Helping Inexperienced Users to Construct Simulation Programs: An Overview of the ECO Project

HELPING INEXPERIENCED USERS TO CONSTRUCT SIMULATION PROGRAMS: AN OVERVIEW OF THE ECO PROJECT

Dave Robertson
Alan Bundy
Mike Uschold
Bob Muetzelfeldt

DAI RESEARCH PAPER NO. 338

Paper accepted for ES-87 Conference,
Brighton, 1987.

Helping Inexperienced Users to Construct Simulation Programs: An Overview of the ECO Project

Dave Robertson †, Alan Bundy †, Mike Uschold †, Bob Muetzelfeldt ‡

†Department of Artificial Intelligence, University of Edinburgh.
‡Department of Forestry and Natural Resources, University of Edinburgh.

Abstract

We provide an overview of the development of ECO, a program which enables ecologists with minimal mathematical or computing skills to build simulation models. The first version of this system used a System Dynamics formalism to represent users' models and relied on simple interface techniques. Subsequent trials revealed that the formalism was insufficiently expressive to represent the sophisticated models which users sometimes required. The system was also over–reliant upon users to drive dialogue during model construction and provided insufficient guidance for inexperienced users. We discuss techniques for solving these problems. Finally, we note the key contributions of this research in the context of related work.

1 Introduction

Ecological researchers are becoming increasingly reliant upon mathematical models as a means of concisely representing their understanding of ecological systems. Having constructed a model of a given system, it is possible to test the validity of the representation using computer simulation and analysis of results. Models which are deemed valid may be used to predict the behaviour of their corresponding real world system when subjected to a specific set of conditions. This capability is particularly necessary in the assessment of environmental impact of resource management decisions.

Ideally, it should be possible for any ecologist to fit his/her description of an ecological system into a modelling framework which allows it to be easily accessed and analysed by other researchers. Currently, this is not possible for the following reasons :

1. Many ecologists do not have the mathematical or programming skills needed to construct ecological models.

2. There has been little standardisation of modelling approaches. Individual modellers tend to write large, one–off, representations using their favourite modelling language and/or mathematical framework. These models are extremely difficult to analyse unless one is familiar with the formalisms involved. Model defects are thus liable to pass unnoticed by the ecological community.

3. Model parameters and relationships are scattered through a wide range of literature and are expressed in different formalisms (*e.g.* mathematical formulae ; Fortran subroutines). Therefore, a large amount of effort is wasted in defining model components which have already been used elsewhere.

Ecologists need to be free to concentrate on investigating the dynamics of the systems which interest them, rather than wasting time learning esoteric programming techniques or deciphering obscure mathematical formalisms. They require an *Intelligent Front End* [Bundy 84], which will help them convert their ecological ideas into a simulation model. An Intelligent Front End is a kind of expert system which builds a formal description of a user's problem through a user–oriented dialogue. This task specification is then used to generate suitably coded instructions for the target computer package. Our research aim was to provide an ecological modelling system which could be used by ecologists with minimal mathematical or programming skills. In order to address the problems (listed above) of our target user group, we considered that the following features were required in the system:

- **A task specification formalism** which is capable of representing a wide range of ecological simulation models. This helps provide a standard representation for different models, tackling problem 2 in the list above.

- **A front end** which would interact with the user in terms familiar to him/her, converting the user's ecological statements into a mathematical formalism capable of translation into source code for a simulation model. The purpose of this dialogue control mechanism is to help overcome the technology barrier of problem 1 by making it easier for users formally to describe the program they require.

- An **automatic checker** of the consistency and ecological sense of the model. This also addresses problem 1 by preventing all syntactic and some semantic errors during the interactive specification phase.

- **A data base and browsing mechanism** for storing and accessing ecological data and relationships. By providing this repository of information, users should find it easier to isolate model structures appropriate to their application (alleviating problem 3).

- **A back end interpreter** to run the completed model and display the results.

The current ECO system, although prototypical, largely achieves these original requirements for a subset of ecological modelling. However, it also exhibits a number of deficiencies which we are attempting to remedy in our current research.

This paper contains a summary of the programs which we have constructed in order to provide the facilities listed above. We begin by describing our first prototype system and its relationship to our original objectives. We highlight some important inadequacies in the basic system and provide a short discussion of our attempts to alleviate these problems. We then summarise the benefits of this project – its contribution to artificial intelligence and ecological research.

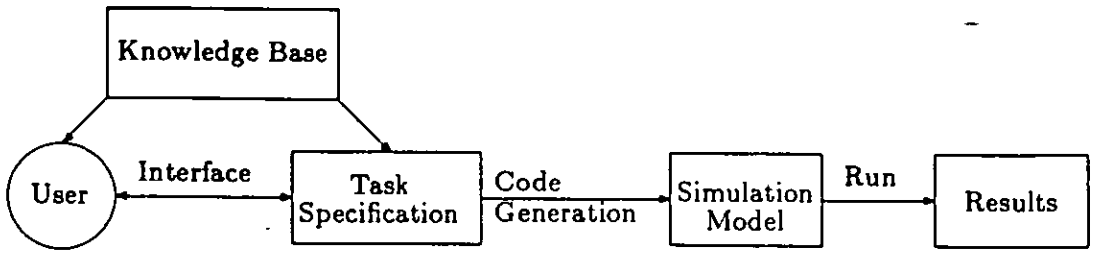

Figure 1: The original ECO system

2 Description of the First ECO System

ECO, ([Uschold et al 86], [Muetzelfeldt et al 85]) is a computer program – written in Prolog – for constructing ecological models. A diagram illustrating the general architecture of the system appears in figure 1. It relies upon a System Dynamics formalism [Forrester 61] to express model structure. This formalism can be manipulated by users, via an interface package, to produce a task specification for the model they require. A knowledge base of ecological relationships is used by the system to perform some simple checks for ecological consistency in the developing task specification. When complete (as determined by the syntactical structure of the formalism), the task specification is automatically converted into a target language (*e.g.* Fortran) and the simulation may then be executed. Recently, we have added the ability to run simulations directly in Prolog, our chosen implementation language, using a special purpose interpreter. This bypasses the code generation phase but does not effect the core of our research – the interface between user and formal task specification. We now consider the main components of the system, in relation to our original objectives from section 1.

2.1 The Task Specification Formalism

ECO can be used to build a special class of models, called System Dynamics models. This methodology encompasses the technique of compartment modelling, commonly used in ecology to model the flow of materials such as energy, nutrients, and pollutants. System Dynamics modelling makes use of a concise schematic representation which helps the ecologist think about the model without mathematical formulae. This representation was adapted and expanded to produce a task specification formalism which helps to bridge the gap between the user's view of the problem in ecological terms and the final Fortran simulation program. Each model is represented by an instance of this formalism which is built up while the user is interacting with the system. Figure 2 shows a diagrammatical representation of this formalism for a very simple model in which wolves are preying upon sheep. Predation is represented as a flow of some material (*e.g.* sheep biomass) from compartment *sheep* to compartment *wolf*, with the rate of flow as a function of the current values for *sheep* and *wolf* compartments and a *coefficient*. The initial values for *sheep* and *wolf* are set to 100 and 10 respectively.

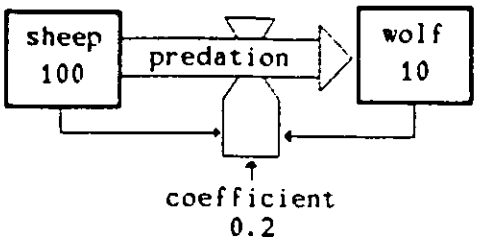

Figure 2: A System Dynamics Model

2.2 Interacting with the User

At the most general level, the ecologist user describes a model in terms of objects (such as trees, sheep, wolves) and relationships between these objects (such as predation, photosynthesis, etc). Equations and parameters defining these objects and relationships can be selected by the user, with automatic connection of appropriate structures in the underlying task specification. The user is free to decide how to approach the task of model construction. For example, submodels can be constructed separately and linked together later or, alternatively, the user can specify all the objects and relationships at the general level before finally attaching equations and parameters.

In order that ecologists should readily accept the system, it is crucially important to have a friendly means of interaction. Initially, users were required to input ecological statements in stylised English. For example, if a user inputs the statement "`wolves eat sheep`", this would be converted into a *predation* flow from a *sheep* to a *wolf* compartment (see figure 2). This allows the user to decide how the model will be constructed but requires that the user remember the syntax of each command. As a means of providing more guidance for users, an alternative menu based interaction system was implemented and, recently, computer graphics techniques are being tested as a more convenient way of eliciting input and displaying the developing model. This removes the necessity for remembering command syntax but provides no help with decisions about strategies for building the model (*e.g.* Should a sheep population be represented as a single entity or as separate individuals). Incorporating this sort of advice into the system will be tackled in future research. Currently, the user must make strategic decisions which are only checked for mathematical and simple ecological consistency by the system.

2.3 Consistency Checking

As the user is building the specification for his/her model, it is continually checked for internal consistency. Two separate types of consistency checking are performed. First, there is a syntactic or mathematical consistency associated with the formalism (*e.g.* destructive circularity should not occur in the task specification, a parameter must have an initial value). Since these consistency rules are few in number and clearly defined, we can ensure that ECO never produces a model which cannot be run – the user is guaranteed to get something that works. Secondly, there is semantic con-

sistency checking which helps maintain ecological sense in the specification. Ideally, we should like to guarantee that a final model will be ecologically sound and, furthermore, will accurately and appropriately describe the behaviour of the ecological system to meet the original goals of the user. This is well outside the capabilities of our current implementation but we do provide limited semantic checking capabilities. For example, if the user says that "`sheep eat wolves`" he/she is warned that this relationship may be the wrong way round. If the system does not recognise a particular object, it will make default assumptions on the basis of the context in which it appears. Thus if the user says "`foo eats sheep`", the system assumes that "`foo`" is a carnivore. All future uses of the object "`foo`" must be consistent with it being a carnivore. However, this rudimentary form of semantic checking is not always desirable, since a user may want to test non–standard ecological theories which are not recognised as valid by the system. A more comprehensive attempt to define specific objects and relationships in terms of general ecological principles is described briefly in section 3.1. This should facilitate improved checking and explanation capabilities.

2.4 Storing and Accessing Ecological Data

During the model building phase, the user has access to a base of ecological knowledge and data. Its primary function is to provide the user with the building blocks necessary for creating the model. This includes such things as ecological objects which may be contained in the models (*e.g.* animals, trees etc), taxonomic information relating classes of objects when possible (*e.g.* primates are mammals), mathematical relationships (with associated contexts indicating their appropriateness), and processes (*e.g.* grazing and evaporation) each with the appropriate types of objects which may participate (*e.g.* only animals may graze). Note that this knowledge is used to perform semantic consistency checking as described above.

Ecologists need the capability to store data from field observations or laboratory studies and retrieve them in a flexible, efficient manner. Often, these observations are made in different contexts and ecologists want to store and retrieve information according to the circumstances in which it was first recorded. For example, an observation may be made that "A tree in plot 5 of the Glentrool plantation was 5 metres high in summer 1976". Another observation may state that "The rate of photosynthesis of Sitka spruce is 10 $mgC\ kg^{-1}\ day^{-1}$ in bright sunlight". We have utilised relationships between items in different observations to provide a structure for browsing through observational records, progressively refining the user's description of the observation he/she wants to find. Ecologists who have used the system find the browsing mechanism easy to understand and operate. For a more detailed description of the ECO browser see [Robertson et al 85].

2.5 Running the Completed Model

Completed models can be passed to a code generation subsystem which translates the task specification into Fortran source code. Due to the constrained nature of our formalism for expressing models, this process was relatively straightforward. The user

can then compile this code, run it and revise the task specification if the program does not behave as expected. Currently, the onus is on the user to decide whether revisions to his/her task specification are necessary. Ideally, there would be a much closer association between the system for eliciting the model specification and the subsystem for running the model so that feedback on program execution can be related to the specification. As a first step towards integrating these systems a Prolog program for running simulations has been developed. This allows test simulations to be executed directly from the task specification (no intermediate translation phase) and provides for the possibility of automatically passing back information from the simulation to influence subsequent model refinement. Because our research effort is directed primarily at formally representing user's models rather than analysis of program execution, we have yet to concentrate on these more sophisticated execution issues.

3 Improving the Original ECO System

The system described in section 2 can construct a particular type of simulation model easily and efficiently, provided that the user knows what he/she wants to do. We tested this version of the system on undergraduate students of ecology and on various visitors to the department. These trials revealed several shortcomings of the original system. The most important of these are that the task specification formalism is insufficiently expressive; the system is too reliant upon the user to drive dialogue during model construction and the modelling guidance provided by the system is insufficient for naive users. We then diverted our attention to exploring ways to combat these difficult problems. Our current progress in each area is summarised below:

3.1 Extending the Task Specification Formalism

Although the System Dynamics formalism was useful for constructing a wide range of simulation models, it could not easily be adapted to represent certain more complex computational structures (*e.g.* models with age class subdivisions or models in which structural components were created and destroyed, perhaps representing births and deaths).

3.1.1 The Submodels Modelling System

A separate program (the Submodels system [Muetzelfeldt et al 87]) was developed to achieve a more flexible way of representing model structure. In this system, users are provided with a library of "base" models, each of which requires a fixed set of input data; generates a fixed set of output data; and performs some procedure in order to obtain output from input. Users may arrange base models hierarchically to represent subunits of the ecological system which they want to describe. Communication between models is achieved by connecting data–flow links between appropriate inputs and outputs. This method allows arbitrarily complex computational procedures to be

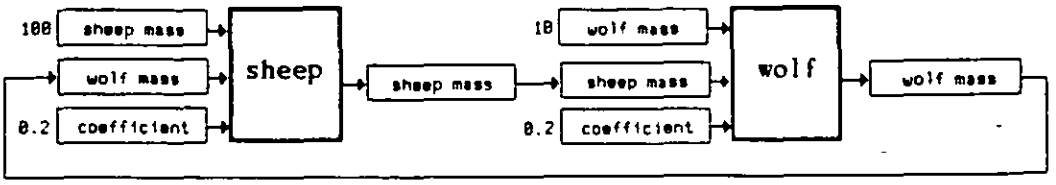

Figure 3: A Submodels Model

incorporated into the model but, like the System Dynamics formalism, places a heavy burden of responsibility on users, who must directly express their models in terms of the computation involved. Figure 3 shows a display, using Submodels symbols, of the System Dynamics model from figure 2.

3.1.2 Use of Typed Logic

Ideally, users should be able to state, in ecological terminology, the problem which their model has to solve and the system should help them convert this description into a computable solution. This raises the problem of how to represent formally these, often qualitative, "high level" statements of modelling problems and how to link these statements to a computable program.

We have performed initial experiments with a formalism in which common ecological statements are represented using a typed logic. Some examples of typical ecological statements expressed in the logic are:

"All wolves prey upon all sheep at all times."
$\forall W \in wolf \ \forall S \in sheep \ \forall T \in time \ \ predation(W, S, T)$

"If animal A preys upon animal B at any instant in time, there will be some probability distribution determining whether A kills B at that time"
$\forall A, B \in animal \ \forall T \in time \ \exists P \in probability\_distribution$
$predation(A, B, T) \rightarrow probability(\lambda T \ kill(A, B, T)) = P$

The procedural structure of the simulation is supplied by introducing fragments of simulation code (schemata), similar to those used in the Submodels system, each being active only under certain conditions of the user's description of the ecological system. This approach to program construction provides greater representational power along with increased ability to represent ecological statements in a form close to that employed by users. It also provides a foundation for future work on dialogue and guidance. A more detailed discussion of these issues appears in Robertson et al 87a].

3.2 Flexible Dialogue Control

Many computer systems (ECO included) tend to force users into a rigidly structured dialogue, designed to suit some "average" user. In the original ECO system, the dialogue was primarily user driven, with the system responding to the user's commands.

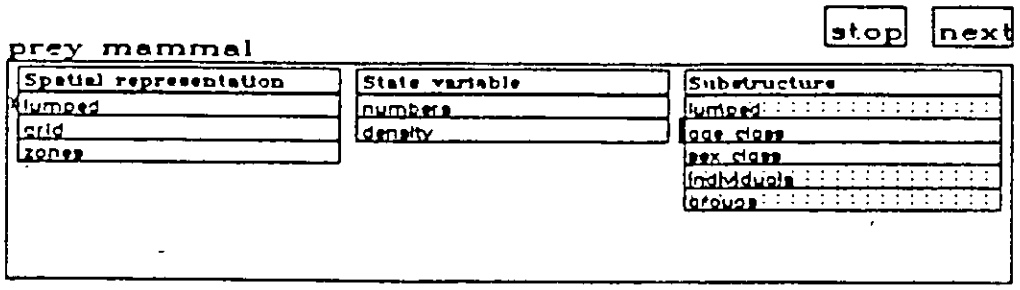

Figure 4: Menu System – Sample Display

As a means of exploring the other extreme of the range of possible dialogue mechanisms we have constructed several simple systems in which the computer plays a strongly active role in guiding the design. Principal among these is a system in which important characteristics of users' models are represented using frame–like structures, possessing attributes which users must instantiate to suit their required model. The system determines the sequence in which these frames are presented to users and suggests values for attributes. The users' role is simply to accept or reject the information offered by the system. A sample of the display produced by this system appears in figure 4. The user has been shown sets of options for three attributes of a prey mammal and has chosen an *age_class* substructure. In response to this choice, the system has excluded the options *lumped*, *individuals* and *groups* because they could not apply at the same time as *age_class*. However, *sex_class* option remains available, since mammals may have both age and sex classes simultaneously. The user may also select options from the *state_variable* or *spatial_representation* attributes. When all the required options have been selected, clicking the "next" button prompts the system to generate a new set of menus for related attributes.

In reality, different types of user require different balances between system and user initiative during model construction. Expert users want freedom to define task specification structure as they see fit. Novice users want to be guided through the model construction process until they become accustomed to the system. A flexible dialogue system is required, which allows users to take the initiative if they want to but continually provides advice as to what it thinks would be a useful move at any time. This suggested to us a dialogue architecture which utilises graphics displays and multiple windowing facilities to simultaneously display different possibilities for interaction. Among the options available to the user would be :

- A graphical display of the model which the user could manipulate by hand (direct user initiative). This approach is similar to that used in the existing ECO program.

- A window in which users may, of their own volition, provide information about

model structures and their goals for the current model. This has been implemented, based on a mechanism for selecting and editing typed logic statements, rendered into English text [Robertson et al 87a]. The left–hand window in figure 5 shows a sample display in which the user has, using a browsing system, selected a sentence (number 218) from the system's knowledge base. This sentence is represented internally as:

$$\forall A, B \in animal \ \forall T \in time \ \ predation(A, B, T)$$

but has been rendered into stylised English to make the logic more understandable to ecologists. The user has edited this sentence by restricting the type of A to $wolf$ and B to $sheep$, forming the expression:

$$\forall A \in wolf \ \forall B \in sheep \ \forall T \in time \ \ predation(A, B, T)$$

which has then been added to the problem description.

- A suggestion box of system advice about model construction. These suggestions are generated by the system, allowing an entire model to be constructed simply by following the system's advice. This part of the system has been only partially implemented (see section 3.3). A display from our current prototype appears in the right–hand window of figure 5. Here the system has used the expression added by the user (see above) in conjunction with the following rule from its knowledge base:

$$\forall A, B \in animal \ \forall T \in time \ \exists P \in probability\_distribution$$
$$predation(A, B, T) \rightarrow probability(\lambda T \ kill(A, B, T)) = P$$

to generate a suggested sentence, rendered into stylised English by the system but represented internally as:

$$\forall A \in wolf \ \forall B \in sheep \ \forall T \in time \ \exists P \in probability\_distribution$$
$$probability(\lambda T \ kill(A, B, T)) = P$$

By referring to the appropriate identification number, the user may get the system to implement this advice.

This architecture would allow smooth and flexible changes of initiative during the session. It also avoids the perennial problem of ordering the sequence suggestions because the user is allowed to choose which to accept at any time. Further discussion of dialogue issues may be found in [Robertson et al 87b].

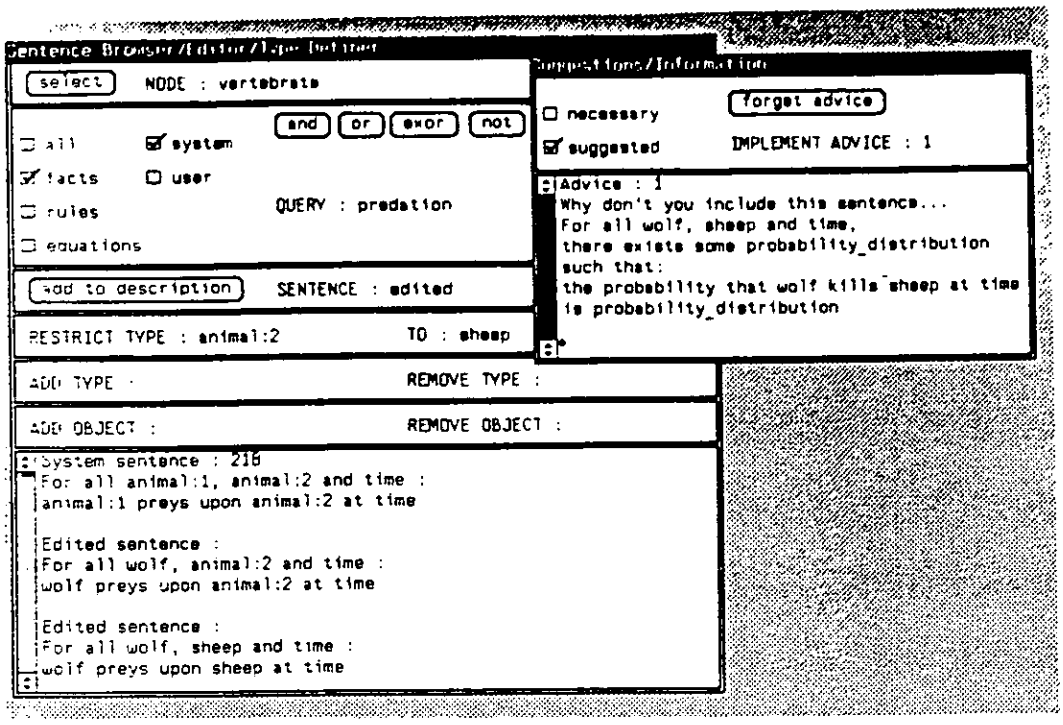

Figure 5: Mixed Initiative System – Sample Display

3.3 Guiding the Design of Specifications

Even when presented with a friendly dialogue system, many users have difficulty in constructing program specifications. This is because they have only a vague notion of what should be included in their model and how to represent it (a problem common to all non–trivial specification systems). For example, the user may be unable to decide whether to represent a sheep population as a single entity or as a number of individual objects and, if the latter option is chosen, he/she may not know the appropriate structures to insert into the task specification. Novice modellers do not know how to idealise the objects in models so that they are consistent with the overall objectives of the model. Without this information, they may construct inelegant specifications or, worse, may leave out crucial structures. The system must be able to advise users about the best structures for representing objects in the model, based on an analysis of existing model structure and a knowledge of the user's goals for the finished program. We have investigated possible methods of providing this form of guidance and hope to provide a working implementation, utilising a typed logic problem description (see section 3.1).

At the start of a session, advice may be provided by asking users to specify their modelling objectives or to provide some of the principal high level components of their model – for example, the fact that wolves prey on sheep. From this general description, the system may be able to select a modelling framework – a predator–prey schema, perhaps – and display this to the user as a suggested structure. If the structure is acceptable, it may be further elaborated, using additional schemata if necessary. For instance, a respiration subschema might be added to the predator (wolf) component of the predator–prey schema. This feature should fit cleanly into

the dialogue architecture mentioned above, allowing the user to obtain guidance in converting his/her initial vague ideas into a final formal specification and making sure that important parts of the specification are included. The resulting system would be an expert modelling consultant rather than merely a convenient tool. A discussion of the guidance requirements in ECO can be found in [Uschold 86].

4 Related Work

The ECO system synthesises Fortran programs from specifications in ecological terminology provided by the user. As a program synthesis system it occupies an important niche on the power/generality spectrum between general-purpose synthesis systems like NuPRL, [Constable et al 86], and the special-purpose, application generators [Horowitz et al 85]. ECO is restricted to the synthesis of a particular class of programs, but this is a much wider class than application generators can typically deal with. It exploits this restriction by synthesising more complex programs than those that can be dealt with by general-purpose systems.

An exciting aspect of our recent work using typed logic for specifying ecological problems, is that it is upwards compatible with the techniques used by the general-purpose synthesis systems. This gives the hope of a smooth transition between weak general-purpose synthesis systems and more powerful special-purpose systems employing domain specific knowledge. Our long range goal is to develop mechanisms for incorporating such domain specific knowledge in a general-purpose framework: to extract specifications from users, to guide the synthesis process and to interpret the results of the program.

ECO is an example of an intelligent front end package (*i.e.* a system which acts as an intermediary between a user and a complex program, making it easier for the user to use the program correctly). Previous work in our group concerning intelligent front ends has included the Mecho system, [Bundy et al 79], which built sets of equations for describing a mechanics problem stated in English, and the ASA system, [O'Keefe 82], which built instructions for a statistics package to analyse the results of a psychological experiment. These three systems have a strong family resemblance to the extent that we have suggested the possibility of a general intelligent front end framework or 'shell' to simplify the generation of similar systems, [Bundy 84].

5 Conclusion

The development of the ECO system can be divided into two phases. Our initial work relied upon a simple System Dynamics formalism which represented users' ecological models in a mathematical framework. Users were assumed to be capable of constructing *solutions* to their ecological problems by directly manipulating System Dynamics constructs. Our justification for this assumption was that ecologists were familiar with System Dynamics and that a large number of ecological problems could be easily represented in this formalism. However, tests of the initial system revealed

that the number of users who fitted into this classification was smaller than we had anticipated. Users sometimes required more complex models than could be represented using System Dynamics. They also wanted to describe their modelling *problem*, using terminology with which they were familiar, and receive guidance in converting this into a computable solution. This requirement provided the impetus for the second phase of development, which continues today. We have constructed prototype systems which allow users to describe their modelling problem using ecological statements – represented in a typed logic. Typed logic permits a much wider range of problems and solutions to be represented than was possible using System Dynamics. These statements can be used to isolate fragments of simulation code (represented in a formalism similar to that used in the Submodels system) which, together, constitute a computable simulation model. We are also designing guidance mechanisms, based on a "suggestion box" system. This will allow the system to take control of dialogue at a user's request, thus elevating ECO from the role of a passive assistant to that of an active participant in the modelling process.

Acknowledgements

This work was funded by SERC/Alvey grants GR/C/06226 and GR/D/44294. We are grateful to members of the Mathematical Reasoning Group in the Department of Artificial Intelligence at Edinburgh University for their practical advice and support during the course of this project.

References

[Bundy 84] A. Bundy. Intelligent front ends. In J. Fox, editor, *State of the Art Report on Expert Systems*, pages 15–24, Pergamon Infotech, 1984. also in proceedings of British Computer Society Specialist Group on Expert Systems 1984 and available from Edinburgh as DAI Research Paper 227.

[Bundy et al 79] A. Bundy, L. Byrd, G. Luger, C. Mellish, R. Milne, and M. Palmer. Solving mechanics problems using meta-level inference. In B.G. Buchanan, editor, *Proceedings of IJCAI-79*, pages 1017–1027, International Joint Conference on Artificial Intelligence, 1979. Reprinted in 'Expert Systems in the microelectronic age' ed. Michie, D., Edinburgh University Press, 1979. Also available from Edinburgh as DAI Research Paper No. 112.

[Constable et al 86] R.L. Constable, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Sasaki, and Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[Forrester 61] J. W. Forrester. *Industrial Dynamics.* MIT Press, 1961.

[Horowitz et al 85] E. Horowitz, A. Kemper, and Narasimhan. A survey of application generators. *IEEE Software*, January:40 – 54, 1985.

[Muetzelfeldt et al 85] R. Muetzelfeldt, M. Uschold, Bundy A., N. Harding, and Robertson D. An intelligent front end for ecological modelling. In *Working Conference on Artificial Intelligence in Simulation*, Flanders Technology International, University of Ghent, Belgium, 1985.

[Muetzelfeldt et al 87] R. Muetzelfeldt, D. Robertson, M. Uschold, and A. Bundy. Computer–aided construction of ecological simulation models. In *International Symposium on AI, Expert Systems and Languages in Modelling and Simulation*, Elsevier Science Publishers, Barcelona, Spain, 1987.

[O'Keefe 82] R. O'Keefe. *Automated Statistical Analysis.* Working Paper 104, Dept. of Artificial Intelligence, Edinburgh, 1982.

[Robertson et al 85] D. Robertson, R. Muetzelfeldt, D. Plummer, M. Uschold, and A Bundy. The eco browser. In *Expert Systems 85*, pages 143–156, British Computer Society Specialist Group on Expert Systems, Coventry, England, 1985.

[Robertson et al 87a] D. Robertson, A. Bundy, M. Uschold, and R. Muetzelfeldt. *Synthesis of Simulation Models from High Level Specifications.* Research Paper RP-313, DAI, 1987.

[Robertson et al 87b] D. Robertson, M. Uschold, A. Bundy, and R. Muetzelfeldt. Dialogue in eco: a system for building ecological simulation models. *in preparation*, 1987.

[Uschold 86] M. Uschold. *Computer-Aided Design of Program Specifications in the domain of Ecological Modelling.* Technical Report DP-35, DAI, 1986.

[Uschold et al 86] M. Uschold, N. Harding, R. Muetzelfeldt, and A. Bundy. An intelligent front end for ecological modelling. In T. O'Shea, editor, *Advances in Artificial Intelligence*, Elsevier Science Publishers, 1986. Also in Proceedings of ECAI-84, and available from Edinburgh University as Research Paper 223.