EDUCATIONAL TOOLS FOR COMPLEX TOPICS: A CASE STUDY FOR NETWORK BASED CONTROL SYSTEMS

O. Mirabella, PhD, M. Brischetto, A. Raucea

Department of Computer Engineering and Telecommunications
University of Catania
Catania , Italy
omirabel@diit.unict.it, mbrischetto@diit.unict.it, araucea@diit.unict.it

Abstract

Engineering students have to learn a lot of complex, interconnected topics in order to develop that multidisciplinary knowledge which is required in facing problems in their professional activities. Usually this is understood especially with the job experience, since during the academic years the various subjects studied are not always seen as being correlated. This bring students to underestimate the importance of certain topics which instead in some context are crucial. For this reason, developing the ability to correlate what they have learned in different disciplines is crucial. This could be achieved with practical activities such as laboratories, but sometime is not the simplest way when the application area is complex and not available in laboratory. In such situation a valuable solution can be the use of a suitable educational software. In this paper we present such a kind of educational tool for the field of Network Based Control Systems (NBCS). Our software wants to help students to deal with NBCS at different level of difficulties, focusing on one or more of its involved problems. Doing this, they are stimulated to link their technical background knowledge, learnt in different subjects, to accomplish a set of given project objectives.

I. INTRODUCTION

As stated by the "Learning by Doing" paradigm, learning is greatly influenced by the experience. In fact, in many fields, knowledge is better acquired if it is generated actively rather than read or heard passively. This paradigm is a very effective learning means, especially for basic mechanical abilities, even if, in fields which require complex knowledge and competences, the experience alone is not sufficient for a full learning. In these cases, in order to accomplish a full learning, it is necessary to reason on the experience to exactly identify what has been learned and to interiorize it, to be able to manage new and different future situations. This is particularly important for engineers, because they are professionals strongly oriented to problem solving. Therefore, it is very important for students in engineering courses, to develop the ability to correlate what they have learnt in different disciplines. In fact, in their future professional activities, in facing problems, the solution will be often found by picking from different multidisciplinary knowledge. Indeed this is not always simple to accomplish especially on account of the scarce cooperation between the teachers of the different courses in coordinating their teaching activities and for the limited time available to the students for each course [1]. This can prevent an unified vision of the various subjects studied and an insufficient ability to link them.

A valid help to this integration can come from the use of suitable educational tools, appropriately designed to allow students to correlate different matters when dealing with specific real problems. A good example of such tools can be represented by simulator programs, in that, they try to reconstruct the application context of certain skills. To this aim simulators have been used in different fields. For example, in [2], the authors examine the effectiveness of simulations in teaching international business, whereas in [3] the authors present an educational software tool which aims to provide a graphical front-end to a Java virtual machine (JVM) with the aim of helping students to learn about JVM architecture and how it works. Instead in [4] the authors present a software tool which emulates a Steam Turbine-based Current Generator with distributed control, where communication between the different devices (sensors and actuators) can be supported by different types of Fieldbuses. All these tools aim supporting and enhancing the learning process and promoting the active and constructive involvement of students.

In this paper we present a multidisciplinary educational tool which wants to be something more than just a simulator. In particular it aims to offer the students a tool by which they can integrate skills acquired in subjects like "Automatic Control" and "Process Control" with those learnt in the "Networks for Process Control" course. All these subjects are

needed in the field of Network Based Control Systems (NBCS), a sector that in the last years has grown in popularity [5]. Both hardware producers and manufacturing company have expressed great interest on the potentiality of this technology. The main challenge of a NBCS is to control a remote process in the desiderate way in spite of the limits introduced by the network. These limits are especially represented by the nondeterministic packets delivery delay which causes a degradation of the control quality. In particular, the delay time impacts on the remote system performances lowering its stability region [6]. Our tool wants to help students to experiment with such a kind of complex and articulated system via simulation, being not simple for them to have one real to work with.

The Network represents a key element of the tool. As we will see in the next sections, it will be modelled not only in terms of delay it introduces in the transmitting packets, but also in terms of reliability, defined as packet loss probability.

Also remote systems and controllers are other key elements in our tool. They are implemented by suitable analytical models. An interesting feature of this software is that students are not limited to use it as it is. In fact, being simple to be customized, students are encouraged to personally add new modules representing remote systems to control, controllers and networks. All this, as we will see, aids the reinforcement of multidisciplinary skills.

The paper is organized as follows: in section 2 a brief description of what is a Network based Control System is given and the most significant points are underlined. In section 3 we present our tool and the learning objectives that we want to reach are described in details. In section 4 the user interface and the main functionalities available are presented whereas in section 5 the software architecture is discussed. In section 6 the way of learning using the tool is described. Finally, in section 7 some conclusions are presented.

II. NETWORK BASED CONTROL SYSTEM: A BRIEF INTRODUCTION

In fig. 1 the classical block diagram of a generic Closed Loop Control System is represented:

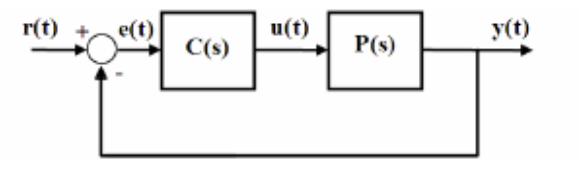

Fig. 1. Architecture of a Closed Loop Control System

Where the typical components are:

– P(s) is the transfer function of the controlled system, in the Laplace domain;

– C(s) is the transfer function of the controller, in the Laplace domain;

– r(t) represents the reference signal, that is, the desired value for the output of the controlled system;

– y(t) represents the real output of the controlled system;

– e(t) represents the error between the desired and the real output of the controlled system: e(t) = r(t) - y(t);

– u(t) is the signal, computer by the controller depending on the value of e(t), to drive the controlled system in order to cancel the difference between y(t) and r(t).

As it is well known, the control loop has to maintain the output signal y(t) of the controlled system as close as possible to the reference signal r(t).

The previous diagram refers to an analog process controller. To-day, most of the control systems provide a digital control, that is, the controller is a digital system such as a microcontroller, a PLC or even a standard desktop computer. The remaining part of the system can either be digital or analog. For example, fig. 2 refers to the case in which the controller is digital whereas the system is analog:

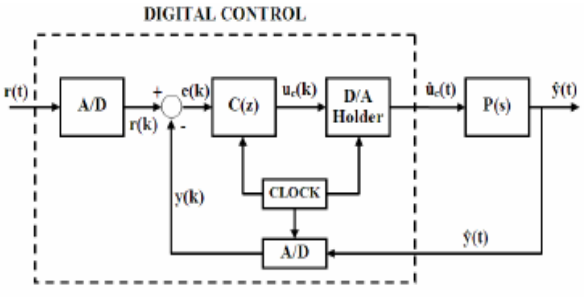

Fig. 2. Architecture of a Digital Closed Loop Control System

Since a digital controller is a discrete system, the Laplace transform of the controller has been replaced with the Z-transform. The interface between the digital part and the analog one, is made up of suitable converters. In particular, at the interface between the output of the digital controller and the input of the analog system a digital-to-analog converter (DAC) system must be used. This system, that we will model as a zero-order hold (ZOH) converter, must convert a discrete-time signal to a continuous-time signal by holding each sample value for one sample interval. The opposite conversion must be done at the output of the controlled system. In fact, this output must be converted, through an Analog to Digital Converter(ADC), to a discrete value, in order to be comparable with the digital reference signal. If we consider the controlled system P(s), together with the ADC and the DAC, we obtain an equivalent discrete block, Pzoh(z), which takes into account the dynamics of both the two converters and the controlled system. The Pzoh(z) must have, at the sampling intervals, the same values of the continuous system. With this transformation, the diagram of fig. 2 can be represented as shown in fig.3, where only discrete models are considered:

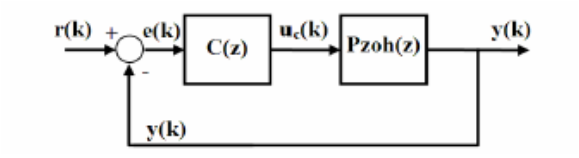

Fig. 3. Block Diagram of a Digital Closed Loop System

In a Network Based Control System, the controller and the remote system are connected through a network as represented in Fig4a, whose effects are not negligible. The block diagram of Fig. 3 in this case becomes as shown in Fig.4b:

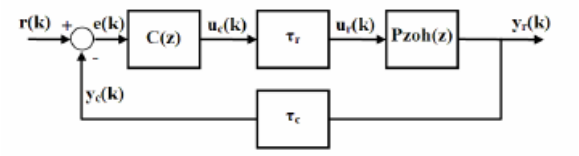

Fig. 4a. Block Diagram with delays in both control and feedback line

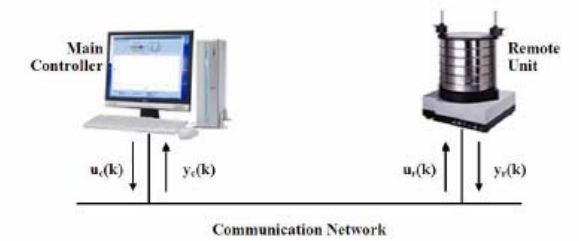

Fig. 4b. Representation of a Network Based Control System

In this case, called $z^{-\tau}$ the time delay operator, and QoS(t) the current quality of service provided by the network at the time t, with reference to the fig. 4b, will be:

$$Ur(k) = Uc\left(z^{-\tau_r}, QoS(t)\right)$$
$$Yc(k) = Yr\left(z^{-\tau_c}, QoS(t)\right)$$

where Q is the transmission delay from the controller to the remote system, and Q is the delay in transmitting a signal from the remote system to the controller. These delays and the QoS(t) depend on different aspects of the network, such as its bandwidth, network congestion, the network protocol used etc.

Therefore, as we can see from Fig.4a, the controller will send over the network a control signal Uc(k) which arrives to the remote system with a certain delay. We refer to the delayed signal as Ur(k). The remote system reacts to the control signal Ur(k), by performing the required operations which produce a new state and new output values. Information about the new system conditions, collected by suitable sensors, must then be sent back to the controller in order to close the control loop. As before, the collected data will be sent as Yr(k), but due to the presence of the network, they will be delayed as Yc(k). All these delays can decrease the quality of control and the controlled system could become unstable [7].

III. LEARNING OBJECTIVES

The tool presented in this paper has been designed to allow students to make experiments over a NBCS of the kind represented in fig. 4. In particular, it aims to show what are the effects of different types of network over a NBCS. In the block diagram in fig. 4 the key elements are the system to be controlled, the controller and the interconnecting network. Students can deal with all or some of these elements. For example new network models can be implemented and tested, to verify their effectiveness in this kind of applications, as well as specific techniques for distributed control.

Of course students should already have some theoretical knowledge in their background. In particular is supposed that they have learnt the theory regarding the analysis and synthesis of closed loop systems, and are able to define the continuous mathematical model of a controller, for example in the form of a PID controller. Usually these knowledge are acquired in different courses, but has been seen in our exerience that, because they are not used in subsequent subjects, often are soon forgotten. In addition these courses are mainly treated from a theoretical point of view, not giving enough emphasis on the practical applications and implementations.

With our tool we want to fill the gap among theory and practice in the specific field of distributed control systems, where control theory apply, but performances are strongly influenced by many factors not taken into account in the theoretic courses. In addition, since the controller must be digital, students should be able to pass from an analog controller model to a digital one, and must code it as a working software program. Therefore, our tool can be used in different ways allowing to obtain various learning targets.

Students find, already implemented, a suite of canonical transfer functions of different order for which suitable controllers have been computed and implemented in order to impose to the resulting closed loop system, some specifications, such as the rise time (the time needed by the controlled system to reach the desired value), the peak overshoot (the highest value reached by the system response before reaching the desired value) and the steady-state error (the difference between the input reference and the output of the controlled remote system when the response has reached the steady state). Students can choose a transfer function among those present in the tool and then can select the characteristics, in term of delay and loss probability, of the network which interconnects controller and remote system. In this way they are able to see how this network influences the system response. Another interesting possibility, as said earlier, is that students haven't to limit their analysis to the systems already present, but they can define new systems, new controllers and new networks.

It is assumed that students who want to use our tool have some prerequisite. In particular they should know:

– the mathematical models of the most common systems;

– the most typical input functions;

– how to synthesize a continuous controller given some specification for the system to control;

– how to pass from analog to discrete domain.

They learn:

– how to code in a programming language the model of a dynamic system, for example with a finite difference model;

– how to implement the discrete controller algorithm in a programming language;

– how to design and implement a network model;

– how to analyze the simulation results, and if it is required, how to modify the controller to take into account the network characteristics.

It is clear that working with this tool involves for students the stimulation of a set of transversal cognitive processes. In fact, they have to use knowledge and skills learned in different subjects. In particular:

1. System Modelling and Automatic Control:
   a. to know, or to be able to obtain, the mathematical model of a system;
   b. to be able to synthesize a suitable controller to drive the dynamics of a system;
   c. to be able to pass from analog to discrete models.
2. Computer Networks:
   a. to know the reference models of the main network architectures;
   b. to be able to define new network models;
   c. to be able to compare and valuate the characteristics of different network architectures.
3. Software Engineering and Programming:
   a. to be able to implement the discrete models of systems, controllers and networks, by mean of suitable algorithms.

To reach all these targets, students must be allowed to integrate in the tool, in a simple way, their systems, controllers and networks models. The modular architecture of our software has been thought with this in mind.

IV. THE NBCS SIMULATOR USER INTERFACE

The user front end is made up of the main window represented below:

Using the "Function" menu, the student can select one of the available systems, characterized by a certain transfer function in the Laplace Domain. Three functions are predefined of first, second and third order respectively plus others representing specific systems, such as a direct current servo motor and a linear slider actuator models.

A predefined controller is associated to each function. Each has been computed to impose a set of specifications to the system dynamics, in response to the selected input. The input function is chosen through the "Input" menu.

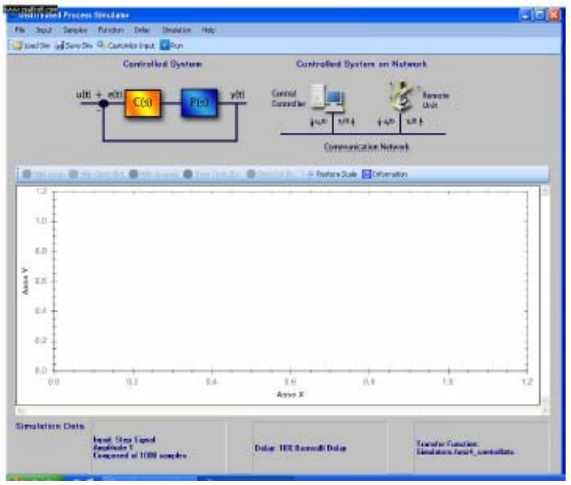

Fig. 5.Main Window of the NBCS Simulator

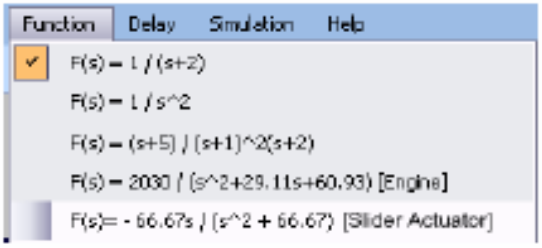

Fig. 6. Some Transfer Functions available to the user

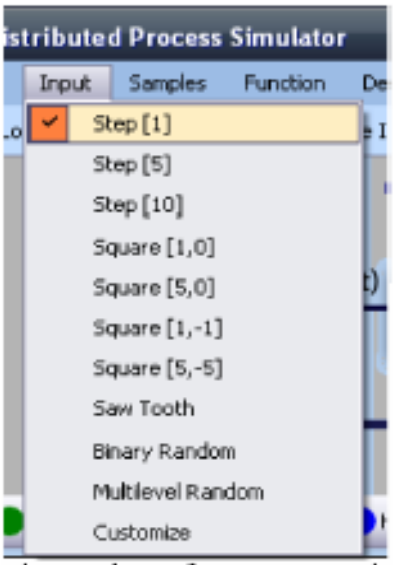

Fig. 7. The Reference Input Signals menu

It is also possible to customize the temporal behaviour of the input function, choosing "Customize" and using the "User Defined Input" Dialog which allow to set the amplitude, period, duty cycle, maximum and minimum values of the input reference signal.

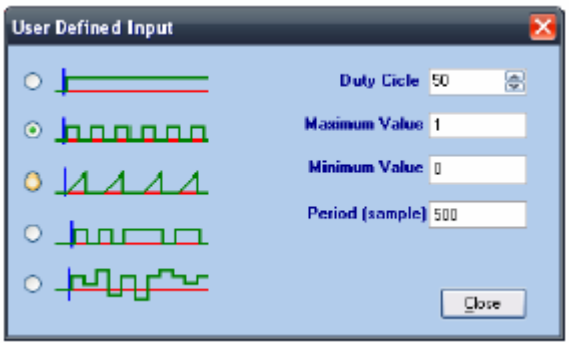

Fig. 8. User defined Reference Input Signals

The number of samples can be set using the "Samples" menu.

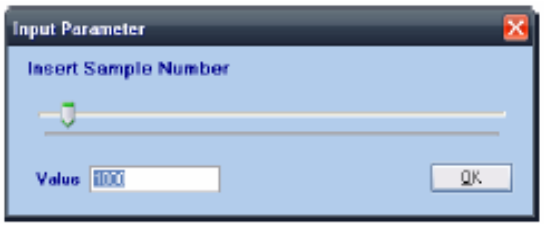

Fig. 9. Customizing the number of samples for the simulation

Finally, the network delay/loss can be defined with the menu Delay:

Fig. 10. Delay models

Three network models are implemented by default:

– Bernoulli Delay: this item allows to select a Bernoulli delay model, that is, a packet sent over the network could be delayed at most of one temporal slot. It is possible to configure the percentage of delayed packets from 0% (no delay) to 100% (delay always);

– Bernoulli Loss: this item allows to select a Bernoulli loss model. If a packet is lost, the destination module will continue to use the previous received value. It is possible to configure the percentage of lost packets from 0% to 100% losses;

– Exponential Delay: this item allows to select an Exponential delay model, that is, a packet sent over the network could be delayed of a random temporal quantity following an exponential distribution. It is possible to configure the mean delay value (in milliseconds).

At the centre of the main window there is a plotting area where the curves relative to the applied reference input function u(t), and to the system output function y(t) are plotted both with and without(ideal case) delay/loss. Below the plotting area, information on the actual running simulation are reported as well. In addition, when the simulation ends, is possible to display the characteristic values of the system response, in particular, the rise time, the peak overshoot, the steady-state error and the settling time, to quantitatively compare the ideal case and the case with delay/loss. An example of this feature is reported below:

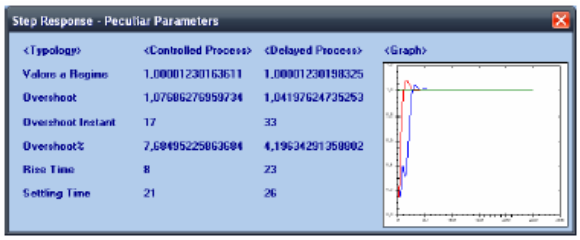

Fig. 11.Step response characteristic values

It is also possible to visualize the system dynamics through another, more realistic, graphical representation. As an example, fig. 12 shows the simulated dynamics of a linear slider actuator and fig. 13 shows the dynamics of a DC servomotor. The representation of the system appears in the space above the plotting area. The user obtains the system output curves, and then, these output values are used to animate the graphical system model in such a way it follows the computed dynamic.

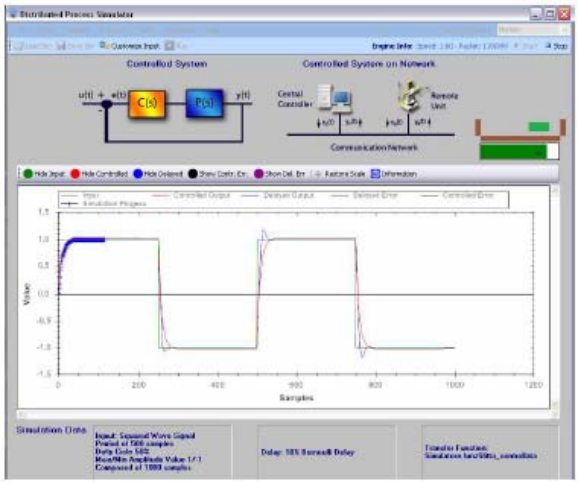

Fig. 12.Linear slider actuator dynamics

In this way, students are able to follow the system dynamics behaviour, at the same time, both on the graphical system model and on the plotting area, with a new cross point curve being plotted over the previous one. This should give a better practical representation of what the plotted values mean for the real system.

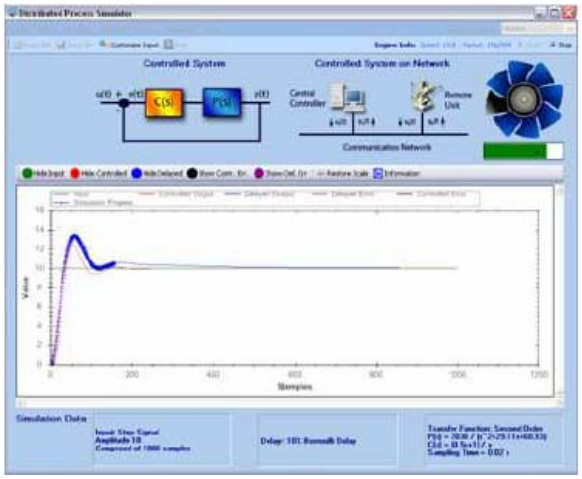

Fig. 13. DC servo-motor dynamics

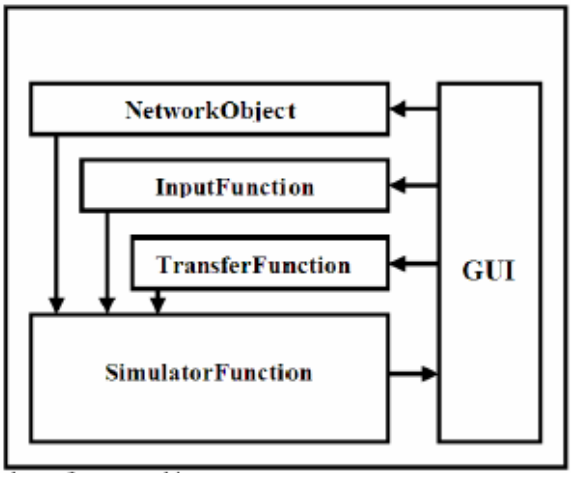

Fig. 14.The Software Architecture

V. THE SOFTWARE ARCHITECTURE

The software has been developed in C# express edition using the object oriented programming paradigm. The architecture has been thought in such a way to allow a simple implementation and integration of new modules which model networks, systems, controllers and input reference functions. This way students don't have to deal with implementation problems that are not strictly related with the learning objectives. In fact, students who want to add new modules have to work only on specific classes. The main classes available for this purpose are:

- NetworkObject: This class is the superclass of all classes representing a network. It allows to define a network model, in particular the distribution functions for packet delay and loss;
- InputFunction: This class is the superclass of all classes representing an input reference function. Its attributes allow to define the type, amplitude, duty cycle, maximum and minimum values. By default the following function types are available: SquaredWave, SawTooth, Step, BinaryRandom, MultilevelRandom;
- TransferFunction: It represents a generic transfer function. New transfer functions, both for controllers and for remote systems to control will be implemented extending this class;
- SimulatorFunction: It represents the main class of the tool. It uses instances of the other classes to model a closed loop system and to compute its dynamic. In particular, depending on the choices made up by the user via the application GUI, it creates at first instances of the classes TransferFunction, InputFunction and NetworkObject according with the user selected menu items, and then computes the samples of the system output variable. These samples are then used to plot the curves in the plotting area. It also computes the characteristic values of the system response in term of rise time, peak overshoot, steady-state error and settling time.

VI. LEARNING WITH THE NBCS SIMULATOR

Students can use the software as it is, without modifications, using the modules already implemented. In this case the learning objectives is reached making a set of simulations choosing one of the systems available and looking at the effects of different network settings on the dynamic of the resulting closed loop system.

The other way to use the tool requires a more articulated work. The starting point is the analytical model of a system, given, for example, in the form of differential equations and a set of specifications regarding the rise time, the peak overshoot, the settling time and the steady-state error. At first, students have to use the methods learned during the Automatic Control course to synthesize an analog controller able to impose, to the given system, the desired behaviour. Then students have to pass to the discrete domain. To do this, they can use suitable tools like Matlab, or can do the required calculations "by hand". In this phase the objective is to obtain the discrete time models of both the remote system and the controller under the form of finite difference equations. Then the resulting models have to be implemented as software modules using the proper classes. This way, students learn both to implement, in a computer, the model of a dynamic system and both to synthesize and implement a suitable digital controller. Also the network which connects the controller and the remote system is object for student experiments. In fact, students can add new distribution functions in order to characterize the delay and the loss probability, or the network can be modelled even in a more complicated way than a software function. For example a physical testbed, like that described in [8], could be interfaced with the software. All this require a previous analysis of the available network typologies, and the ability to device new network models conceived ad-hoc for the particular case study. In addition to the designing and implementation phases, the critical analysis of the simulation results with different network

configurations is also very important. Students should understand the behaviour of the resulting curves, and eventually they should be able to modify the controller or network modules to improve the quality of the distributed closed loop system.

Students who have used the software have expressed great satisfaction. Not big problems have been encountered in the start up to get confidence with the tool being the interface quite intuitive and essential. The main problems arise when a new system to control has to be implemented. This mainly because the theoretical knowledge required to design and implement the system and controller models are usually, more or less, been forgotten and students require a certain time to revise them. This requires usually a week of home work using text books and Internet tutorials. Then the revised knowledge is used to find the analytical model of the assigned system and to compute a suitable controller. This requires some dais and often students use Matlab to quickly validate the resulting closed loop system (considered without network delay effects). If all works as expected, then is possible to begin the implementation of the available classes. To do this, a Z transformation of the analytical model is computed and then the finite difference model of the closed loop system is obtained. This is finally implemented using the *TransferFunction* class. During these steps different skills are used, such as mathematic, process modelling and programming. This phase need usually another week of home work, but could be more in case the student wants also to implement a new network model or modify one already present. Finally the NBCS model obtained can be investigated to see the effects of the chosen design parameters, such as the sampling interval and the PID constants. A set of simulation are run, and if the results are not as expected, the parameters might need to be refined.

Students are asked to produce a report on the simulations done. This is then discussed with the professor during an oral exam to see what the students have learnt. Comparing students who have used the tool with others in the previous years, there has been a sensible improvement on these skills as proved by a major readiness of mind in answering to specific questions.

VII. CONCLUSIONS

In this paper we have presented an educational tool aiming to allow students to "learn by doing" using multidisciplinary skills. In particular, students have to control a remote system via a Network Based Control System, being able to satisfy a set of given constraints. The learning will be more stimulating and effective because applied to a specific and clear problem and because the theoretical issues can be directly applied in practice. The use of simulators like the one presented in this paper, where students are not only users of functionality implemented by other people, allow us to extend the range of learning objectives. The knowledge acquired in different disciplines can be put together and used in a precise application context.

REFERENCES

[1] Portero A.; Saiz J.; Aragones R.; Rullan M.; Valderrama E.; Aguilo J.; "Adopting New Competences Experience in the Face of Engineering Learning" *1ST IEEE International Conference on E-Leaming in Industrial Electronics,* 2006 18-20 Dec. 2006 Page(s):34 - 39 Digital Object Identifier 10.1109flCELIE.2006.347208

[2] Carlyle Farrell "Perceived Effectiveness of Simulations in International Business Pedagogy An Exploratory Analysis" *Journal of Teaching in International Business* vol. 16 Issue 3 ISSN: 0897-5930

[3] Abenza, P.P.G. Olivo, A.G. Latorre, B.L. Miguel Hernandez "VisualJVM: A Visual Tool for Teaching Java Technology", *IEEE Transactions on Education,* vol. 51, Issue 1, pp 86-92, Feb. 2008 ISSN: 0018-9359

[4] O.Mirabella and A.Raucea, "An Interactive tool for the study of the Fieldbus impact on a Steam Turbine-based Current Generator", *in Proc. of ICELIE 06, 1ST IEEE International Conference on E-Leaming in Industrial Electronics,* Hammamet, Tunisia, pp. 119-124, 18-20 Dec. 2006

[5] J. Baillieul and PJ. Antsaklis, "Control and Communication Challenges in Networked Real-Time Systems" *Proceedings of the IEEE Volume 95,* vol. 95, issue 1, pp. 9-28, Jan. 2007 Digital Object Identifier 10.1109/JPROC.2006.887290

[6] Wei Zhang, M.S. Branicky and S.M. Phillip, "Stability of networked control systems", *Control Systems Magazine,* IEEE vol. 21, Issue 1, pp 84-99, Feb. 2001, Digital Object Identifier 10.1109/37.898794

[7] Tipsuwan, Y. Mo-Yuen Chow "Network-based controller adaptation based on QoS negotiation and deterioration" *The 27th Annual Conference of the IEEE Industrial Electronics Society, IECON '01,* Denver, CO, USA, vol.3, pp 1794-1799, Year 2001, ISBN: 0-7803-7108-9

[8] L.Lo Bello, O.Mirabella and A.Raucea, "Design and Implementation of an Educational Testbed for Experiencing with Industrial Communication Networks", *IEEE Transactions on Industrial Electronics,* vol. 54, issue 6, pp. 3122-3133, ISSN: 0278-0046, Dec. 2007.