A Techniques-based Framework for Domain-specific Synthesis of Simulation Models

Alberto Nogueira de Castro Júnior

Ph.D.
University of Edinburgh
1999

Abstract

The formal specification community has produced many languages but few structured design methods. Those which exist tend to be abstract, providing little guidance in tackling problems in particular domains. One way of devising domain-specific design methods is by reconstructing an example in the domain using the target method; then generalising the design structures to cover a class of designs in the domain and finally building an environment in which these structures may more easily be re-applied to new problems. We demonstrate this approach using animal population dynamics models as the domain and Prolog techniques as the target method.

We have identified domain-specific techniques which use a parameterisation method from techniques editing but which contain information specific to the population dynamics domain; we define a problem description language which uses concepts from population dynamics; an interface which allows these concepts to be supplied; and provide an automated system which bridges between population dynamics problem description and the domain-specific techniques needed for model generation.

TeMS – Techniques-based Model Synthesiser, is the system constructed as the main instrument of our research. Because it is an embodiment of our views on the issues addressed, we submitted TeMS to user evaluation by ecological modelling experts, which produced material for a broad discussion of the system itself, its approach to modelling and its potential uses on the ecological modelling scenario.

Acknowledgements

Many people helped me to keep on the right track for finishing this work.

I am very grateful to my supervisor – Dave Robertson, for his support, help and wisdom much needed.

I would also like to thank my second supervisor – Robert Muetzelfelt, for his guidance in the ecological matters of my work.

Many thanks to my wife Goretti for her constant support, love, patience and encouragement without which I could not have done.

Thanks to my parents Alberto and Nazareth for their support, for teaching me to persevere and for being always sure (even when I had doubts).

I am most thankful to people here or back home that gave me strength even when they did not know it: Alane Castro, Mauro, Maurinho and Karen, Maria Macena and family, Paulo and Heloísa Salles, Maurílio and Beth, Marcelo Vieira and Edjard Mota.

My friends in Edinburgh made this period of my life a lot easier, sometimes as if we were a big family, I could not forget to thank Aderson Nascimento, João Cavalcanti, Líria Oliveira, Márcio, Ana, Pedro and Guilherme Brandão, Marco Aurélio, Débora e Felipe, Márcio Fernandes, José and Daniela Carbogim, Nancy Freeman, Teresa, Susan and Atalá Blackman, Patrícia Holanda, Juninho, Tiago, Virginia, Elias and Sofia Biris, May and Charles Wood and many more.

Many thanks to my friends at the Software Systems and Processes Group whom I had the opportunity to meet during all these years: Jessica Chen-Burger, Peter Funk, Renauld Lacouche, Steve Polyak.

Thanks to Wamberto Vasconcelos and Maria Vargas-Vera for their help in the early stages of this research and Birgit Jentsch for her invaluable insights on Social Research.

Many thanks also to the help of people who were kind enough to spend part of their time on the evaluation of TeMS.

I would also like to thank the University of Amazon, in Brazil, and my colleagues from the Computer Science Department back there, who were more than tolerant and backed my long stay in Edinburgh.

To the people at the Artificial Intelligence Department – staff and students – I am very grateful for all the help.

My special thanks to CAPES Foundation, Ministry of Education, Brazil, for their financial support much needed to carry out this project (grant n. 01723/93-8).

And thanks to God.

I dedicate this work to my wife Goretti and my parents, Alberto and Nazareth for their unconditional support and love

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Contents

List of Figures

Chapter 1

Introduction

Formalisation of programming knowledge has been a long-term topic of investigation in Computer Science. Programming techniques – common code patterns specific to a particular programming language – have been widely studied within the logic programming community, although never in a domain-specific context where the actors are not directly involved in the selection, parameterisation and application of techniques needed for program construction. In this chapter we give an overview of a project to investigate the exploitation of the generic method of techniques editing aimed at the design of a class of specifications in a certain domain (population dynamics modelling).

1.1 A few remarks on modelling

Modelling is not always an easy task and in some domains it is particularly difficult. In ecology, for example, a modeller has to deal with peculiarities which make it difficult to devise a model in a structured way. We list three of them:

1. the different levels of organisations – systems in ecology may be analysed at different levels. For example, a system can be analysed at individual or community level, depending on the specific interest of an ecologist. This may blur the focus of interest and the units of study.

2. the complex interactions between a system's constituents – this characteristic of

ecological systems makes it hard to isolate parts of the system and analyse them as separate units. This can be specially hindering for novice modellers, for whom it is much more difficult to capture functional groupings and to understand the web of interactions in a system.

3. uncontrolled sources of variation – unlike other sciences, in ecology it might be difficult to know the initial state of a system, or the actual influence of external factors. These could be detected statistically, but the spatial size and temporal time scales of ecological systems often make the replication of experiments impossible.

These are some of the aspects which makes ecology a difficult domain to apply structured scientific approaches, let alone less formal methodologies involving modelling. Because of its complexity and lack of more formalised (theoretical-based) frameworks, ecology is a typical case were classic modelling approaches have not been adequate.

Ideally, an ecological model should not only exhibit correct behaviour, but should also make explicit the criteria considered in its construction both so it can be evaluated and to augment experience in building other models. However, programs intended as simulation models are still usually seen as "black boxes" which exhibit certain external behaviour, whilst it is very difficult to understand the mechanisms inside.

The common practice still is to define and implement every model "from scratch" and modelling decisions and implementation aspects are mingled according to the modeller's expertise. That process is costly and especially obstructive to novices with little practice in modelling or programming. The lack of explicitness on strategical decisions is hindering to novices in many other areas. The teaching of medical diagnosis for example, has been considered to be "pervaded by the mystique of medical practice as an art, an art acquired by osmosis at the foot of the master" and as a result, diagnostic skills have to be invented from scratch by each neophyte[King 93].

The logic paradigm may provide a clear separation between the axioms defining the structures of a model and the inference methods used, and that could make it easier to evaluate, understand and emulate the modelling process.

1.2 Using structured formal specifications

There are a number of arguments, like the one in the previous paragraph, supporting the use of the logic paradigm as a notation to represent, or to assist the representation, of simulation models[Robertson *et al.* 91, Muetzelfeldt *et al.* 89, Muetzelfeldt 95, Mota 98]. In addition to this, it is interesting for the formal specifications community to apply their methodologies to problems from complex and poorly understood domains like ecology. Thus, if we use logic as a notation to represent simulation models, a starting point is to adopt a logic programming language. Prolog is the most widely accepted of such languages, and consequently would be a reasonable choice.

The construction of programs, which in the context considered here are implementations of simulation models, is a knowledge-intensive activity for which formal and semi-formal approaches have long been sought, the main motivation being to provide programmers with working templates that embody knowledge about programming. These templates would allow the knowledge of standard solutions to be represented and applied to new problems, accelerating program design without compromising desired qualities (reliability, correctness, etc.).

In [Robertson *et al.* 91], Robertson and others described a project that introduced logic-based approaches to tackle knowledge representation in the domain of ecology. One of the results of that project was a tool to generate a model using instantiable pieces of Prolog code called *program schemata.* Program design using schemata is relatively easy, allowing a high-level definition, but is also limited in flexibility.

Another sort of structure, known as *Prolog Programming Techniques*, represents standard patterns for constructing individual predicates [Brna *et al.* 91]. *Techniques editing* is an established general method for addressing program design at a more fine grained level and earlier research has dealt with different tasks within the programming domain[Bowles *et al.* 94, Robertson 91, Vasconcelos 93, Vargas-Vera *et al.* 93].

However, there have been no experiments in tailoring the general methods of techniques editing to the specific demands of a domain of application other than Prolog itself. This thesis describes our attempt to do this by investigating how domain-dependent structures can de *defined, organised* and *used* in a context where knowledge on programming

or hard knowledge of modelling, cannot be assumed.

1.3 A question of control

Because programming techniques are essentially a programmer's tool, their use has generally required a considerable interaction by a user with good understanding of the program transformations being applied[Vargas-Vera 95]. There has been no research where the use of programming techniques is dissociated from the actor controlling program synthesis.

A fundamental difficulty in our target domain is that the styles of description which population modellers understand are different from those needed to control specification synthesis using programming techniques. The work described here shows a way in which the selection and parameterisation of techniques are connected to contextual knowledge supplied by a user with no knowledge of the program transformations that take place during generation.

This distance from the intricacies of program generation is important in our target domain. It is important that modelling systems are seen to be "actually modelling" rather than simply running through data-structures in a purely computational way.

1.4 Aims of this research

The following is the main statement for the research reported here, which makes explicit its primary goal:

> To investigate the use of a generic method for structuring formal specifications (programming techniques), tailoring it to the design of a class of specifications in a target domain (population dynamics modelling).

1.4.1 Research questions

From the preceding statement a myriad of questions might be derived. The following is only a subset of that collection which has guided the development stages of this project.

The central questions are:

- Is the main idea **feasible?** Are domain-dependent programming techniques up to the job specified here?
- If so, how can it be **validated?**
- In the event of the preceding questions having affirmative answers, there is still the issue of how it could fit in the population dynamics modelling context, that implies in answering the question: Is it **adequate?**

The following are more specific questions, which add more detail to the central issues.

- questions about the domain-specific structures to be used:

  What are the structures for this domain-dependent approach?
  How are these structures defined?
  How should they be organised?
  To what extent are they different from generic techniques?

- questions about techniques editing and domain-specific structures:

  Would techniques editing cope with these domain-dependent structures?
  What sort of changes are needed for generic techniques editing undertaking this task?

- questions on bridging knowledge about a problem and synthesis of specifications:

  How can program synthesis be controlled without user interference?
  Would that affect the target domain?

- questions on implementing the ideas developed in the project:

How should the concepts from the project be implemented in a way that would make sense for users from the target domain?

What are the requirements of those users?

What is the suitable way to verify and validate the implemented solution?

- questions on evaluation of the synthesiser:

  Would the synthesiser contribute to some extent to:

  - clearer, more structured, standardised models;
  - a structured, top-downl approach to modelling;
  - less time spent in building models (especially for novice modellers);

  What should be the design for the evaluation experiment?

1.4.2 Goals

The following are the goals of the project, stipulated to address the preceding questions.

- to define a library of domain-dependent structures for program design in animal population dynamics modelling;
- to devise ways in which those structures can be used within the general methodology of techniques editing;
- to design a program generation system in which Prolog predicates are built by that domain-specific techniques editing;
- to use problem-specific knowledge supplied by a user, to control program generation with no need for further user intervention;
- to combine the previous elements in a modelling environment which could lead a user through well defined design steps, then automatically generate a Prolog implementation of the model specified, which could then be tested;
- to carry out a consistent evaluation experiment of the modelling environment, assessing the concepts proposed, the resulting tool and giving a better insight on

how people actually do modelling and how the environment would fit into that scenario.

We believe accomplishing these goals will enlighten not only those directly involved with the use of formal specifications in program design, but also those interested in ecological modelling, since we propose a starting-point to modelling that might support novices in this unsystematised activity. See Section 7.2 for a list of the contributions of this work.

1.5 Structure of the thesis

The remainder of the thesis is structured as follows:

On **Chapter 3** we look at the two areas involved in this research. We start by discussing the two areas in a higher level, briefly discussing approaches to programming and modelling from different yet connected areas in AI. We then narrow the focus of our discussion towards the object of our work by surveying existing methodologies for synthesis of logic programs and analysing the contexts in which they are currently used. Then we discuss some idiosyncrasies of ecology which make it an interesting domain for development and use of Artificial Intelligence methodologies, and also list the main current paradigms for ecological modelling. Finally, we present one approach in which modelling takes advantage of both areas and posit what benefits can be expected.

Chapter 4 addresses the domain-specific data structures used for synthesis of models in a sub-domain of ecology. It is introduced by a discussion on the relationship between knowledge about the domain and the definition and use of data structures for synthesis. It describes a way of defining domain-specific data structures by reconstructing a model from the ecological modelling literature, then extracting structures from it and generalising these structures to a class of models. An initial library of data structures built in this way is listed and explained in this chapter.

Chapter 5 describes TeMS (Techniques-based Model Synthesiser), a modelling environment which uses the structures from Chapter 3 to automatically produce population dynamics models. The chapter starts with a description of TeMS' process-

centred 3-phase modelling approach, which connects a problem description language to the techniques used in model generation. This is followed by a description of the main functional modules in TeMS. Implementation of the tool is then discussed and we demonstrate how the system operates at each stage of the 3-phase modelling approach. In the last part of the chapter, we address validation and limitations of TeMS.

Chapter 6 reports on the rationale, methods and results of the evaluation of TeMS by expert modellers. This evaluation is a piece of *in-depth* qualitative research, aimed to give us a greater appreciation of how the ideas embodied in the system would be seen by modellers and to get an overall picture of how the system performs with respect to what was proposed. The experiment spans formative and summative evaluation and gathers anecdotal accounts of the circumstances under which the system and the ideas in it do well or badly. Two instruments of data-collection – interviews and questionnaires, were used, and the discussion of the results also includes an analysis of how the participants see the modelling activity.

Chapter 7 presents our conclusions and the contributions of the work for the synthesis of logic programs. We sum up the project, examing its achievements in the light of what was brought together by the evaluation and discuss further research on the topic.

Background data for the thesis are included in the Appendices. **Appendix A** contains algorithms for the techniques editor, for the generation module and for the meta-interpreter, all discussed in Chapter 5. **Appendix B** is a listing of the knowledge base for the current implementation of our system, explained on Section 5.2.4. **Appendix C** has the texts used in the evaluation (Chapter 6) – Section C.1 is the introductory text given to participants prior to the experiment; Section C.2 is the text for the semi-structured interview; and Section C.3 is the questionnaire used. **Appendix D** contains results from the interviews – Section D.1 is a table summing-up participants' answers to key questions and Section D.2 contains the transcripts of the interviews.

Chapter 2

Context

This chapter introduces the context in which the ideas discussed in the next chapters were amalgamated in a system (TeMS) and put to use. It gives an indication of what it would feel like for an ecologist to build a simulation model using the framework proposed here.

2.1 A typical example

We illustrate the construction of a standard model through the use of the following scenario, which is the same used by all participants of the evaluation discussed in Chapter 6.

> "Warrawong Sanctuary was the first of the "Earth Sanctuaries", a conservation project carried out in Australia by Dr. John Wamsley since 1969. The goal of the project is to reintroduce plants and animals existing in Australia two hundred years ago (before European settling). All of the land was made free of some predators and it is surrounded by a fox and cat proof fence.
>
> Probably the most remarkable achievement of Warrawong Sanctuary is the breeding of platypus (Ornithorhynchus Anatinusacropus), which had never been done before. We want to have an idea of how platypus population would cope with an accidental invasion of foxes into the reserve.

The current platypus population is 500 from which 95 are young (up to 12 months), 275 are adult (up to 24 months) and 130 are old (more than 24 months). Consider that the maximum individual reproductive rate of platypus is of 0.25 and that they give birth every four months and live no longer than 3 years. If a pregnant fox managed to stay within Warrawong reserve, two foxes would prey on platypus at an individual rate as shown in Figure 2.1. Assume an individual mortality rate of 0.012 for platypus due to a combination of factors (*e.g.* diseases).

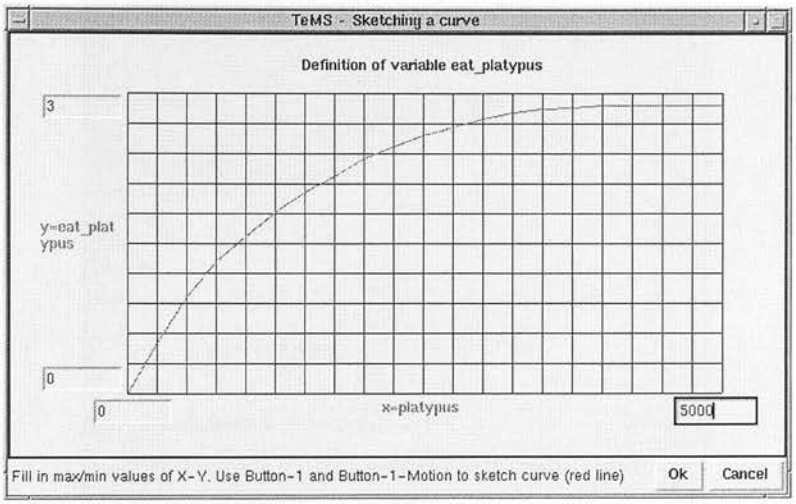

Figure 2.1: Scenario information – eating rate of one fox.

The question we want to answer is: maintaining these conditions, how would platypus population be in 25 years from now?

2.2 Using TeMS to build a model

We now show how a modeller would use TeMS to compose a simulation model from the scenario described in the preceding section.

When the user starts TeMS, a couple of introductory windows are presented to her, who then starts the definition of the model by pressing the button "Describe" on the top of the main window and then by choosing the first (and at this point only) option - "Components". TeMS then prompts the user for the *components* which exist in the

model, that is, what species will form the population modelled. In our case, *platypus* and *fox* are the only two species considered. TeMS has information about some species (e.g. fox) used by context rules in the knowledge base.

Figure 2.2 shows the point at which the user is including a new component (platypus) in the model. Since platypus is not part of the initial set, the system asks the user to choose sort names to be associated with the new component.

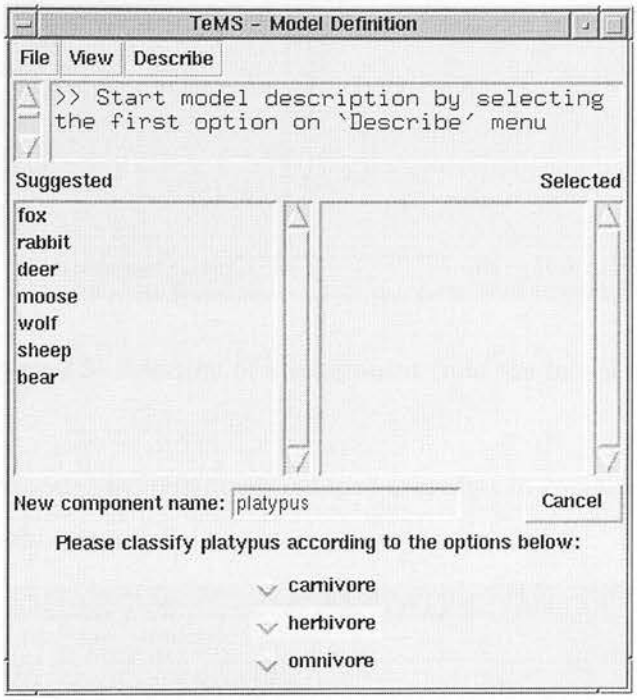

Figure 2.2: Inclusion of a new component in the model. A user's choice of sort label is highlighted.

Figure 2.3 shows the next state of the system, where *fox* has just been selected. It can be seen on the right-hand-side of the middle area that the new species (platypus) is displayed along with the sort-values associated with it, while fox is represented simply by its name, since all the other information is already in the knowledge base.

The next decision point in the definition of our model indicates how the components will be referred to. The two basic choices are to deal only with individuals or to deal

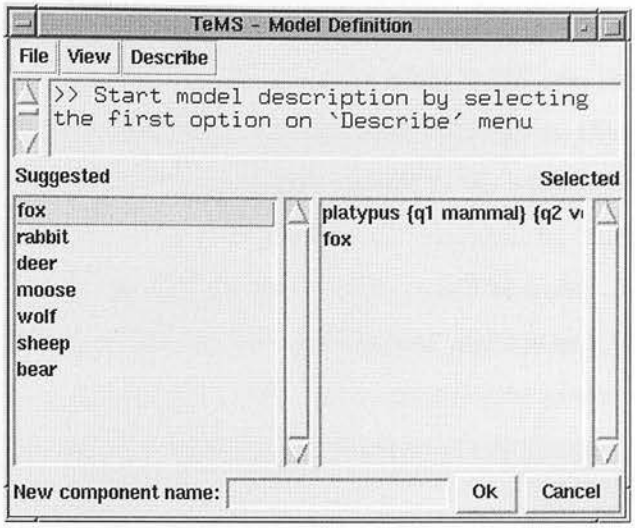

Figure 2.3: Selection of a component from the initial set.

with a population formed by individuals grouped according to common characteristics[1]. Figure 2.4 shows that situation.

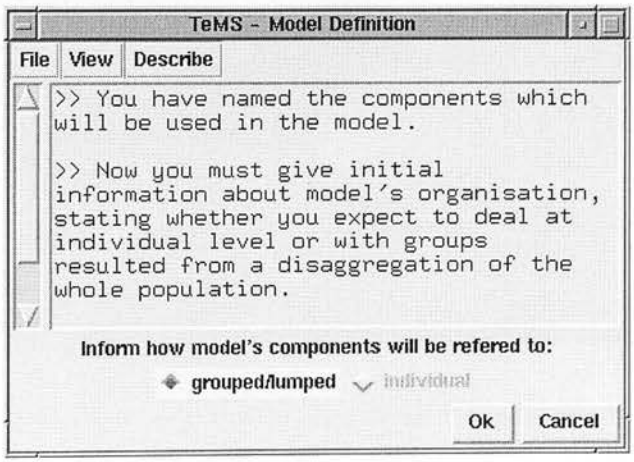

Figure 2.4: Design decision: unit of reference for the population.

Each component in the model may have its parcel of the population grouped in a different way. The population of a component can be disaggregated according to various *dimensions of classifications*, which must be defined by the user.

[1] Currently, TeMS operates only at group level

In the model used here, the platypus population is disaggregated in *age classes*, namely *young, adult* and *old.* Figure 2.5 shows the point when TeMS asks the user the number of individuals (initial population) at each age class of platypus, shown on the window in front. In the window behind that one, we can see in the left-hand-side of the middle area, a list of possible dimensions of classification suggested by TeMS, we see the one selected with the values given by the user (young, adult and old). Note that the user can select more than one dimension of classification, and for each subgroup resulting from the combination of them, an initial population must be given.

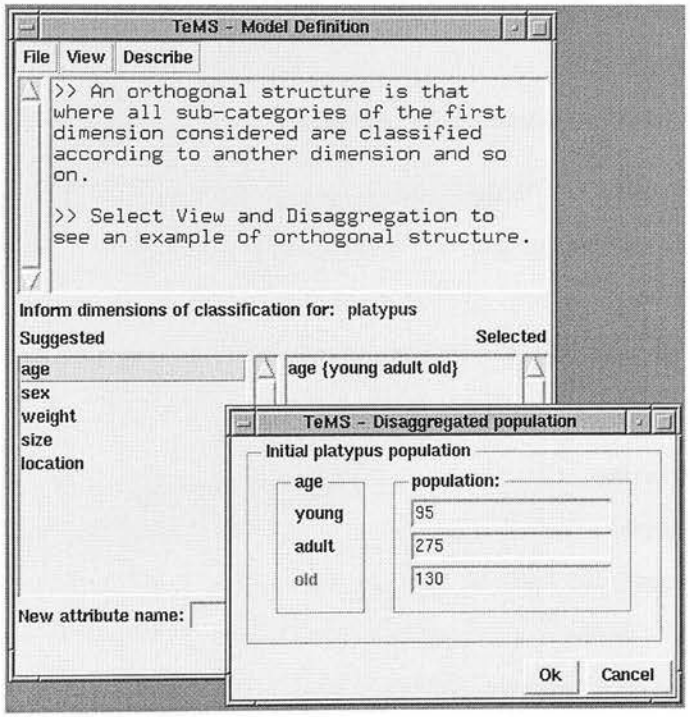

Figure 2.5: Initial values for platypus population (disaggregated in age classes).

When a population is not disaggregated at all (the case of fox population in our model), the user simply press the "Ok" button at the "dimensions-of-classification" window to go ahead. The system will show a warning message and ask for a confirmation, then it will wait for the user to type the initial fox population, as shown on Figure 2.6.

When all initial populations have been given, TeMS prompts the user for a confirmation that time-steps will be the basic time-units used throughout the simulation after the

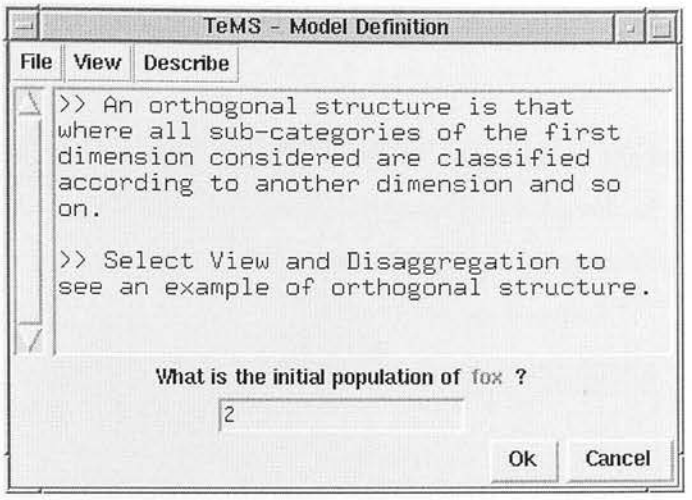

Figure 2.6: Initial values for fox population (non-disaggregated).

model be completed[2]. After that, the definition of processes can start.

Each component has associated with it, a set of ecological processes affecting its population (increasing, decreasing or rearranging it). A process will belong to one of the main groups suggested by TeMS, in our case: natality, mortality or progress on category. On the situation shown on Figure 2.7, the user is informing how *reproduction*, which belongs to the group *natality*, will be represented in the model.

According to the way a population is disaggregated and to the context rules previously stored in the knowledge-base, TeMS presents a list with different instances of processes belonging to the main group selected. Alternatively, the user may create a new process definition with customised or non-standard equations. In our case, a standard way of computing the number of births (using an equation that produces logistic growth of the population), called *logistic_rep_rate*, was selected by the user. As we can see on Figure 2.7, a text giving a short description of the highlighted option is shown beside the list of suggested processes.

Finally, there are areas in the window to indicate the set of time points (P) from a reference period (T) in which the process will be active. That is, TeMS allows the user to pinpoint which ones, within a period of N time-steps, will "trigger" a certain

[2] Other time-structures can be previouslly defined on the knowledge base.

process. Thus, to indicate that the process will be always active, the user should choose $P = \{1\}$ and $T = 1$. If each time-step corresponds to a month of the year and we want to represent that a certain process will be active on January, May and September every year, then $P = \{1, 5, 9\}$ and $T = 12$, as shown on Figure 2.7.

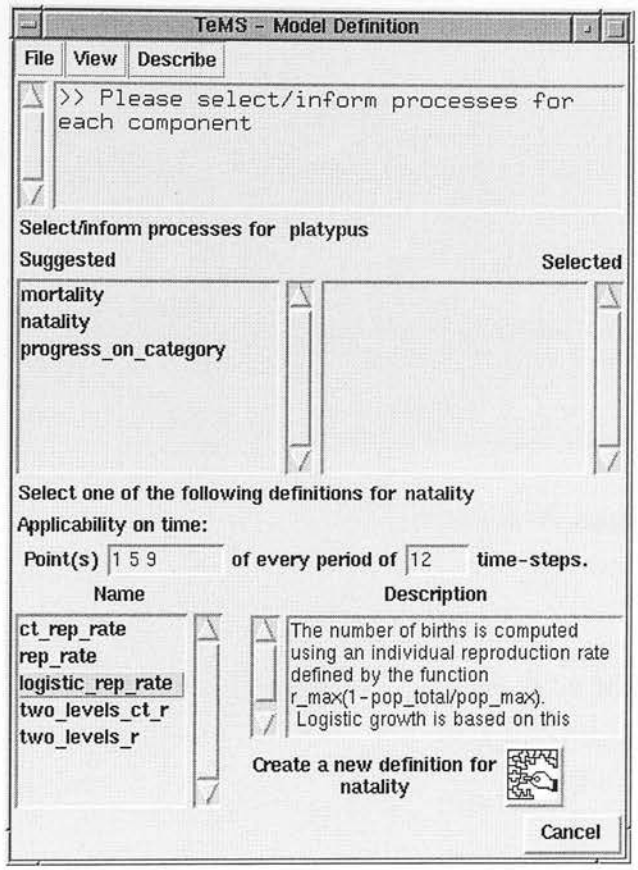

Figure 2.7: Reproduction of platypus is represented in the model by using a standard equation known to produce a logistic growth of the population. The process will be active on the 1st, 5th and 9th points of each period of 12 time-steps.

When a predefined process is used, TeMS will ask the user the values for the variables in the equation. A variable is either a number, an arithmetic expression (which may include other variables), an if-then-else declaration or a two-dimensional graphic function. Figure 2.8 shows the window where the user gives the value for the first variable in the equation of the process "logistic_rep_rate".

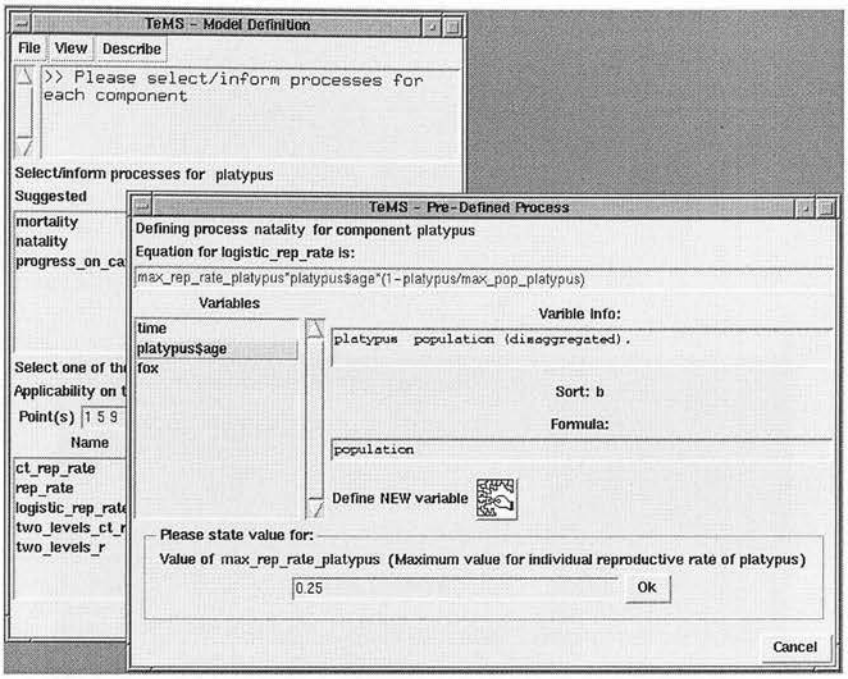

Figure 2.8: TeMS asks the values of all unknown variables in the equation which defines a process.

After defining natality, there are two instances of *mortality* to be defined in our model: The first is due to a combination of factors and is represented by a rate of 0.012, as shown on Figure 2.9.

The other mortality process is due to *predation* and is based on a variable (`eat_platypus`) which is a function of platypus population. An equation for that function is not previouslly known, the user then indicates that the relation between the value of the variable `eat_platypus` and the platypus population will be defined by sketching a graphical function through the declaration `graph(platypus)` as shown on Figure 2.10. The Figure 2.1 is the actual window in which the user can sketch the curve.

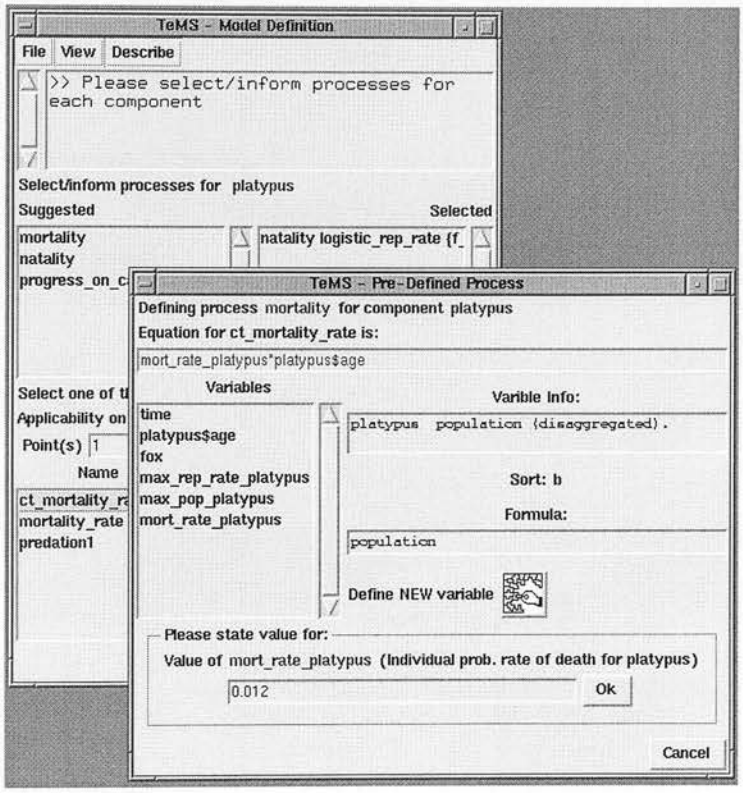

Figure 2.9: Definition of a mortality process for platypus population. In this case, a constant mortality rate of 0.012 is used to represent death due to a combination of factors.

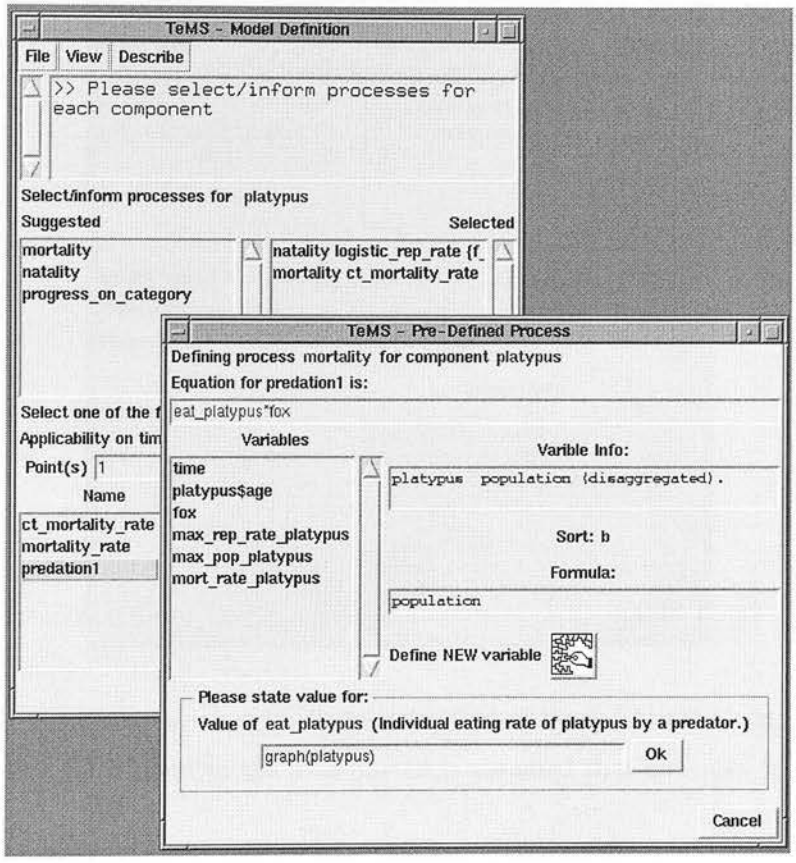

Figure 2.10: *Predation* is another process causing mortality on platypus population. The user chooses to define the variable needed in the equation (`eat_platypus`) by sketching a graph (shown on Figure 2.1).

The final process referring to platypus population is *ageing*, which happens at each period of 12 months, when the individuals of an age-class progress to the next one, i.e. youngsters become adults, adults become old and old platypuses die. The definition of this process is shown by Figure 2.11.

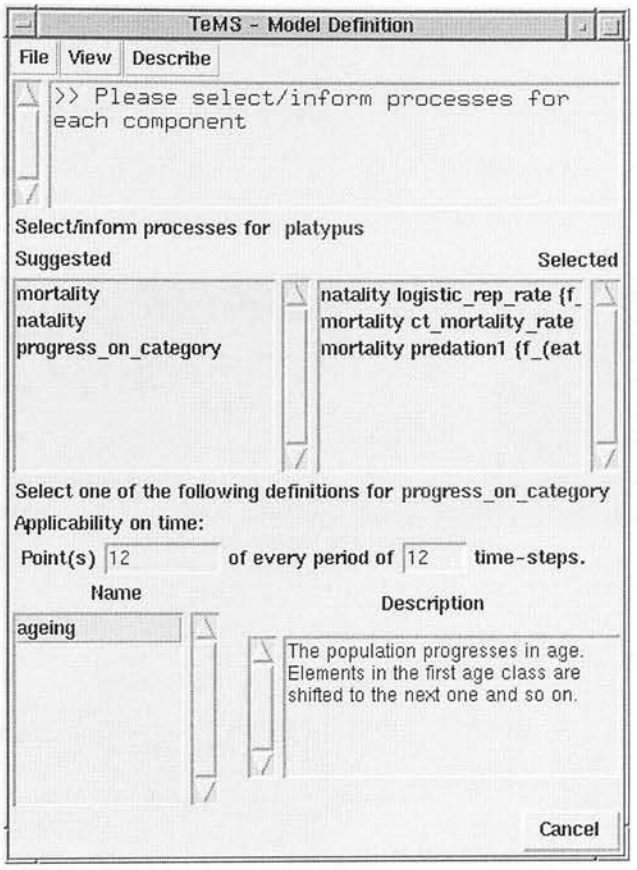

Figure 2.11: Ageing is a process which indicates that at every 12 time-steps, individuals in an age class must move to the next one (it is assumed that elements from the last class will die).

In the scenario described on Section 2.1, there is no dynamics for the fox population, that is, there is no ecological process changing that population and therefore no processes need to be defined. To avoid error situations, the current version of TeMS requires that at least one process must be defined. Thus, to represent our assumption of constant population, we can define a mortality process with a constant rate of value zero (no deaths) which will have no effect on the population. This is shown on Figure 2.12.

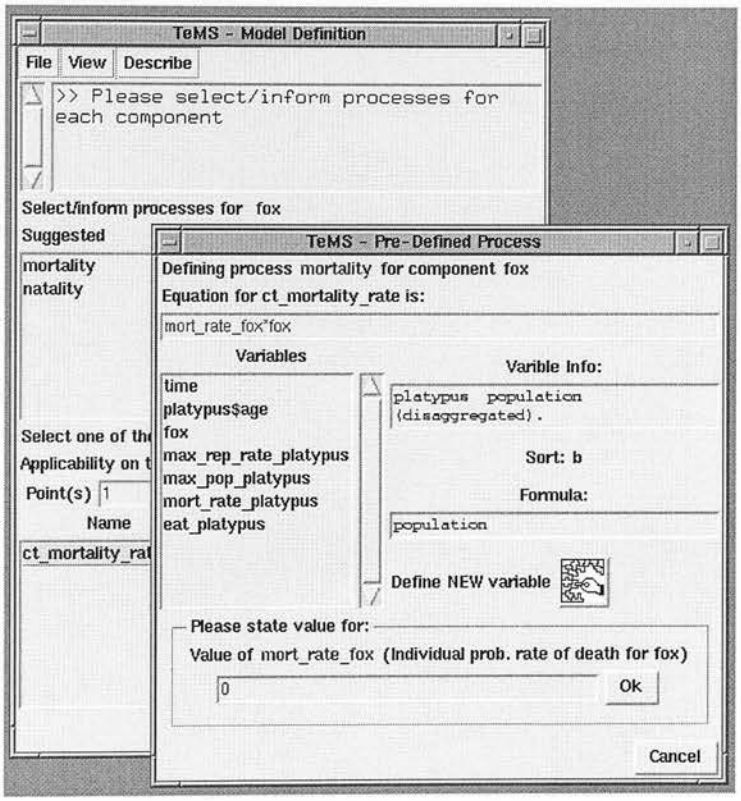

Figure 2.12: A mortality process with mortality rate of zero (therefore with no effect on the population) is defined to represent a constant fox population.

Until now, the interaction between TeMS and the user aimed at representing the scenario presented at the beginning of this chapter and has been used to prepare the resources that will generate the final model, which is a Prolog program.

At this point, TeMS waits for the user to press a button on its main window to start the program generation. Figure 2.13 shows in the window in front, a piece of the resulting model. The predicate highlighted (`population/2`) is the core predicate of the simulation model. In the window behind, there is a listing (normally not displayed to the user) which shows some stages on the generation of `population/2` – from an initial skeleton of the predicate, an argument is added and two programming techniques are applied.

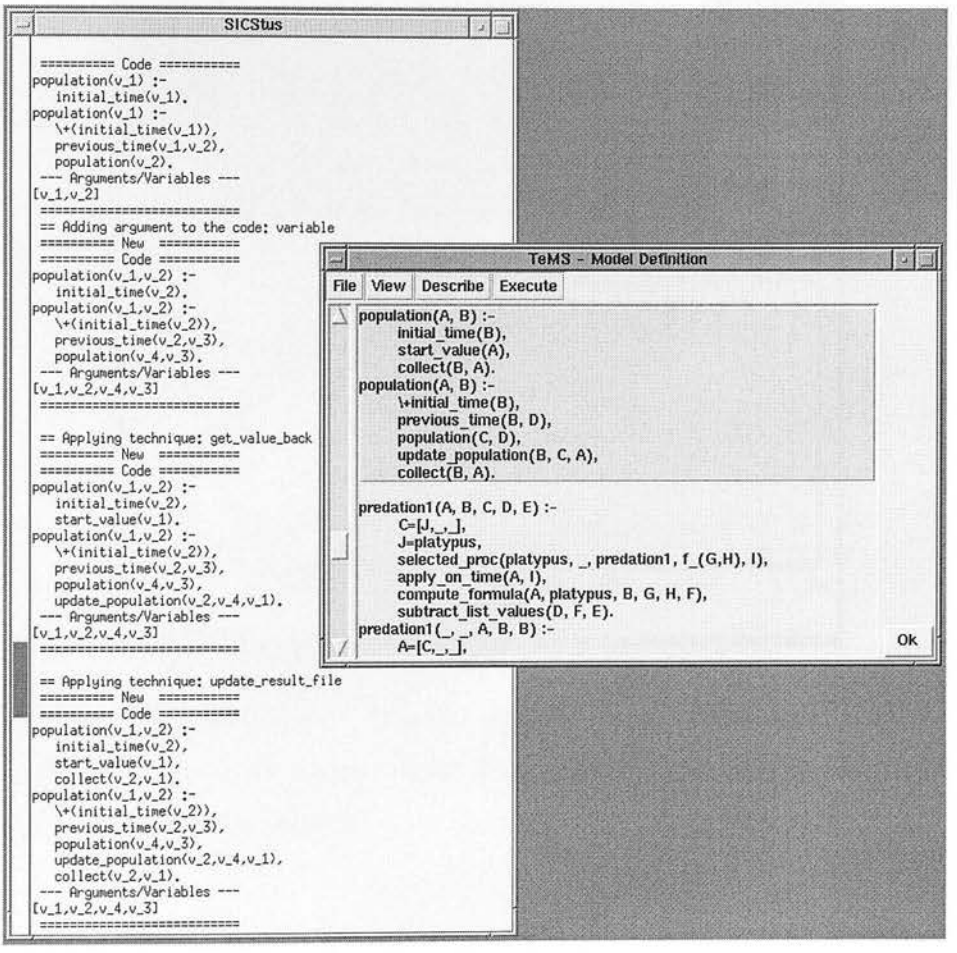

Figure 2.13: Program generation – Window in front: part of the final model, with the core predicate (`predicate/2`) highlighted. Window behind: stages of construction of the predicate `population/4`.

The resulting code can then be used to carry out the simulation within the same environment. TeMS also uses a standard graphical package to plot the population curve, based on population values at each time-step, as shown on Figure 2.14.

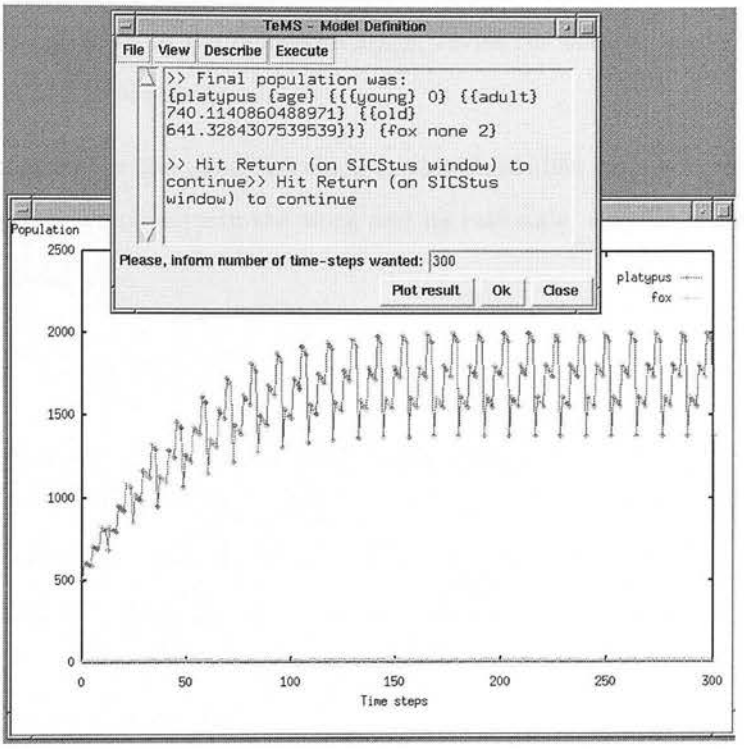

Figure 2.14: Using the model – Window in front: state of the main window after execution of the model. Window behind: The values of population at each time step (from a total of 300) are plotted.

2.3 Concluding remarks

We have illustrated how a modeller, starting from a scenario she wants to explore, can interact with TeMS and go through several design decisions, ending up with a model which is built based on the user choices and domain-dependent structures. The user can then use the model and observe the simulation results within the same environment. It is possible to go back to the definition stage, revise the choices made and generate another model in a exploratory manner.

Now that a context for the ideas discussed in this thesis has been set, we expect that the next chapters, which explain the work and its rationale, may be understood more easily.

Chapter 3

Logic programming and modelling in ecology

We start this chapter with a brief discussion on programming and modelling approaches from different yet connected areas of AI, setting the context in which our choice of method for structuring programming knowledge would be placed. Next, we give an overview of research into the formalisation of Prolog programming knowledge, then we comment on some idiosyncrasies of ecological modelling and introduce our approach in connecting these two areas.

3.1 Addressing programming and modelling

Automating the activities of programming and modelling has been a long-sought goal in Artificial Intelligence. That is understandable, given that both activities involve a great deal of time and effort to master and even then success is not guaranteed. There seems to exist several similarities between programming and modelling. One similarity is the difficulty in finding ways of formalisation which would be suitable to automate both activities. There are, in both areas, issues of knowledge representation and reasoning that make them interesting for developing applications and deepening investigation.

The Programmer's Apprentice project [Rich & Waters 90] for example, a knowledge based system for automated programming, tried to develop a theory of how expert programmers analyse, synthesise, verify and document programs and to assist novice programmers on those phases of the programming task. The system uses *Clichés*, which are structures for representing standard methods for performing some task. Clichés have parameterisable pieces of programming code which can be put together according to predefined specifications attending to certain constraints.

In [Rich & Wills 90], Rich and colleagues argue that clichés are used by programmers in many programming tasks and can be used to reconstruct a programmer's design. Although they show that some inherent difficulties of clichés (syntactic variation, non-contiguousness, implementation variation, overlapping implementations and unrecognisable code) can be overcome when tackling small problems, they also recognise its difficulty in scaling up to more realistic applications. It is also not clear how the technology would migrate to other programming paradigms, nor whether it could be used without assistance in a design recognition or synthesis task.

Other approaches that have modelling as their central theme could give us good insight on how to deal with this. Knowledge Analysis and Design Structuring (KADS) [Wielinga *et al.* 92] for example, is a methodology for developing knowledge based systems. Development is recommended to be divided in three phases: analysis (four models of what the system will do), design (three models of how it will be done) and implementation. Analysis is the aspect of KADS which has attracted the most attention of the knowledge engineering community. KADS analysis involves four layers of abstraction, namely: domain layer, which contains facts about entities in the domain and their relationships; inference layer, which gives a declarative description of the problem solving processes used; task layer, which provides a procedural interpretation to the inference layer; and strategy layer, which connects together the problem solving tasks. The inference layer of a KADS model is created by adapting and instantiating an interpretation model that is identified from a library. Interpretation models are classified by task, and so a hierarchical library of generic tasks models is provided.

The definition of a hierarchy of knowledge based tasks is arguably one of the great achievements of KBS research. However, generic tasks may have limited application

in poorly structured domains, where domain-specific structures might be more efficient. On the other hand, KADS' aspect of building a conceptual model of the domain expert's knowledge, including her problem solving strategies, and take it to its final, complete state before doing any program building, is adequate to an approach dealing with modelling at different levels. Finally, it is true that KADS gives some formalisation for structuring knowledge, but it is not clear how a novice could use it, given that the methodology was intended for use by knowledge engineers.

Knowledge modelling approaches often make use of ontologies. An ontology is an explicit specification of a conceptualisation [Gruber 93]. The ontology of a program can be described by associating the names of entities in the universe of discourse (classes, relations, functions, etc.) with human-readable text describing what the names are meant to denote as well as formal axioms constraining interpretation and well-formed use of these terms. In his paper, Gruber presents Ontolingua, a system for describing ontologies in a form that is compatible with multiple representation languages. The interesting issue in that work is the translation between definitions written in a standard, declarative language (an extended version of first-order predicate calculus) into the forms that are input to a variety of implemented representation systems, and so ontologies written in Ontolingua can be shared by multiple users and ported from system to system.

Ontolingua's goal was to provide a domain-independent translation tool. There is an inherent problem of expressiveness caused by taking a common-denominator approach, it could be the case that the application in a single domain would lessen the compromise when using languages of different power of expression. The set of idioms that Ontolingua can recognise and translate is defined by an ontology – the Frame Ontology, which defines second-order relations that constitute a representational framework similar to clichés and other high level structures.

In [Eriksson *et al.* 94], Eriksson and others describe how a domain ontology was used as the model for the design of knowledge-acquisition tools. The system, Dash, is a metalevel tool that allows developers to generate domain-specific knowledge-acquisition tools from domain ontologies. The project focused on the reuse of ontologies and problem solving methods for knowledge based systems, and the target tools allow non

programmers to edit knowledge structures graphically and incorporate domain terms meaningful to domain specialists. The Dash architecture uses a dialog-design, whose purpose is to establish a high-level structure of the target tool, and a layout-design module, whose task is to instantiate the editors of the target tool given a dialogue structure. The output of Dash is an executable declarative specification of the target tool.

There are two interesting aspects in this project: First, ontology-driven generation allows generation of a domain-specific (as opposed to method-specific) tool. Second, there is a graphical tool which allows a user to inspect most of intermediate representations in the tool generation processes. It is worth observing that even when a well structured representation like an ontology is used, the developer might need to make annotations to it in order to improve representation of other aspects of the task in question (knowledge acquisition).

Another interesting approach to the issue discussed here come from people who are trying to make both programming and modelling available to children, possibly in a learning context. Domain dependent simulation environments using graphical rewrite rules belong to this group. [Smith *et al.* 94] and [Rader *et al.* 98] are examples of such initiatives. The main point is to make programming an easier task, which involves addressing the so-called "end-user programming problem" [Smith *et al.* 94]. By end-users it is meant people who use computers but who are not professional programmers and cannot modify the "applications" they are using unless the designer explicitly built in such modification, and usually that sort of modification is limited to setting preferences. Smith and colleagues also argue that it is necessary to minimize the conceptual or semantic distance between people's mental representations of concepts and the representations that the computer will accept, they support the idea of "programming by direct manipulation" and when associated to a proper set of resources, can form "languageless programming".

The use of graphical rewrite rules is still too restricted to allow any generalisation on how this approach would scale up to more complex situations. Nevertheless, the discussion these approaches propose is an important one. Programming languages are too limiting to modellers who do not know or do not want to struggle with it. It seems

clear that more user friendly approaches need to come into action.

As we suggested in the preceding paragraphs, modelling and programming environments have been tried with different levels of success in learning contexts, although reports of tightly controlled experiments with children in schools are still scarce (e.g. [Brand *et al.* 98, Lewis *et al.* 98]). A good study on the potential of knowledge based tools for educational modelling is presented in [Conlon 97]. Conlon starts by remembering that an analogy between children and knowledge engineers has limits and argues that knowledge acquisition systems will enable more successful model building by children than has been possible with Emycin-like shells. Three new classification-oriented knowledge acquisition tools are discussed and evaluated. Strict-designed experiments judge successful model building according to four kind of measurements: process, product, attitude and learning outcome. Three specific measures associated with quality (correctness, efficiency and conciseness) were formally defined and used to analyse children models semi-automatically.

It is worth mentioning the evidence, based on the findings of that research, that computer modelling can potentially offer valuable support to several classroom activities and it seems to be at least partly in tune with some influential trends and developments, confirming that there is an enormous educational potential for knowledge based modelling in education.

Other work which addresses the issues discussed here from yet another point of view, is intended to be used in a educational context but within a specific domain, that of ecology. [Salles 98] addressed how qualitative reasoning can be brought together with ecological modelling and learning. That work is concerned with qualitative representation of quantitative knowledge, and may refer to magnitudes (relevant information for describing the system at each state) and derivatives (useful for expressing knowledge about the dynamics of the system). An ontology for development of qualitative models (Qualitative Process Theory – QPT) was used to model a number of ecological problems. An important point raised is the potential of the qualitative models developed to generate explanations in learning environments.

Although the focus of Salles work is qualitative reasoning, he pointed out that qualit-

ative modelling may be a complement to quantitative modelling in ecological research. There are at least three issues where that work takes the same direction as the ideas described here: First, the importance of the first phase in modelling – the conceptual model, which is done in qualitative or diagrammatic terms is highlighted. Second, a process-based approach was used in the development of most qualitative models. Finally, Salles think of his work as a contribution for "modelling the modelling process", so do we.

There are some modelling environments that have been used in ecology with different levels of success. We briefly discuss them in Section 5.3.1.

In the next section we narrow the focus of our discussion towards the object of our work by reviewing some methodologies for synthesis of logic programs.

3.2 Structuring logic programming knowledge

Formalising logic programming knowledge has motivated various proposals with the common goal of making explicit programming practices used to write reliable programs. One such approach is that of *Prolog Programming Techniques*, first presented throughly in [Brna *et al.* 91]. Prolog programming skills involves to exploit the relatively simple syntax of Prolog in elaborate ways. Prolog programmers do this by using certain patterns in a systematic way, these patterns – loosely named *techniques* – provide the means for structuring the knowledge involved in the programming. Sections 3.2.1 to 3.2.3 show how these and other related ideas have been exploited in the logic programming community.

3.2.1 Skeletons and additions

A variety of common constructs occurring in Prolog programs have been recognised and received names such as "accumulator pair", "difference structure", "constructing data-structures in the clause head" and "failure driven loop". Despite these terms having a commonly understood meaning at the code level, they often lack precise definition and for this reason, are usually illustrated by giving examples of predicates containing

instances of them.

Sterling, Kirschenbaum and colleagues [Kirschenbaum *et al.* 89, Sterling & Kirschenbaum 93] proposed a now widely accepted approach to develop logic programs, called *stepwise enhancement*. The basic idea is to conduct the construction of well structured, standardised Prolog programs by separating the different basic control flows – the *skeletons* – from the various standard Prolog programming practices, which we refer as *additions*[1]. This approach can be summarised as follows:

1. The first step is to choose the basic control flow needed to solve the problem and embody it in a skeleton. Usually this is done by selecting a skeleton from a previously defined library.
2. Over this skeleton extra computations are included by applying additions to yield an *extension* of the basic skeleton.
3. This extension can now be regarded as a (new) partial program which allows to repeat the process until the final program has been developed.

To illustrate this method, take the skeleton `traverse` which traverses a list partitioning the control of flow into several branches according to some criteria, in our case depending whether the head of the list is a prime number or not, as follows:

[1] Sterling uses the term *technique*, but in this work, we will consider both skeletons and "additions" as specialisations of techniques.

```
%partition traverse
 traverse([X|Xs]) :-
   prime(X),
   traverse(Xs).
 traverse([X|Xs]) :-
   non_prime(X),
   traverse(Xs).
 traverse_n([]).
```

Consider the addition *build* which rewrites a program producing a new one which creates a new list by selectively including elements from the original list. *build* adds an argument (the new list) to the defining predicate and an extra goal to the body of the clause, where the new list will be constructed.

The following is a representation for this addition using the rewrite notation explained in Section 4.3, where $\mathcal{P}$ is the defining predicate and $\{A_1, \ldots, A_n\}$ its arguments; $\mathcal{C}$ is a conjunction of subgoals for $\mathcal{P}$. $Case(X)$ is a conjunction of subgoals involving X and $Relate(X)$ is a conjunction of subgoals involving $X, Ps1$ and Ps.

$$\mathcal{P}(A_1, \ldots, A_n) \leftarrow Case(X), \quad \mathcal{P}(B_1, \ldots, B_n), \quad \mathcal{C} \quad \Rightarrow \begin{cases} \mathcal{P}(A_1, \ldots, A_n, Ps) \leftarrow Case(X), \\ \qquad\qquad \mathcal{P}(B_1, \ldots, B_n, Ps1), \\ \qquad\qquad \mathcal{C}, \\ \qquad\qquad Relate(X, Ps1, Ps) \end{cases}$$

$$\bullet\ \dot{A} = [X|Xs] \wedge \dot{A} \in \{A_1, \ldots, A_n\}$$

$$\mathcal{P}(A'_1, \ldots, A'_n) \leftarrow \mathcal{C} \quad \Rightarrow \quad \mathcal{P}(A'_1, \ldots, A'_n, []) \leftarrow \mathcal{C}$$

$$\bullet\ \mathcal{P} \notin \mathcal{C}$$

The result of applying extension *build* on skeleton *traverse* and instantiating *Relate* is then:

```
only_primes([X|Xs],Ps) :-
  prime(X),
  only_primes(Xs,Ps1),
  Ps = [X|Ps1].
only_primes([X|Xs],Ps) :-
  non_prime(X),
  only_primes(Xs,Ps1),
  Ps = Ps1.
only_primes([],Ps) :-
  Ps = [].
```

This approach to program construction has been extensively used in the development of general purpose tools in different areas of programming [Kirschenbaum *et al.* 94, Bowles *et al.* 94, Vasconcelos 93, Vargas-Vera 95].

3.2.2 Schema-based program definition

A schema is an abstract representation for a class of problems. Building a program from a schema requires us to instantiate the schema for the intended application. The following are examples of approaches in this category.

Gegg-Harrison's schemata

Gegg-Harrison [Gegg-Harrison 89, Gegg-Harrison 93] presented a schemata definition in which precisely defined methods of recursively processing all elements of a list are represented. It focuses on the use of schemata in an Intelligent Tutoring System used to teach recursion.

The following is an example of schema for processing all elements of a list until an empty list is reached. In the representation below, optional arguments and subgoals are denoted by angled brackets and an arbitrary number of arguments is denoted by a double angled brackets can be replaced in the schema.

```
schema_A([],≪&1≫).
schema_A([H|T],≪&2≫) :-
        <pre_preds(≪&3≫,H,≪&4≫)>,
        schema_A(T,≪&5≫),
        <post_preds(≪&6≫,H,≪&7≫)>.
```

This schema can be used to obtain the predicate `append/3`, which concatenates two lists in a third list. This is done by substituting `schema_A` for `append`; `&1` for `L,L`; `&2` for `L,[H|R]`; `&5` for `L,R`; and also `pre_pred` and `post_pred` for the null string. The resulting program is:

```
append([],L,L).
append([H|T],L,[H|R]) :- append(T,L,R).
```

Several other predicates can be generated from `schema_A` (e.g. `length/2` and `reverse/2`).

Although possessing useful capabilities, like the ability to find generalisations of programs and schemata, Gegg-Harrison's schemata deal only with list processing, and there are also limitations on the positioning of arguments.

Barker-Plummer's clichés

Barker-Plummer proposed a way to construct Prolog programs by instantiating program generalisations called clichés. Clichés are a way of representing generalised procedures, and it has been said that they are algorithmic fragments represented in some formalism[Vasconcelos 95]

This concept can be illustrated with the following example which is a cliché to traverse a list checking when the elements of the list meet the condition defined in the predicate `Q/n`.

```
$P/n :- universal($Q/n).

$P([],&Aux).
$P([H|T],&Aux) :- $Q(H,&Aux), $P(T,&Aux).
$end_cliche$
```

Symbols prefixed by $ are cliché parameters (with the exception of `$end_cliche$`, which is the cliché's end-marker) and can be instantiated to any predicate name. The first line of the cliché is the header, with the name of the defining procedure (in this case, `P`), the name of the cliché and the parameters (in this case, `Q`) required to define the procedure. The next lines are the Horn clauses which define `P`.

We illustrate the application of this cliché with the definition of the predicate `numbers_only`, which succeeds if the argument is a list of terms satisfying the `number/1` predicate. This procedure is obtained by instantiating `$Q` to `number/1` and `$P` to `numbers_only`, resulting in the following:

```
numbers_only([]).
numbers_only([H|T]) :- number(H), numbers_only(T).
```

Clichés are similar to Gegg-Harrison's schemata, although using a different notation. They describe generically the flow of control of a procedure, but because it relies heavily on the user's skills – instantiations are left entirely to the user – it can give no guarantee regarding the properties of the programs obtained.

Bundy's recursion editor

Bundy [Bundy *et al.* 91] describes an editor to assist programmers in writing programs by combining partial solutions. The editor assures syntactic correctness of the programs and also assures the correct use of recursion – the recursive definition is guaranteed to terminate and be neither over-defined (where inconsistent set of definitions may happen) nor under-defined (where combination of input/outputs may not be considered in a definition).

There are two basic schemas for the syntactic transformations. One is the primitive recursion schema below:

$$\mu(\flat, \Phi) \rightleftharpoons \nu(\Phi)$$
$$\mu(\sharp(\Psi, \Theta), \Phi) \rightleftharpoons \xi(\Psi, \Theta, \Phi, \mu(\Theta, \Phi))$$

where: μ is the recursive procedure being defined; ν and ξ are the procedures in terms of which μ is being defined; Θ is the recursion variable; Φ is the parameter(s); $\flat$ and $\sharp$

are the constructor functions; Ψ is the constructor parameter; and $\rightleftharpoons$ is the equivalence relation.

And the other is the non-recursive schema:

$$\alpha(\Phi) \rightleftharpoons \beta(\Phi)$$

where: α is the procedure being defined; β is a previously defined predicate; Φ is a parameter; and $\rightleftharpoons$ is the equivalence relation.

Bundy's editor helps programmers devise terminating and well-defined programs, but the set of commands is a "bottleneck" for its use by novices. The editing commands are syntactic transformations of low granularity, and it may be quite difficulty for novice programmers to know how to apply which command. Besides, there is no provision for combine editing commands or add new ones.

Bental's tasks and prototypes

The goal of Bental's architecture to recognise design decisions in Prolog [Bental 94] is try to find design flaws, known as *bodges*. There are two sorts of knowledge represented in this structure: one is about a specific domain (a game of noughts and crosses) and the other is about possible implementations for the sub-tasks and data-structures involved in the solution to a previously defined problem. The *programming tasks* determine what is to be done and the *prototypes* are ways of implementing the tasks. Prototypes are standard programming practices similar to Gegg-Harrison's schemata, which can be used for different purposes (tasks).

3.2.3 More on techniques-based approaches

Clause-level techniques

Bowles [Bowles 94] uses a different approach to techniques to that of Kirschenbaum et al. In this case, techniques are local to clauses, rather than applied across whole predicates, and their intention is to capture the relationship between the head and recursive arguments in the recursive clauses of programs.

The following is a reverse list predicate with an example of a technique (*list head*) where the relation between the head and the subgoals is that one argument is a list and the other is the tail of that list.

```
rev([], R, R).
rev([H|T], R0, R) :-
    rev(T, [H|R0], R).
```

Techniques and program slicing

Vasconcelos [Vasconcelos 95] presents a view of techniques related to the work on *program slicing* [Weiser 82, Weiser 84, Gallagher & Lyle 91], a technique for restricting the behaviour of a program to some specified subset of interest – the slice of a program with respect to program point p and variable x consists of all statements and predicates of the program that might affect the value of x at point p.

Vanconcelos proposed a medium in which programmers can design, test and organise Prolog Programming Techniques. His goal was to provide an automated tool, a *Prolog Techniques Meta-Editor* (TME) which enables techniques to be abstracted from annotated source code and then combined to form new programs. TME defines a most general technique and allows the user to specialise it.

Techniques editing

The concepts of programming techniques were derived from the way in which programmers acquire and use their skills of a particular language. Most of this expert knowledge is still obtained by long experience, which justifies the effort in building better environments to guide the construction of programs. Two of such tools aimed novice Prolog programmers: The first one, described in [Robertson 91] was built directly from the step-enhancement methodology. That editor had a set of skeletons and additions represented in a DCG notation, which could be manipulated under guidance by the user, to ultimately produce a correct Prolog program. The second one, presented in [Bowles 94], defines techniques as common patterns of code which capture the relationship between the head and recursive arguments in the recursive clauses of

programs, i.e. techniques act inside the body of one predicate rather than over a whole program.

The main advantage of those two editors was to concentrate the attention of the user (student) in the main constructors used in Prolog programming, allowing him/her to experience, and maybe to have a better understanding of a standardised and structured program construction. Programming techniques have also been used to manipulate programs in more sophisticated ways than generation of simple programs in a novice-oriented system.

In the next section we list some issues in ecology and ecological modelling and review earlier research trying to address those issues by using the logic paradigm.

3.3 Modelling in Ecology

3.3.1 Ecology

Ecology has been defined as "the scientific study of the interactions that determine the distribution and abundance of organisms" [Krebs 78]. This broad definition suggests how complex the subjects in this science are. Nevertheless, large-scale problems which human beings are facing nowadays (management of natural resources, pollution, worldwide climate changes, etc.) require better ecological understanding. The following aspects of ecology make it difficult to adopt a conventional scientific approach:

- in ecology, subjects are distributed over several "levels of organisation". The focus of interest depends directly on the level in which the problem has been considered (*e.g.* individual level, population level, ecosystem level, etc.).

- ecological systems are characterised by complex interactions. An object of interest affects many other ecological components and in its turn is affected by them. Therefore, in most cases is not possible to understand one object (and its behaviour) separately.

- often the units of study (individual organisms, species, sub-systems, etc.) to a problem are not clearly defined, and can be different depending on the point of

view from which a problem is addressed.

- ecological systems are characterised by many sources of uncontrolled variation and therefore it is very difficult to know the initial state, and the exact influence of external factors. Thus, it is difficult to conduct a well controlled ecological experiment.

These are the sort of issues that challenge other conventional approaches and are interesting to research in Artificial Intelligence because in this case, as in many other real-life situations, it is impossible to provide a complete specification of the problem before beginning experiments with prototypes.

3.3.2 Addressing ecological modelling

Modelling is motived by a problem which needs to be solved, it involves moving from the complex ecological domain to a simulation model implemented as a program. In this task, there is a large amount of knowledge which needs to be identified, represented, organised, and adjusted.

There have been few experiments involving logic programming and ecological modelling, but it seems a consensus that there is no general answer to the issues mentioned in Section 3.3.1. Most work focuses on ways of dealing with restricted situations [Robertson *et al.* 91, Muetzelfeldt *et al.* 89, Muetzelfeldt 95, Brilhante 96, Mota 98].

One of the more comprehensive of the attempts to tackle those issues was the Eco-Logic project [Robertson *et al.* 91] in which the basic idea was to supply logic-based approaches to describe an ecological situation and use that representation to obtain, by successive refinements, a simulation model. One of the main tools produced in the EcoLogic project was the EL system.

EL can be viewed as an Intelligent Front End[Bundy 84] which utilises a sorted logic problem description language to provide an interface for constructing Prolog simulation programs. In EL, the mechanism for generating a complete program lies in a structure of building blocks called Prolog schemata.

The EL System

There are two main sorts of simulation systems: domain independent and domain dependent. Domain independent systems have no explicit knowledge of the problem to be solved and, for this reason, cannot assist the user to make decisions about the model (*e.g.* how to idealise the real problem to the representation language used in the system), except those determined by the nature of the programming language being used. On the other hand, domain dependent systems can be tuned to the type of problem which is to be solved with the simulation model. Although limiting the scope of the system, this makes it possible to provide strong support to the user and may facilitate the tasks of problem specification and orientation of the solution process.

EL is a domain dependent system which has a program description module to help the user in the selection of which schemata must be applied, by reducing the range of schemata suitable for application in a determined context.

Defining the simulation program is the last of the four phases in a complete EL session (the other three are: describing the ecological system; checking the sort hierarchy, and filling in characteristics of the solution). This phase can be divided in two stages:

1. EL extracts the initial queries, which are properly instantiated.

2. EL constructs a program, by the recursive application of schemata.

3.3.3 EL's schemata

A Prolog *schema* is a parameterisable structure for packaging the Prolog code necessary to implement a part of a Prolog program. A schema has packaged inside it not only a piece of code, but also the information about the conditions under which that schema should be used, and the requirements for it to be used along with other schemata. An example of using schemata to generate simulation models is the EL system[Robertson *et al.* 91], a domain dependent system which helps the user in the selection of which schemata must be applied under a determined context. A typical schema definition, like the one used on EL, has the following information:

- *Name* for the schema.

- The simulation *goal* for which the schema solves. If *preconditions* and *actions* execute successfully, and the *subgoals* are satisfied, then the schema is guaranteed to solve this goal.
- List of *subgoals* which the schema produces.
- The piece of Prolog *code* actually supplied by the schema to the final program.
- Procedure calls (*actions*) which must be satisfied before the schema can be applied.
- A condition call which must be satisfied before the schema can be used (*precondition*).

The following is an example of a Prolog schema defining the predicate `attribute/4`. As can be observed, a large part of instantiation work in schemata is done by pattern matching of the parameters.

```
schema([A,' of ',O,' grows logistically'],          % name
       attribute(A,O,T,N),                          % goal
       [initial_time(T),
        last_time(T,T1),
        initial_value(A,O,N),
        parameter(carrying_capacity,O,K),
        parameter(intrinsic_rate_of_increase,O,R)], % subgoals
       [(attribute(A,O,T,N1) :-
            \+ initial_time(T),
            last_time(T,T1),
            attribute(A,O,T1,N2),
            parameter(carrying_capacity,O,K),
            parameter(intrinsic_rate_of_increase,O,R),
            N1 is N2+R*N2*(1-N2/K)),
        (attribute(A,O,T,N3) :-
            initial_time(T),
            initial_value(A,O,N3))],                % code
       [],                                          % actions
       (valid_object_for_type(size, A)
        ; valid_object_for_type(energy, A))         % preconditions
       ).
```

However, schemata are concerned with the definitions of whole predicates. This makes it difficult to allow the program generator to alter the definition of predicates at clause

level. It is impossible for users to customise schemata, other than through the predefined instantiation procedures of each schema. In the schema above for example, one might like to change the last subgoal in the first clause of the code supplied, to "`N1 is N2+R*N2*(1-N2/K)*1.2`". Although there is no need to modify any other subgoals in either clauses, another very similar schema would have to be defined, along with additional instantiation and selection procedures. Of course, we could overcome this difficulty in our example by making the equation itself parameterisable but, in general, it can be difficult to predict which parts of a schema to make flexible in this way.

3.4 Discussion

The approaches for structuring the construction of logic programs discussed here, were split in two main groups: schema-based and techniques-based approaches. The main differences between the two categories being the level of separation between control flow and data flow. Schemata are higher level specifications where information about both flows are often embodied together.

All but one of the approaches discussed aimed to provide tools to support *novice* Prolog programmers. Vasconcelos' goal was to formalise Prolog programming knowledge which could be also be used by experienced programmers. Regardless of the class of users intended, all approaches, bar the parameterisable structures of EL, had only one domain in perspective, that of Prolog programming itself.

Techniques can be applied in several stages of the program construction, but it is not clear that all users will find general techniques easy to use, and one may argue that a better approach would be for the user to specify a set of techniques suitable for her use. However, there is no consensus over what is the "right" set of techniques for a particular domain, or whether they would be used in a way similar to those presented in Section 3.2.

We propose to use the techniques editing methodology to generate simulation models in a sub-domain of ecology and from a practical point of view, we are interested in systems

which can be described by the *structures* which are responsible for their behaviour. We propose to use the step-enhancement method to mirror the way the structures in a ecological situation are put together to form a model.

In the next chapter, we report the definition of a library of the mentioned structures.

Chapter 4

Domain-dependent structures for synthesis

4.1 Modelling implications

It has been argued that by allowing users to adapt predicates at clause level a number of improvements could be obtained, including making explicit standard methods for constructing Prolog predicates, thus encouraging structured yet flexible program development.

Techniques-editing is a general method which, depending on the environment using it, requires different degrees of assistance from the user to make decisions about program generation [Vargas-Vera 95]. In some environments, such as Intelligent Tutoring Systems (ITS), this feature is often considered desirable. Those systems need to maintain a tight control over users' actions either to gain information about their plan or to detect errors or misunderstandings on their decisions and act accordingly.

In ITS, especially if they are aimed to teach or improve programming expertise (e.g. [Robertson 91, Bowles 94]), users may interact directly with the pieces of code resulting from the generation process (or some direct mapping to other visual presentation), making it easier to conduct that process from direct user actions. In that case there is a "short distance" between the external actions and the internal processes controlling

the system, therefore a mapping from user actions to (internal) control decisions is not expected to be complex. This analogy is similar to that of *semantic distance* – the distance between the concepts that the system uses and the ones the user has [Hutchins *et al.* 86].

In our case, the scenario is quite different – we assume the user wants only to describe, in familiar terms, the model she wants to represent, intentionally left unaware of those issues involved in the program generation process. There is then a long distance between external actions and internal control decisions. The problem is to bridge the gap between user's knowledge about modelling and the system's knowledge of techniques editing and use that to guide the model generation process.

4.2 Translating existing models to logic programs

In order to understand how Prolog techniques relate to this form of modelling we must reconstruct a representative example in Prolog. For this, we used a model from the ecological modelling literature[Crête *et al.* 81]. This model was selected because:

- It is a representative population dynamics model, used to explore a predator-prey relationship and it appears to contain reliable data supporting a plausible hypothesis.
- It is small enough to be implemented and tested/refined in a realistic time, yet it embodies the main characteristics which are expected to be needed in this sub-domain as well as allowing us to analyse possible extensions.

4.2.1 Main components

The main components of our case-study were wolf (the predator) and moose (the prey) populations. The goal of the model was to study how wolves could regulate a moose population. Reproduction was represented in either of two ways: using constant values or individual reproduction rates. Predation (the main cause of mortality for moose) varied according to season, as in winter wolves have access to alternative food. A

variant of the basic model included hunting as another mortality factor for moose. Starvation (or indirect consequences of malnutrition) was the only mortality factor for wolves. Both populations were structured in sex and age classes so there is a component in the model for each sex × age combination. This is known in the domain as *disaggregation*. The year was divided into two seasons, summer (June–October) and winter (November–May).

4.2.2 Modelling approach

We have used a process-centred paradigm to implement the model. Most of the simulation models we have seen on the literature are described in terms of the ecological processes involved. Such processes are standard in the literature and represent phenomena affecting the main focus of the simulation. For example, in a model to evaluate the progress of some population over time, reproduction and mortality are typical processes affecting the size of population over a time interval. This *modularity* in the representation of aspects of a problem helps to provide a structured approach to modelling.

The core part of the model is an iteration over discrete time-steps. At each time-step, the processes affecting moose and wolf populations are computed and the size of their populations updated. This procedure is applied iteratively until the final time-point be reached. We next summarise some aspects of the reconstruction of the model.

Two levels of time representation and processing were used in the model. This was because most actions over the population should be taken every season, although some were taken annually. Thus, a *season* was considered as the basic time-unit, and a mapping between *season* and *year* was provided.

Both populations were affected by the same categories of ecological processes, namely: *reproduction*, *mortality* and *ageing*, with different instances of them for each population. Some of those eco-processes required the definition of variables – *starvation* of wolves, for example, depended on the availability of prey which had to be converted from the number of elements to biomass. Thus, a *weight table* giving average moose's weights according to their age was set up.

4.2.3 Prolog implementation of the model

The following is part of a simulation program in Prolog which illustrates how the model described above can be implemented as a logic program.

The predicate which encapsulates the whole model is `model/2`. The first of its two arguments is the time-point (`T`) where the simulation will end and the other is a data-structure (`P`) containing the values for the population – which is the result of the simulation.

```
model(T, P) :-
  open_dc,
  population(T, P),
  close_dc.
```

The core predicate of the model is `population/2` (arguments as in `model/2`). It represents an interaction over time-points implemented as a recursive Prolog predicate in which the base-case sets the initial time and population values for the simulation; in the recursive clause the predicate is reapplied to get the value of the population (`Pp`) in the previous time-point (`Tp`). The predicate `update_population/3` yields the update of population values (from `Pp` to `P`) at a certain time-point `T`. These elements provide the "top-level" control of the model.

```
population(T, P) :-
  initial_time(T),
  start_value(P),
  collect(T, P).
population(T, P) :-
  \+ initial_time(T),
  previous_time(T, Tp),
  population(Tp, Pp),
  update_population(T, Pp, P),
  collect(T, P).
```

The data-structure used to represent the population is composed of one substructure for each component in the model. The actual updating of the population has to be done separately for each component, so it is necessary to traverse the whole data-structure. The predicate `update_population/3` adjusts parameters and transfers flow of

control to update_components/4. The latter actually traverses the data-structure and for each substructure – constituted of a component's name (C), information on how that subpopulation will be organised (O) and the values of that subpopulation in the previous time-point (SPp) – it uses the predicate update/4 for updating the value for the corresponding subpopulation, that is, from SPp to SP.

```
update_population(T, Pp, P) :-
  update_components(T, Pp, Pp, P).

update_components(_, _, [], []).
update_components(T, Pp, [[C,O,SPp]|T1], [[C,O,SP]|T2]) :-
  update(T, Pp, [C,O,SPp], SP),
  update_components(T, Pp, T1, T2).
```

Associated with each component in the model there is a set of ecological processes affecting its population (see Section 5.3.2). The combined effect of these processes constitutes the update of that subpopulation at a certain time-point T, as represented by the predicate update/4. The other arguments for this predicate are: Pop, which is the whole data-structure representing population; Pp, which is the substructure representing the population of the component at the previous time-point; and SP (the result), which is the updated value of that subpopulation.

In this case (i.e. the model stated on Section 4.2.1), rep_rate/5, predation1/5, hunting/5 and ageing/5 represent processes affecting moose population and rep_rate/5, starvation/5 and ageing/5 are processes affecting wolf population. The following is the corresponding definition of update/4.

```
update(T, Pop, Pp, SP) :-
  Pp=[C,_,SPp],
  C=moose,
  rep_rate(T, Pop, Pp, SPp, SP1),
  predation1(T, Pop, Pp, SP1, SP2),
  hunting(T, Pop, Pp, SP2, SP3),
  ageing(T, Pop, Pp, SP3, SP4),
  adj_values(SP4, SP).
update(T, Pop, Pp, SP) :-
  Pp=[C,_,SPp],
```

```
    C=wolf,
    rep_rate(T, Pop, Pp, SPp, SP1),
    starvation(T, Pop, Pp, SP1, SP2),
    ageing(T, Pop, Pp, SP2, SP3),
    adj_values(SP3, SP).
```

Each process has its equation (represented within the predicate `selected_proc/5`) computed using a Prolog meta-interpreter (`compute_formula/6`). Those processes are grouped in "categories" according to the way they affect the population of a component (see Section 5.3.2), and they are implemented by predicates specific to each of those categories (e.g. `distribute_on_first_level/5` for natality and `subtract_list_values/3` for mortality).

The following are definitions of predicates `rep_rate/5` and `predation1/5` for moose population. They represent the effects of *reproduction* and *mortality*, respectively, on a subpopulation. Both processes are computed in a quite similar way: an equation defining the process on that subpopulation (`f_(E,D)`) is contained within `selected_proc/5`; the predicate `apply_on_time/2` assures that process should in fact, be effective at a specific time-point; `compute_formula/6` is where the equation is actually computed and its result is used according to the category of the process. Thus, the result of `compute_formula/6` in `rep_rate/5` is the number of births which must be added to the subpopulation; and the result of `compute_formula/6` in `predation1/5` is the number of deaths which must be subtracted from the current subpopulation.

```
rep_rate(T, Pop, Pp, SPp, SP) :-
  Pp=[C,0,_],
  C=moose,
  selected_proc(moose, _, rep_rate, f_(E,D), Proc),
  apply_on_time(T, Proc),
  compute_formula(T, moose, Pop, E, D, R),
  distribute_on_first_level(moose, 0, SPp, R, SP).
rep_rate(_, _, Pp, SPp, SPp) :-
  Pp=[C,_,_],
  C=moose.
```

```
predation1(T, Pop, Pp, SPp, SP) :-
  Pp=[C,_,_],
  C=moose,
  selected_proc(moose, _, predation1, f_(E,D), Proc),
  apply_on_time(T, Proc),
  compute_formula(T, moose, Pop, E, D, R),
  subtract_list_values(SPp, R, SP).
predation1(_, _, Pp, SPp, SPp) :-
  Pp=[C,_,_],
  C=moose.
```

The rest of this program is built using the same idea of starting from the main flow of control, adding elements to complete definitions at a certain level, then defining each subgoal in the same way. From this example, it is clear that it would be difficult for a inexperienced programmer and modeller, to implement a model from its general description.

In the following sections we show how programs like this one can *automatically* be produced. TeMS – the model synthesiser we shall see on Chapter 5, does not confront ecologists with definitions like these, instead, it uses a domain-specific interface to control the selection and synthesis of them.

4.3 Typical model structures

4.3.1 Overview

There is no consensus over what is the "right" set of techniques for a particular domain, so it is useful to be able to extract techniques information from examples. [Vasconcelos 94] presents a methodology for extracting techniques used in a Prolog program. Similarly to its counterpart in the procedural paradigm [Weiser 84], it uses evaluation over the arguments of a predicate to find all clauses relevant to that argument and to partition the procedure into a set of argument slices which are considered separable techniques. However, in that method techniques are extracted *with respect to a specific query* and techniques must always have *only one argument*; conditions

which may not be met in our domain. Furthermore, example-based methods require a representative case library which does not exist for ecological modelling.

The following techniques and schemas were identified by studying the implemented model, associating those structures with features which are likely to be present in many other population dynamics models. Instead of using more abstract representations for techniques, we show them simply as pieces of Prolog code upon which rewrites must be done, with variables beginning with a capital letter. We shall later demonstrate how these are used in model construction.

4.3.2 List of structures

recursion over time points is the technique which defines the flow of control used in the main predicate of the simulation. It supplies a "ticking clock" which allows all the processes in the model to be calculated at appropriated "ticks" in the sequence of time points determined by the clock. From our implementation of Crête's [Crête *et al.* 81] model, the following skeleton for *regression over time points* was extracted ($\mathcal{P}$ must be substituted by the name of the predicate).

```
P(T) :-
    initial_time(T).
P(T) :-
    \+ initial_time(T),
    previous_time(T,Tp),
    P(Tp).
```

A more general form for this skeleton is:

```
P(T) :-
    TimeLim(T).
P(T) :-
    \+ TimeLim(T),
    Adjacent(T,Tadj),
    P(Tadj).
```

With the latter, more instantiation would be necessary. A possible substitution is {*TimeLim*/`final_time`, *Adjacent*/`next_time`}, which produces the skeleton for *progression over time points*.

population updating as the name suggests, incorporates into the code being edited (*working code*) an updating of the data-structure representing the size of one of the populations in the model. An extra argument is added to the defining predicate and extra goals are added to the body of recursive and non-recursive (base-case) clauses, those extra goals relate the updating from the body with the final value in the head of a clause.

We represent the application of this techniques as a rewrite ($\Rightarrow$) upon a program. As used here, a program is a finite set of clauses of the form:

$\mathcal{P}(A_1, \ldots, A_n) \leftarrow \mathcal{C}$

where:

$\mathcal{P}$ is the defining predicate and $\{A_1, \ldots, A_n\}$ its arguments;

$\mathcal{C}$ is a set of subgoals $\{P_1, \ldots, P_m\}$;

$n \geq 0,\ m \geq 0$;

For clarity, we distinguish different variables in schemata and techniques using prime ($'$) symbols.

The population updating can be represented as:

$$\mathcal{P}(A_1, \ldots, A_n) \leftarrow \mathcal{C} \quad \Rightarrow \quad \mathcal{P}(A_1, \ldots, A_n, V) \leftarrow \mathcal{C}, \texttt{start\_value}(A, V)$$

- if $\mathcal{C}$ does not contain $\mathcal{P}$ as a goal (non-recursive clauses), henceforth: $\mathcal{P} \notin \mathcal{C}$,

$A \subseteq \{A_1, \ldots, A_n\}$

$$\mathcal{P}(A'_1, \ldots, A'_n) \leftarrow \mathcal{C}', \mathcal{P}(B_1, \ldots, B_n) \Rightarrow \begin{cases} \mathcal{P}(A'_1, \ldots, A'_n, V') \leftarrow \mathcal{C}', \\ \qquad\qquad \mathcal{P}(B_1, \ldots, B_n, X), \\ \qquad\qquad \texttt{update\_population}(A', X, V') \end{cases}$$

- $A' \subseteq \{A'_1, \ldots, A'_n\}$

ancillary procedures – It might be the case that the core procedure (`population/2` in our example) needs to be preceded ($\mathcal{C}_{pre}$) or followed ($\mathcal{C}_{post}$) by ancillary procedures (e.g. opening and closing files or loading external programs). The way this operation will be implemented depends on the techniques editor used, in our case (see Section 5.2.3), the editor can add subgoals on top or

bottom of the body of a clause. The following is a technique for encapsulating the core procedure within an outermost predicate (`model/2` in our example).

$$\mathcal{P}(A_1, \ldots, A_n) \leftarrow \mathcal{C} \;\Rightarrow\; \mathcal{P}(A_1, \ldots, A_n) \leftarrow \mathcal{C}_{pre}, \mathcal{C}, \mathcal{C}_{post}$$

traverse_pop is an instance of the skeleton *traverse_n* presented in [Sterling & Kirschenbaum 93]. It embodies the main flow of control for traversing a list partitioning it according to some criteria (see Section 3.2). In our case, the list is partitioned in two cases which represent the way the population of components are referred to, namely: disaggregated (population is organised in sub-groups) and non-disaggregated (see Section 5.3.2 for more details on population organisation).

```
P([H|T]) :-
    disaggregated(H,none),
    P(T).
P([H|T]) :-
    \+ disaggregated(H,none),
    P(T).
P([]).
```

get_value is an addition which may be made to the technique above. It adds an argument to the defining predicate and add extra subgoals to the body of each clause in order to create a new data object with initial population information. We assume that `initial_st/3` is the predicate from the model's specification with initial values for each parcel of the population – In `initial_st`(C, D, V) for example, C is a component name, D is some disaggregation dimension and V is the value of that sub-population. C is within one of the arguments ($\dot{A}$) of the defining predicate.

For components with non-disaggregated population, `initial_st/3` and another goal relating its arguments with the final object in the head (for that reason called a "constructor goal") are added to the body of the clause.

For components of a disaggregated population, `findall/3` (a SICStus Prolog built-in predicate) is used to determine all sub-groups and initial-values for all those sub-groups of a population, along with the constructor goal as before. The

technique is then as follows:

$$\mathcal{P}(A_1,\ldots,A_n) \leftarrow \mathcal{C},\quad \mathcal{P}(B_1,\ldots,B_n) \Rightarrow \begin{cases} \mathcal{P}(A_1,\ldots,A_n,V) \leftarrow \mathcal{C}, \\ \qquad\qquad \mathcal{P}(B_1,\ldots,B_n,T), \\ \qquad\qquad \texttt{initial\_st}(C, none, X), \\ \qquad\qquad V = [[C, none, X]|T] \end{cases}$$

$$\bullet\ \dot{A} = [C|C_s],\ \dot{A} \in \{A_1,\ldots,A_n\}$$

$$\mathcal{P}(A'_1,\ldots,A'_n) \leftarrow \mathcal{C}',\quad \mathcal{P}(B'_1,\ldots,B'_n) \Rightarrow \begin{cases} \mathcal{P}(A'_1,\ldots,A'_n,V') \leftarrow \mathcal{C}', \\ \qquad\qquad \mathcal{P}(B'_1,\ldots,B'_n,T'), \\ \qquad\qquad \texttt{findall}(D_2, \texttt{disaggregated}(C', D_2), D), \\ \qquad\qquad \texttt{findall}([D_1, V_1], \texttt{initial\_st}(C', D_1, V_1), X'), \\ \qquad\qquad V' = [[C', D, X']|T'] \end{cases}$$

$$\bullet\ \dot{A}' = [C'|C'_s],\ \dot{A}' \in \{A'_1,\ldots,A'_n\}$$

$$\mathcal{P}(A''_1,\ldots,A''_n) \leftarrow \mathcal{C}'' \quad \Rightarrow \quad \mathcal{P}(A''_1,\ldots,A''_n,[\,]) \leftarrow \mathcal{C}''$$

This structure illustrates that domain-specific techniques depend not only on data-structures parsed as arguments but also on other predicates from the model specification (e.g. `initial_st/3`).

The following is an example of using the skeleton *traverse_pop* with the addition *get_value*. The predicate `initial_value/2` associates a list of component identifiers, usually their names (`C`) with a data-structure containing the model's initial population (`Pop`). If a component has a non-disaggregated population (i.e. `disaggregated(C,none)` succeeds), the initial value for that subpopulation is a single number represented by `initial_st/3`. If a component is disaggregated in several dimensions (see Section 5.3.2), `initial_st/3` contains one number for each parcel of that component's population, and the sets of dimensions and values are found by the built-in predicate `findall/3`.

```
initial_value( [C|T] , Pop ) :-
  disaggregated(C, none),
  initial_value( T , T2 ),
  initial_st(C, [none], P),
  Pop = [[C,[none],P]|T2].
```

```
initial_value( [C|T] , Pop ) :-
  \+ disaggregated(C, none),
  initial_value( T , T2 ),
  findall(C, disaggregated(C,Dim), All_Dim),
  findall([Dim,Vl], initial_st(C,Dim,Vl), All_P),
  Pop = [[C,All_Dim,All_P]|T2].
initial_value( [] , [] ).
```

conversion is another addition to *traverse_pop* which adds arguments and goals to the defining predicate in order to implement a "conversion" of values. It might be used for example, to compute biomass from population values. It uses a predicate (`factor/3`) as a table for conversion factors. The predicate `for_each/5` applies `factor/3` to every sub-group of the population. In the end, a converted value is structured in the same way of the original.

$$\mathcal{P}(A_1,\ldots,A_n) \leftarrow \mathcal{C},\ \mathcal{P}(B_1,\ldots,B_n),\ V = [[C, none, X]|T] \Rightarrow \begin{cases} \mathcal{P}(A_1,\ldots,A_n,V_2) \leftarrow \mathcal{C}, \\ \qquad \mathcal{P}(B_1,\ldots,B_n,T_2), \\ \qquad V = [[C, none, X]|T], \\ \qquad \texttt{factor}(C,F,X,X_2), \\ \qquad V_2 = [[C, none, X_2]|T_2] \end{cases}$$

$$\mathcal{P}(A'_1,\ldots,A'_n) \leftarrow \mathcal{C}',\ \mathcal{P}(B'_1,\ldots,B'_n),\ V' = [[C', D, X']|T'] \Rightarrow \begin{cases} \mathcal{P}(A'_1,\ldots,A'_n,V'_2) \leftarrow \mathcal{C}', \\ \qquad \mathcal{P}(B'_1,\ldots,B'_n,T'_2), \\ \qquad V' = [[C', D, X']|T'], \\ \qquad \texttt{for\_each}(P_y, X', (Y = [D_y, P_y], \texttt{factor}(C', F', P_y, P)), P, X'_2), \\ \qquad V'_2 = [[C', D, X'_2]|T'_2] \end{cases}$$

- $D \neq none$

$$\mathcal{P}(A''_1,\ldots,A''_n) \leftarrow \mathcal{C}'' \quad \Rightarrow \quad \mathcal{P}(A''_1,\ldots,A''_n,[]) \leftarrow \mathcal{C}''$$

The goals $V = [[C, none, X]|T]$ and $V' = [[C', D, X']|T']$ on the left hand side of this technique assures that a technique (*get_value*) defining an initial data object had already been applied.

chained composition joins one or more processes[1] to the body of a clause, composing arguments in such way that the new information supplied by one process is used as an input-argument to the following and so on. An argument in the last process is linked to an argument in the head of the defining predicate.

This technique adds two arguments to the defining predicate. The first one is the initial structure which will be "passed through" a series of processes (each process is a goal to be added to the body of the clause). The second argument is the composed effect of all processes, that is, the updated data-structure. An extra goal defines the initial structure from a subset of the original arguments.

$$\mathcal{P}(A_1,\ldots,A_n) \leftarrow \mathcal{C} \Rightarrow \begin{cases} \mathcal{P}(A_1,\ldots,A_n,I,O) \leftarrow \mathcal{C}, \\ \qquad\qquad\qquad\qquad Relate(A,I), \\ \qquad\qquad\qquad\qquad P_1(A,I,L_1), \\ \qquad\qquad\qquad\qquad P_2(A,L_1,L_2), \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad\qquad P_m(A,L_{m-1},O) \end{cases}$$

$$\bullet\ \mathcal{P} \notin \mathcal{C}, A \subseteq \{A_1,\ldots,A_n\}, m \geq 0$$

$$\mathcal{P}(A'_1,\ldots,A'_n) \leftarrow \mathcal{C}', \quad \mathcal{P}(B'_1,\ldots,B'_n) \Rightarrow \begin{cases} \mathcal{P}(A'_1,\ldots,A'_n,I',O') \leftarrow \mathcal{C}', \\ \qquad\qquad\qquad\qquad \mathcal{P}(B'_1,\ldots,B'_n,L_m,O), \\ \qquad\qquad\qquad\qquad Relate(A',I'), \\ \qquad\qquad\qquad\qquad P'_1(A',I',L'_1), \\ \qquad\qquad\qquad\qquad P'_2(A',L'_1,L'_2), \\ \qquad\qquad\qquad\qquad \vdots \\ \qquad\qquad\qquad\qquad P'_m(A',L'_{m-1},L_m) \end{cases}$$

$$\bullet\ A' \subseteq \{A'_1,\ldots,A'_n\}, m \geq 0$$

[1] See Section 4.2.2 for a definition of "process" as mentioned here.

partition is a skeleton where the first goal (which can be unfolded as a composition of several checking subgoals) defines the applicability of that clause.

```
P( Key, St, NewSt ) :-
    Case_1(Key),
    P1(St,NewSt).
⋮
P( Key, St, NewSt ) :-
    Case_n(Key),
    Pn(St,NewSt).
```

Applicability of ecological processes according to seasons, where each case condition determines whether the current time point in the simulation is within a particular season, is an example of where this technique would be used.

In our example, the skeleton *partition* with the addition of *chained composition* is used to define the predicate `update/4`, which represents population updating for a certain component. At every time-step, the effect caused by ecological processes on the component's population is computed by this predicate. The following is a listing of `update/4`, where `Comp = wolf` is the case condition.

```
update(T,Ref,P,Pout) :-
  Comp = wolf,
  P = [Comp,_,Pin],
  ageing(Comp,T,Ref,Pin,S1),
  natality(Comp,T,Ref,S1,S2),
  mortality(Comp,T,Ref,S2,S3),
  adj_values(S3,Pout).
```

shift moves part of the contents of every "cell" within a data structure to the next (adjacent) one. Two arguments are included in the defining predicate: the first is the part of the structure to be shifted and the second is the data object to be created. The two *Relate* goals added to the body of recursive clauses relate the two adjacent cells in the data structure and the third goal is the constructor goal

for the new object.

$$\mathcal{P}(A_1,\ldots,A_n) \leftarrow \mathcal{C} \;\Rightarrow\; \mathcal{P}(A_1,\ldots,A_n,X,[]) \leftarrow \mathcal{C}$$

$$\mathcal{P}(A'_1,\ldots,A'_n) \leftarrow \mathcal{C}',\ \mathcal{P}(B_1,\ldots,B_n) \Rightarrow \begin{cases} \bullet\ \mathcal{P} \notin \mathcal{C} \\ \quad \begin{cases} \mathcal{P}(A'_1,\ldots,A'_n,I,S) \leftarrow \mathcal{C}', \\ \qquad \mathcal{P}(B_1,\ldots,B_n,V,T), \\ \qquad \mathit{Relate}(H,\dot{A},V), \\ \qquad \mathit{Relate}(H_2,\dot{A},I), \\ \qquad S = [H_2|T] \end{cases} \\ \bullet\ \dot{A} = [H|R] \wedge \dot{A} \in \{A'_1,\ldots,A'_n\} \end{cases}$$

In our example, the population was organised in age-classes and this technique would be used to shift the values of each class to the adjacent one, a task needed when representing *ageing* in the model. Using the skeleton *traverse* (Section 3.2) with the addition of *shift*, it is possible to obtain the definition of `move_elem/3`:

```
move_elem([],_,[]).
move_elem([H|T],In,O) :-
      move_elem(T,N,T2),
      H = [AgeClass,N],
      H2 = [AgeClass,In],
      O = [H2|T2].
```

4.4 Concluding Remarks

The preceding examples demonstrate that domain-dependent techniques depend upon other domain-specific data-structures. Hence, a library of such techniques, which will probably include other standard data-structures, may only be complete for the set of operations or features of a limited class of applications. The range of operations carried out by an application might have to be known prior to definition of the techniques library, because the design decisions shape the techniques and schemata in the library.

Chapter 5

TeMS

In this chapter we show how the methods and structures for program construction presented in Chapter 4 can be used to define a modelling environment which bridges the gap between domain knowledge elicited from a user and previously defined design structures and methods. The research and implementation of our model synthesiser also incorporates a guided and structured approach to modelling whose main focus is on the design decisions taken during this process.

5.1 An architecture for structured modelling

The modelling process is a semi-formal set of rules which we take to *produce* a model, *implement* it in some formal language, *derive consequences* (predictions) from the model and *evaluate* these in light of the use to which the models are to be put [Haefner 96]. Although much work has been devoted to the creation of general-purpose languages and tools to support one or more of these tasks, the underlying set of rules is often concealed among several layers of representation. It is expected that a user of such languages and tools will, eventually, "discover" those rules, but it is a process relying heavily on the personal profile and experience of each user. Modelling process rules are seldom detailed and specific, most of them are general strategies or guidelines.

In the foreword of [Polya 90][1], Ian Stewart argues that it is inherent in the nature of guidelines that they do not work if taken too literally, they should be interpreted through the "eyes of experience".

This lack of more specific instructions makes modelling a difficult subject especially for novices, who would have no explicit references or "landmarks" to follow when performing this task, hence the need for modelling environments with an explicit path through the design decisions made during the process. We argue that such environments would, to some extent, enhance the understanding of the task.

5.1.1 A knowledge elicitation task

As a prerequisite for building a plausible model, we assume that the builder is knowledgeable about the system which is the basis for the model. Modelling can be considered as a sort of discovery learning, but a very demanding one [Ross 85], and to have knowledge about a system is not sufficient for constructing valid models of it. Defining *what* is to be modelled, what features of the system are important and *how* they interrelate is the first stage a modeller has to go through. Even when a modeller is able to devise a conceptual model with the aspects she wants to have represented, its *implementation* may be a problem. One can have a good idea of a model's main parameters or the final shape it should have, but not know how to put all together in a neat piece of code in some programming language.

A modelling environment with the ability of generating structured programs without relying on the user own knowledge of programming in that specific language, not to mention more subjective aspects like *style*, would be useful for novice and experts alike and could encourage the formation of libraries of well-written models (programs) to facilitate reading and understanding by other modellers.

[1] In his book, George Pólya discusses a number of general problem-solving techniques (called *heuristic strategies*) to which other "generic rules", like the modelling rules mentioned here, bear remarkable resemblance.

5.1.2 From problem description to model generation

The need for constrained languages

All-purpose modelling environments, by definition, are not mainly concerned with the requirements of specific domains, and therefore any resources for facilitating their use and understanding their rationales will be expressed in generic terms (except if domain-specific user-modelling is used). That is, even if an all-purpose environment can make clear to the user its underlying rules, it may be that they still are too generic to be really useful to a novice modeller.

In addition to the preceding issue, if we need *automatic generation* of well-structured programs, which must be consistent implementations of models, general methods will usually not suffice because they provide insufficient guidance. We would have to use domain-specific methods and/or structures such as those shown on Chapter 4 to control generation.

The modelling environment proposed here deals with these issues by constraining the class of models it can deal with. It makes explicit to the user the design decisions which shape the model intended and uses a top-down approach with tight control to guide the user through these decisions. It also uses a previously defined library of domain-specific techniques and schemata along with the knowledge added by the user, to conduct the generation of the implemented model (a Prolog program). Effects to this environment are that the user might have a better understanding of what the crucial questions about the model are, and she could see and evaluate the effect of these questions on the model implemented.

TeMS - Introduction

The model synthesiser described here (**TeMS** – **Te**chniques-based **M**odel **S**ynthesiser) elicits knowledge from an ecologist, leading her through a structured sequence of design decisions and using her definitions to set up the program generation. TeMS can assist novice or expert modellers with little or no knowledge of programming as well as those who want to build their models in an environment able to produce more structured

code.

The system presented here:

- is a tool that leads ecologists through several structured steps for the specification of a model;
- employs its own knowledge-base as well as the knowledge acquired from the user, and generates a runnable Prolog program that is the implementation of the model;
- can execute the constructed Prolog model;
- may record the path taken during the definition process, allowing the display of the choices made during the modelling process.

5.1.3 Three phase modelling for automatic synthesis

Modelling using TeMS consists of going through three phases, as shown on Figure 5.1. In the first phase the user is led through six stages of model description where key features of the model are defined. In the second phase the knowledge elicited is used to select and apply an appropriate set of techniques to build the model described. Finally, in the last phase the user can see the Prolog code for the model and/or run it. If a redefinition or adjustment of the model is needed, the user may start again from some stage on phase I.

This process-based approach to modelling can be used to model population dynamics systems with any number of components. The population can be disaggregated (symmetrically) in any number of dimensions and all variables can be structured (indexed) in the same way. Processes are grouped by categories (see Section 5.3.2) and they refer to which population they affect. Interactions between components and external factors are represented by variables.

As can be seen by the description above, although it is restricted in several ways, this modelling framework can be widely used in several branches of the ecological domain as well as in other domains where the process-based approach is suitable. For more

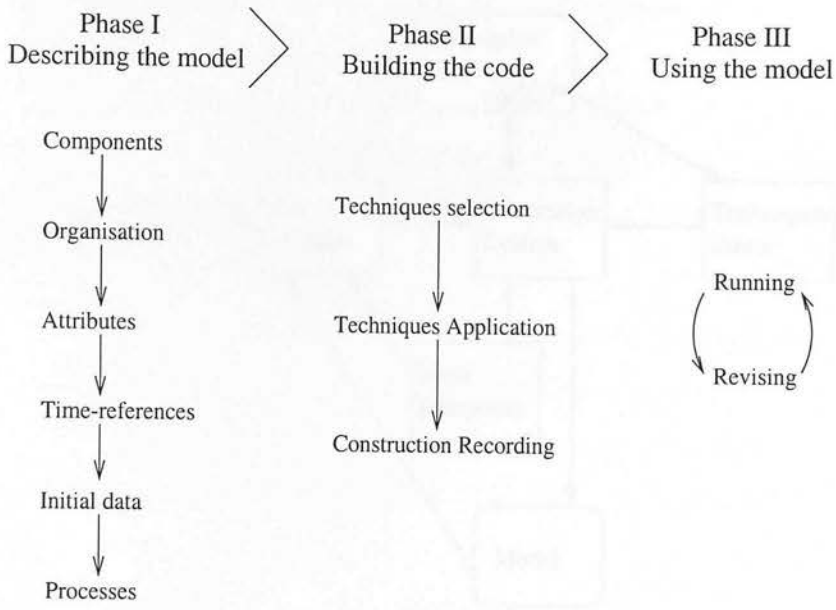

Figure 5.1: Three-phase modelling - modelling in TeMS consists of going through three phases (from left to right), each one with a number of stages (from top to bottom).

details on limitations and potential uses, see Sections 5.4.2, 6.3.5 and 7.3.

5.2 Design

Figure 5.2 shows the general structure of TeMS. The generation system is the core of the tool, defining which predicates will form the final program and in which order they will be generated. For each predicate, the set of techniques editing operations to be carried out by the techniques editor must be specified. The user describes a model through a user interface that also has facilities for executing and displaying the model specified. A Prolog meta-interpreter deals with equations used during the model definition and model execution stages. For each model defined, the knowledge base is extended with knowledge about the design decisions made.

Sections 5.2.2 to 5.2.6 give structural descriptions of the modules on Figure 5.2. The main idea is given to detail the role of each module (and the data-structures used by them) in the generation process. There are some references to figures and examples on Section 5.3. The latter presents a step-by-step description of the three-phase modelling

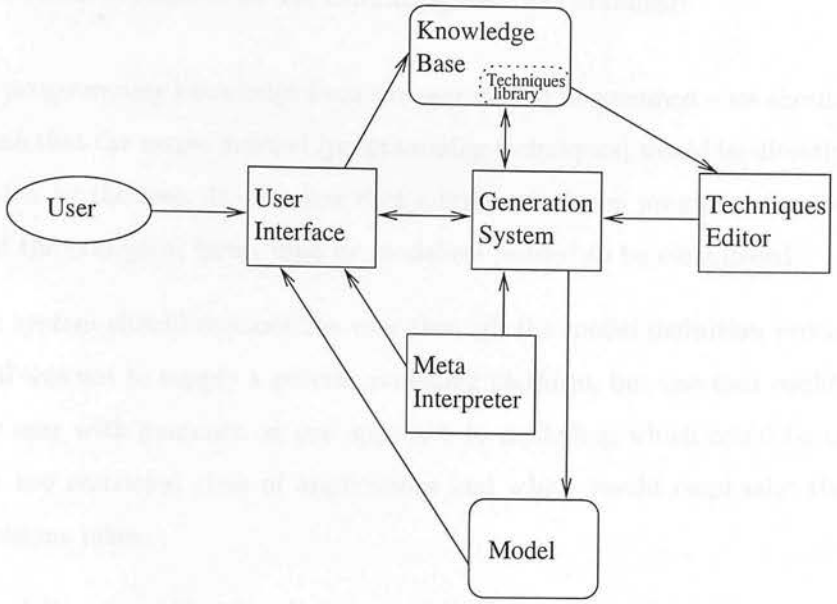

Figure 5.2: Modular structure of TeMS.

as seen by a user.

5.2.1 Design principles

The ideas about modelling that are embodied in TeMS were formed from the following activities:

- literature survey on the ecological modelling literature
- survey on tools supporting modelling currently available
- early formative evaluation including discussion sessions with a small group with expertise in the development of knowledge based systems and modelling

Another contribution to the conceptualisation of the system came because the author, who had some experience with modelling in a different area but was an absolute beginner in ecological modelling questions, since the beginning of the project had tried to gather material and resources to improve his own learning on the area. The fact is that there are only a few structured resources available to support learning in modelling, specially in a complex domain like ecology.

As a result of the preceding activities and motivated by the opportunity to probe the use of Prolog programming techniques in this domain-specific context, a preliminary set of specification features for the defining system was obtained:

- no programming knowledge from the user should be assumed – we should not assume that the target method (programming techniques) would be directly manipulated by the user. It was clear that a linking between programming techniques and the ecological terms used by modellers needed to be established.
- the system should conduct the user through the model definition process – the goal was not to supply a generic modelling platform, but one that could provide the user with guidance on one approach to modelling which could be used in a not too restricted class of applications and which would emphasise the design decisions taken.
- as a follow up of the preceding item, animal population dynamics was chosen as the domain. That choice would allow us to work with models at different levels of complexity and to apply the underlying principles in different subdomains.
- a process-based approach was the notion we thought should be used because it would be easier, especially for novices, to associate with real-life phenomena. A library of predefined processes should be made available to the user, although the possibility of a new process being defined should be left open.
- the main focus should be a class of models in which the user would be able to deal with any number of species. It should be possible to represent populations disaggregated according to several factors or "dimensions".
- the interface, which plays an important role due to interaction with the user at domain level, should supply resources for the user to run the model generated and somehow follow the simulation.

All these features have been implemented in TeMS, although most of them have been adjusted through sessions of formative evaluation.

5.2.2 Generation system

The generation system is the core module of TeMS. It has two distinct yet connected purposes: First, it conducts the problem description phase of our modelling approach. It uses predefined parameters (see Section 5.2.4) to put, through the user-interface (Section 5.2.6), a series of design decisions to the user. After each answer, new predicates are asserted into the knowledge base and the next step is selected and presented to the user.

The second purpose of this module concerns the composition and execution of a set of data-structures used to set up and trigger the program construction. The following is a description of these data-structures and how they are defined and used by this module.

Following the end of each process description, two sorts of data-structures are added to the knowledge base: (1) those used to compute population values at each time point of the simulation (Phase III of our modelling process); and (2) those used to set up program construction, used at Phase II of that process.

The two predicates on the first group are `selected_proc/5`, which usually embodies the equation of each process and the dimensions of disaggregation it uses; and `var/4`, which describes the variables used by the processes. The meta-interpreter (Section 5.2.5) uses both structures to compute, at each time-point, the value of the equations defining a process. These structures are represented as:

`selected_proc(`C`,`CP`,`P`,`D`,`T`)`

where C is the name of the component whose population is affected by process P, CP is the process' category, T are the points on time where P is applicable and D indicates how the process is defined.

For equation-defined processes, D is on the form `f_(`E`,`V`)`; where E is the process' equation and V are the disaggregation dimensions[2] used in E.

`var(`V`,`Ex`,`S`,`E`)`

[2] See Section 5.3.2 for an explanation on disaggregated populations

where V is the name of the variable, Ex is an explanation for it, S indicates whether a variable is from a basic set (e.g. `time`) or used-defined (e.g. `eat`) and E is the equation defining the value of V.

The second group of data-structures are explained next.

Setting up program construction

In order to allow only meaningful predicates to be generated and to do that without user intervention, selection and application of techniques has to be restricted and some data-structures are defined to help in this task.

TeMS knows which predicates will be generated (and in which order) by using a data-structure named the *list of candidates*. Every "candidate" in this list must have associated to it information allowing the generation system to generate a Prolog predicate. This information is represented by two data-structures: the *code construction record* (ccr), which defines the sequence of techniques-editing operations needed to generate a defining predicate; and a set of *binding records*, controlling variable binding at each techniques-editing operation. A techniques-editing sequence within a *ccr* will only be "triggered" through a definition request from the list of candidates.

There are standard ways to add elements to the list of candidates. Some predicates are included by default (the generation system asserts them in the knowledge base), or when a configuration (represented by rules in the knowledge base) can be verified; others are included after the definition of a process during model description. The latter is performed by the algorithm *enter_process* (See Appendix A.1), defined as

$$enter\_process(C, P, CP, E)$$

where: C is the component name, P is the defining process, CP is the category of the defining process, and E is the process' equation.

We illustrate how the representation of an ecological process is included in the model using the following example: Consider the user wants to represent the process *predation*, which is an instance of the *mortality* category of processes, and she will use one of the predefined options (the one called `predation1`) from the knowledge base (see part

B of Appendix B). predation1 is defined by the equation

$$d = eat.P$$

where d is the number of deaths caused by a predator species of population P. Variable *eat* is a variable which represents the eating rate of an individual from the predator species. If we consider a model with two components – moose and wolf – in a predator-prey relationship (moose is the prey and wolf the predator), then *eat_moose* will be the number of moose each wolf eats during the period of time corresponding to one time-step of the simulation. This variable would be a function of the moose population; that is, *eat_moose* would increase or decrease according only to the availability of prey.

Section 5.3.2 explains the use and definition of processes in TeMS. Figure 5.4 in that section, shows a snapshot of a process definition in TeMS with the situation illustrated here. Note that instead of defining *eat_moose* as an mathematical equation, the user chose to define it by sketching a curve, which represents the functional response of *eat_moose* to moose population.

After the user has concluded her definition of the process, TeMS starts setting up program construction for the predicate representing that process in the model. The first step is to check whether process and variables have well-formed formulas. This is done by the meta-interpreter described on Section 5.2.5. Only then are the structures created.

A defining predicate is identified by the following elements: (*a*) a component; (*b*) a disaggregation criteria; (*c*) a definition type (equation, schema or technique); (*d*) a way of computing its effect over the population (according to which category its behaviour is defined – natality, mortality, progress on subclasses).

In our example, the following set of techniques (`T`) would be used: `[proc_head, def_comp, mortality]`. The technique `proc_head` defines the head and first subgoal (splitting the data-structure carrying out population values) of the defining predicate; `def_comp` ensures the process is applied only to the intended component; and `mortality` is the technique definition for that category when the process is defined by an equation and the component's population is disaggregated.

`TS` is the sequence of techniques-editing operations (explained on Section 5.2.3) for

defining a predicate. In this case, TS is:

```
[ unif_pred_name(proc_head,predation1,C1),
  add_technique(def_comp,C1,C2),
  add_technique(mortality,C2,C) ]
```

So the set-up structures for our example are:

```
ccr(predation1,
    [proc_head,def_comp,mortality],
    [unif_pred_name(proc_head,predation1,C1],
     add_technique(def_comp,C1,C2),
     add_technique(mortality,C2,C)],
    C).
bind(predation1,def_comp,[6,new_param(moose)]).
bind(predation1,mortality,[1,2,6,4,5,new_param(predation1),8,9]).
cl([(predation1,[proc_head,def_comp,mortality])]).
```

If we had considered a similar example, where the only difference is that the process would be applied on a non-disaggregated population, the following set of techniques would have been used [proc_head, def_comp, def_no_disaggregation, decr_process_by_formula] and the definitions of TS, ccr, bind and cl for the process would change accordingly.

Once all processes related to a component are defined and the structures for constructing the respective Prolog predicates have been set up, TeMS will define structures for constructing another predicate that chains all the processes for a component. This predicate (update/4) has one definition for each component of a model.

Setting up the construction of update/4 consists of initiating T with techniques proc_head and def_comp2; then adding to it calls for all the processes of a component; and completing the sequence with the technique adj_values, which adds a subgoal for checking consistency of population values.

An example of update/4, which employs techniques *chained composition* and *partition* (discussed on Section 4.3), is shown on Section 5.3.4. The algorithm for this procedure (see Appendix A.2), is defined as

$$compose\_update(C, P)$$

where: C is the component name; P is the set of processes for that component.

According to the choice of processes defined by the user, the predicate `update/4` will have a different sequence of goals, although the basic composition scheme stays the same. *compose_update*(C, P) is the algorithm for defining that predicate.

Any other process definition belonging to one of the categories defined in Section 5.3.2 may be added to the knowledge base without change in the algorithm. In the same way, new categories might have their *modus operandi* embodied in schemata and inserted in the knowledge base. More complex changes would require new ways of linking process definition with program generation, and might require us to redefine the algorithm. Although we have imposed some limitations to predefined processes and variables applied on disaggregated population, we believe the current implementation covers a sensible number of situations which might arise in population models as targeted by our framework (see Sections 5.4.1 and 5.4.2).

Executing program construction structures

Once the data-structures used to set up program construction are defined, all that is needed is to search the knowledge base for new candidates, get their ccrs and binding records, and execute them; that is, to apply an appropriate techniques editing sequence of each candidate in the list of candidates as well as to each new candidate found to be needed during that process.

5.2.3 Domain-dependent techniques editor

We use the techniques editor defined in [Castro 94] which is inspired by the step-enhancement methodology presented in [Kirschenbaum *et al.* 89] and uses the domain-specific structures discussed in Chapter 4. In this editor, techniques application is subdivided in three tasks. The following is a brief explanation of each one.

Initialisation

This is the first task which needs to be carried out for the definition of a new predicate. It finds in the knowledge base the skeleton indicated, checks its applicability and then uses it to compose the first piece of *working code* for the defining predicate. It is defined as

$$unif\_pred\_name(S, P, C)$$

where: S is the name for the skeleton, P is the name for the defining predicate, and C is the code produced.

Adjusting arguments

This operation extends the number of arguments in a working code. The new argument is included in the head of all unit clauses and into the head and body of all recursive clauses. In recursive clauses, there is the choice of including the same argument in the recursive subgoal, only propagating it through the recursion; or to include a new variable instead, which in most cases requires that we have a subgoal relating it with the correspondent argument in the head of the clause. The latter is far more usual, as can be seen in the example shown on Section 5.3.3. Note that at stage (a) of program construction (see Figure 5.5), the predicate `population` has only one argument (`T`). After the technique data_collection has been applied, another argument is included in the head and recursive subgoal of its clauses – stage (b). The new argument is `P` in the head of the clauses and `P1` in the recursive subgoal. The predicate `update_population/3` relates `P` to `P1`.

Appendix A.4 shows the algorithm for this operation (extending the number of arguments of a defining predicate), defined as

$$include\_argument(A, O, C, NewC)$$

where: A is the new argument's name, C is the working code, $NewC$ is the code after the introduction of the new argument and O is an option indicating whether the new argument must be the same in the head and body of recursive clauses.

Techniques application

This operation consists of making the changes specified by a technique upon a working code, which will produce a new version of it.

As we shall see on Section 5.2.2, the construction of a predicate is done through the execution of a sequence of techniques applications. However, the same technique might be used in different sequences and in each one of them the variables in its subgoals will combine with different variables of the working code. Hence, we use a structure to represent the bindings between the variables in a technique and those in the working code. There will be one *binding record* for each technique application.

TeMS generates binding records according to predefined *contexts* – that is, it has standard ways in which variable binding between the technique and the working code will occur. Consider for example, predicates representing the processes natality and mortality (e.g. `rep_rate/5` and `predation1/5` on the model presented on Section 4.2.3), essentially different in their function. They have part of their bodies formed by a similar set of subgoals, which is a result of applying certain technique sequences in the same context – so using the same pattern for the definition of binding records.

Thus, the first step on this operation is to find a binding record for a technique T in the current sequence, and then bind the variables in T to the variables in the working code. There might be cases where the technique will include new variables in the working code.

After the variable binding, the subgoals provided by the technique T can be added to the working code. In techniques editing, the usual procedure is to include subgoals only at the end of the body of each clause. In our case, there are situations[3] where we want to allow the insertion of subgoals also on the top of the body of a clause.

The algorithm for adding subgoals to the clauses of a defining predicate, shown in Appendix A.5, is defined as

$$add\_technique(T, C, NewC)$$

[3] Due mainly to procedural considerations (e.g. to obtain some side-effect by including a subgoal before a recursive call).

where: T is the technique to be applied, C is the working code and $NewC$ is the new code produced.

Section 5.2.2 shows how these operations are ordered and triggered according to the design decisions made by the user. Section 5.2.4 contains comments on the techniques library shown on Appendix B.

5.2.4 Knowledge-base

Appendix B shows TeMS' knowledge base (KB) at its initial state, that is, only with static predicates. It comprises three parts, which are explained next.

Predefined parameters

User interaction during the first phase of our modelling process employs several menus with selection lists or buttons. The values for these lists and buttons are represented on the KB by predicate `menu_s/2`, in which the first argument is the part of the model description to which those values refer and the second one is the list of values. For example, the predicate `menu_s('Attributes', [age, sex, weight, size, location])` lists predefined instances of attributes which may be used to describe the components of a model. In this case, values will be show as a selection list (see Figure 5.3).

As we explain in Section 5.3.2, attributes can be *ordered* or *unordered*. The predicate `dim_info/3` contains this information. Time structures are defined by predicate `time_hier/2`.

Each component c_i defined by `menu_s('Components',[`$c_1, c_2, ..., c_n$`])`, is categorised according to three classes, as represented by the predicate `component_sort/2`.

Finally, rules defining model configurations, which are used to define applicability of ecological processes are also included in the KB.

This initial set can be altered (manually) to suit different applications. Any of the preceding predicates might be modified to embody a different set of instances for those choices in the phase I (model definition) of our modelling process.

Predefined processes

This part of the KB contains definitions of processes that can be presented to the user, given her previous choices during Phase I (model description) of our modelling process. These processes are represented by the predicate `eco_process_app/7` which is an iterative clause with the body representing the applicability of the process definition in the head, as follows:

$$\texttt{eco\_process\_app}(C,CP,P,D,E,V,EP)\ \texttt{:-}\ A$$

where C is the name of the component whose population is affected by the process, CP is the process' category, P is the name of the process, D is a description of the process that will be shown to the user, E is the process' equation, V is the list of variables which must be defined by the user and EP is the name of a technique representing the effect that this process will have of the population of C. A is a set of subgoals which constitute the application criteria for P.

The following is an example representing an instance of *natality*. In this case, we wanted to represent a situation where the rate of population increase must diminish until the numbers reach a saturation point at which no further increase is possible. A well-known model of such population growth is the logistic or Verhulst-Pearl curve[Pearl 39]. The rate of increase in the population at a certain time t is defined by the equation

$$\frac{\mathrm{d}N}{\mathrm{d}t} = r_{max}N(1 - \frac{N}{K})$$

where r_{max} is the innate capacity for increase, N is the number of individuals present and K is the number the habitat can hold at saturation.

This is represented as

```
eco_process_app(Component, natality, logistic_rep_rate,
   'The number of births is computed using an individual reproduction rate defined by
    the function r_max(1-pop/pop_max). \n Logistic growth is based on this function.',
   'max_rep_rate_Component*(1-Component/max_pop_Component)*Component',
   [ [max_rep_rate_Component,
       ['Maximum value for individual reproductive rate of Component']],
```

```
    [max_pop_Component,
      ['Maximum value for Component population']]],
  incr_process_by_formula)  :-
                              \+ disaggregated(Component).
```

The Prolog variable `Component` will be instantiated to every component name whose population is not disaggregated. This instantiation will be used to rewrite the equation so that it might be read by the meta-interpreter. For example, if there is a component named `moose` which has a non-disaggregated population, the preceding equation will be modified and asserted as `max_rep_rate_moose*(1-moose/max_pop_moose)*moose)`, where `max_rep_rate_moose` and `max_pop_moose` will be informed by the user and `moose` will be interpreted as the population of that component in a point in time.

Techniques library

The last part of the knowledge base contains the representation of those design-specific structures discussed on Section 4.3 (techniques and schemata) as well as other structures needed for techniques editing (triggers, ccrs and binding records).

There are two components that define a technique: the piece of Prolog code that it adds to a defining predicate, and the editing operations which allows the system to do so. The first component is represented by the predicate `technique/2`, which in TeMS is also used for representing schemata. This homogeneity is possible because each schema needs only a reference name and the piece of Prolog code it will contribute to the defining program.

There are at least two advantages of using the same representation for both structures. First, the generation system can deal with techniques and schemata in the same way, it takes schemata as a simpler case of techniques (one that does not have the second component mentioned in the preceding paragraph). Second, it might make it easier at some later date to implement a *meta-editor* to include new techniques and schemata in the KB.

Thus, techniques and schemata are represented as

`technique(`N`,[`V`,`D`])`

where N is the name of the technique or schema, V is a list of variables (other than the arguments in the heads of the clauses in D), and D is a set of one or more clauses to be added to the defining predicate. A clause C comprises the head H and the body B, which is a list of subgoals; or, in case of clauses with recursive calls, C is represented as `rec(`H`,`B`)`. Schemata have the list V always empty and the name in the head of the clause always ground.

Sequences of techniques and schemata are defined by the predicate `trigger/2`. Code construction records (`ccr/5`) and variable binding records (`bind/3`) are also here.

During set-up of program generation, dynamic versions of `trigger`, `ccr` and `bind` are asserted to reflect the user's choice of parameters.

5.2.5 Meta-interpreter

A *meta-interpreter* is an interpreter for a language **L** which is written in **L** and, if it can be used to interpret itself, it is also called a meta-circular interpreter.

Many meta-interpreters can be written for a programming language. However, when a language has the same form for programs and data, meta-interpreting is quite natural. This is the case with Prolog, and the literature contains numerous examples reflecting that [Shoham 94, O'Keefe 90, Sterling & Shapiro 94].

For our application, a meta-interpreter with a fine granularity[4] was used to perform the tasks of *inspecting* and *executing* the formulae used to represent magnitudes and proportions in a model. It is defined as

$$\text{check}(F, R)$$

where F is a formula to some variable in a model and R is its value (at certain time point) during simulation. The algorithm for this meta-interpreter is on Appendix A.3.

It is during model description that check(F, R) examines whether a formula is well-formed, that is, if a formula complies with a previously defined syntax. In addition to

[4] See [Sterling & Beer 89] for a discussion of granularity of meta-interpreters

the standard Prolog syntax for arithmetic expressions, this interpreter has to deal with if-then-else expressions; graphically defined values, denoted by the function graph(X); and variables that must be valid at the point in the definition where the user is typing the expression, which also may include indexed variables[5]. During this phase, R is not instantiated.

When using the model (Phase III on Figure 5.1), check(F, R) traverses the formula in the same way it does when inspecting, but it also substitutes each term by its correspondent values and computes the value for the whole expression. At each time point in the simulation, R will be instantiated to the current value of F.

5.2.6 User interface

Given the need for tight control over the user's operations to avoid mistakes and unnecessary operations, a simple interface was designed. It consists of menus, buttons and text-input windows sequenced according to the values entered at each step of the interaction. There are auxiliary windows to provide, for example, textual explanations of options in a menu or a graphical (tree-like) representation of how the population is disaggregated at some point in the process.

Control of the dialogue is divided between system and user. The user maintains the control during input or auxiliary tasks but the system maintains the control during all of the generation phase and when the model is running.

A better account of how the system interacts with the user is given on the next Section.

5.3 Implementation

TeMS was implemented using a common Prolog system (SICStus 3#5) and a standard scripting package (TclTk 7.6) both running on Unix platforms.

SICStus was chosen because it is a reliable and fairly flexible Prolog interpreter and is supported at our local computing laboratory. Versions 3 and newer have a library

[5] The dimensions of population disaggregation (Section 5.3.2) are also used to index for variables

allowing incorporation of TclTk scripts from a SICStus session.

5.3.1 System overview

When surveying programming aids to simulation, [Haefner 96] divided them into three classes: (a) libraries of functions to be used from some programming language, (b) special-purpose languages, and (c) simulation environments that provide a language and a GUI for editing parameters and viewing simulation results. The latter would be the more suitable to place TeMS if we were considering it only as a "simulation facilitator". However, we must have in mind that TeMS is also a modelling facilitator and it has features that are not found in general-purpose simulation environments.

Stella[Stella 98], ModelMaker[ModelMaker 98] and AME[Muetzelfeldt & Taylor 97] are among the general-purpose environments used by modellers. All these environments are based on the Systems Dynamics paradigm and use Forrester diagrams[Forrester 61] to build and represent a simulation model. A first requirement to use such tools is that the user *must* know the paradigm, so there is little or no guidance on how to represent a model without this knowledge. Second, the model has to be fully devised in advance, general-purpose tools cannot ask the user for parts of the model which are *not* present. Third, general-purpose tools may not have direct means of manipulating structures typical from some domains, like disaggregation of population or *location* of individuals or groups of individuals in a eco-system.

TeMS addresses these issues at the price of being less general than systems dynamics based tools and its modelling paradigm is more appropriated to be used when a closer assistance to the various steps of the process is required (e.g. learning).

5.3.2 Describing the model

In this phase the user outlines the model's structure. The user's task consists of selecting options from menus and answering prompted questions according to characteristics of the system she wants to model. The set of questions and standard options presented to the user are stored as predicates in a knowledge base.

After each answer is given (either by selecting one among a set of options, typing some value or sketching a graph), the contents of the knowledge base will be extended and used to define the next step on this elicitation process, that is, the next menu will be presented or a question will be prompted.

The menus and questions presented to the user cover the following elements of the model: components, organisation, attributes, time-references, initial-data and ecological processes. The result of this phase is a knowledge-base "tuned" to a specific model, which will be used on the next phase. Those elements are described next.

Components

When defining a model on population dynamics it is essential to know what the *components* whose population behaviour we will observe during the simulation are. A user can either select among a list of typical components or introduce a new one.

If the latter, the user must also categorise the new component according to categories used by biologists when talking about the animal kingdom. Thus, every component is associated with one element in each of the following sets: {*herbivore, carnivore, omnivore*}, {*vertebrate, invertebrate*} and {*insect, bird, fish, mammal, other*}. This is not the only category structure possible. The sets were chosen based on information given by experts and illustrate the sort classification theory that can be used when describing components in a model.

Such knowledge may later be used to infer relationships between components. The following, for example, is a rule in the knowledge base expressing a predator-prey relationship between two components of a model: Two different elements are taken from the list of components, if one of them is a "carnivore" or "omnivore", the other might be subject to predation. Rules like this are used to select which predefined processes will be suggested to the user, but these are a guide only. Hence, it is not a requirement that the ecological knowledge base should be complete or consistent with these guidelines rules and users may override them.

```
would_be_predator(A, B) :-
  components(L),
  member(A,L),
  member(B,L),
  \+ A = B,
  ( component_sort(A,carnivore)
    ; component_sort(A,omnivore) ).
```

Organisation

The population in a model is usually categorised according to its components. However, each component may have its own way of referring to the elements which constitute its population. The elementary unit in a population may be an individual or, more commonly, a group of individuals which have common attributes.

These groups are defined by the combination of the dimensions according to which a population is *disaggregated.* Those *dimensions of classification* are attributes of a component as shown in the next section. There will be as many sub-populations as there are elements in the Cartesian product of the attributes. The size of a *non-disaggregated* population, on the other hand, is represented by a simple number. These different representations make disaggregation a crucial selective factor in the way population values are computed, as seen in technique *traverse_pop* for example (Section 4.3).

Attributes

Every component may have a set of *attributes* associated with it. Attributes are used to define characteristics which are meaningful to a modeller when talking about a component. Attributes also are important to represent properties which should only apply to elements of certain categories.

Attributes can be inherent in components (e.g. age, sex, weight and size) or they may represent a quality that was for some reason "associated" to it (e.g. location). Attributes' values also can be constant (e.g. sex), or variable over time (e.g. size, location).

Finally, if we consider an attribute as defining a *category*, then its values would define *sub-categories* according to which we might want to represent the dispersal of individuals of a population. Attributes can also be divided in two sorts: *ordered* and *unordered*. Ordered are those attributes in which subclasses are ordered according to some criteria and individuals move to the next subclass always in one "direction" (e.g. age); unordered are those with no predetermined order and so individuals can move to any direction (e.g. location).

Figure 5.3 shows a "snapshot" of model description using TeMS. The window in front shows the user suppling the information that wolf population will be disaggregated by *sex* ($\{male, female\}$) and *age* ($\{pup, adult\}$) classes.

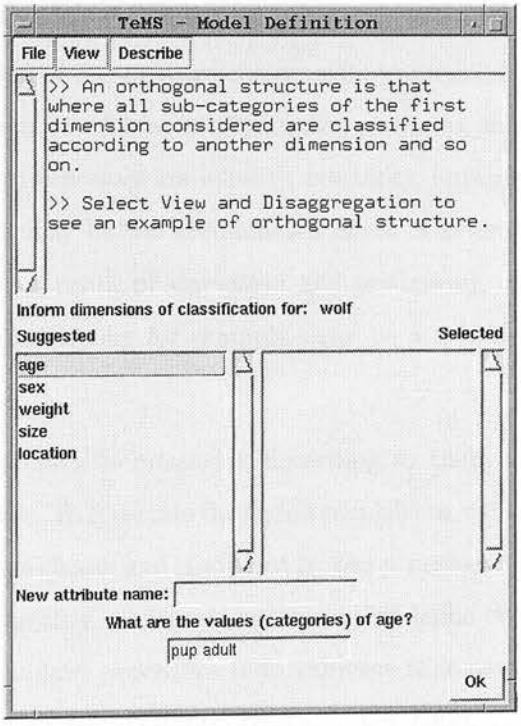

Figure 5.3: Attribute definition in TeMS - the user is filling in values for an age class disaggregation.

Initial data

Once the dimensions of disaggregation are known, the system can prompt the user for the initial values of population that the simulation will start with. If the population is not disaggregated, a simple value is asked for. If the component has a disaggregated population, the system will ask as many values as the number of sub-populations considered. The actual order in which the values are asked depends on the order in which the attributes are given by the user.

Processes

Among the aspects of reality that a model represents, there are actions or operations affecting the population of one or more components, responsible either for increasing, decreasing, or rearranging it. Those actions, called *processes*, must be mirrored during the simulation. Typical processes are natality, mortality, immigration, emigration and ageing. Some of them may be the accumulated effect of several processes (mortality for example, might be a result of starvation and predation), others are instances of more general definitions (ageing for example, may be a specific sort of "progress on subclasses").

We have grouped processes in *categories*, according to their common effect upon a component's population. To illustrate those different effects, consider that a population is disaggregated into age classes and is affected by two processes: *mortality* and *natality*. When representing mortality, whatever way is used to define the number of deaths in each age-class, the standard procedure is to subtract that amount from the current population on each age-class. Natality, on the other hand, requires the number of births to be computed at every age-class and to be added only to the first age-class.

The processes used by TeMS are categorised as follows: *natality, mortality, progress on subclasses*[6] and *migration*. As pointed out in [Solomon 76], reproduction (natality) and mortality are the main references for most studies of population dynamics, which is primarily concerned with the knowledge of how many individuals make up

[6] When a population is disaggregated in *ordered* classes (see Section 5.3.2), it represents the movement of elements from a subclass to the following one (e.g. ageing, growing in size)

the population. However, it is often the case that modellers need also to know how the population will change within its dimensions of disaggregation (progress on sub-classes) or will spread throughout its habitat (migration). Thus, the categories used in TeMS include the main patterns in which a process affects a population, this makes it adequate for most models in this domain.

For each component in the model, the system prompts the user for the processes affecting the size of the population of that component. She must either choose one of the predefined processes shown (the system infers from the knowledge-base which processes may apply) or define a new one.

A process is defined by an equation which may include user-defined variables (e.g. individual reproductive rate, rate of predation). An equation is either an arithmetic expression, an if-then-else declaration or a two-dimensional graphical function. If the latter is used, the graph is represented as a set of equations of the sort $y = ax + b$ and the actual value of a variable is obtained by an interpolation within the corresponding interval.

A predefined process consists of a standard equation along with a set of parameters which must be given values and, optionally, a condition in which that process may apply. For example, the process predation1 is an instance of mortality where the population of a prey species (P) diminishes due to a number of deaths d (P' being the updated value of P). The number of deaths in its turn is the product of the current population of the predator *Pred* by the value of the variable (*eat*), which represents the number of individuals from P that one individual from *Pred* will prey on during the time corresponding to one time-step of the simulation. That is:

$$P' = P - d \quad \text{and} \quad d = eat.Pred$$

The condition `would_be_predator(X,C)` (`C` instantiated to the component's name) completes the definition of that process. In the case of predation1 being selected, the user would be requested to supply a value (or equation) for *eat*.

Figure 5.4 shows the selection of predefined process predation1 where the user is defining the value of eat_moose[7] by sketching a graph of it as a function of moose population

[7] In the situation illustrated by Figure 5.4, *moose* and *wolves* are the only components in a predator-

size.

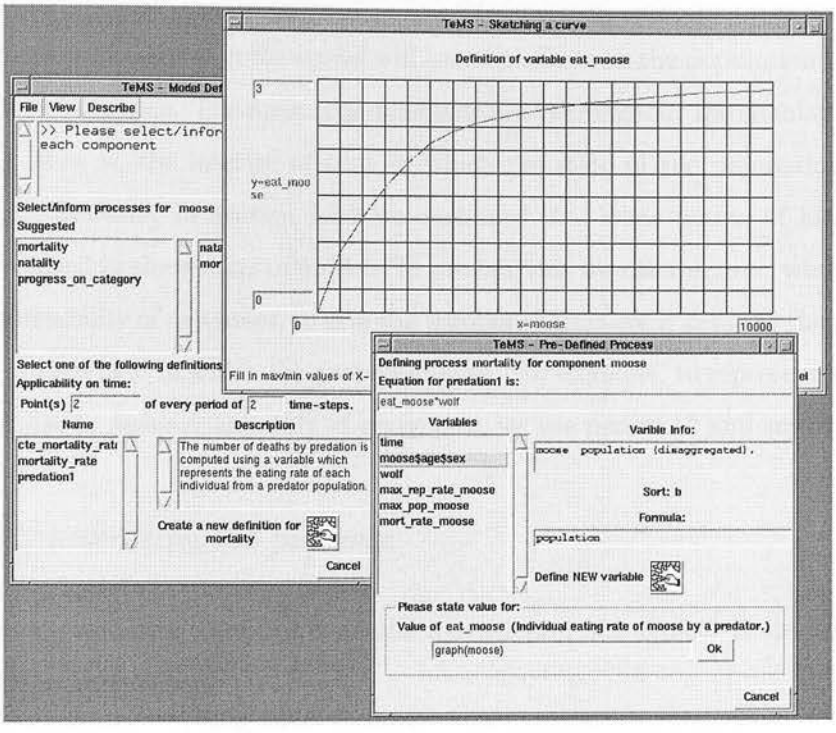

Figure 5.4: Process definition in TeMS - user is representing the process of predation, informing a term used in the process' equation through a sketched curve.

TeMS allows the user to represent relationships by sketching a curve between two terms involved. This resource have been proven to be quite useful, given modellers often know in qualitative terms how the relationship should be (the shape of the resulting curve) but not the equation defining it. On the situation shown on Figure 5.4, `eat_moose` is a term defined in function of moose population, which is indicated by the declaration `graph(moose)`.

TeMS uses a simple interpretative scheme of interpolation to map the curve sketched into expressions that can be processed by a Prolog interpreter. [Cheng *et al.* 98] and [Willoughby 91] discuss sketching tools/resources in detail.

---

prey model

Time Reference

Every process represented in the model will have its effects on the population according to some time reference. The main time-reference used throughout the simulation is the time-step, that is, the interval of time in which the state of the population will be calculated. However, in Section 4.2.2 we explained that some notion of hierarchical time is required in these sorts of model. To provide this we ask the user, when talking about applicability of processes, to give the number of time-steps defining the period T and the subsets of T in which the process applies. For example, to represent a process which applies in January and July of every year, we use period 12 and subset {1 7}.

5.3.3 Constructing the program

The following are some signposts from the construction of a typical model within the framework presented here.

- The "entrance predicate" is by default, the first on the list of candidates for construction. The predicate `data_collection/1` is used as a "flag" (initially set *on* by the generation system). If this flag is *on*, the ancillary procedures (see Section 4.3) will be used to produce:

```
model(T,P) :-
  open_files,
  population(T,P),
  close_files.
```

  Predicates `open_files/0` and `close_files/0` will be defined by schemata and predicate `population/2` is the core predicate for the simulation and will involve some techniques-editing operations.

- The standard way we build the core predicate (see Section 4.2.3) is: We start by taking the technique *recursion over time points* as the main flow of control, then apply technique *population updating* and, if `data_collection(yes)` can be proven, technique *data_collection*, which adds subgoals to record population values at each time-step of the simulation, will also be applied. Upon completion, this

procedure asserts in the knowledge base the predicate's name and arity along with proper ccr and binding records.

Figure 5.5 shows the stages in the execution of the techniques-editing sequence on this predicate's ccr. Stage (a) is the initial state of the working code after initialisation using *recursion over time points* as the skeleton. Stage (b) shows the working code after application of technique *population updating* (note an argument was included in the defining predicate). Stage (c) is the final state of the working code after application of technique *data_collection.*

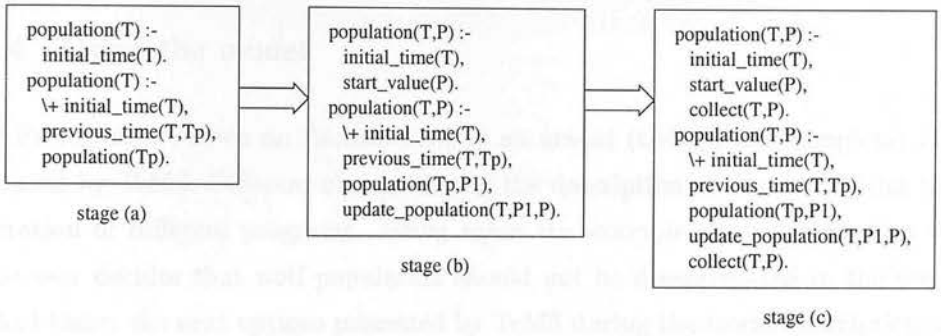

Figure 5.5: Techniques editing in action - three stages in the construction of predicate `population/2`.

- At each technique application when defining the predicate above, new elements are included in the list of candidates (CL) corresponding to those new goals added to the body of the defining predicate. These are `initial_time/1`, `start_value/1`, `collect/2` and `update_population/3`.

- Some of the new elements in the list of candidates are standard in the domain and can be added to the program by schemata.

- Other elements have to be defined through new techniques application. Two examples of the latter were shown in Section 4.3: The first is the predicate `initial_value/2` (included in CL when defining `initial_time/1`), which uses techniques *traverse_pop* and *get_value.* The second is one of the definitions for predicate `update/4` (included in CL when defining `update_population/3`). Note that

there must be one definition of `update/4` for each component in a model whereas other predicates must be defined only once – the generation system uses the list of candidates to assure that.

Finally, the user can run the model generated on the previous phase. The code generated is recorded and loaded into a current Prolog session. A data-file is generated from each simulation, with population values for each component being recorded at every time step. An external Unix package is used to generate a graph.

5.3.4 Using the model

The Prolog code shown on Section 4.2.3 is an actual (though not complete) listing produced by TeMS. Different choices during the description of the model lead to the generation of different programs. Using again the example shown on Section 4.2.3, if the user decides that wolf population should not be disaggregated in the way described there, the next options presented by TeMS during the model description stage would suppress those which are typical of disaggregated populations (e.g. *ageing*) and the code generated would reflect that. Some predicates might be maintained despite the change in population disaggregation (e.g. `rep_rate/4`), that is so because predefined processes are supplied for both disaggregated and non-disaggregated populations and their respective formulas are dealt with by TeMS' Prolog meta-interpreter. The following is the version of `update/4` – the predicate which determines the update of a subpopulation in a time-point (see example on Section 4.2.3) – for the case we described here.

```
% updates population of each component
update(T, Pop, Pp, SP) :-
  Pp=[C,_,SPp],
  C=moose,
  rep_rate(T, Pop, Pp, SPp, H),
  predation1(T, Pop, Pp, H, G),
  hunting(T, Pop, Pp, G, F),
  ageing(T, Pop, Pp, F, E),
  adj_values(E, SP).
update(T, Pop, Pp, SP) :-
  Pp=[SPp,_,H],
```

```
SPp=wolf,
rep_rate(T, Pop, Pp, H, G),
starvation(T, Pop, Pp, G, F),
adj_values(E, SP).
```

The runnable Prolog code for the model defined (except for some utilities) is viewable within TeMS and Figure 5.6 shows how the user see the results of a simulation plotted in a graph. The model represented there is the same described in Section 4.2.

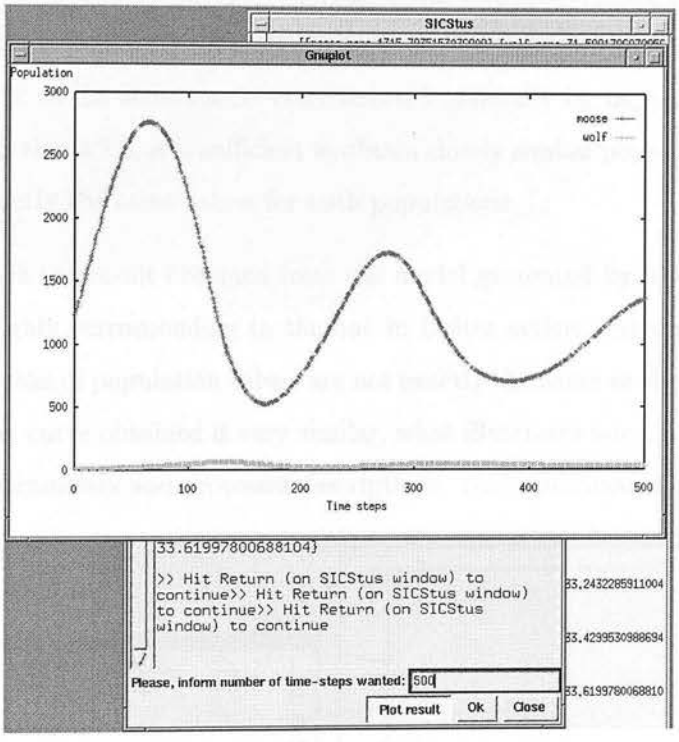

Figure 5.6: Running the model - population curve is plotted using values at each time-step.

5.4 Discussion

Having shown how our views on domain specific synthesis were amalgamated and implemented in a modelling tool, which demonstrated their feasibility, we next address its validation and discuss its scope and limitations.

5.4.1 Validation

Prior to the more thorough and qualitative evaluation described in Chapter 5, we wanted to see whether TeMS could be used to produce a model similar to the one from the literature which we reconstructed during the first stage of this project (Section 4.2).

Figure 5.7 shows two resulting population curves for the same model, the one described in [Crête *et al.* 81], with the process *hunting* being introduced after 360 years. The graph at the left side is a reproduction from Crête's article and the one at the right side is the result of the same model reconstructed manually by us, using Prolog. As explained on Section 4.2.3, it is sufficient to obtain closely similar population behaviour rather than exactly the same values for both populations.

Figure 5.8 shows the result obtained from the model generated by TeMS based on a description roughly corresponding to the one in Crêtes article. Again, although the high and low peaks of population values are not exactly the same as the previous ones, the shape of the curve obtained is very similar, what illustrates our point that given a similar set of parameters and processes descriptions, TeMS produces a similar model.

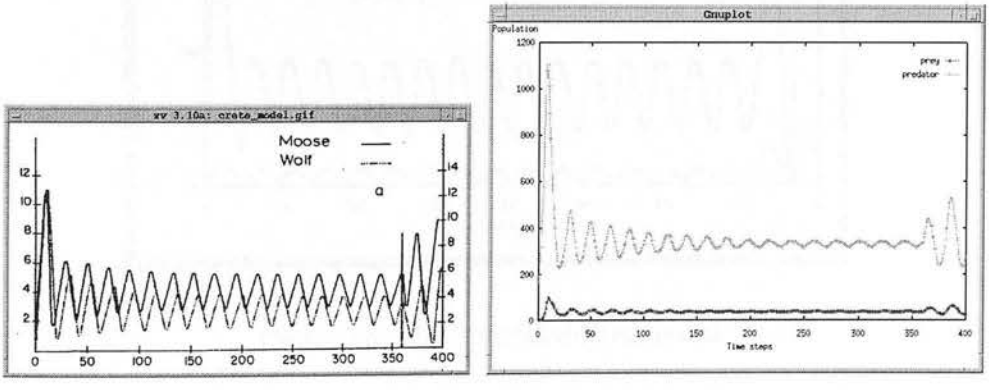

Figure 5.7: Results of two human-constructed versions of the same model.

Another experiment we carried out to gather evidence of TeMS validity as well as to investigate the class of problems it may tackle, consisted of using TeMS to implement population dynamics models that are used as modelling exercises in undergraduate ecology modules [Muetzelfeldt 96].

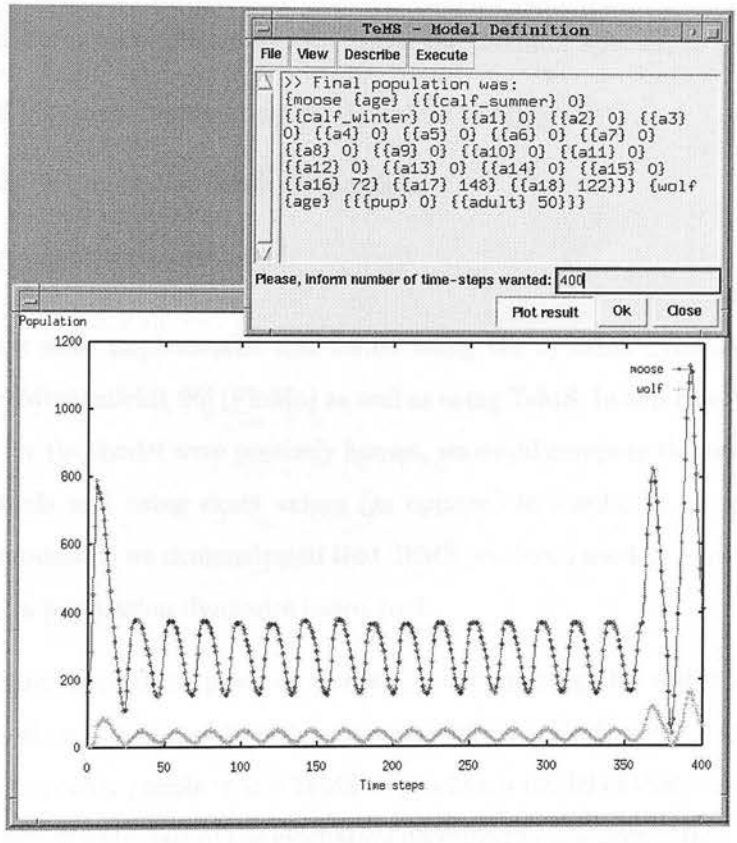

Figure 5.8: TeMS reconstructed model.

The basic intention of the task is to predict how the population size of a particular species (rabbit) will change over time. By successive refinements, a minimum of five versions of the model have to be built as a student goes through the following steps:

1. To implement a one-species model (rabbit population) with constant specific rates of reproduction and mortality;
2. to change the reproductive process in [1] to density-dependent;
3. to add the process *predation* with *fox* as the predator species, to [2];
4. to add processes representing fox dynamics to [3];
5. to make fox mortality density-dependent;
6. to implement variations of [5].

These models were implemented and tested using the systems dynamics based tool described in [Muetzelfeldt 96] (FloMo) as well as using TeMS. In this case, given that all parameters for the model were precisely known, we could compare the results obtained from both tools and using exact values (as opposed to similarity on the pattern of population modelled) we demonstrated that TeMS produced models *equivalent* to those produced by a population dynamics based tool.

Finally, another important piece of evidence corroborating the validity of TeMS in model synthesis was obtained by using a scenario of an ecological situation (Appendix C.1) and asking other people to use TeMS to produce a model of that scenario. During these trials, which were part of the evaluation described in Chapter 5, people (especially those who usually have expectations when describing a model) thought that the models produced were in fact coherent with the design decisions (specifications) made.

5.4.2 Scope and limitations

TeMS addresses the need for explicitness in the modelling process and the goal of producing programs without user intervention in the generation process. To do this it uses a process-centred approach, tight control mechanisms and the use of domain-specific data-structures. The space of models that TeMS can generate is described in

terms of the domain, rather than more abstract mathematical concepts. The general scope in which it can be used comprises population dynamics models with any number of interacting populations, a hierarchically disaggregated population structure, and process definitions controlling the interaction between populations.

On Sections 5.3.1 and 5.4.1, we contrast TeMS with systems dynamics tools. If on the one hand TeMS is obviously less general, on the other hand it has advantages that make it suitable to a number of applications. Section 6.3.3 also discusses this issue using results from the evaluation.

Limitations to the general scope stated at the beginning of this section mean that TeMS does not accommodate some models. These limitations are due mainly to assumptions we have made when designing the tool, but since they would affect only a small number of cases, which were not essential for our purposes, we decided to leave them aside. The following are examples of such issues.

- limitations of our assumptions of the predation process – the function *eat* represents the ability or disposition of predators to eat with respect only to a specific sub-class of prey, that is, the effectiveness of eating something considering simply the amount of that specific thing. By comparison, one may have one's own "preference list" of food types (e.g. in decreasing order: fish, bread, meat, fruits, cookies, vegetables, cereal) when a wide range of food is available. In such circumstances there may be a specific (small) disposition to eat, say, cereal; but when there are no other kind of food available, there could be a huge "disposition" to eat cereal.

  Making an analogy with the approach we used to define *eat* with what we described above, it would be like considering cereal be the only food someone eats, then *eat* would depend simply upon the availability of cereal, that is, no considerations about availability of a preferred kind of food were taken in account.

  We knew there was a "proportion" issue not dealt with there, but since that would be required only for a higher level of sophistication of modelling, it was left aside.

- processes of the same class may be not homogeneous – To illustrate this point,

consider ageing and growing (currently part of the same category of processes). Age classes may be of different sizes but essentially are shifted in one direction with no intervention of any external factors, but simply time. Size classes are also considered only one-direction movement, but in more sophisticated models they may have different speed of growing according to external factors such as temperature, competition, season, etc. Currently, only *time* determines progress on subclasses. It might be made more general by depending on other variables, but that would require more information to be added to the predefined processes and more features added to the interface.

5.4.3 Concluding Remarks

In this chapter we have described a modelling environment which uses a problem description language and allows the construction of a model from this using our chosen formal method.

With the techniques from Section 4.3 in mind, we provided a problem description language which uses concepts from population dynamics, and constructed an interface which allows these concepts to be supplied.

We have built an automated system which ensures appropriate parameterisation of the domain-specific techniques based on the user-defined population dynamics problem description. This connects the problem description to the techniques needed for model generation.

These parts were amalgamated as a modelling tool named TeMS which also has ability to execute the specification in styles familiar to those in the domain. This gives those in the domain an opportunity to check whether the model they have received is the one they expected.

TeMS can automatically generate a model from a problem description. It can be automatic because we employ restricted languages for both problem description and techniques. The space of models that TeMS can generate is described in terms of the domain, rather than more abstract mathematical limitations, so modellers can easily recognise the class of problems it can deal with.

In the next chapter, we address the effectiveness of TeMS, and the ideas embodied in it, through an evaluation by expert users.

Chapter 6

Evaluation

6.1 Introduction

In Artificial Intelligence, the systems produced are often not only a means of investigation but also the embodiment of what was accomplished. However, simply delivering a piece of software is seldom felt to be enough and, as in every other area of research, we wish to present further evidence, which could be attainable through *evaluation* of the results.

The term evaluation is used with a variety of meanings, according to the perspective of each author. A widely used definition is "to make a decision about the significance, value or quality of something, based on a careful study of its good and bad features"[Sinclair 92]. There are several other terms expressing similar concepts which may vary according to the context in which they are used. [SYDPOL 90] for example, when referring to some specific class of AI systems, list the following terms:

evaluation is the act of measuring quality characteristics, i.e. addressing a question "How good or bad?"

validation is a comparison of quality measures with a frame of reference. Validation means deciding on quality, answering a question "Is it good enough?"

verification is the act of checking correctness according to specifications. Question:

"Is the object built right?"

qualification is an assessment of appropriateness or validity in a given context

valuation is an assignment of value or worth by human beings to phenomena, i.e. a subjective assessment as in "How do you value being cured of your arthritis?"

It has been argued that researchers in the area of Artificial Intelligence have often neglected evaluation of the systems produced by them. Most publications give a detailed description of the system's architecture in order to provide sufficient information for others to reconstruct or at least understand the system, but only few include a carefully designed evaluation stage. [Siemer & Angelides 98] argue that the common approach in AI is the "develop-test-review-throwaway" sequence, which does not include an evaluation step. This may be due to the fact that in most of the cases in AI, systems do not usually deal with problems for which there is simply a right or wrong answer and as a result it is not easy to demonstrate that a system produces "correct" results, let alone to evaluate how good or bad it is[Gaschnig *et al.* 83].

Due to the idiosyncrasies of AI systems, there is no single and global methodology to validate them. But validation may also be carried out by means of a combination of evaluation methods and techniques, especially involving *user evaluation* of the system[Lindgaard 94, Cohen 95, Ruddock 81, Draper *et al.* 96]. Users should test a system's competence in their domain of expertise and determine whether it produces meaningful results as well as assess their interaction with the system. A further benefit of having outside users who interactively test a system is the fact that their judgements can be obtained as to whether the right problem has been addressed, because as stressed in [Gaschnig *et al.* 83] the question whether the system is actually going to be used is not the same as whether it fulfils the subtask selected by its designers correctly.

Hence, AI researchers ought to assess the usefulness of their systems as to their strengths and shortcomings. Evaluation instruments help to determine the extent to which a system meets certain requirements and to confirm their research value, propelling research developments by providing suggestions for the overall improvement of the architecture and the behaviour of the system.

6.1.1 Guidelines

AI systems are evaluated primarily in order to test program accuracy and utility and since there is no single and general framework to do so, it is necessary to ascertain the main issues involved and to use the experience of related areas. The following are some basic considerations collected from reported work in evaluation:

- the first point that should be made explicit when designing an evaluation is about its purpose. One should be aware of who it is for, exactly what is being evaluated, and what one hopes to gain from the experiment.
- another issue which must be dealt with before the experiment is the contextual description of the system to be evaluated. One should be able to make clear points such as what the system's objectives are, what will be its role and what is intended to be gained by using the system. Only after that is it possible to take into consideration questions of how good (or bad) a system should be.
- another important point is the question of defining realistic standards of performance of the system against a proper "gold standard".
- [Jackson 90] enumerates three preconditions for evaluation to be meaningful, namely: there must be a clear criteria for success, proper experimental procedures must be followed, and evaluation should be done painstakingly or not at all.
- [SYDPOL 90] list some aspects in an AI system that should be evaluated: the knowledge used in the system, inference mechanisms, user interface and human-machine interaction, and its environmental impacts.
- [Gaschnig *et al.* 83] remind us that complex objects or processes cannot be evaluated by a single criterion or number and people will disagree about the relative significance of various criteria according to their respective interests.
- Finally, some other thoughts about research evaluation in AI
  - The main notion relating to evaluation is to allow us to know if something went wrong and *why*;

- Often AI researchers do not know in advance what a system will look like until it is really finished. [Wyatt & Spiegelhalter 90] draws an interesting analogy between the development of an AI system and that of a drug. He states that neither can be tested thoroughly before they are ready for release;
- It may be very difficult to define the "success" of some systems;
- User evaluation is problematic. One reason is that it may require us to measure changes in attitude, and it is not easy to determine if some method or system is easier, more enjoyable or more reliable than other. A system might allow the users to do something they never could do before and in this case references for comparison would be even fuzzier.

Much effort has recently been spent in the topic of evaluation in AI, as it has long been the case in the field of Social Research. Interesting topics are usability testing[Lindgaard 94, Robson 92, McGraw 92]; assessment of the relationship between a system's architecture and its behaviour[Mark & Greer 93, Shute & Regian 93, Winne 93]; the impact of using a system upon some audience[Siemer & Angelides 98, Ruddock 81] and of course, design and instruments for evaluation largely used in Social Research [Dooley 95, Oppenhein 92, CTI 98, EKSL 98, SSSS 98].

Some sorts of evaluation

With respect to system development, two most frequently mentioned types of evaluation are *formative* and *summative*. Formative evaluations normally occur during design and early development of a project; they have an internal control condition and are concerned with how the system can be made better. Hence they aim to detect the existence of a problem as well as its possible solutions. By contrast, summative evaluations have an external control condition and are concerned with how a system compares with other systems or approaches and for that reason, they may be linked to the making of formal claims (usually related to system goals) about a system.

Another issue is whether an evaluation deals with *quantitative* or *qualitative* data and methods. The first approach dominated the mainstream view of empirical studies in the Social Sciences and uses numerical methods (especially statistical methods) to

measure the effects of manipulating one variable on another variable, usually testing hypotheses. Qualitative methods rely on analysis of narratives, personal accounts and other collections of words to reflect upon a number of variables in order to discover regularities in a context dependent fashion. The tension between these two viewpoints is analogous to the debate in Sociology between positivism and naturalism[Robson 93]. [Parlett & Hamilton 77] introduced the term "illuminative evaluation" to denote an observational approach inspired by ethnographic rather than experimental traditions and methods. It focuses on qualitative methods, inductive analysis and naturalistic inquiry. One of its aims is to discover, not how something performs on standard measures, but what factors and issues are important to the participants in that particular situation, or which seem evidently crucial to a closer observer.

6.1.2 Evaluation goals

We expected TeMS would **contribute to**:

- clearer, more structured, standardised models;
- a structured approach to modelling;
- less time building models (especially for novice modellers);

With the evaluation we wanted to **answer the following questions** from an expert modellers' point of view:

- Has the system reached its goals?
- What are the main advantages/disadvantages of the system?
- Is the modelling approach used in the system any good?
- Do the inference mechanisms seems reliable enough?
- Is there any evidence that the system might be of value as a learning environment?
- How can the system be extended?

6.1.3 Our approach to evaluation

Although many methods of evaluation are available, there is always a danger that if an evaluation is not a rigorous, formal, experimental, controlled, summative process, it will be regarded as a poor alternative. [Twidale 93] presents a good account of limitations of the "expected" sort of controlled evaluation, at least when dealing with certain AI systems.

Given the current state of the system, a large-scale summative evaluation of TeMS would be neither suitable nor feasible. It would be more interesting to gather anecdotal accounts of the circumstances under which the system would do well or badly, to find out what sort of improvements a panel of qualified users would like to see in it, to have a greater appreciation of how the approaches embodied in the system would be seen by the users and, at the same time, to have an overall picture of how the system is performing with respect to what was proposed. Hence, our experiment would have features of both formative and summative evaluation, although we would look for results representing a set of indicators rather than a comprehensive and exhaustive study. For these reasons, we decided on a qualitative sort of experiment, as the method adopted by many AI researchers currently involved in the evaluation of learning technology[Draper *et al.* 96] and interested in detecting the unexpected, inspired by the concepts of illuminative evaluation. As better described in Section 6.2, we used two instruments of data-collection namely: interview and questionnaire. To use more than one instrument would not only allow us to view the problem from a number of angles, but would also facilitate the cross-checking of otherwise tentative findings.

To understand problems and shortcomings of AI systems is more interesting and informative than successes[Gaschnig *et al.* 83] and there can be no better way of exposing the weaknesses of a system than to invite those who work in the task domain of the system to attempt to break it.

6.2 Method

6.2.1 Participants

All participants have, to some extent, been involved in ecological modelling and all of them are computer users, used to interact with a wide range of software. Many of the participants have experience in teaching and/or supporting students. We were especially interested in the opinions of participants whose expertise spans across all these areas.

Although there are people with very similar interests among the participants, their actual involvement with modelling spans from actually building models on a daily basis to managing and advising modellers, as well as teaching or tutoring students and developing new modelling environments. We believed that such a wide range of modelling experience could give us a general insight into the idiosyncrasies of ecological modelling.

A total of fifteen people from five organisations took part in the experiment. An initial set of twelve people, all professionals involved with modelling in Edinburgh and neighbourhood area was proposed during a discussion involving a member of the Department of Artificial Intelligence and a member of the Institute of Ecology and Resource Management who supervised the project. They were listed because their background and current activities seemed suitable to fit the "profile" required by the experiment. All listed people were contacted by e-mail and agreed in take part in the experiment. As the first interviews took place, another three people whose background would suit the requirements of the experiment were suggested by participants and also agreed in take part in the evaluation. The following is a description of the demography by institution, which is presented in Appendix D.1 along with summaries of some topics of the interviews.

Our first idea was to use apprentice modellers, possibly from the ecological modelling module at the University of Edinburgh, to evaluate TeMS. That option was discarded for two reasons: First, novice modellers have, by definition, little or no experience in modelling and tools supporting modelling, and although they could give a precise

answer on the effectiveness of the tool, little feedback on what the critical modelling issues are, would be obtained. In addition to that, we were interested in people's opinions of how modelling tools should deal with such issues, so experienced impressions were needed. Second, to be used by novice modellers, TeMS should be in a more advanced version than it currently is, so avoiding the user forming misconceptions when not distinguishing between modelling and implementation problems. Need for more sophisticated help facilities in the tool and for more tight control in the experiments were other aspects that influenced our choice.

Institute of Ecology and Resource Management

The Institute of Ecology and Resource Management (IERM) at the University of Edinburgh incorporates the Schools of Agriculture, Forestry, Ecological Science and Resource Economics. Its stated mission is to use scientific methods to understand the fundamental processes of biology, agriculture, the environment and economics, and to apply this understanding to the sustainable management of the world's biological resources. The Institute supports programmes of basic, strategic and applied research in temperate and tropical agriculture, forestry, ecology and resource economics with an interdisciplinary approach. Strategic research investigates computer modelling techniques among several other topics and much of its work results in prediction and simulation models and computer-assisted modelling is an important element of research within the Institute [IERM 98].

Seven participants were members of the IERM. A short description of the background of each one follows:

IERM1 electronics degree and Diploma; computer programmer (OO prog.); worked for Nintendo; developing modules for ModMed and AME; some teaching experience.

IERM2 ecology degree, has been developing on an individual-based modelling environment to deal with forestry.

IERM3 ecology degree; currently doing PhD in modelling (QR), used Stella during

undergraduate years; programming in Prolog and C++; works involves qualitative reasoning in ecological modelling

IERM4 ecologist; works with vegetation dynamics, mainly statistical modelling; no maths background.; programming in Pascal for data/statistical analysis; currently lecturing.

IERM5 worked with modelling during PhD (late 60's), building FORTRAN models for plant physiology; currently still does some modelling; lectures a great deal.

IERM6 BSc in ecology; worked in a system dynamics expert system; some teaching experience; programming in Pascal and SAS.

IERM7 1st degree in natural science; PhD in forestry; currently working on measure and modelling water usage by vegetation; some teaching experience; very familiar with computers (OS, programming languages, packages - not modelling packages)

Institute for Terrestrial Ecology

The Institute of Terrestrial Ecology (ITE) is an integral part of the Natural Environment Research Council. The Institute continues to develop long-term, multidisciplinary research and to exploit new technology to understand the science of the natural environment, with particular emphasis on terrestrial ecosystems. The Research Station on the Bush Estate, near Edinburgh, was opened in 1972 as the Institute of Tree Biology. In 1974, the Station became part of the newly formed ITE. The Station was enlarged in 1985, and now has research teams working on many aspects of tropical forestry, pollution, climate change and ecosystem process modelling. The research on ecosystem process modelling aims to understand and to model mathematically how ecosystems function and respond to climate and manipulation by man. Work is being done at the leaf, plant, ecosystem and global scales [ITE 98].

ITE1 degree in zoology; insect ecologist; work with population dynamics of insect pests; teaching experience; lead the modellers team at ITE (although not doing modelling himself lately).

ITE2 degree in ecology; some teaching experience; uses several modelling tools and programming languages (Maestro, Stella, FORTRAN and Prolog).

ITE3 biology 3rd degree, genetics PhD; computing experience during PhD; some teaching experience; quite mathematical background; currently in modelling work.

ITE4 ecology degree, work involves forestry and land use; some teaching experience; some programming in FORTRAN; currently at coordinatory tasks.

Department of Agriculture

Part of the IERM, the Department of Agriculture of the University of Edinburgh carry out research in the fields of crop sciences, animal genetics, animal behaviour and welfare and animal science, the latter with projects like the mathematical modelling of digestive processes in mammals to elucidate profiles of absorbed nutrients.

Agric1 degree in ecology, PhD in environmental physics, post-doc; lecturer; currently doing modelling himself, co-ordinating modelling projects and lecturing.

Agric2 degree in electronics; worked on the industry, PhD in cognitive science; currently building a modelling environment; has been exposed to ecology and ecologists.

Scottish Natural Heritage

Scottish Natural Heritage (SNH) is a government body with offices throughout Scotland and with a remit for both the conservation and enjoyment of all aspects of Scotland's natural heritage. The Research and Advisory Services Directorate is located in Edinburgh, and is charged with commissioning or undertaking research, survey and monitoring, as well as providing advice on the natural and social sciences to the four Regions and other Directorates of SNH. Among its tasks are the development of SNH's Research Programme consisting of externally commissioned and internally managed research and also the audit, survey and monitoring to establish and test procedures for the survey and monitoring of the natural heritage [SNH 98].

SNH1 ecology degree; PhD on impact of rabbits in vegetation; programming in Basic and FORTRAN; has built a large model for real-life based vegetation behaviour; currently is an advisor on land and grazing management; some teaching experience

Department of Artificial Intelligence

The Department of Artificial Intelligence (DAI) at the University of Edinburgh, was established in 1966 as the first European centre for basic research and post-school education in Artificial Intelligence. The Department has a very broad and interdisciplinary research profile, which can be loosely grouped under the broad headings of Automated Reasoning, Intelligent Robotics, Software Systems and Processes, Natural Language Processing, and Non-Symbolic AI. In particular, the Software Systems and Processes group has done work at the intersection of AI and Ecology since 1984, when the Eco-Logic project began and ran until 1990. Since then, the application of formal methods to ecological problems has been the thread of several research projects like this one.

DAI1 civil engineering BSc; programming (including Prolog); just finished PhD on logic-based approaches for ecological modelling; experience with teaching

6.2.2 Material

The evaluation consisted of a qualitative approach. We have therefore used some standard materials[CTI 98, SSSS 98, Robson 92].

interview

We have used semi-structured interviews (see Appendix C.2) both to encourage the acquisition of anecdotal information from the participants and to conduct the two-part interviews according to individual background and interests of each participant. A semi-structured interview is an instrument for data collection often used in qualitative

research, generally when exploratory studies are conducted and not much information is available to design a more structured interview.

questionnaire

The following are some reasons for using a questionnaire in addition to the interviews.

- to get a prompt reaction to the tool and to have a general view of it – participants were asked to fill in the questionnaire according to their opinions immediately after the demo/trial session, in order to get their reactions to what they had just seen and experienced
- to focus on usability aspects of the tool – the "impromptu" application of the questionnaire did not allow the participants time for deeper reflections, which was expected to help them to focus on usability issues, making clearer what problems are due mainly to interface inadequacies
- to improve consistency of the interviews – with the questionnaire we could have an alternative and summarised reference to people's views on the tool
- to introduce the second part of the interview – after the application of the questionnaire, the participants may have started to think about issues concerning the tool, its rationale and related topics and possibly how it would be used/placed in the modelling scenario discussed on the first part of the interview

The structure and presentation of our questionnaire was inspired by QUIS[Chin *et al.* 88] (Questionnaire for User Interface Satisfaction), a well known commercial product used to rate usability of different categories of software. In QUIS, a number of questions are organised in groups, each group has a main component question followed by related subcomponent questions. Each question has rating scales ascending from 1 to 10 and is anchored at both endpoints with adjectives like dull/stimulating.

Although QUIS was designed to support quantitative measurements, it has been refined to reach a small number of items while maintaining a high degree of reliability. We have selected those questions which were applicable to TeMS and change the rating

scales from 1 to 5 with the same adjectives anchored at both endpoints. The questions in our questionnaire are grouped in five different sets rating different aspects of the system. The questions of the first set rated the overall reaction to the system, the following three sets rated respectively: terminology/information, learning and software capabilities. The last set, not part of any standard group, was included to rate the tool with reference to its goals, as stated at the beginning of this Chapter.

The results of the questionnaire application are presented on Section 6.3.2.

Notes from observation

Another resource that we have used were notes from observation of the interaction between the participants and TeMS during the trial sessions. This material was important specially to identify points during the operation of TeMS were users have shown some difficulty as well as impressions on how each participant would interact with the tool.

Models generated

Also during the trial sessions, the participants have generated the model proposed in the introductory material (Appendix C.1). The post evaluation of the models illustrated how people would interpret in different ways the same scenario and would produce different representations for it. These models were also invaluable resource for fixing a few problems that users came across when using TeMS.

In addition to their individual advantages (see [Lindgaard 94, Dooley 95, Oppenhein 92, Parlett & Hamilton 77]), these instruments of data collection were chosen because we believe they were the most appropriate set, given our resources and limitations, to obtain the material we need for our analysis.

6.2.3 Procedure

1. Interview Part I

   Goal: To define participant's background (ecology, modelling, computing), including knowledge/preference of other procedures/tools/methodologies and

teaching experience.

Time: 10 minutes.

2. Short demo of the system

Goal: To show the system working with a simple model, making clear again (as it should be in the introductory text) the main ideas.

Time: 5-10 minutes.

3. Presentation of problem scenario

Goal: To give to the user a description of a problem for which he/she should build a model.

Time: 5 minutes.

4. User interaction with system

Goal: To allow the participant to build a standard model from the problem description given and possibly make their own explorations.

Time: 20 minutes.

5. Questionnaire

Goal: To rate main features and participant's impressions of the system.

Time: 5 minutes.

6. Interview Part II

Goal: To get participant's impressions about the concepts and the product.

Time: 15-20 minutes.

6.3 Results and Discussion

6.3.1 Idiosyncrasies of modelling

We have used the first part of the interview - background description - to find out more about the personal views of the participants on the topic of modelling. We were

interested in knowing how they do modelling, how they learn to do it and how they place ecological modelling among the more general activity of modelling. As we expected, that was not always a straightforward exercise, mainly because the participants might not have reflected on such issues before.

When discussing modelling, one of the first issues to arise is what people understand by "modelling in ecology". The semantics of modelling could, and in most cases will, vary very much depending on factors such as the modeller's background and the nature of her work. Until recently, ecological modelling used to be associated mainly with statistical modelling, largely used to summarise large amounts of data giving an *descriptive* account of what field ecologists have found. With methodologies and tools for simulation modelling improving and becoming easily available, the concept is moving towards a more *exploratory* and when possible, *predictive* approach. Simulation modelling for example, has become more popular, overcoming some antagonism by biologists who would prefer not use more formal frameworks (i.e. involving some mathematics).

These different perspectives on modelling have made evident a schism between field ecologists and modellers, as illustrated by the following statements:

> "I suspect that a high proportion of the ecological work is really descriptive, sort of exploratory work, and it only becomes more formal on the last day, when they design experiments to those particular components which they have observed."
>
> "Simulation modelling is something totally different from traditional modelling that ecologists have done, (...) fitting statistical modelling." (IERM4)

But whatever their personal views on the main concept are or the degree of scepticism with which they see other people's views, modellers agree that modelling is a difficulty intellectual activity which we do not often reflect upon, as pointed out by one interviewee:

> "I think modelling is the fundamental scientific activity, we all do it, and we don't know [how] we do it" (Agric1)

People's ideas on ecological modelling might also be derived from the association they make with other intellectual activities such as programming or experience in other domains such as mathematics or physical systems.

> "I found it [modelling×programming] to be just another language for the same thing" (IERM1)
>
> "they're both tools, but exist in their own right as subjects, essentially tools to be applied to various subjects" (IERM3)

The current practice, as emerged in the interviews, still is to write a model from scratch or, as seems to be more common nowadays, to use someone else's model and rewrite it, so a modeller who wants to model a system would look for models with "typical features" of that sort of system and adapt it to their own purpose, using the first one as a *template.*

> "it's more a matter of taking another model that exists and looking at what it does and try to implement that in a slightly different way, rather than starting from the very beginning and trying to write a whole new model" (ITE4)

Some modellers said they have developed their own templates for defining a model, although they have never made them explicit to others. The software normally used are mainly programming languages like Fortran, Basic/Visual Basic, C/C++. Software like spreadsheets and statistical packages are very often employed and modelling tools like Stella and ModelMaker are now more in use. As the following statement shows, there is now greater motivation for the use of modelling tools:

> "I know that you can in many cases achieve the same thing much more quickly in a much more flexible way by using a modelling tool and that's what interest me" (IERM7)

People learn modelling in different ways. Formal training is available at some undergraduate courses. In biology courses Systems Dynamics is the main media used, probably due to the availability of references and software to work with it. Other areas

(e.g. engineering) use different paradigms. In our sample, 9 participants had formal training at undergraduate level (6 of them in ecological modelling). But many people who are now doing real-life models in ecology had no training in modelling at all (6 in our sample), they would have learnt modelling at the time when they started at some position, and had to learn by unstructured tuition or be self-taught, as a participant said:

> "[I] picked up from books, papers, conferences and by working on it" (IERM7)

Some of the people interviewed said that learning modelling was just a matter of giving a more organised treatment to things they have already been doing, and as put by one of them, intuitive instruments like listing and diagramming are the basic tools for it. This does not depend whether you already have a formal paradigm to construct a model. He said:

> "you are building models anyway in your brain, it's just formalising them" (SNH1)

conceptualisation × implementation

We asked the participants about the way they go from the conceptual model to its implementation suggesting two common approaches, namely:

1. to devise a complete conceptual model (or the nearest of such) and then implement it, or
2. to identify some features of the model, implement it and by running the implemented model, try to identify new features and proceed to a new implementation

Most of the participants (9 out of 15) chose the second one, that is, they would normally run their implemented models to find out other aspects of it they did not think of at the start. Implementation, in that context, is also a way of visualising and reflecting on the conceptual modelling. This is consistent with a point made by a participant who said

that biologists have a tendency to avoid large-scale use of formalisms like mathematics, which may be required by someone who is working only at the conceptual level. People in that group would say:

> "I think it is definitely an interactive process ... you try to put something together and then you see how it would test against real data, real measurements of the system and then if doesn't work the way you expected or you hoped to, then maybe try to change it." (IERM7)
>
> "we try to sort of... deferring, leave that line out until what we need to do, ... then run it so we have it working and then adding more detail each time..." (ITE3)
>
> "some things only become apparent after you implement the first part, so you change the model." (IERM2)

From the remaining six participants, three said they would go as far as possible with the conceptualisation stage, and only when a quite comprehensive view of the model has been reached, they would go for the implementation. In this case, it would be more a matter of finding a correct implementation for the model devised.

> "It saves a lot of time if I work through all the ideas first and then go and do it afterwards. I definitely find that [this] is the best way of doing it, and that's regardless of what I'm doing."
>
> "I always spend a lot of time on the design, before I sit down and program. I was taught to do things that way" (IERM1)

Finally, the last three participants did not think they would fit exactly in either approaches. One of them said it would depend on each problem. For some problems it would be possible to have a reasonably clear mental picture of what the model should look like, enough to know what would be the parts or aspects of the system which should be modelled, and how they would relate to each other. For other systems though, the modeller would implement simpler versions of the model in order to form an idea of the "bigger picture", that is, how the final model should be. A modeller would form their own heuristics of how to identify what would be the best approach.

A participant pointed out that the problem could be reversed, that is, the modeller would also work backwards by using standard models to explain data from the field.

It seems that in this group of subjects formal training has no direct effect on the approach people will take when designing models, since the number of people with and without formal training is similarly distributed among the first and the second approaches and the combination or exclusion of both.

Modelling in ecology × modelling in general

In general, participants would not have a ready answer for a question asking whether modelling in ecology is essentially different, or has very distinguished aspects from modelling in other domains. We should remember that modellers normally do not need (or want) to reflect on such issues. Rather, they use models as simple and undistinguished scientific tools. Occasionally this changes as they speculate about it when trying to apply methodologies from other domains into ecological modelling (e.g. object oriented programming).

Regarding this matter, there were three clearly distinguished groups. The first one, formed by 6 participants, supported the view that there are critical differences between eco-modelling and modelling in other domains, as one participant said:

> "There is a big difference between ecological modelling and engineering modelling, because engineering modelling can be much more tied up into the laws of physics which are general laws. Biology has got fewer laws and we are dealing with a chaotic if not stochastic environment. A lot of ecological modelling depends on empirical relationships, so they [ecological modellers] are not quite sure how reliable those empirical relationships are..." (Agric1)

Another participant pointed out that in some domains (e.g. engineering) a modeller has to deal with a huge number of components for which most physical laws are well known whereas in other domains (e.g. business), there is a not so large number of components with a high interaction between them; in ecology, on the other hand, you have a large number of components with a huge interaction between them and less theoretical

framework to explain their behaviour. The "enormous uncertainty in ecology", as put by another participant was also mentioned several times throughout the evaluation. Ecological systems are characterised by having many sources of uncontrolled variation and when that happens within a complex, rarely obvious, network of interactions, it only makes the matter more complicated. In addition to the inherent complexity caused by heavy and chaotic interaction between many components, measuring and defining the parameters needed to model a real-life system is not usually an easy task either, as corroborated by the following statement of a modeller interviewed:

> "you can't possible model every interaction that you are already aware of in the system, because it would make the model too complicated and there are probably interactions going on that are immeasurable or you don't know about" (IERM7)

Most of the participants from this first group would agree that the modelling issues are similar, but they stressed that because of things like those mentioned here, ecology is a "special case" in modelling. And one would further add:

> "problems may be similar but the way people approach them, the problem solving methods, are different. They [modellers in economics] tend to think in other ways..." (IERM2)

The second group, also formed by 6 participants is formed of participants who could not see any substantial difference between modelling in ecology and modelling in other complex domains. For them, ecological modelling shares the most significant problems with many, if not all, other areas with complex systems.

> "I would think most things have very similar modelling problems" (SNH1)
>
> "most modelling tools I've seen could be used in ecology or economics, for example" (IERM1)

One person mentioned that modelling is a "meta-subject" which would be independent of any specific domain, whatever the level of complexity involved; that is, the general principles would apply regardless. That view was shared with other people

who would say they have their "customised" approach to modelling but the essential points would be the same of other modellers. Another person expressed the view that similar problems are always found when somebody is modelling activities involving human beings.

People from this group also suspected that ecological models have peculiarities that might make them harder to deal with, such as the difficulty of setting up realistic parameterisation of models, or the balance between number of processes and level of details modelled in each one.

Three participants would not fit in any of the two groups. They could see justification for both positions.

People from all groups would have different views on the argument suggested by one participant, according to which ecological modelling would be different because it deals with natural (as opposed to man-made) systems. He said:

> "... ecological modelling is more complex because the system you're trying to model is not something that you, basically, have designed yourself or know about and therefore the model needs a lot of constructs that are unique to modelling natural systems." (Agric2)

Considering the points raised by the participants, we can accept that modelling in ecology shares many, if not all, of the questions in general modelling. However, it is also obvious that some peculiarities of ecology, like the fact that it is dynamic and therefore very difficult to segregate or to establish a threshold for interactions, make it very hard to use general-purpose modelling paradigms straightforwardly. Such difficulties in ecological modelling apply to both conceptualisation and implementation of the model and some of those features were mentioned as the root for what people thought is the most difficult thing when modelling in ecology.

What the main difficulties are?

Considering that a modeller could identify all the important components in an ecological system, the first difficulty he or she would have is to select the ones he or she should

include in the model, given the very complex web of relationships between the components. If the modeller manages the *abstraction* phase, another difficulty would be how to conform their ideas for a model into a modelling framework whatever it would be. The lack of non-deterministic support for representing ecological phenomena and the tendency of some people to be unsympathetic towards maths are only two of the *conformation* problems mentioned by participants.

Realistic determination of all parameters needed for a model is another difficulty mentioned by modellers. Unlike other domains, where the identification and measurement of parameters may be straightforward, in ecology this can become a real impediment. Factors such as the different time scales used and the reproductive habits of every species in a ecosystem can make *parameterisation* a real problem. Parameterisation is then a substantial influence on the way people do modelling, as can be seen by assertions from modellers, such as:

> "... when we have a model, the difficult thing is to parameterise it, because it is just too big a task for most people to go out and measure all the parameters, and that leads to a kind of schism between those modellers who represent a few processes and have a small number of parameters and the others modellers who like to have rich detail but unfortunately they don't have a way to find the parameters' values." (IERM5)

All the previous points are part of

> "the difficult thing about modelling is actually the whole concept, the idea of dealing with things, with objects as systems with complex properties (...) it's a difficult intellectual activity." (Agric1)

Finally, to reach a *correct implementation* of a conceptual model is another difficulty also mentioned by modellers. To know if the model implemented actually corresponds with that which the modeller had in mind is a hard task, especially for those not too familiar with the means used, such as programming.

> "I think the modelling isn't the problem, I've got modelling in my head, great! but how do I get this into a computer and how do I get the numbers?"

"it's the implementation I think, just the technicalities of programming" (IERM3)

"my ideal way of working is with someone who is an expert in programming and can do that job well" (Agric1)

It was said by one participant that in general, given the right resources, time and knowledge for example, modellers would love to build their own programs. The preference for delegating programming (as can be seen in the statements quoted above) might well be due to a lack of easier-to-use programming environments suitable for modellers, another justification for tools like TeMS.

Table 6.1 sums up some of the peculiarities of modelling gathered from the interviews and discussed in this Section.

6.3.2 Questionnaire answers

Our analysis of the answers to the questionnaire aimed to produce a general picture of respondent's opinions on TeMS, specially on its usability topics, as well as to supply another reference to compare and to check consistency with the interviews.

In a piece of qualitative research, one would not expect to see graphics and tables, usually result of statistical procedures typical of quantitative approaches. However, we thought it would be helpful for the reader to have some way of seeing a graphic summary of the questionnaire results and so we present simple frequency distribution graphs for the different groups of questions in the questionnaire.

Figure 6.1 presents the results for the first set of questions named "Overall reactions to the software". Each frequency distribution graph corresponds to the answers to one question. Each one has a five-point scale with two adjectives on the extremes and the three points between which are always named: below medium, medium and above medium. Above each column of a graph, the number of respondents who marked that option is printed.

Our comments on Figure 6.1 as well as on the other summary graphs are mostly descriptive with tentative explanations. While abstaining from referring to statistical

TOPIC	RESULTS	No. of Respondents
Background	Biology	11
	Computing	2
	Others	2
How people do modelling	Using someone else's model	11
	Designing from scratch	3
Conceptualisation and implementation	Identify features, implement, try it and revise	9
	Conceptualise and then implement	3
	Other	3
Are there differences between modelling in Ecology and modelling in general?	Yes	6
	No	6
	Not sure	3
Main difficulties on modelling	To select components	1
	To conform the ideas to a model	3
	To obtain realistic determination of all parameters	2
	To reach a correct determination of a conceptual model	2

Table 6.1: Summary of data collected on idiosyncrasies of modelling

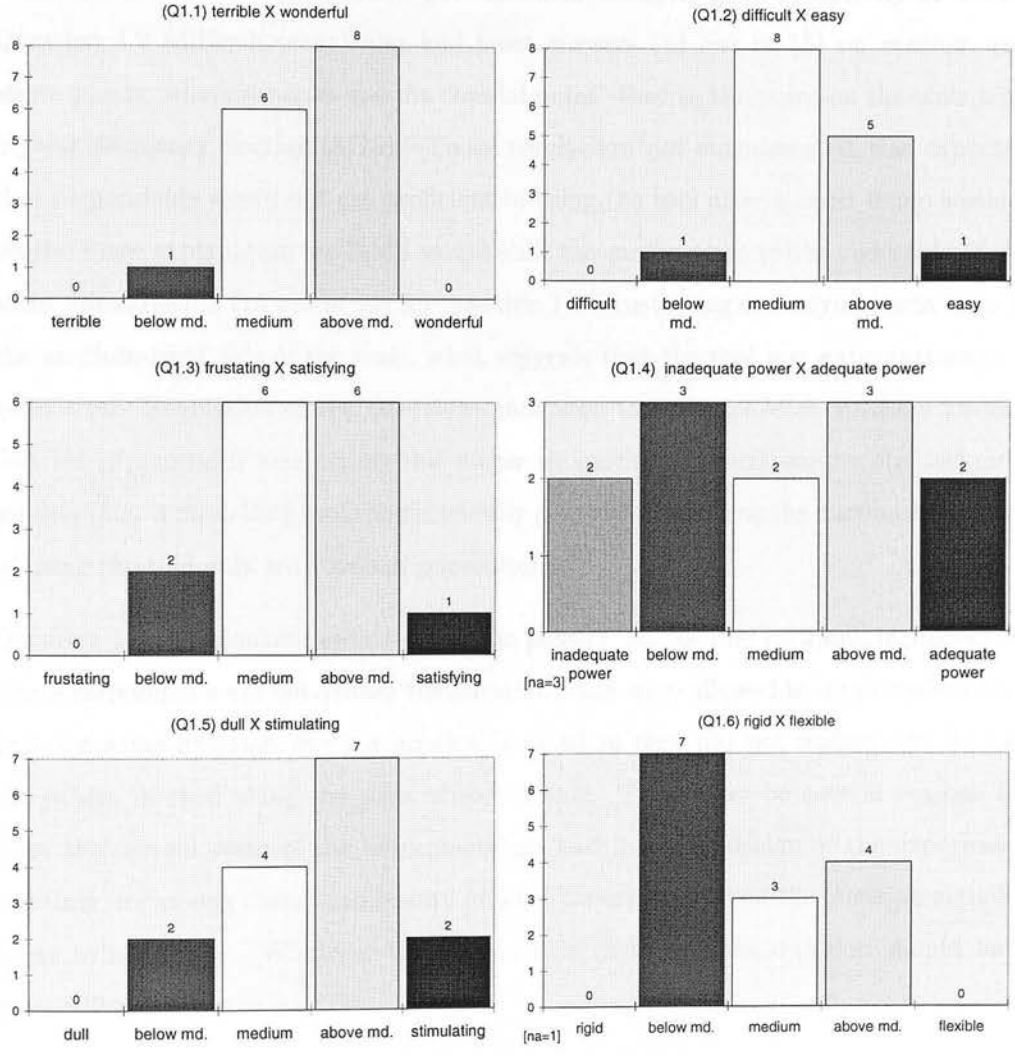

Figure 6.1: Question 1 – Overall reactions to TeMS

coefficients or tests, we often refer to which "side" of the scale most respondents rated a specific aspect of TeMS that is, a tendency to concentrate most answers to either side of the scale - were the main adjectives are, or towards the centre of it.

On question 1.1 (terrible×wonderful) most answers (14 out of 15) are distributed between *medium* and *above medium*[1] points, which indicates good receptivity to TeMS. Question 1.2 (difficult×easy) also had most answers (14 out of 15) on *medium* and above points, where *medium* was the "modal point" that is, the point on the scale with highest frequency density (8/15). Those results are not surprising, it was expected that respondents would not get proficient in using the tool after a short demo session, maybe more explanation on TeMS would shift the marks more to the right side of the scale. Most results (13 out of 15) for question 1.3 (frustrating×satisfying) were also at the medium-right side of the scale, what suggests that the tool met expectations at a good level. Evaluation of the interviews indicated that marks *below medium* for this first set of questions was mainly due either to specific expectations on the technical aspects (e.g. a modelling tool should be fully graphical-based) or the current limitation of using the tool only with animal populations.

Question 1.4 (inadequate power × adequate power) was the one we found inconclusive. Three respondents did not answer the question (they were allowed to do this when they believed some question was not applicable or when they did not understand it) and the others marked along the scale almost evenly. There may be several reasons for that, the current state of the implementation and the tight design of the experiment certainly are among them, uncertainty of what aspect of the tool the question actually refers to is another. Whatever the reason, it is clear that the question should have been differently put.

Answers to question 1.5 (dull×stimulating) were again mostly at the medium-right side of the scale (13 out of 15). That was partially due to people generally being motivated or at least curious to test modelling environments. It is worth mentioning that the two answers on *below medium* points on this questions were from the first two participants to whom the system was introduced, when there was a certain difficulty in introducing the tool properly.

[1] Another way of interpreting this scale is: *terrible* → *bad* → *regular* → *good* → *wonderful*

The last question of the first set (rigid×flexible) has most answers on medium and below medium (10 out of 15). This was expected because of the top-down way of defining models, which could be limiting for experienced users. Those marking *above medium* may have considered potential uses of TeMS as a flexibility factor, given that two among them mentioned its use in different subareas within population dynamics and four of them discussed ways of using it to support learning.

Figure 6.2 shows the answers for the second set of questions on the questionnaire: "Terminology and information" group.

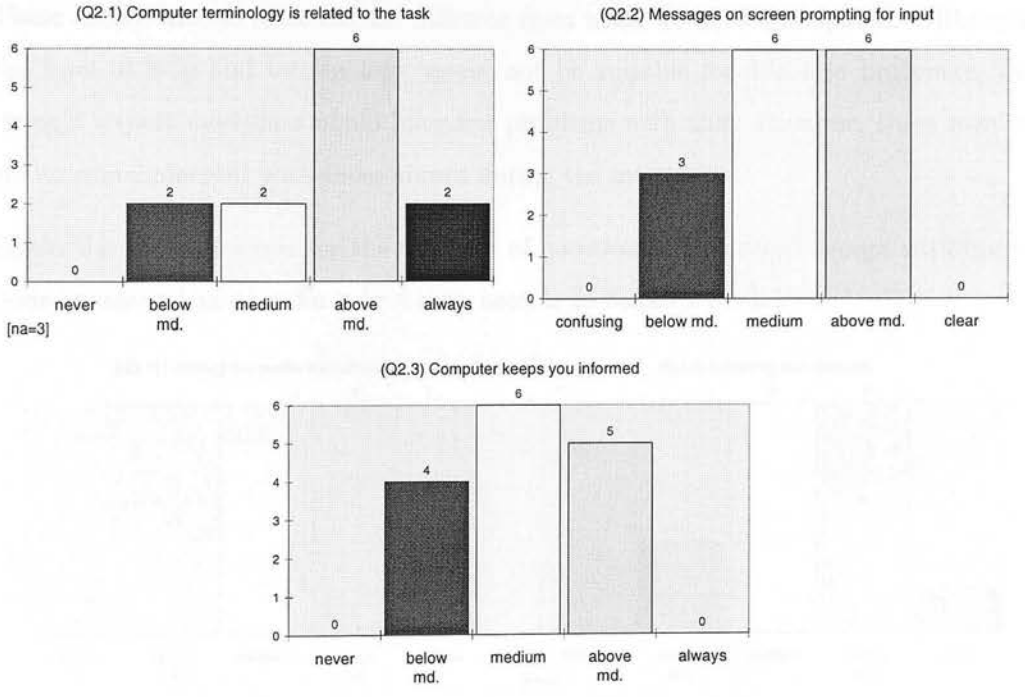

Figure 6.2: Question 2 – Terminology and information

When asked, in question 2.1, if computer terminology was related to the task they were doing (answer ranging from *never* to *always*), answers did not shown a definite concentration around either side of the centre of the scale (although with a the modal point with 6 respondents on *above medium*) and 3 people did not answer this question. In the interviews, "terminology" was also mentioned as one feature needing improvement.

Questions 2.2 (messages on screen: confusing×clear) and 2.3 (computer keeps you

informed: never×always) had answers with a noticeable tendency towards the medium point of the scale, and if from one hand there have been no answers rating the system as "confusing" or "never informing the user", on the other hand the need for more help and information was mentioned in the interviews, specially when suggested that novice users were to be employing the tool.

One respondent marked *below medium* in all three questions of this group. During the interview he mentioned that there should be more prompts or maybe worked examples for each question, which was consistent with his answers.

These results are, to some extent, different from what we have anticipated. Although the level of help and terminology would not be suitable for absolute beginners, we thought expert modellers would have few problems with this. However, these results are all quite coherent with those shown during the interviews.

Figure 6.3 shows answers for the third set of questions: "Learning" group, with questions assessing how easy for a first-time user, is to define a model.

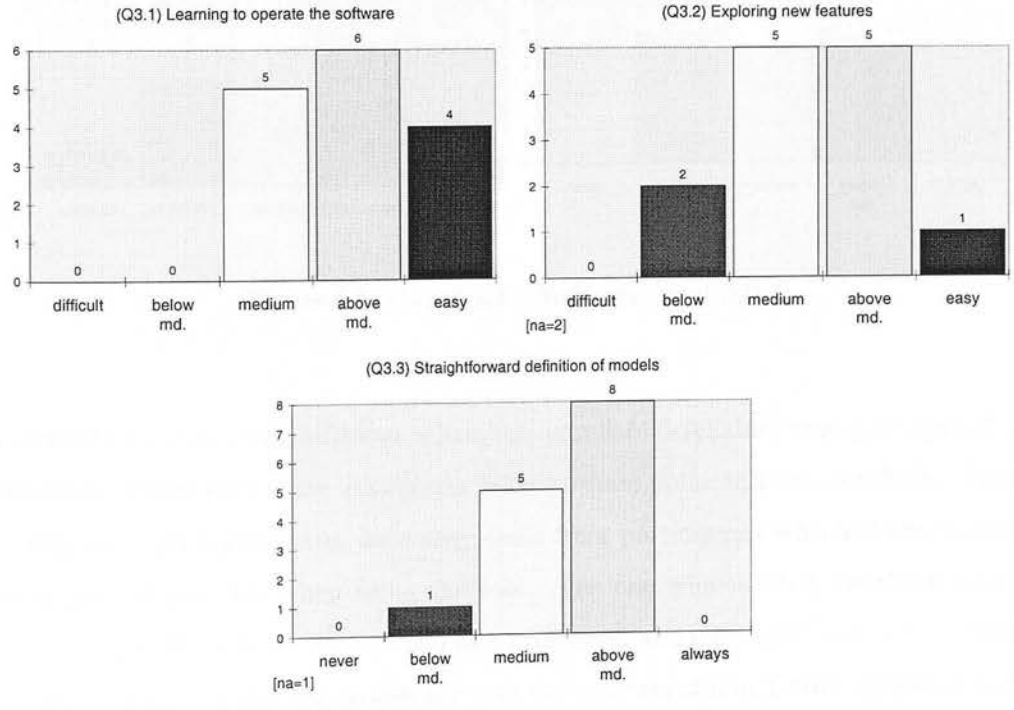

Figure 6.3: Question 3 – Learning

Question 3.1 (learning to operate the software: difficult×easy) all respondents rated the tool at the middle and right-side of the scale. This, compared with question 1.2, shows a consistent feeling that the system is easy to deal with.

Question 3.2 (exploring new features: difficult×easy). Although there was no consensus, many participants have seen in the tool a potential for exploration, at least at the level of modelling decisions.

Question 3.3 (straightforward definition of models: never×always) also shows that most respondents (13 out of 15) rated the tool at the medium and right-side of the scale and the modal point was *above medium.*

Figure 6.4 shows answers for the fourth set of questions: "Software Capabilities" with questions on the reliability of the system and its suitability for experienced and novice users.

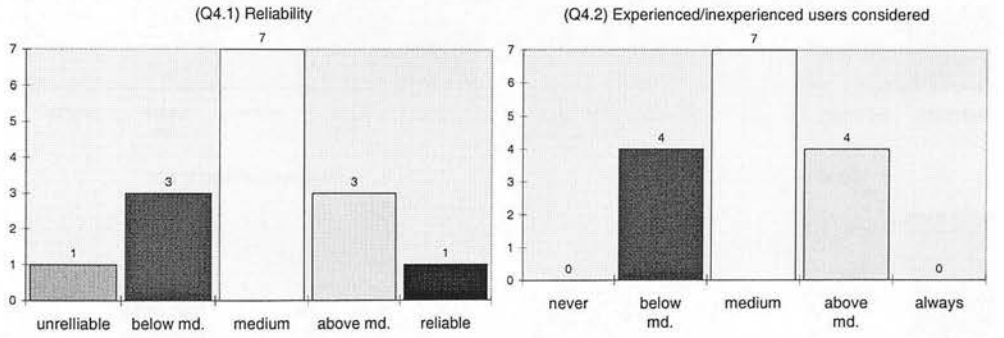

Figure 6.4: Question4 – Software capabilities

Answers for question 4.1 (software reliability: unreliable×reliable) were quite spread on the scale. There were three answers on *below medium* point and one *unreliable.* These marks were quite justifiable, once they came from participants who had experienced some sort of problem when using the tool. The one who marked *unreliable* had a system crash, that is, the system had to be restarted and all models had to be defined again. Although it was said in advance that the tool was still in a prototype version, it is understandable that problems of that sort affect confidence in its reliability. Among the other three participants, problems with slow connections and doubts on how the tool conducts its generation process also have affected the rate on this question.

Question 4.2 (experienced and novice users considered: never×always) had answers showing an acute tendency towards the centre of the scale, which cannot give definite support to the assumption that the tool can be useful for both expert and novice modellers. Issues related to this question are a modelling approach simple enough to be used by both experienced and novice users (see comments on question 5.1) and help and information (questions 2.1-2.3). Consistency was observed with answers to questions 2.2 and 2.3, which also have answers distributed towards the centre of the scale.

Figure 6.5 shows answers for the last set of questions: "Stated Goals".

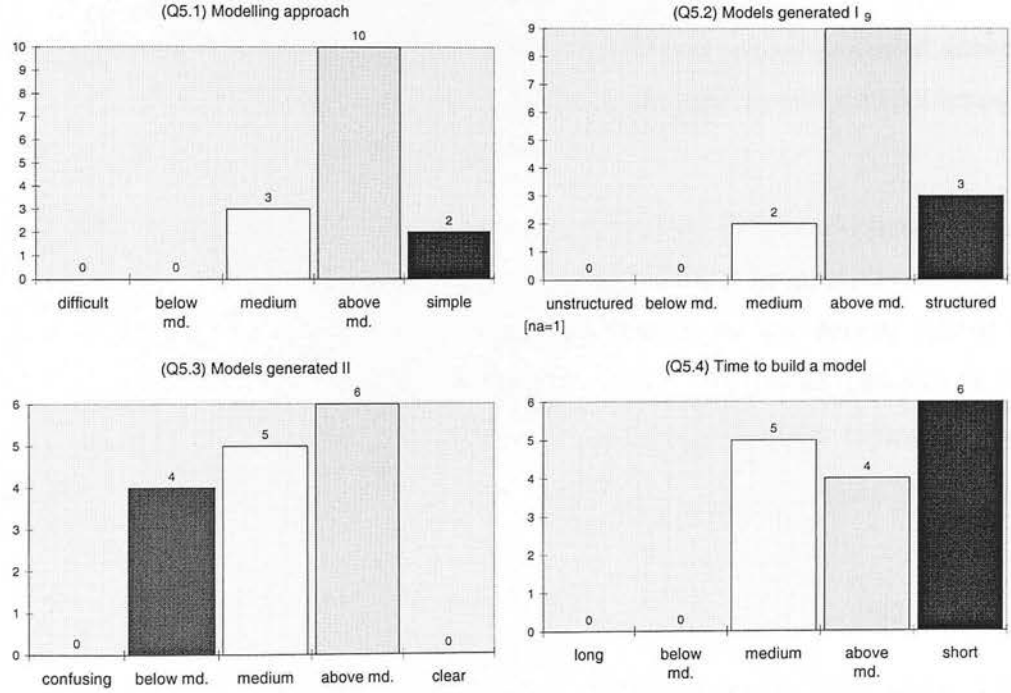

Figure 6.5: Question 5 – Stated Goals

Question 5.1 (modelling approach: difficult×simple) was the question with the largest concentration on one point – 10 out of 15 participants marked *above medium* with the other 5 are on the two adjacent points of the scale. That was corroborated by the interviews and is evidence that the approach is fairly simple and as analysed on Section 6.3.3, it is one of the features which gives TeMS potential to be used as a supporting tool for learning modelling.

Similarly to the previous questions, question 5.2 (models generated I: unstructured×structured), 12 out of 15 respondents rated the system at the right side of the scale (*above medium* and *structured*) and again, this was supported by the interviews, where it was mentioned that the generation of well-structured models was one of the advantages of the system (see Section 6.3.3). Question 5.3 (models generated II: confusing×clear) had answers concentrated at the middle area of the scale. Since the model is generated in Prolog and most of the participants did not know much about that language, we had expected this item would be at the left side of the scale, which was not the case. The factor that could have balanced that aspect was that the model was a result of clear design decisions and that must be an interesting issue to the users.

As we expected, question 5.4 (time to build a model: long×short) presented answers at the middle and right side of the scale, indicating the opinion that the tool actually allows a user to define a model fairly quickly.

The questionnaire was useful in forming a general picture of the participants' views. Although it was developed as a usability measurement device, its questions are related to those points discussed during the interviews, even issues not directly related to TeMS as a "product". The respondents seemed to have answered all questions on the merits of each item separately, as opposed to expressing a "general feeling" at each question (a problem known as the *halo effect*).

6.3.3 TeMS – approach to modelling

One of the results we expected to obtain with the approach used by TeMS was to make clear to the user the design decisions in the construction of models of a certain class. It seems we have succeeded with this: All participants said that TeMS' approach to modelling was clear, following logical steps. They thought it was a structured and easy-to-follow way of defining models.

Similarity to what modellers do in real life was another point participants agreed upon about TeMS. This supports our view that, in population dynamics and other areas within the ecological domain, people often reflect upon systems using a top-down, process-based representation of those systems (see Section 5.2.1), which would help

them to understand causal relationships on the model. Participants mentioned that

> "the concepts are similar" (ITE1)
>
> "it reflects the intuitive way that modellers do things" (IERM3)
>
> "getting people to think in terms of processes is an important part of this (...) I don't think it is necessarily a normal way of thinking" (Agric1)

The step-by-step approach used by TeMS to lead the user through the main design decisions also received support from the participants. Comments such as "it starts people thinking about what's important in modelling" and "the questions asked are essential questions", suggests that it may be useful to help the user to identify "landmarks" in the design process. However, participants also observed that when describing more complex systems, it would be too difficult to keep asking the user questions about the model without giving them more feedback on the current state of the model.

The following are some characteristics that people have mentioned as being distinctive from other modelling environments:

- step-by-step approach

  > "I'm not aware of any modelling tools that are used today that actually guide people through the model construction..." (IERM2)
  >
  > "most modelling tools I have seen were really based in systems dynamics, in drawing the diagrams, while this is more structured. It takes you through steps which I think is good." (IERM6)
  >
  > "this seems very structured and it leads you through to generate the model as you'd say, whereas something like Stella is like a white board where you draw things and make connections between them ..." (ITE2)
  >
  > "I'm pretty sure it is [similar to real-life approach]" (IERM5)

- focus on design questions – people thought the questions asked were in fact, essential questions, that is, questions about essential parameters and how they might be linked together. The effect of focusing on the main modelling questions gives a better understanding of what is really important to the process.

"it starts people thinking about what's important in modelling" (Agric2)

A participant from the Institute of Terrestrial Ecology (ITE) made an analogy with a real-life situation where modellers from ITE, usually with a standard background in biology or ecology, have to explain their models and their modelling approaches to people with different backgrounds (e.g. physicists or mathematicians) coming into the Institute. Those newcomers give an important contribution because they ask the most basic things, considered common-sense by ecologists, and cause reflection on what the more important concepts in ecological modelling are. TeMS was acknowledged as a formalisation of those fundamental questions.

However, to understand the questions might also require an amount of previous knowledge on the modelling terminology, since the approach relies on the semantics of the definitions (e.g. relationships, time structures).

- close assistance to the user

  "the idea here is to guide people through the process" (IERM2)

  "I like the idea that the way it asks you for information, so you have to respond and then it puts it together to make a progress" (IERM5)

- potentially better (at least in some aspects) than system dynamics-based environments

  "The only tool I have really used is ModelMaker (...) but I think this is clearer" (ITE3)

  "I see many advantages in this way of doing things [top-down approach] (...)
  it's more readable (...)
  it has more information from the modelling point of view than *white-board* approaches" (DAI1)

  "it clearly breaks out on age classes, and some of the other things I've seen, well... the traditional systems dynamics sort of model finds it rather difficult to handle age classes" (Agric1)

The ability to represent spatial interaction was another point mentioned as being not well handled by Stella or ModelMaker. Another distinguishing feature is the positive effect of constraining the range of models the tool can generate with its size and usability, so that a smaller system should be better handled by users. A modeller contrasted that feature of TeMS with other modelling environment:

> "The trouble is that on AME you got to think of a huge variety of possibilities so you got an enormous system." (Agric1)

- other

> "I've never seen a front end for Prolog programming like this before" (ITE4)
>
> "it could be useful as a knowledge acquisition tool" (IERM1)

TeMS approach to modelling have tackled the problems with modelling inherent to ecology, mentioned on Section 1.1.

Advantages × disadvantages

The limitation most often mentioned during the experiment was the lack of a graphical resource allowing the user to see a representation of the whole model, showing not only the components but also the processes affecting their populations. Participants also pointed out that a graphical representation of the model, built along the definition stage and dynamically changing at every decision made by the user, would be a way of speeding up their understanding of which features are important for the definition of a model, as users would actually see the effect of their decisions on the whole model.

The semantics aggregated to the terminology used by TeMS was another item which participants found should be improved. Some of the terms used, although quite standard in the technical literature, could have different interpretations according mainly to the level of experience of the user. Thus, when asking about *natality* and *mortality rates* and the implications of *ageing*, TeMS should be very precise and explicit about what is meant by these terms. Also, more detailed explanation should be given to the definition of time-structures used along with the ecological processes.

Opposite opinions were demonstrated on the tight top-down approach used by TeMS and its effect on the flexibility of the system. On the one hand, some people said it might constrain more experienced or more adventurous users (e.g. modellers who are also experienced programmers). On the other hand, people said they liked these constraints, because they would "direct the effort" to the right direction and, as noticed by another participant, they should make the tool less susceptible to errors made by users.

> "research scientists, I think, might find it a framework rather constraining (...) it might well be things that you want to do and you couldn't in that framework" (IERM7)
>
> "it's a good thing these constraints [are] put on me. The constraints put upon me made it easy for me to parameterise and build the model" (IERM1)

Clarity, simplicity of the concepts used and the structure it gives to modelling were the most frequently mentioned advantages of the modelling approach used by TeMS. These characteristics give an easier start to ecological modelling and, as a guided decision-making process, it should be quick to learn how to use the tool as well as to understand the rationale behind it, making more explicit what the basic concepts in that class of ecological modelling are. The following quotations illustrate these views.

> "I think it could be useful to remove some of the mystique and encourage people to get stuck in [built a model]." (IERM1)
>
> "I think that by doing this you're showing students who are put off by modelling that it is quite easy for themselves to build their own models" (ITE4)

It was said that TeMS would provide a good way of having some model up and running and since you can play with parameters, testing several scenarios, it would be a good exploratory tool. The potential to give more information through the design process when compared with "white-board" sort of tools was also emphasised.

Although the participants were exposed to a prototype version of TeMS, which might

have blurred their views on the approach used, they would see the same approach been used in other areas within ecology:

> "it could be used in several areas (agriculture, forestry and many branches of biology) which use Leslie-matrix models" (ITE4)
>
> "I'd think that people in conservation management and also people in pest management would find it extremely useful and these are the two most important areas of applied population dynamics." (ITE1)

A participant thought TeMS could be useful as a knowledge acquisition tool for a class of models in population dynamics, it could help to organise knowledge about systems. The ability to break directly in age classes, not usual in other modelling tools was also largely appreciated.

The modellers who took part in the experiment commented on the use of TeMS to support learning in modelling. The following section examines their views on this topic.

TeMS as supportive tool for learning

The need for learning tools in modelling is quite evident. There still is a lack of organised references for teaching ecological modelling and for introducing real modelling practices. TeMS could help novice users to *explore* model construction, a resource that a self-taught modeller said she did not find when she was starting on modelling:

> "you couldn't go to a library and get a book that teaches you by really describing it and have you started writing models" (ITE3)

All participants said they could see TeMS used to support learning in the ecological domain, many suggested improvements, most of which related to on-line help and the ability to have all parameters available to allow more exploration on the behaviour of the system.

> "I could imagine it might be really nice for use in teaching for instance. You try to get people who haven't done any modelling, maybe undergradu-

ates or other people, to understand how to do modelling and also to begin to understand interactions between animal populations. I think it would be very good for that because I think people could learn to define a model and be happy about modifying it and see how things work, in a trial-and-error sort of approach." (IERM7)

"you can test various scenarios and generate the set of results that can be looked upon in exactly the same way as you have looked upon them actually going out into the field and making real measurements. You must never confuse the two, but from the point of view of initial work of teaching students, it's very powerful indeed." (Agric1)

Although modelling has already been taught formally in some undergraduate courses, the current practice still relies too much on acquiring expertise in programming as well as in modelling. TeMS might help learners to concentrate on the right side of it, as argued by one participant:

"if novice [modellers] have to start from programming, and that's a huge problem, they would learn programming instead of modelling" (IERM6)

Most people supported that TeMS could help novices to understand and improve conceptualisation, at least in the domain of Population Dynamics, since the main design decision questions are highlighted and novices would become more aware of them. One participant was skeptical of the idea of using a tool for helping conceptualisation. He said:

"models come from experience, they come from knowledge of a particular part of the world, so I don't think you would ever use a tool to generate conceptual model, that's the process in reverse" (IERM7)

This view was not shared by other participants: they would rather value the supportive role of modelling tools in shortening the time and effort to get a grasp of conceptualisation. It was stated by many participants that TeMS has the potential to be more supportive than other approaches, provided more help on its design decisions would

be given, that would involve special care on explaining concepts such as age-class disaggregation and different instances of a process category. It would be important to explain to novices the consequences of their actions and, when possible, the effect such actions would have on the final structure of the model.

Improving model visualisation would be a requirement for using TeMS as a learning environment, given that it is thought to be specially important for novices. It was suggested that the current diagrammatic representation of the population disaggregation available on TeMS, could be extended to show population values and other relevant information dynamically. That of course, should be complemented by a model diagram representing the whole model (components, processes and variables).

Other improvements, such as a library of implemented models with classical examples of Population Dynamics were also suggested as a way to enhance usability by novices. Unlike the case of use by experienced modellers, computer efficiency would not be a critical issue for a supportive role of TeMS.

The last points on this possible use of TeMS is that it may suit people with various levels of knowledge, that is, from absolute beginners to experts and, as mentioned before, it can be used in several branches of biology.

All but one participant said they would be interested (or "curious" at least) in seeing how the system would build the model from the users' specifications. They said to see it would improve their understanding of how parameter specification would affect model construction, that is, the causal relationship between them. It was suggested that the system might use those relationships to give warnings of unusual choices that a user might do and not be aware of. On the question of *how* those relationships should be shown to the user, it was clear that simply showing relevant pieces of Prolog code would not be the best choice:

> "a novice user would struggle with Prolog syntax" (IERM2)
>
> "... I think using some kind of mapping, and staying away from using Prolog code initially [would be the best choice]. Perhaps having Prolog code plus an easier-to-understand, more intuitive representation, so you could see what that means." (IERM3)

"... maybe through some kind of graphical representation" (Agric2)

Finally, a participant suggested that TeMS is the sort of tool that could be used in the classroom by groups of 2 or 3 students exploring different modelling choice and it could stimulate debate amongst them.

6.3.4 TeMS – product

From the information collected during the evaluation we would like to identify the main issues raised on the topic of TeMS as a piece of software, a *product* and carry out some analysis from that point of view. We were interested on participants' views on the tool itself, how it would be placed among commercial tools and whether or not a differential factor would be noticed by them.

Taking in consideration the limitations of the experiment, namely: the prototype state of the system, the time constraints and the diverse set of characteristics embodied by the tool, we aimed to find out what the experts thought the current problems are and to rectify those. The idea was to list which improvements should be done on the tool in order to make it more friendly to end-users, especially to novices or novice-like users (a participant pointed out that, according to his own experience, there are many people who do not want to become computer experts, they simply want to use the tool like novices and be able to build their models).

It was expected that the main assets and drawbacks of the tool as perceived by the participants would be the consequences of the advantages and disadvantages of the approach used. Thus, the first and most obvious point has already been mentioned: the need for a graphical way of seeing the whole model, not only the diagram showing how the population is disaggregated. Users would specially benefit from a representation allowing them to see how their choices affect the model. The need for a dynamic picture of the whole model increases when one consider how it would be possible for a user to cope with more complex models, especially with many interactions between components. More graphical resources would also accentuate the differential factor between TeMS and other ways of model generation (e.g. direct programming).

Participants also pointed out the lack of on-line help, which they stressed should be "switchable" in order to be useful for novices without becoming an obstacle for more experienced users. More help to understand how to navigate among the different windows of the system was also mentioned.

On-line help is related with the need for more feedback from TeMS at all stages of the process definition, especially when dealing with novices. Care with the terminology used (e.g. using the term "transition" instead of progress-on-categories) and a way of following changes in the model are key points here. Improvements on feedback to the user would not necessarily imply a major modification in the current state of the tool. One option would be to allow the user to have summarised information on the current state of the model at any stage of the definition process. Even a text-based presentation of the structure would help the user to have a more complete view of the model. Presentation of time structures as a customisable time-line with marked points on it instead of a menu with classes and sub-classes, was another improvement suggested by a participant with experience in building tools for handling different time scales.

TeMS should also record *all* model parameters through time. That feature would make it easier to observe how those parameters change through time, allowing the user to compare different runs of the same model or even different models, reinforcing the use of TeMS as an exploratory tool. Improving editing facilities would be essential for that.

Finally, it was suggested that a MS-Windows version of the system would expand the user base, since both SICStus Prolog and TclTk are now available for Windows platform.

Models Generated

We used the model produced during the trial session to ask the modellers about their expectations on the models they usually produce. We wanted to get their views on the correspondence between the choices made, the model generated (Prolog program) and the results obtained (population values). Obviously we expected that some participants would be too busy during the demo/trial session to reflect on what the results should

or should not be. In this case, they would make a retrospective analysis instead.

We found out that a modeller normally has some expectation on what the population patterns should be, which could be more or less accurate depending on their previous experience and/or knowledge of the actual data from the field as well as on the complexity of model. So, if the modeller knows what the general behaviour of the model should be, it is usually possible for him/her to notice if something is obviously wrong, although there are cases where errors are found in models already used for a long time, which supports the need for some means of quality control and checking.

It was brought to our attention that the value of modelling as a scientific instrument is noticed exactly when expectations are not fulfilled. It is at this point that the modeller has to analyse the model to find the source of deviation and either correct it or form new expectations accordingly, as put by a participant:

> "you usually have an idea because you know something about the system, so you know how you would expect it to react, but your expectations aren't always right, which is why modelling is really useful because you found it that something didn't go quite as you expected so you have to go through the model to find out what it is." (IERM6)

By using standard ways of defining models and allowing for testing the results of the models produced, TeMS might contribute to improve quality control on the generation of models while suppling a platform for evaluation and redefinition of the design process, the latter mentioned as an important aid for the understanding of conceptualisation in modelling.

6.3.5 Future Work

The following are the main ideas for improving TeMS gathered during the evaluation. Most of those ideas are correction procedures for the shortcomings seen through the prototype and indicate some positive priorities for future implementations.

- to implement a graphical representation of the whole model which could allow more feedback on the current state of the model, making it easier to deal with

more complex models and to see the interactions between the components in the model as well as in the details of all parameters;

- to provide switchable pop-up help which could aid the understanding of the basic concepts used (semantics) and how they are used in the model. This should lead to an improvement of the terminology used by the tool.

  A participant was quite specific about where he would like to have more help:

  > "I personally don't know what syntax is it expected by Prolog, I don't know what operators Prolog uses, if it got any precedence rules (...) So, without some extra information I wouldn't be able to write any equation." (IERM7)

  The amount and detail of information on screen should be selected according to the expertise of the user, which should either be selected by the users themselves or by a user categorisation (see item below).

- to have a pre-installed set of practical examples which could be used as a tutorial. These examples should be standard population dynamics models with different levels of complexity and categorised by topics, possibly involving their application in different subareas;

- to improve the editing facilities on TeMS, adding the ability to record all parameters through time and use them when producing evaluation graphs. That should include the ability to draw graphs not only of population values but of any other model parameter, and do this as the simulation happens. To change parameter setting between runs must also be possible;

- to allow other improvements upon the tool:

  - to incorporate user modelling – which would allow TeMS to select resources according to the users' expertise or background, providing different ways of defining a model for different categories of users. It is worthwhile mentioning that such categorisation should identify more subtle differences. An expert modeller may or may not have knowledge of programming and there are

novices who are complete beginners and people who do want to be treated as novices, that is, to be able to use *one* sort of models and nothing else.

- to provide better support when used remotely, especially if a connection is slow;
- to supply predefined processes allowing some variability accounting for stochastic influences on the system modelled;
- graphical interface for the model in a way that users could work themselves backwards to recognise what the inference mechanisms are, that is, the relationship between parameter setting and the code which results. This could be done similarly to a *debugger*, a tool often part of a programming environment, where the error produced is linked to the piece of code which originated it.
- to give information (e.g. a manual) of how more experienced users or those with supervision tasks, could create new "set-ups" (e.g. new definition for processes) which could be used by a group of less experienced users under supervision;
- to provide slots for connecting TeMS with other modelling tools – A great deal of care would be needed to implement this. A participant mentioned his experience in trying to connect his ecological models with spreadsheet-based economic models. He affirmed that it is very difficult to interact with other models (or even sub-models) which do not use the same paradigm (language and references).
- to use the ability of comparing runnings (e.g. by putting several runs in one graph) for modelling a system under different viewpoints (e.g. ecological and economical models for land management);
- to extend explanation (terminology and/or inference mechanisms) using a Natural Language interface;
- to add the ability to do a first interpretation of the results and give warnings and even some explanation when abnormal behaviour happens (e.g. constant population in a pred-prey model).

6.4 Concluding remarks

We have seen that reflecting upon how we do modelling is not an easy exercise. People have different ideas of what *ecological modelling* stands for, what we should expect from it and how we use it. Although ecological modelling has always been used for describing phenomena observed in the field, usually based in statistical models, there was evidence that more and more people are using it for probing and even predict the behaviour of ecological systems through analytical and numerical simulation models. This exploratory approach to modelling is reflected in the way people build their models. Most of the participants in the experiment said they do not wait until they have a complete conceptual model before they start the implementation process. Instead, they use the first implementation of the model to have a better picture of the model and then refine this implementation until a satisfactory version of the model has been reached.

If on one hand this approach is a very pragmatic way of completing a model, where each version is used to "brainstorm" the modeller and generate a new one, on the other hand, it lacks structure and there is no guarantee that a suitable model will always be produced.

Abstraction, conformation to a theoretical framework, parameterisation and correct implementation are the main difficulties in the construction of ecological models. All these factors are shared by many other areas, but are especially critical in ecology.

The modelling approach embodied by TeMS was strongly supported for being clear, well structured and easy to follow. Also, its top-down, process-based approach was supported by the participants because of its similarity to how experts work in real-life. To clarify what the design decisions are in this sort of model was also rated as an important point.

The evaluation revealed what the main drawback in the approach was: the lack of graphical support to see the current state of the model. This however, could be a point for further development.

We had several suggestions on how to improve the tool in order to make better use

of its potential for supporting learning. All participants said TeMS could be used for that, especially providing help in the conceptualisation stage of modelling.

People were aware of difficulties on modelling and were receptive to the idea. They could see the application especially with novices and they had suggested several improvements in order to make the final product more supportive.

Chapter 7

Conclusion

This chapter presents our research conclusions. We sum up the project describing the main stages of it, list the contributions to the areas involved, and discuss further research on the topic.

7.1 Summary

The research reported here aimed to demonstrate what happens when we take a generic method for structuring formal specifications (in this case techniques editing) and attempt to tailor it to the design of a class of specifications in a target domain (in this case population dynamics modelling). The research questions posed on Section 1.4.1, were addressed through the various stages of this project, listed below.

- The project started with a **survey on the existing work** on the synthesis of formal specifications that make use of certain methods, namely: programming techniques and schemata. This survey was carried out having in mind the intention of detecting how those generic concepts would accommodate domain-specific applications. Thus, this study gave us the basis not only of understanding how those methods have been normally used but also of pondering on their utilisation in a context they have never been put before.

- The next stage of the project aimed at the identification of the domain-specific

structures and started with the **reconstruction of a model** from the population dynamics modelling literature in order to **extract techniques** from the reconstructed Prolog model.

- We also have **examined the ecological modelling literature** in order to identify standard patterns of the modelling process in population dynamics. As a result, we were able to **generalise** the set of techniques extracted from the reconstructed model to a class of models in the domain.

- We then built a **techniques editor** that can handle editing operations from interaction with the user or through Prolog predicates which can be automatically asserted by other modules in a more complex system.

- The next stage was the definition of a **problem description language**, later implemented through an user-friendly interface, where design decisions can be elicited from a user and stored in a knowledge base.

- We then devised a **generation system** that can use the problem description to generate the predicates needed to trigger editing operations on the techniques editor.

- The preceding elements were than combined in a **model synthesiser** (TeMS) that implements a simple, structured, process-based approach to modelling. The main idea of the system is to lead a user through a series of strategically sequenced design-decisions. At the end of the problem description stage, the system can automatically proceed with the synthesis of the corresponding model.

- TeMS was **verified** against a series of examples and it was able to produce models accurate with respect to a given description.

- The last stage of this project was the **evaluation** of TeMS by expert modellers. This experiment allowed us to gauge how well TeMS supports people in modelling. The design of the experiment was intended to gather and to analyse the participant's views on the *concepts* embodied in the system as well as in the resulting *tool*. We were also able to get the experts' impressions on questions related to the use of the tool – like its use to support learning in modelling and what improvements it would require – as well as on the practice of modelling.

As can be seen from the preceding list, we have achieved all goals stated on Section 1.4.2, and the contributions from this are listed in the next Section.

7.2 Contributions

The project described in this thesis has shown a way of extending the advantages of using Prolog programming techniques to a domain other than Prolog programming itself, thus applying this method of structuring formal specifications in a hitherto unexplored context. The following is a list of the contributions made by the project.

7.2.1 A library of domain-specific techniques

Our first contribution was to have identified domain-specific techniques which use the parameterisation method of generic techniques but which contain information specific to the population dynamics domain.

This was perhaps the most difficult step, because it is not always easy to identify appropriate domain-specific techniques. Our way of starting to acquire the necessary knowledge is to begin with a sample model (from the modelling literature) and to build the techniques needed to construct it. By making these techniques as general as possible (while still being recognisable within the domain) we generalised from a particular example to a class of models.

7.2.2 A problem description language

With these techniques in mind, we provided a problem description language which uses concepts from population dynamics, and constructed an interface which allows these concepts to be supplied. This enables the style of problem description to be disconnected, initially, from the style of definition of the domain-specific techniques.

7.2.3 A way of automating synthesis

A fundamental difficulty in our target domain is that the styles of description which population modellers understand are different from those needed to control specification synthesis using our chosen formal method.

To tackle the problem we have built an automated system which ensures appropriate parameterisation of the domain-specific techniques based on the population dynamics problem description. This connects the problem description to the techniques needed for model generation. It can be automatic because we employed restricted languages for both problem description and techniques.

7.2.4 A modelling environment

We also had supplied tools for executing the specification in styles familiar to those in the domain. This gives modellers an opportunity to check whether the model they have received is the one they expected – in our case these are population dynamics models with any number of interacting populations; a hierarchically disaggregated population structure; and process definitions controlling the interaction between populations. Since the main boundaries of this class can be described in terms of the domain, rather than more abstract mathematical limitations, it is easier for modellers to determine whether or not the system is appropriate to their needs.

We have used the system produced as the basis for an evaluation experiment, assessing the concepts proposed; the resulting tool and giving a better insight on how people actually do modelling and how the environment would fit on that scenario. We believe the evaluation designed for this project can be used as reference for the assessment of other AI toolkits to which small formative evaluations would not be informative enough and summative evaluation would require resources (time, effort, a "development-quality" version of the system) not often available.

7.2.5 An alternative approach to modelling

The modelling approach embodied in TeMS was strongly supported for being clear, well structured and easy to follow. Also, its top-down, process-based style was supported by the participants of the evaluation because of its similarity to how they believed experts work in real-life. The fact that it makes explicit what the design decisions are in this sort of model was also rated as an important point.

All participants endorsed the use of TeMS to support learning in the ecological domain, many of them having suggested improvements to that effect.

7.3 Future work

There is a number of ways in which the research reported here could be extended. Some of them are discussed in this section.

7.3.1 Improving user-support on TeMS

Several improvements can be made to the current implementation of our model synthesiser and, as we found out through the evaluation, might have a significant effect on its use by novices. The following are some of the improvements detailed on Section 6.3.5.

- to improve the editing facilities on TeMS.
- to provide switchable pop-up help and improve the terminology used in the tool;
- to have a pre-installed set of practical examples that could be used as tutorials;
- to implement a graphical representation of the whole model which could allow more feedback on the current state of the model;

7.3.2 Learning programming from modelling

TeMS emphasises design decisions in modelling, and allows the automated generation of a model derived from those decisions. This feature was thought to be important for novice modellers (see Section 6.3.3), who could then avoid the entanglements of programming when learning modelling.

However, at some point when becoming modelling experts, it is common that modellers want to acquire certain programming skills. This side-effect of the apprenticeship in modelling was evidenced during the evaluation – all participants have experience in programming, and most of them (11 out of 15) are actually skilled programmers.

One way of supporting modellers acquiring programming skills is to use the program construction resources presented in this project and make them explicit to the user. This feature could be added to a future implementation of TeMS through an additional module, a "generation viewer", responsible for communicating to the user the state of the generation process at each step of Phases I and II in the modelling process (see Section 5.1.3). This module would launch a separated (graphical) resource for displaying how each decision would have on the selection and parameterisation of domain-dependent structures as well as previewing the final code (a much better version of what is informally shown on Figure 5.5).

Obviously, by having this sort of resource does not necessarily imply any skill will be improved, but certainly it will make the stepwise enhancement methodology a lot clearer for the neophyte.

7.3.3 Extending the library

The library of domain-specific data structures defined for this project can be extended with the addition of new techniques, schemata and process definitions. Although this would not necessarily alter the class of models the tool can tackle, it would make available to the user new modelling patterns that could then be chosen rather than specified by a new equation each time they are needed.

7.3.4 Other domains

To probe the use of the methodologies in other domains, especially those in which process-based modelling have been used, would form an appealing extension to our work. This path could be taken at several levels, which could require everything from a simple redefinition of part of the knowledge base to the reimplementation of most of the tool.

7.3.5 Linking it to other approaches

Long-term work on the issues addressed here might include the study of how they could be adapted to other programming languages. It may also be the case that other sorts of domain-specific formal specifications – the "ecological agents" proposed in [Mota 98], for example – can be structured in a similar way as we have done with techniques.

Another path worth investigation is a broader use of domain-dependent techniques as repositories of modelling knowledge, which could be made globally available through web-based tools. These tools could address modelling in a way distinct from the one used here – instead of restricting problem description and data-structures in order to generate models automatically, it could use other sources of information about the domain (e.g. databases) to complete the problem description, so it would be able to suggest alternative configurations for the model and from interaction with the user or by using a set of previously established heuristics produce a suitable model.

Bibliography

[Bental 94] D.S. Bental. *Recognising the design decisions in Prolog programs as a prelude to critiquing.* Unpublished PhD thesis, University of Edinburgh, 1994.

[Bowles 94] A. Bowles. A techniques editor for prolog novices. Internal software report, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Bowles *et al.* 94] A. Bowles, D. Robertson, W. Vasconcelos, M. Vargas-Vera, and D. Bental. Applying prolog programming techniques. *International Journal of Human-Computer Studies*, 41:329–350, 1994.

[Brand *et al.* 98] C. Brand, C. Rader, H. Carlone, and C. Lewis. Prospects and challenges for children creating science models. Technical report, Department of Computer Science, University of Colorado, http://www.cs.colorado.edu/ crader/, 1998.

[Brilhante 96] V. Brilhante. Inform-logic: a system for representing uncertainty in ecological models. Unpublished M.Sc. thesis, Department of AI, University of Edinburgh, 1996.

[Brna *et al.* 91] P. Brna, A. Bundy, T. Dodd, M. Eisenstadt, C.K. Looi, H. Pain, B. Smith, and M. Someren. Prolog programming techniques. *Instructional Science*, 20(2/3), 1991.

[Bundy 84] A. Bundy. Intelligent front ends. In Bramer. M.A., editor, *Research and Development in Expert Systems XI, Proceedings of the IV Technical Conference of the British Computer Specialist Group on Expert Systems*, pages 193–203. Cambridge University Press, Cambridge, 1984.

[Bundy *et al.* 91] A. Bundy, G. Grosse, and P. Brna. A prolog techniques recursive editor. *Instructional Science*, 20(2):135–172, 1991.

[Castro 94] A. Castro. TBPG – A techniques-based program generator. Technical paper 29, Department of Artificial Intelligence, University of Edinburgh, 1994.

[Cheng *et al.* 98] P. Cheng, J. Cupit, and N.R. Shadbolt. Knowledge acquisition from cartesian graphs: An ontological framework. In *Proceedings of the 11th Banff Knowledge Acquisition Workshop.* Banff, Alberta, Canada, 1998. http://ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc.html.

[Chin *et al.* 88] J. Chin, V. Diehl, and K. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In E. Soloway, D. Frye, and S. Sheppard, editors, *Human factors in computing systems – CHI'88 Conference Proceedings*, pages 213–218. ACM Press, 1988.

[Cohen 95] Paul Cohen. *Empirical methods for artificial intelligence.* MIT Press, 1995.

[Conlon 97] Thomas Conlon. *Beyond rules: the development and evaluation of knowledge acquisition systems for educational knowledge-based modelling.* Unpublished PhD thesis, University of Edinburgh, 1997.

[Crête *et al.* 81] M. Crête, R.J. Taylor, and P.A. Jordan. Simulating conditions for the regulation of a moose population by wolves. *Ecological Modelling*, 12:245–252, 1981.

[CTI 98] CTI. Research methods and practicals. Technical report, CTI Centre for Psychology, University of York, York, http://www.york.ac.uk/inst/ctipsych/stats.html, 1998.

[Dooley 95] David Dooley. *Social Research Methods, 3rd Ed.* Prentice-Hall Inc., 1995.

[Draper *et al.* 96] S.W. Draper, M.I. Brown, F.P. Henderson, and E. McAteer. Integrative evaluation: An emerging role for classroom studies of cal. *Computers and Education*, 26(1/3):17–32, 1996.

[EKSL 98] EKSL. The experimental knowledge systems laboratory. Technical report, EKSL and the University of Massachusetts Amherst, Amherst, Massachusetts., http://eksl-www.cs.umass.edu/eksl.html, 1998.

[Eriksson *et al.* 94] H. Eriksson, A.R. Puerta, and M.A. Musen. Generation of knowledge-acquisition tools from domain ontologies. *International Journal of Human-Computer Studies*, 41:425–453, 1994.

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

[Gallagher & Lyle 91] K.B. Gallagher and J.R. Lyle. Using program slicing in software maintenance. *IEEE Transactions on Software Engineering*, 17(8):751–761, August 1991.

[Gaschnig *et al.* 83] J. Gaschnig, P. Klahr, H. Pople, E. Shortliffe, and A. Terry. Evaluation of expert systems: Issues and case studies. In F. Hayes-Roth, D.A. Waterman, and D.B. Lenat, editors, *Building Expert Systems*, pages 241–280. Addison-Wesley Pub Co Inc, 1983.

[Gegg-Harrison 89] T. Gegg-Harrison. Basic prolog schemata. Technical Report 89-20, Duke University, 1989.

[Gegg-Harrison 93] T. Gegg-Harrison. *Exploiting program schemata in a Prolog tutoring system.* Unpublished PhD thesis, Duke University, 1993.

[Gruber 93] T.R. Gruber. A translation approach to protable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Haefner 96] James Haefner. *Modeling Biological Systems: Principles and Applications.* Chapman & Hall, 1996.

[Hutchins *et al.* 86] E.L. Hutchins, J.D. Hollan, and D.A. Norman. Direct manipulation interfaces. In D. Norman and S. Draper, editors, *User-centered system design*, pages 87–124. Laurence Erlbaum Associates, 1986.

[IERM 98] IERM. Ierm home-page. Technical report, Institute for Ecology and Resource Management, http://helios.bto.ed.ac.uk/ierm/ierm.htm, 1998.

[ITE 98] ITE. Ite edinburgh. Technical report, Institute for Terrestrial Ecology, http://mwnta.nmw.ac.uk/ite/edin/edin.html, 1998.

[Jackson 90] Peter Jackson. *Introduction to Expert Systems.* Addison-Wesley, 1990.

[King 93] K. King. *Making diagnosis explicit.* Unpublished PhD thesis, University of Edinburgh, 1993.

[Kirschenbaum *et al.* 89] M. Kirschenbaum, A. Lakhotia, and L.S. Sterling. Skeletons and techniques for prolog programming. Technical Report TR–89–170, Case Western Reserve University, 1989.

[Kirschenbaum *et al.* 94] M. Kirschenbaum, S. Michaylov, and L.S. Sterling. Skeletons and techniques as a normative approach to

program development in logic-based languages. Technical Report 25, Ohio State University – CISRC, May 1994.

[Krebs 78] C.J. Krebs. *Ecology: the Experimental Analysis of Distribuition and Abundance.* Harper and Row, 1978.

[Lewis *et al.* 98] C. Lewis, C. Brand, G. Cherry, and C. Rader. Adapting user interface design methods to the design of educational activities. In *Proceedings of the Conference on Human Factors in Computer Systems (CHI-98)*, pages 619–626. ACM SIGCHI, Los Angeles, CA, 1998.

[Lindgaard 94] Gitte Lindgaard. *Usability testing and system evaluation – A guide for designing useful computer systems.* London : Chapman & Hall, 1994.

[Mark & Greer 93] M.A. Mark and J.E. Greer. Evaluation methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 4(3/4):129–153, 1993.

[McGraw 92] Karen McGraw. *Designing and evaluating user interfaces for knowledge-based systems.* Ellis Horwood, 1992.

[ModelMaker 98] ModelMaker. Modelmaker simulation package. Technical report, Cherwell Scientific Publishing, http://www.cherwell.com/modelmaker/index.html, 1998.

[Mota 98] E. Mota. *Time granularity in simulation models within a multi-agent system.* Unpublished PhD thesis, University of Edinburgh, 1998.

[Muetzelfeldt & Taylor 97] R. I. Muetzelfeldt and J. Taylor. The suitability of ame for agroforestry modelling. *Agroforestry Forum*, 8:7–9, 1997.

[Muetzelfeldt 95] R. I. Muetzelfeldt. A framework for a modular modelling approach for agroforestry. *Agroforestry Systems*, 30:223–234, 1995.

[Muetzelfeldt 96] R. I. Muetzelfeldt. Flomo tutorial – population dynamics modelling. Internal note, available from the author. IERM, University of Edinburgh, 1996.

[Muetzelfeldt *et al.* 89] R. I. Muetzelfeldt, D. Robertson, A. Bundy, and M. Uschold. The use of prolog for improving the rigour and accessibility of ecological modelling. *Ecological Modelling*, 46(1/2):1–20, 1989.

[O'Keefe 90] Richard A. O'Keefe. *The Craft of Prolog.* MIT Press, 1990.

[Oppenhein 92] A. N. Oppenhein. *Questionnaire Design, Interviewing and Attitude Measurement.* Pinter Pub. Ltd, 1992.

[Parlett & Hamilton 77] M. Parlett and D. Hamilton. Evaluation as illumination: A new approach to the study of innovatory programmes. In D. Hamilton, D. Jenkins, C. King, MacDonald B., and M. Parlett, editors, *Beyond the numbers games – a reader in educational evaluation*, pages 6–22. MacMillian Education Ltd, 1977.

[Pearl 39] R. Pearl. *The natural history of population.* Oxford Un. Press, 1939.

[Polya 90] George Polya. *How to solve it: A new aspect of mathematical method.* Penguin Books Ltd., 2nd edition, 1990.

[Rader *et al.* 98] C. Rader, G. Cherry, C. Brand, A. Repenning, and C. Lewis. Designing mixed textual and iconic programming languages for novice users. In *Proceedings of the 1998 IEEE Symposium on Visual LanguagesI.* IEEE Computer Society, Halifax, Nova Scotia, November 1998.

[Rich & Waters 90] C. Rich and R.C. Waters. *The Programmer's Apprentice.* Addison-Wesley, 1990.

[Rich & Wills 90] C. Rich and L.M. Wills. Recognizing a program's design: A graph-parsing approach. *IEEE Software*, 7(1):82–89, 1990.

[Robertson 91] D. Robertson. A simple prolog techniques editor for novice users. In G.A. Wiggins, C. Mellish, and T. Duncan, editors, *3rd UK Annual Conference on Logic Programming*, pages 190–205. Springer-Verlag, Berlin, 1991.

[Robertson *et al.* 91] D. Robertson, A. Bundy, R. Muetzelfeldt, M. Haggith, and M. Eschold. *Eco-Logic: Logic-Based Approaches to Ecological Modelling.* MIT Press, 1991.

[Robson 92] Colin Robson. Designing and interpreting psychological experiments. In J. Preece and L. Keller, editors, *Human-Computer Interaction*, pages 357–367. Ellis Horwood Ltd, 1992.

[Robson 93] Colin Robson. *Real world research – A resource for Social Sciences and practitioner-researchers.* Blackwell Pub., 1993.

[Ross 85] Peter Ross. Modelling as a method of learning physical science and mathematics. DAI Research Paper 274, Department of Artificial Intelligence, University of Edinburgh, 1985.

[Ruddock 81] Ralph Ruddock. *Evaluation: A consideration of principles and methods.* Manchester University, 1981.

[Salles 98] Paulo Salles. *Qualitative Models in Ecology.* Unpublished PhD thesis, University of Edinburgh, 1998.

[Shoham 94] Yoav Shoham. *Artificial Intelligence Techniques in Prolog.* Morgan Kaufmann Pub., Inc., 1994.

[Shute & Regian 93] V.J. Shute and J.W. Regian. Principles for evaluating intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 4(3/4):245–271, 1993.

[Siemer & Angelides 98] J. Siemer and M. Angelides. A comprehensive method for the evaluation of complete intelligent tutoring systems. *Decision Support Systems*, 22:85–102, 1998.

[Sinclair 92] J. Sinclair. *Collins COBUILD English language dictionary.* HarpeCollins Pub, 1992.

[Smith *et al.* 94] D.C. Smith, A. Cypher, and J. Spohrer. Kidsim: Programming agents without a programming language. *Communications of the ACM*, 37(7):54–67, July 1994.

[SNH 98] SNH. Snh home-page. Technical report, Scottish Natural Heritage, http://fac1.vet.ed.ac.uk/ecrr/snh.htm, 1998.

[Solomon 76] M.E. Solomon. *Population Dynamics – 2nd Edition.* Edward Arnold Pub. Ltd., 1976.

[SSSS 98] SSSS. Qualitative research text resources. Technical report, School of Social and Systemic Studies, Nova Southeastern University, Fort Lauderdale, Florida., http://www.nova.edu/ssss/QR/text.html, 1998.

[Stella 98] Stella. Stella software. Technical report, High Performance Systems, Inc., http://www.hps-inc.com/products/STELLA/STELLA.html, 1998.

[Sterling & Beer 89] L. Sterling and R. Beer. Metainterpreters for expert systems construction. *Journal of Logic Programming*, 6(1–2):163–178, 1989.

[Sterling & Kirschenbaum 93] L.S. Sterling and M. Kirschenbaum. Applying techniques to skeletons. In J.-M. Jacquet, editor, *Constructing Logic Programs*, chapter 6, pages 127–140. John Wiley & Sons Ltd, 1993.

[Sterling & Shapiro 94] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques.* MIT Press, 2nd edition, 1994.

[SYDPOL 90] SYDPOL. Issues in evaluation of computer-based support to clinical decision making. Research Report 127, Universitetet i Oslo – Institutt for Informatikk, 1990.

[Twidale 93] Michael Twidale. Redressing the balance: The advantages of informal evaluation techniques for intelligent learning environments. *Journal of Artificial Intelligence in Education*, 4(3/4):155–178, 1993.

[Vargas-Vera 95] M. Vargas-Vera. *Using Prolog techniques to guide program composition.* Unpublished PhD thesis, University of Edinburgh, 1995.

[Vargas-Vera *et al.* 93] M. Vargas-Vera, W. Vasconcelos, and D. Robertson. Building large-scale prolog programs using a techniques editing system. Research Paper 635, Department of Artificial Intelligence, University of Edinburgh, 1993.

[Vasconcelos 93] W. W. Vasconcelos. Designing prolog programming techniques. In Y. Deville, editor, *Logic program synthesis and transformation (LOPSTR 93) – 3rd International Workshop, Louvain-la-Neuve*, pages 85–99. Springer-Verlag, 1993.

[Vasconcelos 94] W. W. Vasconcelos. Extracting prolog programming techniques. In T. Pequeno and F. Carvalho, editors, *Proceedings of the XI Brazilian Symposium on Artificial Intelligence*, pages 269–283. Brazilian Computer Society, 1994.

[Vasconcelos 95] W. Vasconcelos. *Extracting, organising, designing and reusing Prolog programming techniques.* Unpublished PhD thesis, University of Edinburgh, 1995.

[Weiser 82] M. Weiser. Programmers use slices when debugging. *Communications of the ACM*, 25(7):446–452, July 1982.

[Weiser 84] M. Weiser. Program slicing. *IEEE Transactions on Software Engineering*, SE-10(4):352–357, July 1984.

[Wielinga *et al.* 92] B.J. Wielinga, A.T. Schreiber, and J. Breuker. Kads: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1):5–53, 1992.

[Willoughby 91] A. Willoughby. Further developments in qualitative graphical reasoning. Technical Report hcrl-tr-66, Human Cognition Research Laboratory, The Open University, Milton Keynes, England, 1991.

[Winne 93] Philip Winne. A landscape of issues in evaluating adaptive learning systems. *Journal of Artificial Intelligence in Education*, 4(3/4):273–296, 1993.

[Wyatt & Spiegelhalter 90] J. Wyatt and D. Spiegelhalter. Evaluating medical expert systems: what to test and how? *Medical Informatics*, 15(3):205–217, 1990.

Appendix A

Algorithms

A.1 Program generation I

The following algorithm shows how a part of the model description is used to set up data-structures for program generation.

$$enter\_process(C, P, CP, E)$$

where: C is the component name; P is the defining process; CP is the category of the defining process and E is the process' equation.

enter_process(C, P, CP, E)
IF C has a non-disaggregated population THEN
IF P is defined by user (new equation) THEN

- check the equation.
- get (from the knowledge base) the first part of techniques sequence and binding records, those corresponding to equation-defined and non-disaggregated kind of process definition. That is: `Sequence1 = [proc_head, def_comp, applicability, compute_eq]` and corresponding bindings.
- get second part of techniques and binding sequences, that is, the one assigning different behaviours for different categories of processes. For example, if CP = natality, then `Behaviour = distribute_on_first_level`.
- build construction records (ccrs) for P.
  For the case above (CP = natality), the technique-editing operations sequence would be:

```
TS = [unif_pred_name(proc_head,Proc,C1),
      add_technique(def_comp,C1,C2),
      add_technique(applicability,C2,C3),
      add_technique(compute_eq,C3,C4),
      add_technique(distribute_on_first_level,C4,Out)]
```

- include P in the list of candidates, assert corresponding ccr, binding records and equation.

ELSE IF P is selected from predefined set THEN
CASE P is defined by
equation THEN
proceed as for user-defined process;
schema THEN
- use arguments for instantiating the schema - no ccr/binding records are needed;
- include P in the list of candidates.
technique composition THEN
- define supplementary data-structures;
- get corresponding ccr and bindings for the composition. The user may be prompted for choosing amongst several choices.
- include P in the list of candidates and assert ccr/binding record.
END CASE
END IF
ELSE (population of C is disaggregated)
proceed as for non-disaggregated population, but with suitable selection from the knowledge base.
END IF

A.2 Program generation II

This the algorithm for defining the predicate `update/4`, which contains the sequence of processes defined for each component.

$$compose\_update(C, P)$$

where: C is the component name; P is the set of processes for that component.

compose_update(C, P)
IF there is no sequence started THEN
- start sequence: `TS = [unif_pred_name(proc_head,update,Code), add_technique(def_comp2,Code,C1)]`
- assert last binding record

END IF
IF P is an empty set THEN
- add last goal to sequence:
  `TS = [ ... , add_technique(adjust,LastCode,CodeOut) ]`
- assert binding record
- include `update/4` in the list of candidates

ELSE
- take the first process (P) from P, let T be the rest of that set
- include P in the sequence (process call):
  `TS = [ ... , add_technique(P,PreviousCode,NewCode) ]`
- *compose_update*(C, T)

• assert binding record
END IF

A.3 Meta interpreter

The following is the definition of our meta-interpreter.

$$\text{check}(F, R)$$

where F is a formula to some variable in a model and R is its value (at certain time point) during simulation.

check(F, R)
if F is a number **then**
succeed
else if F is a variable name **and** $F \in$ variables – list **then** {non-indexed variable}
succeed
else if F is on the form "$A\$B$" **and** $A \in$ variables – list **then** {indexed variable}
check_index(B) {check indexes names and/or values}
else if F is of the form "IF C THEN E_1 ELSE E_2" **then**
check_condition(C)
check(E_1) {reapply to check first expression}
check(E_2)
else if F is of the form "graph(X)" **then** {graphic-defined}
if X is a variable name **and** $F \in$ variables – list **then**
succeed
else if X is of the form "$A\$B$" **and** $A \in$ variables – list **then**
check_index(B)
end if
else if F is of the form "$X(E)$" **and** X is a Prolog built-in **then**
check(E)
else if F is of the form "$X(E_1,E_2)$" **and** X is a Prolog built-in **then**
check(E_1)
check(E_2)
else
error processing
end if

A.4 Techniques editor I

This operation extends the number of arguments in a defining predicate.

$$\textit{include\_argument}(AN, O, C, NewC)$$

where: AN is the new argument's name, C is the working code, $NewC$ is the code after the introduction of the new argument and O is an option indicating whether the new argument will unify (binding) in the head and recursive calls of the defining predicate.

We denote the *head* and *body* of a clause C_i ($C_i \in C$) by C_i:head and C_i:body, respectively.

include_argument$(AN, O, C, NewC)$
  $NewC = \{\}$
  for all C_i in C **do**
    $NewC_i = \{\}$
    $NewC_i$:head = C_i:head
    include A into $NewC_i$:head
    if C_i is a recursive clause **then**
      $NewC_i$:body = C_i:body
      if O **then**
        include A into the recursive goal(s) in $NewC_i$:body
      else
        include an *anonymous* variable to the recursive goal in $NewC_i$:body
      end if
    end if
  end for

A.5 Techniques editor II

This operation add subgoals to the clauses of a defining predicate.

$$add\_technique(T, C, NewC)$$

where: T is the technique to be applied, C is the working code and $NewC$ is the new code produced.

We denote the *head* and *body* of a clause C_i ($C_i \in C$) by C_i:head and C_i:body, respectively; and the operation of appending something to a set by $\uplus$.

add_technique$(T, C, NewC)$
  get clauses TC of technique T
  if there is a building record **then**
    get B {binding set}
  else
    "pretty-display" C and TC
    prompt user for B
  end if
  $NewC = \{\}$
  $NewC$:arg = C:arg
  $NewC$:var = C:var
  $P = C$:arg $\uplus$ C:var
  $k = 0$
  for all B_i in B **do**
    if B_i is a variable **then**
      include B_i into $NewC$:var
    else if B_i is a constant **then** {B_i is an index for P}

bind TC_k:arg to P_{B_i}
end if
$k = k + 1$
end for
for all TC_i in TC **and** C_i in C **do**
$NewC_i$:head = C_i:head
if standard composition **then** {subgoals add at the end of body}
$NewC_i$:body = C_i:body $\uplus$ TC_i:body
else {subgoals *on-top* of body}
$NewC_i$:body = TC_i:body $\uplus$ C_i:body
end if
end for

Appendix B

Knowledge base

```
% kb.pl  -  Last updated on 20/10/97

%=============================================================================
%   PART [A] - PRE-DEFINED PARAMETERS.
%=============================================================================

%
% Pre-defined components (Menu options)
%
menu_s('Components',[fox, rabbit, deer, moose, wolf, sheep, bear]).
menu_s(new_component,[q1, insect,bird,fish,mammal,other]).
menu_s(new_component,[q2, vertebrate,invertebrate]).
menu_s(new_component,[q3, carnivore,herbivore,omnivore]).
menu_s('Organisation1',['grouped/lumped',individual]).
menu_s('Attributes',[age, sex, weight, size, location]).
dim_info(_,age,ordered).
dim_info(_,weight,ordered).
dim_info(_,size,ordered).

%
% Time-related info.
%
time_units(['time-steps(gen.)','year={season,season}',seasons,months]).
time_hier(year,[season,season]).     time_hier(season,summer).
time_hier(season,winter).            time_hier(summer,[jun,jul,aug,sep,oct]).
time_hier(winter,[nov,dec,jan,feb,mar,apr,may]).
time_hier(month,jan).    time_hier(month,feb).    time_hier(month,mar).
time_hier(month,apr).    time_hier(month,may).    time_hier(month,jun).
time_hier(month,jul).    time_hier(month,aug).    time_hier(month,sep).
time_hier(month,oct).    time_hier(month,nov).    time_hier(month,dec).
time_hier(month,[week,week,week,week]).
time_hier(week,[day,day,day,day,day,day,day]).
time_hier(day,mon).      time_hier(day,tue).      time_hier(day,wed).
time_hier(day,thu).      time_hier(day,fri).      time_hier(day,sat).
time_hier(day,sun).

%
% Component's sort hierarchy.
%
menu_s('Components',[fox, rabbit, deer, moose, wolf, sheep, bear]).
component_sort(fox,mammal).          component_sort(fox,vertebrate).
component_sort(fox,carnivore).       component_sort(rabbit,mammal).
component_sort(rabbit,vertebrate).   component_sort(rabbit,herbivore).
component_sort(deer,mammal).         component_sort(deer,vertebrate).
component_sort(deer,herbivore).      component_sort(moose,mammal).
component_sort(moose,vertebrate).    component_sort(moose,herbivore).
component_sort(wolf,mammal).         component_sort(wolf,vertebrate).
component_sort(wolf,carnivore).      component_sort(sheep,mammal).
component_sort(sheep,vertebrate).    component_sort(sheep,herbivore).
component_sort(bear,mammal).         component_sort(bear,vertebrate).
component_sort(bear,omnivore).

%
%  add user-defined components
%
component_sort(Comp,Sort) :-
  predicate_property(component_sort(_,_,_),_),    % check if there is any
  component_sort(Comp,_,Sort).

%
%  Defining a (possible) predator-prey relationship
%
would_be_predator(A,B) :-
  components(L),
```

```
  member(A,L),
  member(B,L),
  \+ A = B,
  ( component_sort(A,carnivore) ;
    component_sort(A,omnivore) ),
  component_sort(B,_),!.     % check it out with new_component

%
%  Cross-examination a process selected.
%       Succeds if a process is confirmed by user or no confirmation is needed.
%       (it will be used by predicate 'validate/4' on 'TechGen' module.
%
cross_examination(Component,immigration) :-
  \+ factor(Component,[location|_]),!,
  confirm(Component,immigration,'You have selected immigration. However,
                                 there is no location representation.').
cross_examination(Component,emmigration) :-
  \+ factor(Component,[location|_]),!,
  confirm(Component,emmigration,'You have selected emmigration. However,
                                 there is no location representation.').
cross_examination(Component,movement_between_classes) :-
  \+ factor(Component,[_,categories|_]),!,
  confirm(Component,movement_between_classes,
   'You have selected movement_between_classes. However, no categories were
    found among attribute representation.').
cross_examination(_Component,_).


%=============================================================================
%   PART [B] - PRE-DEFINED PROCESSES
%=============================================================================

*/-----------------------------------------------------------------------------

   eco_process_app( +Comp ,    ?ProcGroup ,   ?ProcName , ?ProcDescription ,
                    ?Equation, ?EqVariables , ?EffectOnPopulation
                  )
                  :-  "Applicability"

   where:
   Comp             = component's name.
   ProcGroup        = category to which the process belongs to.
   ProcName         = name of defining process.
   ProcDescription  = process' description (presentation text).
   Equation         = process' equation.
   EqVariables      = variables (other the population) in Equation.
   EffectOnPopulation = technique associated with the effect of the process
      on the population of the component:
      if non-disaggregated population: only increasing/decreasing;
      if disaggregated population: the effect for this category of processes.
*/


%
%  disaggregated( +C ) - succeeds if (C) is a component with a disaggregated
%   population. (!) is used to avoid several instances of the same description
%   to be produced when there is more than one disaggregation criteria.
disaggregated(Component) :-
    disaggregation(Component,C,_,_),
    \+ C = none,!.


%  NATALITY ---------------------------------------------------------------
%
%  Population is increased by a number of births computed using individual
%  reproductive rates.
%

%
%             NON-disaggregated population.
%

% --------------
eco_process_app(Component, natality, cte_rep_rate,
   'The number of births is computed using a constant individual reproduction rate.',

   'rep_rate_Component*Component',

   [[rep_rate_Component,['Individual reproductive rate for Component']]],
   incr_process_by_formula
   ) :-
        \+ disaggregated(Component).

% --------------
eco_process_app(Component, natality, logistic_rep_rate,
   'The number of births is computed using an individual reproduction rate defined by the function r_max(1-pop/pop_max).\n
Logistic growth is based on this function.',

   'max_rep_rate_Component*(1-Component/max_pop_Component)*Component',

   [ [max_rep_rate_Component,['Maximum value for individual reproductive rate of Component']],
```

```
    [max_pop_Component,['Maximum value for Component population']]],
  incr_process_by_formula
  ) :-
      \+ disaggregated(Component).

% --------------
eco_process_app(Component, natality, two_levels_r,
  'The number of births is computed using an individual reproduction rate defined by one of two values, depending on whether population
has reached certain level.',

  'if Component >= trigger_pop_Component then r1_Component else r2_Component',

  [ [trigger_pop_Component,['Value of Component population when other value for individual reproductive rate must be used']],
    [r1_Component,['Value of Component individual reproductive rate BEFORE population reaches trigger-value']],
    [r2_Component,['Value of Component individual reproductive rate AFTER population has reached trigger-value']]],
  incr_process_by_formula
  ) :-
      \+ disaggregated(Component).

%
%           Disaggregated population.
%

% --------------
eco_process_app(Component, natality, cte_rep_rate,
  'The number of births is computed using a constant individual reproduction rate for all sub-populations.',

  'rep_rate_Component*Component$Original_DisAgg',

  [[rep_rate_Component,['Individual reproductive rate for Component']]],
  natality
  ) :-
      disaggregated(Component).

% --------------
eco_process_app(Component,  natality, rep_rate,
  'The number of births is computed using an individual reproduction rate for each sub-group of the population.',

  'rep_rate_Component$_disaggregated_*Component$Original_DisAgg',

  [[rep_rate_Component,['Individual reproductive rate for sub-population of Component']]],
  natality
  ) :-
      disaggregated(Component).

% --------------
eco_process_app(Component, natality, logistic_rep_rate,
  'The number of births is computed using an individual reproduction rate defined by the function r_max(1-pop_total/pop_max).\n
Logistic growth is based on this function.',

  'max_rep_rate_Component*Component$Original_DisAgg*(1-Component/max_pop_Component)',

  [ [max_rep_rate_Component,['Maximum value for individual reproductive rate of Component']],
    [max_pop_Component,['Maximum value for Component population']]],
  natality
  ) :-
      disaggregated(Component).

% --------------
eco_process_app(Component, natality, two_levels_cte_r,
  'At each sub-group of the population, the number of births is computed using a constant individual reproduction rate defined by one
of two values, depending on whether the total population has reached certain level.',

  'if Component >= trigger_pop_Component then r1_Component*Component$Original_DisAgg else r2_Component*Component$Original_DisAgg',

  [ [trigger_pop_Component,['Value of Component population (trigger-value) to which other value for individual reproductive rate must be used']
    [r1_Component,['Value of Component individual reproductive rate BEFORE population reaches trigger-value']],
    [r2_Component,['Value of Component individual reproductive rate AFTER population has reached trigger-value']]],
  natality
  ) :-
      disaggregated(Component).

% --------------
eco_process_app(Component, natality, two_levels_r,
  'The number of births at each population is computed using an individual reproduction rate defined by one of two values, depending on
whether that sub-population has reached a certain level.',

  'if Component$Original_DisAgg >= trigger_pop_Component$_disaggregated_ then r1_Component$_disaggregated_*Component$Original_DisAgg else
r2_Component$_disaggregated_*Component$Original_DisAgg',

  [ [trigger_pop_Component,['Value of sub-population (trigger-value) to which other value for individual reproductive rate must be used']],
    [r1_Component,['Value of individual reproductive rate BEFORE population reaches trigger-value']],
    [r2_Component,['Value of individual reproductive rate AFTER population has reached trigger-value']]],
  natality
  ) :-
      disaggregated(Component).


%  MORTALITY ------------------------------------------------------------------
%
%  Population is decreased by a number of deaths computed using a rate which
%  corresponds to the probability of death.
```

```
%

%
%            NON-disaggregated population.
%

% --------------
eco_process_app(Component, mortality, cte_mortality_rate,
   'The number of deaths is computed using a rate which corresponds to the (individual) probability of death by combination of several factors.'

   'mort_rate_Component*Component',

   [[mort_rate_Component,['Individual prob. rate of death for Component']]],
   decr_process_by_formula
   ) :-
        \+ disaggregated(Component).

% --------------
eco_process_app(Component, mortality, predation,
   'The number of deaths by predation is computed using a variable which represents the eating rate of each individual from a predator populatio

   'eat_Component*predator_Component',

   [[eat_Component,['Individual eating rate of Component by a predator.']]],
   decr_process_by_formula
   ) :-
        \+ disaggregated(Component),
        would_be_predator(_Predator,Component).

%
%            Disaggregated population.
%

% --------------
eco_process_app(Component, mortality, cte_mortality_rate,
   'The number of deaths at each sub-group of the population is computed using a constant rate corresponding to the (individual)
probability of death by combination of several factors.',

   'mort_rate_Component*Component$Original_DisAgg',

   [[mort_rate_Component,['Individual prob. rate of death for Component']]],
   mortality
   ) :-
        disaggregated(Component).

% --------------
eco_process_app(Component, mortality, mortality_rate,
   'The number of deaths at each sub-group of the population is computed using
a rate corresponding to the (individual) probability
of death by combination of several factors.',

   'mort_rate_Component$_disaggregated_*Component$Original_DisAgg',

   [[mort_rate_Component,['Individual prob. rate of death for Component']]],
   mortality
   ) :-
        disaggregated(Component).

% --------------
eco_process_app(Component, mortality, predation1,
   'The number of deaths by predation is computed using a variable which represents the eating rate of each individual from a predator populatio

   'eat_Component*predator_Component',

   [[eat_Component,['Individual eating rate of Component by a predator.']]],
   mortality
   ) :-
        disaggregated(Component),
        would_be_predator(_Predator,Component).


%  PROGRESS ON CATEGORY  ------------------------------------------------
%
%  The elements that form a population disaggregated according to certain
%  category (dimension of classification), periodically progress (move to the
%  next value) in that category.
%  Since all sub-poopulation is shifted, the dimension name is used instead of
%  an equation.
%  This process is applicable only on dimensions with ordered values.
%

% --------------
eco_process_app(Component, progress_on_category, ageing,
   'The population progresses in age. Elements in the first age class are shifted to the next one and so on.',
   age,
   na,
   progress_on_category
   ) :-
        disaggregation(Component,age,_,_).

% --------------
eco_process_app(Component, progress_on_category, size_growing,
```

```
   'The population progresses in size. Elements in the first size class are shifted to the next one and so on.',
   size,
   na,
   progress_on_category
   ) :-
        disaggregation(Component,size,_,_).


%=============================================================================
%   PART [C] - TECHNIQUES LIBRARY.
%=============================================================================

/*
   SYNTAX - Both techniques and schemata use the same syntax. TeMS recognises
            them by checking whether the name is instantiated (schema) or not
            (technique).

   technique( name , [ internal_variables , definition_1 ,..., definition_n ] )

   internal_variables = [ v_1 ,..., v_n ]
   definition = [ Head , BodyPart_1 ,..., BodyPart_n ]
   definition = rec( [ Head , BodyPart_1 ,..., BodyPart_n ] )
   Head       = [Functor , Parameter_1 ,..., Parameter_n]
   BodyPart_i = [Functor , Parameter_1 ,..., Parameter_n]
*/

%-----------------------------------------------------------------------------
%     TECHNIQUES
%-----------------------------------------------------------------------------

technique(time_recursion_back,                % recursion over time points
  [ [Tp] ,
    [     [_P,T] ,
          [initial_time,T] ],
    rec([ [_P,T] ,
          [\+,[initial_time,T]],
          [previous_time,T,Tp] ,
          [_P,Tp] ])
  ]).

technique(get_value_back,                     % ''population updating''
  [ [] ,
    [ [_P,T,V,Vp] , [start_value,V] ] ,
    [ [_P,T,V,Vp] , [update_population,T,Vp,V] ]
 ]).

technique(start,
  [ [] ,
    [ [_P,FinalTime,Population] ,
        [population,FinalTime,Population] ]
  ]).

technique(open_result_file,
  [ [] ,
    [ [_P] ,
      [open_dc] ]
  ]).
technique(close_result_file,
  [ [] ,
    [ [_P] ,
      [close_dc] ]
  ]).
technique(update_result_file,                 % data_collection
  [ [] ,
    [ [_P,Time,Population],
          [collect,Time,Population] ],
    [ [_P,Time,Population],
          [collect,Time,Population] ]
  ]).

technique(update_head,
  [ [CompName,CompOrg,CompPop] ,
    [ [_P,_Time,_PopRef,PopiComp,_NewCompPop] ,
      [ =,PopiComp,['.',CompName,['.',CompOrg,['.',CompPop,'[]']]] ] ]
  ]).

technique(def_comp,
  [ [],
    [ [_P,CompName,X] , [=,CompName,X] ] ,
    [ [_P,CompName,X] , [=,CompName,X] ] ]).
technique(def_comp2,
  [ [],
    [ [_P,CompName,X] , [=,CompName,X] ] ]).

technique(def_org,
  [ [],
    [ [_P,CompOrg,X] , [=,CompOrg,['.',X,_]] ],
    [ [_P,CompOrg,X] ] ]).

technique(def_no_disaggregation,
  [ [],
```

```
    [ [_P,CompOrg,X] , [=,CompOrg,X] ],
    [ [_P,CompOrg,X] ] ]).

technique(def_time,
  [ [],
    [ [_P,TimeStep,TimeRef] ,
      [season,TimeStep,TimeRef] ],
    [ [_P,TimeStep,TimeRef] ] ]).

technique(def_sort,
  [ [],
    [ [_P,Sort,X] , [=,Sort,X] ],
    [ [_P,Sort,X] ] ]).

technique(pop1comp,
  [ [],
    [ [_P,Pop1Comp,CompName,CompOrg,CompPop],
      [=,Pop1Comp,['.',CompName,['.',CompOrg,['.',CompPop,'[]']]]] ],
    [ [_P,Pop1Comp,CompName,CompOrg,CompPop] ] ]).


technique(proc_head,
  [ [CompName,CompOrg,CompPop] ,
    [ [_P,_Time,_Ref,Pop1Comp,_I,_O] ,
      [=,Pop1Comp,['.',CompName,['.',CompOrg,['.',CompPop,'[]']]]] ],
    [ [_P,_Time,_Ref,Pop1Comp,_I,_I] ,
      [=,Pop1Comp,['.',CompName,['.',CompOrg,['.',CompPop,'[]']]]] ]
  ]).
technique(incr_process_by_formula,    % process defined by formula
  [ [],                               % increasing effect on population
    [ [_P,Time,PopAllComp,Comp,InPopComp,OutPopComp,ProcessName],
      [selected_proc,Comp,_,ProcessName,[f_,Formula,FormulaDimensions],TimeRef],
      [apply_on_time,Time,TimeRef],
      [compute_formula,Time,Comp,PopAllComp,Formula,FormulaDimensions,Births],
      [is,OutPopComp,[+,InPopComp,Births]] ],
    [ [_P,Time,PopAllComp,Comp,InPopComp,InPopComp,ProcessName] ]
   ]).
technique(decr_process_by_formula,    % process defined by formula
  [ [],                               % decreasing effect on population
    [ [_P,Time,PopAllComp,Comp,InPopComp,OutPopComp,ProcessName],
      [selected_proc,Comp,_,ProcessName,[f_,Formula,FormulaDimensions],TimeRef],
      [apply_on_time,Time,TimeRef],
      [compute_formula,Time,Comp,PopAllComp,Formula,FormulaDimensions,Deaths],
      [is,R,[-,InPopComp,Deaths]],
      [is,OutPopComp,[max,R,0]] ],    % just avoiding negative values
    [ [_P,Time,PopAllComp,Comp,InPopComp,InPopComp,ProcessName] ]
   ]).
technique(natality,                   % process natality causes increasing
  [ [],                               % on "first level" of population
    [ [_P,Time,PopAllComp,Comp,InPopComp,OutPopComp,ProcessName,Dims,CompPop],
      [selected_proc,Comp,_,ProcessName,[f_,Formula,FormulaDimensions],TimeRef],
      [apply_on_time,Time,TimeRef],
      [compute_formula,Time,Comp,PopAllComp,Formula,FormulaDimensions,Births],
      [distribute_on_first_level,Comp,Dims,InPopComp,Births,OutPopComp] ],
    [ [_P,Time,PopAllComp,Comp,InPopComp,InPopComp,ProcessName,Dims,CompPop] ]
   ]).
technique(mortality,                  % process mortality causes
  [ [],                               % decreasing effect on population
    [ [_P,Time,PopAllComp,Comp,InPopComp,OutPopComp,ProcessName,Dims,CompPop],
      [selected_proc,Comp,_,ProcessName,[f_,Formula,FormulaDimensions],TimeRef],
      [apply_on_time,Time,TimeRef],
      [compute_formula,Time,Comp,PopAllComp,Formula,FormulaDimensions,Deaths],
      [subtract_list_values,InPopComp,Deaths,OutPopComp] ],
    [ [_P,Time,PopAllComp,Comp,InPopComp,InPopComp,ProcessName,Dims,CompPop] ]
   ]).
technique(progress_on_category,    % Zeroes are moved to 1st sub-category
  [ [],                            % Previous values of last sub-cat. are lost
    [ [_P,Time,PopAllComp,Comp,InPopComp,OutPopComp,ProcessName,Dims,CompPop],
      [selected_proc,Comp,_ProcessGroup,ProcessName,DimensionOfProgress,TimeRef],
      [apply_on_time,Time,TimeRef],
      [shift_population,Comp,DimensionOfProgress,CompPop,ShiftedInPop],
      [add_list_values,ShiftedInPop,InPopComp,OutPopComp] ],
    [ [_P,Time,PopAllComp,Comp,InPopComp,InPopComp,ProcessName,Dims,CompPop] ]
   ]).

technique(adjust_values,
  [ [],
    [ [_P,I,O] , [adj_values,I,O] ]
  ]).

technique(X,Body) :-
  predicate_property(eco_process(_,_),_),     % if there is at least one...
  eco_process(X,Body).


%-------------------------------------------------------------------------
%     Triggers and suggestions of technique application
%-------------------------------------------------------------------------
%  A technique may have a "trigger list" with the next definitions needed
%  to make a runnable code.
%  Some predicates are represented as schemata, and are automacticly triggered.
%
```

```
:- assert(trigger(start,[(population,[time_recursion_back,get_value_back])])).
:- assert(trigger(time_recursion_back,[(initial_time,schema),(previous_time,schema)])).
:- assert(trigger(get_value_back,[(start_value,schema),(update_population,schema)])).

:- assert(trigger(start_value,[(get_components,schema),(initial_state,schema)])).
:- assert(trigger(initial_state,[(initial_value,schema)])).
:- assert(trigger(update_population,[(update_components,schema),(adj_values,schema)])).

:- assert(trigger(natality,[(distribute_on_first_level,schema)])).
:- assert(trigger(mortality,[(subtract_list_values,schema)])).
:- assert(trigger(progress_on_category,[(shift_population,schema),(add_list_values,schema)])).


%  Extensions to the code after a technique be used may be indicated.
%
suggested_extensions(start,[open_result_file,close_result_file]).


%-------------------------------------------------------------------------
%     SCHEMATA
%     Comments to schemata may include a more legible version of the code
%     embodied on them.
%-------------------------------------------------------------------------

technique(start_value,
  [ [],
    [ [start_value,Population] ,
      [get_components,C],
      [initial_state,C,Population] ] ]).

technique(get_components,
  [ [],
    [ [get_components,C] , [components,C] ] ]).

technique(initial_state,
  [ [],
    [ [initial_state,'[]','[]'] ] ,
    rec([ [initial_state,['.',H,T],['.',Pop,RestPop]] ,
          [initial_value,H,Pop] ,
          [initial_state,T,RestPop] ])  ]).

technique(initial_value,
  [ [],
    [ [initial_value,Component,['.',Component,['.',none,['.',N,'[]']]]] ,
      [disaggregation,Component,none,_,_] ,
      [initial_st,Component,['.',none,'[]'],N] ,
      [!] ] ,

    [ [initial_value,Component,['.',Component,['.',Dims,['.',Values,'[]']]]] ,
      [findall,D,[disaggregation,Component,D,_,_],Dims] ,
      [\+,[=,Dims,'[]']] ,
      [findall,['.',C,['.',D,'[]']],[initial_st,Component,C,D],Values] ]  ]).

technique(previous_time,
  [ [],
    [ [previous_time,T,NexT] ,
      [is,NexT,['-',T,1]] ]  ]).

technique(initial_time,
   [ [] ,
     [ [initial_time,0] ]   ]).

technique(update_population,
  [ [],
    [ [update_population,T,LastPop,NewPop] ,
          [update_components,T,LastPop,LastPop,NewPop] ]   ]).

technique(update_components,
  [ [],
    [ [update_components,Time,LastPop,'[]','[]'] ] ,
    rec([ [update_components, Time, LastPop,
              ['.',['.',CompName,['.',CompOrg,['.',CompPop,'[]']]],T1],
              ['.',['.',CompName,['.',CompOrg,['.',NewCompPop,'[]']]],T2] ] ,
          [update,Time,LastPop,
              ['.',CompName,['.',CompOrg,['.',CompPop,'[]']]],
              NewCompPop],
          [update_components,Time,LastPop,T1,T2] ])  ]).

technique(adj_values,
/*--------------
  % adj_values( +Population, -AdjustedPopulation )
  % Check Population for negative numbers (previously changed -x for 0).
  adj_values(P,0) :-
    number(P),
    P<0,
    show_err_msg(negative_pop),!.
  adj_values(P,P) :-
    number(P).
  adj_values([],[]).
  adj_values([[Group,P]|T],[[Group,0]|T]) :-
    P<0,
    show_err_msg(negative_pop),!.
```

```
  adj_values([[Group,P]|T1],[[Group,P]|T2]) :-
    adj_values(T1,T2).
  --------------*/
  [ [],
    [ [adj_values,P,0],
      [number,P],
      [<,P,0],
      [show_err_msg,negative_pop], [!] ] ,
    [ [adj_values,P,P],
      [number,P] ] ,
    [ [adj_values,'[]','[]'] ] ,
    [ [adj_values,['.',['.',Group,['.',P,'[]']],T],
                  ['.',['.',Group,['.',0,'[]']],T] ],
      [<,P,0],
      [show_err_msg,negative_pop], [!] ] ,

    rec([ [adj_values,['.',['.',Group,['.',P,'[]']],T1],
                      ['.',['.',Group,['.',P,'[]']],T2] ],
          [adj_values,T1,T2] ]) ]).

technique(distribute_on_first_level,
/*--------------
  % distribute_on_first_level( +Component,+Dimensions,
  %                             +InPop,+CompBirths,-PopResult )
  % Sum the births of the population of a Component (CompBirths) and
  % distributes it over the first level of component's population - CompPop
  % (that is the parcel of population referred by the first value of a
  % dimension of classification - the dimension considered is the first of
  % Dimensions).
  distribute_on_first_level( _Component,[Dimension],  %% Only 1 dimension
                             [[SubG,N]|T] , Births  , [[SubG,NewN]|T] ) :-
    atom(Dimension),
    list_sum(Births,Sum),
    NewN is N+Sum.
  distribute_on_first_level(Comp,Dimensions,PopIn,PopBirth,PopR) :-
    list_sum(PopBirth,Sum),
    first_level(Comp,Dimensions,Match),
    findall(P,member([Match,P],PopIn),L),
    length(L,N),
    Parcel is Sum/N,
    add_parcel(Match,Parcel,PopIn,PopR).
  --------------*/
  [ [],
    [ [distribute_on_first_level,Component,['.',Dimension,'[]'],
                                 ['.',['.',SubG,['.',N,'[]']],T],
                                 Births,
                                 ['.',['.',SubG,['.',NewN,'[]']],T]],
      [atom,Dimension],
      [list_sum,Births,Sum],
      [is,NewN,['+',N,Sum]] ] ,
    [ [distribute_on_first_level,Component,Dimensions,PopIn,PopBirth,PopR],
      [list_sum,PopBirth,Sum],
      [first_level,Component,Dimensions,Match],
      [findall,P,[member,['.',Match,['.',P,'[]']],PopIn],L],
      [length,L,N],
      [is,Parcel,[/,Sum,N]],
      [add_parcel,Match,Parcel,PopIn,PopR] ] ,

/*--------------
  % list_sum ( +List , -Sum )          List as in: [ [ SubGroup , Value ] | T ]
  list_sum([],0).
  list_sum([[_,V1]|T],S) :-
    list_sum(T,Acc),
    S is V1+Acc.
  --------------*/
    [ [list_sum, '[]',0 ] ] ,
    rec([ [list_sum, ['.',['.',_,['.',V1,'[]']],T] ,S ] ,
          [list_sum,T,Acc],
          [is,S,['+',V1,Acc]] ]) ,

/*--------------
  %  first_level(+Comp,+Dimensions,-Match),
  %
  %  Match is a list of the same length of Dimensions. An element of Match is
  %  the first value when the correspondent element in Dimensions is an
  %  ordered dimension, or an anonymous variable otherwise.
  %
  first_level(_,[],[]).
  first_level(Comp,[Dim|T1],[First|T2]) :-
    dim_info(Comp,Dim,ordered),
    disaggregation(Comp,Dim,_,[First|_]),
    first_level(Comp,T1,T2),!.
  first_level(Comp,[_|T1],[_|T2]) :-
    first_level(Comp,T1,T2).
  --------------*/
    [ [first_level,_,'[]','[]'] ] ,
    rec([ [first_level,Comp,['.',Dim,T1],['.',First,T2]],
          [dim_info,Comp,Dim,ordered],
          [disaggregation,Comp,Dim,_,['.',First,_]],
          [first_level,Comp,T1,T2],
          [!] ]) ,
    rec([ [first_level,Comp,['.',_,T1],['.',_,T2]],
```

```
        [first_level,Comp,T1,T2] ]) ,

/*--------------
  %  add_parcel(+Match,+Parcel,+PopIn,-PopR).
  %  Add Parcel to those elements (the population part) which match Match.
  add_parcel(_,_,[],[]).
  add_parcel(Match,Parcel,[[TrabMatch,InPop]|T1],[[TrabMatch,R]|T2]) :-
    copy_term(Match,TrabMatch),
    R is InPop+Parcel,
    add_parcel(Match,Parcel,T1,T2),!.
  add_parcel(Match,Parcel,[In|T1],[In|T2]) :-
    add_parcel(Match,Parcel,T1,T2).
  --------------*/
    [ [add_parcel,_,_,'[]','[]'] ] ,

    rec([ [add_parcel,Match,Parcel,
                      ['.',['.',TrabMatch,['.',InPop,'[]']],T1],
                      ['.',['.',TrabMatch,['.',R,'[]']],T2] ],
          [copy_term,Match,TrabMatch],
          [is,R,[+,InPop,Parcel]],
          [add_parcel,Match,Parcel,T1,T2],
          [!] ]) ,

    rec([ [add_parcel,Match,Parcel,['.',In,T1],['.',In,T2]],
          [add_parcel,Match,Parcel,T1,T2] ]) ]).

technique(shift_population,
/*--------------
  % shift_population( +Component, +Dimension, +PopIn, -PopOut )
  % Given a population disaggregated in several categories (dimensions), SHIFTS
  % the population considering ONE specific dimension (dimension of progress).
  % Each value of the dimension of progress defines a LEVEL, zeroes are moved
  % to the first level. The values (population values) previously on the 1st
  % level are moved to the next level and so on.
  shift_population(Component,Dimension,PopIn,PopOut) :-
    disaggregation(Component,Dimension,_,Values),    % get Dimension's values
    findall(D,disaggregation(Component,D,_,_),Dims), % get all dimension names
    length(Dims,N),                                  % how many dimensions?
    pos_list(Dimension,Dims,Pos),        % position of Dimension on dim. list
    Values=[First|_],
    make_list(N,Pos,First,Match),
    findall(P,member([Match,P],PopIn),Lsample),    % get "first level" of pop.
    length(Lsample,Nlevel),
    list_of_cte(Nlevel,0,From),         % list of 0 will be moved to 1st level
    traverse_dimension_list(Dimension,Values,N,Pos,From,PopIn,PopOut).
  --------------*/
  [ [],
    [ [shift_population,Component,Dimension,PopIn,PopOut],
      [disaggregation,Component,Dimension,_,Values],
      [findall,D,[disaggregation,Component,D,_,_],Dims],
      [length,Dims,N],
      [pos_list,Dimension,Dims,Pos],
      [=,Values,['.',First,_]],
      [make_list,N,Pos,First,Match],
      [findall,P,[member,['.',Match,['.',P,'[]']],PopIn],Lsample],
      [length,Lsample,Nlevel],
      [list_of_cte,Nlevel,0,From],
      [traverse_dimension_list,Dimension,Values,N,Pos,From,PopIn,PopOut] ],

/*--------------
  % traverse_dimension_list(+Dimension, +DimensionValues, +Ndimensions,
  %                          +DimensionPos, +From, +PopIn, -PopOut).
  % Shifts the population a each Level (value of Dimension) to the next one,
  % moving the previous values (or zeroes, when 1st) to the current level.
  traverse_dimension_list(_,[],_,_,_,LastValues,PopIn,PopIn).
  traverse_dimension_list(D,[Level|T],Ndims,PosD,From,PopIn,Result) :-
    make_list(Ndims,PosD,Level,Match),
    findall(P,member([Match,P],PopIn),To),
    attrib(Match,From,PopIn,PopOut),
    traverse_dimension_list(D,T,Ndims,PosD,To,PopOut,Result).
  --------------*/
    [ [traverse_dimension_list,_,'[]',_,_,_,LastValues,PopIn,PopIn] ],
    rec([ [traverse_dimension_list,D,['.',Level,T],Ndims,PosD,From,PopIn,Result],
          [make_list,Ndims,PosD,Level,Match],
          [findall,P,[member,['.',Match,['.',P,'[]']],PopIn],To],
          [attrib,Match,From,PopIn,PopOut],
          [traverse_dimension_list,D,T,Ndims,PosD,To,PopOut,Result] ]),

/*--------------
  % attrib( +Match, +From, +PopIn, +PopOut)
  % If a sub-group of population belongs to a level (defined by Match),
  % substitute original value of PopIn for From. Otherwise keep current value.
  attrib(_,[],L,L).
  attrib(Match,[V1|T1],[[TrabMatch,PreviousV1]|T2],[[TrabMatch,NewV1]|T3]) :-
    copy_term(Match,TrabMatch),
    NewV1 is V1 - PreviousV1,
    attrib(Match,T1,T2,T3),!.
  attrib(Match,L1,[V1|T2],[V1|T3]) :-
    attrib(Match,L1,T2,T3).
  --------------*/
    [ [attrib,_,'[]',L,L] ],
    rec([ [attrib,Match,['.',V1,T1],
```

```
                    ['.',['.',TrabMatch,['.',PreviousV1,'□']],T2],
                    ['.',['.',TrabMatch,['.',NewV1,'□']],T3] ] ,
          [copy_term,Match,TrabMatch],
          [is,NewV1,[-,V1,PreviousV1]],
          [attrib,Match,T1,T2,T3],
          [!] ]) ,
    rec([ [attrib,Match,L1,['.',V1,T2],['.',V1,T3]] ,
          [attrib,Match,L1,T2,T3] ]) ,

/*--------------
  % make_list( +N, +Pos, +Elem, -L )
  % L is a list of anonymous variables except on position Pos (Elem is there).
  --------------*/
    [ [make_list,N,Pos,Elem,L],
      [is,ReflxPos,[+,[-,N,Pos],1]],
      [makeL,N,ReflxPos,Elem,L] ] ,

    [ [makeL,0,_,_,'□'] ],
    rec([ [makeL,N,N,E,['.',E,T]],
          [is,Next,[-,N,1]],
          [makeL,Next,N,E,T] ]) ,
    rec([ [makeL,N,Pos,E,['.',_,T]],
          [is,Next,[-,N,1]],
          [makeL,Next,Pos,E,T] ]) ]).

technique(subtract_list_values,
  [ □,
    [ [subtract_list_values,'□','□','□'] ] ,
    rec([ [subtract_list_values, ['.',['.',H,['.',N1,'□']],T1],
                                 ['.',['.',H,['.',N2,'□']],T2],
                                 ['.',['.',H,['.',R,'□']],T3] ] ,
          [is,R,['-',N1,N2]],
          [subtract_list_values,T1,T2,T3] ]) ]).

technique(add_list_values,
  [ □,
    [ [add_list_values,'□','□','□'] ] ,
    rec([ [add_list_values, ['.',['.',H,['.',N1,'□']],T1],
                                 ['.',['.',H,['.',N2,'□']],T2],
                                 ['.',['.',H,['.',R,'□']],T3] ] ,
          [is,R,['+',N1,N2]],
          [add_list_values,T1,T2,T3] ]) ]).


%
%  schemata(-S)      S is the list of known schemata.
%
schemata(S) :-
  bagof(X,schema(X),S).
schema(S) :-
  technique(S,[_,[[Y|_]|_]|_]),
  atom(Y).



%-------------------------------------------------------------------------------
%     CODE CONSTRUCTION's RECORD
%-------------------------------------------------------------------------------
/*
  ccr( Name , TechniqueSequence , OperationSequence , InputCode , OutputCode )

  Name              = predicate name
  TechniqueSequence = Sequence of techniques whose application is needed to
                      obtain the predicate.
  OperationSequence = Sequence of technique-editing operations to accomplish
                      TechniqueSequence. It might refer to an OperationSequence
                      of a SUBSET of TechniqueSequence.
  InputCode         = Code over which TechniqueSequence will be applied on.
  OutputCode        = Code resulted from application of TechniqueSequence.
*/

ccr(Name,[start],
    [ unif_pred_name(start,Name,Initial) ],
    [], Initial ).

ccr(Name,[start,open_result_file,close_result_file],
    [ ccr(Name,[start],_OpSeq,[],Input),
      add_technique_ontop(open_result_file,Input,InterCode),
      add_technique(close_result_file,InterCode,Output) ],
    [], Output ).
bind(model,open_result_file,[]).
bind(model,close_result_file,[]).


ccr(Name,[time_recursion_back,get_value_back] ,
    [ unif_pred_name(time_recursion_back,Name,Initial),
      include_argument(_Var,no,Initial,NewCode),
      add_technique(get_value_back,NewCode,Output) ] ,
    [], Output).
bind(population,get_value_back,[2,1,3]).

ccr(Name,[time_recursion_back,get_value_back,update_result_file],
```

```
    [ ccr(Name,[time_recursion_back,get_value_back],_OpSeq,[],Input),
      add_technique(update_result_file,Input,Output) ] ,
    [], Output ).
bind(population,update_result_file,[2,1]).
```

Appendix C

TeMS Evaluation – Texts

C.1 Introductory Text

Introduction

Modelling in complex domains, such as ecology, is usually a non-structured task for which expertise acquisition is a critical issue. That process is costly and particularly obstructive to novices with little practice in modelling and/or programming. **TeMS** – **Te**chniques-based **M**odel **S**ynthesiser, is a tool designed to support model construction in a restricted part of the ecological domain, by automatically synthesising models (which are implemented as simulation programs) from users' specifications.

TeMS uses logic programming as the paradigm of model construction. It was devised as the means to investigate the use of *Prolog Programming Techniques Editing*, an established general method for addressing program design, in a singular domain[1].

Although we know that TeMS can build a range of models, other issues such as its usability and adequacy of its structured, process-based modelling approach, are topics for user evaluation.

How the system works

As a prerequisite for building a model, we assume that the builder has the knowledge about the ecological system which originates the model. However, to have knowledge about the system often is not enough to start building a model. One can have a good idea of model's main parameters and/or the final shape it should have, but not know how to put all together in a neat piece of code in some programming language.

The lack of a standard path to construct a model is a major difficulty for a novice. If from one hand this feature is accepted as inherent of modelling, on the other hand

[1] Research has dealt with different tasks within the programming domain

the absence of references on "how to do it" (usually acquired only from experience) makes expertise acquisition even more difficult. The modelling approach we use, leads the user through a structured sequence of design decisions and use them to guide an automatic synthesis of the model specified.

What is to be gained?

We expect TeMS would contribute to:

- clearer, more structured, standardised models;
- a structured, hierarchical approach to modelling;
- less time building models (specially for novice modellers);

What do we want to find out?

From this evaluation we want to answer the following questions, from your point of view:

- Has the system reached its goals?
- What are the main advantages/disadvantages of the system?
- Is the modelling approach used in the system any good?
- Do the inference mechanisms seems reliable enough?
- Is there any evidence that the system might be of value as a learning environment?
- How can the system be extended?

Constructing a standard model

In order to give you an idea of how to build a model using TeMS, we propose the following scenario:

Warrawong Sanctuary was the first of the "Earth Sanctuaries", a conservation project carried out in Australia by Dr. John Wamsley since 1969. The goal of the project is to reintroduce plants and animals existing in Australia two hundred years ago (before European settling). All of the land was made free of some predators and it is surrounded by a fox and cat proof fence.

Probably the most remarkable achievement of Warrawong Sanctuary is the breeding of platypus (Ornithorhynchus Anatinusacropus), which had never been done before.

We want to have an idea of how platypus population would cope with an accidental invasion of foxes into the reserve.

The current platypus population is 500 from which 95 are young (up to 12 months), 275 are adult (up to 24 months) and 130 are old (more than 24 months). Consider that the maximum individual reproductive rate of platypus is of 0.25 and that they give birth every four months and live no longer than 3 years. If a pregnant fox managed to stay within Warrawong reserve, two foxes would prey on platypus at an individual rate as shown in Figure C.1. Assume an individual mortality rate of 0.012 for platypus due to a combination of factors (*e.g.* diseases). The question we want to answer is:

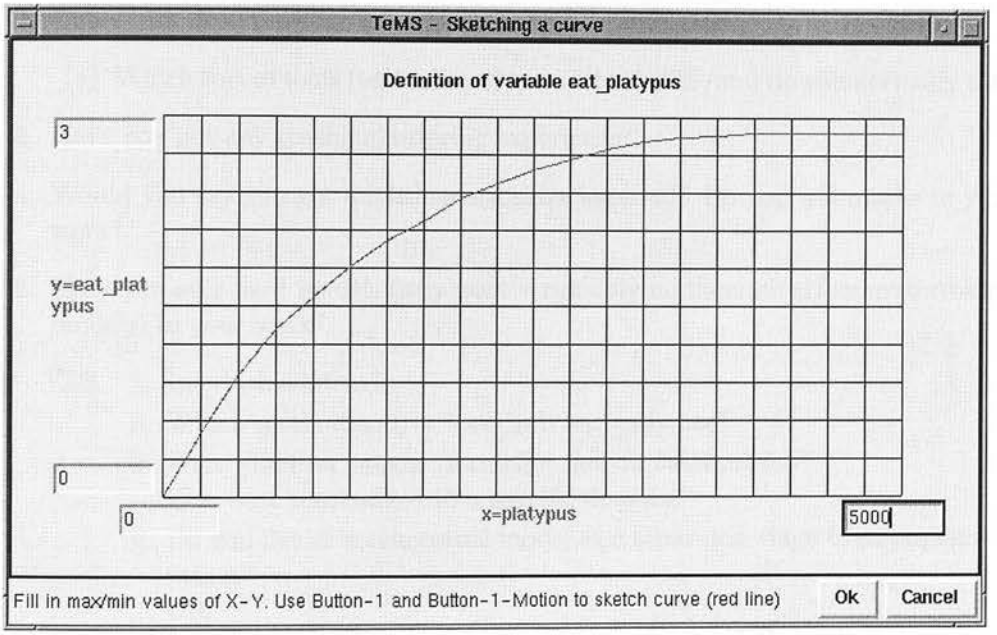

Figure C.1: eating rate of one fox

maintained these conditions, how would platypus population be in 25 years from now?

C.2 Semi-structured Interview

Introduction

Are there any questions/suggestions on the introductory text?

User's background

1. Could you describe your experience (if any) in ecology?
2. Could you describe your experience (if any) in computing?
   (a) Which sort of tools (text editors/spreadsheets/ES/etc) do you normally use?
3. Have you got any teaching/tutoring experience?
4. Would you say you got a mathematical background? Do you use maths in your work?
5. Have you ever used models (**any sort** – not only mathematical/computer-based models) in your work?

   YES
   i. In which context?
   ii. Which method and/or tools you normally use?
   iii. Were you ever taught (formally) *how to build models*?
   Was that constrained to a specific domain?
   iv. Do you devise a conceptual model as a separated stage from implementation?
   OR
   Have you got a sequence of steps you usually follow when devising a model?
   v. Do you think there is any substantial difference between modelling in ecology and in other domains?

   NO Do you think you could use it someday (maybe with data from your work)?

The modelling approach

1. What do you think about *the way* models are defined and built in TeMS?
   Was the approach clear/confuse to follow, easy/difficult to grasp, etc.
2. Do you think the TeMS' approach is similar to something you or other modellers do? If so, in which context?
3. Do you think TeMS may help a user to devise a conceptual model or he/she needs to devise it completely *before* using the software?

4. What do you think about the questions asked by TeMS?
   Are they asked in a sensible sequence?
   Can you think of other questions it might/should be asked?

5. Do you think models are useful/valid scientific tools?

About the model defined

1. Do you think the model produced actually corresponded to your specifications? Why?
2. Did you have any expectation on the population behaviour for the model your defined?
   Do you think *it is possible* to establish any expectation? How?
3. If you have produced more than one model (by modifying parameters and then generating another model), have you observed any difference in the (new) model? How about the population response for those models?

About TeMS

1. What would you say, are the main advantages and drawbacks on this software?
2. What sort of users you think would benefit from this sort of tool? How?
3. How would you place the system amongst other modelling tools that you know of (similarities/differences, advantages/disadvantages, etc)?
4. Do you think the product achieved its stated goals?
5. Do you think the user should have greater access to the system's reasoning mechanisms (knowledge base, inference strategies, etc)?
6. What sort of assistance would you like to have from the system?
7. Are there any other process categories and/or process definitions that you would like to see available?
8. What other features would you like to have in the system?

Conclusion

Is there anything else you would like to add?

C.3 Usability Questionnaire

For each of the following questions, fill in 1-5.

1. OVERALL REACTIONS TO THE SOFTWARE

terrible	1	2	3	4	5	**wonderful**
difficult	1	2	3	4	5	**easy**
frustrating	1	2	3	4	5	**satisfying**
inadequate power	1	2	3	4	5	**adequate power**
dull	1	2	3	4	5	**stimulating**
rigid	1	2	3	4	5	**flexible**

2. TERMINOLOGY AND INFORMATION

   Computer terminology is related to the task you are doing

   never 1 2 3 4 5 **always**

   Messages on screen which prompt you for input

   confusing 1 2 3 4 5 **clear**

   Computer keeps you informed about what it is doing

   never 1 2 3 4 5 **always**

3. LEARNING

   Learning to operate the software

   difficult 1 2 3 4 5 **easy**

   Exploring new features by trial and error

   difficult 1 2 3 4 5 **easy**

   Models can be defined in a straight-forward manner

   never 1 2 3 4 5 **always**

4. SOFTWARE CAPABILITIES

   Software reliability

   unreliable 1 2 3 4 5 **reliable**

   Experienced and inexperienced users' needs are taken into consideration

   never 1 2 3 4 5 **always**

5. STATED GOALS

   Modelling approach

   difficult 1 2 3 4 5 **simple**

   Models generated

   unstructured 1 2 3 4 5 **structured**

   confusing 1 2 3 4 5 **clear**

   Time to build a model

   long 1 2 3 4 5 **short**

Appendix D

TeMS Evaluation – Results

D.1 Comparative table

	IERM1 [26/1/98]
Background - degrees, experience in modelling, programming, teaching and current activities	electronics degree and Diploma; computer programmer (OO prog.); worked for Nitendo; developing modules for ModMed and AME; some teaching experience.
Idiosyncrasies of modelling:	he compares modelling with programming: "I found out to be just another language for the same thing"
How people do modelling - approaches, opinions, etc.	devise/implement; "It saves a lot of time if I work through all the ideas first and then go and do it afterwards. I definitely find that is the best way to do it, and that's regardless what I'm doing."
How people learn to do modelling	formal on programming, informal in ecology. "I always spend a lot of time on the design, before I sit down and program. I was taught to do things that way"
Modelling in ecology X modelling in other domains	No, not at the level of the environment/approach used: "... most modelling tools I've seen could be use in ecology or economics, for example."
Main difficulty on modelling	
TeMS - Approach to modelling:	
Advantages - good things about it	it's quick to learn, it's quick to use
Disadvantages - bad things about it	it's not very flexible just now; lacks graphical interaction;
Differential factor	it could be useful as a knowledge acquisition tool;
Similarity to real-life approach	yes
Inference mechanisms - should they be made available for users? comments on reliability	show it graphically
Viability as a learning environment	"... it could enable people to explore model construction..."
TeMS - Product:	
Current problems	it needs a way of following changes in the model
How far from usable product	need way to show structure; be aware of Windows-based users.
Improvements	more graphical interface; improve ability to explore around by allowing new "set-ups" be defined by more experience modellers (teaches, supervisors, etc.); debugger-like way of showing model's structure
Operation - logistics, demo/trial	problems on selection of processes; system crashed during demo (due to call to undefined predicate)
Expectations when defining a model	
Summary/Conclusion:	
Lessons to be learned - from subjects' statements	there's a lot of people who don't want to become computer experts, ... they always want to be (like) novices.
NOTES 1	When I mentioned to extend the tool by showing the correspondence between the code and the model's parameters, he suggested to implement it as in a debugger - where the error produced is linked to the piece of the code which originated it.
NOTES 2	In contrast to his ideas on modelling, he said he had to learn quite a lot of ecological concepts in order to be able to produce his stuff, but that because ModMed "is quite a specialist tool for a specialist audience".
Quotation 1	about TeMS: "... I think it could be useful for removing some of the mystique and encourage people to get stuck in."
Quotation 2	about TeMS: "... it's a good thing those constraints put on me there. The constraints put upon me made easy for me to parameterise and build the model"
Quotation 3	"I see levels of users, you know, not just novice users but users who don't intend to become anything other than novice users, people who will always just use models. And then there are people who design models, and so on."
Quotation 4	
Quotation 5	

ITE1 [3/2/98]	IERM2 [30/1/98]
degree in zoology; insect ecologist; work with population dynamics of insect pests; teaching experience; lead the modellers team at ITE (although not doing modelling himself lately)	ecology degree, has been developing on an individual-based modelling environment to deal with forestry.
	devise/try/revise
no formal training; learned by doing it in FORTRAN: "I wrote a couple of models in FORTRAN so it was sort of jump in… deep into it and do it"	formal training (Bob's module)
yes; "I think there is an enormous amount of uncertainty in ecology…"	problems may be similar, but the way people approach them - the problem solving methods, are different. They tend to thing in other ways (e.g. spreadsheets)
	"I'm not aware of any modelling tools that are used today that actually guide people through the model construction…"
clear (except for alternatives for natality and mortality)	"I thought the idea is very good and it's a very interesting useful way to go".
too little description of natality and mortality options	it could have problems with more complex models; one can't revise the state of the model
it allows people to try different conceptual frameworks - it seems to be a good exploratory tool.	guidance throughout
yes; concepts are similar	yes
only through graphical representation	yes, but a novice user would struggle with Prolog syntax, so some reinterpretation would be needed. Also some explanation when abnormal behaviour happens (e.g. constant population in a pred-prey model).
yes, it helps conceptualisation	yes, with improvements - as a training tool; suggested it should always explain novices the consequences of their actions (choices, etc.).
like graphical definition of variable (response function)	
not sure how it would cope with more complex examples (e.g.. with age classes of different sizes, or with different equations).	interface slightly unclear; no way of seeing the structure of the model; terminology could be better (e.g. transition instead of progress-on-categories)
it should record all parameters/graphs which went into the model	it needs more guidance
it could allow some variability (stochastic); switchable on-line help; it should provide for slow connections	differentiate users' expertise levels; improve terminology
choose to divide model's population in age as well; expected Return always be equivalent to mouse clicking; asked about age classes with different sizes as well as new equations	problems with mouse buttons; tried to define death by old age as part of a mortality equation
	unexpected but: "… it is consistent with the model specification…"
people with different background (physicists, mathematicians…) give important contribution because they ask the most basic things (considered common-sense by ecologists), causing reflection on what are the *more important concepts* in ecological modelling.	It's difficult to interact with other models (or sub-models) which don't use the same paradigm (language and references). From his experience: spreadsheet-based economic models interacting with eco-models.
He suggested also to vary the amount of information on screen according with the expertise of the user, although he considered that expertise would be selected by the users themselves.	His work might be related to AMPHION and Bob's paper: "I use something that generates C code, so I write both the C code, modules of C code and I write something which specifies which bits are to be used… "
	"… But I'm also using, sort of slight symbolic representation to it, a lot of people change the options in the model to put independent modules that sort of [different ?] algorithms [for different ?] options."
about domain: "… I'd thought that people in conservation management and also people in pest management would find it extremely useful and these are the two most important areas of applied population dynamics."	about his current work: "… the main thing I have to do is to get the thing running and to get people using it. So I tend to spend a lot of time in the interface and not so much on the symbolic representation aspects."
	about Stella/ModelMaker: "… you can't really do that very well in Stella or ModelMaker, they don't handle spatial interaction very well."
	about use by novice X experts: "when you adapt towards novice users then it becomes less powerful for advanced users…"

IERM3 [2/2/98]	IERM4 [11/2/98]
ecology degree; currently doing PhD in modelling (QR), used Stella during undergraduate years; programming in Prolog and C++; works involves qualitative reasoning in ecological modelling	ecologist; works with vegetation dynamics, mainly statistical modelling; no maths background.; programming in Pascal for data/statistical analysis; lecturing
He sees modelling much the same he sees mathematics: "they're both tools, but existing in their own right as subjects, essentially tools to be applied to various subjects." devise/implement/revise; he said modelling is an intuitive thing to do - diagramming, listing, but no formal way of constructing a model formal training (Bob's module) he thinks modelling is a "meta" subject - it independs of the domain; reckons the differences are due to the level of complexity correct implementation: "it's the implementation I think, just the technicalities of programming."	"I think a lot of ecologists see drawing a graph and put along to a set of data as modelling..." he believes a high proportion of ecologists do descriptive work: get data, try to explain what they got by devising experiments (models) and, ideally, testing those hypothesis
model construction is very clear, following logical steps; he likes the step-by-step approach (similar to real-life approach) it's a guided decision-making process lack of feedback on the state of the model he didn't know of a similar tool yes - it reflects the intuitive way the modellers do things "Yes, I think using some kind of mapping and staying away from using Prolog code initially. Perhaps having Prolog code plus an easier to understanding more intuitive representation, so you could see what that means." yes, with more feedback; it is better than "white-board" approach used in Stella	"it makes you understand what the basic concepts are"; he would like more leading it's well structured: "Modelling approach I think it's very simple. It's... the model itself is quite well structured, clear groups and clear components..." lack of graphics and feedback (maybe both); terminology is confusing "yes, I think it could be used by anybody." but also: "If this is going to be used by people who don't have experience in modelling, I'm quite sure that for what you are actually doing, you need quite a bit of help to find your way around."
The user can get confused with more than two populations (due to lack of feedback - see disadvantages) more feedback (visualisation) on the current state of the model; improve terminology; optional extension of existing explanation, specially on mortality and natality process definitions slow connection for demo, reasonable speed on trial (Central Lab); problem when defining new attribute and clicking OK (bug was corrected) no mental picture (expectation) of the results, but a field ecologist would always have some	it needs the ability to edit things; more guidance and feedback through definition; it lacks graphical representation of the model graphical presentation of the model structure (more bits of the model, instead of one at the time); more guidance on selecting options; ability to put several runs in one graph typing problems; wanted more instructions of what to do from 1st menu; odd failure happened: system froze (no explanation); he decided to qualify answers to the questionnaire rather than do an interview.
About modelling tools (he'd use): "I guess intuitive tools, diagramming, picking things up, listing things, basically my brain, I don't have a structured formal way in which I'm constructing a model." He suggested more elaborated explanation of how things are connected and what multipliers do - what they cause on the model. Also suggested connection to some graphical environment (e.g. AME) to create a modelling tutor. About difficulties on modelling: "I think the modelling isn't the problem, I've got modelling in my head, great, but how do I get this onto a computer and how do I get the numbers..."	As one could expect, simulation modelling is not a consensus among ecologists. Also, it seems that he endorses there is a schism between field ecologists and modellers (see quotations 1 and 2). See Note 2 about: "I think I was a bit confused about messages on the screen ... it wasn't always obvious what it was actually doing what the time, there was some screens that I didn't quite understand" It seems I didn't present to the subject a good overview of how the system conducts the definition process. Problems with terminology (mortality rate and time structures) and time constraints (see quotation 3) On eco-mod.: "I suspect that a high proportion of the ecological work is really descriptive, sort of exploratory work, and it only becomes more formal at the last day, when they design experiments to those particular components which they have observed." "Simulation modelling is something totally different from traditional modelling that ecologists have done, ... mathematical modelling ... fitting statistical modelling." "I was a bit confused about messages on the screen, because it wasn't [obvious ?] and I didn't expend much time read them, it went though quite fast..."

ITE2 [28/1/98]	ITE3 [28/1/98]
degree in ecology; some teaching experience; use several modelling tools and programming languages (Maestro, Stella, FORTRAN and Prolog).	biology 3rd degree, genetics PhD; computing experience during PhD; some teaching experience; quite mathematical background; currently in modelling work.
he normally uses someone else's models and rewrite it adapting it to his needs	"The modelling is all FORTRAN programs and all are written from scratch, we don't use any modelling tool, but we do have ModelMaker, but we haven't used it in many big models..."
"It can be interactive yes. It totally depends whether you doing top-down or bottom-up modelling...". He has devised his own heuristics for the design process	uses FORTRAN programming; mostly process-based modelling; devise/try/revise; all from scratch (as opposed to use a modelling tool)
formal training (Bob's module)	no formal training
not sure, he wouldn't though ecological modelling is different from modelling in other domains	no opinion, but she's trying to use OO ideas on her work
	There was (when she started) no systematic way of learn how to do it
it requires an amount of previous knowledge: relationships, time structures, etc.; he mentioned not having enough time to know the tool properly	very clear and structured
fairly well structured, good way of having some model up and running	step-by-step approach: "...it was good. I haven't come across anything like... sort of takes you though step-by-step like that, which will be good."
inflexible in some way	less graphical then other environments (e.g.: ModelMaker)
	clearer then ModelMaker;
yes, it's similar to what modellers (suggested himself) do in real life	
it might not be a benefit to see the inference mechanisms	not clear, it needs better explanation, she said "I would be curious to look at the actual code that it came up with..."
yes, for teaching how to create simple models specially in population dynamics; users could understand conceptualisation better.	yes, with improvements; it could be useful for anyone doing modelling
he got stuck among some menus/options	not sure about more complex problems (models); it lacks editing facilities (a summary at the end of each stage could be used to that)
ability to change parameters and setting between runs; more explanation on key terms like time structures.	editing facilities; graphical feedback; some way of change parameters and to compare runnings
slow connection so sometimes he wasn't sure if a command have been accepted; some problems with mouse/key settings; not sure about questions on the questionnaire	very slow connection; found bug when tried to use capital letters for component names; problems with mouse buttons; questions on the questionnaire weren't too clear; problem with OK button (see B.McIntosh)
	Yes, some.
	modelling is a very difficult task
it may exist a "meta" level on peoples approach to modelling: "I suppose I have my own approach, probably it's the same of you and many other people's approach."	many (most?) modellers use FORTRAN and, on the majority of cases, they rewrite from other people's programs (see idiosyncrasies of modelling)
"...this seems very structured and it leads you thorough ...to generate the model as you'd say. Whereas something like Stella is like a white board where you draw things and make connections between them, so in that way you need to make your own mind about...	About learning how to do modelling: "... you couldn't go to a library and get a book that teaches you by really describing it and have you started writing models..."
	About the step-by-step approach: "... it got them (users) started, which is good."

DAI1 [25/2/98]	ITE4 [29/1/98]
civil engineering BSc; programming (including Prolog); just finished PhD on logic-based approaches for ecological modelling; experience with teaching	ecology degree, work involves forestry and land use; some teaching experience; some programming in FORTRAN; currently at coordinatory tasks
he builds very simple model to get an initial grasp of the framework; uses his own heuristics, refining representation and operations over it; he might mix conceptualisation with implementation issues	adapting other people models' rather than start from scratch (see quotation 1)
not in ecology, but in physical systems	formal training in modelling
yes; other areas (engineering) have well known physical laws to explain either many components or many relations; ecology has both and much less theor. basis	reckons there are similarities between eco-modelling and modelling in engineering, for example.
abstraction/conceptualisation; to know what the real system is, what has been modelled	
it's easy learning to use it;	
it can be used straightforwardly	
different screens popping-up may be confusing; structure diagram is static;	it should have more prompting/help for each question/decision point presented
hierarchical specification - more readable, it has more information than "white-board" approach	"I've never seen a front end for Prolog programming like this before"
yes	
yes	
It needs improvements like, for example: making the current structure diagram dynamic (a clickable source of information); OK for teaching novices (see quotation 3 for qualification on that)	yes, it could help conceptualisation (at least in one sort of modelling); it could be used by several areas (agriculture, forestry and many branches of biology) which use Leslie-matrix models.
after improvements, usable by novices and experts	
structure diagram is not enough; time structures were not clear (not time-line shown); couldn't follow parameters values on time	terminology, it needs more help
different priorities for different class of users; natural language interface for explaining terminology and/or inference mechanisms;	categorisation in specific areas using examples (see notes 1); on-line help; connection with compartment flow tools
He was surprised that the model actually corresponded to what was specified	slow connection; he would like more info on how to operate TeMS
No	
Ecology is not only complex in terms of the number of interactions within systems, but it is also dynamic, so it's very difficult to segregate interactions, to establish a threshold for interactions.	1) Adapting from other people's models is the current practice. 2) "I think is a very important area to develop decision support models"
the disaggregation diagram is important: "... when that picture of the classes that you show in that tree appeared, I could then understand what I was doing"	He mentioned the need for modelling a system under different viewpoints, connecting them all (e.g. ecological and economical models for land management): "... so management support models within ecology and that bring you into business."
	suggested the use of a data-base of examples from different areas to be used as reference when helping the user - a sort of "categorisation" which he reckons would help the system to give help to the users.
About ecological X general modelling: "...civil engineering, the physical laws that rules everything are well known, so the model will be just to satisfy the constraints that such physical laws impose to model...in a more abstract domain like business, "	"in my personal case ... it's more a matter of taking another model that exists and looking at what the other does and try to implement that in a slightly different way, rather than starting from the very beginning and try to write a whole new model."
"... I think the domain does not have many things involved but more many relations ... the ecological model is difficult because there are many things in the domain and many relations..."	"... I think that by doing this you're showing for students who are put off by modelling, that is quite easy for themselves to build their own models, so I think it's a very commendable effort."
"I think this sort of tool could be useful for teaching, and even for experts. Of course in the case of experts, they would need more... maybe more efficient execution."	On help on Stella/ModelMaker: "if someone is coming through "cold", a non-expert user, the on-screen help that you got on those tools is not very great and you would expect to have to read the manual for one day or two to be able to build your model"

Agric1 [13/2/98]	SNH1 [12/2/98]
degree in ecology, PhD in environmental physics, post-doc; lecturer; currently doing modelling himself, co-ordinating modelling projects and lecturing	ecology degree; PhD on impact of rabbits in vegetation; programming in Basic and FORTRAN; has built a large model for real-life based vegetation behaviour; currently is an advisor on land and grazing management; some teaching experience
mentioned scepticism on effectiveness of traditional modelling, but also: "I think modelling is the fundamental scientific activity, and we all do it, we don't know we do it"	learning programming was needed to break the barrier between field ecologist and modeller
goes with devising the furthest possible before implementing; uses example models with students; uses simple programming or spreadsheets: "my ideal way of working is with someone who is an expert in programming and can do that job well"	devise/implement
no formal training. He teaches students: "I work with students, a lot of the time. I have used various models with them"	formal training (Bob's module)
thinks there is a big difference from ecological modelling and engineering modelling, for example (engineering mod. can be much more tied into the laws of physics)	she wouldn't think so: "... I would think most things have very similar modelling problems..."
conceptualisation (see quotation 2)	to adapt your ideas about the system and the model intended to a certain framework (e.g. mathematics)
process-based approach is important (see quotation 1), it would help to understand causal relationships on the model	clear; terminology needs to get used to; the questions asked are essential questions (about essential parameters)
it's clear; it can directly break in age classes; you can test various scenarios	it allows a novice to end up with a model
it lacks documentation/support	it needs more on-line help; more complex models would require too many questions be asked/checked
"it clearly outbreaks on age classes, and some of the other things I've seen, well, the traditional systems dynamics sort of model finds rather difficult to handle age classes"	she doesn't think is too different from ModelMaker
yes, very much similar to what modellers do.	yes
	yes; if the tool will allow rules to be included/removed, it might give warning of changes in the causal/influence network that the user would not be aware of
yes, with improvements: more support; to have a library of implemented models with classical examples of population dynamics	yes, with improvements it could be really good for improve conceptualisation
it could be good to groups of 2 or 3 work on in	
it lacks on-line help	it needs more help
	she can't see how it could be extended (as it is) to something more complex (see disadvantages)
switchable, pop-up help; implement/record standard population dynamics models	On-line help to aid understanding of the structure of the model and the terminology used
KB-house lab used for demo. He mentioned: "I don't think you should expect someone to be able to sit down like this and in ten minutes be able to do everything that it is capable of"	Demo carried out on DAI machine.
"I'm not surprised" - "You can generally look of something and see if something is obviously wrong. But there are other cases..."	it depends on how complex the model is; she would have expectations at least for the general behaviour
1) Conceptualisation is the main part of modelling and also the most difficult part of it. 2) TeMS' process-based approach is important for novices' understanding of the causal relationships within the model.	1) There is a "mental block" for biologists when something involves maths (or a more formal framework). 2) TeMS needs more help and it could be used to warn the user of unexpected effects - between a model's parameters, that is.
maybe his preference to delegate programming is due to a lack of easy-to-use programming tools.	She thinks a step-by-step approach can also be developed in ModelMaker (macros ?).
he inquired on possibility to translate or transpose code for other languages.	
"... get people to think in terms of processes is an important part of this ... I don't think it is necessarily a normal way of thinking."	...my feelings are that as an ecologist you don't need to be taught how to model, because you're building models anyway in your brain, it's just formalising those models
"the difficult thing about modelling is actually the whole concept, the idea of dealing with things, with objects as systems with complex properties ... it's a difficult intellectual activity."	"I'm not sure it is that different really from something like Stella or ModelMaker ... if someone ask you questions is probably easier to come to use novices because they just go through answering questions as you ask them an they will end up with a model."
Contrasting TeMS with AME: "The trouble is that on AME you got to think of a huge variety of possibilities so you got an enormous system."	continuing from quotation 2: "So I guess it's better."
Advantages of TeMS approach: "you can test various scenarios and generate the set of results that can be looked exactly the same as you have looked upon them actually going out into the field and make real measurements." cont.below "You must never confuse the two, but from the point of view of certainly initial work of teaching students, it's very powerful indeed."	

Agric2 [27/1/98]	IERM5 [9/2/98]
degree in electronics; worked on the industry, PhD in cognitive science; currently building a modelling environment; has been exposed to ecology and ecologists	worked with modelling during PhD (late 60's), building FORTRAN models for plant physiology; currently still do some modelling; lectures a great deal
he reckons visualisation is important for novices get a grasp on modelling.	he believes people have the main "templates" for model ecological systems
devise/implement/revise;	devise/implement/revise; most modellers do their own programs: "Some people use ModelMaker, and some people use Stella... Most people write their programs in FORTRAN, Basic, Pascal, C..."
formal in programming informal in ecology. Described as: "determining how to do modelling rather than - sort of - try to learn from other people how to do it."	no formal training available
he thinks ecological modelling is different because it deals with natural (as opposed to man-made) systems and they are also more complex	doesn't see much difference between eco-mod. and other areas' mod., but mentioned all ecological models are a bit different (granularity of process X details)
	parameterise a model
it start people thinking about of what's important in modelling	clear, easy to follow; too simple for practical applications but good for learning about modelling; not sure about spatial representation of processes
easy to follow	it gets a beginner started
it is hard to go back and trying different things; terminology (specially variable names) is confusing	it could be limiting if modeller wants to try it with more complex problems or apply it in other domains
	"I like the idea that the way it asks you for information, so you have to respond and then it puts it together to make a progress..."
	similar to what modellers do
"I don't think the user would want to see the information at the level of Prolog program... maybe some kind of graphical representation..."	yes
yes, with improvements like more support on concepts used (age-classes, for example)	yes (see approach to modelling) "it would be interesting trying out in a class of students"
He found strange to see a non-graphic-based mod. tool	he liked the ability to draw a functional response
he wasn't happy for not be able to define processes the same way he does in AME (e.g. compensating death by old age directly on the equation); it lacks editing facilities	
graphical interface for the model in a way the users could work themselves backwards to recognise what the inference mechanism is; examples of answers for design decisions; ability to see all parameters through time	graphical support; worked examples for a tutorial going from simple to complex models; ability to record all parameter values thorough time and produce graphs of them
problems to run TeMS from Central Lab; Jasper is high motivated by his own modelling environment/approach - he tried to implement scenario model on AME (partial success); tried to include all mortality in one equation	
	too busy during demo to reflect on what results should be, but in retrospective, results make sense
modelling in ecology is more complex than modelling man-made systems. Visualisation is quite important	modellers who like to do programming may find a model generator too constraining, at least for complex problems
"...ecological modelling is more complex because the system you're trying to model is not something that you, basically, have designed yourself or know about and therefore the model needs a lot of constructs that are unique to modelling natural systems."	About parameterisation: "schism between those modellers who represent a few processes and have a small number of parameters and the others modellers who like to have rich detail but unfortunately they don't have a way to find the parameters values"
About showing inference mechanisms: "if they can start with a very simple model and they can sort of... work backwards themselves to recognise what the inference mechanism is."	"I think it does transfer across domains, but maybe the way you set it up its at the moment very constrained"
	"people love to write their own programs. And most people discovered that programming - as long as you have a time ... can be very satisfying"

IERM6 [4/2/98]	IERM7 [6/2/98]
BSc in ecology; worked in a system dynamics expert system; some teaching experience; programming in Pascal and SAS	1st degree in natural science; PhD in forestry; currently working on measure and modelling water usage by vegetation; some teaching experience; very familiar with computers (OS, programming languages, packages - not modelling packages)
	reckons similar problems (ecological modelling) when modelling activities involving human beings
devise/implement/revise	devise/implement/revise
formal training (Bob's module)	no formal training, he says it picked up from books, papers, conferences and working on it.
eco-modelling could be more complicated because of the number of variables	sees differences - the inherent complexity caused by heavy and chaotic interaction and how to measure/define parameters (e.g. reproductive habits of wales)
abstraction: "I suppose the more difficult thing is what to include and what not include in the model."	conforming to a theoretical framework; lack of non-deterministic support for eco-modelling; scale issue
clear, at least for population dynamics models; liked the step-by-step approach	step-by-step is good, but it relies on semantics of the definitions (variable names, etc.) given; it seems to be a reasonable approach
it leads you through the modelling process	it's easy to learn; it should be fast to work with it and try to implement a model; it should be less susceptible to errors made by users
if you want to make it more flexible, the knowledge base will become huge	inflexibility (specially for more experienced modellers)
"most modelling tools I have seen were really based in systems dynamics, in drawing the diagrams - while this is more structured, it takes you through steps which I think is good."	
yes	similar to what experts do
would like to know why results came up that way (reasoning system)	"yes, it would be good for the user to understand that making that selection causes the behaviour to change and in what way it changes."
yes; it's important novices not start from programming, but from actual modelling	yes (see quotation 1)
not very flexible	could be confusing for more than two populations; lacks graphical support; terminology
	some way of seeing the model structure with larger models (e.g. 10 species) and notice interactions between (e.g. clickable boxes with model's parameters); ability to follow other model's parameters through time
Yes, but expectations aren't always right. Modelling is useful to find out what went wrong when that's happens (see quotation 2)	yes, and it's important to understand why a model wouldn't behave as expected
	1) "most of the [ecological] systems are fabulously complex ... so all the time you're simplifying reality". 2) Graphical resources in tools can be the attractive factor over modelling by direct programming (e.g. FORTRAN)
	He's skeptic on supporting tools for conceptualisation: "models come from experience, they come from knowledge of a particular part of the world, so I don't think you would ever use a tool to generate conceptual model, that's the process in reverse"
	New equations would need help: "I don't know what operators Prolog uses ... any precedence rules ... So, without some extra information I wouldn't be able to write any equation"
Mod.approach: "if novices have to start from programming, and that's a huge problem, they would learn programming instead of the modelling, if they're using systems dynamics diagramming tools, they will have to remember which steps to go through"	"I could imagine it might be really nice for use in teaching for instance, you try to get people who haven't done any modelling, maybe undergraduates or other people, to understand how to do modelling..."
"... modelling is really useful because you found it that something didn't go quite as you expected so you have to go through the model to find out what it is."	"... and also to begin to understand interactions between animal populations. I think it would be very good for that (learning), because I think people could learn very quickly and be happy about modify it and see how things work... trial-and-error sort of...
	"I know that you can in many cases achieve the same thing much more quickly in a much more flexible way by using a modelling tool and that's what interest me..."
	"you can't possible model every interaction that you are already aware of in the system, because it would make the model too complicated and there are probably interactions going on that are immeasurable or you don't know about."

D.2 Transcripts

Participant's dialogue is indicated by a capital letter followed by a colon (:). Interviewer dialogue is indicated by (M:). Square braces ([]) mark pieces of the dialogue in which the sound quality on the original recording was not good enough for transcription.

Participant 1 (IERM1)

M: Let me show what I want to do now... explains interview/demo/trial

M: Could you describe your background, your experience with computers, with ecology, both of them or maybe just one of them.

A: Yes, sure. I'm a programmer, not an ecologist - that's something you got to be aware of. I originally trained in electronics then I retrained in computing and [?] a C- based programmer, so OO and procedural programming as opposed to logic programming, OK. I came in to ecology because they needed a programmer here and so I found myself doing modelling work for ModMed project which Colin Legg and Bob Muetzelfeldt were involved with. The idea there was to have a generic environment were we could have specialist modelling modules plugged-in to it. This was all written from scratch, it was all written [?] for Windows environment. So [?] I came from a computer-science side as opposed to for ecology side, my ecology background is nil and my ecology has been, since I started the job here and I've here for almost two years now. So, that's me. I've been exposed recently to things like AME and obviously that's living a way for programming from the [ground up] as a [?] tool to help you program. And that's is my experience in sort of generic programming environment, and I've been writing modules for AME just now.

M: Have you got any teaching experience, tutoring or teaching?

A: Just beginning to have right now. At the moment I've been lecturing HTML [?] that's really the first proper teaching I've done. The students are supposed to be submitting their work through the web, and of course they having problems that I would never... I never though they [?] very easy thing for [?] to do, but I found later it's quite difficult. Say today, my main thing today is student sending me work and me writing back to them saying "don't send it to me, send it to the web". So that's er... my teaching experience just beginning.

M: So you were trained in electronics, your first background?

A: My first was electronics, yes.

M: So you got a quite hard...

A: I got an honours degree in electronics and I did a post-graduate Diploma in optical electronics which is [?] really.

M: So you have a good mathematical basis?

A: Yes, I think so... it's [hard ?] to say. And then I've been trained as a post-graduate diploma in Computer Science. So er... it gives you a good sort of engineering [?] electronics and then it tells you don't want to do that [?].

M: Before the work you do here, have you have any experience with modelling, I mean, modelling in a different context.

A: I would say yes, but some would say no. I've worked for Nitendo [?] in games, depending the way you look at it, that's a modelling exercise.

M: Yes, of course, I meant modelling in any sense, like in programming drawing with paper and flows, block diagrams or whatever.

A: There's a couple of programs [?] the game side of things it's quite interesting because you do use a lot of modelling concepts, although they are... I only realise that because I lived in both worlds, you know, and

before that we used a thing called ROM, which is an object modelling environment from Napier University. (M: What is it called again?) ROME, have you seen it? (M: I've heard about if I'm not wrong, yes) Yes, John Savage and Ken (Kenneth) Barclay and it's er... a programming tool, as opposed to a purely modelling tool. But it uses some of the user-interface tricks that I've seen used in modelling things so you can drawn boxes, have flow diagrams and so on. And that then produces code for you, what is quite nice, but I think... that's probably it really, of course the modelling I've done here is quite different because doesn't use modelling tools as a [?] last two years have been taken [?] and just from scratch.

M: You are in the field of building modelling tools yourself.

A: Well yes. A little, yes. I mean [?] what I am, much more than... I mean, I'm not at all an ecologist as I've said. More or less [?] to build modelling tools is something I'm involved in now. In fact ModMed itself was always intended to be er... a very specialised tool, so we got a specific [planned ?] model in mind and [we built ?] ModMed around that, and then the primary control of the ecologist [we done ?] over that will be pure parameterisation really, as opposed to fundamental changes in model structure. Pure parameterisation. So in some ways, ModMed is very old-fashioned project, in another ways it's quite as far as I see it, quite updated, because we use an OO structure for ModMed. But, having said that, it's a customer-build model for a specific purpose and it's now beginning to look as if more and more modelling has been done in a more generic way, with user-interface tools to enable you to change, as I said the underline structure of the model. So er, that's [? good size — both sides], I think [?]

M: Just one or two questions still about modelling: Have you had any formal training on how to build models, even in computing or electronics.

A: Not really, I would say no. I am not [?] understand modelling to be [here ?] not at all.

M: How is it to grasp the basics of it? Is it hard, more or less?

A: Well, I think building a model is pretty much like building any computing system, once you understand the... [?] systems. I find quite easy to come from the idea of... a sort of abstract idea of encapsulation and so on, over to modelling you got these processes in flows and influences and all this stuff. To me I find out to be just another language for the same thing.

M: Do you think... when you are programming, or working with modelling, with think separated from the conceptual model and the implementation or you think a bit of one and get a grasp of the implementation. Do you know what I mean.

A: Like design first and then implement it?

M: Yes, I mean, like a two-stage work. (A: Yes) Do you always do that?

A: I always design first, I always spend a lot of time on the design, before I sit down and program. I was taught to do things that way. But I also learned from experience that for me at least that is the best way to work. It save a lot of time if I work through all the ideas first and then go and do it afterwards. I definitely find that is the best way to do it. And that's regardless what I'm doing, whether I'm writing a text parser or, you know, a plane model or what I've done most recently, a tri-dimensional thing for AME, [?] is a simple... it's a piece of TclTk... do you know it? (M: Yes, I'm using it) OK, this is TclTk. I designed this over Christmas, just the code to do all this stuff, it took about 5 days and then I wrote in... well, I wrote the first version in an afternoon. I much prefer to work in that way, to spend a lot of time in design and then jump in the implementation once I'm certain the design is... well, you can't be certain, once youre happy the design is going to work, that's is the way I normally do it.

M: And you normally do the same whatever you doing? Whatever task you doing?

A: Yes. That's the way I've came to work. I mean, if I writing a document I still doing that way. I still plan the document and then go and do it. [?] any computer task I'll plan it first and then do it.

M: How about when the task you have to perform is a new task? Let's say when you started to deal with ecological modelling, you had to start to get some grasp from ecology and things like that. Which way you think could be better for a novice modeller, to grasp (A: It depends...) the ecology basics.

A: It depends how deep you have to get into, I think. If you gonna be implementing something... OK, I see two specific things: When I'm producing my stuff I think of grades of users, sort of advanced users,

probably the guy who wrote the program is the advanced user, right down to people who just want to run models which have been pre-built. And then you got various grades in between them. You got people who want to just run [?] models to see what happens, you have people who build the models, people who specify and design the models and the people who implement that specification designed. There is all these different things, and I think that more advanced knowledge, I don't mean advanced in depth, I mean knowledge in advance of the task. It's requires [?] you get. The sort of... [?] can the user go and press a button to see what happens, might learn best for actually doing that, just coming in and see "Oh, what happening here?" [?] in my opinion, what is obviously just my opinion, that user won't need too much knowledge in advance. He would came in and be shown things on his on, and that's great, that could be a task of a model which can be used for teaching. But then when it comes to design the model, then he'd require much more background information before he start up, and when it comes to implement a design for a model, then I think you need again proportionally more information and these could be different [kinds ?] of information. I mean, a physiology [?] modeller would have no knowledge of how to put that into a computer whatsoever. But also he would needs some concepts of modelling. A programmer implementing a physiological model, may ideally require more knowledge of ecology, because the model should be specified and he just would have to implement it. But I felt in my experience [?] that you'll find that very difficult.

M: Do you think there is any substantial difference between ecological modelling and other sort of modelling, modelling in general that is.

A: No, I suppose... most modelling tools I've seen could be use in ecology or economics, for example, but the people who design those models, and let's say designers [who stay ?] from implement, people designing those models must have specialist ecological knowledge, or specialist economic knowledge. The programmer on the other hand, ideally... you know, the guy behind the tool, ideally would not require any specialist knowledge of who is using the tool, but because of the nature of what I have done, which is quite a specialist tool for a specialist audience, I did actually have to learn quite a lot of ecological concepts in order to be able to produce my stuff. But [things like ?] AME and Stella the sort of thing, I don't thing those programmers required to know anything at all about the specifics of the [?] implementation, you know. I don't know, it's difficult to think of ecological modelling as a specific thing, you could... it's general in the sense that a lot of the concepts used are general, but... I don't, it's a very difficult question, very good question.

M: I've been wondering about that for some time now. If someone with an ecological background would be thinking in a different way, would he see that tool, that media to build a model in a different way of a person from economics or whatever would see it, you know? (A: I wonder) If they have this mapping, maybe ecology it's a very particular (peculiar) thing, because there are different degrees of complexity and inter-relation between everything.

A: There is something I was wondering about as well, I was thinking that would be quite interesting to present AME to economists and see what they said. Because obviously it's primary intention is to enable people to produce ecological models, but was just wondering over the weekend about whether economics people would be interested in it. And I've somehow very interested to see that, I wonder... an awful lot of ecology there is qualitative good/bad, more/less, hot/cold (M: and causal reasoning) Yes, economics it seems to me you could [?] be much more quantitative about. They both equally complex subjects I think, I don't anything of economics, so I couldn't say, but it would be very interesting to have an opportunity to contrast how some the different [foundations ?]

M: You could have some hint, because systems dynamics stated with economics, that means something.

A: Yes, there is a lot of crossover there, which it would be interesting to explore would not? really interesting.

—— Demo and trial ——————————————————

A: Comments on questionnaire: Roughly what I've said there was: it's quick to learn, it's quick to use, it's not very flexible just now, but that's... you knew that - you stated that already, and I'm not quite sure what I've had at the end. You possibly [?] into this, but I'd like to see more graphical interaction. I'd like to see... even if it was just a static representation (M: Actually, there is a very basic one) OK, what I mean is with little boxes and... you know, the standard, some sort of (M: And of course, there is no help) That's true. I know that it's difficult to choose a correct sort of graphical terminology, but I do think that boxes and lines ideas it's quite useful, or even if you just have... I don't know, I just feel something is missing, something graphical is missing. It's all menu-based. [?] that's what you have produced, but how do you

do that? If you could [only ?] have. Maybe a summary. You have a... even just... Is it really possible for example, for me to say: "OK, the population of haggis depends upon the population of wolves in some way... or [nessie ?] in some ways..." Is that possible to use, you know, if wolves predate upon geese or something, how would I state that? I mean, can I do that? Let's say, there is a predation relationship here.

M: Actually, when you state one of them is carnivorous or omnivorous and the other is herbivore, it is... a sort of induction (A: There is an implicit predation there?) Yes, but of course, you can say that by specifying a process of predation, you can say "there is predation here", you can put a new process and put an equation. But is not user-friendly.

A: Yes, but there is a problem there, because you can have a herbivorous dinosaur and a carnivorous mouse, you know, the mouse would never eat any dinosaurs. So there is... obviously is [?] development what I was thinking there was if you stated explicitly the predator-prey relationship, then you can [draw ?] some graphics of the link which says "you defining this, click on this box to see the information you've given for this", even if that was a static thing, so you couldn't go into that box and change things, you could at least get a representation of it. So to see you go through this step-by-step and you'd say "OK, now I'll put this information in it, and now I'll put this..." you know, you'd put the information and at the end before I run it, or even after I run, to check on my things, I want to be able to just get summary information. So let's look on what I've put for mouse, "thump" ah, there is the error all right, now I'm going and change that, you see? (M: that's a good suggestion) Maybe you have, let us say a predator-prey relationship, you would draw a line there or something, you wouldn't draw a line to make a relationship, but you may represent it in the summary information by drawing a line, just to say "here you go". It's so that the user can get... just like an aid to the user conception of what he got, I would say... you should do that. As I said, even if it was static, so you couldn't use the graphical interface in order to change things, because obviously that implies a lot of programming work which you don't want to be into, it would help the user understand what he got and even help him with the debugging of the model as well, so that would be a nice idea. Anyway, is there more formal stuff you want to ask me? (M: Yes probably, but not much, I'll try to be quick).

M: Probably you have answered already this on the questionnaire, but what you think about the way the models are described? Are they easy to follow, do you think they're given some structure to the conceptual modelling, or to organise their users thinking?

A: Yes, I think... as I said before, I see levels of users, you know, not just novice users but users who don't intend to become anything other than novice users, people who will always just use models. And then there are people who design models, and so on. There is different levels of complexity on your users. Now, what I just was able to do there was pretty much a novice user task, I was able to... well I did some design, but pretty much was parameterisation, I made some choices and then parameterise them. Now I think if you catered it for another level of user who could design a whole new process, a whole new input process, you could say "well, OK, that was a population dynamics type process which you designed." How about if you could have a more advanced user, someone with perhaps not your... not much knowledge of programming as you, but somebody who wasn't a novice was able to specify a new type of set-up. Because it's a good thing those constraints put on me there. The constraints put upon me made easy for me to parameterise and build the model, but it is only one type of model, so if there was some facility to describe a new type of model, which could then be given to the novice user, do you see? I think that would be excellent.

M: And probably to make the novice user infer how to do different models, making it more general.

A: Yes, because you can either have a modelling tool for a general purpose and it could do anything you like with it. What is all very well for people who know modelling.

M: And also know about that specific language for building models.

A: Yes, but if you want something to be... somebody with minimal training to be able to put their ideas in. If their ideas fit all in one framework then that's a good thing to me. Not everybody would agree with me but I think that's a good thing. People just don't think user within certain limitations could be a good thing. But you must be able to... you know, the teacher if you like, or the supervisor or whatever must be able to say "well, actually this doesn't suit my purposes, I want be able to describe a new way of input things or a new type of models, and then give it to my students or whatever, people whom are trying to get the knowledge from and have them go through it". That's what I see it's missing here. But again, you already said that this is an earlier product and it's constrained to only type just now. So I can see there is room for it. But I actually liked the constraints, I think it's a good thing. Although I can see there is need

for more flexibility in the future.

M: Do you think this way of lead the user to specify a certain sort of modelling, is similar to any other environment you have seen? Do you think it could be useful?

A: Yes, I think it could be useful for extracting opinions, maybe. It could be useful in many ways, but something which immediately struck me was that I was someone with experience of population dynamics in the field for example, then I could with minimal instructions, sit down with something like that and put my knowledge into it. It could be a good knowledge acquisition tool, that's what I'm saying - that's quite good. And also it could enable people to explore ideas... no, not ideas... to explore model construction, because it is constrained, there are only a certain number of things which can change, it might encourage them to get a [?] starting from a white piece of paper is one thing, this... sort of hold your hand along the way, it says "OK, now you may want to do this, now you may want to do that... these are reasonable choices..." I think it could be useful for removing some of the mystique and encourage people to get stuck in. Obviously the more experienced the modeller you speak to, they would say "I don't like, because it doesn't let me do whatever I want", that's what experienced modellers like to do, it's like when I use Visual Basic I don't like, because it doesn't let do certain things that I want to do. Doesn't give me complete freedom, but that's hardly the point. Visual Basic enables all sorts of people do [?] programs. So put constraints on people it's often frustrating for people who know the details, but those for don't know the details or don't want to know the details that's very useful. That's where this falls in. It's like a useful modelling tool for people who don't want to know the details.

M: Yes, that's the idea, and you've said one thing I've never tough of before, you said there are some user who want to be novice, they don't want to do things differently, they just want to build models like this one...

A: Absolutely, there's a lot of people who just want to come along and just use a model, for example, I'm looking on something now for pig farmers, and pig farmers haven't absolutely no interest whatsoever in become computer experts, they just want to know if they change nutrient balance of their feed will they make more money? that's all they want to know, so they always want to be novice users, they'll never want to know anything else. And also you have to keep in mind that perhaps it may be useful to have a thing ultimately being more flexible, for many people its gonna be very useful to put their constraints on it.

M: Do you thing it could be useful to give some insight or to show somehow the reasoning mechanisms on the tool, like the KB or part of the KB. You said about to see things a bit more.

A: I would do it graphically, you know, that's my training, that's my background, to make everything nice and simple. I think we can... I have one those terrible lines but I think it's a good point, that you should always keep in mind whenever you doing science specially when you doing things which tend to go beyond the scientific world, you should always keep in mind the user, even if that is just a conceptual user, you know. When I'm doing a contract for the European commission for example, I still want a nice user-interface with menus and all this stuff, because although hard science may get more information from huge files full of numbers, even the most hard of the scientists, will benefit from a graph or a 3-dimensional chart, or trees or whatever. So I always think that way, let's give an impression, even if to get the ultimate truth you have to look at the details, let's try to give an impression on the screen of what's happening. So I would like to see some sort of graphical representation of what I've done. Both: before I run it, in order to understand what I'm expecting, and after I run it to understand what's happening. Although I'm not qualified to say how you should represent your reasoning process, that's is not a question for me.

M: One of the things I've been thinking about to make as future work, which I really want to pursue, is to give the user, for those users who want it, some relation between the techniques, better saying, the processes and the instantiations the user have made with the final code, saying "OK, you have selected this sort of process, using this sort of parameters and this is somehow represented here is the program..." for some user or some novice programmer who want to know how to build a program to represent something, how to migrate from the conceptual stuff, the definition, to the code itself. Do you think it would be a useful thing?

A: Yes, why not. That would be an optional thing...

M: of course, I was think as a teacher, about students who are learning Prolog and who could learn how to use Prolog to produce simulation models.

A: I wonder how would you do that... I think there's different ways to do it. I don't how the structure of your code is, whether you could click on an equation and say something like "show code" or whether you could click on "moose" and say "show moose code"...

M: Like that question you asked "where this information is represented", but something deeper than that. Like "OK, I've selected an equation, and ask where this equation is dealt with in the program?" What part of the code says, this is an equation dealt with by the program.

A: I don't know, maybe the simplest or... again, as a programmer... I think that for me, as a programmer, try to understand what your tool has produced, so now I'll put a different eye on it - a technical eye, I think it would be useful for me to be able to like... almost like a debugger, you know, when you got a graphical programming environment, and you compile it and gives you an error list, you double-click on error and it jumps to the piece of code that created that error. Maybe like that, except not errors (M: but the final code) yes, so maybe here you got a list of modelled functions, or whatever - relationships, and you go click-click and it goes "jump - here is the bit in the code". It might be easier to go to all the code, not just one piece which deals with that, I don't know, that's something I'd need to think about. (M: Just what I was think over this weekend) yes, "I wonder if I could do that...", anyway, the best way to do it could be just to jump straight to the part which deals with that, without try to interpret it further, just to say "there! start to look here" that's where you'll find help on that, maybe that would be the easy enough to do it.

M: Thank you very much, I wonder if you have anything to add, to mention?

A: Well, something... You do want to be aware of the fact that, of course, this could be used in Windows as well. You've used... you written it in Prolog and TclTk, both of which are available in Windows interrupted by a student So, please be aware of this could work in Windows as well, also most people, if you talking about (M: I have to say I didn't think about it) 90 percent of the people of the world are gonna have more access to Windows computers than there are Unix computers, which is sad but true, I much prefer Unix but most people have a Windows, and the fact is that TclTk and SICStus is a nice bonus for you, works very well in Windows, so that's good, so you can expand your user base, have in mind you have to cut things like middle-button.

Participant 2 (ITE1)

M: Could you tell me something about your background, your experience?

A: I'm an insect ecologist, I work on the population dynamics of insect pests, in agricultural crops, forest crops and in woodlands. I've worked in projects on insect like [?] and [?] in Scotland, mahogany [shoot bug ?] in Latin America, and some things like [?] . Also I work on diversity of insects, again my work spans [?] to tropical zones. I always work in forests and forest [?] plantations in Scotland, Cameroon and in Indonesia.

M: Then I assume your first degree was in biology?

A: In Zoology

M: Would you say you have a strong mathematical background?

A: Not strong, but a moderate mathematical background, but no mathematical training since I was in school.

M: Do you use maths in your work, to do models and to understand what other people do?

A: Only a little.

M: I see you are a user of some computer packages and programs, what sort of tools you use at the moment?

A: I use standard tools, spreadsheets, word-processing tools and statistical tools like SAS, but [?]

M: Do you any sort of modelling tool in your work, I mean, like ModelMaker or Stella, anything like that?

A: No.

M: You do make/build your own models?

A: I haven't constructed any models for about 8 to 10 years.

M: But you have contact with people who work, people who produce some models.

A: Yes, obviously here I have contact with people who build models and [?] previous experience other then modelling.

M: The sort of population dynamics you mentioned, does it involve the relationships between insects and forest.

A: Yes. The sort of things I've done was to look at the relationship between insects and [?] plants, insects in a natural [?] and find tools to get [?]

M: For what you have seen and from your contact with other people, about modelling - I'm talking about modelling because that's what this tool is about, do you think in your own personal experience there are any basic differences between ecological modelling and modelling in other areas, like business, industrial modelling or whatever?

A: Yes, I guess so. I don't have the experience of modelling [?] in order to be able to make an intelligence statement

M: Just your impression, do you think there are essential differences things in ecology? A: Yes, I do. I think there is an enormous amount of uncertainty in ecological modelling, [?] uncertainty is smaller in other modelling systems. That's a huge [?] to take in account.

M: Have you got any experience in teaching or tutoring?

A: Yes, I've actually been doing a little bit of teaching in Edinburgh University over the last two terms, because they start changing the university. Before that, in my previous job which was over 15 years ago I did a lot of teaching.

he takes a phone call

M: You were saying about your experience with teaching at the University. Are you still teaching nowadays?

A: Yes, a little bit.

M: Were you ever trained in modelling? I mean, in a formal module, discipline or anything like that?

A: No, the first job I had 20 years ago was a modelling job, and had no training and in fact it didn't [?] I was very keen in understanding FORTRAN [?] I wrote a couple of models in FORTRAN so it was sort of jump in deep in and do it. I taught myself a few programming languages and I did some modelling about 15 years ago and then a little bit more last [?] not much.

M: Do you think models are, or could be seen as valid scientific tools? Are they useful?

A: Yes, yes.

—— Demo and trial ————————————————

M: Do you think the way models are defined in this tool, is clear, confusing, maybe difficult to grasp the sequence? What do you think about the way models are defined on it.

A: I think is reasonably clear. The structure is clear. What's confusing is the difference alternatives for natality and mortality.

M: Do you think the system should give some more assistance to the user?

A: Yes. You've given to yourself quite a small space which to describe a particular option under natality and mortality. If that was bigger, it would give some clearer way of define each category and to [?] explanatory text on them, or perhaps the key differences underlined.

M: And maybe examples to compare it to?

A: Yes, good idea.

M: Do you think the software's approach is similar to the way other people doing modelling?

A: Yes. I think the concepts are similar, but then what you appears to be able to do is to take a range of different conceptual frameworks and explore it using the package you have developed. What I see often the problem with different conceptual frameworks is that the average person isn't able to explore them, you have to take what people tell you on trust, you can't go and play about with that simple model to explore the consequences of the aspects of the framework you're using [?] play around with. Do you understand what I trying to say? Is just that the key thing, it seems to be it's to be a good exploratory tool.

M: Do you think it could somehow help a novice modeller to learn how to devise at least one sort of conceptual model? I mean, does it give some hint to a novice modeller on how to go through design decisions?

A: Yes, it does. The good thing about the example that you using there is that is a relative simple example. I wonder if I would have though that a novice modeller could use that example if I could [?] simple understanding of zoology, but if was a more complicated example

M: That's because I'm a very novice modeller. I'm asking these questions because the way I've put the design decisions (organisation, disaggregation) that sequence, it's because I've tried to use to put my experience in modelling into ecological modelling. And so I'm trying to check if that makes sense, because a person from a different background (other than a proper ecological background) could think in a different way towards modelling, do you what I mean. The way things are in there, they are a mapping of my views of modelling as a programmer, as engineer.

A: It seems OK to me. I sympathise with it because you got almost all the important things there. You started from the bottom, which is the way all models should start, and the conceptual framework should start. You got the options to disaggregate which is very good. The different components, [?] and the relationships between them and also the flexibility which is the ability to graph of which is excellent. I think that if there is one thing that worries me a little bit would be the idea of uncertainties, [stochastic ?] whether [?] unexpected events, it would be nice put some kind of variability into the system. OK, you can run it as it is, but it would be nice to do a step further and have defined for example a certain level of natality and the consequences of that, the next step might be to say "natality varies random-like within these bounds" what does that do? And you could do the same to mortality feature. And then explore what that does to the model. It would be a kind of sensitivity analysis.

M: That's a good idea for further work on the tool. Which sort of users do you think would benefit from this?

A: Well, the example is good, because that's an example of conservation problem, and I'd thought that people in conservation management and also people in pest management would find it extremely useful and these are the two most important areas of applied population dynamics.

M: Do you think the user would benefit from seeing the inference mechanism on the system, the KB - even through some graphical representation?

A: Yes, perhaps only through graphical presentation (M: of course, Prolog code would be just) Yes. For example, one thing that would concern me if I was using this model and did [?] just done and got an interesting result here, I would like to be able to have a record of everything that went into the model. And [?] the values for natality and mortality [?] but what you might forget it's the particular curve - you see, you need to have a representation of [?] there.

M: I think that's about. You have mentioned some assistance you'd like to see have from the tool. Any other features you think it could improve the accessibility of the tool?

A: No. There's one nice feature that I've noticed using SAS. Do you use SAS? (M: No) It's quite a [?] statistical package and like so many computer packages that came into the age of Windows, it's not doing very well, it doesn't look very good. But they try to do things, one of those things is giving the user help, and you have an option to switch the help on or off. I would suggest you increase the amount of help, the complexity are quite obviously for an experienced this is annoying I don't like to told that. So like the 3rd version or the 4th version, maybe you should add that facility to it. If you aim to increase the amount

of information on the screen, then have the option to decrease it for experienced users (M: I see, switch on and off) yes, switching on and off. And the other thing is I wonder if is because this [?] slow, I don't know whether I was waiting for something or whether I haven't pressed the button (M: That's because of the connection, normally is quite fast, that's because I didn't bother to give any warning "the system is processing" or anything like that). Because I can imagine you or another user could find yourselves in a circumstance like that every now and again. If are in Brazil and need to try off somebody else which doesn't have a Unix machine [?] it would happen with that.

M: Have you got anything to add? Maybe some comment, suggestion?

A: Not really, I rather enjoyed it.

Participant 3 (IERM2)

M: I'll starting first asking you about your background, your experience, what was your first interest.

A: Basically I did an undergraduate degree in ecology and I did some ecological modelling under Robert Muetzelfeldt's, after that finished, I started of as a RA at the University of Edinburgh involving... I was working in a tropical forest modelling project, we used individual-based approaches to model dynamics of one [rack of standards ?] over a period of 50 to 100 years. And I have gone some way towards making the approach flexible and modular and use [?] of symbolic representation of the model structure, but isn't very fully developed because the main thing I have to do is to get the thing running and to get people using it. So, spent it ... tend to spend a lot of time in the interface and not so much on the symbolic representation aspects.

M: I know as it is with this stuff. So, you have got experience with modelling and with computers as well, I presume so (A: Yes). Would you say you have a mathematical background? (A: No) Do you use maths in you work?

A: I use maths but I can't do calculus, really. I use numerical techniques and not analytical techniques so, I don't know any calculus or any very advanced calculus. I use very simple calculus, not advanced calculus, for integration.

M: I wonder, were you ever taught how to model, how to do modelling? (A: Yes) in a formal course, I mean?

A: Yes, I was talking about Robert Muetzelfeldt.

M: I see. which sort of tools you normally use nowadays, when you do modelling?

A: Nowadays I use... well, I probably use something quit... slightly similar to what you've done, I use something that generates C code, so I write both the C code, modules of C code and I write something which specifies which bits are to be used. But the main language I'm using at the moment is C. But I'm also using, sort of slight symbolic representation to it, a lot of people change the options in the model to put independent modules that sort of [different ?] algorithms [for different ?] options.

M: But er... the software tools you are using, you have built them yourself? (A: Yes) OK, you don't have the need to use any other software package like ModelMaker or Stella, or...

A: Well, unfortunately most of the models I make, individual-based models are too complex, Stella and ModelMaker are not very good at spatial interactions, and where I do my modelling there's a lot of spatial interacting and overlapping between damaged shapes caused by tree falling over on the grids and to [?] there is an overlapping and those sort of things is not... you can't really do that very well in Stella or ModelMaker, they don't handle spatial interaction very well.

M: The way you work with modelling, do you always devise a conceptual model, a complete conceptual model beforehand, before to start to implement it, or you try to mix a bit, you devise the main, the core part of the model, then implement it and then go back?

A: I think the second one you described is much more attractive, [?] many [decides ?] some things only become apparent after you implement the first part, so you change the model.

M: Have you ever tried to... or had to do modelling in another domain, like business or industrial processing, or anything like that? or programming?

A: I've done some linking with... as part of the project I was employed on... I had a set of economic models which interacted with them, but most of that models were spreadsheet-based and it would integrate with what I had done, but I thought it was too ambitious [?] to do it at the time, because of the different nature of the models, the complexity [?] because they used a lot of the time spreadsheets and they really organised their models according to spreadsheet ways of doing things... which sometimes wasn't... I mean, for iteration it always a quite an artificial in spreadsheet, if you wanted to iterate a group of figures then you had repeatedly put them into spreadsheets and didn't seem ideal to me. So, interrupted by a person using the other computer in the room so I [basically pull ?] back because I thought work to get us both using the same language and the same reference would be too difficult.

M: Would you say that modelling in ecology have similarities with modelling in other domains, in general?

A: I know of a lot of economic models use compartment flow approaches, but most of the economists I've worked with seems to [?] spreadsheets, all the models in spreadsheets, no matter what they doing they have to iterate for whatever, says simulation modeller [?] use spreadsheets, so... what's the question? (M: Is it similar) yes, they are similar, but sometimes they organise their ideas according to the tools they have, they tend to think a lot in spreadsheet terms.

M: So you think the problems may be similar, but the way people approach them, the problem solving are different, essentially different, because they think in other way.

A: Sure, yes.

— Demo and trial ————————————————————

M: So, can you tell me what was your overall impression of the system?

A: I thought the idea is very good and it's a very interesting [essentially ?] useful way to go. I thought the interface sometimes was slightly unclear on what you should be doing next, or the sequence you have to go through was slight unclear sometimes. And also because I was using it on a PC based system there is a problem with mice. I think, my experience with user interaction is: you need to get people something they can see [rely on ?] to know to click mice buttons, sometimes, mice buttons can be shortcuts but usually is [bar on ?] to something on the screen that people can get [at ?] if they don't know about what mice is a shortcut to. That was the main... oh, I had something on the sequence of operations was slightly unclear, also some of the terminology, I talked about progression in progress_on_categories, to a modeller, the term that they would use probably would be transition between size classes or age classes, so some of the terminology, the modelling terminology which is not probably your domain, so you just trying other people could be improved on... there are words that modellers would relate more quickly than some of the words you've used... I think... [I've already said, but it is ?] potentially very useful way to go, I'm convinced of that.

M: What do you think about the way (the approach), the way models are defined and built on this tool? Do you think it's clear, it's confusing to follow, it's easy, I mean, to get a grasp of the modelling sequences...?

A: I think, in a way, what struck me is not the graphical - you don't have a graphical representation of the structure of the model, but it would [be ?] good to have some... even if it was text-based... of viewing the structure of the model (M: on, actually there is something) yes, that sort of thing would be useful so the user could reviewing the state of the model [where is at?] it would be ideal editing and amend it to make it [better ?]and ...

M: Do you think this approach is similar to anything that you or other modellers probably do?

A: Yes, it's slightly similar to the system I have developed, but it is much... I have... In fact is not similar because the idea here is to guide people through the process of constructing model - I haven't done that. So, no, I'm not aware of any modelling tools that are used today that actually guide people through the model construction. to some extent. I don't know of any.

M: Do you think somehow it could be use as a supportive tool to help novice modellers to get a grasp of conceptual modelling, because sometimes they don't have any references of enough examples and experience to start building a model, they can't do even simple models?

A: Yes I think... yes, it seems to be the way the approach could be adapted for that, but a novice modeller would probably need to know slightly more... also the problem when you adapt towards novice users then it becomes less powerful for advanced users, because with novices it's going to be lots and lots of text, every time they do something to get an idea of the implications of their actions, and that's just annoying if...for more advanced users. So maybe you need some [idea of ?] differentiate the environment according to the level of expertise of the user.

M: Do you think the model actually corresponded to your specifications? Did you have any specification - I mean, expectations? (A: the results?) Yes, the results?

A: Well, I believe [it ?] I haven't though about the structure of the...

M: But if you do something like those specifications, do you have any sort of idea what could... what should come up from this?

A: I actually didn't think about. But... there wouldn't be any cycle, there shouldn't any population cycling because the predator [?] cycling was due to mortality after 3 years, I wouldn't expect platypus numbers to rise like that, [?] a million is quite... (laughs) it's a very large rate of growth for a contained area (M: maybe the foxes aren't doing their job) the problem is not be back on any density dependency. You don't do that in the model, so it is consistent with the model specification, but that could happen.

M: Which sort of users you think could benefit from this sort of tool?

A: At the moment, I think that there gonna be problems specifying more complicated models, so maybe the way to aim it is at novice users, and put lots and lots of text, and every time they do their action it informs them on the consequences of their actions. I can see it being very useful that way. Probably if you go for more advanced users, they probably have a very detailed specification and idea of what they want, the problem I usually find on modelling tools is that they're good at general models, but they not very good at specific models, like distribution throughout a canopy is quite a difficult thing to model, or individual spatial interaction quite often can be difficult to capture. So I think this probably should be aimed towards novice users as a training tool.

M: Do you think it could be useful the user having greater access to the inference mechanism, to the knowledge base or so?

A: Yes, I could see it. I mean, once you have the [larger ?] representation of the model structure, I think it would help a lot but the novice user [?] struggle with the Prolog syntax, so you need to do some reinterpretation, some syntactic [sugar ?] to make it... to produce the knowledge in natural language for example, because novice users wouldn't be acquainted with Prolog syntax.

M: You have mention some already, but could think of any other assistance the system could give the user as it is now?

A: Yes, er... the ones I've mentioned: you'll need to be... a clear sequence of events and not relying on mice buttons and more text [probably ?] or to have some notion of the interaction every time the user does something, they get some message of the implications of what they have done, some analysis that would say [it' s a ?] sensible thing to do [or it could say it's not ?] not a sensible thing to do. Er, you... for example, you could mention on way in which to [have ?] some other outside knowledge, like something is a carnivore, something is a herbivore, which [then you'll ?] probably be able to infer that herbivore can't eat or influence mortality of a predator. At the moment you can probably capture that, but there are some cases where... there are less clear cut situations, I can't think of one, where normally something is not appropriate but in some cases it might be, for example, normally you wouldn't expect population to be constant, so as soon as your user specifies that, it could be perhaps some text warning that isn't the normal [way of working ?] you'll need feedback or some reaction, because is quite an unusual choice, but it's not a wrong choice, but it is unusual.

M: I see. Have you seen any other... I'd better say: From the processes you have seen there specified, the instantiations for natality, for mortality, etc. Can you think of any other which could be added? Just an idea?

A: I'm not totally familiar with what you've gone - got in there at the moment, but I imagine that once you get the idea of implementing other peoples models you already user a vast number of different functions,

many of them having similar shapes but, there is probably a very large number of functions you could use. I couldn't say off hand which one is [?] important to put in at this stage.

M: I think that's it. That was very good. I wonder, have you anything to add, maybe a comment, or suggestion?

A: No no, I think it [worked ?] very worthwhile, a [?] thing to do. I think it has a lot... it could have a lot of potential... (M: thank you, it was very useful) It was very interesting to see.

Participant 4 (IERM3)

M: Could you tell me a bit about your background, your experience?

B: My original background is an ecology degree which involved certain amount of modelling by using systems dynamics, [that was ?] using a computer program called Stella [some ?] modelling environment click pointing, draw and stretch putting things together, graphical modelling environment and... when I was been taught undergrad level modelling was all largely the examples [and work through them was ?] quite clearly structured, so you didn't really need to put at a tremendous amount of thought to it. There wasn't a... the whole modelling component of undergrad degree wasn't huge as quite as small part of [what was done just from my own ?] personal interests and [that ?] I was interested in modelling, so I ended up here, interested in modelling and I work in a project with Robert Muetzelfeldt. I wasn't specially interested in qualitative reasoning to begin with, that's where I ended up, and I guess my background now... I'm fairly... I'm far more confident having spent a year and 3 months doing a PhD about how [you can ?] structure a model. Still have to sit down and then go... every now and then [?] should put things in but I have a bit of real-based modelling experience. That's what I'm doing, not necessary using systems dynamics for everything.

M: You have quite a good experience with computers and package software...

B: Yes, I [?] Prolog, I'm now programming in Prolog and I also program in C++ and can read Basic and [things like that ?].

M: So... you have used in your undergraduate course, software like Stella, maybe Flomo as well.

B: Flomo I saw here. I think Bob wrote Flomo [as I understand it ?]. So I've used Flomo, Stella is like Flomo but it can do more things.

M: Would you say you have a strong, or quite good at least, mathematical background?

B: Mathematical background in terms of actual qualifications it's a bit weak, still I can...

M: But you use maths on your work...

B: I'm quite happy [?] maths. If I don't, initially know how to work it out I can look up a book and [?] no problem teaching myself. I've taught myself quite [?] statistics for an undergrad dissertation, a spatial analysis statistics. So I am confident, but not qualified.

M: What sort of modelling tools are you using at the moment? If any?

B: My brain (M: that's an important one). Well other than [?] I guess intuitive tools, diagramming, picking things up, listing things, basically my brain, I don't have a structured formal way in which I'm constructing a model.

M: OK, you don't have the same, let's say heuristics or even an algorithm, that you follow every time that you start to do a model?

B: I don't think that I've [done ?] enough models for an algorithm [?] I've got two running models from my PhD, I would reckon it would take a good couple of years more experience before I could [some ?] standardise it.

M: But have you got any paradigm you like the most, like systems dynamics?

B: I think systems dynamics seems to be good for many things, and I guess that if you have to translate systems dynamics into some qualitative format then it could encompass a lot more things, I would think the problem of ecological modelling is the data and the information, quite often rather than what [?] actually using the model. You see, there is some kind of structure, brainstorming as the [?] people [would think ?], but diagramming and that's systems dynamics [?] use to include (repeat) and from there, incrementally get more and more structure and refine the idea. There is no tool [?].

M: You told me you got some formal training in how to do modelling, I wonder, have you ever applied, or have you have thought of those things in a different context from ecology - in a different area like business or programming.

B: Absolutely, I think that modelling is a meta-subject. Modelling the techniques that I was taught, were almost definitely specific to ecology, [?] your modelling as much the way I view mathematics, they're both tools, but existing in their own right as subjects, essentially tools to be applied to various subjects.

M: But having said that, do you think there is any basic difference between ecological modelling and general modelling - modelling in banking for example?

B: I think it's a level of complexity. I don't think there is inherently differences, it's just the subject [method ?] that requires something different of the techniques, but the actual approach that is used in the way you have extracted the elements of a system you think are necessary to model and they're important [?] say to that subject area.

M: Going back to your approach of modelling, you said your brain it's your main tool - that's a good statement. Tell me, you always have a complete conceptual model before start to implementing, or sometimes you get a general idea and then try to implement, have some feedback and then go back to form your conceptual model?

B: I think the conceptualisation conceptualisation it's always fairly plastic, so things can change, but it's pretty much a firm [?] body of rules or set of diagrams or something, and although you may change bits here and there, in the way you [?] the whole model is conceptualised in some format before implement it and then once you got some feedback then you change, although that's a [?] approach at the moment, I haven't entered it for model validation, so and I don't have any data sets to compare my models against, so I haven't had to get to the stage of changing it because it doesn't match the real world.

M: Just to conclude this first part, what do you think is the most difficult part of modelling?

B: I would even though I think modelling it's quite an intuitive easy thing to do in terms of a system what are the exact are the important components here, it's the implementation I think, just the technicalities of programming. That's quite a difficult thing on a list where I'm now quite confident with my Prolog skills, having never been taught programming at undergrad and towards the end of the undergrad degree I decided I needed to know how to programming, so I taught myself C++ and here I taught myself Prolog. I think that probably is the biggest barrier. I think most ecologists humans fundamentally operate in a modelling sense, you go around many models [always ?] how to get into a car, however you doing the simplest things and the complicates also. I think the modelling isn't the problem, I've got modelling in my head, great, but how do I get this onto a computer and how do I get the numbers [that define them ?] (M: how you match that through some implementation, that's the point) Although I think that [there's ?] something like formal structure formalising the conceptual idea.

— Demo and trial ————————————————————

M: What do you think of the way models are defined in that tool. Of course it's a very straightforward way of defining things through steps, but is that clear, is it confusing? Is it too constrained? What do you think in general?

B: I think it would be good. The construction of the models is very clear, I think it follows some logical steps and the only thing I would reckon it's that once you get above the level of complexity of having two populations, two compartments then without some option in the menu bar to give you feedback on the current state of your model, then I think you'd easily get confused. Because in Stella, you see your entire model in front of you and if you add on another bit to it, it's still there so you can see as it builds then some kind of feedback process on the model building I think it could be good. With rabbits and foxes, nice, you remember of a few, but if you try to if you've ever seen any of the global models that are made where

you get [factory production ?], culture and then disaggregate down the whole world, that's actually no way you can remember all that, you need to show so you don't forget and get lost.

M: Do you think that approach - going through steps - is somehow similar to what some modellers actually do in real life?

B: Yes, I think it probably is. The first step you've chosen it's the intuitive first step that you would have made in modelling, I think choosing your main components. It's probably roughly what modellers do, I think. But that's not to say that your system doesn't provide a valuable function, because modellers do that because of the model and the first model may take ages, they'd make lots of cock-ups and say "bad" and then after that they find out that it takes far longer than it should have so experience that's why they do it. So I think it probably do.

M: The idea of the tool should also to give support to someone who's learning how to do modelling and would have no experience. In that way, do you think that system , with further assistance, it would help a student to devise a conceptual model? When you have a framework to follow, you think that could give they some hint of how to start to do it, a starting point for novices?

B: Well, if you are giving them a sheet of text I think it would, because of presenting them with possibly even better than things like Stella, using Stella you just got a blank screen (mimics the user - what's this?) whereas this you say "right, choose your components", "specify how your components are structured or disaggregate", the processes, etc. Yes, I think it's good.

M: So you think in that way, the sequence, is it a good one, is it a reasonable one? The sequence in what things are asked for the users like "choose your components, now how about organisation, are will gonna disaggregate this in some way or not?" Does that make sense?

B: Yes, it makes sense to me. The only thing I found at all confusing would be just the lack of feedback on the current state of the model.

M: About the model produced, do you think some way the model results, the final shape of the population, it corresponded to your expectations, I mean, from what you have described there, would you have expected that shape of curve? Roughly?

B: I didn't have a precise picture in my head, I knew that foxes would be [on average ?] while platypus I didn't know exactly what would happen to them and [?] particularly when two foxes [?].

M: I was thinking, maybe not related to the tool itself. But in general, when you start to build a model, do you have an expectation of what should you get from that, or you try to discover as you specify things?

B: I think An ecologist I think, will have some kind of idea because the [?] idea would be based on either going out into the field and looking at the system and reading the literature and see what actually happens in that system, perhaps speaking to native people and asking them what happened, what it is like - in a descriptive manner, so you got an idea. I think the modeller necessarily expects the model to (M: to behave in some way) to behave like the real world does otherwise rather than get the approach the task in hand not [quite lonely ?]. (M: OK) I think it possibly it would be good to I expected the foxes [?] the platypus it could be good to perhaps relate somehow just to expand on certain concepts in it. I think that having talked to students here, in modelling things like mortality rate, they get confused. A lot of confusion can arise not on the choosing of compartments if they are using systems dynamics as the paradigm, you got your state variables very easily or relatively easily you abstract it from the system, if you got a pop of predator/prey the two state variables are compartments are foxes and rabbits or the predator and the prey, but when you get multipliers and variables that influence other things, so population of, let's say fox natality may be influenced by the number of rabbits [or the amount of prey that would be on the model ?] those things I think it would tend to confuse, rather then the actually abstraction of the two important things for the system.

M: You have mentioned what could be some improvements of the tool. Can you remember, or can you think of the main drawbacks and, since I'm asking that, the main advantages you can see in that sort of software?

B: I think the main advantage I can see is a guided decision-making process, which is good. The main drawback, as I said before, is feedback on the current state of the model. I think it could be useful for the

[?] to be added to your work, to ask someone like Bob who has always taught undergraduate modelling for what I consider quite a long time. Have you interviewed Bob yet? (M: No, I won't - Bob is my second supervisor, it would be drive the results) I would ask Bob what he thinks undergrads find or novice modellers find difficult, in my experience they find things like flows easy-peasy, population and state variables easy-peasy, but when you hang around influencing things, then some things get a little confused there. So maybe elaborate on how things are connected and what multipliers do (M: Again because of the causal relationship between components) Yes, what exactly one component is doing, not necessary go on an analogue to the real world, but what its doing causes on the model.

M: Can you see any similarities between this tool or this sort of tool with others you have used or you have seen? Any sort of decision-supporting tool for modelling.

B: No that I remember of.

M: Do you think a user could have any profit from take a look at the inference mechanisms, like the KB? Even through some sort of interface or some graphical representation mapping it?

B: Yes, I think using some kind of mapping and staying away from using Prolog code initially. Perhaps having Prolog code plus an easier to understanding more intuitive representation, so you could see what that means. Yes, I think it could. Because at the moment, is a bit like [?] it works, you put in some things and then suddenly you got numbers which [?] comprehension I think could be. (M: again the problem of feedback on what the system is doing) Yes, on what the system is doing. That's where I see the value of tools like Stella. Although I think yours could be good because it was guiding the initial decision-making process, things like Stella are good on that you can see what's going on (M: what is happening) what's happening, and that is quite important. You need more tuition when you are using something like Stella and then the whole model building process can become less formalised, people aren't exactly clear how to do things and what the initial steps should be, nothing in graphical feedback.

M: And apart from that, can you think of any other sort of assistance the system could give to the user (apart from feedback)?

B: Perhaps you could perhaps just expand definitions, so you got your library of mortality models and when you got just a small bit of description, then I think perhaps expanding that sort and perhaps drag up a separate window, then I think if you do something like that, it could be good. If you give your systems to undergraduates, they could be

M: Have you anything to add? Suggestions, criticisms, comments?

B: I think I said everything, you got feedback - graphical [?], drawing attention to components of a model which aren't compartments like populations, you got variables and multipliers drawing attention to those which could be a great source of confusion. Perhaps slightly more help for fill in models [or something of that type ?]. I think it's perhaps it's reinvent the wheel for you to make a graphical modelling package but I can see the use of yours in perhaps explaining the inferences with [something like ?] AME so you had a (M: making a connection) a modelling tutor, which then you could use rather than reinventing all that graphical stuff, [?] AME's graphical interface, that's perhaps some way of showing how the model looks, something like that.

M: Thanks a lot.

Participant 5 (IERM4)

M: Can you tell me something about your background, your experience?

C: I'm an ecologist, I'm interested in vegetation and vegetation dynamics, the way vegetation changes with time and the way that is influenced by management practices, in particular things like grazing and [?] So I'm mainly interested in that. And in order to understand the way vegetation works, I think you have to understand some of the [?] plants, so I'm interested in sort of behaviour of ecological behaviour of individual plants [?]

M: You work with modelling, or with people who do modelling?

C: With people who do modelling. Yes, I'm not a modeller.

M: Would you say you have some mathematical background?

C: No. (M: not at all). Not at all.

M: But do you use maths somehow in your work?

C: Very little. Statistics, I'm interested in, so Statistics is important, but I wouldn't call that mathematics.

M: I see a computer in your office. Probably you use computers.

C: Yes

M: What sort of software are you using now?

C: For programming I use Pascal, I'm trying to use Delphi which is Pascal for Windows and standard software - word-processing

M: So you do some programming in Pascal (C: Yes) What is that for?

C: Most of what I have done I would say is for data analysis, or statistical analysis.

M: In your experience you deal normally with people who do modelling. Do you think people's approach to modelling is somehow standard, or everyone has a different idea.

C: Lots of different people have different ideas - yes. There are many different approaches to modelling, different objectives. I think a lot of ecologists see drawing a graph and put along to a set of data as modelling, but you than talk to somebody like Robert who has a totally different idea of what modelling means. I'm not suggesting that one [?] wrong, but they're certainly very different. (M: In your I'm sorry, go on) Simulation modelling is something totally different from traditional modelling that ecologists have done, which is [?] mathematical modelling [?] fitting statistical modelling.

M: Do you think people, when they do modelling they first have the complete idea of the conceptual model on their mind, or you think they try to experiment a bit, they devise they got an idea of the main the skeleton of the model and then try to implement and get some feedback and then refine it with time

C: Yes I think I guess most ecologists do the work first, get the data and then begin to wonder what they going to do with the data. And they look for a way of representing what they have observed, so I suspect that a high proportion of the ecological work is really descriptive, sort of exploratory work, and it only becomes more formal at the last day, when they [?] design experiments to those particular components which they have observed. I suspect most ecologists would do the research first and then they'll have [?] design model [?] to represent [?]. Whereas maybe that a more appropriate approach is to design the model very early on the research project and that should then structure the research which follows. I think I see a model as a [?] used in research as being like a hypothesis which should be testable, and you use a hypothesis to design an experiment to test that hypothesis, and that's the way you should use models, but I suspect you don't actually have it in that way very often.

M: So you think in that sense models are valid scientific tools?

C: Yes, it depends on the model sometimes that confuse the issue.

M: Do you have experience with teaching or tutoring students:

C: Yes, but not in modelling (M: in general?) Yes. You could argue that all science is really based on models, conceptual models, [?] (M: abstracting) Yes.

M: You said you don't use modelling yourself, but you mentioned you do some analysis with your analysis with your data, even programming to test some hypothesis and find out some results.

C: This is where the word modelling has many different meanings, and if you take to its limit, as soon as you begin to think about something you're constructing a conceptual model, so in that sense everybody uses models. In terms of constructing simulation models, that's something that I have not done. I've done very simple statistical models, in a sense I suppose any statistical exercise you do, involves the fitting of a

statistical model, [?] this is the way that your data behaves or this is the way system behaves, that is, in essence you're fitting a model, so, if you use the word modelling in its broader sense, then everything you do is modelling.

M: Would be right to say that you don't use simulation models because, probably, your work doesn't require that?

C: Yes.

M: Or is it because you prefer not use it.

C: Well, I'm using simulation models that other produce, so the project on [?] is about simulation models and I'm not coming to the way they designed [?] somebody else is really doing the work [?] me. I've endeavoured in very simple models, produced [?] model, transition model [?] as part of PhD and it was pretty simple compared with most simulation models.

M: That's it for the first part.

— Demo and trial ————————————————————

M: Could you tell me what you think about the way models are defined in this tool, I mean, not exactly the result, because the results I know, were a bit frustrating, but the way models should be defined. Do you think it is easy to follow, difficult, obscure the idea of landmarks or stages you have to go through, is it a good idea or doesn't matter?

C: I think I would have liked the program to lead through the definition more than it did, so if I had to select items from the menu, one at the time, in order to construct the model. If you have to select those items from the menu one at the time in order to construct the model, if you to select those items from the menu one at the time in the [middle of ?] they're presented to you rather than [?] go back to the menu to select an extra. [?] so many things which have to be defined in a certain [way ?] then I would thought it would be better if [various ?] pieces of information were presented to you one [?] I'm not sure how much choice there was in order to you should define things. And the ability to edit what you've put in is actually quite important, I think, specially when you're exploring something new, then you're putting things and you not quite sure what the [?] are going to be, then you have to go back and change it. So I [?] was very important.

I've down on the modelling difficult and frustrating because I think I didn't find it terrible clear what the structure was, as I was going through [?] because you shown me what to do, to some extent, but I think it would be nice to see what the structure is, see what is coming next, in a sense. So you could see which [bit ?] you go through at the time. I think ecologists tend to think graphically, think in pictures [?] see relationships, and there are some special relationships and [?] other things. I wonder if would be possible to devise a way of presenting the structure of the model graphically and tell "here's the bit that you are at the moment, you're putting in predation rate, which refers to platypus in different age groups" and then you could see another box somewhere else which is mortality rate for different age groups, so you would see the whole structure as you put bits in whereas at the moment you only see one bit at the time, you see mortality-adult-platypus, and you can't see what you just put in [?] put in for old individuals. So I think I would like to see the structural thing more clearly.

I think it might in terms of you put here dull/stimulating, it would be much more interesting to use if you could then think of individual parameters and see how that affects the result. So one thing that would be quite nice to do would be to have a graph on screen of your last one, and then do another run with different sort of parameters, you could set the values and to compare and see how the result has changed, [or better ?] if could actually put several runs in one graph it didn't seems as you could. It would be much easier to explore the way the system works if you have several runs displayed at the same time.

[?] it run fast enough.

I think I was a bit confused about messages on the screen, because it wasn't [obvious ?] and I didn't expend much time read them, it went though quite fast, but I'm sure it wasn't always obvious what it was actually doing what the time, there was some screens that I didn't quite understand, with several boxes and screen were you type in the mortality, a box at the bottom and several boxes at the top, I wasn't quite sure what they did, I was clicking to see what happened. So sometimes the screen was a bit confusing in that respect.

[?] I think it would be in a sense I didn't find obvious what to do next, maybe reading the instructions on the screen would be easier. I'm not sure that the user are sort of lead through the model, [?] is clearly as [?]

I think if the model doesn't sometimes it doesn't [were ?] presumably because of the parameters we put in there were [?] somehow. You need to be able to find some [?] other to step back through the model and find out which bits have failed, you need to be able to find to be given some information on the part of that might be to get information [?] to different age groups and mortality rates and mortality [?] and see how they change with time.

It's a yes, I think it could be used by anybody. I think it could be used by somebody who [?] models before [?] It makes you understand the basic concepts of population models, obviously [?] mortality rates and natality are defined.

I'm not sure I understood how the [what it was called? ?] points, the events occurring at certain points which go at 12 time steps, and I'm not sure I entirely understood how that operated, I didn't find that very clear. If we said that natality occurs every 4 months, how does that fit with the natality rate of 0.25? I don't know, I'm not quite sure what was going on there does the .25 mean that 1/4 of potential occasions there is a birth? Or is it the birth rate per month? Or are these 2 pieces of information the same thing so one is redundant? I'm not quite sure I understood what was going on there.

M: explain rate

So this is so what happens once every four months (M: yes) and .25 you saying that (M: every time on) in every four months there is a probability of .25 (M: the probability doesn't change) OK. (M: more explanation, and these concepts of involving time-steps are not obvious) Presumably something there was producing several offspring at [time-step ?] as well, but if you dealing with [foxes ?] which bread once a year [?] (M: I have no idea I know one interesting thing is they eggs as well - that's odd) and they lay eggs one at the time (M: and from those what is the probability of every one hatch and grow on But that's clear what point I should be looking at).

Modelling approach I think it's very simple. It's the model itself is quite well structured, clear groups and clear components and [?] and I think that yes, I found a bit confusing when you try to put it together, knowing what the different components were, you actually working I think in some sort of graphical you showed the graphical outline that's the structure of the model, and that would be quite nice if it actually appeared automatically as you defined it, so you [?] is always in screen and you see which part of that structure (M: you going through and it would be highlighted, or something like that) that would actually make it a lot [better ?] to find your way around the model as you go on it.

It was quite fast, it's [?] to set up (M: If you run it in your own machine it's quite fast) sitting there and define the components it's fast.

Is that all?

M: Yes, I think so

M: I wonder if you have any comment, maybe suggestions other than the ones you have just made.

I think I probably said most things. I think it needs some guide to take you through it. I think some sort of diagram to help you find your way around it, it probably be very interesting, make a big difference.

If this is going to be used by people who don't have experience in modelling, I'm quite sure that for what you are actually doing, you need quite a bit of help to find your way around.

Participant 6 (ITE2)

M: Can you tell something about you background, your experience? Your experience with computers, with ecology, etc.

C: I have got a degree in Ecology and I did modelling with Bob Muetzelfeldt in the graduation course and I was working for over a year and a bit using Maestro, to do simulation model, and I haven't done a awful

lot of modelling for a wee while [?] work and I'm coming back to modelling again now, [?] a year, and in the future I'll have to do more.

M: And what about computer, have you got any environment you use most?

C: I use Windows for model [community is ?] using much.

M: And what about tools? Modelling tools, besides Maestro?

C: Modelling tools themselves not so many, we use things like compartment component modelling, diagram modelling (M: Stella) Stella, that we all use a little bit, I haven't [?] FORTRAN.

M: Have you got any teaching experience?

C: Some. I teach the MSc's introduction to computing, that is just something I have done to [?] it would be a network here in Edinburgh.

M: Working with modelling, would you say you got a mathematical background, do you use maths in your work?

C: Some, yes. But I haven't used a lot of [dialectical style?] solutions to things more numerical. [?] I've done some maths but not strong at mathematics.

M: Which sort of how you implement your model, you do it from scratch, you use some skeleton with the main parts of a model, or some modelling tool?

C: Well it depends, at the moment what I am doing is using I'm recording someone's else modelling, which someone else used FORTRAN 77 and I'm putting in FORTRAN 90. Yes, in the past I've used Maestro so it was already written and most of which was written I added some bits to that code to change things.

M: Were you ever taught how to build a model?

C: Yes.

M: You attended a course or something

C: Yes, I learned at school, Bob Muetzelfeldt's modelling course.

M: And you went through several levels of complexity in models?

C: Yes.

M: Probably you have attended on your honours course was modelling from the ecology point of view. Wasn't?

C: Yes. I mean, my honours project, I build up an expert system using Prolog, using it for [adding ?] environmental knowledge, ecological knowledge, and at then someone could use it as an expert system, so that was my [5-week ?] project.

M: Have you ever applied your modelling experience in a different domain? Like business or industrial process or something like that.

C: I've programmed. I've never [?] modelling but I guess that program to the I written software to drive hardware for the monitoring equipment [?] CO2 concentrations and ever touch [?] a branch, so I had to analyse those trolley valves.

M: Would you say there are significant difference between ecological model and general or other sort of modelling?

C: I am not sure, I don't know, I wouldn't thought so. But having to think about that, and modelling ever [?] come out here [?].

M: When you're building models, you always have your complete conceptual model before implementing, or sometimes you have first some ideas of the conceptual model, implement it a bit and have some feedback and then go back to devise again? Are these complete separated stages or they interconnect?

C: It can be interactive yes. It totally depends whether you doing top-down or bottom-up modelling, if you going top-down then you will build a [raft ?] and you fill the gaps and you then improve the mechanistic parts, [?] subroutines to do anything.

M: Have you got your own algorithm of how to start a model? If someone start to describe a scenario, a situation for you, you can start to devise the model.

C: Yes. (M: do you have your own approach?) I suppose I have my own approach, probably it's the same of you and many other people's approach.

M: It is a sort of well defined heuristics or sequence of steps on how to do something?

C: Yes.

—— Demo and trial ——————————————————

M: What you think about the way the models are defined on this tool? Do you think the approach is clear, easy/difficult to follow?

C: I think you have to have a certain amount of knowledge in advance, for example, about the relationships that have been used, about the relationships the model runs its meaning, [?] what the time-steps are in order to relate to the rate step the user is using. That [means ?] help them is to build a [?] describes what things are.

M: Is that a good sequence, a good way to define model?

C: Models of that type, I guess yes, with population dynamics models. I haven't written very many population dynamics models so it's difficult to say whether it is like to fall short or if it is inflexible in some way and if I try to do something the system will let me do. It seems to have caught the population structures.

M: You think that would be similar to anything other modellers do in real life? When they're devising a model, do you thing they think that way? Is that one of the possible approaches to modelling?

C: Yes, that's one approach for sure. When define the populations and relationships between them, yes.

M: Do you think that could help a novice?

C: I think so, yes. As a teaching tool I would think it could be useful.

M: In which context?

C: Just simply teaching population dynamics, and the way that populations well, population dynamics basically.

M: Do you think it could help the users or the students to get a grasp of how to devise a conceptual model?

C: Yes, yes. (M: I mean, how to organised it, to get some structure to the reasoning) Yes, yes.

M: What do you think about the menus and the questions? Are they reasonable for that task.

C: I think I have not enough chance to really get a feel real feel for it, yes and no, some parts yes, some parts I got locked, the mouse buttons could [?] wrong one do something strange. Apart from that it was fairly well structured, it led you through.

M: Do you think that is an advantage in some way? mentioned white board versus hierarchical approach

C: Yes, [?] there are benefits for both methods, this is a good way of getting a model up running, way we have some idea about what the structures of things is or relationships between the populations.

M: You see any similarities between this and other tools you know of?

C: No, this seems very structured and it leads you through where you want where it wants to go really, to generate the model as you say, whereas in something like Stella is like the white board where you draw things and make connections between the relationships, so in that way you need to make your own mind

about how to handle state variables and the [?] between them.

M: Do you think would help if they user had a greater access to things like the inference mechanism or the knowledge base, or something like that? Could it have made any difference?

C: What sort of information for example?

M: explain possible rules Do you think the user would benefit from some graphical mapping (interface) to the inference mechanisms?

C: Not specially.

M: Do you think the system could give some more assistance for the user? Can you think of any sort of assistance it could be given to the user?

C: I can't think on the top of my head, what anybody else would might like maybe more ability to change parameters and settings between runs. I guess that is the way it is heading anyway, so you could change parameters and see the effects they would have [?]

M: Do you think for the audience we have intended is that a fair set of features, or you think it could have much more features which could help? Maybe this is the same question as before

C: The main features [?] get out of it and look at it, interpret it and explore its behaviour, I mean, obviously things like time-steps are important relative to [?] affecting the behaviour of the model and whether you have processes appearing at different rates, your account for the [?] there [?].

M: Have you got anything to add?

C: None that I can thing of. What is actually the goal you're aiming for, to develop a general tool or only in population dynamics? (M: explain why domain was restricted to population dynamics).

Participant 7 (ITE3)

M: Could you give me some information about your background, experience?

D: I did a biology third degree and a genetics PhD, and didn't do any computing before the PhD, and on the PhD I started a programming and then I came to this job where I had to learn about modelling which was 4 years ago, I've been doing modelling for 4 years.

M: But now you are very experienced in computing?

D: The modelling is all FORTRAN programs and all are written from scratch, we don't use any modelling tool, but we do have ModelMaker, but we haven't used it in many big models, and we did have a small model which... we have ModelMaker such that if anybody else in ITE wants to run a model we have the so in the tropical section they have a [monitored ?] model which we developed for an Italian ModelMaker. So, that was just once an exercise that we did but everything else is just written in FORTRAN.

M: Are these FORTRAN simulation models from a library or you have made from scratched yourself?

D: We have written all from scratch.

M: Have you had any teaching experience?

D: A little, I have various people who come work here and teaching them about the modelling, what we do, but I first have to teach them modelling too.

M: Would you say you have a strong mathematical background?

D: Yes, PhD was quite mathematical, there was some population genetics, quite mathematical.

M: You're certainly still using lots of maths today?

D: Yes.

M: Were you ever taught how to build models, how to do modelling?

D: Well, I started here, we ... I was simply given copies of the models that had been used within this group, and I have to started to learn how they were done and then I have to build other ones, adding at to coding so that I just used.

M: So you've learned by yourself?

D: Yes, well asking a hundred of friends who were here at the moment use his model, perhaps he'd used first, so I had to go through the code and then asking him how it works.

M: Is there any main method you have used? Systems dynamics or something like that, some paradigm... or you do what you need as you need it?

D: All the models we do here are processed based, so er, the other bits, the tree model we have sort of [?] for photosynthesis is processes based, but [with those ?] models you could take that bit if model and put into another one [?].

M: You ever do first the complete conceptual model and then try to implement or you try to think a bit about and implement a bit, and have some feedback?

D: Yes, I do that, we are actually trying to do a whole new model at the moment so we try to sort of deferring, leave that line out 'till what we need to do, and then coded that and then we will be adding more to it, and then run it so we have it working and then adding more detail each time and do another one. I guess is something rather important then add more complexity to it as we go along.

M: Besides ecology, have you had any other sort of modelling experience, like modelling in another area?

D: No.

M: But in your opinion is ecological modelling different from other sort of modelling, business modelling, industrial process modelling?

D: Well I don't really know, but I do (M: think how you fell about it) Well, I suppose one [?] that I have actually learning C++ because I am investigating object oriented design, which I'd like to know more about and I think we should investigate further so I'm actually learning it at the moment, I'm hoping to apply OO techniques for what we do, but I am not able to do that yet, not yet.

M: That's it I think, for the first part. Do you want to add anything, comments maybe, about modelling? Is it hard, easy?

D: It is very difficult because I mean coming here is very difficult to you couldn't go to a library and get a book on modelling that teaches you [?] really described it and have you started writing models and how to work and the [searching the shelves] it is not there anymore this one, but then I started to look at it and it is really didn't discuss the things that we needed here, so really was just the case of taking the existing models and try to work on how it was (M: there was no recipe on how to do general modelling) Not really.

— Demo and trial ————————————————

M: What do you think about the way models are defined in this tool? do you think the approach is clear, is it confusing, is it easy/difficult to grasp the idea?

D: Yes, it's clear, the way you have the components and then you [?] the components, and then have it defined within that [?], you select foxes and anything which would apply the foxes so that was clear.

M: Do you think the way the system works is similar to other tools you have seen?

D: The only one only tool that I have really used is ModelMaker, have you used that? (M: No, no.) That is I mean you have it's done more graphically, where you select component and then that is comes out with a symbol, which you build up a picture, so you have say 2 boxes with a flow, an arrow between them, but that [?] a component when you double click on a component it would come up with a flower in and a flow that [?] a component when you double click on a component it would come up with a flower in and a flow out, which you actually have to talk in the equation, that is a similar sort of thing, but I'd think this is clearer.

M: You think this system could help a novice user to get the idea of how to devise a conceptual model? Do you think someone using it had to have in advance the complete conceptual model or just pieces of it?

D: Something like the description you got here, you have to have that in mind first I think (M: A well-defined scenario, or something) Yes. (M: Do you think the questions asked Sorry) Sorry, it may not be completely defined but obviously what happens when you start to build a model and you can think of other things that you haven't though of, if you have something like that and as you building it you realise that you got to include let's say like immigration and emigration for example, and you have to build that in, then I think that something like you got there, it's a place to start.

M: The questions asked in the menus which pop-up, do you think they are positioned the sequence, is it sensible that sequence? or it would be better to have something more spread out and the user could just pick up something?

D: [?] sorted out how easy it is to go back and change something you've put in earlier, and we actually we tried to go back and tried to change the name, which obviously was a problem, but if you say you would set up the platypus and then you are half-way through the foxes and you want to go back and change something about the platypus, I don't know how easy that would be, and at the end maybe I think at the end of each one you had a list of everything you'd said, all the variables you've got, so I that let's you see all displayed somewhere I think it was, wasn't it? (M: Yes) at the end, if you could then change anything there, [?]

M: The graphical representation you think it would really correspond to the description you made? Do you had an expectation of what you should have?

D: The graph in the end was just the platypus numbers, the actual numbers of them so

M: Do you normally have an expectation, when building models, of what should come out? The actual values?

D: I'm not quite sure what you mean, do you mean the actual way of presenting it or the actual (M: values) Yes, I've got some.

M: Which sort of user you think could benefit from this kind of tool?

D: Anybody who is doing modelling, I don't know how bigger or profound [that could handle ?] it could cope with working in details on a particular area, then you might end up with quite a big complex model, I don't know how fast this would be.

M: Do you think the user should have access to the reason mechanisms like the knowledge base? The generation of code?

D: I would be curious just to look at the actual code that you came up with, but I don't know how many I guess if you haven't done any programming then probably you wouldn't want to see that side of it.

M: But probably you would like to have some assistance of the system. Which sort of assistance a user would want from the system, help facilities, etc.?

D: In terms of just building a model?

M: Yes, using the tool, say, "I should be able to see a graphical representation of the structure of the model" - well, that's possible. Let's say I would be interested in seeing the link, or the correspondence between some variables and the actual effect on the model, or things like that.

D: I actually was going to say I was going to ask you: say you run that, want to run it for 300 time-steps, say I want to have a look at the graph and there is something funny happening, if I then may go back and change it, say the mortality rate or the number of foxes or something, and then run it again, would I be able to see the two? (M: Yes, OK) to compare the difference like (M: Yes, not now, No) that would be something I would want to do (M: to have some sort of way of comparing different) different experiments, changing different increasing the predator rate and see when the population crashed, or whatever. (M: I think that's about sorry) I have just to run 2 or 3 times and I still be able to see all the plots of the end, just to compare them.

M: You think this system could help a novice user to get the idea of how to devise a conceptual model? Do you think someone using it had to have in advance the complete conceptual model or just pieces of it?

D: Something like the description you got here, you have to have that in mind first I think (M: A well-defined scenario, or something) Yes. (M: Do you think the questions asked Sorry) Sorry, it may not be completely defined but obviously what happens when you start to build a model and you can think of other things that you haven't though of, if you have something like that and as you building it you realise that you got to include let's say like immigration and emigration for example, and you have to build that in, then I think that something like you got there, it's a place to start.

M: The questions asked in the menus which pop-up, do you think they are positioned the sequence, is it sensible that sequence? or it would be better to have something more spread out and the user could just pick up something?

D: [?] sorted out how easy it is to go back and change something you've put in earlier, and we actually we tried to go back and tried to change the name, which obviously was a problem, but if you say you would set up the platypus and then you are half-way through the foxes and you want to go back and change something about the platypus, I don't know how easy that would be, and at the end maybe I think at the end of each one you had a list of everything you'd said, all the variables you've got, so I that let's you see all displayed somewhere I think it was, wasn't it? (M: Yes) at the end, if you could then change anything there, [?]

M: The graphical representation you think it would really correspond to the description you made? Do you had an expectation of what you should have?

D: The graph in the end was just the platypus numbers, the actual numbers of them so

M: Do you normally have an expectation, when building models, of what should come out? The actual values?

D: I'm not quite sure what you mean, do you mean the actual way of presenting it or the actual (M: values) Yes, I've got some.

M: Which sort of user you think could benefit from this kind of tool?

D: Anybody who is doing modelling, I don't know how bigger or profound [that could handle ?] it could cope with working in details on a particular area, then you might end up with quite a big complex model, I don't know how fast this would be.

M: Do you think the user should have access to the reason mechanisms like the knowledge base? The generation of code?

D: I would be curious just to look at the actual code that you came up with, but I don't know how many I guess if you haven't done any programming then probably you wouldn't want to see that side of it.

M: But probably you would like to have some assistance of the system. Which sort of assistance a user would want from the system, help facilities, etc.?

D: In terms of just building a model?

M: Yes, using the tool, say, "I should be able to see a graphical representation of the structure of the model" - well, that's possible. Let's say I would be interested in seeing the link, or the correspondence between some variables and the actual effect on the model, or things like that.

D: I actually was going to say I was going to ask you: say you run that, want to run it for 300 time-steps, say I want to have a look at the graph and there is something funny happening, if I then may go back and change it, say the mortality rate or the number of foxes or something, and then run it again, would I be able to see the two? (M: Yes, OK) to compare the difference like (M: Yes, not now, No) that would be something I would want to do (M: to have some sort of way of comparing different) different experiments, changing different increasing the predator rate and see when the population crashed, or whatever. (M: I think that's about sorry) I have just to run 2 or 3 times and I still be able to see all the plots of the end, just to compare them.

M: Would like to add something? Comments, suggestions?

D: It was great, it was good. I haven't come across anything like sort of takes you though step-by-step like that, which will be good.

M: Do you think the idea of leading the user through step-by-step to some well-defined stage is a good point?

D: Certainly when you first start a model, and then let's suppose when you come back to it later, or just going on and change what you built, more experienced you get on what it's doing, but certainly you got them started, that's fine.

Participant 8 (DAI1)

M: Can you tell me about your background, your experience?

E: My background is in terms of my graduation was in civil engineering, and through my civil engineering years I worked as programmer in some very complex let's say, ecological models, but I didn't understand most of them, 95represent and the simulation I didn't understand at all, and I move to AI and my background in terms of AI is more to theorem proving and implementation of some provers and PROLOG, and when I came to Edinburgh I started to work with ecological models using logical based systems, but I am not an expert in ecology, just someone trying to understand ecology using a logic based approach.

M: Would you say you have a mathematical background?

E: No, I would say I would have a more modelling system background, not mathematical background to model real systems but to model computer systems.

M: But independent from the modelling aspect of it, you have a degree in engineering, so you probably you have studied calculus and things like that?

E: Yes, some basics that every engineer... yes.

M: And you still use maths in your work?

E: Yes I do use but more maths not related to numerical analysis but more to an algebra and discreet mathematics.

M: Have you got any experience with teaching or tutoring students?

E: Yes, I used to teach computer programming languages and functional programming languages I taught twice, PROLOG I never taught before, and tutoring yes, a bit of it.

M: Have you ever tried any sort of modelling environment, modelling tool, ecological tool or any other sort?

E: I just tried once the FLOMO, but I didn't progress very much using it, I just built some very simple examples were shown in the manual, in the guide, actually.

M: Were you ever been taught how to build modelling (formally)?

E: No, I was taught how to model systems in general in terms of computer systems, but to model in graduation time I was taught how to model some physical systems.

M: When you think of modelling you always devise a complete conceptual model and then go for the implementation, or you tend to have some insight into the conceptual model, try to implement it and get some feedback and then devise or refine it a bit more?

E: I usually try to get a simple model first to understand how whatever the framework I am using for languages, or for the system, in the case of the FLOMO, and as soon as I understand the basics of how the framework helps me then I try to, I tend to leave more, stay more aside from the tool or the framework and try to build the conceptual model at once.

M: Have you got some heuristics of how to do modelling? Have you got your own heuristics?

E: Yes, my tendency is go first to the representation aspect, what is the best way to represent, and as soon as I feel that I have a good representation I try to go through the operations over the representation and then I just try to improve the representation and then of course the operation for the rest of it.

M: You usually think first in terms of conceptual model? Or you feel you have some mixing with implementation? Have you got any tendency when you devise a model linked with the way you intend to implement it?

E: Yes, there is a bit of mixing, yes.

M: Do you think there is any substantial difference between ecological modelling and modelling in other domains like programming or civil engineering?

E: Yes I think there is because for example in terms of let's say let's take an examples from civil engineering - that's my background - the structures of building, the laws, the physical laws that rules everything are well known, so the model will be just to satisfy the constraints that such physical laws impose to model a structure and how to measure the efforts over the structures and things like this, and for example, in a more abstract domain like business, I think that the way the domain does not have many things involved but more many relations, I think that ecological model is difficult because there are many things in the domain and many relations, and the difficulty is what exactly is important to the model, and how I can be sure that I'm modelling is exactly what is the actual real system, that is the main difficulty for me.

M: You think the most difficult thing about the modelling is the abstraction or the conceptualisation of the thing?

E: Yes.

M: And you think that possibly it could be due to the level of complexity of that domain, of ecology itself?

E: Yes, as I said, it relies on the number of relations between the objects, interactions, it is very dynamic and very difficult to establish a line, to draw lines, "interaction stop here".

—— Demo and trial ——————————————

M: Can you tell me what you think about the way models are defined in this tool, is the approach clear, confusing, easy to follow and get the grasp of the path to define a model?

E: I think that to learn how to use this system is very easy, but in the beginning I felt a bit tired with the different screens that appeared, and I felt a bit lost, I didn't know what I was doing, and I felt that when that picture of the classes that you show in that tree appeared, I could then understand what I was doing, and that would be, I don't know if it is just a reaction to the graphical approach or to the text and window based approach, and dialogue approach, but I felt that if I could in some way see what I was doing, and just the window appeared when it was necessary, that would be better, that was my impression.

M: Do you think this approach is similar to something you or other modellers that you know of, do in real life? Do you think that this step-by- step, hierarchical way of define things is present in real modelling?

E: In real modelling yes, but I've never seen any software that could give this sort of hierarchical specification, I worked just with FLOMO but I see many advantages between (over) this way of doing things and FLOMO's, and one way is the hierarchical stuff, and because the information that I can see things is much more let's say readable than in FLOMO, I don't know new version of FLOMO, I worked with it 2 or 3 years ago.

M: You think this sort of tool could help a novice to get the grasp of how to devise a conceptual model?

E: Considering the aspect of many windows, at first time I would say no because a novice tends, I think, to see things in a more graphical way, I consider myself as a novice, I tried to forget my background and tried to behave as a novice and my impression was that when you showed me that picture, I could say "now I know what I am doing", and if I could have instead of just a sequence of windows and text showing me things I could have just a general window showing me pictures of things, I could click here, and then a window appear there, and then I could have an idea about local things but relating that to the global class definition structure.

M: If you consider this hierarchical approach as opposed to the white board approach like FLOMO, do you

think with some graphical support and some help, a different, more graphical way of driving things, that could help the user?

E: A novice I would say yes. I think that this sort of tools I have just seen has more information from the modelling point of view than a white board, the only thing the I have objection in terms of a novice was these aspects.

M: Do you think these questions they're reasonable questions? The sequence, the decision points on the design?

E: Yes, the season points, I was a bit confused because maybe it would help instead of maybe would help to show pictures saying that you have chosen this stuff and you have these events, now choose how this thing happens in this time line. Because I had to make a division, I have to divide 12 by 4 to know that was 3 times a year, and in the first I didn't understand what I was doing.

M: You had no idea about the time structure?

E: Yes, that would help me to show the time structure you have defined for a classes or whatever and the events.

M: Do you think the models produced they actually they make sense against you specification?

E: Yes, I think so.

M: Do you have an expectation, when you are modelling, of the results?

E: No, just an expectation in terms of "the software really works?", and my surprise was that it worked.

M: Can you tell me if you can pick up the main drawbacks and the main advantage in this sort of tool?

E: I think I didn't understand your question well.

M: Basically, just you think are the good and bad (E: The software, the approach?) Both.

E: The software I think from an expert point of view, I think it can be used straightforwardly, just has to learn this steps and then you can understand how the structured way works, that would be fine. But from a teaching point of view I would, and for novices, not only users but novice modellers, I would say that the majors drawback of the system is that that picture is static in relation to the specification, that would be nice to have a relationship between what you shown and you could click, and if you change anything there (M: some way to check the current state of the model) yes, the current state of the model, and might be interesting as well to see the execution, what is going on, for example with the value of things for example, how many you have 3 values there, young, adult and old, for a certain group of whatever, and if you could see during simulation time what is going on, just a bar showing how that is growing or changing, and if the user could stop the simulation and say I want to change this now and see what is going on from now, I think that would give more dynamic to the specification, because suppose if I want to during to simulate something for a hundred years, but then the user wants to change something and 100 years it's a considerable amount of time-steps, it would be more than 25, it would be 4 times more than what we have done, and the user say "now I want to change something for 70 years", there is an event there that I want to change, and the user would have to wait until everything run and come back to the specification again and set-up everything. It would be interesting to have a sort of interruption button to say "stop the execution, I want to see the specification at this state I want to change it from now". And this is well, basically is it.

M: What sort of users do you think could benefit from this tool, or this sort of tool?

E: I would say that if these drawbacks and of course there might be others that I can't see , and maybe advantages for someone who is not a modeller. But if those things could be offered, I think this sort of tool could be useful for teaching, and even for experts. Of course in the case of experts, they would need more maybe more efficient execution. You translate your code to PROLOG, but you could translate to another code, or you could translate your code for a parallel model, or distributed, whatever and then an expert could really model big things. But I think this sort of tool could be very useful.

M: Do you think the user could somehow benefit from seeing, or from having more details on the inference

mechanisms, like the KB, the rules. Of course, not direct PROLOG code but through some graphical interface or something.

E: Yes, yes, yes.

M: What sort of assistance you think could be good to have from this tool? Of course, you said you had some difficult, you were a bit lost and some support on guiding you through the navigation. You also mentioned some way to see the current state of model or of the definition. Can you think of any further assistance you could get from it?

E: I think that maybe (M: any other sort of features) I think it would depend very much on the kind of user that could use this assistance, for example, you cannot offer such suppose that is interesting for a modeller to see even the inference mechanisms generated in PROLOG, but for a novice that's not important, because you could translate that for a Natural Language for example, presentation. That's would be fine, I think. So that the modeller could see if what he/she has just defined have some link with the NL expression of that stuff like the class structure, all those "ageing" when the thing happens, see the library of or definitions of things. But for a novice, [?] then I think that would depend very much on the user model, I don't think that just to introduce this new any new features to the system would make the system any better, just because you're introducing that.

M: I think that's about it. Have you got anything to add? Comments, suggestions?

E: No.

Participant 9 (ITE4)

M: Let me quickly explain what I'm doing here... I'll ask you a couple of questions about your background and about your ideas on modelling things like that and then we'll try to do the demo of my tool - my system. After that I'll ask you to fill up a simple questionnaire, it's a sort of standard questionnaire just to say - just react to it - it is not actually evaluating the software - just react to what you see - it's a simple one. After that just one or two questions more and that will be it. So, hopefully, everything will last about one hour, given just conditions on the network - because I need to connect to my department and they got some security problems there and you'd have to connect now just through one machine and this machine, of course, is very busy, lots of people outside the university trying to connect, sometimes - well, anyway...

M: Could you tell me something about your background your experience is it ecology?

G: It's ecology, forestry, land use... more of forestry ecology. And temporary areas under the tropics as well, I worked I was working in Cameroon for 3 years and had works in Sudan and Gana, Nigeria, mainly West Africa. On forestry sylvan-culture and also in the ecological aspects on tree growth. And, to some extend, working on nurseries - the project I doing at the moment is working, amongst other people, with Robert Muetzelfeldt to coordinate for a forestry research program of the Department for International Development on this project called the agro-forestry modelling research coordination, so that involves seeking to unite the people who is doing research into agro-forestry with those who are able to model the different aspects of the competition between trees in the [ponds ?] for light, water and nutrients so they have quite a large group of people working together towards the same goal only it has only 3 or 4 months left to go, so the program itself is almost finished.

M: I see you have a good background in modelling. Would you say you have a mathematical background?

G: Not really

M: But do you use maths in your work?

G: I wouldn't say I understand maths particularly well, no, I am able to programming crudely in FORTRAN and understand other people's programs in FORTRAN, but I wouldn't say now I do very much programming myself. I may have done 20 years ago, but not anymore.

M: And nowadays do you use any sort of computing support, like spreadsheets...

G: Oh yes, spreadsheets of course, and SAS - a statistical analysis package which is able to... you are able

to write [?] computer programs with that. And I've looked at ModelMaker but not used to use it [?] and as said just FORTRAN.

M: Were you ever thought how to build a model, how to do modelling?

G: I suppose I was, I did a course - because I did my undergraduate degree here in Edinburgh and there was someone called David Gifford who thought a module called Ecological Modelling, so I did that module, but that was a long time ago. He's been dead for quite a long time, dead for almost 20 years and he went to Brasilia, after left Edinburgh wanted to be professor of ecology, I think actually in Brazil. In the computer room in the Institute of Ecology and Resource Management they still call it "the Gifford room".

M: So it was really a formal training in modelling.

G: Yes, a little bit - just a small course.

M: Have you had any teaching experience?

G: Any?

M: texing?

G: Teaching experience of modelling?

M: Not only modelling, but in general.

G: Or other thing - er, lecturing I do a little bit of a course for the MSc in agro-forestry at the university which I do for 4 days a year or ... So no, not much.

M: Still about the modelling stuff: Do you think that modelling in ecology is substantially different from modelling in other areas, like business or industrial process?

G: Er I don't know - never thought of it. Probably modelling will deal more with flows, inputs of resources, but could a [flow] of nitrogen soil could be treated in the same way of as flow of money, I don't know, probably not. Although they are all... there are so many similarities between modelling in ecology and modelling in engineering, most of the business modelling I would think. But still, I think is a very important area to develop decision support models, so management support models within ecology and that bring you into business. So, I think a lot of ecological forestry - for example - models can't just look at the environmental impact or the effect on the yield of a particular land managed. I would say that further a look at the impact on people - now either that's an impact which comes of the profit that a farmer gets, a profit that a state gets or whether you need to look more [at more projected] model, now so [linear programming] type of model... er... you know, what [approaches] are valid I've to some extend cooperated with a guy called Lloyd [forceps?:-] in the Agricultural Department and he uses linear programming a lot for agricultural decision-making. So, some are aware of the two approaches.

M: I'm a bit curious about the approach normally have with modelling. Yourself, in your experience, how do you start a model? you have first a complete conceptual model idealised and then you do the implementation or you try to idealise, to devise a bit and implement a bit and get some feedback?

G: Yes, in my personal case, probably as Deena will have told you it's more a matter of taking another model that exists and looking at what the other does and try to implement that in a slightly different way, rather than starting from the very beginning and try to write a whole new model.

M: You normally take a skeleton or a basic structure of some model and adapt that one to your needs?

G: Yes, if you trying to build an ecosystem model you'd look around and you'll see "who has got that soil-process model" or "who has got a model of tree grow or something" and then put the two together rather than from starting from scratch.

M: I think that's a good start - for this first bit.

— Demo and trial ————————————————

M: Could you tell me what you think about the way models are defined on this system. The approach is clear, is confusing to follow, is it easy to understand how models are defined there?

G: I think you could probably have more prompts for each of the questions it would be handy to have a little box that when your cursor is over that box then you would have (M: Windows-like) yes, Windows-like, that than explains it the terms and give some possible options. I think probably it would be useful as well if that's a general model, to have goal examples changing according the field, so you could have an engineering set of examples or an agricultural set of examples of forestry set of examples and that would one of the prospections in what type of model is this likely to be and then the instances and the groups and so on, because the examples could be engineered, could be modified to be appropriate to the type of user just like a hot water [padlock ?] if you're building one set of help files you might as well do four set of help files with different strategies.

M: Do you think this approach is similar to other things you have seen around, I mean, either programs or support systems?

G: I've never seen a front end for Prolog programming like this before, so it's not similar to anything I have seen. All that I've really seen are things like ModelMaker and Stella and I haven't used those a great deal and there, it's true, the help you get to building models is... if someone is coming through "cold", a non-expert user, the help the on-screen help that you got on those tools is not very great and you would expect to have to read the manual for one day or two to be able to build your model, and... (M: and have knowledge of System Dynamics as well) Yes, a little bit. The other... Did Deena show you the front-end that we were (sorry, this is not a direct answer) did Deena show you the front-end that we developed for our FORTRAN models?

M: I've read about... is it HyPAR?

G: Hypar, yes, OK - it's written in Delphi... it's only an assistant to parameters setting, because the model has 100 parameters all are different things - it's just explains to people what the parameters are, which are important, which are not important, what values you might like to put in this place, it's not build a model... but er so that's the.... most of the experience that I have had with front-ends but I'd put that in back-ends as well because you can use it to choose, to select the type of graphical output that you want from the model.

M: About the class of users - do you think this tool could be, of course, when properly finished, do you think useful for which class of users for novice (could be supportive) could it assist in tutoring, when someone is teaching students how to create models?

G: Yes, for the type of Leslie-matrix model you are using here that could be applied in a great range of biological examples that could be applied in forestry for example also, competition where you can define the competition rules between individuals species of trees or individuals - individual instances of each species, if you can define those rules in terms of demand of light and demand of nutrients, demand of water, then Leslie matrix models like this would be very useful. So yes, you could use that in agriculture, forestry, in many branches of biology and I think by doing this you showing that is possible for students that are put off by modelling, show them that is quite easy for themselves to build their own models I think it's a very commendable effort.

M: Do you think that could somehow get the grasp of conceptual modelling? I mean, give them some sort of structure to modelling, even it a basic structure?

G: Yes, although this is just one particular type of modelling, yes I think it could be useful.

M: I think... that's mostly...

G: How would you apply this to compartment flow models?

M: I could - let's say, translate some compartment flow models in this approach, and also the other way around, I think so. But there is no direct mapping between them.

G: Because I think for this to be generally useful across a whole range of biological applications you need not only having matrix [points ?] in it but you would need to build in some compartment flow although that's necessary given Stella and ModelMaker and all these other packages that are available for that... but to make it more comprehensible, or maybe you can combine it with AME.

M: Yes maybe, yes, connecting it with one of special slots in AME. Actually, Jasper Taylor - which is the AME responsible person, he tried to put this scenario into AME. Yes, it worked well (more or less), because

there some structures in the age-classes and things like that which are not quickly done in a compartment flow modelling, you have to have some basis to do that - some previous knowledge. I mean, it's possible to do but you have to know how to do it. Well, I think that's about it.

M: Oh yes, you have mentioned which you would like to see of assistance on this tool, but can you think anything to add? any suggestions? comments?

G: About your program?

M: Yes, a bit more assistance to the user, anything like that?

G: Some of the variable parameter names that you were using, were not very self-explanatory, and again, providing an expanded name for them and perhaps boundaries, if you're asking someone to set a rate sometimes people don't know what an appropriate rate might be. So suggestion that a rate of 0.9 is not [lined ?] but people would find that out by trial-and-error. It's mainly that, if I think of anything else, I'll e-mail you.

M: I'll be very grateful - well, I am very grateful. Thank you very much.

G: Can you give me your e-mail?... Because what I want to show you is there is one model - if can find the paper for it there is one model actually amalgamating this Leslie matrix approach for trees in a population with a bio-physical approach to competition for resources and the two are combined together in one model so if I can remember where that was published.

Participant 10 (Agric1)

M: Could you tell me something about your background, your experience?

G: Yes, I always find that a difficult question. My first degree was in ecological science which I did here, and then I did a PhD in environmental physics and then I did a post-doc and then I was brought here as a lecturer in agriculture in [?] crops. My particular interest are in the ecological side of agriculture, that is explaining how farming systems work. [Increasingly ?] I've got also interested in the human-side of agriculture, because many of the things influencing agriculture have nothing to do with the biology of it but it have to do with the decision making by human beings and the way things are [adapted ?].

M: Have you worked yourself with modelling somehow, or worked with people who do modelling?

G: Both. I've built models of various sort, generally crops. I have interacted a lot with people who build models, I have a considerable degree of scepticism of how effective many traditional models are, but I recognise that in some cases there is no alternative to modelling. And I think that in some cases modelling can be able to, at least, have good results as traditional [ways of doing things ?].

M: Would you say you have some mathematical background?

G: Best then most of our undergraduates students these days, but then that's not saying particular much. Yes, I like to work with numbers.

M: So nowadays you still work with, you still use maths in your work?

G: I'm still using simple maths in my work. My differentiation and integration are a bit rusty but I can still do it. I recognise that are various properties of equations that are necessarily consequent from the equations, and then if you're modelling them correctly you got [you answer ?] that is purely because of the mathematical formulation, nothing to do with the biology of it.

M: I see you quite at easy with computers. Are you a computer user at normally... daily?

G: Too much, yes. Although my computers are on all the time... I'm in a much of [computer user ?] for fairly trivial purposes. I'm very much interested in diffusion of information by computers.

M: Have you been involved with teaching or tutoring students, maybe novices?

G: Oh yes, yes, I work with students, a lot of the time. I have used various models with them, apart from

the [featuring program ?]

M: Which sort of... I mean, talking about modelling, do you use any sort of environment, or programming packages, or something like ModelMaker, Stella...?

G: I know that these things exist and I work who use these regularly. I, myself though I'm afraid most of my programming is very crude, it's sort of basic, basic level or spreadsheet modelling, because I can often do the things I want quickly in that sort of environment. Although I'm considering making a great leap forward to Java, after having missed all the intervening stages, because that can bring things that I want to. I'm not by inclination a computer programmer, I'm much more concerned with the information that lead us to build computer model. So, my ideal way of working is with someone who is an expert in programming and can do that job well, because my job is to explain [?] to the programmer I understand what [?] that I know about the system.

M: Were you... have you ever been taught how do modelling, how to create models, I mean, formally?

G: I haven't been taught formally, no. Because my undergraduate and post- graduate education preceded those things. And a lot of what have been [learnt ?] about [from a law?] very few people actually thought of carry out modelling effectively. And there is a lot of bad modelling [because of it ?]

M: When you think... yourself, when you thing about modelling, do you first devise a complete conceptual model before trying any sort of implementation, or you try to have some insight first, and then try to implement something, get some feedback and go back to the conceptualisation stage?

G: I prefer to work with the conceptualisation as far as I can, for some purposes that's all you need to do, I mean, you don't actually... by the time you conceptualise then you remove the need for the model itself, and that is the area that particularly interest me. I think if you don't do it that way then there is a very great danger that your model is not going to adequately represent reality. It obviously goes round in cycles and, if at some point you have to admit alterations to it if the model is producing answers that... answers which you have good reasons to believe are wrong, then you need to go back and re-examine the conceptual model, but I think there is a very [margin ?] stage of model creation even before producing a conceptual model, which involves going and look at the realities that causes the variation in the output of whatever is you trying to predict, certainly in agriculture many of the models of crops are being produced [immediately ?] assume that the key variables are going to be soil [i ?]radiation, temperature and water and there are plenty of practical situations where none of this is responsible for [site to site ?] or [nutrient ?] variations, these are other factors which are often missing from the model, so that's a failure in conceptualise the problem.

M: Do you think there is any substantial difference between ecological modelling and modelling in other areas?

G: There is a big difference from ecological modelling and engineering modelling, because engineering modelling can be much more tied into the laws of physics which are [genuine ?] laws... biology has got fewer [than ?] laws and we dealing with chaotic if not stochastic environment and that means... a lot of ecological modelling requires on... depends on empirical relationships, so they not quite sure how reliable those empirical relationships are, so there is the problems of natural variability. It depends on the particular system you are dealing with. If you are dealing with something like Population Dynamics of a species, then I think that is rather easier to do, rather than looking at several crop [meat ?] variability the sort of things I'm interested in, and then there is a lot of other factors that may come in and completely change the progression of whatever is happening. But that might just me as a plant scientist looking at somebody else's area and saying "it must be a lot easier".

M: Just one more... What do you think is the more difficult about modelling, in your opinion? (G: Compared with what...) What, if you have to start a model or if you have to teach someone how to do modelling, what you think it would be the most difficult part of it?

G: I'll answer a different question to begin with. I think modelling is the fundamental scientific activity, and we all do it, we don't know we do it and there are many people whose the difficulty actually is in conceptualising the whole lot. It's a lot easier just to carry out experiments of [?] treatment and see what happens. But that itself ends on an underlining model, which is a very crude model and probably there is no much relationship to reality. So the difficult thing about modelling is actually the whole concept, the idea of dealing with things, with objects as systems with complex properties. I think one should done that,

then putting the thing to the form that can used by a computer its relatively straightforward. I am sure... it's a difficult intellectual activity.

M: Do you think modelling is a difficult activity to learn?

G: I think lots of people model without real awareness that's what they doing. And some people are able to have this sort of insight where... you can [tear ?] in their heads over a series of changes of arguments, another people find it very difficult to think in that sort of way. Yes, I can see it in my students, some people just want to know what's the answer to something, rather than understand it, and it has to do with understanding. The more modelling you do of systems, the more you realise how little you do understand about things that are really often very important. We can often very well things that are trivial, but I'm interested... I consider myself, being an agronomist, as a sort of engineer, I'm interested in knowing about natural and management systems because I think you need to be able to manipulate them more efficiently and effectively. So I have to deal with the real world it's not good enough to have a rather simplified model system that works perfectly well. What's that got to do with practical farming or practical forestry, or wild land management [?].

M: I think that's it for the 1st part.

—— Demo and trial ——————————————

M: What you think about the way models are defined in this tool, in this system? Do you think it is clear/confusing/obscure? Is it easy to follow, the steps or they are complex, complicated, you don't know where to go next, something like that?

G: Most of it was clear. Sometimes I wasn't sure, but have done it once it would be easy to do it again. I think there is definitely an opportunity for documentation which might... not necessary on screen, just that give you some of the... some more detail about it. Which [ones ?] may be... may be [bridging ?] the ecological terminology and the world of computer does it.

M: Do you think this approach is similar to anything that modellers actually do?

G: Oh yes. Very much so. (M: Go through some landmarks, in a step-by- step sense way?) Yes. Yes.

M: Do you think that could be useful for a novice, for a student, to have this sort of guidance, when they don't have any experience to get ideas from? Do you know what I mean?

G: Yes, I think people how are new to it would probably require more help. I knew what I expected to see and... and to go back to an earlier point, I think even in the system I had lots of questions I could have asked about the simple system, about the way of... the exact meaning of the various representations was... A student would necessarily pick up some of these... these points. On the other hand, maybe I know too much and that makes it more difficult for me because this could be either of all sort of things. And I think when you trained someone in a system like that, then they can go and build lots of them, that's where the power comes in. I don't think you should expect someone to be able to sit down like this and in ten minutes be able to do everything that it is capable of. That would make the system too inflexible and not powerful enough and you should expect people to have to invest some time in actually learning how to do this. And then having done that, be able to generate new models by elaborating the existing ones. So that one of the modes of teaching people, it actually do a step-by-step description of what you're doing and why it is. And then say "right, you do it for this example now".

M: I was just... I didn't think I've shown you... show other feature of the system, namely, new process definition.

G: That was a very powerful sort of thing. But students often find it difficult to think what you mean by a process, and did I do myself at times, because there are processes that can be divided into other processes... and er yes, it is important that you actually.... getting people to think in terms of processes is an important part of this, and when you...

M: Oh, do you think so?

G: Oh yes, yes yes. I don't think it is necessarily a normal way of thinking.

M: Because some people probably would say "Oh, a process-centred way of viewing things is not a very

good one" but of course, it depends on what they're talking about.

G: In this sort of case you might say well, "foxes reduce the population of platypuses" whereas in fact... what's happening is that foxes are killing platypuses which reduces the level of the general population and reducing the potential for new ones being born, it's much more complicated than that. And if you short-circuit the whole lot, as many people do, it could be difficult to get them to think about all the component parts. Often... lots of students would tell me that shortage of water reduces the yield of crops, yes we know that, but why - why lots of the processes in which it is operated... (M: What the causality is) What the causality is, yes.

M: Do you think that in the system you have defined somehow the results corresponded to what you expected?

G: Well, I'm not surprised, because I know this sort of systems, these systems are very much dependent on the values of the system. What I don't know is whether I have actually implemented this system correctly in front of the numbers. It might be me, my implementation of it, or it might be the property of model parameterised in that particular way.

M: But normally you have some expectation of your model...

G: Yes, yes yes. You can generally look of something and see if something is obviously wrong. But there are other cases when the model might be wrong but there's no [end ?] indeed I know of a number of cases of models which have got errors in the model [?] and which people have been using it and have been happy with it for many years, but they still wrong in some occasions. There's nothing you can do about that, except some means of quality control and checking. Presumably the way you check this sort of system would to remove various processes from it and see what happens with the result. So if you had no foxes for example and still large number of platypuses were dying, then you probably want to know, you probably suspected there is something wrong.

M: About this tool... not only this tool, but this sort of tools in general, what do you are the main drawbacks and in the meaning, the main advantages of it?

G: I think there are clear advantages to any sort of modelling software that enables you to actually generate models of real systems. I think that's... you can test various scenarios and generate the set of results that can be looked exactly the same as you have looked upon them actually going out into the field and make real measurements. You must never confuse the two, but from the point of view of certainly initial work of teaching students, it's very powerful indeed.

M: So you think novices would possibly benefit from this sort of tool?

G: Yes, I think so. Because you... if you can, for example, generate some of the classic population dynamics on a system like that and show how it might be, I think that's a very powerful... much powerful than be told these things were happening. Because then you can start to answer a more... even more interesting question, which is "Under what circumstances is it true that... the following outcome takes place..." I mean, the drawbacks for it are that some people find it very difficult to work like this, and you do need... you always need support. If you are using it as teaching tool... has this been designed for a teaching tool, or just in general...?

M: In general, but I think the audience which could really benefit from it would be really novice.

G: Yes, OK. But if you doing that then you need good support from the start, to explain things. Because always there will be people who don't quite understand or alternatively they might have a much too deeper understanding, if you see what I mean, an so they ask sort of questions like "what exactly do you mean by mortality in this particular...". (M: time- steps... what do you mean by it) Time-steps, yes. And many of these things that appeared to be simple initially, are in fact often really quite complex. If you are dealing with students, you're dealing with a very wide range of particular skills, and what they're good at. This sort of things work quite well if you working with groups of 2 or 3 people together, because somebody get different ideas coming in. And that could be very powerful way of doing things. So sometimes not having enough computers can be an advantage.

M: That's a good point. Do you think somehow the user could benefit from having greater access to the inference mechanisms, or to the knowledge base, say, the rules stating when you can have a predator-prey

relationship or a competition between... species or something like that?

G: Yes. I think... (M: I'm not meaning through pure crude Prolog, no, I mean through some graphical...) Yes, because I think that itself is a useful exercise, adding to the list of predators for example, specifying attributes of particular classes so it's the system... but then you will have the system being able to use those particular sort of features but since the system knows what it contains, then you don't need ask you for information which is irrelevant to it, I guess. Yes, I think that's a very powerful sort of thing. I know with time you could develop it in quite a large system. You commented on the fundamental problems of update your bases though, and things you need caring of "are platypuses say in [Queensland ?] the same as platypuses in [New South Wales ?] and things like that. And so you might be have to some way of tackling that sort of thing, dealing with conflicting information as well. But I mean, these are all problems that can be dealt with the way you designed the system.

M: Which sort of further assistance you, as an "almost user" would like to have from tools like this one?

G: What sort of? (M: assistance). How do you mean what sort of assistance do you want?

M: A data-base of examples, or maybe some pop-up help as you suggested...

G: I would like pop-up help. Which doesn't... would not necessary have to be fancy and clicking on a term on it. But might be... have a separated help file where you can just scan down through a list of terms, whatever things you'd need.

M: You could somehow switch on and off...

G: Yes, you don't want it there all the time, to do that. You got a certain amount of help on the screen and if you had any more help in this format then the screen would get very crated and it wouldn't be useful, but would be good to have another... a separated source of help, that sort of thing.

M: I think that's it... Just one more: how would you place this sort of tool among the ones you have seen...

G: This is quite good, because you can have... it clearly outbreaks on age classes, and some of the other things I've seen, well, the traditional systems dynamics sort of model finds rather difficult to handle age classes (M: you have to simulate that...) you have separated... they're all separated instances of the same object, and that's kind of complex, so it's really quite nice like that points to screen. I think you could in... there are rather sort of standard... sort of population dynamics processes, there are a lot of alter relationships and things that could come in to something like that pointing to screen, or maybe they'd become the merging property of the system after having... that would be a nice thing to do wouldn't it? where actually you told that people have discovered there is this relationship and actually you performed your simulation the answer was... oh look, it's exactly the same as was found by some other group of people. So, for it does I think it is really good. I mean, I'm also seen - do you know Robert Muetzelfeldt's AME? - (M: Yes) OK, which does a different sort of things, and it does that very well, but I thing that for this particular class of things, it quite nice. The trouble is that on AME you got to think of a huge variety of possibilities so you got an enormous system.

M: The approach is different, it's a "white board" and you can start to put the things on... (G: Yes) But of course, you have to have some experience, (G: Yes) you have to know how systems dynamics work, (G: yes) you have to know what a flow, a compartment means...

G: I mean, there are all sort of things... it would be nice to have a natural language input into this... (M: That's one thing I haven't thought yet...) And because you actually dealing with various objects, it would be quite easy to translate them to a number languages, because it would be only the words you would have to change, the underlining machinery remains the same.

M: I think that's it (almost on time). Anything to add, any complaint?

G: OK, good. It was quite nice, I've enjoyed that.

Participant 11 (SNH1)

M: Could you tell me something about your background, your experience (H: in modelling?) yes, as well, but not only.

H: I did an ecology degree at Edinburgh, between 1976-1980, I went to Australia and did a PhD on impact of rabbits in the vegetation of the arid zone, I was there until the end of 1987 none of this included modelling, although I started doing some quick [?] the end of my PhD, on the competition between well, actually some population dynamics of rabbits and kangaroos, a competition between rabbits and kangaroos, there comparing their impact in vegetation, but never using any packages at that stage and then came back to Britain, I spent a year working to the Native Conservancy Council, the predecessor of the board where I work now, things on research work on the impacts of grazing on vegetation, and starting to use a very simple model but from the [Macaulay ?] Institute something called [Macaulay ?] Institute that is now in Aberdeen I reckon, a really really simple model, just a couple of equations really, that predicted the number of animals that a particular real cycled support predicted the vegetation and then you can predict knowing how much an individual animal eat so you then say how much there is available for animals and therefore how many animals you can support, very, very simple, we started using it as an advisory tool, but there were all sort of limitations with it. So when [the Colley ?] decided that they would get the money to build the better version of this model and we could add all the flexibility that we wanted, could [?] variation for altitude and latitude, in terms of production, included seasonality of the animals, numbers of animals, when the animals choose to graze, so they took me on to produce this new version of the model which was quite a gamble considering that I haven't done any modelling in my life before, however they were then based in Edinburgh, they moved up to Aberdeen, and I spent a last time, a bit of my time in Aberdeen, but for the first 3 years or something else in Edinburgh working in this model, we started with a brand new model, because the one that I inherited, the [Macaulay ?] one, although the guts of it were very simple, surrounded by all this miasma of programming using Basic I think it was at that stage, and it was impossible to follow and with all the things we wanted to do we just decided to start again with a complete new model, so that took me about 3 years to get a working model, I worked in FORTRAN initially, and by the end of 4 years we had a manual for it which employed some software consultant to [?] using a front-end in Visual Basic, and so by the time I left the [Macaulay ?] which was 4 years later we had a working decision support tool, essentially. I then left the [Macaulay ?] and I came back down here again to work at the Scottish National Heritage, which is the body that I now work for, I am in Advisory Service section in the [?] group, I give advice on land management and particularly grazing management for [?] conservation purposes in particular, and I actually find I use the model quite a lot, so help for set appropriate stock levels because basically it works for sheep, it is designed for a whole systems in the UK, you might have different vegetation times, the altitude, location of [line ?] the grazing [line ?] the numbers of sheep, the size of the sheep, and have along each month of the year and it also the users forging behaviour theory to predict where the animals should go in each any month of the year, in each day, that's in a daily basis, and, predicts the production of vegetation less some extra fall, so that you know what is available to the animals [?] stability of the vegetation in each day of the year, and knowing how much is there and statistics, it lets you handle simple heuristics of where you think the animals will go, and [?] and in the second year [?] it's stabilised by the second year, and it gives you the results for the second year, and usually what people are looking for is things like how much is the animal production for each vegetation type is there to be eaten by the animals, how much do the animals actually get to eat throughout the year, what's the digestibility, I know that is available in the outputs including extra information about biomass live and dead and all sorts of things. So since I've been at the Scottish National Heritage I've not done any modelling work because what we do is give advice and managing people, which is rather unfortunate, but we [?] building now new versions of the model but are not developing a model as such they build the next generation of modelling, all teams of people working on it so that is it.

M: Were you ever formally taught how to do modelling?

H: I actually did Robert Muetzelfelt's course when I was undergraduate, in 1979, but that was called quantitative resource ecology in those days and half of it was statistics and half was Robert's modelling component, which was very basic, it was basically just compartment and flows and time-steps, the basic concepts behind what's modelling, when we actually came to build a computer model, all that was pretty intuitive really, what you need to know to build a computer model is how to program, and specifically they taught the techniques of programming which are appropriate for model building, and you don't even need that now wit things like ModelMaker around, you just need the concepts really, so my feelings are that as an ecologist you don't need to be taught how to model, because you're building models anyway in your brain,

it's just formalising those models, I didn't feel as I need to be taught how to model, I mean yes, it is useful to see examples of other people models and how they did go and how they structured them, and yes you do need to know about time-steps, compartments and flows and the different options on how to conceptualise your system, but what I really need to do is to break that barrier between being field ecologist and building a model that worked was to know how to program, so learning FORTRAN was this well FORTRAN wasn't the best example because it is not very easy with FORTRAN to produce graphical output for instance, it is easier now with these bits you can plug in but in those days when first started it wasn't, you never [?] a computer in those days, and so now with languages that make it easier to produce usable output I think the whole process is a lot awful easier, but there is a sort of barrier there I think between the field ecologists and the modellers, and I find it a lot, I mean, I think model is just a tooi and every ecologist should be able to use, in order to build models or whatever, but a lot of ecologists just have a complete mental block when it comes to producing things in a computer that any [?] processors, it is quite amazing. But I think I haven't had, the block that I had was just knowing enough programming to produce something I could think, "Oh, I can do this" and I take it from there. There certain things about modelling in FORTRAN that aren't explained in FORTRAN books, but that you need to know to be able to build a successful model, I did think actually that when I finish that it would be a very useful textbook, would be a textbook on FORTRAN programming for ecological modellers, with examples and advice, you know, I didn't need to know all this stuff they told you about and that you never going to use it, and they didn't tell me about the things, you know, most of them never mention common blocks and that sort of things you need to know to build a good well structured model, maybe other languages are the better, anyway nobody is going to program in FORTRAN now but C++.

M: Do you always divide first the complete conceptual model and then go for implementation or sometimes you only have an insight of the model and try to implement the main parts, get some feedback and refine it?

H: I've only got experience of building this one model OK, it is a reasonably large model, with lots of bits to it, but only this one, and we built that in stages, I had a concept of the whole thing, but then I would work on little bits and get them working and verifying them, and then go to the next little bit and get it working, and then yes, you try to add them up and make sure that also make sense.

M: Which sort of tools you use, or people in the modelling community you think are using now? You've mentioned ModelMaker.

H: Yes, Model maker was one, actually I said I've working in one model but I did start some work on one model with a chap called [Neil Krat ?] he was quite involved in making building ModelMaker [?] in Nottingham, and we wanted to build an improved vegetation production model, this was when I first started working at the Scottish Heritage, and I though I might have some time to do some research before I knew any better, anyway we started to build this model, and I suppose the way we did that it worked really really well as a team, and because he is an agriculturist by training and I am a ecologist, so and he builds models of crop production, so I guess we can say at that stage we were using ModelMaker because [Neil ?] had a prototype version of ModelMaker, but think anyway it was the first version that was out, commercially, so the way that we did that was that we basically drew diagrams, roughly the sort of things that we wanted and then because Neil was already so familiar with ModelMaker he could just implement it in ModelMaker really quickly and then using that, [?] seeing that we could produce results immediately, which we did, we tested against some data we had available, which was quite good and then we though about ways that we needed to refine and improve it, and put extra bits in, and we could do all that with ModelMaker, just as it was, what was missing was the data, the next step was to go way and actually find relevant the nature relationships, what was the function that applied, what the data was like and that was the step that we never got to do it unfortunately, we knew exactly what the next step was but other priorities to go with, so we stopped at that stage.

M: Would say you have a mathematical background, or better saying: do you use maths in your work?

H: I don't use maths in the work that I do now, because it doesn't require it, much of what I do now is management

M: Have you used it before, when you were modelling?

H: Yes, you have to use maths obviously, I did sometimes think that if I was a mathematician or was working with a mathematician I might have developed different relationships so I could use equations or different

functions, but I might be more familiar with or structured relationships between things in a different way, because all I could use was reasonably simple maths I was familiar with, because I simple didn't know what else was available, so yes I mean, you have to use some maths, but for the model I was building, maths didn't need to get very complicated, it is not a I don't use different equations just step functions.

M: Have you got any teaching or tutoring experience? (H: In what? Anything?) Yes.

H: Yes, I do some teaching for Colin Legg every year. I do lecture on modelling and then run up a [top ?] session with students, and we occasionally have training courses. I've work quite a lot in one way or another, I just did some other teaching in the University on training courses for other staff, and [?] as well, I just do teaching and all sort of things from time to time.

M: What do you think could be the most difficult part of modelling? What you fell people think is the most difficult?

H: Well, if you start using natural models, I think the most difficult part is just this mental block that they have towards implementing, towards using maths quite often or implementing anything, because a lot of people who go into biology I think don't have a mathematical background, I did maths up to my last year at school, reasonably hi-levels at that stage, but not after school, but I always liked it, I not had any problems with it, a lot of people who go into biology I think to drop maths quite early on, and they go into biology because they are not very good with mathematical side of things, so anything which is a little more or even mention to maths turns them off completely , so this mental block when they have to think in equations is going to be very difficult. But as I said, I've never found that a problem, that's OK, so [crystal ?] speaking, what I find is the hardest part of modelling? What takes the longest time is finding all the right information, trying a whole lot of different ways that you can implement a model and deciding which one the actually the best way to go for, that's it. So I guess a model structure that allows you to very easily to put equations and allow you to play around with some of the answers to see what are the results that way I think would be a big help, a lot of times you're just trying different methods for different sub-components and then try to record what was good and bad about different bits, and you ditch an awful lot of it in the end, [literally ?] what goes into the final paper is just a description of the final model.

M: The last one for this first part: Do you think there is any substantial difference between ecological modelling and modelling in other areas, in other domains?

H: I don't have any experience in order domains (M: What is your feeling?) I wouldn't think so, I'd think most things have very similar modelling problems and the modelling structures are already used. I don't know, on the other hand, I think that an ecologist who would read a book on engineering problems, much of them wouldn't be very helpful, but I guess as a programmer you would be constructing very similar modelling building tools for the two of them, I mean ModelMaker gives you examples of chemical models, ecological models, sort of science models there were built as a modelling course. So I guess the answer is to build them it's the same.

—— Demo and trial ——————————————————

M: What do you think about the way models are defined in this tool? I mean, the approach, is it clear confusing?

H: I think the approach is clear, yes. It's fine, I think as I've said on the questionnaire, it took me a bit it takes a little [time ?] it's actually necessary to sit down and do it, then I have to learn the terminology that have been used and I'm not a population dynamics modeller generally so I had to remind myself, and then put all in it as well, lots of different ways of expressing the same thing. Clear, but I need a little time to get used to it, I think.

M: Do you think this approach is similar to what modellers actually do in real life?

H: Yes, yes.

M: Do you think that with proper support or improvements in the system, it could help a novice to learn how to devise a conceptual model, or how to start the design process?

H: I think it would be really good, yes.

M: Do you think the questions asked are sensible questions, or they could be better?

H: They're essential questions really, aren't they? They're all questions about the essential parameters that you use to define your system.

M: About the model you have defined, do you think the results corresponded (more or less) to what you expected? (H: er..) Do you had any expectations when you build a model?

H: Well, that all depends on how complex the model is, if is something simple like this then you have some expectation that comes from how you do certain things, constant mortality and increasing population, I mean, it looks as the mortality isn't more than you like it to increase then you expect the population to keep on going up. But with the mode I worked with, is sufficiently complex, there are lots of things that it does that I couldn't predict, I mean, the individual components are predictable but when you put them all together the system's work isn't, which is the whole point of building a model, if you understand the components then you understand how they interact, but there are just too many things going on and too many calculations to do yourself in your head, and too many things that could interact, it needs a computer to do all the calculations. So although we predicted the general behaviour of the model I built, there are lots of specific behaviours and specific situations which were on it that I wouldn't predict.

M: What, in your opinion could be said is the main drawback on this system, and for that matter, the main advantage of it?

H: I guess at moment probably it does I'd say it does need more [correspond ?] more help on-line, probably a manual or whatever. I think if you're sitting there it would take a little while to get through these stages and, you know, figure out what should be filled in, what is to be collected, what it is meaning by a particular phase, and yes, I think that's the main drawback it just needs some help in there.

M: Which sort of assistance you think it could be useful to get from the tool?

H: Really is two things. I probably need a sort of refreshing course in population dynamics, I mean, all right, if you get population dynamics ecologists who are very familiar with them, no problem, but on the other hand if there were terms that could be used which you could click on them and say "oh, this is the population's whatever", great! Similarly if you're using terms which are specific to this model, so that's some way that you can simply find out what that means in the context of this model. So that's extra help, you could click on terms in order to get explanation.

M: Which sort of user do you think could benefit from this tool, this sort of tool, actually, not only this one? Do you think novice would benefit from it (basically it's different from ModelMaker, for example, where you've got a white board)?

H: Well, is it really that different from ModelMaker? Because you can design in ModelMaker instead of with the graphical diagrammatic interface, you can design everything step-by-step into it, it asks you questions rather than to you know, you just fill in a diagram. So I'm not sure it is that different really from something like Stella or ModelMaker. I guess if someone ask you questions is probably easier to come to use novices because they just go through answering questions as you ask them an they will end up with a model. So I guess [?] it's better. But I can't see how to could you extend this to something more complex, as it goes more complex it would be harder to just keeping asking more and more questions and making sure you got all the right answers. (M: And of course, you have to give some feedback for the user, some way to show the structure of the model) Yes, at that point it would be easier just to leave people to drawn this by their own.

M: Do you think the user could benefit from seeing the inference mechanisms behind the tool, like the KB rules, etc.?

H: Yes, I guess so. I mean, you have to know that in order to answer the questions [?] in this particular case you just been talked what you knew already. (M: Of course, I'm not talking about showing the rules in Prolog, but through some graphical interface saying "OK, if you get this and that sort of interaction, then it's a predator-prey relationship") Yes, what you need is information specific to the rule you've just done, says, in this particular [one ?] what happened is something like there is very hard predation, therefore [?] and discussing it to interpret what comes out in the end of the day. If was just something tell you the information it was already fed in you've fed in it already. (M: But of course, this is not a static thing, it could be dynamic, the user, through some interface, could extend that KB, for example, saying "How about if I've got a configuration of competition between two species, or things like that") Yes, that's true, if there were the relationships being set-up and being feed in. (M: And processes as well, because I have some

pre-defined processes, of course they aren't all of possible processes for computing natality. the user could say "how about if I use this equation" and leave that equation available on the KB for others, forming some sort of Data-base of processes and ways of computing, even guidance strategies. I've had this suggestion - someone told me "How about if we have a data-base of examples the users could go through") Yes, that's could be useful. But I guess what you need out a computer, is if you fed in certain rules, but by doing that there is an interaction set-up, you wouldn't necessarily be aware of it. That's the sort of thing that could be really useful for the computer to handle on it in order to highlight it to you. And for example, if you said you have 3 layers for instance, 3 [?] levels, if you had an invertebrate say, in there, and there was an interaction between invertebrates and mice or some sort of rule saying "animals eats the invertebrates" and a further interaction between the predator and the mice. So if you change the interaction between the predator and the mice, that would not affect the invertebrates, but you may see something in the results, but it's happening because there is an interaction that you can't see looking at the results. You might change something at the top predator and there is some results on the bottom level involving the invertebrates. (M: but you couldn't see the causality between them) yes, as you change the predator you see something happening, but you don't necessary know why that's happening, it's happening because something is changing in this [?] in the middle. So it would be quite useful, I suppose, for the modeller if it could say "OK, you change this, this is the effect that will have on this and that's will influence in all that".

M: That's it. Have got anything to add, maybe some comment? Suggestions? You've made some good suggestions already.

H: Good, I hope I can buy it off the shelves later.

Participant 12 (Agric2)

M: Let me just explain how I intend to do this... I'll just ask you some questions about [your] background and stuff and you have the things then we'll do the demo... after that I'll give you a short questionnaire - it's a standard usability questionnaire, there are ratings, very simple. Of course, as I told you, the software is not finished yet - it's a prototype and what I'm trying to do is check if what I got now it's in the right direction... well, let's see.

M: First thing, could you describe your background. I see you have some sort - a good background in computing and how about ecology?

J: I don't know much about ecology, my first degree was electronics, and after that I was working in industry for a while and I decided I preferred in academia so I did a masters in KBS and I've done a PhD in cognitive science. After that I started work as a RA and my first project was a modelling beliefs of agents during dialogue and the PhD was also something more or less about that but this is sort of AI based stuff... and at the end of that it didn't seem to be any opportunities to get further jobs there we have some difficulties with the funding and I fancied the change anyway, so I moved into the IERM and started to brought in this system, so my only ecological background is in biology A-level, and that is quite a long way, so I picked up quite a lot of stuff... and basically do modelling systems is more a problem of programming to the designs actually going about the systems then you end up modelling, I mean, it's up to the person who uses the system too.

M: But you have been exposed to a lot of ecology stuff, I mean,... people [etc.]

J: Indeed, yes. I've been there for conferences, etc... a lot of jokes.

M: You just told me about your background in electronics and everything. Have you had any formal ... you ever though how to do modelling? How to model?

J: Er... only in the context of demonstrating AME, so, I've helped out demonstrating AME to students and there are a couple of master students that using it to ... I'm er 2nd supervisor to these students.

M: Yes, but I mean... how about YOU - have you any formal training in modelling, or...?

J: No people training [?], of course, Bob has spent quite a lot of time telling me about what modellers do, but er basically since the one of the purposes of AME, is to enable people to do modelling in a top-down

or bottom-up style, because has a possibility to create all these sub-models. We are basically making the run-in [?] in determining how to do modelling rather than - sort of - try to learn from other people how to do it. We are providing OO tools to do it in a new way.

M: In your experience, you thing ecological modelling is anyhow different from modelling in other area?

J: Er... YES, certainly. A lot of... most modelling which is done in the world it's of man made systems, there is models of queueing systems, traffic systems and of industrial processes all of which I have done, and ecological modelling is more complex because the system you're trying to model is not something that you, basically, have designed yourself or know about and therefore the model needs a lot of constructs that are unique to modelling natural systems. For instance, the population model that we just looked at probably wouldn't be to [?] any other projects.

M: Do your think that makes it more complex or only different?

J: I think more complex, for having [?] over other systems ... I would say some [born and born] [?] are potentially more complex

M: What is your opinion: when people start to build a model when they start to modelling, do they always go from a conceptual model to implementation or sometimes they mix it and they try to... "OK, let me try to idealise this stuff, and then jump - try the implementation a little bit and then having some feedback from the results and THEN start to..." Do you think they are, actually - in real life, they are separated stages, always, or sometimes no... What's your [opinion]?

J: I think just as programmers have different techniques, then modellers, given the choice, will probably have different techniques as well.

M: Yourself, do you always devise first and implement later, or sometimes you like to try...

J: No, I tend to build things up as I go along, rather than plan them out first.

M: To have some feedback.

J: Yes [nod].

M: You are working with AME now. Which sort of computer stuff are you using? I know you are using SICStus and TCL and Windows and... Are you using any other simulation software like - to get references - like Stella, Flomo.

J: I had had a look, a very brief look, at the other simulation packages in the field, but I wouldn't consider myself to be an expert in any of them.

M: No, I mean: in order to develop your own stuff you have looked in the different ones and you have compared them. Isn't it?

J: Mmmmm, only Flomo really.

M: Another thing. Have you had any tutoring or teaching experience?

J: Not very much, I thought at school for while, but...

M: But you told me you were giving some training in AME for students,

J: Well yes, I somehow demonstrating to [?] the labs, and also there are some students using it so I have individual sessions with, ... but I have [?] actually done lecturing on AME

M: Is it difficult to talk to novice about modelling? Do they get the grasp of it, or

J: Usually they do, users can pick up what is going on ... visualisation is important for that

M: That's good. I think that's about - about the first part (background). Thank you.

—— Demo and trial ——————————————————

M: What was your overall view of the tool? Not only the implementation as it is now, but the idea - do

you think this is worth, or...

J: It's probably an idea of get people to be able to make a wide range of models and just start people thinking about of what's important in modelling... and I needed to sort to play with it for longer and to try and do more different things, by trying to sort of take unusual concepts and see how they work... how close you can get to represent them in that tool and what sort of compromise you have to make

M: About the way things are done, you thing it was clear, it was easy to follow, I mean, the concepts

J: It is easy to follow, yes. The problem I had was that I felt always that I was into one particular process and I couldn't go back and try to do something else, sort of navigate around the system very easily, it seems to be quite happy as long as you go along with the process, but not happy otherwise.

M: Do you thing those questions and those menus - they are reasonable for a novice? You see, the people I'm ... the audience I'm intending to are only novice people who have no idea how to start modelling.

J: Probably then a bit more background is needed like some people may don't know about the concept of age classes... and when entering equations, you have the available variables appear to one side - so you could specify the variables but it was [wasn't] possible because at that point you weren't looking at the meanings of each one of those variables... so often you have to infer those from the names of the variables

M: Do you think it was very constrained? too constrained, maybe?

J: It seemed fairy [very[?]] constrained, there are sort of hints that other things might be possible, you also had position disaggregation class in there so I would like to see what you were able to do with those

M: I have to say it was intended to be a bit constrained err, you think... in that way, it could be a good thing for novices? I mean, to say: "OK you have not such a great experience on how to do that... how about try to follow these steps?" You think that makes sense? Or maybe there are a different approach to do that?

J: I think obviously - probably, you want to do something very simple first, something that use only one parameter, and then see that working and then extended, rather than try to create a system with several parameters at once.

M: I see. You think that there should be some way of show the inference mechanisms behind the machine, like the KB, the Prolog state of the code, because what is there now is just a log of something just for my use, not for the user, you think it could be useful for the user to have these information, or it would only add to the complexity?

J: I don't think the user would want to see the information at the level of Prolog program... maybe some kind of graphical representation of what was effecting what, that they... I mean, if they can start with a very simple model and they can sort of... work backwards themselves to recognise what the inference mechanism is.

M: And how about facilities like clicking in some variable of the model and some graphical representation or something says: "OK, these facts [?] are represented in the model here and here and there".

J: Yes, because I wasn't sure about what - how I would going about to getting my graph because there... when we created a model there were two populations in it ... and the results came up those [?] only in one graph. Admittedly if there had been a graph for the fox it wouldn't have done anything interesting because there were only two of them, but it seemed like if I wanted to see some other value in the model, there no way to do it.

M: Which other sort of assistance or help could be useful for a user to get from there? Is that a reasonable thing to think about?

J: Which other sort of assistance?

M: Yes, like help facility, like graphical assistance for something or...

J: Perhaps more working examples, because you... basically have a system allowing sort of taking the user step by step construction or reconstruction of something, but the actual input that they have to offer at some point, that should be examples of what kind of entry groups [?] that way.

M: I think mostly that's it. Is there anything you would like to add? comments? suggestions?

J: Mmmm, just that was quite strange to use a system that isn't graphic- based like that is.

Participant 13 (IERM5)

M: Could you tell me something about your background? experience?

J: Yes, sure. OK, so, in my PhD I worked a bit on modelling, way back, a long time ago that was in late 60's, so in those days I used FORTRAN models, I used [?] for plant physiology [?] and try to make models of growth, water mass and CO2 uptaking. And now I'm working on different things, so I now work on CO2 uptaking by rainforest and so, er, we go into [?] field with equipment which ran on towers in the forest, we expose our instruments above the forest then we can measure the uptaking of CO2 in a day and the release in the night by respiration. And so this leads to all kind of modelling approaches because one collected data which can be model in a very simple way or one can use any of the models that are in the literature right now and that are sometimes used by the global planet modelling community. So I don't myself do much modelling these days, that has to be said, I much prefer to work with people who do the modelling. So the modelling I do myself when I have to do it, will usually be in Basic or sometimes I might be using spreadsheet models, and I avoid serious modelling, I prefer to work with the people who do the modelling. And I also do some modelling with students, so with such things as mmm, for example tomorrow we work on... trying to make a simple of the global carbon balance, and the relationship between CO2 emissions by humans and the temperature of the Earth, so that quite simple models can be used to represent the basic processes. This is more or less it...

M: I think... I guess you use computers a lot, which sort of computer software are you using now?

J: I use computers a lot yes. I use computers for writing for a [?] for e- mailing, (M: spreadsheets) spreadsheets, yes, I tend to use Excell spreadsheets, Word 6, Excell 5, and Corel Draw a use as well.

M: You are still active on teaching and tutoring, maybe?

J: Yes, quite considerably yes. So... mainly in... not in modelling, but in environmental biology, of course I did a bit of teaching in the course on statistical methods [?]

M: When you learned about modelling, were you taught formally, I mean, have you attended some discipline or something about modelling?

J: No, in my case, when I started, nobody else was doing this sort of thing, so this was way back, and, well, there were groups in UK doing computer modelling of this sort of growth process in eco-systems and so I guess I'm more or less self-taught. What I do I obviously read books, talked to other modellers... Oh, there is another project which I'm involved which is modelling tree growth, it is an European project in which my group were contributing submodels really, submodels to deal with water use, water transport in water use, so I reasonably in touch with the modelling community.

M: Yourself, or these people you have contact with, which sort of approach, you use when modelling, I mean, system dynamics, or tools like ModelMaker, Stella...

J: Some people use ModelMaker, and some people use Stella... Most people write their programs in FORTRAN, Basic, Pascal, C...

M: Would you say MOST of the people use to build their own programs?

J: Yes, people love to write their own programs. And most people discovered that programming - as long as you have a time - programming can be very satisfying, so people actually like to program. I think that people you'll find in my field do a lot of programming are people that program in all kind of languages, [definitely ?].

M: When you have to do modelling, or teach other people how to do modelling, you first devise a complete conceptual model or you tend to have some insight into the model and then try to implement it and get some feedback and then devise a bit further?

J: Yes, most usually when work with compartment flow paradigm, so modellers usually in things to do with eco-systems and forests, they can [work?] themselves to an idea of carbon flows on water flows or mineral nutrients flows from soil... well, between the soil, the plant and the atmosphere. So, for example, in my field we talk about the soil-vegetation- atmosphere transfer scheme, so we have a pretty, rather broader, idea of what we mean by the transport [part of it ?] from the soil to the atmosphere, and it is partly a physical process and partly a biological process.

M: So you got already some templates of the main...

J: Yes, I think it's true to say that... templates... you can call them templates.

M: In your experience, would you say there is any substantial difference between ecological modelling and modelling in other areas, like, I don't know, business maybe industrial processes?

J: I don't think there is very much difference. I noticed that in business people are using neuron networks sometimes, that hasn't really got into ecological modelling very much, and [?] modelling people often have a [fairly ?] idea what the underline approaches are, they tend to think of process which have been studied under controlled conditions, I mean, on the laboratory, where the physical or chemical processes is reasonable well understood, so it is often a matter of taken a model which comes from someone else's work and then it becomes a submodel in the model of the eco-system or of the forest.

M: What you think is the most difficult thing about modelling, or about ecological modelling? What you feel yourself?

J: Is certainly that eco-systems are all a bit different, when we have a model, the difficult thing is to parameterise it, because it is just too big a task for most people to go out and measure all the parameters, and that leads to a kind of schism between those modellers who represent a few processes and have a small number of parameters and the others modellers who like to have rich detail but unfortunately they don't have a way to find the parameters values. So just as modelling in business, take [a way] of several human physiologists then humans are more or less the same, but if you are studying eco-systems, there is such a huge diversity of them, and the parameters are just elusive.

M: So you think the complexity add more to it...

J: I think the complexity, yes. The high variance between different eco- systems.

—— Demo and trial ————————————————————

M: What do you think about the way models are defined in this tool? The approach to modelling, is it clear, well-defined, easy to follow, or...

J: Yes, it is easy to follow, but because it doesn't... I'm not sure how easy this will cope with spatial [retrogenality ?] for example, which may be very important in population dynamics. But the general approach is very easy to follow. Perhaps you couldn't make any easier. Perhaps the fact that you have to make it very easy means that it have to be extremely simple (M: constrained) [?] too simple for practical applications but good for learning about modelling.

M: Do you think this approach is similar to something modellers do actually do in real life... go through stages or go through standard points on the design process?

J: I didn't understand the question.

M: The way you go through steps, step-by-step here, these steps are they similar to what modellers actually go through at some stage if they have some similar models.

J: Yes, I'm pretty sure. You have to first of all, define the system, define the components, define the interactions between the components and then [?] functions to describe the interactions.

M: It is just because I'm not an ecologist, nor a biologist...

J: I suspect it is standard, it isn't. It would seems to me to be necessary to go thorough more or less the steps you have...

M: What I've done there was try to apply the way I've seen modelling in programming and another domains

to ecology... OK, Do you think somehow this tool, or this sort of tool, not this only, but of this sort, could help a novice user to devise a conceptual model? Or to learn how to devise a conceptual model?

J: Yes, I think it could. I think someone might eventually get frustrated when they want to get into more complex situations, but as a way to a kind of [achieve lift of ?] then this would be fine.

M: About this model, this shape of population that you have produced there by your description, you think that corresponded to your expectation of the model? Did you have any expectation by what you've defined there?

J: I didn't... No, I didn't really think of it in much detail, because I was too busy trying get on screen but I suppose, well retrospectively, one would have expected some sort of saw-teeth, something like that - I didn't think ahead because I was too busy looking at the screen.

M: But you normally, when devising a model, you have some expectation?

J: Yes, well, normally you working in a field that you are familiar with, you learned the behaviour of the system empirically, through experience before you start of, and then you compare the model results with what you think what it should be. Because if there is a major discrepancy there, it makes you think that must be something wrong with the model and then you go back to the model - it's a first true test when you making a model, is isn't?

M: Again about the tool, what are main drawbacks and advantages?

J: The main advantage is that you get a beginner started and the main drawback is the limitations, in many areas, the limitations with [???] and also I'm not sure how well this would work outside the [?] population dynamics, I'm not sure how well it look like for [?] for say, global carbon modelling, I'm not without looking into this, I'm not sure whether the same approach would transfer to another [real ... ?] enquiring, maybe it would.

M: But you think novices would benefit from this sort of tool?

J: Yes, I think it would be interesting trying out in a class of students.

M: Have you seen this sort of approach in any other modelling tools you have experience with? I mean, can you place it among others?

J: Yes, I think of it as being a bit like this software that Robert Muetzelfeldt has been putting together called AME and [?] models like Stella which, which... well those case are compartment flow models but they do, at least, have the environment for a complete novice from they can make a start... but I like the idea that the way it asks you for information, so you have to respond and then it puts it together to make a progress... very [nervous ?]

M: Do you think the user could somehow benefit from seeing the reasoning mechanisms, the inference mechanisms behind the tool, like the KB, to see how the... what are the rules, like "you could have a predator- prey relationship in this and that or that situation"? Of course not directly in Prolog code, because Prolog code is just nothing for a novice, but through some graphical interface, do you the user would benefit from that?

J: Yes, I think [?] the user concentrate on processes much more rigorously [?] they were working with a model like this.

M: What sort of assistance would you like to see in tools like this one?

J: What sort of assistance, what sort of extra-help, and things? (M: as a user, you have just tried). Yes, OK. I think it would be nice to have rather like you got on this example, it would be nice to have some worked examples which you could run through a tutorial and then [?] every time [?] it co .. part of the sitting up... for help you you wanted to, it could prompt you. Maybe a tutorial, a selection of tutorials going from very simple to quite complicated would be useful.

M: You mentioned about the processes. Do you think there are other... I mean, using the same approach, do you think that with different process instantiations it could applied in different domains.

J: I think it could, yes. I think probably when [ ] you want to go to somebody else in a completely different

domain, different perspective than population dynamics and see whether... you could see how it could applied, I think it translates to any domain, really. Perhaps also the commercial, industrial domain and if this were for instance some series [?] processing from, let's say Kodak, then this would be a [concept you ?] go on in all the factories and how one factory [feeds ?] on the other, this could be something which would [be meaningful ?] to this type of programming. I think it does transfer across domains, but maybe the way you set it up its at the moment very constrained, if you have said to me try [?] a model in it without give me any example then I had tried something else, I don't know whether it would work. I think I might.

M: Can you think of any other features in tools like this one - besides the assistance, like graphical support...

J: Graphical support it's really important getting graphs that you can printout afterwards, you know, with several variables, when you have modelled [?] say you have 6 animals you would like to plot out of six a at time.

M: And see again the, as in your case, the functional response that...

J: Yes, that's a very nice feature, that's a good feature that I haven't see before.

M: You would like to be able to edit that and to print out...

J: Yes, sort of [?]. You might want to also have [?] to make small functions sometimes.

M: Have you got anything to add, maybe some comment, suggestion?

J: I think the important think is that it would be good to try it out in actual total novices, specially on students. And perhaps that could be arranged, perhaps if you talk to Bob, he could try it out on complete novices.

M: My first idea was go straightaway for the novices, but now I think the best idea was just to get the experts first and to listen all... because I got lots the feedback and ideas, after those ideas be implemented, to give it to students/novices. Things like the help you said, and the examples, was a good idea, to have a bank, or a data-base of examples which the use can select...

J: Yes, if you wanted to help you by [?] on a class of students, Robert could probably do it, but there will be this silly problem with the software and eXceed stuff...

M: Oh yes, that was other suggestion I got, because I'm using versions of...

Participant 14 (IERM6)

M: Can you tell me something about your background, your experience?

J: Mmmm, I did a BSc in ecological science here, and so I did some modelling with Robert Muetzelfeldt, and then I did about 10 or 15 weeks immediately after I graduated just building up a, well, basically a system dynamics expert system, er, maybe that is why he mentioned me. And then I've gone to Micromed something measuring plots [?] and choose the atmosphere and I also for a year I was working on modelling [modelling ?] trees and [?] ... and how ozone affects [?]

M: Would you say you have a mathematical background, some way?

J: Not very mathematical, I haven't done much maths... I am probably more comfortable with maths than most biologists, but that isn't really being very mathematical.

M: Do you use maths in your work, normally?

J: I use some...

M: equations?

J: Yes, equations... not much [?]

M: have you got any teaching or tutoring experience?

J: A bit of demonstrating... I've demonstrated for computer practicals - that sort of thing, and so [MicroMath ?] practicals.

M: I suppose you are quite familiar with computers, have you used them a lot?

J: Yes, yes, quite familiar.

M: Which sort of computer tools you use at the moment?

J: Er, I do some programming in Borland Pascal, I use Excell, word processors, MathCad, and I use statistical program SAS - I suppose are the [?] things I use.

M: But you have done some programming yourself?

J: Yes, I still do quite a bit of Pascal programming, and I did do some Prolog programming.

M: Were you ever been thought modelling as a formal course, discipline or something like that?

J: Well, as part of the BSc in ecological science, Robert Muetzelfeldt er thought [?] resource ecology, which was basically computing and modelling, a lot of modelling.

M: Was that probably constrained to ecological domain, to ecology?

J: Yes, it was.

M: But do you think is there any basic difference, basic difference, between modelling in ecology and modelling in other areas like business, industrial processes, or something like that... traffic, maybe?

J: Mmm, I think there will be some differences between different domains in the modelling and [?] ecological modelling any model will be [amongst ?] others far more complicated than modelling traffic, traffic can be quite complicated, but because you have to keep to the roads, etc. you don't have anything like as many variables as you have in an ecological system.

M: Do you always devise a complete conceptual model before start to implement, or you have some idea, some insight and then try to implement it, get some feedback and go back to devise again?

J: I usually have some idea, put some on the model, see what happens, so its usually a development cycle.

M: What do you possibly, is the most difficult part in modelling? Or in working with modelling and understanding modelling in ecology?

J: I suppose the more difficult thing is what to include and what not include in the model.

M: Yes, the "abstraction"...

J: Yes, the abstraction, yes.

M: Maybe you have mentioned before... but do you still use any sort of general purpose modelling tool like - systems dynamics in Stella, ModelMaker or anything like that?

J: No, no.

M: OK, that's it for this first part

—— Demo and trial ——————————————————

M: What do you think about the way models are defined on this tool? The way the modelling is done, is it clear, easy to follow?

J: Yes, it's quite clear, for predator-prey populations type. You have to ... for different modelling domains you would have to change things, but

M: Do you think this approach is similar to anything you have done or other modellers do normally... going through this sort of (J: step-by-step) yes, step-by-step design decision-like?

J: Well, most modelling tools I have seen were really based in systems dynamics, in drawing the diagrams

- while this is more structured, it takes you through steps which I think is good.

M: Do you think somehow this tool, or tools like this one could help a novice modeller to get the grasp of how to do a conceptual model, at least some conceptual model?

J: Yes, yes.

M: Because normally they have no experience on how to start.

J: No, and if novices have to start from programming, and that's a huge problem, they would learn programming instead of the modelling, if they're using systems dynamics diagramming tools, they will have to remember which steps to go through, I think is quite... it's very interesting in fact that it leads people thorough. The steps you have to go thorough are quite well defined, there has to be some flexibility but basically the steps are the same, so you could [play ?] programming may as well, be default.

M: You think the questions asked... they make sense, they are good questions, I mean, the step about organisation... they are good landmarks for the design?

J: Yes.

M: About the model that you have defined, do you think the model somehow corresponded to the specification you gave? Normally, when you do modelling, you have expectations of how the population, how the results would be?

J: Yes, you usually have an idea because you know something about the system, so you know how you would expect it to react, but that's not always your expectations aren't always right, which is why modelling is really useful because you found it that something didn't go quite as you expected so you have to go through the model to find out what it is.

M: What in your opinion, are the main drawbacks and, if is there any, the main advantages of this sort of tool.

J: The advantages are that it leads you thorough the modelling process and I guess the disadvantages are that to make it a very flexible modelling tool means that programming has to be very good, you would need a huge knowledge base, but if you team something of like the structure of this with a huge knowledge-base then you'd have an extremely good tool for at least well, definitely for teaching purposes, but also for research purposes. Lots of it depends on how good the knowledge base is.

M: So you ... the sort of users you think would benefit from this tool would be novices, it would be students for example, learning how to do modelling?

J: Yes.

M: How would you place this system among others you have seen? OK, as you told me, systems dynamics is not a good point to compare, but ... How would you place the system among the tools you've got available for modelling?

J: Of my immediately use... my main modelling is in a different domain, so you can't really compare...

M: Here's another one for you: Do you think the user would somehow benefit from a greater access to the reasoning mechanisms, to the knowledge base, even through some nice graphical interface, because, of course, pure Prolog code is not (J: readable)...

J: Even if you knew Prolog it would be very difficult. But yes, specially for the reasoning system... you want to know why it may came to these results, so I think it could be very useful.

M: Can you think of any other sort of assistance the system could give to a user? What difficulty have you found most annoying when using the system?

J: I don't know, because I've been using it for such a short while, and it's a prototype, then it's difficult really to evaluate it. It seems a good approach and I think with more work and aiming some of the... implementing some of the things that you're going to implement [?], then I think it will be an useful tool.

M: I think mostly is it. Do you have anything to add... comments, suggestions?

J: I can't really think of anything.

Participant 15 (IERM7)

M: Could you tell me something about your background, your experience?

S: In modelling, particularly?

M: No, no, in general.

S: Er, well I'm a biologist by training, my first degree was natural sciences at Cambridge University, so I studied animal and plant biology in my final year I specialised in [?] and I was [?] mainly in ecology and physiology, so whole plants issues rather than [?] biochemistry of that type. I did my PhD here in Edinburgh in forestry department on... with wind effects on trees and then I worked at Reading University on a project to deal with [?] and soil it was looking at the way that roots system grow and whether [varying ?] differences in root systems could improve the growth [?] in dry areas and that was the first time I did any modelling really. And after this I went to the Institute of Hydrology and I have been, I was there for 8 years, so I just came recently back to Edinburgh. And my work is mainly on vegetation [what is?] measuring and modelling the amount of water used by different types of vegetation linking into [?] models.

M: Would you say you got a strong mathematical background?

S: I think, well it's all relative, is isn't? I think I'm a reasonable numerate person but I mean, I'm not capable of doing, you know, complicated integrations [?] I don't have a very good grasp of calculus, I basically I understand what it is, but I don't... I'm not very good at doing, you know, algebra and differential equations and things like that.

M: But you use maths in your work?

S: Yes, I use maths, I use things like... a lot of times I'm making models of [?] mainly relatively simple systems of equations and fitting into data by using some non-linear optimisation technique, multiple regression, that kind of thing. So I'm comfortable with that level of mathematics.

M: Have you got any teaching or tutoring experience?

S: Er, not in a formal sense, when I did my PhD here I did a lot of demonstrating in practical [?] and on field trips, ecological field trips, and that was back in the early 80's and since then I've mainly worked in the research environment, but I haven't really done any teaching. I've helped to supervise one or two PhD students.

M: I see you are very familiar with computers, which sort of software are you using... in your daily work?

S: Well, as regarding to say operating systems, to start of, I've used obviously DOS and now Windows, Windows95 and also familiar with UNIX and Xwindows and also worked in other things like on IBM mainframes, CMS, VM-CMS and M-VS operating systems.

M: And how about [?]

S: Other software, then I use [?] most [?] manipulating data and modelling I've used SAS, a package called SAS, I don't know if you know (M: mmhumm, statistical) yes, it originates from America, it's a very large package which can do all kinds of things and you know, re-formatting, manipulating data, plotting graphs and doing a whole range - a full range of statistical tests and it can do optimisation stuff, that kind of thing. But I use nowadays things like standard Microsoft Office tools for word processing I use sigma-plot for doing publication quality graphics.

M: Do you use any modelling tool, like Stella or ModelMaker?

S: I'm very interesting in modelling tools and what you can do with them and I want to use them but... a couple of years ago I bought a copy of ModelMaker and I did the tutorials and everything and I wanted to get started with it but... it wasn't on my sort of critical path of work, you know, I had other on more - had higher priority and so I never really got to use it frankly. I can also program, I mean, I can program in FORTRAN and Pascal and the SAS package has its own programming language inside it so I comfortable

with procedural programming not - I've never done any object oriented type programming, so, oh yeah I was gonna say... I'm used to look at models other people have written in a language like FORTRAN or whatever and I know that you can in many cases achieve the same thing much more quickly in a much more flexible way by using a modelling tool and that's what interest me, it seems a lot the scientists there in a very sort of backwards mindset they just keep on using FORTRAN because that's what they always used and if they [took the trolley ?] to learn how to use these new modelling tools that, I think, they could do it a lot more quickly.

M: Something which just strike me... have you had any formal training in modelling, or have you learned by yourself?

S: No, I haven't made any [?] had any formal training, no, I've just... I've picked up from books and papers [?] software and I go to conferences as well, sometimes agricultural conferences or [ecological ?] conferences where modelling have been discussed. As I mentioned before, I collaborated with modellers, I worked for a couple of years with a guy in US who was doing GCM modelling and I was helping him to improve his model of the [flocks ?] of water and heat between the African land surface and the atmosphere, because I've done quite a few measurements there. So, I think I've got quite a good idea of what modellers do and what the issues are, but I'm not a fully trained mathematician or modeller... Neither are probably 90 percent of people doing modelling.

M: When you have to devise a model, you first devise a complete conceptual model and then implement it or you get some insight of the model, try to implement it and have some feedback and then refine it again?

S: Er, I think is definitely an interactive process, yes. You try to put something together and then you see how it would test against real data, real measurements of the system and then if doesn't work the way you expected or you hoped to, then maybe try to change it.

M: What, in your opinion is the main difficulty, or the less obvious thing about modelling? What you think normally people think is most difficult about modelling.

S: Do you mean the most difficult about the process? (M: yes, the design process) er... well, yes, there is a problem of implementing your ideas you know, if you, say, you examine theoretical ideas in the area you want to model, and you might be able to have an idea of the model you can put together, but if it involves some intractable equations or you know, something like that, could be a problem of you know how to program or whatever to get it to work... that's one problem. But I think there are other problems with modelling, most of the modelling [?] work in a sort of deterministic type modelling, you know, there is no uncertainty attached to any of the parameters, but in nature there is lots of variability, lots of uncertainty and while they are different in errors or different variability in different parts of systems why they add up to produce variability in any point of the system, that's very important but rarely considered in modelling, and there is also issues around the reliability of the data as well, you know, like input data and drive data, [?] errors in it with variabilities and they are usually taken into account, another issue is of scale as well, often measurements are made of rather small scales, you know, in field plots or in the lab or whatever, [?] try to model is an eco-system on the region of the outer surface or something and there is all kinds of problems around the validity of scaling from small systems to large ones, and there's a similar... there's a temporal issue as well, usually you data is only available over a limited time period, but nowadays in environmental sciences, we are often interested what is going to happen in a long term, and [?]. So, when I as working on the African climate, helping this [chap after ?] climate the main issue there was about the draught which has been going on the [?] since the late 70's, but you know, the main trends want to decrease rainfall but still [?] of variability and if you want to calibrate the land surface scheme during some [?] model, there is lots of data about how vegetation, soil [?] that's what control [?], but you know, your measurements need to cover the whole span of typical growing season conditions, so you just have you [?] for two years or something, you know, maybe you would have to two wet years, and you never get conditions on the draught year. I guess of your question, that's is not really to do it, that [what ?] of modellers to do it is the problem of you know, make sure that you test and parameterise the model properly.

M: From this... not exactly difficulty, but these idiosyncrasies maybe, of ecological modelling, do you think there is any substantial difference between ecological modelling and modelling in other domains, like business modelling, or industrial processes?

S: Well, I have little or no first-hand knowledge about those domains, but I think there is an essential difference from modelling physical systems in that, I mean, if you like, man-made physical systems, because

there is much less variability, OK, and there's fewer factors to be taken into account, the problem with ecological modelling is [?] already, the variability every point in the system but also the fact that most of the systems are fabulously complex. So, you can't possible model every interaction that you are already aware of in the system, because it would make the model too complicated and there are probably interactions going on that are immeasurable or you don't know about. So, all the time you're simplifying reality, you just... in a very intuitive way, picking out the interactions that you think are important or may be more important for the particular purpose of your model... but you know, might be relatively unimportant to some other purpose. So, there's... it must be a major problem in validating ecological models as well, like, I see your example here has to do with population model, well, if you create a population model, you know, how can you possibly test it? Probably you got to get, ideally, data on the population of different kinds of organisms and doing that in natural systems may be extremely difficult. ... Your parameters about mortality and rates of reproduction that you need to make the model in the first place, I mean (M: they have to be obtained somehow), yes, they have to be obtained but I mean, they are incredibly difficult to measure, aren't they? specially if you are working, I don't know, any environment like a forest, you working in a tropical forest and the organisms you're interested are the ones who only live up in the canopy, you know, things like that. I once spoke with someone who was interested in Wales, you know, it's very difficult to know what the whale population is, or the demography, you can't really do marking and recapture experiments on them.

M: And of course, you have to have some observation over some time period, some significant time period. It could be very difficult.

S: Yes.

M: Something like 100 years... what happens on the rainforest in Amazon...

S: Yes, that's what I was saying about the temporal scaling problem. And you mentioned also business-type modelling, well, that I think that must be similar to ecological modelling in its complexity, because it involves the behaviour of human beings and that is so difficult to predict, because it depends on so many factors, and then human beings are very non-uniform in their attitude, their education and their motivation.

—— Demo and trial ——————————————————

S: I just thinking you may have a problem when your number of animal increases beyond 2, because presumably you... some are predators and some are carnivores they can... you can define a process for one preying on another, can't you? So, as your number of animals goes up, you number of possible interactions goes up, doesn't it? and you can define those all in different ways using different equations if you want to, so you could end up with a very complicated model, and as it is at the moment, there is no graphical representation isn't there? (M: No) So, whereas you might like to have say a diagram with 10 animals, 10 boxes on it and you can have arrows that show that that kind of animal preys on that [other] kind animal and you can click on the arrow and inside there it could give you the processes by which that [link ?] incorporates, and then it would be much easier to find your way around the model, you know, if you want to change the [platinum ?] predation on a particular ...

M: And of course, when you have more than two, or even with two, only two, you can have quite complex models for example, if I got fox which only adult foxes prey on platypus, let's suppose they got some preferred classes or categories of platypus, for example, the old ones of the younger ones, it could be...

S: Yes, I agree yes. It could become quite complicated.

M: Of course, you can represent that, but it just too difficult to imagine it.

S: Another thing is that at the moment what you've shown me the only graphical output is the trend in the number of individuals through time, but as we were looking at that output, we were saying to ourselves "oh, I wonder why... the population crashed just there", so you might want to be able to generate other graphs that show you the [?] of other variables through time, let's say we might... [?] be nice we could have a graph of the foxes maximum consumption rate through time, to look at in conjunction with the population graph, so we could test the hypothesis that it was the foxes were unable to eat any platypuses at that point, so to actually have a usable tool where people can set up the model and see how it behaves and then understand why it behave the way it did and then modify the model again. I think you need to build some more capability into it.

M: What do you think about the way models are defined on this tool, I mean, the approach, is it clear,

confusing, this step-by-step...

S: Yes, step-by-step, yes, that's ... The one thing I don't like about it is the way some of the pre-defined variable names, you know, you had one with cte instead of constant, but then when you open up a window where the age was stratified, you got things like... you got the name of [?] name platypus and then it says dollar ... it just... it does make you feel as [?] you are looking through a window on to a computer language, you know... There is no need to have a dollar there, from the user's point of view, the dollar, they don't know what the dollar sign means, and it doesn't actually have to be there to make the modelling intelligible, does it? Presumably is there because it's a requirement of the syntax of Prolog. So, you not....

M: Actually, the intention there was just to give some indication that that specific variable was disaggregated somehow in that dimension, but of course, that could be done in different ways (S: Yes). Do you think this approach... I'm sorry, you were saying anything else about it - the variables, of course.

M: Do you thing the way models are defined here is somehow similar to what modellers actually do, go through these stages on the program and try to define them, or it somehow different from anything else?

S: Well, I guess that what they do for this class of model, population predator/prey type of model. I'm not working in that field but I mean, it seems to me to be a reasonable approach, but I mean, the problem with this tool it is [?] you to quite a small domain isn't it? (M: right, yes) So, and... whereas I could imagine it might be really nice for use in teaching for instance, you try to get people who haven't done no modelling, maybe undergraduates or people to understand how to do modelling, and also to begin to understand interactions between animal populations. I think it would be very good for that, because I think people could learn very quickly and be happy about modify, you know, see how things work and so trial-and-error kind of. I think that would be good. But if were saying research scientists, I think it might find a framework rather constraining, I'm sure that might well be things that you want to do and you couldn't really achieve in that framework. And other things, I noticed that you specify your own equations, don't you? (M: yes) but I personally don't know what syntax is it expected by Prolog, I don't know what operators Prolog uses, for instance, or I don't know if it got any precedence rules if you start having complicated equations, you know, brackets and that kind of thing. So, without some extra information I wouldn't be able to write any equation.

M: About the design questions which are asked, do you think they're reasonable, they are asked in a sensible way? I remember you made at least one observation about the hierarchy of vertebrates/invertebrates, but how about the other ones, you think they are asked in a sensible way, because they could have different order in different sectors, maybe more specific questions?

S: Yes, from what I can tell they are asked in a reasonable way.

M: You said it could support somehow, novice modellers, do you think that could help them to devise a conceptual model? Because in my opinion the thing with conceptual modelisation, or conceptual modelling, is that you should be helped by some experience, and when you have no experience to compare to...

S: Yes, modelling... models come from experience, they come from knowledge of a particular part of the world, so I don't think you would ever use a tool to generate conceptual model, that's the process in reverse, you know, you think about a system, you construct a conceptual model in your head, and then you want, you originally want to implement on a computer, you want to see whether your model will actually reproduce the behaviour of the real system, because if it does then you say to yourself "my model is probably a reasonable one, and my understanding of interactions in this system is probably more or less correct".

M: Of course this is a silly question, because the system crashed, but what do you think about the main drawbacks and if is there any advantages of this sort tool, not only this one, but this category, this sort of tools. (S: Any advantages of it) yes, and drawbacks.

S: Well, the advantages are that tools like that are easy to learn, OK. They should be fast to work with and try to implement a model in 3rd generation language or whatever, and also I guess it should be less [?] to you make errors, because various errors are trapped, you know, when you input something into field and because you working with a framework that has been [?] for you. You can easily make errors when you writing a FORTRAN program because there are infinite numbers of ways you can write it, but you know, you can't... that's not true on this kind of tool. There should be the advantages. The disadvantages are the inflexibility, the fact that you never be able to think of things you would like to do with the tool, and when you want to have the capability and you won't be able to understanding how the tool is put together,

that's always the disadvantages of modelling systems.

M: Do you think the user could benefit from having greater access to the inference mechanisms, to the KB, of course, through some graphical mapping, graphical representation, because Prolog code would be too crude to see, but through some graphical representation?

S: The thing is don't believe... I don't quite understanding what you mean by the Knowledge base, I'm not sure whether it would be useful or not.

M: Things like the sort-hierarchy of the components which are pre-defined, or maybe the rules of inference saying "if you I got this and that ... you could have a predator/prey relationship here", some pre-defined processes, for which are given just some crude information, maybe saying "this process would be on when there is any sort of disaggregation in this population" and things like that.

S: Yes, that might be useful, yes. Because you mean that in the case of the disaggregation that you choose, if you select disaggregation and then modifies the subsequent behaviour of the tool, doesn't it? (M: yes). So yes, it would be good for the user to understand that making that selection causes the behaviour to change and in what way it changes.

M: Which are the features that you would like to see in this tool? You have mentioned some. Would you like to have some assistance on how to operate it and graphical feedback...

S: I think the graphical feedback and the ability to look at any of the variables in the model graphically is very important to understand WHY it behaves the way it does. And that's [?] what you want to do [in your own view ?]. If you have a conceptual model of the way the system works, and then you implement any real model and you find that its behaviour is different from what you expected, then you need to understand (M: to trace that) yes, you need to trace WHY is different and it's in that way that you revise your conceptual model. So, I would give that a hard priority. I mean, you mentioned Stella and things like that, earlier on, I know from when I was doing the ModelMaker tutorial, [?] you can look at any variable, you can select and you can see how it behaves through the duration of the simulation, and that's very valuable. And that's one way that tools really score over say, programming in FORTRAN, because it's so difficult to implement graphics, and you can achieve things so much more quickly if you can, very rapidly, graph any variable in the system.

M: I think that's it for the second part, would you like to add anything, comment, suggestions - you have made some suggestions, very good ones.

S: I think it's quite good fun to... I wouldn't mind to try it out with other problems ... [tape ends]