Simulation models: the computer programs

Pedro J. Aphalo

Departamento de Ecología, Facultad de Agronomía
Universidad de Buenos Aires
1417 Buenos Aires, Argentina

Abstract. Concepts and criteria of computer science and program engineering are analyzed in relation to the programming of simulation models. Programs are considered to be useful for the communication and accumulation of knowledge and goals are stated taking this into account. Techniques such as structured programming and literate programming are discussed in relation to the goal of writing comprehensible and modifiable programs. Criteria for the acceptance of program listings for publication are also proposed.

Key-words: simulation, models, computer programming.

Introduction

Models exist independently of their implementation as computer programs, but as their complexity increases, the use of computers becomes unavoidable during their validation and later use as prediction tools. Programming1 can be defined as coding the relations or the algorithm that defines our model in such a way that they, or it, can be read by a virtual machine (computer + translator program) that then becomes an embodiment of our model (a new virtual machine). Programming is usually done with the aid of programs called translators that translate the 'source code' written in a, more or less, human understandable language such as BASIC, FORTRAN or Pascal to machine executable 'object code'. The programming of complex simulation models is not a trivial task, it accounts for an important part of the time spent in the development of such a model.

1 We give a restricted meaning to programming as we consider the analysis and, at least in part, the design phases to be part of the modeling process.

The process of modeling can be considered to be the process of imposing structure on knowledge (Overton 1977) and so making this same structure explicit in computer programs is very important for their comprehension. Moreover, because of the great similarity of the strategies used for ecosystem modeling (Overton 1977) and computer programing (e. g. Dijkstra 1972) methods and tools that can be used to attain this goal are widely available. We shall discuss the criteria used by professional programmers and computer scientists (e.g. Wirth 1973, 1976; Bentley and Knuth 1986; Takara 1985; Zimmer 1985; Hoare 1986) in relation to our more restricted problem.

Our Goals

1. We must be able to prove the correctness of the program, i.e. that it corresponds to our model.

2. The program must be readable. Then we must define who will be the readers: at first the programmer, may be ourselves, but as we are scientists, one of our very important goals is communicating the knowledge obtained, then we must hope other scientists will eventually read our programs and understand them, and even test and modify them.

3. The program must be easy to modify. Models are hypotheses, and in science we seldom reject them altogether, we generally modify them to account for the new knowledge acquired and then start new experiments to test them again.

4. The program must be as portable as possible. If we expect our programs to be used and tested by other scientists it will be very convenient if they can be run on different computers with only minor modifications.

5. The program must be only as efficient as is needed. We must not sacrifice our other more important goals if high efficiency is not an absolute necessity. In the future efficiency will be even less important than it is nowadays as the cost of computing equipment is decreasing year after year.

The techniques we can use

To attain our first three goals our programs must be structured. We shall give only a brief explanation of what structured programming is. There are many good books and papers that fully explain and justify this (Dijkstra 1972; Wirth 1973, 1976, 1985; Weiland 1983) and other related techniques (Weinman, 1985; Zimmer 1985).

In brief the justification for the use of these techniques is the limitation that we have, as human beings, to grasp big complex problems as a whole. We must divide them, starting at the higher level of perception (the top of top-down design), into black-boxes each of which we then further divide at the next lower level, until we solve our problem. Each of these boxes is complete and small enough to be understandable at its level of abstraction. Each box must have a clearly defined input, output and action to perform, so that as long as these subroutines or procedures behave as black boxes what happens inside them is irrelevant to the rest of the program, and because of this any of them can be replaced with a functional equivalent very easily. To prove the correctness of our program we must prove the correctness of each of the procedures individually and their connections, but this is much easier than to tackle the whole program all at once. Moreover, if we have proven the correctness of a program and then replace a procedure, we must only prove the correctness of this procedure and its connections to prove the correctness of the whole revised program.

However structured programming does not insure readability. Knuth has proposed the concept of literate programming (Bentley and Knuth 1986; Bentley, Knuth and McIlroy 1986). The philosophy of literate programming is very relevant as it is centered in the use of all available resources (writing style, layout, typestyle, size, diagrams, etc.) to facilitate the understanding of program listings by readers. The idea is to write the program in a mixture of a programming language and English, and with fewer restrictions in the order in which different parts of the programs are put down. This author has created the WEB system that through different preprocessors creates a compilable source code, and a formatted listing.

How can we choose the appropriate programming language?

It is possible to write clear and understandable programs in almost any language, some of them being easier to use than others for certain kinds of tasks. However, as Kreutzer (1986) states 'there are intimate relationships between thought and language and language and methodology'. Languages are not only notations to communicate with computers, but also cognitive devices that structure perception, representation and reasoning about classes of problems (Kreutzer 1986). As we believe that writing structured programs is a must except for very simple models, we prefer to use languages that ease this task by providing suitable constructs. As our experience is limited to continuous simulation models we shall refer to them in what follows. If we want to duplicate the structure of models in computer programs we will usually need to define nested subroutines, and this is not allowed in some widely used and powerful but unsuitable languages such as Fortran-77 and C. Readable programs must have meaningful identifiers (constant, variable, procedure and function names), and to be meaningful they must usually be long, so the language standard must guarantee long identifiers. The language must be in widespread use among the expected audience, and of course provide all the needed

operators, functions and data types. We prefer to use Pascal, and think that Modula-2 or Ada could be even better for some uses such as programming big and complex models, or when several programmers are working simultaneously on the same project. But as long as one can attain the goals stated, the programming language to choose is that which the programmer finds easier to work with for the particular problem at hand. Even spreadsheet programs can be very useful in some situations.
In the near future we expect two developments to happen that would alter our preference for Pascal for coding continuous simulation models. Efficient implementations of structured simulation languages such as GEST (Ören 1984) or CCSL81 (Crosbie 1984) could become available, and libraries of simulation targeted modules could be developed for ADA and Modula-2. Simula could be used but no implementations are available on personal computers, and we think this to be an important restriction. Smalltalk and other object oriented languages are potentially useful (Kreutzer 1986), and lead to an intuitive view of models.

The publication of program listings

We think that non commercial programs referred to in scientific papers should be available in machine readable form, or at least as printed listings, from the publisher of the journal or a centralized users group or similar organization for a nominal fee. This kind of code publishing has been adopted by some computer related magazines and journals.

Conclusion

Considering programs useful for the communication of knowledge makes programming in many senses similar to writing scientific reports. Even when programs are mere tools for getting the published results, if they are not standard implementations of common algorithms, they must be subjected to analysis by the interested readers. So we hope that in some future time all relevant source code will be published along with articles and only after having been reviewed for its "human readability". This should give more credibility to the simulation models we build and to some extent set up the usually sterile arguments held between some biologists and mathematicians who work in biological modeling (Hall and DeAngelis 1985).

DANMARKS FARMACEUTISKE HØJSKOLE
KEMISK INSTITUT AD
Universitetsparken 2, 2100 København Ø
Tlf. 01 - 37 08 50

ROYAL DANISH SCHOOL OF PHARMACY
DEPT. OF PHARMACEUTICAL CHEMISTRY AD
2 Universitetsparken, DK-2100 Copenhagen
Denmark

Date July 13, 1987
SEJ/msh

Dr. Pedro J. Aphalo
Dept. de Ecologia
Facultad de Agronomia UBA
Av. San Martin 4453
1417 Buenos Aires
Argentina

Dear Dr. Aphalo:

Thank you for your paper. ISEM-J. is replaced by a News Letter, and Ecological Modelling is the society journal. However, your paper is nicely fitted to "Environmental Software", to which journal I have sent it. I hope it is OK. The address is:

Dr. Paolo Zannetti
Environmental Software
Aero Vironment Inc.
825 Myrtle Avenue
Monrovia
CA 91016-3424
U.S.A.

Yours sincerely,

[illegible]

S.E. Jørgensen

Un 05-46