

Teaching Partial Differential Equations through Internet: an interactive approach

Manuel Alfonseca, Juan de Lara, Germán Montoro¹
Dept. Ingeniería Informática, Universidad Autónoma de Madrid
Ctra. De Colmenar, km. 15, 28049 Madrid, Spain
e-mail: {Manuel.Alfonseca, Juan.Lara, German.Montoro}@ii.uam.es

Abstract

This paper presents the procedures, tools and techniques that we use to teach partial differential equations through Internet. This is accomplished by means of an object oriented continuous simulation language called OOC SMP, a compiler (C-OOL) that is able to generate Java applets, and a tool (MGEN) that allows the student to interact with the problem, design the geometry, discretize it, declare equations, conditions, etc. Different interaction levels can be set by the teacher, allowing the student to change only some aspects of the problem definition.

Keywords

Web based simulation, Distance learning, Partial differential equations, OOC SMP.

1. Introduction

The growing popularity of Internet, and the increasing number of computers connected to it, makes it an ideal framework for remote education. Not only educational sciences, but also a large number of disciplines are re-thinking their traditional philosophies and techniques to adapt to the new technologies [Page00]. One of these disciplines is computer simulation.

Higher education tends to move from a teacher-centred paradigm to a student-centred paradigm [Maly98]. In this paradigm, the student plays a more active role in the learning activity. Web-based simulation is an effective framework for such a learning, that simplifies theory understanding, encourages 'learning by discovery' and experimentation and undoubtedly makes the learning process more pleasant. Web based learning using simulations is a good complement and reinforcement to traditional laboratory teaching; in the sense that students can experiment at any time at home.

There is also a change of mentality in students, they are ever more familiar with browsing the Web, playing computer games and interpreting graphics. Such students, in order to use simulation, desire tools that allow quick, easy and visual experimentation [Page99a]. This visual interactive simulation paradigm [Camp98] can result in some dangers, such as uncontrolled parameter changes by the users, which would invalidate the simulation results. Part of the solution could come from providing decision support means, such as the integration of quantitative simulators with qualitative models to provide guidance and explanations [Bred98], some kind of tutoring system, or simply the presentation of typical problem situations and the restriction of parameter changes and values. Another problem is that web published simulation models are made available to a mass of untrained users with a great possibility of misuse [Page99b].

There is a need for adequate tools to help in the elaboration of courses, which should make it possible to express all the possibilities offered by WWW teaching. As an answer to this necessity, we provide a set of tools to ease the development of technical or scientific educational courses through Internet. The courses would be composed of HTML pages and interactive on-line simulations, which help the student to understand the course subject.

The tools we have built integrate with each other easily and result in a very straightforward use. In particular, in this paper we present a tool (MGEN) designed to be used when the course deals with partial differential equations (PDEs). The tool promotes the student interaction with the problem, making it

¹ In alphabetical order

possible to design or change the problem geometry, discretize it, set the initial and boundary conditions, define the equation to be solved, etc. The teacher can also restrict the interaction freedom and make suggestions to the student.

This paper is organized as follows: section two presents the tools we use to generate web courses (OOCSSMP, SODA and C-OOL); section three presents MGEN; in section four an example of use is presented; and finally, in section five we discuss the conclusions and the future work.

2. Generating Web based simulations: OOCSSMP, SODA and C-OOL.

The models presented to the students are programmed in our own continuous simulation language called OOCSSMP [OOCSS00] which was born in 1997 [Alfo97] as an object oriented extension to the old CSMP [IBM72] continuous simulation language, sponsored by IBM in the 70's and 80's. The language was conceived with an educational focus in mind, in such a way that it has been mostly used to generate courses based on simulation [Alfo99], accessible from the Internet. The language has also been extended in other ways such as: handling discrete events, solving second order *PDEs*, generating distributed simulations, synchronizing multimedia elements with the simulation execution, etc.

A compiler (C-OOL, a Compiler for the OOCSMP Language) has been built to compile OOCSSMP models and generate three different object languages and environments:

- C++.
- C++/Amulet.
- Java.

The user interfaces generated can be configured by means of compiler options in the three cases. They allow the user to experiment with problem and answer "what if...?" questions. Our system fits in the Visual Interactive Simulation paradigm [Camp98]. In this paradigm, the user can interact with the simulation results while they are calculated, pause the simulation and change parameters, etc. Figure 1 shows the working scheme of the system.

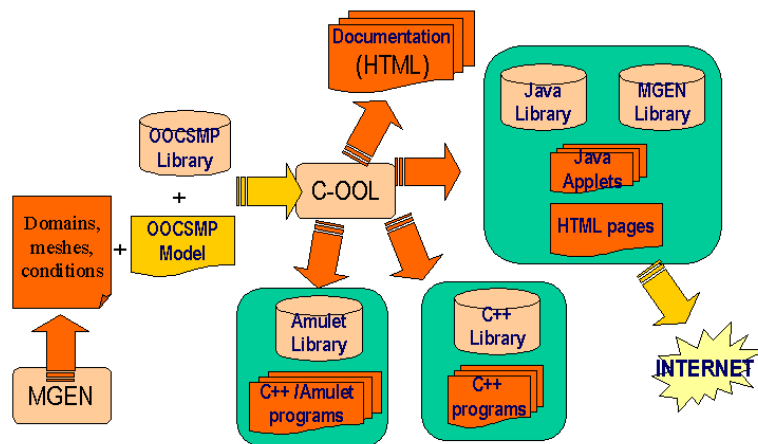


Figure 1: Working scheme of C-OOL

The possibility of configuring the user interface will allow us to adapt the simulation to the target user:

- If we know that the user is naive, we will restrict the possibility of making changes in the model.
- If the user is an expert, we will enhance the possibility to modify model parameters.

The teacher can identify interesting or typical situations in the model that arise when a parameter is modified, an object is added or an output display is changed. These situations of the simulation can be accessed by the student from the same user interface. For example, in the two-body problem, in which one of the bodies is fixed and the other orbits around, depending on the initial conditions set on the second body, we can obtain circular orbits, elliptic orbits, parabolic orbits, or free falls.

To integrate the model generated by C-OOL with web documents, we have extended the language with two higher-level layers, named SODA-1L and SODA-2L. The SODA-1L layer (Simulation Course Description Language 1st Level) allows us to describe web documents containing hypermedia elements that are not available in plain HTML, such as simulations, two dimensional graphics for functions, three dimensional graphics, and maps of isosurfaces. SODA-1L forms a higher language abstraction layer than OOCSSMP, because the models defined in OOCSSMP can be treated as hypermedia elements from the SODA-1L viewpoint.

The level called SODA-2L can group several SODA-1L pages to form a course, a presentation or an article. SODA-2L has primitives to add to the HTML pages navigation links, headers, footnotes, to create and place indexes, etc., which can be inserted in the resulting HTML pages or added as frames. In this level we add interface details common to all the pages, which makes the SODA-1L pages easy to reuse.

Figure 2 shows the organization of the three layers.

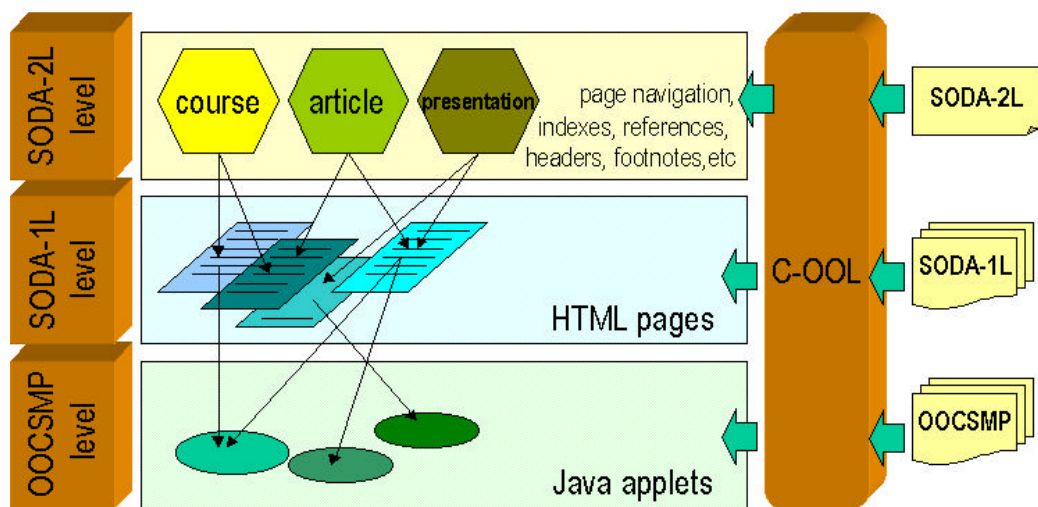


Figure 2: The three layer scheme used to generate simulation based documents.

3. Enhancing student interaction: MGEN.

MGEN is a tool that makes it possible to design graphically the geometry of a problem that can be described by means of partial differential equations, discretize that geometry and add conditions, generating OOCSSMP code that can be reused in the models. It works in two modes:

- Compilative, used as a tool to generate OOCSSMP code, and also read OOCSSMP code for the geometry.
- Interpretive, allowing the end user to design the problem geometry, the conditions, and the equation to be solved at runtime.

The student is thus able to:

- Design a geometry.
- Discretize it.
- Declare the equation to be solved.
- Declare and set initial and boundary conditions.

The possibility to declare equations and conditions inside MGEN means that we have to call an OOCSSMP interpreter to solve the equations. Since the learning process has been developed to be used through the Internet, MGEN and the OOCSSMP interpreter have been programmed in Java. Figure 3 shows a scheme of a user interaction with the system.

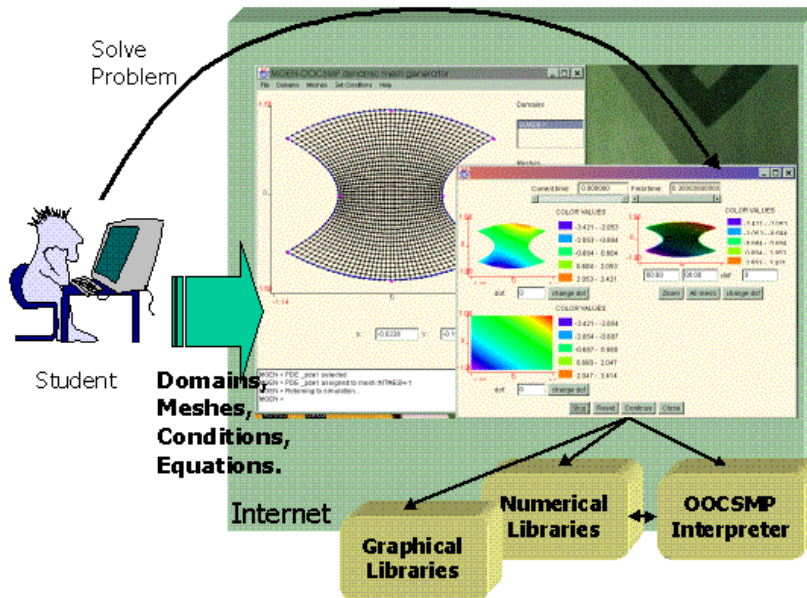


Figure 3: Student interacting with MGEN.

In the figure, MGEN is on the left, allowing the student to design a geometry, discretize it, set the problem conditions, and choose (or define) an equation to be solved. Then, the student changes to the main simulation panel (to the right) and solves the equation. In this panel, the exact solution can be seen on a square (bottom left of the panel) as well as the grid used to solve the equation (upper right of the panel). A more detailed explanation of the MGEN possibilities is given in section 4.

During the simulation, the student can pause the execution, return to MGEN and make changes to the geometry or the equations.

The teacher can reduce the freedom of the student to different extents:

- Propose predefined optimal equations to the student (the student must use the equations proposed by the teacher).
- Propose alternative expressions, which may be used as initial or boundary conditions (the student must use the expressions proposed by the teacher).
- Prevent the student to define equations.
- Prevent the student to define arbitrary initial or boundary conditions.
- Prevent the student to design a geometry. The teacher defines a geometry that cannot be changed, but the student can choose the appropriate gridding technique and simplex.
- Prevent the student to design or discretize a geometry. The teacher proposes a discretized geometry that cannot be changed.

These restrictions are useful to build a web course. The first pages may contain a simulation with few possibilities of interaction. As the student becomes more expert, the next pages will offer models with more possibilities of interaction.

4. An example

This section shows an example of the construction of a set of course pages, in the last of which the student has maximum possibilities of interaction. One equation and several conditions are proposed, but in the last page the student may freely change the equations and conditions and define a geometry.

To illustrate the procedure, we are going to use a progressive example. The course provides growing levels of interaction and freedom.

In the first page, we want the student to solve the two dimensional equation:

$$\frac{dU}{dt} + \frac{dU}{dxx} + \frac{dU}{dxy} + \frac{dU}{dyy} = 0$$

The exact solution of this equation is given by $e^{2t}\sin(x+y)\cosh(x+y)$ [Stri89].

Generating the calculated and exact solutions for the equation will require the teacher to write a few lines in the continuous simulation language OCSMP, as shown in listing 1.

```
[1] DATA exacta[75;75]
[2] DOMAIN qd := QUADRILATERAL(-1, -1, 1, -1, 1, 1, -1, 1
                             , INITIAL(SIN(X+Y)*CH(X+Y))
                             , ESSENTIAL(EDGE(1:4)
                             , -EXP(2.0*TIME)*SIN(X+Y)*CH(X+Y)))
[3] MESH m := ISOPARAMETRIC( qd, QUADRILAT4, ELEMENTS(75,75) )
[4] PDE pde1( 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, IMPLICIT)
[5] m.setPDE(pde1)
[6] DYNAMIC
    exacta[ROW;COL] := EXP(2.0*TIME) * SIN((-1+COL*0.027) +
        (-1+ROW*0.027)) * CH((-1+COL*0.027) +
        (-1+ROW*0.027))
    m.STEP()
[7] TIMER FINTIM:=0.3, delta:=0.005, PLdelta:=0.05
[8] ISOPLOT [C], m
[9] ISOPLOT [S], 1, 1, -1, -1, exacta
[10] GRIDPLOT [E], m
```

Listing 1: OCSMP code to solve the problem.

In this first example the PDE [4], the mesh [3] and the domain [2] are established by the teacher and the student doesn't have the opportunity to change the model. The student is only allowed to start the simulation and see the results. Three output forms have been selected to visualize the results: two maps of isosurfaces that show the calculated [8] and the exact [9] solutions (the latter on a square of side 2), and a graphic to visualize the nodes of the grid [10]. An instant of this model execution is shown in figure 4.

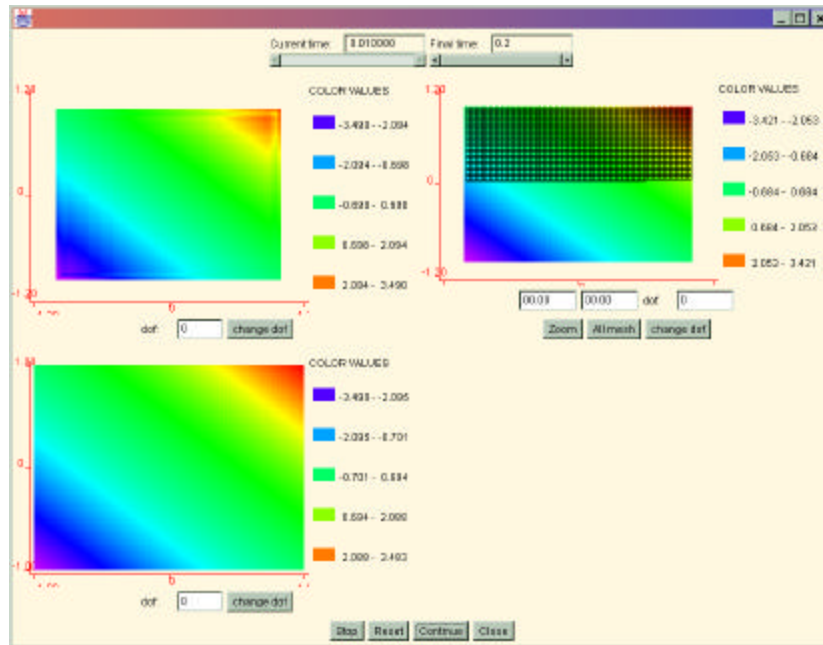


Figure 4: A moment in the solution of the problem.

In the previous figure, the exact solution is shown at the lower left corner, the calculated solution at the upper left corner and the grid on the upper right corner.

Going one step further, the teacher can enable the use of MGEN, so that the student is able to design the domain, mesh it, assign initial and boundary conditions and assign the pre-defined equation to the mesh. At this level of interaction, it is necessary to change some lines in the previous OOC SMP code:

```
[ 2 ] DOMAIN qd := MGEN( DYNAMIC( SIN(X+Y) *CH(X+Y) ) ,
                        DYNAMIC( EXP( 2.0*TIME) *SIN(X+Y) *CH(X+Y) ) )
```

This line defines the functions that the student will be able to choose from, as the initial and boundary conditions.

```
[ 3 ] MESH m := MGEN( )
```

Which means that, using MGEN, the student will design the mesh later.

[5] This line disappears to allow the student to select the PDE to be solved.

With these lines, in the OOC SMP code, the C-OOL compiler will produce an applet window which invokes MGEN, where the student will be able to design graphically the geometry of the problem, discretize it, assign conditions, etc. Figure 5 shows one of the steps while designing the problem geometry with MGEN.

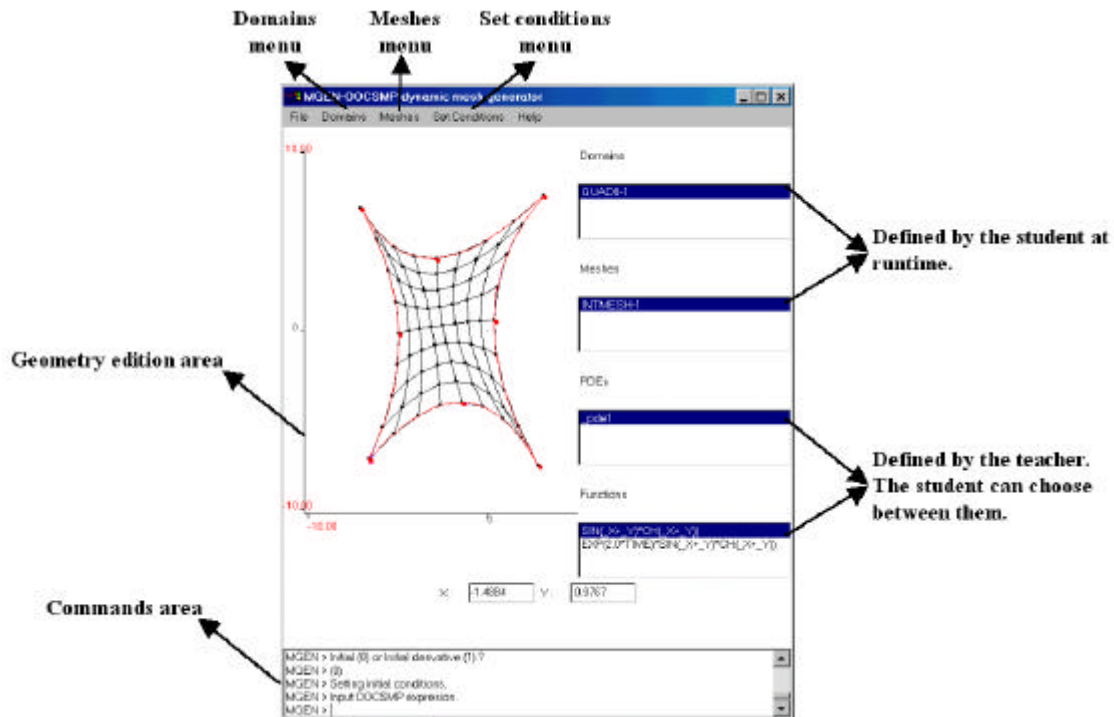


Figure 5: Example of a MGEN window for the above code.

The *domains menu* allows the student to create new domains to be discretized. Students can choose between circular sectors, triangles and 4 or 8 node quadrilaterals. Once selected the student can design it freely in the *geometry edition area*, using the mouse.

The *meshes menu* allows the student to mesh a selected domain as well as set and unset the equation to be meshed. Students can choose between Delaunay, interpolation and elliptic techniques to mesh the domains with triangles (3 or 6 nodes) or quadrilaterals (4 or 8 nodes).

The *set conditions menu* allows the student to establish the initial and boundary conditions (natural or essential). These conditions can be freely established using the *commands area* and/or selected from the function and PDE selection lists to the right, depending on the restriction level set by the teacher.

Figure 3 shows a moment in the execution of the simulation with this degree of interaction.

In the last and higher level of interaction, the teacher allows the student to introduce functions and PDEs using the MGEN interface at runtime. Lines [2] and [4] in the previous OOC SMP code should be programmed thus:

```
[2] DOMAIN qd := MGEN( DYNAMIC(SIN(X+Y)*CH(X+Y)) ,
                      DYNAMIC(EXP(2.0*TIME)*SIN(X+Y)*CH(X+Y)) ,
                      USER )

[4] PDE pde1 ((0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, IMPLICIT),
             USER)
```

With these lines, the teacher offers pre-defined functions and a PDE, but allows the student to introduce new ones at simulation time using MGEN.

This simulation applet can be included easily in an HTML page and in a web course using the SODA-1L and SODA-2L levels. The two first models, with the two lower levels of interaction, are accessible from: <http://www.ii.uam.es/~jlara/investigacion/ecom/pdes/indicepde.html>

These examples show that the system can be adapted easily to different users or to a user through a learning process. The OOC SMP programs are very simple and go from a basic level, where the student

can only visualize the results, to more complicated stages, where the student has total freedom of interaction.

5. Conclusions and future work

In this paper we have presented OOCSMP, a language that makes it easy to write simulation models, in particular those that can be described by partial differential equations; C-OOL, a compiler of OOCSMP that generates C++, C++/Amulet and Java code; and SODA, a set of language extensions to include the simulation models in web courses.

These tools have been enhanced with the development of MGEN, which allows the student to interact with the problem at runtime. Using MGEN, the student can design a geometry, discretize it, declare the equation to be solved, and declare and set initial and boundary conditions.

These possibilities made it necessary to integrate an OOCSMP interpreter with MGEN. In this way, the student is able to make changes to the geometry or the equations during the simulation process.

The teacher has several ways to restrict the student actions so that the course can adapt in a progressive way to the student necessities and capabilities. Novice students will have a very limited control of the system, while advanced students will have total freedom of interaction.

In the future, we are planning to extend the OOCSMP interpreter, so that the student can define entire simulation models at runtime. We are also planning to add some kind of tutoring system to help the student to carry out typical tasks with the environment. Some work has been done, but just for the case when the object language generated by the compiler is C++/Amulet [Alf98]. This tutor could also help the student to choose the most appropriate form of discretizing the geometry, the simplex or the resolution method. This is clearly similar to the approach proposed by the Problem Solving Environments (PSEs) [Aker97] [Rice00], although these environments are specialized in PDEs, and they don't usually deal with course design and construction, student profiles, etc.

We are also working to provide a graphical interface for the construction of the simulation models. Currently, they have to be written using a text editor. Ideally, the tool would provide facilities for collaborative programming across the Internet, covering in this way another aspect of the term "web-based simulation" [Cube98] [Fish98].

Acknowledgement

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL1999-0181

References

- [Aker97] Akers, R.L. Kant, E., Randall, C.J., Steinberg, S., Young, R.L. (1997). "Problem Solving Environments and the Solution of Partial Differential Equations". In Internet at: <http://www-cgi.cs.purdue.edu/cgi-bin/acc/pses.cgi>.
- [Alf97] Alfonseca, M., Pulido, E., Orosco, R., de Lara, J. (1997). OOCSMP: An Object-Oriented Simulation Language. *Proceedings of the 9th European Simulation Symposium ESS97* (pp. 44-48). SCS Int., Erlangen.
- [Alf98] Alfonseca, M., García, F., de Lara, J., Moriyón, R. (1998). *Generación Automática de Entornos de Simulación con Interfaces Inteligentes* (pp. 5-13). ADIE, Octubre Diciembre 1998.
- [Alf99] Alfonseca, M., de Lara, J., Pulido, E. (1999). Semiautomatic Generation of Web Courses by Means of an Object-Oriented Simulation Language. *Special issue of SIMULATION, Web-Based Simulation* 73, 1: 5-12.
- [Bred98] Bredeweg, B., Winkels, R. (1998). Qualitative Models in Interactive Learning Environments: an Introduction. *Special issue of Interactive Learning Environments on "the Use of Qualitative Reasoning Techniques in Interactive Learning Environments"* 5: 1-18.

[Camp98] Campos, A.M.C., Hill D.R.C. (1998). An Agent-Based Framework for Visual-Interactive Ecosystem Simulations. *TRANSACTIONS of the SCS International* 15, 4: 139-152.

[Cube98] Cubert, R.M., Fishwick, P.A. (1998). OOPM: An Object-Oriented Multimodeling and Simulation Application Framework. *SIMULATION* 70, 6: 379-395.

[Fish98] Fishwick, P.A. (1998). Issues with Web-Publishable Digital Objects. *Proceedings of SPIE: Enabling Technologies for Simulation Science II* (pp. 136-142).

[IBM72] IBM Corp. (1972). *Continuous System Modelling Program III (CSMP III) and Graphic Feature (CSMP III Graphic Feature) General Information Manual*. IBM Canada, Ontario, GH19-7000.

[Maly98] Maly, K., Overstreet, C.M., González, A., Denbar, M., Cutaran, R., Karunaratne, N., Srinivas., C. J. (1998). Use of Web Technology for Interactive Remote Instruction. *Proceedings of the Web'97 Conference*. In Internet at : <http://www7.scu.edu.au/programme/posters/1855/com1855.htm>

[OOCSM00] OOC SMP home page: <http://www.ii.uam.es/~jlara/investigacion>

[Page99a] Page, E.H. Buss, A., Fishwick, P.A., Healy, K., Nance, R.E., Paul, R.J. (1999). Web-Based Simulation: Revolution or Evolution?. To appear in *ACM Transactions on Modeling and Computer Simulation*.

[Page99b] Page, E.H., Opper, J.M, (1999). Investigating the Application of Web-based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model. To appear in *Future Generation Computer Systems*, Elsevier Science Publishing

[Page00] Page E.H. Buss, A., Fishwick, P.A., Healy, K., Nance, R.E., Paul, R.J. (2000). Web-Based Simulation: Revolution or Evolution?. To appear in *ACM Transactions on Modeling and Computer Simulation*.

[Rice00] Rice, J. R. (2000). Problem Solving Environments for Scientific Computing. (Deville and Owens eds.). *Session organized in the IMACS'2000 congress*. Book of abstract (pp. 10-15). Lausanne, 21-25 August.

[Stri89] Strikwerda, J.C. (1989). *Finite difference schemes and partial differential equations*. Chapman & Hall, New York.