Published in: Möller, D.P.F. and Richter, O.(eds.), Proc.of the 4th Ebernburger Working Conference on the Analysis of Dynamic Systems in Medicine, Biology, and Ecology, April 5-7, 1990, Ebernburg, Bad Münster am Stein-Ebernburg, BRD, Informatik-Fachberichte 275, Springer, Berlin a.o., 1991, p. 131-145.

Interactive Modeling and Simulation of Environmental Systems on Workstations

Andreas Fischlin[1]

Abstract

Many systems dealt with in environmental sciences such as ecology or environmental biology could be easily modelled and efficiently simulated on personal computers or on workstations. Thanks to their graphical capabilities such computers make it possible to model systems interactively, e.g. supported by graphical structure editors, or allow for interactive simulation featuring sophisticated graphical output of the simulation results. However, in practice this potential remains often underexploited, since traditional, simulation software is mostly batch oriented, largely ignores computer science research, and offers rarely the functionality needed for a sensible interactive use. Instead of porting simulation software from main-frames onto workstations we propose new concepts based on Wymore and Zeigler's modeling theory, enhanced by some new interactive user oriented task concepts. This paper presents a scheme called RAMSES for the architecture of a modeling and simulation environment on a workstation particularly suited for the working with environmental systems. Furthermore it reports on some results which have been obtained by implementing portions of the RAMSES architecture, in particular an open and extensible modeling and simulation environment for the two classical model formalisms SM (Sequential Machine), DESS (Differential Equation System Specification) featuring modular modeling. Finally the modeling and simulation of a system from population ecology is presented as an example to illustrate and evaluate some of the concepts of RAMSES in ecological research.

1 Introduction

The demands for interactive modeling and simulation of environmental systems are old (HOLLING, 1964; DAVIDSON & CLYMER, 1966), whereas the possibility to satisfy them are rather new. Interactive modeling and simulation have never been the strength of the traditional main-frame simulation software, but today, with the wide-spread usage of personal computers and recently the more powerful workstations, the possibility to bring the computational power of the main-frames together with the interactivity and user-friendliness of workstations has become economical and is indeed most attractive.

However, straightforward realizations seem difficult, since an application of existing modeling and simulation techniques has first to overcome several obstacles: First the majority of simulation software has an architecture which is mainly batch oriented (CELLIER, 1975, 1979, 1982, 1984a; ANONYMOUS, 1988); secondly it largely ignores current computer science research (KREUTZER, 1986); and thirdly, maybe most importantly, aside from very few exceptions the software is not founded on a mathematically sound basis, i.e. it ignores the modeling and simulation theory developed in the last decades (ZEIGLER, 1976; WYMORE, 1984).

Interactivity is particularly attractive in the modeling of so-called ill-defined systems (INNIS, 1972; CELLIER & FISCHLIN, 1982; FISCHLIN & ULRICH, 1987). Typical for them is that essential portions of the mathematical properties of the studied system are poorly understood or even un-

[1] Current address: Systems Ecology Group, Institute of Terrestrial Ecology, Department of Environmental Sciences, Swiss Federal Institue of Technology Zürich (ETHZ), ETH-Zentrum, CH-8092 Zürich, Switzerland.

known. This is particularly true for environmental systems, which KARPLUS (1976) places in the middle of his model spectrum calling them grey-boxes. Compared with the black boxes on the one end of the spectrum their modeling appears attractive; on the other hand they lack the simplicity and clarity of the white boxes on the other end. A full list of advantages of interactive modeling in the area of environmental systems has been formulated elsewhere (FISCHLIN & ULRICH, 1987).

The attractive interactive modeling and simulation are, the much it poses difficulties, since it requires the development of dedicated simulation software. Interactive programs have a structure radically different from that of batch-oriented software (NIEVERGELT & WEYDERT, 1980; NIEVERGELT & VENTURA, 1984; FISCHLIN, 1986). The situation is furthermore complicated by the fact, that most simulation studies tend to develop very large computational demands, but sensible interactivity requires that response times remain within certain limits. Hence the design of interactive simulation software must cater to the two conflicting goals of batch and interactive simulations at once.

After having analyzed programming languages and programming styles in the context of simulation and studying current simulation techniques in much detail, KREUTZER (1986) concludes that existing simulation software largely ignores current computer science research. For instance the majority of simulation software has been and still is written with out-dated programming languages poorly fit for their purpose. In a recently published catalog (ANONYMOUS, 1988) from 191 world-wide listed simulation software packages 79% use Fortran, the rest either C or at best some object oriented extensions of C. In the bibliography of textbooks on simulation by KREUTZER (1986) 83% from the programming oriented books use Fortran or simulation languages which are Fortran precompilers. Precompilers separate the modeler only partially from the underlying implementation language. Often either the simulation language reflects the spirit and concepts of the used implementation language, e.g. naming rules for identifiers, or it forces the modeler to use it sooner or later, in cases he/she wishes to program an unavailable algorithm. CELLIER (1979; 1984b) claims that mainly due to the use of the programming language Fortran, the simulation software packages lack robustness. He argues that in order to increase the quality and reliability of the simulation software, one should use formally defined programming languages of type LL(1) to optimally support structured programming (BROOKS, 1979; WIRTH, 1985, 1986).

On workstations interactive modeling and simulation concepts must be simple, user-oriented, and have to support modern graphical user interfaces including items such as menus (pull-down or pop-up), windows, scrolling of window contents, and selection or dragging of graphical objects etc. The latter requires adequate programming support such as structured data types, dynamic memory allocation respectively deallocation (heap-technique), recursion etc. (GUTKNECHT, 1983; FISCHLIN, 1986; WIRTH, 1986). Some of the few programming languages which satisfy the requirement of formal definition and support a good programming practice are the procedural languages Modula-2 (WIRTH, 1985, 1986) and Ada, or Oberon as an example for a formally defined, object oriented language (WIRTH, 1988, 1989a, b).

Finally aside from very few rare cases (ÖREN, 1984; VANCSO, 1990) existing simulation software ignores the whole body of systems and modeling theory which has been developed during the last two decades (KLIR, 1979; ZEIGLER, 1976, 1979, 1984; WYMORE, 1984). Important for the modeling of environmental systems are the levels 1 (IORO) and 3 (System). On level 1 a system is defined by a time basis, the sets of the inputs, outputs, and input segments, plus the I/O relation relating outputs with input segments. On level 3 a system is similarly but in more detail defined by the additional set internal states, the state transition function mapping the product of inputs and internal states to internal states, and the output function mapping internal states to outputs. Furthermore the standard formalisms DESS (Differential Equation System Specification), DEVS (Discrete Event System Specification), and SQM (Sequential Machine) have been formulated to support frequently used classes of mathematical models. For a full summary see VANCSO *et al.* (1987).

From all this follows, that in order to meet todays requirements and to tap the potential of tomorrows computer technology, a new modeling and simulation software has to be developed. There-

fore we have dared to begin with the development of an interactive modeling and simulation software called RAMSES[1] particularly designed for modern personal computers and workstations. Our approach is user-oriented, i.e. a conceptual frame-work derived from research activities, to support interactive modeling and simulation. The software should be particularly well suited for the modeling of ill-defined dynamic systems (INNIS, 1972; CELLIER & FISCHLIN, 1982) as often present in the environmental sciences (KARPLUS, 1976;). For the implementation we chose Modula-2 as the programming language and based the software on the *Dialog Machine* (FISCHLIN, 1986). The RAMSES software architecture is based on systems and modeling theory and it supports an object oriented working with such concepts. This paper focuses on RAMSES' session concept.

2 Interactive Modeling and Simulation with RAMSES

RAMSES provides software support for the activities typically followed by a user who constructs and simulates models of ill-defined systems by grouping them into the four sessions (Fig. 1):

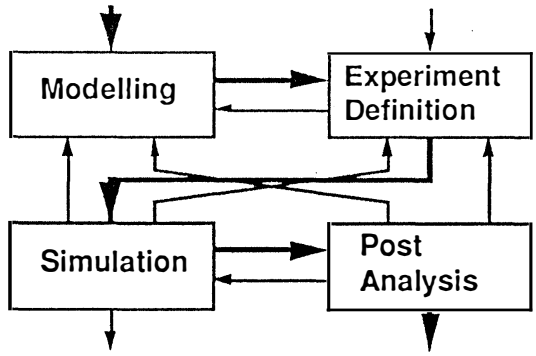

Fig. 1: State transition diagram depicting RAMSES' four task oriented sessions (states) and the possible user movements (transitions). 1) Modeling session resulting in the declaration of models, model objects and the formulation of model equations. 2) Experiment definition which consists of a definition of an experimental frame plus its association with a particular model definition. 3) Simulation session which produces model behavior. 4) Interactive postanalysis of previously computed simulation results.

- Modeling session: This activity serves the declaration of models, model objects, and the formulation of model equations. A mathematical model (not to be confounded with a simulation model) defines a certain mathematical structure but not necessarily a particular time domain, parameter and initial value sets.

- Experiment definition session: It consists of a specification of an experimental frame (ZEIGLER, 1976, 1979) plus its association with a particular mathematical model. Its result is at least an experimental frame or an experiment. The latter fully specifies a simulation model (not to be confounded with a mathematical model) which incorporates in addition to a mathematical structure also a particular time domain, parameter and initial value sets etc.

- Simulation session: Given a particular experiment has been defined and a simulation model exists, the simulation session is used to produce model behavior in time or space or both. Results can be saved for an analysis at a later point in time.

- Postanalysis session: Simulation results previously computed during a simulation session, can be analyzed interactively without having to recompute any model behavior.

Fig. 1 shows a state transition diagram of the meaningful and legal transitions among the RAMSES sessions. Note that iterative model development cycles are supported by various paths.

---

[1] Acronym for Research Aids for the Modelling and Simulation of Environmental Systems

For the user's convenience, sessions can be interrupted and resumed later in the same state they have been left any time. RAMSES has two user interfaces: The end-user interface and the client or programmer's interface. With only very few, but then intentionally introduced exceptions, any function offered by RAMSES can as well be executed via the end-user interface interactively or via the client interface by writing a program.

2.1 The Modeling Session

The modeling session serves the declaration and installation of models, their model objects (Fig 2), and the formulation of model equations.

A RAMSES model definition consists of at least one model and every model usually contains model objects such as state variables, expressions, model parameters, output and input variables for submodel coupling, and auxiliary variables. Every model object can also be declared as a monitorable variable. Not the output variables, which are reserved for the coupling of submodels, but only monitorable variables allow the user to display simulation results. In this respect RAMSES differs from the modeling theory by ZEIGLER (1976) or WYMORE (1984). RAMSES requires to associate with model objects certain real values such as derivatives respectively new values, parameters, initial, minimum, or maximum values. The RAMSES interfaces were built such that mandatory values must always be provided while declaring a model object (Fig. 3). For the easier recognition and identification of model objects by the end-user there are also the optional attributes: long textual description, short identifier, and the unit string (Fig 2) .

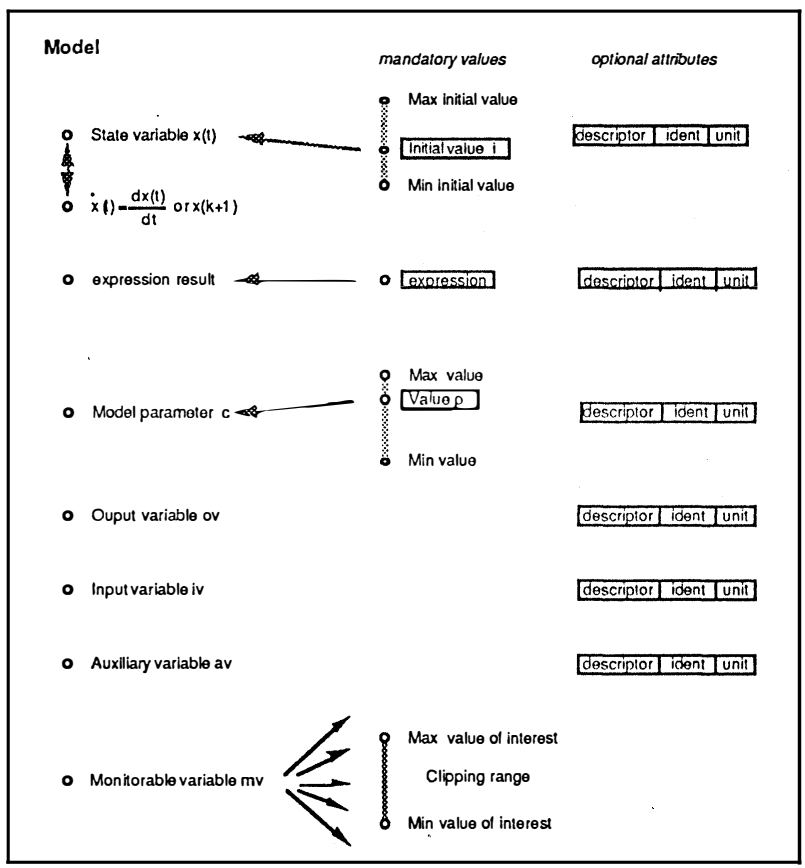

Fig. 2: A RAMSES model definition serves the installation of models and model objects (O): Model objects are state variables plus their derivatives resp. new values, model parameters, expressions, auxiliary, monitorable, input, and output variables.

The purpose of the model and model object declarations is to make their associated real variables known to RAMSES. During simulations RAMSES will then maintain these values: E.g. it uses

the derivatives respectively new values (in case of discrete time difference equations) to compute and repeatedly update the values of the state variables (numerical integration). Otherwise RAMSES ignores these objects and their values. This offers the modeler the potential to use them freely, e.g. in structured data types, in a manner which is rather problem adapted than forced by the idiosyncrasies of the used simulation technique.

```
PROCEDURE DeclareSV   (m: Model; VAR s, ds: REAL; initial, minRange, maxRange: REAL;
                       descriptor, identifier, unit: ARRAY OF CHAR);
PROCEDURE RetrieveSV  (m: Model; VAR s: REAL; VAR defaultInit, minCurInit, maxCurInit: REAL;
                      VAR descriptor, identifier, unit: ARRAY OF CHAR);
PROCEDURE ModifySV    (m: Model; VAR s: REAL; defaultInit, minCurInit, maxCurInit: REAL;
                       descriptor, identifier, unit: ARRAY OF CHAR);
PROCEDURE UndeclareSV (m: Model; VAR s: REAL);

PROCEDURE GetSV       (m: Model; VAR s: REAL; VAR curInit: REAL);
PROCEDURE SetSV       (m: Model; VAR s: REAL;     curInit: REAL);
```

Fig. 3: Excerpt from the client interface of RAMSES showing procedure declarations (they consist in Modula-2 of a heading only). The listed procedures provide all RAMSES functions needed to work with state variables.

There are three basic techniques by which modeling can be done within the RAMSES modeling session: 1) Via the client interface using the host programming language enriched with particular objects needed for modeling and simulation; 2) via an interactive end-user interface accessing individually by entry-forms the functions exported by the client interface in order to add (declare), modify, and remove (undeclare) system theoretical objects (model and model objects) from the model data base. 3) Via the Editing of a graphical representation, i.e. relational digraphs (Fig. 4), of a model system to support a more abstract view. Subsequently relations are specified by functions, i.e. declared as expressions, in a manner which resembles that used by the second technique.

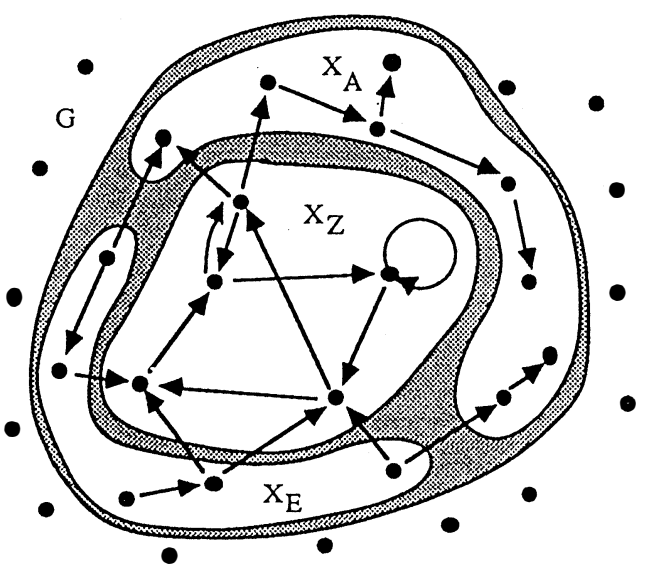

Fig. 4: Relational graph of a general system S represented as a 9-tuple of sets $S = (X_I, X_Z, X_O, R_I, R_{IZ}, R_Z, R_{ZO}, R_O, R_{IO})$. The universe of the system X is partitioned into three sets: Set of state variables X_Z, inputs X_I, and outputs X_O. Where: $X_Z = \{z \mid z \text{ from } z \text{ reachable }\}$, $X_I = \{e \mid e \notin X_Z \wedge (\exists z: z \in X_Z \wedge z \text{ from } e \text{ reachable})\}$ $X_O = \{a \mid a \notin X_Z \wedge [(\exists z: z \in X_Z \wedge a \text{ from } z \text{ reachable}) \vee (\exists e: e \in X_I \wedge a \text{ from } e \text{ reachable})]\}$. Form from ordered pairs within the system universe the following six structure sets: input structure $R_I = \{(e_1,e_2) \mid e_1,e_2 \in X_I \wedge (e_1,e_2) \in R_o\}$, input-state-coupling structure $R_{IZ} = \{(e,z) \mid e \in X_I \wedge z \in X_Z \wedge (e,z) \in R_o\}$, dynamic structure $R_Z = \{(z_1,z_2) \mid z_1,z_2 \in X_Z \wedge (z_1,z_2) \in R_o\}$, state-output-coupling structure $R_{ZO} = \{(z,a) \mid z \in X_Z, a \in X_O \wedge (z,a) \in R_o\}$, output structure $R_O = \{(a_1,a_2) \mid a_1,a_2 \in X_O \wedge (a_1,a_2) \in R_o\}$, and input-output-coupling structure $R_{IO} = \{(e,a) \mid e \in X_I \wedge a \in X_O \wedge (e,a) \in R_o\}$. For the unifications X_T resp. R_T must hold $X_T = X_I \cup X_Z \cup X_O \neq \emptyset$ and $R_T = R_I \cup R_{IZ} \cup R_Z \cup R_{ZO} \cup R_O \cup R_{IO} \neq \emptyset$.

The first technique is the most versatile one with the least restrictions; it even supports non-classical formalisma like cellular automata or recursive model equations etc. However it is less convenient, since it requires programming. The third is most attractive, since it allows to support hierarchical model structure editing: a node representing a subsystem can be collapsed or expanded into a separate window containing again the tuple of the subsystem or the supersystem.

Independent of the three ways the user chooses to work with, RAMSES applies always the same basic technique to manage models and model objects. They are installed or instantiated by calling declaration procedures, which allocate a memory block in the heap to store the object together with its associated values plus attributes. However, the actual objects like state variables or parameters remain fully in the scope of the user model definition. If the modeler uses the first modeling technique via the client interface, this scope corresponds exactly to the scope concept in Modula-2. For instance state variables may be part of any data structure and may be used in any type of statement sequences such as e.g. recursion etc.. Moreover, models and model objects can be removed (undeclared) any time from the model and model object base or be edited in any way. All these functions are realized according to the same principle. Fig. 3 shows an excerpt from the client interface for all procedures needed to manage the model objects of the type *State Variables*.

2.2 The Experiment Definition Session

It consists of the specification of an experimental frame (ZEIGLER, 1976, 1979) plus its association with a particular mathematical model. What results is a simulation model which contains no longer any missing concrete values necessary to fully specify e.g.an initial value problem. Furthermore a time domain or spatial domain for which the model behavior is of interest, and any other needed parameters such as integration method, maximum local error (absolute and relative) etc., have also been defined.

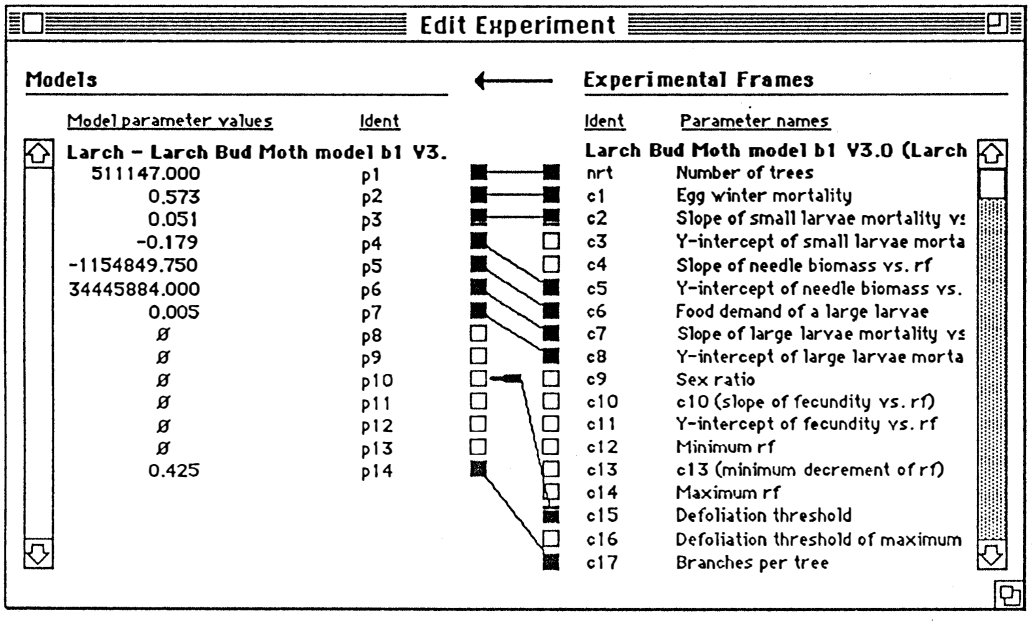

Fig. 5: Editing an experiment by combining a mathematical model with an experimental frame during an experiment definition session. A set of parameters defined during the modeling session have partly no defined values (ø). Specific values with a concrete meaning are contained in the experimental frame. The user can now assign them to the parameters of the mathematical equations via connecting items from the experimental frame with parameters in the mathematical model similar to establishing a plug connection. The direction of the top arrow can be toggled and determines the direction of the assignment.

This session is user oriented, since the thinking in terms of an experimental frame or a particular mathematical model is much easier than to think always in terms of the ever exploding numbers of combinations of the two (Fig. 5). For instance is it much easier to keep track of a number of data

sets, each with its particular time domain, and the number of alternative models which could at least hypothetically be applied. If mathematical models can be freely combined with experimental frames, a technique which has been proposed e.g. by ÖREN (1982), the definition and combination of two members of these classes are easier to manage.

There remains the issue of compatibility between a mathematical model and an experimental frame to be resolved. In RAMSES a mathematical model and an experimental frame are called compatible if their order is the same and the dimensions of the model parameter, input, plus output vectors are the same. RAMSES accepts an experiment or a simulation model for a simulation session only if the combination of model and experimental frame are fully compatible (s.a. Fig. 1).

2.2 The Simulation Session

The purpose of a simulation session is to produce model behavior in time or space or both by solving a simulation model over a particular domain of the independent variables, normally time. This is called an elementary simulation run. A complex simulation experiment formed from several elementary runs is called a structured simulation run. RAMSES allows either to directly execute an elementary or a structured simulation run, each an arbitrary number of times (Fig. 6: *k*, *n*).

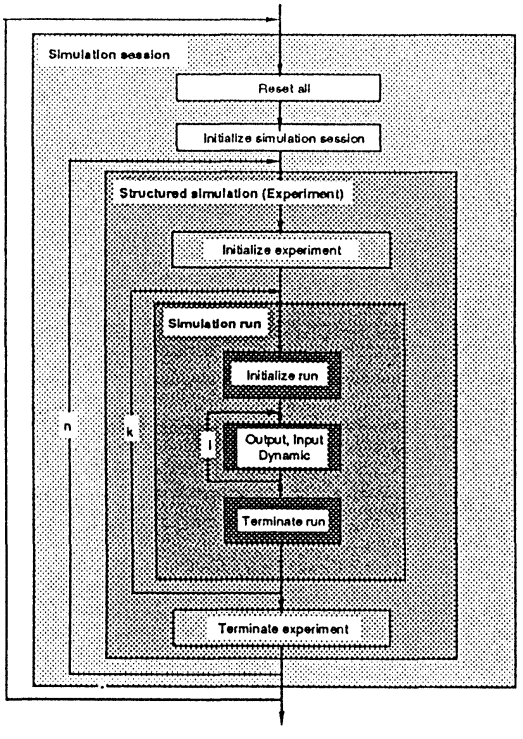

Fig. 6: Structure and flow chart of a RAMSES simulation session. The simulationist may execute directly an arbitrary number *n* of structured or of elementary simulation runs. A structured simulation («experiment») consists of a programmed number *k* of elementary simulation runs. Every simulation run consists of an *Initialize run*, dynamic (includes the sections *Output*, *Input*, plus *Dynamic* s.str.), and *Terminate run*. The dynamic section is executed according to the chosen time step and simulation time an arbitrary number of times *i*. Dark grey shaded areas: mandatory, i.e. must be defined by every model definition program; light grey shaded areas: optional and under full control of the modeler.

RAMSES automatically assigns the initial value *i* to the state variable *x* at the begin of every simulation run, and the value *p* is assigned to the model parameter *c* at the beginning of the simulation session or after any interactive change (Fig. 2). RAMSES maintains also the current values of state variables, parameters, and monitorable variables and remembers their initially specified values for eventual restoration (so-called resetting). During simulation experiments the unknown values, which the monitorable variable *mv* may obtain, are written on the stash file, tabulated in a table, or displayed in graphs (Fig. 7). The latter is only the case if the values fit within a particular range of interest as specified by the modeler; otherwise they will be clipped.

Often at the begin and at the end of an elementary simulation run particular actions must be taken, e.g. to initialize states or compute and record final results. The simulation environment of RAM-

SES provides facilities to install such procedures (Fig. 6: *Initialize run, Terminate run*). Furthermore in structured simulations, which typically execute several elementary runs, there is the possibility to initialize and terminate the whole experiment. This may be particularly useful in the case of a stochastic model, where e.g. means, variances and other data collections ought to be calculated from many runs (Fig. 6: *Initialize experiment, Terminate experiment*). Finally the whole simulation session may also require an initialization; this is provided in the simulation environment by allowing for the installation of an initialization procedure.

A basic purpose of interactive simulation is to allow for the easy monitoring of the simulation results by focussing on particular sensitive or otherwise interesting results in order to be able to temporarily halt or even interrupt the simulation. This is most useful in early stages of model development, where the user may want to abort a particular run quickly, once its main characteristics have become apparent.

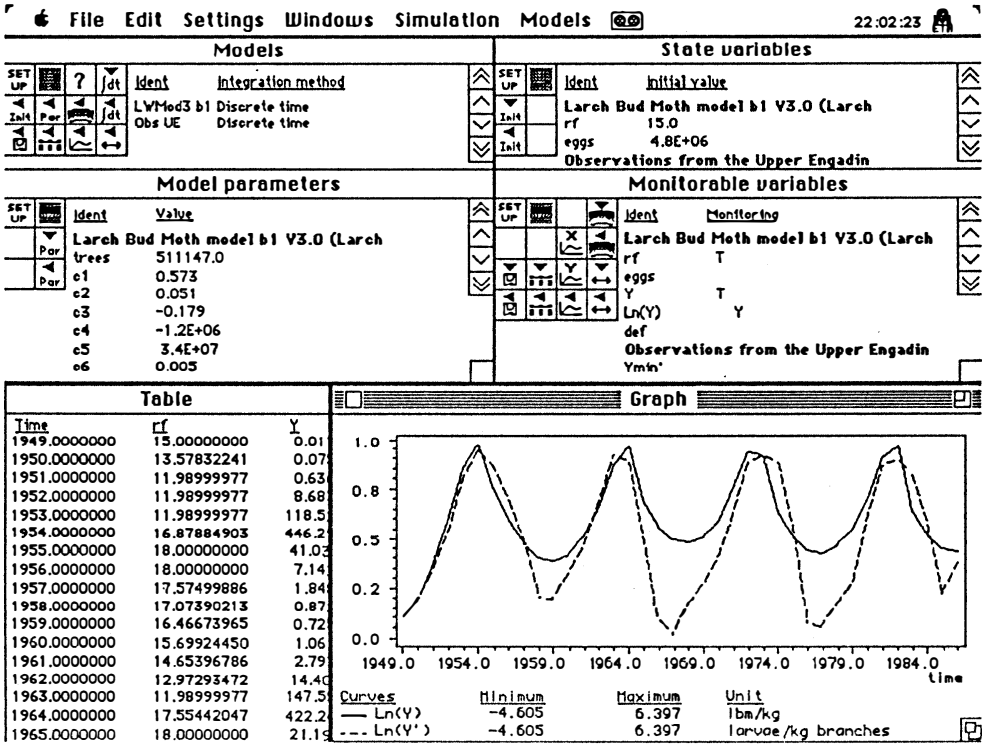

Fig. 7: Typical screen of a RAMSES simulation session. The model shown is the larch bud moth system described in the text, in the graph its simulated behavior is compared with observations.

For user convenience the simulation session of RAMSES allows also to change currently used parameter values, scaling values for monitorable variables, initial values etc. This avoids having to quit completely the simulation session for a run just testing a particular parameter value. This can be considered as working in the simulation session just with a scratch copy of the whole model and its associated values, whereby the values specified within the experiment definition session can any time be resumed by a reset. Note that the reverse is also possible, i.e. they can be copied to the model definition and be stored as an experimental frame. The arising need for selection of models or model objects, e.g. to change the value of a specific parameter, is provided by the so-called IO-windows. Fig. 7 shows a typical screen of a RAMSES simulation session with the menu bar plus its menus, four IO-windows, the table, and the graph window.

Since the user's monitoring is always restricted to a few variables, there arises the need, in particular for long lasting simulations, to save the results for a later inspection and exploration. RAM-

SES adopts the technique of the so-called stash file to temporarily save all potentially interesting results. The data are written according to a formally defined LL(1) syntax for easier scanning and parsing by the postanalysis program. This approach allows also to completely omit the control of the simulation by the end-user interface except for the initiation of a so-called structured simulation run (since it may consist of an arbitrary number of elementary runs it can be considered as a freely programmable, hence versatile form of batch-processing). For batch-processing the client interface can be used to program a structured simulation run, which may then even be executed on another machine, e.g. a super-computer, another currently unused workstation, a host serving as a simulation server, or a set of transputers within the workstation.

2.3 The Postanalysis Session

In the postanalysis session simulation results, previously computed during a simulation session, can be analyzed without having to recompute any model behavior. It serves the interactive exploration of model behavior even in cases where an interactive simulation of the results would last too long. Surprising results, e.g. in behavior of indicator variables, may be traced back to the temporal behavior of other, internal system variables, thus often allowing for a better understanding of the system mechanisms. During such an exploratory data analysis of the simulation results, the visualization and the interactive testing of ad-hoc formulated hypothesis play an important role.

In addition to the simulation results the stash file contains detailed information on the global parameters of the simulation environment, on the model and on its model objects which have been used to produce the data. This allows the postanalysis session to install in RAMSES a model and its objects exactly as they have existed during the simulation session, except that the postanalysis model (IORO level 1) is not used to produce the model behavior but to read the already computed behavior from the stash file and display it for the user by using the ordinary RAMSES monitoring mechanisms from the simulation environment. The latter encompasses graphical representations (scattergrams, line charts with optional error bars, 3-dimensional grid plots, contour maps etc.) and the tabular display of numerical values. This approach has not only the advantage of reducing the implementation work, but also to be easier to learn by the user, since every RAMSES user will be familiar with at least the simulation environment (Fig. 7). Whether he/she actually uses the simulation or the post-analysis session to analyze the behavior of a complex model will be of minor importance.

To grant a virtually unlimited data exchange between the simulation session and the postanalysis session, the secondary storage medium containing the stash file is used to store the simulation results. For an efficient access during the postanalysis session the organization of the stash file is crucial. We chose a formally defined LL(1) syntax, but otherwise it is a sequential text (ASCII) file. The latter is a compromise in terms of efficiency and the need for data exchange with other programs than just the postanalysis session software. For instance spread-sheet applications, statistical packages, or document processors can also open the stash file.

More details on the EBNF describing the stash file syntax, the mathematics, and the internal structure of the postanalysis session can be found in GYALISTRAS (1990).

3 An Application Example from Population Ecology

Population systems are used in many areas of the environmental sciences: for instance in ecotoxicological studies or in pest management. The larch bud moth system has been studied intensively for now over four decades and serves as a useful example to illustrate and evaluate RAMSES.

3.1 The population cycles of larch bud moth

Larch bud moth, *Zeiraphera diniana* GN. (*Lep., Tortricidae*), is a univoltine forest defoliating insect, which periodically attacks larch trees in the European Alps between 1700 to 2000 m a.s.l. (BALTENSWEILER & FISCHLIN, 1988). The mean of the cycle length is 9.2 years and the average amplitude amounts to 226.9 larvae/kg larch branches (FISCHLIN, 1982; BALTENSWEILER & FISCHLIN, 1988). The ecological mechanisms causing these population cycles are only partly understood and there exists a range of competing hypothesis postulated by numerous authors (FISCHLIN, 1980; BALTENSWEILER & FISCHLIN, 1988; CLARK *et al.*1967; WILSON, 1975; ANDERSON & MAY, 1982; MAY, 1981). We evaluated and tested all hypothesis first for their plausibility by comparing their assumptions and statements with all known ecological facts and data from the 40 year study and secondly their capability to predict quantitatively the observed system behavior. The latter has resulted in a family of mathematical models (FISCHLIN & BALTENSWEILER, 1979; FISCHLIN, 1982). Members of this model family have been implemented using experimental versions of RAMSES. The simulation session has been realized by embeding the simulation environment ModelWorks[1] (ULRICH, 1987; FISCHLIN *et al.*, 1990) in RAMSES.

3.2 Using RAMSES to model the Larch Bud Moth System

Several aspects of RAMSES, such as modular modelling, were important during the modeling and the simulation of the larch bud moth system. Members of the model family were modeled as submodels in form of separate modules (level 3 System) and measurements were implemented as parallel data-models (level 1 IORO) (Fig. 8). This allowed to compare simulated results with measured and observed time series (Fig. 7).

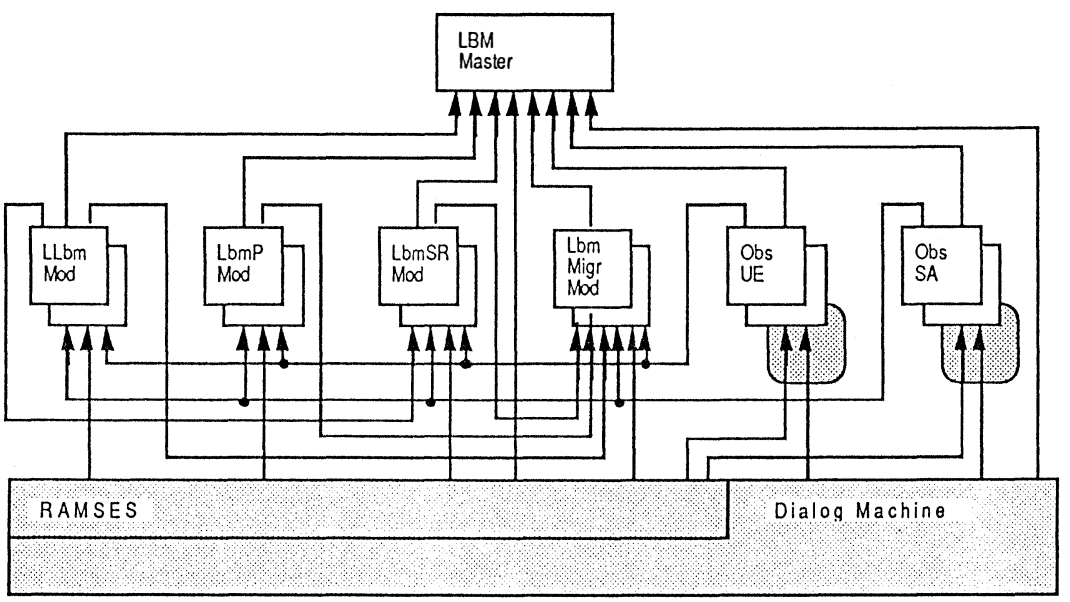

Fig. 8: Module structure of the model definition program implementing the family of population models of the larch bud moth system. ▭ (on top) - program module; ▭ - definition modules in front of implementation modules; → - imports; ⊙ - data files; ▭ (on bottom) - module libraries.

[1]ModelWorks may be used independently from other RAMSES tools, i.e. the modelling and postanalysis session.

The modeling was done with the first technique via the client interface. The master module, the program module *LBMMaster*, combines all modules to a model definition program (Fig. 8). Depending on the current user needs the corresponding submodels can be installed or deinstalled in the RAMSES model base. For instance note the following two submodels: The first submodel, module *LLbmMod*, describes the ecological interaction of the host plant larch (*Larix decidua* MILLER) with the herbivorous insect larch bud moth (*Z. diniana*). The second, parallel data-submodel, module *ObsUE*, mimics the real system by using field data from the Upper Engadine valley in Switzerland, which were sampled from 1949 till the present (BALTENSWEILER & FISCHLIN, 1988). At the begin of a simulation session the parallel data-submodel simply reads the observations stored in a data file into the memory and assigns the measured values during simulations to a monitorable variable, which the simulationist can compare with simulated results (Fig. 7).

4 Discussion

4.1 Strengths and Weaknesses of Interactive Modeling and Simulation

Modeling and simulation of ill-defined systems substantially benefits from interactivity, since iterative system structure identification and systems behavior analysis are typical for these systems, especially also in the field of environmental systems. Interactively connecting and disconnecting submodels, activating or deactivating models are powerful while navigating through a model family which is only partially explored. Systematic model combinations are only possible thanks to such an approach.

The more advanced a study becomes, the more modeling will focus just on a few subsystems or processes. The strengths of interactivity is during the model development phase. Once the structure of a model and its equations become relatively fixed, interactive modeling looses its attractivity and a rather batch-oriented simulation study becomes predominant. The two interfaces of RAMSES, the interactive end-user and the batch-oriented client interface, support these varying needs during the course of a study well.

Especially in the modeling of ill-defined systems there becomes another disadvantage of interactive modeling apparent. It gets easily difficult to keep track of all involved objects and to keep a good overview. This is particularly true for the second technique (s.a. modeling session) where the modeler uses no relational graph editor and works directly on the models and model objects. Therefore hierarchical modeling becomes a necessity; unfortunately RAMSES does currently not support real hierarchical modeling. The latter could be realized by an explicit subordination of submodels which would not only be reflected in the module structure but also would be recognized by RAMSES. Implementing hierarchical modeling such that it starts always from a single root model, would also offer the advantage of steering the user towards a top-down model design.

Disadvantages of interactive simulation become also particularly evident if computational needs are big. The more complex a model, the larger the conflict with the patience of the user, because.the simulationist can only watch a more and more limited selection of the results. For complex systems the interactive simulation degenerates too often to a monitoring of a few, uninteresting indicators, the actual results remaining hidden. Not only does this call for the separation of the simulation and the post-analysis as realized in RAMSES, but also offers the advantage of a transparent use of computing servers, such as a simulation server or a super computer. RAMSES' session concept appears to optimally support such solutions; for instance, except for restrictions inherent to the programming languages available on the Cray-XMP, did we encounter no fundamental difficulties when experimenting with a super-computer implementation of the RAMSES simulation environment for structured simulation runs. Except for the simulation session, which may be trans-

ferred to a simulation server, all other sessions remain on the workstation and can there profit from the available interactivity. This unites and allows to have the best of both worlds, the world of the batch-oriented hosts, strong in number crunching, and that of the workstations, strong in interactive use.

Some rather technical, implementation problems shall also be mentioned: Against common expectations, continuous consistency checking during interactive editing is often impossible, since users violate consistency while editing all the time: E.g. a first portion of an entry may already have been typed, a second not; a completely normal, but inconsistent situation, since some terms may already have been referenced, but are not yet present. Due to the difficulties to have a good overview when working with ill-defined systems, however, it would be crucial to have the computer perform consistency checks. Thanks to the state diagram of RAMSES' session concept (Fig. 1), a solution could be found, namely if the model definition and the experimental frame are not compatible, RAMSES rejects the transition from the modeling or experiment definition session into the simulation session. Inasmuch expert systems could really help to support the modeling process and consistency testing is currently not well understood. We have experimented with little expert systems, but only achieved disappointing results; just complete novices could really profit from the available advices.

Compared with previous implementations of the presented model family made with more cumbersome simulation tools (FISCHLIN & BALTENSWEILER, 1979) or commercially available simulation software (Marr, 1989) indicate that RAMSES is not only efficient and elegant, but also tends to support the user in such a way, that results are obtained in a more systematic manner. This has several reasons, but among the more important ones is certainly the choice of the programming language Modula-2.

Thanks to Modula-2, RAMSES supports elegantly a modular implementation of the members of a model family. Each submodel forms a self-contained unit, typically a Modula-2 module, yet they may exchange informations by IO-links. E.g. in the presented example relationships between submodels, such as the computation of initial states from observations, could be implemented with output to input coupling (Fig. 8). All model equations, no matter how complicated or of which form, could also be elegantly implemented; e.g. the migration model uses recursion to model spatial orientation and moth flying behavior (FISCHLIN, 1982).

The power of the programming language used to implement RAMSES has shown to be most suitable to achieve our originally set goals, because it supported modular and structured programming and helped to implement efficiently the software tools needed for the interactive exploration of model behavior. Moreover the RAMSES user profits as well, e.g. a sensitivity analysis could be implemented by programming a few lines of code and installing the executing procedure as a structured simulation run; the testing of pesticide applications or the heuristical design of a management scheme were also straightforward and easy to realize.

Much effort had to go into the design of a software system which is attractive for the beginner as well as open enough for the specialist who is interested in advanced techniques. This resulted in a basic symmetry between the end-user and the client interface. However, in case of conflict we opted rather for an optimally open system structure, than for the ease of use for the beginner, who is more likely to use just the end-user interface. These design goals are reflected e.g. in the maximal functionality of the client interface. Building on a solid model-based foundation and adding later front-end modules for easier use of the interactive end-user interface by beginners seemed to be an appropriate development strategy. It offered also right from the begin the sophisticated user optimal access to RAMSES. Offering both, the end-user and the client interfaces simultaneously has also the advantage that the user may smoothly transgress from major reliance on the end-user interface to the client interface by implementing step-wise more and more elaborate tasks, such as statistical analysis of data collection from many runs or parameter identification techniques etc. This way the risk to loose investments in complex model implementations can be kept minimal.

RAMSES demonstrated also that modeling and simulation software have to be tightly coupled. In particular interactive modeling and simulation require a common kernel managing dynamically models, model objects, and associated values in a model base accessible by both the modelling and the simulation sessions. The heap technique we adopted to achieve this behavior was surprisingly efficient: Compared with more traditional simulation software architectures the efficiency losses in computational speed were on the average as little as 10-20% (s.a. ULRICH, 1987).

An important disadvantage of RAMSES is of course the rather big software development effort. However, it appears to be unavoidable, since the new concepts required a substantial rewriting or at least restructuring of existing simulation software. For instance modular modeling required to implement integration algorithms anew. This is because the IO-links among submodels must be calculated before the numerical integration of differential equations. The user interface development is often underestimated and forms the dominant fraction of a rigorous, user-friendly interactive program. However, designing a good man-machine interface can be crucial for the final quality of a software. Only thanks to the *Dialog Machine* (FISCHLIN, 1986) could these efforts be minimized and the port of the simulation environment ModelWorks to another target machine, given a *Dialog Machine* was available, could be accomplished in only a few days labor.

Finally either the design as well as the use of the RAMSES tools were facilitated by the modelling theory. This is because the software architecture could be based on a model base kernel. It provides the installation (or deinstallation) of systems theoretically well understood objects, such as state variables or model parameters etc. Furthermore the user is assisted in or even guided to adopt a structured modelling approach. RAMSES recognizes and hence maintains values only if they conform to the mathematically defined concepts of the standard model formalisms SM, DESS, or DEVS. Yet this does not prevent the modeler from using rather unconventional formalisms such as recursion.

4.2 Perspectives

Since workstations and personal computers become increasingly popular and ever more powerful, interactive modeling and simulation appear to gain rapidly in importance for the analysis of ill-defined systems. This is particularly true in the field of environmental systems, where much remains to be done. Although heuristic approaches still dominate the field, and not all concepts are yet well understood, it seems that progress can be achieved along the described lines. In order to improve the health of our environment it is hoped that the modeling of environmental systems will not just remain a highly specialized activity reserved to a few specialists, but will actually contribute to the better understanding and solving of environmental problems.

5 References

ANDERSON, R.M. & MAY, R.M., 1980. *Infectious diseases and population cycles of forest insects.* Science, **210**: 658-661.

ANONYMOUS, 1988. *Catalog of simulation software.* Simulation **51** (4): 136-156.

BALTENSWEILER, W. & FISCHLIN, A., 1988. *The larch bud moth in the Alps.* In: Berryman, A.A. (ed.), *Dynamics of forest insect populations: patterns, causes, implications.* New York a.o.: Plenum Publishing Corporation, p. 331-351.

BROOKS, F.P. JR., 1979. *The mythical man-month - Essays on Software Engineering.* Reading, M. a.o.: Addison-Wesley Publ. Comp., 195pp.

CELLIER, F.E., 1975. *Continuous-systems simulation by use of digital computers: a state-of-the-art-survey and prospectives for development.* In: Hamza, H. (ed.) Proc. of the international Symposium SIMULATION'75, Zurich, Switzerland, June 1975, (To be ordered from:) Acta Press, P.O. Box 354, CH-8053 Zurich, p. 18-25.

CELLIER, F. E., 1979. *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.* Diss. ETH No 6483.

CELLIER, F.E., 1984a. *Simulation software: Today and tomorrow.* Zeitschr. Schweiz. Ges. Automatik, **4**: 7-22.

CELLIER, F.E., 1984b. *How to enhance the robustness of simulation software.* In: Ören, T. I., Zeigler, B. P., Elzas, M. S.(eds), *Simulation and Model-Based Methodologies: An Integrative View*, 651pp., Springer, Berlin a.o., p. 519-537.

CELLIER, F.E. (ED.), 1982. *Progress in Modeling and Simulation.* Academic Press, London a.o., 466pp.

CELLIER, F.E. & FISCHLIN, A., 1982. *Computer-assisted modeling of ill-defined systems.* In: In: Trappl, R., Klir, G.J., Pichler& F.R. (eds.), *Progress in cybernetics and systems research. Vol. VIII: General systems methodology, mathematical systems theory, fuzzy sets.* Proc. of the 5th European Meeting on Cybernetics and Systems Research, University of Vienna, Austria, April 8-11, 1980, p. 417-429.

CLARK, L., GEIER, P., HUGHES, R. & MORRIS, R.F., 1967. *The ecology of insect populations in theory and practice.* London: Methuen, 232pp.

DAVIDSON, R.S. & CLYMER, A.B., 1966. *The desirability and applicability of simulating ecosystems.* Ann. N.Y. Acad. Sci., **128**: 790-794.

FISCHLIN, A., 1982. *Analyse eines Wald-Insekten-Systems: Der subalpine Lärchen-Arvenwald und der graue Lärchenwickler Zeiraphera diniana Gn. (Lep., Tortricidae).*. Diss. Eidg. Tech. Hochsch. Zürich No. 6977, 294pp.

FISCHLIN, A., 1986. *Simplifying the usage and the programming of modern workstations with Modula-2: The 'Dialog Machine'.* Internal report, Project-Centre IDA/CELTIA, Swiss Federal Institute of Technology Zürich (ETHZ), Zürich, Switzerland, 13pp.

FISCHLIN, A. & BALTENSWEILER, W., 1979. *Systems analysis of the larch bud moth system. Part I: the larch-larch bud moth relationship.* Mitt. Schweiz. Ent. Ges., **52**: 273-289.

FISCHLIN, A. & ULRICH, M., 1987. *Interaktive Simulation schlecht-definierter Systeme auf modernen Arbeitsplatzrechnern: die Modula-2 Simulationssoftware ModelWorks.* Proceedings, Treffen des GI/ASIM-Arbeitskreises 4.5.2.1 *Simulation in Biologie und Medizin*, February, 27-28, 1987, Vieweg, Braunschweig, p. 1-9.

FISCHLIN, A., ROTH, O., GYALISTRAS, D., ULRICH, M. & NEMECEK, T., 1990. *ModelWorks - An interactive simulation environment for personal computers and workstations.* Manual for Version 2.0. Systems Ecology Group, Internal Report 8, Swiss Federal Institute of Technology Zürich, Switzerland, 182pp.

GUTKNECHT, J., 1983. *System programming in Modula-2: mouse and bit-mapped display.* Internal report No. 56, Department of computer science, Swiss Federal Institute of Technology, Zürich, Switzerland, 58pp.

GYALISTRAS, D., 1990 (In prep.). *Interactive post-analysis of simulation results on a workstation.* Systems Ecology Group, Swiss Federal Institute of Technology, Zürich, Switzerland.

HOLLING, C.S., 1964. *The analysis of complex population processes.* Can. Entomol., **96**: 335-347.

INNIS, G.S., 1972. *Simulation of ill-defined systems: Some problems and progress.* Simulation, **19**: 33-36.

KARPLUS, W.J., 1976. *The spectrum of mathematical modeling and system simulation.* In: Dekker, L. (ed.), Proc. of the 8th AICA Congress on Simulation of Systems, Delft, The Netherlands. North-Holland Publ. Co., p. 5-13.

KLIR, G.J., 1979. *Computer-aided system modeling.* In: Halfon, E. (ed.), *Theoretical Systems Ecology*, Academic Press, New York, p. 291-323.

KREUTZER, W., 1986. *System simulation: programming styles and languages.* Sydney a.o.: Addison-Wesley, 366pp.

MARR, G.R., 1989. *Better interaction with ACSL simulation programs.* In: Allen, R.W. (ed.), *Modeling and simulation on microcomputers*, 1989, The Society for Computer Simulation International, p. 69-73.

MAY, R.M. (ED.), 1981. *Theoretical ecology. Principles and applications.* Blackwell Scientific Publications, Osney Mead, Oxford, 2nd ed., 489pp.

NIEVERGELT, J. & WEYDERT, J., 1980. *Sites, modes and trails: Telling the user of an interactive system where he is, what he can do, and how to get to places.* In: R. A. GUEDJ et al. (eds.), *Methodology of Interaction*, North-Holland, Amsterdam, p.327-338.

NIEVERGELT, J. & VENTURA, A., 1984. *Die Gestaltung interaktiver Programme.* B. G. Teubner Stuttgart, 124pp.

ÖREN, T.I., 1982. *Computer-aided modeling systems.* In: Cellier, F.E. (ed.), *Progress in Modeling and Simulation*, Academic Press, London a.o., p.189-203.

ÖREN, T.I., 1984. *GEST - A Modeling and Simulation Language Based on System Theoretic Concepts.* In: Ören, T.I., Zeigler, B. P., Elzas, M.S.(eds.): *Simulation and Model-Based Methodologies: An Integrative View*, 651pp., Springer, Berlin a.o., p. 281-335.

ULRICH, M., 1987. *ModelWorks. An interactive Modula-2 simulation environment.* Post-graduate thesis, Project-Centre IDA, Swiss Federal Institute of Technology Zürich (ETHZ), Switzerland, 53pp.

VANCSO-POLACSEK, K., 1990. *Theory and practice of computer assisted simulation and modeling on professional workstations.* Diss. ETH No., 109pp.

VANCSO, K., FISCHLIN, A. & SCHAUFELBERGER, W., 1987. *Die Entwicklung interaktiver Modellierungs- und Simulationssoftware mit Modula-2.* In: Halin, J. (ed.), *Simulationstechnik*, Informatik-Fachberichte 150, Springer, Berlin, p. 239-249.

WILSON, E.O., 1975. *Sociobiology the new synthesis.* Belknap Press of Harvard University Press, Cambridge a.o., 697pp.

WIRTH, N., 1985. *Programming in Modula-2, Third, Corrected Edition.* Springer-Verlag, Berlin a.o., 202pp.

WIRTH, N., 1986. *Algorithms & data structures.* Prentice/Hall International, Inc., 288pp.

WIRTH, N., 1988. *The programming language Oberon.* Software - Practice and Experience, **18**: 171-90.

WIRTH, N., 1989a. *From Modula to Oberon.* Institut für Informatik ETHZ, Swiss Federal Institute of Technology Zürich, Switzerland, internal report 111: 3-10.

WIRTH, N., 1989b. *The programming language Oberon (revised report).* Institut für Informatik ETHZ, Swiss Federal Institute of Technology Zürich, Switzerland, internal report 111: 11-28.

WYMORE, A.W., 1984. *Theory of Systems.* In: Vick, C. R., Ramamoorthy, C. V.(eds.): *Handbook of Software Engineering*, Van Nostrand Reinhold Company, New York.

ZEIGLER, B.P., 1976. *Theory of modeling and simulation.* Wiley, New York a.o., 435pp.

ZEIGLER, B.P., 1979. *Multilevel multiformalism modeling: an ecosystem example.* In: Halfon, E. (ed.), *Theoretical Systems Ecology*, Academic Press, New York, p. 17-54.

ZEIGLER, B.P., 1984. *System theoretic foundations of modeling and simulation.* In: Ören, T.I., Zeigler, B. P., Elzas, M.S.(eds.), *Simulation and Model-Based Methodologies: An Integrative View*, 651pp., Springer, Berlin a.o.

Acknowledgements: I thank Dr. Olivier Roth for the reading of the manuscript, the sincere discussions, and many valuable suggestions for improvements. Many thanks are due to Markus Ulrich, especially for his original design of the simulation environment ModelWorks. I thank Dr. Olivier Roth, and Dimitrios Gyalistras for their substantial contributions to new ideas and the implementation of RAMSES sessions. Particular thanks also to the first users, namely Thomas Nemecek and also Harald Bugmann, who have been willing to serve as «guinea-pigs» for the testing and exploring of the numerous concepts and their validity in their daily research. Finally I owe heartily thanks to Prof. Dr. W. Schaufelberger, not only for his substantial support, but also for his unceasing encouragement, which made this research and the associated development only possible.