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; Kreutzer 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

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

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.

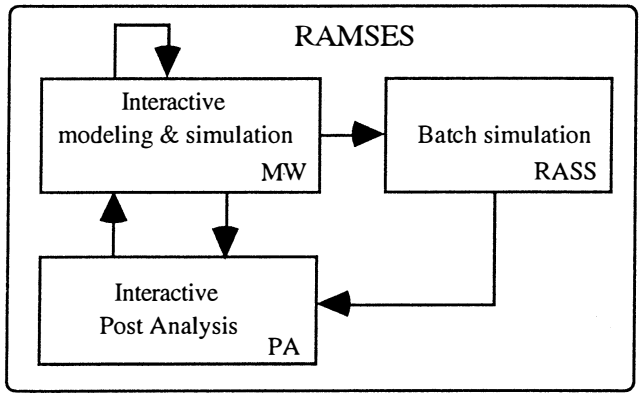

Fig. 1: State transitions while developing and solving models using RASS within RAMSES.

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).

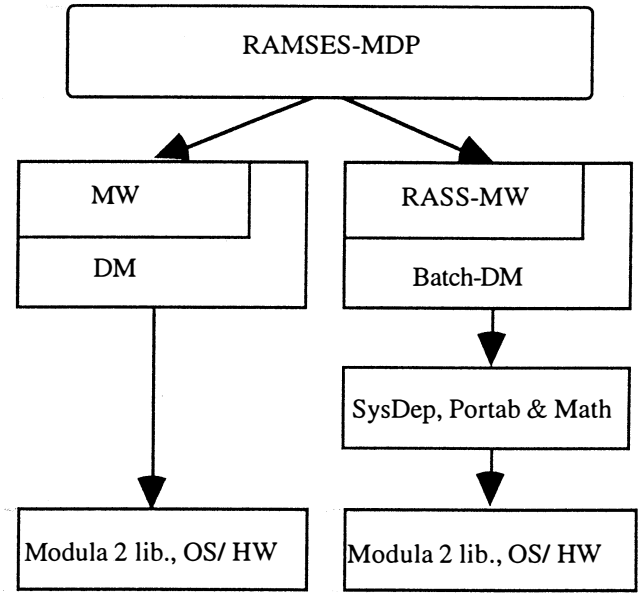

Fig. 2: Architecture of RAMSES and RASS.

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.

$t_{t\ (RAMSES)} = t_s$
$t_{t\ (RASS)} = t_s + t_c + t_a$

Since $t_a << t_s$ we get:

$t_{t\ (RAMSES)} = t_s$
$t_{t\ (RASS)} = t_s + t_c$

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.

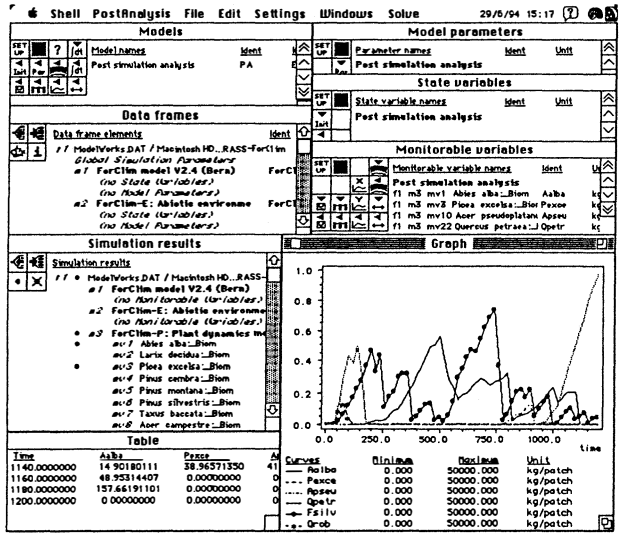

Fig. 3: Typical screen, here from the case study *ForClim*, of a RAMSES post-analysis session on the simulationist's personal computer. This session supports the interactive analysis of the uploaded simulation results by processing a stash file, which RASS previously produced on the simulation server.

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:

$$t_{t\,(RAMSES)} = t_{t\,(RASS)} = t_c + t_{s\,(RASS)}$$
$$\Rightarrow \quad t_{s\,(RAMSES)} - t_{s\,(RASS)} = t_c$$

A gain in performance occurs if

$$t_c < (t_{s\,(RAMSES)} - t_{s\,(RASS)})$$

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 $t_{s\,(RAMSES)}$ = 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 interactive 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.* Routdlege, 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 revized 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. & Breitenecker, 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.