Integration of Simulation tools for the construction of Educational Applications

Juan de Lara, Manuel Alfonseca
Dpto. Ingenieria Informática, E.T.S. de Informática
Universidad Autónoma de Madrid
28049 Madrid
{*Juan.Lara, Manuel.Alfonseca@ii.uam.es*}

Key words: Continuous Simulation, Web based Simulation, Multimedia enriched Simulation, Distributed Simulation, Course generation, Numerical resolution of partial differential equations.

1 Introduction

The currently most successful hypermedia system is the World Wide Web (*WWW*), wich has many advantages on traditional hypertext applications. This has brought around the current proliferation of educational courses in the *WWW*, which run from a simple transposition of lecture notes, to pages including more sophisticated elements, such as animated graphics, simulations and so forth.

We have been working for some time on the developement of advanced simulation tools, that simplify the generation of educational courses on the *WWW*. The language we are using is an extension of the old *CSMP* (Continuous System Modelling Program) language, sponsored by IBM [1]. We call the new language *OOCSMP* [2], for its main difference with *CSMP* is the addition of object-oriented constructs which make much easier the simulation of complex systems based on the mutual interaction of many similar agents (which can be modelled as collections of objects). We have used this language to build a course on Newton's gravitation and the solar system [3], a course on ecosystems [4], and a basic course on electronics [5].

2 The OOCSMP language

The OOCSMP language is a true extension of the old *CSMP* simulation language, in the sense that *CSMP* programs can still be compiled and executed by the *OOCSMP* compiler. The *OOCSMP* language include the following cappabilities :

1. Object-oriented constructs : classes, objects, methods, object collections, etc.

2. Solution of systems of bidimensional second order PDEs by the finite element method [6] and/or several schemes of the finite differences method [7]. Equations can depend on time. Different resolution methods can be combined when solving an equation. There are several primitives to define basic

domains. The domains have to be discretized by means of a mesh generator. In OOCSMP several mesh generators are available :

   (a) Isoparametric methods.
   (b) Delaunay triangulation.
   (c) Elliptic generation.

   An equation or system of equations has to be assigned to the resulting mesh, together with a resolution method. Meshes can be concatenated, to obtain more complex meshes.

3. An algebra of vectors and matrixes. The basic *CSMP* operations and blocks have been extended and specific blocks for vector and matrixes have been added.

4. Instructions to define different runs of the same model, all of them are accessibles from the same program.

5. Instructions to handle discrete events, similar to the *IF* ... *THEN* and *IF* ... *THEN* ... *ELSE* constructions of tradicional programming languages. When a discrete event occurs, and a variable that has to be integrated is modified, the integration method is reset, and the current value of this variable is set as the new "initial" condition. Same thing happens if a discontinuous block (such as *STEP*, *IMPULS*, etc) appears inside an integral.

6. Primitives for using and synchronizing multimedia elements.

7. Incorporation of external parameters to the simulation.

8. Instructions that make it possible the construction of distributed simulations.

9. New graphical output forms. The programmer can combine several in the same simulation. They include :

   (a) A graphical mesh generator (called *MGEN*) for the resolution of PDEs.
   (b) 2-D and 3-D animated graphics, plots for vectors.
   (c) Iconic plots.
   (d) Outputs to view the model equations graphically, in the form of blocks and connections.
   (e) Outputs to visualize the mesh nodes used to solve a *PDE*, and maps of isosurfaces.

10. Special comments instructions, fot the automatic generation of documentation files for the model, in *HTML* format.

11. Instructions to configure the appearance of the *HTML* page where the Java applet generated by the compiler will be placed. In this way the generation of web simulation courses is made easy.

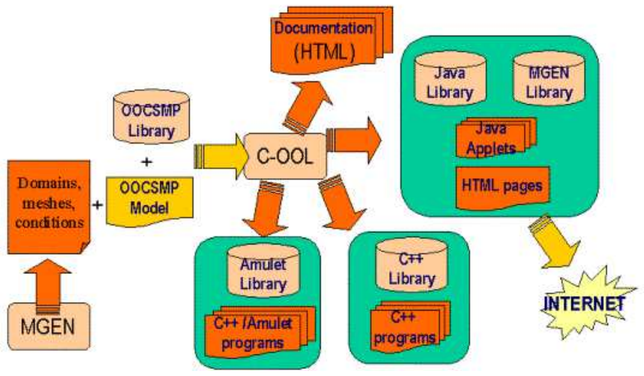

Figure 1: Scheme of the procedure.

3 The C-OOL compiler

Our compiler is called *C-OOL* (a **C**ompiler for the **OO**CSMP **L**anguage) and generates Java code for the Java Virtual Machine, or C++ code for *DOS*, *Unix* or *Windows 95*, using the *Amulet* [8] library, in the last two cases. Figure 1 shows the *C-OOL* architecture.

The compiler generates a user interface that allows the user to explore and interact with the problem, making easy to answer *"what if . . . ?"* questions. The user interface is configurable by means of compiler options and provides the following features :

1. Add or delete simulation objects during the simulation execution.
2. Move simulation objects between object collections.
3. Change or view the model variables or the object attributes during execution.
4. Change the domain geometry or the mesh properties when solving a *PDE*, if the tool *MGEN* is active.
5. Change between different simulation runs if available.

4 The courses generated

The courses we have generated with these tools can be found at :
http://www.ii.uam.es/˜jlara/investigacion

A procedure for the course generation has been designed and used to construct all the courses. It has siz steps :

1. Design the course on paper.
2. Build the necessary simulation models.

3. Adapt the main model to the course page.

4. Decide which outputs form will be used to display the results.

5. Include multimedia elements.

6. Add *OOCSMP* instructions to control the *HTML* appearance

5 Conclusions

The extensions added to the *CSMP* language turn it into a powerful and flexible simulation language, capable of solving PDEs, generate distributed simulations and synchronize the simulation execution with multimedia elements.

The set of procedures and tools we have developed makes it very easy to automatically build web courses based on simulation. The models used in the courses are automatically documented in form of *HTML* pages.

6 References

[1] 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.

[2] Alfonseca, M.; Pulido, E.; de Lara, J.; and Orosco, R. 1997. *"OOCSMP: An Object-Oriented Simulation Language"* . In Proceedings 9th European Simulation Symposium ESS97. SCS Int. Erlangen, 44-48.

[3] Alfonseca, M; de Lara, J.; and Pulido, E.; 1998. *"Semiautomatic Generation of Educational Courses in the Internet by Means of an Object-Oriented Continous Simulation Language"*. In Proceedings ESM'98. SCS Int, 547-551.

[4] Alfonseca, M; de Lara, J.; and Pulido, E. 1998. *"Educational Simulation of Complex Ecosystems in the World-Wide Web"*. In Proceedings ESS'98. SCS Int, pp. 248-252.

[5] Alfonseca, M; de Lara, J.; and Pulido, E. 1998. *"Generación Semiautomática de cursos Electrónica para Internet Mediante un Lenguaje de Simulacióon Contínua orientado a Objetos"*. (In spanish) III Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica TAEE'98, Madrid, pp 125-130.

[6] Zienkiewicz, O.C ; Taylor, R.L. 1989. *"The Finite Element Method"*, 4th edn, vol. I, McGraw-Hill; New York.

[7] Strikwerda, J.C. 1989. *"Finite difference schemes and partial differential equations"*. Chapman & Hall; New York.

[8] Myers, B.A. & al. 1997, *"The Amulet v3.0 Reference Manual"*, Carnegie Mellon University School of Computer Science. Technical Report no. CMU-CS-95-166-R2 and Human Computer Interaction Institute Technical Report CMU-HCII-95-102-R2.