EMBEDDING CIRCUIT SIMULATIONS INTO ELECTRONICS DESIGN COURSEWARE

Saravanan Solaimalai[1] *and Peter Hicks* [2]

Abstract *¾ 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 ¾ Courseware, electronic circuit design, simulation, on-line Web-based tutorial.

INTRODUCTION

The advent of communication and information technology, particularly the World Wide Web, is bringing about a revolution in education. Higher education in particular has and still is undergoing significant changes to make the most of this revolution. There have been numerous efforts [1] to demonstrate the power of interactive computer-based learning material (courseware) as an aid to teaching and learning. By and large, these coursewares aim to benefit from the added interactivity computer-based simulations have to offer.

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 [2] 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 [3], and its benefits are numerous [4] - [7]. 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.

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 [2]. There is also a steep learning curve associated with simulations, and students using these tools often require close supervision and support [7]. 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

The use of the World Wide Web (WWW) to deliver learning material is a relatively recent phenomenon. Its advantages over classroom teaching include greater flexibility and accessibility, the potential for more interactivity, and portability due to its platform independence [1]-[4]. The Web is also more cost-effective [2] than any other alternative medium, and acts as an environment that can foster user motivation [3].

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

Due to the nature of its implementation, a single instance of the simulation tool (on the Web server) is sufficient for such a system, thus saving the cost and portability issue involved in installing the simulation tool in multiple machines. The saving of cost is particularly attractive to institutions of higher education as some simulation tools may not form a major component of a particular course but could come into use in certain instances. Licensing of the simulation tool may have to be altered, but would undoubtedly be cheaper. Other benefits of a Web-based educational simulation include wide accessibility, controlled access, efficient maintenance and increased integration [8]. Anyone with a standard Web browser (and sufficient permission) can access the simulation tool at any time of the day, and administrators can control access to it in an efficient and centralised manner. Maintenance of the simulation tool is also a matter of updating or repairing a single instance of it, i.e. on the server. Increased integration on the other hand refers to the fact that the simulation tool can be integrated with other Web browser-supported services with little effort.

Web-based Simulations 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 [11]. 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) [12]. 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 Engineeering (ESEE) is to extend the Web-based EDEC courseware material by embedding access to electronic simulation tools within it.

POSSIBLE APPROACHES

The diversity of methods employed in Web-based simulations makes it difficult to review every possible approach here. In general, established approaches for Web-based simulation delivery come under three categories [13]. 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*.

Of the three approaches outlined above, Remote Simulation (with CGI technology) was chosen as the initial project implementation method 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.

ESEE PROTOTYPE IMPLEMENTATION

The architecture of ESEE is essentially that of the Internet client-server architecture and can best be understood with reference to Figure 1. 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.

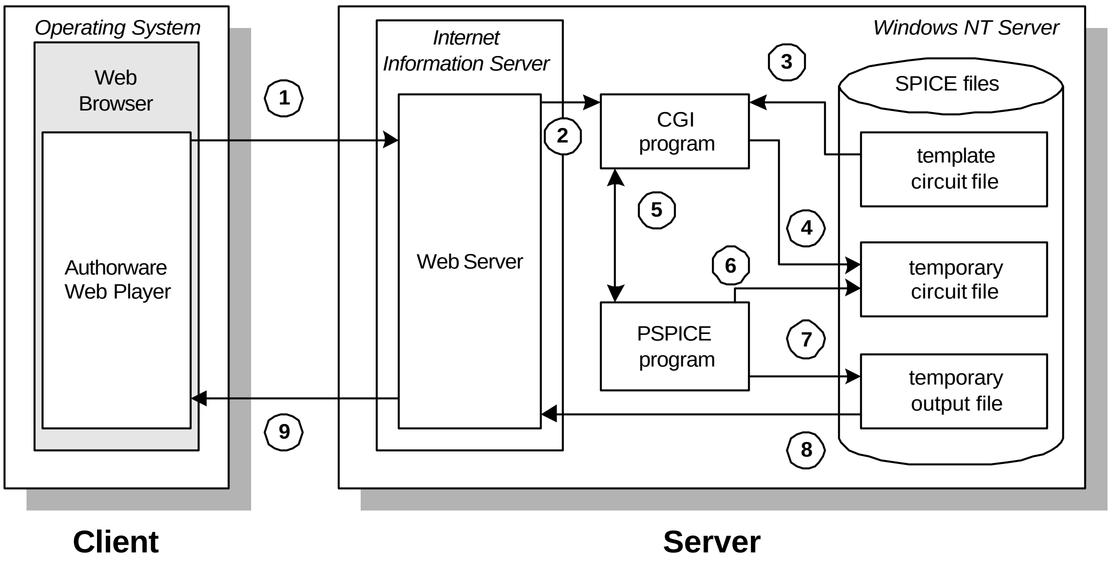


FIGURE 1
SIMPLIFIED FLOW DIAGRAM FOR THE WEB-BASED SIMULATION ENVIRONMENT.

The diagram in Figure 1 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 …
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

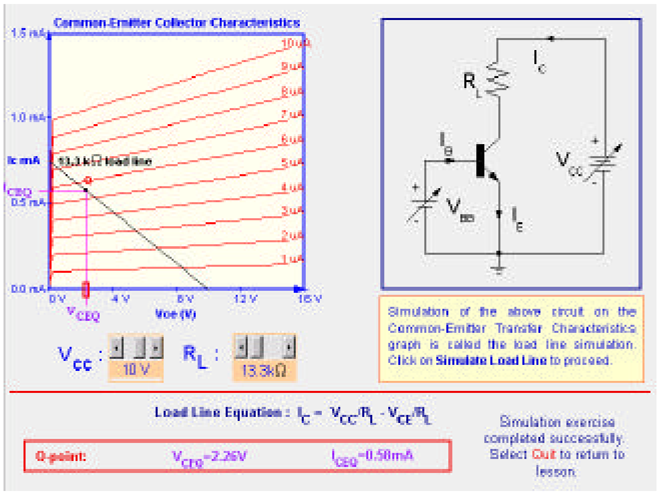
Examplars

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.

In order to extend the interactivity of the simulation environment beyond Authorware's standard interactive components, ActiveX controls were used. Values of all circuit components, including those which were not modified by the user, are sent to the Perl script for processing. A temporary circuit is generated by the Perl script, replacing the element values with those obtained from the Authorware Web-packaged piece.

SPICE is invoked to simulate the circuit described by the temporary circuit file using a CGI program. 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.


Simulation of the above circuit on the Common-Emitter Transfer Characteristics graph is called the load line simulation. Click on **Simulate Load Line** to proceed.

Load Line Equation : $I_C = V_{CC}/R_L - V_{CE}/R_L$

Q-point: V_{CEQ}=2.26V I_{CEQ}=0.58mA

Simulation exercise completed successfully. Select Quit to return to lesson.

FIGURE 2
GRAPHICAL POST-SIMULATION PROCESSING

The idea behind graphical post-simulation processing (Figure 2) 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.

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.

Subsequently, an evaluation exercise to study the effectiveness of EDEC as an electronic engineering courseware was conducted by external evaluators on the 20th of February 2002. The focus of the evaluation were the Web-based simulations exemplars embedded into the EDEC Courseware using the methods introduced in the ESEE project. Among other things the report has concluded is that embedding simulations into a courseware like EDEC adds to the value of the courseware itself. Students enjoyed the ability to explore the effects of varying circuit parameters and observing the outcome in real-time. Visualisation, like that provided in the exemplar, aided student comprehension of the behaviour of a simulated model, such as a diode I-V characteristics.

IMPROVING THE ESEE PROJECT

Component, or simulation model reuse is an important aspect of Web-based simulations. As Ernst[7] describes it, there are several key features that distinguish the future of simulation practice, impelled by the maturing of Web-based simulation, from the current and traditional approaches. These are:

- Digital object proliferation – as with the recent changes in the Web strategy, Web content of the future will be more object-oriented than document-oriented. With these changes, digital objects, in this context, simulation models, will increasingly become a commonplace on the Web.
- Software standard proliferation – numerous standards have been introduced in relation to interoperability of software. These have been critical in the defining and adaptation of Web-based simulations approaches. HLA, UML, CORBA, XML and OLE/COM are important standards in this respect.
- Model construction by composition – the proliferation of digital objects coupled with the intrinsic features of object-oriented approaches, such as inheritance and data encapsulation, would give rise to model construction by composition of existing digital objects.
- Proliferation of simulation use by non-experts – with the wide accessibility rendered to simulation tools by the Web, these normally expensive expert systems will become available to the masses.
- Multi-tier architecture and multi-language systems – the diversity of programming tasks dictates a diversity of programming styles and therefore languages. Multi-tiered application development, an increasingly dominant approach, is necessitated by the specialisation and optimisation of applications.

In the ESEE projects, some of these features are more apparent than others. ESEE was developed to cater for the needs of a specialist group, the education community, in particular the higher education institutions. Building the interface to the SPICE simulation software into the Authorware Web application allows non-experts (with reference to the SPICE simulator) to get a handle on executing circuits designs. This is particularly important requirement for the project as the system was designed as a courseware to teach electronic engineering related subjects and not how to use the simulator itself. In ESEE, model composition existed at simulator level, whereby SPICE models from third party could be added to the simulator model library, which was sufficient for its purpose.

As a prototype system, ESEE took the simplest path possible for a Web-based access to a remote simulation program. The CGI interface, a World Wide Web Consortium (W3C) standard established since 1993, is a means by which an external application can interact with a HTTP (Web) server. However, the CGI mechanism has its pitfalls[15]. The most pronounced of these is the process creation overhead and the security vulnerability it introduces.

Some argue that a solution to this problem would be to extend the Web server using its Application Programming Interface (API). Several Web server producers, e.g. Microsoft and Netscape, provide these API extensions, which aim to eliminate the shortcomings of the CGI interface. Such APIs, however, are proprietary in nature and owing to the sharing of environment, can cripple the Web server in the event of a malfunction.

By far, Java server-side solutions present the most attractive of CGI alternatives[1]. Besides the code mobility that the Java programming language provides, the Servlets single greatest advantage over a CGI program is that it can be launched once to service many clients. New Servlet requests require only a lightweight thread context switching, incurring much lower overheads. This, combined with Java's multi-threading, object-orientedness, rich class library and portability make it an increasingly popular replacement for CGI-Perl.

Simple tests were carried out with examplar programs to validate the notion that Java Servlet would serve ESEE better from a Web-based application point of view. Test results demonstrated that the two different implementations were, not surprisingly, comparable. There was no significant dissimilarity in response time, i.e. the speed of processing the simulation request. This can be attributed to the simplicity of the tasks involved. Perl is an interpreted language, which means, it is compiled every time it is invoked. Java, on the other hand, is compiled into bytecodes during coding. Nevertheless, bytecodes would still have to be 'interpreted' or converted into machine code before it can be run on the Java Virtual Machine. Hence, both mechanisms introduce an overhead not present in binary programs, such as C++. Conclusion from initial tests suggest the simplicity of the server tasks render the Perl-CGI approach more appealing than the Java Servlet approach.

Using XML for data interchange

ESEE exemplars used non-standard data exchange methods. It highlights the problems typically faced in data interchange with use of non-standard or proprietary interfaces between different software. The two most important software components that make up the interaction in ESEE are the Authorware application and the SPICE simulator. While the well-defined and documented format of SPICE netlists (input circuit files/decks) establishes a standard interface to circuit simulations on SPICE-based simulators, the diverse range of output formats has the exact opposite effect.

The various implementers of the SPICE simulator have taken the liberty to format the output data in a manner that suites their style of presentation. Although the outcome from an analysis is the same, a common extraction method cannot be used to parse spice output files from the different simulators. One way around this problem was to introduce another abstract component to the architecture, the Metadata Store. By devising a method that allows the Authorware application to make sense of the data in the output file rather than simply make a calculated guess as to where a particular data is located, the dependence on its proprietary structure/format can be eliminated.

The description of the data contained in the output file is called metadata. The metadata store would therefore contain converted SPICE output files describing the data contained in a particular output file. From this description, the Authorware application would be able to locate the data of interest for a particular simulation.

The metadata files would be written as XML (eXtended Markup Language) files. XML, a subset of the industry standard SGML, is syntax for the transport of information for exchanging structured data. The XML file will consist of a purpose written Schema [14] for SPICE circuit outputs.

Figure 5 shows the ESEE physical architecture after the XML components are incorporated. The XML Interpreter, XML Converter together with both the XML Output and DTD (Schema) stores would essentially make up the abstract component Metadata Store. Work on the XML conversion is still in progress.

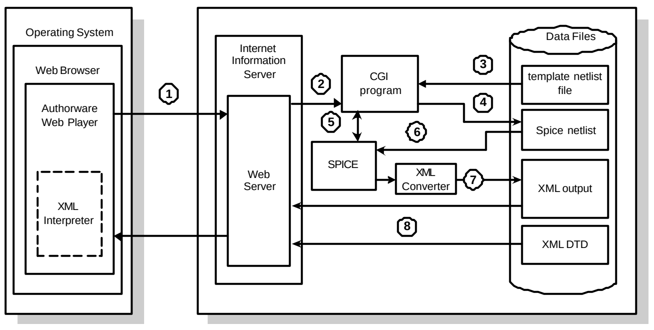


FIG 5
THE ESEE ARCHITECTURE WITH THE XML-MEDATA STORE

DISTRIBUTED PARALLEL PROCESSING

One of the initial objectives of the project was to design the system for future expandability yet keeping it simple for developers to construct new simulation exercises. Designing a system that meets such requirements is possibly made easier with a more modular or object-oriented approach. Performance is also a critical aspect of the overall design of the system as it is anticipated as being one of the more influential of factors from a user's point of view. The underlying mechanism has to cater for future expansions without allowing it to take a toll on the overall performance.

The Internet (and hence the Web) is a natural heterogeneous distributed collection of computers. Designing a system to operate on a single platform has it risks. If the simulation program, or any other program the Web-based simulation attempts to access terminates abnormally, it has the potential of affecting the other more critical services on the platform. Also, designing a system for one platform can make it useless in the future if there is a forced migration onto another system. Given that higher education institutions don't always use the same system for the same purpose, it is reasonable to expect that the proposed system would have less appeal if it were designed for only one kind of platform. These sorts of typical problems with Web application development can be addressed by using middleware, of which CORBA is one.

CORBA

CORBA stands for Common Object Resource Broker. It is a specification (rather than a group of products) that was developed by the Object Management Group (OMG)[16]. OMG contends that these specifications make it possible to develop a heterogeneous computing environment across all major hardware platforms and operating systems.

Unlike CGI and its alternatives such a Java Servlets, the CORBA implementation is component-based (object-based) and preserves the modularity that objects provide. Although

CGI-like architecture is currently the predominant model for creating 3-tier client-server solutions, the uptake of CORBA is increasing by the day[17]. Compared to competing standards like Microsoft's DCOM and Java's Enterprise Java Beans (EJB), CORBA is not vendor (or language) specific and has a large support base. CORBA specification is implementable in numerous languages, including C, C++, Java, Perl, and Python.

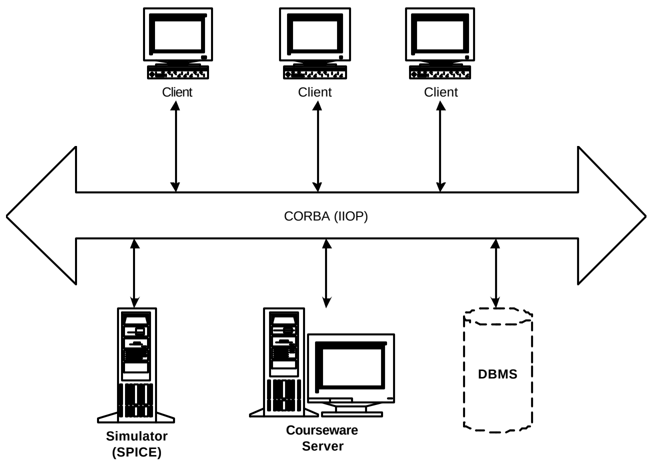


FIG 6
ESEE ON CORBA ARCHITECTURE

In ESEE's context, the implementation of the CORBA architecture would look like Figure 6. As can be seen from the figure, this architecture facilitates the addition of other components to the system such as a Database Management System (DBMS) to monitor student progress. More importantly, this lends itself to a more stable and possibly faster system. The different objects, i.e. applications, can reside on different machines, e.g. SPICE on a Unix server, the courseware in a dedicated Web server, and DBMS on an NT Server. Having a dedicated computer to run the different applications can also potentially increase speed of response although it is subject to network messaging latency.

CONCLUSION AND FUTURE WORK

In terms of feasibilty, the ESEE project has been successful in achieving its primary target of embedding an educational simulation into the EDEC courseware. Specifically, the exemplars created in this project demonstrate that controllable simulation can be implemented within a Web-packaged Authorware piece in a manner that is suitable for integration with the eventual Web-based EDEC courseware. Work of introducing XML for data interchange into the current CORBA-based system is underway.

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] Hicks, P.J., Dagless, E. L., Jones, P. L., Kiniment D. J., Lidgey, F. J., Massara, R. E., Massara, Taylor, D., 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.

[2] Thomas, R and Schnurr, C, "Simulations for education: the potential and reality", Active Learning, No. 9, December 1998, pp. 65-66.

[3] 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.

[4] Zillesen, P van S, "Using Educational Computer Simulations", Van Hall Institute, http://www.xs4all.nl/~eszet/personal/educsim.html 1998.

[5] 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.

[6] 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.

[7] 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.

[8] Veith, T L, "World Wide Web-based Simulation", International Engineering Education Journal, Vol. 14, No. 5, 1998, pp. 316-321.

[9] 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.

[10] 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.

[11] Hicks, PJ, 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/

[12] Teaching and Learning Technology Programme (TLTP), see http://www.ncteam.ac.uk/tltp.html

[13] 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

[14] Page, E. H. and Opper, J. M. (1999). "Investigating the Application of Web-based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model". Elsevier Science. Available at (Accessed 15 June 1999).

[15] Colburn, R., "CGI Alternatives," in CGI Programming in a Week. Indiana: Sams. net, 1998, pp. 251-265.

[16] "CORBA Success Stories". (2001). [Web] Object Management Group Inc. Available at http://www.corba.org/success.htm (Accessed 31 January 2002).