Proceedings

Session "Software Tools and Products"

F. Breitenecker
I. Husinsky
editors

ARGESIM Report No. 2

ISBN 3-901608-02-8

ISBN 3-901608-02-8

ARGESIM Report No. 2

ARGE Simulation News (ARGESIM)
c/o Technical University of Vienna
Wiedner Hauptstr. 8-10
A-1040 Vienna, Austria
Tel: +43-1-58801 5386, 5374, 5484
Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at
WWW: <URL:http://eurosim.tuwien.ac.at/>

Printed by *CA Druckerei*, Vienna, Austria

Foreword

Developments over the last years show that beside the classical tools theory and experiment simulation becomes more and more the third major tool for problem solving in application and research. Nowadays simulation is found in nearly every application area, research activities result in new methodologies and tools for simulation, and more and more simulation software, simulators, and simulation systems are offered on the market.
The *EUROSIM Congress*, the European Simulation Congress, an international event normally held every three years, aims to be a common forum for presenting European and international recent results and applications in simulation, and to stimulate the exchange of ideas and experiences among scientists and engineers active in simulation.
EUROSIM is the Federation of the European Simulation Societies, acting as a European forum for Simulation Societies and promoting the advancement of system simulation in industry, research, and education.

All these intentions are reflected in the 5th European Simulation Congress *EUROSIM 95*, the 2nd Congress after the formal foundation of EUROSIM. The scientific programme consists of invited and contributed papers to regular sessions and to "Special Interest Sessions", of contributions to the session "Software Tools and Products", and of posters.

The invited and contributed papers to regular sessions and to "Special Interest Sessions" are published in the Congress Proceedings printed by Elsevier Science B.V. and in a Late Paper Volume (ARGESIM Report, ISBN 3-901608-01-X). The Proceedings contain eight invited papers and 212 contributed papers, the Late Paper Volume contains 20 contributed papers. The papers were selected by the International Programme Committee from 459 abstracts received.

The session "Software Tools and Products" presents papers dealing with State-of-the-Art and new features of simulation languages, simulators, and simulation environments. These contributions also passed the review process and are published as ARGESIM Report (Proceedings EUROSIM'95 - Session "Software Tools and Products", ISBN 3-901608-02-8).

The reviewed Poster Session completes the scientific programme. The abstracts of the 111 posters are published as ARGESIM Report (EUROSIM'95 - Poster Book, ISBN 3-901608-03-6).

It is interesting to compare the titles of papers and posters presented at previous European Simulation Congresses with those at the present congress. Even a brief glance through the four volumes of Proceedings and Late Paper Volumes shows that in this twelve year period considerable, remarkable, and sometimes astonishing advances have been made in a number of different areas. For example, developments in parallelism and distributed processing are now not only being seen in simulation applications but are also frequently used.
Object-oriented methods are being implemented now, and artificial intelligence and knowledge-based tools appear to be an established part of system modelling and simulation methodology. The availability of improved graphic algorithms and tools is also leading to some very interesting and innovative research and application in terms of man-machine interface and of animation and visualisation, both for discrete-event and continuous-system simulation.

New developments in terms of mathematical modelling and simulation techniques as well as in terms of general methodology are of little significance unless they are stimulated by the requirements of the real world in terms of industry, business, agriculture and the sciences. We are very pleased, therefore, that application papers are so well represented. This also applies to papers on parallel and distributed simulation, where beside graphics the fastest development can be observed.

We are also pleased that the idea of "Special Interest Sessions" could be realized. These sessions deal with recent developments in areas where methodology and application are considered together. The results of the closing discussion at the end of these sessions are summarized in manuscripts which will be edited and published in abbreviated form in *EUROSIM - Simulation News Europe* (SNE), the newsletter of the EUROSIM member societies. Some of these papers will be prepared for publication in EUROSIM's scientific journal *SIMULATION PRACTICE AND THEORY*. A separate role is played by the Industry Session on "Model Exchange and Software Independent Modeling" where people mainly from industry report on this topic without necessarily having to publish a paper in the Proceedings. Furthermore, we are pleased, that the contributions to the session "Software Tools and Products" show a very broad spectrum of simulation software, and that the Poster Session presents new ideas under development.

The European Simulation Congress *EUROSIM 95*, held in Vienna (Austria) at the Technical University of Vienna from September 11 through September 15, 1995, is organized by *ASIM* (Arbeitsgemeinschaft Simulation), the German speaking Simulation Society, in co-operation with the other member societies of EUROSIM: *AES* (Asociación Española de Simulación), *CSSS* (Czech & Slovak Simulation Society), *DBSS* (Dutch Benelux Simulation Society), *FRANCOSIM* (Société Francophone de Simulation), *HSTAG* (Hungarian Simulation Tools and Application Group), *ISCS* (Italian Society for Computer Simulation), *SIMS* (Simulation Society of Scandinavia), *SLOSIM* (Slovenian Society for Simulation and Modelling), *UKSS* (United Kingdom Simulation Society).

The moral co-sponsorship of *CASS* (Chinese Association for System Simulation), CROSSIM (Croatian Society for Simulation Modelling), *IFAC* Advisory Board Austria, *IMACS* (International Association for Mathematics and Computers in Simulation), *JSST* (Japanese Society for Simulation Technology), *LSS* (Latvian Simulation Society), *OCG* (Austrian Computer Society), *PSCS* (Polish Society for Computer Simulation), *ROMSIM* (Romanian Society for Modelling and Simulation), *SCS* (Society for Computer Simulation), *SiE* Esprit Working Group "Simulation in Europe" supports this congress.

A successful conference is always due to the efforts of the many people involved. To this purpose, particular acknowledgement goes to the members of the Scientific Committee for their contributions in the paper selection process, to the members of the Local Organizing Committee, and more especially to the head of this committee, to Mr. Manfred Salzmann. We would like to thank *Unseld + Partner* and *CA (Creditanstalt)* for sponsoring the printing of this report.

Felix Breitenecker Irmgard Husinsky

Technical University of Vienna

Scientific Committee

F. Breitenecker (Austria), Chairman

H. Adelsberger (D)
M. Alexik (SQ)
W. Ameling (D)
S. Balsamo (I)
I. Bausch-Gall (D)
S.W. Brok (NL)
F.E. Cellier (USA)
V. Ceric (HR)
L. Dekker (NL)
J. Dongarra (USA)
V. De Nitto (I)
H. Ecker (A)
G. Feichtinger (A)
P. Fishwick (USA)
J.M. Giron-Sierra (E)
H.J. Halin (CH)
N. Houbak (DK)
R. Huntsinger (USA)
I. Husinsky (A)
T. Iversen (N)
A. Jávor (H)
K. Juslin (FIN)
G. Kampe (D)
E. Kerckhoffs (NL)
W. Kleinert (A)
P. Kopacek (A)
W. Kreutzer (NZ)
M. Lebrun (F)
F. Lorenz (B)
F. Maceri (I)
D. Maclay (GB)
H. Mang (A)
Y. Merkuryev (LV)
Z. Minglian (VRC)
I. Molnar (H)
D.P.F. Möller (D)
D. Murray-Smith (GB)
F.J. Pasveer (NL)
R. Pooley (U.K.)
W. Purgathofer (A)
P. Schäfer (D)
T. Schriber (USA)
A. Seila (USA)
W. Smit (NL)
F. Stanciulescu (RO)
A. Sydow (D)
H. Szczerbicka (D)
S. Takaba (J)
I. Troch (A)
G.C. Vansteenkiste (B)
W. Weisz (A)
J. Wilkinson (GB)
R. Zobel (GB)
B. Zupancic (SLO)

Organization Committee

Felix Breitenecker, Dept. Simulation Techniques,
Institute for Technical Mathematics, Technical University of Vienna

Irmgard Husinsky, Dept. for High Performance Computing, Computing Services,
Technical University of Vienna

Local Organizers are the Dept. Simulation Techniques of the Technical University of Vienna and the Dept. for High Performance Computing of the Computing Services of the Technical University of Vienna, and the "ARGE Simulation News".

Local Organizing Committee

M. Salzmann;

K. Breitenecker, B. Gabler, M. Holzinger, C. Kiss, M. Klug, N. Kraus, M. Lingl,
I. Mannsberger, J. Schuch, G. Schuster, H. Strauß, S. Wassertheurer, W. Zeller

About ARGESIM

ARGE Simulation News (ARGESIM) is a non-profit working group providing the infra structure for the administration of **EUROSIM** activities and other activities in the area of modelling and simulation.

ARGESIM organizes and provides the infra structure for

- the production of the journal EUROSIM Simulation News Europe
- the comparison of simulation software (EUROSIM Comparisons)
- the organisation of seminars and courses on modelling and simulation
- COMETT Courses on Simulation
- "Seminare über Modellbildung und Simulation"
- development of simulation software, for instance: mosis - continuous parallel simulation, D_SIM - discrete simulation with Petri Nets, GOMA - optimization in ACSL
- running a WWW - server on EUROSIM activities and on activities of member societies of EUROSIM
- running a FTP-Server with software demos, for instance
  - * demos of continuous simulation software
  - * demos of discrete simulation software
  - * demos of engineering software tools
  - * full versions of tools developed within ARGESIM

At present ARGESIM consists mainly of staff members of the Dept. Simulation Technique and of the Computing Services of the Technical University Vienna.

In 1995 ARGESIM became also a publisher and started the series **ARGESIM Reports**. These reports will publish short monographs on new developments in modelling and simulation, course material for COMETT courses and other simulation courses, Proceedings for simulation conferences, summaries of the EUROSIM comparisons, etc.

Up to now the following reports have been published:

No.	Title	Authors / Editors	ISBN
# 1	Congress EUROSIM'95 - Late Paper Volume	F. Breitenecker, I. Husinsky	3-901608-01-X
# 2	Congress EUROSIM'95 - Session Software Products and Tools	F. Breitenecker, I. Husinsky	3-901608-02-8
# 3	EUROSIM'95 - Poster Book	F. Breitenecker, I. Husinsky	3-901608-03-6
# 4	Seminar Modellbildung und Simulation - Simulation in der Didaktik	F. Breitenecker, I. Husinsky, M. Salzmann	3-901608-04-4
# 5	Seminar Modellbildung und Simulation - COMETT - Course "Fuzzy Logic"	D. Murray-Smith, D.P.F. Möller, F. Breitenecker	3-901608-05-2
# 6	Seminar Modellbildung und Simulation -COMETT - Course "Object-Oriented Discrete Simulation"	N. Kraus, F. Breitenecker	3-901608-06-0
# 7	EUROSIM Comparison 1 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-07-9
# 8	EUROSIM Comparison 2 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-08-7

For information contact: ARGESIM, c/o Dept. Simulation Techniques,
attn. F. Breitenecker, Technical University Vienna
Wiedner Hauptstraße 8-10, A - 1040 Vienna
Tel. +43-1-58801-5374, -5386, -5484, Fax: +43-1-5874211
Email: argesim@simserv.tuwien.ac.at

Table of Contents

Foreword iii

Committees v

About ARGESIM vi

"HYPAS" - Software Simulation Package for Electro-Hydraulic Drive Installations
Ionescu Fl., Hasler Fl., Ciomaga A. 1

Computation of Noise Radiated by Fluids Flow around Rotating Blades in the Subsonic Case with the DECIVENT Software
Peyraut F. 5

Simulation Assistance for Analog Simulation
Jendges R. 9

Marketing in Simulation
Unseld H.G. 13

Simulation and Animation Software from Wolverine
Crain R.C. 17

Building Discrete Event Simulators with Create!
Rüger M. 21

CAD-I++: Advanced Interface between CAD and Simulation System SIMPLE++
Kronreif G. 25

Insights into Successful Simulation Software Products
Annino J. 29

Metamodelling of Simulation Tools
Frank M. 33

Modular Application Objects: Closing the Gap between Flexibility and Ease of Modelling
Geuder D. 37

Micro Saint Simulation as a Means of Evaluating Optimal Conveyor Management Strategies
Laughery K.R. 41

Quantitative Design of Material Handling System Using Predictive Simulation Modeling
Lu Ch.J.J., Tsai K.H., Yang J.J.S., Wang Y.M. 45

Simulation for Production Scheduling: Integration of Simulation Techniques into Planning and Scheduling
Popp V. 49

Top Down Design with VHDL-A
Trihy R., Kundert K. 53

VASIMS: A Software Package for Validating and Simulating FMSs Modeled by Petri Nets
Amer-Yahia C., Haddab Y., Achour H., Zerhouni N. 57

Simulation-Aided Investment Analysis for Manufacturing
Klug F. 61

A Scalable 3D Animator with Open Interfaces
Kirchner H., Helbing R. 65

The Next Generation of Simulation Tools: A Focus on Usability in Micro Saint
Laughery K.R., Drury C.E. 69

X": A Vision for Worldwide Simulation
Symons A. 73

Simulation Using SITA
Jonin G., Sedols J., Tomsons Dz. 77

ECHTZEIT-ERWEITERUNG for SIMULINK on PCs
Bausch-Gall I. 81

Linkage of a CAE-Simulation-Tool to a Realtime-Operating System
Beck K. 83

ACSL For Real Time Simulation
Gauthier J.S. 87

Automatic Code Generation for Multi-DSP Networks on the Basis of SIMULINK Block Diagrams
Kiffmeier U. 91

Technical Computing Environment MATLAB - New Developments
Stahl H. 95

The Simulation System ANA V2.0
Goldynia J.W., Marinits J.M. 99

Intelligent Simulation Interface for LATISS Simulation System
Merkuryev Y., Merkuryeva G., Mazversitis A., Goldmanis A. 103

Architecture of a Simulator for Research in Advanced Traffic Management System Design
Ingle R.M. 107

MAPLIS - Simulation with Work Sheets in Hyper Space
Tettweiler W. 111

Vibratory Analysis of Thin Shell Structures in Medium and High Frequency Range with the ACTU Software
Peyraut F. 115

A Multi-Purpose Simulator for an Oil Production Plant
Øgård O., Halvorsen I., Telnes K., Sjong D. 119

Analysis of Multilayer Medium Acoustic Behaviour with the SIMAMCO Software
Peyraut F. 123

The Module partDEQ - Solving Partial Differential Equations Using SIMUL_R
Ruzicka R. 127

Extremely Exact and Fast Computations
Kunovský J., Mikulášek K. 131

Interactive Simulation of a Fuzzy-Logic-Controlled Nonlinear Servomechanism
Korn G.A. 135

Simulation-Based Analysis and Optimization of Complex Systems Using SIMPLORER
Knorr B., Strunz C. 139

New Developments in the Parallel Simulation System mosis
Schuster G., Breitenecker F. 143

Environment for the Simulation and Design of Control Systems
Zupančič B., Klopčič M. 147

Distributed Object-Oriented Simulation Environment: An Implementation of Time Warp Using PVM
Beraldi R., Nigro L. 151

Simulation-Based Optimization of Production Systems by SENSIM
Stief E. 155

omsim2maple - A Translation Utility for OmSim Simulation Code
Sørlie J.A. 159

Author Index 163

"HYPAS" - Software Simulation Package for Electro-Hydraulic Drive Installations.

Authors:
Fl. Ionescu, Professor at Fachhochschule Konstanz, Department of Mechanical Engineering, 55 Braunegger Str., 78462 Konstanz, Germany

Fl. Hasler, Student at Fachhochschule Konstanz, Department of Mechanical Engineering, 55 Braunegger Str., 78462 Konstanz, Germany

A. Ciomaga, Student at Fachhochschule Konstanz, Department of Mechanical Engineering, 55 Braunegger Str., 78462 Konstanz, Germany

1. Introduction

The program package is based on an original method of structured methodology and on a high experience in the field of hydro-pneumatical design and mathematical modelling. HYPAS frees the user from the difficult task of describing the system analytical behaviour and offers the possibility to design and to simulate the installation's behaviour using its own constructive elements. The scheme's set-up is made through a standardised representation. The basics are the non-linear and linearised mathematical models, which have been developed and implemented taking into consideration the specific physical behaviour of the studied components.

2. How to construct a simulation scheme

The construction of the simulation schemes is based on the association of some graphical symbols to the compound elements of the electro-hydraulic drive installations. The de-facto design of these symbols and the set-up for the connection points with other constructive elements is done with the help of a graphical symbols' editor. These symbols hide in their background mathematical models which are used in the simulation stage as a descriptor for the operation mode. Both the graphical symbol and the associated mathematical model can be adjusted during the program running.

The fundamental objects are classified from the functional point of view through object-folder definition, which group the elements with the same functionality but are different from the constructive point of view. The compound elements of these object-folders are the background used to construct the electro-hydraulic drive installations which are to be simulated.

The connections among the different elements of the simulation scheme are made by means of the connection lines. These connection lines include a set of segments which link different objects and which intersect each other.

The macro-definitions are the general form for a composite element representation. These represent a collection of elementary objects or of other macro-definitions connected in a complex way. The definition for these macros was introduced to ease the design and the comprehension of the scheme, by dividing it in different aggregates from the functional and/or constructive point of view, as well as to separate the constructive elements of the electro-hydraulic drive installations. Obviously, the macro's associated mathematical model will be made by the reunion of all associated models of the included elements. The internal structure of a macro-definition can be observed by means of a window which allows both the view and the change of its internal structure. With the help of this window as well as of the options offered by the program one can modify the macro-definition's structure and the structure for the included macros, at any level, deleting or adding objects or modifying the connections among them. These changes will modify the mathematical model for the selected macro-definition.

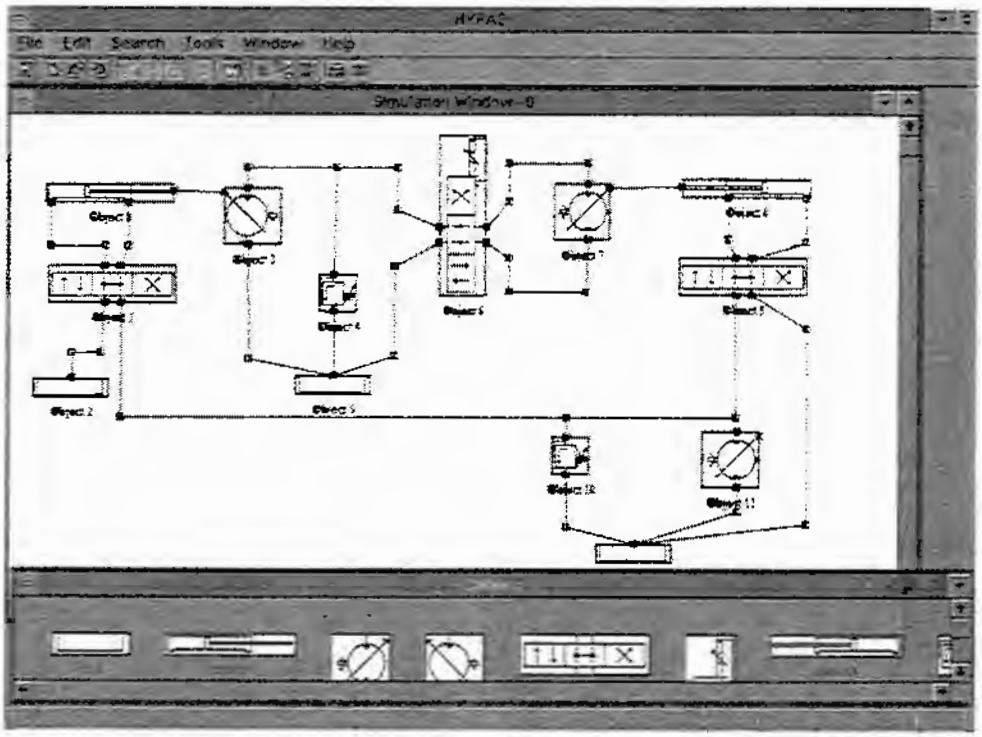
To start the design of a particular scheme one must open a simulation window, which will keep the scheme, and the desired catalogues which contain the objects included in the electro-hydraulic drive installation. To construct a scheme one must select the needed symbols from the catalogues and add them to the simulation window through a drag-and-drop like technique and then must connect them

according to the construction mode of the installation. The definition for the macros is made both through the selection of a group of objects and through the explicit construction.
The possible changes in the structure of an already designed scheme include:

- objects rearranging,
- object resizing,
- rearranging of the connection points within the associated graphical symbol of an object,
- object renaming,
- connection removing,
- object deleting.

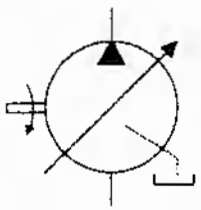
The program allows to change the graphical symbols associated to different objects and to define new symbols by means of the graphical symbols' editor as well as to create new object folders and to add new objects in existent catalogues.

The following picture shows an example of a simulation scheme drawn by means of the simulation program.

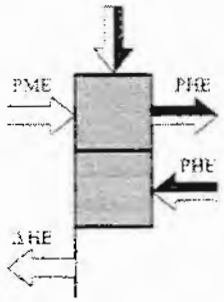


3. Types of Diagrams

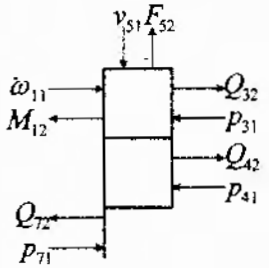
A researcher must have the adequate physico-mathematical methods for phenomena understanding in the elements' and installations' conception and auto-development stage. According to our point of view, these are realised in a graphical way to free the user from the very often difficult mathematical modelling operation. The design of these symbols is possible by means of the diagrams. We have designed four types of diagrams to help the user during the simulation process: ***Functional-Standard-Diagram, Energetical-Block-Diagram, Informational-Block-Diagram and Analogic-Block-Diagram***. We shall briefly describe each of them in the following.



i. The ***Functional-Standard-Diagram*** describes the functional behaviour of the installation (containing: engine, pomp, actuator, valves, lines, and so on) by means of some standardised symbols (at the left you can see a pomp viewed by means of the functional-standard-diagram).



ii. The ***Energetical-Block-and program flow Diagram*** shows the mechanical energy conversion into the hydraulic energy and the reverse process. One installation consists in different modules each of them containing maximum seven gates (at the left you can see a pomp viewed by means of the energetical-block-and program flow diagram).

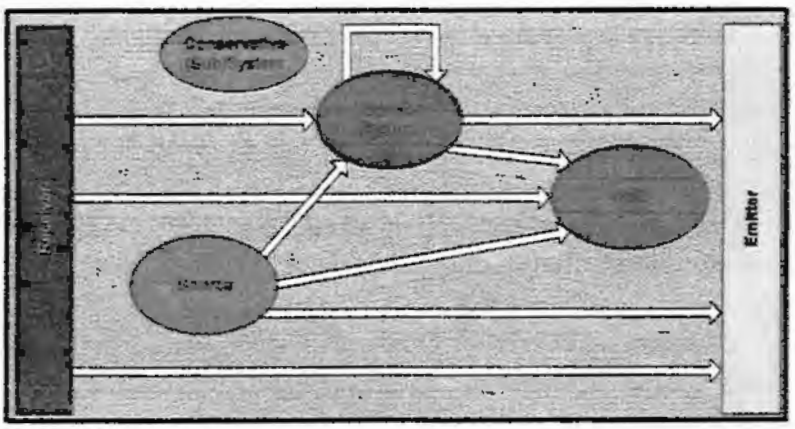


iii. The ***Informational-Block-Diagram*** is obtained from the energetical one by means of the energy decomposition into its components: flow and force variables. The variables are represented as orientated vectors, but their orientation during the operating time is not a fixed one, taking into consideration the harmonic variables involved in the process. The installation consists in associated modules, each of them containing maximum seven gates and each gate operate with maximum 2 variables (at the left you can see a pomp viewed by means of the informational-block diagram).

iv. The ***Analogic-Block-Diagram*** contains the exact linearised or simplified mathematical model of the installation and offers the mathematical flow of the information. The mathematical models are decomposed up to the elementary information layer. The informational hierarchy contains the following layers: group proprieties, semimodule, module, chain and installation. This diagram also contains all the mathematical operations with their variables and non-liniarities which describe the behaviour of the semi-module or module.

4. Mathematical Modelling

The Systemic Model Description (SMD) is referring to the design mode for a generalised physical system in the simulation program. The SMD concept is based both on the objective reality and on its scientifically perception as one can observe in the following picture.



Systemic Model Description

The Systemic Model Description assumes the Model Systemic dichotomy:

- According to the *Energy exchange with the environment:*
  Transfer - Exchange with the environment
  Conservative - No energy transfer with the exterior

- Terminal energy exchange blocks:
  Source - Specialised energy spring
  Sink - Energy flow terminal

- Surface energy exchange units (SEEU):
  Receiver - Input, Sensor - translation unit from environment to the inner world of the system
  Emitter - Output - energy translation unit to the environmental world

This description hides in its background the schemes' creation mechanism as compound systems. Except the surface subsystems, all subsystems can be on their turn aggregated. By means of this method, one can form a hierarchy with different depth description levels.

Observations:

- The number of Receivers or Emitters defines the number of system Input or Output gates, respectively.
- A *Source* has no inputs and one output at least. A *Receiver* has at least one input and no outputs.
- A *Conservative* system has no I/O gates. In order to exist an energy flow there must exist at least an energy *Source*.
- The highest hierarchical system is a *Conservative* system.

This „natural" way for the system description offers the possibility to implement the Object Oriented technique. Other consequences of this method consist in its independence from the field of study, and in the possibility to use the same hierarchy to solve another problem, taking into consideration problem's particularities though.

5. Numerical Simulation

The Integration Methods used by the program are:
- Euler
- Runge-Kutta 3rd Order
- Runge-Kutta 4th Order
- Runge-Kutta - Gill
- Gill

Each integration method can be chosen individually for each object or globally for the entire scheme. The selection of the optimal method highly depends both on each model's structure and parameters and on the integration step so it is not possible to recommend a specific method. During the new models' creation and optimisation the chosen solution will be the one which is the closest to the real behaviour of the given element for as many parameters as possible.

6. Conclusion

The presented program is based on an accurate mathematical theory. This theory allows the use of the same program to solve the simulation problems which appear in other fields of research.

7. Literature

Ionescu, Fl. Computer Aided Design of Hydraulic and Electrohydraulic Drive Installations. Proceedings of the 9th Trienal World IFAC Congress, Budapest, Hungary, Pergamon Press, 1st Volume, pp. 569-574, 1985.

Ionescu, Fl. Numerical Simulation of Large Hydrostatic Drive Systems. Proceedings of the 3rd Cairo Univ. Conf. on Current Adv. in Mech. Design & Production, Cairo, Egypt, December 28-30, 1985, pp. 135-142.

Ionescu, Fl., Stoffel, B. Contribution to the Automatic Generation of Mathematical Models for the Computer Assisted Analysis and Synthesis of Hydraulic Drive Systems. The 2nd International Conf. on Fluid Power, 19-21 March 1991,Tampere, Finland.

COMPUTATION OF NOISE RADIATED BY FLUIDS FLOW AROUND ROTATING BLADES IN THE SUBSONIC CASE WITH THE DECIVENT SOFTWARE

PEYRAUT FRANÇOIS

SCIENCES INDUSTRIES CONSEILS
14, Avenue de Sceaux
78 000 VERSAILLES
FRANCE
Tel. : (1) 39 49 05 27
Fax : (1) 39 02 77 23

1 - INTRODUCTION

In this paper, we present the DECIVENT software. This is an acoustic simulation's software devoted to ventilator blade's fast design. It allows engineers to compute the noise radiated by fluids flow around rotating blades in the subsonic case.

The acoustic calculation is based on the Lighthill's aeroacoustic analogy ([3]). The computation is made in free and far field. Considering the subsonic case, we only compute thickness and load noises.

Conciliating the necessities of rapidity and accuracy, a simplified approach based on radial equilibrium has been choosen for the steady force's computation. Numerical and experimental correlations have valided this approach.

DECIVENT has been developed with fortran 77.

2 - NOTATION

Bold quantities represents vectors. The other quantities are real or complex numbers following the context.

3 - THEORY

Assuming periodic the different noise's sources, we consider the tone noise which maximum intensity appears at blade's passing frequency.

Taking the Lighthill's aerodynamic theory, the wave equation for the acoustic pressure p can be written :

$$\phi P - 1/c^2 \, \delta^2 P/\delta t^2 = \delta^2 T_{ij}/\delta x_i . \delta x_j - \text{div } \mathbf{f} + \delta Q/\delta t \qquad (1)$$

with c the sound speed and Q the flow rate. f represents the blade's force applied to fluid, T_{ij} is the Lighthill's stress tensor, ϕ symbolize the Laplacian operator and div the divergence operator.

The left hand member of equation (1) concern the acoustic wave's propagation and the right hand member is related to the various mechanichs of noise's generation.

The term $\delta Q/\delta t$, of monopolar type, represents the thickness noise due to fluid's volume moved by the blade's rotation.

The term div f, of dipolar type, is called load noise. It represents the forces applied to fluid by the blades.

Finally, the quadrupolar noise, associated to the term $\delta^2 T_{ij}/\delta x_i.\delta x_j$, is connected to the stress tensor which acts in the fluid near the blades. As it's a second order term at the considered Mach number, we can neglect it.

By using the Green's function G of the problem :

$$G(\mathbf{X},t,\mathbf{Y},\tau)=\delta(t-\delta-r/c)/(4\pi r) \tag{2}$$

with **X** the reception point, **Y** the emission point, t the reception time, τ the emission time, δ the Dirac's function and r the distance beetween the reception and emission points :

$$r=||\mathbf{X}-\mathbf{Y}|| \tag{3}$$

we express the acoustic pressure in integral form :

$$P_e(\mathbf{X},t) = \int_T \int_S m_v V_n \, dG(\mathbf{X},t,\mathbf{Y},\tau)/d\tau \, dS(\mathbf{Y}) \, d\tau \tag{4}$$

for the thickness noise and :

$$P_c(\mathbf{X},t) = \int_T \int_S f_i \, dG(\mathbf{X},t,\mathbf{Y},\tau)/dy_i \, dS(\mathbf{Y}) \, d\tau \tag{5}$$

for the load noise.

These integrals are computed over time (T) and spacial ranges (S=blade's surface). m_v represents the fluid's density and V_n the blade's normal speed.

In the thickness noise, we have only quantities connected to blade's geometry and machine's working. No aerodynamic computation is needed.

On the contrary, for the load noise, it's necessary to compute the aerodynamic forces applied to fluid by the blades.

DECIVENT is intended for beginning project's design. So time computation for fast iterative calculation must be short. These requirements exclude the using of tridimensionnal software of fluid's mechanic with the object of representing the aerodynamic forces with much accuracy.

That is the reason why we have choosen a radial equilibrium approach mixed with a Lowson type empiric estimation of unsteady forces ([1] and [4]).

This approach gives us entirely satisfaction because, even if discrepancies with experimental results reach occasionally 10 %, the general tendencies are respected.

Taking into account time's periodicity of the acoustic emission and assuming that the section is thin and that the acoustic sources are compact, the integration of (4) and (5) is analytic over the azimuth and the chord and numeric over the blade's wingspan.

4 - VALIDATION

DECIVENT has been valided for thickness noise by comparisons with numerical computations made by the ONERA on a two-blades helicopter's rotor in stationary flight ([2]). The comparison's results, on figure 1, are extremely conclusive.

Moreover, DECIVENT has been experimentally corraleted with ventilators of the CDV company. The discrepancies, oscillating beetween 1 % and 10 % (see figures 2 and 3), are reasonably good for a tool devoted to fast design at a beginning project. We must remark that this last correlation was made without knowing exactly test procedures.

5 - CONCLUSIONS AND FUTURE PROSPECTS

DECIVENT is a blade's acoustic design tool conceived for fast using by no-specialists. It allows to seek the best noise-flow compromise for rotating machines as ventilators, propellers, pumps, compressors ...

The future prospects can be classified on three levels.

DECIVENT has been soon valided by existing measures. But it could be interesting to refine the discrepancies beetween theory and experiment by making additional tests in which SCIENCES INDUSTRIES CONSEILS would be involved.

The second prospect's level concern the coupling of DECIVENT with the sound propagation in pipes.

The third prospect's level is the taking into consideration of aerodynamic phenomena's finest computation (wide band noise, vortex, drag noise ...)

REFERENCES

[1] "Etude Aéroacoustique des Spectres de Raies Générés par les Ventilateurs Axiaux en Régime Subsonique", J.P. Bridelance, thèse de Docteur Ingénieur, ENSAM Paris, 1982

[2] "Calcul du Bruit de Raies Emis par un Rotor d'Hélicoptère en Champ Lointain", Michel Caplot, thèse de l'Université de Technologie de Compiègne, 1985

[3] "On Sound Generated Aerodynamically. I General Theory", M. J. Lighthill, Proceedings of the Royal Society, Series A, Vol. 211, pp. 564-587, 1952

[4] "Theoretical Analysis of Compressor Noise", M. V. Lowson, J. Acous. Soc. Am., Vol. 47, n°1, pp. 371-385, USA, 1970

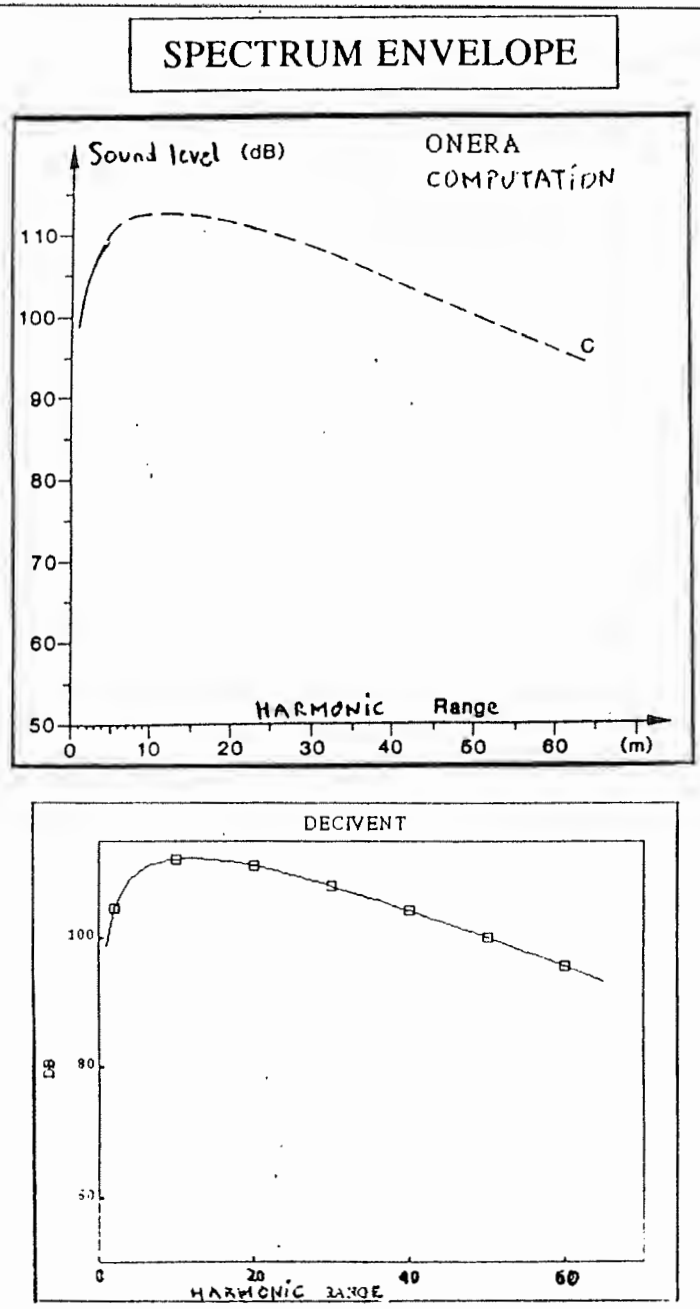


Figure 1

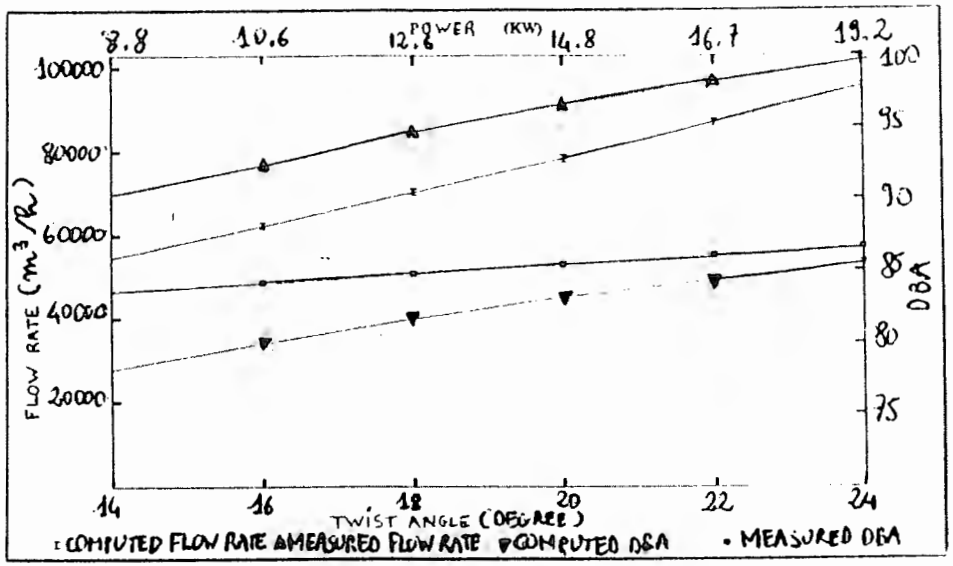


Figure 2 - 9-blades ventilator with a radius of 0.7 meters

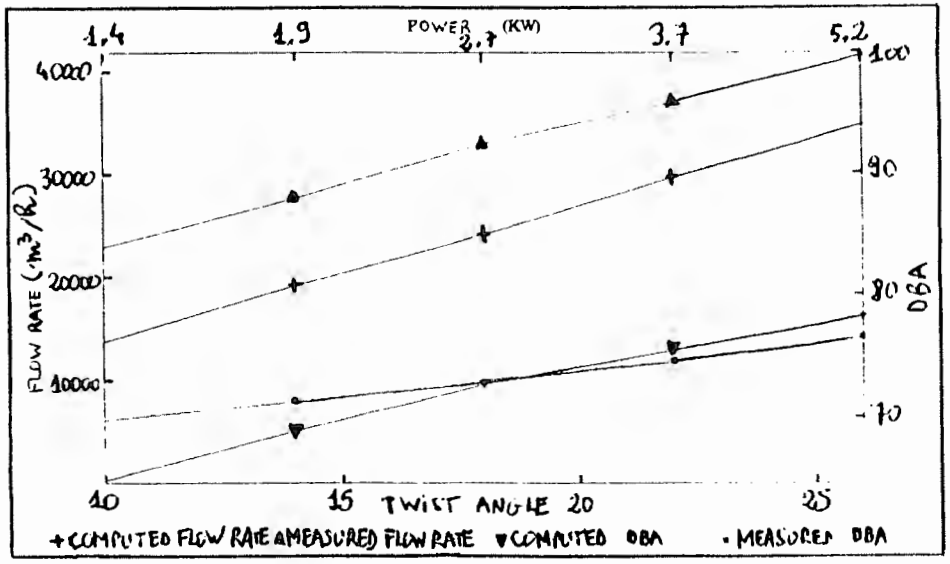


Figure 3 - 9-blades ventilator with a radius of 0.5 meters

Simulation Assistance for Analog Simulation*

Ralf Jendges
Institute for System Design Technology
German National Research Center for Computer Science
GMD-SET, 53754 St. Augustin, Germany
email: jendges@gmd.de

Abstract

Successful analog simulation is a necessary part of the design process of integrated circuits. On the other hand, the simulation can fail due to limitations of the circuit simulator or incorrect specifications. Often the error messages are only vague and the user is left without support to find the cause of the problem. This paper describes an approach to improve this situation by assistance for simulation. We present a toolbox named SAINT (Simulation AssIstaNT). SAINT supports the user with several ways to analyze the simulation problem, eliminate the problem and get the simulation running. The system can be used to locate simulation problems and to overcome convergence problems. The methods to localize simulation problems are based on partitioning and simulation of the parts. These methods and some examples are discussed in detail.

1. Introduction

Analog simulation is an integral part of the design process of integrated circuits. Circuit simulators are widely used to verify the electrical performance of circuits. In addition these simulators can be used to analyze microsystems on a behavioral level. The mechanical, optical or chemical parts of the system are taken into account by modelling the corresponding equations [Paa93].

At the same time, circuit simulation is one of the algorithmically most complex and CPU consuming parts of the design process. In spite of considerable progress in the development of simulators there are still limitations and the simulation can fail. The main reasons for a failing simulation are errors in the specification (input file) or some weaknesses of the numerical method. Typical examples for the latter are convergence problems related to the Newton-Raphson method and numerical instabilities of the integration method. Often the error messages give only vague information about the cause of the problem. Sometimes the error message is even missing. The user is left without support but with the task to make a diagnosis, eliminate the problem and get the simulation running. The probability for problems during the simulation increases with the size and complexity of the simulated system. At the same time, it becomes more difficult to analyze and overcome the problem. Thus, there is a strong demand to assist the user in overcoming problems which occur during the simulation. We will call this „simulation assistance".

Experienced users are familiar with special techniques to deal with simulation problems [Kie94]. These techniques are limited because they are performed manually and usually only modify the controlling parameters and options of the simulator. Knowledge-based systems were introduced to bypass simulation problems [Kel89] [Zan90]. These systems speed up the diagnosis but the modifications are still restricted to controlling parameters and parasitic elements. The advantages of topological changes like partitioning are not used. Recent work applies homotopy methods for the special problem of robust computation of dc operating points [Mel93]. The corresponding simulation package is proprietary.

This paper describes a novel approach to simulation assistance. First, we present a toolbox named SAINT (Simulation AssIstaNT). SAINT supports the user with different ways to analyze the simulation problem and improve the simulation. Secondly, we study in detail our methods to localize simulation problems. At last, we state some results obtained with the implemented localization techniques.

* This research is supported in part by the German Ministry of Education, Science, Research and Technology (BMBF) under grant 13MV0183

2. The SAINT system

The scheme of the proposed diagnosis procedure (fig. 1) is based on the steps:

- analyzing the simulation results
- extracting certain data
- selecting an appropriate strategy
- modifying the simulator input according to this selection
- repeating the simulation and analyzing the new results

The user controls this iterative procedure which can be repeated successively to get the desired diagnosis result.

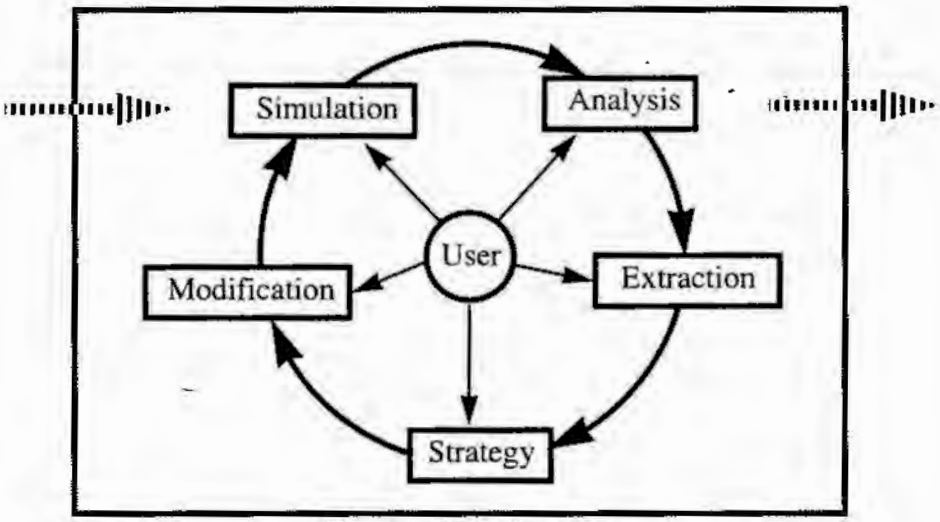


Figure 1: diagnosis process

The purpose of the simulation assistant SAINT is to offer a set of possibilities to obtain additional information to overcome simulation problems. The available techniques are combined in a toolbox to make the diagnosis more efficient. Currently SAINT helps to locate simulation problems and to improve dc convergence. The techniques to find the dc operating point are based on continuation methods [Deh94]. The methods to localize problems will be discussed in the next section.

To improve the simulation one can modify the input or the source code of the simulator which is used. We decided to change only the input file but not the simulator itself. Information about the improvement due to this changes is extracted from the simulator output. In this sense SAINT is based on a black box approach to control the simulator. This approach has essential advantages. It is easier to interface the simulation assistant to different simulators and to integrate it in an analog system design environment [Deg89]. This is a useful property concerning analog design automation.

SAINT has a graphical user interface. The user can select the input file, different kinds of modification and controlling parameters. SAINT performs corresponding actions like parsing the input file, partitioning of the system, generating modified input files, running the simulator, analyzing the simulation output and displaying results.

The diagnosis is partially-automated to minimize the necessary effort of the user. This also decreases the probability of new errors created by manual operations. The time spent for the design process can be reduced this way.

SAINT is implemented as an experimental system for the simulator SPICE3 [Joh92]. Therefore it is easy to adapt SAINT for the numerous simulators which accept the SPICE input format. The package is coded in C/C++ and the graphical user interface is written with Tcl/Tk [Ous94].

3. Localization of simulation problems

To localize problems is one important way to assist in understanding and solving simulation problems. SAINT has the capability to partition the system and repeat the simulation for the parts automatically. Successful or failing simulation of the parts indicates the local or nonlocal character of the problem.

The localization techniques were developed for the transient analysis. But dc convergence problems can be investigated too if they are transformed manually to transient problems. This is easily done by ramping all sources.

Partitioning is a well-known method for the design of integrated circuits [San77][Joh91]. Our algorithm is based on a modified node-tearing method [San77]. The costfunction is defined as the number of adjacent elements. Elements are combined to parts in such a way that the costfunction becomes minimal. The partitioning can be controlled with parameters for minimal and maximal part size. It is possible to specify „global nodes". The adjacency relations of this nodes will be ignored by the partitioning algorithm. This is useful to handle e.g. the power supply.

SAINT offers a choice between two types of coupling for the parts. In the first case we use data from the failing simulation of the original system to apply signals to the connections which are cut [Jen94]. The extracted data are the voltage waveforms up to the time when the simulation aborts. Linear approximation is used to predict the signals for the subsequent time interval. The parts are simulated separately without exchange of data between them. So, strictly speaking, this kind of coupling is a decoupling.

The second type of coupling is a waveform relaxation method [Rue87]. Controlled voltage and current sources exchange the signals between the parts. The simulation of the parts is performed iteratively. Figure 2 shows the corresponding algorithm where the subscript i denotes the part index and the superscript k denotes the iteration count. While the first type of coupling produces an error due to the linear approximation, the second coupling is free from this disadvantage. On the other hand, the first technique leads right from the beginning to an exact solution in the time interval up to the failure of the original simulation.

initialize all waveforms y_i^0
$k := 1$
while (*iteration limit not reached*)
{
  $i := 1$
  while ($i <=$ *number of parts*)
  {
    compute waveforms y_i^k *for part i for* $t \in [0, t_{max}]$ *while waveforms for other parts j are fixed:*
    $y_j = y_j^k$ *for* $j < i$
    $y_j = y_j^{k-1}$ *for* $j > i$
    $i := i + 1$
  }
  $k := k + 1$
}

Figure 2: basic waveform relaxation algorithm

The simulation of the parts can be repeated for different partitionings to test the system. If we detect a part with simulation problems, we can split this part to locate the problem more precisely. This procedure can be repeated successively. With every step the actual part becomes less complex. Therefore, even a single successful step will be useful because analyzing the failure is simplified.

4. Examples

The localization methods were already applied successfully for various examples including resistor networks, a SRAM and a BIC-monitor.

We present here the application for a simple resistor network to illustrate the basic concept. The resistor network consists of two voltage sources V_1 and V_2, eleven equal resistors and one resistor R_t which is time dependent (fig. 3). R_t is modeled by a subcircuit such that $R_t = a \cdot t + b$ with $a < 0$ and $b > 0$. Thus, there is a zero at the time $t_0 = -b/a$ and the simulation of the entire circuit fails.

The choice of appropriate controlling parameters forces the algorithm to decompose the network into two parts. The dashed lines in figure 3 indicate the partitioning. These parts **A** and **B** are simulated automatically. We observe that the simulation aborts for part **B** which contains the resistor R_t. Hence, the attempt to determine the

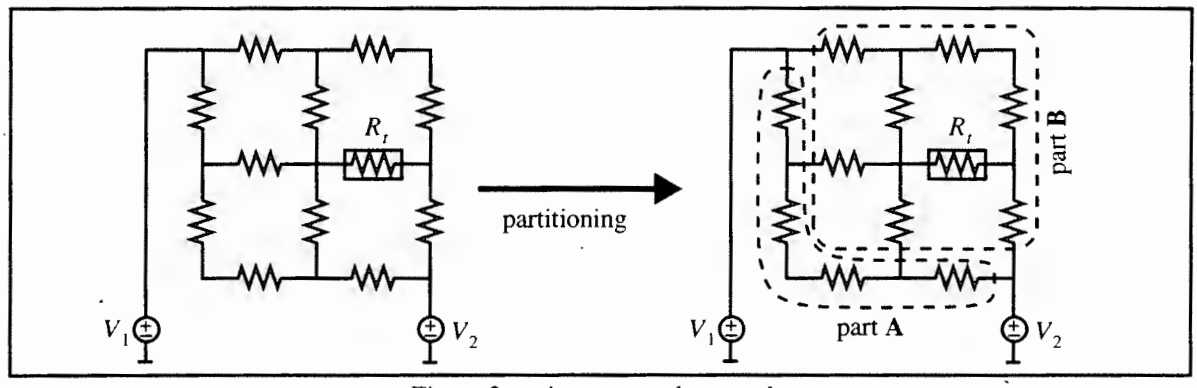


Figure 3: resistor network example

part which causes the problem was successful. The same technique can be performed successively if the system is more complex.

5. Conclusion

A novel approach to analyze and overcome problems which occur during simulation has been presented. The simulation assistant system SAINT provides several useful capabilities including localization of problems and improvement of dc convergence. Costly diagnosis processes like partitioning, modification of input files and iterations are automated to reduce the diagnosis time.

The methods to locate simulation problems are based on partitioning and simulation of the parts. A waveform relaxation technique allows to perform the simulation of the parts iteratively. This procedure can be repeated for different partitionings to detect faulty parts and to localize more precisely.

Current work is dedicated to a multilevel generalization of the waveform relaxation technique. This will improve the convergence of the iteration process. Further investigations are focusing on the approximation of solutions which can not be computed for the entire circuit.

References

[Deg89] M.G.R. Degrauwe et al., „Towards an Analog System Design Environment", IEEE Journal of Solid-State Circuits, Vol. 24, No.3, June 1989

[Deh94] M. Dehlwisch, R. Jendges, „Fehleranalyse für die Analogsimulation", Tagungsband 6. ITG-Fachtagung Mikroelektronik für die Informationstechnik, Berlin, 1994

[Jen94] R. Jendges, M. Dehlwisch, „Partitionierung und Resimulation zur Analyse simulationskritischer Systeme", Tagungsband 9. ASIM Symposium Simulationstechnik, Stuttgart, 1994

[Joh91] W. John, W. Rissiek, K.L. Paap, „Circuit Partitioning for Waveform Relaxation", Proceedings European Design Automation Conference EDAC, 1991

[Joh92] B. Johnson et. al., „SPICE3 Version 3f User's Manual", University of California, Berkeley, 1992

[Kel89] T.M. Kelessoglou, D.O. Pederson, „NECTAR : A Knowledge-Based Environment to Enhance SPICE", IEEE Journal of Solid-State Circuits, Vol.. 24, No.2, April 1989

[Kie94] R.M. Kielkowski, „Inside SPICE, overcoming the obstacles of circuit simulation", McGraw-Hill, 1994

[Mel93] R.C. Melville, L. Trajkovic, S.-C. Fang, L.T. Watson, „Artificial Parameter Homotopy Methods for the DC OperatingPoint Problem", IEEE Trans. on CAD, CAD-12, No. 6, June 1993

[Ous94] J. K. Ousterhout, „Tcl and the Tk toolkit", Addison-Wesley, 1994

[Paa93] K. L. Paap, M. Dehlwisch, R. Jendges, B. Klaassen, „Modeling Mixed Systems with SPICE3", IEEE Circuits & Devices, Vol. 9, No. 5, September 1993

[Rue87] A. E. Ruehli (Ed.), „Circuit analysis, simulation and design", Part 1 and 2, North-Holland, 1987

[San77] A. Sangiovanni-Vincentelli, L. Chen, L.O. Chua, „An Efficient Heuristic Cluster Algorithm for Tearing Large-Scale Networks", IEEE Trans. on Circuits and Systems, CAS-24, No. 12, Dezember 1977

[Zan90] M. Zanella, P. Gubian, „A Learning Scheme for SPICE Simulations Diagnostics", Proceedings IEEE International Symposium on Circuits and Systems, New Orleans, 1990

Marketing in Simulation

H. G. Unseld
Unseld + Partner
Business and Marketing Consulting • Simulation!
Lerchenfelderstraße 44/9, A-1080 Vienna, Austria
E-mail: hgunseld@unseld.co.at

1. Introduction

In simulation we frequently find ourselves debating the „obvious" market demand and the „obvious" right product we can offer to satisfy this demand. Unexpected low sales, however, show that „obvious" facts do not necessarily end up with happy simulationists and satisfied customers. Therefore, the „obvious" question is: What are the elements of successful marketing for simulation in industrial and other profit-oriented organisations?

Marketing is applied in virtually all areas, where a demand meets a product. This process always requires human interaction and involves all steps of the communication cycle. Simulation can be considered as an art of communication, too. Marketing in simulation is a bundle of elements describing the process, where a person or a company is ready to pay money for getting a better insight into various fields, which usually gain more individual importance to the customers, if analysis and intuition do not work any more. This paper describes a new approach based on experience and reflections among marketing experts.

2. Needs and Conditions of Profit-Oriented Organisations

The evolution and changes of modern industrial management structures, i.e. Business Reengineering, leads to organisations with a strong focus on core business and core functions. Support functions are reduced to their minimum. Managerial functions imply extensive information technology structures.
For people working in these organisations it is vital to know how to organise themselves in order to support the core functions. To perform their jobs well the required knowledge of the people concerned has to be within their reach and span of competence. Communication trainings and workshops are meant to support the transition phase to the „new" organisation. Their objectives are in tune with the company's objectives. Their direct goals are clear, concise and easy to understand; their contributions are measurable. Their relation and access to information technology (IT) determines their possible degree of integration into the overall company strategy.
The IT system represents the company's backbone and provides state-of-the-art services and performance. This scenario describes an „ideal" situation. The management will undertake all actions to achieve these goals. And so far, all is perfect. This is the world, where the simulationist brings in his ideas to challenge industrial ambitions.

The reality, however, deeply contrasts depending on the history of the company, the type of industry and the competitive environment.

In all projects with profit-oriented organisations a simulation provider faces more or less the following facts

- increasing time pressure on professionals
- management afraid of chaos and information security: subjective high complexity
- inconsistent IT structures and data, lack of documentation and experts
- insufficient IT knowledge and user trainings, sometimes combined with a lack of professional capacity

Each simulation application project, however, requires precise data, an in-depth understanding of the behaviour of the modelled reality (=company) and a minimum knowledge of the process itself. The obvious gap between idealistic and realistic view must be filled in by confidence. The level of confidence needed for providing the information leads to an unmatched heavy burden on even small simulation projects. To handle simulation implementations successfully, a complete simulation package consisting of a bundle of software and service elements is needed, with more emphasis on service - from the confidence point-of-view.

As an interesting fact, IT systems providers find themselves trapped with the gap of real and perceived core complexity in „crafts-men-type" and „professional" customers (see Fig 1). A given degree of complexity is under-estimated by „professional customers" and overestimated by „crafts-men-type customers". The higher the degree of complexity will grow, the bigger the span of subjective complexity will become. This explains, why simulation as a „complexity-uncovering-product" has to fully rely on at least *one* person at the customer knowing the core complexity.

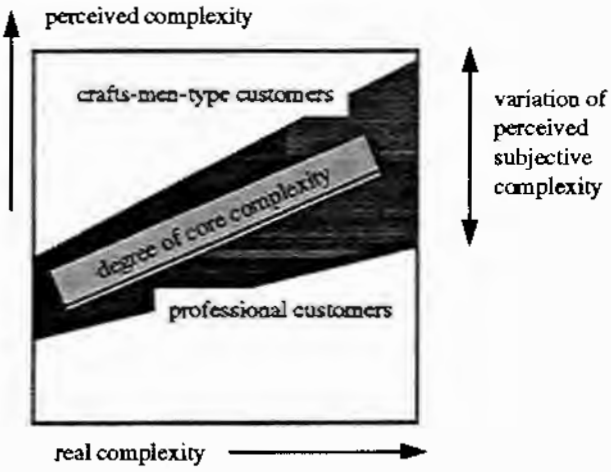


Fig. 1: Real and perceived core complexity

Is investment in simulation worthwhile? Starting with simulation technology can generally be compared with the early stage of the life cycle in Computer Aided Design in terms of know how and expertise as well as soft- and hardware investments. CAD improves the design process for new products and stimulates indirect revenues. Simulation improves the process to better and faster utilisation of the permanent cost-raising resources in a changing company environment. It primarily contributes to cost reduction and new service requirements. Its Return-on-Investment as an „design and optimisation tool" is determined by the inherent potential of the investments done, or to be done.

3. Success Factors for Simulation

Success factors are not a guaranty for success. They are rather ideas for a specific mind set.

Efficient tools. The power and flexibility of modern state-of-the-art object oriented tools in the hand of IT experts, communication experts and simulation experts guaranties a customer the best-of-its-art solution. A customer embarking on a simulation project never accepts less.

Confidence. Getting known - often detailed - inside and confidential information and delicate subjects, a customer needs to trust the simulationist right to begin with. No confidence - no data - no simulation! Since substantial investments are involved confidence must be granted for a long time.

Advocate. In every organisation simulation involves many people in many departments. The majority at the end has to vote PRO SIMULATION. In this context key account marketing and an internal advocate will help.

A new marketing approach must support all success factors. The challenge remains in learning individually which factor is prime and which is second.

4. Marketing Matrix for Simulation

The four classical elements of a marketing matrix are:

- Product mix: describing the product and its features
- Market communication mix: describing the communication between product and market
- Distribution/supply mix: describing the product's way into the market
- Price mix: describing the price conditions for supply and sales

A matrix for simulation will add the timing aspect, the product life cycle aspect, the service aspect and the technology transfer.

Timing. The interaction between customer and simulation provider differs during the phases of a project. Gaining confidence with the appropriate individual positioning of simulation is mandatory during the pre-sales phase. The marketing activities during the project phase will focus on supporting the confidence received on an individual basis. Both activities support the lifelong relationship with a customer.

Product life cycle. Simulation by its modelling nature as well as the relevant soft and hard ware enjoy short and long life cycles, both being determined by the user and the user's community and their relationship; marketing needs define the elements how to describe it. The product/market-matrix for simulation software products outlines possible conflicts.

Technology transfer. The cross functionality of simulation requires efficient communication on most critical and serious technical and organisational topics. A transfer of know how and technology is inevitable and offers synergy especially for demanding customers. This process must be structured and governed. Marketing has to value it.

Service and confidence. Simulation providers always deal with sensitive data. Sometimes even on items, where the internal IT know how is not sufficient. This provides another opportunity to serve the customers needs. The ultimate service goal is to kick any hurdle away for starting a simulation project to the benefit of both parties.

The major factor is the timing synchronisation, yet the timing of the relation between simulation provider and customer on simulation topics.

5. Simulation Performance Package SPP

Simulation providers dealing with the above listed facts and with the established demand for a simulation package usually apply classical marketing methods for advanced software marketing or expert and consultants services marketing. In this paper a new approach is shown, integrating all elements to a simulation performance package SPP.

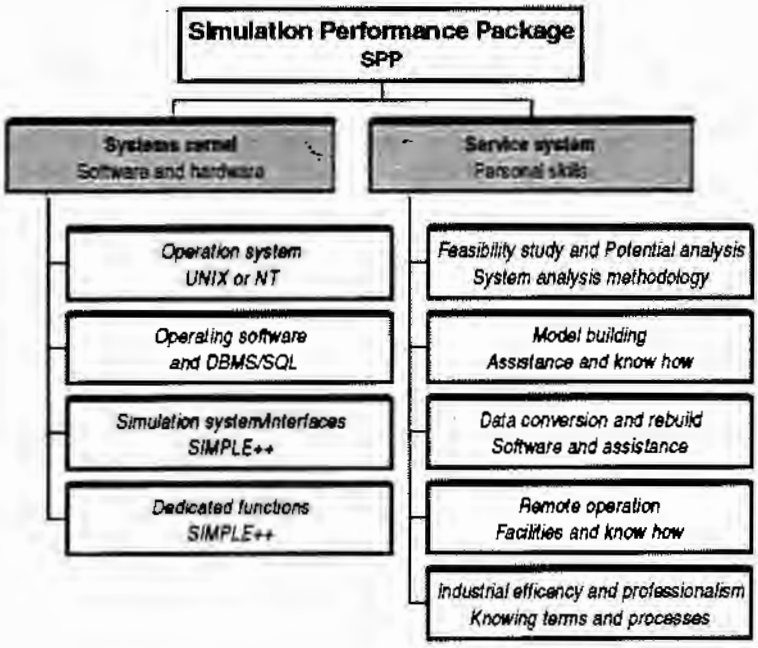


Fig. 2: Simulation Performance Package for a Manufacturing Site

The package consists of two eqnally important parts: the systems core functions, and the associated service system. Under the term „*systems core functions*" we understand all system soft and hard ware products dealing with simulation: ranging from dedicated operating systems to very special routines i.e. for optimising the throughput in a cutting machine. The relevant simulation software is an integral part of this system. The „*service system*" consists of all available services: ranging from presales software performance comparisons to provision of experts' consnltation for very specific technologies. Fig. 2 shows a set of elements of a classical simulation implementation in a manufacturing site based on SIMPLE++. The goal of marketing is to verify the customer's match, and to evaluate the differences during the whole product-in-customer life cycle. Depending on the maturity of the customer the package will be abopted accordingly.

6. Conclusion

Modern simulation technology and a concerted approach according to a proposed new marketing mix matrix will help the simulationists getting closer to customers from profit-oriented organisations. The challenge will remain with the persons dealing with the systems and how they are able to contribute to their customers wealth. This will allow them to generate more revenues for their own business and contribute to a wider acceptance of the simulation idea.

References:

Belz, C. et al (1991): Erfolgreiche Leistungssysteme - Anleitung und Beispiele, Stuttgart

Zerr, K. (1994): Systemmarketing - Die Gestaltung integrierter informationstechnologischer Leistungssysteme als Aufgabe des Marketing, Wiesbaden

Simulation and Animation Software From Wolverine GPSS/H, PROOF, and SLX

Robert C. Crain
Wolverine Software Corporation
7617 Little River Turnpike, Suite 900, Annandale VA 22003, USA
(703) 750-3910 • FAX (703) 642-9634

Wolverine Software has been providing extremely efficient, high-quality simulation software since 1977. This paper presents an overview of Wolverine's current simulation and animation products, GPSS/H and PROOF Animation, and of its forthcoming next-generation simulation product, SLX.

The widespread success of GPSS/H stems both from the superiority of its original design and from years of improvements and enhancements. Although it requires some programming-style effort, GPSS/H provides a natural modeling framework that can be readily used *without* extensive programming experience. It is equally well-suited for modeling simple systems and for modeling large, complex systems. GPSS/H is presently applied worldwide in modeling manufacturing, distribution, transportation, hospitals, computers, telecommunications, and many other types of queueing systems.

Animation is often considered a requirement for a simulation study because it is a meaningful, proven way to show results to an audience with varied backgrounds. Proof Animation is a powerful general purpose animation tool. It is not tied to a specific area, application, or simulation language, nor is it limited in the size of the systems it can animate.

SLX is Wolverine's next-generation successor to GPSS/H. While SLX retains some of the tried-and-proven fundamental concepts of GPSS/H, such as its Transaction-flow world-view, Facilities, Queues, and Storages, SLX is far more than a new implementation of GPSS/H. SLX is a layered modeling system with powerful extensibility mechanisms which facilitate the development of higher-level, graphically oriented, application-specific modeling tools.

GPSS/H

GPSS/H is a discrete-event simulation language. Models are conveniently developed in a text-based environment, and subsequently compiled *directly into memory* and executed. Rapid prototyping and iterative model development are encouraged by *exceptionally* fast compilation and execution.

GPSS/H follows the intuitive and natural *process-interaction* approach to modeling. The modeler specifies the sequence of events, separated by lapses in time, which describes the manner in which "objects" flow through a system. A GPSS/H model thus resembles the structure of a flowchart of the system being modeled. This intuitive modeling approach contributes greatly to the ease and speed with which simulation models can be built. After the model has been built, the process representation is executed by GPSS/H, with the activities of "objects" *automatically* controlled and monitored.

An "object" in a GPSS/H model might be a patient, a telephone call, or any other type of discrete entity. The representations of these entities in GPSS/H are called *transactions*. As the model executes, many transactions may be flowing through the model simultaneously—just as many "objects" would be moving through the real-world system. In addition, multiple transactions can execute GPSS/H model statements at the same instant in time *without any special action required of the modeler*.

The focus of many simulation projects is the use of system resources such as people, machines, conveyors, computers, physical space, and so on. In a GPSS/H simulation model, transactions ("objects") *compete* for the use of these system resources: as transactions flow through the process representation, they *automatically* queue up when unable to gain control of a necessary resource. The modeler does not need to specify the transaction's waiting time or its queueing behavior. Hence, the passage of time in a GPSS/H model can be represented *implicitly*, as in the case of a part waiting for a machine to be free, as well as *explicitly*, as in the case of a part being processed by a machine.

As is the case in most real-world systems, a GPSS/H model may consist of multiple processes operating simultaneously. Furthermore, each process may in some way affect the other processes in the system. GPSS/H provides the capability for multiple parallel processes to interact with each other *automatically*. Transactions ("objects") may be sent between processes; they may control or share common resources; or they may influence the (global) operation of all processes.

Important Features Of GPSS/H

Several unique characteristics make Wolverine's GPSS/H an ideal choice for a general simulation environment. A key feature of GPSS/H is the ***conceptual flexibility*** to model a wide range of *different types* of systems: any system that can be described as a process *flow*, with objects and resources acting upon each other, can be modeled. This may include people on a mass transit system, tasks in an office environment, or data flow within a computer network.

Definition flexibility is also provided within the language: complex math formulas, expressions, and constants can be used virtually anywhere in the model. To promote *model readability*, elements and entities may be specified by names instead of numbers. *Basic simulation output data,* such as queueing and service statistics, are *automatically* provided without any programming.

GPSS/H also allows ***flexibility in the selection of hardware platforms:*** GPSS/H runs on PCs, SUN SPARC workstations, VAX/VMS computers, and mainframes. On the PC, GPSS/H Professional runs as a true 32-bit application under DOS, Windows, OS/2, or Windows NT, providing tremendous speed as well as model size that is limited only by the computer's available memory.

The ***file and screen I/O*** *built into* GPSS/H provide a variety of ways to get data into a model and to produce custom output. A very intuitive "picture" type of format specification, which follows the "what you see is what you get" convention, is used to specify custom output.

A complete ***scripting language*** is available to construct experiments and control model execution. The experimental specifications and parameters, like any other model data, can be read in from a data file or from the keyboard if desired.

The GPSS/H ***Interactive Debugger*** conveniently provides for rapid model development and verification. The debugger provides a "windowing" mode that displays source code, model status, and interactive user input as the model runs. GPSS/H also provides *"just in time"* debugging. Even when the debugger was not invoked at the beginning of a run, if an error occurs during model execution, the debugger automatically "pops up" to allow the modeler to explore the cause of the error.

Features Of GPSS/H Release 3

GPSS/H is continuously improving and evolving. A few of the more significant recent additions to the widely-used GPSS/H Professional version are:

- The BLET Block and the LET Statement can now be used to assign a value to ***any*** GPSS/H data item. Unless you need the rarely used range-type assignments, *there is no longer any reason to use* the ASSIGN, SAVEVALUE, and MSAVEVALUE Blocks.
- GPSS/H now supports built-in random-variate generators for 23 additional statistical distributions (26 in all), and GPSS/H Professional now comes bundled with Unifit II, the highly-regarded distribution-fitting software from Averill M. Law and Associates.
- GPSS/H Professional supports user-written external routines in both C and FORTRAN.
- CHECKPOINT and RESTORE statements allow a model to save its state at a given point during execution, then make repeated runs using that state as the starting point.
- The SYSCALL statement and the BSYSCALL Block, which take an operating system command line as an operand, allow a running GPSS/H model to *shell out* to the operating system to run other software.
- The INSERT compiler directive allows model code to be read from multiple files during compilation.
- The operations that can be performed on Transactions in a User Chain have been extended. New SCANUCH and ALTERUCH Blocks allow examining and changing the Parameters of such Transactions without having to UNLINK and reLINK them.

Run-Time Versions Allow Economical Model Distribution

Sometimes a model is intended to be used by many people, each of whom must have a copy of the simulation software in order to run the model. Because of the number of users, the cost of the simulation software itself may render the project too expensive. Wolverine's Run-time GPSS/H offers a solution.

Run-time GPSS/H is identical to Wolverine's 32-bit GPSS/H Professional for personal computers, except that it costs less and can only run models which have been previously compiled with the regular Professional version.

Security is another important feature provided by the run-time version. Since only *pre-compiled* models can be run, the end user cannot view or edit the model "source" code. Hence, confidential models can be safely distributed.

PROOF ANIMATION

Proof Is A *General Purpose* Animator

Proof Animation can be used to animate the full range of applications, from areas such as Business Process Reengineering to the classic applications such as health care, manufacturing, and traffic. The size of the system to be animated is not an issue when using Proof Animation.

With Exceptional Features And Performance...

Proof Animation uses vector-based geometry to provide a large animation canvas and the ability to zoom in or out while maintaining crisp, clear images. Proof Animation's features include post-processing for maximum performance, built-in drawing tools and CAD import/export for ease of creating animation layouts, dynamic bar graphs and plots used for displaying statistics, a multi-windowing display, a unique presentation-making capability, and smooth, realistic motion for animations regardless of the size, complexity, or application. All versions run as 32-bit applications that require only a 386 or better CPU, a

math coprocessor, and at least a VGA-compatible video card.

Demo versions of animations can be prepared using the optional Demo-Maker feature. Copies of the demo can be reproduced and distributed free of charge and can be viewed by anyone. No licensed copy of Proof is needed to *view* an animation prepared with the Demo Maker.

And An *Open* Architecture

Proof Animation was built not only to work easily with Wolverine's GPSS/H simulation software, but also to provide affordable, powerful, easy-to-use animation software to modelers who use other simulation and programming languages. Proof Animation is driven by ASCII files. As a result, any software capable of writing ASCII text files can be used with Proof Animation.

In contrast to Proof, most animation software from other vendors is directly coupled to their simulation software. In other words, one cannot use *their* animation software without also using *their* simulation software. Worse yet, in some cases the simulation and animation software are sold only as a pair, so both must be purchased regardless of the needs of the user.

Although vendors of such tightly coupled packages often claim that their approach is the *only* way to add animation to a simulation, Proof Animation provides a mix-and-match option that allows software selection to be based on optimal functionality and price.

Post-Processing Makes The Difference

Post-processing means that the animation runs *after* the simulation has executed.

Three great advantages result from the post-processing approach. First, PC hardware is not shared between the simulation and the animation. This leaves the entire CPU for running the animation. Second, it provides the abilities to jump back and forth in time during the animation playback, to speed up or slow down the viewing speed, or show all or a specific portion of an animation. These features make it easy to investigate unusual system behavior or highlight points of interest. Third, you don't have to run the simulation model to see the animation. With small (or demo) models, this may not seem important. *But if your model requires, say, ten minutes or an hour to run and you need to show your boss (or customer) a certain part of the animation again, the importance of post-processed animation becomes quite apparent.*

Vector-Based Geometry Gives Realism, Allows Direct Import of CAD Files

One of the advantages of *vector geometry* is that an animation can be much larger than a single screen. With the ability to zoom in or out and pan side to side, larger layouts are easily navigated to show the "big picture" or zoomed in to whatever level of detail is necessary. Vector-based geometry also allows moving objects to realistically *rotate* around corners.

Another advantage of vector-based geometry is that if a CAD drawing already exists for the system to be animated, the effort of (re)drawing the system layout can be avoided. Proof Animation's built-in CAD Import/Export feature can convert industry-standard .DXF files into Proof Animation layout files, *and vice versa.* Credibility of the study is enhanced when viewers see the system's animated behavior taking place on a familiar CAD drawing of the system.

Smooth Motion Is Critical To Realism

The maximum-performance design of Proof Animation achieves *very* smooth motion by updating the screen 60-70 times per second. Other software can often sustain rates of only 5-10 updates per second. Objects that move smoothly across the screen are dramatically more realistic than those that jump across the screen.

Smart Paths Handle Accumulation

Proof Animation provides two kinds of motion: *absolute* and *guided.* Absolute motion causes an object to be moved directly between two points. In contrast, guided motion follows a predefined path. Such paths play an especially important part in transportation, product flow, and material-handling animations.

Using paths is very simple because Proof Animation does all the work. Once an object is placed on a path, it will follow that path until it visually comes to rest at the end of the path or until it is placed elsewhere or destroyed. All objects traveling on the same path can be stopped simultaneously and resume movement at a later time. Paths provide real animation power.

Accumulating paths provide even greater power. On accumulating paths, Proof Animation reflects physical reality by *visually* queueing objects when bottlenecks occur. This often makes a simulation model of the system much simpler to construct, because such queueing need not be explicitly represented in the model. Accumulating paths can be used to represent certain types of conveyors, cars at a traffic signal, customers in bank lines, and so on.

Multiple Display Windows Provide Complete Flexibility

The animation screen can be divided into separate windows. Within each window, the view can be independently manipulated using zooms, pans and rotations to include all or any portion of the animation canvas.

With this feature it is easy to maintain a window containing updated statistics in constant view while panning and zooming to different areas of the layout.

Proof Makes Presentations Easy

A professional-looking presentation can be built *completely* with Proof Animation. Its Presentation Mode lets users specify scripted sequences consisting of bit-mapped screen images or slides, full animations,

and/or segments selected from full animations. These presentation elements can be linked together using fades, dissolves, and other special effects to produce a polished presentation.

Slides can be created directly in Proof Animation or in any software package capable of exporting industry-standard ".PCX" image files. There are many such packages available, and virtually all of them can produce very high-quality charts, graphs, and slides. Proof Animation can both read and write these .PCX images, so one can save Proof Animation screen images as .PCX files and incorporate them into presentations as slides when animation is not needed.

Presentations can also incorporate selectable menus defined by the presentation developer. These menus can be set up by topic, giving the viewer or presenter complete control and flexibility of what to show.

SLX

SLX is a completely new layered modeling system, designed with the benefit of Wolverine's nearly 20 years of experience in the simulation industry. Its individual layers are as follows:

- *Level 0* is the SLX kernel, a language loosely modeled after the widely popular C language, including a run-time library. Unlike C, it provides complete run-time error-checking. It also directly supports a number of primitives (such as a generalized "wait until") essential to simulation.
- *Level 1* consists of data structures, subroutines, operators, macros, and statement-definitions, all written in SLX. These provide additional simulation primitives to support higher levels of SLX. A user can augment Level 1, adding similar capabilities of his or her own design.
- *Level 2* is the "new" GPSS/H, retaining many of the basic concepts while implementing them in a far more general way.
- *Level 3* is the level at which application-specific packages will be developed, e.g., manufacturing, telecommunications, health care. Wolverine will develop some of these packages, but we anticipate that many will be developed by third parties.
- *Level 4* will contain very high level, graphically oriented packages for use by non-simulationists.

Perhaps the greatest strength of SLX lies in its novel extensibility mechanisms, which facilitate the construction of higher layers from components contained in lower layers. Users of upper layers can ignore lower layers. However, if their requirements are not met at a given level, they can move down one or more levels, *without exerting extraordinary effort and without losing protection against potentially disastrous errors.* For example, a user wishing to add a new statement to the GPSS/H level of SLX has access to the same mechanisms we used to implement the built-in statements — a hallmark of the *open* architecture of SLX.

SLX retains an important concept from GPSS/H: its Transaction-flow world-view. This world-view has proven to be extremely flexible and powerful, *yet easily learned,* and lends itself well to graphical representation.

A second important concept retained from GPSS/H is its *total run-time error-checking* and complete reproducibility of run-to-run results. In a simulation, events take place in complex (usually random) circumstances, thus one expects to have unanticipated events. (That's a major reason for simulating: to have such events occur in a model instead of in the real system.) Determining the cause of an unexpected event can be quite difficult in a complex simulation. It would be totally unacceptable to allow undetected "programming-type" errors, such as referencing beyond the end of an array, to muddy the waters further. Such errors must be unfailingly trapped.

SLX is not a truly object-oriented language, although it has been influenced by the object-oriented programming (OOP) paradigm and it does make heavy use of objects. In SLX, objects are used in two ways. *Passive* objects are used for modeling entities which have no "executable" behavior. For example, a parking lot could be modeled as a passive object. *Active* objects have executable behavior patterns. Customers in a supermarket are a good example of entities that would probably be modeled as active objects. SLX active objects are roughly equivalent to GPSS/H transactions.

SLX objects can have a number of *standard properties*. All standard properties are comprised of explicitly identified sections of executable code. The *initial* property is invoked when an object is created. The *final* property is invoked when an object is about to be destroyed. The *actions* property specifies the behavior pattern for an active object. The *clear* and *reset* properties specify what should be done to an object when statistics are cleared or reset. The *report* property is used for the obvious purpose.

SLX presently operates in a highly interactive, window-based environment which features a fully integrated editor, compiler, and debugger, with work underway on tools to support building models graphically. On "typical" machines, "typical" models can be compiled at rates exceeding 1000 statements per *second.* Both compilation *and execution* errors are highlighted in the model source. SLX contains a number of truly impressive innovations in simulation technology. It provides the sophisticated capabilities that will be required to meet rapidly-changing modeling needs in the second half of the 1990s.

Building discrete event simulators with Create!

Michael Rüger

Dept. of Computer Simulation and Graphics
Otto-von-Guericke-Universität Magdeburg
P.O.BOX 4120
D-39016 Magdeburg, Germany
michael@isg.cs.uni-magdeburg.de

Abstract

Create! started as a project for developing a discrete event simulator for material flow and logistics systems. Since then it has been enhanced to form a multi-platform, powerful integrated development environment (IDE) for discrete event simulation.
This paper will show which steps where necessary to arrive at where we are now and how several quite different simulation and evaluation tools where built using this environment.

Introduction

Through recent years new graphical simulation tools have appeared on the market and gained a remarkable success in the simulation community. Most of these environments come with a set of prepacked components aimed at a certain application field, some provide a more or less general extensibility.
Solutions to the problem of extensibility range from simple programming language interfaces providing access to external C(++), FORTRAN or Pascal code to combined library and simulation language approaches for extending the set of simulation elements. Finding a both flexible and intuitive way of specifying the behavior of elements or processes is another challenge in this context.
Allowing a user to create new or modify existing elements requires the existence of a language or process specification of some sort to describe the desired behavior. Compiling/linking on the one hand or interpreting this code on the other both have their pitfalls. While the first generally produces faster models it introduces a potential danger of corrupting the system through unwanted side effects or simply programming mistakes. Interpreting user code allows trapping errors at runtime but is somewhat slower.
Another important issue is the range of supported platforms. Using an application and it's data on different platforms implies support for different filename conventions, character encodings and user interface look-and-feels.
By choosing Smalltalk as the underlying development environment, most of the above problems could be solved quite easily. ObjectWorks™-Smalltalk is binary portable across all supported platforms, thus allowing us to use Create! on workstations as well as on PC's or Macintoshs™. The incremental compilation and dynamic binding provides a way to add code at runtime while providing the safety of an interpreted system.

In the following we will show how these capabilities are used to solve the problems mentioned above and together with the modelling and development framework form a powerful environment with low threshold and high ceiling.

The Create! cloud[1]

Based on this underlying framework is a two step approach to the development of simulators. The first step is the creation of

- a generic runtime environment for building models, performing simulation runs and evaluating the results and
- an integrated development environment containing all necessary tools to create new libraries of elements and objects.

Combining the libraries with the generic simulation environment in the second step results in a new simulator for the end user (*Figure 1*).

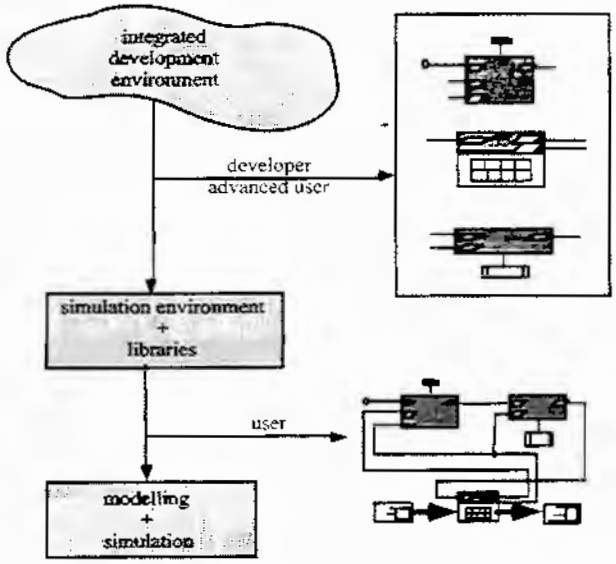


Figure 1: Create! IDE and runtime environment

1. The word *cloud* came up during the early Create! development when there was just this cloudy vision but no idea how to realize it.

Simulation kernel and modelling language

The Create! simulation kernel is based on extended finite automata. Each automaton is defined by a set of states, state transitions and functions attached to the transitions. Transition functions are defined (coded) using an object-oriented programming language which is syntactically close to C++ and conceptually close to Smalltalk-80. It supports inheritance for data structures and simulation elements as well as function polymorphism depending on function name and parameters.

The notion of finite automata is extended in so far as some transitions may be guarded, so, depending on the outcome of a boolean expression (condition), one of two alternate transitions will be executed. Each automaton is executed by a state machine which reacts to input signals and performs the appropriate transition depending on the current state.

This concept named SEC (State, Event, Condition) has later been extended to allow the modelling of concurrent processes based on the SEC automaton definition (Par-SEC: Parallel State, Event, Condition). Although the concept is named *parallel* SEC, processing of simulation events takes place in a strictly sequential manner. Nevertheless, as each state machine has its own execution space, one can think of these as executing in parallel, without worrying about mutual exclusion on variable access etc.

An example for utilizing this concept is a transport system with several vehicles, all sharing the same behaviour. The behaviour of a vehicle would then be described by the automaton definition. At runtime a state machine will be forked for each vehicle in the simulation model.

Elements are completely encapsulated. Communication between elements is realized by sending messages along connected in- and outputs. There are two different ways of communication: message passing with and without handshaking. The first one is used for information exchange like sending of orders and receipts, the second one for realizing the material flow. Figure 2 shows the general layout for an element with it's in- and outputs.

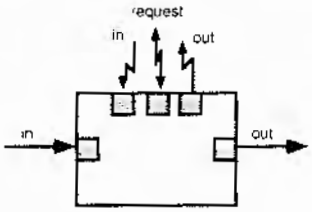


Figure 2: general element layout

Figure 3 illustrates the handshaked communication through a plug in combination with the automaton definition. Plugs define the communication protocol through a set of pins, transmitting a signal or message, and guards, promoting a boolean value.

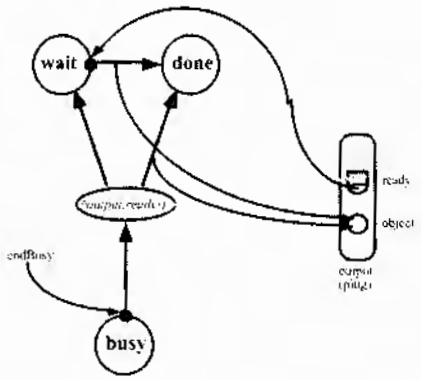


Figure 3: communication through plug

Element definition

Based on the concepts layed out in the previous chapter, the user can define Create!-elements using the element editor. The editor allows the interactive editing of constants (parameters), variables, in- and outputs as well as automata definitions and transition functions. Figure 4 shows the editor while working on the elements of the LogiChain!-environment.

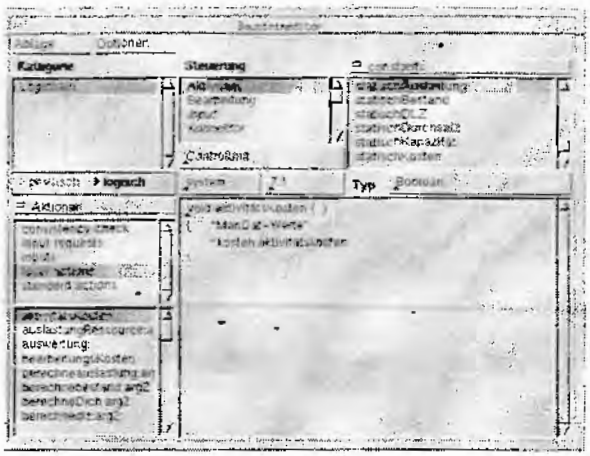


Figure 4: Element-Editor

In order to display the elements in the model editor, each element has one or more graphic symbol(s) (icon) associated with it. Icons are defined in a freely scalable vector graphic format based on the Office Document Architecture (ODA). By using the icon editor, the user may choose from one of the predefined icons from the libraries or define a new one.

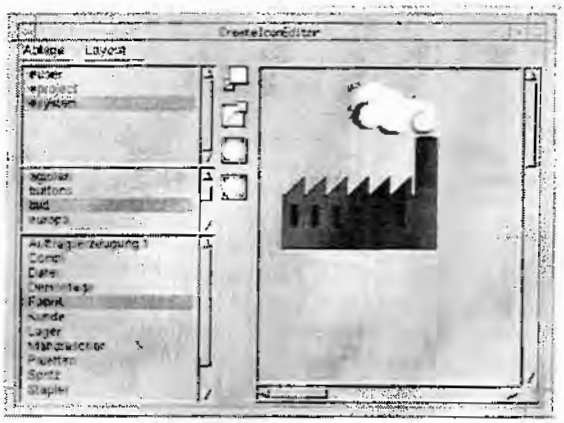


Figure 5: Icon Editor

Using the type information provided within the element editor, dialogs are generated to allow entry and modification of element parameters.

Figure 6: generated parameter dialog

Trace, evaluation and animation

Both evaluation and animation use a generic trace format. The trace is generated by generic calls coded into the elements during the simulation run. An early version supported element specific (load, state) trace events only. The current implementation is based on an ASN.1 compatible file format and supports both element and object specific (creation, attribute change, destruction) trace events.

Evaluation and animation depend to a large extent on the application domain. Nevertheless a set of basic evaluation and animation methods is desirable. Based on a generic framework, domain specific evaluations can be added when needed.

The solution choosen in Create! is to provide a method to generate a stream of events during the simulation and work on this stream online or offline. The object-oriented nature of the underlying Smalltalk system makes it easy to transparently handle both internal and external streams or pipes.

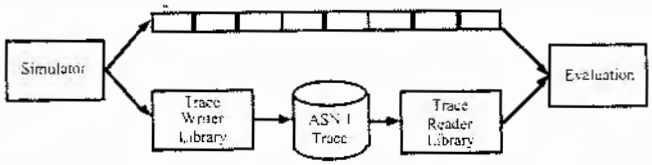


Figure 7: trace event stream

The tracer interface allows element related trace events like *state*, *load* or *object count*, as well as object related events like creation, modification of attributes or deletion.

All events are piped through an evaluation network which consists of single data processors realizing tasks as counting, averaging or other statistical filter functions. Connected to the outputs of this network are visualization components like graphs or animation views. As the evaluation sees events only, it does not depend on the kind of elements generating these events, thus allowing us to provide a generic set of evaluation and animation methods.

Figure 8 shows the results of an amount evaluation based on the increment and decrement trace events. In this particular case, the line chart displays the variation of amounts in a buffer. The same evaluation can be used for every element providing the appropriate trace events and thus allows for an easy integration of new elements into the existing framework.

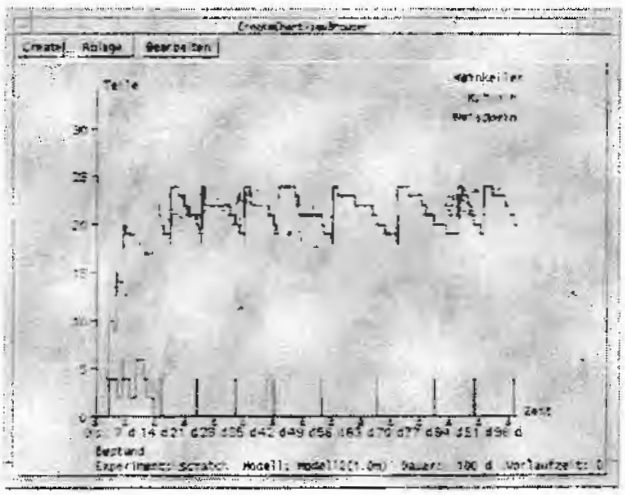


Figure 8: line charts of amounts in a buffer

Model and experiment management

When building models and performing experiments on them, the user in general has to use functions of the underlying operating system for organizing his data. Create! provides the concept of a simulation project to support the management of models, associated parameter sets and results of simulation runs(Figure 9). It relieves the user from coping with different file systems and name conventions on all supported platforms including DOS™ and UNIX™.

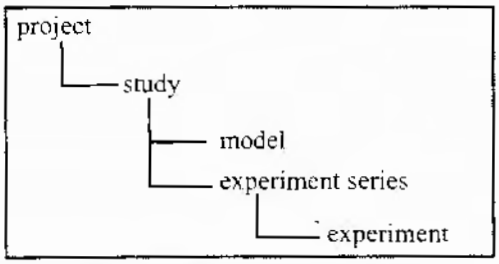


Figure 9: organization of a simulation project

A project is organized into studies which in turn contain models and experiment series. Models come in two flavors: working versions and frozen ones. Frozen models may not be modified and form the basis for experiments, which therefore can be replicated at any time.

Each of these project parts can be created, renamed, duplicated or deleted without the need to use the commands as provided by the underlying operating system.

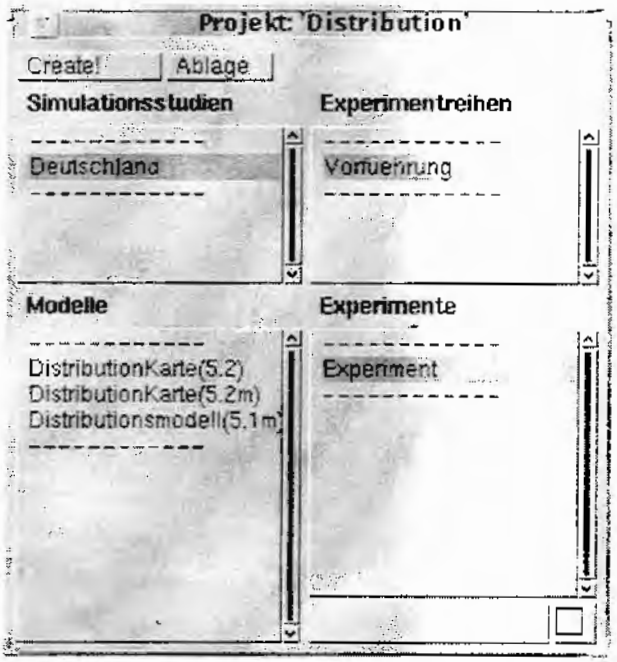


Figure 10: project management window

Projects provide yet another functionality. Each project works as a a container for the installation of element and type definitions, libraries and icons. In this way, parts of an environment which are specific extensions for a simulation project, are encapsulated within this project without cluttering the overall system space containing the default elements and libraries. When entering a project, these components are dynamically loaded into the current environment, and hidden again when leaving the project.

Simulation environments

Currently the following environments have been built using the Create! IDE:

Create!-LSG
simulation for strategic decision support

Create!-Batch
simulation for batch oriented production systems

Create!-Simple
simple single server environment for teaching purposes

Create!-Structure
simulation of logistic systems at the structural level

Create!-LogiChain
analysis of process chains (no simulation!)

All share the same runtime kernel with some minor simulator specific changes in the user interface. Figure 11 shows a snapshot of the model editor within the Create!-Structure environment. The same editor is used in the other tools as well.

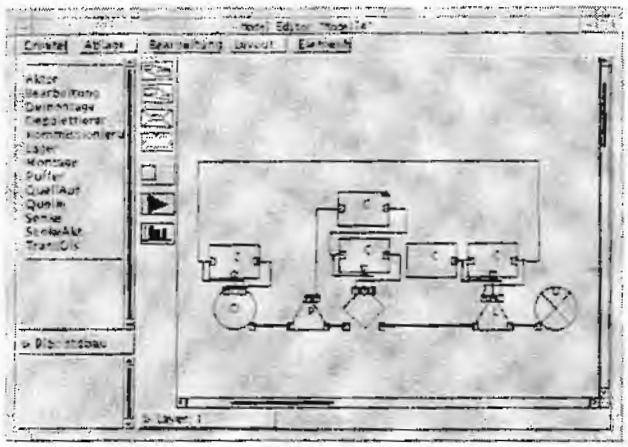


Figure 11: Create!Structure model editor

A somewhat interesting exception is the Create!-LogiChain environment which makes no use of the simulation kernel. It was built using the IDE's abilities to support graphical, element based modelling environments. Elements (activities) are placed on a regular grid. Controlled by parameters provided with the element definitions, elements of the process chain auto-connect to their next neighbors. After constructing the chain, a generic computation algorithm, which in turn calls functions defined within the elements, is run on the model and provides the results of the analysis.

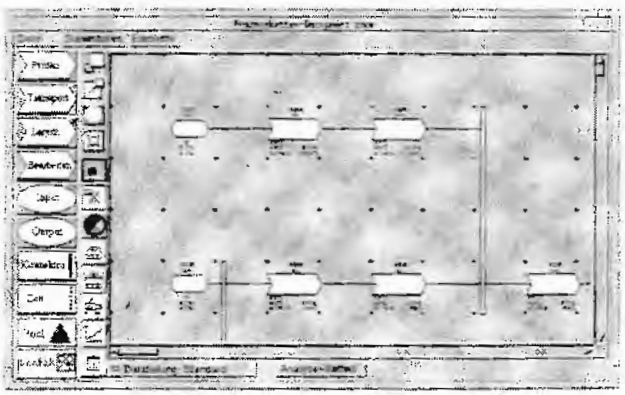


Figure 12: LogChain Designer

References

Carrie A., 1988. Simulation of Manufacturing Systems. Chichester: John Wiley & Sons Ltd.

Cota B.A., Fritz D.G. and R.G. Sargent, 1994. Control Flow Graphs as a Representation Language, in *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, Fl. December 1994, pp. 555 - 559

Paul R.J., 1993. Activity Cicle Diagrams and the three Phase Method, in *Proceedings of the 1993 Winter Simulation Conference*, Los Angelos, Ca, December 1993, pp. 123-131

M. Rüger:, An object oriented divide-and-conquer approach for modelling control logic in logistic systems, In: SCS European Simulation Multiconference, Nürnberg (Deutschland), 1990

Rüger M., Hoppe U. und H. Kirchner, 1990. Objektorientierte Modellierung von Bausteinen innerhalb der Simulatorentwicklungsumgebung „Create!", in*Fortschritte in der Simualtionstechnik*, Band 1, Wien, 140-144

R. Schmidt, A. Schürholz, M. Rüger, CREATE! - Simulation aided development of control software, In: SCS summer simulation conference , Austin (Texas, USA), 1989

Schruben L.W. 1983. Simulation Modelling with Event Graphs, *Communication of the ACM* 26: 957-963.

Schruben L.W. 1990. Simulation Graphical Modelling and Analysis (SIGMA) Tutorial. in *Proceedings of the 1990 Winter Simulation Conference*, New Orleans, LA, December 1990, pp. 158-161

Som, T.K. and R.G. Sargent, 1989. „A Formal Development of Event Graph as an aid of Structured and Efficient Simulation Programs", in *ORSA Journal on Computing*, 1, 2, 107-125

SIMPRO 1984. *GBS - Die graphische Basissprache für das SIMPRO-System*, IMPRO Gmbh Berlin.

Zeigler, B.P. 1984. *Multifaceted Modelling and Discrete Event Simulation*, Academic Press.

World-Wide-Web

The Create! home page can be reached at

http://simsrv.cs.uni-magdeburg.de/~create

CAD-I++: Advanced Interface between CAD and Simulation System SIMPLE++

G. Kronreif
Unseld + Partner
Business and Marketing Consulting • Simulation!
Lerchenfelderstr. 44/9, A-1080 Vienna, Austria
E-mail: kronreif@unseld.co.at

1. Introduction

Attending to the increase in production automation and to the utilization of computer aided technologies in design, planning and manufacturing, simulation technique has been becoming an established tool for planning, realization and operating of technical systems. Initially a tool for providing cover against planning risks, simulation now is utilized in every particular stage of planning and realizing the system as well as for control of the manufacturing process. Because of the unquestionable advantages of this tool, computer aided simulation will gain more importance in future. Considering this, an improvement of the simulation systems in respect to flexibility, interfaces to particular (hardware-) components of the manufacturing systems, user-orientation for modelling and experimentation tasks as well as complete integration of simulation technique into the whole planning cycle will be necessary.

Above all, integration of the two essential CA-techniques for design and layout planning - Computer Aided design (CAD) and (Computer Aided) simulation - seems to be advisable. Generally the CAD drawing of the plant layout contains many data used by the simulation system, as there are information concerning the physical elements of the plant (e.g. machines), information to the flow of materials, transport routes, and others. Further data could be available in PPS (product planning systems) and/or in plant control systems. When opening the access to data stored in other CA-systems and using them for (partially) automated generation of a simulation model, the efforts and costs necessary for a simulation study can be decreased dramatically.

There are different ways to form a combination between CAD functionality and simulation features:

- ⇨ *Simulation within CAD-environment*
  The CAD-system serves as a carrier for an embedded sub-system with basic simulation features. This arrangement is used especially in simulation of robotics systems (e.g. CATIA and CATIA-Robotics).

- ⇨ *CAD functions within simulation environment*
  In general there are some CAD functions available in most simulation systems for design of the animation layout (i.e. animation background and facilities). Therefore making a simulation/animation model consists of two steps: drawing of the animation layout and definition of links between the simulation model and the particular elements of the animation layout.

- ⇨ *CAD-Simulation interface*
  The goal of such a CAD-Simulation interface is to use the full functionality of each tool during the appropriate planning stage. Previous approaches of a combination between CAD and simulation systems are limited in exporting the CAD-layout data for providing a more realistic graphical representation of the simulation model. Hence, the CAD layout is being reduced to a static animation background - the information included in the CAD layout, like transport routes, gets lost.

The CAD-simulation interface presented in this paper is more than an export function for CAD data. Beside drawing of the plant layout, the CAD software serves for definition of a hierarchical structure of

the plant components and for a static analysis of the plant layout. In addition to the data included in the CAD drawing file, information from other sources, like PPS systems or plant control systems, can be processed as well. After development of an appropriate design for the plant layout, a data exchange system creates the necessary structures for automatic generation of the simulation model. For a first prototype we used CAD system AutoCAD. Simulation is executed with SIMPLE++ - the object-oriented structure of simulation models in SIMPLE++ and the advanced features for automatic model generation provided with this system are forming the basis of an efficient data exchange. A second prototype using CATIA as CAD environment is in planning stage.

2. CAD Software AutoCAD

AutoCAD (by Autodesk Inc.) is the most common CAD software used in architecture, electronics, engineering. It is a powerful CAD tool with many applications available (e.g. AutoCAD Designer, AutoSurf, 3D-Studio...). AutoCAD supports various hardware platforms and operating systems (DOS, WINDOWS, WINDOWS NT, HP-UX, SUN, etc). Some important features of AutoCAD used for the presented interface system CAD-I++ are:

- ⇨ Powerful 2D and basic 3D drafting and designing tools.
- ⇨ Graphical user interface with pull down menus, dialogue boxes, accelerator keys, icon toolbar and floating toolbox.
- ⇨ AutoCAD menus can be customized to suit user needs.
- ⇨ AutoCAD Development System **ADS** supports a programming interface to C or C++ code for development of custom AutoCAD applications. Furthermore there are additional interfaces to other graphic standards and database systems (AutoCAD SQL Extension **ASE**).
- ⇨ Performance: fast redraw, pan and zoom speed - important for large drawings.

3. Simulation Package SIMPLE++

SIMPLE++ is the abbreviation for **SIM**ulation in **P**roduction, **L**ogistics and **E**ngineering design and its implementation in C++. The simulation package SIMPLE++ is almost becoming a standard software for object-oriented, graphical and integrated modelling, simulation and animation. All the features - graphical user-interface, system architecture and implementation - are corresponding to the demands of object orientation.

A major strength of SIMPLE++ is the significant increase in productivity when building, changing and maintaining models. The most powerful features of description-block, list and language concepts are provided in a single, integrated simulation environment. Features of object-orientation like inheritance, hierarchy, reusable objects or even models as part of models make this simulation software to one of the most successful simulation systems nowadays.

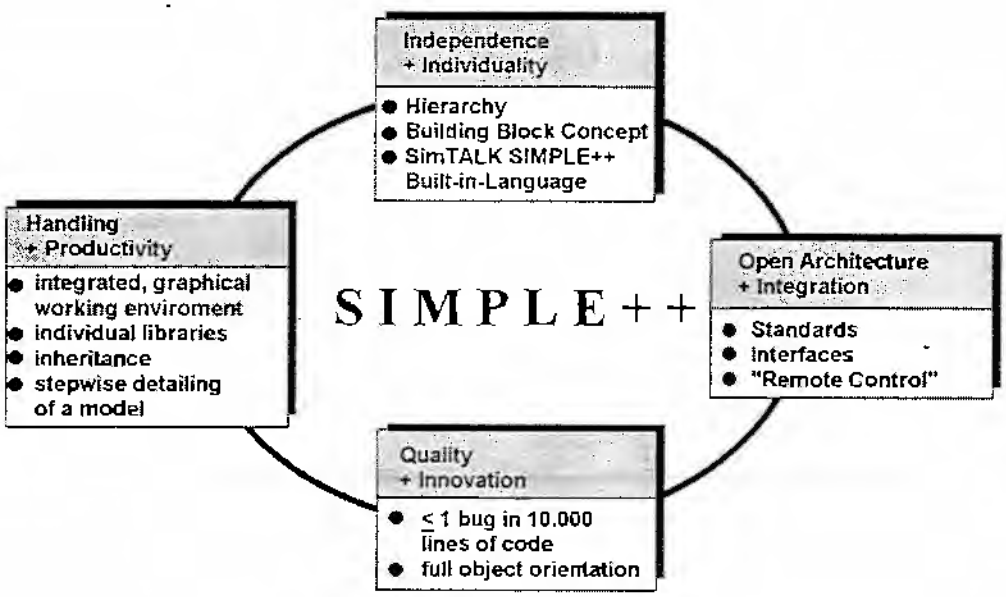


Fig. 1: Requirements in functionality for SIMPLE++

Graphical and intuitive model handling to increase productivity is a key requirement for the widespread adoption of simulation throughout the economy. In SIMPLE++ this is achieved with a unique integrated and incremental working environment, individual libraries (application templates), inheritance and a graphical object interface.

„Users should be independent" - consequently they are free to make an infinite range of tailored application objects which are serving themselves as templates for efficiently creating models. Individuality ensures that the simulation model has a lifelike animation and functionality.

4. Layout generation and simulation with CAD-I++

The presented system CAD-I++ is both, an extension of the features of the used CAD-system as well as an interface between CAD and simulation. To fulfil these tasks, CAD-I++ defines an additional link to a third 'component' (beside CAD- and simulation system): to a database system for storing and processing **non-graphic information (NIGI)**, like instruction sheets, list of orders, characteristic data of the particular facilities, and others.

4.1 Tasks performed within CAD environment

The main task of the CAD system is the generation of the plant layout. With CAD-I++ the user is endowed with all necessary functions for efficient layouting. After definition of the planning area and all existing cut-off regions, the next step is to define and insert the facilities of the plant (i.e. (tooling) machines, assembly systems, storage, etc.). The system CAD-I++ follows to a strictly separation between **graphic data** (dimensions of the facility, positions of the input/output unit) and **non-graphic data** (characteristic data of the facility, like mean time between failure MTBF, mean time to repair MTTR, ...). Graphic data are part of the layout drawing and have therefore to be defined during the layout generation (after input of the facility dimensions, a scaled symbol of the facility is inserted into the planning area). On the other hand, the non-graphic data are stored in the database system and can be displayed in special data sheets on request.

After (or during) the arrangement of the facilities, the planner can define a **hierarchical structure** in order to gain more clarity for the later simulation process. To set up a 'multi-layer' structure of the plant, CAD-I++ defines three groups: *Facility* (i.e. the plant 'hardware'; machines, storage, ...) - *Work Centre* (i.e. group of facilities with equal characteristics) - *Module* (i.e. group of Work Centres; e.g.

cost unit). The described model hierarchy is fully supported in later simulation because of the object-oriented structure of the simulation system SIMPLE++.

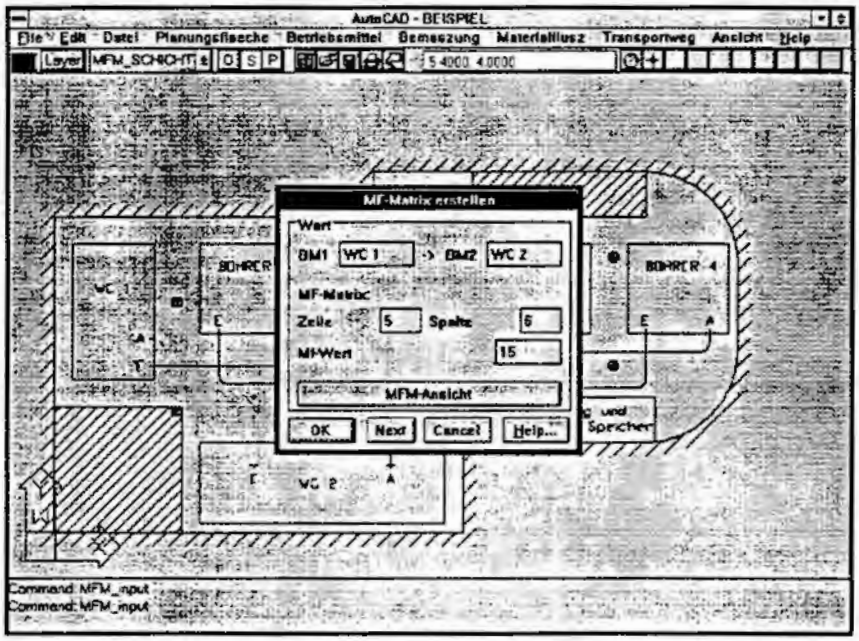


Fig. 2: Interface CAD-I++: Definition of the MFM

Prior to the dynamic analysis in SIMPLE++, the interface system CAD-I++ offers certain functions for static analysis of the flow of material (MF). To supply the system with the necessary data for this task, the matrix of the flow of material (MFM) either can be defined by means of data masks (see Fig. 2) or calculated using the non-graphic information. Combined with functions for graphical input of the transport paths by means of 'Point-and-Click'-Method, the valuation of the generated layout can be carried out by means of a detailed graphical display of the MF relations (Sankey diagrams).

4.2 Tasks performed within simulation environment

After static analysis of the layout, the appropriate CAD data can be exported to the simulation system SIMPLE++ via an interface file. A template model in SIMPLE++, including all the necessary functions for automatic model generation, transforms this data structure into a complete simulation model structured into the same hierarchy defined during layout generation. Beside processing the geometrical data stored in the interface file, the simulation model has access to the non-graphic information as well. No additional effort for modelling and definition of the input data for the particular simulation runs is necessary.

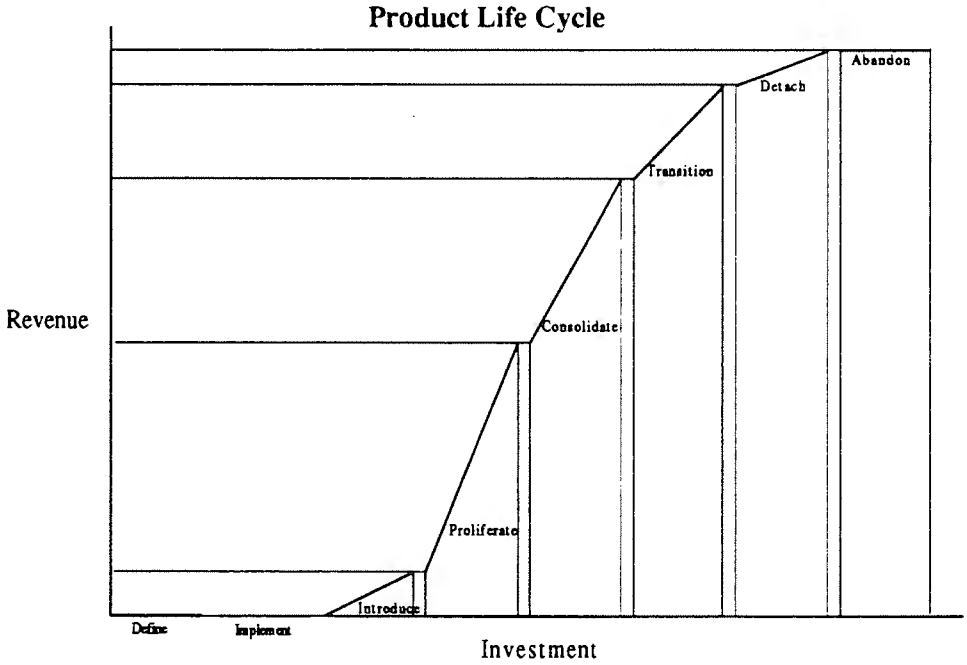
5. Summary

Using simulation technique, manufacturing systems can be designed, optimized and estimated. The plant layout, nowadays often developed by means of CAD-systems, stores the results of the layout planning in a concentrated way. The aim of the presented interface system CAD-I++ is to close the gap between these two planning tools in order to use the capacity of rationalization existing in manufacturing planning. With this interface system, all the planning data can be integrated and used by both systems. The arrangement of the facilities as well as the definition of the transport routes can be done within familiar CAD environment. After static analysis of the plant layout, CAD-I++ offers an automatic generation of an object-oriented, hierarchical structured simulation model for simulation system SIMPLE++ for further dynamic analysis of layout manufacturing process. Additional to the improved efficiency of planning, the integration of CAD information with CAD-I++ should gain more acceptance for the tool 'simulation'.

Insights into Successful Simulation Software Products

Joseph Annino, President
CACI Products Company
3333 N. Torrey Pines Court
La Jolla, CA 92037 USA
Tel: (619) 457-9681
Fax: (619) 457-1184

Organizations need new products just to survive. The need for new products is confirmed by the fact that the profits from today's technology leaders come primarily from products they introduced in the past five years. While the need for products is great, so also is the risk of product introduction -- most products never make it to market and those that do often make little or no profit. With failure so common and its cost so great, a strategy is needed for the successful introduction of products. It is helpful to consider such a strategy in the context of the product life cycle.

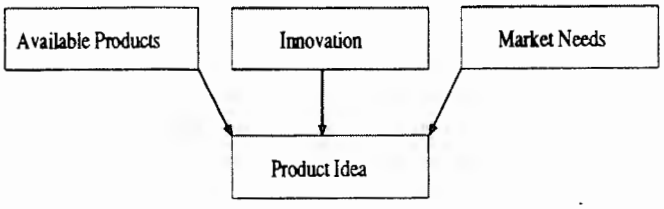


Define

It all begins when some visionary, drawing on talent, ingenuity and a predisposition to new ideas, conceives a product that fits the organization's strategic direction.

The creative idea encompasses an understanding of evolving technology, and a strong sense of what the market needs. Such good ideas are not uncommon, but thinking the idea through to an implementable product plan requires a combination of intelligence and commitment that is rare. In most organizations persistence is also essential. The intuitive visionary is often discouraged by premature requests for financial justification, such as Return on Investment, when little is known about either the needed investment or its potential return.

Product Definition



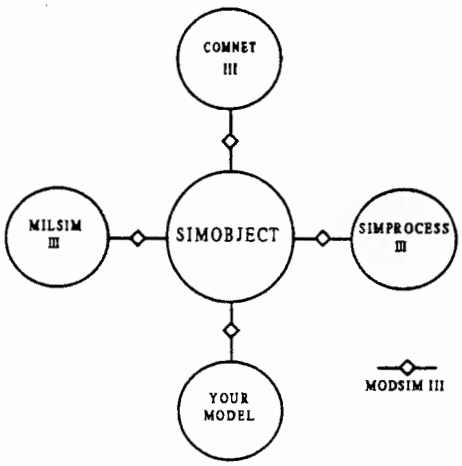
If the visionary has a history of success, the endless hand-wringing over whether to proceed with development can probably be dispensed with. The product idea must, of course, be tested. One excellent market test is to see if outsiders will put up money or other resources. You should contact organizations you regard as prime prospects, and make them a special "for-sponsors-only" investment offer. They get the opportunity to influence product functionality, and are its first users. Offer to repay their investment as a percentage of sales. If necessary, pay more than their investment.

Implement

Eighty percent of all products that are started fail at this stage. A company must have a great idea, a superior product development technology, a talented development team and strong management support, or it will be starting a game that it cannot win.

Time-to-market is an increasingly important competitive factor in high-technology products where the underlying technology advances rapidly, and the markets change quickly. One strategy is to develop a Platform that greatly facilitates the development of a family of products. By providing a jump start on product development, a Platform can sharply reduce the time, cost and risk of bringing a product to market.

Product Platform



A product Platform is collection of the common elements, especially the underlying core technology, that can be used across a range of products. In the above diagram, the SIMOBJECT Platform incorporating simulation and graphics, is specialized using the MODSIM III[1] language to develop members of the product family: COMNET III[2], SIMPROCESS III[3], and MILSIM III.

Differentiation, a way of distinguishing a product's value from competing products, is achieved both at the Platform level, and at the individual product level. The Platform provides advanced concepts and a consistent look and feel throughout the product line built upon it. When the Platform is enhanced, the entire family of products built on it is enhanced.

Of course what makes great products is not technology alone -- technology is perishable -- great products are built by talented hard-working people who are committed to the commercial success of the product. The most successful product developments are those by small, intensely focused teams, working closely and under a tight schedule. The mutual dependence of the team generates the high level of commitment needed to take product ideas to market.

The difficult creative work and long hours that go into successful product development must be rewarded with appropriate incentives. Technical people should be partially rewarded on the product's sales success -- just as salespeople are. This keeps them focused on what's important.

While product development is itself a consuming activity, marketing must begin **before** the implementation is complete. One technique is to organize a Charter Group of users by offering special benefits. Offer a special price and a money-back guarantee to those who commit to buy early. Work with these early adopters to perfect your interface, functionality, installation and support procedures, and sales literature. True, your charter users may point out a lot of things you don't want to hear. You may find that your sales literature is misdirected, or that your pricing is wrong, or your product has no appeal or that its appeal is different from what you thought.

It may be necessary to cut your losses.

Introduce

The best way to introduce a product is to try to sell it! Market research has been mostly poor at predicting the success of a truly new product. People don't know what they want until it exists. Once you have something to show -- even if it is only an initial version -- prospects and customers will let you know what they like and dislike about it. You can then make needed changes.

Even the greatest product cannot succeed if nobody knows about it. You must aggressively promote the product with advertising, direct mail, conferences and trade shows.[4]

Proliferate

Continually infuse the product with significant enhancements and promote relentlessly.[5] Enhancement ideas come from the imagination of developers, often stimulated by the needs of customers and prospects. From the long list of potential enhancements, choose those that expand the market.

As the product achieves success it will attract competitors who know of that success, can examine your product, and will seize the opportunity for low risk profits by entering the market. With an aggressive enhancement program, you can discourage these competitors by forcing them to play catch-up. You want them to look elsewhere for opportunities.

No matter what you do, however, at some point product proliferation will slow and you need to be positioned with a successor product. The pace of change is so great that a large installed base

offers little protection -- look at what happened to Lotus 1-2-3, dBase, Word Perfect, Turbo Pascal, ...

For this reason, technology companies must be willing to initiate the development of advanced products, that compete with their own successful but aging products, or their competitors will surely do it for them. In 1988 Microsoft wisely hired David Cutler, the visionary behind Digital's wildly successful VAX VMS operating system, to lead the development of its NT operating systems -- a superior competitor to their own already successful MS Windows.

Consolidate

Most high-technology products are eventually replaced completely by new technology. Because of the entry of effective competitors, loss of sales momentum, or market changes, revenues will slow down. Ironically, a large customer base also constrains product enhancement by demanding upward compatibility. This constraint creates an opportunity for competitors with a vastly superior product.

In addition, technological limits make successful products vulnerable[6]. For example, Intel based PC's appear to have reached a technological limit. **It costs a great deal to get a small performance improvement.** Intel's CISC (complex instruction set computer) is being threatened by IBM's new Power PC RISC (reduced instruction set computer). Just as the IBM Selectric typewriter, with its wonderful correction tape and type-balls, was threatened and then obliterated by word processors.

Transition

At this stage it is apparent that revenue will continue to decline. Cost cutting should continue in order to maintain profits. Be prepared to offer your customers a transition path to your highly-desirable successor product.

Detach

The decline of revenue continues to the point where the product is no longer profitable, but cannot simply be abandoned because there are customers using it. By raising the support fees and aggressively cutting costs the product may be made marginally profitable. Customers must be moved to the successor product.

Abandon

Customers who did not move to the successor product are notified that they can use the product "as is" but that it will no longer be supported.

---

[1] *A Quick Look at MODSIM III*, CACI Products Company

[2] *A Quick Look at COMNET III*, CACI Products Company

[3] *A Quick Look at SIMPROCESS III*, CACI Products Company

[4] Annino, Joseph, *How to Promote Your Own Products*, La Jolla, CA: Genesee Publishing, 1989

[5] Ibid., Annino

[6] Foster, Richard, *Innovation*, New York: Summit Books, 1986.

Metamodelling of Simulation Tools

M. Frank, Technische Universität Dresden, Fakultät Informatik

1. Introduction

Usually, three classes of discrete simulation software are distinguished

- simulation languages SL (SIMSCRIPT, SIMULA etc.)
- general purpose simulation systems GPS (SLAM, GPSS etc)
- special purpose simulation systems SPS (DOSIMIS, SIMPLE++, WITNESS etc.)

This classification is mainly based on the different levels of predefined means for model description. For instance, SPS offer predefined modules describing subprocesses of a special application area whereas GPS offer blocks describing events. The greater complexity of SPS-modules leades to decreasing expenses for aquisition but to a certain loss of flexibility for application, too. For this reason SPS designers frequently don't realize the modularity consequently. Generally, the modelling means offered force the user of SPS to describe the process to be simulated not only by (more or less complex) modules of typical subprocesses but also by additional statements of a programming language.

This hybrid way of model desciption confines the loss of flexibility but leaves the expense for aquisition and efficient handling of SPS on such a level that specialised users are required. For a more simplified utilization one should look for another way to ensure flexibility without renouncement of consequent modularity.

Obviously, there are two ways:

- the creation of SPS based on GPS on the one hand /1/
- the extension of SPS by means for descriptive module insertion.

2. Meta modelling

Some years ago we started investigations in this direction. They have shown that , to a certain extent, flexibility of a pure modular SPS can be guaranteed by metamodelling. Metamodelling means the description of all elements offered for model building on an abstract level. The subject of investigations mentioned is the simulation system TOMAS, the modelling conception of which is consequently modular. Referring to TOMAS, metamodelling is particularly the formal characterization of the modelling modules and functions.

The TOMAS' modelling modules are called operators and can be combined to an operator net flowed through by operands.

The entities which are called operands represent (passive) temporary elements (material parts, messages, jobs etc.) which after its generation move through the operator net in

order to be processed, transported, stored or distributed etc. by operator objects. An operator object is derived from a predefined operator type by specifying its parameters. The specification is supported by predefined functions.

Model building requires the following user activities, only:

- selection and denomination of suitable operators
- positioning of operator symbols on the screen
- drawing the needed connections dependent on the possible flow of operands
- specification of all denominated operators via its parameters.

The TOMAS metamodel describes formal features

- of models that can be build (global parameters).
- of the operators offered
- of the predefined functions

Global model parameters to be described are

- the maximal number of operator objects which are allowed for a model
- the maximal number of preceding and succeeding objects of an object
- the number of operator types offered

The operator description refers to its coupling conditions and its parameters including for instance:

- the minimal and maximal number of input respectively output valencies
- the number of freely usable valencies, (freely usable means that the connection with different operators is allowed; the complemental subset consists of valencies for special operators, only)
- the subset of allowed adjacencies (possible neighbour operators)
- the total number and the types of parameters (there are four types: alternative, scalar, function and list parameters
  - ∇ each parameter type has to be described by some further formal features respecting parameter dependences etc.

Metamodelling of functions includes for instance

- the function type and its general denomination
- the number of the assigned syntax graph
- the number of function parameters

  Per each function parameter are to be assigned
  - ♦ the type (real, integer, ...)
  - ♦ the assigned scope of values (to be selected from a predefined set)
  - ♦ a possibly given speciality (as for instance 3-σ-Intervall)

Dependent on the function type further features are to be fixed.
Summarizing one can say: the meta model defines the model room. That is the set of all models that can be build based on the defined metamodel.

3. System and surface generation

Based on this conception of metamodelling we designed an additional component for system generation and partial adaptation. Using this generation component the TOMAS administrator can derive a desired system version by

- changing the global parameters
- deletion respectively addition of operator parameters
- deletion respectively addition of operators or functions

The component is able to adapt the surface for the model building automatically except the type of parameters or functions added don't yet exist. The basis for this adaptation are predefined widgets of menu and formula windows (OSF/Motif).
All data describing the meta-model are stored in a comprehensive file called GENA.DAT. Under the precondition that the parameters (of operator or functions) added don't extent the already defined set of parameter types, the generation component is able to adapt the model building component automatically. The basis for the automatic adaption are predefined procedures for the generation of windows and window hierarchies by using predefined widgets (OSF/Motif).
Morever the generation component is used while the process of building a special model is running in such a way that all windows needed for this model in order to input the model data are dynamically generated. For this reason there is no additional expense for adaption the model building surface to changes of the modelling means as it would needed in the case of a static surface definition. On the other hand, only such windows are produced which really are needed for the special model. Only if a not existing parameter type is demanded the administrator has to extend the generation component.
Recently, such an extension has been realized in order to equip TOMAS with an interface to the post-run-animation-system PROOF. PROOF needs an ACII-Trancefile of all events to be animated. For this reason the generation component has been extended by

- a global alternative parameter allowing to decide whether post-run-animation is desired or not (This parameter is important for the dynamic generation of the model input surface)
- additional alternative parameters of the differant operator types describing all events which can reasonably be protocolled for animation
- a specification file describing the protocoll text of the animateable events in accordance to the PROOF owned syntax
- routines for the generation of windoes respectively window hierarchies allowing the choose of events which are really desired to be protocolled .

Of course, the simulator has to be exented, too, especially by routines for the tracefile generation based an the specification file, mentioned. Based on the concept of meta modelling under discussion the expense for the extension mentioned was comparitive small.

4. Concluding remarks

Up to now. the means for system extension has been focused to the surface for model development and manipulation, only. That´s why our experiences have shown that functional extensions cause much higher expenses for adapting the model data files and the surface for model building than for changing the modular simulator program.
The idea of meta-modelling has been elaborated in detail by Stephan Strohmeyer /2/. Substantial contributios to its actualimplementation have been realized by Simone Bersiner /3/, Kathrin Sippach /4/, Alexander Hartmann /5/.

References:

/1/ Apsel, T.: Attributierbare Modellkomponenten zur Unterstützung von simulationsunerfahrenen Anwendern mittels eines universellen Simulationssystems.
in: Biethahn (Hrsg.): Simulation als betriebliche Entscheidungshilfe: Neue Werkzeuge und Anwendungen in der Praxis; Braunlage 1995, S. 63-74

/2/ Strohmeyer,S.: Theoretisch-methodologische Probleme der Entwicklung und Nutzung diskreter Simulationsmodelle und daraus resultierende Konsequenzen für die Gestaltung des Zugangs zu fachgebietsorientierten Simulationssystemen.
Dissertation A, Technische Universität Dresden, Informatik-Zentrum 1987

/3/ Bersiner, S.: Neukonzipierung der Generierungskomponente des Modellierungs- und Simulationssystems TOMAS/32.
Diplomarbeit, Technische Universität Dresden, Fakultät Informatik 1994

/4/ Sippach, K.: Untersuchungen zur Anpassung der Oberfläche der TOMAS-Modelleingabe an veränderte bzw. neue Modellbausteine.
Diplomarbeit, Technische Universität Dresden, Fakultät Informatik 1995

/5/ Hartmann, A.: Entwurf und Implementierung einer Schnittstelle TOMAS-PROOF zur Post-Run-Animation eines TOMAS-Modells.
Diplomarbeit, Technische Universität Dresden, Fakultät Informatik 1995

Modular Application Objects: Closing the Gap between Flexibility and Ease of Modelling

Dietmar Geuder
AESOP GmbH, Königsstr. 82
70173 Stuttgart, Germany
E-mail: geuder@aesop.de

ABSTRACT

Compared to pure simulation languages modern graphically oriented simulation environments offer a comfortable efficient user interface. This increase in modelling efficiency, however, came at the expense of modelling flexibility and scope of application. In this paper, we will propose an approach which offers the ease of modelling of building block oriented simulators while keeping the flexibility of simulation languages.

1 MOTIVATION

During the last 40 years simulation systems have come a long way. Starting with general purpose programming languages, explicit simulation languages optimised for modelling and simulation evolved. Nowadays graphical simulation systems and modelling environments are widely used (Schmid, 1994).

With pure simulation languages, modelling requires a comparably high level of programming skills and is therefore mostly performed by specialists. Implementing the model on this level provides for the substantial flexibility to model the real world to as much detail as necessary. Also, a vast variety of application areas can be covered with the same language.

Modern graphical simulation tools being targeted for specific application areas usually offer a standard set of building blocks or templates. With these a model can easily be built graphically on the computer screen with a few mouse clicks. The model is then adjusted to the real system by setting default parameters and selecting standard operation rules. To allow this, these tools are mostly domain restricted and are shipped with one or several sets of templates for specific application areas (Hillen, 1993; Banks, 1994). Obviously this approach is by far more powerful as far as speed and ease of modelling is concerned. Due to the graphical user interface, models can be built by practitioners with only limited simulation experience and no programming skills.

This increase in efficiency, however, leads to a significant loss in flexibility. The fixed set of application related templates restricts the variety of systems and the detail which can be modelled. With existing systems one can thereby either find high flexibility and a wide scope of application or a comfortable, easy to use modelling environment. An optimal solution would incorporate the advantages of both type of systems:

- an integrated graphical user interface
- application specific building blocks or templates

for the *ease of modelling* and

- application objects configurable and changeable by the user
- ability to create new application objects in an easy way
- availability of an integrated programming language for specific control strategies

for *flexibility and application scope*.

While conventional systems compromise in either efficiency in modelling or in application scope, an ideal toolset will provide the flexibility to cover different application areas as detailed as necessary by fully adaptable and reconfigurable application objects (see Figure 1).

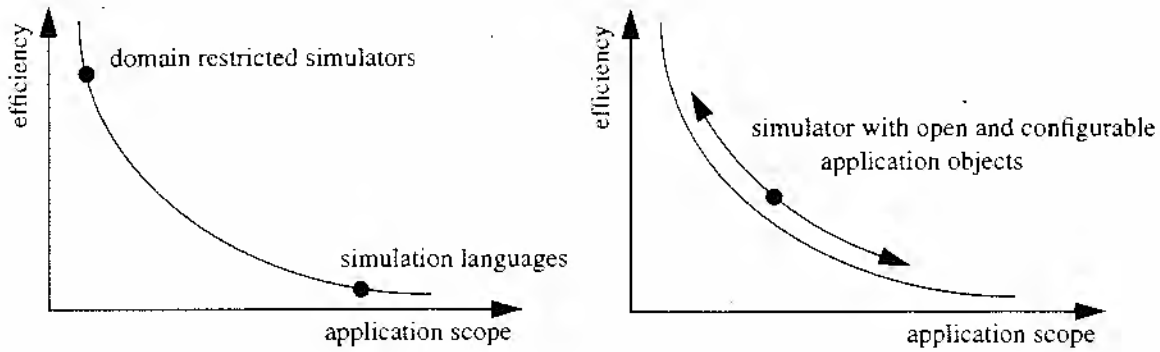


Figure 1: Configurable application objects open to the user combine efficient modelling with a wide application scope.

2 FLEXIBILITY BY MODULAR APPLICATION OBJECTS

SIMPLE++ an object oriented modelling and simulation environment offers hierarchical modelling, generic objects, an integrated programming language and a graphical user interface. This serves as a basis for implementing an approach which incorporates both types of features described above.

2.1 A Layered Approach of Open Objects

Instead of fixed high level building blocks we propose application oriented objects which are composed of generic objects on a lower level. Their functionality is determined by the default behaviour of the basic objects and their parameter settings. Application specific functionality can be added by high level program statements while domain specific data is represented by user defined attributes and variables.

By this process building blocks are created which are specific for a given application area. With definable control and data structures they can be as detailed as necessary. Generic basic objects together with configurable attributes and an integrated programming language provide for the required flexibility to cover a wide variety of application areas. With a hierarchical modelling environment the functionality can even be refined in an entire hierarchy of levels. Figure 2 shows a library containing basic objects and application objects. The internal structure and the dialogue mask of a buffered processor is depicted as an example.

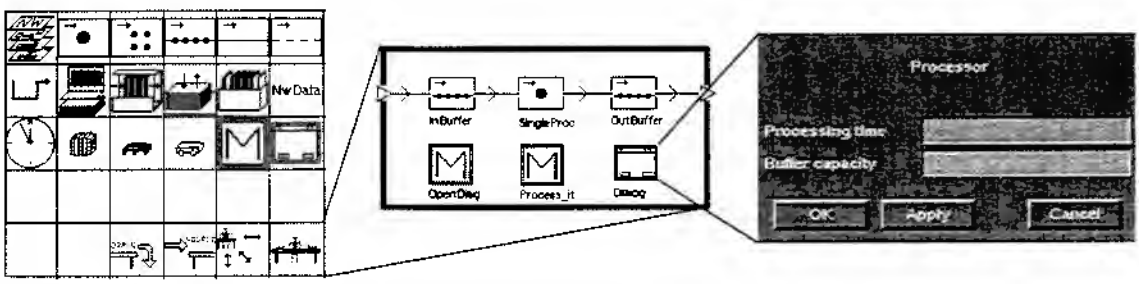


Figure 2: Example for a processor with buffers as an application object being composed of basic objects. With its dialogue window it appears as an integral high level application object.

These application objects are an integral part of the modelling environment and are represented as classes of objects in the library. Thanks to configurable dialogue boxes, these modular objects look like standard fixed templates to the user and can be employed accordingly.

That is, they can graphically be inserted in a model, and the parameters and control rules be set as necessary. This guarantees the user friendliness and ease of modelling modern domain restricted simulators offer.

2.2 Flexible Modification of Application Objects

While for modelling the application objects resemble standard parametrisable objects, they are nevertheless open for the user for modification. Should at any time additional parameters or operating rules be required or should the overall structure of the building block need to be changed, this can easily be done to any detail as the internals of each application object are fully accessible. The same graphical integrated modelling capabilities apply for all hierarchical layers. Changes like adding, deleting or redefining the structure or functionality are easily possible and there is no need to fall back to a different description mode.

Starting with a given set of application objects users can thereby actually create their own objects targeted to their individual modelling purposes by modifying the internal structure, the number and type of parameters and control strategies. In this way one is not restricted to a given set of parameters or attributes being predefined for domain restricted building blocks. Also, additional hierarchy levels may be introduced with this concept which allows incremental refinement of existing models.

In an object oriented modelling environment as offered by SIMPLE++ these changes will automatically be propagated to the instances of the objects in the model by inheritance. That is, added or modified functionality of a class in the library will automatically become active for all instances of that class in a model. This provides for very effective and productive modifications as necessary for common ‚what if' scenarios. Also, a full set of variants can be created for these application objects by deriving child objects and making the necessary modifications.

2.3 User Interface for Different Skill Levels

The hierarchy and the configurable dialogue boxes as described above make the user interface adaptable to different skill levels. Depending on the expertise of the user, he can simply use the application objects as they are and adjust them by setting parameters and selecting operation rules. He will thereby stay on a comparably high level requiring only limited practise. More experienced users can also change the internal structure of the application objects by opening the building block and graphically changing it in the same way the global model is changed. If necessary, the entire functionality of the building block can be adapted by progressing to the next lower level and programming new control structures with the built-in language.

Different views of the simulation environment and degrees of changeability can thus be distinguished as depicted in Figure 3. Information hiding of data irrelevant to certain user groups is directly supported.

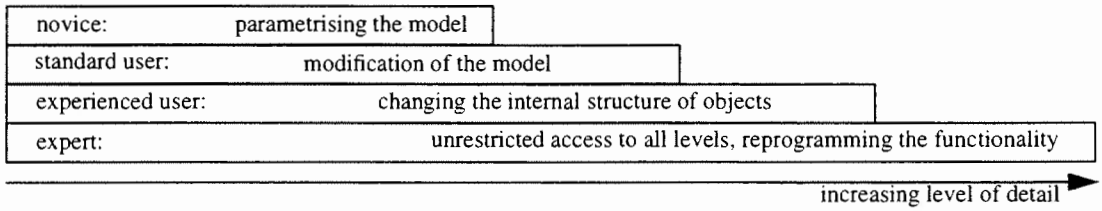


Figure 3: Different views and access levels of a model are supported by hierarchy and user defined dialogues. Information hiding and access restrictions can thus be implemented.

3 APPLICATION EXAMPLES

Various sets of modular application objects were created using the above approach. These include libraries for manufacturing, warehousing, business process reengineering, and staff and shift patterns. In Figure 2 two sets of application objects for chemistry and transportation are shown as an example.

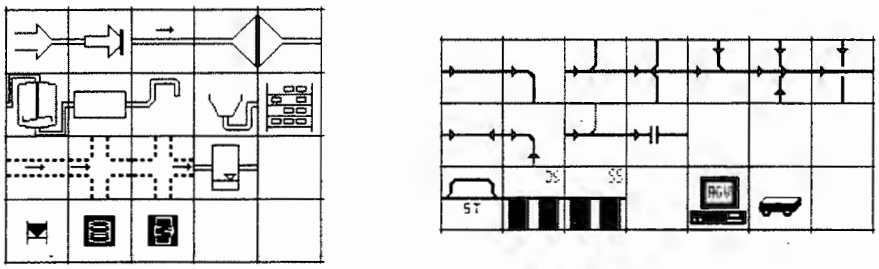


Figure 4: Application objects for modelling of chemical processes (left) and for an AGV system (right).

The application objects in the above figure were created with the SIMPLE++ modelling environment. Based on its graphical user interface for modelling it offers the possibility to create new application objects (AESOP, 1995). For this purpose generic basic objects are available which fall into different categories such as processing elements, entities, control structures, data structures, displays and dialogues, and interfaces. An integrated high level programming language with a variety of data types guarantees the implementation of specific functionality.

The application objects shown above are based on the same generic objects. These are aggregated in 1 up to 3 hierarchical layers depending on the complexity of the object in question. The functionality of these objects was specified in collaboration with engineers from the respective field and the graphical representation (icons) was chosen accordingly. By a double click on any of these objects, a dialogue box will open which displays only the parameters and menus which are relevant for the planner or engineer using the model.

Sets of application objects for different fields are shipped with SIMPLE++ and were used in various projects. Being modular and open for the users, they were also modified and adjusted to the specific requirements of simulation studies. In addition they serve as templates for variants and for new objects to be created.

4 RESUME

Based on a hierarchical approach and generic objects a variety of high level application objects can be created for different domains. With definable dialogue masks these objects are as easily usable and configurable as the fixed templates modern domain restricted simulation systems offer. However, being modular and open for the user, they offer the flexibility to change any part of them as far as necessary to customise them for the specific systems to be modelled. The modular application objects thereby combine the ease of use of domain restricted simulators with the flexibility of programming languages.

REFERENCES

AESOP GmbH (1995) *SIMPLE++ Reference Manual Version 3.0.* Stuttgart, Germany

Banks, J. (1994) *Software for Simulation.* Proc. 1994 Winter Sim. Conf., Orlando, FL

Hillen, D.W., Werner, D. (1993) *Taylor II Manufacturing Simulation Software.* Proc. 1993 Winter Sim. Conf., Los Angeles, CA

Schmid, B (1994) *Simulationssysteme der 5. Generation.* In: Fortschritte in der Simulationstechnik vol. 6, Berlin 1994

Micro Saint Simulation as a Means of Evaluating Optimal Conveyor Management Strategies

by K. Ronald Laughery, Ph.D.
Micro Analysis and Design, Inc.
Boulder, Colorado USA
Work was performed while Dr. Laughery was
at the University of Nottingham, Nottiingham, United Kingdom

In recent years there has been a great increase in the use of simulation as a means of evaluating manufacturing investment alternatives. Dunlop Cox Ltd., designs, develops and manufactures vehicle seats and seat mechanisms in Nottingham, England. The factory is structured on flow line, cellular manufacturing basis and it operates using Just In Time principles. Simulation was chosen as one of the tools to improve the performance of the factory. Of particular interest was the conveyor and paint plant that was used in the manufacturing process. This was seen as the limiting factor on total plant productive capacity. Several conveyors are used to transport seat components on carriers (flight bars) from the production cells to the paint plant, which is a bottleneck of the factory, and back to the cells again. The aim of the project was build a model which could then be used as a tool to discover ways that the conveyor hardware, software, and or usage practices could be changed to improve the efficiency of the plant.

Of particular interest in the study reported here is the use of dynamically programmable flight bars. The current system used a movable bolt on each flight bar to indicate to which particular production cell it was assigned. An alternative was to make an investment in a hardware and software system that would allow the allocation of flight bars to production cells to be dynamic as the requirements of the cells vary on a moment-to-moment basis. However, this would require a substantial investment. The question addressed in this study was how great an increase could be obtained by this potential investment.

The Conveyor System

The ten production cells at Dunlop Cox Ltd. share common paint plant facilities. Several conveyors transport the seat components from the cells into the paint plant, through the painting process and back to the cells again for assembly. The overall layout of the conveyors is illustrated in the Figure 1. At each cell, components are manually loaded onto load jigs hung form carriers (flight bars), which stop on a siding conveyor. From here they are released manually onto the main factory conveyors when the operator pushes the release button. A flight bar takes the components throughout the entire system and returns the painted components to the siding of their home cell, where they are unloaded and new parts take their place.

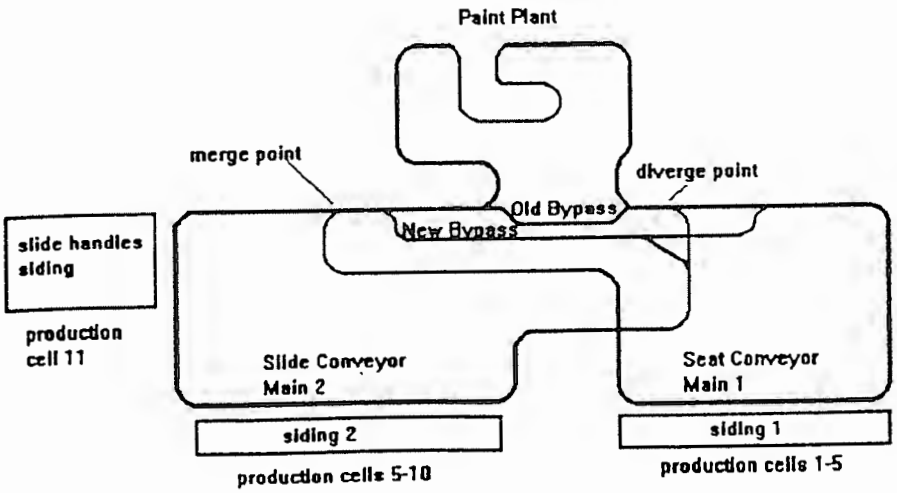


Figure 1. Overview Layout of the Conveyor System

To add to the system's complexity, there are two main conveyors in the system, each dedicated to one half of the factory. The conveyors merge before the paint plant and diverge again after the painting area as shown in Figure 1. The paint plant itself has its own conveyor. There is a siding conveyor on one side of the shop for five cells, four of which are involved in production and one in maintenance. The other half of the factory has two siding conveyors, one for five adjacent production cells and the other for a remote cell which requires a different siding structure. The conveyor system also has many other small aspects of operation that are too lengthy to mention here but are important to the efficient operation of the system. These aspects of the system have evolved over several years and are important contributors to plant performance and, as such, needed to be included in the simulation analysis.

Each cell has a number of flight bars allocated to it and no other flight bars can enter the cell. The conveyor control system is able to direct the flight bars into the correct cell by sensing the fixed bolt on the side of the bar. The investment being considered was a method of dynamically allocating flight bars to each cell during the course of operations based upon moment-by-moment supply and demand.

The Simulation Project

The steps followed to build the simulation and answer the question of "Would a dynamically programmable system be worth it?" were as follows:

- define the system to be modeled
- build the model
- collect data
- validate the model
- run experiments using the model
- revise the model and repeat experiments

Below, these steps are discussed as they were followed in this project.

Defining the System to be Modeled - Some previous work done on modeling the conveyor system provided a starting point for the Micro Saint modeling effort. This was supplemented with the plan of the electrical services layout of the conveyor system. Additionally, we make frequent visits to the company. Conveyor maintenance personnel also turned out to be helpful in gathering information about the system.

Building the Simulation Model and Collecting Data - Once a general understanding of the conveyor system had been attained the model was developed and tested gradually using the Micro Saint computer modeling system. The first version of the model including only one of the two factory conveyors and the paint plant. The model was extended as progress was made in developing the conveyor logic and collecting information about the system.

Once the building of the physical structure of the conveyor system into the model was completed, the main concern was how the system is operated by each individual cell. To ensure that the model adequately represented the real system, it was necessary to study the behavior of each individual cell by interviewing the cells leaders. The following aspects of each cell's behavior were included in the simulation:

- shift working hours
- beginning/end of shift practice (actual period of loading/unloading the flight bars)
- flight bar allocation
- maximum number of flight bars that fit onto the siding
- number of operators (loaders)
- criteria to release an empty flight bar

The development of the logic for dynamic flight bar allocation was carried out during the last stages of model development. The strategy used for assigning flight bars dynamically was based upon input

from the Managing Director of the plant as well as insights gained during model development. Also, during model testing, insights for better allocation schemes were gained and subsequently implemented in the model that was used for experimentation.

As we were gaining this knowledge and data about the system, we were concurrently building and refining a Micro Saint model of the system. In the Micro Saint model of the system, the conveyors are represented as chains of tasks, which are sections of the conveyors. Tasks form the top level network diagram of which the final version is shown in Figure 2. Some of these tasks were actually decomposed into subnetworks in the final model.

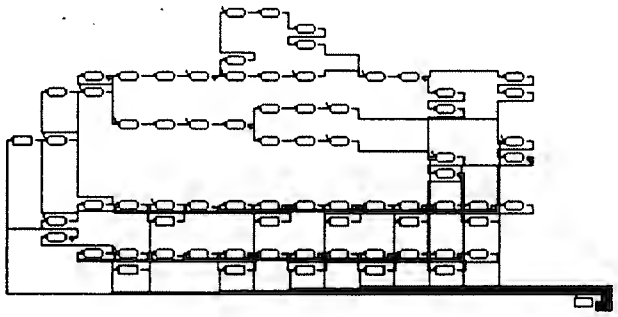


Figure 2. Micro Saint Network Diagram

The model simulated:

- The flight bars as they flowed through the system
- All flight bar queuing rules and release as in the actual conveyor.
- Routing based upon the system logic (which could be readily changed in the model).
- The potential for dynamic flight bar allocation to the work cells.

Data generated by the model included 1) the percentage utilization of the paint plant, 2) the number of loaded flight bars that were painted per unit time, 3) the number of flight bars that went through the paint plan either empty or with the same load twice, and 4) the average flight bar cycle time.

Validation and Verification - Throughout model development, validation and verification practices were carried out on a continuous basis. The validity of the model was improved as knowledge of the system increased through several interviews and visits to the shop floor.

Experimentation - The behavior of the model in eight different scenarios was studied varying three factors, 1) the existing fixed flight bar allocation systems vs. dynamic flight bar allocation, 2) the current production rate vs. a 25% increase and 3) a consistent material flow onto the conveyor vs. a variable material flow.

Results - The simulation results are presented in Table 1. The first three columns of this table describe the experimental condition. The fourth column of the table titled "paint plant utilization" is an aggregate measure of the number of cells in the paint plant vs. total paint plant capacity if used 24 hours/day. Utilization of the paint plant remains at around 50-60% because the shut-down hours are included in calculations. The fifth column entitled "number of painted flight bars" shows to the number of flight bars painted excluding empties and those flight bars going through the process twice. When a flight bar is either painted empty or painted twice because the cell was full upon return of the flight bar, this is reported in the sixth column titled "number of flight bars painted empty/twice." The seventh column refers to the overall average cycle time of the flight bar.

We were also able to plot measures over time, allowing comparisons such as that shown in Figure 3.

Table 1. Statistics Generated by the Simulation

Production level	Material flow variation simulated?	Flight bar allocation scheme	Paint plant utilization	Number of painted flight bars	Number of flight bars painted empty/twice	Average flight bar cycle time
Current	No	fixed	52.64	946	45	98.5
		dynamic	50.69	948	5	97.7
	Yes	fixed	52.64	946	45	98.5
		dynamic	50.59	948	5	97.7
Current +25%	No	fixed	61.67	1138	24	127.3
		dynamic	61.67	1147	15	130.8
	Yes	fixed	61.51	1122	37	119.7
		dynamic	61.67	1147	15	127.6

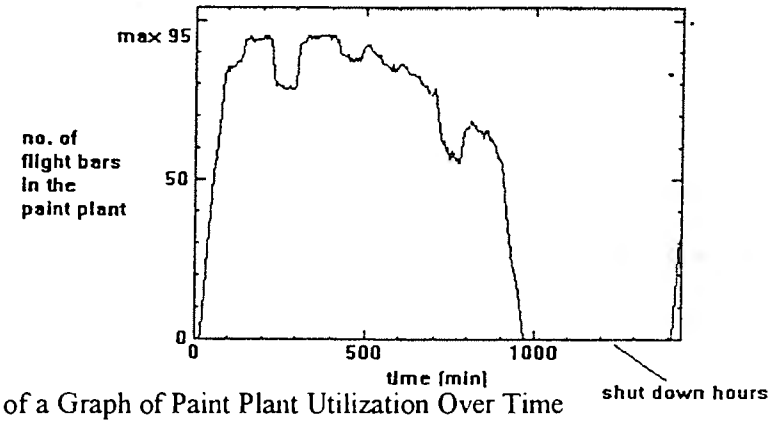


Figure 3. Example of a Graph of Paint Plant Utilization Over Time

Discussion of the Results

With the existing flight bar allocation, output goals are not fully achieved in the +25% production models. Using a dynamic flight bar allocation scheme gave somewhat better output figures, but the number of loaded flight bars released still did not reach the necessary value to sustain this production rate.

The logic used for dynamic flight bar allocation resulted in some improvements to the operations of the conveyors. The dynamic flight bar allocation decreased the proportion of empty flight bars compared with that of all four scenarios in the fixed allocation models - the current and +25% production models with and without material flow variations. There were significantly fewer empties released from the sidings and the flight bars passed by the home cell more rarely. Thus, the results indicate a more efficient use of the conveyors is possible.

However, the increased formation of queues in +25% production models, especially the one at the end of the slide conveyor, occurred even in a greater extent in dynamic flight bar versions compared with that of fixed allocation models. Hence, final conclusions about the usefulness of dynamic flight bar allocation cannot be made without a more detailed consideration of the queuing effects and potential modifications to the flight bar allocation scheme.

Summary

In all, this study provided both useful data and insights regarding the potential effectiveness of a significant capital improvement to the Dunlop Cox conveyor system. Of course, the decision to implement the dynamic flight bar allocation scheme will take into account many other factors such as projected demand, other potential plant limitations, and business constraints. However, this study did provide the decision makers with valuable projections on which these decisions can be based. Additionally, if the scheme is implemented, the simulation model can be used to fine-tune the allocation scheme at virtually no cost.

Quantitative Design of Material Handling System Using Predictive Simulation Modeling

Chris J. J. Lu, K. H. Tsai
Advanced Technology and Research Corp., Springpointe Executive Center
15210 Dino Drive, Burtonsville, MD 20866-1172, USA

and

Drs. Jackson J. S. Yang & Yu Michael Wang
University of Maryland, Mechanical Engineer Dept. College Park and Baltimore, MD

Abstract

The design of material handling system (MHS) includes the interface with existing automatic material processing machines (AMPMs), the comprehension of entire facility operation, the development of system control architecture, and the integration of all elements in the system. The staging buffer size, type of staging equipment, and required throughput of transportation links among machines are considered as the main design parameters of MHS and should be addressed first. This paper provides a generic numeric method to determine the above design parameters using predictive simulation model. Moreover, other interested system characteristics can be analyzed by using this method as well. All the material handling devices met the found requirements will be capable of handling the load of material in the automated system and may be considered as the candidate equipment. The criteria of selecting the final design and facility from these candidate equipment are based on the allowance of capital, floor space constrains, equipment reliability, and designer's preference.

1. Introduction

The plantation of automatic material processing machines, the installation of material handling equipment (MHE), and the integration of automation system are considered as the three major design elements in any type of facility automation. A generic method using the accelerated simulation was proposed to determine the type and number of AMPMs and to optimize the operation schedule [1]. The traditional design of MHS involves manual recording and experts consulting. This type of analysis is usually too subjective and unsystematic. Simulation methods are developed to overcome this obstacle.

A numeric determinative method, predictive simulation, is used in this study to find the required design parameters of MHE, such as the required minimum buffer size of the staging device and throughput capacity of the transportation equipment. Moreover, the type of staging device, FIFO or intelligent retrieval (I.R.), may be determined as well. Once the parameters of MHS and AMPMs are obtained, the theoretical optimized facility layout can be achieved by the topology optimization concept [2, 3]. A life cycle automation system can be then integrated and implemented by applying the accelerated simulation, the predictive simulation, topology optimization concept, Real-time Control System methodology (RCS), and facility layout CASE tool [4, 5].

2. Predictive Simulation

A successful system design must start with a set of good data. Thus, to acquire fundamental knowledge of the entire factory is the first step while designing the MHS. Certain common characteristic information may be categorized as a standard format for all types of MHS. The required design parameters can be determined once the standard dada set is ready.

The predictive simulation model applies the Markov chain simulation that predictively determines the paths of material from machines to machines. The predictive simulation can be run faster than the accelerated simulation and real-time simulation [1]. The assumption of the predictive simulation is that every object in the model will behave as its pre-assigned characteristics. Every automatic machine is operated at regular condition and each material is sent to an appropriate destination and is processed at the earliest period as soon as this material is handled by the AMPM. The predictive simulation is composed of a series of discrete sampled simulation period. However, the simulation is pseudo-continuous within each sampled period. Predictive simulation is perfectly suitable for analyzing a complex and large scale system while applying the traditional simulation is too tedious. The merits of predictive simulation are its extremely high simulation rate and simplicity of simulation algorithm.

2.1 Nodes Modeling

In this simulation model, AMPMs and MHE are considered as nodes and links in the network graph, respectively. Each node serves as an individual device emulator to simulate quantitative characteristic of the hardware device in the system. **N** different input buffers are assigned to each node if there are **N** operation plans exist in the designed facility. Only one operation is allowed to run at a node within one sampled period. Each node handles the appropriate material according to its operation plan. The simulation of handling event is called node operation and is modeled by applying the Monte Carlo method and Queuing algorithm. After the material has been handled, the characteristic of that material will be updated by the node. According to this updated characteristic, the material is then sent to the appropriate input buffer of the next destination node according to the earliest availability criteria (EAC).

The data structure of a single node, $\mathbf{Node}_{j,k,l}$, is a three dimensional array, where j, k, and l represent the operation plan (**OP**), sampled period (**P**), and simulation properties, respectively. Four entries, material processing capacity, received material volume, proceeded material volume, and miss-cutoff material volume of $\mathbf{OP}_j$ at $\mathbf{P}_k$, are included in the simulation property. The data structure of the nodes model is a four dimensional array and can be expressed as $[\mathbf{Node}_{j,k,l}]_i$, where i is the number of nodes.

The maximum volume that can be handled by a node (automatic machine) during a single simulation sampled period is called the capacity of that node. The processed material and the received material volumes are updated as soon as the material is handled by the source node and assigned to the new destination node. The node simulation mechanism increases the processed volume in the source node and the received volume in the destination node by one unit.

The transportation time for every material between nodes is not considered because the perfect material handling system is assumed [1]. Therefore, to achieve the objective (maximize the throughput of the material flow) is to send the material to the earliest available automatic machine resource. Because the volumes of throughput, capacity, and the processed material of each node are "real-time" updated in the simulation world, the destination node may be determined as the node with minimum queue length. Such a pre-determined algorithm is used to determine the destination node as soon as the material has been handled by the source node in this model. Moreover, the volume of processed material should be equal to the volume of received material of each node at the end of each period if the perfect MHS is applied. This criterion may be used to detect simulation errors.

Two important features, miss cutoff and FIFO (First In First Out), are implemented in this simulation model. First feature is to test if the material will miss its cutoff time for the destination operation. Cutoff time is the latest time that one operation is allowed to run without delaying the downstream operation during a unit operation cycle (one day). Cutoff time can be mathematically derived by using Eq. (1) once the final dispatch and the throughput rate of each operation are known.

$$\begin{cases} P_{cur} \le P_{des} \le P_{cut} & : \text{Make the Cutoff Time} \\ P_{cut} < P_{des} & : \text{Miss the Cutoff Time} \end{cases} \quad (1)$$

The second feature is to analyze the FIFO queuing property for the input buffer of the destination node. According to the queuing theory, a node does not need such a device if its arrival materials are in a FIFO order. When the material is sent to its destination node, the destination operation period has been determined. Thus, a node requires an automatic intelligent retrieval type device if there is any material in its queue with a later destination period than the destination period of a new arrival material. If a material is handled and sent to the new destination node, $\mathbf{Node}_{dest}$, with a destination operation period, $\mathbf{P}_{dest}$, the type of input buffer device of $\mathbf{Node}_{dest}$ can be determined by using Eq. (2),

$$\begin{cases} \sum_{i=Pdes+1}^{Max\_P} \mathbf{Rec}_i = 0 & : \text{FIFO Type} \\ \sum_{i=Pdes+1}^{Max\_P} \mathbf{Rec}_i > 0 & : \text{Intelligent Retrieval Type} \end{cases} \quad (2)$$

where $\mathbf{Rec}_j$ is the received material volume of $\mathbf{Node}_{dest}$ at $\mathbf{P}_j$.

Another important quantitative design parameter is the minimum required buffer size of each node and can be determined if we have the time series of the peak number of the material volume in the buffer at all sampled periods. The volume of material in the buffer ($\mathbf{Buf}_{i,j}$) of $\mathbf{Node}_i$ at $\mathbf{P}_j$ is given in Eq. (3).

$$\mathbf{Buf}_{i,j} = (\mathbf{Rec}_{i,j} - \mathbf{Pro}_{i,j}) + \sum_{k=j+1}^{k=Max\_Period} \mathbf{Rec}_{i,k} \quad (3)$$

where $\mathbf{Rec}_{i,j}$ and $\mathbf{Pro}_{i,j}$ represent the received and processed material volumes for $\mathbf{Node}_i$ at $\mathbf{P}_j$, respectively.

The minimum buffer size of a particular $\mathbf{Node}_i$, $\mathbf{Min\_Buf}_i$, is the peak number of all the required buffer size of $\mathbf{Node}_i$ at all simulation sampled periods, as shown in Eq. (4)

$$\mathbf{Min\_Buf}_i = \text{Max.}(\mathbf{Buf}_{i,j}) \qquad j = 0,\ldots,\text{Max\_Period} \quad (4)$$

2.2 Links Modeling

The minimum required buffer sizes of nodes and throughput capacities of links which connect nodes are the two major design parameters of MHS. The throughput capacity in this simulation model is defined as the volume of handled (transported and delivered) material within a time unit (sampled period). $\mathbf{Link}_{i,j,k,l}$ is used to represent the characteristics of all links in this predictive simulation model, where i, j, k, and l represent sampled period, source node, destination node, and simulation statistic properties, respectively. Three entries, minimum buffer size of destination node, material volume received by destination node at destination period, and material volume sent from source node at current period, are included in the simulation statistic property. A delivered material is staged in the local input buffer of its destination node until its destination period.

The values of both $\mathbf{Link}_{i,j,k,1}$ and $\mathbf{Link}_{i,j,k,2}$ are updated (through a function call) when the material is handled and sent to the new destination node. The minimum required buffer size for $\mathbf{Node}_k$, $\mathbf{Min\_Buf}_k$, can be determined by Eq. (5) once $\mathbf{Link}_{i,j,k,1}$ is found.

$$\mathbf{Min\_Buf}_k = \mathbf{Link}_{i,j,k,0} = \sum_{p=i+1}^{p=Max\_P} \mathbf{Link}_{p,j,k,1} \quad (5)$$

This simulation model is a closed system, therefore no material will leave or disappear from the system. Eq. (6) shows a useful balance relationship to check for leaking error of simulation models.

$$\sum_{p=0}^{p=Max\_P} \mathbf{Link}_{p,j,k,1} = \sum_{p=0}^{p=Max\_P} \mathbf{Link}_{p,j,k,2} \quad (6)$$

The simulation time will be increased by the amount of "time factor" for every simulation cycle. The behavior of the simulation model is simulated based on the simulation time and the elapsed time. The larger the value of time factor is, the faster the simulation is. However, the simulation resolution will be decreased as the time factor is increased.

2.3 Simulation Modeling Summary

The capacity of each node at every sampled period is determined at the initialization stage using "Op.data" and "Param.data". Materials are then introduced into the simulation model by a system input device simulator. This simulator is modeled as a function and is called whenever the sampled period changes. Each material is simulated to be handled and assigned its coveted operation plan by applying Monte Carlo method and Queuing algorithm. The appropriate destination node of

[1] A perfect material handling system is an intelligent system which will automatically deliver every material in the system to a desired location at an appropriate time. Moreover, the capacity of the system is assumed infinite.

this handled material is also determined using the EAC. If no unexpected malfunction during the operation is assumed, the earliest node that will be able to handle certain materials in a period is the node with the smallest quantity in its operation buffer. This unique predictive property is suitable for the purpose of system design and analysis. The limitation of the predictive simulation is that one material can not go through two nodes within one sampled period. Thus, the sampled window effects the fidelity of the simulation result. The entire simulation flow chart is shown in Figure 1.

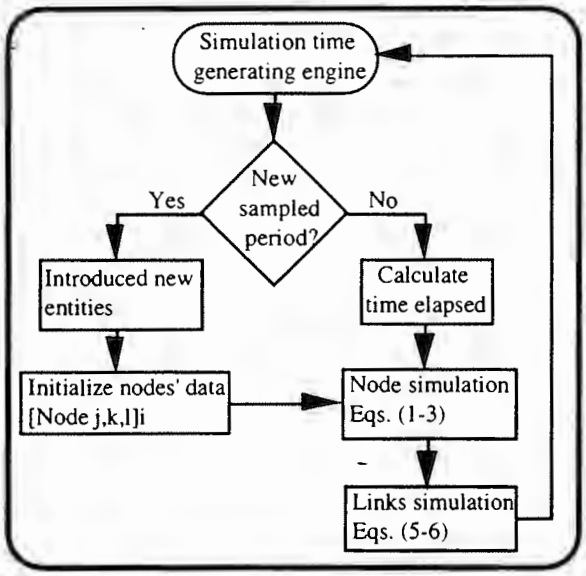


Figure 1. Predictive Simulation Algorithm Flow Chart

3. Example - USPS P&DC, San Diego, CA

The U. S. Postal Service San Diego mail processing and distribution center (P&DC) is a large scale facility in California. This facility is planned to extend and re-layout as an optimal P&DC by 1996. The predictive simulation is used to analyze the mail flows, staging requirements, and several other design parameters. This facility handles three to four million mail pieces every day, including letter and flat mail which have separate independent properties. In this paper, we use the letter mail analysis as an example. There are at least five major types of mail processing machines needed to be used to process the letter mail. They are Optical Character Readers (OCRs), Bar Code Sorters (BCSs), Letter Sorting Machines (LSMs), Distributed Bar Code Sorters (DBCSs), and Letter Mail Labeling Machine (LMLMs).

3.1 Standard Design Data Set Acquisition

The first step of this methodology is to acquire the standard data set that describes the characteristics of the entire system behavior. The standard data files of the facility provided by the San Diego field experts and are shown in Figure 2.

3.2 Simulation Modeling & Scenarios Setup

The simulation model of the San Diego P&DC was implemented in Language C and ran on Silicon Graphic Onyx Reality Engine-2 computer. Input data files were automatically read in while this software was running. A tray was used to represent the unit of a material (entity) and contained 300-450 of mail pieces. The simulation was setup to run a five day scenario. The output report files were generated at the end of the simulation. The analyzed data in the output report were extracted from the middle three days (second, third, and fourth day) for the accuracy purpose.

1.Volume.data-material volume of each input operation plan.
2.Profile.data-input profile distribution over 24-hour period.
3.Param.data-operation plan characteristic parameters, such as throughput, bin used number, bundleable, flush tray number, and order of operation plan.
4.Dens.data-bin density distribution of each operation plan.
5.Op.data-operation plan schedule over 24-hour period.
6.Cutoff.data-cutoff time of each operation.
7.Flush.data-machine flush schedule over 24-hour period.
8.Manu.data-units of manual sorting at each sampled period.
9.Center.data-served operation plans of center storage devices.
10.Node.data-general information of each node (AMPM).

Figure 2. Standard Data Set of San Diego, CA, P&DC.

3.3 Simulation Results

In this study, simulation searched all possible paths that mail would travel within the P&DC. The data describes each path which includes not only its source and destination, but also the throughput volume at all sampled periods.

3.3.1 Minimum Required Buffer Size & Types

One of the main goal of this simulation analysis is to find the minimum required buffer size of the MHS. Table 1 shows the detail data of the minimum required buffer size of every OCR extracted from the output report file of the simulation. Two different analysis methods, Node analysis - Eq.(4) and Link analysis - Eq.(5), are both applied in the simulation model. The result of these two methods are similar to each other. The larger buffer size of the results derived from these two methods should be used as a design base parameter for the safety purpose. Safety factors should be also applied to these parameters when the buffer device is designed and implemented.

Machine (Node)	Min. Buf (Node Analysis)	Min. Buf (Link Analysis)
OCR-0	154	153
OCR-1	150	144
OCR-2	129	130
OCR-3	102	84
OCR-4	102	82
OCR-5	121	100
OCR-6	180	166
OCR-7	200	180
OCR-8	150	132

Table 1. Minimum Required Buffer Size of OCRs.

Because of interchangeable properties, it is logical to arrange the same type of machines into like-groups, similar to a farm layout. All of the paths that previously went from machine to machine are now combined with similar paths to form new paths that carry trays of mail from group to group. Figure 3 shows the 5 day buffer size time series of OCR-0, OCR group, and P&DC. Another useful information for designing the staging equipment is the type of retrieval device, FIFO or intelligent retrieval (I.R.). Table 2 shows the required volume of buffer size and staging type of all groups in the P&DC. If the staging design is a centralized storage device system, the

minimum buffer size of the center storage device is 5950 trays applying the same method. OCR group is the only group that does not require an intelligent retrieval staging device. This is because the OCR group is the first mail processing station that all types of incoming mail may be processed in a random order.

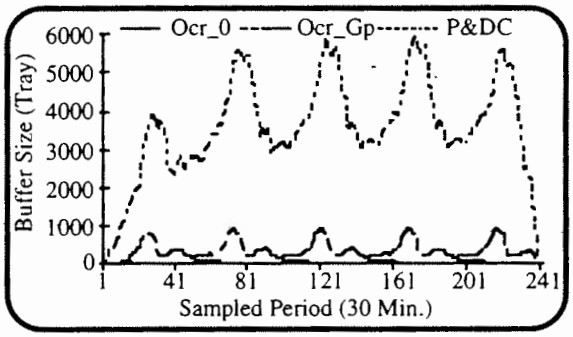


Figure 3. Five Day Buffer Size Time Series of OCR_0, OCR Group, and P&DC.

Ocr	Bcs	Dbcs	Lsm	Lmlm	Manu	P&DC
938	925	2997	919	260	747	5950
FIFO	I.R.	I.R.	I.R.	I.R.	I.R.	N/A

Table 2. Min. Required Buffer Sizes of All Nodes' Groups.

3.3.2 Maximum Throughput Capacity Requirement

Another important design parameter of the MHS is the required throughput capacity of each link that connects nodes. The predictive simulation provides the complete throughput time series of every link. Figure 4 illustrates the throughput time series for BCS group. The minimum required throughput capacity is the peak value of the time series. Table 3 shows the minimum required throughput capacity matrix for all the machine groups in the P&DC.

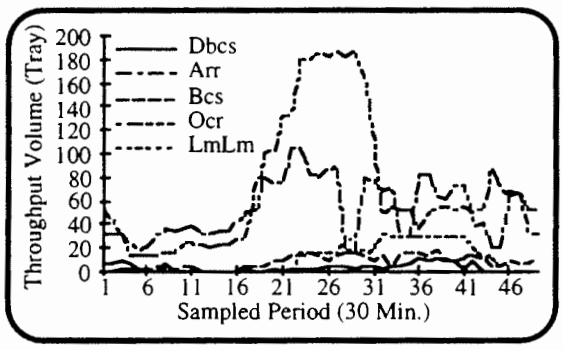


Figure 4. Throughput Time Series for Destination: BCS Group .

3.3.3 Operation Characteristics

The steady state status of volume of miss-cutoff tray and the volume of turn around mail are the most important aspects in the P&DC operation. The phenomenon of increasing volumes of miss-cutoff trays and turn around mail over time is considered as a result of a poor design because it means the machines are not capable of handling all the input mail. The steady state status of both miss-cutoff mail and turn around mail volumes are observed from the result of simulation.

From \ To	Arr	Ocr	Bcs	Lsm	Mamu	Dbcs	Lmlm	Disp
Arr	0	320	105	59	57	78	0	0
Ocr	0	0	187	28	0	156	0	227
Bcs	0	35	18	37	0	254	19	328
Lsm	0	0	0	245	40	0	0	1651
Manu	0	0	0	0	34	0	0	158
Dbcs	0	34	11	40	0	266	0	532
Lmlm	0	0	31	0	0	0	0	0
Disp	0	0	0	0	0	0	0	0

Table 3. Throughput Requirements for All Groups in the P&DC.

4. Conclusion

The ultimate purpose of this study is to provide a generic method to design an optimal automated facility. The material handling system is usually the most important component of a modern system automation. This paper proposes a quantitative analysis method, predictive simulation, to synthesis the required design parameters for the MHS. These parameters include the minimum required throughput capacity of every path, minimum required buffer size of every staging device, the types of staging buffer devices, and analyzed data of the information of every automatic machine and every operation.

The facility layouts found by applying this method are the best layout that can be derived from mathematical model. The system behavior should be simulated in its entirety before any components are installed on the floor of the facilities. A CASE tool called CSAT provides all these perfect features for integrating the entire automation facility and supporting the life cycle system design. Once all these stages are accomplished to the satisfaction of the end users of the system, the implementation of the final design can be initiated with the confidence that the system will be reliable and efficient.

Acknowledgments

This research was supported by the USPS Headquarters under Task Order #104230-88-D-2546. The authors wish to thank Dr. Anthony Barbera, Mr. Clyde Findley, Mr. Phil Feldman, and Mr. Keith Breadford for their advice and discussion.

References

[1] Lu, J. C., Tsai, K. H., Yang J. C. S., and Wang M. Y. "Real-Time Operation Plan Optimization Using Accelerated Simulation Modeling", International Journal of Modeling & Simulation, Accepted for publication.

[2] Findley, C., Feldman, P., and Lu, J. C., "An Analysis of Material Handling and Mail Processing Equipment Configuration in USPS Processing & Distribution Centers", Final Report of USPS Automation Research Project, Washington DC, May, 1993.

[3] Yang, J., and Lu, J. C., "Optimized Design For Automation Systems Using 4-D Simulation", Proceeding of International Workshop On Functional Modeling Of Complex Technical Systems, Ispra, Italy, May 13-14, 1993.

[4] Yang, J., and Lu, J. C., "A Life Cycle Automation Toolset Using The Real-Time Control System", Proceeding of US/ROC Workshop On Automation, Taipei, Taiwan, ROC, July 1-4, 1993.

[5] Yang, J., and Lu, J. C., "Developing A Life Cycle Toolset For The Modern Automation System", Proceeding of Sino-American Technology & Engineering Conference, China, October 11-15, 1993.

Dipl.-Math. Veit Popp, ExperTeam SimTec GmbH, Dortmund

Simulation for Production Scheduling:
Integration of Simulation Techniques into Planning and Scheduling

1. Why simulation...

The problem of production planning and scheduling is characterized by a simultaneous scheduling of all resources and materials needed. Traditional MRP systems use scheduling algorithms that do not represent the finite capacity of a manufacturing system precise enough and do not provide a means for analyzing a manufacturing system's ability and agility to satisfy diverse customer demands through the use of different planning strategies and scheduling technologies.

This is the point where event based simulation comes in and provides techniques for a total coordination of the elements of the production process, giving a clear understanding of the details of all processes that influence the manufacturing performance. Simulation predicts the impact of each order on manufacturing performance and the production process. As all characteristics of the production system and all capacity constraints are represented in the simulation model (strategic, operating and procedural aspects), a better understanding of the dynamics of the production system is achieved. Comparing results of different simulation alternatives provides a better communication based on facts for planners and schedulers.

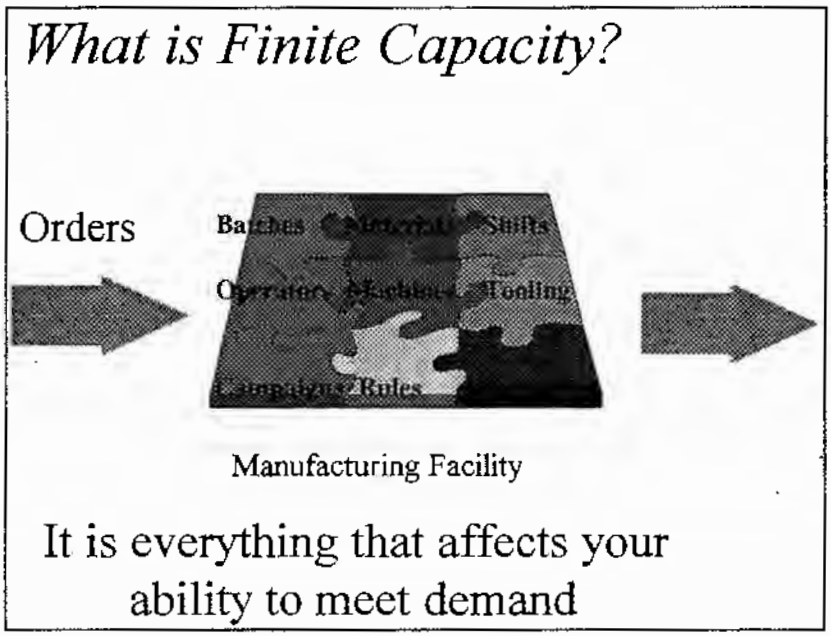
The problem of the representation of the complexity of the production process in a model - the "factory in the computer" - is simplified if the organizational and structural models developed in a capacity design system can be transfered and reused for finite capacity planning and scheduling for production control. Especially simulation models for evaluating capacity design questions and production strategies can help to understand the dynamics of the production process and provide a template for implementing a scheduling model.

The benefit of event based simulation for scheduling is well acknowledged, therefore one of the main demands on MRP systems in the future is providing "real" simulation techniques.

2. Modeling

A simulation of a manufacturing system provides a realistic portrayal of its detailed operations from which its performance can be accurately estimated. Because all aspects of the manufacturing system, including management's operating philosophies and procedures, are contained in the computer model, a common understanding of the planning and scheduling functions is established. This allows for comparisons of alternatives and facilitates communication to managers and executives of any problem areas and improvement possibilities. Simulation implements the operation strategy and creates feasible, coordinated and intelligent schedules.

Because manufacturing systems are complex and large, simulation systems have been specifically developed for them. A good manufacturing simulation system provides the capabilities as follows: allows diverse scheduling-development philosophies, demand-driven simulation using actual orders and build-to-plan forecasts, bill of materials to any level of detail, multiple resource requirements and constraints for processing steps, detailed process planning procedures, menu-driven interactive query functions to support reporting and problem analysis using manufacturing terminology, future projections of order status and displays of operation contention in specific time periods, advanced built-in graphics with automatic chart generation, RDBMS with complete data structure specificity, manufacturing specific functions, problem configuration and data collection specification, capabilities for multiple performance measurement and estimation.

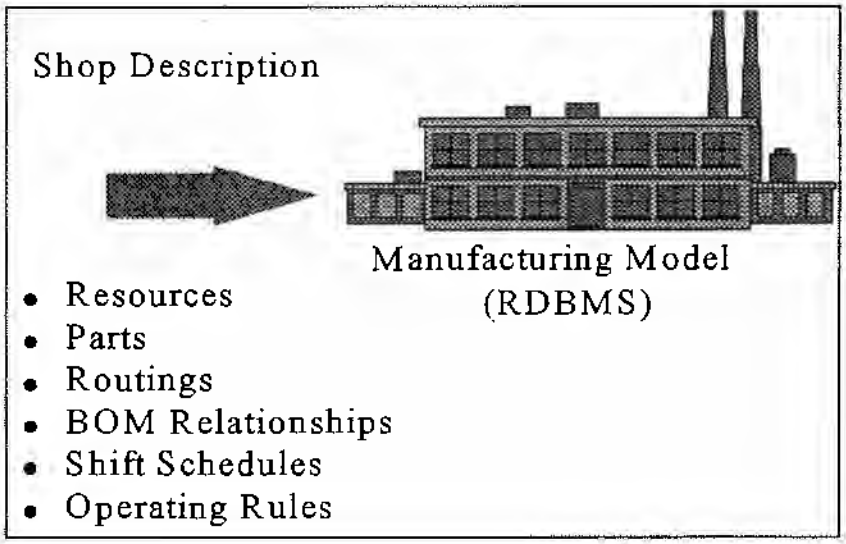


Traditional simulation systems need great expenditure of modeling work and often are inflexible to support permanent model modifications. Because of the amout of data that has to be processed (orders, process plans, jobsteps) the computing performance does not allow to use a software implementation based on a simulation language to be interpreted at runtime. A new approach supports both requirements: easy but accurate modeling and runtime performance. The event based simulation - also called time dynamic simulation - is used to combine the production data model available in MRP systems with the well known rule-based queuing algorithm.

One of the main advantages of the simulation system *FACTOR Production Manager* is that is makes use of the all the data relevant to the manufacturing process available in MRP and shop flor data collection systems.

The underlying model consists mainly of data already available from MRP systems: orders, bill of materials, process plans, resource data, shift calendars, maintenance intervals, personal with qualification for specific jobs etc. Building the model is reading that data through an interface from the MRP and shop flor data collection system into the simulation system. The interface has to be "intelligent" to provide some kind of modeling tasks: bill of materials and process plans are connected by assigning material needed to specific job steps, fixed set-up times have to be converted to part-depending set-ups and rules when to apply, fixed waiting times are eliminated (perhaps considering transportation time) as the time a load is waiting in a queue for processing is evaluated by the simulation depending on the dynamic situation.

In the simulation model generated, all aspects of the specifics of the shop decription is taken into account: operating strategy, multiple finite capacity constraints of job steps and technological sequencing rules.



3. Simulation

Simulation can now be performed by reading the MRP order data and the actual status of the manufacturing system (by a data collection system) on a day-to-day basis. The simulation takes into consideration all technological, strategic and manufacturing rules that were implemented at every resource. Sequencing is done by selection rules at resources that can be controlled by global rules (Earliest Due Date, Least Dynamic Slack, etc.) or resource specific rules (Minimum Set-Up, etc.). By using the multiple constraints defined in the job steps the simulator determines the best fit of strategies selected based on the actual jobs to be performed.

Simulation can be repeated with parameter adjustment as often as desired to satisfy the actual goals of manufacturing (meet due days, minimize set-up, decrease inventory level etc.). Simulation is rerun if new order data is available, a severe breakdown conflicts with the actual schedule or the actual status varies considerably from the schedule. Especially for the last item a degree of deviation from the schedule has to be defined and a threshold that indicates the necessity for reruning the simulation based on the actual status of production. This is caused by accumulation of small breakdowns or a conflict between strategic goals and their feasibility.

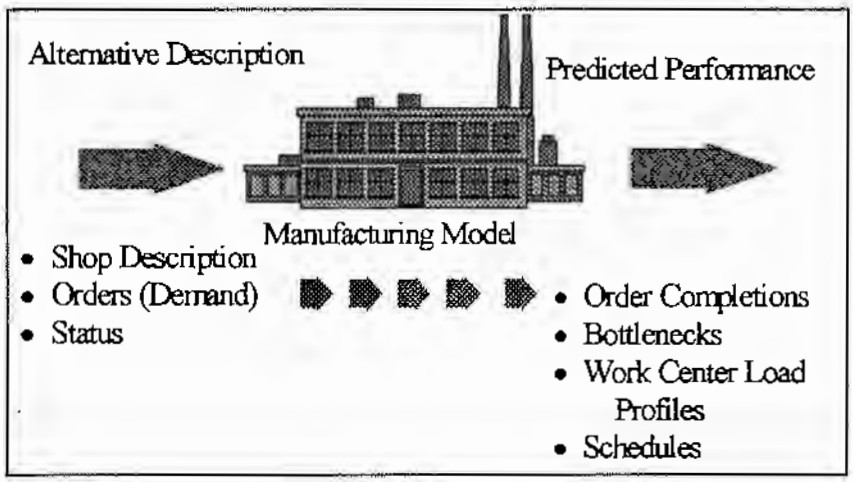
4. Results

When a relational database system is part of the scheduling system, it is easy to work with results of the simulation to obtain information on why the manufacturing system representation behaved the way it did, that is, produced a specific plan. Global and specific information is available: "Are there orders late, which orders are late, which job steps waited for the resource that caused the lateness, which are the bottleneck resources and when, did the job step wait for a resource or was material not available" etc.

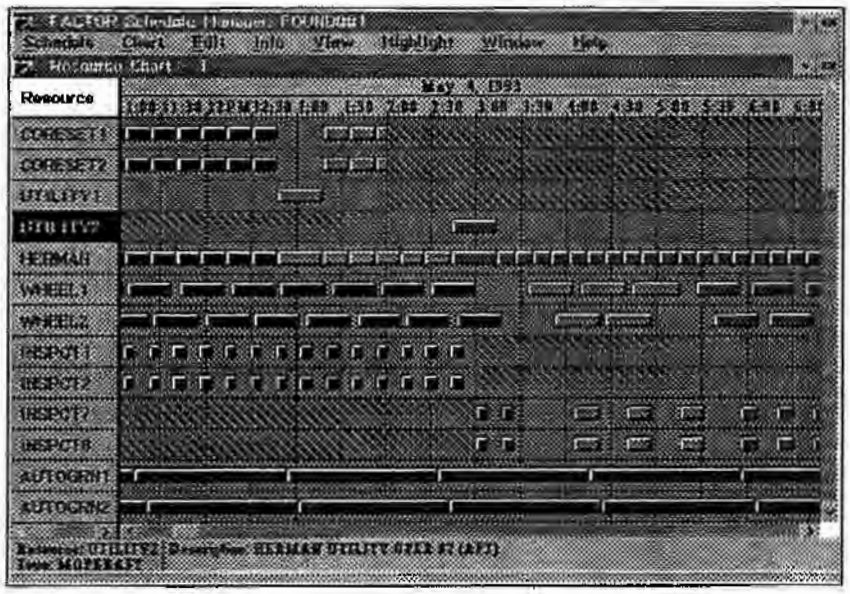
Also very detailed information is available: for each job step the simulation determines

- when are resources and materials allocated by requests of specific job steps
- when starts the operation, e. g. are all resources and materials needed available
- when is the operation finished and frees all resources not needed any more.

Simulation provides a feasible and intelligent schedule not only for resource allocation but also for material requirements, thus supporting JIT concepts.



Alternatives allow for what-if-scenarios to explore more advanced operating strategies. The final schedule then is used for execution, detailed material plans and feasible promise dates.



5. Summary

The advantages of event-based simulation for planning and scheduling are:

- simulation allows realistic and technological scheduling and the ability to perform various what-if-scenarios to develop the best strategies that meet the company's goals
- no complex modeling is needed as the model is generated mainly in the data interface to the MRP system and the sequencing rules at resources
- simulation always produces a feasible and intelligent schedule
- produces all necessary data to get full transparency of the manufacturing process and to feed back results to other systems involved in the manufacturing and decision making process
- considers the actual status of the production system when simulation is performed and therefore provides a full control loop for the manufacturing process
- schedulers are free to solve real problems, not ones we can predict ahead of time.

Top Down Design with VHDL-A

Richard Trihy and Kenneth Kundert
Cadence Design Systems
trihy@cadence.com

Abstract

Traditionally circuit simulation has seen significant application in the verification of integrated circuits and systems. Its application to system level design has been limited by the need to express circuit and multi-disciplinary model behavior in terms of a limited number of built-in circuit primitives. However the new Analog Hardware Description Languages (AHDLs) provide the needed flexibility and functionality to describe behavior at the architectural or system level.
This paper discusses the benefits of AHDLs in the design of circuits and systems. By bridging the gap between higher level behavioral or system level descriptions and circuit level descriptions an AHDL, such as VHDL-A brings structure and continuity to the design process. Language constructs that "shield" the user from the underlying simulator complexities allow designers to efficiently utilize the behavioral modeling capability. The top down design of a 9-bit A/D converter using Cadence's prototype implementation of VHDL-A is presented as an example[1].

1 Introduction

Simulation of analog circuits and systems as performed by traditional simulation strategies is time-consuming and cpu-intensive. The system to be simulated must be described in terms of a limited set of built-in primitives, consisting of resistors, capacitors, transistors, controlled sources etc. While the list of primitives may include higher level behavioral macromodels, the design or architecture must still be expressed in terms of this limited set of building blocks. This requirement acts to restrict the designer when generating a higher level description of a system's behavior.

The cost of circuit simulation is primarily affected by the size or number of nodes in the circuit as well as the complexity of the component models that must be evaluated at each iteration. Hence in order to speed up the simulations the size of the circuit should be reduced and its models simplified.

AHDLs allow the user to describe the essential behavior of a system more easily than is possible with traditional circuit simulators. By reducing the circuit and model detail, faster simulations result. The ease with which system level behavior can be expressed and then simulated opens up a number of useful applications that were previously cumbersome or even impossible.

This paper addresses the capabilities of SpectreHDL[3], a mixed abstraction tool that combines different AHDLs, e.g. VHDL-A, Verilog-A, and circuit level descriptions. There are two key aspects to SpectreHDL's languages. SpectreHDL employs an equation formulation methodology that treats across and through (voltage and current) sources symmetrically and independently of the simulator's underlying matrix formulation. This makes the language easier to use and understand[2].

The SpectreHDL langauges support a set of built-in functions that can be used to model a variety of different behaviors. A set of functions for shaping waveforms and controlling the underlying simulator engine are provided. This paper illustrates by example the application of SpectreHDL to the top-down design of circuits and systems.

2 Top Down Design

Traditionally post layout verification of integrated circuits has been the dominant need of high performance analog simulation. Analog circuit simulation does find application late in the design process but only after most of the detailed design decisions have been made. Other more behavioral approaches are often taken at the architectural level of the design. These tools do not extend to circuit level simulation.

The evolution of flexible AHDLs, such as VHDL-A, that permit the merging of accurate circuit level analog simulation and mixed signal simulation with higher level behavioral descriptions has introduced new applications for these tools in the design of analog and mixed signal circuits and systems.

This uniformity in the design tools between system and circuit level descriptions opens up new possibilities for the design process.

- The system under design can be described and simulated at a high level of abstraction early in the design process. This aids design experimentation and model reuse. The essential system behavior and its dependence on component nonidealities can be identified with the aid of simulations that would be prohibitive at the circuit level. This is illustrated in the case of a 9-bit A/D converter below.

- These higher level descriptions serve as a communication tool between the circuit and system level designers. They document the interface between system modules. Furthermore the AHDL descriptions can be used by the test engineers to debug the early versions of their testbenches.
- Functional level models for the system elements can be used to verify the connectivity of the circuit. Errors in circuit connectivity is a common and potentially very expensive mistake. It can be avoided with inexpensive simulations that employ AHDL models that characterize the interface of the system elements.
- AHDL can be used to construct measurement blocks or testbenches for analyzing components or circuit behavior. For example a standard testbench for opamps might perform a number of analyses to measure offset, slew rate etc. of different opamp circuits. For top-down system design the measurement blocks perform measurements of interest and they can be reused across abstraction levels. As the circuit details of the system are refined and expanded the testbench serves as a check on the behavior. Figure 1 shows the block diagram for an A/D converter and it includes a testbench to measure INL and distortion. It will be demonstrated in the next section.
- By mixing levels of abstraction it is possible to speed up the resulting simulations. Key pieces of the design can be described at the circuit level, while the remainder is modeled at the behavioral level.
- Finally AHDL allows for analyzing complex systems that employ multi-disciplinary models. Models of different disciplines can be combined in the system description to investigate the overall system (and not just electrical) behavior.

3 Design Example: 9 bit A/D

This section describes the top-down design of a 9 bit A/D converter. The example is designed to illustrate the desirability of developing testbenches early in the design process as well as to give a flavor of the types of analyses that can be inexpensively performed with behavioral models of the system.
Five levels of design are implemented for this A/D component. At its most simple level it consists of a single AHDL module description and the most complex version mixes behavioral and circuit level descriptions of the entire system.
The converter is a parallel pipeline implementation. It consists of four parallel paths that are time multiplexed to increase throughput. Each of these paths consists of a four stage 9 bit pipeline A/D converter with digital error correction. A block diagram of the circuit is shown in figure 1.

3.1 Simple Model

Using the most simple system model, suitable measurement blocks can be developed and tested. These measurement blocks are reused repeatedly as the design advances. They serve to ensure that the design is correct, and they identify the effects of component nonidealities.
The measurement blocks that are implemented are an ideal 9 bit D/A converter which allows the reconstructed output analog waveform to be compared with the input waveform and an integral nonlinearity (INL) measurement block. The INL block calculates the error between the input analog signal and the reconstructed output waveform. This integral nonlinearity error is expressed in terms of least significant bits (lsbs).
Furthermore we can perform fourier analysis on the reconstructed output waveform to investigate the distortion effects of various nonidealities in the system.

3.2 Analysis of pipeline converter

The first implementation of the pipeline converter consists of three 3-bit stages, which are combined to produce a single 9-bit code. In addition an implementation of the converter that uses an additional stage and employs digital error correction[1] to remove error due to component nonidealities is created. Diagrams of the 4-stage pipeline converter and the expanded 3-bit stage are given in figure 2. AHDL descriptions are written for the 3-bit stage, as well as for the input sample and hold and digital combinational logic.

To test this error correction feature, both the correcting and non-correcting A/D's were simulated with A/D nonlinearity error infused into the models. A VHDL-A module for this nonlinear 3-bit A/D is shown in template 1. The nonlinearity is modeled by introducing random offsets at the A/D transition points.

Figure 3 shows a portion of the output waveforms from the INL measurement block (expressed in lsbs) for 3 different transient simulations. The first plot shows the quantization error for a pipeline converter with no nonidealities. The second plot is for the case of a nonlinear A/D converter. The third plot is the result of simulating the circuit with nonlinearity and additionally digital error correction. This shows that the digital error correction does indeed remove some nonlinearity in the system's components. It would be prohibitively expensive to perform these simulations at the circuit level.

3.3 Parallel pipelines

The 4 bit stages can now be combined in parallel with an analog time domain multiplexer to complete the overall system. The interesting nonidealities that can be examined here are the effects of path mismatch on the harmonic content of the output waveform.

The effects of gain and offset mismatch were investigated. For a sampling frequency (Fs) of 10MHz and an input sine wave with frequency of 625KHz (Fs/16), the harmonics produced from gain mismatch are shown in table 1 below. This data was obtained from a transient simulation followed by a fourier analysis of the reconstructed output waveform.
It can be seen that the dominant harmonics occur at 3Fs/16, 5Fs/16, 7Fs/16, 9Fs/16, which is the behavior predicted in[1].

Next the circuit was simulated with a D.C. input, and gain mismatch was replaced by offset mismatch. In this case the dominant harmonics were seen to occur at frequencies of Fs/4, Fs/2 and 3Fs/4, i.e at multiples of the multiplexer sampling frequency, as expected.

3.4 Mixed abstraction representation

The final representation of the circuit replaces the AHDL description of the sample and hold with a detailed circuit (transistor) level representation, see figure 4 The transient simulation results for a 200KHz sine wave and an A/D clock frequency of 10Mhz are

shown in figure 5. The first plot shows the input sine wave and the reconstructed output waveform, which is generated from the ideal D/A probe measure block. The second plot shows the reconstructed waveform for one of the A/D channels and the last plot shows the reconstructed waveforms for all four channels. The true output of the A/D is a composite of these waveforms.

Table 2 compares the simulation times (on a SPARCstation 5) of the mixed abstraction circuit with a purely AHDL implementation. There is an order of magnitude speed difference.

4 Conclusion

This paper has illustrated the benefits of an analog hardware description language for top down design of integrated circuits and systems. The ease with which circuits and systems can be described and simulated with AHDL opens new possibilities for design methodologies. Investigation of design trade-offs can be performed efficiently at the architectural level and measurement modules can be written to quickly analyze the circuit or system behavior across different levels of abstraction. In addition standardization will increase model availability, interchange and reuse.

References

[1] Cormac S. G. Conroy, David W. Cline and Paul R. Gray. "An 8-b 85-MS/s Parallel Pipeline A/D Converter in 1um CMOS", IEEE journal of Solid State Circuits, vol 28, No. 4, April 1993.

[2] Daniel FitzPatrick and Kenneth Kundert. "Simplified approach to formulating Analog Behavioral Models", ICEHDL Las Vegas, Nevada, January 1995.

[3] SpectreHDL Reference Manual. Cadence Design Systems.

Table 1: Harmonics for Gain Mismatch

Harmonic	Relative Magnitude
1	**0 dB**
2	-280.573 dB
3	**-32.0978 dB**
4	-283.165 dB
5	**-33.4639 dB**
6	-295.362 dB
7	**-35.2253 dB**
8	-289.381 dB
9	**-37.3804 dB**

Table 2: Comparison of Simulation times

Abstraction	Simulation Time
Mixed Circuit and AHDL	36m 15s
AHDL only	3m 2s
Simple AHDL A/D model	1m 24s

```
NATURE electrical IS
  ACROSS V;
  THROUGH I;
END electrical;

ENTITY adc3 is
GENERIC(mismatch : REAL := 0;
        risetime : REAL := 5e-9;
        vref : REAL := 1.0;
        falltime : REAL := 5e-9;
        hi  : REAL := 5.0);
PORT(vin, d0, d1, d2 : electrical);
END adc3;

ARCHITECTURE basic_adc3 OF adc3 IS
  VARIABLE s2 : REAL := vref/2*(1.0 + mismatch*(random(2) - 0.5));
  VARIABLE s1 : REAL := vref/4*(1.0 + mismatch*(random(1) - 0.5));
  VARIABLE s0 : REAL := vref/8*(1.0 + mismatch*(random(0) - 0.5));
  VARIABLE out0, out1, out2, x : REAL;

BEGIN
RELATION BEGIN
    x := vin.V;
    out0 := 0;
    out1 := 0;
    out2 := 0;

    IF x > s2 THEN
       out2 := hi;
       x := x - s2;
    END if;
    IF x > s1 THEN
       out1 := hi;
       x := x - s1;
    END if;
    IF x > s0 THEN
       out0 := hi;
       x := x - s0;
    END if;

    d2.V <= transition( out2, 0, risetime, falltime );
    d1.V <= transition( out1, 0, risetime, falltime );
    d0.V <= transition( out0, 0, risetime, falltime );

END RELATION;
END basic_adc3;
```

Template 1: 3 bit A/D model with nonlinearity

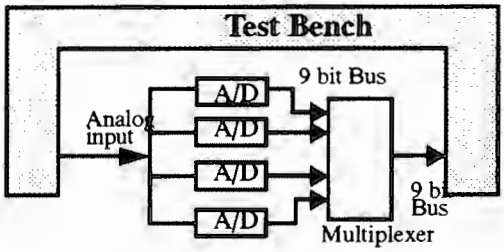


Figure 1: Parallel Pipeline A/D and Testbench

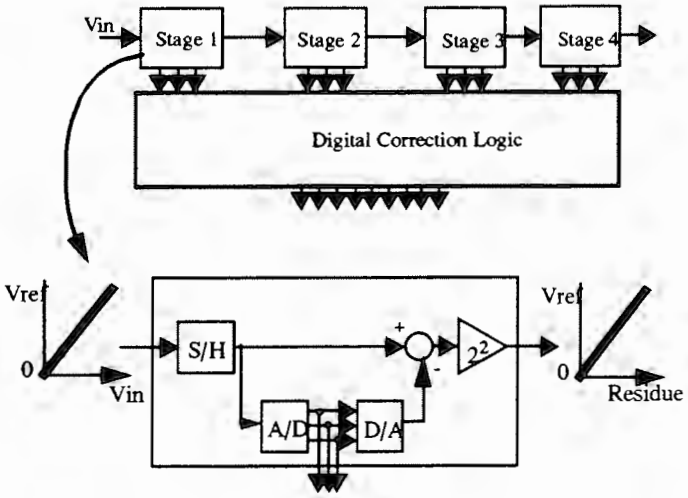


Figure 2: Pipeline A/D and Digital Correction Logic

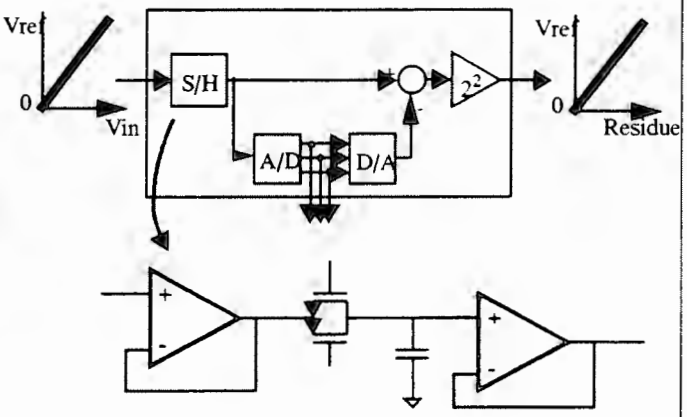


Figure 4: Mix of transistor level schematic and AHDL

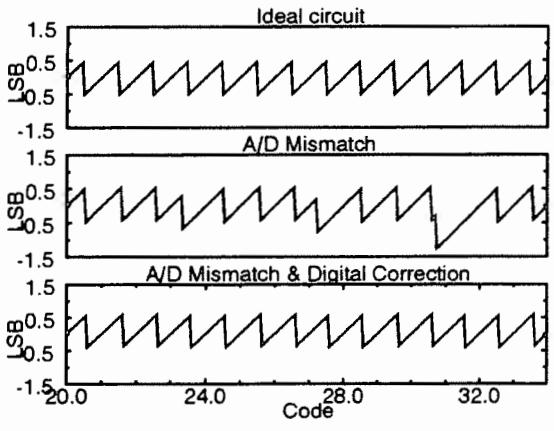


Figure 3: Simulation results with and without correction logic

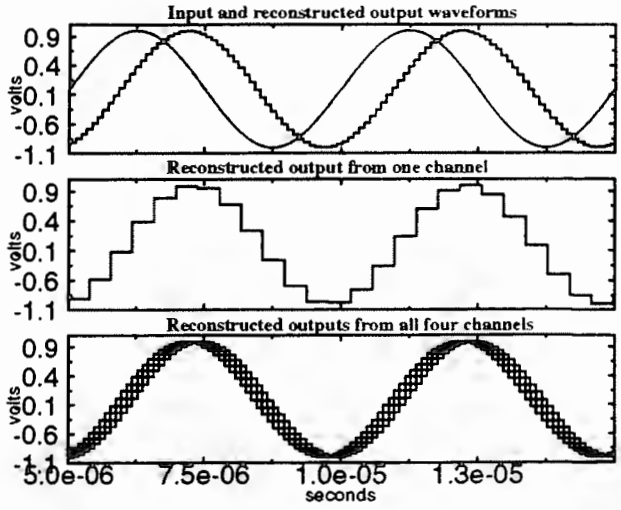


Figure 5: Transient Simulation results for mixed abstraction circuit model

VASIMS: A software package for validating and simulating FMSs modeled by Petri nets

C. AMER-YAHIA[1,2] Y. HADDAB[1]

[1] Université de Tizi-Ouzou, Institut d'Electronique, LCSP, B.P. 17 R.P. 15000 Tizi-Ouzou, Algeria
Fax number: 213 3 21 29 68

H. ACHOUR[1] N. ZERHOUNI[2]

[2] Ecole Nationale d'Ingénieurs de Belfort, LMP, 8, Bd Anatole France, 90016 Belfort Cedex, France
Fax number: 33 84 58 23 42

Abstract: VASIMS, a software package to assist the drawing , analysis, and simulation of flexible manufacturing systems is presented. The Petri net description, performed by the software user, follows a textual or a graphical approach. The system validation is based on the interactive use of the three well-known methods of analysis which allow behavioral and structural properties of Petri nets to be verified. A new algorithm which calculate all minimal invariants is used for the first time. All the programs constituting this computer-aided tool use the BORLAND C++ language. This object-oriented language makes the tool flexible and easy to use.

Key-Words: Petri net, FMS , Validation, Simulation, Object-oriented language.

1. Introduction:

Once a Petri net (PN) model of a physical system is constructed, a qualitative analysis may be performed. The analysis aims to investigate the behavior of the model and provides an assurance that the whole process is well defined. There are three main methods of analysis which allow behavioral and structural properties to be verified, for example liveness, boundedness, deadlock, and so on [1],[2].

VASIMS, a software package to assist the drawing, analysis, and simulation of flexible manufacturing systems (FMSs) is presented. The PN description, performed by the software user, follows a textual or a graphical approach. The system analysis is based on the interactive use of the three methods above-mentioned. The available processing allows the analysis of autonomous PNs (ordinary and generalized) as well as non-autonomous PNs (timed and synchronized).

All the programs constituting this computer-aided tool use the BORLAND C++ language. This object-oriented language makes the tool flexible and easy to use.

VASIMS is organized as it is shown in figure 1. It is constituted of two separate parts : the editor part and the processing part. A file serves as an interface between the two parts.

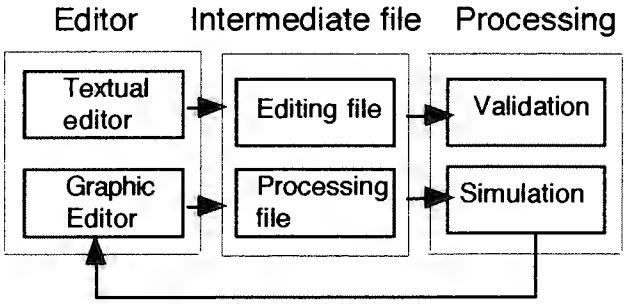


Fig.1 VASIMS'software structure

The resulting properties, such as liveness, boundedness,deadlock,safeness, reachability, persistence, and so on, when the different methods of analysis are applied to a manufacturing system model lead to the validation or the invalidation of the system. In the case where the manufacturing system model does not present the required properties, the proposed computer-aided tool makes it possible to find out and correct the design mistakes.

2. The PN editor

The implementation of a PN model on a computer uses two different approaches. It is possible to synthesize the PN model by means of a textual editor which allows to describe the PN structure and initialize the system model. It is, as well, possible to describe the manufacturing system, one hopes to design and control, by a graphic editor. However, because of the graphic nature of PNs, a particular care has been taken in the design of the graphic editor.

The net is drawn on a squared area which appears on the computer screen.

This area may occupy all the screen. When the menu or a sub-menu is needed, it is called and it appears on the screen. Once the needed function has been executed, the menu desappears, leaving the whole screen to the drawing operation.

A mouse may be used. It provides a flexibility in the access to PN components or menus.

In addition to the ordinary operations such as components displaying, deleting, names changing, and so on, the editor offers certain possibilities such as the zoom operation, the arcs drawing operation, etc... .

For a better PN drawing, an arcs drawing method is proposed. The method decides of the structure of the arc with regard to the relative positions of the place and the transition to join. The arc drawing is a part of an ellipse with the point $O(x_1, y_2)$ as a center (see figure 2).

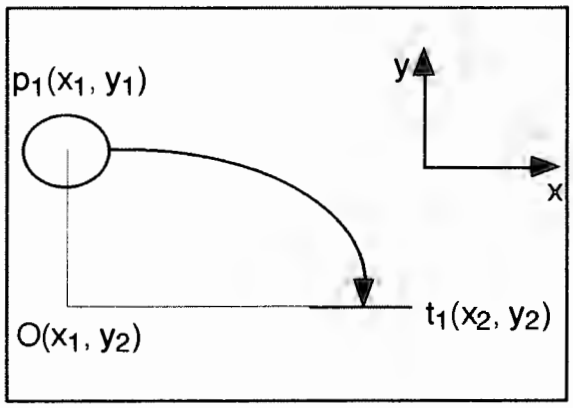


Fig. 2 Arcs drawing strategy

A PN modeling an FMS is constituted of a number of components that may represent a machine, a stock, or any entity of the system. In order to avoid confusions, references and names, displayed on the screen, are given to every place and every transition of the net. This way of doing seems to be very useful, particularly when a PN simulation is concerned.

Building large PNs is possible with VASIMS. Indeed, the authorized building space is not limited to the computer screen, but may be extended to a virtual screen whose dimensions, fixed by the user, are much larger than the real screen dimensions, and inside where the computer screen may move.

Because an object oriented language (the BORLAND C++) is used for writing VASIMS, the editor, from the conception point of view, is very different from editors that use ordinary languages [6],[7]. VASIMS proposes the following conception structure:

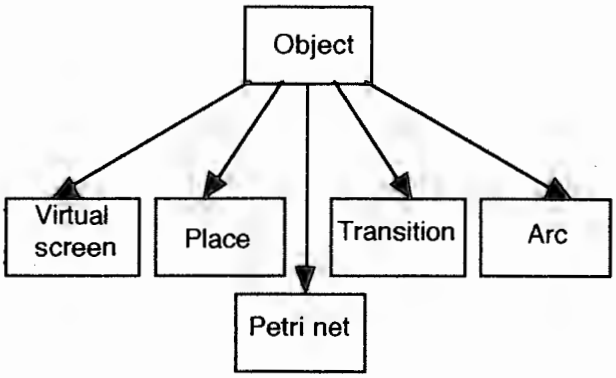


Fig.3 VASIMS'conception structure

The above structure is constituted of six classes. The object class contains all the data and methods that are common to the derived classes. All the classes inherit properties of the object class. The PN class contains all the pointers that are necessary to the manipulation of the net components. The virtual screen is a class which allows to manipulate the virtual screen with regard to the real computer screen.

The object-oriented language enables every place, every transition, and every arc of the net to be represented by an object. VASIMS introduces the idea of a PN by defining three sequences of objects, including all components of the net.

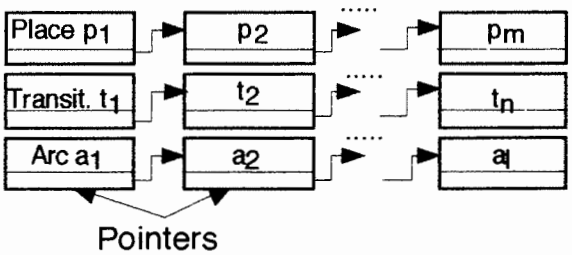


Fig.4Sequences of places, transitions, arcs

Each component of a sequence pointes on the following component by means of a pointer. Thus, the substraction or addition of components are widely facilitated. The object "PN" contains all the pointers that are necessary to the manipulation of the sequences. This way of doing presents a real advantage: the number of places,

transitions, and arcs is not limited. Thus, the size of a PN is only limited by the memory size of the user computer.

3. The intermediate file:

The PN drawing operation has led to the creation of two files: the editing file and the processing file.

The editing file receives all the data concerning the editing operation, such as the position of the different PN components on the virtual screen, the name of every place and every transition of the net, and all the PN drawing informations. It allows a repetitive call of any memorized PN in order to display it on the screen and, eventually, make some modifications. This file is used when a simulation of a PN is concerned.

The processing file is built with informations taken from the editing file. These informations are set in a way to be directly used by the processing algorithms, as it is shown in figure 5.

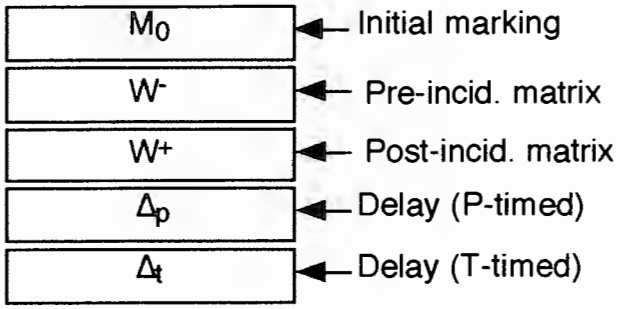


Fig.5 Processing file

4. Validation

This operation of analysis verifies the behavioral and structural properties of PNs by means of a number of tasks that are performed on the data contained in the processing file. the results of these processings validate or not the PN model.

VASIMS uses several methods for analysing PNs. Firstly, it makes it possible to draw the reachability tree (if the net is bounded) or the coverability tree (when the net is unbounded) of a PN according to its initial marking. This operation is done on the file in order to enable the analysis of large systems. Secondly, it allows to obtain all marking and firing invariants (P-invariants and T-invariants) which provide powerful tools for studying structural properties of a given PN independently of its initial marking. A new and powerful algorithm is used, in this case, for the first time [3]. Thirdly, the study of structural properties, as mentioned above, is done by means of a number of algorithms implemented in the proposed computer-aided tool. Finally, in order to facilitate the analysis of very large systems, VASIMS enables the reduction of the system model to a simpler one, while preserving the system properties to be analysed.

5. Simulation:

The computer-aided tool presented may perform, as well, the simulation of flexible manufacturing systems by enabling and firing transitions. The firing of transitions causes an evolution of the marking which corresponds to an evolution of the state of the system. Questions like the type and the number of machines one may buy, stocks capacities and system architecture one may use, and so on, are accurately answered by the simulation operation.

VASIMS allows the simulation of both synchronized and timed PNs.

In a synchronized PN [4], an external event is associated with every transition of the net. The firing of a transition will occur under the following conditions:

i) the transition is enabled,

ii) the associated event occurs.

The evolution of a synchronized system is described by the PN fundamental equation in which a new matrix is inserted, namely the events matrix E. The fundamental equation of PNs:

$$M_i = M_0 + W \cdot S$$

becomes:

$$M_i = M_0 + W \cdot E \cdot S$$

where M_0 and M_i represent respectively the initial and the actual markings, W is the incidence matrix, and S is the firing sequence. The event matrix E is an n x n matrix such that $E = \text{diag}(E_i)$, with $i = 1,2,...,n$, and where n represents the number of transitions of the synchronized PN. E_i ($i = 1,2,...,n$) represents the event corresponding to the transition t_i, with

i = 1,2,...,n. The simulation of externel events is done in the two following manners. When the number of transitions is small, a keyboard key is associated to every transition of the net. Pressing a key means the occurence of the event of the corresponding transition. When the number of transitions is large, the external event is simulated by pressing on the corresponding transition by means of the computer mouse.

In a timed PN [5], the functionning of a system is time dependent. The timing may be associated with the places (the PN is said to be P-timed), or with the transitions (the PN is said to be T-timed). Timed PNs are well suited for modeling systems in which a certain time may elapse between the start and the end of an operation. They are useful for evaluating the performances of a system. For example, the simulation of a P-timed PN is done in the following manner:

a) A time delay d_i is associated with each place p_i.

b) A counter C_i, with an initial zero value, is associated with each place p_i.

c) When a token is deposited in place pi, the following operations happen:

i) The token remains in this place for, at least, a time delay d_i. This token is said to be unavailable for this time.

ii) The time delay d_i is transmitted from the place pi to the counter C_i.

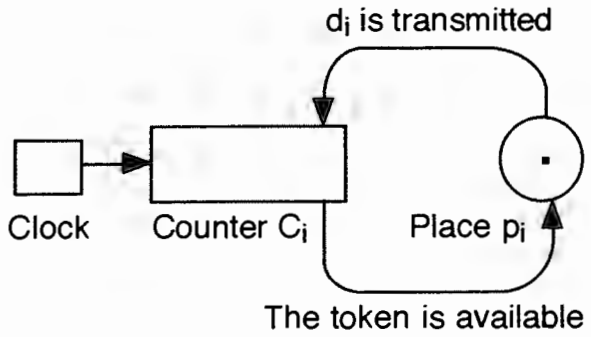


Fig. 6 Simulation of a P-timed PN

iii) The counter counts down at each clock pulse.

iv) When the d_i has elapsed the counter is reset to zero, a pulse is transmitted to the place pi in order to make the token available (see figure 6).

6. Conclusion:

We have presented a computer-aided tool which assist the drawing, analysis, and simulation of flexible manufacturing systems. The tool presents several interesting characteristics. The most notable characteristics are the introduction, for the first time, of the algorithm proposed in [3] which allows to obtain all minimal invariants of PNs, the utilization of an object-oriented language (the BORLAND C++) which makes the tool flexible and easy to use, and the possibility of drawing very large pNs because the authorized building space is not limted to the computer screen but may be extended to a virtual screen. Some improvements of this computer-aided tool are steel going on.

References:

[1] DAVID, R., and ALLA, H., 'Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems'. Prentice-Hall, London, 1992.

[2] MURATA, T., 'Petri nets: properties, analysis and applications'. Proc. IEEE, Vol. 77, 1989, pp. 541-580.

[3] AMER-YAHIA, C., and ZERHOUNI, N., 'A mathematical method for calculating minimal invariants in Petri nets'. (In preparation).

[4] MOALLA,M.,PULOU, J.,and SIFAKIS,J., 'Réseaux de Petri synchronisés'. Journ. RAIRO, Automatic Control Series, Vol. 12, 1978, pp. 103-130.

[5] RAMCHANDANI, C., 'Analysis of Asynchronous Concurrent Systems by timed Petri Nrets'. PHD Thesis, MIT, Cambridge, MA, 1973.

[6] LE MER,E.,'OVIDE: A software package for verifying and validating Petri nets', IFAC Software for Computer Control, Madrid, Spain, 1982, pp. 255-260.

[7] COLOM, J.M., MARTINEZ,J., and SILVA, M. 'Packages for Validating Discrete Production Systems Modeled with Petri nets'. In Applied Modelling and Simulation of Technological Systems, Elsevier Science Pub. B.V., 1987, pp. 529-536.

Simulation-Aided Investment Analysis for Manufacturing

Florian Klug
Institute for Operations Research and Systems Theory
University of Passau
Innstraße 33, 94032 Passau, Germany
E-Mail: klug@moni.fmi.uni-passau.de

INTRODUCTION

During the planning phase of a manufacturing plant, a main objective of the designer of the system is choosing between different design options. This implies working out a cost function which minimises the overall cost of the planned manufacturing system over its lifetime at a given, expected performance level. Due to the long timescales required for large technical projects the resulting decision problem can be extremely complex in its dynamic behaviour and underlying uncertainties.

Simulation is one of the decision support technologies for the planning phase of such systems. It gives the designer a tool for validation of technical and logistic options at an early point in the project, which reduces risk and - by enabling early recognition of errors - shortens the planning cycles.

Traditionally, the values for the relevant technical variables - obtained by simulation runs - have been used for decision support. This approach, however, leads to inefficiencies due to cost factors being neglected in favour of technical factors. While the method is usually sufficient to ensure that the chosen design will satisfy all the technical restrictions and boundary conditions, the final choice of design (among several sufficient ones) is often down to intuition. If we want to optimize this final selection then we need to introduce a cost-related target.

SIMULATION-BASED COSTING

To support such cost-based target functions a costing system has been developed, which is suitable for simulation and which consequently, could be integrated in the existing simulator for manufacturing systems, OSIRIS.

The new costing approach is required to support assessment of the design alternatives. Furthermore, it needs to allow integration of features which are specific to a particular simulation model, a requirement which made traditional costing system inapplicable.

Traditional costing methods which were developed with different requirements in mind tend not to meet the requirements of simulation based costing and must therefore, be excluded from the list of candidate costing methods. Some, more recent approaches to costing attempt to achieve a more realistic mapping from the real world to the costing model and therefore, need to be investigated as to whether they can be integrated into a simulator. The following, four methods have shown initial promise:

- Riebel's unit cost calculation
- machine hour accounting
- logistics costing
- activity based costing

Since each of the methods only fulfills some of the requirements for a fully operational cost simulator we need to be selective and re-combine as well as modify elements from each of the approaches in order to arrive at a costing system which can be used for simulation. This new approach was developed application-independent, and was then integrated into the OSIRIS manufacturing simulator, for verification. There are some general requirements for simulation-oriented costing methods - we included the following in our implementation:

- a realistic, cost-based representation of the manufacturing process
- user friendly input mechanisms for cost parameters
- re-use of existing accounting data in order to avoid duplication - acquisition of cost data at source
- cost attribution according to Riebel's relevance criteria.

The following applications are possible:

a) calculation of unit cost for a segment of the manufacturing plant
b) economy studies for single components as well as for the manufacturing plant as a whole are enabled by input/outputanalyses.
c) sensitivity analysis of cost drivers through simulation leads to better planning solutions.
d) cost based control of the manufacturing process is enabled. Up to now, control strategies for flow of materials has been based primarily, on quantity-related or time related parameters (e.g. load factor, throughput-time). In order to improve the quality of micro-economic decision making, these need to be transformed into cost factors.
e) analyses of overall value enhancements, based on individual value enhancements by cost factors or groups of cost factors
f) simulation support for logistics-based costing

SIMULATION-AIDED INVESTMENT

In economics, the evaluation - and assessment of comparability - of different options is regarded as a classical investment problem. Investment theory approaches decision problems by looking at cash flows between input and output categories. Charting the cash flow will not only show the current financial situation of a company and its solvency over time, but will also give indications about the profitability potential of alternative investment plans.

Practitioners who were forced to deviate from the rigorous notions of accounting as used by investment theorists, paved the way for the use of cost-oriented parameters in investment analysis. Cost-Volume-Profit models (CVP) have cost-theoretic foundations whereas techniques such as net present value (NPV), payback-method, accounting rate of return, or internal rate of return (IRR), as well as some modern Operations Research models are based on the theory of capital budgeting.

We believe that it is possible to use cost-oriented investment analysis (which is not based on the theory of capital budgeting) for solving known types of investment problems. Historically, these cost-oriented procedures were developed for purely practical reasons: it tends to be very difficult for companies to conduct sufficient data acquisition for assessing future input and output. This leads decision makers to turn to existing cost data in order to support their decisions.

One might feel that because of its cost-theoretic foundations, cost-integrated simulation is just another kind of cost analysis technique. However, traditional CVP techniques are static methods

where observations are made at a particular point in time and where there is a very limited time horizon. They are therefore often termed "subgoal-oriented techniques". Cost-integrated simulation by contrast, simulates the life-cycle of the plant which is to be built. We therefore propose that the taxonomy extended by a new class of "dynamic investment calculations". The great advantage of this method is its practical usefulness while at the same time, overcoming virtually all of the shortcomings associated with static techniques.

Having established that simulation based costing can be used for investment analysis we investigated in what respects results would deviate from those of classical input/output analysis if investment simulation was used.

A theorem by Lücke shows that one can indeed use cost terms as the basis of an investment analysis technique and that, if certain preconditions are satisfied, there will be no deviations as long as relevant costs of project capital are considered over the time period in question. This also proves that we were justified in accounting for these costs in our cost analysis model which was developed prior to the work reported here.

REFERENCES

Cooper, R.: Activity-Based Costing - Wann brauche ich ein Activity-Based Cost-System und welche Kostentreiber sind notwendig?, in: *Kostenrechnungspraxis* (5/90), p.271-279

Gausemeier, J.; Erfolgspotentiale integrierter Ingenieursysteme (CAE), in: Gausemeier, J.(Hrsg.); *Proceedings CAD '94*; Paderborn 1994, p.1-25

Gutenberg, E.: *Grundlagen der Betriebswirtschaftslehre, Band I: Die Produktion*, 23. Auflage, Berlin, Heidelberg, New York 1979

Hartberger, H.: *Wissensbasierte Simulation komplexer Produktionssystem*, München 1990

Hummel, S.: Die Forderung nach entscheidungsrelevanten Kosteninformationen, in: Männel, W. (Hrsg.): *Handbuch Kostenrechnung*; Wiesbaden; Gabler 1992, p.76-83

Kaplan, R.S.: Ein einziges Kostensystem ist zuwenig, in: *HARVARDmanager* 3/1988, p.98-104

Klug,F.: Kostenintegrierte Fertigungssimulation, in: *Simulation in Passau*, 1994 Heft 1

Klug, F./ Fortmann, Ch.: Do Simulation Models In Manufacturing Make Any Sense Without Cost Analysis, in: *CISS - First Joint Conference of International Simulation Societies Proceedings*, hrsg. von Halin, J./ Karplus, W./ Rimane, R., Zürich 1994, S.415-419

Klug, F./ Fortmann, Ch.: Kostenintegrierte Simulation als betriebswirtschafltiches Bewertungsverfahren fertigungswirtschaftlicher Systeme in: *Simulationstechnik 9. Symposium in Stuttgart Oktober 1994* hrsg. von: Kampe, G./ Zeitz M., Braunschweig/Wiesbaden 1994, S.579-584

Plinke, W.: *Industrielle Kostenrechnung für Ingenieure*; Berlin, Heidelberg, New York; Springer Verlag 1989

Riebel, P.: *Einzelkosten- und Deckungsbeitragsrechnung; Wiesbaden*; Gabler 1990

Schneider, A.: Prozeßorientiertes Controlling und Rechnungswesen auf der Basis der I-Linie; in: Reichmann, Th. in Zusammenarbeit mit Sema-Group Paris(Hrsg.): *DV-gestütztes Unternehmens-Controlling*; München Vahlen 1993, p.24

Spur, G.: Simulation zur Auslegung und Optimierung von Produktionssystemen in: *Zeitschrift für wirtschaftliche Fertigung* 77 (1982) Nr.9, p.446-452

Witte, T.: *Simulation von Produktionssystemen mit SLAM*; Bonn, Paris, Reading; Addison-Wesley 1994

A Scalable 3D Animator with Open Interfaces

Heiko Kirchner, CePLuS GmbH Magdeburg
Ralf Helbing, Universität Otto von Guericke, Magdeburg

Introduction

The animation system AniPLuS was originally developed by the CePLuS GmbH Magdeburg. It is intended as a post-run animation tool for three-dimensional modeling and visualization of dynamic processes.

Animation is often used to provide a visual understanding of an abstract model that is used in a simulation system.

Today's simulation software allows the creation of very sophisticated models of real systems. However, this can only be done using abstraction techniques to formulate the reality in terms the simulator can understand.

Based on the same simulation model, several animations can be created, each of them tailored to it's specific purpose.

Since three-dimensional animation becomes more and more realistic, a user can intuitively apprehend the simulation and imagine how the real process works. This is especially useful in a presentation where some of the audience may be unfamiliar with either the simulation or the real system.

Furthermore, the animation can help the simulation process itself by revealing possible flaws in the simulation model.

Of course, creating the animation means additional effort, but this can easily be offset by the benefits mentioned above.

AniPLuS

Components

Our animation system consists of four major parts:

- the kernel which manages and processes all animation data and contains an interactive 3D modeler for the animation model and an interface to import geometry descriptions for the objects and their layout
- a rule editor to transform the simulation run into a series of movements and other visible alterations of objects
- an underlying graphics system to render the animation frames in rapid succession
- a graphical user interface

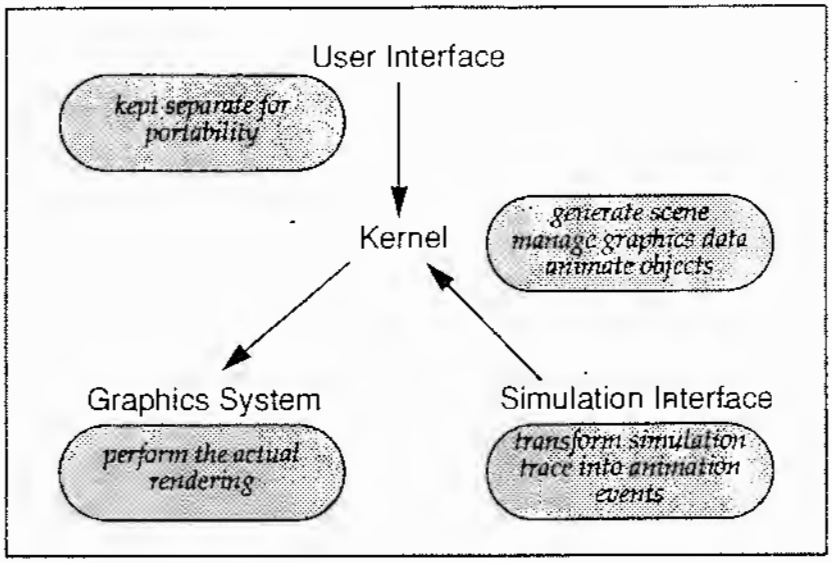


Fig 1: AniPLuS architecture

AniPLuS is available on several types of UNIX workstations, including SGI, SUN, HP, and IBM as well as the popular PC based LINUX system. If available in a fast (hardware accelerated) implementation, the OpenGL graphics interface will be used. Otherwise a very simple PHIGS-like implementation based on generic X11 is the graphics system. Although the OpenGL version is expected to be faster especially for models with a large number of small polygons, all[1] rendering functions are available in both versions. Providing two versions with essentially the same feature set allows to meet user-specific requirements for rendering quality and speed while still having a cost effective solution.

Ports to other systems like OS/2 are planned.

Creating Animations

In order to create animations, the animation system must solve at least three problems:

- import of geometry data
- transformation of simulation results into visible animation events
- running the animation

Thus, an animation consists of a static scene describing the initial configuration of the animation model and an ordered list of animation events.

The scene contains all the geometric information needed to display the objects. There are two kinds of geometry information: layout data and shape data.

[1] with the exception of things like gouraud shading and antialiased outlines

Layout information is stored in configuration files and defines the position, orientation, relative size, color, name and reference ID of each object. Shape information is defined externally in a DXF file. An object can have more than one shape. This is useful to indicate different conditions of an object in a more interesting fashion than only with colors, e.g. a transportation vehicle with or without load.

In order to create a scene, the user must:

- build a DXF description for each class of objects using a geometric modeler of some kind
- import these descriptions into AniPLuS
- create and place/edit instances of objects

Steps 2 and 3 are performed in the integrated scene editor. Once an object is placed in the scene, it can be selected and edited with respect to its position, orientation, size and color. The scene can then be saved for later use with an animation.

The list of animation events is what brings the scene to life. Each event can be thought of as a transition that affects an object in a certain aspect. At present, there are seven kinds of events: move, rotate, scale, color, shape, hide and show. The first three events can be arbitrarily long in duration whereas the rest of the list occurs immediately at the given time. Several data entries are common to all events: type, start, length and the targets object-ID. The rest of the event information varies with the type. Move/rotate/scale events have their respective final state as parameter. Color and shape events need the name of the new color. Hide and show events need no further information.

Interfacing the Simulator

As we visualize the behaviour of simulated objects, the creation of the animation model starts with the simulation. Currently, the focus is on discrete event simulators. There are five steps to build an animation from a given simulation model:

- identify which objects can be and must be visible
- build a scene with these objects using the scene editor
- set up a number of rules that transform simulation events and actions into animation events using the rule editor
- use these rules to automatically perform the transformation

Here step three is the most interesting one. It allows to create different animations based on one simulation run. Each animation can be focused on specific parts of the model and neglect less important issues.

Usually, each class of simulation objects (from the first step) will have a visible representation. Objects from that class can then be instanced in the animator, that is made visible at certain places in the layout. Simulation events implying the move of a specific object from one place to another will have a corresponding rule to rephrase this action in terms of animation events mentioned earlier. There can be a series of animation events generated from a single simulation event (an object moving on a path would have to perform a number of moves and perhaps turns to reach its target).

All these animation events can either be written to a file for later execution or be performed on-line.

Future work

AniPLuS is still in an early phase of its development and many improvements must be added to make it a stable and useful tool.

Fields of interest are:

- providing various levels of detail (adequate for the current hardware and desired frame rate) for the creation of high-quality animation with detailed objects vs. simple animations for previews and low-end hardware
- addition of an interface to the element-oriented simulator Create!
- a clean, easy-to-use GUI
- an OS/2 version of AniPLuS
- accelerated movements, text output and textures

Current work is focusing on two fields: the Proof™ interface and the use of cinematic knowlegde in the creation of asthetic animations.

The Proof™ interface will allow the use of many existing 2D Proof™ animations, layouts and objects in a 3D world. However, it requires a cleanly defined general simulation interface which can be used for other simulators, too.

Film techniques as used in movies would help the viewer better understand the simulation, since the language of film is intuitively known by almost everybody. Furthermore, their use allows the creator of an animation a much better control on how to fulfill his communicative goals and therefore provide an effective animation in terms of time spent by the viewer to „get the message".

Several models from different application fields with varying levels of detail and size have been built.

Current research about the integration of simulation and animation continues at the University of Magdeburg.

The Next Generation of Simulation Tools: A Focus on Usability in Micro Saint

by K. Ronald Laughery and
Catherine E. Drury
Micro Analysis and Design, Inc.
Boulder, Colorado USA

Simultaneously, the world of discrete event simulation software is both mature and, yet, in its infancy. It is mature in that the necessary modeling power of simulation tools is largely defined and supported by existing commercial packages. Discrete event models of virtually unlimited size and complexity can now be developed with commercial software packages on computers that sit on virtually every engineer's desk. However, it is in its infancy in that the number of *actual* users of discrete event simulation is a small fraction of the number of *potential* users. If one simply looks at the number of examples of discrete event simulation studies performed by the engineering community, it is obvious that relatively few engineers are aware of the potential for discrete event simulation, and far fewer are frequent users. Furthermore, discrete event simulation has wide potential use outside of the engineering community. For example, the management sciences could use simulation in many aspects of business process design and development. Yet, simulation is almost unknown to most managers.

One could simply attribute this lack of awareness and application of discrete event simulation to many potential factors that are beyond our control including:

- *It takes time to raise awareness and build a market for new software technologies.* However, if one considers what happened when computer spreadsheet software was introduced twenty years ago, it is obvious that users will quickly accept new technologies if the technology can be used to solve a wide variety of problems.

- *Simulation is inherently too complicated.* Certainly, one can create examples of system issues that require highly complex model designs. However, there are many straightforward problems that lend themselves to simple formulations with simulation. By and large, these straightforward problems are still not being addressed with simulation.

- *The simulation software market has not had sufficient time to develop mature tools.* Computers were invented to do simulation.[1] There has certainly been time.

Certainly, the above factors have contributed to slow growth in the use of simulation. However, we suggest that *the primary factors slowing the use of simulation is the same today is it was twenty years ago - most software still demands too much of the user in the way of computer programming skills*. While many commercial software packages have provided a more usable environment for specific modeling problems (e.g., robots), once the user must represent something out of the ordinary, some form of programming is usually required.

The last few years have seen an increase in the availability of graphically-based model building tools such as Micro Saint™, ProMod™, and Arena™. Also, more sophisticated windowing environments are making the models more transparent to the developers as they build and run their simulations - another key to enhanced usability. However, there are still many aspects of discrete event simulation packages that could be substantially improved to enhance usability.

From the outset, our mission in the development of Micro Saint has been to enhance usability without sacrificing modeling power. Many innovations over the past ten years have resulted in leaps forward in usability. However, we still struggle to develop the product and market education strategies that will lead to

[1] ENIAC, the computer that is widely perceived to be the first electronic digital computer, was developed in the 1950s at the University of Pennsylvania to simulate ballistics trajectories.

the mass market for simulation use. What are the factors that have kept the mass market from our doorstep? The remainder of this paper presents the three factors that we see limiting the growth in the market and some of the initiatives we are undertaking to address these factors at Micro Analysis and Design.

Limiting Factor #1 - The wide variety of skills of the potential user base - Simulation does require that the user have significant analytical skills. Even if we were able fully automate the process of translating a process flowchart into a functioning simulation, analytical skills would still be required to determine how best to chart the process and the appropriate level of detail for the problem at hand. Users must understand the basic concepts of systems analysis and measurement and many potential users do not. To some degree, the problem lies with the schools and universities. While we teach children the methods and issues when adding columns of numbers, as in a spreadsheet, we do not teach the methods and issues of system function decomposition. Accordingly, when new users of our products show up for training, we have a wide diversity of systems analytic skills. For example, in a recent Micro Saint training class, we had individuals with advanced degrees in engineering and computer sciences as well as individuals with no prior programming experience. This was a typical class.

The wide variety of skills in the potential user base creates a conflict to the developer and designer of simulation software - do we create software for the systems analyst or do we create software for the broader market which demands less analytical skills? Often, software elements that facilitate use by the novice simulationist will impede the experienced analyst. We propose two solutions to this conflict that are being used increasingly by modern software; layering and wizards.

Layering - Complex computer software today is often designed to provide many layers of usability and, hence, complexity to the user. For example the novice user of a word processor can create letters and other basic documents without learning how to format and write macros. Simulation software can do the same. For basic systems analysis problems with straightforward process flow and resource constraints, the software can provide usable tools. For complex systems, the user could have access to more aspects of the model of the systems behavior, such as complex decision behavior and constrained resource modeling.

An example of the concept of layering is presented in figures 1 and 2. Figure 1 presents a task description in Micro Saint where a complex set of resource requirements, as defined in the release condition. Figure 2 presents a new feature we are developing which will allow the user to define simple resource requirements associated with tasks.

Wizards - For the past several years, MicroSoft™ has been incorporating wizards into their software products. By invoking a wizard, the user is presented with a series of questions that helps him to use a feature of the software (e.g., set up a form for data entry into a database). Experienced users need not use wizards to use the features.

Figure 1. Defining complex resource requirements in Micro Saint

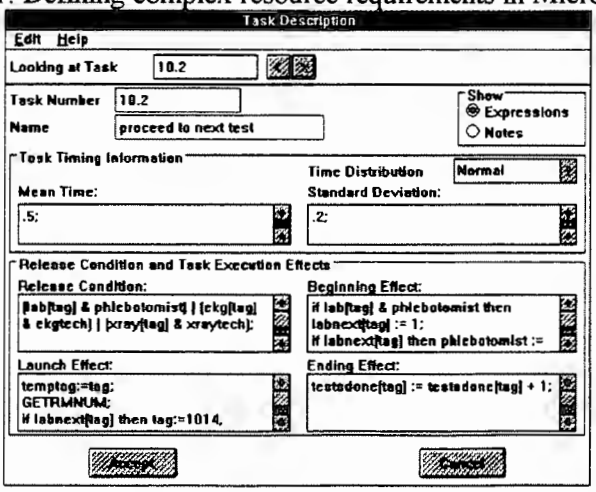
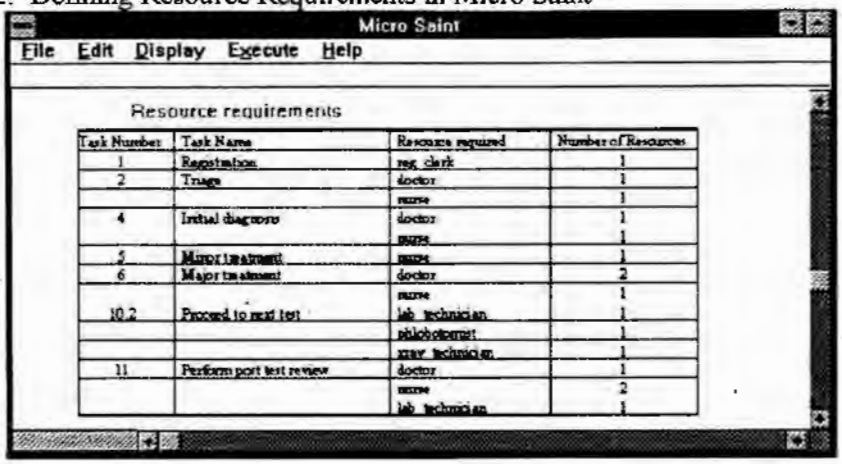
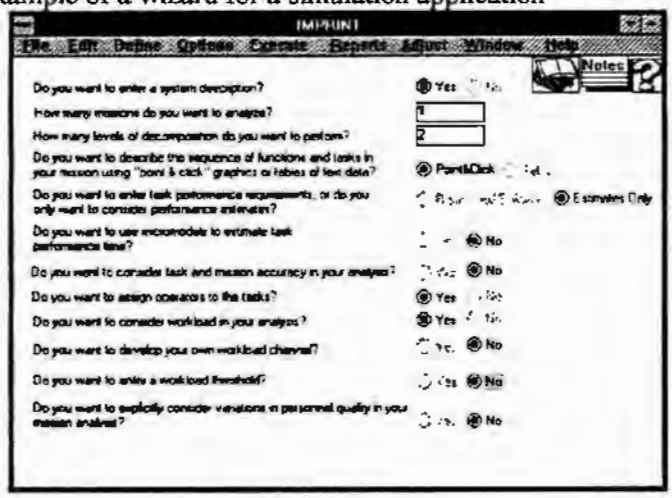


Figure 2. Defining Resource Requirements in Micro Saint



Many aspects of model development lend themselves to the use of this type of technology. These are, in a sense, expert systems that can assist users in a range of activities - from determining the appropriate level of detail for the model to building custom model data collection. While we have not incorporated wizards into our commercial software, we have developed wizards for several custom simulation packages. An example of a wizard user interface is presented in Figure 3.

Figure 3. An example of a wizard for a simulation application



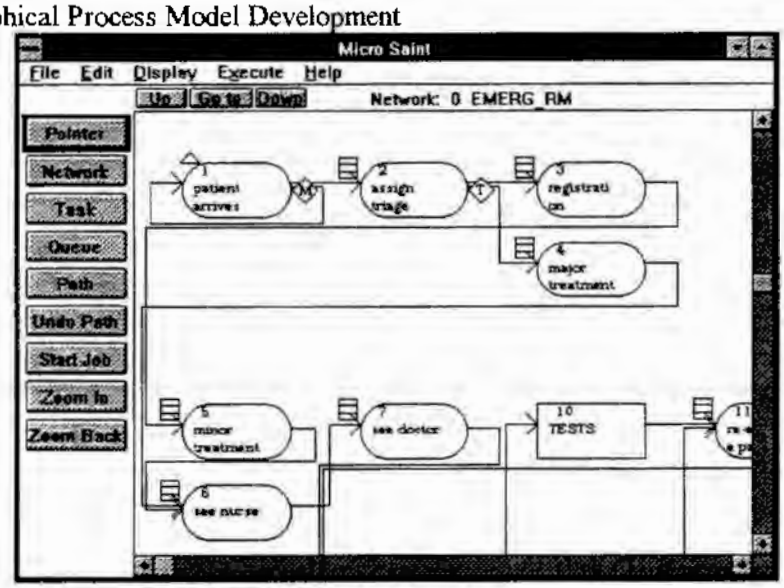
Limiting factor #2 - Dramatic differences between simulation tools, both structurally and in look and feel - A common problem we hear from potential users of simulation tools as they review the suite of tools that are available is that the tools all look so different. This leads to confusion on the part of the potential customer and makes it difficult to make a purchase decision. Potential customers like to think that they are comparing "apples to apples," which in today's market is not always easy to tell. These differences also limit the degree to which experienced users are able to change simulation systems. This is in contrast to, for example, spreadsheet software where all products are quite similar.

Why is the difference between simulation tools so dramatic? We suggest that it is the long history of discrete event simulation that has created these differences. Since the early 1960s, models have been built using higher level languages. A set of the simulation tools evolved using this language-based approach. In the mid 1980s, a set of tools emerged using modern software techniques such as language parsers and graphical development environments. These approaches are inherently different .

Time and the marketplace will allow the best approaches to emerge as standards. This is beginning to happen now. For example, many software packages provide a graphical process model development environment as exemplified in Figure 4 as the foundation for model design. As those of us building and

selling simulation software listen to the features that the marketplace likes and does not like, other aspects of different simulation tools will look increasingly alike.

Figure 4. Graphical Process Model Development



In the future, there will be forces beyond the marketplace that will encourage increased commonality. A Simulation Software Vendors Group now exists to direct and promote simulation software technology. This will result in some standardization of software functionality and look and feel.

Limiting Factor #3 - Price - Simulation software is still quite expensive in comparison to software that is targeted at similar markets (e.g., computer aided design software). This is part of a "chicken and egg" problem. Until there is a larger market, the fixed costs of developing the simulation software must be paid by a smaller number of users, thereby keeping costs high. However, the high costs are keeping the market small.

To expand the market, some software vendors, including ourselves, are promoting the use of simulation through the Universities. By providing simulation software at a low cost to the emerging user, the student, we hope to expand the base of users in the long run, thereby allowing us to lower our price. Additionally, the market must be sensitive to price and the vendors must continue to hear from potential buyers that price is a significant factor in deciding which software package to buy.

Summary - The wide variety of skills of simulation users, the differences in structure and look and feel of the software and the vast price differential are all areas that are driving our product development process. Our long-term goal is to increase the base of simulation software users by enhancing our products' usability, working with Universities to create a new market and eventually lowering the price to make simulation software a product that is on everyone's computer. In order to achieve this goal, we will continue to solicit ideas from the simulation community on new methods and constructs that will make simulation more intuitive to the experienced user as well as the vast number of users that we have not yet reached. Please contact us via email at sales@madboulder.com to comment.

X'': A Vision for Worldwide Simulation

Andy Symons, X-Prime Ltd.
Orteliusstraat 176-1, 1057 BK AMSTERDAM, NL
tel :+31 20 689 2022, fax: +31 20 6168142, email: andysymo@xs4all.nl

Keywords: modelling, simulation, client-server, object-orientation, internet, networks

Abstract: *a client-server architecture is proposed for the cooperative establishment of powerful, interactive, user-friendly modelling and simulation facilities in a worldwide network environment.*

Introduction and overview

X' (read "X prime") is an experimental system which has been under development since 1991 [1]. It has proved some important basic assertions about interactive modelling and simulation:

- that models can be captured directly *in user terms* (such as mathematical equations)
- that a compute-ready internal model can be *generated directly* from such a user model without use of a programming language
- that this generation (compilation) can be done *incrementally,* i.e. as the model is entered
- that experimentation on a model can be separated from the modelling process itself
- that the whole modelling and simulation process can be usefully controlled by an *object-oriented* graphical user interface (OO-GUI)

Further, it has been shown theoretically [2] that the automatic generation (compilation) of models can easily be extended to *parallel* computer architectures

However, as a development group of 4 people part-time, we had a big problem: we had too much work to make useful progress... so we did the only thing possible: we made the system concept even bigger!

The new concept described in this paper (dubbed, X'' or "X prime prime") is based on the idea of interacting clients and servers operating locally or over networks such as the Internet. The great advantages of this approach are:

a) separate clients and servers (of various hue) can be developed concurrently and independently by separate organisations
b) the potentially usable computing power for simulation is unlimited: any resource, connected by a network, can be employed

This paper is an introduction to the architecture and an invitation to universities and similar institutions to participate in the X'' programme by developing your own contributions in the form of client and server software. You may for example like to develope a better model editor client with direct translation of mathematical formulae, or a graphical input method such as bond graphs or process diagrams. You may like to implement a powerful compute server based on a parallel computing platform connected to the Internet. The *quid pro quo* is the possibility to combine this with our software or that from other institutions in the programme.

This is no unambitious plan, but it is eminently practical. We have already solved the 'difficult' problems. Everything required by the X'' specifications is based on technology that is either already available elsewhere or has been proven by us in the first X' project. The novelty lies in the new *combination* of the technologies.

Client-server principles

The basic idea of a client-server architecture is that the programs concerned with interfacing to the user (the *clients*) are separated from the programs concerned with the organisation of bulk data and, in our case, also the powerful computations required (the *servers*). In principle, clients and servers can be implemented on any platform. Usually (and easiest to visualise for X''), clients will be implemented on PC's or workstations with a windows-oriented graphical user interface (GUI); examples are a PC with MS-Windows, an Apple Macintosh or a UNIX workstation such as a SUN with X-Windows. Servers will usually be implemented on bigger machines such as UNIX mini-computers or mainframes, although small implementations will be made for test purposes and for use with small models, e.g. for teaching. Clients and servers are connected by some kind of network such as the Internet.

Those wishing to know more about client-server architectures are referred to the abundant literature on the subject.

So how can client-server ideas be applied to modelling and simulation? ... (see figure 1)

The X" Clients

We identify three types of X" client:

a) *Model Editor* clients
b) *Experiment Control* clients
c) *Results Analysis* clients

A *Model Editor* client edits *Model Components*. Model Components (synonyms: 'model objects', 'sub-models' or 'model block') are descriptions of items in the real world. Instances of Model Components can be invoked in other Model Components and can be characterised by (not necessarily constant) *parameters*. Thus an hierarchy is formed which we call a *Model*. Note that the Model is kept quite separate from numerical solution methods, which in X'' are part of an *Experiment* (see below). New Model Components can be *inherited* from other Model Components with full function overloading; therefore, the X'' Model Description Language can be said to be truly *object-oriented*.

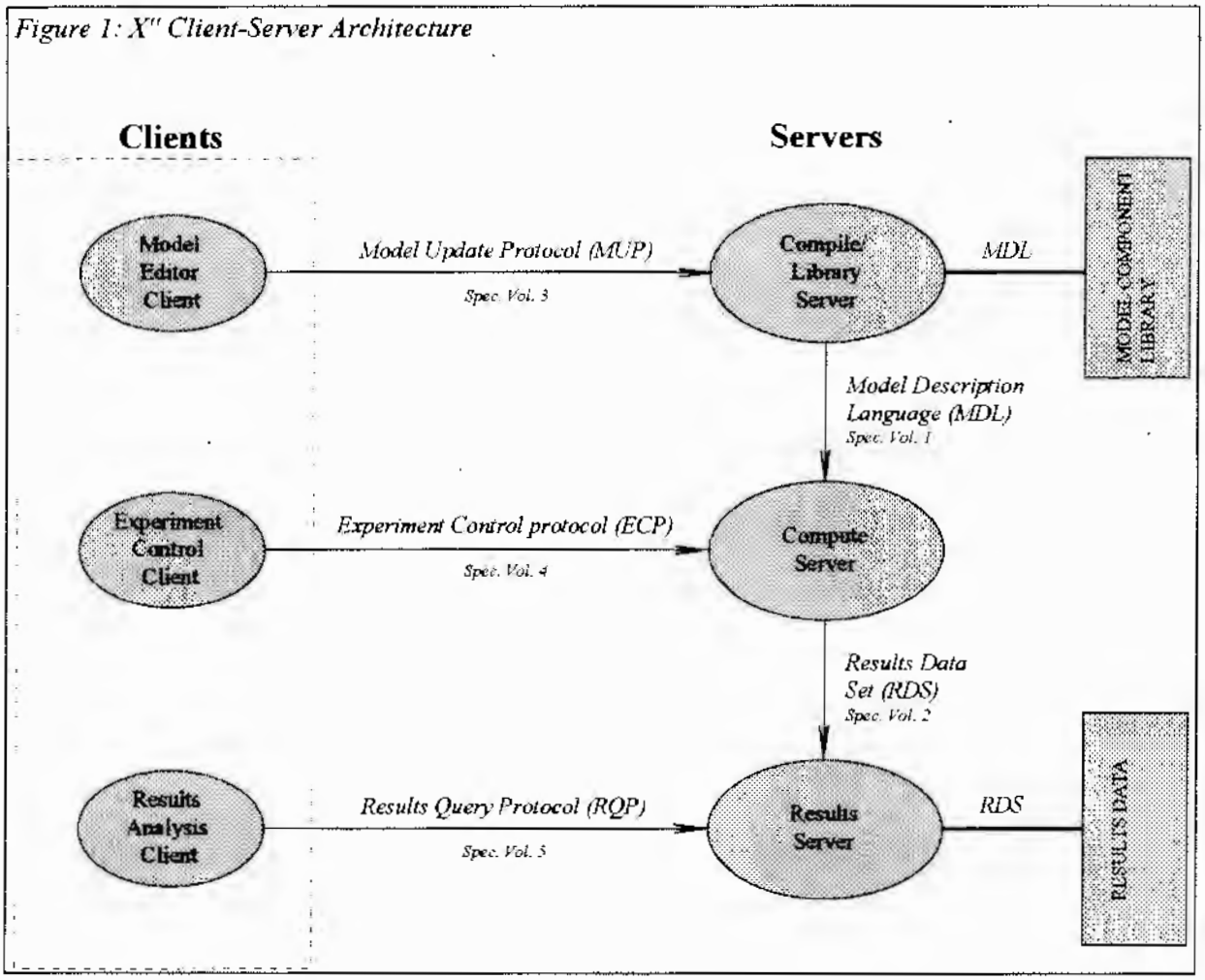
An *Experiment Control* client interacts with the user to ascertain what the user wants to *do* with the Model. This includes setting initial values, specifying simulation begin and end times, manipulating operator parameters and, if required, influencing the numerical methods which will be employed to calculate the results. The results can also be displayed as they are generated, if required. Running simulations can be 'paused', 'rewound', 'forward wound', influenced by operator parameters and restarted.

A *Results Analyzer* client gathers the results of one or more experiments and represents them in a variety of graphical forms. Results of experiments on one or a *number* of different models (including different versions of the same basic model) may be combined in one graph. Graphs can be displayed, printed or exported to other documents in, for example, a proprietary word processor.

The X" Servers

Servers are used to store data and do computations 'behind the scenes'. They only interact with users via clients. They will ultimately be implemented on middle range or large machines, including massively parallel configurations.

Figure 1: X'' Client-Server Architecture



We identify three types of server:
a) *Model Compile/Library* servers
b) *Compute* servers
c) *Results* Servers

A *Model Compile/Library* server converts the (external) model description entered by the user via a Model Editor client into an internal form suitable for (parallel) computing. It also keeps information on the model topology such as where variables are defined, the class and type of each variable and so on. A Model Compile server can act in an interactive mode so that a user is continually warned of syntactical and semantic errors and is continuously informed of the completeness of the model as it is built. Model Components can be saved and retrieved in a library. For this, existing file transfer protocol (FTP) technology can be employed. Model components include a text description to 'advertise' them to other users. These descriptions could be used to form search (e.g. WAIS) databases.

A *Compute Server* receives an experiment specification, which refers to a particular model[1]. The Model is already compiled as far as is possible without knowing the target compute platform. An *Assembly* step maps this internal compiled model onto the machine code and architecture of the target platform, then the calculations are done according to the experiment specification, using suitable numerical

[1] References could, for example, be based on the Universal Reference Language (URL), so the model components can be anywhere on the Internet).

methods. Several experiments can be carried out on the same model. The nature of X'' Model Descriptions is such that the computations can be carried out on massively parallel machines without any intervention from the modeller. The results are saved using the X-Prime Results Data Set (RDS) standard.

A *Results Query* server provides access (for a Results Analysis client) to a potentially vast wealth of results calculated from any number of experiments on any models. Preliminary location, extraction and sorting functions are provided, which can be refined further by the client.

The X" specifications

The key, of course, to a successful global client-server environment is a common definition of the data. Data in this case means the formats of the files used to save models and results, and the protocols used to transfer instructions and data between the clients and the servers. The principle is that all parties can develope software independently with the assurance that they can use each other's products, provided they adhere to the interfacing and data standards. To this end X-Prime is preparing the following specifications[2]:

Vol. 0: Introduction and Overview. The Architectural relationship between all the other documents. Read this first!

Vol. 1: Model Description Language (MDL). The detailed description of the internal file format for a model component.

Vol. 2: Results Data Set (RDS). The detailed description of the internal file format for the calculated results data.

Vol. 3: Model Update Protocol (MUP). The protocol used between a Model Editor Client and a Model Compile and Library server.

Vol. 4: Experiment Control Protocol (ECP). The protocol used between an Experiment Control Client and a Compute Server.

Vol. 5: Results Query Protocol (RQP).
The protocol used by a Results Analysis client to query the results database.

References

[1] "X-Prime: An Interactive Simulation Package". J.A. Burton, M.H.H. van Dijk, A. Symons.
Published in "Computational Systems Analysis 1992" - Proceedings of the 4th International Symposium on Systems Analysis and Simulation, August 25-28, 1992, Berlin, Germany. Elsevier Science Publishers B.V., P.O. Box 211, 1000 AE AMSTERDAM, The Netherlands. Editor: A. Sydow. ISBN 0 444 89780 1, pages 457-462.

[2] "What's the point in Parallelisation?". Author: A. Symons.
Published in "Massively Parallel Processing Applications and Development" - Proceedings of the 1994 EUROSIM Conference on Massively Parallel Processing Applications and Development, Delft, The Netherlands, 21-23 June 1994. Elsevier Science Publishers B.V., P.O. Box 211, 1000 AE AMSTERDAM, The Netherlands
Editors: L. Dekker, W. Smit, J.C. Zuidervaart. ISBN 0 444 81784 0, pages 321-331.

[END OF DOCUMENT]

[2] Documents can be retrieved from the X-Prime WWW homepage on *http://www.xs4all.nl/~andysymo/x-prime*

SIMULATION USING SITA

G.Jonin, J.Sedols, Dz.Tomsons
Institute of Mathematics and Computer Science, University of Latvia
Rainis Boulvd. 29, LV-1459 Riga, Latvia
Phone: +371-2-227815 ; Fax: +371-7820153 ; E-mail: jonin@mii.lu.lv

ABSTRACT

Simulation of any real system aims to some destination. If a new system is under development, the simulation is used as a tool for the investigation how the system will act under certain conditions. For existing systems, the simulation is applied in case when something should be changed. A great amount of simulation tools, like GPSS, ACSL, SIMULA, SITA, etc., are built in the world [1]. Any of them has certain advantages for the use to some of research directions - system design and specification, prototyping, testing, calculation of characteristics, data processing and analysis, hypothesis testing, forecasting, training, etc. The features and advantages are described that can be got using simulation system SITA in order to solve the above mentioned problems. The research branches, where SITA has been used, are illustrated with corresponding examples.

1. SYSTEM SPECIFICATION

SITA (Simulation and ITeration Algorithms) is offered for simulation of discrete event systems. It implements it's own simulation and specification language on the IBM-compatible PCs. The language contains a small set of basic elements that provides possibility to describe model's action scheme in a graphic form. Thus, it is easy to learn and to use the language for the system specification. The set of the basic elements is enough for description of large and complicated systems. New elements, more useful for some models, can be derived from basic elements. The usage of derived elements is helpful in the specification process. The description becomes more understandable. Thus, SITA is suitable both for teaching and learning, and for scientific and practical research.

The specification and simulation language SITA has it's software implementation - simulation systems SITA/B, SITA/C, and ITA (ITeration Algorithms) [2]. These systems provide the possibility to describe the models in different forms. So, SITA/B allows to build the action schemes of quite simple models by graphic elements. In ITA, anyone can describe quite simple models in a table form. ITA builds the system of equilibrium equations on input data, and solves it by using iteration method. By analytic calculations, as ITA does it, it is possible to get the system characteristics more quickly than by simulation.

The system SITA/C provides the model description in C-like programming language supplement by some simulation operators and pseudo-graphic notation. The simulation and analytic modelling by the same simulation program can be performed using SITA/C. Thus, the system is suitable to validate either how perfectly a model simulates the real system, or how accurate the given formula describes the behaviour of the real system.

The system SITA has been widely used for specification and simulation of teletraffic systems. In the current paper, a case study on simplified model of a factory supply department illustrates the SITA facilities. The department has N vehicles taking required elements from warehouse to workshop. Factory has K warehouses. In every of them V workers have been working to load the vehicles. The distance between any warehouse and the workshop differs. In the all warehouses, the average service time B is equal. The vehicles form the Poisson flow into warehouses with

parameter L. They choose i-th warehouse with probability P_i (i=1, 2, ..., K; $P_1+P_2+...+P_K=1$). There is possible to get the required elements with probability R in every warehouse. If the necessary elements are not in the warehouse, the vehicle is going to next one. Fig.1 shows the model's action scheme, built by SITA, for this system. Performing the simulation program, the information on system characteristics can be got, for example, to calculate the time consumed by a driver of vehicle in order to get the corresponding elements.

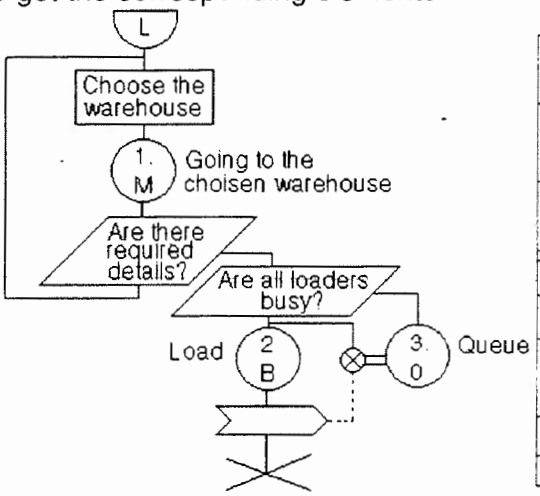


Fig.1. The model's action scheme for the factory supply department.

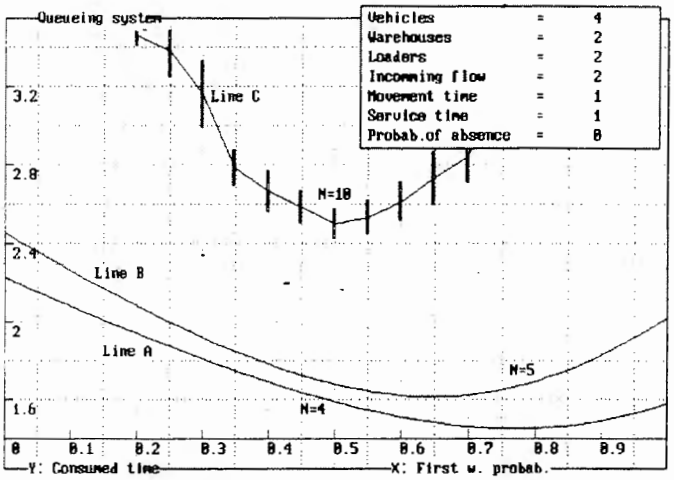


Fig.2. Simulation results.

2. CALCULATIONS AND DATA ANALYSIS

SITA contains a great amount of tools for processing and analysis of simulation results - for calculation of statistics, for building of histograms, for graphic building, etc. The usage of the tools provides possibility to apply the simulation as numerical method by which it is possible to estimate approximately the characteristics of the real system.

For example, Fig.2 displays the coherence between the average time, consumed by a driver of vehicle in order to get the required elements, and the probability by which the drivers choose the nearest warehouse. The parameters of the model are shown in the right upper corner of the Fig.2. Looking on the line A, someone can find: if there are four vehicles (N=4) and drivers are choosing the nearest (first) warehouse with probability 0.8, they spend the least average time for elements supply. Such information is helpful for the supply manager of the factory managing the elements deliver for the workshop.

3. NUMERICAL METHODS AND HYPOTHESIS TESTING

The system SITA provides both simulation and analytic modelling. SITA/C allows to do that by the same simulation program. However, the analytic modelling can be applied only under corresponding requirements. Thus, SITA can be used to validate how perfectly a formula or a statistical model describes the behavior of the real system. Besides, SITA provides possibility to save results of various trials. That allows to estimate how essentially the change of some parameters influences the model's behaviour and so on.

In Fig.2, the lines A and B display results calculated by analytic modelling, but the line C shows those got by simulation. Looking on the figure, someone can claim the following hypothesis: in the described model, if the number of working vehicles is quite large ($N \geq 10$), the less average time,

consumed for elements deliver, is spent in the case when the drivers are choosing the corresponding warehouse with probability 0.5. In order to examine that, the simulation results can be transferred to some statistical programs, for example, MINITAB, SPSS, etc. [4]. Subsequently, its tools can be used for testing of the hypothesis.

4. PROTOTYPING AND DEBUGGING

It is proved that the simulation system SITA can be a suitable tool for design and debugging of real-time systems [3]. SITA/C is useful for prototyping of these systems. In such system, both program modules, and technical devices can be substituted by their prototype. Besides, the "external environment" can be simulated using prototype. For this purpose, several actions should be done. At first, the model of the system must be built by SITA/C. Then the simulation program had to be debug and run; the results should be analyzed, and the efficiency of the simulated system must be estimated. Often, it is necessary to build some alternative models.

When the most appropriate model has been chosen, it's parts are replaced with corresponding programs and devices. SITA/C uses Turbo C (or Borland C) compilator. Thus, the system is more suitable for prototyping of systems built by programming language C (or C++). In order to SITA for debugging and prototyping of other systems, additional interfaces should be implemented between SITA/C and corresponding programming language.

5. TRAINING

Simulation has taken an important place in the training. Usually, visualization, economy (time-consumption, job-consumption, materials, etc.), safety, etc. are meant as the main advantages of the simulation in the instruction. SITA is a suitable system for training purposes, too. The system is easy-to-learn. Using SITA, students can describe the same model in various forms. That is why the system is suitable for training programming and modelling.

The system has a wide set of tools for data processing and analysis. Thus, SITA can be used to simulate the experiments and the results can be utilised for teaching and learning methods of the theory of probability and mathematical statistics. [4].

SITA animation facilities permit to use the system for demonstration of different processes [5]. Simulation animation is an efficient way how to show the development (dynamic) of the simulated system. At the same time, it is also appropriate tool for model validation and debugging. In SITA/C Trace mode, it is possible to form the file where the system records all the simulated events. A special SITA program translates this file to the ProofAnimation trace file. Subsequently, animation of simulation results can be performed by ProofAnimation.

Besides, SITA/C includes dialogue-building tools. That provides mutual interaction between either SITA and user, or SITA and another software system. The last feature is essential for contemporary simulation and animation. The facilities of simulation animation and dialogue-building are the main condition for implementation of simulation games by SITA. The games take especial place in the teaching and learning process. They are mainly used for training the management and decision-making. The games can be used both for student training, and for further education of specialists.

For example, the game has been designed based on the above described model. The player can play the game in one of two modes. In the first one, he is the dispatcher of the factory. His main duty is to deliver the elements in the appropriate time giving corresponding instructions to every driver,

which warehouse to attend. In the second mode, user plays the driver's role. The simulation program controls the remaining drivers. In both modes, SITA dialogue means provide the user's control over the corresponding drivers. When the game is over, he can compare efficiency of his strategy to the one computed by computer. Playing the game, user can get experience for analysis of different situations and for decision-making under uncertainness. Besides, the player better understands how the stochastic systems work.

6. CONCLUSIONS

Simulation is a powerful tool for the solution of various tasks in different application areas - science, hardware, education and other areas. The mentioned above enumeration of the application fields is not closed. It is only showing those classes of the tasks, where simulation system SITA is used.

REFERENCES

1. Kindler E. (1985) "Simulation languages" (in Russian), Energoatomizdat, Moscow, Russia, 288 p.
2. Jonin, G.L., Sedol, J.J. (1991) "SITA - The Language for the Description of the Teletraffic Systems Simulation Algorithms" in *Proceedings of 13th International Teletraffic Congress*, Copenhagen, Denmark, pp.219-224.
3. Jonin, G.L., Kalnina, D.A., Sedols, J.J. (1991) "The Application of the System SITA/C for Simulation of Package-Line Communication Networks" (in Russian) in *Proceedings of International Conference on Functionability Problems of Communication Networks*, Novosibirsk, Russia, pp.143-149.
4. Jonins, G., Sedols, J., Tomsons, Dz. (1994) "The Application of Simulation System SITA for Statistical Analysis and Education" in *Proceedings of 15th Nordic Conference in Mathematical Statistics*, Lund, Sweden.
5. Tomsons, Dz., Galibeckis, J. (1995) "The Animation Facilities within Simulation System SITA" in *Proceedings of European Simulation Multiconference*, Prague, Czech Republic, pp.837-840.

ECHTZEIT-ERWEITERUNG
for SIMULINK™ on PCs

DA/AD-board

The ECHTZEIT-ERWEITERUNG is an extension for SIMULINK to allow **realtime simulation** on a PC under MS-DOS and MS-Windows.

No additional software is necessary. You can use **directly** SIMULINK models for your realtime simulation. No code generation and compilers are required. Models with MATLAB- and S-functions in MATLAB-code or as MEX-files can be simulated in a realtime environment without any changes. Only discrete blocks and some blocks with memory are not allowed. These blocks will be added in a future release.

Sampling times from as low as 0,3 ms up to some minutes can be reached on a Pentium PC. Simulation results can be monitored during the realtime process using a ONLINE-GRAPHIC.

Application examples are:

- Evaluation of measurement values
- Hardware in the loop simulations
- Processcontrol
- Monitoring

Special simulation hardware and software is not necessary. The product supports access to standard data acquisition boards, as **Lab-PC+** from National Instruments, **DAS-1600** from Keithley Instruments and **PC30DS** from Meilhaus. Other boards are available on request. The product supports additonally boards for pulse generation and DA/AD-convertion.

With the ECHTZEIT-ERWEITERUNG we supply a very costeffective extension for realtime simulation. Current applications include testenvironments in industrial environments and at research labs and experiments for educational purposes.

The industry price (inclusive data aquisition board) is DM 4.950,00. Please ask for university prices.

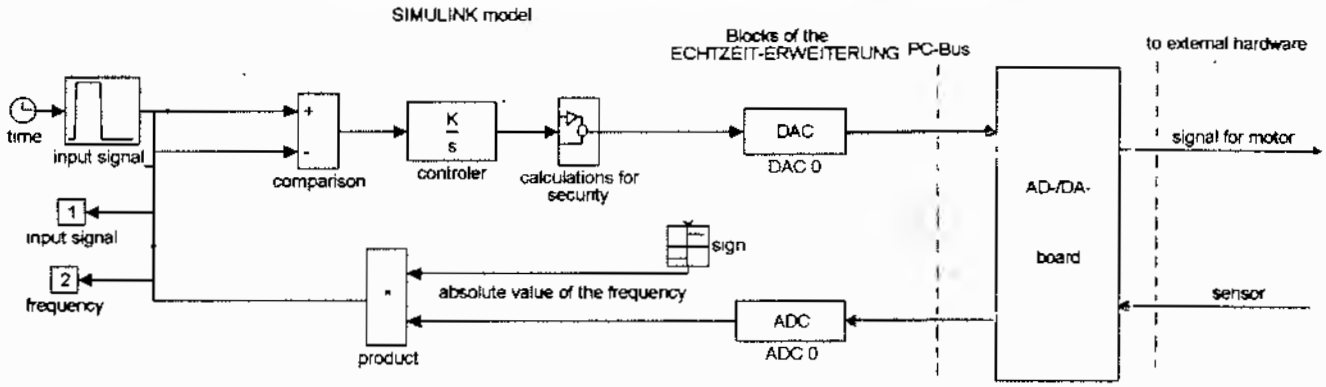
Example for a realtime simulation

The ECHTZEIT-ERWEITERUNG is called from the **MATLAB** command window as:

[t,y] = rtrun('MODELL',[T0 Tstop], Tsamp, X0);

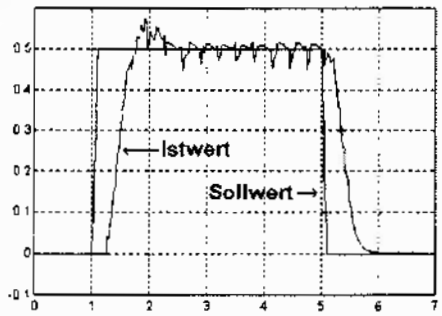
with starttime T0, stoptime Tstop, sample time of the realtime process Tsamp and initial conditions X0.

Realtime SIMULINK simulation model

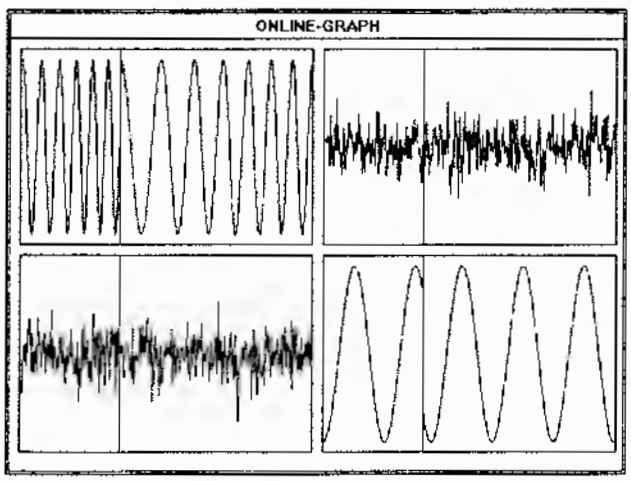


Realtime run

```
% Parameter of the I-controler
K = 3;
! Start realtime simulation
[t,y]=rtrun('drehzreg',7,0.01);
plot(t,y);  ! Plot results
axis([0 7 -0.1 0.6]);
grid;
```



ONLINE-Graphics



Linkage of a CAE-Simulation-Tool to a Realtime-Operating System

Klaus Beck
ARS Integrated Systems GmbH
Starnberger Strasse 22
82131 Gauting/München
Email: kb@ars-isi.de

Abstract

The use of the SystemBuild product for modelling a continuous engine coupled with an event based controller is presented in this paper. The example makes use of a recently developed interface between SystemBuild and the pSOSim product, a simulator for embedded tasks based on the pSOS+ real-time operating system. This integration allows high fidelity simulation of asynchronous and periodic controller tasks running on a real-time embedded kernel in an automotive environment.

1. Introduction

Popular computer aided control design products can effectively simulate continuous, discrete and hybrid models. However, real-time automotive controllers are not only periodic in time but are often asynchronous, i.e. event driven, having to respond to hardware generated interrupts. As a result, it has not been possible to use conventional control design software to perform high fidelity closed-loop simulations consisting of a continuous behavioural model of the plant combined with an accurate representationof a real-time asynchronous controller.

This paper demontrates the use of recent enhancements to the SystemBuild product that enables the closed-loop simulation of a continuous of discrete time behavioural model and a real-time event driven controller. The bevavioural model of the plant is numerically integrated within SystemBuild which controls the progression of the simulation time. The state-event cabability within SystemBuild is utilized to find the occurrence of sensor detected physical events which correspont to asynchronous interrupts within the control micro-processor. The event occurrences are utilized by a newly developed interface between the SystemBuild simulator and pSOSim to synchronize the execution of real-time control tasks to the simulation of a behavioural model. pSOSim is a simulator for embedded tasks based on the pSOS+ real-time operating system. pSOSim will also simulate the behaviour of pSOSelect, a highly scaleable version of pSOS+ that can require as little as 2K of ROM.

The example control problem used in this paper is a multi-tasking engine management controller that is both event-driven and time periodic. Modern engine control software is typically synchronized to the physical rotation of the crankshaft by means of sensors that generate interrupts in the engine controller. The tasks that are associated with the crankshaft sensors, are asynchronous in the time domain, but periodic in the crank angle domain. The control software also has time periodic tasks that calculate the engine speed which is used as a control variable.

2. Software Architecture

The simulation architecture, which is illustrated in Figure 1, consists of two processes running on the same UNIX workstation. The control loop is closed within the SystemBuild graphical modeling environment which completely encapsulates the simulation. The child pSOSim process is created automatically at the start of each simulation.

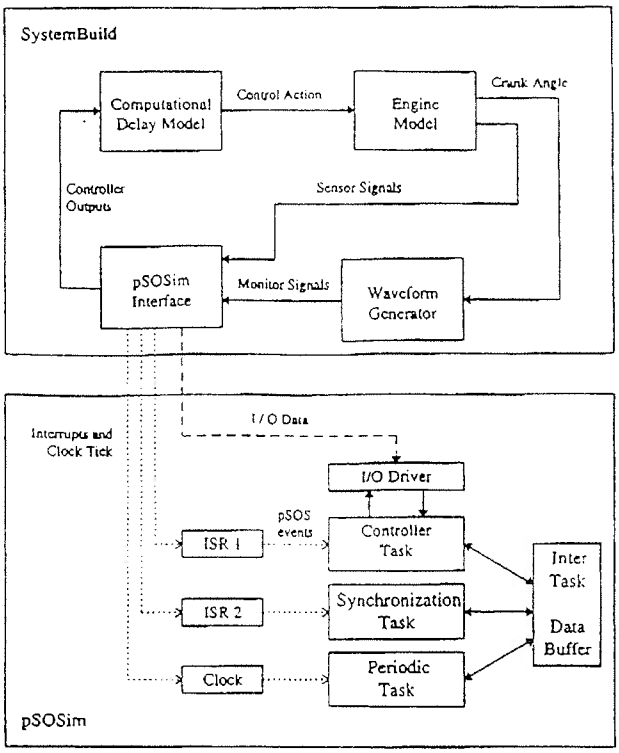


Figure 1, SystemBuild and pSOSim software architecture

The SystemBuild simulator performs the numerical integration of the system dynamic equations and controls the progression of time during the simulation. The four components of the SystemBuild model include the behavioural model of the engine, a waveform generator, the pSOSim interface and computational delay. The engine model is described later in this paper. The waveform generator uses the crank angle from the engine model to create an oscillating monitor signal for each rotational event. These monitor signals are input to the SystemBuild state-event detection algorithm where each zero crossing corresponds to the occurence of an event. The detected events can be time periodic, periodic with respect to a system state variable (such as the crank angle), or completely asynchronous. The magnitude and shape of the monitor signals are not important, provided that their zero-crossings occur at the proper time.
The sensor information from the engine model and the monitor signals are input to the pSOSim interface block. The interface block can have an arbitrary number of sensor inputs and is able to monitor multiple event signals. If the system has asynchronous events, the interface block must be defined within a continuous subsystem. In which case, each asynchronous event and the controller time clock will require a unique monitor signal. The monitor signal for the clock is generated from within the pSOSim interface block based on the specified clock tic interval. If there are no asynchronous events, then the interface block can be defined within a discrete subsystem where the sample rate corresponds to the controller clock tic interval.

Each time one of the monitor signals experiences a zero-crossing, SystemBuild supends the integration algorithm and releases control to the event processing software within the interface block. The SystemBuild simulator is blocked during event processing, after which the integration algorithm will compute a new consistent operating point and resume the simulation. In effect, the dynamic equations are numerically integrated piece-wise continuously between event.

The event processing software in the interface block communicates the sampled sensor signals and an event identifier to pSOSim through an inter-process communication (IPC) interface that uses a socket for data transmission and a UNIX signal for the handshake. The interface block will then wait for pSOSim to complete execution of the appropriate task in response to the event, and read the controller outputs back from pSOSim through the same IPC interface. When the pSOSim process receives the handshake signal from SystemBuild, it will automatically execute the appropriate

interrupt service routine (ISR) that corresponds to the event identifier. Each ISR can create a pSOSim+ event, causing data sampling and/ or task execution. Both the reading of the sensor signals and the writing of the controller outputs are done through a device driver that is written specifically to correspond to the communication protocol that is used between pSOSim and SystemBuild. The interface between the device driver and the pSOS+ tasks are written to emulate the actual device driver used in an embedded processor. As a result, the simulated ISRs and the tasks are the same as those that would be used within an embedded processor in a vehicle.

The computation in pSOSim as a result of an event are instantaneous with respect to the SystemBuild simulation. As a result, a computation delay must be modeled within SystemBuild between the pSOSim interface block and where engine model applies to controller outputs.

This simulation framework will not show the effects of preemption due to the occurence of events while the controller is executing a task. The exact timing characteristics of the controller software, including the potential for task preemption, are target specific and can be verified by running the controller software on a real-time processor.

The pSOS+ controller tasks can be user written, or automatically generated by the AutoCode software. AutoCode is a customizable code generator for SystemBuild models. AutoCode generated tasks that run on pSOS+ are realizable using a special set of template files.

3. Engine Model

The plant model used in this paper is a nonlinear 4-cylinder spark-ignition engine model [I]. Note that exhaust gas recirculation has been neglected.
The SystemBuild state event detection capability is used both in the engine model, and by the pSOSim interface.

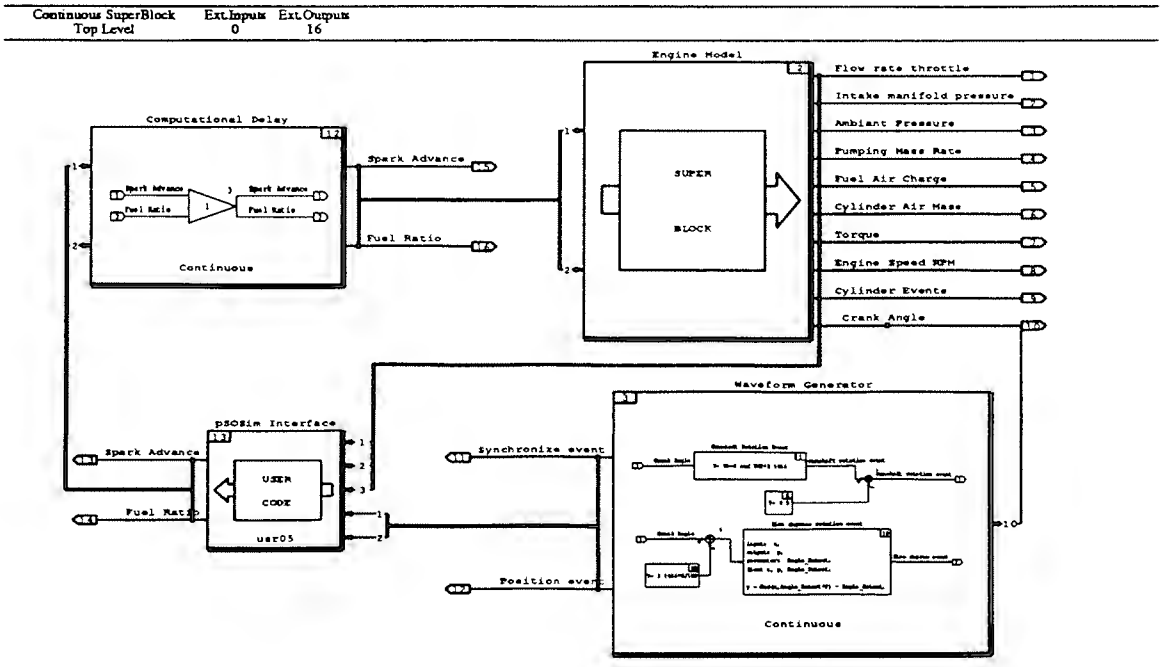


Figure 2, Top Level SuperBlock model

4. Real-time Controller

The real-time engine management controller is a multi-tasking controller that uses one periodic task and two asynchronous tasks. The periodic task is driven by a real-time clock in the embedded processor or by SystemBuild in this simulation. The two asnychronous tasks are driven by crankshaft mounted sensors that generate interrupts as the engine rotates. One of the sensors generates an interrupt every 5 degrees of crankshaft rotation and is called the position interrupt. The other sensor generates an interrupt every 360 degrees of engine rotation and is used synchronize the controller to

the crankshaft rotation. The three tasks cooperatively calculate two state variables, engine speed and engine position, and computes the control actions. Communication of calculated state variables between these tasks is accomplished through the use of a global inter-task data buffer.

The periodic task executes at 10 ms and performs the single function of calculating the engine speed, in revolutions per second (RPS). This is done by reading the position interrupt counter from the data buffer and deviding that value by the sample period. The calculated engine speed is then written back to the data buffer and the position interrupt counter is reset to zero.

The controller task is driven by the position interrupt and three functions. First it increments the position interrupt counter that is used by the periodic task. Second, it increments the position variable in the data buffer that represents the current physical position of the crankshaft. And lastly, this task calculates the appropriate control actions based on the computed state variables and the sensor values read from the device driver. The resulting control actions are finally written back to the device driver.

The synchronization task is driven by the less frequent interrupt that occurs every 360 degrees of crank rotation at top-dead-center for the first cylinder. This task resets the postion state variable which synchronizes the physical orientation of the crankshaft with the control software. During the startup phase, this task also enables the control task to safely take action by setting a flag in the data buffer once the control software computes the actual crankshaft orientation.

The control task employs a control algorithm that is implemented as table lookup structure that uses four inputs. These inputs the computed engine speed and the three sensor signals, intake mass flow rate, intake manifold pressure, and the ambient air pressure. The two computed control actions are the sparc advance and the fuel-air ratio. The engine model in SystemBuild employs the fuel-air ratio to manipulate a throttle-body fuel injector and the sparc advance to manage the engine performance.

This controller and engine model could be enhanced to support the simulation of various controllers and engines, including a sequential port fuel injection system where the injection timing of each individual port fuel injector is being sequenced with the timing of its respective cylinder. The number of sensor and actuator channels can be increased or decreased. The frequency at which the crankshaft position sensor generates events and the periodic task frequency are also customizable.

5. Implementation in a Real Vehicle

Implementation of a controller simulated in this environment requires the following steps:

1) Tie the ISR's controlling asynchronous tasks to externally generated interrupts.

2) Replace the ISR's controlling periodic tasks with events linked to timers. For example, the pSOS+ ev_every command can be used to generate an event every n pSOS+ timer ticks. For controllers running faster than the pSOS+ timer ticks, and ISR linked directly to a timer can be used, and the appropriate pSOS+ event can be created in the ISR.

3) Replace the I/O drivers with drivers for the physical I/O devices.

6. Conclusions

In this paper a coupling between two simulators has been presented and demonstrated. The simulators are SystemBuild, a general purpose simulator, and pSOSim, a simulator for applications running on the pSOS+ real-time operating system or pSOSelect. The combined environment allows accurate simulation of high fidelity plant models with periodic and event driven controllers. The actual embedded code for the controllers is used, which can be either hand written or automatically generated from SystemBuild.

ACSL For Real Time Simulation

Joseph S. Gauthier
MGA Software, Inc., 919-B Willowbrook Drive
Huntsville, Alabama 35802 U.S.A.
Tel: (205) 881-0947; Fax: (205) 883-5516

Introduction

The Advanced Continuous Simulation Language (ACSL) is a continuous system simulation language (CSSL) based upon the Society for Computer Simulation's CSSL standard. Based on Fortran and C, it permits the description of simulation models in terms of non-linear ordinary differential equations. Models written in ACSL are transportable among computer platforms ranging from PC compatibles to Unix workstations to mainframe supercomputers.

ACSL was first adapted for real time operation in 1975 for the U.S. Army Missile Command in an Analog-Digital hardware-in-the-loop environment. In this custom environment, ACSL resided on a Control Data mainframe host which would communicate with multiple analog consoles, digital minicomputers and actual missile guidance hardware in real time. This system supported radar, infra-red and electro-optical test environments for missile seeker and autopilot testing.

Since this first real time version of ACSL, there have been many other custom implementations of ACSL in real time world-wide. ACSL for real time (ACSL/rt) is available commercially on PC compatibles running Windows 3.1, Silicon Graphics workstations and Harris Night Hawk workstations.

There are many definitions of "real time". For ACSL, a real time simulation is a time critical simulation. If a real time ACSL model uses a ten millisecond integration step size, then it will take a single integration step every ten milliseconds of real time. The model must finish taking its integration step within ten milliseconds of real time or the simulation fails.

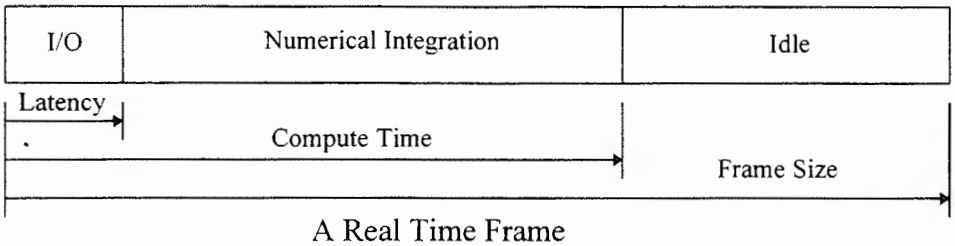
The special features of ACSL/rt required for real time operation are:

- Synchronize numerical integration steps with a real time clock
- Communicate with the outside world via real time analog and digital I/O channels
- Record time history data on disk for later analysis

These features are handled automatically by ACSL/rt so that models written in ACSL are platform independent.

The Real Time Frame

Real time simulation is concerned with hardware I/O rates. This update rate defines a time interval known as a *frame*. A simulation's integration step size is interpreted by ACSL/rt to be the size of the real time frame. On any particular computer system, ACSL/rt will see to it that the user's model is activated at each real time frame in order to take a single integration step. The frame size imposes an upper limit on the amount of work which can be done during an integration step.



A Real Time Frame

As shown, a frame is concerned first with performing I/O with the real time hardware. The time measured from the start of the frame to the end of hardware I/O is called the *latency* time. This is the time delay between the start of the frame and the point at which the model gets control to take an integration step. Latency also includes any overhead time required by the computer system to transfer control to the frame handler. The latency defines the smallest possible frame size which can be achieved by a computer system. The latency is determined by: the computer system's hardware architecture, software architecture, and the amount of time that the computer's CPU must be involved with moving data between the hardware I/O board and memory.

The *compute time* is the time measured from the start of the frame to the end of the model's integration step. It defines the smallest possible frame size which can be achieved by a particular simulation. The compute time is the sum of the latency and the time required to take an integration step. The factors affecting the integration step performance are: the computer's scalar floating point speed; the selection of the integration algorithm; and the total number of model equations. (The number of states in the model has minimal effect.)

ACSL/rt also supports multi-section simulations. Portions of the simulation can be assigned to different integration algorithms with different integration step sizes. In such systems, the section with the largest integration step defines the frame size.

Real Time Synchronization

ACSL/rt automatically handles the details of synchronizing with the real time clock. The simulation is only concerned with specifying the frame size by defining an integration step size. ACSL can use two different techniques to achieve synchronization: polling or interrupts. In the polling approach, ACSL continuously reads a clock driven counter until the frame interval time has elapsed. In the interrupt approach, the computer's hardware and software architecture are used to activate an interrupt handler which initiates frame processing.

The advantage of the polling approach is that latency is minimized by avoiding the overhead of interrupt processing and computer context switching. A disadvantage of polling is that the CPU is dedicated to the real time process. During the "idle time" described in the real time frame diagram, the CPU is spinning in a tight loop, locking out any other lower priority tasks.

The advantage of the interrupt approach is that the CPU can truly be idle during the "idle time" described in the real time frame diagram. This frees the CPU for other tasks. The disadvantage of interrupts is the potential for increased latency required for interrupt processing and context switching.

External interrupts can be handled with either the polling or interrupt approach. In the polling approach, the external signal is physically connected to a counter so that a change in counter value signals the start of a frame. In the interrupt approach, the external interrupt is assigned to the software interrupt handler which controls frame processing.

Real Time Hardware I/O

ACSL/rt simulations specify I/O operations to real time hardware boards by using a set of device independent operators:

- WDAC writes a scaled floating point value to a digital to analog channel
- WDIG writes an integer value to a digital channel
- RADC reads a scaled floating point value from an analog to digital channel
- RDIG reads an integer value from a digital channel

These operators handle the details of initializing the I/O board(s) and arranging for the exchange of data between the board(s) and computer memory. These generic operators are implemented for a variety of I/O devices. Specific devices are selected during ACSL/rt installation so that the ACSL models themselves remain transportable. The real time I/O operators provide default operations when used outside of real time, so that models can be checked out all digitally without requiring a special real time configuration.

Real Time Data Recording

The simulation's time history data needs to be written to disk for later analysis. The problem is that disk I/O is slow and not deterministic. In the PC implementation of ACSL/rt all real time processing, hardware I/O and data recording is handled by a single CPU. To minimize compute time, data is written to memory during real time and dumped to disk at the end of a run. Provision is made to record any number of data values for 10000 time points during a run on the PC.

Workstations configured for real time operation have multiple CPU's for true parallel multitasking. ACSL in real time on a workstation can dedicate a CPU to perform unlimited data recording

ACSL/rt for Windows on the PC

ACSL/rt on the PC runs a Watcom Fortran implementation of ACSL which operates as a 32 bit Windows/NT application. Microsoft's "win32s" software allows Windows to handle 32 bit NT applications. This configuration was selected for performance and for source code compatibility across DOS, Windows, Windows/NT and Windows/95 operating environments. The advantage of ACSL/rt on the PC is its price and ease of use. The primary disadvantage is its performance relative to workstation solutions.

I/O latencies as small as 50 microseconds have been measured under this PC configuration for short real time runs. It has been found that Windows adds almost a millisecond of overhead at irregular intervals when the same simulation is run for a longer duration. Windows was not designed to be a real time operating system with deterministic performance.

In the PC world, Keithley Metrabyte and National Instruments build real time I/O boards for the standard PC bus architecture. Synchronization on the PC version of ACSL/rt is done by polling the two cascaded 16 bit counters on a Keithley DAS-1600 board. Timer resolution is 100 nanoseconds.

ACSL/rt for Silicon Graphics

The Silicon Graphics (SGI) version of ACSL/rt makes use of SGI's "React" real time software package running on a multiprocessor architecture. SGI was selected for ACSL/rt because it is one of the major workstation manufacturers and because it has a vested interest in real time operations due to its market niche in real time video processing. The advantage of SGI is in dealing with a major hardware vendor, using Unix and the availability of a parallel processing solution.

SGI uses a multiprocessor approach to maximize deterministic performance in real time. The first processor (processor 0) runs the Irix (SGI's Unix) operating system and system daemons. The other processors can be dedicated to individual real time tasks. Since most real time I/O boards use the VME bus architecture and SGI uses a proprietary system bus, SGI computers configured for real time add a VME bus to the existing proprietary bus.

ACSL/rt uses SGI's React as an interrupt driven real time system. As a result, latency will be higher than in a polling implementation. SGI guarantees a latency of less than 200 microseconds, but this does not include hardware I/O time. SGI's CPU's are among the fastest in the workstation world, so what is lost in latency is made up during integration.

ACSL/rt for the Harris Night Hawk

The Harris Night Hawk is a computer system designed for real time simulation. Its CX/UX (Unix) operating system supports the Posix 1003.4 standard. This revision of the Posix standard provides for accessing high precision timers (one nanosecond resolution) to implement a polling solution to real time synchronization. Latency is about 20 microseconds. Like Silicon Graphics, Harris uses multiple CPU's to achieve deterministic real time performance. Unlike SGI, Harris uses a standard VME bus to facilitate the use of standard real time I/O boards.

The advantage of Harris is its demonstrated dedication to the real time simulation market. Its hardware and software solutions have been optimized for real time. In addition, Harris is an experienced real time systems integrator.

Other Platforms

ACSL/rt can easily be ported to other hardware/software platforms as the real time market develops. Potential future systems include Sun, HP, DEC Alpha and IBM workstations.

Summary

ACSL/rt extends the demonstrated success of ACSL's system simulation solution to the real time world. The details of solving the real time problem have been encapsulated into ACSL/rt so that simulations can easily move from the all digital world to the hardware in the loop world and from low end platforms to high performance platforms.

AUTOMATIC CODE GENERATION FOR MULTI-DSP NETWORKS ON THE BASIS OF SIMULINK BLOCK DIAGRAMS

Dr.-Ing. U. Kiffmeier
dSPACE GmbH, Technologiepark 25, D-33100 Paderborn, Germany
E-Mail: ukiffmeier@dspace.de

Abstract

For advanced controller prototyping and sophisticated real-time simulations the performance of a single processor may not be sufficient. A solution to these advanced performance requirements can be a multiprocessor implementation. This paper describes a tool, that allows graphical-oriented programming of multiprocessor systems for real-time applications. SIMULINK block diagrams are used to describe not only the model dynamics, but also the network structure of an application. The code generated by the Real-Time Workshop for each processor is automatically augmented by appropriate communication functions and embedded in a real-time simulation frame for multiprocessor systems. The close integration of the proposed tool into the SIMULINK environment allows easy handling of real-time multiprocessor applications.

1 Introduction

The real-time implementation of control systems by automatic code generation based on SIMULINK block diagrams has found its way into practice and is now widely used. The advantages of this approach are obvious:

- Easy graphical programming of system dynamics and shortened developement cycle.
- High portability of the application between different real-time platforms.
- Close integration into the MATLAB/SIMULINK® environment offering easy access to powerful toolboxes for analysis, synthesis and optimization of control systems.

The implementation of real-time code on single processors, e.g. digital signal processors (DSP's), is supported very comfortably by the Real-Time Workshop™ [Mat94] and the dSPACE Real-Time Interface to SIMULINK [dSPACE95]. If the performance of a single DSP is not sufficient, e.g. for a sophisticated Hardware-in-the-Loop simulation, an implementation on a network of multiple DSP's may be necessary. When facing a multiprocessor implementation, a number of problems can arise:

- The application must be distributed over the different processors of a network under the constraint that the CPU load of all processors should be nearly identical.
- The communication connections between the processors must be implemented. Deadlocks must be avoided and data transfer should be as fast as possible. In most control engineering applications a synchronization of the processors is necessary.

The programming of multiprocessor applications can be considerably simplified using a graphical-oriented approach. In this case, a SIMULINK block diagram is used to define the whole multiprocessor model including the communication connections. Based on such a block diagram description the *Multiprocessor Option for the dSPACE Real-Time Interface to SIMULINK* (RTI-MP) allows fully automatic code generation for multi-DSP networks with a single mouse-click. The implementation and background of RTI-MP will be described in the following.

2 SIMULINK Block Diagrams for Multiprocessor Systems

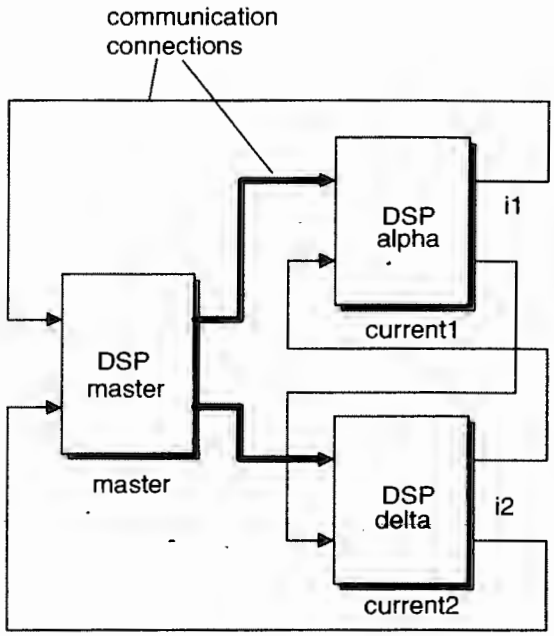


Fig 1: Network layer of a multiprocessor SIMULINK model

Fully automatic code generation is only possible, if all necessary information about a multiprocessor application is contained in the SIMULINK block diagram. This not only concerns the submodels to be implemented on each processor, but also the structure of the network and the communication connections between the DSP's. An obvious approach is to organize the block diagram in a hierarchical form as shown in Figs. 1 and 2. The top level of the model, the so called *network layer*, describes the number of processors involved, along with their connections. The submodels to be implemented on the different processors can be opened with a mouse-click on the corresponding DSP block on the network layer. This level of the block diagram, called *application layer*, can contain arbitrary SIMULINK blocks including hand-coded S-functions in 'C'. The signals to be exchanged with other processors are defined by special *ComPort icons*. For example, in Fig. 2 the signal *i1* is send from DSP „*alpha*", port 4 to DSP „*master*", port 2. The behaviour of the Simulink model is not affected by the ComPort Mux/Demux icons. This makes it possible to verify the dynamics of the whole multiprocessor model in an off-line simulation before implementing it on the real-time hardware.

The definition of the block diagram, i.e. the distribution of a simulation task over the different processors of a network, is the user's responsibility. In many cases the distribution of the model directly follows from the structure of the real system. For example, in an automotive application, the four wheels of a car could be simulated on four processors. The number of processors is not limited by software. Errors in the network structure, e.g. missing connections, are recognized and reported by RTI-MP automatically.

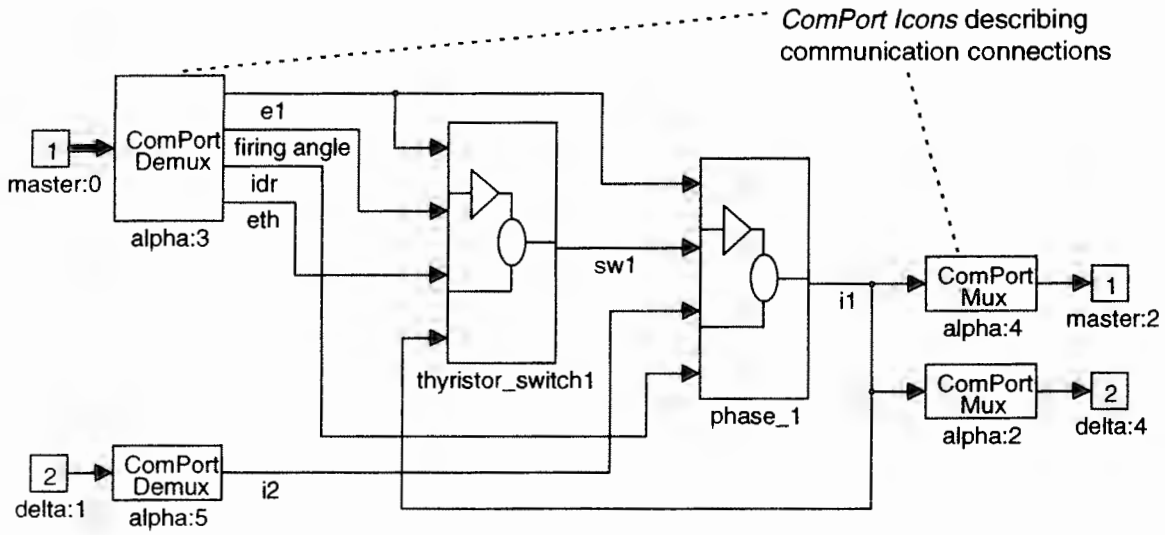


Fig 2: Application layer of a multiprocessor SIMULINK model

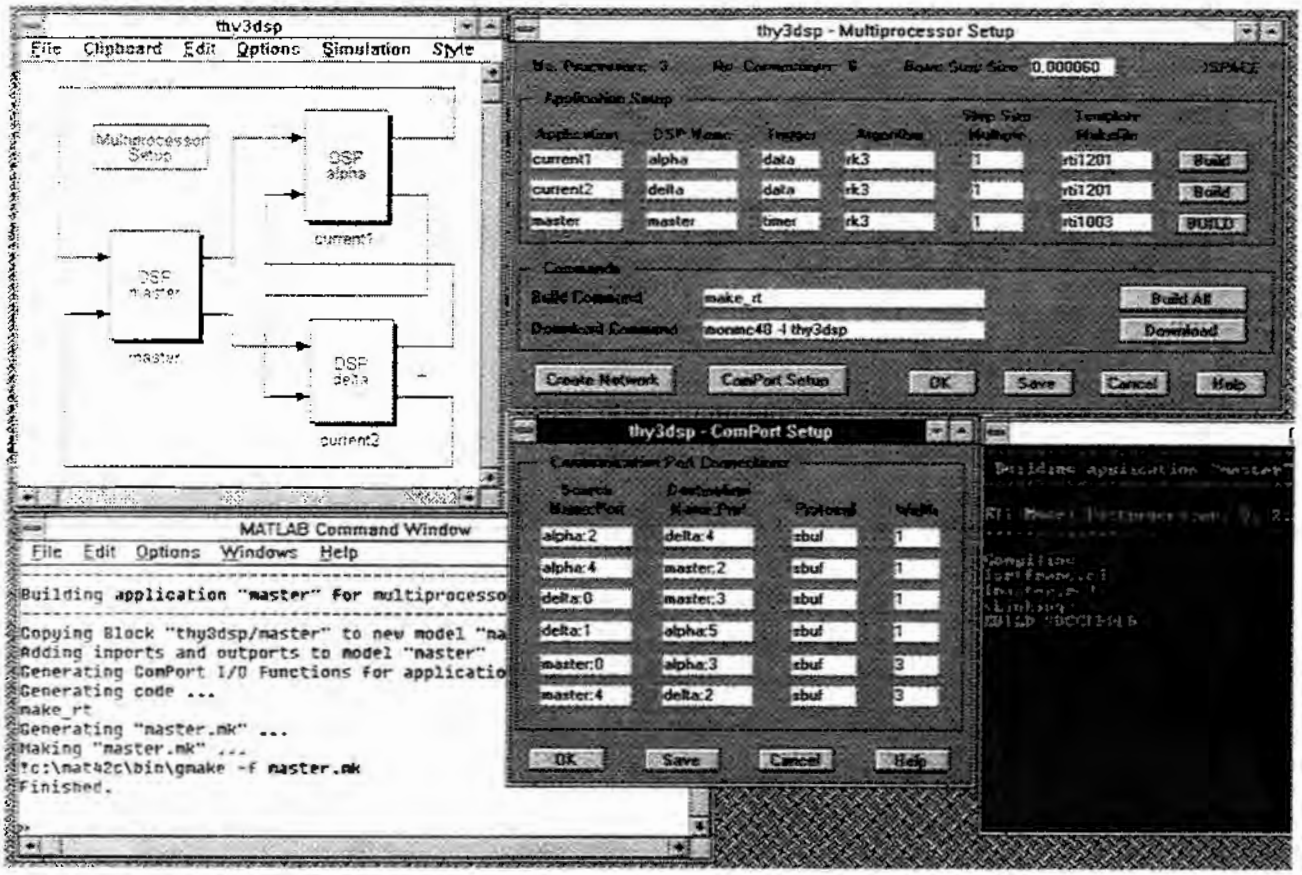


Fig 3: Automatic code generation with RTI-MP

The real-time simulation parameters are specified in a *Multiprocessor Setup* menu shown on the screen copy in Fig. 3. The following parameters can be assigned individually for each submodel of a multiprocessor application:

Algorithm The integration algorithm. Five different methods with fixed step size from *Euler* to *Runge-Kutta 5* are available.

Step Size Multiple Multiple of a basic real-time step size, used for integration of the model. The individual selection of the integration algorithm and step size allows the user to balance the load of the different DSP's in the network. For example, a small subsystem with fast dynamics can operate with a smaller step size than a large subsystem with slow dynamics.

Trigger This field defines whether the evaluation of a submodel is controlled by *timer* interrupts or by incoming *data*. In the first case, a new cycle of model evaluation is started at fixed time steps with a timer interrupt event. If a DSP is defined to be triggered by *data*, the corresponding submodel is evaluated as soon as a new vector of input data is received. This mode allows the fastest possible response on the inputs.

The parameters of the communication connections are specified in a second menu, the *ComPort Setup* menu also shown in Fig. 3. The generation of the communication functions and timing of the model evaluation is described in section 3.

With the **Build** and **Build All** buttons, automatic code generation can be started either for a single submodel or for the whole network. The Real-Time Workshop then generates a C file describing the dynamics of the corresponding submodel. This C code along with a set of communication functions generated by RTI-MP are embedded in a special real-time simulation frame for

multiprocessor applications. After successful compilation, the object files for each DSP in the network can be loaded to the multiprocessor hardware using the **Download** command. Before starting the application, the loader program verifies that all processors and the necessary connections are available on the hardware.

3 Communication in a Multi-DSP Network and Timing of the Model Evaluation

One of the most important factors for an effective multiprocessor system is a fast interprocessor communication. A processor very well suited for parallel processing applications is the TMS320 C40 DSP from Texas Instruments. Besides its impressive floating point performance of 50 Mflops, the C40 DSP offers 6 *communication ports* for high speed data transfer between the computing nodes. Each of the 6 communication ports achieves a transfer rate of 20 MBytes/sec. A DMA coprocessor enables data transfer parallel to the normal CPU operation. RTI-MP only uses communication mechanisms, that take advantage of the DMA coprocessor.

A very fast data transfer method is the *Virtual Shared Memory* (VSM) mechanism. In this case, the data is transfered continuously by DMA and written directly into the memory of the receiver DSP. While the DMA coprocessor is receiving data, the CPU executes its normal model evaluations at the same time. A synchronization with the sender is not performed, i.e. the receiver uses the available data as present at the time of reading.

In most control engineering applications it is desirable to have all input signals of a submodel result from the same simulation step of the sender; for example, if the submodel contains logical blocks. Since this is not guaranteed with the VSM, it is often better to use a *Swinging Buffer* (SBUF) mechanism for interprocessor communication. In this case, 3 buffers are allocated on the receiver DSP. While the DMA coprocessor is filling one of the buffers with new incoming data, the CPU reads its inputs from another. As soon as a new buffer is filled, the CPU can acquire it as new input, so that the data used by the CPU will always be consistent. Timing of the Model Evaluation

To achieve a computation flow free of deadlocks, the different submodels must be evaluated in a proper order. RTI-MP recognizes automatically whether or not a submodel has a direct feedthrough. According to this information, and the structure of the multi-DSP network, RTI-MP assigns the order of model evaluation for each DSP. To minimize the input/output delay of a submodel, the outputs are computed as early as possible and the inputs are read as late as possible. For systems without direct feedthrough, the outputs are computed before reading the inputs.

4 Conclusions

Automatic code generation for multiprocessor systems on the basis of SIMULINK block diagrams allows the user to access new classes of real-time performance. A key problem is the implementation of the communication between the processors and the timing of the model evaluation. Automatic generation of the communication functions with RTI-MP and simple graphical-oriented programming with Simulink simplifies the handling of multiprocessor systems enormously. Of course, it is the user's task to distribute a control application over the processors of a network.

References:

[Mat94] *Real-Time Workshop User's Guide*. The MathWorks Inc., 24 Prime Park Way, Natick, Mass., 1994.

[dSPACE95] *Real-Time Interface to SIMULINK User's Guide*. dSPACE GmbH, Technologiepark 25, 33100 Paderborn, Germany, 1995.

[Sch95] F. Schneider, F. Wienand and H. Rake: *Parallel Simulation of Complex Technical Processes*. EUROSIM Congress '95, Sept. 11.-15., Vienna, Austria.

scientific COMPUTERS

Technical Computing Environment MATLAB - New Developments -

Helmuth Stahl

Scientific Computers GmbH, Franzstr. 107, 52064 Aachen, Germany

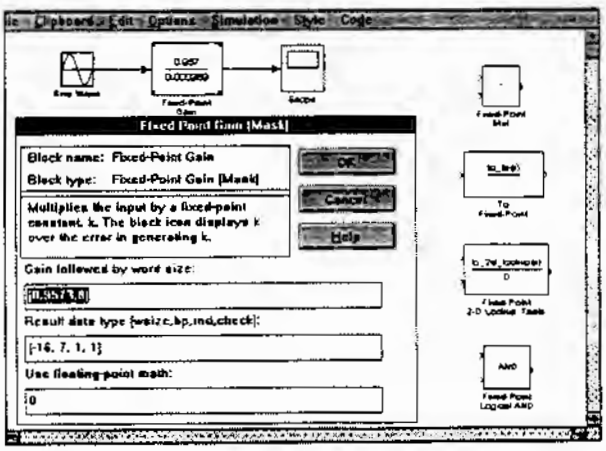
It is well known that ***Toolboxes*** extend the technical computing environment MATLAB for various application specific areas. Quite similar to this so-called ***Blocksets*** are available now to extend SIMULINK directly. These Blocksets constitute block libraries which may be used in SIMULINK simulation models and may be combined with other blocks as usual. It is important to know that the block functions are also supported by the Realtime Workshop.

The following pages shall give a general overview about new MATLAB related products which are just coming up or will come up in very near future.

• Fixed-Point Blockset - Block library for Integer Arithmetic in SIMULINK

The Fixed-Point Blockset offers SIMULINK V 1.3c users to simulate models or parts of them with fixed-point arithmetic. This new feature is based on a collection of more than twenty additional blocks like logical elements, blocks for the basic calculations and Lookup Tables. Users may perform calculations in an unsigned or the 2's complement format with 8, 16 or 32 bits. Besides that you may switch between floating and fixed-point format.

This allows to simulate effects that typically arise in control systems and digital filters applications which have been implemented on fixed-point hardware. When using the Realtime Workshop, C source code will also be generated for these fixed-point blocks.

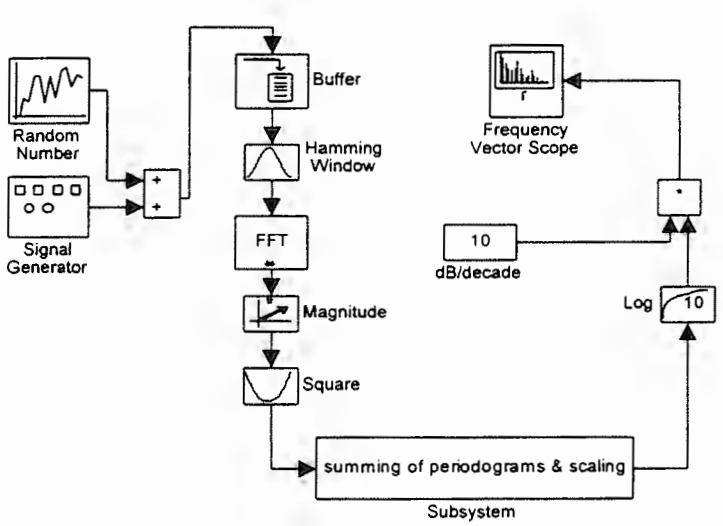


The figure illustrates some typical blocks of the Blockset and a dialog box for the fixed-point gain block. The dialog box offers the above mentioned ways to represent data types (*wsize=word size, bp=binary point location, etc.*) and to select floating point arithmetic. The block displays the gain quantization error when using an 8-bit word length.

• DSP Blockset - Block library for signal processing in SIMULINK

With more than 100 additional blocks the new DSP (Digital Signal Processing) Blockset extends SIMULINK in the signal processing areas. It targets engineers who need an easy-to-use software tool to develop and simulate DSP algorithms. This becomes increasingly important in areas like mobil communication techniques, medical or consumer electronics. Some of the major DSP Blockset diagnostic functions are:

- Basic operations in digital signal processing (e.g. FFT, correlation, etc.)
- Data buffering for parallel computing.
- Additional signal sources and sinks (e.g. FFT Scope)
- Filtering, filter design and windowing techniques (see Signal Processing Toolbox)
- Complex arithmetic (e.g. calculation of magnitude and phase) and vector-oriented math operations
- Compatible to Realtime Workshop, i.e. SIMULINK and the Realtime Workshop may be used to generate portable C code from a SIMULINK model for use on an external DSP board or processor.



DSP Blockset: SIMULINK block diagram to determine the power density spectrum according to the Welch method

The DSP Blockset is available on all MATLAB platforms. MATLAB 4.2c, SIMULINK 1.3c and the Signal Processing Toolbox 3.0b are required.

• LMI Control Toolbox - Linear Matrix Inequalities Control Toolbox

The LMI Control Toolbox is a new toolbox that extends the MATLAB product family in areas of control engineering and general math. LMIs are convex optimization problems that have to be solved e.g. in linear algebra, interpolation or control engineering and system identification.

The following equation depicts a typical example for an LMI problem:

$$A' * X + X * A + Q < 0$$

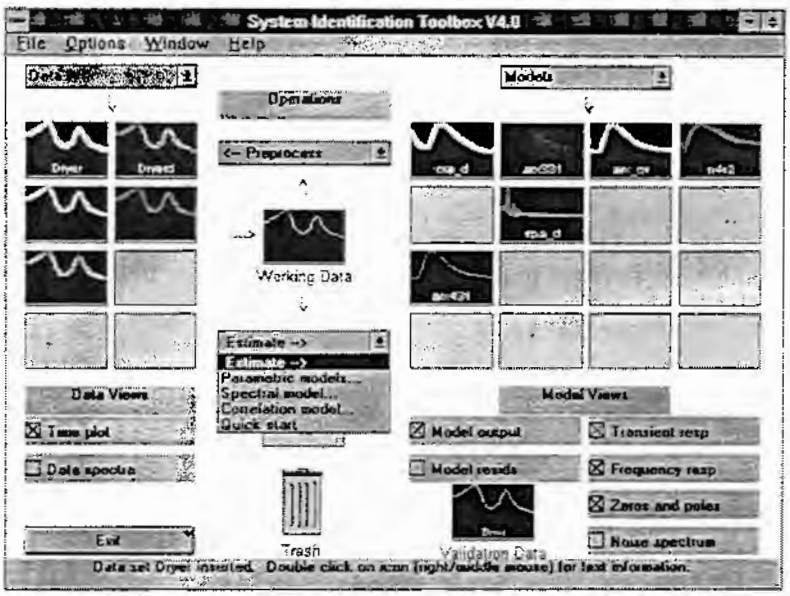
The strength of the LMI Control Toolbox is the availabilty of optimized algorithms and a powerful GUI environment.

scientific COMPUTERS

LMI will be available on all MATLAB 4.2c platforms. For control applications the Control System Toolbox is strongly recommended.

- **System Identification Toolbox** - Update: Version 4

The MathWorks responds to the request of many users and provides a graphical user interface (GUI) for the System Identification Toolbox. This new version includes a completely new GUI. Less experienced users and novices are guided throughout the system identification process (data preprocessing, modeling and model validation). Experienced users may compare different methods and actions transparently.



With the exception of one additional parameter esti-mation method for state space models the toolbox functionality did not change. As in the past all commands may also be entered via the MATLAB command line. The manual has been adapted with respect to the GUI.

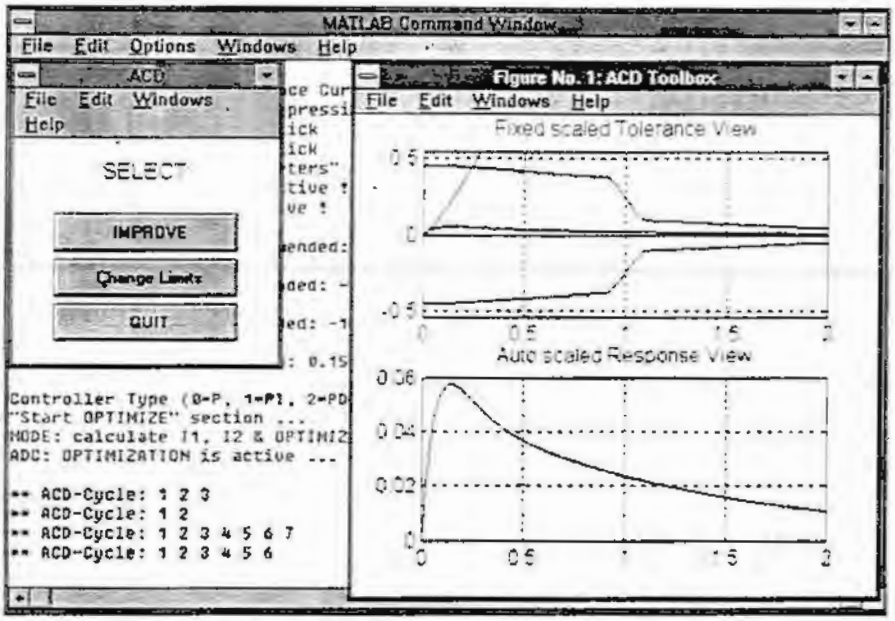
The System Identification Toolbox is available for all MATLAB 4.2c platforms.

- **ACD Toolbox** - Automatic Control Design Toolbox

This new toolbox provides powerful and very fast functions for a completely automated controller design which may also be carried out by non-experts in the control area. ACD especially provides all parameters for an ARW (Anti-Reset-Windup) controller structure to avoid undesired windup effects of compensators with an integral part. The major ACD characteristics are:

- results are controller transfer functions, also observer-based reduced-order state controllers or the parameters for the standard controller types: P, PI, PID-T_1, PD-T_1.
- works in a reliable way also in cases where the system to be controlled is unstable or contains pure time delay or system zeros are located in the right half of the s-plane.
- The Control System Toolbox is required.
- The desired responses and the corresponding tolerance regions may be defined graphically in the time domain. The ACD Toolbox may also be integrated into a user-written user interface.

- ACD is suitable to calculate good initial conditions for the Nonlinear Control Design Toolbox and the Quantitative Feedback Theory Toolbox.
- Closed-loop stability is always taken into account and guaranteed.



- Controller parameters for an ARW-structure are generated automatically, also for standard controllers.
- A reference filter is generated automatically. This means that optimized behaviours will be achieved both for disturbance and reference inputs.

- Robust controller design: several system models are allowed (parameter varied models or even different model structures).

● NAG Foundation Toolbox - Library of the Numeric Algorithms Group

The NAG Foundation Toolbox consists of a comprehensive set of more than 240 M-files with algorithms from optimization, statistics, partial and ordinary differential equations, integration methods, data fitting, etc.

The Toolbox is based on the NAG Foundation libraray which is well known for its comprehensive Fortran library containing more than 1100 routines for numerics and statistics. MATLAB users may access a set of these functions within the MATLAB environment. The function names of the corresponding M-files and the function call syntax are derived from the Fortran routines.

The NAG Foundation Toolbox is currently available under the MS-Windows-MATLAB.

● PCDAQ - PC Data Aquisition System for MATLAB

PCDAQ is a menu-driven PC software package for data aquisition and filtering for MATLAB. The aquisition is carried out in realtime while using Keithley Instruments' DAS1600 board.

Functionality: graphical representation of curves with printout capabilities on standard printers; measured data is stored in MATLAB format for post-processing within one of the various Toolboxes.

The simulation system ANA V2.0

J.W. Goldynia[1], J.M. Marinits[2]

ANA V2.0 is a CSSL based simulation system with a graphically driven frontend based on block scripts. It is able to handle state and time events in models at arbitrary precision and is available free of charge. The ANA product familiy is well established in education since 1986.

Introduction

The ANA V2.0 project has been initiated to support forthcoming control engineers with a powerful and yet flexible simulation system free of charge. The predecessor, the simulation program ANA 1.x, was invented in 1986 and is in wide spread use for control education at universities, technical colleges and courses. The secret of the success of ANA is the very simple user interface which allows newcomers to use the system from the scratch without any introduction.

The design goals of ANA V2.0 were to support the same userbase but to enhance the capabilities of the system tremendously and to define a solid base for future extensions. For that reason the capabilities of ANA V2.0 also satisfy the need of the professional user in industry and research.

The block oriented user interface of ANA 1.x is based on block diagrams. The contents of a block may include both differential equations and sequential algorithms to realize discrete or nonlinear behavior.

Since there is a need to introduce a full flagged CSSL within ANA V2.0 this new system is based upon a layer model. The frontend module ANAide covers all duties of input as well as presentation of results and composes an input script to ANAmdl - the CSSL compiler stage. The block library consists of ANAmdl templates which are assembled to a model.

Control engineers often deal with nonlinear and/or switching systems and therefore ANA V2.0 supports time events as well as state events. The possibility to determine the occurence of an event at an arbitrary accurency is just one of the advantages of ANA V2.0 towards ANA 1.x. An outstanding feature of this CSSL is the powerful Pascal like procedure processing which accompanies the state based ODE coding. This language has been successfully used to simulate complex hybrid models like power electronics circuits, helicopter flight dynamics and sewage works dynamics.

The ANA V2.0 CSSL-Compiler produces either PI code for immediate interpretation or C language code which needs to be compiled and linked with the runtime stage ANAsim. Integration algorithms for control engineering problems are adapted to handle events. Advanced explicit one step methods within the Runge-Kutta family of algorithms can be choosen amongst others to provide best results.

[1]Dipl.-Ing. Dr. Johannes W. Goldynia (goldynia@iert.tuwien.ac.at)
[2]Dipl.-Ing. Johann M. Marinits (marinits@iert.tuwien.ac.at)

Institut for Electrical Control Engineering, Vienna University of Technology,
Gusshausstr. 27-29/375, A-1040 Vienna, Phone ++43 1 58801 3906, Fax ++43 1 5058907

Some extensions, currently in process of development are a real time euhancemeut for hardware in the loop simulation, fuzzy logic extension aud artificial neural networks by improving ANAmdl as well as analysis in the frequency domain like bode, locus and root locus plots.

The ANA V2.0 project is a multiplatform approach and currently available for MS Wiudows 3.1, MS Windows NT, MS Windows 95 and X/Motif without license fees and can be obtained via anonymous ftp from `iert.tuwien.ac.at` (look for ANA2).

Example 1

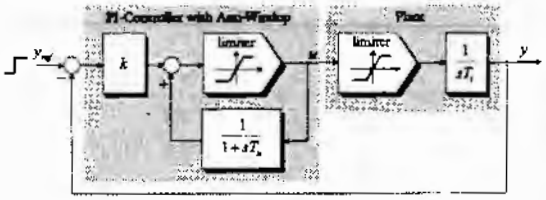


Figure 1: Control circuit with Auti-Windnp measure

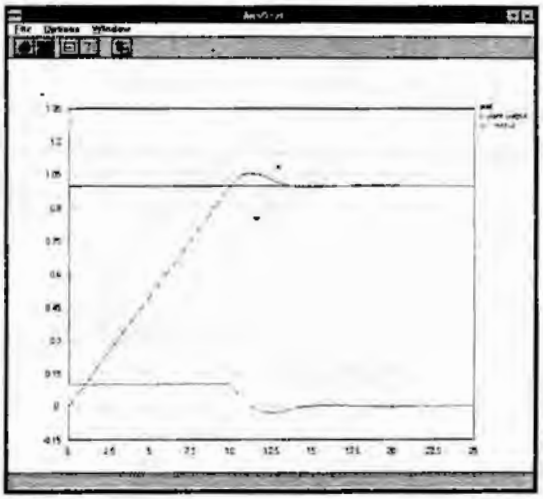


Figure 2: Step response simulated with ANA V2.0

Figure 1 shows a typical example of a simple control loop. The input of the model is straight forward since the graphical editor of ANA V2.0 is based on block diagrams. Figure 3 preseuts the graphical scheme including an additional block called `control`. Within this block the terminate condition (`te = 25`) for the simulation and the communication interval (`h = 0.25`) is set. All used blocks are part of standard libraries which can be extended or redefined by the user.

Figure 2 confirms that this structure of a PI-controller avoids any windup effect in the integral part of its structure. The plant output y is nearly a ramp function because the controller output is limited to the upper bound from the start.

The apperance of a block can be individually changed. This includes resizing and adding texts as well as drawings. Signal paths may be named. The user interface is intuitive. It can be driven by tool buttous, menus or mouse actions on the drawing pane. The bottom statusline informs the user about the current activity.

Example 2

The strength of ANA V2.0 is the CSSL called ANAmdl. This language is equipped with a powerful mechanism for handling time and state events. Each type of block element is defined by an ANAmdl description which acts as template if a block is used.

The statements in this template can be subdivided into information for the graphic interface (lines startiug with `$`), declarations of inputs, outputs, parameters, states and variables and finally the equation body consisting of intialization, a simulatiou part and event controlled procedures (performed sequentially).

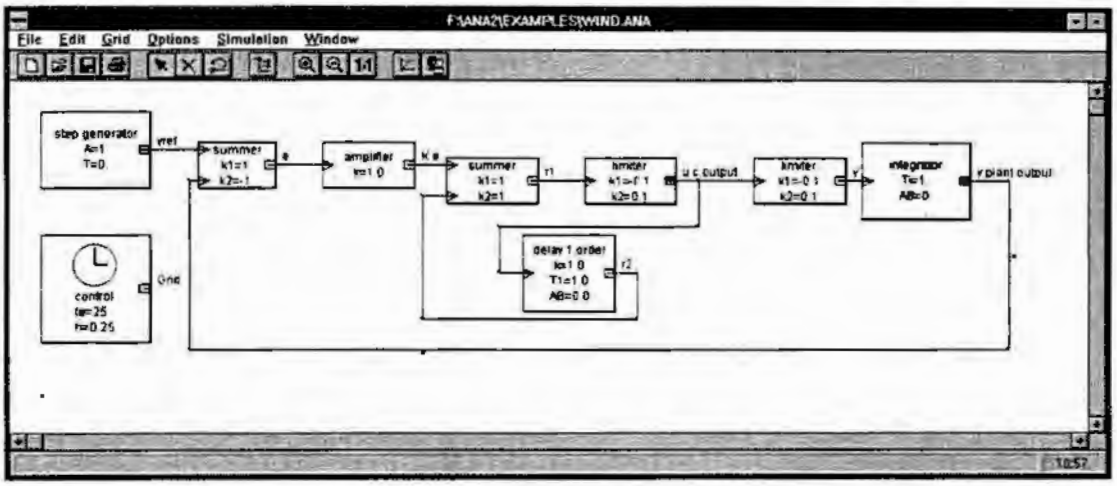


Figure 3: Control circuit drawn with ANA's graphic editor

A oue dimensional model of a 'Jumping One-Wheel' is presented in order to illustrate the state event capability of ANA V2.0. Figures 4 and 5 give a formal description, figure 8 shows the corresponding implementation.

Lines 36 to 44 show the differential and algebraic equations. Lines 45 to 54 put the state event dependent equation of the force f into action. The variable `state` is declared as `DISCRETE` and is therefore able to memorize its value. In line 72 it's initial value is assigned. The `SWITCH` statement distinguishes `state == 0` (mass 2 standing on ground) and `state == 1` (mass 2 up in the air). In the case where mass 2 is standing on grouud line 48 is carried out and the `ONRISE` condition on line 49 is constantly checked to trigger a call to the `PROCEDURE jump`.

Since ANAmdl allows arbitrary complicated `ONRISE` conditions, the evaluation of such a conditiou is controlled by specifying a boundary for the independent variable *time*, called *event precision*. ANA V2.0 guarantees that the associated procedure is called only once if the condition changes from *false* to *true*.

The procedure code of `jump` (line 59 to 63) changes the variables `state` and `fly` and saves an extraordinary communication record due to the statement `STORE ALL`. This ensures that the visualisation handles the discontinuities of the model properly.

Figure 6 and figure 7 show sample results. The variable `fly` was introduced to animate the state switching to the display.

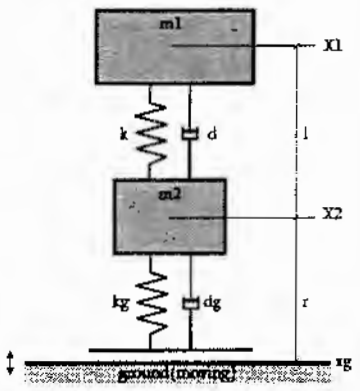


Figure 4: One dimensional model of a 'Jumping One-Wheel'

$$
\begin{aligned}
\frac{dx_1}{dt} &= v_1 \\
m_1 \frac{dv_1}{dt} &= -m_1 g - d(v_1 - v_2) - k(x_1 - x_2 - l) \\
\frac{dx_2}{dt} &= v_2 \\
m_2 \frac{dv_2}{dt} &= -m_2 g + d(v_1 - v_2) + k(x_1 - x_2 - l) - f \\
f &= \begin{cases} k_g(x_2 - x_g - r) + d_g(v_2 - v_g) & : x_2 - x_g < r \\ 0 & : \text{else} \end{cases}
\end{aligned}
$$

Figure 5: Simplified equations

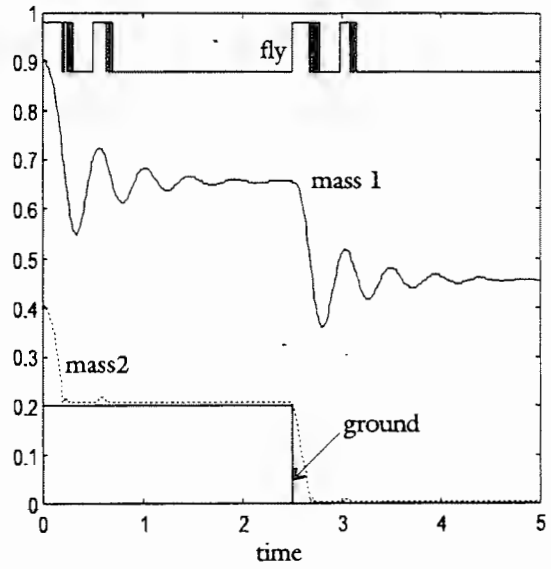


Figure 6: 'Jumping One-Wheel' going downstairs

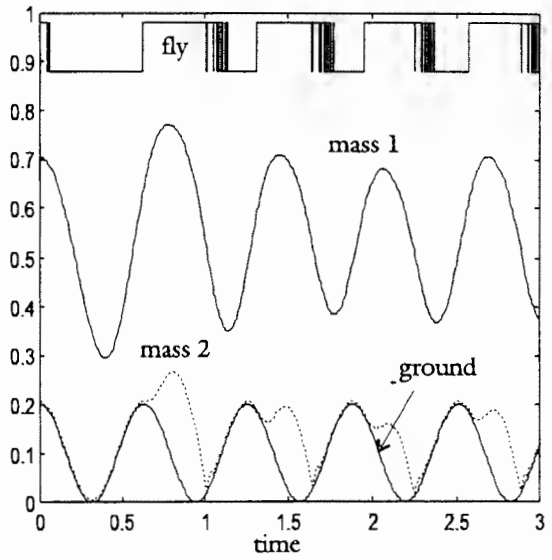


Figure 7: 'Jumping One-Wheel' driving on a 'sinusoidal gravel path'

```
1   BLOCK JFEDMAS;
2   $ BLOCKDIM -1 -1 6 16
3   $ TEXT "One Wheel 1"
4   $ TEXTALIG 1 1 1 1
5   $ ICONNAME "*** NO PIC ***"
6   $ ICONDIM 4 4
7   $ ICONALIG 1 1 1
8   INPUT
9      xg "[m] position of ground";
10     vg "[m/s] velocity of ground";
11  OUTPUT
12     X1 "[m] position mass 1";
13     X2 "[m] position mass 2";
14  PARAMETER
15     Ax1 = 0.705 "[m] X1(0)";
16     Ax2 = 0.205 "[m] X2(0)";
17     m1 = 50 "[kg] = upper mass 1";
18     m2 = 10 "[kg] lower mass 2";
19     k = 5e3 "[N/m] spring between";
20     d = 200 "[Ns/m] spring damping";
21     l = 0.5 "[m] spring length";
22     kg = 3000e3 "[N/m] spring to ground";
23     dg = 500 "[Ns/m] spring - ground damping";
24     r = 0.005 "[m] c.o.grav. mass 2 - ground";
25     g = 9.81 "[m/s^2] acc. due to gravity";
26  STATE
27     x1 "position mass 1";
28     v1 "velocity mass 1";
29     x2 "position mass 2";
30     v2 "velocity mass 2";
31  VAR
32     dist; a1; a2; a3; a11; a21; f;
33     state DISCRETE "ground - air";
34     fly DISCRETE;
35  SIM
36     x1 .= v1;
37     x2 .= v2;
38     a1 = -g -d/m1*(v1-v2);
39     a2 = -g +d/m2*(v1-v2);
40     dist = x1 - x2 - l;
41     a11 = k/m1*dist;
42     a21 = k/m2*dist;
43     v1 .= a1 -a11;
44     v2 .= a2 +a11 + f/m2;
45     SWITCH state
46     CASE 0:
47        // standing on the ground
48        f = kg*(r-x2+xg) + dg*(vg-v2);
49        ONRISE x2-xg > r DO jump;
50     CASE 1:
51        // up in the air
52        f = 0;
53        ONRISE x2-xg < r DO stand;
54     ENDSWITCH
55     X1 = x1;
56     X2 = x2;
57  ENDSIM
58
59  PROCEDURE jump;
60     state := 1;
61     fly := 0.98
62     STORE ALL;
63  ENDPROCEDURE
64
65  PROCEDURE stand;
66     state := 0;
67     fly := 0.88;
68     STORE ALL;
69  ENDPROCEDURE
70
71  INIT
72     state := 0;
73     fly := 0.88;
74     x1 := Ax1;
75     x2 := Ax2;
76  ENDINIT
77  ENDBLOCK JFEDMAS;
```

Figure 8: Blockscript

Intelligent Simulation Interface for LATISS Simulation System

Yuri Merkuryev, Galina Merkuryeva, Ainars Mazversitis, Agris Goldmanis
Dept. of Modelling and Simulation, Riga Technical University
1, Kalku Street, LV-1658 Riga, Latvia
E-mail: merkur@itl.rtu.lv

The paper discusses design of the Intelligent Simulation Interface (ISI), being used within a user-friendly intelligent discrete-event simulation system LATISS (LATvian Intelligent Simulation System). LATISS is a version of the generic interactive system for modelling and simulation GISMOS (Vangheluwe et.al., 1994). It is based on the same approach, when knowledge regarding a system under simulation is encapsulated in models. These models are organized in accordance with the MSL (Model Specification Language) formalisms. The main difference between these simulation systems is that in GISMOS all knowledge about the simulation process itself is also represented in the form of models. On the contrary, in LATISS only a part of knowledge about the simulation process is encapsulated in models (namely, knowledge about optimization algorithms to tune parameters of simulation models). The simulation process here is controlled by ISI, which supports intelligent abilities of the simulation system as well. Another essential part of LATISS is an object-oriented model base, where all models are kept. Simulation models are developed in the HGPSS simulation language, for both UNIX and LINUX platforms.

1. Abilities of simulation interface and its programming

The Intelligent Simulation Interface controls all processes in LATISS. Following are main functions, supported by ISI:

1. **The control function.** ISI supports control of the simulation process and also supports decision making, necessary to realize stages of the simulation process, e.g.

- **control of the simulation process**, aiming to achieve goals of simulation, specified by the user (e.g., analysis of a system under simulation, sensitivity analysis, optimization of system parameters). It is based on schemes of the simulation procedure, corresponding to various goals of simulation;
- **support of decision making,** accompanying realization of the simulation procedure. It is aimed to ensure a possibility to perform most of simulation stages automatically, thus not asking from the user for a deep knowledge in the field of the simulation theory. As a result, LATISS could be used by domain specialists, who are experts in their professional areas, but are not experienced in simulation studies. In order to support necessary statistical procedures, an interface with standard statistical software tools is foreseen. It is aimed to support such stages as analysis of input data (analysis of outliers, design of histograms, testing for fitness to various probability distributions), strategic design of simulation experiments (fractional factorial design, Plackett-Burman design), tactical design of simulation experiments (correlation analysis, evaluation of the necessary number of simulation runs), analysis of simulation results (design of confidence intervals, comparison of outputs from alternative designs). A possibility for the user to change the decisions, made by ISI, is provided as well.

2. **The advising function.** ISI supports the user in making decisions, concerning "manual" implementation of stages of the simulation procedure (for example, when implementing actions, which could not be performed automatically). It provides the user with necessary information, thus

advising him in making decisions. For instance, ISI provides the user with recommendations on use of various optimization algorithms in different situations.

3. **The explanation function.** The user can turn to ISI, asking it to explain him its decisions and recommendations. For instance, if ISI recommends to use some already existing models, which are kept in the model base, in order to model parts of the system to be simulated, the user can ask it to ground these recommendations.

4. **The learning function.** ISI is capable to learn from its own experience, generating new knowledge - corresponding facts and rules to be used for future decision makings. For instance, it collects information about effectiveness of optimization algorithms in various situations; when some of already observed situations occur, those algorithms, which have been found as the most efficient ones, are recommended.

All these functions are based on manipulations with knowledge, which include acquisition of knowledge, its normalization and application, and generation of new knowledge as well. Knowledge in ISI is organized in the following ways:

- **facts** which describe properties of objects, ISI is dealing with (e.g., properties of various algorithms to design simulation experiments: full factorial design, fractional factorial design and Plackett-Burman design);
- **rules** which describe rules, related to use of these objects (e.g., when to use each of the algorithms);
- **metarules** which are rules of a higher level, that can change existing rules or use them in order to develop new facts and rules;
- **comments** which are knowledge that can be treated by ISI and presented to the user in the form of comments.

Knowledge about MSL models is incorporated in LATISS also in the model base, in the form of its *Rule* elements. ISI manipulates with knowledge by means of **tools** which are procedures, that deal with facts and rules. Each tool consists of a PROLOG part that treats facts and rules, and of a C++ part that implements rules. Graphical ISI abilities are developed with C++ language using MOTIF library.

Currently, the research prototype of ISI is available. Its operation is illustrated below for the case study, when a manufacturing system with 4 working stations and a transport robot is simulated with a goal to find an optimal capacity for each working station.

2. Examples of ISI interactions

Building of a model is supported mainly by three windows illustrated in fig. 1. The first window is aimed to select the useful atomic models. A corresponding Class name, Application area and a Filter may be used to simplify the search procedure. Selected atomic models are presented with icons in the second window. It defines the model's structure and is used to form the coupled model's USE part. Working algorithm describes the sequence of atomic models or their interaction. This information is used to formalize the DYNAMIC part of the coupled model. Building of new atomic models and their icons is supported by ISI. Additional logical models are introduced to describe complex working algorithms. Animation of corresponding manufacturing process is available in the middle window. ISI realizes also the following intelligent functions concerning the model base:

- design of requests in accordance with a current stage of the simulation procedure and existing restrictions. As a result of that requests, corresponding models are found in the model base (e.g., models of separate parts of the system under simulation) and used to make the overall model of the system, and to run simulation;
- optimization of the model base (e.g., extracting unfitted models).

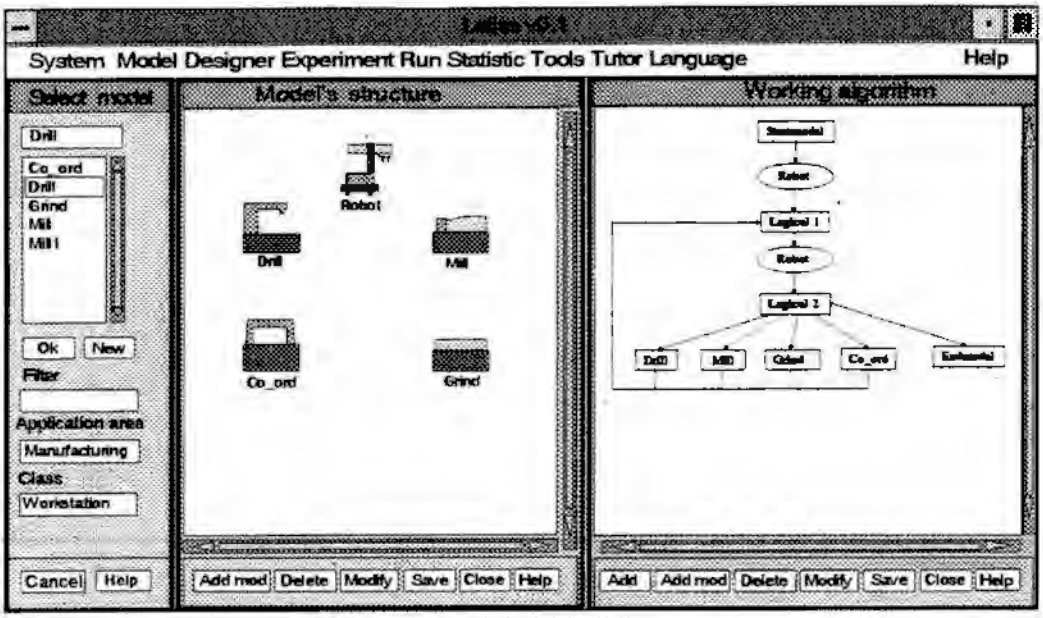


Fig. 1. Model building

ISI supports **specification of several simulation goals.** They are: G1- Process evaluation, G2- What If Analysis, G3- Sensitivity Analysis, G4- Bottleneck Analysis, G5- Comparison of alternatives, G6- Optimization of parameters, G7- Prediction, G8- Metamodelling, G9- Simulation Training. The goal of simulation may be specified by the user directly. Short description of each goal is provided. A special Goal Specification Inference procedure is developed as well. It is based on specification of each goal by a set of attributes. For example, the following G6 specification is used: a performance function exists and should be optimized, the corresponding model parameters should be founded. Goal specification by ISI means is illustrated in fig. 2,3.

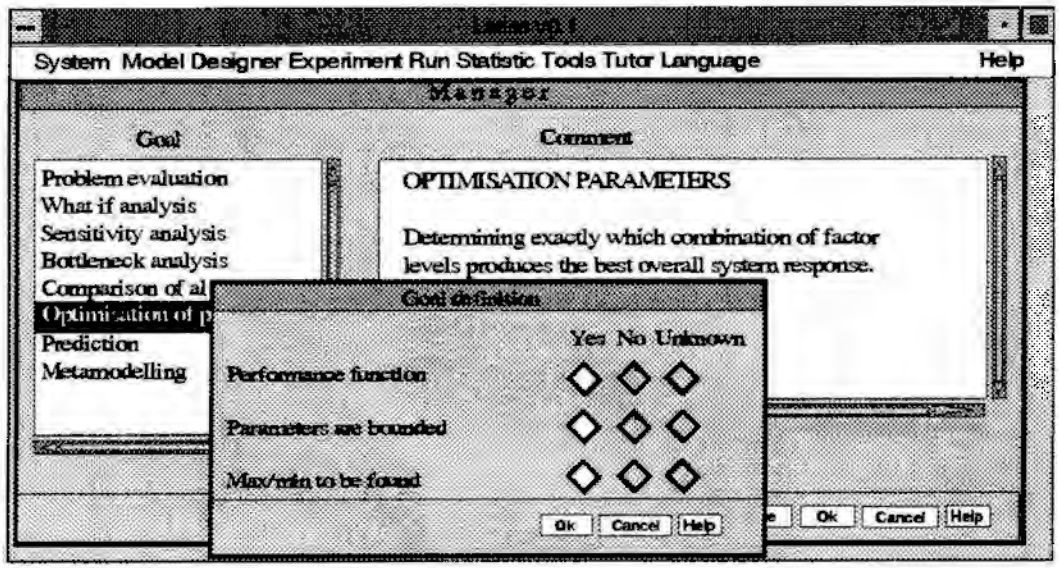


Fig. 2. Goal selection

Once the goal G6 is selected, the parameters of the model should be optimized. ISI incorporates knowledge and experience about 14 optimizers and their applications. The user initializes the characteristics of the model, e.g. number of parameters, their nature, etc. (see fig. 4). ISI makes selection based on knowledge inference, gets an available optimizer and presents explanations. The tool which realizes the corresponding optimizer is activated. Additional decision making procedure is provided if several optimizers are available.

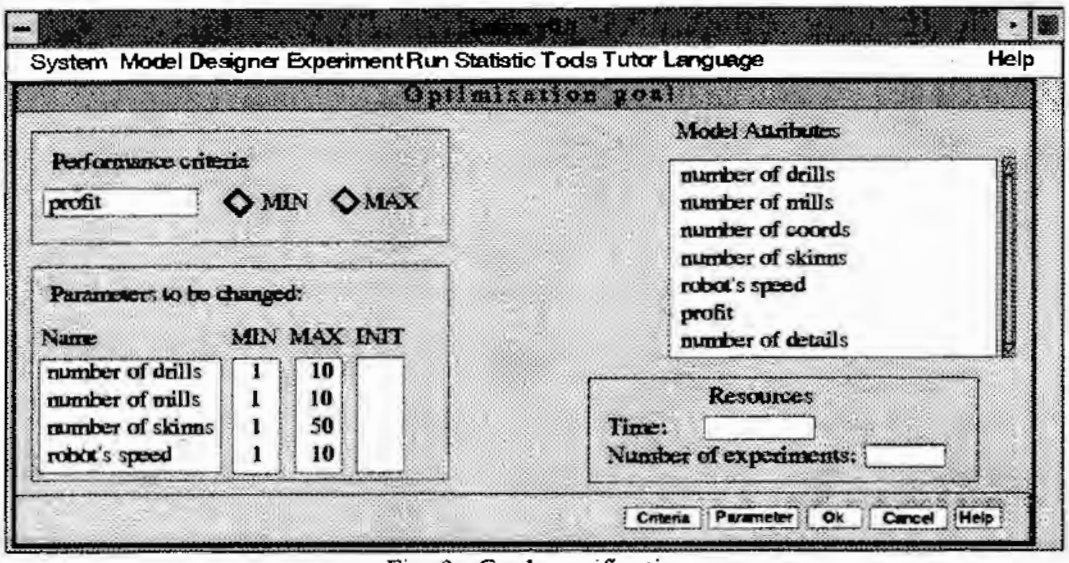


Fig. 3. Goal specification

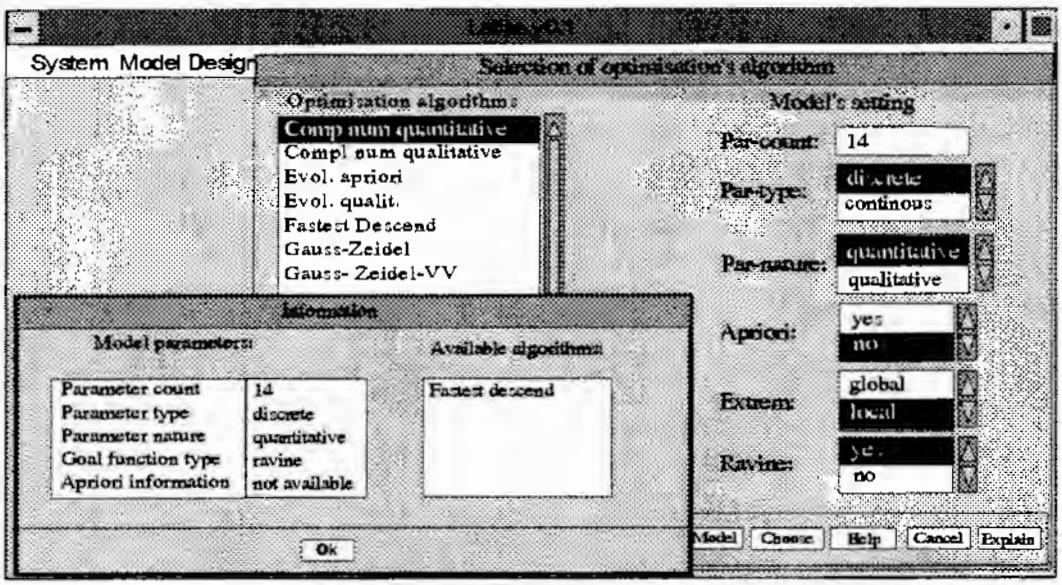


Fig. 4. Selection of optimization algorithm

The above described simulation interface incorporates knowledge and experiences. It is aimed to support intelligent abilities of the LATISS simulation system, e.g. goal searching, explaining and learning. By means of the ISI interface, LATISS operates in different ways adapted to the type of the user and the level of his experience. The non-experienced in simulation user will prefer to perform the most of simulation stages automatically. The most experienced one will prefer ''manual'' implementation of the simulation procedure and will ask for advises and recommendations in necessary cases. TUTOR tools to improve the own abilities of users of the LATISS simulation system are supposed to be developed as well.

H.Vangheluwe, G.Vansteenkiste, V.Visipkov, Y.Merkuryev, G.Merkuryeva and A.Teilans. (1994). Design of a user friendly modelling and simulation environment. Proc. of the 1994 European Simulation Multiconference. Ed. by Dr.A.Guasch and Dr.R.M.Huber.

ARCHITECTURE OF A SIMULATOR FOR RESEARCH IN ADVANCED TRAFFIC MANAGEMENT SYSTEM DESIGN

Richard M. Ingle

Georgia Tech Research Institute
Georgia Institute of Technology

ABSTRACT

Real-time traffic management is a key part of the overall vision of a future Intelligent Vehicle-Highway System (IVHS). The rapidly approaching IVHS era brings many technical challenges and questions related to the design, implementation, operation and maintenance of future Advanced Traffic Management Systems (ATMSs). This paper provides an overview of an Advanced Traffic Management System/Center Simulator developed at Georgia Tech and currently being used as a research tool to help answer questions that arise as IVHS technology makes its way into tomorrow's Advanced Traffic Management Systems. The architecture, technical challenges and fundamental implementation decisions are summarized. Brief descriptions of the traffic model, database, displays and associated simulated traffic video coverage are provided.

1 INTRODUCTION

The IVHS era brings high technology to the traffic management arena. For example, larger numbers of more capable sensors, automated control devices, automated monitoring devices, automated dispatching systems, advanced information systems, predictive traffic modeling systems, and various other support systems will incrementally find their way into Traffic Management Systems (Vostrez et al. 1992). The rapidly approaching IVHS era also brings many technical challenges and questions related to the design, implementation, operation and maintenance of future Advanced Traffic Management Systems. In order to start addressing many of the issues related to future IVHS-era ATMSs, simulation of the full ATMS environment is indicated. Simulation can help answer questions related to types of sensors needed, appropriate sensor coverage, design of automated and semi-automated support systems, appropriate degree of automation, and various human factors issues associated with vehicular, highway and traffic management center (TMC) systems. In order to meet this need, especially for the case of research into human factors issues associated with the design of future traffic management centers, we have developed a TMC simulator and are currently using it to conduct an intensive program of such research. As the research program continues, the capability of the TMC simulator is continuing to evolve.

This paper presents an overview of the hardware and software architecture of the TMC simulator, a brief discussion of some of the technical challenges addressed, a survey of the implementation decisions made on the basis of an extensive trade study, and a summary of initial performance observations.

2 TECHNICAL CHALLENGES

Some of the major technical challenges encountered in the development of the TMC Simulator included satisfaction of the requirements for rapid reconfigurability, real-time operation, and responsive CCTV simulation. Some of the details related to meeting these challenges are included in the architecture discussion that follows. Special attention is given to the CCTV system simulation, since this proved to be the most interesting and challenging problem solved.

3 ARCHITECTURE OF THE TMC SIMULATOR

The TMC simulator provides real-time, interactive operation, and is rapidly reconfigurable to allow for a wide variety of user interfaces and displays, ATMS reconfiguration, a variety of control methods, and varying degrees of automation. It provides simulated TMC inputs including traffic and roadway sensors, visual CCTV sensors, probe vehicles, cellular

telephone dialog, voice communication systems, and database services. It provides simulated TMC outputs associated with intersection control devices and algorithms, roadway access devices and control algorithms, variable message signs, highway advisory radio, commercial TV and radio, cable TV traffic channel, traffic bulletin board, and voice communication output. Operator support systems include adaptive traffic control, predictive traffic modeling, incident detection and location, response advisory, and information dissemination systems. The traffic model (AUTOS) was developed at Georgia Tech and is based on the Greenshield's speed-density flow model. The CCTV traffic video simulation (AUTOGRAPH) was also developed at Georgia Tech and is hosted on a Silicon Graphics Onyx / Reality Engine 2 system, making heavy use of the graphics-rendering hardware on this machine.

3.1 TMC SIMULATOR HARDWARE

After conducting detailed requirements and performance studies, the hardware architecture selected for hosting simulator included Silicon Graphics Indigo2 XZ machines for the four Operators' Workstations, the Experimenter's Workstation, the Traffic Model Server, and the Large Display Server. As discussed earlier, a Silicon Graphics Onyx / Reality Engine 2 was used as host machine for the simulated traffic video system. An additional bank of video monitors, whose configuration varies with experiments, uses Sony Trinitron PVM-1350 color video monitors. A Barco Retro Graphics 801 serves as Large Display and a plethora of video switching and converting devices are used in the implementation of a flexible video assignment and display system. The Operators' and Experimenter's Stations are rounded out with 486 PCs equipped with resistive membrane touchscreens which are configured as control panels and displays. These are generally associated with communications interfaces. An audio communications network with wireless headsets, Metamorphosis adjustable desks and Hag ergonomic task chairs complete the hardware used to implement the simulator.

3.2 TMC SIMULATOR SOFTWARE

Rapid software reconfigurability, including user interfaces, databases, displays, control devices, etc., was achieved by use of a unified TMC database and related control processes to drive the traffic model, the video simulator and all operator displays, and support system inputs and outputs provided the setting for achieving flexible software reconfigurability. The development of a script language, based on the public domain tool command language (Tcl) from the Department of Electrical Engineering and Computer Science at the University of California - Berkeley, for use in prescribing scenarios and configurations and the use of user interface builder tools, such as VAPS (a Virtual APplicationS development tool sold by Virtual Prototypes, Inc.), contributed heavily to flexible user interface reconfigurability.

An extensive trade study (Ingle et al. 1993) was done early in the simulator development program to locate and evaluate available commercial and government hardware and software for possible use in meeting various requirements of the simulator. None of the existing commercial or government traffic models met the interaction and real-time operation requirements, let alone ease of modification to allow for simulated support systems, etc. We opted, therefore, to start from an existing Georgia Tech traffic model called TERMINUS (Gilmore et al. 1992). Starting from this model, we created a much larger capacity, versatile model with hooks in place for the needed support systems. The resulting traffic model, called AUTOS, is a macroscopic flow model capable of running 5-10 times faster than realtime on a 5000-link network and 50 times faster than realtime on a 1700-link network. This performance permits us to use a copy of the model to achieve the simulated predictive traffic management support system. The AUTOS model is implemented using C++ object-oriented design. It is driven by TMCS database map data and traffic specifications. AUTOS (Gilmore et al. 1994) provides a macroscopic model of traffic flow using Greenshield's speed-density flow model:

$$u = u_f(1 - k/k_j)$$

$$q = uk,$$

where q is flow, u is speed, k is density, u_f is free flow speed and k_j is the jam density. AUTOS models traffic flow on both freeways and surface streets and allows for various types of intersection control (uncontrolled, signed, fixed signal, actuated signal, fixed meter, actuated meter), signal phases, turning percentages, delays and various parameters of link control (directionality, number of lanes, length, free-flow speed, road condition factor, initial load, load deltas for rush hour build-up/down, current and maximum vehicle counts, etc.). In addition, parking lots, high occupancy vehicle lanes, reversible lanes and turn lanes are modeled.

Perhaps the biggest technical challenge was achieving a responsive simulated traffic surveillance video capability. Since our interviews with traffic management experts all over the country consistently underscored the importance of traffic surveillance video in present and future TMCs, we elected to implement high fidelity traffic video simulation using computer animation. The traffic video simulation was implemented by taking 360° photographs at 38 key traffic locations in Atlanta. These photographs were digitized, existing traffic was removed, background and foreground images were separated, lane paths were defined, and animated traffic using texture-mapped graphical techniques were applied to produce animated traffic flowing between the background and foreground planes. We were able to achieve a very realistic animation, with densities, speeds and vehicle-type mix controllable by the traffic model and model control processes. Incident scenarios are easily choreographed and emergency vehicles (police, ambulance, tow trucks, etc.) arrive in accordance with dispatching actions. Where other visual confirmations are possible, such as variable message signs, these features are also depicted in the traffic video simulation. Smooth lane changes are also rendered. Some atmospherics are implemented, although this aspect has been at low priority to date. Many sites can be concurrently ready for display at any time and up to four sites can be concurrently rendered for viewing on any number and combination of monitors, video windows, or on the large screen display. The operators have full control of camera selection, 360° pan and zoom (1X to 6X). This traffic video simulation, called AUTOGRAPH, integrated with the traffic model has already attracted attention from traffic engineers, managers and researchers from many countries and has led to requests to extend the capability to include additional features, such as tunnels and bridges. Figure 1 provides a top-level depiction of the software architecture.

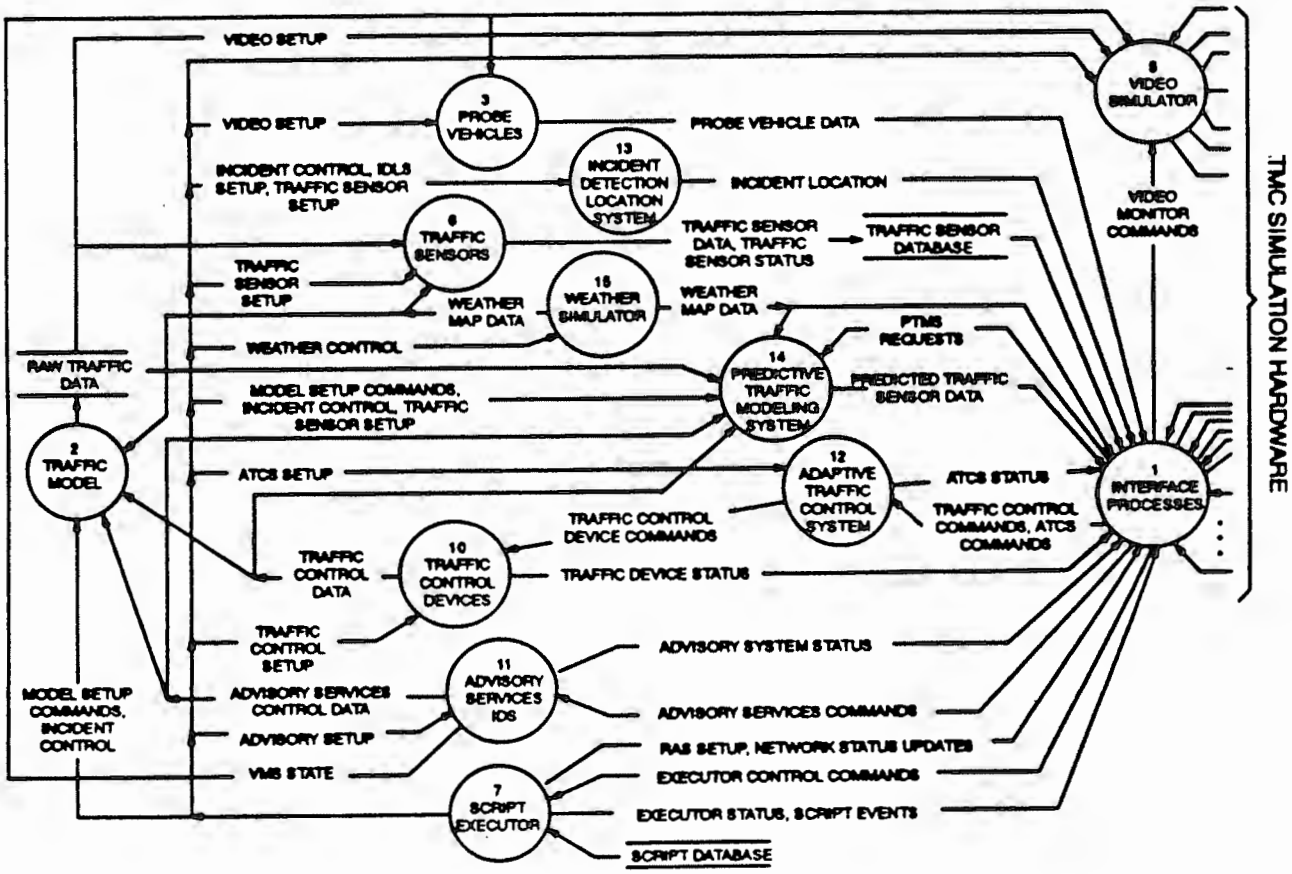


Figure 1: TMC Simulator Software Interfaces

4 SUMMARY AND RESULTS

A very capable ATMS Simulator has been implemented and is currently being used in a research program investigating human factors issues in the design of future TMCs. Simulation capabilities will continue to evolve over the next year and the simulator should be useful for a wide range of ATMS research and training purposes. The traffic model, AUTOS, is capable of running much faster than realtime for network sizes up to 5000 links. A sophisticated Traffic Video Simulator, AUTOGRAPH, is integrated with AUTOS and produces very realistic simulated traffic video animations. AUTOGRAPH taxes the current state of the art in graphical simulation to render four concurrent video simulations. It is anticipated that the next generation in Silicon Graphics compute power will allow this limit to expand.

ACKNOWLEDGMENTS

This work is sponsored by the Federal Highway Administration under Contract DTFH61-92-C-00094. Ms. Nazemeh Sobhi is the FHWA Contracting Officer's Technical Representative. Dr. Richard Ingle is director of the simulator design and development effort. Dr. Michael Kelly is principal investigator for the program of human factors research being performed using the traffic simulator as a tool. *Notice -- This document is disseminated under sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its content or use thereof. The United States Government does not endorse products or manufacturers. Trademarks or manufacturer's names appear herein only because they are considered essential to the object of the document. The contents of this report reflect the views of the author and project team who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official policy of the Department of Transportation*

REFERENCES

Gilmore, J. F., K. J. Elibiary, and H. Forbes. 1994. Intelligent Control in Traffic Management. To appear in *Proceedings of the International Conference on Vehicle Navigation and Information Systems*. August 31 - September 3, 1994 Nagoya, Japan.

Gilmore, J. F., K. J. Elibiary, and R. J. Peterson. 1992. A Neural Network Approach to Special Event Traffic Management In *Proceedings of the National Conference on Advanced Technologies in Public Transportation*. Transportation Research Board, San Francisco, California.

Ingle, R. M., S. M. Blankenship, D. J. Folds, J. M. Gerth, M. J. Kelly, D. R. Stocks, P. D. West, C. J. Whaley, and B D. Williams. 1993. Traffic Management Center Simulator Trade Study. Technical report on contract DTF61-92-C-00094, Georgia Tech Research Institute, Georgia Institute of Technology, Atlanta, Georgia.

Ingle, R. M., N. Sobhi, and B. D. Williams. 1994. An Advanced Traffic Management System Simulator for Intelligent Vehicle-Highway Systems Research. In *Proceedings of the 1994 Winter Simulation Conference*. December 11-14, 1994. Orlando, Florida. pp. 1455-1460.

Vostrez, J., J. Arlook, E. S. Greene, D. C. Roberts, M. G. Sheldrick, and J. Sussman. 1992. *Strategic Plan for Intelligent Vehicle-Highway Systems in the United States*. Washington, D. C.: Intelligent Vehicle-Highway Society of America.

AUTHOR BIOGRAPHY

RICHARD M. INGLE is a Principal Research Scientist and head of the Systems Technology Branch of the Georgia Tech Research Institute's Electronic Systems Laboratory. He received a B.A. degree in mathematics from West Georgia College in 1967, and M.S. and Ph.D. degrees in applied mathematics from the Georgia Institute of Technology in 1969 and 1980 respectively. His research interests lie in computational aspects of modeling and simulation, and applied mathematical analysis. His career has intertwined teaching with applied research. He has also developed various other simulation and display systems in current use by the USAF and NASA.

MAPLIS - Simulation with Work Sheets in Hyper Space

Wilfried Tettweiler, Krailling

Summary: Recent releases of Lotus 1-2-3, QuattroPro for Windows and MS-Excel have shown considerable enhancements of these products, supporting three-dimensional spread sheets, "What-when"-analysis and spread sheet simulations etc.. New elements in the MAPLIS PC release reflect these developments by means of interfaces to access data of different file formats for popular PC software, supporting multi-dimensional views on the data contained, analysis, consolidation, aggregation (sums) and application of formulas to this data. Formulas and constant data from work sheets can be converted into MAPLIS-language models. Multi-dimensional computations can be vectorized to a high degree due to the tensor concept of MAPLIS. This permits high performance computing with a vector processor on an upgraded personal computer.

1 Introduction

Once more it is questionable if simulation with spread sheets means the birth of a good idea or the death of a bad one. Since Microsoft's Excel introducing "what-when"-analysis created innumerable spread sheet models, there is growing concern about the missing clock mechanism and the "intuitive" use of the Solver to manipulate categorical data. For example: interest rates (no bank pays 6.32789% interest on your money) or numbers of children (do you have 0.5 children?). The methodical search for a solution using MAPLIS language which is based on a tensor concept to store multidimensional objects in the model's data base aims at avoiding that sort of conflict resulting from mere computing with scalar data. The following printout shows the first part of the contents of a MAPLIS variable named EAGNWDC with six dimensions ZELLE, GEBALT, WGN, NATION, ALTER and GESCHL:

```
(ZELLE     )
(GEBALT    )
(WGN       )
(NATION    )
         --> ALTER
     |   +------------------+
     |   | EAGNWDC          |
     V   +------------------+
    GESCHL

(1    )
(B8889     )
(EINFM     )
(DEUTSCH)                        (AUSLAND)

         1          2                    1          2
        +----------+----------+          +----------+----------+
MAENNL  |          1          2|  MAENNL  |          5          6|
WEIBL   |          3          4|  WEIBL   |          7          8|
        +----------+----------+          +----------+----------+

(1    )
(B8889     )
(GESCH     )
(DEUTSCH)                        (AUSLAND)
         1          2                    1          2
        +----------+----------+          +----------+----------+
MAENNL  |          9         10|  MAENNL  |         13         14|
WEIBL   |         11          0|  WEIBL   |         15         16|
        +----------+----------+          +----------+----------+
```

Header of the Printout with names of the dimensions used

First two values for the dimension WGN

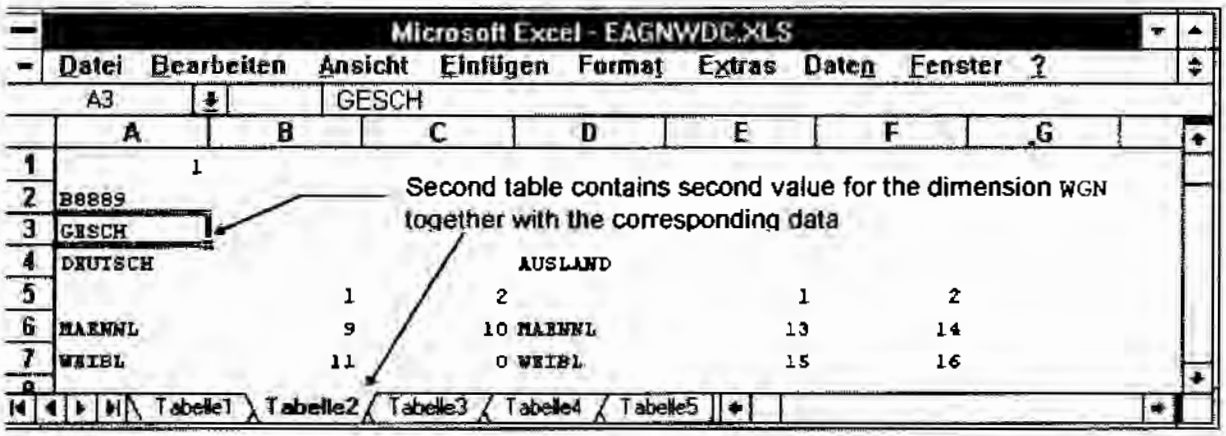
Furthermore MAPLIS gets welcome help from newly emerging trends in the development of spread sheet programs.

2 New characteristics of standard spread sheet programs

Recent releases of Lotus 1-2-3, QuattroPro 5.0 for Windows and MS-Excel have shown considerable enhancements of these products.

2.1 Supporting three-dimensional spread sheets

Seeing multi-dimensional tables being reflected as a sequence of two-dimensional tables is routine for users who are familiar with standard statistical packages like SPSS etc. Likely, one can prepare multi-dimensional data in appropriate sets of spread sheets. Any practical minded user can easily understand and handle those, because they meet his view of data categorially rather than continuously. The following example corresponds to the MAPLIS variable shown above, the upper left corner of each worksheet contains the respective value settings for the first three dimensions ZELLE, GEBALT and WGN (the actual value of WGN is GESCH):



Each cell B6..C7 and E6..F7 contain frequencies of units with idetical properties. Cells A1..A7, D4..D7 and B5..F5 define values for the corresponding properties.

Selective printout of a MAPLIS variable with any value combination (as wanted, not as predefined by the order or structure of the spreadsheet) serve to analyse and consolidate data. Aggregation level changements eliminate (or exchange) dimensions by computing row or column sums or vice versa add dimensions by distributing data according to given probabilities.

In addition a sequence of spread sheets is the most natural way to reflect also periodical alterations of data. The clock mechanism of MAPLIS is as period-oriented as DYNAMO, its predefined variable TIME allows for the saving of sets of variables according to their occurence in time in sets of spreadsheets. Support of three-dimensional spreadsheets in standard spreadsheet packages provides easy access to variable's contents being reported in the form of a time series.

2.2 "What-when"-analysis and spread sheet simulations

New functions as e.g. scenario-manager, if-then-else-analysis and break-even-analysis etc. have set standards for modelling not only in the area of financial calculation or business management. The problem connected is that the search for the optimum may bring forth only relative optima or even worse, rather unrealistic results (see introduction). MAPLIS-language performs strict compatibility checking to identify and avoid problems undetected in mere spreadsheet computing, e.g. multiplication of interest rates with number of houses.

3 New elements in the MAPLIS release for personal computers

MAPLIS can offer solutions that are adequate to the problems mentioned above.

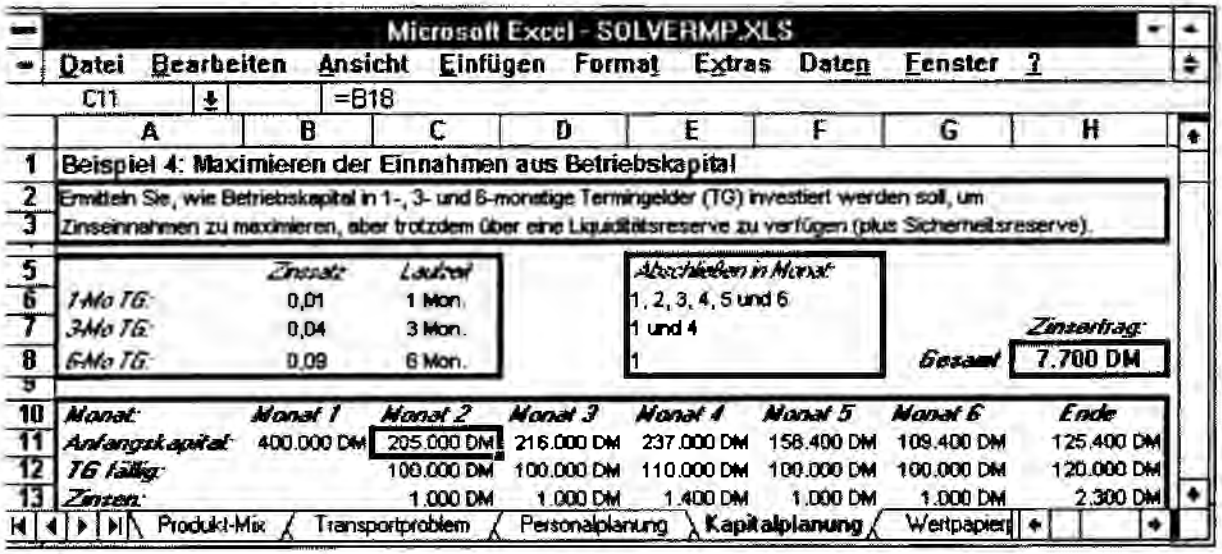
3.1 Interfaces within MAPLIS to process standard data formats

The MAPLIS command `INOUT_MEDIUM` now accepts de facto standards like ODBC and IDAPI to process data immediately from data bases and spread sheets. This allows for the collection of data for processing even from quite different sources. This data can then be examined and connected logically or arithmetically under any aspect within the multi-dimensional concept of MAPLIS.

3.2 Integrating spread sheet models into MAPLIS

The formulas and constant data from work sheets can be converted into MAPLIS-language models either directly or through filter programs.

The following Excel Solver spreadsheet



is translated into a model in MAPLIS language as follows:

```
add_model solvermp                          <-- typical definition of a scalar dimension
n_of_values EINS (1)
-- A1: 'Beispiel 4: Maximieren der Einnahmen aus Betriebskapital
-- A2: 'Ermitteln Sie, wie Betriebskapital in 1-, 3- und 6-monatige Terminge
-- A3: 'Zinseinnahmen zu maximieren, aber trotzdem über eine Liquiditätsrese
-- B5: ^Zinssatz
-- C5: ^Laufzeit
-- E5: 'Abschließen in Monat:          <-- Strings are translated to comments
-- A6: +"1-Mo TG:"
-- B6: 0.01
...
-- G10: 'Monat 6
-- A11: 'Anfangskapital:
add_variable   B11 (EINS)               <-- Declaration and initialization of the
print_formats  B11 (16.9)                   variable corresponding to cell B11
let            B11 = 400000
add_variable   C11 (EINS)               <-- Declaration and computation of the variable
print_formats  C11 (16.9)                   corresponding to cell C11
compute        C11 = +B18
...
end_model
```

The same work sheet can contain the model as well as the data related to it. It is thus possible to perform a plausibility test on a model first by means of the spread sheet program based on scalar data. One can then continue processing multi-dimensional data with the corresponding MAPLIS-language based task after having changed the MAPLIS variable definitions to multiple dimensioning, e.g. . `add_variable EAGNWDC (ZELLE by GEBALT by WGN by NATION by ALTER by GESCHL by TIME)` according to the example above.

3.3 Benefits using a vector processor

Computations can be vectorized to a high degree due to the tensor concept of MAPLIS. This permits high performance computing with a vector processor on an upgraded personal computer.

4 State of development and preview

The MAPLIS-language was first published 1982 in german language in the proceedings of the "1. Symposium Simulationstechnik" (1st Symposium for Simulation Technique, Informatik-Fachberichte 56, Springer 1982) and one year later in english language in the proceedings of the "Mathematical Modelling in Science and Technology (Pergamon Press 1983). Various realization platforms since that time were main frames (IBM, Cyber and CRAY). A number of applications based on MAPLIS models are in the range from population development prognosis to investment planning.

A personal computer DOS version of MAPLIS has been in use since the beginning of 1994, a Windows user interface is in work. All parts of MAPLIS-computations, which can be vectorized or parallelized respectively, will soon be speeded up by a ZORAN vector processor and by means of INMOS transputers.

Wilfried Tettweiler, Brahmsstr. 12, D-82152 Krailling
Tel.: +49-89-8573777 Fax: 8572183
E-Mail: 100421.2304@CompuServe.com

VIBRATORY ANALYSIS OF THIN SHELL STRUCTURES IN MEDIUM AND HIGH FREQUENCY RANGE WITH THE ACTU SOFTWARE

PEYRAUT FRANÇOIS

SCIENCES INDUSTRIES CONSEILS
14, Avenue de Sceaux
78 000 VERSAILLES
FRANCE
Tel. : (1) 39 49 05 27
Fax : (1) 39 02 77 23

1 - INTRODUCTION

This paper presents the ACTU software which is devoted to structural vibrations and sound emission analysis outside thin shells structures. The sound can be generated by rotating machine's vibrations (pump, compressor, motor ...). These machines create periodic motions which propagate as sound waves in interior and exterior fluids and as vibratoring waves in the shells.

The quasi-analytic method used in ACTU allows low, medium and high frequency's computation. In addition, this method don't impose any restrictions on boundary conditions.

At the present time, ACTU works in vibratory field for cylindrical shells and axisymetrical loading. All the boundary conditions are allowed.

ACTU has been developed in fortran 77.

2 - THEORY

ACTU applies to thin shells. Therefore, we can use the Kirchhoff-Love's theory in which a little piece of transverse matter is considered as a rigid body. Thus we neglect normal striction and transverse shear strains ([4]).

With cylindrical shells and axisymetrical loading, the longitudinal displacement u, the azimuthal displacement v and the radial displacement w are solutions of the following partial derivatives equations :

$$\delta^2 u/\delta z^2 + k^2 u + \mu/R\ \delta w/\delta z = 0 \qquad (1)$$

$$(1-\mu)/2\ \delta^2 v/\delta z^2 + k^2 v = 0 \qquad (2)$$

$$\mu/R\ \delta w/\delta z + h^2/12\ \delta^4 w/\delta z^4 + (k^2 + 1/R^2) w + \mu/R\ \delta u/\delta z = \alpha\ \delta p \qquad (3)$$

δp represents the pression applied to the shells and z the longitudinal variable along the cylinder's axis. R is the radius and h the thickness. We have :

$$k = \Omega[m_v(1-\mu^2)/E]^{1/2} \quad (4)$$

with k the wave number, Ω the circular frequency, m_v the density, μ the Poisson's coefficient and E the Young's modulus.

The coefficient α is defined by :

$$\alpha = (1-\mu^2)/(Eh) \quad (5)$$

The longitudinal and radial equations (1) and (3), dependant on each others by the terms $\delta w/\delta z$ and $\delta u/\delta z$, are solved by an analytical Fourier series's expansion and a plane wave's approach ([2] and [3]). In that case, the solutions of (1) and (3) are completely known except for the modal amplitudes. Consequently it's necessary to solve numerically the modal amplitude's linear system.

Moreover our method allows all boundary conditions in vibratory field.

3 - VALIDATION

ACTU has been correlated with a completely analytical method. This method assumes that longitudinal and radial vibrations are independant ([1]). On that hypothesis, the longitudinal resonant frequencies are given by :

$$f_k = k[E/m_v]^{1/2}/(2L) \quad (6)$$

where L is the cylinder's length.

Consider a cylinder with :

$$E=2.1E11\ N/m^2;\ m_v=7800\ kg/m^3;\ L=1\ m;\ R=0.25\ m;\ h=0.01\ m;\ \mu=0.285$$

By using equation (6) the third longitudinal resonant frequencies are :

$$f = 2594,\ 5188,\ 7782\ Hz \quad (7)$$

The comparison beetween (7) and ACTU's results is shown in table 1. The discrepansies don't exceed 3.9 %. Hence ACTU is valided in vibratory field.

Frequency (Hz)			
Analytic	2594	5188	7782
ACTU	2646	5363	8087
error	2 %	3.4 %	3.9 %

Table 1

In the same way, we have considered a cylinder with :

$$E=2.1E11\ N/m^2;\ m_v=7800\ kg/m^3;\ L=80\ m;\ R=0.25\ m;\ h=0.01\ m;\ \mu=0.285$$

By using again equation (6), we can easily find the fifth longitudinal resonant frequencies:

$$f = 32.4, 64.9, 97.3, 129.7, 162.1\ Hz$$

On figure 1, we have plotted the cylinder's dynamic response against frequency. It may be seen that analytical frequencies correspond exactly with ACTU computation.

A key has been put in ACTU for the amplitude's correlation. This key allows us to have input acceleration to zero. We can thus compare displacements and rotation computed by ACTU and statics calculation made with the finite element software I-DEAS. We have considered a cylinder with :

$$E=2.1E11\ N/m^2;\ m_v=7800\ kg/m^3;\ L=0.936\ m;\ R=0.25\ m;\ h=0.01\ m;\ \mu=0.285$$

We have assumed free rotation and radial displacement equal to zero at cylinder's ends and we have taken the following longitudinal displacement :

$$u = 0.1\ m \text{ at cylinder's top; } u = -0.1\ m \text{ at cylinder's bottom}$$

Observing figures 2, 3 and 4, it may be seen that displacements and rotation amplitudes are perfectly the same. Therefore, in addition of frequency's calculation, ACTU allows us to make statics computation. That's an important advantage for ACTU's users who want to test the structural resistance of a thin shell assembly.

4 - CONCLUSIONS AND FUTURE PROSPECTS

ACTU is helpful for silent installation's design (choices of material and geometry for instance). In consequence of its quasi-analytic approach, ACTU computes quickly, allowing engineers to work rapidly for a design project. Moreover, this approach is better than finite element for medium and high fréquencies and has no limitations for boundary conditions.

At the present time, a more complicated geometry than cylinder and a general loading are studied with an analytic approach to guarantee the ACTU's fast computation.

Finally, beyond the vibratory study, useful because a good acoustic behaviour needs a good vibratory approach, the acoustic emission's analysis is in progress.

REFERENCES

[1] "Modélisation des coques minces élastiques", Destuynder P., cours de l'Ecole Centrale, 1986/1987

[2] "Elastic Wave Transmission Through Plate/beam Junctions", Langley R.S. and Heron K.H., Journal of Sound and Vibrations, 143(2), p. 241-253, 1990

[3] "Comportement Vibratoire de Réseau de Plaques Assemblées", Rebillard E.et Guyader J.L., Journal de Physique IV, Colloque C5, supplément au Journal de Physique III, Volume 4, P. 121-124, 1994

[4] "Vibrations of Shells and Plates", Soedel W., Ed. Marcel Dekker (New-York), 1981

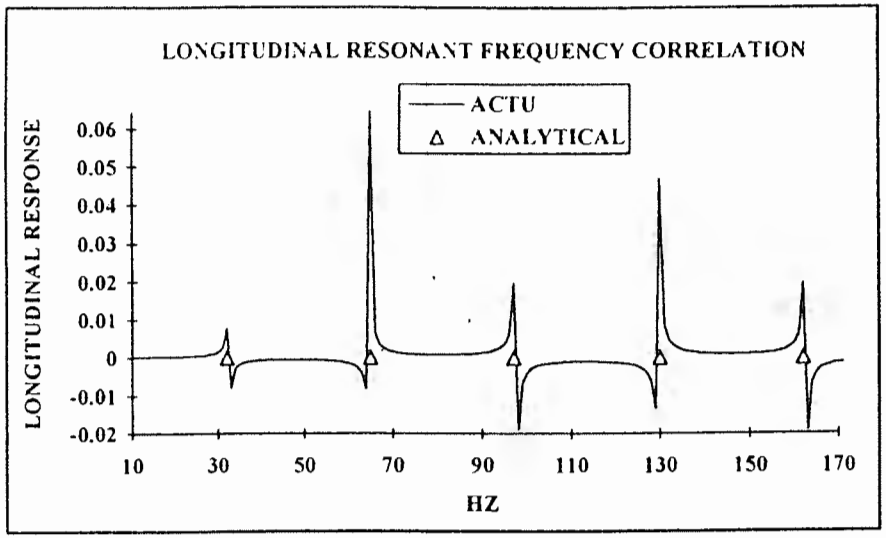


Figure 1

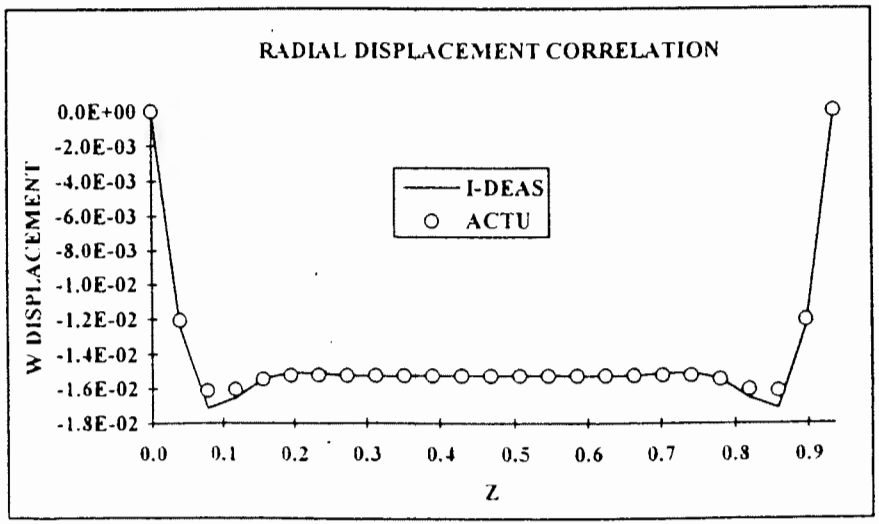


Figure 2

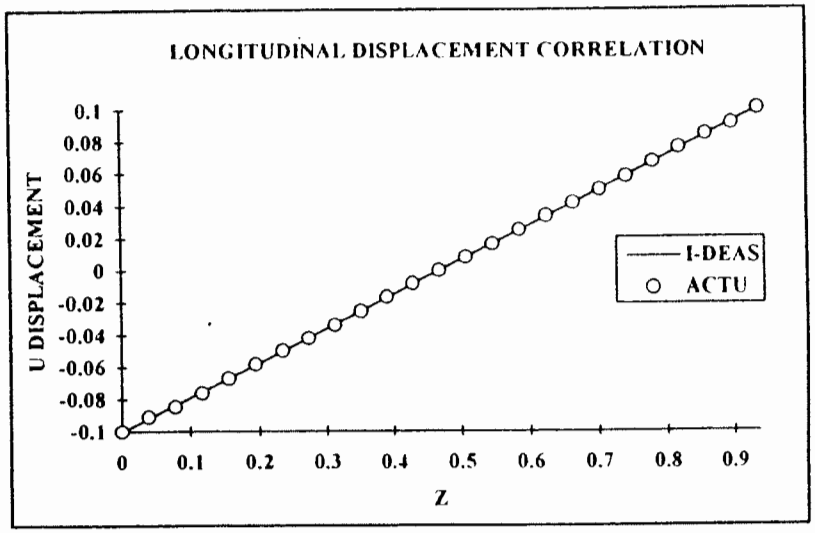


Figure 3

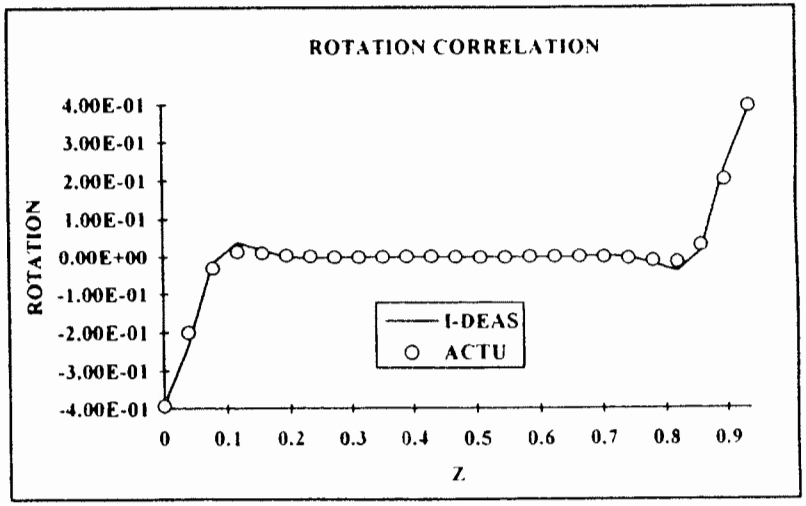


Figure 4

A Multi-Purpose Simulator for an Oil Production Plant

Ommund Øgård[1], Ivar Halvorsen[1], Kåre Telnes[2], and Dag Sjong[2].

ABSTRACT

This paper describes a multi-purpose dynamic simulator for the Heidrun oil production plant. The simulator integrates a commercial dynamic model with the actual process control system to enable dynamic simulation, control system verification and operator training within the same framework and based on the same models and configuration data. The control system configuration can be imported directly into the simulator. This will simplify simulator maintenance and ensure consistency between the simulator and the real plant. The entire system runs on standard Unix work stations. The simulator has so far been used for initial operator training, control system verification and controller tuning.

INTRODUCTION

The usage of dynamic simulation is rising in today's industry. The Norwegian offshore oil production industry uses dynamic simulation for different purposes. Typical examples are: design and verification of control structures, verification of the process design, testing of operational procedures, operator training and for verification of process modifications during the plant's life cycle.

There has so far been a clear distinction between simulators used for different engineering purposes and simulators used for operator training. "Engineering simulators" for the oil production process are based on detailed and accurate multi-component models. Engineering simulators are usually maintained and used during the plants life time. The OTISS simulator from the british company SAST dominates this market segment. Engineering simulators run on standard workstations and are moderately priced.

Training simulators, in contrast, are tailor made to match the training requirements and to mimic the operators' interaction with the control system. The models are traditionally different from those used in the "Engineering simulators", emphasising real time performance and robustness. Special HW might be required and the price is high. The deviating models in "Engineering" and training simulators increase the maintenance costs because two models must be updated when the plant undergoes modifications.

The rest of this paper will describe the HOPE (Heidrun OPerational Experience) simulator. The HOPE simulator integrates an OTISS simulator and Simrads's AIM-1000 process control system to enable operator training, control system verification and engineering studies on the same simulator. The HOPE development project consisted of two parts: the simulator and an operator support system for on-line execution of operational procedures. Woods, Backer, Wahl and Telnes (1994) describe this operator support system in detail.

PROJECT OBJECTIVES

The main objectives for the HOPE simulator are stated below.

1. To develop a multi purpose dynamic simulator for the Heidrun offshore oil production plant that enables both operator training and system verification.
2. The simulator should use the same configuration files as the control system and the Engineering simulator, in this case an AIM-1000 distributed process control system and an OTISS model.
3. The development costs should be lower than for a traditional training simulator.
4. The functionality should be the same as in other training simulators. Both instructor based and self training should be supported.
5. The simulator should run on standard work stations located in a normal office environment both on board and in the operation centre.

The scope of the HOPE simulator did only include the main process on the platform. This includes the systems for oil separation, gas production and produced water handling.

1. SINTEF Automatic Control, N-7034 Trondheim, Norway. (Ommund.Oegaard@regtek.sintef.no)
2. Statoil R&D Centre, N-7005 Trondheim, Norway. (ktel@fou.tr.statoil.no)

The requirements do not state that the training environment should be identical to the control room on the platform. More emphasis was put on making the simulator flexible and facilitate multiple usage. One rational for this is the fact that only experienced operators are recruited to the Heidrun platform. The focus of the training is therefore moved more on accustoming the operators to the process dynamics and the operational procedures, than on training elementary control room functions. A copy of the control room environment was therefore not important as long as the HCI was nearly equal to the HCI in the control room.

The basic training simulator functionality in the HOPE simulator include:

- An instructor Human Computer Interface (HCI) for control and operation of the simulator
- Selection and setting of initial conditions and scenarios
- Selection and setting of disturbances while the simulator is running
- Taking snapshots of the current state and saving initial conditions
- Go to a previous snapshot and replay from snapshots

TECHNICAL SOLUTIONS

The system architecture is illustrated in the data flow diagram in Figur 1. The re-used components are shaded. This includes nearly all of the AIM-1000 system, its internal communication protocol and the OTISS simulator. The most extensive SW components in the HOPE simulator are therefore re-used. Some enhancements had to be made in the basic AIM-1000 software to implement snapshots, replay and scenario functions. These modifications, however, do not interfere with the HOPE simulator's ability to import AIM-1000 configurations directly.

The Maritime Information Technology Standard (MiTS) protocol was used for all inter-process communication in the SW developed in the HOPE project. MiTS defines a application program interface that allows flexible network transparent inter-process communication. MiTS is based on TCP/IP and is implemented on different unix dialects, Windows and several real time operating systems. The HOPE simulator is thus designed to utilize several cooperating computers connected in a standard local area network. MiTS is described more in detail in Rødseth and Haaland (1993).

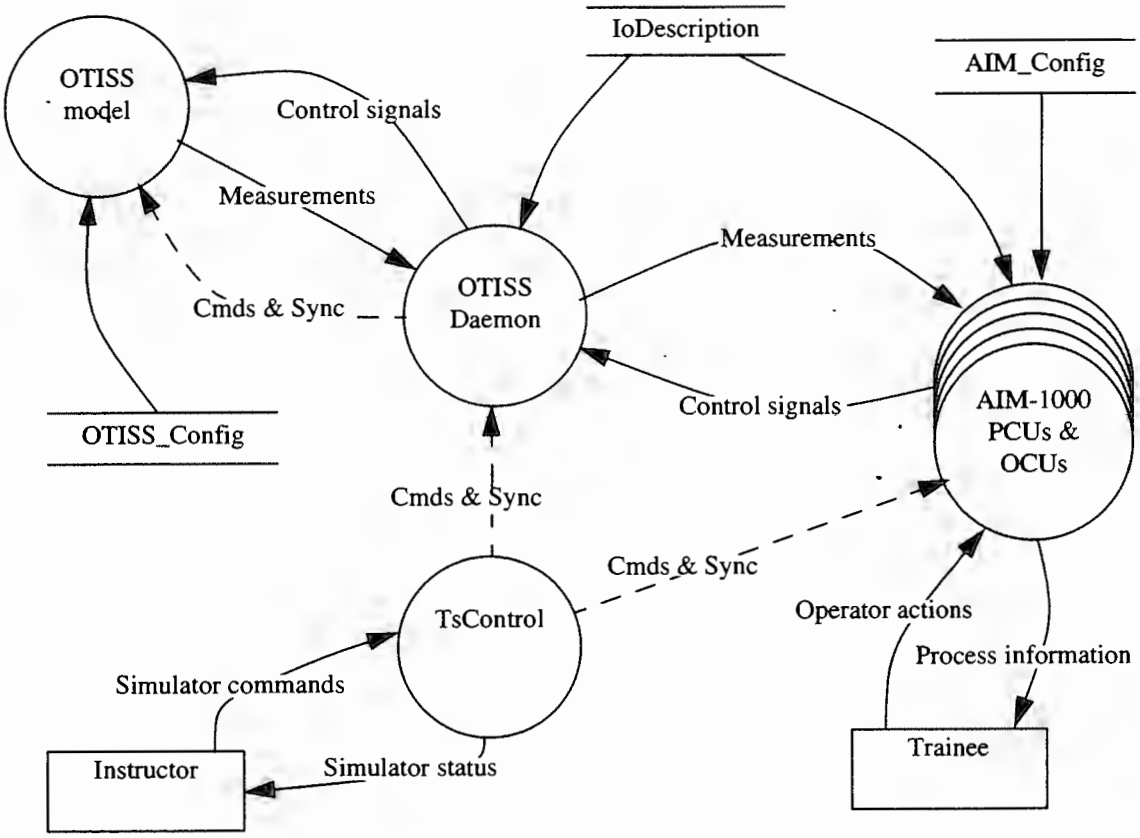
The following SW components in Figur 1 was developed in the HOPE project:

1. TsControl, the instructor HCI, which also synchronises the simulator to real time and coordinates the data exchange between AIM-1000 and OTISS.
2. The command protocol from TsControl to OTISS Daemon and AIM-1000, and the data exchange protocol between the OTISS Daemon and AIM-1000.
3. The OTISS daemon which implements the MiTS interface to SAST's proprietary internal protocol.
4. The IoDescription cross connection file defining the data transfer between OTISS and AIM-1000 by mapping OTISS and AIM-1000 tag names.
5. Some utility programs for setting up the IoDescription.

The key point that made the HOPE simulator feasible, was the ability to run the AIM-1000 system software nearly unmodified on Unix work stations. Each Process Control Unit (PCU) and Operator Control Station (OCU) executes as an independent Unix process. This allows flexible allocation of the PCUs and OCUs on the available computers.

The HOPE simulator runs on four standard HP9000/735 computers. The OTISS simulator and the OTISS Daemon run on one computer. TsControl and the 19 PCUs and 3 OCUs are distributed on the other three. These three are equipped with special AIM-1000 keyboards and are used as operator stations during training. This configuration runs in real time with a data exchange frequency at 1 Hz, which is the same as in the plant.

Fray (1995) distinguishes between emulated, stimulated and hybrid simulator architecture. The emulated models plant, control system and HCI. The stimulated architecture models the plant and uses control system HW for the rest, while the hybrid only uses control system HW for the HCI. In this sense the HOPE simulator is a completely software stimulated architecture where the AIM-1000 system's ability to run under Unix, saves the costs and complexity associated with stimulated solutions using control system HW.



Figur 1 The HOPE simulator architecture

The selected technical solution allows a new control system configuration to be imported directly from the process control system's data base to the HOPE simulator. Only minor modifications have to be carried out in the model to reflect a change in the control system configuration. If for example a new transmitter tag has been added to the control system, a corresponding transmitter module has to be configured into the model, and an extra line has to be inserted into the IoDescription file. This new entry in the IoDescription file is needed to link the measurement from the transmitter in the model to the input of the new transmitter module in control system.

The OTISS model may also run completely disconnected from the HOPE simulator using its own internally modelled control system. This is achieved by a set of SW switches that allow the user to select whether control signals are to be fetched from the OTISS-Daemon or from the internal controllers. OTISS is used as a stand alone process simulator if the internal controllers are enabled. Consequently, only one model has to be maintained for the Heidrun plant.

THE DEVELOPMENT PROJECT

The HOPE development project consisted of three main phases. First, software development. Secondly, simulator configuration and thirdly simulator verification. We used standard structured technics as described in Yourdon (1989) for system analysis and design. The programming was done in ANSI C. We used the POSIX API to access the operating system, OSF/Motif to build the HCI and MiTS for all inter-process communication. About 4.5 man year were spent in this phase. All the software was developed in this phase, but the test configuration did only include a small number of AIM-1000 and OTISS tags.

The goal for the configuration phase therefore was to establish the full simulator. The main activity in this phase was to establish and test all the cross connections between the full OTISS model and the AIM-1000 configuration and to set up static default values for AIM-1000 modules which did not have a counterpart in the OTISS model. This task involved extensive search for, and pairing of, tag numbers in the OTISS and AIM-1000 configurations. We used the script language Perl to automate as much as possible of these tasks.

The numbers of connection points and defaults set in the IoDescription file are 1150 and 1400, respectively. Circa one man year was spent on this task. The result was a simulator that ran satisfactorily around different steady state values.

The final simulator verification phase was required to make the training simulator fully operable, i.e. to allow the trainee to start up the platform from a shut down state and execute normal operational procedures. This phase included a detailed walk through of the control structures and logics required to start up and shut down the process.

This phase also served as a useful verification of both the control system and the model. Although the control system configuration proved to be of good quality, the needs for some modifications were reported. Shortcomings in the OTISS model were mainly caused by mismatches in the granularity between the model and the control system. This activity will probably be continuing until the model has been updated with real operational data and the AIM-1000 system on the platform has become stable. The platform will start oil production by the end of 1995.

RESULTS AND EXPERIENCES

The simulator has so far been used successfully for control system verification and initial operator training. It has also been used to fine tune the control loops by a semi automatic method. Inputs and outputs resulting from system perturbation are logged and used off-line to calculate controller parameters. The method is described in Schei (1994).

The most serious problems during the start up period of the HOPE simulator originate from missing details in the OTISS model. These details are not needed for the operational studies the model originally was designed for, but are important to provide the control system with the necessary inputs and to create realistic operator training. However, it has generally not been difficult to extend the model to overcome such problems, but the total time spent on this has caused some delay in the start-up of the initial training.

One lesson learned is therefore to ensure that the specification of the granularity in the model matches both training requirements and the interfaces to the control system. In addition means must be provided for adding simple interface modules to tailoring the model to the requirements of the control system. Such a flexibility is needed because the configuration of the simulator will start before the control system configuration is thoroughly tested. The the simulator development project must thus be prepared to deal with changes in the control system configuration. To ensure consistency between the control system on the plant and that in the simulator all modifications must be done in the model.

The HOPE simulator concept has been taken over by the AIM-1000 system vendor, Simrad Norge, and is likely be re-used in one of Statoil's next major field developments.

The HOPE simulator's accurate process model together with it's copy of the process control system SW and HCI open interesting possibilities for using the system as a test bed for operator support systems and new control strategies. New control strategies can be implemented directly in the control system and tested in a realistic environment. The MiTS protocol gives any other MiTS process access to the measurements from the OTISS model. New operator support tools might therefore be fully integrated and receive on line data.

REFERENCES

Fray, R. (1995). "Compact simulators for fossil-fueled power plants". IEEE Spectrum, Vol. 32 No 2.

Rødseth, Ø.J., Haaland E. (1993). ``MITS: An Open Standard for Integrated Ship Control". Proceedings of ICMES 93, Hamburg September 1993.

Schei, T.S. (1994). "Automatic Tuning of PID Controllers Based on Transfer Function Estimation". Automatica, Vol. 30, No 12, pp 1983-1989. Elsevier Science Ltd.

Woods, E.A., H. Backer, P.E. Wahl and K. Telnes, (1994). "The HOPE Procedure Handling System, "Pre prints of AIRTC'94, 605 - 609, València, Spain.

Yourdon, E. (1989). "Modern structured analysis". Prentice-Hall, Inc.

ANALYSIS OF MULTILAYER MEDIUM ACOUSTIC BEHAVIOUR WITH THE SIMAMCO SOFTWARE

PEYRAUT FRANÇOIS

SCIENCES INDUSTRIES CONSEILS
14, Avenue de Sceaux
78 000 VERSAILLES
FRANCE
Tel. : (1) 39 49 05 27
Fax : (1) 39 02 77 23

1 - INTRODUCTION

This paper deals with the SIMAMCO software which allows the engineers to predict acoustic behaviour of multilayer media. Because of their high heterogeneity, source of many impedance's breaking, these media constitute efficient sound insulation. They are used in various fields like automotive industry, aeronautics, rail transport, shipbuilding, the building trade ...

To compute the reflexion and transmission acoustic response of multilayers with an harmonic incident plane wave, we have used a plane wave approach ([1]). The amplitudes of local waves in each layer are computed by a transfer matrix method. This method is originally due to W.T.Thomson ([6]).

By using a semi-analytical method which combines a plane wave approach with a transfer matrix technique, SIMAMCO gives results very quickly. Consequently, this software is really suitable for design stage.

A possible anisotropic behaviour of layers and an eventual damped behaviour of viscoelastic type are taken into account.

We have written SIMAMCO in fortran 77.

2 - NOTATION

Bold quantities represents vectors. The other quantities are real or complex numbers following the context.

3 - THEORY

We analyse the problem within the frame of small perturbation's theory around a configuration at rest. The fluids upstream and downstream of the multilayer are assumed perfect and we neglect the voluminal forces of gravity.

We also assume that the incident wave is harmonic plane and propagates in free field. In other words, the only multilayer attack wave comes from the source and the only reverse wave is radiated by the multilayer.

We assume in the same way that the transmitted wave downstream of the multilayer is plane and propagates in free field.

Finally, to permit a quasi-analytic solving of the problem, two layer's hypotheses are made :

1) the layers are homogeneous
2) they are infinite in the perpendicular directions to depth axis

With infinite layer's hypotheses, it can be seen that the problem is independant of x variable. This spacial variable is perpendicular to the plane defined by the propagation direction y and the layers depth axis z. Taking into account this geometrical property, the Snell's law ([5]) leads to write the displacements $\mathbf{u}$ with the form :

$$\mathbf{u}(\mathbf{X},t) = \mathrm{Re}[\mathbf{U}(z)\exp(-i\Omega\sin(\Theta)y/c_0)\exp(i\Omega t)] \qquad (1)$$

t is the time, Ω the circular frequency, Θ the incident angle, c_0 the upstream sound velocity and i the complex number whose square is equal to -1. The Re notation means real part and dash above quantities the conjugate of this quantities.

By combining equation (1) with the laws of momentum's conservation and material's behaviour, we can demonstrate that the vector $\mathbf{U}$ is solution of ([4]) :

$$\Omega^2(m_v\mathbf{Id}-\sin^2(\Theta)/c_0{}^2\mathbf{A}^{22})\,\mathbf{U}(z) + i\Omega\sin(\Theta)/c_0(\mathbf{A}^{23}+\mathbf{A}^{32})\,d\mathbf{U}(z)/dz + \mathbf{A}^{33}\,d^2\mathbf{U}(z)/dz^2 = 0 \qquad (2)$$

which is an homogeneous second order differential system with no-constant coefficients.

m_v is the density, $\mathbf{Id}$ represents the identity matrix and $\mathbf{A}^{ik}$ $(2\le i,k\le 3)$ are 3x3 matrix dependant of circular frequency and layers mechanical properties. For each couple of (i,k), the coefficients of these matrix are defined by :

$$(\mathbf{A}^{ik}(\Omega))_{jh} = a_{ijkh}(\Omega)\ (1\le j,h\le 3) \qquad (3)$$

where $a_{ijkh}(\Omega)$ are the coefficients of the complex stiffness matrix.

With a first derivative unknown well ajusted to the problem, the second order differential system (2) is turned into a first order system. This system is solved in each layer. The solution is then known analytically except for the integration constants. These constants are determined by displacements and stresse's continuity at layers interfaces in conjunction with a transfer matrix method ([3]).

4 - VALIDATION

SIMAMCO has been valided by numerical and experimental correlations in frequency field, for normal incidence, and in oblique incidence field. Reference curves are issued from ([2]).

Figures 1 and 2 represent the transmission coefficient expressed with decibel as a function of frequency. These two figures concern a microvon panel in water for normal incidence. The microvon's mechanical and geometrical properties are :

- density = 700 kg/m^3
- longitudinal wave velocity = 200 m/s
- longitudinal loss angle tangent = 0.2
- transversal wave velocity = 50 m/s
- transversal loss angle tangent = 0.2

Reference results are numerical (curve named Plan Infini) and experimental (curves referenced by L=0.1, L=0.2 and L=0.3). In each case, we note a perfect correlation with SIMAMCO computation.

On figures 3 and 4, we have plotted the transmission coefficient expressed with decibel against incident angle for a 3 cm steel panel in water. The reference result is issued from a numerical calculation. We note again a perfect concordance with SIMAMCO computation.

5 - CONCLUSIONS AND FUTURE PROSPECTS

The acoustic software SIMAMCO is helpful for sound insulation design, especially for acoustic comfort problems (transport, building trade ...) and military ship's discretion (submarines).

We have successfully valided it in all tested configurations. Its reliability is consequently one of its greater advantage.

With its analytical approach of the propagation problem, SIMAMCO allows users to input quickly data (no tedious mesh generation). For the same reason, results are got very quickly.

SIMAMCO is conceived to take into account various materials (elastic or viscoelastic, isotropic or anisotropic) and stays open for new material's integration (porous or composite materials, elastomere ...).

Finally, SIMAMCO could be easily adjusted to deal with vibratory problems.

Because of its short response delay and its simplicity of using, SIMAMCO is really adjusted to multilayers acoustic design. SIMAMCO is complementary to experimental tests. It allows cost reduction by advising the engineer in material's choice.

REFERENCES

[1] "Wave Propagation in Elastics Solids", J.D. Achenbach, North-Holland, Amsterdam, 1973

[2] "Caractérisation de Matériaux Utilisés en Acoustique Sous-Marine à l'Aide de la Mesure en Cuve des Coefficients de Réflexion et de Transmission de Panneaux de Dimensions Finies", Christian Giangreco, Thèse de l'Université Technologique de Compiègne, décembre 1990

[3] "Calcul et Visualisation de l'Amortissement d'une Onde Harmonique dans un Multicouche Viscoélastique", François Peyraut, Thèse de l'Université Paris 6, décembre 1991

[4] "Amortissement Dans des Composites Viscoélastiques", Nadia Labed, Thèse de l'Université Paris 6, décembre 1992

[5] "Crystals Acoustics", M.J.P. Musgrave, Holden Day, San Francisco, 1970

[6] "Transmission of Elastic Waves Through a Stratified Solid Medium", W.T. Thomson, J. Appl. Phys. 21, 89, 1950

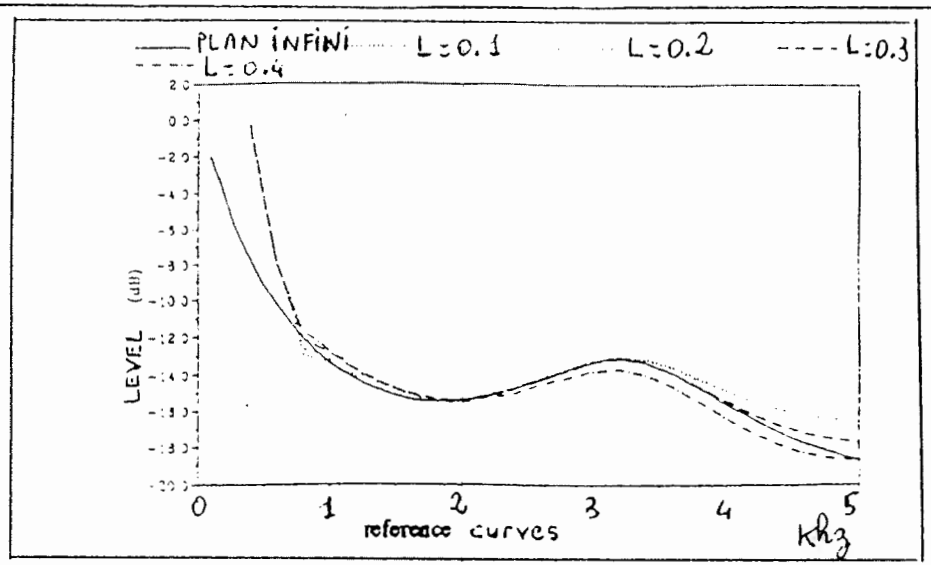


Figure 1

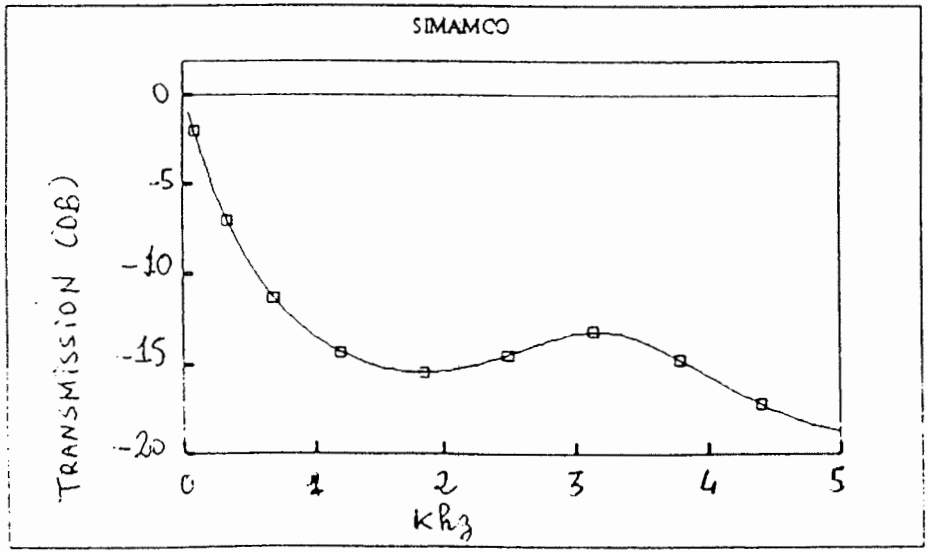


Figure 2

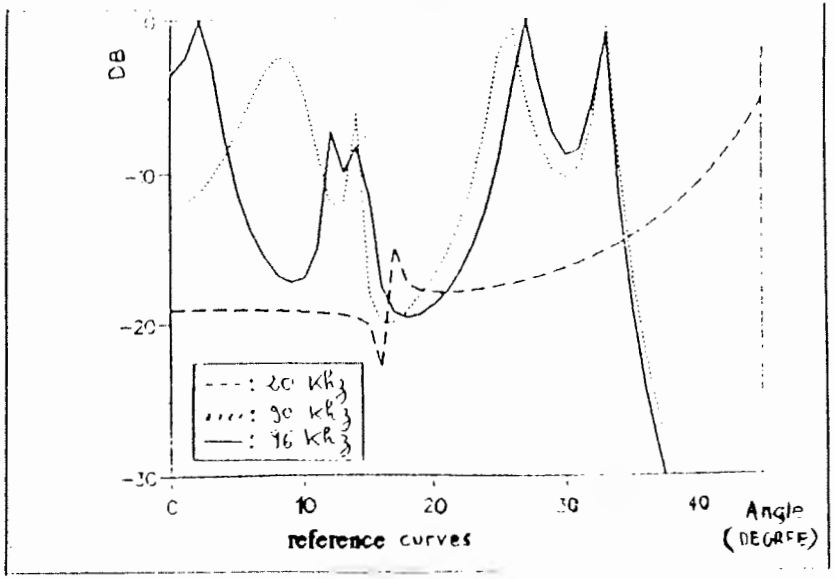


Figure 3

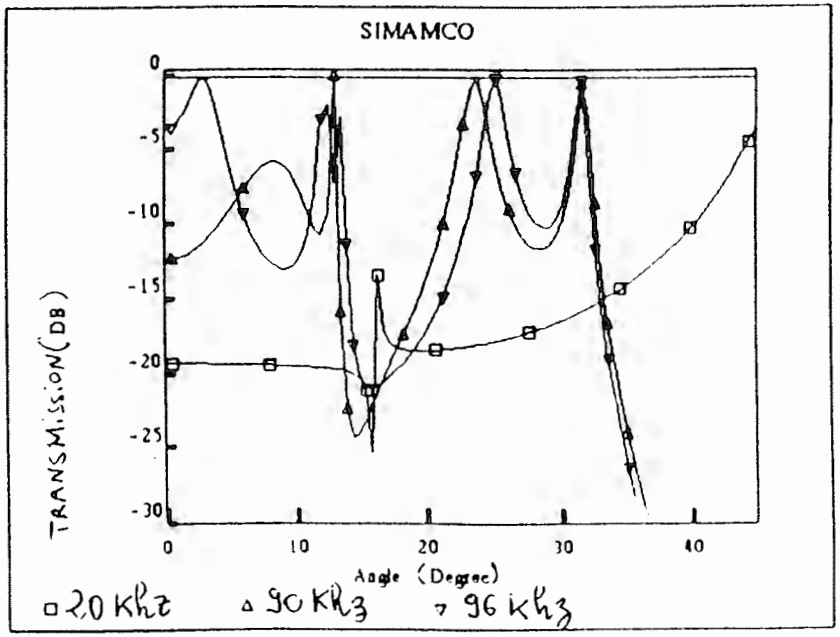


Figure 4

The module partDEQ - solving partial differential equations using SIMUL_R

R. Ruzicka
SIMUTECH
Hadikgasse 150, A-1140 Vienna

Abstract

This contribution will show all features of the simulation language SIMUL_R, which make it possible to solve systems of partial differential equations (PDEs) and display their solutions. The modelling features are contained in a module called *partDEQ*. Some examples will show how to use partDEQ to solve PDE problems.

Introduction

Most modern simulation languages for continuous systems' simulation contain features for solving ordinary differential equations (ODEs). But in many cases ODEs are not as accurate as is necessary to describe all dynamic features of a real system:

think of electronical circuits with long wires or e.g. the bus of a computer; these cannot be simply modelled by using for example a capacitor and a resistor to show the effects of voltage reflections;

or the absorbtion of pollutants by a filter: you need to model the process of absorbtion to be able to predict, when the filter will be filled;

or the closing of valves in waterworks: the waves of water pressure and velocity have to be modelled using PDEs to simulate extreme and dangerous states.

There are a lot of tools, each spezialized in one specific field of PDE simulation: finite element methods for mechanical purposes or special numerical libraries for special types of PDEs. Nevertheless those are difficult to use for general purposes and mostly have no user friendly interface for modelling and presentation of results.

How to add PDE features to a simulation system for ODEs?

A modelling utility for PDEs in an environment for ODEs, like SIMUL_R, should be

- fully integrated into the ODE system (PDEs and ODEs combinable)
- easy to use
- the result functions must be easy accessible
- the results should be drawn over the locality dimensions without user "chin ups"

SIMUL_R features for PDEs

SIMUL_R a priori offers some features, which help modelling PDEs:

- a very powerful macro and meta language
- sorting of equations
- easy modelling of implicite problems
- so-called *cross plots*

Meta and macro commands

SIMUL_R contains all macro features known from the programming langugage C, but extended by very important points:

+ recursive macros

+ varying number of parameters for the same macro
+ meta loops (for, while)

Equation Sorting

The equation sorting algorithm of SIMUL_R - sorting occurs within the DERIVATIVE section of a model for finding dependencies of variables -

+ automatically detects algebraic loops,
+ searchs for a minimal set of variables, necessary to be iterated for solving an implicite problem,
+ transforms the equations into a zero search problem (yet available algorithms, choosable by menu: damped Newton, DASSL),
+ delivers information messages, e.g. about the structure of the equation matrix

  banded matrices can be easily detected, necessary variable resorting can be performed.

+ SIMUL_R offers special banded matrices methods for the Newton and DASSL algorithms.

Implicite problems

can be modelled in SIMUL_R as

$0=g(x)$ zero point notation

or

$x=f(x)$ fix point notation

(as well for vectors and matrices). Therefore it is much easier to use special algorithms for PDEs, like Crank-Nicolson's method.

Cross plots

mean, that the time variations of a set of variables (e.g. variables representing the solution of a PDE over the locality dimension) can be drawn in the vertical y direction of a plot (with constant horizontal x position) and are automatically connected.

These features have been used to build up the partDEQ module with a library of macros, which translate a PDE notation into a set of equations and algorithms, which can be treated by usual SIMUL_R.

Modelling PDEs

PDEs of the form

$$\frac{d^k y}{dt^k} = f\left(t, y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, .., \frac{d^{k-1}y}{dt^{k-1}}, x, \frac{dy}{dx}, \frac{d^2y}{dx^2}, .., \frac{d^m y}{dx^m}\right)$$

where

t	time (first independend variable)	*f*	an arbitrary expression
x	locality (second independend variable)	*k*	maximum derivative order over t
y	state variable	*m*	maximum derivative order over x

are solved and can be inserted as parameters of a partDEQ macro; additionally the initial conditions and the length of locality is specified.

The macros are also available for up to 3 locality variables (3 dimensions) with mixed local derivatives.

Special macros are used to specify boundary conditions (fixed conditions, or switchable conditions).

The results are sampled in user specified array variables, which can be simply plotted using cross plots.

Currently the two methods

- method of lines
- Crank Nicolson (implicite method)

are available, which only differ - at the user interface - in the name prefix of the macro.

Examples

Introductional example

The first example shows a simple model of sucking water into wood:

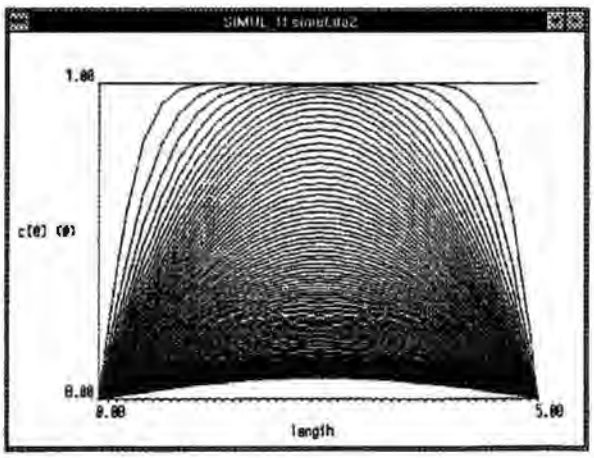
$$\frac{dy}{dt} = D * \frac{d^2y}{dx^2} \qquad \text{with} \qquad y(t0,x) =$$

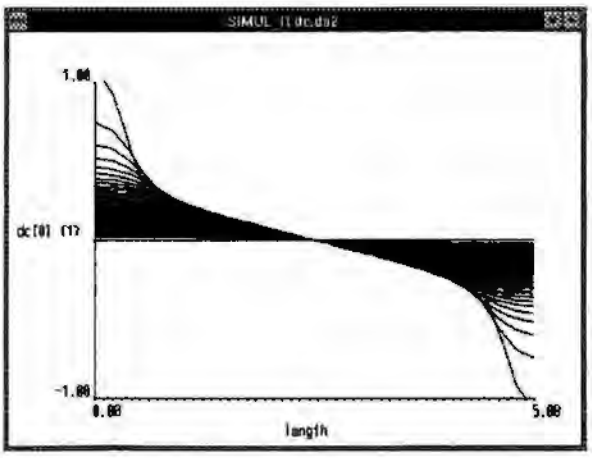
At time *T* a step from 1 to 0 is performed on the left and right boundaries. Here the first and second order locality derivatives are named *dyx* and *dyxx*, respectively.

The model:

```
#set n=50#                          " number of discretization points "
#include 'simcomac.def'
#include 'partdeq.def'              " include module partDEQ "

wood {

        CONSTANT tend=1000, len=1, D = 1e-5, T=0.2;
        float yl, yr, y[#n+1#], dyx[#n+1#], dyxx[#n+1#];
        int i;

        DYNAMIC {
                DERIVATIVE {
                        yl=1-#step(T);                      " left bound "
                        yr=1-#step(T);                      " right bound "
                        #partDEQlrc(1,D*dyxx[i],i,len,yl,yr,y,1,dyx,dyxx)
                }
                TERMINATE t>=tend;                          " termination condition "
        }
}
```

It is easy to draw the state variable results and the locality derivatives over *x*:





Net of water pipes

The second example shows pressure waves in a pipe net of waterworks: each pipe is described by the two coupled PDEs (for pressure *p* and velocity *v*)

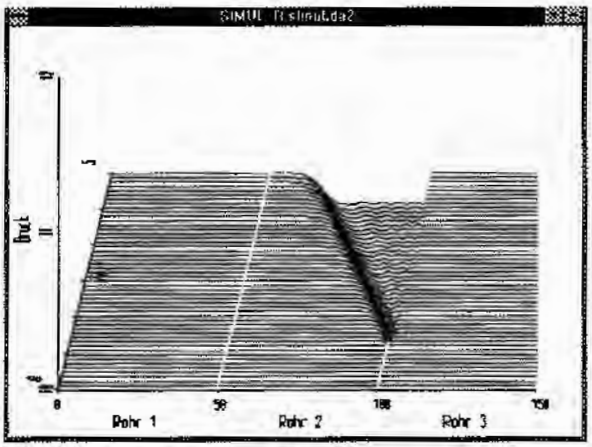
$$\frac{\partial p}{\partial t} = -\frac{c*c}{g} * \frac{\partial v}{\partial x} \quad \text{and} \quad \frac{\partial v}{\partial t} = -\frac{\partial p}{\partial x} * g - \frac{\lambda}{2*D} * v * |v| \quad ,$$

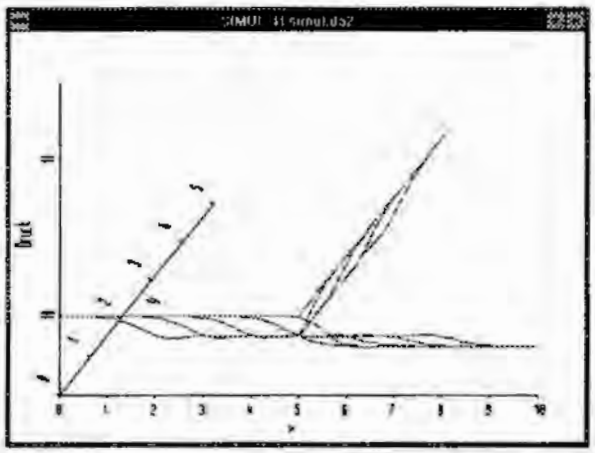
the net is described using bond graphs.

On the left hand side, there is a water container with constant pressure, on the right hand side and at the top there are two valves. A water consumer opens a valve: this leads to water reflections through the entire net.

A pipe is modelled using partDEQ macros for the PDEs and is formulated as a macro. The net is modelled by the bond graph tool BAPS using this macro. This allows for an easy changing of the net's topology.

The pictures show the pressure in the three pipes: the first wave after closing one valve and five states (at five different points of time over the real topology).





Future developments

In the near future there will be available methods for

- automatic adaption of the discretization points and
- new methods for computing the differential quotients.

Literature

- R.Ruzicka: *SIMUL_R - eine Simulationssprache mit speziellen Befehlern zur Modelldarstellung und -analyse*, Informatik Fachberichte 179, Proceedings of the 5th Symposium Simulationstechnik, Springer, Aachen, 1988
- R.Ruzicka: *Environments For SIMUL_R*, 3rd European Simulation Congress, Proceedings, Edinburgh, 1989
- J.Niwinski, R.Ruzicka: *Online Simulation With SIMUL_R*, SCSC 91, Baltimore Maryland, 1991
- R.Ruzicka: *SIMUL_R PARALLEL - Hardware-in-the-loop Simulation mit Transputern unter Windows*, Fortschritte in der Simulationstechnik, Band 6, Proceedings of the 8th Symposium Simulationstechnik, Vieweg, Berlin, 1993
- R.Ruzicka: *Optimierung technischer Systeme mittels Evolutionsstrategien - ein Standardverfahren in SIMUL_R*, Fortschritte in der Simulationstechnik, Band 9, Proceedings of the 9th Symposium Simulationstechnik, Vieweg, Stuttgart, 1994

EXTREMELY EXACT and FAST COMPUTATIONS

Jiří Kunovský, Karel Mikulášek

Department of Computer Science and Engineering, Technical University Brno
Božetěchova 2, CZ–612 66 Brno, The Czech Republic
Telephone: +42-5-7275234 (+42-5-41212219), Fax:+42-5-41211141
e-mail: kunovsky@dcse.fee.vutbr.cz

1 Abstract

The paper deals with the numerical solution of differential equations. The numerical integration employs the Modern Taylor Series Method. The useful properties of the Modern Taylor Series Method are demonstrated on two problems: in homogeneous linear differential equations and nonlinear differential equations with time dependent coefficients. An application of multiple word arithmetic to the solution of stiff systems is shown.

2 Introduction

The best-known and most accurate method of calculating a new value of a numerical solution of a differential equation is to construct the Taylor series in the form

$$y_{n+1} = y_n + h * f(t_n, y_n) + \frac{h^2}{2!} * f^{[1]}(t_n, y_n) + \ldots + \frac{h^p}{p!} * f^{[p-1]}(t_n, y_n), \tag{1}$$

where h is the integration step.

The Modern Taylor Series Method is used for the computation. The main idea behind the Modern Taylor Series Method is an automatic integration method order setting, i.e. using as many Taylor series terms for computing as needed to achieve the required accuracy.

Methods of different orders can be used in a computation. The 1st order method (ORD=1) means that when computing the new value y_{n+1} only the first Taylor series term is taken into account

$$y_{n+1} = y_n + h * f(t_n, y_n), \tag{2}$$

the 2nd order method (ORD=2) uses Taylor series terms up to the second power of the step h

$$y_{n+1} = y_n + h * f(t_n, y_n) + \frac{h^2}{2!} * f^{[1]}(t_n, y_n), \tag{3}$$

etc.

3 Homogenous Linear Differential Equations

Let us solve the differential equation

$$y' = y, \quad y(0) = 1. \tag{4}$$

A numerical solution of (4) by the Taylor series method (using (1)) is

$$y_{n+1} = y_n + h * y_n + \frac{h^2}{2!} * y_n + \ldots + \frac{h^p}{p!} * y_n + \ldots \tag{5}$$

$(y = y' = y'' = \ldots\ldots = y^{(p)})$,

or

$$y_{n+1} = y_n * (1 + h + \frac{h^2}{2!} + ... + \frac{h^p}{p!} + ...). \quad (6)$$

The numerical solution of (4) (using (5) or (6)) will depend on the number of Taylor series terms used.

Note : The analytical solution of (4) is

$$y = e^t. \quad (7)$$

Results illustrating the use of the Taylor series for applying a numerical integration method are shown in Tab.1. Tab.1 demonstrates the results of a numerical solution of the differential equation (4) after one computation step (with the integration step h=1s).

Reduced value y(1)	ORD	Time (ms)
2.	1	0.084
2.	2	0.140
2.	3	0.195
2.7	4	0.248
2.71	5	0.307
2.718	6	0.365
2.7182	7	0.422
2.7182	8	0.468
2.718281	9	0.531
2.7182818	10	0.589
2.71828182	11	0.649
2.718281828	12	0.693
2.7182818284	13	0.757
2.71828182845	14	0.828
2.71828182845	15	0.861
2.71828182845904	16	0.911
2.71828182845904	17	0.983
2.71828182845904	18	1.033

Tab.1

In each line of Tab.1 the "Reduced value y(1)" of the numerical solution of the differential equation (4) and the time evaluation of computation - for the method order ORD used are printed. In order to make the results of the numerical solution more clear and illustrative only the digits tallying the digits of the exact solution are shown (for the exact solution of the equation (4) for $t_1 = h = 1s$, in view of the equation (7), we have

$$y(1) = e^1 = 2.718281828459045235.....).$$

It follows from Tab.1 that the requirement of a higher method order is justifiable - with the same integration step h a higher method order (i. e. with more terms of the Taylor series) can yield a higher accuracy (it approximates better the exact solution).

3.1 Experimental Time Evaluations

Tab.1 brings time evaluation of the computation. For instance, using the 17th method order requires 0.983 ms.

If we wanted to reach the same accuracy (at point t=1s) by the 4th order Runge-Kutta method, we would have to use a substantially shorter integration step and the computation time would be 271.229 ms (Tab.2).

h(s)	Reduced value y(1)	Time (ms)
1	2.7	0.299
0.1	2.7182	2.691
0.01	2.718281828	27.500
0.001	2.71828182845904	271.229

Tab.2

Note: All time evaluations were obtained on the ACA 32000 computer (based on National Semiconductor 32000 processor).

3.2 Accuracy and Word Width

The increase in the accuracy of the result is not unlimited. In Tab.1 the accuracy stops increasing when the ORD reaches the value of 18. This is caused by an underflow during the computation of the higher order Taylor series terms. The addition of these terms changes neither the resulting value of y_{n+1} nor the absolute error - the absolute error has reached its saturated value ESAT (the value of ESAT depends on the word width of the arithmetic of the computer used).

The accuracy of the result can be influenced in a considerable way by the word width. For instance when the word width is 32 bits and the integration step h=1s, only the accuracy $1.863 * 10^{-9}$ of computation of the equation (4) can be reached. With a specially constructed 128-bit arithmetic a very high computation accuracy $4.408 * 10^{-39}$ for the equation (4) can be reached very fast even with the integration step h=1s.

4 Differential Equations with Time Dependent Coefficients

The high accuracy and the high speed of the Modern Taylor Series Method is demonstrated on the following system of equations

$$y' = aycost \quad y(0) = 1 \tag{8}$$
$$x' = -axcost \quad x(0) = 1 \tag{9}$$
$$z = xy \tag{10}$$

The system of equations (8), (9), (10) was deliberately designed for the variable z to characterize the accuracy of the computation.

The accuracy of the computation by the Modern Taylor Series Method is preserved even if the variables reach values of 10^{43} and 10^{-44} by order of magnitude. The numerical solution of the system (8),(9) reaches these values for a=100. The achieved high accuracy of the computation of the expression

$$z = x \cdot y = 1$$

is based on the fact that the new value of y_{n+1} is calculated in each step automatically until the adding of the next Taylor series terms has no effect.

Note: A special comparison has been completed. The value of z(100) (for a=100) has been calculated by the Modern Taylor Series Method and by the 4th order Runge-Kutta method. The result z(100)=1.000000000000000 was achieved after 136 seconds of computation using the Modern Taylor Series Method. In the same time the RK-4 method was able to yield only the value z(100)=1.0064.

5 Stiff Systems

Let us focus our attention on particular problems with the integration of stiff problems. It is well known that in stiff systems an extremely small integration step must be used. If, however, we combine Modern Taylor Series Method and multiple word arithmetic, the computation can be done with an extremely large integration step.

Let us consider a system of linear differential equations

```
y1'=y2,                  y1(0)= 1                              (11)

y2'=-a*y1-(a+1)*y2,      y2(0)=-1                              (12)
```

with the exact solution

$y_1 = e^{-t}, \qquad y_2 = -e^{-t}.$

The general solution of the system (11),(12) is in the form

$y_i = A_i e^{-t} + B_i e^{-a*t}$

$i = 1, 2 \qquad A_i, B_i \qquad$ are constants

and the eigenvalues of the matrix of the system are

$\lambda_1 = -1, \qquad \lambda_2 = -a.$

The results of the computation, for the integration step h=1, accuracy $EPS = 10^{-6}$ and for 30 terms of the Taylor series (ORD=30) are shown in Table 3. The table shows the effect of using the multiple word arithmetic. If the computation is done with the word length 8 bytes, the coefficient a can at most have the maximum value $a = 10^2$. When using the word length 512 bytes, the value of a can be $a = 10^{175}$.

The described positive effect of the use of multiple word arithmetic can also be used to solve intricate nonlinear stiff systems.

Word length (Byte)	a	Time of computation (s)
8	10^2	0.0012
16	10^5	0.0090
32	10^{10}	0.0500
64	10^{21}	0.1000
128	10^{43}	0.3800
256	10^{87}	1.2100
512	10^{175}	4.3400

Tab.3

6 Conclusion

We consider the following points to be a contribution to the methodology of the numerical solution of differential equations:

- Direct use of Taylor Series Method
- Use of variable number of terms of the Taylor series
- Use of multiple word arithmetic
- Applicability

INTERACTIVE SIMULATION OF A FUZZY-LOGIC-CONTROLLED NONLINEAR SERVOMECHANISM

Granino A. Korn

G.A. and T.M. Korn Industrial Consultants, RR1, Box 96C, Chelan, WA 98816

ABSTRACT

DESIRE/NEUNET software (DOS or UNIX), originally designed for combined simulation of neural networks and dynamic systems, is equally suitable for interactive simulation of fuzzy-logic control systems. A single page of code, much like ordinary mathematical notation, produces complete sets of nonuniformly spaced fuzzy-set membership functions as vector differences of simple hard-limiter neuron activations, implements rule table and defuzzification, and solves the nonlinear differential equations for a servomechanicsm. An interpreted experiment-protocol program sets and modifies fuzzy-set centers, servo parameters, and rule-table entries. When the interpreter calls for a simulation run, a fast runtime compiler compiles fuzzy-logic operations, rule-table references, and differential equations within 0.05 sec - an unnoticeable delay. The solution then starts immediately and produces color graphics. Interactive rule-table modifications easily permit experiments with different fuzzy-set partitions and nonlinear control schemes.

TRULY INTERACTIVE SIMULATION OF COMPLETE DYNAMIC SYSTEMS

A fuzzy-logic controller is a function generator which produces a desired control function from sensor inputs. Rule-table-based fuzzy-logic controllers let designers use intuition or prior knowledge by working with simple linguistic rules, such as

IF output error is positive AND output rate is positive THEN torque is negative

We will simulate the application of such a controller to an electromechanical servomechanism which is described by nonlinear differential equations.. Its servo motor drives its load so that the servo output angle **x** = *x(t)* follows an input waveform **u** = *u(t)* after an initial transient. **x** is read from an angle pickoff, and its time derivative **xdot** is read from a tachometer on the motor shaft. The controller produces the motor control voltage **V** as a function of the servo error **e = x - u** and the output rate **xdot**. **V** is then amplified and causes a motor torque **torque**, which is limited by motor-field saturation. For convenience, we scaled the effective moment of inertia of motor and gears to equal 1. The dynamics of motor, gears, and load are modelled by nonlinear differential equations.

We employ "direct-executing" simulation software (DESIRE/NEUNET) incorporating a runtime compiler which translates even large simulation programs so quickly that there is no noticeable translation delay. This technique permits truly interactive modelling. An interpreted *experiment protocol program* - much like an advanced basic program - sets up parameters, function tables, and arrays and then calls for a simulation run with a **drun** command. At this point, the simulation-run code (*DYNAMIC program segment*), automatically compiles into fast binary code, which then runs immediately.

DESIRE lets you enter and edit experiment protocol and scalar and/or vector differential equations, plus neural-network and fuzzy-logic models on the CRT screen. Reference 3 lists the complete program. The DYNAMIC program segment segment describing the servo and controller has only 13 program lines. The servo equations of motion are entered in quite ordinary, easily readable mathematical notation:

d/dt x = xdot ¦ d/dt xdot = torque - R * xdot

where **R** is a motor damping coefficient. The motor torque is modelled with

torque = - maxtrq * sat(g * V/maxtrq)

Motor field saturation limits the torque between - **maxtrq** and **maxtrq**. We used a hard limiter, but you can replace **sat()** with a soft limiter such as **tanh()**.

FUZZY-LOGIC CONTROLLER

The servo controller is a function generator producing **V** as a function of **e** and **xdot**. A simple linear controller might implement

```
V = - k * e - r * xdot
```

Instead, we experiment with a more powerful nonlinear two-input *fuzzy-logic controller*. The range of each of the controller input variables **e** and **xdot** is partitioned into 5 fuzzy sets, which could be labelled negative-large, negative-small, zero, positive-small, and positive-large. The rule table holds 25 entries corresponding to different combinations of **e** and **xdot** values.

GENERATING NONUNIFORMLY SPACED MEMBERSHIP FUNCTIONS

We need a one-dimensional array (vector) **mbre** of *Ne* triangular membership functions **mbre[j]** of **e**, and a vector **mbrxdot** of *Nxdot* membership functions **mbrxdot[k]** of **xdot.** For extra resolution near zero error and output rate, *we space our fuzzy sets more closely near zero.* The DESIRE experiment-protocol program reads nonuniformly spaced membership-peak coordinates **E[j]** and **Xdot[k]** from short **data** lists,

```
data - 2 * emax, - 0.5 * emax, 0, 0.5 * emax,
  2 * emax  ¦  read E
data - 2 * xdotmax, - 0.5 * xdotmax, 0,
  0.5 * xdotmax, 2 * xdotmax  ¦  read Xdot
```

To produce, say, **mbre**, the DYNAMIC program segment uses DESIRE **VECTOR** assignments [2,3] which first define hard-limiter functions **mbe** starting at **E[j]** and then subtract successive pairs of such limiter functions[4] to generate the nonuniformly spaced triangular functions **mbre[j]** shown in Fig. 1a.

The one-dimensional membership functions for **r** and **xdot** combine into joint membership functions **M12[j,k]**, with

```
MATRIX M12 = mbre * mbrxdot
```
(*product/sum logic*)

or

```
MATRIX M12 = mbre & mbrxdot
```
(*min/max logic*)

This produces the two-dimensional membership functions **M12** of **e** and **xdot** as elements **M12[j,k]** of a rectangular matrix **M12**. But the DESIRE declaration

```
ARRAY M12[Ne, Nxdot] = m12
```

also lets us access the joint membership functions **M12[j,k]** as elements **m12[i]** of a one-dimensional array (vector) **m12**, in the sequence

(negative-large **e**, negative-large **xdot**)
(negative-large **e**, negative-small **xdot**) ...
(negative-small **e**, negative-large **xdot**)
(negative-small **e**, negative-small **xdot**) ...
. . . . . .

The controller output **V** can then be obtained neatly as the inner product of the joint-membership-function vector **m12** and the rule-table vector **ruletabl**:

```
DOT V = m12 * ruletabl
```

The rule-table entries **ruletabl[i]** are programmed in another **data** list in the experiment-protocol program.

This convenient method of combining "singleton" rule-table entries and joint membership functions is *not* restricted to two-dimensional joint membership functions, since **m12** could be combined, in turn, with a third set of one-dimensional mebership functions by product or minimum fuzzy logic; the resulting two-dimensinal membership-function array **M123** is again made equivalent to a new one-dimensional array **m123**, and this process can be continued as needed.[3]

INTUITIVE DESIGN WITH LINGUISTIC RULES

The controller rules, presumably, ought to counter observed servo errors with torques of opposing sign. If this is done with a large gain parameter **k**, the output will overshoot and oscillate, so we damp the response with a torque component opposing the output rate **xdot**. It may or may not be useful to decrease these effects for small absolute errors and output rates. As a starting point, one can approximate a linear servo or a bang-bang controller.

Figure 1b shows the servo response with the rule table

```
k = 200  ¦  r = 40
data -8 * k-r, - 8 * k-r, -8 * k, -8 * k+r, -8 * k
data - k - r, - k - r, - k, - k + r, - k + r
data - r, - r, 0, r, r
data k - r, k - r, k, k + r, k + r
data 8 * k-r, 8 * k-r, 8 * k, 8 * k+r, 8 * k+r
```

This approximates the response of a linear servo, [1] but produces less than linear damping for large ¦**xdot**¦ values.

We can now experiment with new linguistic rules for improving the servo response. To lower the sinusoid-following error without causing too much of an initial transient, we substituted a narrower zero-zone fuzzy set for the servo error **e** via the new **data** list

```
data - 2 * emax, - 0.1 * emax, 0, 0.1 * emax,
     2 * emax  ¦  read E
```

and tried to kill servo damping in this low-error zone with the modified rule table

```
k = 150  ¦  r = 120
data - 8 * k - 2 * r, - 8 * k - r, - 8 * k, - 8 * k + r,
     - 8 * k + 2 * r
data - k - 2 * r, - k - r, - k, -k + r, - k + 2 * r
data - r, 0, 0, 0, r
data k - 2 * r, k - r, k, k + r, k + 2 * r
data 8 * k - 2 * r, 8 * k - r, 8 * k, 8 * k + r,
     8 * k + 2 * r
```

Figure 1c shows the resulting, entirely different response. The servo now follows the sine table with remarkable accuracy (note that we displayed *one thousand times* the servo error **e** !). The example illustrates the remarkable possibilities of experiments with membership functions and rule table.

USING A NOTEBOOK FILE

Computer simulation experiments require you to record and recall many successive parameter lists, rule-table entries, simulation time histories, and modified-program files. To keep track of such material, DESIRE/NEUNET maintains a ***notebook file*** which automatically records file operations and also lets you lets you record selected program or data lines, and your own comments, with command-mode or programmed **note** *line-number list* and **note** *'comment'* statements.

REFERENCES

1. Korn, G.A.: *Interactive Dynanic-system Simulation*, McGraw-Hill, New York, 1989.

2. -- : *Neural-network Experiments on Personal Computers and Workstations*, MIT Press, Cambridge, MA, 1991.

3. -- : *Neural Networks and Fuzzy-logic Control on Personal Computers and Workstations*, MIT Press, Cambridge, MA, 1995.

4. Geva, S., and J. Sitte: A Constructive Method for Multivariate Function Approximation, *IEEE Trans. on Neural Networks*, **3** : 622-624, 1992.

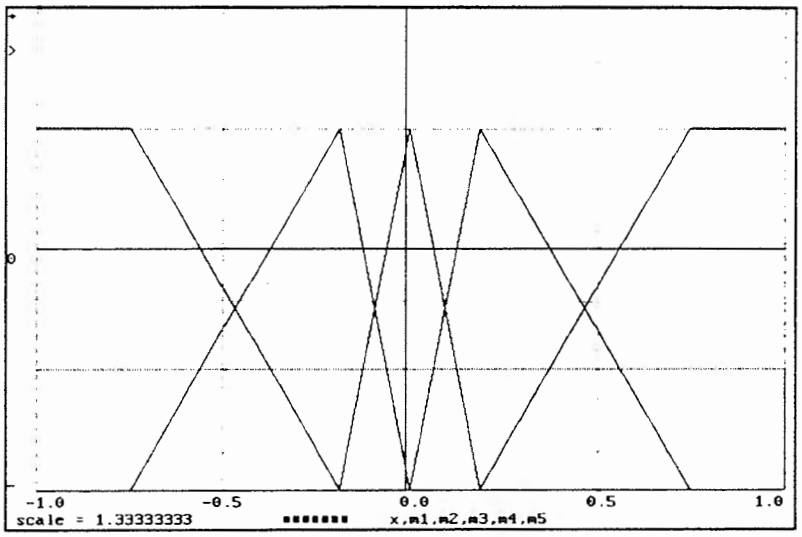


Figure 1a

These servo-error membership functions provide extra resolution near zero error.

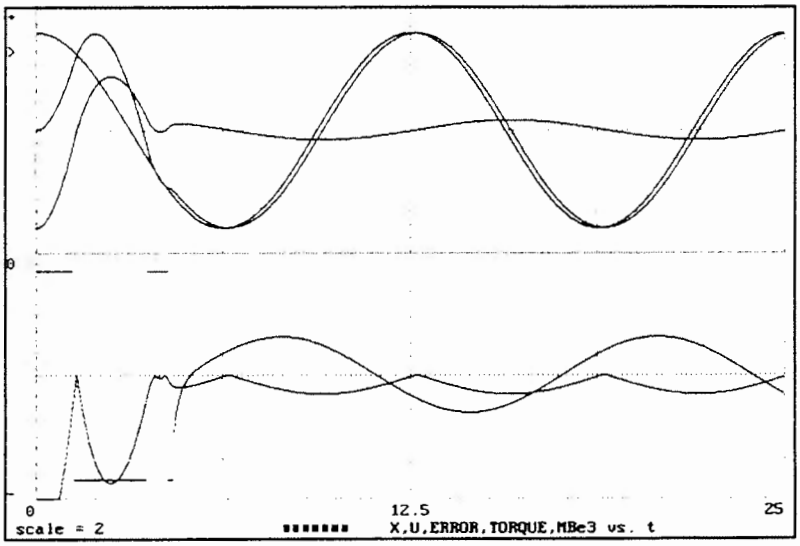


Figure 1b

Servo input, output, error, torque, and zero-error membership function vs. time. The original display was in color.

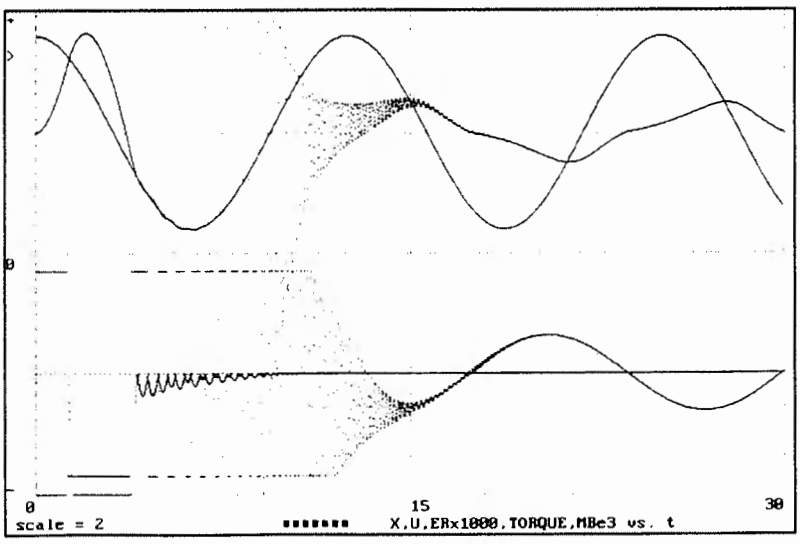


Figure 1c

Servo response with a radically non-linear rule table. The system hunts initially and then follows the sine wave extremely accurately.

Simulation-Based Analysis and Optimization of Complex Systems Using SIMPLORER®

Dipl.-Ing. B. Knorr, cand. ing C. Strunz
Simec GmbH & Co KG, Germany

The growing complexity of system designs as well as decreasing development times require an efficient analysis and optimization of technical systems during computer-aided system engineering. A lot of technical tasks can be reduced to a problem of parameter optimization by selecting relevant system parameters and criteria. The solution of such problems can be realized by various optimization techniques.

Modern mixed-mode/mixed-signal simulation systems, like SIMPLORER®, allow an easy, fast and efficient model creation even for extremely complex systems. The time necessary to examine the desired and the optimal system behavior is considerably high. Based on the automation of system analysis and optimization an optimization module is presented. This module is oriented on approaches which are typical in engineering.

The engineer can use computer-aided analysis and optimization tools in a wide area of technological processes starting from the adjustment of optimal closed-loop controller parameters up to limit analyses. To illustrate the procedure of optimization using SIMPLORER® the dynamic behavior of a battery-powered electrical drive system is examined.

System Development Using SIMPLORER®

The mixed-mode/mixed-signal simulation system SIMPLORER® allows the simulation of mixed technical systems, e.g. of electronic and power-electronic circuits also under consideration of control units. The simulation problem can be modelled in three different description languages. Electric/electronic circuits can be described by lumped components, closed-loop control systems can be represented with block diagrams (signal flow graphs) and control schemes can be modelled using state graphs. Consequently, it is possible to simulate heterogenous systems, e.g., mechanical, pure electronic, and control engineering problems within one system.

SIMPLORER® was recently extended by an optimization module. This module can be used in all cases, where complex interactions and various restrictions make it difficult to find the optimum solution for the development of new products. Great emphasis is put on the analysis and optimization under consideration of the required function on one hand and on the other on the given technological data. So the engineer has the chance to evaluate alternative product designs under technological and economical aspects before the real process will be modified.

Optimization of a Battery-Powered Electrical Drive System

The dynamic behavior of a battery-powered d.c. drive is examined according to the influence of varying system parameters. From the simulation results the optimum solution is extracted.

1. Simulation Model

The system consists of a d.c. motor which is powered by a battery. The electrical circuit of the d.c. motor (armature resistance, armature inductivity, counter e.m.f.) and the transistor pulse chopper are generated by the component library of the SIMPLORER® network modul. The mechanical part of the d.c. motor and the speed regulator are modelled on block diagrams (signal flow graphs) from the SIMPLORER® block diagram modul.

For an illustration of the entire system see Figure 1.

2. Optimization

Various analyses of the battery-powered d.c. drive model are used to optimize the dynamic behavior and to show parameter limits of the system. The aim of system optimization is to ensure the best quality factor by variation of given system parameters. The quality factor of a system stands for the deviation from accepted system behavior.

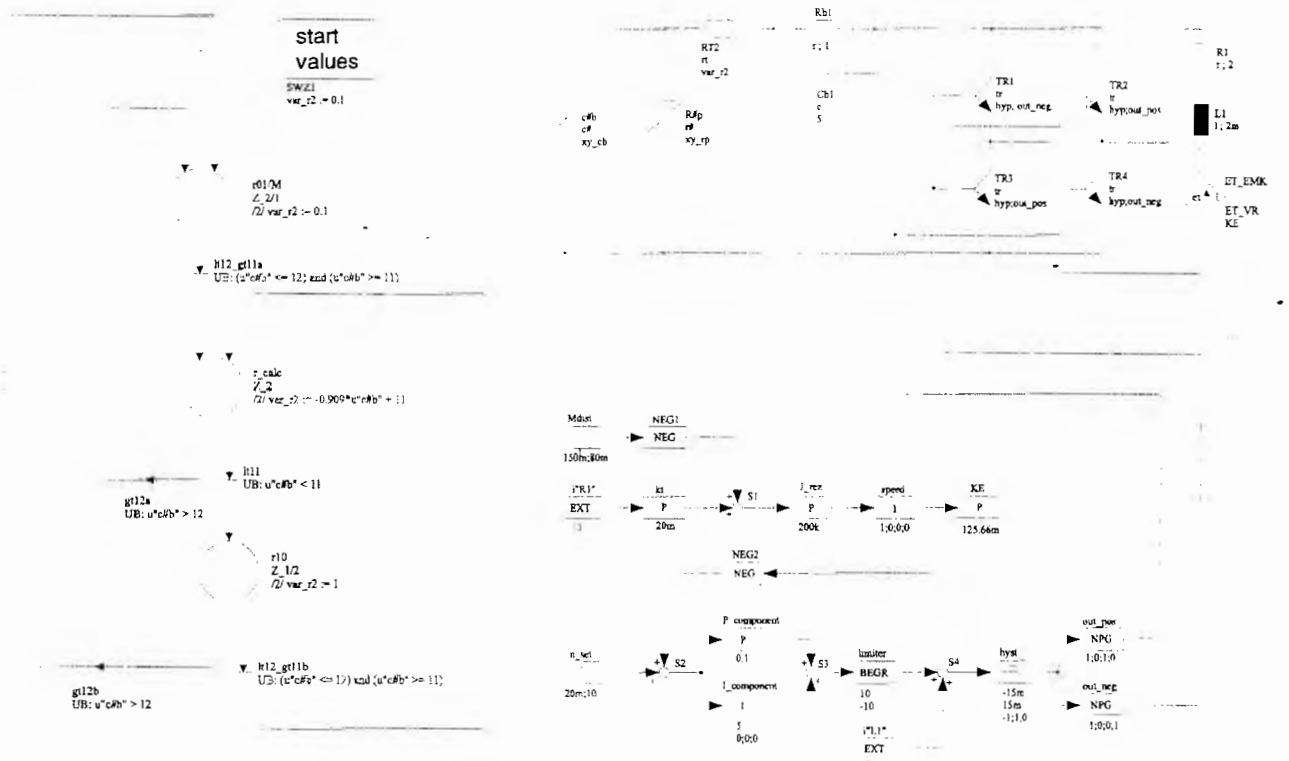


Figure 1: Simulation Model of the Battery-Powered Drive System

2.1. Introduction to the SIMPLORER® Optimizer

To realize an automatic system design the new SIMPLORER® optimization module is exploited. It supports the following kinds of analyses:

- Trend Analysis
- Monte-Carlo-Analyis
- Worst-Case-Analysis

Furthermore, the optimization module provides techniques of intelligent algorithms. A quality criterion (the target function) has to be maximized or minimized as a result of optimization.

The character of an optimization task is represented by its analysis. The analyses differ in the kind of parameter variation. The Trend Analysis is a more straight approach in opposition to the Monte-Carlo-Analysis which assumes a random parameter variation. In a Worst-Case-Analysis parameter limits of a system are interesting. Intelligent algortihms generate the new parameter set for the optimizer based on the results of the previous simulation run.

The use of an analysis combination as well as the use of a single analysis makes it possible to solve an optimization task. Every analysis has its own characteristics and can so be a helpful tool for computer-aided system engineering.

2.2. Usage of the Characteristic Value Library for the Specification of Target Functions

The quality of a system can be described by a number of various characteristic values. Even during an optimization these values have a great importance.

To automize the system design it is necessary to calculate the selected characteristic values during the simulation and not after the simulation. Petri Nets are the basis therefore, which can efficiently be described using the SIMPLORER® state graph module.

Supported by a library with prepared calculation algorithms of characteristic values it is easy to include optimization criteria in the actual simulation task. The particular macro which calculates the target function has to be specified by its parameters and can be evaluated in the same way.

2.3. Optimization of the Dynamic Behavior by Adjustment of Optimum Controller Parameters

2.3.1. Specification of the Optimization Task

The transfer function of the control process is fixed. A PI-Controller is part of the control process. The parameters of its transfer function can freely be selected. The controller realizes a correction of the stationary and dynamic behavior which results in a change of the system quality factor. The adjustment of the PI-Controller's parameters leads to an optimum dynamic behavior of the drive system especially if the reference value and the disturbance value will be modificated.

The d.c. drive has to be examined in no-load operation, in motor operation and in braking operation. The initial parameter set of the considered PI-Controller is P-Part=0.1 and I-Part=5. The simulation model has the system behavior shown in Figure 2 below, where the reference and actual value of the rotation is visible. In modifying the counter torque deviations between these variables are conspicuous. The controller's parameter set has to be determined, so that deviations from the rotation reference will be corrected fast and free of overshoot.

The calculation algorithms of this target functions have to be included via the Characteristic Value Library. The target functions of interest are listed in chart 1:

Transient Part	Characteristic Values
no-load operation t=20ms to t=80ms	maximum overshoot time to maximum
motor operation t=80ms to t=180ms	minimimum rotation control times
braking operation t=180ms to t=400ms	maximum rotation control times
overall transient	integral squared error

Chart 1: Target functions

2.3.2. Realization of the Optimization Task

An efficient way to realize an optimization of multiple target functions (so-called polyoptimization) is the use of the Monte-Carlo-Analysis.

By specification of desired limits or ranges for the target functions it is possible to select the simulated parameter sets during optimization. Thus, the optimizer generates a data set that meets all given specifications. The available storage methods are either all solutions, all permissible solutions or the Pareto Set.

In the case of the battery-powered d.c. drive a Monte-Carlo-Analysis with 100 simulation runs has been realized. The P-Part was evenly distributed between 0.1 and 7 and the I-Part was evenly distributed between 5 and 15. All solutions has been stored in a special file to keep the complex interdependencies also after the optimization.

2.3.3. Evaluation of Optimization Results Using Day-Optim

With the help of the postprocessor DAY, embedded in the simulation system SIMPLORER®, the optimization results can be filtered and sorted. Combined with the optional statistic evaluation part of DAY the simulated data sets can be pureposefully post-analysed. During the decision procedure the user has almost no restrictions. So he can use his own experience and special knowledge of the system.

Optimization can be applied in a number of ways. A variety of features that insures flexibility gives the user a complete control over the resolution of the results. Possible methods to find the optimum parameter set are:

- optimization of a single target function by sorting (rise or fall)
- optimization of multiple target functions by:
  - ▸ setting bounderies (filter)
  - ▸ generating the Pareto Set

and sensible combinations of these methods can rapidly increase the efficiency of the decision procedure.

Optimizaion by generation of the Pareto Set:
The comprehensive estimation of multiple criteria often causes the improvement of one criterion by simultaneous deterioration of another criterion. In this case the result of optimization can be a compromise, the so-called Pareto Set.

For the example of the Monte-Carlo-Analysis of the electrical d.c. drive the Pareto Set was found with the following adjustments:
(1) min. rotation in motor operation ➡ Maximum
(2) rise time in motor operation ➡ Minimum
(3) settling time in motor operation ➡ Minimum
(4) integral squared error ➡ Minimum
and Day-Optim has generated these solutions:

Data Set	P-Part	I-Part	Integral Squ. Error
23	6.8206	13.29	1.8356
48	6.2893	13.2	1.858
60	6.7171	11.13	1.8585
73	1.0867	14.43	4.2728
87	0.9418	13.66	4.7895
88	0.4036	11.74	9.4902
99	0.3484	11.14	11.024

Chart 2: Pareto Set

The decision on an optimum parameter set was caused by a minimum integral squared error (set 23). The transient behavior of the system before and after the optimization is shown in the Figures 2 and 3 below.

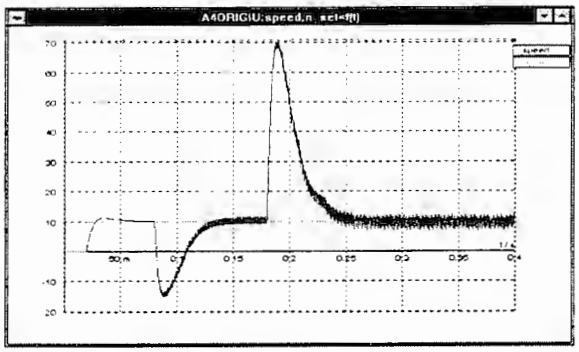


Figure 2: Transient Behavior of Rotation before Optimization

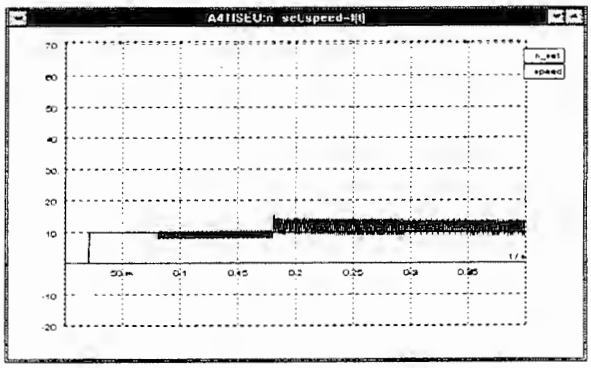


Figure 3: Transient Behavior of Rotation after Optimization

2.4. Analysis of Trends by Varying Counter Torque

A Trend Analysis was made to show limits of system loadability. The optimum controller parameters (P-Part=6.8206 and I-Part=13.29) has been inserted in the simulation model. The counter torque was evenly increased in the range of 5mNm to 80mNm.

Target functions to be obtained were (motor operation):

- maximum and minimum rotation
- time peroid of acceptable battery power and
- the integral squared error .

Results of the Trend Analysis with increasing counter torque are presented in Figure 4, 5 and 6.

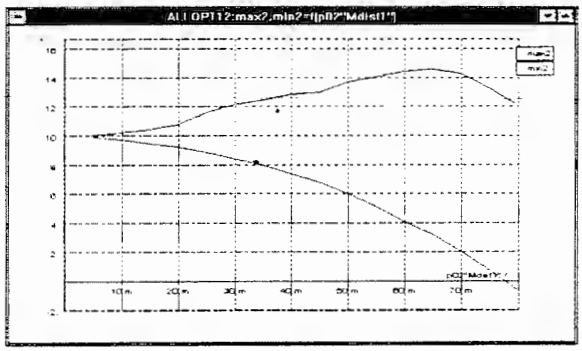


Figure 4: Maximum and Minimum of Rotation

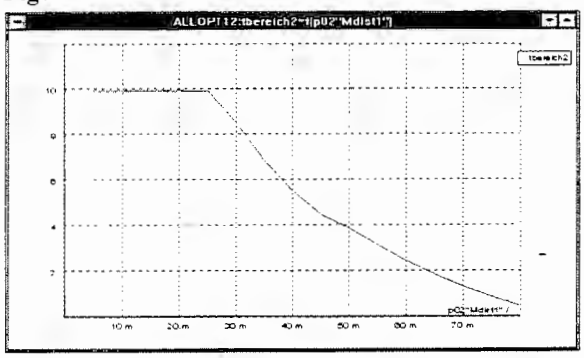


Figure 5: Time Period of Acceptable Battery Power

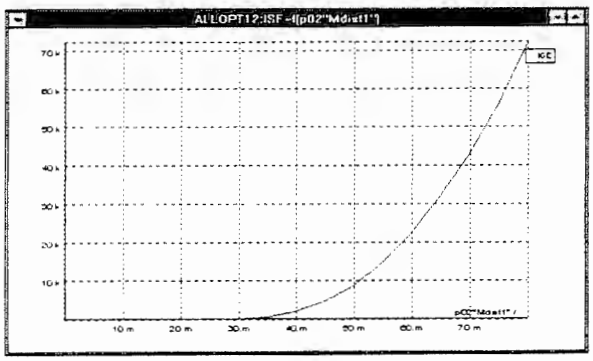


Figure 6: Integral Squared Error

3. Conclusion

By means of optimization of a d.c. drive various optimization techniques has been introducted. The peculiarity of this analysis is the intelligence of evaluation. The engineer is now able interactively to make a decisive, process-oriented classification based on simulated data sets.

New Developments in the Parallel Simulation System mosis

G.Schuster, F.Breitenecker
ARGE Simulation News, Technical University of Vienna, Wiedner Hauptstr. 8-10, A-1040 Wien, Austria
E-Mail: guenter@osiris.tuwien.ac.at

Abstract

This contribution describes the newest developments in the modular CSSL simulation system mosis that were implemented within the last year. After a short description of this simulation language the new features will be described in detail. Those developments include:

- graphical user interfaces under Microsoft Windows and the XWindow system
- handling of implicit models
- object-oriented definition of simulation models
- other new developments under development including graphical modeller

1. Description of mosis

mosis (modular simulation system) is a CSSL-type compiling simulation language on a "C"-basis with special features for modular development and parallelization of simulation models. The first version of mosis was implemented as a part of the dissertation of G.Schuster which was meant as a practical implementation of the *Model Interconnection Concept*. This is a concept for describing bigger simulation models from several independent models that are connected via special data links; those models can even be simulated on different processors in a multiprocessor network (*parallelization*). During this work, many features of other simulation languages were included in this system, so with the first release in 1994 it had most features of commercial systems and by now it is a full all-purpose high-performance simulation language which can be used on parallel systems (including workstation clusters) and singleprocessor systems as well.
mosis is distributed as *freeware*, i.e. it may be used and copied by anyone with no restrictions (although copyrighted software), provided the package or parts of it (or executable programs created from it) are not sold for commercial profit (except a copying fee). The system (complete packages, source code and documentation) can be downloaded from the TU Vienna simulation server <ftp://simserv.tuwien.ac.at>. Special services for commercial customers are offered by the software company "Advanced Technical Software GmbH" which is located at Vienna, Austria.

1.1 The Model Interconnection Concept

The *Model Interconnection Concept* (MIC) is the theoretical background of the parallelization strategy, modular development and external system connection in mosis. According to the definition in this concept, a model can be one of the following:

- an algorithmic description of a dynamic system (also with discrete model parts)
- a test model (a model which is only used for validation of another model)
- a predefined function or a constant or
- even an interface to an external - "real" - system or another simulation package.

A complete simulation model can be defined by connecting several independent models. For this reason, the parts of the whole model can have several input and output signals for data exchange with other simulation models. Parts of the complete simulation can be devices existing in real ("Hardware in the Loop"), interaction with humans ("Man in the Loop") or can be performed by other specialized simulation systems (Simulator Coupling). The parts of a connected model can be simulated on a distributed processor network which gives a simple and efficient strategy for parallelization.

1.2 Existing Features

With the first version of mosis (which was released shortly after the completion of the dissertation of G.Schuster at the end of 1994), although an experimental simulation system, mosis already had most features of modern commercial CSSL languages with a remarkably higher simulation speed (even much faster on one single processor than other compiling simulation languages). The main features of mosis 1.0 are:

- High level model description language on a "C"-basis, including a preprocessor (superset of "C"-preprocessor), very fast compiler. The language follows the CSSL standard (sections) with the

possibility to insert any "C"-statements within the code; user-functions can be called from within the model. Names and functions follow the standard ANSI-C name conventions.
- It includes a user-friendly make utility that performs all steps (translation, compilation and linkage) to translate the models and user-functions to the run time system.
- Several models can be defined and linked simultaneously to the run time system.
- Dynamic model instancing at run time level; several instances of one model are possible (e.g. on different processors in a network).
- High-level interpreter language (similar to "C") for simple functions, loops, graphics at run time level; hardware independent: all commands are exactly the same either used on a multiprocessor UNIX-network or on a simple PC.
- All simulation are done in the background; during a run it is always possible to enter user commands (start, stop, display, ...)
- All simulations are done in double precision (IEEE 64 Bit)
- Easy interfacing with other systems: Connection to other simulators or to external systems (hardware in the loop, man in the loop) by accessing interface routines from the simulation program.
- Many experimentation commands for optimization, graphical output etc.; easy way to implement new commands in "C".

The following hardware platforms have already been supported in version 1.0:

- MS-DOS under Borland C/C++ 3.1
- MS-DOS 32Bit (386+) under Watcom C/C++ 10.0
- UNIX systems under PVM (parallel virtual machine) as a communication system.

In the following sections the new features of the recent release will be described:

2. Graphical User Interfaces

In the previous implementations mosis had quite a simple user interfaces: The DOS versions (Borland, Watcom/32 Bit) had a command line input (with history lists etc.) and several graph screens that could be viewed by pressing a special function key (or by an interpreter command), while the UNIX version only had a very basic possibility to edit the interpreter commands and an interface to the "gnuplot" program to draw curves on a graphical interface (e.g X Window). By the new version of mosis, versions for two of the most commonly used graphical user interfaces (GUIs) have been implemented: For the PCs a version for Microsoft Windows (32 bit) is now supplied; for the UNIX world the XWindow implementation of the mosis simulation system has been completed. Both versions are quite similar to use (except the visual appearance dependent from the windowing system) and offer the following features:

- A command window consisting of a menu bar, a output area (where all output from the simulation system is written to) and an input line. The text in the input line may be edited (plus cut/copy/paste); a history list is also supplied. The text in the output area can also be transferred to other application by "Copy"; a certain number of lines off top may be recorded (user-selectable).
- Many commonly used commands can be entered by menu commands with the mouse.
- up to 32 simultaneous graph windows; they can be resized, moved and iconized.

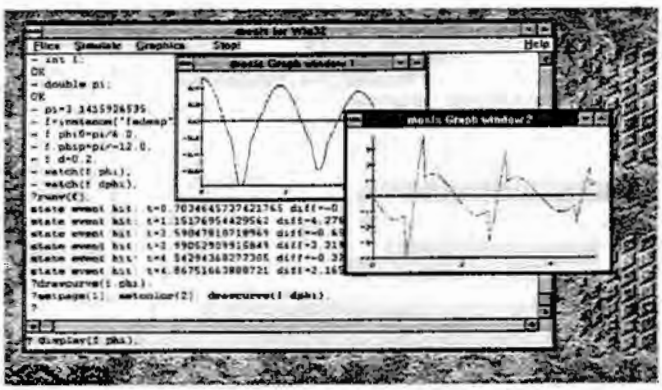


Fig. 1 shows a simple screenshot of the MS-Windows version with two open graph windows. For future compatibility and higher efficiency, the *Win32s* system has been used. With this and other special techniques, (in contrary to other simulation systems) the Windows version is not at all slower than the DOS-version (which itself runs extremely fast).

In the MS-Windows version also a graphical make utility is provided which corresponds to an IDE (integrated development environment) in other programming languages. By using this, the files that should be translated to the final simulation program can be entered in special fields, they can be edited by pressing a single button or removed from the make list. Moreover, the make process can be performed automatically and the resulting file can be executed by a single button press.

3. Implicit Model Definition

This new feature of mosis refers to the fact that not all problems in simulation technique can be easily described from ordinary differential equations (ODEs). In the recent years, much emphasis has been laid on the research of differential algebraic equations (DAEs). In the ODE approach, a dynamical change in a state is described as

$$\dot{x} = f(t,x)$$

while in the DAE approach the dynamics are described as

$$G(t,x,\dot{x}) = 0$$

(f, G are functions). Each ODE can also described as a DAE ($f(t,x) - \dot{x} = 0$), but not vice versa. There are several algorithms that can handle this type of problem (by a solution finder and iterations), usually in connection with other features like stepsize control, automatic handling of stiff systems etc. The probably most famous algorithm of this type is DASSL by L.Petzold which has also been integrated in the new mosis simulation system. In order to be able to use this algorithm, new syntactical structures had to be implemented in the mosis model description language:
A new type of variables - the algebraic variables - have been added to the existing variable types; they can be distinguished from the "normal" state variables. Those variables are declared with the algebraic keyword, e.g.:

```
algebraic x,y;
state z;
```

which determines that the variables x and y are part of the vector of algebraic variables, while z is inserted in the state vector. The DAEs are defined in a new special block - "`implicit`" - which exists beside the ODE block "`derivative`". Within the implicit block, the DAEs are defined in the form of

```
0 = G(t,x,x`);
```

(where G is replaced by the appropriate function of t, x - or another algebraic variable - and its first derivative x` (mosis-syntax for $\dot{x}$)). If an equation can be written as an ODE but it should be nevertheless calculated toghether with the other DAEs, it can be written in the known mosis syntax for ordinary differential equations (of first or higher order):

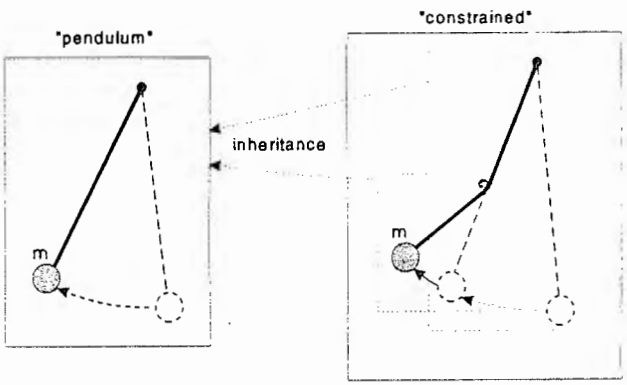
```
x` = f(t,x);
```

where f has to be replaced by the actual function describing the dynamic behaviour.

With this construction, it is possible to have two sections that describe the dynamics of a system that are calculated by two different integrators - one with DASSL and the other one with another integration algorithm, e.g. a Runge-Kutta one - which could increase the simulation speed by only having to calculate the implicit parts by the implicit algorithm and the other parts by a much faster one.

4. Object-Oriented Modelling

The new version of mosis incorporates an object oriented modelling approach which gives the user much higher flexibility and maintainability in writing simulation models and a clear model structure. The main key of this feature is the possibility to inherit methods of other simulation models; even of several different models if e.g. the described system consists of several parts that have been already described as mosis-models. By use of the `inherit` statement it is possible to use all methods of the given model name. A simple example for this would be a constrained pendulum that may hit a pin positioned at a certain angle. For this model the description of a general pendulum (without the constraints) can be used by inheriting the previously defined methods: The constrained pendulum can be described as a general pendulum that switches to another pendulum with a smaller length when the pin is hit. When the smaller pendulum swings back and reaches the switching angle, it turns again back to the first one. For higher performance, those two general models (with different lengths) can occupy the same address space (as only one model is active at a time) by use of the



shared keyword. Switching between those models (in an appropriate discrete section) can be done by issuing the activate command. The main structure of a mosis/2 model would be:

```
model constrained()
{
  inherit shared pendel LongPend, ShortPend;
  // use methods of "model pendel"

  LongPend:dynamic     // append to dynamic section in "LongPend"
  { sevent(LongPend.phi-Dphi,_NEG_,SwitchShort); }
  ShortPend:dynamic
  { sevent(ShortPend.phi-Dphi,_POS_,SwitchLong); }
  discrete SwitchLong  { activate LongPend; }
  discrete SwitchShort { activate ShortPend; }
}
```

With the "LongPend:dynamic" statement it is possible to append certain statements to the dynamic section of the pendulum model called *LongPend*. In the corresponding *discrete*-blocks it can be switched between the short and the long pendulum model (as the two models are used alternatively). It would be also possible to prevent the system to execute a section of the inherited model, to replace it or to add special features (shown above).

5. Other Implementations

Besides the described extensions and new implementations, other programs and modules have either been written or are under development. Those are:

- a graphical modeller tool (multi-hierarchy) under MS-Windows (Win32s) that produces mosis-code, but also simulation code for other common simulation systems.
- graphical environment for the run time system (model connections, instancing, ...)
- a bond graph modelling tool
- special graphical output modules: 3-dimensional graphics, animation etc.
- a real-time system with interface routines to several A/D-D/A boards
- interface-connections basing on the MIC to other simulation systems via PVM
- other algorithms for integration (DAEs,. ODEs), frequency analysis, optimization, visualisation etc.

Some of them will probably be presented at the congress. Several other enhancements and new features will be developed in the future.
Information about the mosis-system (and the complete package) can be obtained at the congress (exhibition area) or at the following internet-addresses:

<http://eurosim.tuwien.ac.at> World Wide Web-Suite of ARGESIM/EUROSIM
<ftp://simserv.tuwien.ac.at> Simulation software server at the TU Vienna
E-Mail: guenter@osiris.tuwien.ac.at

References

Cellier E.F.: Continuous System Modeling, Springer, New York 1991
Schuster G.: Definition and Implementation of a Model Interconnection Concept in Continuous Simulation, Dissertation, TU Wien, 1994
Breitenecker F.: Comparison of Simulation Software: Constrained Pendulum, Simulation News Europe 7/March 1993

Environment for the simulation and design of control systems

Borut Zupančič, Marko Klopčič

Faculty of Electrical and Computer Engineering
University of Ljubljana, Tržaška 25
61000 Ljubljana, SLOVENIA
Email: borut.zupancic@fer.uni-lj.si

KEY WORDS: digital simulation, simulation language, control, real time systems, fuzzy control

1. Introduction

The domain of modelling, simulation, analysis and design of control systems is known as a discipline which beside the signal processing area perhaps most drastically influenced the development of simulation hardware and software in the past. Simulation language SIMCOS, which was in its original concept general purpose CSSL type language, was expanded by numerous features, that make it extremely applicable in control design and simulation area. Although the user interface, the graphics and some other things can not be compared with some commercial simulation tools, there are some implemented ideas, which indicate important advantages in the field of control systems' design and simulation. The fuzzy controller design and simulation environment as the last extension shows the applicability also for the most advanced control approaches.

2. Basic principles of simulation language SIMCOS

The simulation language SIMCOS [1, 2, 3, 4] is a CSSL - type equation oriented language that was developed at the Faculty of Electrical and Computer Engineering in Ljubljana. It works as a compiler. The model, which is coded in CSSL syntax or is described by a graphical block oriented simulation scheme, is processed by the compiler into FORTRAN modules and a model data base. The FORTRAN modules are further processed by a FORTRAN compiler and subsequently linked with appropriate libraries into an executable simulation program. The supervisor program automatically handles all the above procedures and is able, together with a highly interactive user interface, not only to simulate the model but also to perform simple experiments (e.g. change of model constants, output specifications, function generators' breakpoints) and complex ones (e.g. parameter studies, optimization, linearization, ...).

All basic possibilities for a model description are available: The built in nonlinearities and signals, multidimensional function generators, the procedural block, basic and advanced univariable and multivariable dynamical precompiled submodels (transfer functions, state space, lead-lag, ...) in continuous and discrete forms as well as different controllers with integral wind up protection are implemented. Using a special pre-processor a hierarchical modelling is also available. Well tested and previously developed programs for particular components can be reused as submodels in higher hierarchical levels. Because of sorting algorithm an user need not to take care about the order of a model statements. Many integration algorithms (single step, multistep, low order, high order, extrapolation, stiff, ...) cover all commonly appeared numerical problems giving the simulation tool an appropriate numerical robustness.

For modelling, simulation and design of control systems the following implemented features are particularly important:

- extension of continuous simulation concept to the discrete one by so called generalized simulation operator,
- extension of the language with harware-in-the loop and man-in-the loop simulation possibilities implementing real time possibilities,
- extension of the simulation towards a complex experimentation system,
- possibilities for modular and hierarchical modelling of control systems,
- extension with design and simulation possibilities of fuzzy control systems.

3. Generalized simulation operator

Sometimes it is very efficient to simulate or to experiment with a model in discrete or in difference equation form. Such models can be easily derived from continuous linear models using well-known discretization transformations. Especially in experiments that demand many simulation runs (e.g. optimization) a lot of time can be saved using the mentioned approach.

The genaralization of the simulation was achieved by so called generalized simulation operator i.e. the integrator as the basic operator of continuous simulation was expanded to the generalized simulation operator which can be by the proper parametrization either the integrator or the discrete delay as the basic operator of discrete (difference equation) simulation.

4. Real time possibilities

Conventional computer systems are very limited for real time applications because of insufficient hardware capabilities, redundant and too complex software equipment, inefficient programming and poor input-output facilities. As SIMCOS is general purpose simulation language which works on conventional hardware (PC) with standard operating system (MS-DOS) and low cost process interface (PCI 20000), the simulation speed is of course limited. Using SIMCOS in hybrid configuration some other limitations occur so the speed is even more limited. But the use in slower simulations (or experimentations) is quite satisfactory.

The real time simulation was enabled by the following modifications and extensions:

- the generalized simulation algorithm was synchronized with real time,
- simple CPU low cost simulation algorithms were included (Euler integration algorithm, discrete simulation),
- the communication with real signals was enabled.

Program modules for data acquisition and transmission are specific for particular hardware equipment (PCI 20000). Because the language is opened to the user, it is not difficult to implement a specific hardware.

5. Experimental environment

It is well known, that simulation run is only one useful experiment in different studies. Modern tools include also more complex experiments as optimization, linearization, parameter study etc. Some experimental implementation concepts are known from the literature but compiler orientation of SIMCOS does not give the possibility to fully implement them. For this case we introduce three experimental sections (INITIAL, TERMINAL and DYNAMIC). Each model is compiled for in advance selected experiment. An experiment is treated as an execution of a particular method (simulation, optimization, linearization, parameter study, user experiment) on a model. Each higher level experiment allows also the execution of the pure simulation.

Optimization is a very useful experiment in many applications. It is particularly effective for the design of PID control schemes. An user should define a criterion function, constraints and parameters to be optimized. Linearization as another implemented experiment is also very important for control systems' design. It is used to obtain the linearized model (in state space form) from a non-linear one. It is well known, that most of design methods are limited to linear models. Parameter study is an experiment in which the influence of a parameter to simulation results can be efficiently studied. All these experiments are realized with pre-programmed INITIAL and TERMINAL sections. User experiment enables a user to create his own experiment.

6. Possibilities for modular and hierarchical modelling of control systems

Beside elementary modelling constracts, which are available in SIMCOS library, higher level modelling constructs. which can be grouped into two different classes, are available:

- precompiled objects and
- hierarchical sub models

Precompiled objects are used as standard (frequently) used blocks, written in Fortran language. The user can use a template to program his own blocks. To realise the concept of precompiled objects an efficient function analyser and a self configuration procedure of a simulation model during run time were implemented. Function analyser recognises each model function by the aid of an installation precompilled objects file. This file has an information, whether the precompiled object is statical or dynamical, continuous or discrete, univariable or multivariable, with or without delay attribute. Each function, which is called with inputs and parameters and returns outputs, is specifically pre-processed enabling appropriate sorting algorithm and efficient and correct calculations in run time. Self configuration enables that all states, derivatives and predictions are appropriately organised into common vectors of states, derivatives and predictions. On this way advanced univariable and multivariable dynamical control systems' constructs (transfer functions, state space, lead-lag, PID controller with integral wind up protection, fuzzy controller,...) in continuous and discrete forms are implemented.

Using a special pre-processor a hierarchical modelling is also available. Well tested and previously developed programs for particular components can be reused as sub models in higher hierarchical levels. The hierarchical pre-processor is in some way similar to MACRO concept in simulation languages. The internal variables of sub models are exchanged with working names so that each hierarchical model can be reused several times. Because of sorting algorithm the user needs not to take care about the order of model statements.

7. Extension with design and simulation possibilities of fuzzy control systems.

The extension with fuzzy design and simulation was implemented in two steps. As the optimization as the most important approach in conjunction with conventional PID schemes was not suitable for the design of fuzzy controllers, we decided to design a special graphically oriented design tool. Then the fuzzy controller as a precompiled block was included in the simulation library on the way, that it can automatically handle the results of the design phase. To decrease the development time, an existing fuzzy package, which contains also designing tools written in C language

was adopted. The adaptation was accomplished by an interface between fuzzy controller algorithm written in C too and simulation language SIMCOS.

Fuzzy controller designing tool Fuzzy controller design is more complex than classical PID controller design, because there are many parameters, including shape and position of segments. Because of the nature of the fuzzy controller settings, graphical representation is very helpful and simplifies design. The designing tool provides facilities for fuzzy controller design in graphical environment. User can choose among three standard shapes of segments and user defined segment type. There are four fuzzy set operations available. Rules can be defined by table or explicitly as `IF...THEN...` statements. The number of input and output variables and segments is limited by available memory. Designed fuzzy controller settings are written to a file in two available formats - ASCII and binary. Program described in this paper is written in C++. Objets were used to create the user interface.

Fuzzy controller implementation in SIMCOS Fuzzy controller implemented in SIMCOS can have two input and two output variables. The range of input variables is from -1 to +1, while output variables are given in the range from -1000 to +1000. The user has to make appropriate scaling before and after calling the fuzzy controller. If user does not define fuzzy controller properly and there exist such combinations of input variables that output fuzzy set is empty, this error is detected at runtime (during simulation). To provide user the information about the error an error flag is set and returned to SIMCOS.

Structure of fuzzy controller data The fuzzy controller settings are written to a file in binary form prepared for direct use by the fuzzy controller program. This file contains the following data:

- number of input variables,
- number of output variables,
- number of membership functions for each variable,
- number of rules,
- membership functions defined as integer arrays,
- centres of gravity and areas of output membership functions and
- rules.

Implementation of the fuzzy controller algorithm The fuzzy controller algorithm consists of three basic steps: fuzzification, calculation of output variables and defuzzification. To achieve fast fuzzification, the fuzzy membership functions are written to file as arrays of membership functions. The fuzzification is therefore performed simply by picking membership values from an array. In the next step the fuzzy decision logic is executed. The result of each rule is the centre of gravity for appropriate output segment and its area. Defuzzification is the last step in fuzzy algorithm. In this program the centre of gravity of all output segments is calculated. Its position represents the output value of the fuzzy controller.

8. Example: Design and simulation of fuzzy control of a laboratory pneumatic plant.

The model which describes the laboratory pneumatic plant is given by the transfer function

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \qquad \omega = 9.325 \qquad \zeta = 2.238$$

Y(s) ... pressure in the vessel *[mA]*
U(s) ... control signal to the pneumatic valve *[mA]*

Using the fuzzy controller designing tool and simulation approach the fuzzy controller was set experimentally. The truth table is shown in Table 1.

Table 1. Truth table (E ... control error, dE ... error derivative, mid ... middle, ze ... zero)

		E				
		-big	-mid	ze	+mid	+big
dE	-big	-big	-big	-mid	ze	+mid
	-mid	-big	-big	-mid	+mid	+big
	ze	-big	-mid	ze	+mid	+big
	+mid	-big	-mid	+mid	+big	+big
	+big	-mid	ze	+mid	+big	+big

Fig. 1,2 and 3 show the membership functions for control error and its derivative (E and dE) and for output controller variable U. The appropriate SIMCOS program is shown in Fig. 4.

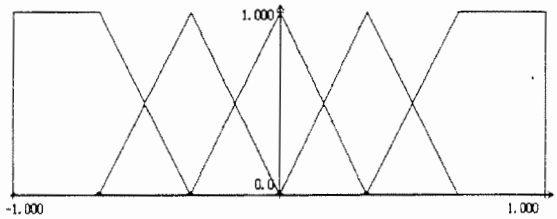


Fig. 1. Membership functions for input variable E

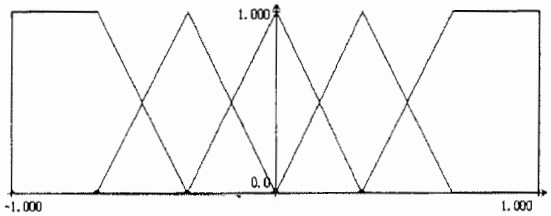


Fig.2. Membership functions for input variable dE

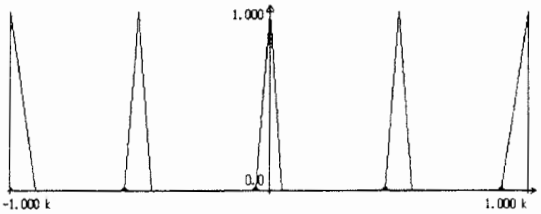


Fig. 3. Membership functions for output variable U

```
PROGRAM FUZZY
ARRAY E(2), U(2)

" reference signal
REF = STEP(T, 2.)

" error - the first input to the fuzzy controller
E(1) = REF - Y
" error derivative - the second input to the fuzzy controller
CONSTANT Tf = 0.01, Kd = 0.1
E(2)=DIFF(E(1),Kd,Tf)

" fuzzy controller - scaling of input and output variables is required
PROCEDURAL (U(1), U(2) = E(1), E(2))
    E(2) = E(2) * 5.0
    call FUZZY(U, E, FERR)
    U(1) = U(1) * 10.
END
UI = INTEG(U(1), 0.)

" second order pneumatic system
CONSTANT omega=9.325, zeta=2.238
Y=SECOR(UI, zeta,omega)

" simulation run duration and error flag
CONSTANT TFIN=5, FERR = 0
CINTERVAL CI = 0.01
TERMT T.GT.TFIN
PREPAR      Y, E(1), E(2), REF, U(1), UI, FERR
OUTPUT 20, Y, E(1), E(2), REF, U(1), UI, FERR
END
```

Fig.4. The fuzzy model of control system in SIMCOS is the following

Fig. 5 depicts the reference (REF), the controlled variable (Y) and the controller output variable UI.

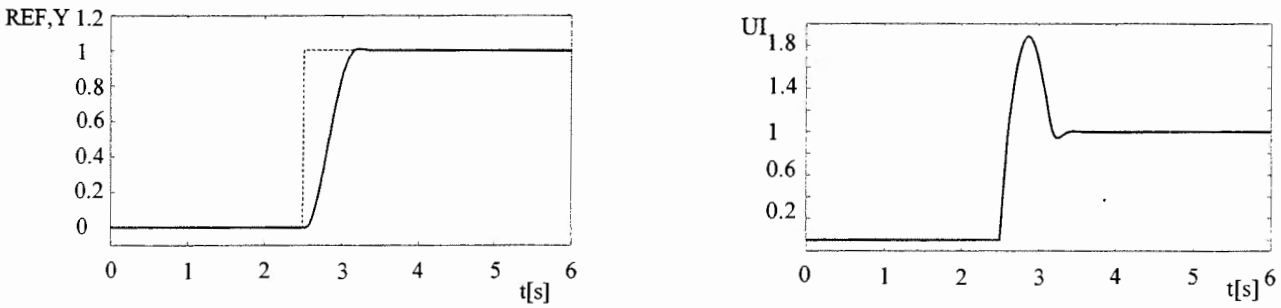


Fig. 5. Reference (REF), the controlled variable (Y) and the controller output variable (UI).

9. Conclusions

All parts of described environment were realized and tested in off line simulation studies as well as in hardware-in-the-loop experiments using real time facility. Although SIMCOS is in its nature general purpose simulation tool, the described extensions make it much more powerful and useful for the design and simulation of control systems. Fuzzy implementation as the last included feature shows, that the simulation concept is wide and open enough also for the inclusion of most advantages artificial intelligence based methods. SIMCOS is successfully used also in education process in conjunction with modelling, simulation, continuous and discrete control.

10. References

[1] Matko, D., B. Zupančič, R. Karba: *Simulation and Modelling of Continuous Systems - A Case Study Approach*, Prentice Hall Int., London, UK, 1992.

[2] Jekl, M., B. Zupančič, R. Karba: Hybrid simulation system based on digital simulation language SIMCOS, *Proceedings of the 8th symposium Simulationstechnik*, Berlin, Germany, 1993, pp. 171-74

[3] Zupančič, B., D. Matko, R. Karba, M. Šega: SIMCOS - digital simulation language with hybrid capabilities, *Proceedings of the 4th symposium Simulationstechnik*, Zürich, Switzerland, 1987, pp 205-12.

[4] Zupančič, B., D. Matko, R. Karba, S. Divjak, M. Jekl: The Concept of a new Hybrid Experimenting System, *Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, Ireland, 1991, pp. 1318-19.

Distributed Object-Oriented Simulation Environment: An implementation of Time Warp using PVM

Roberto Beraldi and Libero Nigro

Dipartimento di Elettronica, Informatica e Sistemistica
Universita' della Calabria - I-87036 Rende (CS) - Italy
beraldi@telecom.deis.unical.it, l.nigro@unical.it

Abstract

This paper introduces DOSE, a development framework which facilitates the construction of a simulation model on the basis of library light-weight objects and a customizable scheduling control structure. The simulation model can be targeted for execution either on a uniprocessor machine or on a networked computing environment abstracted by PVM. As a case study, the performance of a Time Warp implementation is given.

1. Introduction

The work described in this paper aims at experimenting with parallel or distributed discrete-event simulation[1], which has the potential of allowing the exploitation of the computing resources of modern multiprocessors as well as networked computer systems. The objective is to provide an effective support to the analysis of complex and computational-intensive applications like broadband communication systems[5,6,7], wireless cellular radio networks[8] and so forth. Mathematical models used to asses the performance of such systems could require excessive simplification in the hypotheses. Simulation models are thus used to this end. The current state of ongoing work is represented by DOSE - Distributed Object-oriented Simulation Environment, a set of software tools which ensure a smooth transition from sequential to distributed simulation. A DOSE simulation model is achieved by selecting, instantiating, configuring and linking objects belonging to a reusable library specialised in a given class of application problems. The simulation model can be targeted for execution on a single processor or it can be decomposed into a collection of interacting subsystems to be allocated onto a networked system. DOSE is characterised by its openness and extensibility. New library objects, relevant to specific application areas, can be made ready to use to DOSE users. The runtime support of DOSE relies on an asynchronous light-weight architecture, DART[2,3], which has been developed for real-time applications[4]. DART is based on active objects and customizable scheduling through programming. It can easily be integrated with different communication systems and protocols. DART is truly object-oriented and permits reusable and extendible libraries of active objects to be established via inheritance. DART is available in C++ and Oberon-2. DOSE prototype implementations are in current use on concentrated DOS/Win or UNIX platforms for the simulation of communication systems[5], and in particular ATM (Asynchronous Transfer Mode) based Broadband ISDN systems. In order to meet the high speedups required for the simulation of more complex systems, e.g., mobile wireless radio networks[9], a distributed realisation of DOSE has been implemented on a standard network of UNIX workstations, possibly heterogeneous, abstracted by PVM[10] (Parallel Virtual Machine). PVM is responsible for configuration in-the-large management and inter-subsystems communication and synchronisation. Distributed simulation control centres on the Time Warp mechanisms[11], especially developed to work with DART and PVM. The implementation language is C++. The paper is organised in two parts. First the DOSE environment is summarised, by focusing on the configuration issues and the DART-based simulation level. Then the paper discusses the performance of the realised Time Warp implementation. Finally, some directions of future work are given.

2. An overview of DOSE

DOSE consists of a set of tools which help the end-user in making and executing an actual simulation model on the basis of reusable and extensible *library objects*. As shown in Figure 1, DOSE tools split logically into two classes: those responsible for configuration management (*configurer* and *analyser*) and those specifically dealing with the actualisation of a configuration (*spawner* and *actualiser*) on a networked environment handled by PVM[11].

2.1. Configuration level

A *configurer tool* is provided which supports the operations of an application-expert, e.g., through a friendly graphical user interface, for selecting, instantiating, initialising, connecting and allocating the *basic simulation objects* (**bso**s) relevant to a chosen problem domain. The configurer tool works on an *abstract representation* of a simulation model, where the basic building blocks are objects equipped with *input* and *output typed ports*. A port type essentially captures a set of messages which can be transmitted or received through the port. The type of an input port coincides with the type of the object to which the port belongs. The type of an output port is defined as the type of an object an input port of which can be linked to the output port. The number of output ports (*out-degree*) of an object is dependent on the "require part" of the object, i.e., the number and kind of communications required by the object behavior (see also later in this paper). The number of input ports (*in-degree*) is amenable to topology requirements. It can be defined at the object creation time. To facilitate the configuration of complex systems, the configurer enables *composite objects*, named *subsystems*, to be established. A subsystem is the unit of programming in-the-large and it is associated to a physical processor. A whole configuration is stored on a *configuration file*, which can also be directly generated by a standard editor. The configuration file is structured in three sections: a *create*, a *connect* and an *allocate* section. The create section lists all the bso instances involved in a simulation model. Object instantiation is expressed by the CREATE command:

CREATE OBJ <obj_id> **OF TYPE** <obj_type> **WITH**
<in-degree,value><attrib,value>... <attrib,value>

which states that a new object instance of obj_type has to be created. The new instance will be referred to by the unique identifier obj_id. Moreover, the in-degree and the internal attributes of the new object will be initialised according to the specified attribute-value list. The connect section of the configuration file captures the topology issues of the simulation model, and consists of a list of CONNECT commands, each being structured as in the following:

CONNECT OUTPORT <port_id> **OF** <obj_id> **TO**
INPORT <port_id> **OF** <obj_id>

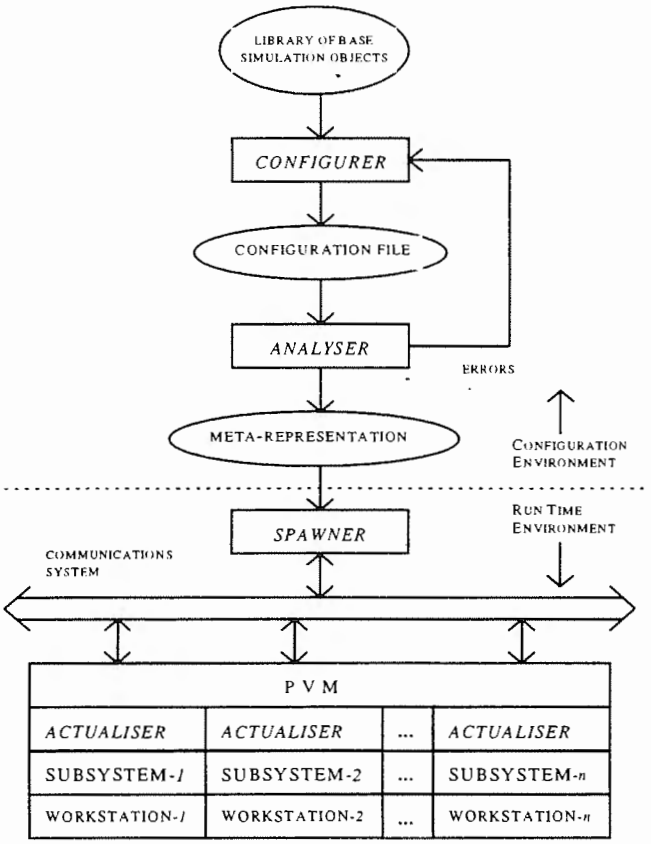


Figure 1: Dose framework.

Finally, the allocate section of the configuration file specifies the number of subsystems into which the simulation model is split, the bso instances allocated to each subsystem and the mapping of sybsystems to processors. An example of an ALLOCATE command follows:

ALLOCATE OBJECTS 0-15;17;20-25 **TO** <subsystem_id>
ON <host_name>

Object instances are named according to their unique identifiers. A single ALLOCATE command is sufficient to define the object organisation of a single subsystem. Furthermore, the following version of the ALLOCATE command:

ALLOCATE ALL TO <subsystem_id> **ON** <host_name>

can be used when the simulation model is targeted for execution on an uniprocessor machine. The *analyser* tool is responsible for checking the correctness (i.e., completeness and type-safe connections of the topology) of the configuration contained in a configuration file. Toward this, a *meta-representation* of the simulation model is built consisting of *object descriptors* where, for instance, the initialisation data are organised into suitable data structures. Object descriptors have been designed so as to be more easily translatable into the corresponding runtime representations within the different address spaces of a networked environment.

2.2. Simulation level

A *Spawner* tool is responsible for actualising a meta-representation of a simulation and of initiating the execution of the resultant distributed simulation system. The actual steps for carrying into execution a distributed simulation are summarised in the following. First all the subsystems are activated on the associated workstations of a networked system, then they are fed with the relevant object descriptors. Within each separated subsystem, an *actualiser* tool is in charge of converting object descriptors into equivalent runtime objects and actual connections. Finally, the spawner sends a message to the various subsystems for starting the simulation work. The runtime representation of a DOSE simulation model relies on an asynchronous, distributed, object-oriented architecture, DART[2,3], which has been developed for real-time applications[4]. DART is a minimal model whose design mirrors the *spirit of Oberon*[12]: "make it as simple as possible". More specifically, DART centres on the concept of light-weight active objects, which communicate one with another by message passing. The dynamic behaviour (*lifecycle*) of an object is modelled as a finite state machine which evolves through a succession of states[13]. Transition from one state to another is triggered by the arrival of an expected event (message). To each state is associated an action which is executed each time the state is entered. Action execution cannot be suspended nor pre-empted. *Action is the unit of scheduling*. Synchronisation relies upon communication and object lifecycle. A fundamental object of a DART subsystem is a programmable *scheduler* (minimal executive) which transparently collects all the message transmissions among the objects and dispatches them according to a given control strategy[14,15]. To exemplify, a basic DART scheduler rests on an *event-queue* for message buffering and an *event-loop* at each iteration of which a message is extracted from the event-queue and dispatched to its destination object. A scheduler can be customised to reason upon a *real* or *virtual clock*. As a consequence it can deliver a discrete-event simulation framework in a straightforward way. Time intervals are programmed by setting *timers*. A timer holds a *fire time* and a *timeout message* which is sent to its destination object at the timer expire time. All the timers waiting to fire, are stored into a ranked *timer-queue*, with the most imminent timer to fire at the beginning. When no more "instantaneous" messages exist on the event-queue to be dispatched, the basic simulation scheduler forces firing of the most imminent timer and adjusts accordingly the virtual clock. Flexibility of scheduler replacement is a key factor to effectively support a uniprocessor or networked execution of a DOSE simulation model. In the former case it is sufficient to adopt the basic simulation scheduler, where in the latter a distributed, e.g. based on Time Warp[11] or on a conservative strategy[1], is mandatory. DART has been implemented in C++ and Oberon-2. Active objects are equipped with event handlers[3] which allow for type-safe event communications. Objects can be organised, through inheritance, into hierarchies of reusable components. DART addresses *interoperability* by allowing different subsystems to be developed using possibly different programming languages.

3. Performance of a Time Warp implementation

Time Warp[11] (TW) is an optimistic asynchronous strategy supporting general purpose parallel discrete-event simulations (PDESs). Significant successes have been achieved from its use across a wide-range of applications[1]. Our case study concerns an implementation of TW in DOSE using a standard, networked environment managed by PVM. Key features of the realised prototype are: (a) it is a minimal and efficient kernel, whose behavior can be tuned to the application at hand; (b) it has been achieved according to the DOSE/DART approach in standard C++, hence is portable. A whole simulation model is split into a collection of subsystems (*logical processes*). Concurrency is ensured by mapping the subsystems on PVM tasks, by allocating tasks one per processor (*true parallelism*) and by executing object actions concurrently within subsystems (*apparent parallelism*). Each subsystem essentially carries a sequential simulation. Its progress in the simulated time is represented by its Local Virtual Time (LVT). Current time for the whole simulation is contained in the Global Virtual Time (GVT)

variable. Inter-subsystem messages are marked with two timestamps: the *sending time ts* (equal to the LVT of the sender sybsystem) and the *receiving time tr*, i.e. the time at which the message-event is to be handled by the receiver subsystem. Besides the event-queue and the timer-queue, the TW scheduler uses also an *input-queue* and an *output-queue*. The input-queue stores copies of received external messages, ranked by their *tr*. A receiver subsystem translates an external message into an equivalent timer whose fire time coincides with the message *tr*. The output-queue serves to annotate *undo messages* (see below) directed to subsystems to which external messages have been sent. LVTs are allowed to take different values as a consequence of load conditions and/or processor speed. Therefore it is possible to receive an external message whose *tr* is strictly below LVT (a *straggler*) giving rise to the basic TW *clock synchronisation problem*. TW needs frequent state saving in order to cope with *causality errors*, i.e., processing stragglers. A state version at a simulated time T consists essentially of copies of the object statues and of the timer-queue, performed just before any action at time T is executed.

3.1. Virtual clock synchronisation

On the arrival of a *straggler* with receiving time T, a subsystem must *roll back* to time T cancelling, on itself and on partner subsystems, all the effects of previous erroneous computation carried at times T'>T. Depending on the availability of state versions, roll back can cause re-installation of a state to a simulated time T''<=T. After that, forward computation is repeated until straggler time (*coasting phase*) then continuing by processing straggler and finally going further into future. To undo effects on other subsystems, standard TW uses *anti-messages*. An anti-message m^- either annihilates the corresponding positive-message m^+ stored in the input-queue of a subsystem S, or can trigger a roll back in S in the case m^+ has been already processed. Due the overhead normally accompanying PVM messages (e.g., pack/unpack operations for transmitting/receiving message arguments on the network of heterogeneous UNIX workstations), and considering the unpredictable latency of message transmissions on a shared LAN which can facilitate the spreading of erroneous computation[16] TW-DOSE uses "aggressive cancellation" (all the anti-messages are immediately sent for any previously sent positive-message with *ts*>T) but minimising the number of anti-messages. A single **undo** message, carrying an identification of the sending subsystem S and a timestamp T, is transmitted to every partner subsystem P for undoing effects caused by computation of all the messages sent by S to P with *ts*>T.

3.2. GVT update

A critical issue of every TW implementation is GVT management. The value of GVT is, at any real time *t**, the minimum among the LVTs and the timestamps of events that have not yet been processed at *t**, including the undo messages, of all the subsystems. Obviously, no event with a timestamp smaller than GVT will be ever rolled back, so storage associated with state versions less than GVT can be reclaimed for reuse (*fossile collection*). A frequent GVT update is clearly desirable both to conserve memory and for an early detection of simulation end. However, GVT-update operations are time-consuming and then can degrade performance due to the overhead required by the involved synchronisation. GVT is periodically updated by a subsystem acting as the *GVT manager*. GVT manager normally receives a request for a GVT-update by a subsystem which reaches a critical threshold in memory usage, measured in terms of the number of state versions and of the input-queue length. After that, the GVT-manager sends a control message to all the schedulers advising about the GVT-update. Every scheduler then stops dispatching message-events and enters a *barrier* synchronisation. All in-transit messages arriving during barrier, are actually received by the relevant subsystem at the end of the barrier and are scheduled on the timer-queue. An undo message is left pending until termination of GVT-update. The GVT-update protocol continues with each TW-scheduler proposing a value for the GVT to the GVT manager, which selects the global minimum among the received proposals and, finally, broadcasts it to all the subsystems. On receiving the new GVT value, every subsystem first reclaims fossile collection and then resumes its normal dispatching activity. The adopted strategy for GVT updates proves effective in coping also with expensive PVM memory requirements.

3.3. Simulation experiments

Performance measures of the realised DOSE TW prototype have been derived using a simple closed queueing network benchmark (CQNB), consisting of N fully connected switches. Each switch contains Q FIFO queues connected in tandem. Every queue models a non pre-emptive service station. A job arriving at a switch is served sequentially by the Q servers and is thereafter routed to one of N neighbouring switches (including itself) with equal probability. The service time of a job at a server is exponentially distributed, with minimum service time greater than zero. Initially, every switch is assigned J jobs. The parallel computing architecture used for the experiments, consists of four dedicated workstations (a Sun SPARC 1, a Sun SPARC IPC, a Sun SPARC 5 and a HP9000 Apollo) connected by a 10 Mb/s Ethernet LAN shared with other users. Experiments were conducted at times when the network was "light loaded" by other users in order to minimise their effects on the measurements. The CQNB is split into four subsystems, one per processor. At configuration time each subsystem is assigned a number of switches with associated values of Q and J. Base simulation objects involved are a *queue* with server and a *router*. Simulation results are collected on separate files, one per subsystem. Since the parallel virtual machine used is composed of workstations with sensible different computing speeds, a first concern was static *load balancing*. A CQNB with N=16, Q=20, J=32, an average service time of 3 time units, was preliminarily simulated for 1500 time units, both on a uniprocessor machine and on the four processor architecture. Different load configurations (see Table 1), labelled from L1 to L5, were considered. A load configuration specifies the number of switches allocated to each separated processor. Table 2 shows the effects of different load configurations on the completion time (i.e., the elapsed real time) and the "efficiency", the latter being defined as the ratio between the number of messages processed by the sequential simulator and the number of messages processed by the parallel simulator. As one can see from Table 2, L2 and L4 load configurations are almost equal in the completion time and efficiency. Figure 2 shows details about the distribution of the computing load among the four processors. In particular, the percentage of the messages processed by each processor with respect to the total number of messages handled by the parallel simulator is portrayed.

	Sparc1	IPC	HP9000	Sparc5
L1	4	4	4	4
L2	3	3	5	5
L3	3	3	6	4
L4	3	3	4	6
L5	2	4	5	5

Table 1: Load configurations

	L1	L2	L3	L4	L5
C.Time(sec)	104	75	81	71	86
Efficiency	0.42	0.60	0.55	0.60	0.51

Table 2: Static load balance measurements

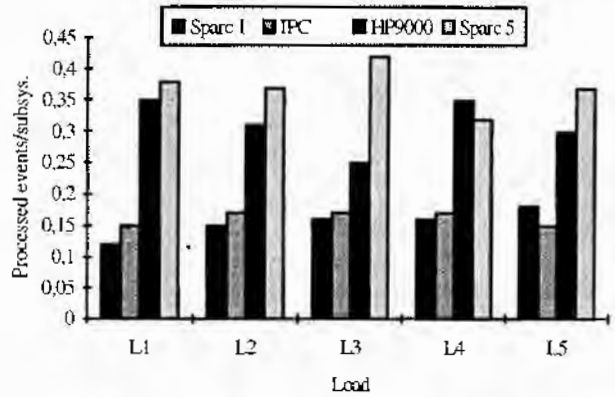


Figure 2: Processed events versus load configurations

The results in Table 2 and Figure 2 have been achieved using a state saving rate (SSR) of 6 and a maximum number of state versions (MNSV) equal to 12. A value *s* for SSR means that the subsystem status is copied on the occurrence of the *s*-th LVT change since last saved state version. The value of MNSV represents the maximum number of state versions existing before asking for a GVT-update. Figure 3 shows the completion time versus MNSV for different SSRs. It was used to select a pair <SSR, MNSV> for speedup evaluation. Figure 4 portrays the speedup as a function of Q keeping constants N=16 and J=32. Each data point has been achieved as the average of 5 runs. It is worthy of note that the sequential simulation uses the basic simulation scheduler, not TW, running on the fastest machine (Sun SPARC 5).

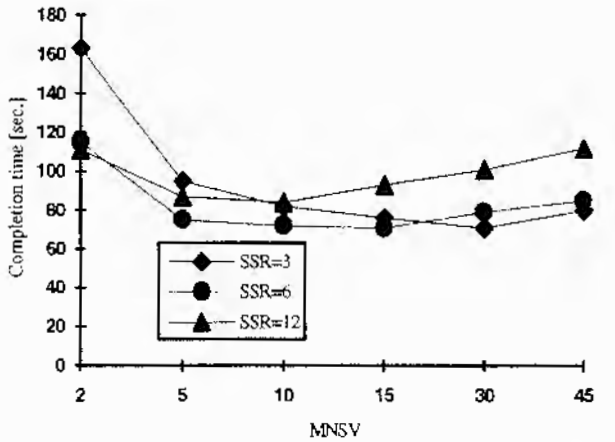


Figure 3: Completion time versus MNSVs

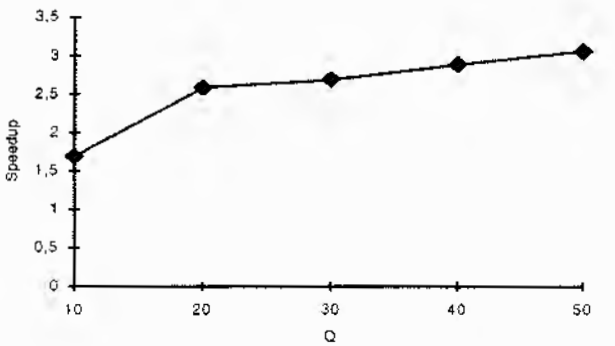


Figure 4: Speedup for CQNB (N=16,J=32,Q varying,4 processors)

4. Conclusions

Due to their low cost and high computing power, networked environments are today widespread. Their usefulness in general parallel computing is enhanced by recent high-speed communication means (e.g., based on ATM or FDDI). The effective use of a networked system is facilitated by a standard software layer like PVM which simplifies programming by abstracting details about protocols, data conversion operations, synchronisation concerns aad so forth. All of this is of obvious interest in supporting parallel discrete-event simulation (PDES), provided adequate development tools are available. With respect to concurrent applications, PDESs add the difficulties of coping with causality errors which in turn introduce hard synchronisatinn problems. This paper presents DOSE - Distributed Object-oriented Simulation Environment - which has a flexible configuration tool allowing the application expert to configure a simulation model on the basis of reusable objects, and of spawning the model for execution either on a uniprocessor machine (Dos/Win or Unix platform), or on a networked context managed by PVM. The following factors are noteworthy: (i) DOSE relies on an asynchronous software model based on light-weight objects in the absence of pre-emption and of context-switch. operations; (ii) DOSE has a programmable runtime support (scheduler); (iii) DOSE is programmed in the popular language C++. The paper furnishes performance data of a developed Time Warp mechanism. The ongoing activity is geared toward: (a) optimising the Time Warp implementation, e.g., by introducing an incremental object state saving technique; (b) completing the realisation of library objects for the parallel simulation of complex systems like mobile wireless networks.

Acknowledgements

The authors wish to thank B. Kirk and S. Marano for their helpful comments during preliminary drafts of this paper.

References

1. Fujimoto R.M. Parallel Discrete Event Simulation, CACM, 33(10), 1990, 31-53
2. Kirk B. and Nigro L. A Distributed Architecture for Real-Time, in Oberon-2, in *Advances in Modular Languages*, P. Shulthess (ed.), Universitatsverlag ulm GmbH, 1994, pp. 325-366.
3. Nigro L. A real-time architecture based on Shlaer-Mellor object lifecycles, *JOOP*, 8(1), March-April 1995, 20-31
4. Nigro L. and Tisato F. An object-based architecture for real-time applications, *in Advances in Object-Oriented Software Engineering*, Mandrioli D. and Meyer B. (eds.), Prentice-Hall, 1992.
5. *IEEE Communication Magazine*, issue on Network level modelling and simulation, March 1994
6. Beraldi R., Iera A., Marano S. and Sestito R. A per-burst Combined Policing Mechanism for ATM Networks in the case of ON/OFF and Loss Sensitive Sources: The DTAT, in *Proc. of Third Int. Conf. on Computer Comm. and Networks (ICCCN)*, San Francisco (CA), 1994.
7. Beraldi R., Iera A., Marano S., Ruggi R., Buffer Statistical Multiplexing for Bursty data Calls in ATM Networks, in *Proc. of the IEEE Singapore ICCS*, Singapore, 14-18 Nov. 1994.
8. Beraldi R., Iera A., Marano S. and Salerno P. A New dynamic Reservation Multiple Access Protocol for Supporting Multimedia Traffic in Third Generation Cellular Systems, *in Proc. of the IEEE Singapore ICCS*, 14-18 Nov., 1994.
9. Edbom G. and Stjernholm P. Simulation of Cellular Networks, in *Proc. of the IEEE/VTS, 44th Vehicular Technology Conference*, 1994.
10. Geist A. et al. *PVM:Parallel Virtual Machine - A Users Guide and Tutorial for Networked Parallel Computing*, (The MIT Press, 1994)
11. Jefferson D. et al. Distributed simulation and the Time Warp operating system, in *Proc. of the 11th ACM Symposium on Operating System Principles*, 1987, 77-93.
12. Wirth N. A plea for lean software, *IEEE Computer*, **28**(2), pp. 64-68, February 1995.
13. Shlaer S. and Mellor S.J. *Object lifecycles: modeling the world in states*, (Yourdon Press, 1992)
14. Nigro L. Control extensions in C++, *JOOP*, 6(9), pp. 37-47, Feb. 1994.
15. Nigro L. and Veneziano G. Control abstractions in Modula-2: a case study using advanced backtracking, Informatica, 18(2), June 1994, 229-243
16. Carothers C.D., Fujimoto R.M., England P. Effects of Communication Overheads on Time Warp Performance: An Experimental Study, *8th Workshop on Parallel and Distributed Simulation*, Edinburgh, July 1994, 118-125.

Simulation-Based Optimization of Production Systems by SENSIM

Elke Stief
Institute for Manufacturing Automation and Production Systems
Prof. Dr.-Ing. K. Feldmann
University Erlangen-Nuremberg, Germany
91058 Erlangen, Egerlandstr. 7-9, Germany
E-mail: stief@faps.uni-erlangen.de

ABSTRACT

This paper describes the tool SENSIM which supports the optimization of production systems by simulation using design of experiment (DOE) methods. The tool has been developed by the Institute for Manufacturing Automation and Production Systems. The paper is structured as followed: First the implementation of DOE methods and their connection to the modular system SIMULATION is described. The performance of the DOE methods implemented in the system will then be demonstrated using a case study.

INTRODUCTION

Due to prevailing buyer's markets and to a growing dynamic of innovations, the economic situation for companies has changed fundamentally in recent years. This has led to a variety of changes and to an increasing complexity of production systems in the companies.

In order to support the planning of production facilities simulation has gained general recognition as a suitable tool. Simulation can for example be used to evaluate alternatives for improvement according to identified negative trends. In addition it is possible to obtain predictions of future system behaviour.

In general, a large number of experiments have to be carried out to optimize plant performance. With regard to the growing dynamic in the planning stage of both production and product it is necessary to increase the effectiveness of simulation. This can be supported by SENSIM, a tool which enables the user to work according to a fixed pattern implying the required experiments.

When creating a tool the implemented two issues are important: first the methods used and second the procedure of experimentation. Therefore, the tool SENSIM is based on a scientific approach. The design of experiments is part of it.

DESIGN OF EXPERIMENT

The DOE methods are considered to be the proprietary tool for improvement and optimization of processes and products according to given objectives. Information on how to improve an industrial process can be obtained by going through a cycle of minor modifications, or variants, of the current process and repeatedly running these in sequence. The methods can be used as a way to avoid large numbers of time consuming experiments.

Factorial design is a particularly useful method which may be used to investigate either quantitative or qualitative variables. The following Factorial Designs have been implemented in the tool:

- 2n-Factorial Designs.
- 2n-p-Factorial Designs.
- The Taguchi Method.
- The Shainin Method.

The 2n factorial experiments provide information on all possible two factor interactions if the experiment is designed properly. The Taguchi Method and the Shainin Method are special factorial designs.

IMPLEMENTATION OF SENSIM

The module SENSIM has been developed to automatically support the process of model optimization by DOE methods. It enables the user to specify the optimization problem to which the optimization process is to be applied.

Due to the scientific approach the tool is based on the following requirements which have been taken into consideration:

1. Recognition of an existing problem.
2. Formulation of the problem.
3. Agreeing on factors and levels to be used in the experiment.
4. Specifying the variables to be included.
5. Definition of the inference space of the problem.
6. Random selection of the experimental units.
7. Assignment of treatments to the experimental units.
8. Outline of the analysis corresponding to the design before the data are taken.
9. Collection of the data
10. Analysis of the data.
11. Conclusions.
12. Implementation.

The design of experiment methods are dealt with in the steps 5, 6 and 7. The sections prior to these three describe the preparation of the design, and section 8 allows the experimenter to modify the design before putting it into operation.

The architecture of SENSIM consists of the following three building blocks

- REVERS,
- a graphical user–interface and
- an interface for coupling the module with the modelling tool SIMULATION.

The building–block REVERS provides the four kinds of factorial designs mentioned above. REVERS runs the implemented program in accordance to the input received from both the design selected by the user and the results of the experiments received from the system SIMULATION.

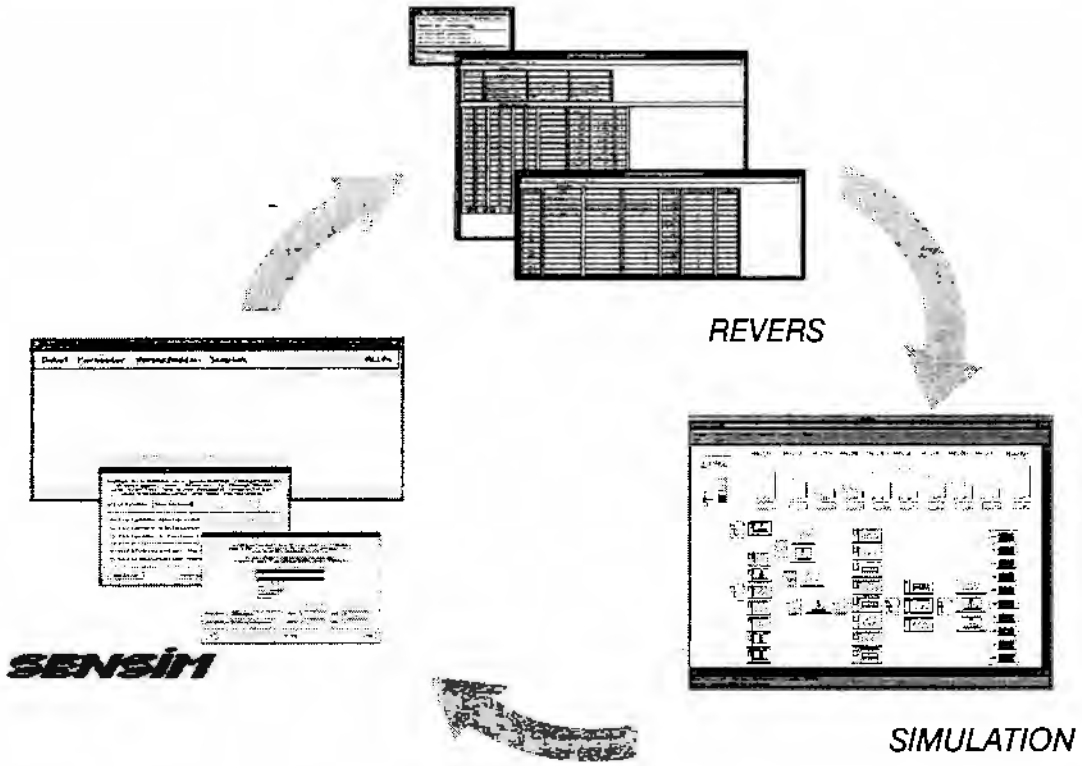


Fig 1 Interplay between SENSIM and the related tools REVERS and SIMULATION

When working with the module, the user has to handle a distinct number of input parameters as well as to cope with a high quantity of output data, generated by the tool SIMULATION. In order to get both an easy survey and a user friendly structure the user–interface has been divided into an input part, a control part and an output part.

The *input–part* enables the user to choose a factorial design. In addition to this the variables to be examined can be selected and a limited number of possible levels for each

of these variables and the objective function can be identified. This objective function however may be any performance parameter that can easily be received as a result of the experiments. The system offers defaults of useful performance parameters of the objective functions and gives support when fixing the variables and their possible levels. The input is passed forward to the building-block REVERS. Informations about relevant variables and components of the simulation model are needed for generation of batches and preparation of the simulation experiments. The task of the *output-part* is to visualize the results and to give recommendations for optimized settings for the parameters of the simulation model. On the one hand the *control part* of the interface enables the user to stop the running optimization process and to restart the process if necessary and on the other hand to skip to a help mode, which supports the user to handle the tool.

The interface couples the tool SENSIM with the modular system SIMULATION in order to run experiments.

CASE STUDY

The experience of using the tool SENSIM is presented by a case study. In this example the tool SENSIM and the simulation was used to optimize the behaviour of a production system. The model of the production system consists of about 20 components and approximately 30 different jobs are modelled. The object of the study was to obtain initial experience on the optimization of model-behaviour using simulation and DOE-Methods. Key concerns were to gain knowledge about the time involved by running the tool and about the validity of the results.

REFERENCES

Box, George, and Draper, Norman R., *Evolutionary Operation*, John Wiley & Sons, Inc. Petersen,1969.

Harro, *Grundlagen der Statistik und der Statistischen Versuchsplanung*, Ecomed Verlagsgesellschaft mbH, 1991.

Anderson, Virgil L. and McLean, Robert A., *Design of Experiments - A Realistic Approach*, Marcel Dekker, Inc., New York, 1972.

Abels, S.,*Modellierung und Optimierung von Montageanlagen*, Ph.D.,Fertigungstechnik - Erlangen; 37,Hanser,1993.

Abels, S., and Stief, E., *Knowledge-Based Optimatization of Simulation Studies*, in Proceedings of the Conference on Modelling and Simulation 1994 ESM'94, Barcelona, Spain, pp.186-190,1994.

`omsim2maple` – A translation utility for OmSim simulation code

James A. Sørlie

S3–Automatic Control; Royal Institute of Technology; S-100 44 Stockholm
`http://www.s3.kth.se/~sorliej/`

Abstract: The purpose of this contribution is to present a tool for translation of OmSim simulation code into Maple, as well as to illustrate some potentials of symbolic manipulation when applied in modeling and identification. OmSim simulation code is effectively a sorted set of modeling equations. The reported utility translates these equations into Maple statements, which then allows symbolic reduction and manipulation of the equations. A principle motivation in this work has been the desire to use analytical derivatives in parameter estimation. Furthermore, Maple provides facilities for optimized C-language code generation. To illustrate its use, the translator and symbolic manipulation are used in deriving an extended Kalman filter and generating numerically optimized C-language subroutines. Currently, most commercially available code-generators preclude the use of symbolic manipulation.

Keywords: software, symbolic manipulation, code generation, extended Kalman filter, grey-box modeling

1 Introduction

An ongoing project at the S3 department in Stockholm is the development of methodologies for Grey-Box Modeling and Identification – an approach to building mathematical models based on both mechanistic knowledge of the process, as well as empirical measurement data; *cf.* [1]. An identification tool-kit, IdKit, has been developed [2] consisting of application independent identification software. In using the tool-kit, a model builder provides model structures, *i.e.* modeling equations, in the form of C-language subroutines. The original objective behind the immediate project was to facilitate the definition and management of model structures. Through time, the focus has changed as a result of an awakening to the potentials of symbolic manipulation.

Omola and OmSim are respectively an object-oriented modeling language and a simulation environment which have been developed by the Computer-Aided Control Engineering (CACE) group in Lund [3–6]. The modeling language provides a high-level *acausal* data description of a dynamic object. The simulation environment provides a model compiler as well as state-of-the-art facilities for numerical simulation. Due to its developmental status, OmSim currently lacks export facilities of a low-level data description, *i.e.* the equations of the compiled model. This is unfortunate since such a modeling language is ideal for managing multiple realizations of a given model. This is of particular interest when investigating alternative model structures via a grey-box methodology; *cf.* [7, 8].

It would thus appear that in its present state, OmSim is a closed software system. However, as part of its debugging facilities, the software does provide a means of "dumping" the compiled simulation program code. This program code itself resembles the uninteresting primitives of a reverse-Polish-notation calculator. Of interest are the corresponding modeling equations which are interspersed throughout this debugging output as program comments. These equations appear in computational order and, rather serendipitously, the grammatical syntax of the Omola equations is strikingly reminiscent to that of the computer algebra program Maple [9]. This last observation, combined with knowledge of the configurable code generation capabilities in Maple [10], led us to investigate an interface between the two program packages.

2 Program Description

In order to keep matters simple, our objective has been, at least initially, to translate only continuous-time elements of the Omola language. This reflects the fact that, aside from transport delays, much of the mechanistic information used in grey-box modeling is most naturally expressed as continuous-time relations.

Two versions of the utility have been developed. The tool was initially prototyped as a Unix shell script in order to verify the feasibility of the idea. A second version has been developed using C++ and GNU's `flex` and `bison` utilities. This later version's sophistication was deemed necessary in order

```
1   #!/usr/local/bin/bash -norc
2   # FILE:  omsim-filter.sh -- Convert an OmSim log-file to Maple statements.
3   # $Id: omsim-filter.sh,v 1.7 1995/03/08 17:06:30 sorlie$
4   #-----------------------------------------------------------------
5   # Step 1:  Convert the OmSim simulation code to Maple definitions.
6   #-----------------------------------------------------------------
7   # Delete from the log-file all lines except the symbolic equations.
8   # These lines have a program-code line-number followed by two percent
9   # symbols, as well as containging an equals symbol.
10  #-----------------------------------------------------------------
11  cat $*  | sed -n '/^.*%%.*=.*$/p' |\
12  #-----------------------------------------------------------------
13  # Remove the line number, percent symbols and leading whitespace.
14  # Convert the end-of-line character to a Maple command terminator (:).
15  # Convert any equalities (=) into Maple bindings (:=).
16  # Convert the period in Omola names to an underscore.
17  # Finally, comment out equations designating the OmSim inputs.
18  # This makes the script-output suitable as "include" input for Maple.
19  #-----------------------------------------------------------------
20  sed 's/^.*%%[    ]*\(.*\)$/\1\:/;
21       s/ = / := /g;
22       s/\.\([^0-9]\)/_\1/g;
23       s/^\(.*\)\*input\*.*\..*\,[ ]*\(.*\)\,.*$//'       |\
24  #-----------------------------------------------------------------
25  # Convert conditional assignment statements to piecewise function calls.
26  #-----------------------------------------------------------------
27  sed  's/^\(.*:=.*\) if \(.*\) then \(.*\) else \(.*\):$/\1 \
28  piecewise\(2,\2,\3,\4\):/g' | cat
29  #-----------------------------------------------------------------
30  # End of processing Step 1.
31  #-----------------------------------------------------------------
32  # Step 2:  Convert the OmSim parameter dump to Maple definitions.
33  #-----------------------------------------------------------------
34  # Find any assignment (:=) or relation (=) statements in the log file,
35  # excluding those in the comments to the simulation code.  The result
36  # should be the parameter mapping and propagation of parameters
37  # in computational order, found in the Parameter debug output.
38  #-----------------------------------------------------------------
39  cat $*  | egrep -v '%%' | egrep '[ :]= ' |\
40  #-----------------------------------------------------------------
41  # Convert the end-of-line character to a Maple command terminator (:).
42  # Convert any equalities (=) into Maple bindings (:=).
43  # Convert the period in Omola names to an underscore.
44  #-----------------------------------------------------------------
45  sed 's/^\(.*\)$/\1\:/g;
46       s/ = / := /g;
47       s/\.\([^0-9]\)/_\1/g;'     | cat
48  #-----------------------------------------------------------------
49  # End of processing Step 2.
50  #-----------------------------------------------------------------
```

Figure 1: Listing of `omsim2maple` shell script.

to deal more directly with the complexity of the Omola grammar.

To illustrate the basic mechanisms of the translator, and its sheer simplicity, we include a listing of the shell-script prototype in Figure 1. Worth noting is the translation of conditional statements using the Maple `piecewise` function (lines 27–28). Notice however that the shell script does not account for the possibility of nested conditional statements in Omola. This problem, along with the translation of relation-operator statements and various matrix operations are examples of the complexity which warranted the development of the C++ version of the translator.

2.1 Creating the Translator Input

The input to the translator is a debugging "dump" which the user creates using the OmSim simulator and its logging facilities. Upon successful instantiation of an Omola model (see [5]), one turns logging on and then selects the simulator's Debugging → Parameter part and Debugging → Simulation code menu selections. (It is assumed that the log-file is initially empty.) At this point, one may run the translator, specifying the log-file as input and redirecting the standard output to a file. This output is suitable for loading

```
1   SingleBlockModel ISA IdKit::SignalModel WITH
2   signals:
3     Wv, U,  Wy    ISA IdKit::SignalVector;        % Inputs.
4     Y             ISA IdKit::SignalVector;        % Outputs.
5     Xy            ISA IdKit::SignalVector;        % States.
6     dXy_dt        ISA IdKit::SignalVector;        % State derivatives.
7   parameters:
8     P             ISA IdKit::ParameterMatrix;     % Physical parameters.
9   END;
```

Figure 2: A "wrapper" class in Omola.

into Maple using Maple's File → Include ... menu selection.

2.2 Parameter and Signal Names

Once loaded into Maple, all variable signals and unbound parameters are available for symbolic manipulation. Logically, their names in the translated equations are fully qualified according to the hierarchical class structure of the Omola model. To facilitate processing in Maple, it proves useful to encapsulate the Omola model in a "wrapper" class which defines both parameter and signal name mappings. The wrapper class provides top-level (meaning unqualified) references to the parameters and signals of interest. Note that this encapsulation mechanism places no restrictions on the hierarchic structure of the actual Omola model.

An example of a wrapper class suitable for use with IdKit is shown in Figure 2. In using this wrapper, the translator's output will include Maple assignments for the signals `Wv`, `U`, `Wy`, *etc.*, and the unbound parameters `P`. It is these short names which one references Maple. The use of the wrapper class will be clarified in the following section's example.

3 Symbolic Manipulation

We intend to illustrate the use of the translator along with some of the potentials of symbolic manipulation of the model equations. In [3, 11], symbolic manipulation is discussed in the context of model compilation and index reduction, *i.e.* reduction of differential-algebraic equations to ordinary differential equations. In [12], symbolic manipulation is used in linearization and the *future* potentials of code generation are mentioned. In what follows, we assume we have run the `omsim2maple` translator on the OmSim log-file and loaded the model equations into Maple.

3.1 Extended Kalman Filter

Here we develop the equations for a continuous-discrete extended Kalman filter (EKF); *cf.* [13]. The system for this example is based on a drum-boiler/turbine model of a power plant [2]. Space does not permit going into the details of the model. Suffice it to say that the plant consists of two

```
1   IdKitDrumBoiler ISA IdKit::SingleBlockModel WITH
2     dboil ISA DrumBoilerSimulation;
3   signal_model:
4     Wv.dim := 2;  U.dim  := 2;  Wy.dim := 2;
5     Xy.dim := 4;  Y.dim  := 2;  dXy_dt.dim := 4;
6
7     Wv = dboil.StateDist.Wv;
8     Xy = [dboil.StateDist.Xv; dboil.Process.Xz];
9     dXy_dt = [dboil.StateDist.Xv'; dboil.Process.Xz'];
10    U  = dboil.Process.U;
11    Wy = dboil.Dbserver.Wy;
12    Y  = dboil.Dbserver.Y;
13  parameter_matrix:
14    P.rowdim := 7;  P.coldim := 2;
15    dboil.Process.Tau     := trans( P[1,1..2] );
16    dboil.Process.Alpha4  := P[2,1];
17    dboil.Process.Xz0     := trans( P[3,1..2] );
18    dboil.Process.Uss     := trans( P[4,1..2] );
19    dboil.StateDist.sigma := trans( P[5,1..2] );
20    dboil.Observer.sigma  := trans( P[6,1..2] );
21    dboil.Process.NonLin  := int( P[7,1] );
22  END;
```

Figure 3: Use of the Omola wrapper class.

```
1   A := jacobian(  dXy_dt, [ x[j]  $ j=1..vectdim(Xy) ] ):
2   G := jacobian( dXy_dt, [ wv[j] $ j=1..vectdim(Wv) ] ):
3
4   # Time-update of the estimated state.
5   convert(dXy_dt,list):
6   dx_hat_dt := convert(subs({'wv[j]=0' $ j=1..vectdim(Wv)},"),vector);
7
8   # Time-update of the prediction error covariance.
9   Rx := array( symmetric, 1..vectdim(Xy), 1..vectdim(Xy) ):
10  dRx_dt := evalm( A &* Rx  +  Rx &* transpose(A)  + G &* transpose(G) ):
```

Figure 4: Time-update equations in Maple.

state equations augmented by a linear shaping filter which models two state disturbances as a diffusion process. The stochastic inputs Wv and Wy are normalized continuous and discrete Gaussian white noise, respectively.

$$\frac{d}{dt}x(t,x,u,wv,P) = \begin{bmatrix} 0.002wv_1 \\ 0.003wv_2 \\ \frac{4.20u_1 - 0.99x_3u_2}{P_{1,1}} + P_{5,1}x_1 \\ \frac{0.2112x_3u_2 - 0.9600x_4}{P_{1,2}} + P_{5,2}x_2 \end{bmatrix}$$

$$y(t_k,x,u,,wy,P) = \begin{bmatrix} 1.056P_{2,1}x_3u_2 \\ +0.32\,(15 - 15P_{2,1})\,x_4 \\ +P_{6,1}wy_1 \\ x_3 + P_{6,2}wy_2 \end{bmatrix}, k \in N$$

A simulation model of these equations has been programmed in Omola. The signal model "wrapper" for the model is shown in Figure 3. Note in line 21 the mapping for an Omola "realization parameter" [4, Ch.8]. This parameter may be used to switch between nonlinear and linearized realizations.

Time-update equations: The calculation of the time-update equations in Maple is shown in Figure 4. The prediction error covariance is reduced *symbolically*, rather than performing the matrix multiplications numerically. Note that here we use continuously evaluated Jacobians rather than constant Jacobians (*i.e.* evaluated instead using the results of the last measurement update) across the interval between measurements. Technically this is

```
1   C  := jacobian( Y, [x[j]  $ j=1..vectdim(Xy) ] ):
2   H  := jacobian( Y, [wy[j] $ j=1..vectdim(Wy) ] ):
3
4   # Innovation vector, i.e. the prediction error or residuals.
5   convert(Y,list):
6   convert(subs({'wy[j]=0' $ j=1..vectdim(Wy)},"),vector):
7   innovations := evalm( vector(['y[j]' $ j=1..vectdim(Y) ])-" );
8
9   # Covariance of the innovations, and its square root.
10  Ry := evalm( C &* Rx &* transpose(C) + H &* transpose(H) ):
11  S  := CholeskyFactorization(Ry):
12
13  # The Kalman gain matrix.
14  K  := evalm( Rx &* transpose(C) &* transpose(inverse(S)) ):
```

Figure 5: Meas.-update equations in Maple.

necessary because of the coupling of the differential equations for the state estimate and prediction error covariance. These couplings are important if the measurement sample interval is long with respect to the dynamics of the system, particularly in the case of stiff systems; *cf.* [2].

Measurement-update equations: Using the standard filter equations, the expression for the Kalman gain involves the inverse of a full matrix. Taken symbolically, this introduces complexity which leads enormous expressions for the measurement update equations. Even for a state vector of moderate dimension (four in this example), it is far more efficient to evaluate the matrix multiplications *numerically* rather than symbolically. An additional reason for deferring to numerical evaluation of the measurement update equations is the desire to include a mechanism for multi-rate sampling and missing measurements; *cf.* [14]. Also, use of the square-root information variant of the Kalman filter (as is shown in Figure 5 and implemented in IdKit), requires instead only the inverse of a triangular matrix.

3.2 Code Generation

Here we use symbolic manipulation for the conversion of the analytically derived results into an *optimized* form, suitable for numerical application. Figure 6 shows a template for automatic generation of a C-language subroutine, using the `macroC` share-ware package [10] in Maple. Figure 7 shows the output, the time-update equations for the extended Kalman filter. Note again the use of Maple's `piecewise` function in handling the Omola realization parameter.

Plainly, this is nothing more than an application of already existing tool. Our contribution is this: There are a number of simulation packages on the market, that provide "configurable" low-level code generation capabilities. All these packages are geared towards accelerating *model* simulation. However, equation export at this low a level precludes the further use of the equations in symbolic processing. As the EKF example illustrates, only

```
genC( [ [defineC,'piecewise(dummy,cond,expr1,expr2)','((cond)?(expr1):(expr2))'],
        [ commentC, '---------------------------------'],
        [ commentC, ' predictor time update equations '],
        [ commentC, '---------------------------------'],
        [ functionm, void, EKF_time_update,
          [ [ double, ['*x', '*u', '**Rx']],
            [ double, ['*dx_hat_dt'       ]],
            [ double, ['**dRx_dt'         ]]],
          [ [ commentC, ''                  ],
            [ commentC, ' state estimate '  ],
            [ matrixm, 'dx_hat_dt', dXy_hat_dt],
            [ commentC, ''                  ],
            [ commentC, ' erorr covariance ' ],
            [ matrixm, 'dRx_dt', dRx_dt     ]] ]] ):
```

Figure 6: A macroC template in Maple.

```
# define piecewise(dummy,cond,expr1,expr2) ((cond)?(expr1):(expr2))
/*---------------------------------*/
/* predictor time update equations */
/*---------------------------------*/
void EKF_time_update(x,u,Rx,dx_hat_dt,dRx_dt)
double *x,*u,**Rx;
double *dx_hat_dt;
double **dRx_dt;
  {
  double t1,t2,t4,t6,t10,t11,t15,t21,t23;
  double t7,t12,t14,t17,t19,t24,t28,t32,t35;

  /* state estimate */
  t1 = (P[6][0] != 0);
  t2 = x[2]*u[1];
  t4 = 1/P[0][0];
  t6 = P[4][0]*x[0];
  t10 = P[3][0]/P[3][1];
  t11 = x[2]-140/33*t10;
  t15 = u[1]-P[3][1];
  t21 = 1/P[0][1];
  t23 = P[4][1]*x[1];
  dx_hat_dt[0] = 0;
  dx_hat_dt[1] = 0;
  dx_hat_dt[2] = piecewise(2,t1,3/10*(14*u[0]-33/10*t2)*t4+t6,
    -99/100*P[3][1]*t4+t11+21/5*t4*(u[0]-P[3][0])-21/5*t10*t4*t15+t6);
  dx_hat_dt[3] = piecewise(2,t1,8/125*(33/10*t2-15*x[3])*t21+t23,
    34/161*P[3][1]*t21*t11-24/25*t21*(x[3]-14/15*P[3][0])+112/126*t10*t21*t15+t23);

  /* erorr covariance */
  t1 = (P[6][0] != 0);
  t2 = piecewise(1,t1,P[4][0],P[4][0]);
  t4 = 1/P[0][0];
  t7 = piecewise(1,t1,-99/100*u[1]*t4,-99/100*P[3][1]*t4);
  t12 = piecewise(1,t1,P[4][1],P[4][1]);
  t14 = 1/P[0][1];
  t17 = piecewise(1,t1,132/625*u[1]*t14,34/161*P[3][1]*t14);
  t19 = piecewise(1,t1,-24/25*t14,-24/25*t14);
  t21 = t2*Rx[0][3]+t7*Rx[2][3]+t12*Rx[1][2]+t17*Rx[2][2]+t19*Rx[2][3];
  t24 = t2*Rx[0][0]+t7*Rx[0][2];
  t28 = t12*Rx[1][1]+t17*Rx[1][2]+t19*Rx[1][3];
  t32 = t12*Rx[0][1]+t17*Rx[0][2]+t19*Rx[0][3];
  t35 = t2*Rx[0][1]+t7*Rx[1][2];
  dRx_dt[0][0] = 1/250000;
  dRx_dt[2][2] = 2*t2*Rx[0][2]+2*t7*Rx[2][2];
  dRx_dt[2][3] = t21;
  dRx_dt[0][2] = t24;
  dRx_dt[3][1] = t28;
  dRx_dt[0][3] = t32;
  dRx_dt[3][0] = t32;
  dRx_dt[2][1] = t35;
  dRx_dt[3][2] = t21;
  dRx_dt[1][2] = t35;
  dRx_dt[2][0] = t24;
  dRx_dt[1][3] = t28;
  dRx_dt[3][3] = 2*t12*Rx[1][3]+2*t17*Rx[2][3]+2*t19*Rx[3][3];
  dRx_dt[0][1] = 0;
  dRx_dt[1][0] = 0;
  dRx_dt[1][1] = 9/1000000;
  }
```

Figure 7: C-code for EKF time-updates.

through further processing is it possible to get truly optimized *predictor* simulation code. To our knowledge, none of the currently available packages support symbolic manipulation and, thus, preclude the possibility of simulating an EKF predictor.

4 Conclusions

We have presented a tool which translates a debugging "dump" from a modeling/simulation package into sorted modeling equations, suitable for symbolic manipulation. To demonstrate the tool's utility, an extended Kalman filter (EKF), which requires analytic derivatives, was derived through symbolic manipulation. Numerous commercial simulation softwares currently on the market now support automatic code generation, but not symbolic manipulation. As we have shown, today's computer algebra packages are more than capable of performing the final conversion to the low-level C-language code. We see the need for a choice in the level of equation export in order to fully realize the potentials of symbolic analysis and code generation. Consider, for example, implementing other *higher-order* nonlinear filter approximations [13].

References

[1] T. Bohlin. *Interactive System Identification: Prospects and Pitfalls.* Springer-Verlag, 1991.

[2] S. F. Graebe. *Theory and Implementation of Gray Box Identification.* Ph.D. thesis TRITA–REG–9006, Royal Institute of Technology, Stockholm, Sweden, 1990.

[3] S. E. Mattsson, M. Andersson, and K. J. Åström. Object-oriented modelling and simulation. In D. A. Linkens, editor, *CAD for Control Systems*, pages 31–69. Marcel Dekker, 1993.

[4] B. Nilsson. *Object-Oriented Modeling of Chemical Processes.* Ph.D. thesis TFRT–1041–SE, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1993.

[5] M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems.* Ph.D. thesis TFRT–1043–SE, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1994.

[6] M. Andersson. OmSim and Omola Tutorial and User's Manual. Technical Report TFRT–7504–SE, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1994. Manual and software available at URL: `http://control.lth.se/~cace/`.

[7] T. Bohlin. Derivation of a 'designer's guide' for interactive 'grey-box' identification of nonlinear stochastic objects. *Int. J. Control*, 59(6):1505–1524, 1994.

[8] J. A. Sørlie. On a general computer-aided methodology for building models of industrial processes. Technical Report IR–S3–REG–9504, Royal Institute of Technology, Stockholm, Sweden, 1995.

[9] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, 1991.

[10] P. Capolsini. *MacroC - génération de code C depuis Maple.* INRIA - Université de Nice, Laboratoire I3S, June 1992. In Maple, type `with(share): readshare(macroC,numerics):`.

[11] F. E. Cellier and H. Elmqvist. Automated formula manipulation supports object-oriented continuous system modeling. *IEEE Control Systems Magazine*, 13(2):28–38, 1993.

[12] P. J. Gawthrop. Symbolic modeling in control. In D. A. Linkens, editor, *CAD for Control Systems*, pages 127–146. Marcel Dekker, 1993.

[13] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*, volume 64 of *Mathematics in Science and Engineering Series.* Academic Press, 1970.

[14] T. Bohlin. Computer-Aided Grey-Box Identification. Technical Report TRITA–REG–8403, Royal Institute of Technology, Stockholm, Sweden, 1984.

Author Index

Achour H. 57
Amer-Yahia C. 57
Annino J. 29

Bausch-Gall I. 81
Beck K. 83
Beraldi R. 151
Breitenecker F. 143

Ciomaga A. 1
Crain R.C. 17

Drury C.E. 69

Frank M. 33

Gauthier J.S. 87
Geuder D. 37
Goldmanis A. 103
Goldynia J.W. 99

Haddab Y. 57
Halvorsen I. 119
Hasler Fl. 1
Helbing R. 65

Ingle R.M. 107
Ionescu Fl. 1

Jendges R. 9
Jonin G. 77

Kiffmeier U. 91
Kirchner H. 65
Klopčič M. 147
Klug F. 61
Knorr B. 139
Korn G.A. 135
Kronreif G. 25
Kundert K. 53
Kunovský J. 131

Laughery K.R. 41, 69
Lu Ch.J.J. 45

Marinits J.M. 99
Mazversitis A. 103
Merkuryev Y. 103
Merkuryeva G. 103
Mikulášek K. 131

Nigro L. 151

Øgård O.

Peyraut F. 5, 115, 123
Popp V. 49

Rüger M. 21
Ruzicka R. 127

Schuster G. 143
Sedols J. 77
Sjong D. 119
Sørlie J.A. 159
Stahl H. 95
Stief E. 155
Strunz C. 139
Symons A. 73

Telnes K. 119
Tettweiler W. 111
Tomsons Dz. 77
Trihy R. 53
Tsai K.H. 45

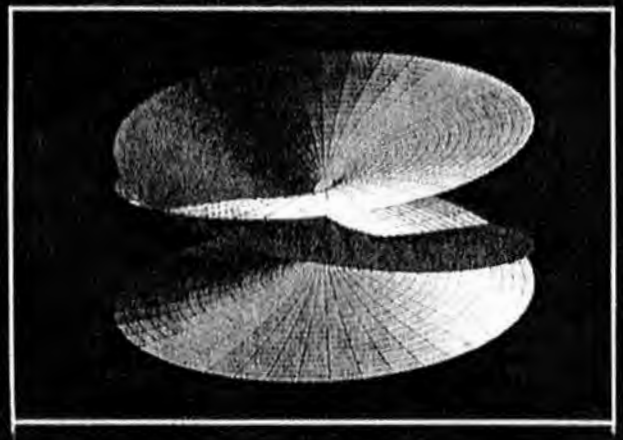
Unseld H.G. 13

Wang Y.M. 45

Yang J.J.S. 45

Zerhouni N. 57
Zupančič B. 147

MATLAB®
Die Software für math.-techn. Berechnungen

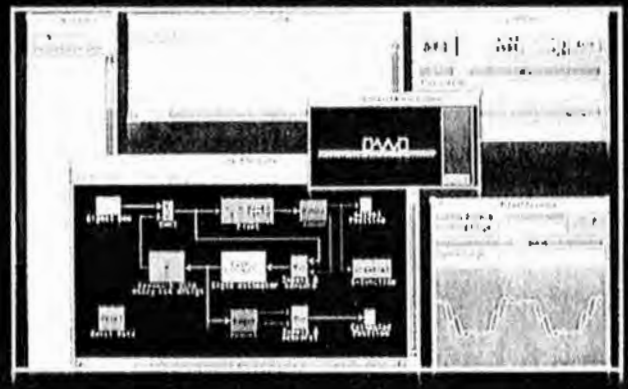


MATLAB® für Ingenieure und Naturwissenschaftler. Einfach anzuwenden. Ersetzt aufwendige Eigenprogrammierung.

Anwendungsgebiete:
- Gleichungsdefinition, Matrizenarithmetik
- Grafische Darstellung, 2D+3D
- Gleichungsbasierte Simulation nichtlinear. Systeme
- Auswertung von Versuchsdaten, Visualisierung, Animation, Algorithmen Entwicklung
- Formelauswertung, Statistik
- Eigenwertrechnung, Polynomarithmetik

Eigenschaften:
- Interaktive Anwendung, einfache Syntax
- PCs, Workstations und Mainframes
- Eigene Funktionen mit Fortran und/oder C
- Speichern und Wiederverwend. benutzereig. Funktionen
- Lesen und Schreiben beliebiger Dateiformate
- MATLAB ab **DM 1.600,-**, TOOLBOXEN ab **DM 990,-**

SIMULINK™
In MATLAB integriertes Simulationssystem für nichtlineare dynamische Systeme

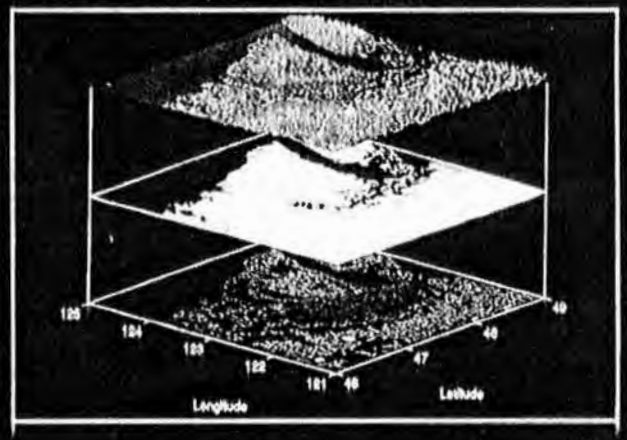


SIMULINK™ für die grafische blockbildbasierte Modellierung, Analyse und Simulation.

Modellierung:
- Lineare, nichtlineare, kontinuierliche und diskrete Modellteile in einem Modell
- Blockorientierte grafische Eingabe, aufbauend auf MS Windows (PC), X/Motif (Unix-Workstation) oder Macintosh Windowing
- Teilmodelle, Zahl der Hierarchie-Ebenen praktisch unbegrenzt, viele Standardblöcke verfügbar
- Eigene Blöcke in MATLAB-, C- oder Fortran-Code
- Speicherung in lesbarem MATLAB-Code

Systemuntersuchung:
- Bestimmung des eingeschwungenen Zustands
- Linearisierung nichtlinearer Modelle
- Parameteroptimierung, Reglerentwurf, Signalanalyse mit MATLAB-Toolboxen
- Generierung von C-Quellcode: C Code Generator

TOOLBOXEN
Anwendungsspezifische Ergänzungen für MATLAB



TOOLBOXEN (TB) zur Ergänzung von MATLAB und SIMULINK mit leistungsfähigen, fachspezifischen Zusatzfunktionen.

Signalverarbeitung: Signal Processing TB

Regelungstechnik und Systemidentifikation: Control System TB, Nonlinear Control Design TB, Robust Control TB, µ-Analysis and Synthesis TB, System Identification TB, State Space Identification TB

Simulation mechanischer Systeme: MECHMACS (Ergänzung zu SIMULINK)

Meßdatenerfassung, -verarbeitung, Steuerung, Regelung in Echtzeit: ECHTZEIT-ERWEITERUNG (Ergänzung zu MATLAB und SIMULINK)

Universell einsetzbar: Optimization TB, Neural Network TB, Chemometrics TB, Spline TB, Statistics TB, Image Processing TB, Symbolic Math TB

Software mit Zukunft

scientific COMPUTERS

D-52064 Aachen, Franzstraße 107, Tel. 0241/260 41, Fax 0241/449 83
Geschäftsstelle München:
D-85744 Unterföhring, Firkenweg 7, Tel. 089/99 59 01-0, Fax 089/99 59 01-11

Unseld + Partner

Business- and Marketing Consulting Simulation!

What we can offer our customers !

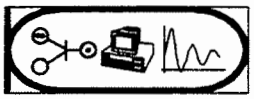
Unseld + Partner :

- focuses on simulation topics only and provides efficient industrial simulation experience, gained in many industrial projects
- consists of a group of highly educated simulation experts representing the highest concentration of such know-how in Austria
- offers simulation expertise in industrial projects, while being receptive for sophisticated new strategies in freight centres and intercompany logistics, tackling especially multimodal aspects.
- provides advice on EU-programs (concerning railways, IT and Telematics) to leading members of Austrian industrial organisations and government bodies
- is a completely independent Austrian company
- has co-operation contracts and contacts with many prominent international research and university institutes

Of course state-of-the-art simulation software technology for instance SIMPLE++ and VISIO are used and professional project management skills are provided.

→ Without Simulation no Innovation ←

Unseld + Partner Business- and Marketing Consulting Simulation!
A-1080 Vienna Lerchenfelderstraße 44/9 phone +43-1-4030371-0* fax +43-1-4030371-90



ARGESIM Reports

No.	Title	Authors / Editors	ISBN
# 1	Congress EUROSIM'95 - Late Paper Volume	F. Breitenecker, I. Husinsky	3-901608-01-X
# 2	Congress EUROSIM'95 - Session Software Products and Tools	F. Breitenecker, I. Husinsky	3-901608-02-8
# 3	EUROSIM'95 - Poster Book	F. Breitenecker, I. Husinsky	3-901608-03-6
# 4	Seminar Modellbildung und Simulation - Simulation in der Didaktik	F. Breitenecker, I. Husinsky, M. Salzmann	3-901608-04-4
# 5	Seminar Modellbildung und Simulation - COMETT - Course "Fuzzy Systems and Control"	D. Murray-Smith, D.P.F. Möller, F. Breitenecker	3-901608-05-2
# 6	Seminar Modellbildung und Simulation - COMETT - Course "Object-Oriented Discrete Simulation"	N. Kraus, F. Breitenecker	3-901608-06-0
# 7	EUROSIM Comparison 1 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-07-9
# 8	EUROSIM Comparison 2 - Solutions and Results	F. Breitenecker, I. Husinsky	3-901608-08-7