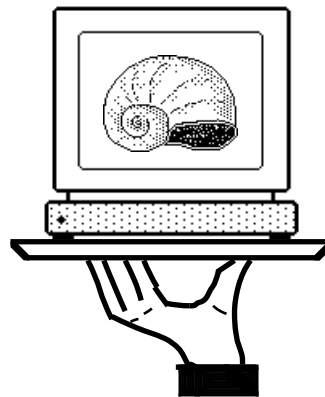


SYSTEMÖKOLOGIE ETHZ
SYSTEMS ECOLOGY ETHZ

Bericht / Report Nr. 21

Introducing RASS - The RAMSES Simulation Server

J. Thöny, A. Fischlin, D. Gyalistras



January 1995

Eidgenössische Technische Hochschule Zürich ETHZ
Swiss Federal Institute of Technology Zurich

Departement für Umweltnaturwissenschaften / Department of Environmental Sciences
Institut für Terrestrische Ökologie / Institute of Terrestrial Ecology

The System Ecology Reports consist of preprints and technical reports. Preprints are articles, which have been submitted to scientific journals and are hereby made available to interested readers before actual publication. The technical reports allow for an exhaustive documentation of important research and development results.

Die Berichte der Systemökologie sind entweder Vorabdrucke oder technische Berichte. Die Vorabdrucke sind Artikel, welche bei einer wissenschaftlichen Zeitschrift zur Publikation eingereicht worden sind; zu einem möglichst frühen Zeitpunkt sollen damit diese Arbeiten interessierten LeserInnen besser zugänglich gemacht werden. Die technischen Berichte dokumentieren erschöpfend Forschungs- und Entwicklungsergebnisse von allgemeinem Interesse.

The first part of this report has been published. Please cite as:

Thöny, J., Fischlin, A. & Gyalistras, D., 1994. *RASS: Towards bridging the gap between interactive and off-line simulations*. In: J. Halin, W.K.a.R.R. (ed.), *CISS- First Joint Conference of International Simulation Societies*, Zürich, Switzerland, The Society for Computer Simulation International, P:O: Box 17900, San Diego, Cal. 92177, USA, pp. 99-103

Adressen der Autoren / Addresses of the authors:

J. Thöny, Dr. A. Fischlin, D. Gyalistras
Systemökologie ETH Zürich
Institut für Terrestrische Ökologie
Grabenstrasse 3
CH-8952 Schlieren/Zürich
SWITZERLAND
e-mail: sysecol@ito.umnw.ethz.ch

Introducing RASS – The RAMSES Simulation Server

Jürg Thöny
with contributions from A. Fischlin and D. Gyalistras

RASS forms a new component of the RAMSES¹ modelling and simulation concept. It allows to automatically translate interactive RAMSES model definition programs and to execute them in a batch mode on a high performance computer.

This report describes RASS and its implementation RASS 1.1. It is a collection of three documents:

1. RASS: Towards bridging the gap between interactive and off-line simulations.
2. RASS 1.1 User's Guide
3. RASS 1.1 Developer's Guide

The first document is an introduction to the general ideas and concepts behind RASS and was presented in 1994 at the *CISS - First Joint Conference of International Simulation Societies*, Zuerich.

The user's guide describes how to install and use RASS on Unix workstations. It also contains a tutorial guiding the users through a sample transition from RAMSES to RASS.

The developer's guide is a description of the RASS modules and the development environment, as required for further development.

¹ Research Aids for the Modelling and Simulation of Environmental Systems

RASS: Towards bridging the gap between interactive and off-line simulation.

Jürg Thöny, Andreas Fischlin and Dimitrios Gyalistras
Systems Ecology
Institute of Terrestrial Ecology
Swiss Federal Institute of Technology Zürich (ETHZ)
CH-8952 Schlieren/Zürich

ABSTRACT

Interactive model exploration is an important step in the process of model-building of ill-defined systems such as ecosystems, a task which is well supported by RAMSES (Research Aids for the Modelling and Simulation of Environmental Systems). However, interactive simulation is of minor interest at a later stage of research, when large scale batch oriented simulation experiments are needed.

RASS, the RAMSES Simulation Server, typically located on a high performance computer, translates, compiles, links, and executes in a batch mode interactive RAMSES model definition programs [MDP]. MDPs must be given in source form and formulated according to one or any combination of the following standard formalisms: SQM (Sequential Machine), DESS (Differential Equation System Specification), or DEVS (Discrete Event System Specification). The simulation results generated and returned by RASS can then be explored interactively by means of the post-analysis component of RAMSES.

RASS was found to allow for automatic translation of interactive programs developed for a graphical user-interface with windows and menus in order to execute them in a batch mode to produce correct and reliable simulation results. Three complex case studies from the field of ecology and engineering showed that performance gains of 1'100-7'200% (overhead included) can be obtained when RASS is run on a SUN S10 server relative to the time needed when running the MDP interactively on an average personal computer. Since all transfers showed to be user-friendly and smooth, we concluded that RASS offers RAMSES simulationists an efficient and attractive alternative for solving interactive MDPs whenever off-line simulations are needed, or when it is a necessity because of the high computing requirements.

1 INTRODUCTION

The simulation of ill-defined systems, e.g. ecological systems, challenge most existing simulation software by specific, sophisticated requirements (Cellier and Fischlin, 1982; Kreuzer 1986; Fischlin and Ulrich 1987; Vancso *et al* 1987; Vancso 1990).

At an early stage of a research project, an interactive simulation environment is of paramount help for the

modeling of ill-defined systems. However, at later stages, i.e. when simulation studies such as sensitivity analysis, parameter identification, or stability properties are of prime interest, batch production of simulation results is needed. In order to satisfy both needs during the entire course of a research project, a simulation software must support both interactive and batch simulations.

The RAMSES [Research Aids for the Modelling and Simulation of Environmental Systems] software (Fischlin 1991) was designed to support interactive, modular modeling and simulation of ill-defined systems with one or any combination of the classical model formalisms SQM [SeQuential Machine], DESS [Differential Equation System Specification], DEVS [Discrete Event System Specification] (Zeigler 1976, 1979; Wymore 1984).

In RAMSES, any model implementation is made in form of a so-called MDP [Model Definition Program] (Fischlin *et al.* 1994). A MDP represents an interactive program and therefore shares the common problems of all sophisticated interactive software. These are the limited portability and the computational overhead of the user interface.

Thus the following questions arise: Is it possible to reuse an interactive simulation program, such as a RAMSES-MDP for non-interactive batch calculations without any changes on the source level? Which problems need to be solved to correctly run MDPs within a batch simulation environment? How valid are the simulation results? Which gain in performance can be achieved thanks to reduced graphics overhead and more powerful machines?

The here presented solution RASS [RAMSES Simulation Server] forms a new component of RAMSES. RASS receives interactive MDPs in source form, runs them off-line on a simulation server, and generates simulation results, which can be again explored interactively by the Post-Analysis component of RAMSES. First we present the architecture of RASS and discuss the conceptual problems which had to be solved. Three case studies serve to demonstrate the obtainable performance gain and the quality of the simulation results. Finally, portability issues and future enhancements are discussed.

2 MATERIAL

The current implementation of RAMSES covers interactive MDP development and simulation by several

components such as the standard DM [Dialog Machine] (Fischlin and Schaufelberger 1987; Fischlin *et al.* 1987), the standard MW [ModelWorks], and the PA-session [Post-simulation Analysis].

RASS was implemented in Modula-2 using MacMETH (Wirth *et al.* 1992) on Macintosh computers and EPC Modula-2 (Anonymous 1992) on SUN workstations. It uses a new batch oriented implementation of the DM [Batch-Dialog Machine].

Three MDPs, *ForClim*, *Diversity*, and *NumInt* were used as case studies.

All simulation experiments were performed either on an Apple Macintosh Quadra 700 for RAMSES or on a SUN S10 for RASS. On the Macintosh we measured the time inside the MDPs with no other applications running during simulation. On the SUN we measured the simulation/experiment with the Unix time command during a minimal work load.

3 RESULTS

3.1 Simulation with RASS

A RASS task is conceptually a RAMSES simulation session (Fischlin 1991). However, RASS implements the simulation session in a different way.

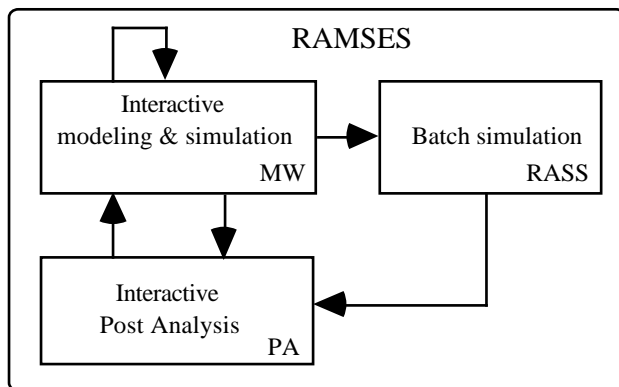


Fig. 1: State transitions while developing and solving models using RASS within RAMSES.

A typical modeling-simulation cycle (Fig. 1) involves the following steps:

- 1 Develop a model interactively in the RAMSES simulation and modeling session, and run local simulations interactively.
- 2 Produce model behaviour with RASS in a batch mode.
- 3 Analyse the simulation results interactively in the PA session.

3.2 Design and Implementation

RASS was designed to fulfil the following main requirements:

RASS has to achieve a high portability in two ways: First it should be easily portable. Second any MDP including its corresponding input/output data files should be exchangeable between the batch and interactive simulation environment without any changes.

The simulation results of RASS have to be reliable, i.e. the batch processing must not hamper the validity of the simulation results.

The transition from the interactive RAMSES to the batch oriented RASS should be smooth, and require a minimum of user interactions.

3.2.1 RASS Architecture

RASS resides on top of several software layers, which enhance portability and software maintenance (Fig. 2).

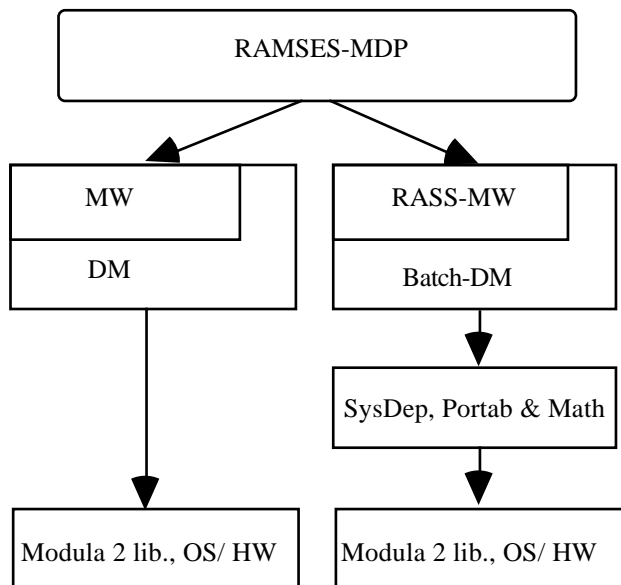


Fig. 2: Architecture of RAMSES and RASS.

All RAMSES software is implemented on top of the DM using the language Modula-2. The DM is a library that allows to program interactive applications independently from the underlying graphical user interface. All interactive DM-commands are designed as a program state transition with a known precondition and a defined postcondition. The DM allows the transition only if the precondition is true and the postcondition can be fulfilled; otherwise the interactive DM modally demands a user intervention, usually with a default answer to the request.

Since RASS is not interactive, a new DM implementation [Batch-DM] was necessary. The Batch-DM has the same programmatic interface, but has no visible user interface. An optional terminal like I/O is provided. The various DM components are maintained internally, such

that the state transitions are the same as in the interactive DM.

The Batch-DM is built solely on top of the two modules *SysDep* and *Portab*. The only exception is the *DMMathLib*. *SysDep* and *Portab* hide the local Modula-2 library and the OS/hardware. To prevent performance loss, we implemented the *DMMathLib* with direct calls to the machine dependent math library, instead of encapsulating these functions in *SysDep*.

3.2.2 Running MDPs with RASS

Essential is that any MDP has the same execution thread regardless whether executed interactively by MW or by RASS.

A RAMSES-MDP has very few interactive components. However due to the open system architecture such as the DM interface, it is possible to add interactive elements (Fig. 2). Since most of them are potential branches in the execution thread, special care has to be taken. The Batch-DM handles the core user interface components of the DM as follows:

Components without default actions are not implementable in the Batch-DM; they lead to a program abort. However, in most cases they can be easily replaced by the programmer with another DM function.

Dialogues: DM dialogues entities have default values. Therefore the dialogues lead to a predictable postcondition.

Menus: The menu structure is maintained internally, but no menus are shown on a screen. However, execution of menu commands is possible under program control.

Windows: No windows are provided, however the text written on them is directed to a standard output file.

3.2.3 Data and Result Files

RASS is based on the same DM library interface as the standard MW. Both expect and produce files in textual form. Therefore the input and output files are fully interchangeable.

3.2.4 Current Implementation

A simulation experiment to be solved by RASS is given by a MDP, and the corresponding input data files - only these have to be transferred and converted according to the conventions of the specific simulation host. The transfer and translation can be accomplished, e.g. by the means of FTP [File Transfer Protocol].

To provide a user-friendly transition from interactive RAMSES to RASS we created a tool called *RASSMakeMake*. It generates automatically a make script that produces the executable simulation program.

The output of a simulation can be explored locally or transferred back to the host where the interactive PA resides.

3.3 Case Studies

We selected three case studies to cover all three currently by RAMSES supported standard modeling formalisms (SQM, DEVS, and DESS):

a) *ForClim* (Bugmann 1994), a SQM that models the stochastic species succession of forests and is currently used to study the impact of climatic change on forests (Bugmann and Fischlin 1994). *ForClim*'s input data files contained machine specific characters which had to be removed before the *ForClim* was able to process them under RASS. We ran that experiment that generates the so-called reference output, which was not numerically identical; but, since *ForClim*'s output depends on pseudo-random numbers it is highly dependent on the precision of the floating point instructions. However, the results deviate only insignificantly from the expectations, and were therefore interpreted as correct.

b) *Diversity* (Fischlin *et al* 1994), a DEVS simulating the reinvasion of species and diversity restoration on an island, which has been hit by a volcanic eruption. RASS returned exactly the same number of years for diversity restoration as the interactive version.

c) *NumInt*, a DESS that compares the position of an earth satellite computed with fixed step integration methods of various orders with an analytically determined expectation. The purpose of *NumInt* is to explore the range of valid simulation results, limited either by too big or too small step sizes (rounding errors). The RASS results were the same for lower order integration methods, whereas those of the higher order integration methods differed for the smaller step sizes due to the rounding errors.

All three MDPs were transferable without any changes. The only exception was *ForClim* which required initially one iteration.

3.3.1 Performance Measurements

In order to compare a simulation cycle of an interactive MDP between interactive RAMSES and RASS, we started measurements only from the moment of an already developed RAMSES-MDP and neglected constant terms if they were approximately the same on both hosts, e.g. the time to build and link/load the MDP.

Terms:

t_t = Turnaround time.

t_s = Time to execute a simulation experiment.

t_c = Time to transfer MDPs and/or data files to the simulation host and transfer the result files back to the PA host.

t_a = Time to set up the interactive analysis [PA] of the results.

$t_t(\text{RAMSES}) = t_s$

$t_t(\text{RASS}) = t_s + t_c + t_a$

Since $t_a \ll t_s$ we get:

$t_t(\text{RAMSES}) = t_s$

$t_t(\text{RASS}) = t_s + t_c$

MDP	MW Mac $t_t=t_s$	RASS SUN		
		t_s	t_c	$t_t=t_s+t_c$
ForClim	37800	3134	132	3266
Diversity	638	13	120	133
NumInt	70758	860	115	975

Table 1: Turnaround time in seconds of three case studies in the interactive RAMSES environment on a Macintosh Quadra 700 and the batch RASS environment on a SUN S10.

The over-all performance gain obtainable by the simulationist was a factor 12-82 for t_s and a factor 11-72 for t_t (Table 1).

3.3.2 Post Analysis

The RAMSES-PA session allows to interactively explore simulation results previously written to a stash file by MW or RASS (Fig. 1). The PA supports an arbitrary number of stash files, simulation runs per file, and models per run, limited only by the computer's available memory. Based on MW, PA mimics the model behaviours by reading the results from stash files, instead of computing them on-line (Fig. 3). Not only does it allow to compare stored results among several stash files, but also with interactively simulated model behaviours.

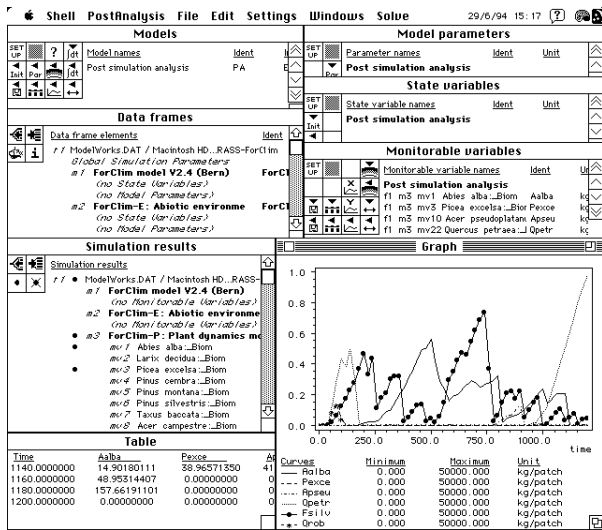


Fig. 3: Typical screen, here from the case study *ForClim*, of a RAMSES post-analysis session on the simulationist's personal computer. This session supports the interactive analysis of the uploaded simulation results by processing a stash file, which RASS previously produced on the simulation server.

We tested PA with ten *ForClim* simulation runs, which lasted under RASS $t_s = 145s$ and produced a 944kB

stash file. For each run were documented 95 variables at 600 simulation time points. Transfer of the stash file to the local personal computer required 26s, such that, together with the transfer of *ForClim* to the RASS-server, $t_c = 76s$. Loading of the PA-session required ca. 4s, and setting up of the workspace (Fig. 3), including the preparation of all runs within PA, additional 12s, yielding $t_a = 16s$. The graphical representation and tabulation of six variables from a single simulation run required ca. 6s (Fig. 3), whereas the simultaneous inspection of the same variable from all ten runs required 27s. These times increased by ca. 14 % (to 7s and 31s, respectively) when the stash file was directly accessed via a LAN from the mass storage device of the simulation server. Thus, the average time to inspect one simulation run amounted to ca. 30s, via the LAN to only 27s.

The combined use of RASS and PA compares favourably with the 85s needed for a single *ForClim* simulation with the interactive MW on the simulationist's computer. We concluded that the RAMSES-PA not only allows to efficiently and flexibly analyse batch simulation results, but that it may even be of interest for inspecting model behaviours interactively during a model development phase.

4 DISCUSSION

4.1 Domain of RASS

From the perspective of the simulationist, the break even point for RASS depends on the available computer infrastructure. The easier to use and the faster the connection between the interactive and the RASS host is, the smaller simulation experiments may get to be still suitable.

The actual break even point is given by:

$$t_t(\text{RAMSES}) = t_t(\text{RASS}) = t_c + t_s(\text{RASS})$$

$$\Rightarrow t_s(\text{RAMSES}) - t_s(\text{RASS}) = t_c$$

A gain in performance occurs if

$$t_c < (t_s(\text{RAMSES}) - t_s(\text{RASS}))$$

A typical t_c for a small model is less than 3 minutes. The average performance gain for t_s is a factor 12 to 82.

Given a model with $t_c = 3$ minutes and a modest performance factor of $t_s = 12$, the break even point for a single simulation run is reached for $t_s(\text{RAMSES}) = 196s$. Note, if a simulation experiment is executed within a loop, the smaller t_c becomes, and therefore the sooner the break point may be reached.

For *ForClim* this was found to be already the case if a structured simulation experiment consists at least of two runs (See 3.3.2 Post Analysis).

4.2 Portability

In order to make Modula-2 code portable, special care had to be taken. Modula-2 is a formally well-defined language with some exceptions. In particular the type

transfer functions and the LONG type language extensions caused portability problems. Therefore RASS and the Batch-DM were written in a portable subset of Modula-2.

There exists no standard library for Modula-2. However, in order to port RASS to a different machine only the modules *SysDep*, *Portab*, and *DMMathLib* have to be re-implemented. *SysDep* and *Portab* consist of 593 lines of source code in the EPC Modula-2 implementation.

Since the RASS binary to text conversion for numbers is dependent of the IEEE floating-point standard, it runs only on machines which follow this standard.

4.3 Planned Improvements

We plan to implement a RASS shell with communication facility. This would allow a user transparent transfer of MDPs and/or data files.

Not every powerful host provides a Modula-2 compiler. It is planned to apply a Modula-2 to C translator to RASS and MDPs in order to run simulations on these hosts. Since most C compilers generate optimised code, an additional gain in performance could be expected.

5 CONCLUSIONS

RASS is capable of solving correctly and efficiently any simulation experiment, given it is defined in form of a RAMSES-MDP (Model Definition Program).

This is possible regardless of the original design for interactive usage. We could demonstrate that the 'Dialog Machine' provides a solid basis to write interactive programs such as a RAMSES-MDPs, since they could be executed in a batch mode with only few, insignificant restrictions. Although the correctness of any MDP solved by RASS can not be proven, at least for the case-studies RASS produced correct results. Thus, the uploading from an interactively developed RAMSES-MDP to a RASS batch simulation server is possible and can even be implemented in a smooth user transparent manner.

In the tested "real-world" case studies, we obtained substantial average gains in performance (e.g. between 1'100% and 7'200% for the simulationist's turnaround-time (Table 1)). Therefore RASS allows to profit in two ways: First it allows to freely engage in interactive development of complex simulation models on widely available PCs or work-stations using the interactive RAMSES software. Second, at later stages, i.e. when complex, well defined simulation experiments are of prime interest, the RAMSES-MDP can be easily transferred to more powerful machines, e.g. a super-computer, running the RASS simulation server.

Since RASS is highly portable and can be implemented easily on any host-computer, RASS provides a user-friendly simulation environment. It allows smooth transitions: a) from interactive work-station based to simulation server based off-line simulations, b) from the server back to simulationist's work-station for an inter-

active exploration and analysis of the simulation results under the RAMSES post-analysis [PA] session.

REFERENCES

- Anonymous. 1992. *EPC Modula-2. User's Reference Manual*. Second Edition. January 1992. Edinburgh Portable Compilers Ltd.
- Bugmann, H. 1994. "On the Ecology of Mountainous Forests in a Changing Climate: A Simulation Study." Ph.D. thesis No. 10638, Swiss Federal Institute of Technology Zürich (ETHZ).
- Bugmann, H. and Fischlin, A. 1994. "Comparing the behaviour of mountainous forest succession models in a changing climate." In Beniston, M. (ed.), *Mountain Environments in Changing Climates*. Routledge, London & New York: 206-221 (invited, refereed).
- Cellier, F.E. and Fischlin, A. 1982. "Computer-assisted modelling of ill-defined systems." In *Proceeding. of the 5th European Meeting on Cybernetics and Systems Research*, (University of Vienna, Austria, April 8-11, 1980), McGraw-Hill, Washington, N.Y., 417-429.
- Fischlin, A.; Gyalistras, D.; Roth, O.; Ulrich, M.; Thöny, J.; Nemecek, T.; Bugmann, H. and Thommen, F. 1994. *ModelWorks 2.2: An interactive simulations environment for work stations and personal computers*. Second, completely revised edition. Internal Report No. 14, Systems Ecology Group, ETH Zurich.
- Fischlin, A. and Schaufelberger, W. 1987. "Arbeitsplatzrechner im technisch-naturwissenschaftlichen Hochschulunterricht." *Bulletin SEV/VSE*, 78 (Januar): 15-21.
- Fischlin, A. 1991. "Interactive Modeling and Simulation of Environmental Systems on Workstations." In Möller, D.P.F. (ed.), *Analysis of Dynamic Systems in Medicine, Biology, and Ecology*. Informatik-Fachberichte 275, Springer, Berlin a.o. 131-145.
- Fischlin, A.; Mansour, M.A.; Rimvall, M. and Schaufelberger, W. 1987. "Simulation and computer aided control system design in engineering education." In Troch, I., Kopacek, P. & Breitenacker, F. (eds.), *Simulation of Control Systems*, Pergamon Press, Oxford a.o. 51-60.
- Fischlin, A. and Ulrich, M. 1987. "Interaktive Simulation schlecht-definierter Systeme auf modernen Arbeitsplatzrechnern: die Modula-2 Simulationssoftware ModelWorks." In *Proceedings of Simulation in Biologie und Medizin*, February, 27-28, 1987, Vieweg, Braunschweig: 1-8.
- Kreutzer, W. 1986. *System simulation: programming styles and languages*. Sydney a.o.: Addison-Wesley.
- Vancso, K.; Fischlin, A. and Schaufelberger, W. 1987. "Die Entwicklung interaktiver Modellierungs- und Simulationssoftware mit Modula-2." In: Halin, J. (ed.), *Simulationstechnik*, Informatik-Fachberichte 150, Springer, Berlin: 239-249.
- Vancso-Polacsek, K. 1990. "Theory and practice of computer assisted simulation and modeling on professional workstations." Ph.D. thesis No. 9104 Swiss Federal Institute of Technology Zürich (ETHZ).
- Wymore, A.W. 1984. "Theory of Systems". In *Handbook of Software Engineering*, Van Nostrand Reinhold Company, New York.
- Wirth, N.; Gutknecht, J.; Heiz, W.; Schär, H.; Seiler, H.; Vetterli, C. and Fischlin, A. 1992. *MacMETH. A fast Modula-2 language system for the Apple Macintosh. User Manual*. 4th. completely revised ed., Departement Informatik ETH Zürich, Switzerland.
- Zeigler, B.P. 1976. *Theory of modelling and simulation*. Wiley, New York a.o.
- Zeigler, B.P. 1979. "Multilevel multiformalism modeling: an ecosystem example." In *Theoretical Systems Ecology*, Academic Press, New York, 17-54.

RASS¹ 1.1 User's Guide

EPC-Version

Jürg Thöny

1 INTRODUCTION.....	1
2 DISTRIBUTION AND INSTALLATION.....	1
3 RUNNING MDPs WITH RASS.....	2
3.1 Developing MDPs.....	2
3.2 Transferring MDPs to the RASS host.....	2
3.3 Building executable simulation programs.....	2
3.4 Running executable simulation programs	3
4 TUTORIAL.....	4
4.1 Developing Sensitivity MDP	4
4.2 Transferring the MDP Sensitivity to the RASS host	5
4.3 Building the executable simulation program Sensitivity.....	6
4.4 Running the executable simulation program Sensitivity.....	7
5 TROUBLE SHOOTING	9
6 LITERATURE	10

¹ RAMSES Simulation Server

1 Introduction

This user's guide describes how to install and use RASS on Unix workstations using the EPC Modula-2 compiler.

This text assumes that you are familiar with ModelWorks (Fischlin *et al.*, 1994). It is highly recommended to read also (Fischlin, 1991) and (Thoeny *et al.*, 1994). You should also be able to work with Unix workstations.

RASS allows to run interactive ModelWorks Model Definition Programs [MDPs] in batch oriented environments like Unix workstations.

The generated results of a batch simulation should be the same as on the interactive RAMSES host, but you have to consider the following aspects:

1) The calculated results don't have to be numerical identical. This is due to the different floating point hardware on different hosts. Especially simulations with very small step-sizes may produce detectable differences. It is impossible to say on which host the quality of the results is better.

2) RASS ModelWorks is build on top of the Batch-DM (Thoeny *et al.*, 1994). No user interactions are possible. A RASS program will follow the default execution thread; it simulates default answers on every interactive DM component (i.e. pressing the enter key in every modal dialogue).

2 Distribution and installation

RASS consists of the libraries *libMWLib.a*, *libMWLib_x.a* and the shell scripts *MOD2mod*, *DEF2def*, *listdef*, and *RASSMakeMake*. You have to install the libraries in your linker search path and the shell scripts in the command search path of your shell. It is recommended to install the RASS library in `/usr/local/lib` and the shell scripts in `/usr/local/bin`. Please ask your system administrator to install these files.

If the installation in `/usr/local` isn't possible, you can also install RASS local to your home directory. After copying the file *libMWLib.a* to the desired location, you have to run:

```
ranlib PATH_OF_LIB/libMWLib.a PATH_OF_LIB/libMWLib_x.a
```

To allow the linker to find the library, you have to specify the location using the `LD_LIBRARY_PATH` environment variable. Please make sure, that the RASS shell scripts are in the command search path of your shell (consult the documentation of your favoured shell).

The EPC Modula-2 environment has to be installed on the host prior using RASS. This has to be done by your system administrator.

3 Running MDPs with RASS

A simulation program is usually developed in the interactive RAMSES environment. Then the Modula-2 source code and the input files are transferred to the RASS host. There you can build an executable simulation program to finally run your simulation experiments.

3.1 Developing MDPs

Usually a MDP is developed on an interactive RAMSES host, where you should also plan and test your simulation experiment. Beside the default strategy of the Batch-DM (default answers to dialogues), RASS will simulate exactly one user event for you:

If you have an experiment installed in your MDP, RASS will execute it, otherwise RASS will execute one simulation run.

Therefore, you can test your simulation program on the interactive RAMSES host prior transferring it to the RASS host. If you start your simulation respectively experiment and get the desired result by answering all eventual modal dialogues just by pressing the enter key, you can expect to get the same behaviour on the RASS host.

3.2 Transferring MDPs to the RASS host

A developed MDP consists of one or several Modula-2 source files and optional data files. They have to be transferred to the RASS host. Because most hosts have a different definition of text files, it is highly recommended to use FTP (File Transfer Protocol) to transfer files, since it translates them according to the specification of the target host.

The EPC Modula-2 compiler expects source files with the extension `.mod` and `.def`. All modules must have the name `Modulename.mod` or `Modulename.def` for definition modules. The naming is case sensitive. If you transfer files from the RAMSES environment, they are usually named using the extensions `.MOD` and `.DEF`. The two shell scripts `MOD2mod` and `DEF2def` performs this renaming for you. To use these scripts, change to the directory where your source files reside and call both scripts.

3.3 Building executable simulation programs

After you have transferred your source and data files to the RASS host, you can build an executable simulation program. This involves both, the compilation of the sources and the linking with the MW library. The shell script `RASSMakeMake` helps you with this task. Run `RASSMakeMake` in the directory where your MDP resides. It will generate a file named `Makefile`. The Unix tool `make` uses this file to perform the compilation and linking for you. Simply call `make` in the directory where your MDP and the generated `Makefile` reside. Note, that `RASSMakeMake` assumes, that the first (alphabetically) module without corresponding definition module is the main module.

It is possible that you get some compiler error messages. The reason is usually that Modula-2 is not defined exactly the same on various compiler implementations. Please read the report "Practical considerations on writing portable Modula-2 code" (Thoeny, 1994) to get more information about writing portable Modula-2 code.

If you change the source code, you have to call *make* again in order to bring your executable simulation program up do date.

You can use *make* with three optional parameters: *all*, *clean*, and *depend*. *make all* is the same as *make* without parameter. *make clean* will remove all object files (the result of the compilation process). *make depend* will add the inter module dependencies to the *Makefile*. This is useful, if you plan to change your definition modules on the RASS host.

3.4 Running executable simulation programs

The executable simulation program has the same name as the main module of your MDP. It can be started by typing its name.

If you have stored your data files in another location as the directory where you start your executable simulation program, you have to specify the path in the environment variable M2PATH. Consult the documentation of your favoured shell on how to set environment variables. The syntax of M2PATH is:

```
path[:path]
```

where [:path] stands for an optional repetition of :path. The pathes can be given relative to the start directory or to the root directory.

Examples using csh:

```
setenv M2PATH Datafiles
setenv M2PATH Datafiles1:Datafiles2
setenv M2PATH /home/users/goofy/RASS/Models/MyModel/Datafiles
```

If you have used DM library routines which are not yet implemented in the Batch-DM, a file named *RASS.NYI* will be created. *RASS.NYI* contains a list of all calls to those routines. Each line contains one entry. An entry can have two forms:

```
NotYetImplemented : RoutineName in ModuleName
FATAL NotYetImplemented : RoutineName in ModuleName
```

RASS.NYI contains any number of entries of the first form. If an entry of the second form exists, the program was aborted inside the listed routine. Note that the file *RASS.NYI* will only be generated if you have called at least one of the routines, which are not yet implemented. After execution of a simulation program, you should check the existence of *RASS.NYI*. If it exists, examine its content. In order to prevent the examination of an old *RASS.NYI*, it is recommended to remove or rename it prior to the next execution of a RASS program.

The filing of the MW library on RASS behaves the same as on the interactive RAMSES. As a result, you will only get a stash file if at least one monitoriable variable is activated for filing, or if you switch the filing on by calling *SimBase.SetDocumentRunAlwaysMode*.

4 Tutorial

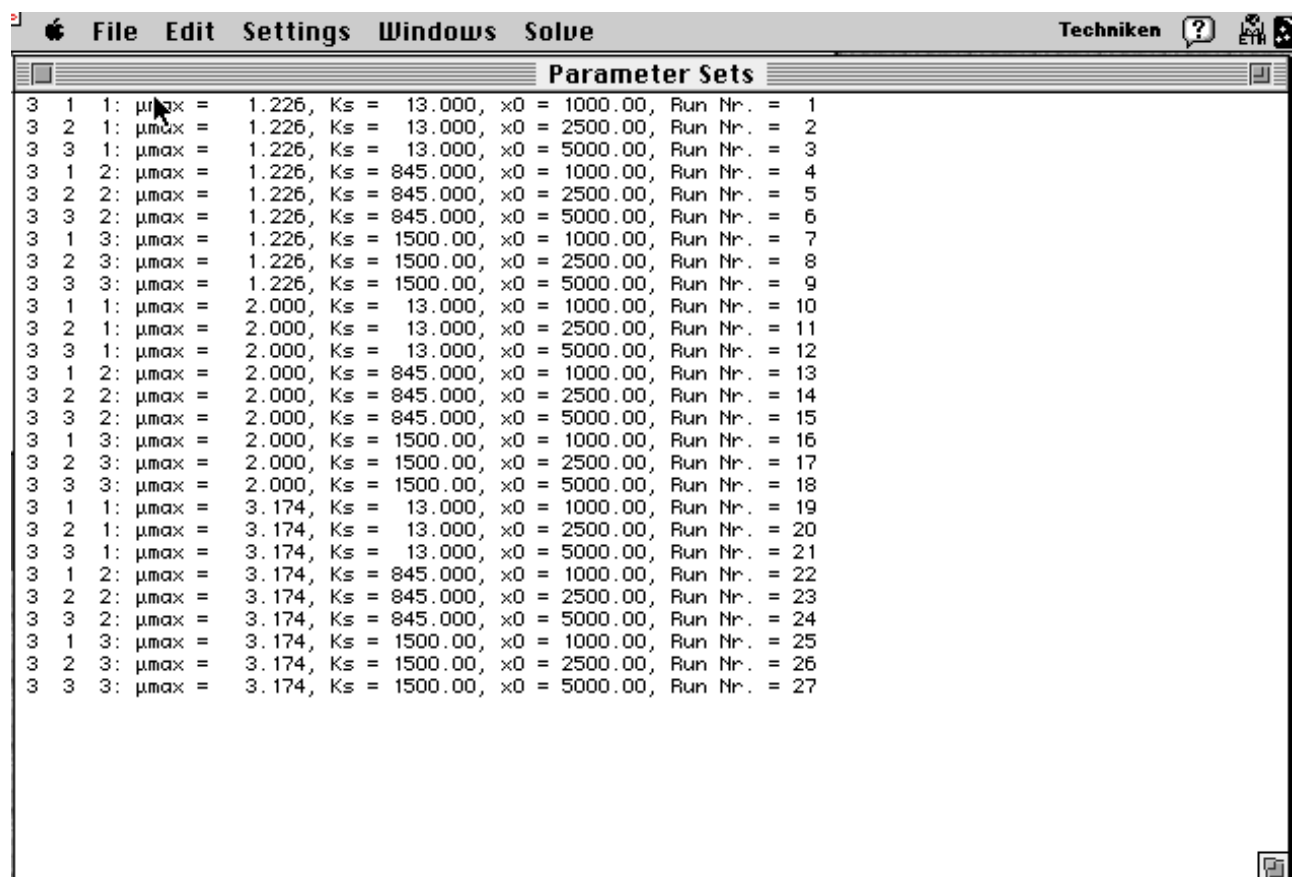
This tutorial guides you through a sample RAMSES to RASS transition. The transfer using FTP is not described here.

The chosen sample model is Sensitivity (File name Sensitivity.MOD) as described in (Fischlin *et al.*, 1994). It demonstrates a parameter sensitivity analysis of a Michaelis-Menten algae growth model. It creates an additional window, where it writes the chosen parameter sets.

At the begin, Sensitivity reads its model parameters from the data file Sensitivity.DAT.

4.1 Developing Sensitivity MDP

The model is already developed and produces the following output when ran from within RAMSES (On a Macintosh, after some window rearrangement):



Run Nr.	Parameter	μ_{max}	K_s	x_0	Run Nr.
3 1 1	μ_{max}	1.226	13.000	1000.00	1
3 2 1	μ_{max}	1.226	13.000	2500.00	2
3 3 1	μ_{max}	1.226	13.000	5000.00	3
3 1 2	μ_{max}	1.226	845.000	1000.00	4
3 2 2	μ_{max}	1.226	845.000	2500.00	5
3 3 2	μ_{max}	1.226	845.000	5000.00	6
3 1 3	μ_{max}	1.226	1500.00	1000.00	7
3 2 3	μ_{max}	1.226	1500.00	2500.00	8
3 3 3	μ_{max}	1.226	1500.00	5000.00	9
3 1 1	μ_{max}	2.000	13.000	1000.00	10
3 2 1	μ_{max}	2.000	13.000	2500.00	11
3 3 1	μ_{max}	2.000	13.000	5000.00	12
3 1 2	μ_{max}	2.000	845.000	1000.00	13
3 2 2	μ_{max}	2.000	845.000	2500.00	14
3 3 2	μ_{max}	2.000	845.000	5000.00	15
3 1 3	μ_{max}	2.000	1500.00	1000.00	16
3 2 3	μ_{max}	2.000	1500.00	2500.00	17
3 3 3	μ_{max}	2.000	1500.00	5000.00	18
3 1 1	μ_{max}	3.174	13.000	1000.00	19
3 2 1	μ_{max}	3.174	13.000	2500.00	20
3 3 1	μ_{max}	3.174	13.000	5000.00	21
3 1 2	μ_{max}	3.174	845.000	1000.00	22
3 2 2	μ_{max}	3.174	845.000	2500.00	23
3 3 2	μ_{max}	3.174	845.000	5000.00	24
3 1 3	μ_{max}	3.174	1500.00	1000.00	25
3 2 3	μ_{max}	3.174	1500.00	2500.00	26
3 3 3	μ_{max}	3.174	1500.00	5000.00	27

4.2 Transferring the MDP Sensitivity to the RASS host

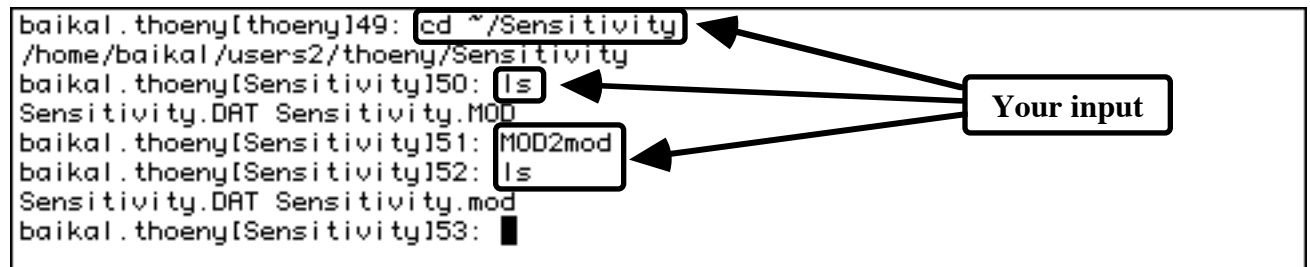
Create a directory named Sensitivity in your home directory on the RASS host:

```
mkdir ~/Sensitivity
```

Transfer Sensitivity.MOD and Sensitivity.DAT from your RAMSES host to the RASS host into this directory using FTP.

On the RASS host change to the directory Sensitivity, check for the presence of the MDP and data file, and call the shell script *MOD2mod*:

The output on the terminal should look like:



```
baikal.thoeny[thoeny]49: cd ~/Sensitivity
/home/baikal/users2/thoeny/Sensitivity
baikal.thoeny[Sensitivity]50: ls
Sensitivity.DAT Sensitivity.MOD
baikal.thoeny[Sensitivity]51: MOD2mod
baikal.thoeny[Sensitivity]52: ls
Sensitivity.DAT Sensitivity.mod
baikal.thoeny[Sensitivity]53: █
```

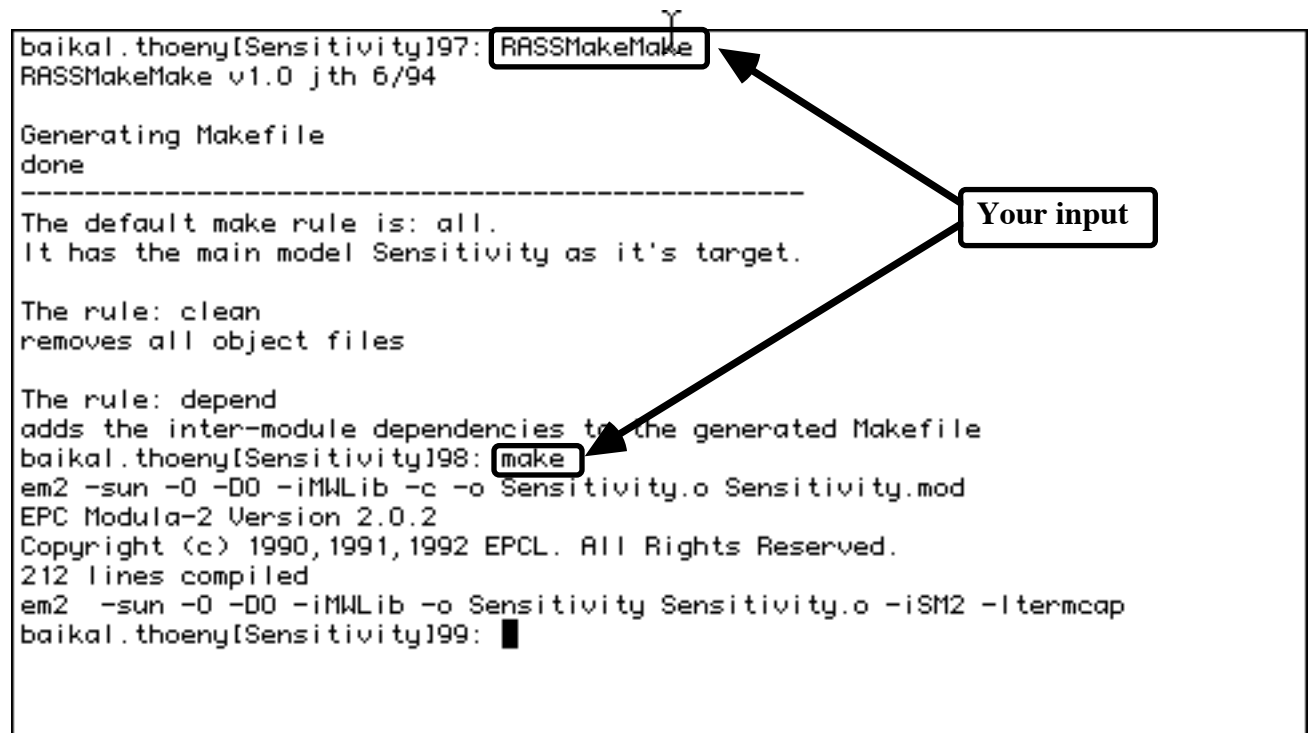
You don't have to call *DEF2def* because the Sensitivity MDP contains no definition modules.

4.3 Building the executable simulation program Sensitivity

The shell script RASSMakeMake generates the Makefile, which you can subsequently use to build Sensitivity. In the directory ~/Sensitivity use the commands:

```
RASSMakeMake  
make
```

The output on the terminal should look like:



```
baikal.thoeny[Sensitivity]197: RASSMakeMake  
RASSMakeMake v1.0 jth 6/94  
  
Generating Makefile  
done  
-----  
The default make rule is: all.  
It has the main model Sensitivity as it's target.  
  
The rule: clean  
removes all object files  
  
The rule: depend  
adds the inter-module dependencies to the generated Makefile  
baikal.thoeny[Sensitivity]198: make  
em2 -sun -O -DO -iMWLib -c -o Sensitivity.o Sensitivity.mod  
EPC Modula-2 Version 2.0.2  
Copyright (c) 1990,1991,1992 EPCL. All Rights Reserved.  
212 lines compiled  
em2 -sun -O -DO -iMWLib -o Sensitivity Sensitivity.o -iSM2 -ltermcap  
baikal.thoeny[Sensitivity]199: █
```

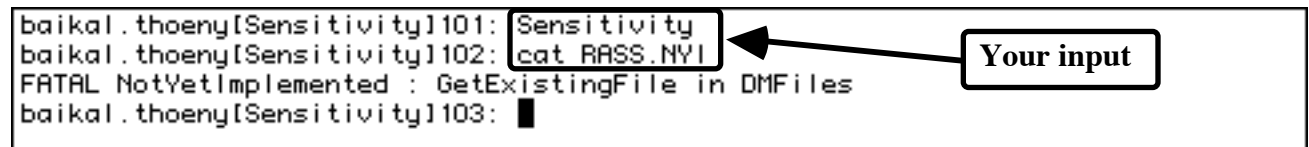
4.4 Running the executable simulation program Sensitivity

Now you can run Sensitivity which will be finished very fast. Then analyse the RASS.NYI:

```

baikal.thoeny[Sensitivity]101: Sensitivity
baikal.thoeny[Sensitivity]102: cat RASS.NYI
FATAL NotYetImplemented : GetExistingFile in DMFiles
baikal.thoeny[Sensitivity]103: █

```



Whoops! This transition went wrong. The best thing at this point is, to return to your RAMSES host and look at the MDP. You will find `GetExistingFile` twice; first in the import list,

```

FROM DMFiles IMPORT GetExistingFile, TextFile, GetReal, SkipGap,
  ReadChars, Close, Response;

```

and secondly inside the procedure `ReadAndSetParameters`.

```

GetExistingFile(parFile, 'Open parameter file "Sensitivity.DAT"');

```

As you can see, the call to `GetExistingFile` has no parameter for a default file name. Therefore, RASS was unable to continue and terminated your program. In order to prevent this, we replace `GetExistingFile` with `DMFiles.Lookup` in the import list,

```

FROM DMFiles IMPORT Lookup, TextFile, GetReal, SkipGap,
  ReadChars, Close, Response;

```

and inside the procedure `ReadAndSetParameters`.

```

Lookup(parFile, "Sensitivity.DAT", FALSE);

```

Now we can test our changes on RAMSES and transfer the `Sensitivity.MOD` again to the RASS host. There we have to rerun the shell script `MOD2mod`. We don't have to rerun `RASSMakeMake`, since the structure of the MDP hasn't changed.

Rerun `make`, which builds us finally the new executable simulation program of `Sensitivity`.

An other run of Sensitivity gives us the following output, which happens to be exactly the same as on RAMSES:

```

baikal.thoeny[Sensitivity]124: Sensitivity
3 1 1:  $\mu_{max}$  = 1.226, Ks = 13.000, x0 = 1000.00, Run Nr. = 1
3 2 1:  $\mu_{max}$  = 1.226, Ks = 13.000, x0 = 2500.00, Run Nr. = 2
3 3 1:  $\mu_{max}$  = 1.226, Ks = 13.000, x0 = 5000.00, Run Nr. = 3
3 1 2:  $\mu_{max}$  = 1.226, Ks = 845.000, x0 = 1000.00, Run Nr. = 4
3 2 2:  $\mu_{max}$  = 1.226, Ks = 845.000, x0 = 2500.00, Run Nr. = 5
3 3 2:  $\mu_{max}$  = 1.226, Ks = 845.000, x0 = 5000.00, Run Nr. = 6
3 1 3:  $\mu_{max}$  = 1.226, Ks = 1500.00, x0 = 1000.00, Run Nr. = 7
3 2 3:  $\mu_{max}$  = 1.226, Ks = 1500.00, x0 = 2500.00, Run Nr. = 8
3 3 3:  $\mu_{max}$  = 1.226, Ks = 1500.00, x0 = 5000.00, Run Nr. = 9
3 1 1:  $\mu_{max}$  = 2.000, Ks = 13.000, x0 = 1000.00, Run Nr. = 10
3 2 1:  $\mu_{max}$  = 2.000, Ks = 13.000, x0 = 2500.00, Run Nr. = 11
3 3 1:  $\mu_{max}$  = 2.000, Ks = 13.000, x0 = 5000.00, Run Nr. = 12
3 1 2:  $\mu_{max}$  = 2.000, Ks = 845.000, x0 = 1000.00, Run Nr. = 13
3 2 2:  $\mu_{max}$  = 2.000, Ks = 845.000, x0 = 2500.00, Run Nr. = 14
3 3 2:  $\mu_{max}$  = 2.000, Ks = 845.000, x0 = 5000.00, Run Nr. = 15
3 1 3:  $\mu_{max}$  = 2.000, Ks = 1500.00, x0 = 1000.00, Run Nr. = 16
3 2 3:  $\mu_{max}$  = 2.000, Ks = 1500.00, x0 = 2500.00, Run Nr. = 17
3 3 3:  $\mu_{max}$  = 2.000, Ks = 1500.00, x0 = 5000.00, Run Nr. = 18
3 1 1:  $\mu_{max}$  = 3.174, Ks = 13.000, x0 = 1000.00, Run Nr. = 19
3 2 1:  $\mu_{max}$  = 3.174, Ks = 13.000, x0 = 2500.00, Run Nr. = 20
3 3 1:  $\mu_{max}$  = 3.174, Ks = 13.000, x0 = 5000.00, Run Nr. = 21
3 1 2:  $\mu_{max}$  = 3.174, Ks = 845.000, x0 = 1000.00, Run Nr. = 22
3 2 2:  $\mu_{max}$  = 3.174, Ks = 845.000, x0 = 2500.00, Run Nr. = 23
3 3 2:  $\mu_{max}$  = 3.174, Ks = 845.000, x0 = 5000.00, Run Nr. = 24
3 1 3:  $\mu_{max}$  = 3.174, Ks = 1500.00, x0 = 1000.00, Run Nr. = 25
3 2 3:  $\mu_{max}$  = 3.174, Ks = 1500.00, x0 = 2500.00, Run Nr. = 26
3 3 3:  $\mu_{max}$  = 3.174, Ks = 1500.00, x0 = 5000.00, Run Nr. = 27
baikal.thoeny[Sensitivity]125:

```

RASS.NYI contains now:

NotYetImplemented : CreateWindow in DMWindows

Since the batch-DM was designed for non interactive usage, it is unable to create a window, but it redirects the text written to a window to the standard output file.

5 Trouble shooting

Symptom	Possible solutions
The linker doesn't find libMWLib.a or libMWLib_x.a	<p>The libraries are not in the linker search path. Eventually ask the local system administrator to put it into an accessible place.</p> <p>If the libraries are not in the linker search path (i.e. local to your home directory), use the environment variable LD_LIBRARY_PATH to tell the linker where it can find the library.</p>
The linker complains about an "out of date" symbol table of libMWLib.a.	Use ranlib to bring the symbol table up to date.
The executable simulation program doesn't find my data files.	<p>The file names are case sensitive on Unix systems. You may have to rename some files.</p> <p>If your data files aren't in the start-up directory, specify the search path with the environment variable M2PATH.</p>
A MDP file consists only of one line on my Unix workstation.	You should transfer all your MDPs using FTP in text (ASCII) mode.
I'm using Fetch as the FTP Client on my Macintosh computer, and it doesn't allow me to transfer my RAMSES MDPs in text mode.	<p>RAMSES on Macintosh computers sets the type of the MDPs to 'MoTx'. Fetch currently only allows the transfer of files with the type 'TEXT' in text mode. Use an other FTP client (e.g. XferIt sfalken@apple.com)</p> <p>It is also possible to set the preferences of the RAMSES shell (Modeling session, programming session, MiniShell) to "export mode". Thereafter the RAMSES shell will set the type of the touched MDPs to 'TEXT'.</p>

6 Literature

Fischlin, A., 1991. *Interactive modeling and simulation of environmental systems on workstations*. In: Richter, D.P.F.M.a.O. (eds.), *Dynamic Systems in Medicine, Biology, and Ecology*, Berlin: Springer, 131-145.

Fischlin, A. *et al.*, 1994. *ModelWorks 2.2: An Interactive Simulation Environment for Personal Computers and Workstations*. Internal Report # 14, Systems Ecology, ETHZ, .

Thoeny, J., 1994. *Practical considerations on portable Modula 2 code*. Internal Report # 20, Systems Ecology Group, ETHZ, .

Thoeny, J., Fischlin, A. & Gyalistras, D., 1994. *RASS: Towards bridging the gap between interactive and off-line simulations*. In: J. Halin, W.K.a.R.R. (ed.), *CISS - First Joint Conference of International Simulation Societies*, Zuerich, Switzerland, The Society for Computer Simulation International, P.O. Box 17900, San Diego, Cal. 92177, USA, p.^pp. 99-103.

RASS¹ 1.1 Developer's Guide

Jürg Thöny

1 LIBRARY ARCHITECTURE	1
2 MODULE DESCRIPTIONS.....	2
2.1 Portability Layer	2
2.1.1 SysDep.....	2
2.1.2 Portab	6
2.2 DM-Modules	6
2.2.1 DMConversions	6
2.2.2 DMFiles	6
2.2.3 DMMaster.....	7
2.2.4 DMClock.....	7
2.2.5 DMMathLib	7
2.2.6 DMMessages	7
2.2.7 DMStorage.....	7
2.2.8 DMStrings.....	7
2.2.9 DMSystem	7
2.2.10 DMWindowIO.....	7
2.2.11 DMWindows & DM2DGraphs.....	8
2.2.12 DMEditFields & DMEnterForms.....	8
2.2.13 DMMenus.....	8
2.3 MW Modules.....	8
2.4 AuxLib Modules	8
3 EPC RASS DEVELOPMENT ENVIRONMENT AND TECHNIQUES.....	9
3.1 Libraries	9
3.2 Makefiles	11
3.3 Tools.....	14
4 FUTURE DEVELOPMENT OF RASS.....	16
5 LITERATURE	16

¹ RAMSES Simulation Server

1 Library Architecture

The base architecture of RASS is described in Thoeny (1994).

	0	1	2	3	4	5	7	6	25	24	28	27	26	44	40	41	42	43	29	11	22	21	30	31	32	33	10	9	18	34	35	38	12	13	14							
0	O	X	X	X	X	X	X	X																																		
1		O					X													X								X	X						X							
2			O			X	X		X	X																										X						
3				O		X		X	X																												X					
4					O	X													X				X				X	X	X							X						
5						O	X								X				X									X	X							X						
7						O	X							X	X	X	X	X	X	X				X			X	X	X							X						
6							O							X	X				X								X	X					X	X			X					
25								O	X										X										X								X					
24									O	X	X	X																														
28										O	X	X																										X				
27											O	X																										X				
26												O																										X				
44													O	X		X	X	X						X		X	X	X				X	X				X					
40														O	X	X		X	X				X	X		X	X	X									X					
41															O	X	X	X						X		X	X	X										X				
42																O								X		X												X				
43																	O	X	X					X		X	X	X		X					X			X				
29																	O	X						X	X		X	X	X		X							X				
11																		O	X	X							X	X										X				
22																			O	X								X	X									X				
21																				O									X									X				
30																								O																		
31																									O	X	X	X	X	X	X	X	X	X	X	X	X	X				
32																										O													X			
33																												O	X	X		X							X			
10																													O	X	X								X			
9																													O										X			
18																																							X			
34																																							X			
35																																							X			
38																																							X	X	X	X
12																																							O	X		
13																																							O	X		
14																																								O		

Table 1: Current RASS MW dependency matrix generated by HierSRC.mod. Note that the DM is combined into one entry and the imports of SysDep are omitted.

- | | | |
|-----------------|------------------|---------------|
| 0 ImportMWHHigh | 26 MatBase | 32 JumpTab |
| 1 SimIntegrate | 44 MWFFunctions | 33 MWDocProcs |
| 2 SimDeltaCalc | 40 MWRunTimeSys | 10 MWVars |
| 3 SimGraphUtils | 41 MWRTCHandlers | 9 MWTypes |
| 4 SimEvents | 42 MWMonitoring | 18 MWErrors |
| 5 SimObjects | 43 MWDefaults | 34 MWDocUtils |
| 7 SimBase | 29 MWObjects | 35 TFBBase |
| 6 SimMaster | 11 MWSimLib | 38 DM Lib |
| 25 MWSGUBase | 22 MWSimLibAux0 | 12 RASSNotYet |
| 24 Matrices | 21 MWSimLBase | 13 SysDep |
| 28 MatShape | 30 MWEvtBase | 14 Portab |
| 27 MatCopy | 31 MWFiling | |

2 Module descriptions

2.1 Portability Layer

2.1.1 SYSDEP

SysDep encapsulates the interface to the local System. It's implementation module has to be reimplemented on every new RASS host. The interface is the following:

```

(* file IO *)
CONST
  SDBNullDevice      = -1;
  SDStdInput         = 0;
  SDStdOutput        = 1;
  SDStdError         = 2;
VAR
  SDEol              : CHAR;          (* readOnly variable *)
  SDDirSeparator     : CHAR;          (* readOnly variable *)
  SDM2PathEnvVarName : ARRAY[0..10] OF CHAR; (* readOnly variable *)

TYPE
  SDFile              = INTEGER;
  SDFileResult        = (SDDone, SDFileNotFound, SDTooManyFiles,
                        SDNoWriteAccess, SDNoReadAccess,
                        SDFilenameWrong,
                        SDOtherResult);

PROCEDURE SDLookup (VAR file : SDFile; fileName: ARRAY OF CHAR;
                   readOnly, new, binary : BOOLEAN);
PROCEDURE SDDelete (fileName      : ARRAY OF CHAR);
PROCEDURE SDRename (oldName, newName : ARRAY OF CHAR);
PROCEDURE SDClose (VAR file : SDFile);
PROCEDURE SDFlush (file : SDFile);
PROCEDURE SDEOF (file : SDFile) : BOOLEAN;
PROCEDURE SDTruncateFile(file : SDFile);
PROCEDURE SDFileRead (file : SDFile; buffer: ADDRESS; size: LONGCARD): LONGCARD;
PROCEDURE SDFileWrite (file : SDFile; buffer: ADDRESS; size: LONGCARD): LONGCARD;
PROCEDURE SDGetFileSize (file : SDFile) : LONGCARD;
PROCEDURE SDGetFilePos (file : SDFile) : LONGCARD;
PROCEDURE SDSetFilePos (file : SDFile; pos : LONGCARD);
PROCEDURE SDLastFileResult () : SDFileResult;

```

Three standard files are defined: SDStdInput, SDStdOutput and SDStdError. A undefined file is identified by SDBNullDevice. Every operation on the SDBNullDevice file has no effect at all.

An end of line in a text file is defined by SDEol on interface level. SysDep will translate them to the local usage (i.e. line-feed on Unix, carriage-return on a Mac, or carriage-return/line-feed on a MS-DOS PC). The client of SysDep has to use SDEol in order to get the desired result.

SDDirSeparator exports the character which is used by the operating-system to separate directory levels (i.e. ":" on a Mac, "/" on Unix, or "\n" on a MS-DOS PC).

SDPathEnvName contains the name of the environment variable which identifies the search path for M2 files. It is used by the Lookup procedure of DMFiles. It is set to "PATH" on the Mac implementation (denotes the PATH section of the User.Profile), and to "M2PATH" on the SUN implementation.

SDFile is the type of the file variables which can be used by the procedures from SysDep which operates on files.

Every file procedure of SysDep generates a result of the type SDFileResult. The result of the last file operation can be requested by the function SDLastFileResult.

SDLookup opens a file with name *fileName* and access mode *readOnly*, *new* and *binary*. It returns the file identification in the VAR parameter *file*. If *readOnly* is TRUE, the file access is opened read only. If *readOnly* is FALSE, the file has read and write access. If *new* is TRUE, a new file will be created. If a file named *fileName* already exists, it will be truncated to length zero. If *new* is FALSE, the file must exist to complete the operation successfully.

SDelete deletes the file *fileName*, SDRename renames the file *oldName* to *newName*, and SDClose closes the file, which is identified by *file*. SDFlush flushes the file buffer of the operating system.

SDEOF returns TRUE if the access pointer has reached the end of file. SDTruncateFile truncates the file at the current file position.

SDFileRead/Write reads/writes the content of *buffer* with *size*. The number of bytes actually read/written are returned as the function value.

SDGetFileSize returns the size of file in bytes. SDGetFilePos returns the current position in the file. SDSetsFilePos sets the current position to *pos*.

```
(* dynamic memory i.e. Heap *)
PROCEDURE SDAlloc (VAR a : ADDRESS; byteCount: LONGCARD);
PROCEDURE SDDealloc (VAR a : ADDRESS);
PROCEDURE SDValidPtr(a : ADDRESS) : BOOLEAN;
```

SDAlloc allocates a heap-block of size *byteCount*. The heap-block is aligned to Portab.MaxAlign. SDDealloc deallocates a previous allocated heap-block. SDValidPtr returns TRUE if *a* might be a valid address.

```

(* Time *)
CONST
  SDJan = 1; SDFeb = 2; SDMar = 3; SDApr = 4;  SDMai = 5;  SDJun = 6;
  SDJul = 7; SDAug = 8; SDSep = 9; SDOct = 10; SDNov = 11; SDDec = 12;
  SDSun = 1; SDMon = 2; SDTue = 3; SDWed = 4;  SDThu = 5;  SDFri = 6; SDSat = 7;
(* hour is always within range [0..23] *)

TYPE
  SDTime;
  SDDate = RECORD
    year, month, day,
    hour, minute, second, sec100,
    dayOfWeek          : INTEGER;
  END; (*RECORD*)

PROCEDURE SDGetTime    (VAR time : SDTime);
PROCEDURE SDTimeToDate (   time : SDTime; VAR date : SDDate);
PROCEDURE SDDateToTime (   date : SDDate; VAR time : SDTime);

```

SysDep encapsulates the time in the opaque type `SDTime`. `SDGetTime` returns the current system time in the parameter *time*. `SDTimeToDate/SDDateToTime` are routines to convert between `SDTime` and `SDDate`.

```

(* Environment *)
PROCEDURE SDGetArgCount  (): CARDINAL;
PROCEDURE SDGetArgument (argNum : CARDINAL; VAR argument: ARRAY OF CHAR);
PROCEDURE SDGetEnvVar   (varName : ARRAY OF CHAR; VAR varValue : ARRAY OF CHAR);
PROCEDURE SDGetEnvVarListElem(   varName : ARRAY OF CHAR;
                                index   : INTEGER;
                                VAR elemValue : ARRAY OF CHAR);

```

Program arguments are only supported on systems which supports them. `SDGetArgCount` returns the number of arguments. A specific argument can be retrieved by `SDGetArgument`.

The value of an environment variable can be accessed by `SDGetEnvVar`. If the environment variable with name *varName* doesn't exist, a string with length zero is returned. On the Mac implementation `SDGetEnvVar` returns the content of the section *varName* of the `User.Profile`. Since an environment variable can contain a list (i.e. the `PATH`) and the divider of list elements is system dependent, a procedure `SDGetEnvVarListElem` is provided.

```

        (* Termination *)
PROCEDURE SDSetTermProc (    p : PROC; VAR done : BOOLEAN);
PROCEDURE SDExit          (status: INTEGER);
PROCEDURE SDExitMsg      (status: INTEGER; module, procedure, reason : ARRAY OF CHAR);
    
```

SDSetTermProc allows to add cleanup procedures. On termination of a program, they are called in the reverse order of their installation.

SDExit terminates a program with the status code *status*. SDExitMsg writes also a termination message to SDStdError.

```

        (* Interprocessing *)
TYPE
    SDLocalMsgID   = INTEGER;
    SDLocalMsgProc = PROCEDURE(ADDRESS);
PROCEDURE SDSetLocalMsgHandler(    msgID : SDLocalMsgID;    handler : SDLocalMsgProc;
                                VAR done : BOOLEAN);
PROCEDURE SDGetLocalMsgHandler(    msgID : SDLocalMsgID; VAR handler : SDLocalMsgProc);
PROCEDURE SDSendLocalMessage (msgID : SDLocalMsgID; msg : ADDRESS);

(* channels are not designed yet. Idea: a channel is a connection between two host
   consisting of two two-way streams:
       command stream: text
       data stream:   binary data with translation facility
*)
    
```

A message handler is a procedure with an address parameter. Many message handlers can be installed by SDSetLocalMsgHandler. SDGetLocalMsgHandler retrieves the message handler with the identification *msgID*. A message handler can be triggered to send the message *msg* by calling SDSendLocalMessage. This mechanism can be used to implement the DialogMachineTask procedure (e.g. calling GetNextEvent in the Mac implementation to enable the co-operative multitasking).

2.1.2 PORTAB

Portab exports a set of base types which differ among Modula-2 implementations. This supports the programming of portable Modula-2 code. Main clients of Portab are DMConversions and SysDep.

```

TYPE

CARD16 = SHORTCARD; (* SUN EPC Modula 2 *)
CARD32 = CARDINAL;  (* SUN EPC Modula 2 *)
CARDADR = CARDINAL; (* SUN EPC Modula 2 *)
INT16  = SHORTINT;  (* SUN EPC Modula 2 *)
INT32  = INTEGER;   (* SUN EPC Modula 2 *)

(*
CARD16 = CARDINAL;  (* MacMETH *)
CARD32 = LONGCARD;  (* MacMETH *)
CARDADR = LONGCARD; (* MacMETH *)
INT16  = INTEGER;   (* MacMETH *)
INT32  = LONGINT;   (* MacMETH *)
*)
VAR
  MaxAlign : INTEGER; (* the biggest alignment for the base types *)

```

Typically the definition module of Portab has to be redefined for every new RASS implementation. I recommend to define the whole type-set anew and to keep the other sets for documentary purpose as comment.

MaxAlign is the maximum alignment boundary of all the base types (i.e. LONGINT/LONGREAL). MaxAlign is calculated by standard Modula-2 code and is not guaranteed to be correct. Please check its actual value when you encounter strange behaviour with dynamic data (i.e. wrong calculations).

2.2 DM-Modules

For a description of the DM interface see the according DM literature (Fischlin, 1986; Fischlin *et al.*, 1987; Fischlin & Schaufelberger, 1987; Keller, 1989). This documentation describes only the special Batch-DM implementation details. The concept of the Batch-DM is described in (Thoeny *et al.*, 1994).

2.2.1 DMCONVERSIONS

The module DMConversions is implemented using only standard Modula-2 code; no calls to system software are made. It is therefore fully portable. However, its efficiency could be improved.

2.2.2 DMFILES

DMFiles uses SysDep.SDPathEnvVarName as the search path for Lookup. The single path components are extracted using SysDep.SDGetEnvVarListElem. This provides a portability between various RASS platforms.

2.2.3 DMMASTER

Most of the procedures and functions of DMMaster are not yet implemented. The procedure DialogMachineTask calls the local message handler zero from SysDep. Therefore RASS could be easily extended by adding a handling of DM-Events.

2.2.4 DMCLOCK

Only the procedures Now and Today are implemented.

2.2.5 DMMATHLIB

The DMMathLib is implemented with direct calls to the C-Math library.

2.2.6 DMMESSAGES

The messages are written to the SDStdOutput. Since the writing is done using a file variable, it is easy to change it to another device. Abort writes the three paragraph parameters to SDStdOutput and calls SDExit with exit status one.

2.2.7 DMSSTORAGE

DMStorage is fully implemented, but the levels are ignored.

2.2.8 DMSTRINGS

All procedures which are string resource operations are not implemented.

2.2.9 DMSYSTEM

The DMLevel is forced to the DMSystem.startupLevel. All ScreenSize and color related procedures return zero. The ComputerSystem, Keyboard, and CPU are reported as unknown. FPU is FALSE. The SystemVersion and RomVersion are reported as unknown. InitProcs are not supported. The installed TermProcs are called only once - at program termination.

2.2.10 DMWINDOWIO

All write operations are redirected to SDStdOutput. Since the writing is done using a file variable, it is easy to change it to another device. SetPos tries to achieve the desired position in X-coordinates. If the desired column is smaller than the current X-coordinate then a new line is inserted. All graphical output related procedures are not supported. All mouse position procedures report the position (0,0).

Some WriteLongXYZ procedures are not implemented yet.

2.2.11 DMWINDOWS & DM2DGRAPHS

All DMWindows and DM2DGraphs procedures are not implemented. Later implementations should maintain the window data structures internally to support programs which rely on the internal state of these structures.

2.2.12 DMEDITFIELDS & DMENTRYFORMS

All procedures are not implemented yet. This gives the default behaviour.

2.2.13 DMMENUS

The menus data structures are maintained internally but the menus are not shown on the screen. This was necessary, because many clients request the internal state of a menu (i.e. using MenuExists, CommandExists, IsCommandChecked) to maintain the program state.

2.3 MW Modules

The interface is exactly the same as that of the interactive Version 2.2. For a description see the according MW literature (Fischlin *et al.*, 1994). The basis was the MW 2.2 library. All interactive components have been commented out. As a consequence of this procedure, the curve attributes are not defined and can not be written to the stash files. The default attributes are written instead (i.e. autoDefCStr, autoDefSStr). This allows to open the stash files from within the PostAnalysis session.

The only change in the program control is the procedure RunSimEnvironment. If an experiment is installed, it calls SimExperiment instead of calling RunDialogMachine. If no experiment is installed it calls SimRun.

2.4 AuxLib Modules

The core of the AuxLib modules were ported and included in the RASS library.

The Matrix package is only available in a old version (O. Roth's version 1991). The new (af) version is right now not portable from MacMETH to an other Modula-2 compiler.

The body of the DisplayArray procedure of StochStat is commented out. The reason is, that the implementation tries to work with simulated open arrays (defined as a POINTER TO ARRAY[0..8000] OF REAL), which leads to a page fault on protected memory systems.

3 EPC RASS Development environment and techniques

This chapter describes the RASS development environment on a Unix Workstation using the EPC Modula-2 compiler. The RASS development environment is organised in a hierarchical tree of directories. On the top-level are: SysDep, AuxLibDev, DMLibDev, MWLibDev, tools, and Models.

DMLibDev is subdivided into CoreHigh and OptHigh. All modules contained in CoreHigh are referenced by the MW library. The modules in OptHigh are modules which are referenced by many MW simulation models. Note, that this doesn't reflect the interactive DM's kernel and optional module structure. (cfg. Table 2).

MWLibDev is subdivided into High.MOD and Base.MOD, which reflects the organisation of the Mac development environment.

The RASS MW library contains all modules (object and definition) of the mentioned hierarchy.

The directory tools contains shell scripts which are of use for the user of RASS.

The directory Models contains some sample MDPs, which are included for testing purposes.

3.1 Libraries

The Unix tool *ar* is used to create the libraries.

Since *ar* is restricted to filenames with a maximum of 15 characters (including "." and extension), special care has to be taken for files with longer names. All editing is done on the files with the original name length. They are copied to files with names of 15 characters prior compilation or library archiving. To enable the EPC compiler to find the real names, a special file named `__ALIASES` has to be included in an archive. This file contains aliases for module names in the form: `ModuleName:ShortName` (cfg. Table 2).

Library	Modules	Aliases
SysDep	SysDep Portab RASSNotYet	NONE
DMCore	DMFiles DMConversions DMMaster DMStrings DMSystem DMWindowIO DMWindows DMClock DMStorage DMMessages DMMathLib	DMConversions:DMConvers
DMOpt	DM2DGraphs DMSyntaxForms DMEditFields DMMenus	DMEditFields:DMeditF DMSyntaxForms:DMSyntaxF
AuxLib	TabFunc TFTypesA TFBase TFDocProcs TFEdit TFFuncs TFMenus Matrices JumpTab Histograms StochStat MathProcs RandGen Lists MultiNormal Jacobi RandNormal ReadData MatBase MatCopy MatShape RandGen0	TFTypesAndVars:TFTypesA
MWBase	MWDefaults MWFunctions MWSimLBase MWDocProcs MWMonitoring MWSimLib MWDocUtils MWObjects MWSimLibA0 MWErrors MWRTCHandlers MWSimLibA1 MWEvtBase MWRunTimeSys MWTypes MWFiling MWSGUBase MWVars	MWSimLibAux0:MWSimLibA0 MWSimLibAux1:MWSimLibA1 MWRunTimeSys:MWRunTimeS MWRTCHandlers:MWRTCHand MWMonitoring:MWMonitori
MWHigh	SimBase SimGraphUtils SimObjects SimDeltaCalc SimIntegrate SimEvents SimMaster	SimGraphUtils:SimGraphU SimIntegrate:SimInte SimDeltaCalc:SimDeltaC

Table 2: All RASS libraries with the corresponding modules and the aliases. The library files are named libXXX.a and libXXX_x.b (where XXX is the name of the library). The libXXX.a files contain all object files and the libXXX_x.a files all definition modules and the __.ALIASES file.

All RASS libraries are combined into *libMWLib.a* and *libMWLib_x.a*. These two files should be installed in /usr/local/lib. This location is in the *ld* search path and can be accessed by anybody on our SUN-Network. Note that *ranlib* has to be called every time the libraries are copied.

3.2 Makefiles

The Makefiles are structured in the same fashion as the RASS directories. The main Makefile is in the root directory of the RASS development environment. It is not possible to use any Makefile in the sub-directories directly, they are to be called from within the main Makefile. All Makefiles contain three main rules:

1. all: this is the first rule, and therefore also the standard rule. It has the library of the directory where the Makefile resides as it's target. It generate also some basic library test programs.
2. clean: this rule deletes all object and library files.
3. depend: this rule adds dependency rules to the end of the Makefile.

```
depend :
    -csh -c 'setenv M2FLAGS "$(M2FLAGS)"; listdepf > makedep'
    echo '/^# DEPENDS/a' > eddep
    echo >> eddep
    echo '.' >> eddep
    echo '.,$$d' >> eddep
    echo '$$r makedep' >> eddep
    echo 'w' >> eddep
    -ed - Makefile < eddep
    rm -f eddep makedep

# DEPENDS -- make depend needs this line
```

It is a csh script which uses *listdepf* (see Tools) to generate a list of dependency rules. Then it uses *ed* to edit the Makefile (i.e. to find the line with "# DEPENDS", to delete the rest of the file, and to add the dependencies). Finally it removes the temporary files makedep and eddep, which have been generated during this procedure.

Main Makefile:

```
M2GLOBALFLAGS = -sun -D0 -nobounds -norange
RASSHOME=$(PWD)
```

Definition of the flags for all calls to the driver program of the EPC Modula-2 compiler. The switch *-sun* activates the compiler's SUN mode, and tells the linker to use the SUN library. Note, that the switch *-sun* works also with the EPC Modula-2 compiler on the IBM AIX machines. The home of the development environment is set to the value of the environment variable *PWD* (this is the string of the current working directory).

```

all :
  (cd SysDep; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)
  (cd DMLibDev/OptHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)
  (cd DMLibDev/CoreHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)
  (cd AuxLibDev; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)
  (cd MWLibDev/Base.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)
  (cd MWLibDev/High.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" all)

clean :
  (cd SysDep; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" clean)
  (cd DMLibDev/OptHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)"
   clean)
  (cd DMLibDev/CoreHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)"
   clean)
  (cd AuxLibDev; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" clean)
  (cd MWLibDev/Base.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" clean)
  (cd MWLibDev/High.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" clean)
  rm -f *.o *.a

depend :
  (cd SysDep; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" depend)
  (cd DMLibDev/OptHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)"
   depend)
  (cd DMLibDev/CoreHigh.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)"
   depend)
  (cd AuxLibDev; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" depend)
  (cd MWLibDev/Base.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" depend)
  (cd MWLibDev/High.MOD; make M2GLOBALFLAGS="$(M2GLOBALFLAGS)" RASSHOME="$(RASSHOME)" depend)

```

The rules all, clean and depend call the corresponding Makefiles rules in the RASS hierarchy.

```

mwlib :
echo -n > __.ALIASES
rm -f libMWLib.a
-sh -c 'mkdir tmpLib; cd tmpLib;\
    for f in ../../lib*[*x].a ../../lib*[*x].a;\
    do ar x $$f;\
    rm -f __.*;\
    ar cur ../libMWLib.a *;\
    rm -f *;\
    done;\
    cd ..; rmdir tmpLib;\
    ranlib libMWLib.a'

mwlib_x :
echo -n > __.ALIASES
rm -f libMWLib_x.a
-sh -c 'mkdir tmpLib; cd tmpLib;\
    for f in ../../lib*_x.a ../../lib*_x.a;\
    do ar x $$f;\
    if [ -f __.ALIASES ]; then\
        cat -s __.ALIASES >> ../__.ALIASES;\
    fi;\
    rm -f __.*;\
    ar cur ../libMWLib_x.a *;\
    rm -f *;\
    done;\
    cd ..; rmdir tmpLib;\
    ar ur libMWLib_x.a __.ALIASES; ranlib libMWLib_x.a'

```

The rules mwlib and mwlib_x create the RASS MW object library resp. definition library, which contain all libraries of RASS.

3.3 Tools

The rule depend of the RASS Makefiles call a shell script named *listdepf*.

```
#!/bin/csh
foreach file (*.mod)
  set obj=`basename $file .mod`.o
  em2 -twy $M2FLAGS $file | \
  sed /is\ found\ in/\!d | \
  sed "s/.*is found in.\\(.*\\)/"$obj": \1/" | \
  uniq
end
```

listdepf uses the EPC Modula-2 compiler to get a list of dependencies for all *.mod files in the current directory. If RASS has to be ported to another Modula-2 compiler, *listdepf* has to be rewritten. The Modula-2 program GetImports.mod (also in the tools directory) could be used within a new version of *listdepf*.

The shell scripts *MOD2mod* and *DEF2def* rename all .MOD files in the current directory to .mod (resp. .DEF to .def).

RASSMakeMake is a tool which helps the user of RASS to generate a skeleton Makefile for his MDPs. Only one MDP is supported per directory.

```
#!/bin/sh
echo "RASSMakeMake v1.1 jth 12/94"
#
# we don't overwrite an existing Makefile
echo
if [ -f Makefile ]; then
  echo "The Makefile already exist. Please remove or rename it first"
  exit
fi
#
# generate a list of the object files in OBMS and assume, that
# the first module without a definition module is the main module
# and therefore the target.
OBMS=""
mainmod=`basename "$1" .mod`
for f in *.mod ; do
  OBMS="$OBMS `basename $f .mod`.o"
  if [ ! -f `basename $f .mod`.def ]; then
    if [ "$mainmod" = "" ]; then
      mainmod=`basename $f .mod`
    fi
  fi
done
export OBMS
export mainmod
echo "Generating Makefile"
#
# put the content of the Makefile to stdout and redirect it into a file
```

```

# called Makefile (why not)
{
sed "s/^X//" << TEXT_STOP
X
XM2CFLAGS = -sun -D0 -iMWLib_x
XM2LFLAGS = -sun -D0 -lMWLib
X
X.SUFFIXES: .o .mod
X
X.mod.o:
X      em2 $(M2CFLAGS) -c -o \$.o \$.mod
X
TEXT_STOP
echo -n "OBMS = $OBMS"
echo
echo
echo "all : $mainmod"
echo
echo "$mainmod: \$(OBMS) Makefile"
echo "  em2 \$(M2LFLAGS) -o $mainmod \$(OBMS) -iSM2 -ltermcap"
echo
sed "s/^X//" << TEXT_STOP
Xclean :
X      rm -f *.o *.a
X
Xdepend :
X      -csh -c 'setenv M2FLAGS "$(M2CFLAGS)"; listdepf > makedep'
X      echo '/^# DEPENDS/a' > eddep
X      echo >> eddep
X      echo '.' >> eddep
X      echo '.,\$\$d' >> eddep
X      echo '\$\$r makedep' >> eddep
X      echo 'w' >> eddep
X      -ed - Makefile < eddep
X      rm -f eddep makedep
X
X
X# DEPENDS -- dont remove this line, make depends needs it
TEXT_STOP
} > Makefile
#
# tell the user what we have done
echo "done"
echo "-----"
echo "The default make rule is: all."
echo "It has the main model $mainmod as it's target."
echo
echo "The rule: clean"
echo "removes all object files"
echo
echo "The rule: depend"
echo "adds the inter-module dependencies to the generated Makefile"

```

4 Future development of RASS

The directory `pvm` contains some prototypes of MDPs, which run on distributed systems. The `pvm` library is necessary to use them (installed on `baikal.ethz.ch`, obtained from `netlib2.cs.utk.edu`).

The `AuxLib` should be completed, such that at minimum all sample models of the RAMSES distribution are supported.

The internal windows data structures of `DMWindow` should be maintained, such that a program which relies on them shares the same behaviour using the `Batch-DM` or the interactive `DM`.

The transition from RAMSES to RASS could be improved. Especially the conversation of text files (MDPs and data files) is not yet supported by the RASS tools. Only the renaming (`MOD2mod` and `DEF2def`) is provided.

`DMSystem` does report the `ComputerSystem` as unknown. This is annoying, since it is therefore not possible to see where a stash file is created. To keep `DMSystem` portable it should be able to get a `ComputerSystem` identification string from `SysDep`.

5 Literature

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

Fischlin, A. *et al.*, 1994. *ModelWorks 2.2: An Interactive Simulation Environment for Personal Computers and Workstations*. Internal Report # 14, Systems Ecology, ETHZ, .

Fischlin, A., Mansour, M.A., Rimvall, M. & Schaufelberger, W., 1987. *Simulation and computer aided control system design in engineering education*. In: I. Troch, K., P. & Breitenecker, F. (ed.), *Simulation of Control Systems*, 459pp., Pergamon Press, Oxford a.o., p.^pp. 51-60.

Fischlin, A. & Schaufelberger, W., 1987. *Arbeitsplatzrechner im technisch-naturwissenschaftlichen Hochschulunterricht*. *Bulletin SEV/VSE*. , **78**: 15-21.

Keller, D., 1989 *Introduction to the Dialog Machine*. (Bericht Nr. 5, Projektzentrum IDA, ETH Zürich): 37 pp.

Thoeny, J., Fischlin, A. & Gyalistras, D., 1994. *RASS: Towards bridging the gap between interactive and off-line simulations*. In: J. Halin, W.K.a.R.R. (ed.), *CISS - First Joint Conference of International Simulation Societies*, Zuerich, Switzerland, The Society for Computer Simulation International, P.O. Box 17900, San Diego, Cal. 92177, USA, p.^pp. 99-103.

BERICHTE DER FACHGRUPPE SYSTEMÖKOLOGIE
SYSTEMS ECOLOGY REPORTS
ETH ZÜRICH

Nr./No.

- 1 FISCHLIN, A., BLANKE, T., GYALISTRAS, D., BALTENSWEILER, M., NEMECEK, T., ROTH, O. & ULRICH, M. (1991, erw. und korr. Aufl. 1993): Unterrichtsprogramm "Weltmodell2"
- 2 FISCHLIN, A. & ULRICH, M. (1990): Unterrichtsprogramm "Stabilität"
- 3 FISCHLIN, A. & ULRICH, M. (1990): Unterrichtsprogramm "Drosophila"
- 4 ROTH, O. (1990): Maisreife - das Konzept der physiologischen Zeit
- 5 FISCHLIN, A., ROTH, O., BLANKE, T., BUGMANN, H., GYALISTRAS, D. & THOMMEN, F. (1990): Fallstudie interdisziplinäre Modellierung eines terrestrischen Ökosystems unter Einfluss des Treibhauseffektes
- 6 FISCHLIN, A. (1990): On Daisyworlds: The Reconstruction of a Model on the Gaia Hypothesis
- 7 * GYALISTRAS, D. (1990): Implementing a One-Dimensional Energy Balance Climatic Model on a Microcomputer (*out of print*)
- 8 * FISCHLIN, A., & ROTH, O., GYALISTRAS, D., ULRICH, M. UND NEMECEK, T. (1990): ModelWorks - An Interactive Simulation Environment for Personal Computers and Workstations (*out of print[] for new edition see title 14*)
- 9 FISCHLIN, A. (1990): Interactive Modeling and Simulation of Environmental Systems on Workstations
- 10 ROTH, O., DERRON, J., FISCHLIN, A., NEMECEK, T. & ULRICH, M. (1992): Implementation and Parameter Adaptation of a Potato Crop Simulation Model Combined with a Soil Water Subsystem
- 11 * NEMECEK, T., FISCHLIN, A., ROTH, O. & DERRON, J. (1993): Quantifying Behaviour Sequences of Winged Aphids on Potato Plants for Virus Epidemic Models
- 12 FISCHLIN, A. (1991): Modellierung und Computersimulationen in den Umweltnaturwissenschaften
- 13 FISCHLIN, A. & BUGMANN, H. (1992): Think Globally, Act Locally! A Small Country Case Study in Reducing Net CO₂ Emissions by Carbon Fixation Policies
- 14 FISCHLIN, A., GYALISTRAS, D., ROTH, O., ULRICH, M., THÖNY, J., NEMECEK, T., BUGMANN, H. & THOMMEN, F. (1994): ModelWorks 2.2 – An Interactive Simulation Environment for Personal Computers and Workstations
- 15 FISCHLIN, A., BUGMANN, H. & GYALISTRAS, D. (1992): Sensitivity of a Forest Ecosystem Model to Climate Parametrization Schemes
- 16 FISCHLIN, A. & BUGMANN, H. (1993): Comparing the Behaviour of Mountainous Forest Succession Models in a Changing Climate
- 17 GYALISTRAS, D., STORCH, H. v., FISCHLIN, A., BENISTON, M. (1994): Linking GCM-Simulated Climatic Changes to Ecosystem Models: Case Studies of Statistical Down-scaling in the Alps
- 18 NEMECEK, T., FISCHLIN, A., DERRON, J. & ROTH, O. (1993): Distance and Direction of Trivial Flights of Aphids in a Potato Field
- 19 PERRUCHOUD, D. & FISCHLIN, A. (1994): The Response of the Carbon Cycle in Undisturbed Forest Ecosystems to Climate Change: A Review of Plant–Soil Models
- 20 THÖNY, J. (1994): Practical considerations on portable Modula 2 code
- 21 THÖNY, J., FISCHLIN, A. & GYALISTRAS, D. (1994): Introducing RASS - The RAMSES Simulation Server

* Out of print

- 22 GYALISTRAS, D. & FISCHLIN, A. (1996): Derivation of climate change scenarios for mountainous ecosystems: A GCM-based method and the case study of Valais, Switzerland
- 23 LÖFFLER, T.J. (1996): How To Write Fast Programs
- 24 LÖFFLER, T.J., FISCHLIN, A., LISCHKE, H. & ULRICH, M. (1996): Benchmark Experiments on Workstations
- 25 FISCHLIN, A., LISCHKE, H. & BUGMANN, H. (1995): The Fate of Forests In a Changing Climate: Model Validation and Simulation Results From the Alps
- 26 LISCHKE, H., LÖFFLER, T.J., FISCHLIN, A. (1996): Calculating temperature dependence over long time periods: Derivation of methods
- 27 LISCHKE, H., LÖFFLER, T.J., FISCHLIN, A. (1996): Calculating temperature dependence over long time periods: A comparison of methods
- 28 LISCHKE, H., LÖFFLER, T.J., FISCHLIN, A. (1996): Aggregation of Individual Trees and Patches in Forest Succession Models: Capturing Variability with Height Structured Random Dispersions
- 29 FISCHLIN, A., BUCHTER, B., MATILE, L., AMMON, K., HEPPELLE, E., LEIFELD, J. & FUHRER, J. (2003): Bestandesaufnahme zum Thema Senken in der Schweiz. Verfasst im Auftrag des BUWAL
- 30 KELLER, D., 2003. *Introduction to the Dialog Machine, 2nd ed.* Price,B (editor of 2nd ed)

Erhältlich bei / Download from

<http://www.ito.umnw.ethz.ch/SysEcol/Reports.html>

Diese Berichte können in gedruckter Form auch bei folgender Adresse zum Selbstkostenpreis bezogen werden /
Order any of the listed reports against printing costs and minimal handling charge from the following address:

SYSTEMS ECOLOGY ETHZ, INSTITUTE OF TERRESTRIAL ECOLOGY
GRABENSTRASSE 3, CH-8952 SCHLIEREN/ZURICH, SWITZERLAND