

TECHNIQUES FOR EMBEDDING CIRCUIT SIMULATIONS INTO ELECTRONICS DESIGN COURSEWARE

Saravanan Solaimalai¹ and Peter Hicks²

Abstract *✎*The integration of Web-based simulations into on-line tutorial courseware can be used to create a very powerful learning environment. Preliminary work has focused on use of the Common Gateway Interface (CGI) mechanism to establish communication between an electronic circuit simulator, namely PSPICE, running on a remote server and a Web-packaged Authorware application called EDEC running on a client machine. A graphical front-end allows users to alter simulation models without any knowledge of the simulation language. Simulation results are processed using the Authorware scripting language before being displayed both in numerical and graphical formats. Results are presented which illustrate the effectiveness of this technique as an aid to learning the principles of electronic circuit design. The current system is not entirely portable and limited to a single computer system. Introducing distributed computing methods with CORBA means that the system could be extended to benefit from the distributed and parallel nature of the World Wide Web.

Index Terms *✎* Courseware, electronic circuit design, simulation, on-line Web-based tutorial.

INTRODUCTION

The use of the computer as an educational tool has become firmly established with the arrival of multimedia and the World-Wide Web. Many examples exist which demonstrate the power of interactive computer-based learning material (courseware) as an aid to teaching and learning.

Simulations for Education

The integration of computer simulations into courseware has long been recognised as a powerful tool for learning, giving learners an environment in which they can explore subtlety and variation. More specifically, Thomas and Schnurr [1] define simulations used in the context of education as educational simulation, i.e. "one that is used to teach about the system modelled by simulation rather than the process of modelling itself". Computer simulation is quite popular in instruction [2], and its benefits are numerous [3] - [6]. These include its suitability for contemporary approaches to active learning whereby the learner is invited to explore and discover the important concepts in the domain to be mastered instead of being given direct instructions about it.

Students are also free to accomplish this using their own preferred approach and at their own pace, unlike in the traditional classroom method where students are expected to absorb most if not all of what is being taught.

Simulation tools assist students in developing an in-depth understanding of complex mathematical and physical concepts, which otherwise would just be seen as equations in a textbook. Simulation-based learning methods allow students to experiment with a model and therefore acquire the ability to generate and evaluate hypotheses. Knowledge that is self-discovered also stays more firmly than knowledge that is told. Thus simulations indirectly generate interest in a subject area. Simulations are very useful for modelling real world situations that would otherwise be impossible, too dangerous, too expensive or too time consuming to perform. In engineering education, for example, computer based simulations are essential tools. Working with simulators more closely emulates the actual environment of today's engineers and helps prepare engineering students to face the demands of the industry.

Simulation in this discussion is restricted to interactive simulation, i.e. programs that run a model with parameters supplied by the student. The student's input to the model determines the subsequent behaviour of the model, which after completion of the simulation is displayed either as numerical values, diagrams, pictures, animation or description of its new state. This key feature also sets simulation apart from an animation or pre-programmed hypermedia.

Using simulations in education, however, has its drawbacks. In general, they are costly to buy or produce, with commercial simulations normally being targeted at designers or researchers and not teachers or students [1]. There is also a steep learning curve associated with simulations, and students using these tools often require close supervision and support [6]. Simulation-based course authors should take into consideration the need for a guiding and focusing tutor (or other experts) when attempting to use simulation in an education environment.

Web-based Simulations for On-line Learning

World-Wide Web based simulations are not uncommon in the education community, although they vary a great deal from one another due to the myriad of Internet related technology available for implementation.

¹ Saravanan Solaimalai, Dept. of Electrical Eng. and Electronics, UMIST, PO Box 88, Manchester, M60 1QD, UK. Saravanan.Solaimalai@stud.umist.ac.uk

² Peter Hicks, Department of Electrical Engineering and Electronics, UMIST, PO Box 88, Manchester, M60 1QD, UK. P.J.Hicks@umist.ac.uk

Web-based educational simulation can overcome some of the limitations of traditional simulation software discussed in the previous section. Due to the nature of its implementation, a single instance of the simulation tool (on a Web server) is sufficient for such a system, thus saving the cost and portability issue involved in installing the simulation tool in multiple machines. Other benefits of a Web-based educational simulation include wide accessibility, controlled access, efficient maintenance and increased integration [7].

Some selected examples of simulation-based learning environments that are presently in use are the SPICE Internet Package (SIP) [8], an integrated on-line learning environment for antenna education using Mathematica [9], and the MultiVerse environment [1].

Educational Simulation for Electronic Engineering

Over the last ten years a consortium of universities in the UK has been developing computer-based learning material for use in electronic engineering education [10]. Known as the Electronics Design Education Consortium, or EDEC for short, the group has produced a wide range of modules spanning analogue and digital circuit design, high level system design and testing and design for testability. The project is funded under the UK national Teaching and Learning Technology Programme (TLTP) [11]. In common with much of the courseware produced under the TLTP initiative EDEC was developed using a commercially available authoring tool, in this case Macromedia's Authorware Professional. Using the latter's Shockwave technology the EDEC modules can be packaged for delivery over the Web.

A primary objective for EDEC was to enhance the educational value of Electronic Computer Aided Design (ECAD) tools, with special emphasis on the use of Berkeley's SPICE (Simulation Program with Integrated Circuit Emphasis) software for circuit design. SPICE, which has become the de facto industrial standard for computer aided design for electronic circuits over the years, is the most commonly used analogue circuit simulator today. It is widely used in electronic engineering related courses in most universities. As such the emphasis on SPICE as a simulation tool is appropriate.

The concept of Educational Simulation for Electronic Engineering (ESEE) is to extend the Web-based EDEC courseware material by embedding access to electronic simulation tools within it.

WEB-BASED SIMULATION APPROACHES

Approaches for Web-based simulation delivery have been grouped into three categories by Lorenz *et al* [12]. These approaches are focussed around the different methods of sending simulation requests (simulation parameters) and displaying their responses (simulation results): *Remote*

Simulation, *Local Simulation*, and *Remote & Local Simulation*.

Remote Simulation

Remote simulation, as the name suggests, is a scenario whereby the simulation model is executed on the server machine and results are sent back to the browser. In remote simulation (Figure 1) the user typically specifies values of parameters for a simulation model on the Web browser.

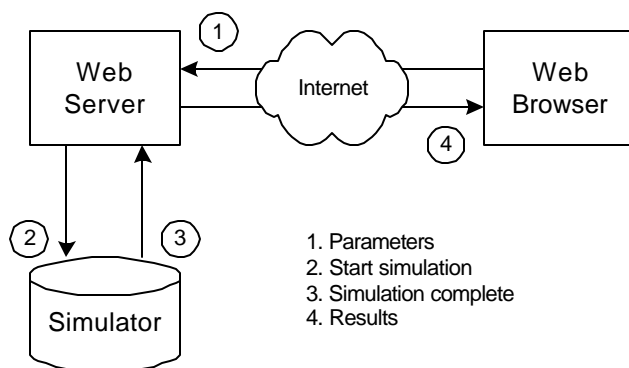


FIGURE 1.
REMOTE SIMULATION MODEL.

This is normally done on an HTML form and submitted to the server by pressing a submit button. The Common Gateway Interface of the Web server is used to transfer data to and from a simulation program. Simulation results are normally returned as a new HTML document. This may be in any of the standard MIME types, for example html text or images (gif, jpeg). Note that the role of the CGI can also be replaced by less common alternatives such as Active Server Pages, or Allaire's Cold Fusion.

This approach is most common and is suitable for incorporating existing off-the-shelf software. Establishing a link between these software and the Web server would normally only be a matter of writing a simple CGI script. However, remote simulations are not well qualified for observing dynamic processes at work and since the simulation software, after being invoked, runs independent of the Web server, it does not permit interruption of a running simulation by the user.

Local Simulation

Local simulation (Figure 2) involves execution of the simulation almost entirely on the client, or Web browser. Such a simulation application is typically built using Java Applets although the VRML (Virtual Reality Modelling Language) used to be a more dominant tool in this approach. Java simulation applets that are loaded onto the Web browser upon user request are independent programs, that is they have all the necessary information and logic to execute without an established connection with the Web server.

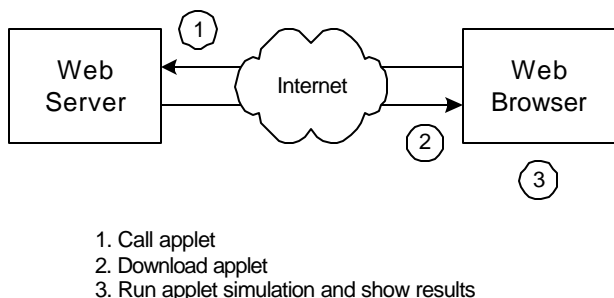


FIGURE 2.
LOCAL SIMULATION MODEL.

These simulations normally incorporate a user interface and therefore allow observation of dynamic processes. Users also have direct control over the progress of the simulation. However, such simulations have to be coded as applets from the start, and are therefore more resource intensive.

Remote & Local simulation

This approach is shown in Figure 3. Simulations run remotely on the Web server (that supports Java Servlets). The result is then returned to the Web browser and an applet is used to visualise it on the client’s machine. Alternatively, the applet can be used to extend the visualisation of the result by providing additional local simulations. A virtual connection is normally established between the servlet and the applet allowing continuous data flow between the two. Typically the user interface is also written as a Java applet.

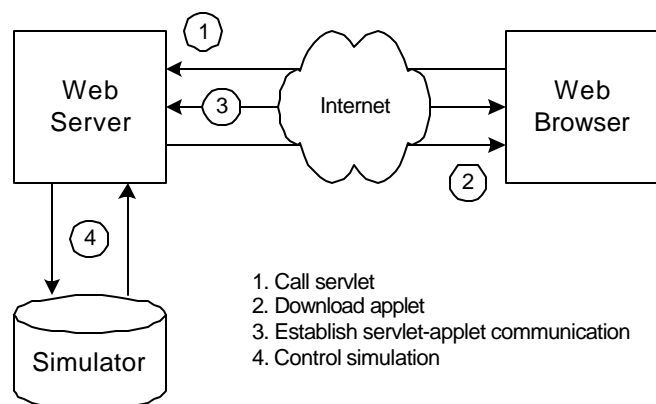


FIGURE 3.
REMOTE AND LOCAL SIMULATION MODEL.

The biggest advantage of this approach is the portability of the code involved as the application is written almost entirely in Java. It also enables dynamic visualisation of the simulation progress and direct handling of the running simulations.

Of the three approaches outlined above, Remote Simulation with CGI technology was initially chosen for reasons of popularity, wide spread availability and relative

ease of programming when compared to two other approaches. Having chosen CGI as the implementation method, Perl was the obvious choice for the programming language:

- Perl is the most popular and appropriate language for writing CGI programs.
- Being an interpreted language, Perl scripts are more portable than code written in compiled languages.
- Perl makes code modification easier for developers than using C/C++ or Java.
- Perl is best for short and simple tasks, and therefore suitable for this project.

The Microsoft Internet Information Server (IIS) was chosen as the Web server and the evaluation version of MicroSim’s PSPICE was selected as the SPICE simulation engine.

SYSTEM ARCHITECTURE

The architecture of ESEE is essentially that of the Internet client-server architecture and can best be understood with reference to Figure 4. Communication between the two entities is made possible by the TCP/IP protocol suite. Note that the client’s Operating System can either be a Windows or a Macintosh operating system as the Authorware Web Player is only supported on these two platforms. The Authorware Web Player runs as an add-on (installed as a plug-in) to the Web browser and is where the Web-packaged courseware would execute.

The Internet Information Server plays a central role on the Server side. All Web, or HTTP transactions have to go through the Web Server. The Courseware Files storage is part of the computer’s permanent storage. Perl programs, stored within the Perl Scripts storage area, execute in a separate region of the server’s memory space, known as the CGI process space. This is a feature of a CGI program that ensures that its operation is independent of the server’s operation. The PSPICE program is invoked by the CGI program and instructed to simulate a circuit file from the SPICE Files storage. The output data is then sent to the simulation interface running on the Authorware Web Player.

The diagram in Figure 4 depicts the simplified sequence of events (shown with numbers) that occur when a simulation is run on the system. An explanation of the numbered events is given below:

1. The Authorware Web Player (where the simulation interface executes) requests for a Perl script (CGI program), passing to it the parameters for the circuit simulation.
2. The Web Server spawns a new CGI environment and causes the CGI program to run.
3. The CGI program reads the template circuit file, and ...

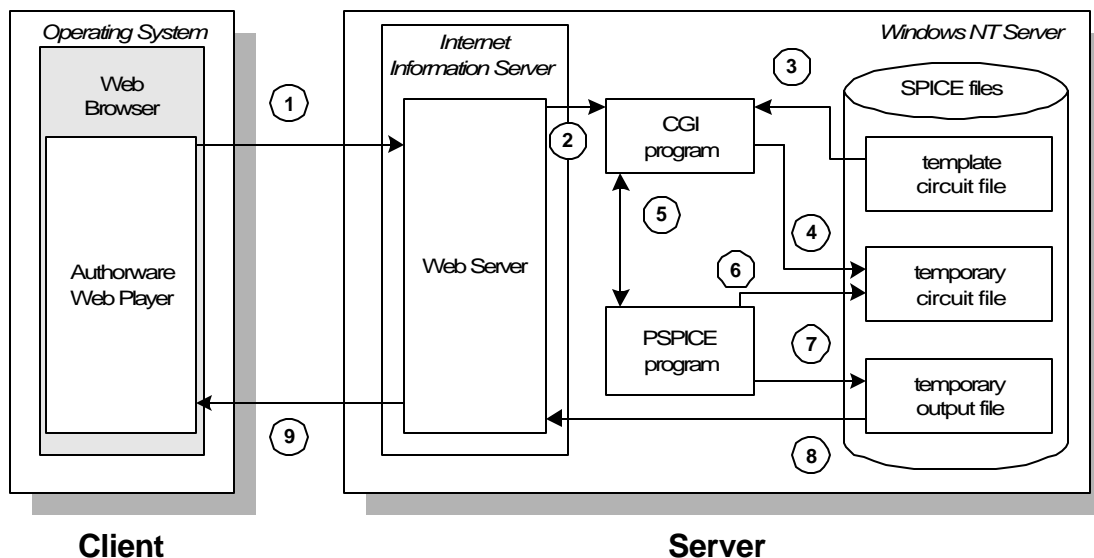


FIGURE 4.
SIMPLIFIED FLOW DIAGRAM FOR THE WEB-BASED SIMULATION ENVIRONMENT.

4. ... creates a temporary file describing the circuit with the parameters provided by the user.
5. The CGI program then invokes the PSPICE program, and
6. runs a simulation of the circuit file created in (4).
7. PSPICE writes the results of the simulation into an output file.
8. The client requests for data from the output file
9. which is then processed in the Authorware Web Player

User Model

Adapting or redesigning the simulation interface to promote learning is a distinctive step that sets the interface design for this project apart from other software interface design processes. Tait [14] describes a simulation-based learning environment as consisting of four concentric layers, like an onion.

In the context of this project the **Model** at the centre of the simulation-based learning environment represents the description of the circuit being simulated, written in the SPICE 'language'. This quantitative description is made executable by using PSPICE, the analogue simulation tool. Given the input file, PSPICE produces the output for the variables of interest, or conditions being monitored. The iteration of this process of generating successive outputs given the variation in the inputs would eventually help describe the behaviour of the model to the learner.

Working out from the centre, the **Simulation Interface** exists because the raw data that flow to and from the **Model** need to be translated into a format more comprehensible to the learner. Specifying input parameters for a circuit, for instance, has to be done in the SPICE language so that

PSPICE can make sense of it. The simulation interface eliminates the need for the user to know this language by providing a mechanism which translates changes made to selected interface objects into values in SPICE language. Similarly, output of a simulation would be displayed in a more readily understood form such as graphs and animations. Such is the importance of the simulation interface.

The purpose of the next layer, the **Learner Interface**, on the other hand is to enhance the simulation interface by making the interaction correspond directly to the concepts or procedures that are to be learned. It ultimately helps the learner achieve the goals of a simulation exercise. In the context of this project, an example would be an interface designed to help a student observe the implications of increasing the collector current of a BJT (bipolar junction transistor) circuit on the transfer characteristics of the transistor.

The role of the outer **Learner Support** layer is especially important in any simulation-based learning environment as simulations can be difficult to learn. Its function is simply to help the student learn from using the simulation. The learner support is typically provided through teachers or trainers. However, it may well be replaced by providing it as part of a computer-based learning environment, which has to be intimately linked with the learner interface.

The learning process of a student using this system follows the 'preparation/observation/reflection' structure described by Tait [14]. *Preparation* involves the user setting up the simulation to behave according to a certain hypothesis. *Observation* simply involves the learner noticing the intermediate changes in the simulated model before it

comes to an end. While *reflection* refers to the learner confirming the hypothesis or correcting it based on the results obtained from the simulation.

A typical scenario of a student learning session would therefore start with the student being prepared for a simulation exercise, by a tutor, with the notes and instructions on how to use the simulator. The student then goes off to a computer that is equipped with applications to access the Web-based simulation environment. He/she then uses the Web browser to go to the Web site where the virtual learning environment is set up. There would be an authentication dialogue box, which the user would have to use in order to log into the system. The courseware is accessed as an Authorware Web-packaged piece running on the Web browser. Having been briefed with theories and concepts governing the simulation by either the tutor or by the courseware itself, the student proceeds to the learner interface of the simulation. The student should then be given instruction on how to use the simulator and asked to perform a certain simulation. A set of controls on the learner interface provides the student with the ability to vary the input parameters for the simulator. On clicking a start button, the simulation begins. On completion, the simulator returns the simulation results to the Web browser to be displayed in a manner that contributes to the learning process of the student. The student can then either continue to perform another simulation with different sets of input conditions, or choose to quit the simulation.

IMPLEMENTATION

A number of exemplars have been created to demonstrate the potential of embedding educational simulation into the EDEC courseware. These use simple linear dc circuits and bipolar transistor circuits to illustrate some of their fundamental characteristics. Circuit diagrams are typically captured from a schematics capture application as a Windows Meta File (.WMF) and imported into Authorware.

In order to extend the interactivity of the simulation environment beyond Authorware's standard interactive components, ActiveX controls were used. The exemplars use three main ActiveX Controls: the *Microsoft Forms 2.0 ComboBox*, *TextBox*, and *Scrollbar* control. Where necessary custom ActiveX controls can be created for use within a piece.

After assembly and testing as a standalone application, the exemplars were Web-packaged according to standard Authorware procedures. Authorware provides three built-in functions, *ReadExtFile*, *ReadURL* and *PostURL*, to invoke scripts from an Authorware Web-packaged piece. The *ReadExtFile* function is used to call a Perl script that executes the PSPICE program. Values of all circuit components, including those which were not modified by the user, are sent to the Perl script for processing. At the receiving end, the Perl script decodes the URL-encoded string and the decoded data is written to a temporary circuit

file. The file name for the temporary circuit is generated by the client and is sent to the Perl program along with other information appended to the URL. The Perl program then creates a circuit file by reading the circuit description (netlist) from a pre-prepared template circuit file and writing it to the temporary file, replacing the element values with those obtained from the Authorware Web-packaged piece. In other words, the temporary file is identical to the template circuit file with the exception of the element values, which have been input by the user.

Invoking SPICE

SPICE is invoked to simulate the circuit described by the temporary circuit file using a CGI program. The output file that SPICE creates has the same file name as that of the input circuit file with the only difference being the extension. Output files have an extension of '.out' instead of '.cir', and, like the input file, are in text format. Although slight variations exist between the SPICE simulation output files of different manufacturers' ECAD tools, there is a general pattern that is adhered to, which makes manual interpretation possible.

The SPICE output file is downloaded by the Web-packaged Authorware piece for post-simulation processing. Performing post-simulation processing on the client machine instead of the server aims to achieve two significant advantages, (i) it takes the output processing burden off the server and (ii) less data is transmitted over the network making repeated execution faster. Two forms of post-processing have been implemented in this project: numerical and graphical. The former method simply parses the output file in search of a specific parameter and reads its value as string of text. The problem of searching for a specific element output in the output file is approached using a relative text mapping method. Output values are initially read as strings and later converted using Authorware's string manipulation commands. Since numbers in SPICE are represented in a scientific notation not interpretable by Authorware, a conversion process is necessary.

The idea behind graphical post-simulation processing is to present the numerical values in a visual form, which can help enhance the understanding of simulation results. A custom-made plotting program was used which was coded entirely using the Authorware scripting language. In one case Gnuplot, a public domain plotting program, was used to generate the plots instead. Gnuplot was invoked using a Perl script in the same way SPICE was launched.

Various authoring 'wizards' and tools have been developed in an attempt to simplify the authoring process. One such wizard is the *Perl Script Wizard*: its purpose is to eliminate the need for courseware developers to learn the Perl programming language.

Testing

As in any software development, testing is an important part of quality assurance. It ultimately aims to review the

requirements specification, design and coding methods employed in this project. Testing of the Web-based simulation system was carried out in three distinct stages: unit testing, integration testing, and validation testing. In the last case the tests were conducted with a random selection of electronic engineering students, academics, computer support staff, and other prospective users of the system.

CONCLUSION AND FUTURE WORK

The project has demonstrated the feasibility of embedding educational simulations into electronic design courseware, in particular, the EDEC courseware.

The EDEC courseware, however, is but one of numerous other legacy teaching applications that make use of simulation tools. Although the Perl/CGI approach could possibly be applied to most other web-based simulation applications, there are already scores of others that use different approaches, e.g. the Java Applet-Servlet-Applet approach. This contributes to the lack interoperability between these applications and hence is a hindrance to future integration. Furthermore, most of these systems consists of a single computer running several resource consuming applications. Currently, work is underway to extend and distribute the simulation processing onto several machines with different processing capabilities. This would also provide a heterogenous distributed environment that mimics the true nature of Web applications. The use of OMG's CORBA specification with Java programming language is expected to provide a means of integrating these systems together with the added benefit of platform independence.

ACKNOWLEDGEMENT

The authors wish to acknowledge the contributions of colleagues in the EDEC consortium whose names are too numerous to list here. The Teaching and Learning Technology Programme is jointly funded by the four UK higher education funding bodies, HEFCE, HEFCW, SHEFC and DENI, whose support is gratefully acknowledged.

REFERENCES

- [1] Thomas, R and Schnurr, C, "Simulations for education: the potential and reality", *Active Learning*, No. 9, December 1998, pp. 65-66.
- [2] Jong, T de, Andel, J van, Leiblum, M, and Mirande, M, "Computer assisted learning in higher education in the Netherlands, a review of findings", *Computers & Education*, Vol. 19, 1992, pp. 381-386.
- [3] Zillesen, P van S, "Using Educational Computer Simulations", Van Hall Institute, <http://www.xs4all.nl/~eszet/personal/educsim.html> 1998.
- [4] Hensgens, J, Rosmalen, P van, and Hahn, B, "Microelectronics simulation-based training on a virtual campus", *Elsevier Science B. V.*, Displays 18, 1998, pp. 221-229.
- [5] Fisher, S E and Michielssen, E, "Mathematica Assisted Web-based Antenna Education", *IEEE Transactions on Education*, Vol. 41, No. 4, Rapid Publications Supplement CD, November 1998.

- [6] Jong, T de, Joolingen, W van, Scott, D, Hoog, R de, Lapied, L and Valent, R, "SMISLE: System for Multimedia Integrated Simulation Learning Environments", in Jong, T, Sarti, L. *"Design and Production of Multimedia and Simulation-based Learning Material"*, Kluwer Academic Publisher, Netherlands, 1994, pp. 133-165.
- [7] Veith, T L, "World Wide Web-based Simulation", *International Engineering Education Journal*, Vol. 14, No. 5, 1998, pp. 316-321.
- [8] Regnier, J W and Wilamowski, B M, "SPICE Simulation and Analysis through Internet and Intranet Networks", *IEEE Circuit and Devices Magazine*, May 1998, pp. 9-12.
- [9] Fisher, S E and Michielssen, E, "Mathematica Assisted Web-based Antenna Education", *IEEE Transactions on Education*, Vol. 41, No. 4, Rapid Publications Supplement CD, November 1998.
- [10] Hicks, P J, Dagless, E L, Jones, P L, Kiniment D J, Lidgey, F J, Massara, R E, Taylor, D, and Walczowski, L T, "A computer-based teaching system for Electronic Design Education: EDEC", *The International Journal of Engineering Education*, Vol. 13, No. 1, 1997, pp. 20-28. See also <http://edec.brookes.ac.uk/>
- [11] Teaching and Learning Technology Programme (TLTP), see <http://www.ncteam.ac.uk/tltp.html>
- [12] Lorenz, P, Dorwarth, H, Ritter, K and Schriber, T, "Towards a Web based simulation environment", *Proceedings of the 1997 Winter Simulation Conference*, available at <http://www.informscs.org/wsc97papers/prog97.html>
- [13] Tait, K, "The Design and Production of Simulation-based Learning Environment", in Jong, Ton de, Sarti, Luigi, *"Design and Production of Multimedia and Simulation-based Learning Material"*, Kluwer Academic Publisher, Netherlands, 1994.