accounting mechanism, two other useful features that are outside the scope of this work).

When the ES receives the ACER file, it has to compile that file into an executable for its own architecture. This part of the ES, termed the AIRES Compiler (AC), is the only one that has to do with a specific architecture. In fact, the core of the construction of an ES is the implementation of an AC on the architecture at hand. This implementation has to encapsulate the particular architecture from the point of view of the EC, and of course there is no possible standard for the AC implementation if viewed from the ES architecture side, exactly as there is no standard when implementing the software driver for a hardware device, from the hardware side. The present state of the research on AIRES indicates, however, that the AC implementation is possible on very different ES architectures. For example, at the moment two such implementations are being carried out: one uses a Matlab-based architecture, and translates the ACER into a Simulink diagram and some runtime information, the other is based on LabVIEW, and translates the ACER into a sequence of subprocedure calls accessing a shared memory. Although no completely general statement is possible, then, it appears that an AC can be implemented on virtually any architecture an ES may be based upon (note that the two quoted ones, that are being at present addressed, are *completely* different). All the validation tasks to be performed on an ACER file pertain of course to the AC, that has to produce the necessary diagnostics when required. The choice of maintaining the control scheme and the experiment parameters together in the ACER file has been made basically for clarity, as there are no references that may be broken and every experiment is completely defined in one file. This causes a very small overhead for the ES and is perfectly acceptable.

Once the ACER is compiled into an executable, the experiment can be performed immediately or put in a queue. The first choice is highly advisable in the case of an interactive experiment, but apparently requires some reservation mechanism, or similar feature, to manage concurrent requests. The second is the natural choice for batch experiments, and since AIRES is at a preliminary stage, this is where research efforts are now concentrated. At the end of the experiment, data can be made available for download or e-mailed to a specified account. Note that, due to its generality, the use of AIRES does not limit users to perform control systems experiments in the strict sense of the term, but allows for many other kinds of experimental activity as well. For instance, experiments can easily be designed and performed to gather data aimed at off-line system identification.

The advantages yielded by AIRES are numerous. Apart from those already mentioned, the "physical control system encapsulation" realized by the standard allows to perform the same experiment on different architectures, to specify the same control strategy and have it implemented and tested on different systems, and so on. In one word, AIRES is (also) a way to expose the users to all the problems that arise when using control specification tools (e.g., CACSD software) that conceal, at the design stage, most details on the actual control system implementation. Becoming aware of that matter is very important in control education, see e.g. (Amadi-Echendu and Higham, 1997; Bialkowski, 2000; EnTech, 1994), and being exposed to those problems is a particularly useful form of "realism".

4. EXTENSIONS

Since the main scope of AIRES is to create a standard which is able to encapsulate the simulation architecture of any remote lab, some extensions to the ASC and ACER files will become necessary in the future to satisfy some server requests. Some

of these have already been found and solved. For example, by adding a `special` field in the parameter definition of a block will allow the parameter to perform special functions, such as to be tuned on-line during the experiment.

Once developed the needed extensions, an experiment server could use AIRES in various contexts and functionalities. Some examples are reported below.

- Real and virtual laboratories.
- Batch and interactive experiments.
- On-line tunable parameters.
- On-line signal plots.

Since one key feature of AIRES is to standardize the definition of a controller scheme and of the data obtained during the experiment, the use of the AIRES do not exclude the use of other additional features (e.g. features already developed in the remote lab), such as those reported below.

- Visual feedback through a live camera.
- On-line animation of the experiments in VRML.
- Download data in special formats (e.g. Matlab or LabVIEW formats), though preserving the ASCII format anyway.

Of course, since AIRES does not require any additional feature other than a standard browser, it will be a server task to inform users about the required specification in order to run an experiment with extra features (for example, the client must be informed whether it would be useful or not to have a Java Virtual Machine locally installed).

5. EXPERIMENT EXAMPLE

In this section, an example of a control experiment is provided. The experiment concerns the control of the water level in a tank (see Figure 4) which is physically located at the University of Siena. This and other experiments can be accessed through the Automatic Control Telelab (ACT), a remote laboratory which allows users to design and test remote controllers on real plants (Casini *et al.*, 2004). Although the ACT was born to work with the Matlab/Simulink environment (i.e. a controller had to necessary be a Simulink model), it has been recently improved in order to join the AIRES standard, too. The AIRES version of the ACT can be found in `http://act.dii.unisi.it/aires`. For the moment, experiments can be run in batch mode (online camera is provided), but work is in progress also to allow user interaction.

Let us assume to design a feedback control experiment as reported in Figure 5, where the block

Fig. 4. The tank process for water level control.

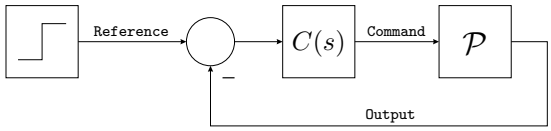

Fig. 5. Scheme of the control experiment reported in the example.

$\mathcal{P}$ denotes the physical plant (i.e. the tank), the reference signal is a step, and the controller to use is

$$C(s) = \frac{50\,s^2 + 55\,s + 5}{s^2 + 10\,s}.$$

Moreover, let us assume that we want to record the reference, the command and the output signals. The experiment stop time is assumed to be 150 seconds while the time step is set to 0.1 seconds.

The ACER file describing this experiment request is as follows (the ASC file is not reported for lack of space).

```
{GENERAL
  (EXPERIMENT LevelExperiment)
  (DESCRIPTION simple transfer-function
    feedback controller)
  (TIMESTEP 0.1)
  (DURATION 150)
} // end GENERAL section
{BLOCKS
  (reference : step
    (vi, 0)
    (vf, 1)
    (tstep, 10)
  )
  (sum : sum
    (signs, "+-")
  )
```

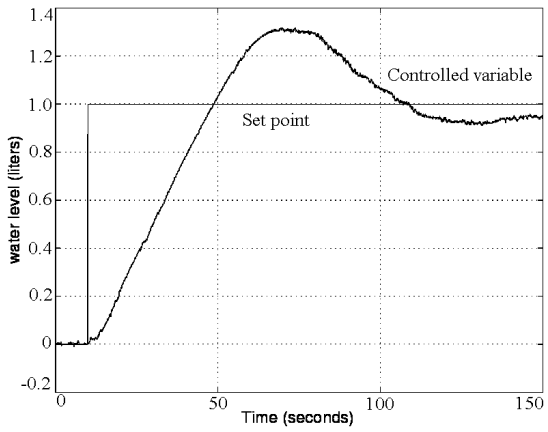

Fig. 6. Reference/Output plot of the water tank experiment.

```
  (controller : tf
    (num, [50 55 5])
    (den, [1 0 10])
  )
} // end BLOCKS section
{CONNECTIONS
  (reference.1, sum.1)
  (sum.1, controller.1)
  (controller.1, system.1)
  (system.1, sum.2)
} // end CONNECTIONS section
{RECORD
  (system.1, Output)
  (controller.1, Command)
  (reference.1, Reference)
} // end RECORD section
```

Once the file is sent to the server, it is compiled and the experiment is started. At the end of the experiment it is possible to download data in the AIRES format, i.e. as a text file whose fields are delimited by tabulations. These data can be used to perform off-line analysis or to plot the signals dynamics as reported in Figure 6 where the reference and output signals obtained in this experiment are shown.

6. CONCLUSIONS

This paper has attempted to draw the reader's attention onto the necessity of agreeing a standard for the remote operation of laboratory experiments. This would allow anybody connected to the net to use any experiment that conforms to the standard, without the necessity of employing specific software. After discussing some general aspects, the AIRES standard has been presented.

The proposal of AIRES is of course preliminary, and not only because the scope of this paper has been limited to batch experiments. For example, with simple (and only morphological) modifications, AIRES could adopt the XML format, as suggested in (Pastor *et al.*, 2003), or be complemented with menu driven interfaces to ease the use on the part of students. This may make its textual files less readable for the human, but would also allow to use a number of standard tools for their manipulation.

At present, research is underway to implement two AIRES servers adopting very different experiment architectures, and to build a "lightweight" and cross-platform editor capable of importing AIRES configuration files and generating AIRES experiment requests. In detail, a Matlab-based AIRES server is already available at the Automatic Control Telelab of the University of Siena, while a LabVIEW-based AIRES server is being set up at the Cremona site of the Politecnico di Milano.

REFERENCES

Amadi-Echendu, J.E. and E.H. Higham (1997). Curriculum development and training in process measurements and control engineering. *Engineering Science and Education Journal* **1997**(June), 104–108.

Bialkowski, W.L. (2000). Control of the pulp and paper making process. In: *Control system applications* (S. Levine, Ed.). pp. 43–66. CRC Press. Boca Raton, FL.

Casini, M., D. Prattichizzo and A. Vicino (2004). The Automatic Control Telelab. A web based technology for distance learning. *IEEE Control Systems Magazine* **24**(3), 36–44.

Dormido Bencomo, S. (2004). Control learning: present and future (plenary lecture). In: *Proc. IFAC World Congress b'02*. Barcelona, Spain. pp. 81–103.

EnTech (1994). Competency in process control—industry guidelines, version 1.0.

Gasperini, D., F. Schiavo, W. Spinelli, C. Veber and A. Leva (2004). A set of hardware and software tools for control education. In: *Proc. 2nd IFAC Workshop on Internet Based Control Education IBCE'04*. Grenoble, France.

Leva, A. (2002). A hands-on experimental laboratory for undergraduate courses in automatic control. *IEEE Transactions on Education* **46**(2), 263–272.

Pastor, R., J. Sánchez and S. Dormido (2003). A XML-based framework for the development of web-based laboratories focused on control systems education. *International Journal of Engineering Education* **19**(3), 445–454.

Poindexter, S. E. and B. S. Heck (1999). Using the web in your courses: what can you do? what should you do?. *IEEE Control Systems Magazine* **19**(1), 83–92.