Developing adaptive web courses based on simulation: A thick client approach

Manuel Alfonseca, Juan de Lara, Alfonso Ortega[1]

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

Abstract
In this paper we present some extensions that we have added to our tools to develop adaptive simulation based web courses. The tools allow the construction of web documents enriched with visual interactive simulations and other hypermedia elements. For this purpose, we use a continuous simulation language (*OOCSMP*) that is composed of several abstraction layers: the first describes the simulation models' behaviour; the second describes pages or slides; and the third builds courses, articles or presentations. The new extensions allow including different texts, images and simulations depending on the user characteristics. The compiler for the OOCSMP language (named C-OOL) generates HTML pages prepared to run entirely on the client, using technologies such as Java, JavaScript and cookies.

Keywords
Web-based simulation, Web-based applications, Distance learning, Automatic code generation, Adaptive courses.

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

1. Introduction
The web is changing the way we work. In fact, more and more frequently, we see educational courses, articles, and presentations when navigating in the web. These publications sometimes can be accessed on-line, sometimes must be downloaded. On-line documents run from a simple transposition of lecture notes, to pages including more sophisticated elements, such as animated graphics, simulations and so forth.

The Internet is accessed by millions of people, with very different characteristics, knowledge and preferences. This means that it would be good for Internet-published materials to be adpative in the sense that different people would see different views. In some sense, this is already done in the best commercial web pages, such as Amazon (Amazon, 2000).

In the case of educational materials, this need is even greater. There are many levels of students, depending on their previous experience, their interests, or their capabilities, both on the subject matter objective of the course, and on the use of computers. The best courses would consist of different study trajectories, adapted both to individual students and to the path they have followed during their prior navigation through the course (de Bra99). A few existent course writing systems already provide some of these opportunities (Carro et al 1999, Brusilovsky et al 1998, Weber and Specht 1997, de Bra et al 1998).

Our approach is somewhat different, in the sense that our courses are based on thick-client technologies, such as Java, Javascript or cookies. Using these tools, the student can select information adapted to a certin level, but can also access other pages, paragraphs or views that would correspond to a different level, which would make the course more flexible and allow dynamic changes in perspective which usually are not possible with other courses.

This article is organized as follows: section 2 presents the system we use to develop documents for the web containing interactive simulations, section 3 describes the extensions that we have added to the system to produce adaptive web courses, section 4 shows an example and section 5 expounds the conclusions and the future work.

2. OOCSMP, SODA-1L, SODA-2L, an overview.
The *OOCSMP* continuous simulation language was conceived in 1997 (Alfonseca et al 1997) as an object oriented language. A compiler (*C-OOL*) was built for this language in order to produce C++ code or Java applets from the simulation models. This approach would simplify the generation of simulation based web courses. In fact, several of them have been generated using this language (for gravitation, partial differential equations, ecology and basic electronics) which can be accessed from:

http://www.ii.uam.es/~jlara/investigacion

The language and the compiler have been designed with an educational focus, with the following features:

[1] In alphabetical order

- It is possible to include several forms of output displays in the same simulation.
- The user interface can modify parameters, object attributes or even add or delete objects during the simulation execution. This gives the student more interaction possibilities.
- The user interface can be configured by means of compiler options. For example, if we are going to present a simulation to a naive user, we will restrict the interface to prevent the user to change model parameters. On the contrary, if the user is an expert, we will provide more possibilities to change the model, such as buttons to add or delete simulation objects.
- Alternative simulations can be designed, to be accessed from the main simulation. In this way, the teacher can plan interesting situations that arise when a parameter is changed, an object is added, etc.
- Multimedia elements (video, dynamic text, sound or virtual reality) may be included and synchronised with the simulation execution. Thus, model behaviour explanations may be presented in the appropriate moment.

Two higher level layers (de Lara and Alfonseca 2001) allow describing the HTML pages of the document, and grouping them to form educational courses, articles or presentations.

The description of document pages is covered by a set of instructions that we call SODA-1L (Simulation Course Description Language 1st Level). This set of instructions allow 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 OOCSMP, because the models defined in OOCSMP can be treated as hypermedia elements from the SODA-1L viewpoint.

The level called SODA-2L (Simulation Course Description Language 2nd Level) can group several of the SODA-1L pages to form a course, a presentation or an article. SODA-2L has primitives to add navigation links to the pages, headers, footnotes, create and place indexes, etc. They can be 'embedded' in the resulting HTML pages, or added as frames. In this level we usually add interface details that are common to all the pages, which makes the SODA-1L pages easy to reuse. A picture of the three layers and their interrelations is shown in figure 1.

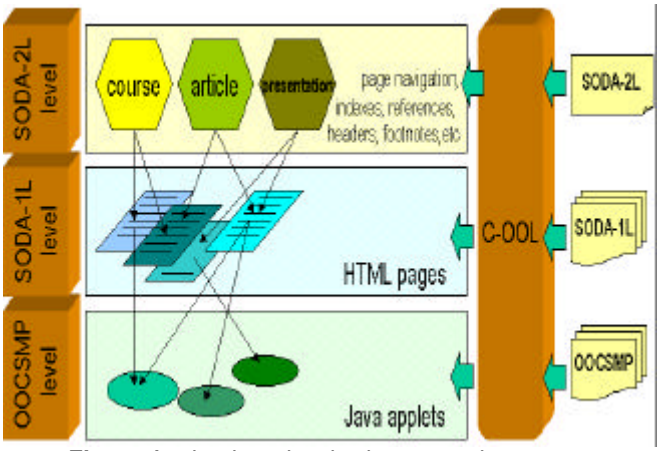

Figure 1: The three level scheme used to generate simulation based documents.

3. Extensions to generate adaptive courses.

We have extended our system with instructions that permit the adaptation of the document pages, depending on the user type. For the moment, the new instructions have been added in the SODA-1L and the SODA-2L levels.

At the SODA-2L level, the course designer has to identify the types of user that will access the document pages. The syntax for this instruction is:

```
USER="<user_type1>"(,<user_type2>")*
```

This instruction is not obligatory, and it is not necessary to include it if just one type of user is targeted. If the instruction is included, an entry page is created automatically for the document, that is constructed using JavaScript. In this page, the user has to provide her name and type. This information will be kept in the user's computer by means of cookies. If the system detects the user has accessed the document before, it asks her whether she wants to start again or continue from the last page in her last visit.

In this entry page, the user is also presented a text describing the different user types. In this way, she can choose the most appropriate. The description of the user types is accomplished by means of the SODA-2L instruction:

```
USERDEFINITION= "description of user type-1
(in SODA-1L format)",
(, "description of user type-n (in SODA-1L
format)")*
```

During her access to the document, the user will be presented the information designed for the type of user she belongs to. But, at any moment, the user can access the information for other user types. If the user access frequently the information of other user types, she is asked by the system if she wants to change her user type. In this way she will be able to check the type of information each type of user receives, and choose the

most appropriate. The number of times the user can access information of other user types before she is offered to change her user type can be configured by means of the SODA-2L instruction:

```
USERCHANGE <n>, "<message>"
```

Where <*message*> is the message that is presented to the user when she is offered to change the level. If this instruction is not present, the user is not asked for change.

We think that allowing the user type to be changed is very important, because the user may not know *a priori* which user type she belongs to; or her characteristics may change with time. User change is not permitted in adaptive learning systems such as TANGOW (Carro et al. 1999).

At the SODA-1L level, the designer must describe the HTML pages of the document. This level provides instructions to add text, images, links, tables, simulations, 2D-graphics, 3D-graphics, maps of isosurfaces, etc. to the page. All these instructions have been extended in such a way, that now it is possible to express different alternatives for the different users defined in the SODA-2L level. For example, the instruction to add text to a page now has the following syntax:

```
DESCRIPTION [USER="<user_type>"], [<text in SODA-1L format>]
(, [USER="<user_type>"], [<text in SODA-1L format>])*
```

In this way, for each paragraph in the text, it is possible to include different texts depending on the user type. When processing this instruction, the compiler generates a piece of JavaScript code. This code will generate an HTML table with two columns and one row, the column to the right has the text of the corresponding user level. The left column has a widget that allows the user to change her level (and the associated text on the column to the right). This widget is controlled by means of JavaScript functions, that the compiler generates automatically. Thus, with these extensions we provide adaptive presentation and adaptive navigation.

If the previous instruction is inserted without the `[USER="<user_type>"]` parameter, the compiler generates plain HTML code for the paragraph.

Other SODA-1L instructions, to include images, simulations, etc., have a similar syntax structure. The case of the simulations is especially interesting, because of the possibility of changing a simulation depending on the user type. For example, if the user is naive, we will restrict her possibilities to change model variables by generating fewer buttons in the user interface, perhaps we will include multimedia elements in the simulation. On the contrary, if the user is expert, we will make available the possibility to change parameters, by generating more buttons in the user interface, even buttons to add or delete objects during the simulation execution (Alfonseca et al 1999a).

4. An example: A course on gravitation.

We have experimented with the adaptation capabilities of our system using a course on gravitation we had developed previously (Alfonseca et al 1999b). The course has been enhanced with adaptation capabilities and with one more page. The course consists of 7 pages:

- A simple description of Newton's Mechanics with different solutions to the two-body problem (a free fall following different orbits: a circle, an ellipse, a parabola, a hyperbole and a straight line).
- A model of the solar system, as a practical example of the n-body problem. For convenience, the solar system is shown in two separate parts: the inner system (from Mercury to Jupiter) and the outer system (from Jupiter to Pluto), with different time scales.
- The same model, using a Virtual Reality panel and a 2-D plot (this page is new). The user can click on the 3D planets, and a table is shown with planet's data, including its mass, diameter, distance to the sun, etc.
- A model of the Sun-Earth-Moon system. The time and plot scales are adjusted to make the two orbits distinguishable.
- The discovery of Neptune by John Couch Adams and Urbane Jean-Joseph Le Verrier, indicating how this discovery transformed an apparent failure of Newton's Mechanics into an outstanding success. The discovery is illustrated by a double simulation of Uranus's orbit, in the presence and in the absence of Neptune.
- A geo-stationary satellite which keeps its distance to the Earth constant has been simulated using the same model, changing only the values of the constants and the instanced objects (members of the class Planet). The effect of the Moon on the satellite's orbit is illustrated by performing a double simulation in the presence and in the absence of the Moon. To test the second case, we only have to change the mass of the Moon to zero.
- Finally, the last page leaves the student freedom to experiment with the simulated solar system, providing the ability to change the planet parameters and the universal constants (the mass of the Sun or the gravitational constant), to play at answering what-if questions.

For this course, we consider two user types: expert and naive. Listing 1 shows the SODA-2L script necessary to build the course.

```
[1] INCLUDE "macros.csm"
[2] INCLUDE "styles.csm"
[3] COURSE "Universal Gravitation and Newton's
    Mechanics" BACKGROUND="WHITE"
[4] USER= "Advanced", "Naive"
[5] USERDEFINITION="Select this \BOLD{user type}
    if you have some notions about
    \BOLD{Newton's mechanics}", "Select this
    \BOLD{user type} if you don't know anything
```

```
    about     \BOLD{Newton's     mechanics}     or
    simulation"
[6] FONT    TITLE    TYPE="Tahoma",    SIZE="+4",
    COLOR="BLUE"
[7] FONT TYPE="Arial,Helvetica", SIZE="+2"
[8] AUTHOR J.de Lara, M.Alfonseca, A. Ortega
[9] EMAIL                    Juan.Lara@ii.uam.es,
    Manuel.Alfonseca@ii.uam.es,
    Alfonso.Ortega@ii.uam.es
[10]    WEBADDRESS    http://www.ii.uam.es/~jlara,
    http://www.ii.uam.es/~alfonsec,
    http://www.ii.uam.es/~alfonso
[11]    NAVIGATION  [TABLE, 80]
[12]    SIMULATIONS   -noFrame   -noScaleWindow   -
    noLeyenda -WIDTH= 500 -HEIGHT= 350
[13]    PAGE "grav.csm" NAVIGATION [2]
[14]    PAGE "igrav1.csm" NAVIGATION [1,3]
[15]    PAGE "igrav2.csm" NAVIGATION [2,4]
[16]    PAGE "igrav3.csm" NAVIGATION [3,5]
[17]    PAGE "igrav4.csm" NAVIGATION [4,6]
[18]    PAGE "igrav5.csm" NAVIGATION [5,7]
[19]    PAGE "igrav6.csm" NAVIGATION [6]
```

Listing 1: A SODA-2L script used to compile the course.

The third line indicates that we are compiling a course, alternatives are presentations and articles. The next line (4) declares the user types, and line 5 describes these user types. Lines 6 and 7 declare the font type to be used in all the course pages. Lines 8 to 10 declare the authors' data. This data can be accessed naming variables `AUTHOR`, `EMAIL` and `WEBADDRESS`, in page indexes and headers. In this way, the indexes and headers become more general. Line 11 defines the navigation links appearance, they will be presented in a table filling 80% of the HTML page. It is also possible to include them in the form of lists. Line 12 sets some options on the user interface that will apply by default when compiling the simulation models. Lines 13 to 19 declare the course pages (SODA-1L files) and the navigation between them: each page will have a link to the next and to the previous. Declaring the navigation in this way allows reusing the document pages in other documents.

Examples of SODA-1L code can be found in (deLara and Alfonseca 2001). In our example, the page should include a simulation model (programmed in OOCSMP), that changes depending on the user type. If the user is inexpert the model will have a dynamic text explaining what happens in the simulation. This text is synchronised with the simulation execution.

The OOCSMP model presents several interesting situations that occur in the 2-body problem. Depending on the initial conditions of the problem, we can simulate a free fall following different orbits: a circle, an ellipse, a parabola, a hyperbole and a straight line. These alternatives are accessible from the same simulation program. The OOCSMP code necessary for this problem (for the advanced user) is shown in listing 2.

```
[1]   TITLE GRAVITATION
[2]   DATA G:=0.00011869, PI:=3.141592653589793,
            MS:=332999
[3]   INCLUDE "Planet.csm"
[4]   Planet Earth("Earth",1,0,1,-6.29,0.107, 0)
[5]   DYNAMIC
[6]     Earth.STEP()
[7]   TIMER delta:=.0005,FINTIM:=2,PLdelta:=.01
[8]   METHOD ADAMS
[9]   \
[10]  TITLE ELLIPSE
[11]  TIMER FINTIM:=100, PLdelta:=.05
[12]  DATA Earth.XP0:=-8.5
[13]  PLOT 3.2, 2.5, -4.5, -11
[14]  \
[15]  TITLE ELLIPSE
[16]  TIMER FINTIM:=140, PLdelta:=.3
[17]  DATA Earth.XP0:=-8.8
[18]  PLOT 8.0 , 2.5, -8.7 , -57.31
[19]  \
[20]  TITLE PARABOLE
[21]  TIMER FINTIM:=75, PLdelta:=.1
[22]  DATA Earth.XP0:=-8.9
[23]  PLOT 2.5 , 2.5, -26.0, -120.0
[24]  \
[25]  TITLE HYPERBOLE
[26]  TIMER FINTIM:=75, PLdelta:=.1
[27]  DATA Earth.XP0:=-9
[28]  PLOT 2.5 , 2.5, -57.0, -140.0
[29]  \
[30]  TITLE FREE FALL
[31]  TIMER PLdelta:=.001
[32]  DATA Earth.XP0:=0
[33]  PLOT 0.5 , 1.5, -0.5 , -0.5
```

Listing 2: OOCSMP code for the 2-body problem (file grav00.csm).

The first line declares the model's title. Line 2 declares some constants. Line 3 includes an OOCSMP file that contains an OOCSMP class named *Planet*. This class encapsulates all the behaviour of a Planet. More details about this can be found at (Alfonseca et al 1999b). Line 4 declares an object that represents one of the bodies, the second one is supposed to be at the origin of coordinates. Lines 5 and 6 are the main simulation loop that just invokes a method on the object. Line 7 and 8 declare some simulation parameters. At line 9 begins the declaration of the different simulation alternatives. Each '\' begins the declaration of a variation in the main problem. In our case, the variations consist in changing the initial position of the first body, the scale of the graphic and some simulation parameters. These instructions will generate a button each. When the user clicks on the button, the execution of the corresponding variation of the main model will start.

The model for the inexpert user is almost the same, but we have added an explanation to each situation. Information about embedding and synchronising multimedia inside simulations can be found in (Alfonseca and de Lara 2000).

Figures 2 and 3 show a piece of this page for both user types. On the left side of both simulation applets, it can be seen the buttons to change between the different model variations. At the bottom of the applet, there are some buttons to control the simulation execution. Note that the advanced user interface has two mor buttons: the one labeled with 'Earth' is used to change the attributes of the body that orbites, and the labeled with 'Globals' to change global variables such as the mass of the immobile

body, the universal gravitation constant, etc. In this way, the advanced user can also choose to design her own experiments (restricted to the equations of the model), and has more possibilities of interaction. On the other hand, the simulations of the naive user are more guided, and textual explanations are given below the graphic, the explanations change depending on the variation of the model that is being executed. In the example, the user has clicked on the botton labeled as *1*, the *Ellipse* experiment.

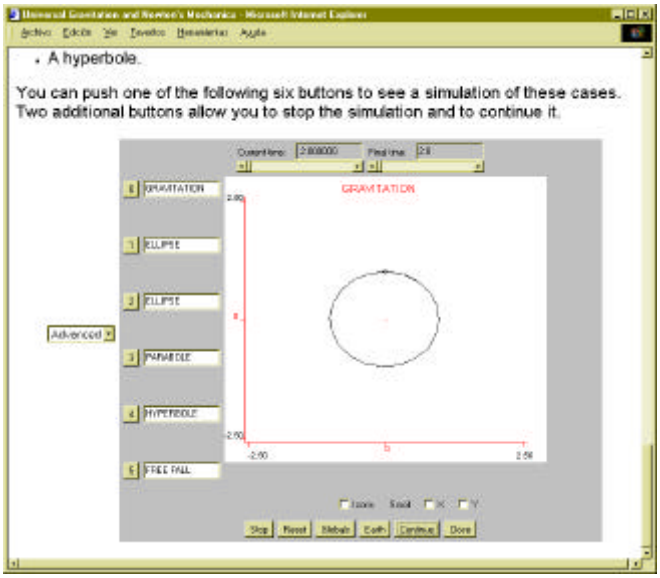

Figure 2: The page for the naive user

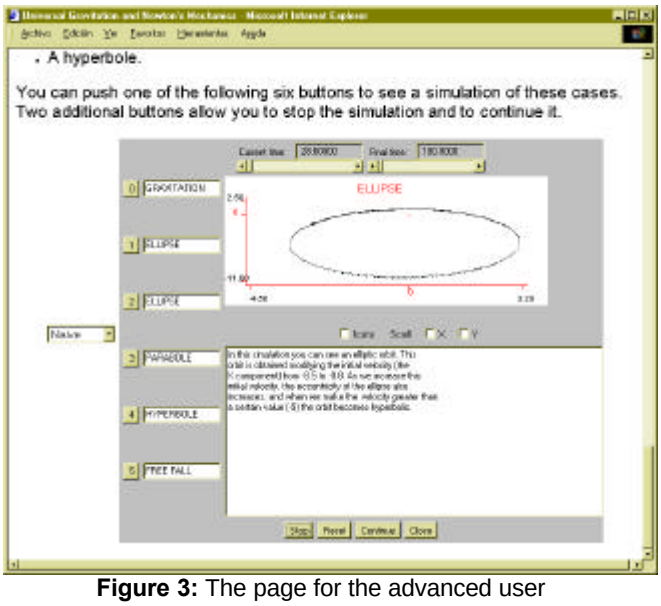

Figure 3: The page for the advanced user

5. Conclusions.

This paper has presented some tools that make easy the construction of adaptive web courses based on simulations. Our approach uses thick-client computation, that reduces the need of an expensive server. We think this is a good approach, because of the large number of users that can access potentially to the course at the same time, and the growing capacity of the personal computers nowadays.

The three-layer language used to integrate interactive simulations with the construction of web documents puts stress on several key points in the development of web based applications, such as maintainability, reusability (of pages, headers, footnotes, simulation models, etc), easy testing, common look, etc. In the future we expect to test the system with real students and to obtain feedback from them. Other useful elements can also be added to this system, such as the possibility of proposing exercices to the student, or to indicate her that path that has followed and the remaining pages to complete a lesson, etc.

7. References

Alfonseca, M., Pulido, E., Orosco, R., de Lara, J. 1997. *"OOCSMP: an object-oriented simulation language"*. ESS'97, Passau, pp. 44-48.

Alfonseca, M., de Lara, J., Pulido, E. 1999. *"Dynamical object generation during the execution of continuous simulation models"*. Proc. ASOO'99, Third Argentine symposium on Object Orientation, Buenos Aires, pp. 89-102.

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, Vol 73, num.1, July 1999, pp. 5-12.

Alfonseca, M., de Lara, J. 2000. *"Integration of Simulation and Multimedia in Automatically Generated Internet Courses"*. Computers and education in the 21st Century. (Ortega, M. and Bravo, J. eds.). Kluwer Academic Publishers. pp. 47-54.

Amazon web page: http://www.amazon.com

Brusilovsky, P., Eklund, J., and Schwarz, E. 1998. *Web-based education for all: A tool for developing adaptive courseware* . Computer Networks and ISDN Systems, 30 (1-7), 291-300.

Carro, R.M., Pulido, E., Rodríguez, P. 1999. *"Designing Adaptive Web-based Courses with TANGOW"*. In Advanced Research in Computers and Communications in Education. Eds: Cumming, G., Okamoto, T., Gómez, L. Vol 2, pp. 697-704. IOS Press. Amsterdam.

De Bra, P., Calvi, L. 1998. *Towards a Generic Adaptive Hypermedia System* . Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia, pp. 5-11.

De Bra, P. 1999. *Design Issues in Adaptive Hypermedia Application Development* . pp.: 29-39. 8th International Word Wide Web Conference. Toronto. In Internet at: http://wwwis.win.tue.nl/asum99/debra/debra.html

de Lara, J., Alfonseca, M. *Towards and authoring tool for the construction of simulation based web courses* . IEEE Multimedia, special issue on Web engineering. January-March. 2001. pp 42-49

Weber, G. & Specht, M. (1997*). User modeling and adaptive navigation support in WWW-based tutoring systems* . Proceedings of User Modeling '97 (pp.289-300).