DECLARATIVE SOFTWARE TOOLS FOR LEARNERS

R.S.Miller, D.R.Brough and J.H.Briggs†

Department of Computing, Imperial College of Science, Technology and Medicine, University of London, 180, Queen's Gate, London SW7 2BZ, England.

†School of Information Systems, Kingston Polytechnic, Penrhyn Road, Kingston-Upon-Thames, Surrey KT1 2EE, England.

Accepted for presentation at the International Prolog Education Group Conference "Knowledge Based Environments for Teaching and Learning", Genova, Italy, 1991.

ABSTRACT

In this paper we argue the case for the development of general modelling software environments for use in education, where the emphasis is on provision of facilities for domain description rather than computational specification. We offer a categorisation of types of modelling useful for different areas of the school curriculum, and identify opportunities for new educational modelling tools in terms of this categorisation.

BACKGROUND

Modelling is increasingly becoming recognised as a bona-fide and even necessary type of learning activity in schools. The new National Curriculum in the U.K. specifies that modelling should be part of the experience of children across a wide range of the curriculum (DES, 1990) Modelling in this context can be defined as explicitly representing one's ideas about something in a formal way. Exactly what this means is discussed in the next section, but the pedagogical aim of this approach to learning is that explicit, unambiguous expression and testing of one's ideas may help the learner in understanding the material being modelled. It often leads to the refinement or change of these ideas.

Programmers are familiar with this syndrome - the acid test of a processing strategy or procedure is it's conversion from an idea into an algorithm which can be run on a computer. The subsequent process of debugging often leads to a refinement of ideas in the mind of the programmer. But it is possible to take the analogy between programming and modelling too far. Modelling is to do with describing the world, whereas programming is to do with instructing a computer.

These two different activities often become confused, especially when using "declarative languages" such as Prolog to program. Confusion arises because in a well structured program, it is often possible to re-interpret certain parts of the computer code as a model of the world - in computing terms programs (or parts of them) can be regarded either via their *declarative semantics* or via their *procedural semantics*.

This paper argues for software modelling tools which help to disambiguate the situation. We first describe a view of modelling in general. We then discuss how software tools might in principle support the activities of model construction and testing. Finally we discuss some different types of model, both in terms of their underlying views of the world and in terms of their paper or on-screen representation, and discuss possibilities for specific modelling tools based on these ideas.

DECLARATIVE MODELLING

A model in the most general sense can be defined as a collection of statements about the world (or an aspect or subset of it - called the *domain*). A model is *declarative* in that it is made from declarations or assertions which, for the purposes of the model, the modeller regards as true in the world. These statements are constructed from a pre-defined vocabulary and syntax who's expressive functions do not differ (at least in theory) from one modeller to the next. Such a definition encompasses a wide variety of entities. Examples are collections of mathematical relationships (equations or inequalities), the production rules of an expert system, causal loop diagrams or structured texts.

The correctness of a model can only be judged in terms of it's correspondence with the domain. At first sight this puts the modeller in somewhat of a dilemma. Since the model is only a description of the domain in the first place such judgements will be entirely subjective; according to the modeller it is necessarily a "good" model (unless he or she simply has a change of mind). However, in a formal modelling system the provision of a formal vocabulary usually goes hand in hand with a formal or mechanical method of generating new statements about the domain from the model. Such statements are extra to the model and can be regarded as *consequences* of that model. The correctness of the original model can be more objectively determined from the observed truth of falsity of these consequences.

For example, suppose that a mathematical model of the physical world includes the mathematical equation

force = mass x acceleration

Given particular values for *force* and *mass*, it is possible to use mathematical calculation to generate a value for *acceleration*. So one of the consequences of the model is that "when force is 20 and mass is 5, acceleration is 4". This statement can be tested for its validity in the real world by experimentation. It should be stressed that the processing method used to generate model consequences is separate from the modelling vocabulary in that the same model could be processed according to several different methods.

We can see that the process of model construction and testing as described above bears a close relation to what is commonly described as the "Scientific Method". However, the Scientific Method is somewhat more than this - for example it implies specific strategies for rigourous experimentation in order to compare model output with the real world. Moreover, Science traditionally uses the formal vocabulary and processing methods of mathematics for modelling. But modelling need not be via mathematics, and although formally generated model consequences are often mathematical (e.g. graphs or tables of numbers) they could equally be in the form of, for example, "advice" statements from an expert system.

Experts engaged in modelling may switch between the processes of model statement construction and model processing (i.e. consequence generation) very many times in the course of building and refining a model. Moreover, familiarity with both the formal vocabulary and its associated formal processing methods might make these switches largely subconscious and hence difficult to identify. Nevertheless, educationalists introducing modelling into the school curriculum need to be aware that this essentially investigative approach to learning involves the two separate stages described above, and that children need to be taught this principle explicitly.

COMPUTER SUPPORT FOR MODELLING

Computers can be useful tools for the modeller when engaged in either of the types of activity described above. As symbol manipulators, they are obvious candidates for use in model processing ("consequence" generation). But computers can provide powerful environments for model construction as well. In this section, we firstly discuss the desirable general characteristics of such authoring environments. We then discuss the possibility of facilities within modelling applications for processing of models.

Appropriate Primitives for Model Construction

A formal language consists of a set of symbols together with a set of rules for combining them. In the context of a modelling language, the symbols constitute the atomic *modelling primitives* from which models may be built. Such a vocabulary might consist of mathematical or logical symbols, or graphical entities such as nodes and connections between them. A computer modelling environment should clearly provide a supply of such building blocks for the modeller - in the same way that word-processors provide letters, spaces and carriage returns.

The important principle here is that the *granularity* of these building blocks should reflect the conceptual basis on which the models are to be constructed. To return to our analogy with word-processing; it would be pointless to provide a tool where it was necessary to draw four straight lines in order to add a capital 'M' to the document. The granularity of 'lines' is too small. It would be equally annoying if whenever the user typed 'M', the word 'Mother' appeared on the screen - the computer has made unwarranted presumptions about the user's intentions because the granularity of 'words' is too high. It is not always easy to decide of an appropriate granularity however. In the case of word processors it would be possible to argue a case for 'qu' appearing on the screen when the user typed 'q'.

Earlier we criticised programming languages as being inappropriate as modelling tools because they mix the procedural and declarative semantics of models. They may also be criticised because they generally provide primitives of too fine a granularity for modelling.

A way of addressing the issue of providing an appropriate granularity of modelling primitives is to provide *default* constructs which can be overridden. For example, it could be that the middle course to steer in our word-processing analogy is to have 'qu' appear on the screen, but allow the user to delete the 'u' subsequently if he or she so desires.

Structural Checking

Formal models are structures or collections of structures built from modelling primitives. Rules define what structures are allowed. Where a model is being constructed on a computer screen, an authoring package can either forbid illegal structures from being formed in the first place, or report the existence of such structures when asked. Examples of illegal structures might be mathematical expressions without proper bracketing, or directed graphs containing loops (in some system where loops implied an inconsistent model). The ideal tool should provide explanations of its prohibitions in terms of the intended semantic functions of its modelling primitives. For example, if the bracketing is incorrect or incomplete in a mathematical expression, it should explain that this leads to ambiguities in interpreting the intended precedence of mathematical operators.

Model Processing

Once a model has been created with the support of a modelling tool, the tool can provide automatic mechanisms for generating some consequences of the model. We stated earlier that the expressive functions of modelling primitives and syntax should be properly understood by the modeller. It is the tool's function to ensure that the methods of model processing it provides have a proper correspondence with these expressive functions of the model components.

For example the computational procedure of *resolution* used to generate a new logical statement from two others is generally held to properly correspond with the intended expressive function of *logical implication* used within logical models or theories. In this respect, a useful notion from the tool developer's point of view is the idea of an underlying *computational model* for each user-specified declarative model. The procedural semantics of this computational model should have a correspondence with the declarative semantics of the user's model.

Many modelling packages allow the user some control over the nature of the underlying computational procedure. For example, STELLA (Richmond, 1985), a System Dynamics Modelling package for the Apple Macintosh, allows the user to select from a menu of computational procedures (e.g. "Euler's Method" or "Runge Kutta") which all correspond to mathematical rate-of-change relationships. If such control is provided by a tool, it is important to separate "control" statements from model statements. Unfortunately many packages do not do this effectively. This mirrors a confusion between the two notions of model construction and model processing outlined above.

For example, STELLA mixes elements of computational vocabulary (such as "pause") with its vocabulary of modelling primitives. Worse still, it uses the "=" symbol as a computational variable assignment in some contexts and as mathematical equality in others. The net effect of this is that for some STELLA models it is difficult to see whether aspects of the model's output (in STELLA's case an on-screen simulation) are truly consequences of the model, or are due to direct intervention on the part of the modeller using computational commands such as "pause".

So far this paper has discussed modelling and modelling tools in very general and abstract terms. We next identify different types of models in order that we can outline more specific ideas about types of tools, and make some concrete proposals for new tools.

A CLASSIFICATION OF MODELS

There are two types of categorisation according to which models might be classified. One is a *semantic* categorisation corresponding to different ways in which a modeller might be regarding a domain. The other is a *syntactic* categorisation which corresponds to the different types of symbols or vocabulary being used to represent the modeller's ideas.

Static versus Dynamic Modelling

As regards semantic categorisation, perhaps the most easily identifiable distinction is between *dynamic* and *static* models; that is, between models which aim to describe how or why a domain changes in time, and those that aim to describe an unchanging state of affairs. Within dynamic modelling, time may either be modelled as a *continuous* variable, or via discrete *state*

transitions (Aris, 1978) (Roberts et al, 1983). The distinction between dynamic and static models runs orthogonal to most of the other classifications we shall mention.

Objects and Variables

Another fundamental decision for the prospective modeller is whether to regard the domain as consisting of *objects* or consisting of *variables*. In object-type models, individual objects are generally ascribed individual *properties*. In the case of dynamic models some of these properties might be described as the object's *behaviour* given certain *stimuli* or *messages*. Such a model might also contain *relations* which hold between two or more objects, either unconditionally or under certain global conditions. In variable-type models, variables may take different *values* which can correspond to properties of the whole domain as well as to properties of individuals within it. *Relations* may hold between the variables which correspond to relations between these global properties. Because model variables often correspond to amounts of a particular type of domain object, variable-type models are sometimes described as being at a different *level* than object-type models.

From Quantitative to Qualitative

Numbers and mathematical relations have traditionally played a major role in modelling, particularly in describing relations and values in variable-type models. Models utilising mathematics in this way are called *quantitative* models. Models who's variables may take non-numerical values and who's relationships are non-mathematical are said to be *qualitative*. The Tools For Exploratory Learning project (Bliss & Ogborn, 1988) has identified an important sub-category of qualitative models which can be labelled as *semi-quantitative*. Semi-quantitative models use relationships such as "is-less-than" and properties such as "increases" which imply an ordering on the sets of values that variables may take. Such values might typically be "large", "very large", "low", etc. Much of the work in A.I. on *qualitative reasoning* (de Kleer, J, 1985) (Kuipers, B, 1982) (Forbus, K, 1982) (Hayes, P, 1985) involves the development of semi-quantitative modelling vocabularies and associated model processing methods.

Causal versus Logical

Dependencies within models (between variable values, object behaviours, relations and properties) might either be regarded as *causal* or as *logical* (or *legal*). Where dependencies are grouped in disjunctive sets, a model can be described as *non-deterministic* (Miller & Brough, 1989). If there are no such disjunctions the model is *deterministic*. Linked to the notion of dependency are the notions of *domain boundary* and *independence*. The modeller must usually decide on a sub-set of the real world which he or she wishes to describe. In variable-based models the boundary between this sub-set (i.e. the domain) and the rest of the world is identified by the model's *independent variables*.

Model Representation

A syntactic classification of models concerns the manner in which the above concepts are represented, either on paper or on a computer screen. Probably the commonest representational formalism (for quantitative models) is *mathematics* - models are expressed as lists or collections of equations. The counterpart of this for qualitative modelling is *logic*, using the syntax of first order predicate calculus or some variation or sub-set of it. *Text* and

pictures can of course be used to express many ideas about a domain, but in the case of formal modelling some structure is usually imposed to enable model processing. *Diagrams* appear to provide a natural vocabulary for the expression of many types of model.

Work in Human-Computer Interface design has lead to the idea of *direct manipulation* environments, which in turn has encouraged the use of *diagrams* to express models. In particular, the use of *directed graphs* has become popular. Representational formalisms can consist of amalgams of these different types of syntax; for example STELLA uses a combination of directed graphs and mathematics to describe models.

We do not claim that the above classifications constitute an exhaustive set. However, the framework we have provided is useful when thinking about possibilities for new modelling tools. In the next section we offer some ideas in this respect.

NEW TOOLS - SOME OPPORTUNITIES AND IDEAS

It would be easy to utilise the above model classifications to construct specifications for modelling tools from an abstract viewpoint. One could for example make conjectures about the possibility for a dynamic, object-type, semi-quantitative, non-deterministic modelling tool with a text-based interface. However, useful specifications arise from practical needs within the curriculum as well as from theoretical considerations.

The authors have been exploring a number of different modelling approaches. As contributors to the Tools For Exploratory Learning Programme they have developed three pieces of modelling software as part of a project exploring the effects of modelling on children's reasoning. There is one tool for each of the qualitative, semi-quantitative and quantitative categories of modelling. All three tools have similar graphics-based, direct-manipulation interfaces. In this last section we focus on work in the semi-quantitative area to exemplify the issues discussed above.

The semi-quantitative tool (IQON) has been the main focus for development work. IQON allows the learner to construct simple models based on 'causal loop diagrams' (Roberts et al, 1983). In terms of the categorisation developed above these models are variable-type dynamic causal models. A typical model is shown in fig.1.

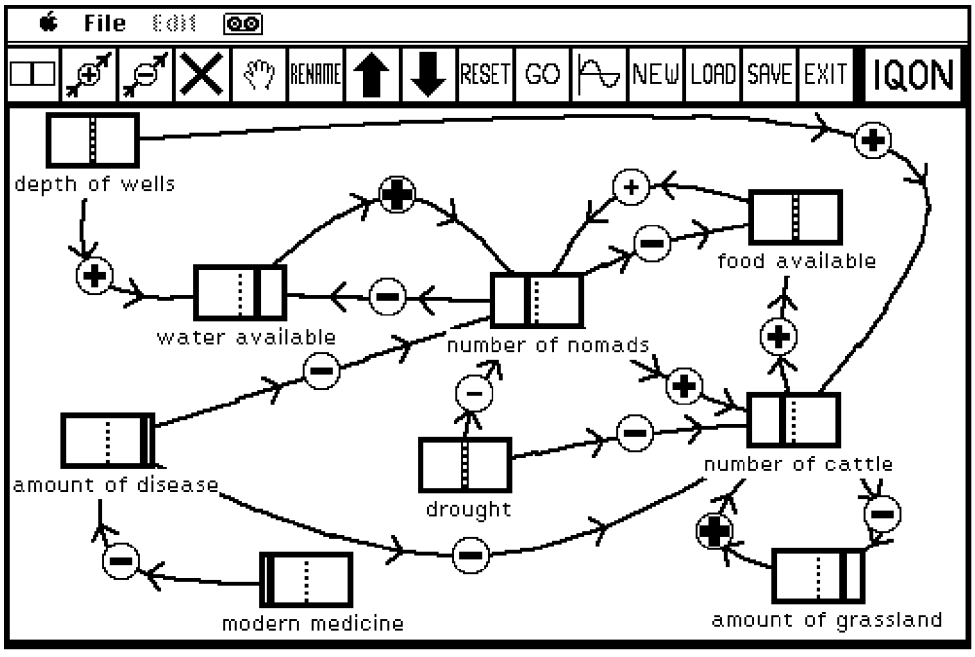

fig.1 An IQON model of the factors affecting nomadic populations in the Sahel Desert

Once a model has been constructed, the tool generates model consequences as an animation of variables' changes of values over time given an initial set of conditions (time is taken as a continuously varying parameter). To do this IQON constructs a default mathematical model, hidden from the user but based on the user's semi-quantitative ideas, which it processes using standard iterative techniques. These consequences are indicated to the modeller in terms of movement of the central bar within variable 'boxes' such as those in fig.1.

There are a number of ways in which the central ideas of IQON can be extended and modified. One is to take a state-transition approach to representing change (Miller & Brough, 1989). This fits with ideas discussed earlier from work in Qualitative Reasoning. Such an approach would eliminate the need for default quantitative model construction, and possibly improve the match between IQON models' declarative semantics and the procedural semantics of the tool's computation.

Other ideas for IQON variations include an IQON-like tool for static modelling, but with facilities for *dynamic input* and *output*. Part of the original motivation for this last idea is to test the extent to which children interpret IQON model output dynamically.

SUMMARY

We have outlined the *declarative* approach to modelling, in order to distinguish modelling from *programming*. We have described modelling as consisting of the separate activities of *model construction* and *model processing* (i.e. *consequence generation*). We have argued the need for computer tools which allow the clear separation of these activities. We have recommended that tool designers ensure that tools provide modelling primitives of proper *granularity*, that tools provide *default* modelling structures where appropriate, and that tools provide model processing whose *procedural semantics* have a close correspondence with the *declarative semantics* of the models.

We have given both a *semantic classification* and a *syntactic classification* of models. The semantic classification is in terms of *dynamic* and *static* models, *object-type* and *variable-type* models, *causal* and *logical* models, *quantitative*, *qualitative* and *semi-quantitative* models, and *deterministic* and *non-deterministic* models. We have made suggestions for several new educational modelling tools based on these classifications.

ACKNOWLEDGEMENTS

The Tools for Exploratory Learning Programme is funded by the ESRC and is a collaboration between researchers at Imperial College, Kingston Polytechnic, Kings College and The Institute of Education. The authors would like to thank their co-directors, Jon Ogborn and Joan Bliss and other members of the Programme for their contributions towards the ideas in this paper.

REFERENCES

Aris, R. (1978) *Mathematical Modelling Techniques* Pitman.

Bliss, J. & Ogborn, J. (1988) *Tools For Exploratory Learning* InTER Occasional Paper, InTER Publications, University of Lancaster

DES & Welsh Office, (1990) *Attainment Target 5: Information Technology Capability* in TECHNOLOGY in the National Curriculum, HMSO

Evans, J. (1986) *Structures of Discrete Event Simulation* Ellis Horwood: Chichester.

Forbus, K. D. (1982) *Qualitative Process Theory* Artificial Intelligence Laboratory AI Memo 664 MIT: Cambridge Mass.Hayes, P. J. (1985) *The Second Naïve Physics Manifesto* in Hobbs, J. R. & Moore, R. C. (eds) Formal Theories of the Commonsense World, Ablex: New Jersey.

de Kleer, J. & Brown, J. S. (1985) *A Qualitative Physics Based on Confluences* in Hobbs J. R., Moore R. C. (eds) Formal Theories of the Commonsense World, Ablex: New Jersey.

Kuipers, B. (1982) *Commonsense Reasoning About Causality: Deriving Behaviour From Structure* Tufts University Working Papers in Cognitive Science No. 18

Miller, R. & Brough, D. (1989) *A Logical Approach To Modelling Dynamic Systems* in Learning With Artificial Intelligence; Proceedings of P.E.G. Conference, Uppsala, Sweden

Miller, R., Ogborn, J., Turner, J., Briggs, J. & Brough, D. (1990) *Towards A Tool To Support Semi-Quantitative Modelling* in Proceedings of International Conference On Advanced Research On Computers In Education, Tokyo

Richmond, B., (1985) *The STELLA User Guide* High Performance Systems, Inc, New Hampshire, USA.

Roberts, N., Anderson, D., Deal, R., Garet, M. & Shaffer, W. (1983) *Introduction To Computer Simulation* Addison Wesley: New York.