13

Domain oriented modelling: a balance between simulation and programming

M. Cecilia C. Baranauskas
Osvaldo L. de Oliveira

State University of Campinas UNICAMP
Brazil

ABSTRACT

Simulation and modelling have both been defended in literature as rich computer based learning environments. Apart from the educational benefits each one offers these also have their drawbacks. This paper analyzes the gains and the shortcomings which might occur from simulation/modelling as an activity for learners and proposes 'domain oriented modelling' as a balance between both. Finally, prototype software in development is described in order to illustrate the proposed approach.

Main conference themes: artificial intelligence, software
Educational areas: secondary education, higher education
Study topics: design/technology, science/engineering
Secondary keywords: human computer interface, modelling, learning systems, problem solving, simulation

J. D. Tinsley et al. (eds.), *World Conference on Computers in Education VI*

INTRODUCTION

Modelling is a very common technique used to study the behaviour of many real systems. Modelling of either a real or a hypothetical dynamic system to observe/analyze its behaviour in time is a process with three main phases:

- the construction of a model which represents relevant aspects of the system being studied;
- experimentation and analysis of the created model;
- comparison of the constructed model with real systems.

We can term 'simulation' that part of the modelling process which involves basically the second phase of experimentation (execution of the model) and analysis of results. A model of the domain is embedded simulation systems and the user experiments in isolation from the domain with the modelled phenomenon by altering the input parameters of the model and observing/analyzing its output.

Simulation and modelling have both been defended in literature as rich computer based learning environments—deJong [1], Schank [2], Hassel [3], Millwood [4], Schecker [5] and Miller [6, 7]. Simulation systems mimic the behaviour of the phenomenon studied and the user's role is typically to 'discover' the rules which govern the phenomenon through some process of scientific investigation. Computational systems for modelling can constitute powerful learning environments as they engage the learner in the basic cycle of expression, evaluation and reflection within the considered domain. The demands of the computer for formal expression of a model force learners to define their knowledge of the subject in a more precise way. Furthermore, the execution of the model by the machine provides for an evaluation which can motivate learners to question the model, to re-evaluate their prior knowledge and re-express it, continuing the cycle of activities in the 'constructionist' (Papert [8]) style of learning.

Apart from the educational benefits these systems also have their drawbacks. The main criticisms of simulation centre on the lack of access users have to the underlying model which is considered a 'black box' (Goodyear [9]). Modelling therefore makes more demands on a user than simulation (Ross [10]). The aim of this paper is to discuss the benefits and drawbacks inherent in the use of simulation and modelling, and to propose domain oriented modelling as a balance between both. Finally, some prototype software in development is described.

FROM SIMULATION TO PROGRAMMING: BENEFITS AND DRAWBACKS

The discussion about learning environments for modelling/simulation must address two important interrelated factors: the learning effort demanded of students performing the task and the degree of learner control embedded in the system's design. Goodyear [9] suggests that there is a continuum of increasing learner control and increasing cognitive load from simulation to modelling by programming. He points out that the more autonomy a system provides for the learner, the greater the effort demanded of the user. For simulation/modelling based learning environments this effort/control continuum ranges from the use of 'pure' simulation to the use of general purpose programming languages for modelling (Fig. 1).

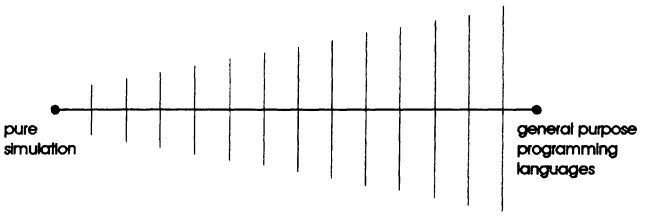

Fig. 1

At one end of the continuum the user of pure simulation is restricted to altering the input parameters of the model and observing its output. The random examination of the external behaviour of a 'black box' system can lead the user to the feeling of 'being lost' during the interaction. In recent years new advances in interface design and in theories of human computer interfaces have again raised interest in the design of simulation for use in educational contexts. The new approaches embed simulation in educationally oriented supportive environments (deJong [1], Baranauskas [11]) and put more control in the hands of the user.

At the other end of the continuum the users are enabled to create their own simulation by building a model to be executed by the computer. In the broadest sense the term modelling refers to the use of a formal language to represent some knowledge and the term model refers to the representation itself. Therefore general purpose programming languages can be viewed as computational environments for modelling.

Historically the first simulation/modelling systems were built using the general purpose language FORTRAN. Logo and SmallTalk brought the subject into an educational context. The problem with the use of programming languages is that these generally offer very little support for separation of the

phases which can be discerned in constructing and processing a model (Miller [7]). Furthermore the use of a programming language in modelling is seen by the user as giving instructions to the machine, rather than as a higher level description of a model for a specific domain.

The complexity of the modelling task is reduced by the use of general purpose modelling languages; these provide a variety of programming constructs which diminish the cognitive load associated with the task of programming.

On the one hand general purpose modelling languages give the user a higher level of control compared to simulation based environments; on the other hand, these make many more demands on a user. Ross [10] has noted that the phases of creating the model and testing it are specially demanding besides the difficulties inherent in both defining what is to be modelled and in the design of the model. To create the model students need to know the computer modelling language. Learning the programming techniques and features of the modelling language during the task of modelling increases the cognitive load. In testing the model students need to be able to judge the results. More importantly, "they need to be able to classify surprising results as the consequence of design flaw, implementation flaw or as not a flaw at all, but merely outside their present understanding of the task" (Ross [10], pp. 97-98).

In our view the problem with general purpose modelling languages is that the primitives are too general and have a level of granularity different from the level of conceptual primitives which should be used to describe the domain. Our proposal is to achieve a synthesis between the extremes mentioned, changing the emphasis from programming to the construction of models in a specific domain (Fig. 2).

By restricting the modelling environment to a specific domain we aim to achieve a balance between the benefits and difficulties found in the extremes of the continuum.

ENVIRONMENTS TO SUPPORT MODELLING

In a typical situation users of the proposed modelling environment build a model of the phenomenon they want to study, propose an experiment (based on initial hypotheses), 'run' the model and observe/analyse the result. Based on the feedback generated by the output users can make adjustments in the model and re-execute the model, continuing the cycle of activities.

Software for modelling has been traditionally directed to expert users with formal competency in mathematics (Millwood [4]). The proposal of modelling software for use in educational contexts must shift the focus of attention to the

aspects of user-system interaction which enables the learner to construct, test, and refine the model in a more concrete way.

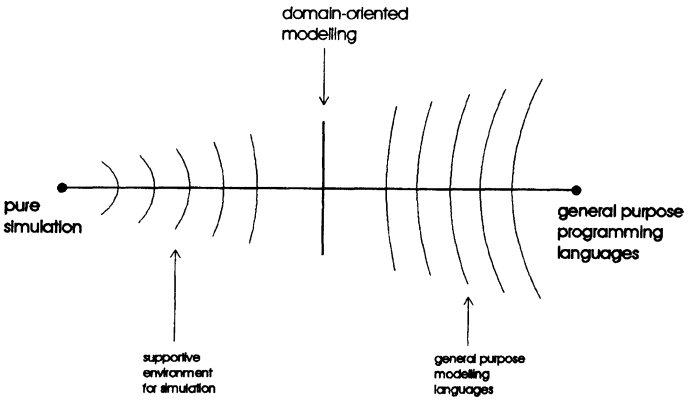

Fig. 2

One of the first computational systems for modelling used for educational purposes was the Dynamic Modelling System—DMS (Schecker [5]). In DMS the model is constructed through equations and numeric algorithms in a Basic-like language. Stella (Lindfield [12], Schecker [5]) is an impressive environment for exploring dynamic systems. Here models are composed of diagrams which are constructed out of primitives representing quantities accumulating over time, rate of change and functional relations. The model is built by direct manipulation of the primitives represented by icons. After running the model the system generates graphics of variables in time and the diagram which represents the model, can also be animated. Spreadsheets like Excel also have been mentioned in literature as environments for modelling (Millwood [4], Miller [7]). New environments are currently being proposed some of which are designed with younger users in mind: IQON, SMOD, QMOD (Miller [7]).

Basically these environments for creation and simulation of models should enable users to express the models they have in mind. Also the environment must enable the user to "perceive" and interpret the results of execution of the model. The question concerning the design of modelling environments for learning we must ask is: what features should the system have in order to facilitate learners to express, evaluate and increase their own ideas about the domain? To answer this question knowledge of the target domain is not

enough; in our view a more profound understanding of the user's actions with computational systems and within the domain is involved.

We argue that users interested in a specific domain should not be obliged to have programming knowledge in order to create and evaluate their models. Our aim is to provide to the user interested in Genetics, for example, a specific language for Genetics with which a model can be made.

DEVELOPING A PROTOTYPE

A domain oriented modelling environment is a computational system which has elements (direct manipulation interface elements, symbolic statements, etc.) which allow the user to build and execute models in that domain. By interacting with the elements of the environment the user is interacting with concepts of the underlying domain. Furthermore we can understand the activity of building a model as the activity of aggregating concepts to produce more complex structures.

'Newton' is a prototype of a computer based learning environment for modelling in the domain of Mechanics. This domain deals with the conditions of resting or moving bodies. In the system models are created through drawing of bodies, defining the mechanical behaviour of each body and describing aspects related to the interrelation between the represented bodies. Each body has standard properties which are automatically created with it, such as: name, position, angular coordinate, etc. These properties can be referenced when the user defines the behaviour of each body or when the user defines the interrelations between them. The mechanical behaviour of each body is defined in a diagrammatic language. The description of the dynamic relation between the bodies and the effects of the interaction between bodies is made through a symbolic minilanguage.

After running the created model the user can ask for graphics showing relations between the elements used in the definition of the bodies or the variation of these elements over time. The user can follow the execution of the model with instruments such as chronometers, rulers, etc. Also the system provides an animation of the mechanical behaviour of the bodies involved in the model.

Our design proposal is to restrict the modelling environment to a specific domain so that the model can be expressed in a 'language' which is 'transparent' and as close to the domain as possible. The debugging phase of the modelling process should also be facilitated through adequate mechanisms provided by the system.

CONCLUSION

We have described computational modelling as the activity of using the computer to express a model, with the subsequent aim of exploring possible consequences of the model and evaluating through a feedback not only the constructed model, but also the user's knowledge of the targeted system/phenomenon under focus.

Research has shown considerable evidence of the educational benefits of modelling based environments. The activity of modelling has been defended as a rich environment in which users can develop abilities in problem solving, decision making, presentation and communication of the understanding they have about a specific domain. However, literature has also addressed the difficulties the learner faces in mastering computational modelling environments. One of the main problems is the degree of learner control embedded in systems and the proportional learning effort demanded of the user (Goodyear [9]).

Our concern in this paper is to achieve a balance between the degree of control put in hands of the user and the effort of learning computational concepts demanded to build models. The prototype system designed is now being developed; through it we aim to investigate the idea and analyze principles for the design of domain oriented modelling in educational contexts.

REFERENCES

1. deJong, T. (1991) *Learning and Instruction With Computer Simulations.* Education & Computing, **6**, pp. 217-229.

2. Schank, R. C. (1990) *Teaching Architectures.* Technical Report #3. Northwestern University, August.

3. Hassel, D. J. and Webb, M. E. (1990) *Modus: The Integrated Modelling System.* Computers & Education, **15** (1-3) pp. 265- 270.

4. Millwood, R. and Stevens, M. (1990) What is The Modelling Curriculum? Computers & Education, **15** (1-3) pp. 249-254.

5. Schecker, H. (1993) *Learning Physics by Making Models.* Physics Education, **28**, pp. 102-106.

6. Miller, R., Brough, D. R. and Briggs, J. H. (1991) Declarative Software Tools for Learners. Proceedings of Sixth International PEG Conference. Rapallo, Italy, pp. 177-184.

7. Miller, R., Ogborn, J., Briggs, J., Brough, D., Bliss, J., Boohan, R., Mellar, H. and Brosnan, T. (1993) Educational Tools for Computational Modelling. Computers & Education, **21** (3) pp. 205-261.

8. Papert, S. (1986) *Constructionism: A New opportunity for Elementary Science Education.* A Proposal to the National Science Foundation, Massachusetts Institute of Technology. Media Laboratory, Epistemology and Learning Group, Cambridge, MA.

9. Goodyear, P. (1987) *A Toolkit Approach to Computer-Aided Systems Modelling.* In Lewis, R. & Tagg, E. (eds.), Trends In Computer Assisted Instruction. Blackwells, Oxford, pp. 88-96.

10. Ross, P. (1986) *Modelling as Method of Learning Physical Science and Mathematics. In* Weinstock, H. & Bork, A. (eds.), Designing Computer-Based Learning Materials. Nato ASI Series, Series F: Computers And Systems Sciences **23**. Springer Verlag, Berlin/Heidelberg, pp. 95-118.

11. Baranauskas, M.C.C. and Weller, D. (1994) *Extended Interfaces for Learning Environments Based on Simulation.* Accepted proposal for presentation at the ED-MEDIA/AACE, Vancouver, Canada.

12. LindField, G. R. (1992) *The Role of the STELLA SoftWare Package in Simulation and Modelling.* International Journal of Mathematical Education Science and Techonology, **23** (6) pp. 865- 880.