Informatica **21** (1997) Number 4, pp. 579–705

Volume 21 Number 4 November 1997 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Simulation Modelling Methodology and Education

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica

An International Journal of Computing and Informatics

Basic info about Informatica and back issues may be FTP'ed from `ftp.arnes.si` in magazines/informatica ID: anonymous PASSWORD: <*your mail address*>
FTP archive may be also accessed with WWW (worldwide web) clients with
URL: `http://www2.ijs.si/~mezi/informatica.html`

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 1997 (Volume 21) is
- DEM 50 (US$ 35) for institutions,
- DEM 25 (US$ 17) for individuals, and
- DEM 10 (US$ 7) for students

plus the mail charge DEM 10 (US$ 7).
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, Fin-Pro d.o.o., Stegne 27, 1000 Ljubljana, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 61 1773 900, Fax (+386) 61 219 385, or use the bank account number 900–27620–5159/4 Ljubljanska banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Engineering Index, INSPEC, Mathematical Reviews, MathSci, Sociological Abstracts, Uncover, Zentralblatt für Mathematik, Linguistics and Language Behaviour Abstracts, Cybernetica Newsletter

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 1000 Ljubljana, Slovenia.

Guest Editorial: Simulation Modelling Methodology and Education

Ray J. Paul & Simon J.E. Taylor
Centre for Applied Simulation Modelling
Department of Information Systems and Computing,
Brunel University, Uxbridge, UB8 3PH, England
ray.paul@brunel.ac.uk

1 Introduction

Simulation Modelling, and the tools and techniques which it embraces, has been with us since at least the 1960s. Arguably it has contributed to some of the major advances in computer science, and has presented some of the greatest challenges. The simulation language SIMULA (Dahl and Nygaard 1966) has contributed much to what is commonly understood as object-oriented programming. Advances in Human-Computer Interaction have also come from the demands made by the sophisticated user interfaces required for visual simulation and visual interactive modelling (Kuljis 1996). Parallel discrete event simulation (Bagrodia 1996; Nicol 1996; Fujimoto 1993; Taylor, Fatin and Delaitre 1995) presents some of the greatest challenges to high performance computing as does Distributed Interactive Simulation to distributed computing (Dahmann 1995; Fullford 1996). Simulation and Virtual Reality are closely linked (Barnes 1996) and also have implications for Distributed Computing (Kloeber, Jr and Jackson, Jr (1995); Pratt and Johnson 1995, Macredie, *et al.* 1996)). Simulation has also greatly contributed to the performance analysis of computers and communication systems (Jain 1991).

The purpose of this Special Issue is to introduce, or to reacquaint, the reader with those to whom this important field. Simulation Modelling Methodology and Education have been chosen for this (re-)introduction. Methodology concerns the process by which Simulation Modelling is applied, from the formulation of a problem to be solved, to suggested action to remedy fault. Education is close to the heart of Simulation Modelling and is at least as important as the Simulation Modelling process. Interaction with clients, or training those who will perform Simulation, has pedagogical implications.

We begin this Special Issue with a review dedicated to giving the interested reader an overview of Simulation Modelling Methodology and Education with respect to recent advances in the field. The internationally refereed papers that follow have been selected to present a wide range of stimulating and thought provoking issues currently attracting interest in the simulation community.

2 Simulation Modelling Methodology and Education

Simulation is dedicated to the study and analysis of systems as is, unsurprisingly, found in many areas such as manufacturing, road traffic planning, goods transportation, construction, military applications to name but a few. For example, simulation is being more widely used in business process design (contributions can be found in Bhaskar, *et al.* (1994); Bridgeland and Becker (1994); Giaglis, Doukidis and Paul (1996); Shapiro (1994); Swami (1995); and Tumay (1996)). To give the reader a feel for the variety of uses for which this technique can be employed, it is useful to present a short review. (Banks 1997; Pidd 1992; Robinson 1994) give excellent summaries of the uses of simulation in the study of systems. We present an outline of these below.

- Simulation enables the study of, and experimentation with, the internal interactions of a complex system, or of a subsystem in a complex system;
- Informal, organisation and environmental changes can be simulated and the effect of these alternations on the model's behaviour observed;
- Much knowledge about a system can be gained from designing and developing a simulation model. This in itself may be of great value toward suggesting improvement in the system under investigation;
- Incrementally changing simulation inputs and observing the resulting outputs yields valuable information into which model components are the most significant and how they relate to other model components;
- As simulation studies a system as it evolves over time, the time-base of a simulation can be controlled. Manipulation of a computerised model in this way allows model conditions to be investigated which take a very short time to occur in the real system;
- Simulation allows repeatability. A system can be studied repeatedly under the same conditions to investigate how the system changes over time. In

a similar vein, a simulation can be reproduced several times to enforce reliability over output data generated;

- Simulation can be used as a pedagogical device to reinforce training and system understanding;
- Simulation can be used to experiment with new designs, policies or strategies prior to implementation, so as to prepare for the consequences of real system change and (hopefully) minimise the time and cost of introducing this change; and
- Simulation allows the effect of extreme conditions in a system to be investigated thus providing the opportunity to increase safety in the real system under study. Similarly the implications of legal requirements made by external organisations on a system can be studied.

How should the process of simulation be carried out? Many authors suggest commensurate methodologies for this purpose (Balci 1994; Banks 1997; Fishwick 1994; Nance 1994; Pegden, Shannon and Sadowski 1995). To introduce this process, we shall use that suggested by Paul and Balmer (1993) as this agrees with the general layout, from the definition of the problem to be discussed to reporting on what has been discovered. Please note that, loosely speaking, the following steps define a sequential generation of deliverables from the start to the end of a simulation study. In reality, the steps are carried out in a more or less concurrent manner, with much prototyping involved. For those that are interested, Kelton (1996) and Kleijnen (1996) give excellent overviews of some of the statistical issues underpinning the simulation process.

1. Problem formulation.

   The formulation of the problem to be investigated is the most important deliverable in the process of simulation. The manner in which the problem is defined depends very much on who is carrying out the simulation work and who is/are the problem owners. Either way it is paramount that clear and open communication between the groups involved in the project is guaranteed. This first stage very much involves all related parties getting to know each other's working environment and motivations, as well as understanding what is the problem with the investigated system. The first deliverable from discussion is a clear statement of the formulated problem with fully stated objectives and a project plan as to how the objectives are to be satisfied. Several authors have suggested guidelines for a successful simulation project (Musselman 1994; Nordgren 1994; Pidd 1996a; Robinson 1994) which have implications for the management of software development. Lehaney and Paul (1994) and Lehaney and Paul (1996) review the popular field of soft systems methodology and suggest how it may be linked to systems analysis by simulation in the hope of better problem formulation.

2. Formal Simulation model

   The formal simulation model is the end product of the thoughts and ideas conceptualised in the formulation of the problem. The development of this model is an art as much as a science and aims to abstract the essential features of a problem to identify the critical model components relevant to the problem and to specify the relationships between these. It is typically specified using a tool that reflects the structure of the problem (for example Activity Cycle Diagrams (Carrie 1988; Paul 1993; Pidd 1992), Event Graphs (Buss 1996; Shruben 1983), Process Diagrams (Schriber 1974) and Object oriented representations (Hirata and Paul 1993)). Balci (1988) and Ceric and Paul (1992) present a comparative overview of several modelling approaches.

   As indicated previously, the depth to which a model component must be modelled is not initially clear. Neither is the manner in which the data complementing model behaviour (Leemis 1996) gives an introduction to data modelling in simulation. As the development of the formal simulation model progresses, greater understanding of the system accrues. Initially modelling a system as simply as possible allows, with cooperation of the user, model components to be represented in as simple or complex form as the problem dictates. Unsurprisingly, when experimentation on the operational model takes place, reporting results to the user can result in a shortcoming of the model being simulated. The outcome of this is that certain system components might require more detail to be added. This is an entirely valid activity, as it shows that greater understanding of the system has resulted.

3. Computerised simulation model

   Once the formal simulation model has been agreed as representing, to the best knowledge of the groups involved, the system being investigated, the model must be translated into a computerised form. There are many simulation programming languages and simulation modelling packages that may be used to achieve this end (Banks 1997; Banks 1996; Law and Kelton 1991; Nance 1995). Hlupic (1995); Hlupic and Mann (1995); and Hlupic and Paul (1993) suggest an approach to selecting the correct language or package which can implement the formal model under constraints such as expertise, cost, appropriateness, etc. Paul and Hlupic (1994) and Standridge *et al.* (1996) discuss developments in integrated computer-aided simulation modelling environments. Pidd (1996b) and Kuljis (1996) review HCI and simulation software with respect to understanding how far simulation has come with some cautionary tales to others involved in HCI.

In parallel with the development of the formal model and the computerised model are two concepts known as verification and validation. Verification concerns making sure that the computerised model executes correctly and parallels software testing very closely. Validation ensures that the model being executed is a model of the system under study and has not been allowed to drift away from the original specification. Reviews of the techniques used in verification and validation can be found in Balci (1994) and Sargent (1996) as well as recent advances in Arthur and Nance (1996).

4. Operational Simulation model

   The operational simulation model is the combination of the computerised model with an experimental design. The experimental design is a specification of how the model is to be experimented with, and what input parameters will be required as a consequence of this. Fu and Hu (1996) discuss some of the technical approaches used in experimentation. Tao and Nelson (1994) suggest a general framework to experimentation.

5. Experimental output and reporting

   The experimental output is derived from the experimentation with the operational model. The group performing the experiments must interpret the experimental output with respect to the system under investigation. Typically this involves analysis of how the model components interact and influence each other. To complete the simulation study, the results of analysis and modelling must be reported. The link between documentation and the modelling process is discussed in Patel, Gardner and Taylor (1996).

As the final part to this review, we consider Simulation Modelling education. Why is education in this area required? If one takes the view that simulation has a place in many different disciplines (as indicated by the introductory section) for the analysis of many diverse systems, then one must reflect on how the simulation process should be taught and where to teach it. The roles of those involved in simulation work also calls for some form of simulation education. For example, a user might choose to develop a simulation solution for their own problems. If this is the case, then the user must consider how simulation should be used. Alternatively, the user might employ specialists in simulation. In this case, for the simulation project to stand some chance of success, the user and the specialist must develop some common understanding of each other's skills and expertise. Again, simulation education becomes a requirement of this process if the user is to gain some understanding of what simulation requires and what it is (realistically) capable of.

There are several alternative solutions to this need. A user purchasing simulation software can (and is sometimes required) attend a training course run by the manufacturor of the software. The drawback with this is, although is it useful, the training course is just that; the user is trained in the use of software rather than the process of simulation. Instead, to gain a wider understanding of the simulation process, a user should seek a training course with a broader skill base. Muller (1996) discusses the need for real models in training, while Farrington, Messimer and Schroer (1994); Lorenz and Schriber (1996); Ståhl (1996); Silverman (1996); Paul and Hlupic (1994) discuss the impact and content of courses aimed at audiences with a focus on Simulation Modelling. Angelides and Paul (1993), Siemer and Taylor (1995); Taylor and Siemer (1996) and Atolagbe and Hlupic (1996) address how computer-aided learning can support the diverse needs of Simulation Modelling.

3 The Selected Papers

A fundamental component of simulation modelling methodology is visual simulation (sometimes called visual interactive modelling). Used to promote communication between client and simulationist for model development, validation (visual debugging) and reporting, the strength of the visual interface has played an increasingly vital role in the success of a simulation project. The visual interface is a major part of the interaction between human and computer. Human Computer Interaction (HCI) addresses the nuances of the interplay between the two and is therefore of particularly relevance to simulation. In the first of the papers of this special issue, Kuljis (1996) introduces many of the HCI issues in simulation by reviewing six simulation software packages. The points considered are simulation environment, data input, visual modelling, results display and user-assistance. The paper concludes with some general recommendations that can be used to improve the quality of HCI in simulation.

Pidd (1996) takes a different but complementary approach to this issue. The latest developments in HCI are reviewed. These include interface design, user-centered design, principles of learning, and the effect of the interface on the user. These are considered in the light of simulation and the needs and problems of different user groups. The paper concludes with a cautionary rebuke for developers of overly complex user interfaces.

The process of simulation modelling involves the building of, and experimentation with, a valid model of a given system with respect to a set of study objectives. By far the balance of work in simulation addressing this assumes the relative ease of objective definition and the identification of relevent system components. In reality one of the greatest reasons for the failure of simulation projects is the inappropriate choice of objectives and elements; the simulation was carried out correctly but failed to address and identify the right problem. A series of techniques which have found popularity in mainstream software development techniques are soft systems methodologies (SSMs). These complement the harder technological techniques that exist by focusing on the more humanist aspects of a development project. The goal of SSMs is therefore to correctly identify user requirements by formally involving the problem owners and stakeholders in the development such. In an attempt to gain the advantages in correctly identifying the objectives of a simulation project, Lehaney and Paul (1996) have combined aspects of SSMs with traditional simulation methodology. The paper reviews SSM and the suggested combined approach and illustrates this with a test case in the health service. This contribution represents an important first step and highlights that there is still much work to be done to successfully combine the two.

In contrast to this work is that considered by Giaglis, Paul and Doukidis (1996). Rather than attempting to improve the simulation modelling process, this attempts to assess the impact of simulation modelling methdology on an existing toolset. Approaches to business redesign in recent years have included business process re-engineering, continuous process improvement, total quality management and organisational transformation. These differ significantly in scope and range of use but have in common a need for some form of modelling. Business process modelling has recently received widespread attention and has been acknowledged as addressing this modeling requirement. Simulation is a natural choice to support business process modelling. A drawback is the complexed involve in applying simulation to business problems is proving to be a significant challenge. This contribution considers the technical, political and social requirements that business changes makes of modelling and suggests how simulation could solve these in the light of inter-organisational business processes. The paper concludes with suggestions as to how business prcoess modelling should engender success.

Macredie, *et al* (1996) take a more technological perspective and presents an informative overview on issues concerning the links between virtual reality and simulation. The paper explores this link and introduces a novel architecture required for distributed virtual reality access and suggests that an efficient underlying architecture is fundamental to the successful impact and evolution of a wide range of virtual reality applications.

Simulation is a technique which forms part of many disciplines and can be found in undergraduate and postgraduate degrees in engineering, manufacturing, business studies and science. There are many different reasons why the presence of simulation in a ciriculum can to be prove desirable. The appreciation of how systems work through the process of modelling and simulation could be argued as being the main driving force. To help stimulate ideas in those educators currently involved in simulation pedagogy, and to foster ideas in those wishing to introduce simulation, the remaining three papers of this special issue focus these aspects.

The first paper of this section by Stohl (1996) presents the experiences of an educator involved in simulation education for over twenty years. Making the observation that finite resources limit what can be achieved in a course in simulation, the author suggests a very simplistic approach should be used to emphasise important features of simulation methodology. Using the course language MICRO-GPSS as an example, the paper discusses the course features that should be used in successful delivery. Taylor and Siemer (1996) take a complementary stance to simulation education. Reviewing a current simulation course, suggestions are made as to how large numbers of students can be given high quality education. The paper suggests that computer-aided learning techniques such as an Intelligent Tutoring System should be used. The features of such a system are discussed. Computer-aided learning support to education are further discussed in Atolagbe and Hlupic (1996).

4 Conclusions

We have presented a diverse review of Simulation Modelling and Education. What follows is an eclectic mix of papers in this area which have been selected to provoke thought and to encourage the reader to delve further into what we consider to be a vibrant and fascinating subject. We hope that the reader is inspired to make their own investigations and, in the long run, their own decisions as to the impact Simulation Modelling could have on their future endeavours.

References

[1] Angelides, M.C. and Paul, R.J. (1993) Developing an Intelligent Tutoring System for a Business Simulation Game, *Journal of Simulation Practice and Theory*, 1, 3, p. 109-135.

[2] Arthur, J.D. and Nance, R.E. (1996) Independent Verification and Validation: A Missing Link

in Simulation Methodology? in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 230-236, Association for Computing Machinery, Coronado, California.

[3] Atolagbe, T.A. and Hlupic, V. (1996), A Generic Architecture for Intelligent Instruction for Simulation Modelling Software Packages, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J.,p.856-863, Association for Computing Machinery, Coronado, California.

[4] Bagrodia, R.L. (1996) Perils and Pitfalls of Parallel Discrete-Event Simulations, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 136-143, Association for Computing Machinery, Coronado, California.

[5] Balci, O. (1994) Validation, Verification, and Testing Techniques Throughout the Cycle of a Simulation Study, *Annals of Operations Research*, 53, p, 121-173.

[6] Balci, O. (1988) The Implementation of Four Conceptual Frameworks for Simulation Modelling in High Level Languages, in *Proceedings of the 1988 Winter Simulation Conference*, Eds. Abrams, M., Haigh, P. and Comfort, J., p.287-295. Association for Computing Machinery, San Diego, California.

[7] Banks, J. (1996) Software for Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 31-38, Association for Computing Machinery, Coronado, California.

[8] Banks, J, Carson II, J.S. and Nelson, B.L. (1997) *Discrete-Event System Simulation, Second Edition*, Prentice-Hall International Series in Industrial Systems Engineering, Eds. Fabrycky, W.J. and Mize, J.H., Prentice-Hall, New Jersey.

[9] Barnes, M. (1996) Virtual Reality and Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 101-110, Association for Computing Machinery, Coronado, California.

[10] Bhaskar, R., Lee, H.S., Levas, A., Petrakian, R., Tsia, F. and Tulskie, B. (1994) Analyzing and Reengineering Business Processes using Simulation, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 1206-1213, Association for Computing Machinery, Lake Buena Vista, Florida.

[11] Bridgeland, D. and Becker, S. (1994) Simulation Satyagraha, A Successful Strategy for Business Process Reengineering, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 1214-1220, Association for Computing Machinery, Lake Buena Vista, Florida.

[12] Buss, A.H. (1996) Modelling with Event Graphs, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 153-160, Association for Computing Machinery, Coronado, California.

[13] Carrie, A. (1988) *Simulation of Manufacturing Systems*, John Wiley and Sons, Chichester.

[14] Ceric, V. and Paul, R.J. (1992) Diagrammatic Representations of the Conceptual Model for Discrete Event Systems, Mathematics and Computers in *Simulation*, 34, 317-324.

[15] Dahl, O. and Nygaard, K. (1996) SIMULA - an Algol-based simulation language. *Communications of the ACM*, 9, 9, p. 671-678.

[16] Dahmann, J.S. and Wood, D.C. (1995) Eds. Special Issue on Distributed Interactive Simulation, *Proceedings of the IEEE*, 83, 8.

[17] Farrington, P.A., Messimer, S.L. and Schroer, B.J. (1994) Simulation and Undergraduate Engineering Education: The Technology Reinvestment Project (TRP), in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 1387-1393, Association for Computing Machinery, Lake Buena Vista, Florida.

[18] Fishwick, P.A. (1994) *Simulation Model Design and Execution: Building Digital Worlds*. Prentice-Hall, New Jersey.

[19] Fu, M.C. and Hu, J.-Q. (1996) A Comparison of Perturbation Analysis Techniques, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 295-301, Association for Computing Machinery, Coronado, California.

[20] Fujimoto, R.M. (1993) Parallel Discrete Event Simulation: Will the Field Survive?, *ORSA Journal of Computing*. 5, 3, p.218-230.

[21] Fullford, D. (1996) Distributed Interactive Simulation: It's Past, Present and Future, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 179-185, Association for Computing Machinery, Coronado, California.

[22] Giaglis, G.M., Paul, R.J. and Doukidis, G.I. (1996) Simulation for Intra- and Inter-Organisational Business Process Modelling, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 1297-1304, Association for Computing Machinery, Coronado, California.

[23] Hirata, C.M. and Paul, R. J. (1996). Object-Oriented Programming Architecture for Simulation Modelling. *International Journal in Computer Simulation.* 6, 2, p. 269-287.

[24] Hlupic, V. (1995). A Comparison of Simulation Software Packages. *Proceedings of the 1995 EUROSIM Conference.* EUROSIM'95, Eds. Breitenecker, F. and Husinsky, I., p.171 - 176, Vienna, Austria.

[25] Hlupic, V. and Mann, A. S. (1995). SimSelect: A System for Simulation Software Selection, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.720 - 727, Association for Computing Machinery, Arlington, Virginia.

[26] Hlupic, V. and Paul, R.J. (1993) Simulation Software in Manufacturing Environments: A Users' Survey. *Journal of Computing and Information Technology,* 1, 3, 205-212.

[27] Jain, R. (1991) *The Art of Computer Systems Performance Analysis: Techniques for Experiemntal Design, Measurement, Simulation and Metamodelling,* Wiley, New York.

[28] Kelton, W.D. (1996) Statistical Issues in Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 47-53, Association for Computing Machinery, Coronado, California.

[29] Kleijen, J.P.C. (1996) Five-Stage Procedure for the Evaluation of Simulation Models Through Statistical Techniques, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 248-254, Association for Computing Machinery, Coronado, California.

[30] Kloeber, Jr, J.M. and Jackson, Jr, J.A. (1995) Issues in Using a DIS Facility in Analysis, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.1229-1236, Association for Computing Machinery, Arlington, Virginia.

[31] Kuljis, J., HCI and Simulation Packages, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 687-694, Association for Computing Machinery, Coronado, California.

[32] Law, A.M. and Kelton W.D. (1991) *Simulation Modelling and Analysis, Second Edition*, McGraw-Hill, New York.

[33] Leemis, L.M. (1996) Discrete-Event Simulation Input Process Modelling, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 39-46, Association for Computing Machinery, Coronado, California.

[34] Lehaney, B. and Paul R.J., Soft Systems Methodology and Simulation Modelling, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 695-700, Association for Computing Machinery, Coronado, California.

[35] Lehaney, B. and Paul, R.J. (1994) Using Soft Systems Methodology to Develop a Simulation of Outpatient Services, *Journal of the Royal Society for Health*, August, 197-200.

[36] Lorenz, P. and Schriber, T.J. (1996) Teaching Introductory Simulation in 1996: From the First Assignment to the Final Presentation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 1379-1386, Association for Computing Machinery, Coronado, California.

[37] Macredie, R.M., Taylor, S.J.E., Yu, X. and Keeble, R. (1996) Virtual Reality and Simulation: An Overview. in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 669-674, Association for Computing Machinery, Coronado, California.

[38] Muller, D.J. (1996) Simulation: "What To Do With The Model Afterward", in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 729-733, Association for Computing Machinery, Coronado, California.

[39] Musselman, K.J. (1994) Guidelines for Simulation Project Success, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 88-95, Association for Computing Machinery, Lake Buena Vista, Florida.

[40] Nance, R.E. (1995) Simulation Programming Languages: An Abridged History, in *Proceedings of*

the 1995 Winter Simulation Conference, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.1307-1313, Association for Computing Machinery, Arlington, Virginia.

[41] Nance, R.E. (1994) The Conical Methodology and the Evolution of Simulation Model Development, *Annals of Operations Research*, 53, p.1-46.

[42] Nordgren, W.B. (1995) Steps for Proper Simulation Project Management, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.68-73, Association for Computing Machinery, Arlington, Virginia.

[43] Nicol, D. (1996) Principles of Conservative Parallel Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 128-135, Association for Computing Machinery, Coronado, California.

[44] Patel, N. V., Gardner, L. A. and Taylor, S. J. E. (1996) Hypermedia for Manufacturing Simulation. *International Journal of Manufacturing System Design*, 2, 2, p.153-161

[45] Paul, R.J. (1993) Activity Cycle Diagrams and the Three Phase Method, in *Proceedings of the 1993 Winter Simulation Conference*, Eds. Evans, G.W., Mollaghasemi, M., Russell, E.C. and Biles, W.E., p.123-131, Association for Computing Machinery, Los Angeles, CA.

[46] Paul, R.J. and Balmer D.W. (1993) *Simulation Modelling.* Chartwell-Bratt Student Series, Lund, Sweden.

[47] Paul, R.J. and Hlupic, V. (1994) The CASM Environment Revisited, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 641-648, Association for Computing Machinery, Lake Buena Vista, Florida.

[48] Paul, R.J. and Hlupic, V. (1994b) Designing and Managing a Masters Course in Simulation Modelling, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 1394-1398, Association for Computing Machinery, Lake Buena Vista, Florida.

[49] Pegden, C.D., Shannon, R.E. and Sadowski, R.P. (1995) *Introduction to Simulation Using SIMAN, Second Edition*, McGraw-Hill, New York.

[50] Pidd, M. (1996a) Five Simple Principles of Modelling, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 721-728, Association for Computing Machinery, Coronado, California.

[51] Pidd, M. (1996b) Model Development and HCI, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 681-686, Association for Computing Machinery, Coronado, California.

[52] Pidd, M. (1992) *Computer Simulation in Management Science*, Third Edition, John Wiley and Sons, Chichester, England.

[53] Pratt, D.R. and Johnson, M.A. (1995) Constructive and Virtual Model Linkage, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.1222-1228, Association for Computing Machinery, Arlington, Virginia.

[54] Robinson, S. (1994) *Succcessful Simulation*, McGraw-Hill International, Maidenhead.

[55] Schruben, L. (1983) Simulation Modelling with Event Graphs, Communications of the ACM, 26. p. 957-963.

[56] Sargent, R.G. (1996) Verifying and Validating Simulation Models. in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 55-63, Association for Computing Machinery, Coronado, California.

[57] Schriber, T.J. (1974) *Simulation Using GPSS*, Wiley, New York.

[58] Shapiro, R.M. (1994) Integrating BPR with Image-based Work Flow, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 1221-1228, Association for Computing Machinery, Lake Buena Vista, Florida.

[59] Siemer, J., Taylor, S. J. E., and Elliman, A. D. (1995) Intelligent Tutoring Systems for Simulation Modelling in the Manufacturing Industry. *International Journal of Manufacturing System Design*, 2, 3, p.165-175.

[60] Silverman, R. (1996) Simulation for Computer Science Majors, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 1395-1400, Association for Computing Machinery, Coronado, California.

[61] Swami, A. (1995) Building the Business using Process Simulation, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C.,

Kang, K. Lilegdon, W.R. and Goldsman, D., p.1081-1086, Association for Computing Machinery, Arlington, Virginia.

[62] Ståhl, I. (1996), Teaching the Fundamentals of Simulation in a Very Short Time, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 1387-1394, Association for Computing Machinery, Coronado, California.

[63] Standridge, C.R., Kelly, J.F., Kelley, T. and Walther, J. (1996) Progress in Modular Simulation Environments, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 714-720, Association for Computing Machinery, Coronado, California.

[64] Tao, Y-H. and Nelson, B.L. (1994) A Conceptual Framework for Simulation Experiment Design and Analysis, in *Proceedings of the 1994 Winter Simulation Conference*, Eds. Tew J.D., Manivannan, S., Sadowski, D.A., and Seila A.F., p. 681-688, Association for Computing Machinery, Lake Buena Vista, Florida.

[65] Taylor, S.J.E., Fatin, F. and Delaitre, T. , (1995) Estimating the Benefit of the Parallelisation of Discrete Event Simulation, in *Proceedings of the 1995 Winter Simulation Conference*, Eds, Alexopoulos, C., Kang, K. Lilegdon, W.R. and Goldsman, D., p.674-681, Association for Computing Machinery, Arlington, Virginia.

[66] Taylor, S.J.E. and Siemer, J. (1996) Enhancing Simulation Education with Intelligent Tutoring Systems, *Proceedings of the 1996 Winter Simulation Conference*, ed. Charnes J.M., Morrice D.J., Brunner D.T. and Swain J.J., p. 675-680, Association for Computing Machinery, Coronado, California.

[67] Tumay, K. (1996) Business Process Simulation, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 93-98, Association for Computing Machinery, Coronado, California.

HCI and Simulation Packages

Jasna Kuljis
Department of Applied Technology and Computing,
Roehampton Institute London,
West Hill, London SW15 3SN, England
Phone: +44-181-392-3672, Fax: +44-181-392-3271
jkuljis@roehampton.ac.uk

Keywords: simulation, HCI

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

Computer-based simulation modelling is one of the domains that is particularly demanding in terms of user interfaces. It is also an area that often pioneers new technologies that are not necessarily previously researched in terms of human-computer interaction (HCI). Issues that influence the 'usability' of such systems are examined. Several representative systems were investigated in order to generate some general assumptions with respect to those characteristics of user interfaces employed in simulation systems. There is a need for simulation systems that can support the developments of simulation models in many domains, which are not supported by contemporary simulation software. Many user interface deficiencies are discovered and reported. On the basis of findings in this research, proposals are made on how user interfaces for simulation systems can be enhanced to match better the needs specific to the domain of simulation modelling, and on how better to support users in simulation model developments.

1 Introduction

The population of computer users is increasing in numbers and expanding to all areas of human activities. HCI research is driven by that expansion. At the same time as computer systems become more 'usable', that brings in more new users and new uses of computers. Despite all the advances and improvements in today's interactive computer systems we are all too well aware that there are still many deficiencies in current user interfaces. Computer systems now have to cater for all kinds of task domains and for all types of user populations. This means that user interfaces have to bridge the gap between often computer illiterate users and computer systems.

Computer-based simulation modelling is one of the domains that is particularly demanding in terms of user interfaces. It is an area that covers aspects of user interfaces that usually span over several application areas. It is also an area that often pioneers new technologies that are not necessarily previously researched in terms of human-computer interaction. Simulation systems therefore are worthy of examination from the user interface point of view. Particular attention in this research is therefore also placed on investigating issues related to interaction styles, interaction objects, screen layout design, navigation through interfaces, user support and assistance. This could result in more awareness among the simulation community of the importance to provide user interfaces that better match the needs specific to the domain of simulation modelling.

2 What is HCI

Only a small number of computer systems today are designed to run autonomously. Most of them are interactive systems in which human users and computers interact through a user interface. The success of a computer system is often dependent on how easily the user can learn to use the user interface. Today, the user interface is the first thing many people ask about when discussing a new software application. To users, the interface is the system (Hix and Hartson, 1993). Computer systems often lack good user interfaces for a variety of reasons, including the lack of a good user interface design methodology and the lack of good tools to implement a user interface. An interface is often the single most important factor in determining the success or failure of a system (Larson, 1992). It is also one of the most expensive. Smith and Mosier (1984) conducted a survey of people concerned with information systems design who on average estimated that 30 to 35 percent of operational software is required to support the user interface. Bobrow *et al.* (1986) claim that the user interface often constitutes one third to one half of the code of typical knowledge-based systems.

This claim is reinforced by Myers and Rosson (1992) who argue that anywhere from an average of 48 percent to a maximum of nearly 100 percent of the code for an interactive system is now used to support the user interface.

Human-computer interaction concerns itself with the domain of interaction between humans and computers, and all the issues associated with that activity: the interaction itself (as a process) and knowledge about that interaction. There is no agreed upon definition of the range of topics which form the area of human-computer interaction. The following definition is one from ACM SIGCHI (1992): "Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them". Benyon and Murray (1988) make an important distinction between the terms Human-Computer Interaction and Human-Computer Interface. Interaction includes all aspects of the environment such as the working practices, office layout, provision of help and guidance, and so on. The interface is the part of a system with which the user comes into contact physically, perceptually or cognitively. In this paper we are mostly concerned with the interface part of simulation systems.

Research into HCI is now well established. The research spans across many disciplines. As a consequence the theories and methodologies developed within HCI are beginning to be used in a prescriptive way to develop new software products. In this paper we examine the usability and appropriateness of such approaches when dealing with simulation software development. We aim to examine user interfaces for discrete event simulation packages. In particular, to investigate issues that influence 'usability' of simulation systems. Usability has many meanings to many different people. The term is often used by software vendors who claim that their products have attributes such as high level of software usability, or 'user friendliness'. However, this terminology mostly indicates that software has a Windows like GUI. There is no generally agreed definition of usability. A definition of usability proposed by the International Standards Organization (ISO) and listed in Booth (1989) states: "The usability of a product is the degree to which specific users can achieve specific goals within a particular environment; effectively, efficiently, comfortably, and in an acceptable manner." This definition does not explicitly specify operational criteria that might lead to an understanding of what we should evaluate. A more operational definition is given by Shackel (1991) who suggests that any system should have to pass the usability criteria of effectiveness, learnability, flexibility, and user attitude. We assess the usability of current simulation systems based on the definition of usability that we adopted from Shackel (1991). Shackel's four distinguishable and quantifiable dimensions effectiveness, learnability, flexibility, and attitude are not mutually exclusive in the sense that measures of, for example, effectiveness can, at the same time, also give some indication of system learnability. Effectiveness refers to levels of user performance, measured in terms of speed and/ or accuracy, in terms of proportion of task(s), proportion of users, or probability of completion of a given task. Flexibility refers to variations in task completion strategies supported by a system. Learnability refers to the ease with which new or occasional users may accomplish certain tasks. Attitude refers to user acceptability of the system in question. There are some problems, however, in setting appropriate numerical values against usability goals. This approach would be more appropriate when the usability goals are set during the design stage of requirements specification than as an evaluation criteria once a system is finished. Although quantitative data is required to accurately assess the usability of a system, it is qualitative information that informs designers how to change an unusable system. Our objectives are both to assess the usability of current simulation systems and to identify usability defects. The usability evaluation of the representative simulation system was carried out using structured walkthrough (Booth, 1989), i.e. we worked through a series of tasks the user might be expected to perform looking for sources of potential difficulties. We have based the examination on simulation software for personal computers partly because the issues of interaction are not dependent on the computer platform and partly because the PC platform predominates among commercial simulation systems. Therefore, the results and findings can be generalised across the whole spectrum of simulation software regardless of the host system.

3 HCI Features In Examined Packages

We have reviewed six discrete simulation systems: XCELL+, Taylor II, ProModel for Windows, Micro Saint for Windows, WITNESS for Windows, and Simscript II.5 for Windows. We examine the following interaction characteristics of these systems: input-output devices employed, interaction styles, and use of graphics. We are also interested in: type of simulation system, application areas, hardware platform, operating system, and hardware requirements. Table 1 provides a summary of the main characteristics of each system examined and general user interface features for each of the reviewed systems.

For each simulation system we examine the user interface for the three identified modules: data input/model specification, simulation experiments, and presentation of output results. We are interested in

	XCELL+	Taylor II
Application area	Manufacturing	Mainly manufacturing
Software type	Data-driven simulator	Data-driven simulator
Interaction style	Menus and fill-in	forms
Interaction devices	Keyboard	Keyboard and mouse
Navigation facilitated using	Function keys	Mouse, arrow keys, and function keys
Terminology	Manufacturing	Manufacturing
Screen layout	Overcrowded	Good
No. of colours	12	16
Use of colours	Hideous	Good
	ProModel	Micro Saint
Application area	Mainly manufacturing	General Purpose
Software type	Data-driven simulator	Data-driven simulator
Interaction style	Windows GUI	Windows GUI
Interaction devices	Keyboard and mouse	Keyboard and mouse
Navigation facilitated using	Mouse, arrow keys, and F1	Mouse, arrow keys, and function keys
Terminology	Manufacturing	No specific domain
Screen layout	Good	Good
No. of colours	64	16
Use of colours	Good	Good
	Witness	Simscript II.5
Application area	Mainly manufacturing	General Purpose
Software type	Data-driven simulator	Language
Interaction style	Windows GUI	Windows GUI
Interaction devices	Keyboard and mouse	Keyboard and mouse
Navigation facilitated using	Mouse, arrow keys, and F1	Mouse, arrow keys, and F1
Terminology	Manufacturing	No specific domain
Screen layout	Poor	Good
No. of colours	16	64
Use of colours	Too extensive	Good

Table 1: Main System Characteristics

adopted interaction styles, modes of interaction, screen design and layout, interaction flexibility, supported functionality, navigation styles, use of colour, and the possibility to import and export data.

We also examine what kind of user support and assistance is provided and analyse how this provision is facilitated. Finally, we evaluate each system against the usability criteria set in the previous section. To achieve that we test each of the six listed systems on the task of developing a small queuing model (a bank). The test is performed by a user with a high computer literacy, low domain knowledge, and no knowledge of any of the six simulation systems. The ability to accomplish the task must be based solely on consulting the user manuals and help (i.e. without formal training). We try to identify which of the general usability principles are applied and also establish where the usability defects are in each examined system. We want to find out where in a system users might run into problems and what kind of problems they are likely to encounter. When examining the user interface we are particularly interested in three aspects: firstly, how the user interface for a particular system aids the user in a model development process; secondly, can the user modify the existing interface to either accommodate the user's own preferences or to adjust the modelling environment to the needs of a particular model; and thirdly, does the system facilitate user interface development.

3.1 Simulation Environments

The users of simulation software are often experts in special application fields. Kömper (1993) points out that these users, being laymen in information science, should be supported as much as possible by the simulation tool. The areas of concern are: validation, the development of simple models, and the development of complex models which contain, for example, non-typical phenomena in a special problem class. She advocates that the simulation tools should support the model builder in the first phases of becoming familiar with the tool, and in further work to model more complex phenomena without being forced to learn new concepts of model building. She sees the development of task specific user interfaces which relate to the respective knowledge as an aid to modelling. Bright and Johnston (1991) point out the necessity of 'ease of use' of simulation software. They see 'ease of use' as a combination of structural guidance in the model development process, error prevention, help provision, and the rate at which familiarity with the software is gained. However, they are concerned that the provision of these requirements in visual interactive modelling software will hinder generality and reduce its power. Regardless of whether the simulation systems are data-driven simulators or simulation languages, it is becoming common to provide some sort of integrated model building environment. The current simulation systems are typically visual interactive simulation systems that provide an integrated simulation environment. This type of environment includes a collection of tools for designing, implementing, and validating models; verifying the implementation; preparing the input data; analysing the output results; designing experiments; and interacting with the simulation as it runs. Such systems are inherently complex and inevitably employ a broad selection of interaction styles and often use technology in a novel way. The provision of completely self-sufficient simulation environments is important for the following reasons:

- it reduces the development time,
- it supports application consistency,
- it can aid the developers throughout the development cycle,
- it can support model completeness,
- it can provide checks of model validation.

The experience we have gained during this research convinced us that the model development process is generally not well supported. Five of the six examined simulation packages are data-driven simulators (XCELL+, Taylor II, ProModel for Windows, Micro Saint, and Witness for Windows) that use some sort of diagramming tools for representation of model logic, and one is a simulation language (Simscript II.5 for Windows). All six systems provide modelling environments. Two systems are general purpose (Micro Saint, Simscript II.5) whereas the other four are manufacturing or mainly manufacturing. Graphic elements for the representation of the model logic are pre-defined for all simulators, and cannot be changed for two of them (XCELL+ and Micro Saint). Names of model elements (i.e. machines, parts) are pre-defined and cannot be changed for any of the simulators, although the user can provide labels for individual instances of elements to describe better the domain related elements (i.e. an element machine can be labelled 'clerk' or 'bank teller' in a bank model using Taylor II). Similarly, all examined simulators use fixed, pre-defined, and unmodifiable attribute names (fields). Data entry is usually facilitated through pre-defined, unmodifiable fill-in forms which use the system's own element names, attribute names (fields), which usually have default values provided. Data validation is not a common facility (only in Taylor II and ProModel for Windows). Background drawing tools are rarely facilitated (Taylor II, ProModel for Windows, Simscript II.5 for Windows), as is importing graphics from other applications (ProModel for Windows, Micro Saint, Simscript II.5 for Windows). Icon editors are more common (not provided in XCELL+ and Micro Saint), even though the majority of them only provide elementary drawing capabilities. The user is rarely allowed to control statistics collection (only in Micro Saint and Simscript II.5) and the way the statistics is displayed (only Simscript II.5 gives complete freedom). Report customisation is rarely allowed. If this facility is provided, only a limited set of options can be exercised. On-line help, if available, usually does not extend to anything more than an overview of basic system concepts. Context sensitive help is scarce and good context-sensitive help is almost non-existent (the exception is ProModel for Windows). On-line help for error messages is not available on the examined systems. The customisation of the modelling environment is virtually an unknown commodity. A limited customisation is offered only in ProModel for Windows. The development of separable user interfaces for particular simulation problems is possible only in Simscript II.5 for Windows, which facilitates user interface development by providing templates for menus, fill-in forms, and several types of graphs that can be then tailored to suit the problem.

3.2 Data Input

The least effort in existing simulation systems has been invested into data input facilities. It is apparent that this part of the system is considered as less important than, for example, the visual simulation part. Most of the papers on simulation systems only briefly mention the data input capabilities of systems, if at all. However, there is room for a great deal of improvement in the domain of data input and/or model specification that would improve existing simulation systems. We have already mentioned that data validation is supported in only two of the six examined simulation systems. None of the systems offers database capabilities for keeping multiple variations of a model. Data input forms, if available, are generally poorly designed. There is no help provision for individual data fields. Importing data files is supported in four of the examined systems. The format of imported data is usually an ordinary ASCII text file. Therefore, there is much to be improved in the way the simulation data is communicated to the systems.. Table 2 summarise user interfaces for data input/model specification for the systems examined. Most of the systems keep data in text files that can be accessed and modified from other environments. None of the examined systems provides database facilities. Simulation languages, like SIMSCRIPT II.5 for example, can provide most of the data input features (menu-driven system, data-input forms, on-line help, data validation, etc.) but at the expense of an extensive time-consuming programming effort. Some of the systems provide limited input error checking and model verification facilities.

3.3 Visual Simulation

Visual programming tools are standard features in all visual interactive simulation (VIS) systems, and drawing tools are very common. Dynamic icons and animation are supported by most visual simulation systems. The interactive change of the simulation parameters and of the speed of animation whilst the simulation is being executed, are also often provided. Panning and zooming is another quite common facility. Most of the current simulation systems have some form of visual animation of a simulation run. Animation speed in simulation systems is commonly made adjustable by their users. There are some problems with the animation speed that are not envisaged by the software developers. Simulation software is built for a particular hardware configuration and therefore for a particular processing speed (in MHz). The speed of animation (moving icons) is dependent on the computer processor speed. Hardware developments are much faster than software developments and by the time simulation software, based on a particular configuration, has reached the market it may well happen that the market has already adopted much faster computers. The user will probably install software on a much faster computer than it was intended for. Even though the user may have a facility to change animation speed, the slowest available speed may still be too fast for an animation observer. Table 3 provides a summary of some of the user interface features that are relevant for the design of simulation experiments.

3.4 Simulation Results

Considerable effort has been invested in the presentation of simulation results. Often this effort lacks proper insight into the particular needs of simulation systems. Most simulation software offers graphical facilities for the representation of simulation statistics. Graphical coding can provide a powerful way of displaying quantitative data. In particular, graphs are able to abstract important relational information from quantitative data. The main advantages of using graphical representations (Preece *et al.*, 1994) are that it can be easier to perceive:
– the relationships between multidimensional data,
– the trends in data that are constantly changing,
– the defects in patterns of real-time data.

Besides the standard reports for commonly occurring performance statistics (e.g. utilisation, queue sizes and delays, and throughput) some of the newer software allows the development of tailored reports. The user can choose the form of representation (e.g. textual output, table, representational graph, bar diagram, pie chart, histogram, time series), colours, labels, etc. In most of the VIS software, partial statistical results can be viewed during the simulation run. Graphs are updated dynamically during the simulation run. Some of the software has facilities to save the statistics into files that can then be printed. Rarely is there a facility to print the whole screen at any point of a simulation run or when the statistics are displayed/presented on the screen. Table 4 gives a summary of presentation of simulation results capabilities of the reviewed software.

3.5 User Support

As a rule documentation is highly erratic, written in a technical jargon and rarely organised in any structured manner. If it contains an index (this is not always the case) it usually requires the user to know the exact terminology used to be able to find the topic of interest. Similarly, the installation procedures are generally badly documented and off-putting ("Insert next disk" - Yes? Which is the next disk if they are not numbered?). All of the systems examined, except Simscript II.5, require a security device to be plugged either in a serial or a parallel computer port (it varies from system to system). Considering the high cost of simulation software it makes some economic sense from the vendors point of view. However, from the customers point of view it shows a basic mistrust in customers' honesty and adds to the general unfriendliness of the simulation systems. User manuals for simulation systems are usually poorly written and need a lot of improvements. Of the six simulation systems we have examined only two provide Tutorial (Taylor II and Micro Saint), three provide Getting Started (XCELL+, ProModel for Windows, and Micro Saint), two provide Reference manuals (Pro Model for Windows and Simscript II.5). An index is provided in all of them except XCELL+, but it usually lists only system concepts using a particular simulation system's terminology. Generally, terminology used in the user manual is too technical. Examples, if provided, are not followed throughout the development process and are therefore of not much use. The summary of characteristics of printed manuals is given in Table 5.

On-line help very rarely provides help for all facilities and tools in the simulation environment. Context-sensitive help is a rare commodity (it is only provided in Taylor II and ProModel for Windows) and is almost unheard of for system messages (i.e. error messages).

	XCELL+	Taylor II
Model logic representation	Diagramming tools	Diagramming tools
Graphic elements	Pre-defined cannot be changed	Pre-defined can be changed
Model elements	Pre-defined	Pre-defined
Element names	Up to 10 characters	Up to 8 characters
Attribute names	Pre-defined cannot be changed	Pre-defined cannot be changed
Default values provided	Yes	Yes
Fill-in forms design	Not applicable	Good
Importing files supported	No	Yes
Data validation supported	No	Yes
Model validation supported	Partially	No
	ProModel	Micro Saint
Model logic representation	Diagramming tools	Diagramming tools
Graphic elements	Default or user selected	Pre-defined cannot be changed
Model elements	Pre-defined	Pre-defined
Element names	Up to 80 characters	Up to 20 characters
Attribute names	Pre-defined cannot be changed	Pre-defined cannot be changed
Default values provided	Yes	No
Fill-in forms design	Good	Well balanced
Importing files supported	Yes	No
Data validation supported	Yes	No
Model validation supported	No	No
	Witness	Simscript II.5
Model logic representation	Diagramming tools	Program
Graphic elements	Pre-defined cannot be changed	Not provided
Model elements	Pre-defined	User defined
Element names	Up to 8 characters	User defined
Attribute names	Pre-defined cannot be changed	User defined
Default values provided	Yes	Can be programmed
Fill-in forms design	Poor and often confusing	Not applicable
Importing files supported	Yes	Yes
Data validation supported	No	Can be programmed
Model validation supported	No	Can be programmed

Table 2: Data Input / Model Specification

An on-line tutorial is provided only in ProModel for Windows. Demonstration disks are provided for three of the examined systems Taylor II, ProModel for Windows, and Micro Saint, and of these three only ProModel provides a professional and carefully thought out product. Table 6 provides a summary of on-line user assistance offered in the six examined simulation packages.

4 Usability Of Simulation Packages

The usability evaluation of the six examined simulation packages indicates that only Pro Model for Windows provides an adequate level of support. Good user manuals are provided only for XCELL+, ProModel for Windows and Witness for Windows. Good on-line help is provided only for ProModel for Windows. Similarly ProModel for Windows is the only package that:

- employs the natural structure of a dialogue that achieves a close match with the user's task organisation instead of being matched to the internal structure of the application;
- provides consitency in screen layout and organisation, in terminology, in system messages, in actions, etc.;
- provides feedback that always keep users informed about what is going on;
- provides clearly marked exits;
- has error messages expressed in plain language (no codes) that precisely indicate the problem, and constructively suggest a solution.

Effectiveness of the system can be hindered if there are: defects in navigation through the system, prob-

	XCELL+	Taylor II
Background drawing tool	No	Yes
Icon editor	No	Yes
Importing graphics supported	No	No
Zooming supported	No	No
Panning supported	No	Limited (with arrow keys)
Interactive speed change	Yes	Yes
Interactive time change	Yes	Yes
Interactive change (other)	Yes	Yes
	ProModel	**Micro Saint**
Background drawing tool	Yes	No
Icon editor	Yes	No
Importing graphics supported	Yes	Yes
Zooming supported	Yes	No
Panning supported	Yes	No
Interactive speed change	Yes	Yes
Interactive time change	Yes	No
Interactive change (other)	Yes	Yes
	Witness	**Simscript II.5**
Background drawing tool	No	Yes
Icon editor	Limited Capabilities	No
Importing graphics supported	No	Yes
Zooming supported	Using Virtual Windows	No
Panning supported	Limited	No
Interactive speed change	Limited to 3 speeds	Can be programmed
Interactive time change	No	Can be programmed
Interactive change (other)	No	Can be programmed

Table 3: Simulation Experiment

lems in screen design and layout, inappropriate terminology, inappropriate feedback or complete lack of feedback, problems with modality, inconsequential redundancies, and problems in matching with user tasks. Learnability can be impeded if there are: defects in navigation, problems in screen design and layout, inappropriate terminology, inappropriate feedback or complete lack of feedback, and problems in matching with user tasks. Flexibility is impeded if there is no user control over the system and if the system imposes the order in which the steps in a task are performed. The user attitude towards the system can be seriously affected by any of the above usability defects.

The usability defects were identified in examined simulation systems. Again only ProModel for Windows has few problems. The only serious obstacles present in ProModel are its terminology and visual objects that are appropriate solely for the manufacturing domain. Therefore, in other domains there can be a problem in analogical mapping of the problem to an unsuitable domain. That would cause problems in matching the user tasks. XCELL+, Witness for Windows, and Simscript II.5 have serious defects in navigation through the system. Feedback is inadequate in all five packages except in ProModel for Windows. Consistency of a system is assessed based on the degree to which the system performs in a predictable, well organised and standard fashion. XCELL+ has an inconsistent use of function keys, Taylor II an inconsistent use of interaction devices, and Simpscript II.5 suffers from the inconsistency across its system modules. Modality is the state of the system operation that the user selects to perform a particular function (how easy is it to move between modes; how many different modes must the user know; how easy is it to recognise where one is at any time?). Only ProModel for Windows and Micro Saint provide clear feedback on the mode that the system is currently in. The four other systems either use modal dialogue which require the user to respond before any action can be taken or do not provide a clear indication in which mode the system is currently in. Only ProModel for Windows gives the user's feeling of being in control. It seems that the idea of adaptive user interfaces has not yet reached simulation software developers. There is no provision, or if there is it is marginal, for user interface customisation. However, there is the possibility to create new simulation systems for limited domains with a custom made interface appropriate for the model domain. These capabilities are currently limited to bespoke program-

	XCELL+	Taylor II
Control of statistics collection	System	System, User partially
Default statistics provided	Yes	Yes
Graphics supported	No	Yes
Graph types	NA	Histogram, Pie chart, Bar, Line, Scatter, Gantt, Area graph
User defined presentation	No	Partially
Flexibility of presentation	None	Small
Tabular form statistics	Yes	Yes
Exporting files supported	Yes	Yes
Printing statistics tables	Yes	No
Printing graphs	NA	No
	ProModel	Micro Saint
Control of statistics collection	System, User can only reduce the set	User
Default statistics provided	Yes	Yes
Graphics supported	Yes	Yes
Graph types	Pie chart, Plot, Histogram, Bar, Line, Vertical line and Step graph	Bar, Line and Step graph
User defined presentation	No	Partially
Flexibility of presentation	Small	Small
Tabular form statistics	Yes	Yes
Exporting files supported	Yes	Yes
Printing statistics tables	Yes	Yes
Printing graphs	Yes	Yes
	Witness	Simscript II.5
Control of statistics collection	System	Modeller
Default statistics provided	Yes	NA
Graphics supported	Yes	Yes
Graph types	Histogram, Time Series	Histogram, Pie chart, Line graphs, Dynamic bar graphs, Trace and X-Y plots, Dials, Meters
User defined presentation	Partially	Yes
Flexibility of presentation	Small	Great
Tabular form statistics	Yes	Yes
Exporting files supported	Yes	Yes
Printing statistics tables	Yes	Yes
Printing graphs	No	No

Table 4: Simulation Results

ming that requires a substantial development effort. Sometimes user interface development can be facilitated using an object-oriented approach that reduces development time. An example is Simscript II.5 that provides ready-made user interface object templates like menus, forms, etc. supplied in the C language library. Only ProModel for Windows provides a minimal customisation of modelling environment. There is a facility for guiding an inexperienced user through the necessary steps of model development. Experienced users can choose the order in which to perform the steps of tasks.

Hlupic (1993) provides a comprehensive set of criteria for the evaluation of simulation systems. Even though the criteria are drawn for manufacturing systems they can be used for evaluating more general purpose simulation software if some of the specifics related to manufacturing systems are ignored. She concentrates primarily on modelling capabilities, taking the user interface into account broadly as "user friendliness", "user support" (documentation, tutorials, demonstration models), "modelling assistance" (on-line help, prompting, logic checks), and "visual aspects" (animation, icon editors/library). However, what basis is used exactly to determine "user friendliness", for example, is not elaborated.

Many authors argue that the advantages of VIS include better validation, increased credibility (and hence model acceptance), better communication between modeller and client, incorporation of the deci-

	XCELL+	Taylor II
Tutorial	Not provided	Not thorough enough
Getting started	Part of the User Guide	Not provided
User guide	Yes	Yes
Reference manual	Not provided	Not provided
Index	Only XCELL+ glossary	Global for all manuals
Terminology	Manufacturing	Manufacturing Technical
	ProModel	Micro Saint
Tutorial	Not provided	Yes
Getting started	Yes	Yes
User guide	Yes	Yes
Reference manual	Yes	Not provided
Index	System concepts	System concepts
Terminology	Manufacturing	General Simulation
	Witness	Simscript II.5
Tutorial	Partial	Yes
Getting started	Not provided	Not provided
User guide	Yes	Yes
Reference manual	Not provided	Yes
Index	System concepts	System concepts
Terminology	Manufacturing	General Simulation

Table 5: Printed Manuals

sion maker into the model via interaction, and learning via playing with the VIS. However, there is little published empirical evidence to substantiate these claims. In addition, animation can be used to enhance a model's credibility and, according to Law and Kelton (1991), it is the main reason for animation's expanding use. Swider *et al.* (1994) feel that animation can provide convincing evidence that model behaviour is representative of the system under study. Cyr (1992) see advantage of using animation in its ability to demonstrate problems with the model itself which would otherwise be difficult to detect. Kalski and Davis (1991) point out that summary statistics sometimes do not show the active interactions of processes in a system, and they advocate the use of animation as an aid to the analyst in identifying the system status under which, for example, bottlenecks occur. There are many animation proponents in the simulation community, especially the software vendors, claiming the benefits of animation.

However, there is very little published empirical evidence which would suggest how to design effective animation. Carpenter *et al.* (1993) conducted an experiment with 47 subjects to examine how well the animation communicated the operation of the simulation model. They considered combinations of three aspects of animation - movement, detail of icons, and colour. The results suggested that movement of icons is more important than their detail or colour in communicating the behaviour of a simulation model with moving entities. Swider *et al.* (1994) used 54 subjects to obtain objective and subjective measures in determining which combinations of animation presentation and speed were best for displaying violations of model assumptions. Based on the results of this study, Swider at el. (1994) recommend: the use of pictorial display with moving icons for simulation models with moving entities; the facility to set the presentation speed to make discrete differences visible; and to avoid overloading the user with too much visual information. The results of the above two studies are not surprising and they match our intuition and commonsense. However, their importance is in substantiating our intuitive judgement with some more concrete evidence. Animations with moving icons are often used in current simulation systems even though presentation of animation is not often well thought about. Ideally, it may seem desirable to present information on the screen that has characteristics similar to the objects we perceive in the environment. The visual system could then use the same processes that it uses when perceiving objects in the environment. Graphical means of description must be given preference over written ones because they present information in a more compact manner. Factors that contribute towards the meaningfulness of a stimulus are the familiarity of an item and its associated imagery. The graphical representation of constructs for different applications should give definite information about the type of model component it represents, such as waiting queues, customers or servers in queuing systems or stores, or suppliers in store keeping systems (Kömper, 1993). Designing manufacturing applications, as suggested by Preece *et al.* (1994), might benefit from the use of realistic im-

	XCELL+	Taylor II
Model examples provided	Yes	Yes
Help type	One text screen	Isolated text screens
Navigation through help facilitated using	Not applicable	Mouse and arrow keys
Help text	Short information on system	Complete version of printed material
Index of topics	Not applicable	Yes
Search facility within help	Not applicable	Searches for a first occurrence of a given string
Tutorial	Not provided	Not provided
Context-sensitive help	Not supported	Only relevant page
Help on help	Not provided	Not provided
Printing help text supported	No	No
Demonstration disk	Not provided	Elementary
	ProModel	Micro Saint
Model examples provided	Yes	Yes
Help type	Hypertext	Limited Hypertext
Navigation through help facilitated using	Mouse point and click on link nodes	Mouse point and click on link nodes
Help text	Differs from printed manuals	Differs from printed manuals
Index of topics	Yes	Yes
Search facility within help	Yes	Searches for a first occurrence of a given string
Tutorial	Interactive lessons	Not provided
Context-sensitive help	Yes	Not provided
Help on help	Extensive	Limited
Printing help text supported	Yes	Yes
Demonstration disk	Professional	Yes
	Witness	Simscript II.5
Model examples provided	Yes	Yes
Help type	Isolated text screens	Hypertext
Navigation through help facilitated using	Mouse point and click on cross reference buttons	Mouse point and click on link nodes
Help text	Differs slightly from printed manuals	Differs from printed manuals
Index of topics	Yes	Yes
Search facility within help	Not provided	Yes
Tutorial	Not provided	Not provided
Context-sensitive help	Not provided	Not provided
Help on help	Not provided	Yes
Printing help text supported	No	Yes
Demonstration disk	Not provided	Not provided

Table 6: On-line User Assistance

ages in helping the users design and create objects. However, they anticipate that there might be a problem in the high-cost of real-time image generation and that for the actual needs of an application, such a degree of realism is often unnecessary. Nevertheless, we believe that it can help if some approximations of real life objects are used. Stasko's (1993) animation design recommendations state that animation should provide a sense of context, locality, and the relationship between and after states. Furthermore that the objects involved in an animation should depict application entities and that the animation actions should appropriately represent the user's mental model. If these recommendations were followed, the effectiveness, learnability, and the enthusiasm of a wider user population to use simulation systems might increase. The eye-catching, appealing nature of animation can tempt designers to apply too many facets to an interface. Animation is, however, another attribute in which the often quoted design principle "less is more" does apply. Nevertheless, if the screen design is kept clean, simple, and well organised some redundant information can be quite useful to the user. The moderation principle is something that many simulation system developers should learn about. User interfaces that have screens crowded with too many objects, large numbers of offensive colours and incompatible colour schemes is more of a rule than an exception.

The attribute names used in the examined simulation packages support mainly modelling in manufacturing domain. However, if the problem modelled is from a different domain manufacturing than the modeller has to map the entities of the domain modelled into the manufacturing domain. The analogical mapping that the modeller has to perform throughout the modelling process can cause many problems. Firstly, a heavy demand on the user's memory is required in order to perform constant translation of used objects to what these objects actually represent. Secondly, the concepts that have to be used have no close and natural association with the tackled problem. Thirdly, the tasks that have to be performed during the modelling process may not at all be related to the problem at hand. The effectiveness and learnability will therefore be seriously hindered. This will not promote the user's positive attitude towards the system. An essential aid in model development can be facilitated by selecting model components which are relevant to the model builder's modelling requirements. Pidd (1992b) points out that the graphics components must be directly linked into the simulator itself, so to avoid displays which are not a direct result of state changes in the simulation and that the model builders should be given the freedom to lay out the screen display by use of interaction devices, choosing how to represent the entities as the simulation proceeds from a provided set of icons. Our proposal is that simulation environments should provide model developers with the following:

- Several pre-defined problem domains.
- Facility to create new problem domains.
- A facility to design and/or choose graphical representations for elements in a problem domain.
- A facility to set default values for a problem domain.
- A facility to set defaults for statistical data collection.
- A facility to set defaults for the graphical presentation of simulation results.

5 Conclusions

Traditionally, HCI has focused on how to design for data availability - is the available data legible and accessible? It is the domain actor's task to use the available data to answer questions and detect and solve problems. The perspective of representation design shifts emphasis away from the display of available data as signals to be interpreted, towards a focus on communicating what is signified by the data (Woods and Roth, 1988). Frequently, what data are relevant depends on the state of domain or the intentions of the problem solver. There is very little published material on usability of simulation systems or models. It seems that the simulation community is not particularly interested in evaluating user interfaces of simulation systems, and to examine what changes would enhance the usability of such systems and thus widen the user group. That is rather strange since simulation systems probably include a much wider scope for interaction paradigms than most other computer software. Advances in computer technology, especially in computer graphics, are much more readily incorporated into simulation systems than in others.

References

[1] ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group (1992). *ACM SIGCHI Curricula for human-computer interaction*, Technical report, New York: Association for Computing Machinery, Inc.

[2] Benyon, D. and D. Murray (1988). Experience with adaptive interfaces. *The Computer Journal*, 31, 5, 465.

[3] Bobrow, D.G., S. Mittal and M.J. Stefik (1986). Expert systems: Perils and promise. *Communications of the ACM*, 29, 9, p.880-894.

[4] Booth, P. (1989). *An introduction to human-computer interaction.* Hove: Lawrence Erlbaum Associates.

[5] Bright, J. G. and Johnston, K. J. (1991). Whither VIM? - A developers' view, *European Journal of Operational Research*, 54, 3, p.357-362.

[6] Carpenter, M.L., Bauer Jr., K.W., Schuppe, T. F. and Vidulich, M. V. (1993). Animation: What's essential for effective communication of military simulation model operation?, *Proceedings of the 1993 Winter Simulation Conference*, Los Angeles, California, Association for Computing Machinery, p.1081- 1088.

[7] Conway, R., Maxwell, W. L., McClain, J. O. and Worona, S. L. (1990). *User Guide to XCELL+ Factory Modeling System*, San Francisco: The Scientific Press.

[8] Cyr, R.W. (1992). Using animation to enhance a marine- terminal Monte Carlo simulator, *Proceedings of the 1992 Winter Simulation Conference* Arlington, Virginia, Institute of Electrical and Electronic Engineers, p.1000-1003.

[9] Hix, D. and R.H. Hartson (1993). *Developing user interfaces. Ensuring usability through product & process.* New York: John Wiley & Sons, Inc.

[10] Hlupic, V. (1993). *Simulation modelling software approaches to manufacturing problems.* Unpublished PhD dissertation, London School of Economics, University of London.

[11] Kalski, D. R. and Davis, A. A. (1991). Computer animation with CINEMA, *Proceedings of the 1991 Winter Simulation Conference* Phoenix, Arizona, Association for Computing Machinery, p.122- 127.

[12] Kömper, S. (1993). A systematization of requirements for the user interface of simulation tools, *Systems Analysis and Modelling Simulation*, 11, 2, p.107-119.

[13] Kuljis, J. (1994). User interfaces and discrete event simulation models, *Journal of Simulation Practice and Theory*, 1, 5, p.207-221.

[14] Kuljis, J. (1995) Usability of manufacturing simulation software. *International Journal of Manufacturing System Design*. 2, 2, p.105-120.

[15] Kuljis, J. and R. D. Macredie (1996) Supporting model development through simulation systems. *Journal of Intelligent Systems*. 6, 1, p.25-44.

[16] Larson, J.A. (1992). *Interactive software. Tools for building interactive user interfaces.* Englewood Cliffs: Prentice Hall.

[17] Law, A.M. and D.W. Kelton (1991). *Simulation modeling and analysis*, 2nd ed., New York: McGraw-Hill

[18] *Micro Saint Builder* (1992). CA: Micro Analysis and Design Simulation Software, Inc.

[19] Myers, B. A. and M. B. Rosson (1992). Survey on user interface programming. *Proceedings of HCI Conference on Human Factors in Computing Systems*, New York: ACM, p.195-202.

[20] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T. (1994) *Human-Computer Interaction*, Wokingham: Addison-Wesley.

[21] ProModel for Windows (1993). Orem, Utah: Promodel Corporation.

[22] Shackel, B. (1991). Usability - context, framework, definition, design and evolution. In Shackel, B. and S. Richardson (eds.) *Human Factors for Informatics Usability*, Cambridge: Cambridge University Press.

[23] *Simscript II.5 for Windows User's Manual* (1993). La Jolla, CA: CACI Products Company.

[24] Smith, S.L. and J.N. Mosier (1984). The user interface to computer-based information systems: A survey of current software design practice. *Behaviour and Information Technology*, Vol. 3, No. 3, pp. 195-203.

[25] Stasko, J.T. (1993). Animation in user interfaces: Principles and techniques. In Bass, L. and P. Dewan (eds.). *User Interface Software*, Chichester: John Wiley & Sons, pp. 81-101.

[26] Swider, C. L., Bauer Jr., K. W. and Schuppe, T. F. (1994). The effective use of animation in simulation model validation,*Proceedings of the 1994 Winter Simulation Conference*, Orlando, Florida, Institute of Electrical and Electronic Engineers, p.633-640.

[27] *Taylor II Simulation* (1993). Tilburg, Netherlands: F&H Logistics and Automation B.V.

[28] Thomas, L .J., McClain, J. O. and Edwards, D. B. (1989). *Cases in Operations Management. Using the XCELL+ Factory Modeling System*, Redwood: The Scientific Press.

[29] Visual C++ (1993) *Microsoft. Windows Simscript II.5 User's Manual* (1993). San Diego: CACI Products Company.

[30] *WITNESS User Manual Release 307, Version 7.3.0* (1991). Redditch, UK: AT&T Istel Visual Interactive Systems, Ltd.

[31] Woods, D.D. and Roth, E.M. (1988). Cognitive systems Engineering. In Helander, M. (Ed.) *Handbook of Human-Computer Interaction*, Amsterdam: North- Holland, p.3-43.

Model Development and HCI

Mike Pidd
Department of Management Science,
The Management School, Lancaster University,
Lancaster LA1 4YX, England
Phone: +44-1524-593870, Fax: +44-1524-592417
m.pidd@lancaster.ac.uk

Keywords: simulation, HCI

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

Much of the progress within discrete simulation has gone hand-in-hand with general developments in computing. Recent years have seen software developers putting great efforts into improving the user interfaces of discrete simulation systems. This too parallels developments elsewhere. This paper considers what further benefits there might be for users and developers of simulation software from more careful attention to interface design.

1 Introduction

As computers have grown more powerful, so have users' expectations of how they might be used. Today's school students now assume that computers will be fast, friendly and easy to use. Many of them, certainly in the USA and Western Europe, are video game experts, connoisseurs who are easily bored with experiences that do not live up to their expectations. (See Microserfs, Coupland (1996), for amusing examples of this). These people will be the next generation of simulation modellers and users and they may not be satisfied with the interfaces available in today's discrete simulation software. Thus, it seems important that the designers and vendors of simulation software start to take the question of user interface design very seriously. This paper presents the basic requirements for simulation software that stem from a particular view of modelling that is argued elsewhere (Pidd, 1996a, 1996b). These are linked to current theories and ideas about HCI (Human Computer Interaction or the Human Computer Interface) to present desirable features of future discrete simulation software.

1.1 Historical background

It is probably true to say that developments in discrete simulation software have proceeded hand-in-hand with general developments in computer software. There have been a few cases in which discrete simulation software has led the field, examples being object orientation in Simula (Dahl and Nygaard, 1966) and the introduction of interactive program generators by Clementson (1991) in his ECSL system. However, in most cases, discrete simulation software developers have been willing to pick up and use whatever tools and ideas appear within the computing community. Examples of this being the use of animated graphics for visualisation by Hurrion (1976) and the use of source level debuggers to support program development.

The purpose of this paper is to examine some of the links between simulation modelling and what is now known as HCI. Given that the idea of simulation is to use a computer as a dynamic model of some situation or system, then it would seem that the question of appropriate HCI is an important one. This paper begins by considering the different ways in which discrete simulation software is provided, looks at the different roles which people play in developing and using simulation models and then speculates on how things might be improved. The question at issue is, what can simulation users and developers learn from the field of HCI? In addition, is there anything that the HCI community can learn from discrete simulation?

2 The Use of Discrete Simulation Software

2.1 Types of Discrete Simulation Software

Different authors have their own ways of discussing the various types of discrete simulation software that are now available. Law and Kelton (1991) divide this software into two main groups, those which require programming and the rest. The latter they describe as simulators. The former category includes general purpose programming languages and libraries such as C++ or FORTRAN, as well as dedicated simulation programming languages such as the SIMSCRIPT family. A rather more finely drawn categorisation is found in Pidd (1992), who has no less than seven categories, ranging from 'do-it-yourself' (in a general purpose programming language), through to Visual Interactive Modelling Systems (VIMS) such as Witness (Lanner Systems, latest edition) and Microsaint (Micro Analysis and Design, latest edition), via flow diagram or block diagram systems such as HOCUS (Szymankiewicz, McDonald and Turner, 1988) and GPSS.

This paper uses a simple tripartite classification as follows.

- 1. Software that requires all users to develop some true program code. This includes general purpose programming languages as well as simulation programming languages. It also includes those languages that provide support for visualisation and those that do not.
- 2. Visual Interactive Modelling Systems which are primarily based around some kind of visual motif for virtually all of their functions. In these systems, examples of which include Witness, Microsaint, ProModel and AutoMod, a visual interface is used for model building, controlling simulation runs and for interaction as a simulation run proceeds.
- 3. Layered systems in which the user can operate at a number of different levels, such as visual interactive modelling, automatic program generation, direct coding and low level bit twiddling. A few systems currently available incorporate some of these facilities and an example would be ARENA.

2.2 Groups of people involved in simulation projects

Though simulation conferences typically focuse on those people who conduct the technical aspects of simulation projects, there is quite a range of people who may be involved across the set of activities that make up such projects. These may be different people or might be different roles that are conducted by one or more people. Examples include the following.

- 1. *Modellers*: whose main role is to understand the system being simulated and to capture that understanding in a model that is implemented in some software system or other. Thus the software interface must offer support to modellers as they develop, usually in a stepwise manner, their abstract and symbolic models.
- 2. *Programmers*: who have the task, in some projects, of taking model conceptualisations and realising them in program code. In an ideal world this would enable the programmers, though coding, to operate at a level which relates to the problem being modelled, rather than just the computing aspects of the work.
- 3. *Project managers*: who may be responsible for ensuring that the project meets the needs of the clients, is on-time and is properly conducted in a technical sense. This group of people may wish to establish standards that enable projects to be properly managed and consistently presented to clients.
- 4. *Customers*: who are footing the bill for the project and whose questions are being addressed in the project itself. This concern of this group may be to ensure that they get value for money and this may be aided by suitable HCI considerations.
- 5. *Users*: who may not be the same as the customers, but who may need to use a simulation model on one or more occasions to address issues of interest. This group may not be expert in simulation and may have limited expertise in computing. Thus the interface needs to be very supportive and should build on their previous experience and expertise.

In addition, of course, there may be software vendors keeping an eye on what is happening.

2.3 Discrete simulation modelling

A computer simulation involves experimentation on a computer-based representation of some system or other. Discrete event simulation is suited to systems made up of objects that change state and display dynamic behaviour. That is, those which change state through time. Thus, to build a discrete event model, the analyst must attempt to tease out and represent a number of features of the system being modelled. These are as follows.

1. *The main entities of the system*: that is, those classes of object whose behaviour seems important to the operation of the system. Note that word seems, if the modeller were absolutely sure of the important entities and if this were not a subject for thought, debate and argument, then it might not be necessary to model and simulate the system at all.

2. *The logical operations of these entities*: that is, the significant behaviour in which the classes of object engage as they persist, or appear, or disappear within the system being modelled. Once again, the previous sentence includes a weasel word, significant. This suggests that the modeller must make judgements about what features of the entities' behaviour deserves inclusion within the model.

3. *The logical interactions of the entities*: systems would be very simple to model and to simulate if the objects within them did not engage with one another. It is these complicated interactions through time that determine how the system itself will appear to behave and some of these interactions may be hard to tease out or may occur only rarely.

4. *The statistical distributions*: the entities will engage in activities, alone or whole other object classes and these activities may occur at irregular intervals and for time periods that cannot be precisely determined in advance. It is normal to model these durations and intervals using statistical distributions that are believed to be good representations of the observed behaviour. Thus an important HCI task is to ensure that the interface offers support on this model building, especially that it eases the gradual development of a model.

3 Basic Ideas of Human Computer Interaction/Interface

3.1 Interface design

The main concern of HCI is to ensure that computer software and hardware is well suited to its users and to the tasks that they wish to perform. Early work on HCI tended to focus on the specialised needs of people such as pilots, engineers running large, complex systems and others such as air traffic controllers who worked with computers in situations of great pressure, risk or danger. More recently, the emergence of the Apple MacIntosh and other windowing-type operating systems has brought some of the benefits of the this early work to users of general purpose computer systems. HCI includes the design of hardware as well as software, but for the purposes of this paper, HCI is restricted to the software that enables users to interact with computer systems. This software interface should reflect the characteristics of the people who wish to use the system and the tasks and functions that they have in mind as they approach the computer system.

Lansdale and Ormerod (1994) suggest that the development of good software interfaces requires the co-operation of two groups of people, designers and psychologists. The designer must ensure that the software is able to achieve the tasks which the user may require of it. The psychologist must focus on the user to try to understand how the user might wish to conduct these tasks. This co-operation occurs during task analysis which aims to understand the logical structure of complex tasks, to identify the information needed by the users to carry out these complex tasks, to understand the activities conducted by users with the interface and to appreciate the conditions that may affect their performance.

On the same theme, Norman (1990) suggests that the interface designer must bear three things in mind if the interface is to be well-designed.

- 1. The types of people who may wish to use the system.
- 2. The tasks that these people wish to accomplish.
- 3. The tools that are most appropriate in enabling them to do those tasks.

This is an extension of what Norman (1988) terms 'user-centered design'.

3.2 User centered design

Norman (1988) suggests that designers should take account of seven principles if they wish to achieve a user-centered design.

- 1. *Use both knowledge in the world and knowledge in the head* . Knowledge in the world is that which is available externally, especially visibly and which will suggest what the function of some system component may be. Knowledge in the head is that which the user is able to internalise as they become practised and proficient and for which few cues are needed. Hence, at its simplest level, there should always be enough clues on-screen for the naive user to know what to do and yet the expert user should also feel comfortable and should not be frustrated by a having to step through a mind-numbing routine of protocols to achieve some simple end.
- 2. *Simplify the structure of tasks.*: This requires the designer to take account of the limitations of human memory. This is often conceptualised as short term memory, which is often believed to be capable of holding 7 plus or minus 2 items and long term memory, which seems to develop conceptual frameworks within which new experiences are considered. Thus, software must not overload short term memory, and it should be a good fit with the users' conceptual frameworks. Software that clashes markedly with a user's likely conceptual framework is likely to be difficult to learn and will be error prone in use.
- 3. *Make things visible.* Designers should struggle to ensure that sufficient information is visible to the user to enable them to do what they wish, within the limitations of the software. This clearly relates to the first principle of using knowledge in the world - it requires the designer to ensure that suitable cues are available to the user.

- 4. *Get the mappings right.* This principle requires the designer to ensure that the user can understand the links between the options, the information presented and the tasks that they wish to perform. The links between the users' intentions and the actions open to them in the software need to be made very clear. It also relates to the need to ensure that users have appropriate conceptual models, otherwise they may misunderstand the function of the system and its components.

- 5. *Exploit the power of constraints*: Constraints are those are restrictions that are deliberately built into the software to ensure that dangerous or bizarre actions cannot be attempted. This might mean, on occasions, frustrating the user who wishes to cut corners.

- 6. *Design for error.* This principle requires the designer to assume that the user may make mistakes. It is vital, therefore, to ensure that, if mistakes and slips occur, the behaviour of the software is safe and helpful to the user. It is unreasonable and downright stupid to assume that even experts never make mistakes.

- 7. *When all else fails, standardise.* The idea of this final principle being to limit the amount of learning that the user must achieve in order to accomplish their required tasks.

Clearly, these principles are inter-dependent and overlap with one another. But their overall message is very clear. Focus on the user and make things easy for them. This requires the designer to know who the likely users will be and to understand what they may wish to do.

3.3 Some principles of learning

Kay (1990) builds on Bruner (1962) in suggesting that human cognition is made up of three interlinked aspects as follows, from the simplest through to the more abstract. As learning occurs, an individual moves through these levels.

- 1. *The enactive mentality.* In this, there is a focus on direct experimentation, on trying things out. It relates to very concrete thinking. Expressed in computer interface terms, this requires the user to be aware of exactly where they are in the system and exactly what is open to them at that point. Given that Bruner (op cit) argues that this is the way in which many of us start to learn about new things, this suggests that the interface be designed to ensure that its exploratory use can do no harm.

- 2. *The iconic mentality.* In this, the individual tries to recognise, compare and configure different aspects of their surroundings. This is rather more abstract than an enactive approach in that it suggests that a person may wish to change the world in some way or other and to make simple inferences. In computer interface terms, this means that the user needs recognisable images and objects whose function is clear. This should then reduce the need for learning by enactment and experimentation.

- 3. *The symbolic mentality.* In this, the individual tries to tie together chains of reasoning in a way that may be very abstract. In essence, the person has conceptual models which are applied to the task at hand. There is thus a need to see the connect between different symbols and to understand how they may be linked to achieve particular tasks. In discrete simulation terms, the interface should support modelling.

In these terms, simulation modelling operates with a symbolic mentality, but Kay(op cit) points out that this will not be achieved unless the other two levels are in place. That is, the ability to conduct symbolic reasoning sits on iconic and enactive foundations. Thus, useful discrete simulation software must make provision for all three.

3.4 The effect of the interface

Finally, it is important to bear in mind the comment made by Lansdale and Ormerod (op cit). Though they are insistent that good, user-centered design must be based on a good fit between task, technology and the user, they argue that one other aspect is of crucial importance. This is that the users' goals may not be static through time. There are two reasons for this, one of which may not be obvious.

The obvious point is that the task itself may change for reasons quite unconnected with the software. Its scale might increase (e.g. air traffic controllers might need to deal with a much more crowded airspace) or other changes may take away much of the task (e.g. decisions on the sharing of air space amongst national traffic controllers). Thus a sensible task analysis needs to look at possible changes in the task.

The less obvious point, and the one that matters for discrete simulation, is that the nature of the interface may change the task itself. To use a non-simulation example, consider word processors. They have two main virtues. The first is that they enable any trained user to produce work that is well-presented, hence a conference can provide presentation guidelines that enable authors to meet their requirements for camera-ready copy. The second is that it changes the nature of the task by allowing people to adopt a different approach to their writing. Writing with pen and ink requires a sentence, a paragraph even, to be composed in the head before it is committed to paper. This forces the pen-and-paper writing style to be thoughtful and considered (in the opinion of Stoll (1995)) or ponderous, in the opinion of others. Using a word processor, especially one with a good outliner, allows the user to hammer away at the keyboard with a stream of ideas that can later be expressed as lucid prose. Hence the software, and its interface, can change the nature of the task.

4 Implications for Discrete Simulation Software

4.1 User-centered design

Norman's seven principles can, perhaps, be reduced to four when viewed within the use of today's window-centric world. These are as follows.

- a) Simplify the structure of tasks.
- b) Get the mappings right.
- c) Exploit the power of constraints.
- d) Design for error.

This is not to say that today's windowing operating systems are perfect, far from it. However, adherence to their norms at least helps to enforce Norman's other three principles. A user who is familiar with other windowing software will at least have useful knowledge in the head and will know where to look for knowledge in the world. Such systems are also essentially visual (which presents severe problems for some users of computers with disabilities) and they provide some form of standardisation.

Principles a) and b) above relate to Kay's point about symbolic, iconic and enactive mentalities. That is, they relate to the need to ensure that the conceptual model of the software is abundantly clear in its interface and for it to fit with that assumed by the user. Those of the user are provided, in part at least, by the conceptual frameworks of their long term memory. The interface and its metaphors need to be consistent, clear and must make sense to the user, but this is not easy.

Consider for example, the metaphor that is usually employed in Visual Interactive Modelling Systems (VIMS). In these, the user develops a model by placing icons on-screen and by linking them with directed arcs in a network of some kind, see Figure 1. Though it may not be obvious, there are at least two ways in which this can be done - and this will not be obvious to the naive user.

- 1. Make this a task network. This is the approach employed in software such as Microsaint (op cit). Each node represents a task and each of these tasks may employ several resources which are committed during the duration of the task. Thus, for example, a medical consultation may require the co-operation of a patient, a doctor, a nurse and some specialised equipment. These resources are freed at the end of the task. One of these resources is modelled as the system entity which flows through the task network.
- 2. Make this a machine network. This is the approach commonly taken on manufacturing-oriented VIMS such as Witness (op cit). Each node represents a system resource that will be occupied as a system entity (usually an item being manufactured) passes through it. The node represents a machine that may well be capable of performing several tasks.

These distinctions will be not be obvious to a naive user from the interface nor may they be obvious from simple examples. The user may suddenly be brought to the jarring realisation that their conceptual model is a task network (or vice versa) and the simulation software assumes a machine network (or vice versa).

Principles c) and d) relate to the need for safety. They aim to minimise the risk that the user may make slips (accidental errors and abuses) or errors (deliberate actions that turn out to be misconceived). The interface and its metaphors must be clearly designed with this in mind. In the case of discrete simulation software, if designers wish to take these ideas seriously, then serious issues about various statistical aspects of the model and the simulation need to be taken into account. As a simple example, most VIMS permit the use of Normal distributions, but even sophisticated users may sometimes forget that Normally distributed variates range from plus to minus infinity. In most cases, users and modellers are probably assuming bounded Normal distributions. Whether this is what they will get may not be clear. If it is what they get, the bounds may not be stated anywhere that is easily accessible to the user.

4.2 Needs of different groups

It is probably true to say that early simulation software systems concentrated on the needs of modellers and programmers. The idea being to give maximum support to the technical aspects of the task in hand. That this effort has been successful is evident in the widespread use of VIMS and the growing acceptance of layered systems and approaches. Though the early, first-generation user interfaces of these systems now seem somewhat crude, the latest generation does seem to have taken some of the lessons of user-centered design to heart - assuming, that is, that the technical people are the users.

But what of the other groups of people involved in a simulation project? When a simulation model is used

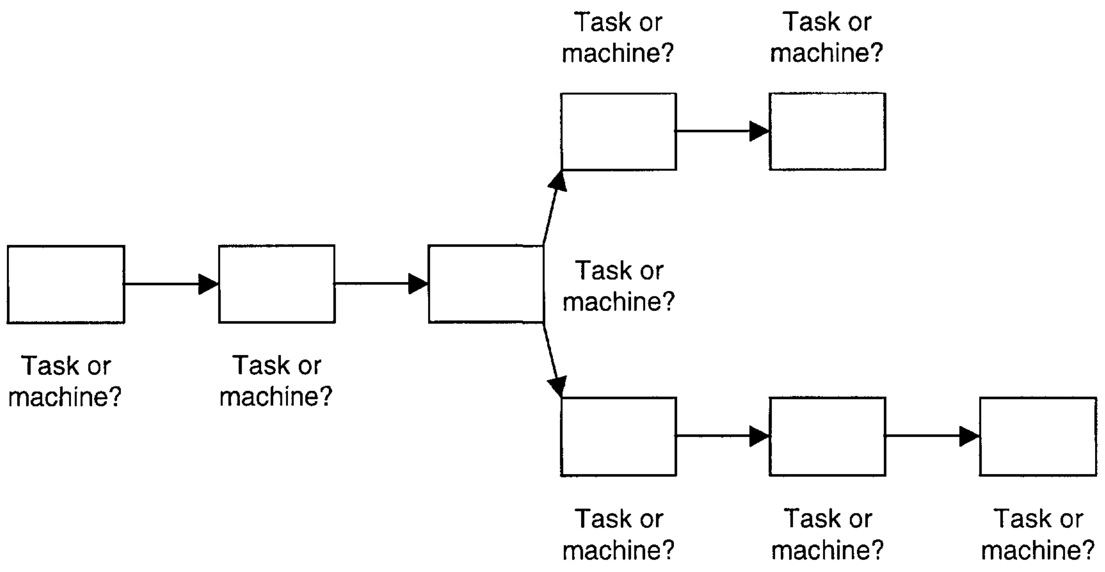


Figure 1: Developing a model by placing icons on-screen

or viewed by someone who is not the developer, the user is modelling - comparing their conceptual model of the system with the perception of what the simulation model is doing. How can this be supported?

The most common approach has been to improve the quality of graphics and visualisation. Welcome though this is, it does carry some problems. Last year the author visited a state-of-the-art engineering company who were planning the development of a highly expensive flexible manufacturing line. As part of this, they developed a discrete simulation which they were keen to show off. The graphics and visualisation were stunning with 3 dimensions, correct proportions, panning and zooming, etc.. But, the simulation itself was, in analytical terms, probably a waste of time. Simple calculations showed where the bottlenecks would occur and, given the discrete nature of the system components, it was pretty obvious what design would be needed. The author has a strong suspicion that this simulation may have hindered a sensible decision making process rather than supporting it. Though graphics and visualisation are important, it must be remembered that their role should be a support to people in their thinking. Sometimes a simple representation may be a better support to thinking than a complex one.

As with user-centered design for the analyst, so too for the other groups, the basic questions concern the tasks to which the different users wish to put the system. It may be that this implies that well designed systems should offer support to these users as they experiment with simulation models. If Bruner and Kay are correct, then these users will wish to enact scenarios in an attempt to learn about the model and to make inferences about the system being simulated. If appearances are deceptive, perhaps superb visualisation is not the be-and-end-all for non-technical users?

References

[1] Bruner E. (1962) *On knowing: essays for the left hand.* Harvard Univ Press, Cambridge, Mass.

[2] Clementson A.T. (1991) *The ESCL Plus system manual.* AT Clementson, The Chestnuts, Princes Road, Windermere, Cumbria, UK.

[3] Coupland D. (1995) *Microserfs.* Flamingo, London.

[4] Dahl O. and Nygaard K. (1996) SIMULA - an Algol-based simulation language. *Comm ACM*, 9, 9, p. 671-678.

[5] Hurrion R. (1976) The design, use and requirements of an interactive visual computer simulation language to explore production planning problems. PhD Thesis, University of London.

[6] Kay A. (1990) Interview. In Laurel B. and Mountford S.J. (1990) *The art of HCI design.* Addison-Wesley, Reading, Mass.

[7] Lanner Systems (latest edition) *WITNESS User manual.* Lanner Systems, Redditch, Worcs, UK.

[8] Lansdale M.W. and Ormerod T.C. (1994) *Understanding interfaces: a handbook of human computer dialogue.* Academic Press, New York.

[9] Law A.M. & Kelton W.D (1991) *Simulation modelling & analysis*, second edition. McGraw-Hill, New York.

[10] Micro Analysis & Design (1992) *Getting started with Micro Saint for Windows.* Micro Analysis & Design Simulation Software Inc., Boulder CA.

[11] Norman D.A. (1988) *The psychology of everyday things.* Basic Books, New York.

[12] Norman D.A. (1990) Interview. In Laurel B. and Mountford S.J. (1990) *The art of HCI design.* Addison-Wesley, Reading, Mass.

[13] Pidd M. (1992) *Computer simulation in management science*, third edition. John Wiley & Sons Ltd, Chichester.

[14] Pidd M. (1996a) *Tools for thinking: modelling in management science.* John Wiley & Sons Ltd, Chichester.

[15] Pidd, M. (1996b) Five Simple Principles of Modelling, in *Proceedings of the 1996 Winter Simulation Conference*, Eds. Charnes, J.M., Morrice, D.J., Brunner, D.T., and Swain, J.J., p. 721-728, Association for Computing Machinery, Coronado, California.

[16] Stoll C. (1995) *Silicon snake oil: second thoughts on the information superhighway.* Doubleday, New York.

[17] Szymankiewicz J, McDonald J. and Turner K. (1988) *Solving business problems by simulation.* McGraw-Hill, Maidenhead, Berks, UK author biography

Soft Modelling Approaches to Simulation Model Specification

Brian Lehaney
Faculty of Business, University of Luton,
Park Square, Luton, Beds, LU1 3JU, England

Keywords: simulation, soft systems methodology, health care systems

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

Simulation is both popular and powerful, but reportage of simulation case studies indicates that in many cases process is treated cursorily, and end-user-acceptance of final models is not forthcoming. Whilst texts often proclaim the importance of process, this is usually left to the discretion of the modeller. A range of problem structuring (or soft) methodologies have been developed to address process issues. However, these can be both slow and unwieldy. This paper outlines a case which utilises the principles of soft methodologies in a relatively quick and dirty approach to process.

1 Introduction

Simulation provides managers with a powerful means to assess the demands on resources created by variable patterns of arrivals and service rates, such as those experienced in hospital outpatients departments. Analytical techniques such as queuing theory may not be of help in such situations, if their assumptions are too rigid or unrealistic or the situation is too complex. However, simulation does not provide a panacea, and much depends upon the way in which it is used. In order to ascertain the appropriate system specification, and model parameters, and in order to assess the value of the simulation, process of discusion an debate must be underataken. Indeed, that very discussion and debate may provide resolution in itself.

Despite the possibility of good intentions, simulation textbooks tend to ignore, or mention only fleetingly, the important processes of problem formulation and logical model development. As Paul and Balmer (1993, p5) note 'Experience of this process of model formulation is not easy to provide in the context of the essentially artificial 'practical' exercises in either a textbook or academic course'. In fact simulation texts give no real guidance as to how this process should be undertaken.

In recent years problem structuring (soft methods) has flourished as an area of academic and practitioner interest. A range of methodologies has been developed, including Soft Systems Methodology (Checkland, 1981, Wilson, 1984). These draw from the following principles:

- emphasis on problem identification, problem structuring, and problem resolution, rather than on problem solution;
- acceptance of multiple problem-perspectives;
- belief that the researcher will affect the situation;
- consensus and participation rather than imposition;
- continual re-evaluation;
- no automatic acceptance of existing structures;
- involvement of those being researched in the research process;
- concern to elicit repressed or minority views;
- challenging approaches to 'norms'.

Whilst the approaches vary in many ways, the arguments in their favour are that they provide:

- structured approaches to problem identification;
- increased sense of model ownership, and hence increased likelihood of model confidence;
- increased probability of implementing recommendations;
- a reference framework which may be particularly useful in tricky situations, especially for the less experienced analyst;
- improved communication through the use of a known modelling methodology;
- an important reminder of the process to authors who are writing up case material - process is often omitted in case articles.

For a number of reasons, Soft Systems Methodology is advocated as potentially the most suitable for combining with simulation for the following reasons:

- each approach provides important features missing from the other;
- the two can be linked in a truly complementarist manner;
- SSM is probably the best known soft method;
- SSM can be used to undertake the important problem structuring phase of simulation, assisting in identifying system boundaries and system activities;
- SSM creates an activity-based model which can be converted to an activity cycle diagram;
- simulation depicts a range of options which assists in assessing their feasibility and desirability;
- simulation produces dynamic models which can be used to investigate interactions;
- SSM enables 'hidden' activities to be identified, by means of issue-based root definitions;
- the use of SSM reduces the chance of implementation failure which results from misunderstandings.

2 What is Soft Systems Methdology?

Soft systems methodology (SSM) is an approach to modelling developed by Checkland (1981) and Wilson (1984), as an alternative to traditional 'hard' approaches, which are based on the prescriptive use of techniques, for clearly defined problems. Such techniques are of little use in the initial analysis of a human activity system, which is likely to have problem areas that are not clearly identified, and which are unstructured. Soft system methodology enables the people involved in running a system (Actors), those responsible for controlling it (Owners), and those who receive its benefits (Customers), to participate in the process of developing a system model, which is likely to encourage acceptability of the model. SSM may be used to aid the identification of system boundaries and system activities, particularly in complex systems. This is particularly useful in a case where a 'hard' technique may eventually be applied, such as in the simulation of an out-patients system. Activity areas within the National Health Service (NHS) are typically 'messy', and particularly suited to an SSM approach, and to the application of simulation.

The general seven stages of soft systems methodology are shown in Figure 1. The methodology is iterative in approach, and is not prescriptive as to a starting point. The unstructured problem situation (1) is described by a rich picture (2). This is used to search for relevant systems which are described by a root definition (3) and conceptual model (4). Although other forms of root definition are possible, those which are both consensus-based, and issue-based are the most interesting and most useful. A root definition describes a system in terms of the following components.

- **Owners**, the roles of monitoring and taking appropriate control actions;
- **Actors**, the roles of running the system;
- **Customers**, the beneficiaries (or victims) of the system;
- **Transformation**, the system inputs coverted to outputs;
- **Weltanschauung**, the world view, the perspective taken in defining the system;
- **Environment**, the constraints on the system.

The above can be formed as the mnemonic CATWOE, and may be used to deter the modeller from omitting important system elements. Any omissions should be deliberate and rational.

3 A Conceptual Combination of Simulation with SSM

Figure 2 shows how SSM and simulation may be combined. Phase 1 comprises the early stages of SSM, in which finding out about the problem situation is undertaken, the problem situation is expressed (rich picture), and the system is described (root definition). From the root definition, a conceptual model is formed which contains the minimum set of activities needed to support the root definition. The conceptual model is compared with the system (Phase 1a), and, where appropriate, the root definition is changed, and a new conceptual model developed. If the activity-level of the conceptual model is too broad, selected activities are expanded until the appropriate level of resolution is reached. As primary task root definitions are used, it is likely that the systems identified within the conceptual model will match those of the organisation. Following temporary entities through the system may result in systems which do not match those of the organisation and which may not have owners. At this stage, if such systems are identified, it may be that control actions are taken. If the organisation appoints an owner of a newly-identified system, for example, this may be sufficient to address any problem situations which have arisen.

If Phase 7 indicates that model output and system output do not match as well as is desired, the route of change must be through Phase 4, as adjusting the

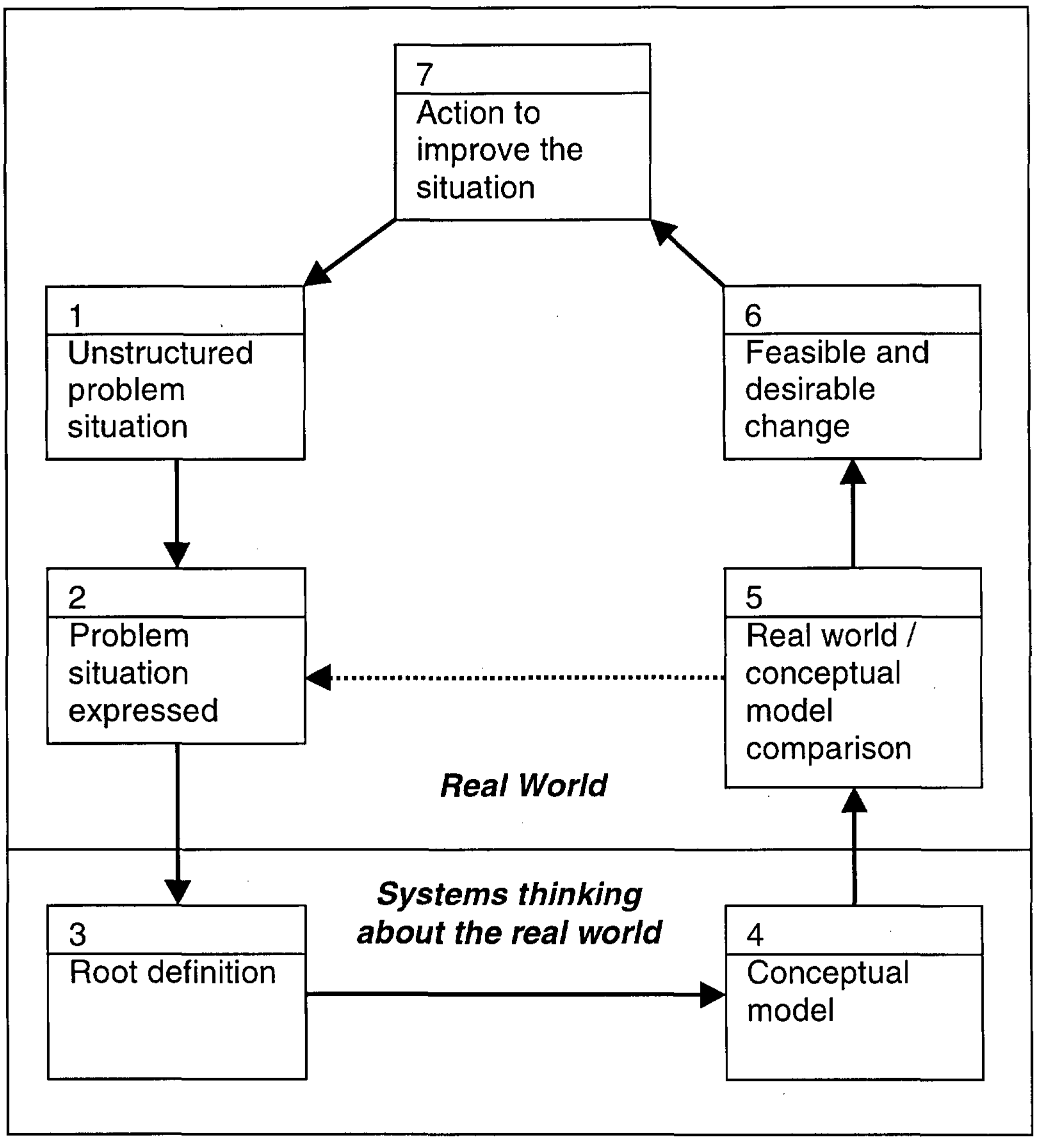

7
Action to improve the situation
1
Unstructured problem situation
6
Feasible and desirable change
2
Problem situation expressed
5
Real world / conceptual model comparison
Real World
Systems thinking about the real world
3
Root definition
4
Conceptual model

Figure 1: The SSM Seven Stage Process

model ad hoc is likely to lead to self-fulfilling validation prophecies. (The modeller must make judgements regarding minor changes.) The route through Phase 4 preserves the integrity of the model, and mimics the 'Rich-Picture-to-Root-Defintion- to-Conceptual-Model-to-Rich-Picture-to-Root Definition' circle of SSM, which should not be broken by direct input from the Rich Picture to the Conceptual Model. (The latter should be built solely from the root definition.)

If the conceptual model development has been undertaken rigorously and with 'good' participation, it should be unnecessary to revisit the early SSM stages in the first iteration. These should, however, be revisited as part of the overall process, as once policy changes have been explored and implemented, the system, being dynamic, has led to another investigation. The cycle then continues.

The concept proposed is appealing in many ways, but it is not without its difficulties. As Mingers (1995) notes, despite Checkland's assurances that SSM is time-independent, it is in fact time consuming. A useful approach is one which enables an accepatble simulation model to be built quickly, but within a framework which embodies the principles of soft methodologies.

4 A Quick and Dirty Case

A hospital outpatients department was the subject of a recent investigation. Table 1 shows the process. The discussions were essentially focus groups geared to patient throughflow. The key stakeholders were easily identified very quickly. From this identification the rest follows.

Table 1 may appear to propose a sequential approach to a complex problem (Note that the initial First Contact was reassigned as a key stakeholder after the first two stages). In fact many of the activities in Table 1 may be undertaken in parallel, and the process is iterative. These activities can be grouped as a conceptual model, the root definition ofr which is as follows:

A system owned by a group of key stakeholders, run by analysts, key stakeholders, and other stakeholders, who use simulation modelling as an aid to develop and implement operational policy which meets internal and external criteria.

Using the CATWOE mnemonic:

- Customer: unspecified
- Actors: analysts, key and other stakeholders
- Transformation: to develop and implement operational policy
- Weltanschaung: by using simulation modelling as an aid, operational policy may be developed and implemented
- Owners: key stakeholders
- Environment: internal and external criteria

The system which is being simulated is the outpatients department. The root definition and conceptual model here relate to a system of resolution which utilises that simulation model, but is not itself being simulated. If it were to be simulated (not impossible) then another overaching root definition and conceptual model for a system of resolution would be needed. The important point is that instead of the simulation being used simply to help solve a single specific problem at an operational level, it is now part of a system to resolve on-going complex situations at a strategic level.

The above process has been used sucessfully to develop a simulation of an outpatients department. The term 'successfully' is used here to denote that the end users have welcomed the recommendations resulting from the investigation, and are taking control actions (patient rescheduling) accordingly. The above process is now seen as an important part of their monitoring and control procedures.

The detailed methodology outlined below should be accompanied by critical reviews of the overall project, its process, its timing, and its outcomes to-date, by means of scheduled meetings of analysts and stakeholders, and by any other means considered to be useful by the analysts and stakeholders. The frequency and duration of meetings may be adjusted as the project develops and in order to ensure that analysts and stakeholders have sufficient opportunity to cover any major issues relating to the project which they wish to raise. Some stages may be undertaken in a different order to that shown below, and some stages may be undertaken in parallel. The terms 'model' and 'system model' refer to diagrams, numbers, and words that describe the analysts' and stakeholders' views of the operations of the system (eg flowcharts). This is distinct from a 'computer simulation', although the latter will be based on a 'system model'.

5 Conclusion

Simulation literature is sparse in the extreme with regard to process, and the principles of soft methods are useful for anyone planning or examining an investigation involving simulation.

SSM is conceptually the most appropriate soft approach to combine with simulation, but it is time consuming and unwieldy.

A quick and dirty approach to developing simulation models which utilises the principles of soft methods may encourage end-user confidence in the early stages, and if an over-arching 'soft' framework is utilised, this confidence is fostered and maintained.

References

[1] Checkland, P.B. (1981) *Systems Thinking, Systems Practice*, Wiley, Chichester.

[2] Mingers, J. and Taylor, S. (1992) The Use of Soft Systems Methodology in Practice. *Journal of the Operational Research Society*, 43, 4, p.321-332.

[3] Paul, R.J. and Balmer, D. (1993) *Simulation Modelling*, Chartwell Bratt, Bromley.

[4] Wilson, B. (1984) *Systems: Concepts, Methodologies, and Applications*, Wiley, Chichester.

Actors	Actions
A, FC	agree the broad nature and scope of the project
A, FC	establish the initial key stakeholders
A, KS	establish the broad area of investigation
A, KS	re-establish the key stakeholders
A, KS	establish the nature and scope of the project
A, KS	agree working objectives
A, KS	agree an initial project process and timetable
A, KS	establish other relevant stakeholders
A, KS	agree initial model of system boundaries and system activities
A	build initial computer simulations
A	run initial computer simulations using 'rough' data estimates
A, KS	evaluate initial computer simulations
A, KS	establish other relevant stakeholders
A, KS	decide roughly how the system could be best examined 'on the ground'
A, KS, ORS	inform ORS of the nature and scope of the project
A, KS, ORS	refine the process of 'on the ground' system examination
A, KS,	agree a timetable to examine the system 'on the ground'
A, KS, ORS	examine the system 'on the ground'
A, KS, ORS	collate information
A, KS	refine the system model
A, KS	refine the nature and scope of the project
A, KS	refine the working objectives
A	refine the computer models
A	run the refined computer models using 'rough' data estimates
A, KS	establish confidence in results
A, KS	refine data requirements
A, KS, ORS	collect and collate additional data
A	run refined computer simulation using refined data
A, KS	experiment with scenarios and analyse results
A, KS	determine the external and internal criteria which policy should address
A, KS	assess which policy options are both feasible and desirable
KS	establish how policy is to be implemented (practical, technical)
KS	implement policy
A, KS	assess the success of policy in meeting internal and external criteria
A, KS	reassess the nature and scope of the project
Key	
A	Analysts
FC	First Contact
KS	Key Stakeholders
ORS	Other Relevant Stakeholders

Table 1: Simulation Modelling Process

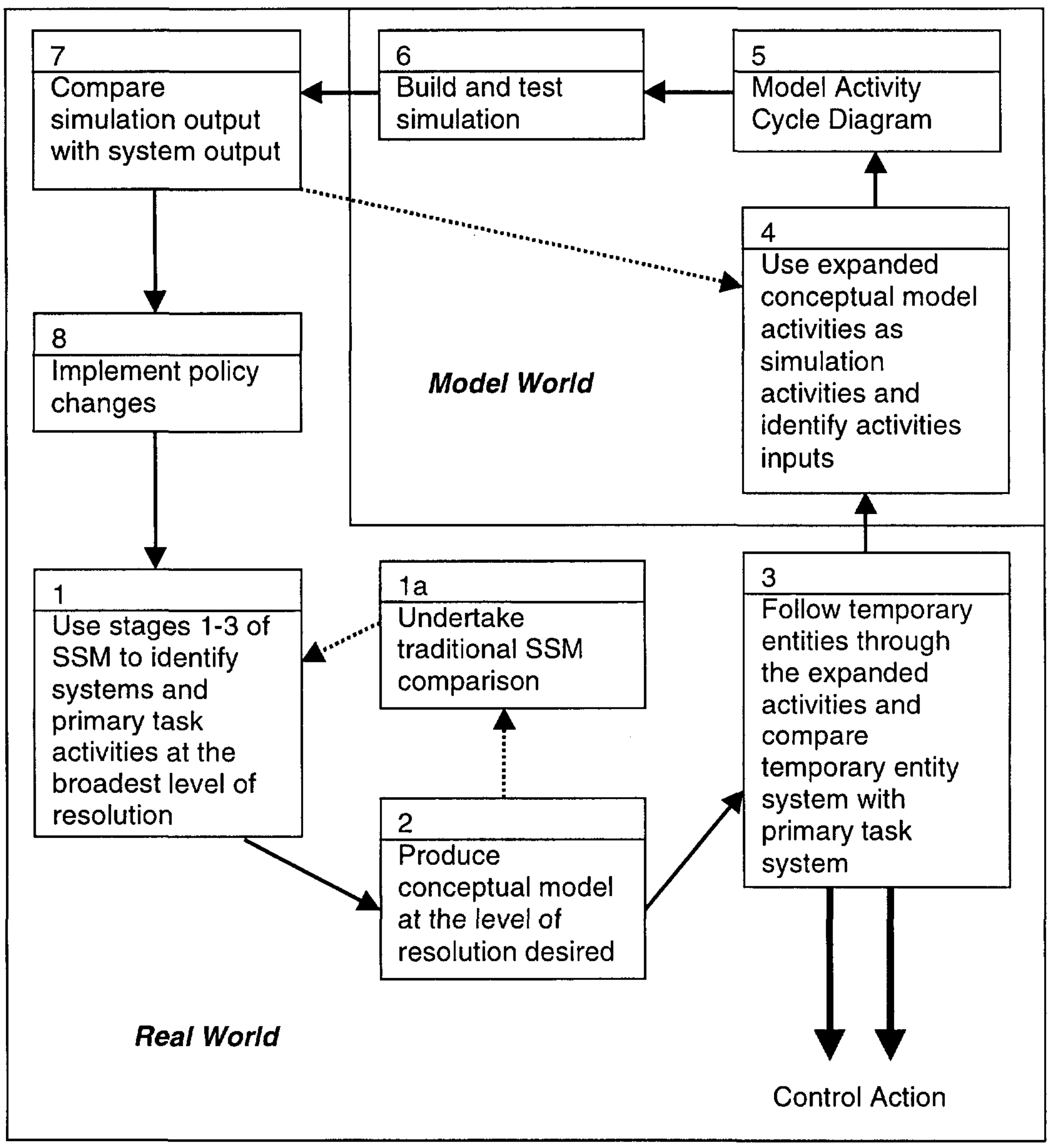

7
Compare simulation output with system output
6
Build and test simulation
5
Model Activity Cycle Diagram
4
Use expanded conceptual model activities as simulation activities and identify activities inputs
8
Implement policy changes
Model World
1
Use stages 1-3 of SSM to identify systems and primary task activities at the broadest level of resolution
1a
Undertake traditional SSM comparison
3
Follow temporary entities through the expanded activities and compare temporary entity system with primary task system
2
Produce conceptual model at the level of resolution desired
Real World
Control Action

Figure 2: A Conceptual Combination of Simulation with SSM

Simulation for Intra- and Inter-Organisational Business Process Modelling

George M. Giaglis and Ray J. Paul
Department of Information Systems and Computing,
Brunel University,
Uxbridge, UB8 3PH, England
Phone: +44-1895-274000, Fax: +44-1895-251686
george.giaglis@brunel.ac.uk, ray.paul@brunel.ac.uk
AND
Georgios I. Doukidis
Department of Informatics,
Athens University of Economics and Business,
Patission 76, Athens 104 34, GREECE

Keywords: simulation, business process modelling

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

Business process modelling (BPM) is an increasingly emerging field of simulation application. Although it has been practically demonstrated that simulation can be an effective tool for business redesign, there does not exist a comprehensive framework to explain the characteristics of business processes and identify specific requirements for their modelling. Furthermore, hardly any attention has been paid to the modelling of inter-organisational business systems. In this paper, we examine the nature of business processes in the light of modern change management approaches and propose a set of requirements for their modelling. We then concentrate on inter-organisational processes and argue that modelling problems can be much more difficult to overcome when more than one business is involved, mainly due to the multiplicity of decision making levels involved and the subsequent need for multi-level output analysis. Based on an empirical study, we illustrate the practical problems of modelling inter-organisational business systems and suggest desirable characteristics of simulation packages for that purpose.

1 Introduction

A multitude of change management concepts have emerged during recent years to help modern enterprises in their efforts to adapt in a constantly changing business, social, and technological environment. The most popular of these approaches include:

- Business Process Re-engineering (BPR) (Hammer 1990, Davenport and Short 1990)
- Continuous Process Improvement (CPI) (Harrington 1991)
- Total Quality Management (TQM) (Oakland 1993)
- Organisational Transformation (OT) (Adams 1984)

Each approach differs significantly in the scope and range of the anticipated changes, the management tools utilised to achieve change, and the business contexts in which they can be used. However, they all have in common that they require businesses to model the ways in which they currently operate, to identify opportunities for change, and to design and implement alternative ways of carrying out business processes. In view of the above, Business Process Modelling (BPM) has recently received widespread attention and has been acknowledged as an integral part of any change management project. Different tools and techniques have been proposed for BPM, and simulation has been identified as an extremely useful tool for this purpose (see for example Tumay 1995, Swami 1995, Bhaskar et al 1994). Despite the availability of these tools, it has been reported that companies are usually facing significant practical problems when trying to model in detail the way they operate (Hansen 1994) or to implement changes in existing environments (Galliers 1994).

Many reasons can be attributed to this difficulty, the primary ones being:

- the complexity of most real-world business processes

- their stochastic and usually unpredictable behaviour
- the interdependencies between individual tasks
- the informal nature of many tasks which makes their analysis and documentation profoundly problematic
- the different perceptions of users regarding the way in which work is being done.

Such modelling problems can become very significant in large, complex organisational settings, especially in cases where more than one business is involved (inter-organisational systems). For example, Business Process Re-engineering projects, where inter-organisational processes almost always play an important role (see for example Riggins and Mukhopadhyay 1994), have been reported to have a large proportion of failures in practice. In this paper, we first examine the nature of the problems of business modelling in general and identify a set of requirements that should be met if a process modelling exercise is to be successful. We then concentrate on the modelling of Inter-Organisational Business Processes and examine their distinct characteristics and requirements for modelling. Simulation modelling is assessed as a potential tool for BPM (at intra- and inter-organisational levels). Finally, an example of inter-organisational business modelling is presented to help clarify some practical modelling problems that might arise.

2 Business Processes: Characteristics and Modelling Requirements

2.1 Process-Based Organisational Approach

Most of the change management concepts identified above (especially BPR) adopt a new look to organisations based on the processes they perform rather than the functional units, divisions or departments they are divided into. Processes are defined as 'structured, measured sets of activities designed to produce a specified output' (Davenport 1993).

Typical examples of major business processes include the purchasing of raw materials from suppliers, the use of these materials to produce goods and/or services, the delivery of these goods/services to customers, the acquisition of new customers, the development of new products according to customer needs, etc. It is obvious that each of these processes requires the co-operation and synchronisation of different functional units in order to be successfully performed. It is also typical for a business process to cross organisational boundaries and extend to third parties (customers, suppliers, etc.). Of course, processes can be described at different levels of detail depending on the abstraction put into analysing the organisation. Typical business processes like those identified above, can be divided into sub-processes which can be further split until the level of individual business tasks.

2.2 Requirements for Business Process Modelling

It has been argued (Willcocks and Smith 1995, Galliers 1993) that businesses and business processes are sufficiently complex systems and therefore carefully developed models are necessary to understand their behaviour in order to be able to design new systems or improve the operation of existing ones. As businesses are essentially 'socio-technical' systems, we can distinguish the basic requirements of the decision makers regarding the modelling process in two separate areas: 'Technical' requirements which refer to those needs that call for the application of engineering principles in process analysis and design, and 'Political' requirements which refer to the needs that emerge from the social nature of business systems. These requirements include:

a. Technical Requirements

- **Formal Modelling** Formal engineering principles should be adopted during the modelling process in order to enable the development of models that can be readily understood and agreed upon by all parties, thus providing a common basis for decision making.
- **Quantitative Modelling** Managers need to have quantitative information that will allow for informed decision making (e.g. cost-benefit analysis) and for direct comparison between alternative system designs.
- **Stochastic Modelling** Modelling should take into account the stochastic nature of business processes, especially the way in which they are triggered by external factors and should allow for representation of and experimentation with situations where a great degree of uncertainty exists. Sensitivity analysis of business models becomes a significant issue in this case.
- **Model Documentation** Models should be easy to document for exchanging information between modellers, analysts, and decision makers. Model documentation can also be used as a reference in subsequent modelling exercises and/or if the model development teams change.

- **Model Adaptability or Reusability** Models should be easily updatable to follow changes in actual processes. Thus, they can be continuously used for future modelling exercises. Reusable models could assist in reducing the cost of model building and can provide an additional means of justifying the initial investment.

- **Objective-driven Modelling** BPM is usually performed having in mind specific business goals to be achieved through the process redesign exercise. The evaluation of alternative configurations is therefore highly dependent on the objectives of the particular study. Business models should reflect this requirement of decision makers and allow for output analysis that can be configured according to objectives so as to provide alternative views of measuring business performance.

b. Political/Social Requirements

- **Feasibility of alternative designs** Modelling and decision making in business contexts should take into account such factors as legislation restrictions, user acceptance of changes, etc. It is not sufficient to simply derive a particular system configuration that seems to optimise business performance, if the changes required in business processes cannot be practically implemented for this configuration.

- **Communication of Models** Business models are often used in brainstorming sessions of business management in order to assist in deciding changes. The models should therefore allow for easy communication of results between different parties. Moreover, generating alternatives and modifying the model during the decision making process is another very important aspect of business modelling, as managers clearly need to be able to interact with the models as new information or ideas emerge during brainstorming sessions.

- **User Friendliness** Modelling tools should be easy to use to allow users of the processes to be personally involved in the modelling exercise. The personal involvement of users should increase the confidence of the whole enterprise in the change initiative, thus enabling a greater degree of acceptability of the derived results.

- **Business Process Modelling** approaches should combine the requirements identified above if they are to be proven useful tools for business decision making. In the next section we will assess the potential of simulation modelling as a suitable BPM technique.

2.3 Simulation as a Tool for Business Process Modelling

Simulation can be an invaluable tool for BPM, as it incorporates characteristics and capabilities that can accommodate all the requirements identified above.

- **Formal Modelling** Simulation is a formal and robust technique. It does not rely heavily on mathematical abstraction therefore it is suitable for modelling even complicated management systems (Pidd 1992).

- **Quantitative Modelling** Simulation is basically a numerical technique, therefore it can be used to generate quantitative output data on various parameters that influence a business system performance. Output Data Analysis, Experimental Design, and other techniques can be employed to ensure a significant degree of mathematical robustness at every stage of a simulation project.

- **Stochastic Modelling** Statistical representation of real-world uncertainty is an integral part of simulation models. Indeed, simulation is perhaps the most suitable modelling technique regarding its ability to capture the dynamic and stochastic behaviour of real systems. Sensitivity Analysis for example can be employed to assess a simulation model's validity with respect to variations in the values of (unknown) system parameters.

- **Model Documentation** The development of a simulation model requires certain assumptions about the real system which can be documented as the model is being developed. Therefore, documentation of simulation models can be a relatively easy task. However, users in practice do not always pay enough attention to the documentation process. Simulation packages which allow for automatic documentation of models can prove particularly useful for this purpose, although they cannot entirely replace the modeller's role.

- **Model Adaptability or Reusability** Simulation models can easily be updated to reflect changes in real world processes. With respect to BPM, a great opportunity exists to integrate workflow capabilities in simulation environments to support not only the modelling and redesigning exercise, but also the actual carrying out of business tasks, thus resulting in highly flexible and continuously reusable models.

- **Objective-driven Modelling** Due to their flexibility, simulation models can be customised to serve multiple purposes of management. Modellers can specify alternative output measures and apply different output data analysis techniques to simulation models, thus allowing for multiple uses of a single business model according to management requirements.

- **Feasibility of alternative designs** Simulation as a process is meant to help with identifying appropriate solutions to complex decision problems. The feasibility of alternative designs in a business context is essentially a step that has to be built into the assumptions made during model development. If certain managerial, legislative, or other restrictions occur, they should be taken into account during the experimentation phase by adhering to these assumptions. In this way, 'political' requirements can be easily accommodated by simulation models.

- **Communication of Models** Simulation models, especially when combined with graphical, animation, and/or interactive characteristics are probably the best means of communicating the essence of a model to managers and decision makers.

- **User Friendliness** Simulation models for business process analysis can be as friendly as their developers choose them to be. In general, simulation allows for a great degree of user friendliness (e.g. through graphical user interfaces) which is supported by the majority of existing simulation packages.

In the next section we will concentrate on inter- organisational business settings and try to understand the characteristics of inter-organisational business processes, to identify additional modelling requirements, and to assess the potential of simulation in this context.

3 Inter-Organisational Business Processes: Characteristics and Additional Modelling Requirements

3.1 Characteristics of Inter-organisational Processes

Problems of BPM become even more difficult to tackle when considering processes which extend beyond the boundaries of a single organisation and include multiple businesses in the value chain. Examples of such processes include purchasing (supplier involvement), shipping (when third parties are employed to transport goods), sales (customer involvement), etc. Inter- organisational involvement might exist even in processes which seem at first to be internal, but require (explicitly or implicitly) the co-operation of third parties. A single organisation cannot control the behaviour of external parties in the way it can with its own resources (people, equipment, etc.). Therefore, the degree of uncertainty is substantially increased with possible implications for the validity of the derived models. Modelling of inter-organisational processes must be exercised with great care to avoid such pitfalls and sensitivity analysis becomes an extremely important issue in this case.

Furthermore, modelling requires extensive data collection and organisation in order to understand and structure the behaviour of the system under study. In the case of external players, such data might not be available, so businesses usually have to rely on additional assumptions that might further jeopardise the validity of the business model.

Decision making becomes extremely difficult in situations where multiple players are involved, since the decisions made by managers in one firm are affected by the (uncontrollable) behaviour of outside parties. For example, a company might decide to adopt a Just-In-Time method of production based on results derived by a, perhaps simulation, modelling exercise showing that such a strategy could streamline the company's operation, cut down on operating costs, and increase timeliness and quality of service to customers. However, the company's suppliers might not be able or willing to deliver raw material at short notice and in small quantities (an essential prerequisite of JIT implementation). If managers decide to adopt such a JIT approach and redesign the respective business processes without a prior assessment of the possible outcomes according to alternative supplier reactions, results might be unexpected. Interactions between players should always be taken into account in inter-organisational modelling and decision making.

There are cases where the decision to model inter-organisational business processes comes from all the parties involved. For example, a company might agree with its suppliers to develop a common model of their trading communication in order to identify opportunities for change. Such a case is presented in the following section. In such a setting, most of the aforementioned problems become less important, since the behaviour of all parties is generally more controlled and input data can be more easily available. The new problem that arises in this case is the multiplicity of decision making levels. When for example, two companies (Company A and Company B) develop a single model to represent their trading interface, decision making can be performed either at a single-site level (e.g. Company A experiments with the model to identify opportunities for change within its own operations) or at a multi-site level (i.e. joint brainstorming sessions of the two companies). In the first case, the model should allow for experimentation only with those parts (sub-models) which represent processes that are performed by Company A (since A cannot control or change the processes of B), and should also allow for multiple output analyses, both at an intra-organisational level (to assess impacts of changes within the company) and at an inter-organisational level (to understand the consequences of individual decisions to the whole system, as these might influence the reactions and future behaviour of Company B).

In the second case, multi-level output analysis is again of paramount importance, since companies need to assess the impact of changes both on their individual performance and in the efficiency of the value chain as a whole.

We can conclude that inter-organisational business processes differ *significantly* from processes limited to one organisation. Although the requirements reported in previous sections for BPM are still valid, Inter- organisational Business Process Modelling (IBPM) requires additional considerations which will be outlined below, after presenting a small-scale case study of Inter- organisational modelling.

3.2 A Case Study of Inter-Organisational Simulation Modelling

The work presented here was part of a wider BPR project aiming at redesigning trading communication between a major pharmaceutical company in Greece (Company A) and its main distributors. The subproject that is described here involved the medical division of Company A which sells small equipment and medical consumables to hospitals and healthcare institutions, via a number of distributors throughout Greece. Each distributor covers a specific geographic area and is responsible for delivering products to customers from its own inventories which are replenished by the main inventory of Company A at regular intervals.

One of the problems identified by the management of Company A (due to space constraints we will be limited only to one of the areas that was actually examined in the project) was the long time it took for a hospital to receive goods from the time it had placed an order. The long lead times were causing customer dissatisfaction and their reduction was considered as a top priority by the management of Company A.

The ordering process was analysed and found to be unnecessarily complicated: Customers placed their orders *either* with Company A *or* with Company A's salesmen who visited the customer sites or directly to the distributors. However, all orders had to be authorised by Company A before the distributors proceeded with order execution. This policy resulted in unnecessary delays as the distributors and the salesmen had to forward the orders to Company A, which gave the authorisation (usually with no amendments) and forwarded the orders again to the distributors. Figure 1 illustrates the process. A simulation model to represent the aforementioned ordering procedure will necessarily include activities performed and controlled by Company A (including Order Receipt, Order Processing, and Order Forward to Distributors), but also activities performed by the distributors (e.g. Forward Order to Company A), by the salesmen, and by the customers. Furthermore, certain activities require a degree of co-operation between parties, thus reducing the flexibility of individual firms to initiate changes in the respective processes.

For example, looking at the model from Company A's perspective, one might want to assess the possibility of changing the ordering process in such a way that salesmen have greater independence and can authorise orders and forward them directly to the distributors, thus relieving Company A from the additional administrative burden. If this scenario is simulated and seems to improve performance in Company A, it might still not be applicable in practice because the distributors might not accept having too many communication channels open to receive orders, as this could influence their own throughput. In this case, the burden of order authorisation in Company A is simply transferred down the value chain to individual distributors. The requirement is for a simulation model that will enable Company A not only to identify the impact of changes in their own performance, but also to measure the influence that the same changes might have on other parties (such as the distributors) in order to be able to more safely 'predict' their reaction to changes.

Another solution might be for distributors to proceed with the orders they receive from customers without waiting for an authorisation from Company A. Such a policy would surely eliminate the delay of orders delivered to distributors and waiting idle until they are authorised, thus resulting probably in reducing the overall order lead time (which is the objective of the whole exercise). However, this is again a policy that has to be implemented by a party outside of the decision sphere of Company A. Distributors might not be willing to adopt such a policy as it would mean an additional level of responsibility for them. Therefore, they have to be persuaded that such a policy would be in their interest too. How can this happen if the model used does not provide output data to assess the performance of the individual players? And what would happen if distributors wanted to experiment only with the parts of the whole model that describe their own behaviour to improve their understanding about the impact of proposed changes on them? If the simulation model is not modular and easy to decompose, they would probably have to build a new model from scratch just to represent the same workings which are already included in the initial model.

A third option for Company A and its distributors might be to co-operate in a joint effort to improve the

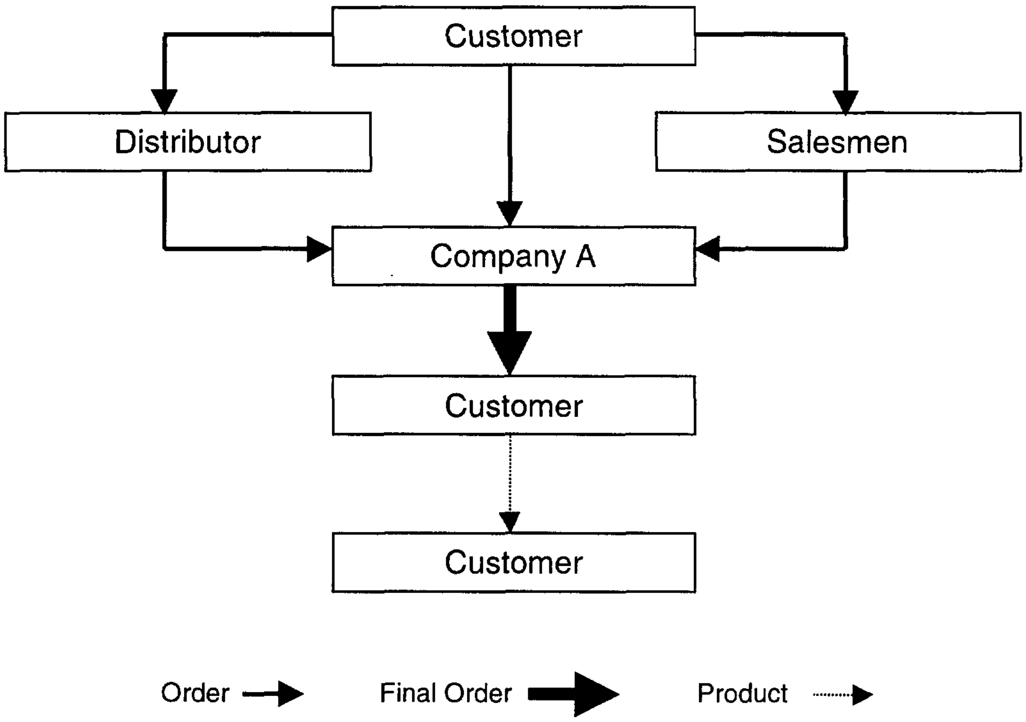


Figure 1: Ordering Process

quality of service they offer to their customers. They might, for example, want to investigate the possibility of adopting EDI to electronically exchange orders, thus reducing unnecessary delays during communications. In this case, the requirement would be for a simulation model that will allow them

1. to fully understand the interactions implied by the new way of communication before committing to changes,
2. to assess their individual performances under the new scheme,
3. to be able to measure the overall improvement in the value chain in terms of total order lead time reduction, and
4. possibly to communicate the results with their customers to persuade them to adopt EDI as a means of order submission as well.

Surely, there are a lot of alternative ways to implement the ordering process in a way that can fit Company A's management objectives. These ways need to be modelled keeping an eye on the influence changes might have on the performance of individual players as well as the whole system. Such a level of modelling and analysis cannot easily be implemented with existing simulation packages, as it requires modular model design and multiple levels of output analysis.

A way to overcome the modelling problems would be to use a general purpose programming language to implement a modular model. An approach is to implement each player (customers, distributors, salesmen, and Company A) as an independent sub- model. Each sub-model communicates with others when necessary (for example when an order is forwarded) via a message passing mechanism which initiates the creation of an entity in the TO model, while it places the respective entity in an idle state in the FROM model. For example, when distributors forward orders to Company A for authorisation, orders remain idle in the distributor submodel, while a 'new' order is created in the Company A submodel. When the order is authorised, it is 'sunk' in the Company A submodel and the respective entity in the distributor submodel becomes busy again (forwarded to be processed).

Although this message passing mechanism facilitates 'independent' modelling of the various levels of decision making of the system and allows for output analysis of submodels, the whole process is clearly not user-friendly and cannot provide the necessary degree of adaptability and reusability required for business models. A far better solution might be to use a user-friendly simulation package for model development, provided that a package to satisfy the aforementioned requirements actually exists on the market.

3.3 Additional Requirements for Inter-Organisational Business Modelling

Based on the analysis of the case study presented above, we can derive the following additional requirements for Inter-organisational Business Modelling:

- Modular Model Design A holistic approach to business modelling is necessary to identify implicit interdependencies among processes, even when they are performed by different organisations. On the other hand, different parties should be able to use suitable sub- models to assess their own performance (but, at the same time, keeping an eye on the influence of 'local' changes on 'global' performance). There is clearly a need for modelling conventions that will allow for modular model implementation and for experimentation with selected sub-models.

- Modular Model Analysis Models that represent the workings of more than one business unit should also allow for a multi-level output data analysis to accommodate the decision making needs of the individual parties involved as well as any of their combinations. Business process configurations that are derived from these models should optimise the performance of individual firms and, at the same time, streamline the efficiency of the whole system. Improvements achieved by individual players should not be lost downstream or upstream in the value chain.

- Model Decomposition and Integration Implementation of modular models should be achievable even if this is not the initial target of the modelling exercise. For example, two firms might develop models independently of each other and at a later stage wish to link these models into a concrete inter-organisational model. To enhance model reusability, individual models should be easy to link, without extensive modifications. In the same way, a single model might need to be decomposed to sub-models, when for example departments of an organisation need to assess their individual performance. Perhaps the only way to achieve problem-free model decomposition and integration, is by defining standard interfaces between models.

3.4 Simulation as a Tool for Inter-Organisational Business Modelling

Simulation models have the potential to become powerful tools for modelling inter-organisational business processes. However, in contrast to intra- organisational simulation, existing commercial products do not generally provide the necessary functionality to meet the requirements identified above.

- Modular Model Design Although simulation as a technique can theoretically be used for modular model development and use, existing simulation packages do not generally include such characteristics. At the time being, the requirement for modular model design can only be met if a general-purpose programming language or a simulation programming language is used to develop the model. We are aware of no existing simulation packages (i.e. software that allows model development and use without or with very limited programming) that include advanced functionalities to assist the user in modular model design. This does not mean that modular model development cannot be done with simulation packages. Rather it means that the user is left alone in performing this task, i.e. the package does not guide him/her towards defining appropriate sub-models that will clearly indicate the 'decision territories' of the firms involved in modelling.

- Modular Model Analysis Things become even more difficult when considering the issue of analysing inter-organisational models for the purpose of decision making. Again the problem seems to be that existing simulation platforms do not provide multiple levels of output analysis. Even worse, in the majority of cases output analysis is not provided at aggregate levels at all, and is usually limited to performance indicators within single functional units of the model (resources, activities, or queues). This however, cannot satisfy the requirement for assessing whole, inter-function extended, business processes which is usually needed in business change projects.

- Model Decomposition and Integration At the current status of non-existence of industrial standards to define the interconnectivity issues between simulation model components, this requirement cannot be easily satisfied. Only if individual models are developed on the same platform, can they be connected in a relatively painless way. If different simulation environments are used, then model integration is usually simply not possible, and a new model should be developed from scratch. Regarding model decomposition, things are slightly better since a single model can always be truncated to include only a subset of its initial range. Even in this case however, the user will probably need to carry out sub-model definition, to define new output analysis measurements, to 'fill' gaps generated by model truncation, etc.

4 Discussion: The Road Ahead

Business process modelling carries several distinct characteristics that differentiate it from other types of modelling problems that simulation has been traditionally used for. This highlights the need for a focused approach to the development of software packages for business process simulation. The requirements identified in this paper are basically meant as guidelines for prospective users or developers of business simulation models. Of course, each individual requirement carries a different weight depending on the objectives and characteristics of a particular change management project. Two avenues for further research can emerge from the aforementioned analysis. The first refers to the development of a formal framework to explain the characteristics of business modelling and assess simulation under a variety of practical business problems. The second refers to a more holistic concept that might be necessary for an integrated approach to business modelling. This approach should combine simulation modelling with other computer-supported modelling, experimentation, and decision making tools in an effort to integrate business process modelling (as part of a redesign exercise) with business performance monitoring (as part of the actual carrying out of business tasks). Computer Aided Business Modelling and Monitoring (CABMM) can be the next target of applications to integrate Simulation, Expert Systems, Workflow Software, and internal business applications in an effort to institutionalise process modelling and measurement in modern businesses.

Acknowledgements

The case study work described in this paper was partially funded by Johnson & Johnson Hellas S.A. as part of a wider Business Process Re-engineering project undertaken by the Electronic Commerce/EDI research group of the Athens University of Economics and Business.

References

[1] Adams, J.D. (1984) *Transforming Work*. Miles River Press, Alexandria, VA, USA.

[2] Bhaskar, R., Lee, H.S., Levas, A., Petrakian, R., Tsai, F. and Tulskie, B. (1994) Analysing and Reengineering Business Processes Using Simulation. In the *Proceedings of the 1994 Winter Simulation Conference*, Lake Buena Vista, FL, USA, December 1994, p. 1206-1213.

[3] Davenport, T.H. and Short, J.E. (1990) The new industrial engineering: information technology and business process redesign, *Sloan Management Review*, Summer, p.11-27.

[4] Davenport, T.H. (1993) *Process Innovation: Reengineering Work Through Information Technology*, Harvard Business School Press, Boston, Massachusetts.

[5] Galliers, R.D. (1993) Towards a Flexible Information Architecture: Integrating Business Strategies, Information Systems Strategies and Business Process Redesign, *Journal of Information Systems*, 3, 3, p.199-213.

[6] Galliers, R.D. (1994) Information Systems, Operational Research and Business Reengineering, *International Transactions of Operational Research*, 1, 2, p.1-9.

[7] Hammer, M. (1990) Re-engineering work: don't automate - obliterate, *Harvard Business Review*, July/August, p.104-112.

[8] Hansen, G.A. (1994) *Automating Business Process Reengineering: Breaking the TQM Barrier*, Prentice- Hall, Englewood Cliffs, New Jersey.

[9] Harrington, H.J. (1991) *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity and Effectiveness*, McGraw-Hill, New York.

[10] Oakland, J.S. (1993) *Total Quality Management: The route to improving performance*, 2nd ed., Nichols Publishing, New Jersey.

[11] Pidd, M. (1992) *Computer simulation in management science*, 3rd ed., John Wiley, Chichester.

[12] Riggins, F.J. and Mukhopadhyay, T. (1994) Interdependent Benefits from Interorganisational Systems: Opportunities for Business Partner Reengineering, *Journal of Management Information Systems*, 11, 2, p.37-57.

[13] Swami, A. (1995) Building the Business Using Process Simulation. In the *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, USA, December 1995, p. 1081-1086.

[14] Tumay, K. (1995) Business Process Simulation. In the *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, USA, December 1995, p. 55-60.

[15] Willcocks, L. and Smith, G. (1995) IT-enabled Business Process Reengineering: From Theory to Practice, In The *Proceedings of the 3rd European Conference on Information Systems*, Athens, Greece, p.471-485.

Virtual Reality and Simulation: an Overview

Robert M. Macredie, Simon J.E. Taylor, Richard Keeble and Xiaoning Yu
Department of Information Systems and Computing,
Brunel University,
Uxbridge, UB8 3PH, England
Phone: +44-1895-274000, Fax: +44-1895-251686
robert.macredie@brunel.ac.uk, simon.taylor@brunel.ac.uk,
richard.keeble@brunel.ac.uk, cspgxxy@brunel.ac.uk

Keywords: simulation, virtual reality, distributed interactive simulation

Edited by: R.J. Paul
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

Virtual environments can be used as an effective tool for training and education. This paper cites many recent examples of such systems and divides current work into virtual environment development and the development of supporting technology for distributed access. A novel supporting technology for distribution is introduced.

1 Introduction

A survey of the recent literature indicates that current research focusing on training is concerned with the development of virtual environments and the development of supporting technology to distribute access. Examples concerning the development of virtual environments show the wide range of applications used for training purposes.

Suzuki *et al.* (1995) describe a driving simulator which allows a user to drive through a virtual town. ROADNET (Katz 1995) is a proposed system which utilises distributed interactive simulation technology to support a network of vehicle simulators. The paper describes the architecture and functionality of ROADNET. Similarly Stevens and Neal (1995) propose the use of distributed interactive simulation to support air traffic control simulation with the National Simulation Capability programme of the Federal Aviation Administration of the USA.

Mizumaki *et al.* (1995) discuss an indoor distribution simulation system based on virtual reality technology. The system is designed to give the user the feeling of touring a substation for routine inspection. It is proposed that the validation of equipment operating procedures, training of unskilled workers and work schedule preparation can be improved by the use of this system.

Merril, Raju and Merril (1995) present a prototype virtual reality surgical simulator for laser iridotomy training. The aim of the work is to help physicians acquire the necessary skills without patient risk. Delingette *et al.* (1995) present a craniofacial surgery simulation testbed which makes extensive use of virtual reality techniques. Virtual reality has been used to train surgeons in new procedures and can determine their competence level before going 'live' (Ota *et al.* 1995). Zeigler *et al.* (1995a and 1995b) discuss a virtual reality medical training system for arthroscopy based on computer graphics and virtual reality techniques.

Coleman (1995) presents the view that leaders in the medical profession suggest that accreditation as a trained physician in the field of endosurgery must begin with a training period in a simulated field. The aim is to develop manual dexterity skills before participating as a clinical assistant to an experienced operator. The Virtual Clinic is introduced as a highly realistic and interactive training system which offers surgeons the chance of gaining clinical expertise before encountering an actual surgical event.

As summarised by Zeltzer, Pioch and Aviles (1995) the US Navy sponsors the Virtual Environment Technology for Training programme which addresses key aspects of this technology. The initial prototype used for evaluation is based on training the officer of the deck on a submarine.

Vegas is a virtual reality application dedicated to the experimentation with evacuation scenarios (Sims 1995). The main goal is to communicate understanding as to the process of a disaster by the use of visualisation to present data generated by the simulator. It is hoped that it will allow users to understand fire and smoke spread. The system has been used to predict egress times from an underground subway station and evacuation times from a cross-channel ferry, as well as from several buildings. Bethel, Jacobsen and Holland (1995) discuss the combination of simulation with virtual reality and visualisation tools to aid training in environmental remediation. Stansfield *et al.* (1995)

focus on virtual reality to train people for small team, close quarters operations.

Stansfield *et al.* (1995) present research aimed at developing a distributed, shared virtual reality simulation. Situational training of inspectors and escorts under programs to verify compliance with nuclear non-proliferation treaties is being used to investigate the efficacy of this technique.

Hollands and Mort (1995) discuss the use of virtual reality to visualise the simulation of manufacturing systems. It is observed that people unfamiliar with the simulated processes quickly understand the dynamics and the logic involved, and those familiar with the real system can immediately relate to the simulation output. The authors indicate that it seems probable that many manufacturing systems will not be suitable for, or benefit from, modelling using virtual reality techniques. However, with the provision of affordable virtual reality tools it is hoped that experience will show if the general manufacturing simulation community will find such features worth the extra modelling effort.

Bienvenue, Curtis and Thakkar (1995) investigate if educational researchers can use virtual environments in the classrooms of the USA for training and learning, and if teachers can look at virtual reality as being a discovery tool rather than a means to demonstrate principles.

Wakefield and O'Brien (1995) discuss the creation of a set of virtual reality and simulation tools for the detailed planning of construction activities.

Bowen-Loftin (1995) reports on the use of virtual environments for the training of NASA astronauts and ground-based personnel for a variety of activities such as repairing the Hubble Space Telescope. It is argued that this technology offers significant cost savings and increased training throughput. The move towards a shared virtual environment is discussed.

Macedomia (1995) describe a network software architecture for solving the problem of scaling very large distributed simulations. Logical partitioning of the virtual environment is accomplished via associated spatial, temporal and functional characteristics of entity classes. A vehicle simulation is given as an example of a practical application. Pullen (1995a and 1995b) discusses Distributed Virtual Simulation, a new technology that extends virtual reality to the networked environment. Future requirements of networking resources are discussed to supporting this proposed system. Faigle *et al.* (1995) discuss how high performance computing can further enhance virtual environments.

The complexity and risks associated with the operation of the power transmission and distribution systems have increased the importance of personnel training for such systems. Garant *et al.* (1995) present a virtual reality training simulator prototype to support this activity. The simulator is implemented on the basis of a distributed client-server architecture.

Mastaglio and Callahan (1995) present a virtual environment called the Close Combat Tactical Trainer (CCTT) which is claimed as the first fully distributed interactive simulation compliant training system. Ongoing work at Brunel University is currently developing underlying technology to support distributed virtual environments. In this paper we present a novel approach to supporting distribution.

2 Towards a Desktop Virtual Environment

Typically, current systems are designed based on the following basic design principles, including:

- there is no central computer for event scheduling or conflict resolution,
- autonomous simulation nodes (computers) are responsible for maintaining the state of one or more simulation entities,
- there is a standard protocol for communicating "ground truth" data; receiving nodes are responsible for determining what is perceived, and
- simulation nodes communicate primarily changes of their state. Dead reckoning is used to reduce communications processing.

A system designed on the above principles suffers from high communication volume.

Object oriented programming is known to be a programming paradigm which deals with complex systems in such a way that maintainability, extensibility and reusability are accommodated (Maffeis 1993). Object orientation provides a natural basis for concurrency since the independency required for concurrent execution is made possible as each object contains its own data and procedures. Inter-object coordination is achieved by message passing.

One approach to addressing the problems mentioned above is to propose a new architecture which takes advantages of the features of the object oriented programming in the hope of reducing the volume of messages. In this paper, we present a prototype architecture with an object oriented structure.

This prototype object oriented architecture is composed of non-autonomous simulation nodes. With an appropriate user interface, the nodes appear as an integrated virtual environment to the user. Simulation entities are modelled as simulation entity objects which are distributed at different nodes and are migrated dynamically between the nodes during the simulation run. This mechanism of object management presents the possibility of reducing message volume. Communication efficiency can therefore be improved by two measures. One is to dynamically group closely related simulation entity objects to a specific node. Alternatively, a point to point communication approach can be used instead of broadcast. The system is currently implemented on a network of IBM compatible PCs, with facilities providing real time communication, an interactive display and interface media.

3 The System Architecture

As an object oriented simulation system, the basic system elements are objects located at different machines. These machines are called nodes and are connected by the network. There are two types of objects in our prototype. One class of object is used to manage system facilities such as keyboard and mouse operation, network operation and display operation. These objects are usually supplied by system support packages. The other class of object is used specifically for simulation. In this paper we focus on the class of simulation objects.

At any single node, there are three major objects: the message handling object, the situation display object and the simulation object.

3.1 The Message Handling Object

The message handling object controls the messages passed between local objects and the network. Unlike current approaches which broadcast all messages on the network (Pullen, 1995), it uses both broadcast and point to point approaches to send the messages to keep the number of redundant messages as low as possible. It broadcasts all messages about the updated states of local entity objects to other nodes via the network. Control is achieved by a point to point communication approach. The message handler also receives the messages from outside the node and distributes them to the proper local objects.

3.2 The Situation Display Object

With update messages, the situation display object is used to display the states of all entity objects distributed at the nodes. A function may be used in this object to restrict some object state display according to the different training scenarios. The display updates with simulation time advance. The update messages are from the local simulation object and the message handling object. The messages from the local simulation object include all changes of its derivative entity objects' states. The messages from the message handling object are the update messages from all other entity objects located at the other nodes. On some occasions "Dead Reckoning" algorithms can be used in this object to reduce the transmission of state updates.

3.3 The Simulation Object

The simulation object at a node contains a series of entity objects which represent physical entities in the real world (a simulation object controls its derivative entity objects simulation in each time loop). It is responsible for creating and deleting its derivative entity objects. It transfers the messages from its derivative entity objects to its other derivative entity objects or to the other objects located at the other nodes via the computer network. Another important task of the simulation object is to control the local simulation time advance. More detail about entity object management and simulation time management will be introduced in the rest of the paper.

4 System Management

4.1 Object management

- 1 Entity object

  In our prototype of system, entities in the real world are modelled by entity objects which are located at different nodes in the system. Each entity object has a unique identity that distinguishes it from other entity objects. During each simulation time loop, each entity object is activated by different messages and the entity object states can be updated. At each node, there is a table recording distribution of all entity objects in the system. When an entity object is created, deleted or transferred to an other node, a message is sent to all nodes in the system to inform them of this change and update the entity objects' distribution tables.

  There is an owner relation between the entity objects. When we say the entity object A is the owner of the entity object B, it means the entity object A controls the entity object B in them system. The ownership can be changed. Each entity object has at most one owner. The ownership is used to judge if the entity object should be migrated or not. To reduce control messages passed over the network, any entity object should be located at the same node as its owner object. An attribute in the object is used to describe the ownership between entity objects.

- 2 Object list

  At each node, a list is used to link all entity objects located at local nodes. During each simulation time loop, all entity objects are activated in the list order by control messages which are transferred from the simulation object. With the entity objects location changing, the list is dynamically changed by the system. When a new entity object is created or moved to a node from another node, it is automatically added at the end of the list as a result of its constructor. When an entity object in the list is deleted or moved out to another node, it also automatically removes itself from the list using its destructor.

- 3 Object Migration

  Reducing the messages passed via the network and keeping the system load balanced can improve the system efficiency. This can be realised by object migration. Here we only discuss entity object migration. One of the basic functions of entity object migration is to group entity objects together which have common ownership. Control messages are passed between entity objects. These messages are used by one entity object to control another entity object. They are only passed between the entity objects which have common ownership. Since messages passed locally are more efficient than the messages passed via the network, keeping control local can improve system efficiency.

  Ownership can be changed during the simulation. When the owner of one entity object is changed, the entity object is immediately moved to the node where the new owner object is located.

  Object migration also facilitates load balancing. Entity objects can be created, deleted, or moved dynamically during the simulation session. At each node, there is an entity object list recording the entity objects located at that node and an entity object location table recording the location of other entity objects at other nodes. A detection object is used at each node based on the above list and table to detect the load states of nodes in the system. When the load of one node is heavy while there is a node with no entity object located, some entity objects can be moved to the idle node to maintain load balance.

4.2 Time Management

Time management is used to control the advancement of simulation time during the simulation session. The simulation object at each node controls time management. To control the simulation time advance, three different time notions should be distinguished in time management.

T_{li} refers to the earliest message timestamp in the ith entity object. T_l refers to the earliest message timestamp in all entity objects. So

$T_l = min(T_{li}|i = 1, ..., n)$

Here n is the number of entity objects in the simulation.

Pseudo real time T_{pr} is a scaled real time. A scale factor f is used to compress or expand real time. It is variable. Different scenarios can set different values of scale factor f. So

$T_{pr} = f * T_r$

where T_r is the current real time. The simulation current time T_c is used for each simulation loop time advance.

$T_c = min(T_l, T_w)$

When $T_c = T_l$, the entity objects process their own messages with timestamp T_l and change the related states. Other entity objects update their own simulation state at this time by using techniques such as Dead Reckoning.

When $T_c = T_{pr}$, entity objects update their own simulation state at this time by again using techniques such as Dead Reckoning

Usually, the logical time is earlier than the pseudo real time, but sometimes, there could be delay caused by nodes processing or network communication processing. To compensate for this, an adjusting value T_a should be added to the pseudo real time, i.e.

$T_a = T_{delay} + T_{process}$

where T_{delay} is message delayed time compared to its timestamp, $T_{process}$ is the time the message object needs to process the message. So, the adjusted pseudo real time becomes

$T_{pr'} = T_{pr} + T_a$

At each node, there is a message list in timestamp order. Each message is processed when the timestamp is the same as simulation time.

4.3 Message Management

Similar to the classes mentioned above, there are two major kinds of messages in the system. One kind of message serves the operating system, i.e. opening or closing a window, expending or reducing the size of window, moving the mouse, etc. The other kind of message is used for simulation execution, i.e. entity objects state update, an event happening, moving an entity object, etc.

For efficient communication, a broadcast approach is used for entity object state update and a point to point communication approach is used to transmit control messages between entity objects which are distributed at different nodes.

The simulation object at each node distinguishes which messages are only used for local entity objects and which messages should be transferred to the message handling object.

Message order affects the whole simulation event order, especially concurrent events. All messages have a time stamp marking effective time. All entity objects will process the messages in time stamp order. This guarantees the processing of concurrent events in the correct order. All message handling objects in the system run in parallel and are synchronised. In each time loop, message handling objects send all messages created by local entity objects to other nodes. When a node has no messages to broadcast, it broadcasts a time advance request message to inform other nodes that it has nothing to send and requests time advance. At the same time, each message handling object holds all messages until it has received messages from all nodes and then transfers them to the display object and simulation object in timestamp order.

5 Conclusion

This paper has introduced virtual reality and simulation within a training context as being a field which concerns interface design and distributed access. We have described a novel architecture which will be used to underpin future in virtual environments.

References

[1] Bethel, W., Jacobsen, J. and Holland, P. (1994) Site remediation in a virtual environment. *Proceedings of the SPIE*, The International Society for Optical Engineering, 2178, p.78-87.

[2] Bienvenue, L. A., Curtis, D. H. and Thakkar, U. (1995) Virtual environments in K-12 learning and discovery: a grand challenge in education? *Computer Graphics*, 29, 4, p.43-4.

[3] Bowen-Loftin, R. (1994) Virtual Environments for Aerospace Training. In *Proceedings of WESCON/94, Idea/Microelectronics*, p.384-7. IEEE, New York, USA.

[4] Coleman, J. E. (1994) Virtual reality in medicine. In *Proceedings of VR '94, the Fourth Annual Conference on Virtual Reality*, p.169-73. Mecklermedia, London, UK.

[5] Delingette, H., Subsol, G., Cotin, S. and Pignon, J., (1994) Virtual reality and the simulation of craniofacial surgery. In *Proceedings of Montpellier '94*. 3rd International Conference. Interface to Real and Virtual Worlds, p. 399-408. EC2, Nanterre, France.

[6] Faigle, C., Fox, G. C., Furmanski, W., Niemiec, J. and Simoni, D. A. (1993) Integrating virtual environments with high performance computing. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, p.62-8. IEEE, New York, USA.

[7] Garant, E., Daigle, A., Desbiens, P., Okapuu-von Veh, A., Rizzi, J.-C., Shaikh, A., Gauthier, R., Malowany, A. S. and Marceau, R. J. (1995) A virtual reality training system for power-utility personnel. In *Proceedings of the IEEE Pacific Rim Conference on Communications*, Computers and Signal Processing, p.296-9. IEEE, New York, USA.

[8] Hollands, R. J. and Mort, N. (1995) Manufacturing systems simulation: mixed mode and virtual reality simulation. In *Proceedings of Fourth International Conference on Factory 2000 - Advanced Factory Automation*, p.651-7. IEE, London, UK.

[9] Katz, W. (1994) ROADNET - distributed interactive simulation applied to driver training, city planning and transportation research. In *Proceedings of the Twenty-Sixth Annual Summer Computer Simulation Conference*, p.936-41. SCS, San Diego, California, USA.

[10] Macedomia, M. R., Zyda, M. J., Pratt, D. R., Brutzman, D. P. and Barham, P. T. (1995) Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. In *Proceedings of the Virtual Reality Annual International Symposium '95*, p.2-10. IEEE Computer Society Press, Los Alamitos, California, USA.

[11] Maffeis, S. (1993) ELECTRA - Making Distributed Programs Object-Oriented, *Proceedings of the Symposium on Experiences with Distributed and Multiprocessor System IV*, USENIX. San Diego, California, USA.

[12] Mastaglio, T. W. and Callahan, R. (1995) A large-scale complex virtual environment for team training. *Computer*, 28, 7, p.49-56. Merril, G., Raju, R. and Merril, J. (1995) Changing the focus of surgical training. *VR World*, 3, 2, p.56-8, 60- 1.

[13] Mizumaki, Y., Morita, S., Asano, K. and Kamiji, N. (1995) Power substation simulation system using virtual reality technique. *Transactions of the Institute of Electrical Engineers of Japan*, 115-C, 2, p.267-72.

[14] Ota, D., Loftin, B., Saito, T., Lea, R. and Keller, J. (1995) Virtual reality in surgical education, *Computers in Biology and Medicine*, 25, 2, p.123-37.

[15] Pullen, J.M. and D. C. Wood. (1995) "Networking Technology and DIS", *Proceedings of the IEEE*, 83, 8, p.1156-1167.

[16] Pullen, J. M. (1994a) Networking for distributed virtual simulation. In *Proceedings of INET'94, the Annual Conference of the Internet Society*, held in conjunction with JENC5, the 5th Joint European Networking Conference, p.241-7. Plattner, B. R. and Kiers, J. P. A. The Internet Society, Reston, Virginia, USA.

[17] Pullen, J. M. (1994b) Networking for distributed virtual simulation, *Computer Networks and ISDN Systems*, 27, 3, p.387-94.

[18] Sims, D. (1995) See how they run: modeling evacuations in VR. *IEEE Computer Graphics and Applications*, 15, 2, p.11-3.

[19] Smith, A. B. (1995) Using virtual reality for the simulation of infrared environments for human training. In *Proceedings of the 33rd Annual Southeast Conference*, p.101-9. ACM, New York, USA.

[20] Stansfield, S., Shawver, D., Miner, N. and Rogers, D. (1995) An application of shared virtual reality to situational training. In *Proceedings of the Virtual Reality Annual International Symposium '95*, p.156-61. IEEE Computer Society Press, Los Alamitos, California, USA.

[21] Stansfield, S., Shawver, D., Rogers, D. and Hightower, R. (1995) Mission visualisation for planning and training. *IEEE Computer Graphics and Applications*, 15, 5, p.12-14.

[22] Stevens, R. J. and Neal, W. J. (1994) Distributed air traffic management simulation: the next application- domain for DIS. In *Proceedings of the Twenty-Sixth Annual Summer Computer Simulation Conference*, p.828-33. SCS, San Diego, California, USA.

[23] Suzuki, T., Yasuda, T., Yokoi, S. and Toriwaki, J. (1994) Construction of virtual town and driving simulation in it using graphics workstation. *Journal of the Institute of Television Engineers of Japan*, 48(10): 1318-25.

[24] Wakefield, R. R. and O'Brien, J. B. (1994) A 'virtual reality' type simulation system for construction automation system development. In *Proceedings of Automation and Robotics in Construction XI, the 11th International Symposium on Automation and Robotics in Construction*, p.239-47. Chamberlain, D. A. Elsevier, Amsterdam, Netherlands.

[25] Zeltzer, D., Pioch, N. J. and Aviles, W. A. (1995) Training the officer of the deck. *IEEE Computer* Graphics and Applications, 15, 6, p.6-9.

[26] Ziegler, R., Fischer, G., Muller, W. and Gobel, M. (1995) Virtual reality arthroscopy training simulator. *Computers in Biology and Medicine*, 25, 2, p.193-203.

Teaching the Fundamentals of Simulation in a Very Short Time

Ingolf Ståhl
Stockholm School of Economics,
Box 6501, S-113 83 Stockholm, Sweden
Phone: +46-8-736-9425, Fax: +46-8-30-4762
iis@hhs.se

Keywords: simulation, education

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

This paper informs about the development of the micro- GPSS system on the basis of feed-back from some 5000 students during a period of two decades, providing a stream-lined simulation system which within ten class room hours of teaching make students prepared to write fairly advanced simulation programs of practical interest.

1 Introduction

Simulation has proved to be a very powerful tool not only in engineering but also in business administration. Against this background it is surprising that most business schools and quite a few engineering schools do not give their students any substantial amount of teaching in simulation. The main reason for this it that student time is an increasingly scarce resource in a very crowded curriculum. Often only a handful of hours can be spent on simulation. This, in turn, often leads to the teaching of simulation being limited to a very broad overview, without allowing for any "hands on" experience. Most teachers with this approach believe that the learning of a simulation language or package would take too much time. If any computer is used, it is limited to the input of data into an already existing model.

This approach has several draw backs compared to allowing the students to actually work with a simple simulation package on concrete problems. In this latter way the student can actively learn the whole process of doing simulation for a problem, from delimiting the actual problem, formulating the question to be answered by simulation, gathering data, outlining the simulation program graphically, coding the program, verifying, validating and documenting the program, running the program a sufficient number of times, doing a statistical analysis for drawing significant conclusions, and presenting the results in a form suitable for a potential user, with a focus on the implementation aspects. In particular, it is important to allow the students to be involved in some kind of modelling, since the modelling aspect, in my view, is of fundamental importance for understanding simulation.

To allow for such an approach to be feasible in a short time, a very simple simulation system is required. Let us already at this stage exemplify the time span that we are discussing, namely ten class room hours and all together at most a week of student time including reading and project work. (We are hence not discussing a full semester course, involving perhaps all together a month of student time.) The question is then what can be accomplished in such a very short time.

I shall in this paper discuss what we have done at two universities in Sweden: namely one business school, the Stockholm School of Economics (SSE), and one technical university, the Chalmer's Institute of Technology (CIT).

At SSE all 300 students will in their third year study simulation, having 10 class room hours of instruction in the micro-GPSS system to be discussed below, spending roughly 10 hours on reading and preparing shorter exercises and spending about 20 hours on a larger GPSS program. This course of ten hours, called *Integrative Computer Applications*, has the particular aim of integrating elements of Operations Research, Corporate Finance, Accounting and Statistics. The focus is on a student project, roughly 100 blocks in size, regarding some kind of cash flow forecast and involving decisions on the purchasing, production, inventory and borrowing policy of a smaller corporation. An example of such a project is presented in detail in Ståhl (1996a).

At the department of Transport Technology at CIT around one hundred students annually do roughly the same thing, also using micro-GPSS, with the main difference that the project work is more directly focused on transportation technology.

2 Why GPSS?

The question then naturally arises why we in the teaching discussed above use a version of such an "old-fashioned" language as GPSS. I shall divide this discussion into two parts: a. Why we use a GPSS-based language, to be discussed in this section and b. why we use our own special version of GPSS, micro-GPSS, to be discussed in the next section. The discussion in this section hence applies to all versions of GPSS, be it GPSS/H, GPSS/PC, GPSS/R, HGPSS, GPSS/V, GPSS/VI, SIMPC, GRAMOS-GPSS, micro-GPSS, etc.

As I see it, the following factors are favorable for GPSS when it comes to choosing a simulation system for an educational task like that one specified above.

- Ease of learning. The very short time available requires a special simulation language. We can hence directly rule out the alternative of doing simulation in a General Programming Language, like C++ or Pascal, or a simulation language that is built on a GPL, like Simula, SLX or YANSL. This point is further strengthened by the poor background in programming, in particular as regards the business students

- The GPSS World View, involving temporary components, like customers, which move through the system to be served by various permanent servers provides a very problem oriented approach which has proved to have great benefits both from the modelling point of view and of making GPSS fun to learn.

- The provision of a great amount of automatic output, like block statistics, server statistics, waiting line statistics, tables, etc. This automatic provision is very important since the novice will generally not know what kind of output is relevant. The study of this standard output is an important part of the learning process. Furthermore one avoids having to spend much of the initial time on print commands.

- The block diagrams make it easy for students to study, discuss and document the logic of a program. The teaching can also be more visually oriented. One can start by looking at the main structure of model before looking at the details of the program syntax. The availability of standardized block symbols is a great advantage. Compared to other simulation systems with block symbols, GPSS has the advantage of having many blocks coming in pairs, with one block being the mirror picture of the block doing the "opposite" thing, e.g. SEIZE and RELEASE. The GPSS symbols are easy to read and generally clearly distinct from each other.

- GPSS is a "fun" language. Simple things can be done in a simple manner. (As discussed below micro- GPSS focuses even more on this.) One important factor in this respect is the unity of program and experiment conditions which is an advantage for the novice, but perhaps a disadvantage for the professional user. Among other special features that are regarded by some as disadvantages, but are advantageous when it comes allowing the student to do interesting things at a very early stage, one can mention the concept of a facility, providing service to only one customer at a time and hence not needing a capacity definition.

- GPSS allows for very compact programs. With only a few blocks one can produce quite interesting programs. The advantage of GPSS in this respect is self- evident when comparing with languages like SIMSCRIPT, MODSIM and SIMULA, but it appears that also compared with languages like SIMAN and SLAM, GPSS will generally lead to shorter programs. The shorter code has the advantage among other things of making the programs easier to "overview".

- GPSS has, due to its long history, the great advantage that there are many program examples and much literature available, in fact over a dozen text books, (Ståhl 1993b and Schriber 1995), including what many teachers regard as the best text book ever on simulation, Schriber's famous Red Book from 1974. The availability of well-documented programs is important for the novice, when doing a project work, since he is then more likely to find a text book example to build upon.

- There are reasons to believe that GPSS with its more than 35 years' tradition is better debugged than other simulation languages. The constructors of one GPSS version can use other GPSS versions for checking correctness. Furthermore, there has been a much higher degree of scientific scrutiny of GPSS than of other simulation languages. The internal mechanism of GPSS is better documented and therefore better understood. The user is therefore less likely to run into strange and unwanted effects in certain situations. (Schriber and Brunner 1995 and Ståhl 1996b) .

- Moderately limited versions of GPSS (allowing for programs of up to 150 blocks) are available at very low cost (20–40) in several GPSS versions (e.g. GPSS/H, and micro-GPSS). The student therefore has the additional incentive of working in a simulation system that he knows he can run on his own PC and continue to use also after the simulation course is over.

- GPSS is a good starting point for later moving to many other simulation systems The existence of GPSS similar packages in object oriented languages like Simula (e.g. GPSSS and DEMOS) makes a switch to some object oriented languages simpler. Since SIMAN contains many basic structures similar to that of GPSS, several of our GPSS students at CIT later learnt SIMAN much faster than their fellow students not having taken the GPSS course.
- Finally it should be stressed that GPSS is a General Purpose Simulation System and that for simulation problems, not only focused on manufacturing, but e.g. combining physical and financial streams, and where one wants to investigate the full implications of uncertainty by many runs, GPSS is often more suitable than systems that are very good for animation of production systems, but otherwise fairly weak.

3 Why MICRO-GPSS?

The development of the micro-GPSS system has been a gradual process, starting already in 1978. The first versions of micro-GPSS were virtually proper subsets of GPSS V, with the main difference being the internal mechanism (Ståhl 1996b). Having taught 200 - 300 students a year for almost 20 years, in courses with a decreasing amount of time allotted to them, but with the ambition of still providing the students with the same amount of simulation knowledge, we have, however, step by step changed the original system into the present one, simplifying it to make it easier to learn and use.

This development of micro-GPSS can be seen as a reaction against the development of increased complexity of the GPSS language over time. One measure of this increased complexity is the ever higher number of block types. GPSS I (1961) had some 20, GPSS III (1965) around 40, GPSS/360 (1967) 44, GPSS V (1970) 48, GPSS/H (version 1, 1977) 58 and GPSS/H (version 2, 1989) 62 block types. This development towards more and more block types has in turn lead to increased redundancy in the way that several blocks carry out the same job. This complexity has also made GPSS increasingly more difficult to learn and has lead to students learning a steadily decreasing share of the full language and therefore making more logical errors. Although many of the developers themselves (Gordon 1979 and Henriksen 1983 and 1985) have raised critique against this development, it has yet continued due to the developers' interest in compatibility with older versions of GPSS. Such compatibility is of interest to old users who have already done a lot of programming in earlier versions of the language, but it is of no interest to the novice who desires to have as easy a learning process as possible with regard to the goal of being able to do a certain kind of simulation.

During the process of development, we have spent much time reflecting on the design. Questions have arisen not only with regard to how to cut out a pure subset of GPSS/360, GPSS V or GPSS/H, but also whether or not we should change this or that feature of GPSS. I have in the process tried to formulate the general principles that have guided us in this work (Ståhl 1993a). The main ones are the following, divided into three groups, dealing with A. ease of learning, B. ease of use and C. safe programming.

A. Ease of learning

A1. The main aim is that the students after a very short time shall be able to do simulation projects of practical value. First, as mentioned above, we have only a very limited time of teaching at our disposal.. This has lead to our aim that one should be able to learn most of micro- GPSS in less than ten hours so that one after that can do some student projects of practical interest. Some time should also be left for the issues of collection and evaluation of input data, the main principles of experimental design, statistical analysis of the output, aspects of implementation, etc., i.e. issues that are left out in many courses where all time is spent on the mechanics of a difficult-to-learn simulation language.

A2. As we want the students to focus on modelling (and experimentation), and not on syntax detail, the language should be such that one in the course does not have to learn a new block type every time that a new and different thing shall be done. It is from a pedagogical point of view often preferable that the new aspects can be handled using already known block types, even if the program thereby becomes a few blocks longer.

A3. When students frequently make the same mistake, one must always consider the alternative of changing the syntax instead of forcing them to learn strange features that, for example, might depend on hardware limitations of computers in the 60's or on pure mistakes made by developers in the early years of GPSS. Since we, in contrast to GPSS/H, are not bound to compatibility with earlier GPSS versions, we have in this way over the 20 years of teaching allowed our 5000 students to change the language syntax. Some examples are given in section 5 below.

A4. We have in micro-GPSS been very keen to keep and increase the "fun aspect" of GPSS. If the simulation system is fun to learn from the beginning, this will in itself provide strong incentives for further learning. It is here important that the language provides a possibility for the students to do interesting things after only a very short period of learning. Preferably the students should already after one or two class room hours be able to write some non-trivial simulation programs, i.e. students should be able to do simple things in a very simple fashion. One should not sacrifice the ease of introduction for the sake of having sophisticated features for the advanced user. One should in particular restrict what appears as unnecessary details, e.g. avoid commas that are not absolutely essential. (See section 5 below). It is important to "encourage users to forge ahead and experiment rather than present barriers that lead to discouragement" (Banks 1995). It should be mentioned that this positive aspect of learning has given the micro-GPSS courses increasingly favorable student ratings, much higher than for comparable computer courses.

A5. We have also developed micro-GPSS with the aim of facilitating the teaching in computer labs as well as facilitating self-studies in front of the computer, instead of learning by lectures in ordinary class rooms or text book studies. In this connection it is important that micro- GPSS allows several programs to be run in a stream, with both program listing and output presented one screen at a time, without the student having to leave the GPSS system.

A6. micro-GPSS was developed with the aim of being so simple that it could be completely covered in a pedagogical manner, with many examples in a book of reasonable size, say a maximum of 400 pp, and hence with a moderate price. This was accomplished with Stahl (1990). The student should not have any need for a difficult-to-read manual in order to find features not covered in the text book or in class.

A7. For micro-GPSS it has been very important not to presuppose any pre-knowledge of programming. This is one difference between how micro-GPSS and GPSS/H are positioned. As mentioned in Henriksen (1985), the focus of GPSS/H is on persons with considerable programming experience. Most of our business students do not have any programming knowledge except possibly some very rudimentary knowledge of (Visual) BASIC. Even for my technical students at CIT, who some year earlier had programming in Pascal, this presupposition has appeared to be advantageous.

B. Ease of use

B1. It should be very easy to do the coding of the programs. Since there are great variations between students as regards their preferences for input systems, one should provide a choice between conventional coding using an editor and coding by choices in a menu, either GUI based or text based. For easy coding by an editor, micro-GPSS has in contrast to other GPSS versions a completely free format so that students do not have to worry about the fact that certain words have to start in a certain column; every line can start in column 1. In contrast to GPSS/H, no distinction is made between upper-case and lower-case letters. As regards coding by use of a menu in an interactive dialogue mode, there are two systems for micro-GPSS, our own GPSSMENU and GPSSGUI for DOS and GPSSEDIT for Windows (Nywall 1992). Finally, these input systems aim at making syntax rules self-evident or easily available, so that the student does not have to refer to a text book..

B2. It must be easy to read the program listing and the output. micro-GPSS does not provide a lot of advanced output that the novice does not know how to read and would find confusing. In contrast to GPSS/H the output always fits into an 80 column screen. It should also be easy to read the extended program listing provided by the system. micro-GPSS provides a neat and easy-to-read program listing, with an automatic line-up of operators and operands, independent of the appearance of the original code

B3. Since it in teaching is important to focus on block diagrams, it is first of all essential that every block type has a corresponding block symbol. This is the case of micro-GPSS, but not of GPSS/H. Furthermore it is in this context important to have facilities for the automatic generation of block diagrams, in micro-GPSS by GPSSDIA, as well as systems for coding by clicking in a menu of block symbols (see B1).

B4. Since we also want to focus on experimentation, it is essential that it is very easy to make replications of the runs and also to make a statistical analysis of these repeated runs. Running a program 20 times with different random numbers is done by SIMULATE 20.

C. Safe programming

C1. Closely related to ease of learning, but also to ease of use, is the principle of safe programming. We want to minimize the risk of logical errors, i.e. that that program produces unwanted and erroneous output. In this way a great amount of student time spent on debugging can be saved. If the simulation language is made as safe as possible with regard to logical errors, this also reduces the need for an extensive debugging system which, in turn, requires a lot of student time to learn. To secure safe programming we want to stress the "Lead us not into temptation" principle, implying that the simulation language should not be excessively permissive, allowing constructions that with a significant probability lead to logical errors. It is better that an unsuitable construction leads to a syntax error message and execution stops right away than have it lead to a difficult-to-find logical error. micro-GPSS is therefore much less permissive than ordinary GPSS; e.g. servers must have symbolic names, parameter usage is restricted, etc.

C2. It is also important that the simulation language has an extensive error trapping system with as clear error codes as possible. This error system is very developed in micro-GPSS. All of my 5000 students have been asked to report on errors with no or an unclear error message. This has lead to constant improvements.

C3. Closely connected with the idea of C1 is the aim that students should not run into surprises and unexpected logical errors due to not having learnt the full language. It is of special importance that the language does not have any reserved words, in particular reserved words that lead to strange logical errors. We must avoid problems such as, e.g. that SEIZE XID1 and SEIZE XJD1 in GPSS/H lead to completely different results. Unless the student knows that XID1 is a reserved word with special meaning in GPSS/H, he can make a serious logical error.

4 Selection Of Micro-Gpss Blocks

On the basis of the principles discussed above we have developed a GPSS version with much simpler syntax. For example, instead of the 62 block types of GPSS/H we have arrived at only 22. These 22 block types can be seen in figure 1 below. The question is then how we arrived at these 22 block types. This is discussed in detail in Ståhl (1992b), but the main ideas are as follows:

With few block types there is less to learn. It is also in line with the "modelling" principle of point A2 above. Speaking, to the contrary, for the inclusion of more block types is the idea that short and compact programs in general are preferable to longer programs as regards solving a specific task. With a new block type, one can sometimes with one block do what would require several blocks to do with the earlier existing block types. This is one factor that has lead to the continued increase in the size of the GPSS language mentioned above.

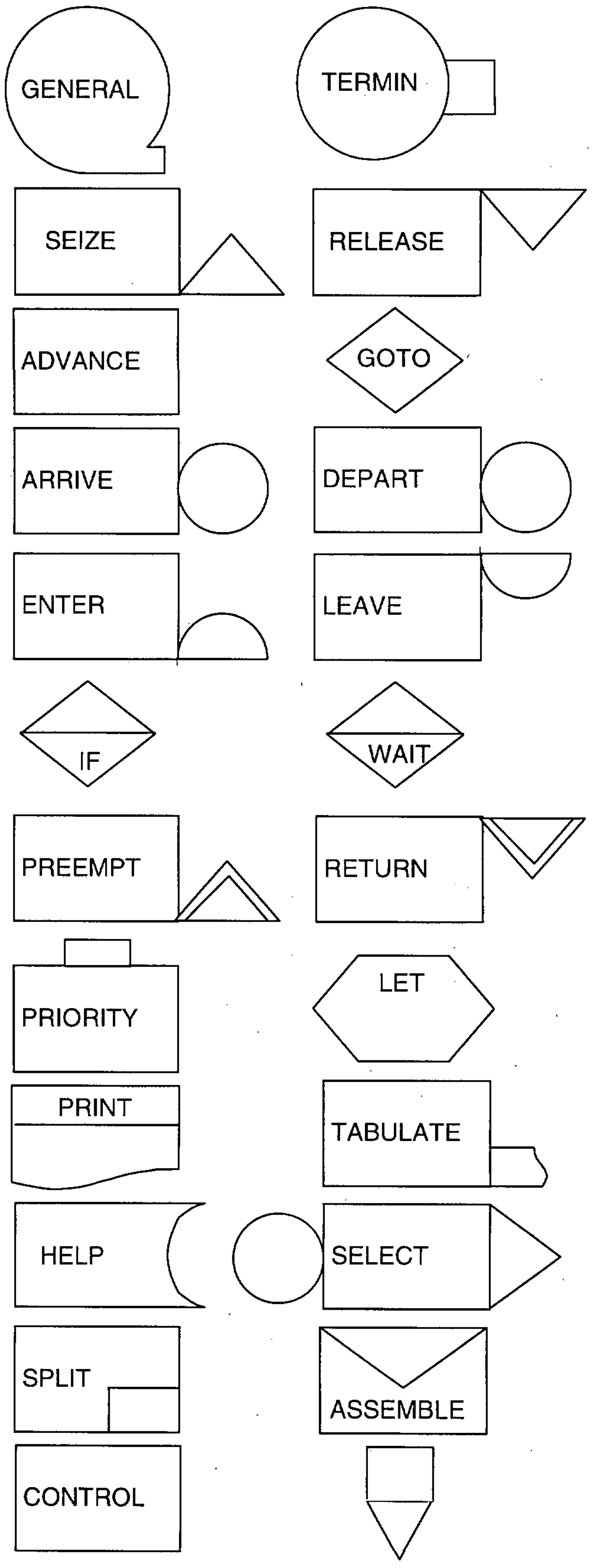


Figure 1: Menu of micro-GPSS blocks

A critical issue is then when to adopt a new block type, if this leads to shorter programs. Our general principle was that we started with a quite small "compulsory" core set of block types. We would extend this set with another block type only if this adoption would lead to significantly shorter programs as regards applications that appear to have a frequent usage. When establishing what kind of applications have a fairly frequent usage, the only alternative available to us was to look at the examples in existing GPSS text books. If applications were not dealt with in these text books they were not regarded to be in frequent usage. Furthermore, we did not adopt a new block type, like LOOP of "old" GPSS, if the use of one block of such a type could easily be replaced with two blocks of already adopted types.

Another consideration speaking for limiting the number of block types was our interest in easy-to-use programs for coding via a dialogue menu, with, for example, block symbols on which to click. With 62 block types like in GPSS/H, the menu itself would hardly fit readably into one computer screen. The 22 block symbols of micro-GPSS fit nicely into the left fourth of the screen as seen in figure 1, produced by GPSSGUI. A final consideration favoring the limitation of the number of blocks is the possibility for the micro-GPSS user to create his own new block types from one or several existing block types (Ståhl 1993c).

The set of 22 block types has proved to be quite powerful and micro-GPSS programs are in general not longer than corresponding programs in other GPSS dialects. The reason for this is that the increase in the number of blocks due to fewer block types is compensated by some new micro-GPSS features. The most important one is that for collecting statistics in a queue, e.g. in front of a facility, we can just add ',Q'. Thus the micro-GPSS block SEIZE SAL,Q does the same thing as the following blocks in ordinary GPSS: QUEUE SALQ, SEIZE SAL, DEPART SALQ.

We have hence been able to rewrite 99 percent of the programs in leading GPSS text books with almost the same amount of code (at most 20 percent difference). For example, for the 29 programs in Schriber's famous "red book" from 1974 the average number of blocks used is virtually the same (18.6 in Standard GPSS and 18.8 in micro-GPSS).

5 Some Other Micro-Gpss Features

It should first be mentioned that of the 22 block types, five do not exist in "old GPSS", namely ARRIVE, GOTO, IF, LET and WAITIF. ARRIVE, the natural counterpart of DEPART, replaces QUEUE and has (almost) the same syntax; GOTO replaces TRANSFER, but with a simpler syntax. Instead of TRANSFER ,BYE (with BYE as B operand) and TRANSFER .4,,BYE (with BYE as C operand), micro-GPSS uses GOTO BYE and GOTO BYE,.4,.with BYE always as A operand.

LET replaces ASSIGN and SAVEVALUE (and BLET in GPSS/H). IF replaces TEST and GATE blocks with an address, while WAITIF replaces TEST and GATE blocks without an address. In contrast to TEST, IF works with a "straight logic", namely that we proceed to the address in the C operand, if the stated condition is true (not false as in "old" GPSS). In micro-GPSS we write IF P1 less than 0,BYE instead of TEST GE P1,0,BYE as in "old" GPSS versions, when we go to BYE if P1less than 0.

Among the many improvements as regards the block syntax , one can here mention that the PRINT statement is much more versatile in micro-GPSS, allowing us e.g. to write PRINT 'PROFITS ',X$PROF. Matrix handling as well as reading and writing the contents of the matrix to or from a file is also handled in a very compact and simple manner in micro-GPSS. Furthermore, the production of an output graph can be handled by a single block. Standard Numerical Attributes can be given easy- to-remember names. The SELECT block has a more understandable syntax.

Also as regards control statements, micro-GPSS requires a lot less learning. Instead of the 34 control statements of GPSS/H, micro-GPSS has only 13, where CAPACITY replaces STORAGE for the definition of capacity and SEEDS replaces RMULT, since it changes the seed, not the multiplier. SIMULATE has, as mentioned, an A operand determining the number of runs. Micro-GPSS also has facilities for automatic statistical analysis (e.g. of Student's t-distribution) and for optimization. It has several built-in statistical functions. Micro-GPSS also allows for animation by a very simple interface to Proof Animation (Ståhl 1992a). and for simple tracing and step by step animation through the block diagram.

The whole of micro-GPSS syntax is presented in Ståhl (1990). A shorter introduction to micro-GPSS is provided in Ståhl (1995) for which there is a tutorial diskette with restricted versions of the micro-GPSS system and GPSSDIA as well as 49 program examples. This contains roughly the material covered in the ten hour course presented above.

Due to these and several other simplifications, micro- GPSS is much easier to learn than traditional GPSS. We have thus with micro-GPSS been able to cover the same material in 10 hours that required 22 hours when using "old" GPSS. Micro-GPSS has also in tests compared favorably as regards learning time with other systems such as SIMAN and WITNESS.

6 Two Program Examples

To give some idea of the simplicity of micro-GPSS we shall present two program examples. The first program, which will produce block statistics, facility statistics, queue statistics and a table on waiting times so that we, for example, can say what percentage of the 30 patients had to wait more than two hours for the doctor. I have not found any other simulation language which with so little code can do so much. The program is run 5 times with different random number streams. This is a program that students can write already after two hours of teaching.

```
simulate   5
qtable     doc,0,10,20
generate   18,6
seize      doc,q
advance    25,5
release    doc
terminate  1
start      30
end
```

The next program will illustrate a problem that appears to be much more difficult to program in many "modern" simulation systems.

"At a small store customers arrive at a rate of every 4 to 10 minutes (7 + 3 minutes; assume a rectangular distribution for all time data.) In the store there are two people working, Boris and Naina. Customers first go to Boris and choose the goods and find out how much they have to pay. This takes between 3 and 7 minutes. Next they go to Naina to pay for the goods and obtain a receipt. This also takes between 3 and 7 minutes. Finally, they return to Boris to pick up their goods after presenting the receipt. This takes between 1 and 3 minutes. There is one waiting line in front of Boris and one in front of Naina. Customers returning to Boris to pick up the goods have to start at the end of this line again. The program should be written so that times spent by customers in the store can be easily measured. How many customers will spend more than 15, 20, 25 minutes, etc. in the store? Assume that the store is closed after eight hours and that the mentioned statistics refer to customers having left the store at this closing time. Calculate also by repeated runs whether there is any significant risk (e.g. happening in one out of ten cases) that a customer has to spend more than an hour in the store."

```
simulate  20
qtable    store,0,5,20
generate  7,3
arrive    store
seize     boris
advance   5,2
release   boris
seize     naina
advance   5,2
release   naina
seize     boris
advance   2,1
release   boris
depart    store
terminate
generate  480
terminate 1
start     1
end
```

This problem has been solved in micro-GPSS in ten minutes by students who have studied micro-GPSS for three hours. I have at earlier Winter Simulation Conferences asked sales representatives of various modern animation based simulation packages to solve this problem and it has taken them over half-an-hour to do so in their system. The difference is that in GPSS you only need a simple sequence of three SEIZE-ADVANCE-RELEASE, while these other systems require Boris to be located at one spot and the program must then for each customer keep track of whether she comes to Boris for the first or the second time, something which is difficult, obviously not only for the novice.

7 How To Proceed From MICRO-GPSS

As discussed above, students can proceed fairly far in simulation after ten class room hours of micro-GPSS. The most clear proof that our business students after these studies can do fairly interesting simulation is given by the project work mentioned in section 1 above. The students here write a GPSS program involving on average one hundred blocks regarding the operations of a smaller corporation. An example of such a project with the micro- GPSS program is presented in Ståhl (1996a).

The question is then what long term benefits the students can have of this. Even if the students never do any more simulation there are clearly several valuable lessons to be learnt. One clear benefit is that simulation illuminates the connection between the actual physical processes and its counterparts in the accounting system. This has a pedagogical merit and has appreciably increased our business students' understanding of the relationship between accounting procedures and the physical and financial flows in a company. Another benefit is that the students in the course get a clearer understanding of concepts such as pseudo-random numbers, exponential, normal and Erlang distributions, experimental design, etc. than they could get by just reading about it. They also become better informed buyers of simulation services, knowing that simulation in certain situations can be a powerful, but yet fairly inexpensive, tool.

Many of our students have, however, proceeded in simulation. We have at SSE an optional longer course in simulation, focused on a larger project done in a Swedish corporation. In this course, the remaining parts of micro- GPSS are covered as well as Proof Animation with the micro-GPSS interface to this. There is also a focus on verification, validation, documentation and implementation. Quite a few of these student projects have been continued on a consultant basis or as a Master's thesis. It should be mentioned that all students have stayed with micro-GPSS. There has obviously not been any need to migrate to GPSS/H, in spite of the fact that we supply a program GPHM which can translate almost any micro-GPSS program into GPSS/H code.

Most of these projects have been fairly small in size, 200 - 300 blocks. The longest micro-GPSS program is 500 blocks. This gives an idea of how we see micro- GPSS positioned. Since micro-GPSS, as well as GPSS/H, in contrast to e.g. HGPSS (Claeys et al., 1995), does not allow any hierarchical structure, we see micro-GPSS positioned as not only a pure teaching device, but also as suitable for rapid prototyping in simulation, for smaller models in line with Woolsey's "Quick and dirty" approach. When it comes to larger models than the 300 - 500 blocks discussed here, object oriented simulation languages like SIMULA or MODSIM III are clearly preferable. However, before committing oneself to a major effort in such a language a rapid prototype in GPSS can be a suitable starting point, as in cases like that of the Swedish ice-breaking operations (Jennergren et al. 1995).

8 Further Information On MICRO-GPSS

Micro-GPSS is available on the PC and Macintosh as well as on SUN and VAX workstations.

A demonstration diskette with scaled down versions of the micro-GPSS interpreter and GPSSDIA and with 50 program examples is available free of charge, together with a short tutorial, from the author. Professor R. Born (1995) has produced a very pedagogical computer slide show to accompany this tutorial.

More information on micro-GPSS and its use in the courses at the SSE is also available on the web starting at WWW.HHS.SE/secc/staff/#IIS.

References

[1] Banks, J. (1995) Semantics of Simulation Software. In OR/MS Today, December 1995, pp. 38 - 40.

[2] Born, R. (1995) *Computer Slide Presentation to Accompany Ingolf Ståhl's Simulation Made Simple with micro-GPSS*, Northern Illinois University. DeKalb.

[3] Claeys, F., Vangheluwe, H. and Vansteenkiste, G.C. (1995) HGPSS: A Hierarchical Extension to GPSS, in *Proceedings of the 1995 Simulation Multiconference*, Prague.

[4] Gordon, G. (1979) The Design of the GPSS Language, in (Eds) Adams, R.N. and Dagramici, A., *Current Issues in Simulation*, Wiley, New York.

[5] Henriksen, J.O. (1983) *State of the Art GPSS*, Paper presented at the 1983 Summer Computer Simulation Conference, Vancouver, Canada.

[6] Henriksen, J.O. (1985) *The Development of GPSS/85*, Paper presented at the 18th Annual Simulation Symposium, Tampa, Florida.

[7] Jennergran, L.P., Lundh, L., Tornqvist, U. and Wandel, S. (1995) Icebreaking Operations in the Northern Baltic, in (Ed) Miser, H.J., *Handbook of Systems Analysis: Cases*, Wiley, New York.

[8] Nywall, J. (1992) *GPSSEDIT Manual*, Mimeo, Karlstad College, Karlstad, Sweden.

[9] Schriber, T. J. (1974) Simulation Using GPSS, Wiley, N.Y.

[10] Schriber, T. J. and D. T. Brunner (1995) "Inside Simulation Software: How It works and Why It Matters." In C. Alexopoulos, K.Kang, W.Lilegdon and D. Goldsman (Eds.) Proceedings of the 1995 Winter Simulation Conference, SCS, La Jolla.

[11] Ståhl, I. (1990) Introduction to Simulation with GPSS: On the PC, Macintosh and VAX, Prentice Hall International, Hemel Hempstead, U.K., 1990.

[12] Ståhl, I. (1992a) Animation with micro-GPSS and Proof, in *Visualisierung und Prasentation von Modellen und Resultaten der Simulation*, ASIM, Heft, Nr.31.

[13] Ståhl, I. (1992b) *Principles behind the Design of an Easy-to-Learn Simulation Language: Implications for the Selection of Block Types*, EFI Research Paper 6485.

[14] Ståhl, I. (1993a) Principles behind the Design of an Easy-to-Learn Simulation Language, in (Eds) Robert, R.S. and Monroe, S., *Simulation Applications in Business Management and MIS*, SCS, San Diego.

[15] Ståhl, I. (1993b) GPSS Will Prevail - Some Reasons for the Resiliance of the GPSS Simulation Ideas, in *GPSS-Users' Group Europe*, ASIM Heft nr. 36, Magdeburg.

[16] Ståhl, I. (1993c) *Simulation Made Simple with micro-GPSS: A Short Tutorial with Seven Lessons*, Stockholm School of Economics, Stockholm.

[17] Ståhl, I. (1995) Simulation Made Simple with micro- GPSS: A Short Tutorial with Seven Lessons, Stockholm School of Economics, Stockholm.

[18] Ståhl, I. (1996) Steps Towards a Better Internal GPSS Mechanism, In J.M. Charnes, D.J. Morrice, D.T. Brunner and J.J. Swain,Proceedings of the 1996 Winter Simulation Conference, SCS, La Jolla.

Enhancing Simulation Education with Intelligent Tutoring Systems

Julika Siemer
Department of Information Systems,
London School of Economics and Political Science,
Houghton Street, London WC2A 2AE, England
Phone: +44-171-955-6029, Fax: +44-171-955-7385
j.siemer@ise.ac.uk

Keywords: simulation, education, artificial intelligence

Edited by: R.J. Paul

Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

The demand for education in the area of simulation is in the increase. This paper describes how education in the field of simulation can take advantage of the virtues of intelligent tutoring with respect to enhancing the educational process.
For this purpose, this paper gives an overview of what constitutes the objectives and the content of a comprehensive course in discrete event simulation. The architecture of an intelligent tutoring system is presented and it is discussed how these sophisticated learning aids offer individualised student guidance and support within a learning environment. The paper then introduces a prototype intelligent tutoring system, the simulation tutor, and suggests how the system might be developed to enhance education in simulation.

1 Introduction

Simulation provides the pedagogical challenge of educating students in a wide range of problem understand ing skills (Paul & Hlupic 1994). A recent approach to the use of computers as tutors is that of intelligent tutor ing systems (Kaplan & Rock 1995). Intelligent tutor ing systems provide helpful guidance and adaptation to the student by exploring and understanding the needs of the individual student. This paper proposes the use of in telligent tutoring to enhance education in simulation. For this purpose, we present an overview of what constitutes the learning objectives and content of a comprehensive course in simulation. The paper continues to describe the architecture of intelligent tutoring systems and how these sophisticated learning aids can provide individualised tutoring. The paper presents a prototype intelligent tutoring system, the simulation tutor, and suggests how the system might enhance the delivery of skills and knowledge in simulation.

2 Education in Simulation

An example of a typical course in discrete event simulation is taught at the Department of Computer Science and Information Systems, Brunel University, England. The course is dedicated to the objective of educating the student in problem understanding, and is a final year option open to B.Sc. Computer Science and B.Sc. Computing in Business students. This is achieved through an overview study of simulation approaches, and aims to provide the student with practical experience in different simulation languages and environments used in industry. The main modes of delivery of the course are lectures and practical case studies. The underlying themes ask questions such as, "How may a simulation model be built, using an appropriate simulation tool, which represents the best understanding of the system under study?" and "Using such a model, how may we experiment with it to help us understand more about that system?" Accordingly, the course objectives include the management of the simulation process, the development of simulation skills and promotion of awareness of the place of simulation relative to other disciplines (orientation).

2.1 Management of the Simulation Process

This concerns the key educational themes of model development and experimentation. Model development addresses the means by which a simulationist can confidently develop and subsequently demonstrate to the owner of the system the validity of a model developed for that system. Experimentation concerns the formulation of the simulation experiments to be performed on the simulation model.

2.2 Development of Simulation Skills

The student is taught a variety of techniques for this purpose. For example, during the course, the student is in troduced to activity cycle diagrams, automatic program generators, visual interactive simulation modelling, simulation program structures, simulation software tools, the handling of stochastic input and output of the model, and issues relating to model confidence. This course of study is backed up with a workshop program to reinforce these practical skills, so giving the opportunity for the student to apply what he or she has learnt in order to concretise the knowledge delivered via the lecturing programme. For example, the student discovers activity cycle diagrams and then is given the opportunity to pursue a simple case study to model a system in terms of this technique. An automatic program generator is then introduced to give the student some experience in transferring the model into a simulator. Case studies of increasing complexity are introduced to address other aspects.

3 Simulation Orientation

Simulation Orientation is intended to present simulation as being part of a larger field concerning techniques relevant to decision making. The reason for this issue being addressed is exactly why simulation has found itself as part of other courses ranging from Operational Research to Management Science. It is important that the student is aware that simulation is not an isolated discipline and has strong links to other fields.

This course has proven to be extremely popular amongst undergraduates. While gratifying that this is the case, the size of the student cohort undertaking this scheme of study has increased the pressure placed on the educators responsible for course delivery. This is especially true for the practical case study elements of the course which require significant tutorstudent interaction. These were some of the factors motivating a study launched to examine methods to alleviate this pressure. The most interesting possibility which arose from this study was that of a recent innovation in computer-based training which takes advantage of Artificial Intelligence. This is termed *intelligent tutoring systems.*

4 Intelligent Tutoring Systems

The objective of intelligent tutoring systems (ITSs) research is to provide a new plateau of instructional capability by integrating artificial intelligence into computerised learning systems. Intelligent tutoring systems provide helpful guidance to the student and are intended to complement socalled conventional computer-based training systems with the features of an equivalent "on- line" human tutor. It is therefore the goal of an ITS to make the computerbased teaching process more adapt able to the student by exploring and understanding the student's special needs and interests, and by responding to these as a human tutor does. In order to provide this adaptability to the student an ITS makes use of three knowledge models. These are the *tutoring model*, which regulates the instructional interactions with the student, the *domain model*, which contains the knowledge about the domain to be taught, and the *student model*, which represents any information about the student. Adding the fourth component, the *user interface*, completes an ITS (Kaplan and Rock, 1995). To better understand the means by which ITS can aid education in simulation, we now review the four components.

4.1 The Domain Model

The domain model of an ITS contains the knowledge about the subject area to be taught. The ITS uses its domain knowledge to reason about and solve a problem which has been set for, or by, a student. This knowledge has to be represented in such a way that it supports reasoning that resembles the human problem-solving process within the teaching domain. Furthermore, different knowledge representations may be required to support the application of different teaching strategies. One rather pragmatic way of looking at what the domain model represents is to consider it as being a computer based training tool and postponing decisions which have to be made concerning knowledge representation, etc. As will be seen, it is this approach which has given rise to our current implementation cycle.

4.2 The Tutoring Model

An ITS should exhibit the various tutoring characteristics encountered in the classroom. These are encapsulated in the tutoring model. It must therefore have control over the selection and sequencing of material to be presented to the student. The system must also be able to respond to questions concerning the subject material and must be able to apply learning strategies to determine when students need help and what kind of help is required. For this purpose the tutoring model incorporates different teaching strategies (O'Neil, Slawson and Baker 1991).

Teaching strategies are used to present material and depend on the subject matter and the instructional objectives of the ITS. A teaching strategy determines the style of material delivery that is employed in order to lead the student and to indicate the times at which intervention is required. Many ITSs apply different teaching strategies in different teaching situations. Reviews have been made by Siemer, Taylor, and Elliman (1995) and ElsomCook (1991).

An ITS should ideally include a selection of different teaching strategies for intervention. The strategy may be chosen according to the peculiarities of a tutorial situation, such as the student"s needs and preferences, experience and the domain of discourse (Angelides and Tong 1995).

4.3 The Student Model

To carry out intelligent tutoring a tutor has to have a good understanding of the student being taught. For this reason an ITS uses a student model to represent the student"s emerging knowledge and skills of the subject matter. An ITS may use its student model to analyse the input of the student during tutoring interaction. The student"s input may come in a variety of forms ranging from answers to questions posed by the ITS or moves taken in a game, to commands delivered within an editor. This information can sometimes be complemented by the student"s educational history.

The most common approach is the representation of the student"s knowledge in the form of an overlay model. Here the student"s knowledge is represented as a subset of an expert"s knowledge. As the student moves from his initial state of knowledge towards mastery, parts of knowledge are added to this expert knowledge subset. To determine the student"s knowledge state the system generally refers to the same knowledge base using different interpreters. However, the overlay model allows only missing parts of knowledge to be represented. To represent misconceptions, i.e. a different conception of some part of knowledge, the overlay model is usually complemented by a educational bugs catalogue. The bugs catalogue is a library of common errors made by students. It represents typical deviations a student can make from the ideal expert behaviour. Student diagnosis is carried out by matching the student"s faulty behaviour against the bugs in the bugs catalogue. From this remedial action can be taken by the system.

4.4 The User Interface

The user interface is generally recognised as a fourth component of an ITS architecture (Woolf and Hall 1995). As the student interacting with the ITS is working in an area he or she does not understand well the system has to ensure that any complications the student is exposed to when operating the system are kept to a minimum. A well designed interface can add considerably to the way in which the student will conceptualise the domain being taught (O"Malley 1990).

The user interface determines how students interact with the ITS (Bonar 1991). The teaching material may have to be presented in different formats, such as text, graphics, animation or video (Alpert, Singly and Carroll 1995) depending on the needs and requirements of the current situation, such as the teaching strategy employed or the nature of the material to be presented. A well designed human interface allows the ITS to present instruction and feedback to students in a clear way. At the same time it can provide students with tools and techniques to state problems and hypotheses to the ITS.

5 StudentCentred Tutoring

An ITS employs the knowledge of its three models to provide studentcentred tutoring, i.e. the adaptation of the tutoring process to the student"s needs and preferences. In order to provide this adaptability for a student, an ITS should apply suitable teaching strategies and presentations for each subject matter unit as needed, choosing the form that is most beneficial to the student for a particular instructional situation (Miller and Lucado 1992). The selection of an appropriate teaching strategy and its presentation requires knowledge about the needs of a student. The student model may provide information about the student's experience with particular remedial strategies and different ways of presentation. Accordingly, it may provide information on approaches that have proven to be successful or unsuccessful within earlier interventions, and the tutoring process may adapt accordingly.

A further common discriminating factor to consider in order to distinguish between students with different needs, at a stage where the creation of a full cognitive model has not yet been achieved, is the differentiation between student advancement stages. A system may, therefore, adjust its tutoring to the advancement stage of a student. A fundamental idea in education is that as students learn more about a subject, one can teach them progressively more advanced ideas. Accordingly, a novice may require different tutoring from an advanced student. Whilst the competent student may be able to appreciate and integrate shallow and subtle tuition, the novice student might require a detailed explanation of intermediate ideas (Silverman 1992).

A further issue of studentcentred tutoring is the issue of proactive instruction and reactive instruction (Alpert, Singly and Carroll 1995, Silverman, 1992). An ITS may provide adaptability to the student by providing both help that may be invoked by the student or by the system. An advanced student, for example, may recognise his need for help and may decide to activate system help. However, when the student is a novice, or when the teaching domain is broader, systeminvoked help seems more appropriate. The student may require intervention when the student makes a mistake without realising it, or when the student does not know what to do next. A student who has problems with the English language, for example, may appreciate a systemactivated spellchecker or grammarchecker that assists him incrementally during the task as each difficulty occurs. Similarly, the PAT (Pump Algebra Tutor) system (Barker 1995) offers studentinvoked help. The student has to ask for intervention by the system when he feels that help is required for the mathematical problem he has to solve. Attempts have been made to bring the two modes to gether by establishing a balance according to the student's individual preference (Moyse and ElsomCook 1992, Winkels 1992, Milheim and Martin 1991). The MoleHill system which teaches Smalltalk programming (Singley, Carroll and Alpert 1993), for example, offers both active and reactive instruction to its users.

ITSs have been introduced thus far as a novel development in computerbased training. These offer individ ualised student guidance and support within the learning environment. In order to demonstrate how these educational benefits can be used to enhance simulation education the next section introduces a prototype ITS for simulation which has been developed as part of this research.

6 The Simulation Tutor: An Intelligent Tutoring System for Simulation

The simulation tutor is an ITS currently under develop ment at Brunel. Given the quite heavy investment in development that an ITS requires, a prototyping approach has been taken. This has used a paperITS to form the basis for evaluation and definition of ITS interaction and the simulation tutor. A paperITS is essentially the domain model and user interface implemented using conventional computerbased training technology (in this case Asymmetrix Toolbook (Asymmetrix Corporation, 1994)). The tutor and student models are emulated by using a research assistant. To provide a forum for discussion and ongoing development, the domain model has been decomposed into subcomponents representing the major elements of the simulation course (Paul and Balmer 1993). These are:

- 1 **The need for simulation**. This is first considered to make it clear to the student as to the reasons for when simulation should and should not be used. This follows the course text and uses examples generated by the educator and by the class in open discussion.

- 2 **Discrete event simulation**. The goal of this lecture is to introduce the students to the key terminology of discrete event simulation and its constituents. Again the course text is used and is backup by worked examples indicating how each elements works and its consequences.

- 3 **Different modelling approaches**. Following on from 2., different modelling approaches are considered. Worked examples are given.

- 4 **Activity cycle diagrams and the three phase approach**. This modelling technique is selected as one technique to study in depth. Further terminology and examples are introduced.

- 5 **Case study I**. The first of two case studies is intro duced. Students are taken through different stages of model development via the first simple case study.

- 6 **Visual interactive simulation**. A simple simulation environment and features are introduced with respect to the earlier elements of the course. The importance of animation in simulation is demonstrated.

- 7 **Model testing and validation**. Following on from 6., issues of model testing and validation are covered.

- 8 **Case study II**. A more complex case study is at tempted. This is used to summarise the previous elements of the course.

- 9 **Discrete event simulation software and languages**. This reviews the current state of the art.

- 10 **Sampling methods**. Sampling methods are reviewed.

- 11 **Planning and interpreting discrete event simulation experiments**. Again a review of the subject area.

- 12 **Summary**. The course is pulled together with respect to the learning objectives.

Initial work performed on the simulation tutor concerned the first four elements. Essentially, the domain model was an animated version of the course text. This added an extra dimension to the usual forms of course support and focused on relating the components of the simulation program to the flow of entities through a system. Interaction with the tutor and student models was performed via a research assistant who discussed the learning needs with the representative student groups.

7 Evaluation of the Prototype

The prototype was evaluated with a group of representa tive students who used the system. The students had recently completed part of the simulation course at Brunel University. They were asked to interact with the system under supervision. The experiment was supported by a questionnaire which was developed based on suggestions made by Schneiderman (1987) to gain feedback from the students about the way they perceived the needs of tutoring and learning.

The results of the evaluation show that the simulation tutor has the potential of offering adaptive tutoring to the student. The tutor coordinates the information from its knowledge models to provide adaptability to the needs and preferences of the players. In general comments alluded to good structuring and presentation of the teach ing material. Informational content was adequate, given that it was supported by course texts and lecture notes. However, more effort should have been made in the presentation of the graphics. Video should be included in the next release. The use of animation to demonstrate the working of the simulation model and the simulation itself was good. Students were keen to have model elements linked to the physical system via video sequences.

In regard to the user interface users expressed a general liking of the use of a windows environment with easy to use, selfexplanatory buttons and features. The contextsensitive help facility providing hints and analogues was useful as was the use of a mapping system; the students responded well to being able to see where they were in relation to the other parts of the course. There was, however, general dissatisfaction with the lack of response that the prototype provided. This was to be expected as the tutoring parts of the system had not been fully implemented. Once this was explained to the students, the response was that if the role of the tutor was documented correctly, then the intelligent tutor would be more than satisfactory.

The overall response to the system was encouraging, and the feedback provided by the students suggests ideas to be incorporated into the next version of the prototype.

8 Conclusions

This paper has reported on an attempt to introduce ITS techniques into simulation education. An initial investigation has shown that ITSs could improve the educational process by providing individual guidence and student centred learning. Further work is currently address ing other elements of the simulation course and the implementation aspects of the simulation tutor.

References

[1] Alpert, S.R., M.K. Singley, and J.M. Carroll. (1995) Multiple multimodal mentors: delivering computer based instruction via specialized anthropomorphic ad visors. *Behaviour and Information Technology*, 14, 2, p.69-79.

[2] Angelides, M.C. and A.K.Y. Tong. (1995) Implementing multiple tutoring strategies in an ITS for music learn ing. *Journal of Information Technology*, 10,1, p. 52-62.

[3] Asymmetrix Corporation. (1994) Asymmetrix Multimedia Toolbook Kit. Asymetrix Corporation.

[4] Barker, D. (1995) Seven New Ways to Learn: Carnegie Mellon University is changing the way in which teachers teach and students learn. *Byte*, 20, 3, p. 54-55.

[5] Bonar, J.G. (1991) Interface Architectures for Intelligent Tutoring Systems, In *Intelligent Tutoring Systems: Evolutions in Design*. ed. H.L. Burns , J.W. Parlett and C.L. Redfield. p. 3568. Lawrence Erlbaum Associates, Hillsdale, New Jersey,

[6] ElsomCook, M.T. (1991) Dialogue and Teaching Styles. In *Teaching Knowledge and Intelligent Tutoring*, ed. P. Goodyear. p. 61-84. Ablex Publishing Corporation, New Jersey.

[7] Kaplan, R. and D. Rock. (1995) New Directions for Intelligent Tutoring. *AI Expert*, 10, 2, p. 30-40.

[8] Milheim, W. D. and B. L. Martin. (1991) Theoretical Bases for the Use of Learner Control: Three Different Perspectives, *Journal of Computer-Based Instruction*,18, 3, p. 99-105.

[9] Miller, M.L. and Lucado, S.R. (1992) Integrating Intelligent Tutoring, Computer Based Training, and Interactive Video in a Prototype Maintenance Trainer, In *Intelligent Instruction by Computer*. ed. M.J. Farr and J. Psotka. p. 127-150. Taylor & Francis, New York.

[10] Moyse, R. and M. ElsomCook. (1992) Knowledge Negotiation: An Introduction. In *Knowledge Negotiation* ed. R. Moyse and M.T. ElsomCook. p. 1- 20. Academic Press, London.

[11] O'Malley C. (1990) Interface Issues for Guided Discovery Learning Environments. In *Guided Discovery Tutoring: A Framework for ICAI Research*. ed. M. ElsomCook. p. 24-41. Paul Chapman Publishing, UK.

[12] O'Neil, H.F., D.A. Slawson, and E.L. Baker. (1991) Design of a Domain Independent Problem Solving Strategy for Intelligent Computer Assisted Instruction. In *Intelligent Tutoring Systems: Evolutions in Design*. ed. H.L. Burns, J.W. Parlett, and C.L. Redfield. p. 69- 104. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[13] Paul, R.J. and D. Balmer. (1993) Simulation Modelling. Chartwell Bratt.

[14] Paul, R.J. and V. Hlupic. (1994) Designing and Managing a Masters Degree Course in Simulation Modelling. In *Proceedings of the 1994 Winter Simulation Conference.* ed. S. Manivannan J.D. Tew. D.A. Sadowski and A.F. Seila. p. 1394-1398. ACM Press.

[15] Schneiderman, B. (1987) Designing the User Interface Strategies for Effective Human Computer Interation. Addison Wesley, California.

[16] Silverman, B.G. (1992) Critiquing Human Error, A Knowledge Based Human Computer Collaboration Approach. Academic Press, London.

[17] Siemer, J., S. J. E. Taylor and A. D. Elliman. (1995) Intelligent Tutoring Systems for Simulation Modelling in Manufacturing Industry. *International Journal of Manufacturing System Design.* 2, 3, p. 165-175.

[18] Singley, M.K., J.M. Carroll, and S.R. Alpert. (1993) Incidental Reification of goals in an intelligent tutor for Smalltalk. In *Cognitive Models and Intelligent Environment for Learning Programming.* ed. E. Lemut, B. du Boulay, and G. Dettori. p. 145-155. SpringerVerlag, Berlin.

[19] Winkels R. 1992. Explorations in Intelligent Tutoring and Help, Frontiers in *Artificial Intelligence and Applications Series*, IOS Press, Amsterdam.

[20] Woolf B.P. and Hall W. 1995. Multimedia Pedagogues Interactive Systems for Teaching and Learning. *IEEE Computer.* May, p. 74-80.

A Generic Architecture for Intelligent Instruction for Simulation Modelling Software Packages

Tajudeen A. Atolagbe
Department of Information Systems and Computing,
Brunel University,
Uxbridge, UB8 3PH, England
Phone: +44-1895-274000, Fax: +44-1895-251686
Tajudeen.Atolagbe@brunel.ac.uk

Keywords: simulation, education, artificial intelligence

Edited by: S.J.E. Taylor
Received: April 14, 1996 **Revised:** July 14, 1996 **Accepted:** March 26, 1997

This paper describes architecture for an intelligent interactive instructional simulation modelling environment. It revolves around the production of tutorial and courseware authoring for different simulation software packages with a generic user interface shell. The generic shell is a "front end" which provides a uniform graphical user interface to diverse simulation modelling software tutorials. An object- oriented perspective is combined with tutorial activities based on the task classification structure to form the interactions between objects. The development environment brings together objects that are functionally coherent and allows them to share common resources within the shell.

1 Introduction

Organisations adopt different strategies for assessing training needs and ways of implementing them. Because of the increasing use of simulation modelling as an analysis tool and the complexity of developing simulation models, computer based training is a viable option for implementing simulation software training. The provision, maintenance, and continuous updating of courseware, both in academic and industrial establishments, will continue to require huge investments. If an organisation is to sustain its competitive position, full integration of training with business activities, and a framework within which the tutorials needs and strategies for their development are vital.

The Intelligent Simulation Training System (Draman 1991) provides a generic architecture for building an intelligent simulation software training system. It uses a generic modelling of simulation scenarios for drilling the learner. The system is domain dependent, and its portability is limited. Its domain knowledge is specifically for simulation modelling. The TOTS system (Richard and Webb 1987) provides a domain-independent intelligent tutoring shell for task oriented domains and is based on *"procedural network"*. The shell is domain specific and does not allow the user to interact directly with the target software. The EXCALIBUR (Richard, Webb and Craske 1987) produced a generic knowledge base system for learning. The system allows the learner to interact with the tutorial program(s).

Using simulation modelling software involves learning the model development skills, knowing what procedures exist for modelling exercises, and how to accomplish various conceptual model formation. It also involves recalling the procedures whenever a similar situation arises. It can be inferred that using a simulation software package requires analytical and cognitive skills.

The aim of this research is to develop an instructional simulation software environment and provide for the learner to interact directly with various simulation software packages. This method permits the learner to acquire procedural knowledge, which forms the foundation for cognitive skill acquisition (Patrick 1992). The curricular consists of practical activities and illustrations, which help the learner to develop basic skills and knowledge for using the package.

An object oriented approach that provides a generic, direct manipulation graphical user interface for an interactive instructional simulation modelling development environment is presented. The software provides for cross platform integration and development of interactive tutorials for simulation software packages. The object-oriented approach allows functionality of the simulation software package to be mapped directly into a tutorial structure.

The tutorial operations and learners mental model can be used to obtain the systems requirements, which forms the basis for initial interface design. The architecture embraces systems for producing, authoring and delivery of tutorials for different simulation software packages with a shared graphical user interface. The shell is capable of seamless integration of audio, text and graphical images, which are used at various levels during instruction. The shell is a graphical user interface suitable for accessing heterogeneous instructional software and it can be utilised for verse delivery of tutorials. It embraces systems for the production and authoring for different simulation software packages with common front end across the packages. The "shell" also incorporates different learning aids, and learner interaction is based on direct manipulation dialogue.

Learning hierarchies proposed by Gagne (1965) can be employed to yield task classification structure. This method allows tutorial tasks to be represented in small modules thereby allowing portability and ease of maintenance. Alderman (1978) suggested that hierarchical structure of subject materials is suitable for a *"rule-by-practice"* learning method. The tutorial curriculum can be classified into hierarchical structure based on its functionality of the simulation software and its operational techniques.

The tool possesses three unique features: unique front end, text inputs which provides for a cross-platform courseware development, and an application software environment. These features provide facilities for courseware authoring and maintenance, and for application software to run independent of the tutorial. It can provide the learner with a content rich tutorial environment with an interactive guide.

2 Teaching Simulation Modelling Software

There is a large gap between the functional behaviour of a simulation software package and the learner's perception of software operations. This misapprehension can be attributed to *"cognitive strain"* which users adopt as strategies designed to minimise cognitive load during problem solving and concept- attainment tasks (Bruner et al., 1956). The difficulty of usage can be exacerbated by the process of models representation and analysis. The functionality of the simulation software can be classified into a hierarchical task structure. Constructing a conceptual model can depict the behaviour of simulation software. Teaching operations of the software requires cognitive association of all instructional activities, structured into hierarchical array of sub-activities. We identified the following strategies for effective teaching of simulation modelling software:

- tutorial tasks organised in a pedagogy structure,
- minimising the cognitive load on the learner,
- providing meaningful error messages during development,
- provision of interactive tutorials,
- use of animation and gaming- simulation as an alternative teaching strategy and
- providing context sensitive help.

The architecture provides for the development of procedural knowledge, direct mapping of structured curriculum tasks to an application model based on appropriate pedagogical structure.

The functionality of a simulation package and the tutorial operations can be classified into a pedagogy structure. The shell allows for direct mapping of simulation problem into predefined structure of the tutorial. The tutorial operations are represented by a root object, on which a hierarchical decomposed task object depends. The tutorial tasks of the software are arranged in pedagogy order, reflecting stages of building a simulation model and post simulation analysis. The sub-task level consists of diverse models of the software behaviour represented in varying levels of description.

An interactive tutorial that permits the learner to interact with simulated scenarios at certain levels during instruction can be provided within the shell. The architecture also uses animation to display the conceptual model of a scenario, with icons representing different activities. This method allows long-term transfer of skills and reinforces the proficiencies of using the software.

The user interface for simulation software consists of data and functions objects, and their attributes. The learner activities involve manipulation of these objects. The object-oriented model consists of the following semantic elements: all objects are classified into types defined by their behaviour and inter-relation with other objects and their attributes. Their operation and their inherent characteristics can define the behaviour of the objects.

The system provides the learner with an opportunity to exploit most features the simulation package such as ProModel PC, SIMFACTORY 11.5, WITNESS etc. A structured decomposition of the domain knowledge elements and functions are progressively presented as the learner advances through the curriculum. This technique enhances learner participation in building a simulation model of the problem. The shell also provides for unobtrusive "walk-through" stages of model development. Simulation model of discrete event systems, passes through different activities and the system clock advances asynchronously. Constructing a simulation model involves depicting the activities and entities of the problems through time. This can be represented and presented during instruction by using a structured decomposition of the operations and taking the learner through stages of developing a simulation model.

2.1 System Architecture

An *"object oriented"* technique effectively captures and displays systems requirements during the development phase. Objects depicting various operations represent the generic "shell". The shell is based on the four-layer architecture. These consist of the following: the presentation system, the application and tutorial module layer, knowledge representation and domain expert layer, and the main kernel.

- The presentation layer consists of the graphical user interface (GUI), which consists of a dialogue system that communicates with all objects in the system. It also provides for direct manipulation of all objects on the GUI and provides feedback to the learner. It provides an effective and transparent interface across the shell.

- The application module consists of a simulation modelling software package and online documentation provided by the vendor. It can be linked directly to the tutorial system and the courseware authoring environment, as shown in figure 1. The courseware authoring environment provides a platform for tutorial development and for courseware maintenance. It provides for components that can be used for integrating text, graphics, animation and for creating or importing dynamic multimedia objects. This is a hypermedia representation and other media objects can be integrated. A tutorial unit may consist of a combination of graphics, digital video images and animation.

- The knowledge base contains structured domain knowledge, and simulation modelling problem, plus simulation modelling examples. The structured knowledge is organised into hierarchical task units. The control system is the domain dependent problem solving strategy that determines how the tutorial can be manipulated into the learner's requirements. The knowledge base is linked to a domain *"blackboard"* which contains all the instructional strategy and control of the tutorial. It executes rules, which depend on the learners' behaviour during instruction.

- The kernel consists of inheritance characteristics of all objects and their classes. It executes the codes and provides means for sharing objects and manages the memory. The kernel provides preemptive threads and multiple threads operations. It allows for each tutorial unit to provides a preemptive *"threads"* to execute other units.

Each object has defined characteristics, which can be embedded into the software. The presentation system provides a uniform representation of the objects. It also provides a mechanism for assessing both the applications and the tutorials modules. The GUI allows the user to communicate directly with the system, (Burton & Brown 1982) whilst the shell consists of a set of routines, which allows the learner to control the instruction. It reflects the general tasks classification structure and semantics for both the tutorials and the application software. The GUI consists of icons that represent objects, which the user can manipulate while under instruction. By providing a unique front end, courseware designers can write diverse tutorials for different simulation software packages.

Research findings suggest that acquisition of procedural knowledge can be improved if learner can observe and interact with visual models of the abstract concepts, (Burton and Brown 1982). Shneiderman, (1987) suggested that learners can learn 'quickly' by manipulating the interface object. Tutorial activities depend on the hierarchical structured tasks represented as objects.

The architecture exploits both hypermedia and artificial intelligent methods for courseware authoring. Angelides and Paul (1993) suggested that for an instructional tool to be effective, it should incorporate *"intelligent tutoring"* components. The artificial intelligence components can be embedded within the tutorial environment. The generic tool allows the user to suspend a tutorial unit and interact directly with the simulation software or to seek other method of learning i.e. to choose a demonstration option or observe the animation of the scenario. Some simulation software vendor provided tutorials, for example ProModel PC, permits the learner to interact with a subset of the real application software. These tutorials do not teach the compete functionality of the software. This method may result in a reduction of the learners' skills level development.

The instructional paradigm is based on different techniques of teaching a simulation software package. The tutorial unit provides a *"step-by-step"* guide to constructing a simulation model and illustrating the use of all the components of the software. The tutorial objective is for the learner to develop a procedural knowledge for a specific simulation software package. This approach is similar to SOPHIE, I, II, III, (Burton & Brown 1982) which is based on developing the users skills in troubleshooting electronic circuit. The systems are based on expert systems tutoring environment, but offer limited control to the user.

The control system provides for dynamic process communication between the "blackboard" and the event manager, as shown in figure 2. It allows the learner to use their own model and generates simulation models based on their tutorial activities. The

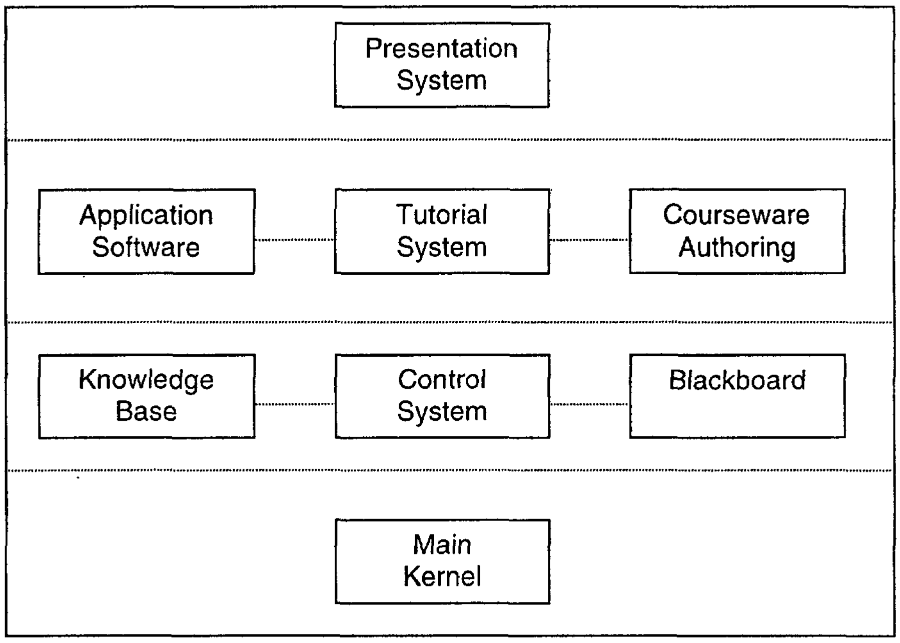


Figure 1: Schematic Representation of the Shell

event manager controls the operations of the software and the tutorial activities. The control system consists of inference mechanism, which controls the structure of the program, and interpreter rules that are embedded in the system. It compares the learner input with the information in its knowledge base and decides which information satisfies the rules. This technique is used for accessing the tutorial network and it limits the search and retrieval time.

The *"kernel"* can execute only one tutorial at a time, but allows user interruption at any time. During interruption it "spawns" supplementary threads to perform other tasks. By separating the graphical user interface from the underlying instructional applications and application software, the shell can be easily transfer to other environments. Other advantages of this approach are versatility, code reusability and ease of updating, (Booch 1990).

All the components of the shell send and receive event messages via a protocol. The controller performs this function and co-ordinates the scheduling of processing time between application system and the tutorial system. The protocol determines the order of the events based on priority values set in the program. User interruption triggers events, which provides a multitasking environment and allocates the resources.

The blackboard comprises of the student model and a model generator. It can provide for communicating between the two models. The student model provides the learner with different problems based on the user *"overlay"* model. The "overlay" model describes the learner in relation to its interactions with the system and competence level. It maps problem domain into tasks classification structure of the tutorial. It also presents increasingly complex exercises based on the current tutorial unit level.

The model generator allows for a hierarchical structured tutorial domain knowledge representation. It provides a structured approach to building a simulation model and consists of combination of the following related phases:

- 1 identification of simulation concepts and software components,
- 2 formation of the relationship between concepts and software,
- 3 hierarchical approach to model representation and mapping problem domain into predefined structure of the tutorial,
- 4 the output analysis.

It first identifies the tutorial needs of the learner and presents the tutorial as represented by rules. It employs the "overlay utilisation rule" to generate simulation model exercises and tutorials.

2.2 Simulation Model

The second layer of the architecture allows for interactive development of simulation models and the representation of the systems components. For a discrete-event simulation model, the states of the *"entities"* change at discrete times. The entity can be represented as objects, characterised by their attributes.

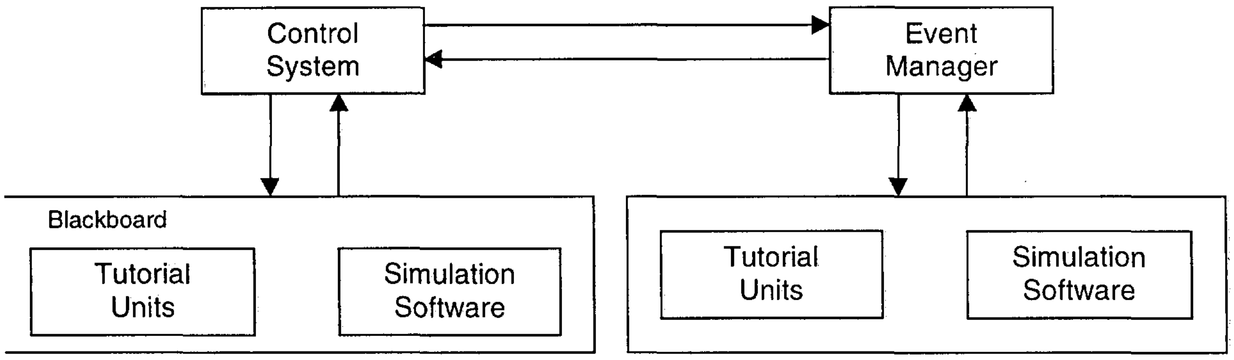


Figure 2: Inter-Process Dialogue

An object transforms into different states of activities within an interval. All the objects relates to events, which have dynamic characteristics determined by the user. The *"entities"* advance through different states and behaviour asynchronously and depends on dynamic message routing. The learner can interrupt an example of a simulation model of a scenario at any time. Learner interruption allows for manipulation of the attributes and allows the learner to suspend the current task or perform another operation. The scope and characteristic of the simulation model must be appropriate to the learner's tutorial level and should permit instructional control. Constructing a simulation model of a scenario involves the follows phases: hierarchical representation of the scenario, relating scenario to a classification structure depicting the functionality of the software, and components of the package. Post simulation analysis illustrates output collection, and statistical analysis.

3 Curriculum Development

All simulation software packages possess different characteristics and functionality, they also differ in how entities are represented. Developing curriculum for a simulation-modelling package should be a dynamic process set within a context, which unifies the simulation modelling theory and practices. The representation of the domain should consists of the tutorial units with specified internal structures that relate to the theory. It should allow the software to run independently of the tutorial. This representation is similar to that of a Lisp program, which captures the learners' behaviour during instructions. It is essential to identify which paradigm will guide the way simulation theory is applied in the tutorial. The difficulties of writing tutorials for simulation software package without relating to simulation theory is complicated. To minimise these obstructions, the following procedures were formulated:

– The curriculum is decomposed so that all the contents and instructional strategy reflects the functionality of the software.

– The curriculum relates to information on simulation modelling theory at appropriate level in the tutorial.

– Developing models that can be supported by the simulation software

– A hierarchical representation of building the model, demarcating conceptual representation and statistical analysis.

– The functionality of the simulation software should be demonstrated in the tutorial.

This representation allows modularity and each composite part is independent of the other. The advantage of this architecture is its simplicity and object oriented nature, plus its usefulness as an integrated courseware authoring tool.

3.1 Tutorial Units

Discrete event simulation software consist of three major components - library, control shell and report generator, (Pidd 1992), and the tutorial units can structured to depict the functionality of the software and these components. The tutorial units can be represented as structured tasks and subtasks, elicited from the tutorial as a tree. The subtask unit portrays the characteristics of the software package and its functionality. Hierarchical task analysis can be applied to organise the operations involved in using the software into a task classification structure, (Gagne 1965). The tutorial consists of problem solving operations, with various degrees of complexity within domain. Simulation modelling operations can be decomposed into a coherent sequence. The instructional paradigm is based on the combination of hierarchical task analysis, learner control technique and learner cognitive preference.

Decomposition of the tutorial tasks into structured units can minimise the cognitive complexity of the operations. A simulation modelling exercise can be decomposed into smaller subsystems and the curriculum task units can be represented as "tree" in an hierarchical structured network. This consists of several units, arranged in pedagogy array of tasks operations. A unit is represented as a node and it consists of several subtasks, which can be depicted as leaves. The tree node can be linked tutorial units, depicting a component of the software and the leaf nodes correspond to a command in the software. The contents of a node can easily be linked to other child nodes, as shown in figure 3, and addition instructions can be attached as leave at different levels.

The tasks are depicted as graphs with individual units, as nodes and prerequisite relations as links. Instructions are encapsulated on each node. The units are represented by a semantic network and each node represents a tutorial unit, which can be linked to other similar operation. The arcs represent the *"segment"* of tutorial unit.

Each node in the structured task tree can represent a production rule, which controls the presentation frame. All the nodes possess attributes, which indicate the names of up to two or more child nodes that are dependently linked. Navigation through the tutorial depends on the learner's path, which can be directly mapped into the task structure.

Each unit is represented as object and every object has one or more activities. All the nodes consist of classes and subclasses. The tutorial can be represented into subtask levels. These consist of a hierarchical model building process and other operations, which are represented as classes and subclasses in the tasks structure.

The sequence of instruction from a basic model representation to a complex simulation model and analysis employs the *"overlay utilisation rule"*. This rule considers the learners' *"current knowledge"* before presenting the tutorial and can provide control. The learner control provides the choice of content, direct manipulation of instruction, priority of actions and tutorial scope. It also provides control for sequence selection, tutorial option and review. It incorporates commands, which can allow the learner to navigate through the tutorial freely.

Tutorial knowledge for a simulation software package can be generated from training and user manuals, task classification structure and structured interviews with experienced users, plus case studies of diverse simulation projects. Knowledge is elicited from the tutorials after careful analysis of the task classification structure which is mapped

4 User Interface Objects

The user interface objects provides an interactive, *"direct manipulation"* of the curriculum objects and all other functional components of the system. The user interface can support unlimited set of semantic messages and provides direct mapping of tutorial knowledge to the functionality of the software, to a representation on the GUI. All the object have a semantic constituent represented as sets of rules that can manipulate the objects on the interface. Some of the rules describe units of lessons referenced by unique name, each unit consists of sets of classes and objects stored as sets. The tutorial task decomposition is utilised as object hierarchies, which depicts the tutorial functions and dialogue specification. This can be used to form a specification of the graphical user interface and depict the object oriented hierarchical structure. The GUI consists of data and object functions which can be inherited. Each application comprises of framework objects. The encapsulated object is specified by its behaviour and state. It permits control of data between objects. Some classes were incorporated for the specific needs of the tutorial. They include the following:

- Action-Class is a lower level class, which maintains a list of actions that are displayed to the learner when appropriate. Each window has an action associated with it. This class is responsible for freeing the memory associated with each action.

- The feedback-Class is a component of the intelligent tutoring system. It provides a content sensitive feedback to the learner. This depends on the type of error and the level of the learner in the task classification structure.

- The Navigation-Class helps the learner to keep up with the current position in the presentation program. It also monitors all the windows titles in the presentation. It allows the learner to navigate throughout the program. The navigation system address context, location, and progress indicator.

- The Windows-Class controls the graphical user interface portion of the sub-frame windows in the presentation. It refreshes the user interface when required, and options chosen by the user. The option can also be used to display textual multiple-choice questions. Texts that cannot be displayed are scrolled.

- Model-Class controls all simulation model examples presented to the learner. The learner can use this option to interact with the simulation model to change any of the variables or use the instructional gaming system

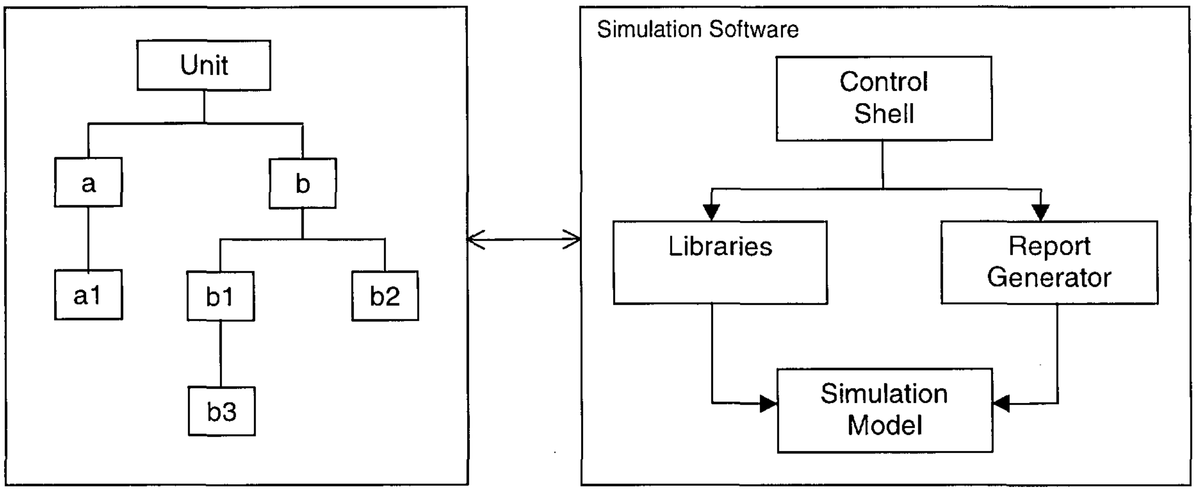


Figure 3: Structured Task Unit and Simulation Software Components

These classes present a highly interactive choice of principles and methodologies, which can be used by the courseware designer.

Polymorphism of a single task to refer to different sub-task and inter-related tasks that can be satisfied by rules represented by the knowledge elicitation method. The form of object orientation is as proposed by Booch (1990). All tasks and operations are modelled as objects, closely linked to the structures and behaviour of the package, incorporating single and multiple inheritance. All objects that have common characteristics and behaviours can by grouped together.

4.1 Portability to other platforms

The major factors for portability are modularization of instruction, with code reusability, file transfer, adaptation to different environment, and cross platform integration of simulation software packages. A "transfer" command from the courseware authoring option allows the text file and the interface to be transfer to diverse environment. Courseware authors can also import data from text files into the tutorials. Choice of development software is crucial for the development of the tool, as the primary objective is to produce a tool that will enable the transfer of both the interface and the files to a target environment. The only feasible programming languages are Visual C++ or Visual Smalltalk. Visual C++ allows partial transfer of its code to other platform, but it is complicated and it has a long development cycles. Visual Smalltalk allows easier transfer of codes to major wider area and is easier to learn. Although both languages support object orientation, Visual Smalltalk has a better memory management and supports multiplatform development. Object Orientation methodology is being used in the design of the software but the development environment will be Visual Smalltalk.

5 Conclusions

The paper describes architecture for an intelligent instructional system for simulation modelling software packages. It describes an integrated environment for teaching simulation modelling, courseware production and delivery. The curriculum development environment provides a dynamic process set within a context which unites the simulation modelling theory and the functionality of the software. The tutorial development framework allows for flexible courseware authoring and maintenance. The development environment allows the transfer of both the front end and the text file to diverse environments.

The direct manipulation of user interface object allows the tool to be adaptable and can simulate learning by allowing the user to manipulate objects of the system and adapt a preferred learning styles. It allows the learner to interact with the simulation software package whilst under instructional control. The tutorial of simulation software can be tailored extensively to suit this methodology. The architecture provides for multi-platform utilisation. This can be achieved by structuring the tutorial into tasks units and by separating the tutorials from the rest of the application. The shell can manage cross platform transfer across the various hardware platforms of Smalltalk and can be implemented within a range of curricular.

The object-oriented architecture ensures a uniform approach for analysis and design of the system, and consistencies of the system for courseware authoring and maintenance. The framework objects can be encapsulated, and allow control of data between all objects. Its inhabitant characteristics allow changes to be made to the tutorials when the software functionality is improved. The interactive development environment allows for the courseware author to visualise the functionality of the software and its components.

This reduces the time required for analysis and development of the courseware and improves the courseware quality. It also helps to standardise the curriculum and improves usability of the system. The architecture may also provide a platform for development and implementation of such systems.

References

[1] Alderman, D.L. (1978) Evaluation of the TICCIT Journal of Computer Assisted Instructional System in the Community College. *Final Report Vol. 1 Educational Testing Services*, Princeton, NY.

[2] Angelides M.C. & Paul R.J. (1993) Towards a Framework for Integrating Tutoring Systems and Gaming-Simulation. In *Proceedings of the Winter Simulation Conference*, (eds.) G.W. Evans, M. Mollaghasemi, E.C. Rusell, and W.E. Biles, p.1281-1289. ACM Press, Los Angeles, California.

[3] Booch, G. (1990) *Object oriented design with applications.* Benjamin/Cummings Publishing Company, Inc. USA.

[4] Bruner, J.S., Goodnow, J.J, Austin, G.A. (1956) *A study of thinking.* New York: Wiley.

[5] Burton, R.R. and Brown, J.S. (1982) An Investigation of Computer Coaching for Informal Learning Activities. In Sleeman, D. and Brown, J.S. (eds.) *Intelligent tutoring systems*, Academic Press, London.

[6] Draman, M. (1991) A Generic Architecture for Intelligent Simulation Training Systems, In *Proceedings of the Winter Simulation Conference.* ACM Press, Los Angeles, California.

[7] Gagne, R.M. (1965) *Learning Hierarchies.* New York: Holt, Rinehart, and Winston.

[8] Mitchell, P.D. (1978) A Classroom in a Computer for Lesson planning Practise. In Meardy, J, (ed.). *Perspectives on Academic Gaming and Simulation* p.191-204. Kogan Page.

[9] Patrick, J. (1992) *Training research and practice*, Academic Press.

[10] Pidd, M, (1992) *Computer simulation in management science*, Third Edition, John Wiley. London.

[11] Richard, T., Webb, G.I. and Craske, N. (1987) *Object Oriented control for ICAL systems.* Technical Report, T., La Trobe University, Boundoora, Australia.

[12] Richard, T., Webb, G.I. (1987) *Topic structuring in ECCLES: A simplified CAL authoring Methodology.* Technical Report 9, La University, Boundoora, Australia.

[13] Shneiderman, B. (1987) *Designing the user interface: strategies for effective human computer interaction*, Addison-Wesley, USA.

Steepest Descent Optimisation in the Secondary Space of Redundant Manipulators

Jadran Lenarčič and Bojan Nemec
Jožef Stefan Institute, Jamova 39, 1111 Ljubljana, Slovenia
Phone: +386 61 1773 563, Fax: +386 61 1258 058
jadran.lenarcic@ijs.si

Keywords: redundant manipulators, steepest descent optimisation, optimum step size

Edited by: Rudi Murn
Received: April 15, 1997 **Revised:** September 9, 1997 **Accepted:** September 30, 1997

To minimise a cost function in the secondary level of control of redundant manipulators, the gradient of the cost function is projected on the null space of the primary task. This projection, however, does not guarantee the maximum decrement of the cost function. In the present paper we rotate the null space of the primary task in order to find the maximum optimisation step. By this improvement, the optimisation procedure can provide the desired solution in much less iterations.

1 Introduction

One of the emerging issues of the new generation of manipulators, in particular service robots, is the kinematical redundancy. The mechanism of a redundant manipulator possesses superfluous degrees of freedom with respect to the motion constraints posed by the assigned primary task. Hence, as a secondary task, it can operate by simultaneously optimising a set of performance criteria or can avoid obstacles and ill-conditioned configurations. Assume that the primary task is to place the hand in a given spatial position and orientation. The rest of the mechanism will still be able to move (see Fig 1). This is the so-called self or secondary motion of the mechanism and is a typical property of kinematically redundant manipulators. From this viewpoint also the human arm contains a degree of redundancy. By rotating the elbow in a fixed pose of the hand, the human arm mechanism accommodate its static and dynamic characteristics.

A majority of efforts in treating redundancy have been concentrated at the kinematic level of control with respect to different type of criteria with robots moving in complex environments avoiding obstacles, as reported in [1], and undesired or ill-conditioned configurations, as reported in [2, 3]. Many authors based their approach on the utilisation of the pseudoinverse of the Jacobian matrix associated with the primary task, such as [4], which is a purely numeric approach. It has intrinsic inaccuracies and accumulates errors that become larger as the velocity increases. To overcome these difficulties, some authors developed symbolic solutions to this problem, such as [5]. It must be pointed out, however, that no symbolic solution can be developed for a general-type redundant manipulator unless certain conditions are met by the mechanism.

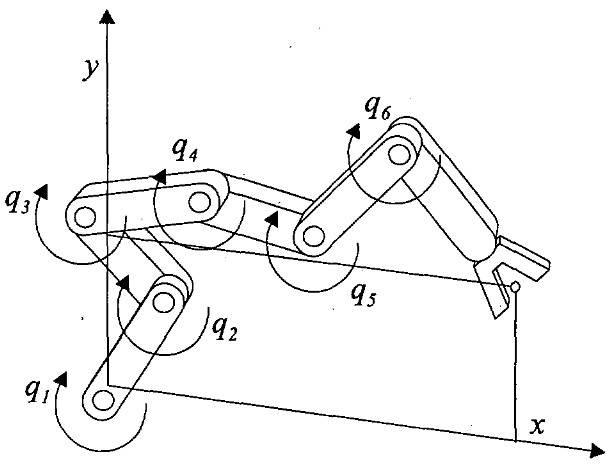


Figure 1: A 6-degree-of-freedom manipulator that positions end effector in x,y plane is kinematically redundant

The calculation schemes based on the pseudoinverse are procedures of local optimisation [6]. They implicitly minimise a norm of joint velocities. Yet, the central point and a distinctive property of various pseudoinverse-based methods is in the determination of the null space projection operator associated with the secondary task [7]. A proper selection of the null space projection operator in combination with the null space vector provides a secondary motion of the manipulator that respects different types of criteria, such as joint torque minimisation, obstacle avoidance, singular pose avoidance, etc.[8, 9]. [10, 11] reported some results obtaining global optima with integral type of criteria. [12] optimised the weighted null space projection operator in order to avoid instabilities. An alternative to the pseudoinverse-based methods is the extended Jacobian method introduced by [13]. [14] described a new formulation of this algorithm that is well suited for more general choices of the secondary criterion. Characteristics of different optimum solutions were studied in [15].

In a vast variety of applications, the secondary task of a redundant manipulator, formulated in terms of an optimisation, is based on a gradient-type procedure. Usually, not to disturb the primary task execution, this gradient is mapped in the null space of the Jacobian matrix pertinent to the primary task. Unfortunately, the projection in this null space may distort the gradient vector, so that in a final consequence it may not provide a decrease of the cost function and the optimisation procedure may become inefficient. In the present work, to overcome the difficulty, we rotate the null space operator in order to achieve the best direction and amplitude of the optimisation step in the domain of the secondary task. By this improvement, the optimisation procedure becomes more effective.

2 Mathematical background

Let the primary task of a redundant manipulator be to achieve some desired poses $\mathbf{p}$ (such as position and orientation of the hand) that are given as a function of joint coordinates $\mathbf{q}$ (angles or linear displacements in joints), where vector $\mathbf{p}$ is of dimension m and $\mathbf{q}$ of dimension n. Typically, vector $\mathbf{p}$ is a set of non-linear independent algebraic equations where $\mathbf{q}$ are arguments of trigonometric functions. It is expressed in the well know differential form [16] by

$$d\mathbf{p} - \mathbf{J}_\mathrm{P} d\mathbf{q} = 0, \tag{1}$$

where $\mathbf{J}_\mathrm{P}$ is a $m \times n$ Jacobian matrix that incorporates the derivatives $\partial \mathbf{p}/\partial q_\mathrm{i}$. We assume that $n > m$ and

$$\mathbf{J}_\mathrm{P}^+ d\mathbf{p} - d\mathbf{q} = 0. \tag{2}$$

If $\mathbf{J}_\mathrm{P}\mathbf{J}_\mathrm{P}{}^\mathrm{T}$ of dimension $m \times n$ isn't singular, the following

$$\mathbf{J}_\mathrm{P}^+ = \mathbf{J}_\mathrm{P}{}^\mathrm{T}(\mathbf{J}_\mathrm{P}\mathbf{J}_\mathrm{P}{}^\mathrm{T})^{-1} \tag{3}$$

is the so-called unweighted pseudoinverse of $\mathbf{J}_\mathrm{P}$ whose dimension is $m \times n$. The utilisation of the pseudoinverse in solving the task to move a redundant manipulator in a given $\mathbf{p}$ implicitly leads to a minimisation of joint velocities [16]. To satisfy other criteria, one can introduce additional optimisations that must not interfere with the primary task. Let the secondary task be expressed similarly to the primary task

$$d\mathbf{s} - \mathbf{J}_\mathrm{S} d\mathbf{q} = 0, \tag{4}$$

where the objective is to achieve a $\mathbf{q}$ that corresponds to given values of secondary task coordinates $\mathbf{s}$ (for instance joint torques), and $\mathbf{J}_\mathrm{S}$ is the Jacobian that includes derivatives $\partial \mathbf{s}/\partial q_\mathrm{i}$. If $\mathbf{s}$ is of dimension l, $\mathbf{J}_\mathrm{S}$ is of dimension $l \times n$. If $n > l$, we can take advantage of the $n \times l$ pseudoinverse

$$\mathbf{J}_\mathrm{S}^+ d\mathbf{s} - d\mathbf{q} = 0. \tag{5}$$

Arrange now the increment of joint coordinates into the primary $d\mathbf{q}_\mathrm{P}$ and the secondary $d\mathbf{q}_\mathrm{N}$ (subordinated) part, so that $d\mathbf{q}_\mathrm{N}$ does not produce any change of primary coordinates $\mathbf{p}$

$$d\mathbf{q} = d\mathbf{q}_\mathrm{P} + d\mathbf{q}_\mathrm{N}. \tag{6}$$

By multiplying with $\mathbf{J}_\mathrm{P}$ we get

$$d\mathbf{p} = \mathbf{J}_\mathrm{P} d\mathbf{q}_\mathrm{P} + \mathbf{J}_\mathrm{P} d\mathbf{q}_\mathrm{N}. \tag{7}$$

Since

$$\mathbf{J}_\mathrm{P} d\mathbf{q}_\mathrm{N} = 0, \tag{8}$$

we have

$$d\mathbf{p} = \mathbf{J}_\mathrm{P} d\mathbf{q}_\mathrm{P} \Rightarrow d\mathbf{q}_\mathrm{P} = \mathbf{J}_\mathrm{P}^+ d\mathbf{p} \tag{9}$$

Let

$$d\mathbf{q}_\mathrm{N} = \mathbf{N}_\mathrm{P} d\mathbf{q}_\mathrm{S} \tag{10}$$

where $d\mathbf{q}_\mathrm{S}$ is an arbitrary n-dimensional vector of joint increments associated with the secondary task, while $\mathbf{N}_\mathrm{P}$ is of dimension $n \times n$

$$\mathbf{J}_\mathrm{P}\mathbf{N}_\mathrm{P} = 0. \tag{11}$$

According to [17]

$$\mathbf{N}_\mathrm{P} = \mathbf{I} - \mathbf{J}_\mathrm{P}^+\mathbf{J}_\mathrm{P} \tag{12}$$

lies in the null space of $\mathbf{J}_\mathrm{P}$ so that $d\mathbf{q}_\mathrm{N}$ and $d\mathbf{q}_\mathrm{P}$ are orthogonal. The null space projection operator $\mathbf{N}_\mathrm{P}$ is Hermitian and idempotent, $\mathbf{N}_\mathrm{P}{}^\mathrm{T} = \mathbf{N}_\mathrm{P}, \mathbf{N}_\mathrm{P}\mathbf{N}_\mathrm{P} = \mathbf{N}_\mathrm{P}$.

The kinematic redundancy, in our opinion, must be understood as an instantaneous property associated with the dimension of the null space of the primary task. We thus define the degree of redundancy as the rank of the null space projection operator

$$D = rank\,\{\mathbf{N}_\mathrm{P}\} = n - rank\,\{\mathbf{J}_\mathrm{P}\}\,. \qquad (13)$$

D is the achievable order of the secondary motion that can change throughout the workspace of the manipulator mechanism in dependence on $\mathbf{q}$. We assume in this work that $l \geq D$.

By substituting (10) into (6) and by multiplying with JS we have

$$\begin{aligned} d\mathbf{s} &= \mathbf{J}_\mathrm{S} d\mathbf{q}_\mathrm{P} + \mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P} d\mathbf{q}_\mathrm{S} \qquad (14)\\ \Rightarrow d\mathbf{q}_\mathrm{S} &= (\mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P})^+ (d\mathbf{s} - \mathbf{J}_\mathrm{S} d\mathbf{q}_\mathrm{P}). \end{aligned}$$

The secondary task coordinates depend on $d\mathbf{q}_\mathrm{N}$ and $d\mathbf{q}_\mathrm{P}$. We denote

$$\begin{aligned} d\mathbf{s}_\mathrm{P} &= \mathbf{J}_\mathrm{S} d\mathbf{q}_\mathrm{P},\\ d\mathbf{s}_\mathrm{N} &= \mathbf{J}_\mathrm{S} d\mathbf{q}_\mathrm{N}\\ \Rightarrow d\mathbf{s} &= d\mathbf{s}_\mathrm{P} + d\mathbf{s}_\mathrm{N}. \qquad (15) \end{aligned}$$

If we take into account (6,10,14), a complete increment in joint coordinates can be written as

$$d\mathbf{q} = \mathbf{J}_\mathrm{P}{}^+ d\mathbf{p} + \mathbf{N}_\mathrm{P}(\mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P})^+ (d\mathbf{s} - \mathbf{J}_\mathrm{S}\mathbf{J}_\mathrm{P}^+ d\mathbf{p}), \qquad (16)$$

which is the know task priority approach where the first part of the joint increment is the particular solution associated with the primary task. It is of a higher priority in comparison with the second part which is the homogeneous solution associated with the secondary task [1, 4].

3 Secondary domain

In this work, an increment in joint coordinates is referred to as the secondary motion (self-motion) of a redundant manipulator when it does not produce any increment in the primary task coordinates. In connection with the definition of the manipulability ellipsoids [18], a sphere $d\mathbf{q}_\mathrm{S}^\mathrm{T} d\mathbf{q}_\mathrm{S} = 1$ produces an ellipsoid in l-dimensional space of $d\mathbf{s}$ whose principal axes are the eigenvectors of $\mathbf{J}_\mathrm{S}\mathbf{J}_\mathrm{S}^\mathrm{T}$ and their lengths are the related singular values. It is clear, however, that only a part of elements $d\mathbf{s}$ in this ellipsoid can be accomplished by the secondary motion of the redundant manipulator. Note that there is some controversy in the definition and utilisation of manipulability when different types of task coordinates are treated simultaneously, such as linear and angular velocities. [19, 8] dealt with this problem.

The secondary motion can only be assembled in the null space of the Jacobian matrix $\mathbf{J}_\mathrm{P}$ where a vector $d\mathbf{q}_\mathrm{S}$ is projected through $\mathbf{N}_\mathrm{P}$ into $d\mathbf{q}_\mathrm{N}$. Thus, an element of a sphere $d\mathbf{q}_\mathrm{S}^\mathrm{T} d\mathbf{q}_\mathrm{S} = 1$ is transformed in the sphere $d\mathbf{q}_\mathrm{N}^\mathrm{T} d\mathbf{q}_\mathrm{N} \leq 1$. The null space of $\mathbf{J}_\mathrm{P}$ is spaned by the n-dimensional orthonormal eigenvectors of matrix $\mathbf{N}_\mathrm{P}\mathbf{N}_\mathrm{P}^\mathrm{T} = \mathbf{N}_\mathrm{P}$, denoted by $\mathbf{E}_\mathrm{P} = (\mathbf{e}_1, \mathbf{e}_2, \ldots\ldots, \mathbf{e}_\mathrm{P})$. Thus, any $d\mathbf{q}_\mathrm{N}$ as a function of parameters $\gamma = (\gamma_1, \ldots\ldots, \gamma_D)^\mathrm{T}$

$$d\mathbf{q}_\mathrm{N} = \mathbf{E}_\mathrm{P}\gamma \qquad (17)$$

is an element of a sphere if

$$d\mathbf{q}_\mathrm{N}^\mathrm{T} d\mathbf{q}_\mathrm{N} = \gamma_1^2 + \gamma_2^2 + \ldots. + \gamma_D^2 = 1. \qquad (18)$$

The values of $d\mathbf{s}_\mathrm{N}$ (15) that are functions of $d\mathbf{q}_\mathrm{N}$ constrained by (18) form that part of the vector space of $d\mathbf{s}$ that can be accomplished by the secondary motion of the manipulator. Vectors $d\mathbf{s}_\mathrm{N}$ produce an ellipsoid whose principal axes are the l-dimensional eigenvectors of matrix $\mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P}(\mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P})^\mathrm{T}$ (denoted by $\mathbf{u}_1, \mathbf{u}_2, \ldots\ldots, \mathbf{u}_D$), while the corresponding singular values of $\mathbf{J}_\mathrm{S}\mathbf{N}_\mathrm{P}$ are the lengths of these principal axes. The secondary manipulability ellipsoid characterises the domain of the secondary task bounded by (18). It describes the potential of a redundant manipulator to solve the secondary task with no interference with the primary task in a given location determined by the joint coordinates $\mathbf{q}$ [20].

4 Best optimisation step

Assume that the secondary task of a redundant manipulator is to minimise a quadratic cost function (which is a very reasonable assumption for a vast variety of practical implementations, e.g. joint torque minimisation)

$$c = \mathbf{s}^\mathrm{T}\mathbf{W}\mathbf{s} \rightarrow min|_q \;\;, \qquad (19)$$

where $\mathbf{W}$ is a diagonal $l \times l$ full-rank matrix of positive weights w_i, while $d\mathbf{q}_\mathrm{N}$ is constrained by the primary task. An iterative procedure is to provide a series of $d\mathbf{q}_\mathrm{N}$ that step by step minimise c and do not interfere with the primary task. In a commonly used gradient projection technique to minimise a scalar cost function [17, 8], the joint displacement is

$$d\mathbf{q} = k_\mathrm{P}\mathbf{J}_\mathrm{P}^+ d\mathbf{p} - k_\mathrm{S}\mathbf{N}_\mathrm{P}\mathbf{J}_\mathrm{S}^+ \left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\}. \qquad (20)$$

where k_S and k_P are selected in order to assure the convergence of the numerical procedure. The trouble with such an approach is that in general the projection of the gradient in the null space of $\mathbf{J}_\mathrm{P}$ may not provide the maximum decrease of the cost function. Hence, the iterative procedure may not be able to locate the desired minimum in an acceptable number of iterations. There are two remedies for it. One is to adapt the projection vector directly in the domain of the secondary task. This can be done by searching the optimum projection vector produced by joint increments that lie the secondary domain. The other way is to adequately rotate the null space of $\mathbf{J}_\mathrm{P}$. In the present paper we show only the second possibility.

It is well known [16] that a weighted pseudoinverse

$$\mathbf{J}_\mathrm{PA}^+ = \mathbf{A}^{-1}\mathbf{J}_\mathrm{P}^\mathrm{T}(\mathbf{J}_\mathrm{P}\mathbf{A}^{-1}\mathbf{J}_\mathrm{P}^\mathrm{T})^{-1} \tag{21}$$

minimises a performance criterion of joint velocities in the form of $\dot{\mathbf{q}}^\mathrm{T}\mathbf{A}\dot{\mathbf{q}}$ where $\mathbf{A}$ is a positive definite weighting matrix. The weighted null space projection operator for the weighted pseudoinverse $\mathbf{J}_\mathrm{PA}^+$ is given by

$$\mathbf{N}_\mathrm{PA} = \mathbf{I} - \mathbf{J}_\mathrm{PA}^+\mathbf{J}_\mathrm{P}. \tag{22}$$

This null space projection operator is idempotent but in general is not Hermitian. It rotates the null space of the Jacobian in what was termed the effective null space by [12]. It is the central point of many attempts to resolve redundancy with solutions possessing different characteristics that assist to avoid obstacles and singularities or minimise a given criterion. In [7] one can find a variety of possibilities later implemented and improved by other authors. For instance, [10, 11] used this approach in order to incorporate some global properties in the optimisation of dynamic joint torques. [21] showed the limitations of these methods when they are implemented independently on a real mechanism without considering its kinetic behaviour. Lately, [12] reported a weighted null space projection operator suitable for a weighted joint torque optimisation with the aim to avoid instabilities that arise from unrealisable joint velocities.

We show here that it is possible to determine a weighted null space projection operator that provides a steepest descent minimisation of a given cost function. For this purpose we search for an optimum $d\mathbf{s}$ that minimises a quadratic form $c' = c(\mathbf{s} + d\mathbf{s})$ (where c is given in (19)) with respect to the primary task (1).

The associated Lagrangian, in which we substitute $d\mathbf{s}$ by $\mathbf{J}_\mathrm{S}d\mathbf{q}$, is as follows

$$\begin{aligned} L = \mathbf{s}^\mathrm{T}\left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\} + 2d\mathbf{q}^\mathrm{T}\mathbf{J}_\mathrm{S}^\mathrm{T}\left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\} + \\ + d\mathbf{q}^\mathrm{T}\mathbf{J}_\mathrm{S}^\mathrm{T}\mathbf{W}\mathbf{J}_\mathrm{S}d\mathbf{q} + \lambda^\mathrm{T}(d\mathbf{p} - \mathbf{J}_\mathrm{P}d\mathbf{q}) \end{aligned} \tag{23}$$

Here,λ is the vector of Lagrangian multipliers. The derivative of L with respect to $d\mathbf{q}$ must vanish

$$\left\{\frac{\partial L}{\partial dq_\mathrm{i}}\right\} = 2\mathbf{J}_\mathrm{S}^\mathrm{T}\left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\} + 2\mathbf{J}_\mathrm{S}^\mathrm{T}\mathbf{W}\mathbf{J}_\mathrm{S}d\mathbf{q} - \mathbf{J}_\mathrm{P}^\mathrm{T}\lambda = 0, \tag{24}$$

as well as its derivative with respect to λ so that (1) holds. As a result,

$$d\mathbf{q} = \frac{1}{2}\mathbf{A}^{-1}\mathbf{J}_\mathrm{P}^\mathrm{T}\lambda - \mathbf{A}^{-1}\mathbf{J}_\mathrm{S}^\mathrm{T}\left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\} \tag{25}$$

where we introduced the following notation

$$\mathbf{A} = \mathbf{J}_\mathrm{S}^\mathrm{T}\mathbf{W}\mathbf{J}_\mathrm{S}. \tag{26}$$

Since $\mathbf{A}$ is singular when $l > m$ and cannot directly be inverted, one can utilise either the damped least square solution

$$\mathbf{A}^{-1} = (\mathbf{J}_\mathrm{S}^\mathrm{T}\mathbf{W}\mathbf{J}_\mathrm{S} + \alpha\mathbf{I})^{-1} \tag{27}$$

or the formulation

$$\mathbf{A}^{-1} = \mathbf{J}_\mathrm{S}^+\mathbf{W}^{-1}(\mathbf{J}_\mathrm{S}^+)^\mathrm{T}. \tag{28}$$

Multiplying (25) by $\mathbf{J}_\mathrm{P}$ gives

$$\frac{1}{2}\lambda = (\mathbf{J}_\mathrm{P}\mathbf{A}^{-1}\mathbf{J}_\mathrm{P}^\mathrm{T})^{-1}(d\mathbf{p} + \mathbf{J}_\mathrm{P}\mathbf{A}^{-1}\mathbf{J}_\mathrm{S}^\mathrm{T}\left\{\frac{\partial c}{\partial s_\mathrm{i}})\right\}. \tag{29}$$

Then by substituting λ again in (25), by taking into account (21,22), as well as $\mathbf{A}^{-1}\mathbf{J}_\mathrm{S}^\mathrm{T} = \mathbf{J}_\mathrm{S}^+\mathbf{W}^{-1}$, and by introducing scalar constants $k_\mathrm{P}, k_\mathrm{S}$ to control the iteration step we get

$$d\mathbf{q} = k_\mathrm{P}\mathbf{J}_\mathrm{PA}^+d\mathbf{p} - k_\mathrm{S}\mathbf{N}_\mathrm{PA}\mathbf{J}_\mathrm{S}^+\left\{\frac{\partial c}{\partial s_\mathrm{i}}\right\}, \tag{30}$$

which is analogous to the standard formula (20) with a weighted pseudoinverse and the corresponding null space projection operator, while the weighting matrix is given by (26) and its inverse by (27) or (28).

The approach (30) turns out to be numerically very effective if compared to (20). The necessary number of iterations to solve the primary and the secondary task can drastically be decreased. While in some cases the formulation (20) fails to carry out the imposed optimisation in the secondary level, the approach formulated in (30) guarantees the best solution to the primary and to the secondary task simultaneously.

5 Conclusions

The paper deals with a steepest descent local minimisation in the level of the secondary task of a redundant manipulator. In previously reported pseudoinverse-based optimisation techniques, the gradient of a given cost function is directly mapped in the null space of the primary task. The projection in the null space may distort the gradient and the optimisation procedure may be troubled. In our work, we overcome the problem by utilising an optimum weighted pseudoinverse. By this improvement, the optimisation procedure becomes more effective and numerically robust. It finds a solution in less iterations and its potential to locate the optimum solution is greater.

References

[1] Maciejewski, A.A., Klein, C.A. (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments, *Int. J.Robotics Res.* Vol. 4, No. 4, pp.109-117.

[2] Baker, D.R., Wampler, C.W. (1988) On the inverse kinematics of redundant manipulators, *Int. J.Robotics Res.*, Vol. 7, No. 2, pp. 3-21.

[3] Chang, K.-S., Khatib, O. (1994) A dynamically consistent strategy for manipulator control at singularities, Advances in Robot Kinematics, *Kluwer Academic Publ.*, Dordrecht, pp. 221-228.

[4] Nakamura, Y., Hanafusa, H., Yoshikawa, T. (1987) Task-priority based redundancy control of robot manipulators, *Int. J.Robotics Res.*, Vol. 6, No. 2, pp. 3-15.

[5] Ghosal, A., Roth, B. (1988) A new approach for kinematic resolution of redundancy, *Int. J.Robotics Res.*, Vol. 7, No. 2, pp. 22-35.

[6] Nenchev, D.N. (1989) Redundancy resolution through local optimisation: A review, *Int. J. Robotic Systs.*, Vol. 6, No. 6, pp. 769-798.

[7] Hollerbach, J.M., Suh, K.C. (1987) Redundancy resolution of manipulators through torque optimization, *IEEE J. Robotics and Automat.*, Vol. 3, No. 4, pp. 308-316.

[8] Yoshikawa, T.(1996) Basic optimization methods of redundant manipulators, *Laboratory Robotics and Automat.*, Vol. 8, No. 1, pp. 49-60.

[9] Khatib, O. (1996) The impact of redundancy on the dynamic performance of robots, *Laboratory Robotics and Automat.*, Vol. 8, No. 1, pp. 37-48.

[10] Nedungadi, A. Kazerounian, K. A (1989) local solution with global characteristics for joint torque optimisation of a redundant manipulator, *J.Robotic Systs.*, Vol.6, No. 5., pp. 631-654.

[11] Kazerounian, K., Wang, Z. (1988) Global versus local optimization in redundancy resolution of robotic manipulators, *J. Robotics Res.*, Vol. 7, No. 5, pp. 3-13.

[12] Hu, B., Teo, C.L., Lee, H.P. (1995) Local optimization of weighted joint torques for redundant robotic manipulators, *IEEE Trans. on Robotics and Automat.*, Vol. 11, No. 3, pp. 422-425.

[13] Baillieul, (1986) J. Avoiding obstacles and resolving kinematic redundancy, *IEEE Int. Conf. Robotics and Automat.*, pp. 1698-1704.

[14] Klein, C.A., Chu-Jenq, C., Ahmed, S. (1995) A new formulation of the extended Jacobian method and its use in mapping algorithmic singularities for kinematically redundant manipulators, *IEEE Trans. on Robotics and Automat.*, Vol. 11, No. 1, pp. 50-55.

[15] Park, J., Chung, W.-K., Youm, Y. (1996) Characteristics of optimal solutions in kinematic resolutions of redundancy, *IEEE Trans. on Robotics and Automat.*, Vol. 12, No. 3, pp. 471-478.

[16] Whitney, D.E. (1969) Resolved motion rate control of robot manipulators and human prostheses, *IEEE Trans. Man-Mach. Sys.*, Vol. 10, No. 2, pp. 47-53.

[17] Liegois, A. (1977) Automatic supervisory control of the configuration and behaviour of multibody mechanisms, *IEEE Trans. Sys., Man, Cybern.*, Vol. 7, No. 12, pp. 868-871.

[18] Yoshikawa, (1985) T. Manipulability of robotic systems, *Int. J. Robotics Res.*, Vol. 4, No. 2, pp. 3-9.

[19] Doty, K.L., Melchiorri, C., Schwartz, E.M., Bonivento, C. (1995) Robot Manipulability, *IEEE Trans. on Robotics and Automat.*, Vol. 11, No. 3, pp. 462-468.

[20] Nenchev, D.N. (1992) Restricted Jacobian matrices of redundant manipulators in constrained motion tasks, *Int. J. Robotics Res.*, Vol. 11, No. 6, pp. 584-597.

[21] Maciejewski, A.A. (1991) Kinetic limitations on the use of redundancy in robotic manipulators, *IEEE Trans. on Robotics and Automat.*, Vol. 7, No. 2, pp. 205-210.

Encapsulation — the Key to Software Component Re-Use

Raimund K. Ege
School of Computer Science
Florida International University, University Park, Miami, FL 33199
phone: (305) 348-3381 fax: (305) 348-2336
ege@cs.fiu.edu

Keywords: software, object-orientation, encapsulation, programming languages

Edited by: Rudi Murn
Received: January 24, 1997 **Revised:** September 9, 1997 **Accepted:** September 23, 1997

Successful software today is built from components. Modern software engineering techniques, such as those based on the object paradigm, support the specification, implementation and re-use of components. Encapsulation is the term used to describe this basic knowledge representation principle of packaging information and its related functionality. This paper explores the concept of encapsulation as it is used in object-oriented programming languages, analysis and design methods, distributed systems and data bases. The paper uses a specification language - KAPSEL - that facilitates the specification of re-usable object-oriented software components to illustrate advanced encapsulation features.

1 Introduction

The packaging of software components is a key feature of object-oriented software technology. An object is an instantiation of a re-usable software component – a class. Object-based technology already allowed the specification and implementation of such components. Object-oriented technology goes a few steps further by allowing that these components are related and organized into inheritance hierarchies.

Encapsulation is the concept of packaging software components [Parnas et al., 1985] by specifying their interfaces to their users or clients and hiding from them their internal implementation details. In a true object-oriented context, where classes belong to class hierarchies and whole-part associations, encapsulation issues can become quite complicated. The purpose of the research reported in this paper was to investigate and explore variations in the specification of encapsulation for object-oriented software components and their relationships. The result of the research is a specification language - KAPSEL - which allows a great degree of flexible but strictly controlled access to software components.

There is some confusion and dispute among the object-oriented community regarding the correct definition of encapsulation [Knight, 1992]. This paper is not an attempt to solve the dispute. Instead, we offer our own definition of encapsulation:

> Encapsulation is the collection and protection of attributes for an object and its exclusive, associated services.

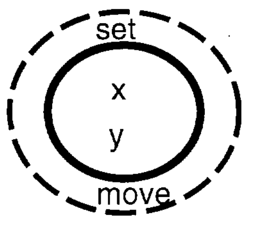


Figure 1: an encapsulated point

As example, consider a "Point" class (see Figure 1), which defines two attributes ("x" and "y") to represent the coordinates of a point, and two services ("set" and "move") that allow to store some initial values for coordinates and allow them to be changed, i.e. for the point to move. Instances of class "Point", i.e. point objects, each encapsulates 2 attributes and 2 services. Conventional object-oriented wisdom would assume that the attributes are hidden by each object, only to be modified via the specified services. But the degree of encapsulation can be more arbitrary: given a "point", are its "x" and "y" coordinates accessible; are its services "set" and "move" invocable ? It is the purpose of encapsulation specification to control the degree of visibility or hiding that each software component allows.

Object-oriented technology knows many ways and modes of controlling access to elements of software components. The goal of this paper is to describe our categorization of forms and variations in encapsulation control and to introduce a specification language

"KAPSEL" that enables the broadest and most detailed of access control.

2 Encapsulation

The focus of our work is not so much the pure concept of encapsulation, but rather its specification. If encapsulation is "collection and protection", then our question is: how can it be specified and controlled ? While encapsulation deals with the visibility of attributes and services, we ask how this visibility can be specified in the most flexible way and how it can be controlled.

In the following we will discuss various issues that relate to the understanding and specification of encapsulation:

- unit of encapsulation
- granularity of encapsulation
- class hierarchy encapsulation
- whole-part association encapsulation

Our discussion of KAPSEL will rely on an understanding of these issues.

Encapsulation Unit

The most natural unit of encapsulation is an object: it has attributes and services that can be protected from access by others. Most object-oriented programming languages follow this model. For example, if an attribute of an object is declared to be "private", i.e. to be protected and not visible, then it is not accessible from any other object [1]. Typically, this degree of encapsulation is specified for all objects of the same class as part of the class declaration. Notable exception among the object-oriented programming languages are C++ [Ellis and Stroustrup, 1990] and Java [Gosling and McGilton, 1996] which consider the class as unit of encapsulation: for example in C++, given 2 points with private coordinate attributes, one point is able to access the coordinates of the other point, whereas in Smalltalk [Goldberg and Robson, 1983] or Eiffel [Wiener, 1995], one point is not able to "see" the coordinates of another.

"Object as encapsulation unit" is more natural, whereas "class as encapsulation unit" appeals to the software developer whose unit of concern is the specification of a set of objects: and this set of objects is the class.

[1] of course, the object itself still has access to all its private attributes.

Access Granularity

Access to attributes and services can be controlled. The smallest "granule" (i.e., unit of access specification) can be either a single attribute or a single service. The largest granule can be that "all services" or "all attributes" are accessible. Smalltalk is an object-oriented programming language that treats encapsulation control in-the-large: all attributes (instance variables) are not visible, all services (instance methods) are generally accessible. Eiffel, C++ and Java allow a per attribute/service specification of accessibility.

Also of importance is the nature of the access: what effect does the access have on the state of the object. Access to an attribute is possible in a "initialization" (to set stored value for an attribute once, initially), "read"(to get the stored value for an attribute) or "write"(to change the stored valued) sense. Access for a service typically is meant to allow the sending of a message to the object to request the performance of a service. The service invocation can further be classified into one that would read only, initialize, or write, based on whether the service will affect the state of the object.

Again, languages disagree here: in C++ for example, if an attribute is "public", i.e. visible, then it can be initialized, read and written from "outside", whereas in Eiffel, if an attribute is exported, it is only readable: Eiffel maintains that while attributes might be made visible, they are still part of an object's internals, and outside change has to be affected through appropriate messages to the object.

Object creation and deletion can also be considered within the scope of encapsulation specification: to create an object is typically a service provided by the class and therefore can be specified as visible or not; to delete an object is an operation upon an object and therefore can be specified as visible or not, as well.

Specification can be active or passive. Active encapsulation control specifies which other objects or classes can "see" specific attributes or services. Examples are Eiffel, where attributes and services are explicitly exported; or C++ where the "friend" mechanism allows the specification of a class that can access all attributes and services of an object. Passive specification classifies attributes and/or services as visible or not: something that is visible is then accessible to any client object that would want to. Active encapsulation specification is typically not transitive, that is if class A actively allows class B access, and B allows C, then C does not have access to A. Eiffel modifies this concept slightly in that it allows all subclasses of B to access A, and all subclasses of C access to B. And, of course, active encapsulation specification is not symmetric.

Class Hierarchy

Inheritance has to be considered in the context of encapsulation specification in 2 ways:

1. which attributes and services of the superclass are accessible to the subclass ? There are 3 modeling choices:

   (a) the subclass is just like any other regular client, and therefore has access to only the visible attributes and services;

   (b) the subclass has unrestricted access to all attributes and services of the superclass;

   (c) the superclass can explicitly and selectively make attributes and services visible to that subclass.

   C++ is an example of the third choice: it uses the keyword "private" to deny subclass access, and the keyword "protected" to explicitly allow access by the subclass. Most other object-oriented programming languages use choice (b), which seems liberal and does not protect a superclass from subclass problems. The first choice is too limited: the subclass is in the same family as the superclass and therefore should gain some privileges.

2. how accessible are inherited attributes and services in an instance of a subclass ? In general, attributes and services should maintain their accessibility in subclasses; the subclass otherwise violates the "behavior-compatibility" maxime in true object-oriented class hierarchies. C++, for example, allows that subclasses restrict the visibility of inherited attributes and services via its "protected" and "private" inheritance mechanism. Eiffel has a similar capability in that subclasses are allowed to change the export status of features (attributes or services). Both mechanisms allow the definition of classes that are "less" than their superclasses: a typical example is to define a "Stack" class as a subclass of "LinkedList" by renaming and hiding the appropriate services. However a "Stack" is NOT a "LinkedList": the subclass is not behavior compatible to its superclass.

Whole-Part Association

Encapsulation control can also be specified for complex objects that contain other (or refer to other) objects. It is possible to view these contained objects as "regular" attributes of the object, whose accessibility can be specified in detail.

However, it is desirable to consider the special relationship between the whole and part objects. For example, consider a point object and a line object; further consider that the line object is defined as 2 points: head and tail. Is it necessary for the line to go through the public protocol to access the coordinates of its head and tail points, or could one specify special privileges for the whole that contains the parts? Of course, this special access should not hold for all line objects. Moreover, what about a special relationship between the head and tail points of a single line ? Should the head point be able to access private attributes of the tail point in the line[2] ? In KAPSEL, we are introducing a "key" concept that would allow the line object to gain special access privileges to its head and tail points and to grant them to other objects (see Section 4 of this paper).

For both class hierarchy and whole-part association encapsulation control specification graphs can be used to visualize object and class dependencies.

3 Encapsulation Support

Encapsulation is at the heart of object-orientation, however, support for encapsulation control and specification varies in object-oriented programming languages and analysis and design methods and also in distributed object approaches[Ege, 1994]. In this section we will discuss encapsulation support at these 3 levels: analysis and design methodologies, programming languages and distributed object systems.

OOA & D

In our opinion, current OOA&D methodologies fall into two categories in respect to their support for encapsulation specification and control:

1. *The OO Pioneers:* like Smalltalk, these methodologies consider attributes as part of the internal state of an object that should not be an immediate concern to the analyst. Services are considered part of the public protocol of an object, and therefore freely accessible to any client. Examples of true "OO Pioneer" methods are the Booch [Booch, 1994] approach and "Responsibility Driven OO design" [Wirfs-Brock et al., 1990].

2. *The OO Practitioners:* their approach is more practice-oriented, attributes can be specified but typically are not considered to be directly accessible. Services are accessible in general. Whole-Part associations are explicit and therefore freely accessible. Examples of methods are OMT [Rumbaugh et al., 1991] and the Shlaer/Mellor [Shlaer and Mellor, 1992] approach.

Both groups of methods lack serious support and tools to enable encapsulation control. OO analysis and design methods in general lag significantly behind

[2]of course, in C++ that would be the case anyway.

such capable object-oriented programming languages as Eiffel and C++. One notable exception is the "Law of Demeter" [Lieberherr, 1989] which deals with encapsulation guidelines during object-oriented design.

Programming Languages

As mentioned earlier in the paper, Smalltalk takes the wholesale approach that attributes should be hidden inside an object for the sole access by the object itself. Services make up the protocol of messages that an object understands: any client object can send any message as long as there is such a service. Smalltalk stands out as the pioneer of true OO understanding, however, various improvements in the area of encapsulation control have been proposed for it.

Newer languages, such as Eiffel, C++ and Java, have significantly more language constructs to deal with encapsulation control.

They allow a single attribute or service as smallest "granule" of specification; both support active (list of targets) or passive exportation; both can give dedicated access to its subclasses; and both can vary the access control to attributes and services inherited by a subclasses (e.g. "private" and "protected" inheritance in C++ and Java).

They differ in their treatment of the encapsulation unit: in Eiffel the unit is the object; in C++ and Java the unit is the class[3]. Another difference is their treatment of explicit exportation: in Eiffel explicit exportation gives access to the target class and all its subclasses, whereas subclasses of "friends" in C++ don't have access to the attributes and services of the class which extended the "friendship".

In Smalltalk, Eiffel, C++ and Java whole-part associations are treated uniformly as attributes, they enjoy the same level of encapsulation control as regular attributes.

DOS / DB

Of growing importance is encapsulation control in large systems that are assembled from object-oriented components. Rarely is it feasible to enforce a single policy (most likely via the adoption of a single programming language). The question is how to specify and control access to and from objects in an environment of heterogeneous object-systems. There are 2 basic approaches: a distributed object system that follows a common architecture (e.g. CORBA [Group, 1991]); or a unifying single[4] repository of objects in a common object-oriented data base.

[3]C++ treats accessibility of members in a class as a scope issue: it supports class scope, but not object scope.

[4]This is meant to be a logically single repository: notwithstanding that the data base can support multiple servers/volumes/clients.

Encapsulation control in such systems is still in its infancy: CORBA, for example, takes the approach in its interface definition language (IDL) that only the visible features of a class need to be declared, no variation is allowed. Attributes or services that are listed in an IDL definition of a class are accessible. The only additional control measure is that attributes can be declared read-only.

Encapsulation control in object-oriented data bases [Loomis, 1995] is still dominated by its data definition language, which typically is either Smalltalk or C++.

4 KAPSEL - The Language

KAPSEL's goal is to facilitate the re-use of object-oriented software components. We designed its features to be as inclusive as possible to allow experimentation with the encapsulation control measures discussed in this paper and to allow evaluation. KAPSEL is used in our IMN project [Ege and Stary, 1992] that combines object-orientation with constraints in the context of building task-oriented user interfaces[5].

KAPSEL is based on Smalltalk-80 [Kolla, 1996]. We chose Smalltalk-80 because of its very open and flexible system approach, but also because it has the least flexibility of its own to vary encapsulation control.

The unit of encapsulation in KAPSEL is the object. Since KAPSEL is based on Smalltalk, it is therefore also possible to express control for a class, since a class is represented and is available as a "class description" object in Smalltalk.

The granularity of access control is a single attribute or service. Read or write access can be specified: write access includes read access.

Active and passive exportation is supported. For active exportation a target class is specified, it can gain regular or transitive access. Transitive access allows the target class to further extend this access privilege to other classes. Passive exportation can be specified for subclasses or in general. General access allows any other object in the system to access the attribute or service.

KAPSEL supports the concept of "key" access: when a KAPSEL class is specified, the specification may include a key, which is a set of access specifications per attribute or service; this key is given to the creator of an instance of the class; the creator can access the new object according to the key, or can give the key to other objects; keys can be copied; keys can be reduced, where access privileges to attributes and/or services can be removed; in order to gain access to an object via a key, an object has to register with the object that is to be accessed[6]. This "key" concept allows

[5]KAPSEL also allows the specification of constraints. This aspect of KAPSEL is not within the scope of this paper

[6]The creator object is automatically registered.

```
Kapsel subclass: #Point
   instanceVariableNames: 'x y'
   instanceMethodNames: 'moveByX:Y:'
   classVariableNames: ''
   classMethodNames: ''
   keyDictionaries: #( x->general y->general
                       moveByX:Y:->(general write))
   poolDictionaries: ''
   category: 'Kapsel-Examples'.
```

Figure 2: KAPSEL Point Class

```
Kapsel subclass: #Point
   instanceVariableNames: 'x y'
   instanceMethodNames: 'moveByX:Y:'
   classVariableNames: ''
   classMethodNames: ''
   keyDictionaries: #(x->(VisualComponent read)
         y->(VisualComponent read)
         x->(Line write transitive) y->(Line write transitive))
   poolDictionaries: ''
   category: 'Kapsel-Examples'.
```

Figure 3: KAPSEL Point Class

very flexible, per object, access specification of encapsulation control. It is also the implementation vehicle that is used in our implementation of KAPSEL encapsulation enforcement.

Examples

KAPSEL examples would best be shown interactively in the context of the Smalltalk-80 class browser. For this paper we extracted the relevant definition statements: they resemble Smalltalk-80 syntax.

The first example (see Figure 2) shows a simple "Point" class with two attributes ("x" and "y") to represent its coordinates and a service that allows to logically move a point by a distance (i.e., change its coordinates).

The coordinates "x" and "y" are exported and made visible to all other objects in the system via "general" exportation. Access privilege defaults to "read" if not otherwise specified. The service "moveByX:Y:" is also made accessible, but with a "write" privilege, allowing the service to change the state of the object. If not specified, then service access defaults to "general write".

Figure 3 shows a variation of the "Point" class where "x" and "y" are not generally exported, but are rather made available explicitly to class "VisualComponent" for read access, and also made available to class "Line" for transitive write access which allows the "Line" class to export access to its points.

Figure 4 shows a linked list class: it makes its services available in general, but limits access to its "firstNode" attribute to its subclasses and allows only read access.

Figure 5 shows class "Stack" as a subclass of "LinkedList". It defines services "push" and "pop" to enable ordinary stack processing. It also restricts the accessibility of its inherited services to "hierarchy" hiding the fact that "Stack" is-a "LinkedList" from other objects that are not instances of subclasses.

Figure 6 again shows another version of the "Point" class: it defines a "creatorKey" to allow write access to the "x" and "y" coordinates.

Whenever an instance of class "Point" is created, the creating object is allowed to retrieve the creator key from the "Point" class. Keys can be used to register with a class to gain special access to an object. Keys can be passed and copied freely from object to object. Keys can be reduced, i.e. access privileges can be removed from it, but, of course, none can be added.

Implementation

KAPSEL has been implemented [Kolla, 1996] as an extension to Smalltalk-80 under the VisualWorks environment running on a SparcStation. Our goal is to use KAPSEL in the context of our user interface specification project [Ege and Stary, 1992]. Figure 7 shows

```
Kapsel subclass: #LinkedList
   instanceVariableNames: 'firstNode'
   instanceMethodNames: 'append: insert: removeFront removeEnd'
   classVariableNames: ''
   classMethodNames: ''
   keyDictionaries: #(firstNode->(hierarchy read))
   poolDictionaries: ''
   category: 'Kapsel-Examples'.
```

Figure 4: KAPSEL LinkedList Class

```
LinkedList subclass: #Stack
   instanceVariableNames: '
   instanceMethodNames: 'push: pop'
   classVariableNames: ''
   classMethodNames: ''
   keyDictionaries: #(append:->hierarchy insert:->hierarchy
         removeFront->hierarchy removeEnd->hierarchy)
   poolDictionaries: ''
   category: 'Kapsel-Examples'.
```

Figure 5: KAPSEL Stack Class

some of the classes of the KAPSEL system, and an example "Point" class definition.

5 Conclusion

Encapsulation is at the heart of useful re-use specification in object-oriented software. Encapsulation specification and enforcement is important, however, it needs to be as open and flexible as possible to enable a hardware-like "off-the-shelf" approach to software construction. Software is by its name "soft", i.e. malleable, rather than rigid. The encapsulation specification language introduced in this paper – KAPSEL – strives for a good balance between control and flexibility in setting up interfaces to re-usable software components.

References

[Booch, 1994] Booch, G. (1994). *Object-Oriented Analysis and Design with Applications, 2nd Edition.* Benjamin/Cummings.

[Ege, 1994] Ege, R. K. (1994). *Object-Oriented Programming with C++.* Academic Press, AP Professional.

[Ege and Stary, 1992] Ege, R. K. and Stary, C. (1992). Designing maintainable, reusable interface. *IEEE Software.*

[Ellis and Stroustrup, 1990] Ellis, M. A. and Stroustrup, B. (1990). *The Annotated C++ Reference Manual.* Addison Wesley, Reading, MA.

[Rumbaugh et al., 1991] James Rumbaugh et al. (1991). *Object-Oriented Modeling and Design.* Prentice Hall.

[Goldberg and Robson, 1983] Goldberg, A. and Robson, D. (1983). *Smalltalk-80: The Language and its Implementation.* Addison Wesley, Reading, Mass.

[Gosling and McGilton, 1996] Gosling, J. and McGilton, H. (1996). The Java language environment. *http://www.javasoft.com/doc-/language_environment/.*

[Group, 1991] Group, O. M. (1991). The common object request broker: Architecture and specification. OMG Document Number 91.12.1.

[Knight, 1992] Knight, A. (1992). Encapsulation and information hiding. *The Smalltalk Report.*

[Kolla, 1996] Kolla, S. (1996). Implementation of KAPSEL: an object-oriented language. *FIU Master's thesis.*

[Lieberherr, 1989] Lieberherr, K. J. (1989). Formulations and benefits of the law of demeter. *SIGPLAN Notices,* 24(3).

[Loomis, 1995] Loomis, M. E. (1995). *Object Databases - The Essentials.* Addison-Wesley.

```
Kapsel subclass: #Point
   instanceVariableNames: 'x y'
   instanceMethodNames: 'moveByX:Y:'
   classVariableNames: ''
   classMethodNames: ''
   keyDictionaries: #(x->general y->general
                      moveByX:Y:->(general write)
                      (x->write y->write) asCreatorKey )
   poolDictionaries: ''
   category: 'Kapsel-Examples'.
```

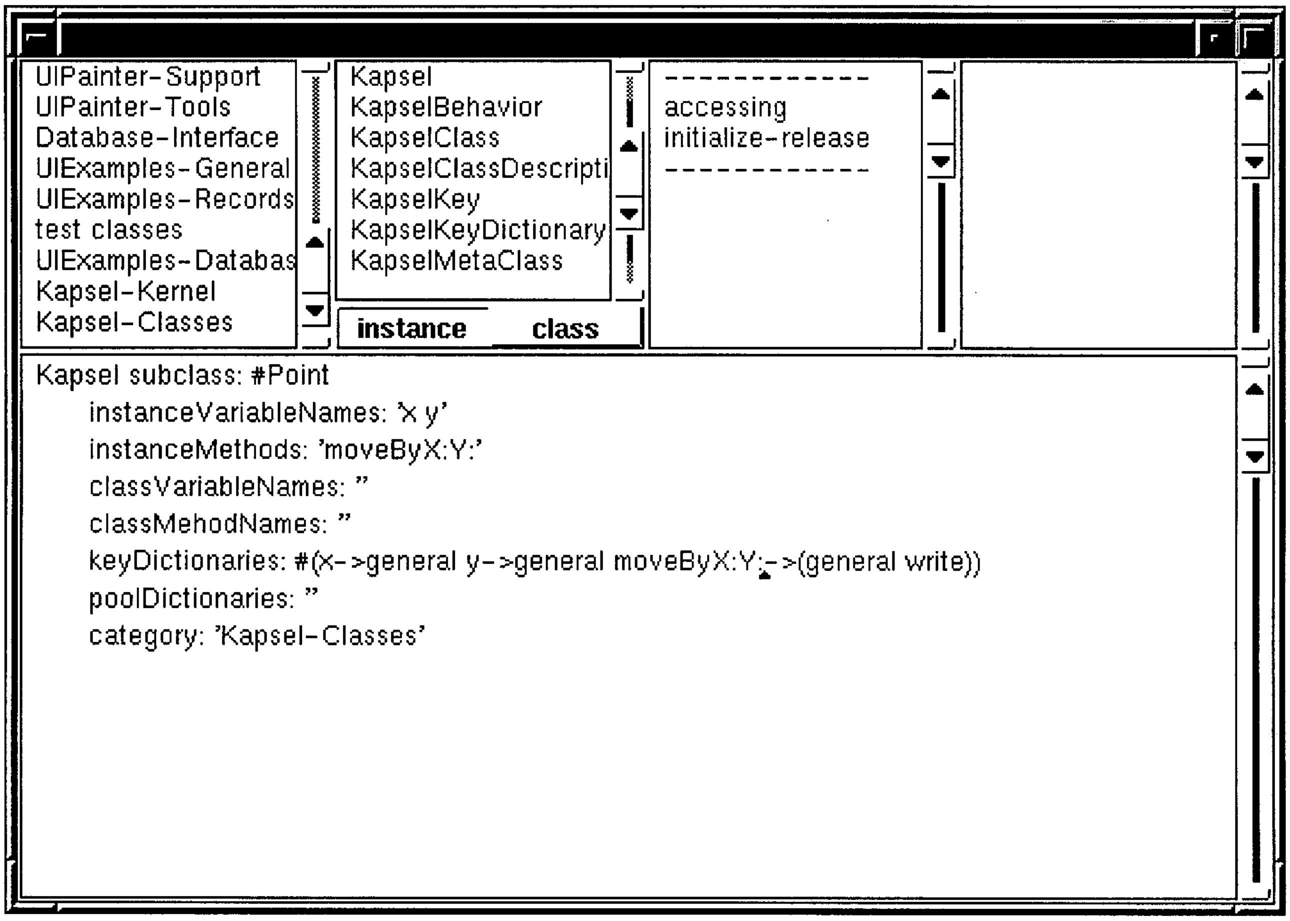
Figure 6: KAPSEL Point Class

[Parnas et al., 1985] Parnas, D. L., Clements, P. C., and Weiss, D. M. (1985). The modular structure of complex systems. *IEEE Transactions on Software Engineering*, SE-11(3).

[Shlaer and Mellor, 1992] Shlaer and Mellor (1992). *Object Lifecycles, Modeling the World in States.* Yourdon Press.

[Wiener, 1995] Wiener, R. (1995). *Software Development Using Eiffel.* Prentice Hall.

[Wirfs-Brock et al., 1990] Wirfs-Brock, R., Wilkerson, B., and Wiener, L. (1990). *Designing Object-Oriented Software.* Prentice Hall, Englewood Cliffs, NJ.



```
Kapsel subclass: #Point
    instanceVariableNames: 'x y'
    instanceMethods: 'moveByX:Y:'
    classVariableNames: ''
    classMehodNames: ''
    keyDictionaries: #(x->general y->general moveByX:Y:->(general write))
    poolDictionaries: ''
    category: 'Kapsel-Classes'
```

Figure 7: Smalltalk System Class Browser: KAPSEL classes

Holographic Neural Networks and Data Compression

Robert Manger
Department of Mathematics, University of Zagreb
Bijenička cesta 30, 10000 Zagreb, Croatia
Phone: (385) 1 4680 111, Fax: (385) 1 4680 335
manger@math.hr

Keywords: artificial neural networks, holographic neural technology, data compression

Edited by: Rudi Murn
Received: June 27, 1997 **Revised:** September 2, 1997 **Accepted:** September 16, 1997

This paper evaluates data compression capabilities of holographic neural networks. An outline of a possible "holographic" compression method is given. Experiments are described, where the proposed method has been simulated on a series of data files. The obtained results are presented and discussed. Conditions are identified, which are necessary for proper functioning of the method.

1 Introduction

Holographic networks [6] are an interesting and promising new type of artificial neural networks. Contrary to the usual "connectionist" approach [3], where networks are built of many simple processing elements, holographic networks consist of a small number of powerful components. Namely, a single holographic neuron is already able to learn stimulus-response associations, so that it is functionally equivalent to a whole conventional network. Another important characteristic of holographic networks is that they internally represent information by complex numbers. Magnitudes of those numbers are interpreted as confidence levels of data, and phase angles serve as actual values of data.

The memory of a holographic neuron is realized as a complex matrix, whose (k, l)-th entry in fact measures the influence of the k-th stimulus element on the l-th response element. During the training process, all associations are encoded and enfolded onto the same memory matrix. Thus training can be viewed as a form of *data compression*, where many stimulus-response pairs are squeezed together in a relatively small block of memory. Similarly, responding to the learned stimuli can be interpreted as data decompression, since the original stimulus-response pairs are again recreated (at least approximately).

Data compression has already been considered as a possible application of holographic neural networks [2]. There is an interesting example in [1], showing how a neuron with a 1024×1 memory matrix can learn 512 random associations and reproduce them with the mean response error 2% of full scale. In this example, stimulus length is 1024 and response length is 1. The author concludes that his neuron in fact compresses the given set of stimulus-response pairs to 1/128 of its original size.

This estimation, although correct, is in a way too optimistic, since it compares the memory matrix size to the size of the table with explicitly written stimulus-response pairs. But in many applications long stimuli do not need to be recorded at all, since they can be reproduced from much shorter (even implicit) data. Thus it would be more appropriate to compare the memory matrix with the list of responses (in the previous example the matrix is larger than the list). So the question remains: are holographic networks really as suitable for data compression as they look on the first sight?

The aim of this paper is to evaluate holographic neural networks as a prospective tool for data compression. The paper has no intention to define precisely all details of a "holographic" compression algorithm, but only to assess if such algorithm is feasible or not. Section 2 reviews the theory of holographic neural networks. Section 3 gives an outline of a hypothetical data compression method based on holographic training. Section 4 presents our experiments, where data files have been coded and reproduced by a holographic neuron. Section 5 summarizes the results of those experiments - compression rates and reproduction errors are listed. Section 6 gives a conclusion, i.e our answer to the initial question.

2 Holographic Neural Networks

As mentioned before, a single holographic neuron has already the functionality that is usually expected from a whole network, i.e. it can map stimuli to responses and it can learn associations. A *stimulus* to a neuron is a complex vector of the form

$$S = [\lambda_1 e^{i\theta_1}, \lambda_2 e^{i\theta_2}, \dots, \lambda_n e^{i\theta_n}],$$

and a *response* is also a complex vector of the form

$$R = [\gamma_1 e^{i\phi_1}, \gamma_2 e^{i\phi_2}, \dots, \gamma_m e^{i\phi_m}].$$

As we see, our holographic neuron internally works with complex numbers (written here in polar notation). It means that some form of data conversion is needed to switch between external (usually real or integer) and internal (complex) representation. This conversion should map actual data onto phase angles and their confidence levels onto magnitudes. The available software emulator [2] performs such data transformation automatically.

The neuron internally holds a complex $n \times m$ *memory matrix* X. Obviously, this matrix has enough entries to measure separately the influence of each stimulus element on each response element. But even more, a single matrix can be used to record a whole set of different associations, given by different stimulus-response pairs. Namely, the encoded associations can be superimposed one on another and stored into the matrix. After superposition, each single association can still be reproduced approximately. The described property of the memory matrix relies heavily on the fact that complex numbers are used instead of real numbers.

Learning one association between a stimulus S and a desired response R reduces to the following update of the memory matrix:

$$X = X + \bar{S}^\tau \left(R - \frac{1}{\sum_{k=1}^{n} \lambda_k} SX \right). \tag{1}$$

Here $\bar{S}^\tau$ denotes the conjugated transpose of the vector S. All occurences of X on the right-hand side refer to the old (previous) value of that matrix, and X on the left-hand side is the new (updated) value. Note that we deal here with a straightforward formula, which involves no iteration but only a fixed number of arithmetic operations.

To learn a set of stimulus-response associations, the basic step (1) must be repeated in turn for each association, so that all updates are accumulated in the same matrix X. Each subsequent encoding of a new association slightly disturbs previous encodings. Therefore the whole training sequence must be repeated several times in order to stabilize (i.e. several *epochs* are needed). The number of required iterations is almost always less than 20 [2]. Consequently, holographic training is usually faster than training in conventional neural networks.

Suppose that training has been completed. Then the response R^* to a given stimulus

$$S^* = [\lambda_1^* e^{i\theta_1^*}, \lambda_2^* e^{i\theta_2^*}, \dots, \lambda_n^* e^{i\theta_n^*}]$$

is computed as follows:

$$R^* = \frac{1}{\sum_{k=1}^{n} \lambda_k^*} S^* X. \tag{2}$$

A detailed mathematical analysis of the expression shown here can be found in [2] or [7]. The analysis demonstrates that R^* can be interpreted as a sum of many components - each component corresponds to one of the learned responses. If S^* happens to be equal to one of the learned stimuli S, then the corresponding learned response R occurs in R^* as a component with a great magnitude (confidence). At the same time, the remaining components have small magnitudes and different directions, so that they produce a relatively small disturbance. Consequently, R^* should have approximately the same direction (thus value) as R. The worst-case response error is expected to be

$$\phi_{\text{error}} \approx \frac{1}{\pi\sqrt{8}} \tan^{-1}(\sqrt{p/n}), \tag{3}$$

where p is the number of associations learned. This estimation is fairly realistic if our neuron has been trained with random (uniformly distributed) data.

From (3) we see that the learning capacity of our neuron is limited, and it depends on the length of the stimulus vector. For problems where the stimulus vector is short, the number of associations that can accurately be encoded is small. Then we need a special preprocessing operation called *stimulus expansion*, which artificially increases the stimulus vector length. There are several stimulus expansion procedures in use. The most popular one is called *higher order statistics* [7]; it supplements the (internally represented) original stimulus elements with their conjugates and products (order 2, 3, ... etc). Another expansion procedure [4] interprets the original stimulus elements as angles, and computes *sines and cosines* of their multiples.

The equation (2) shows that after training each of the computed response elements can depend on any of the given stimulus elements. The influence of the k-th stimulus element on the l-th response element is proportional to the (k,l)-th entry of the matrix X. If the (k,l)-th entry is relatively small, then the k-th stimulus element cannot significantly influence the l-th response element, and the entry can be "deactivated" (treated as zero). *Memory optimization* is a special operation where all matrix entries with magnitudes below a given threshold are deactivated. After optimization it is advisable to perform few more training epochs - the neuron will then "adapt" to the loss of some parts of its memory.

Note that the trained neuron in fact stores the learned stimulus-response associations in its memory matrix X. The size of this storage is $n \times m \times$ (number of bytes needed to represent a complex number). Stimulus expansion increases the storage size in order to reduce the response error. On the other hand, memory optimization reduces the storage (zeros do not need to be stored explicitly) on the expense of increasing the response error.

3 A Data Compression Method

Suppose that we are given a sequential file consisting of p (let say real) values $r_1, r_2, \dots, r_p$. We would like to design a method to store the file in the memory of a holographic neuron. Or, differently speaking, we would like to design a procedure to train the neuron with the values $r_1, r_2, \dots, r_p$, so that after training each of those values is available (at least approximately) as a response to a suitable stimulus. Let us now discuss how such a method could look like.

Since the values $r_1, r_2, \dots, r_p$ are generally independent, each of them must be used as the response in at least one training example. Obviously, one training example per value is enough. Also, for $j_1 \neq j_2$ the stimuli associated with r_{j_1} and r_{j_2} respectively must be different (otherwise the neuron would be exposed to contradictory data); thus the stimulus associated with r_j must be equivalent to the index j. For simplicity, we can assume that our original training set consists of the pairs (j, r_j), $j = 1, 2, \dots, p$. The used informal abbreviation (j, r_j) in fact denotes the pair consisting of the stimulus vector S and the response vector R, so that S is equal to the (internally represented) vector $[j]$ and R is equal to the (internally represented) vector $[r_j]$. As we see, both vectors have length 1, and the set consists of p training examples. After training, r_j can hopefully be reproduced by stimulating the neuron with j.

For the success of our file storage method it is crucial that the learned values r_j are reproduced with satisfactory precision. In order to assure this, we will have to introduce some form of stimulus expansion. Namely, the original stimulus is obviously too short, since it leads to the memory matrix consisting of only one complex number. The introduced procedure should expand the stimulus vector in each training example to a fixed length $n >> 1$, while keeping the response vector intact. In this way, the procedure should produce a transformed training set, again with p training examples. After transformation, the stimulus length will be the chosen n and the response length still $m = 1$. It means that the corresponding memory matrix will consist of n complex numbers. The length n has to be chosen big enough to ensure precise reproduction of any file with the given size p.

If we wish that the described storage method is also a compression method, we must require from the memory matrix to be smaller than the original file. This requirement can be met after training, by memory optimization. Namely, the optimization procedure will take into account the results of training, and it will discard all those memory elements which are not necessary for reproducing that particular file. Thus the minimum possible storage will be formed, which still assures the required precision in reproduction. Let us denote by $\bar{n}$ the number of active matrix elements after optimization.

Putting it all together, our hypothetical data compression method should be based on a combination of stimulus expansion, training, and memory optimization. The corresponding data flow diagram is given in Figure 1.

It is reasonable to assume that one complex number requires twice as much storage as one real number. Thus we can compute the *compression rate* (the ratio between the size of the compressed and the original file) as $2\bar{n}/p$, expressed as a percentage. This estimate is optimistic, since it does not take into account a small overhead necessary to record the positions of still active stimulus elements.

Let us now consider the computational complexity of our proposed compression method. If the file size p becomes bigger, the expanded stimulus size n should also increase in order to retain the same precision in response. In the worst case (random files), the response error behaves according to the equation (3) - i.e. the error stays the same only if n is kept proportional to p. Thus in the worst case one learning step (1) requires $\Omega(p)$ operations, and the whole training procedure (at least p learning steps) requires $\Omega(p^2)$ operations. This estimation means that our method is not able to code directly very big files - it would be computationally too expensive.

The only way how to deal with a big file is to divide it into smaller blocks. A practical compression method should use a fixed block size and it should produce the compressed file as the union of its compressed blocks. Each block should be compressed separately, according to the basic algorithm shown in Figure 1. The whole "block" method will then have linear computational complexity.

The question remains how to choose the block size for our block method. From the computational point of view, smaller blocks are better. But we must also take into account how the block size influences the overall compression rate. If our file consists of random numbers, then there is no influence; namely the equation (3) then indicates that for a fixed error level the compression rate (i.e. the ratio n/p for any block) is always the same. However, this reasoning is not valid if our file posseses some kind of regularity (redundancy) - namely the neuron can "learn" that regularity and use it more efficiently to reproduce a larger block. For instance, if our file is a tabulated linear function, then the neuron should learn and store only one value (i.e. the slope), which represents 10% of a 10-values block and only 1% of a 100-values block. Thus from the compression point of view it is safer to use larger blocks. Putting it all together, our method should use blocks that are small enough to assure reasonably fast computation, and still big enough to enable efficient compression.

Note that the proposed holographic compression

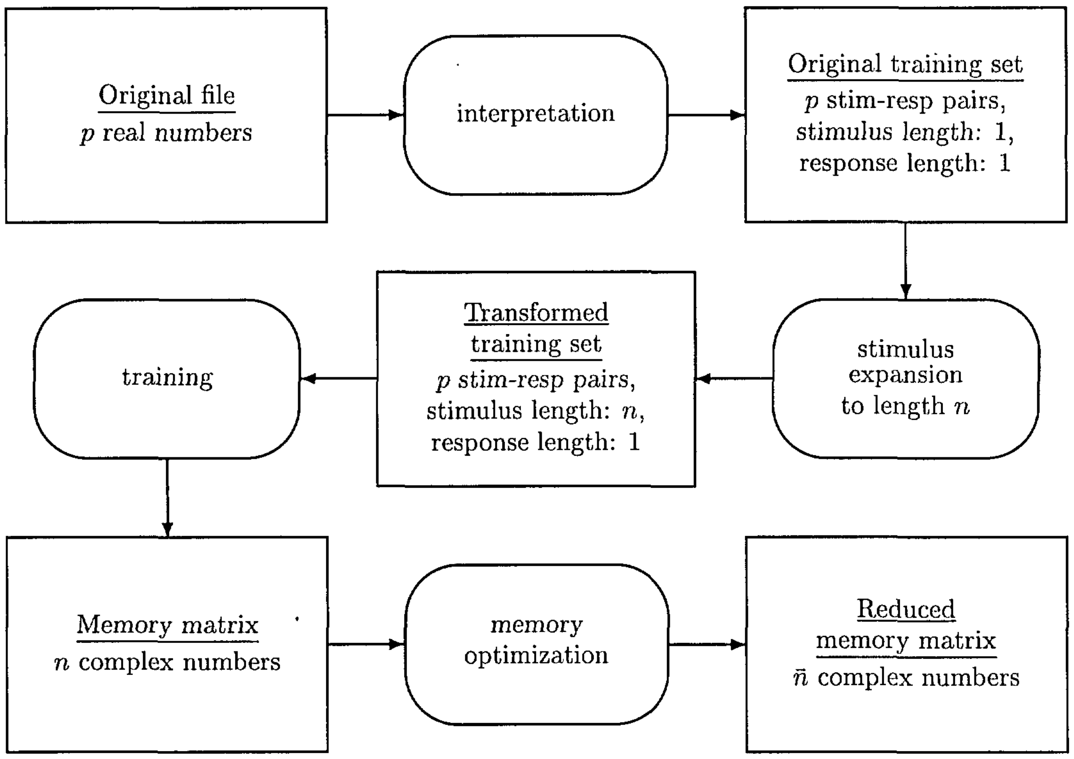


Figure 1: Outline of our data compression method.

method bears some resemblance to the widely used JPEG compression standard [8]. First, our method is also *lossy*, i.e. it only approximately reproduces original data. Also, both algorithms are *block* algorithms, i.e. they divide a big file into smaller (equally sized) blocks which are compressed independently. Finally, our method is also *parametrizable* in the sense that accuracy can be traded for better compression.

4 Experiments

In order to test data compression capabilities of holographic neurons, we have performed a series of experiments. Our experiments tried to mimic the proposed basic holographic compression method, i.e. they followed the outline shown in Figure 1. In each particular experiment, a different combination of data file and stimulus expansion procedure was chosen. After the initial training, the optimization phase was conducted very slowly. During the whole training and optimization process, the current compression rate and the current reproduction error were measured and recorded. In this way, the interdependence between compression efficiency and reproduction quality was explored.

We used altogether 15 different test files; all of them had the same size $p = 100$, and consisted of real numbers spanning the range between 0 and 1. These test files should be regarded as blocks in our hypothetical block compression method. According to our experience, the proposed block size 100 is already too large from the computational point of view - training lasts about a minute on a Pentium-based PC. Still, we worked with such big blocks in order to give more chance for successful compression.

Our 15 test files have been chosen in order to form three essentially different groups of 5. Members of the first group are very "regular" files, obtained by tabulating elementary functions. Files from the second group are still regular but more demanding; they have been constructed by repeating or combining the simple patterns used in the first group. The third group comprises extremely "irregular" files, i.e. sequences of random numbers (uniformly distributed between 0 and 1). One expects that regular files are easier to compress than more demanding ones. Also, truly random files should be incompressible by definition.

For each test file we tried two stimulus expansion procedures: the higher-order statistics and the sine-cosine procedure. This gives altogether 30 experiments. The expanded stimulus length was always $n = 200$, which is sufficient for exact reproduction of most files of size $p = 100$. Note that the initial holographic memory matrix (before optimization) is 4 times larger than the original file. Optimization should reduce this matrix size as much as possible, so that the final (still acceptable) matrix is hopefully smaller than the original file.

To measure the current reproduction error, we com-

file name	file description	stimulus expansion procedure	optimal compression rate mean error ≤ 0.0001	mean error ≤ 0.0010	mean error ≤ 0.0100
linear	$r_j = \frac{j}{100}$, $j = 1, 2, \ldots, 100$	hi.ord.stat	2%	2%	2%
		sine-cosine	364%	228%	52%
step	$r_j = 0$ for $j = 1, 2, \ldots, 50$, $r_j = 1$ for $j = 51, 52, \ldots, 100$	hi.ord.stat	8%	8%	4%
		sine-cosine	12%	12%	8%
poly-nomial	$r_j = \frac{j(j-100)}{2500} + 1$, $j = 1, 2, \ldots, 100$	hi.ord.stat	> 400%	> 400%	32%
		sine-cosine	> 400%	> 400%	108%
sine	$r_j = \frac{1}{2} \sin \frac{2\pi j}{101} + \frac{1}{2}$, $j = 1, 2, \ldots, 100$	hi.ord.stat	228%	200%	36%
		sine-cosine	2%	2%	2%
sine+ cosine	$r_j = \frac{\sqrt{2}}{4}(\sin \frac{2\pi j}{101} + \cos \frac{2\pi j}{101}) + \frac{1}{2}$, $j = 1, 2, \ldots, 100$	hi.ord.stat	244%	174%	26%
		sine-cosine	292%	202%	44%

Table 1: Summary results - experiments with regular files.

puted the *mean absolute response error*, i.e. we computed the mean absolute difference between original and reproduced file values. Since our files consisted of values between 0 and 1, the chosen error measure can also be interpreted as "relative to full range". For instance, error 0.0100 simply means that the expected difference between an original value and its corresponding reproduced value is 1% of full range.

The current compression rate was computed as $2\tilde{n}/p$, expressed as a percentage, where $\tilde{n}$ denotes the current number of active matrix elements. The rationale for this estimate has already been given in the previous section.

Let us now explain in more detail how the two chosen stimulus expansion procedures work in our particular case, where the original stimulus associated with the file value r_j is simply its index j.

- The higher-order-statistics procedure uses the (internally represented) index j and its conjugate to form products of 1, 2, 3, 4, ... etc. factors. The expanded stimulus is simply a list of unique products of that kind (each one of them has either a different number of factors or a different proportion of conjugated factors). As many products are generated as necessary to complete the vector of n elements.
- The sine-cosine procedure interprets the original stimulus j as an angle between 0 and 2π, i.e. instead of j it uses $\alpha_j = 2\pi j/(p+1)$. Then the procedure computes $\sin(k\alpha_j)$ and $\cos(k\alpha_j)$ for $k = 1, 2, \ldots, n/4$. These $n/2$ numbers (internally represented), together with their conjugates, form the expanded vector of n elements.

The two expansion procedures can lead to completely different results. Namely, after expansion the training process will try to correlate r_j with the elements of the expanded stimulus vector. In the first case r_j will be compared to something analogous to powers of j, and in the second case to something similar to sines and cosines of multiples of j. The different nature of the two expansion procedures can be illustrated by analogy with classical numerical approximation methods. More precisely, the higher-order-statistics expansion is analogous to the Lagrange interpolation, where a function is approximated by a polynomial, while the sine-cosine expansion is analogous to the Fourier analysis, where a function is expressed as a combination of trigonometric functions [5].

5 Results

The results of our experiments have been summarized in Tables 1, 2, and 3. Table 1 describes the behaviour of regular test files (first group), Table 2 the behaviour

file name	file description	stimulus expansion procedure	optimal compression rate mean error ≤ 0.0001	mean error ≤ 0.0010	mean error ≤ 0.0100
multiple linear	$r_j = \frac{j}{100}$ for $1 \leq j \leq 25$, $r_j = \frac{2(j-25)}{100}$ for $26 \leq j \leq 50$, $r_j = \frac{3(j-50)}{100}$ for $51 \leq j \leq 75$, $r_j = \frac{4(j-75)}{100}$ for $76 \leq j \leq 100$	hi.ord.stat	> 400%	> 400%	148%
		sine-cosine	> 400%	> 400%	210%
multiple step	$r_j = 0$ for $1 \leq j \leq 50$ or $76 \leq j \leq 88$ or $95 \leq j \leq 100$, $r_j = 1$ for $51 \leq j \leq 75$ or $89 \leq j \leq 94$	hi.ord.stat	312%	292%	150%
		sine-cosine	386%	318%	142%
amplified sine	$r_j = c_1 \cdot j \cdot sin\frac{4\pi j}{101} + c_2$, $c_1 = 0.006563739$, $c_2 = 0.5824075$, $j = 1, 2, \ldots, 100$	hi.ord.stat	> 400%	> 400%	34%
		sine-cosine	> 400%	> 400%	118%
accelerated sine	$r_j = \frac{1}{2}\sin 4\pi(\frac{j}{101})^2 + \frac{1}{2}$, $j = 1, 2, \ldots, 100$	hi.ord.stat	254%	246%	92%
		sine-cosine	386%	366%	168%
modulated sine	$r_j = \left|\sin\frac{4\pi j}{101}\right|$, $j = 1, 2, \ldots, 100$	hi.ord.stat	> 400%	> 400%	62%
		sine-cosine	> 400%	> 400%	16%

Table 2: Summary results - experiments with more demanding files.

of more demanding files (second group), and Table 3 the behaviour of random files (third group). Each row corresponds to one experiment - the best possible compression rate for each of the three chosen error thresholds is reported. In some cases the optimal compression rate is greater than 400%, which means that the initial stimulus was still not long enough to assure the required precision.

As we see from Table 1, our holographic method can compress regular files, but only with a modest precision in reproduction. If we can tolerate the mean response error 1% of full range, then we can achieve compression rates between 20 and 50%. Some files can even be compressed to 2% of its original size and reproduced with no error at all (row 1 and 8 of Table 1). But these examples are rather exceptional, namely they are based on a situation where the file value (response) happens to be proportional to one of the expanded stimulus elements. In this rare occasion the optimization procedure quickly isolates the right stimulus element and deactivates all the others. However, for a slightly more complicated situation (row 5 and 10 of Table 1) where the response is proportional to the sum of two stimulus elements, the results are not so spectacular - namely optimization ends up with much more than two elements. The reason for this is internal (complex) representation of data within a holographic neuron, and the fact that arithmetic operations and data conversion do not commute. For instance, if we add x and y externally and then represent this sum internally, we would not obtain the same value as if we first convert x and y and then add them internally.

According to the results in Table 2, our holographic method is only occasionally successful in compressing more demanding files. To a human eye, the plotted versions of the given files would still look regular in some sense. However, the holographic neuron is usually not able to recognize such subtle regularity. It is very hard to determine which of the demanding files are "regular enough" to be compressed and which are not.

Table 3 shows that our holographic method was not successful in compressing the given random files. In each experiment and for any error threshold, the optimized neuron memory was bigger than the original file. This is contrary to our expectations: namely a good method should preserve the original size of an incompressible file. If a high precision in reproduction was required (error 0.0001 - column 4 of Table 3) then the stored file was 2 to 4 times bigger than the original. For a low precision (error 0.0100 - column 6 of Table 3) the stored file was considerably smaller, but it still had the size at least 1.6 of the original. It is important to note that the results obtained for different random files are quite similar. Therefore we

file name	file description	stimulus expansion procedure	optimal compression rate mean error ≤ 0.0001	mean error ≤ 0.0010	mean error ≤ 0.0100
random-1	first 100 values from a pseudo-random sequence	high.ord.stat	316%	238%	172%
		sine-cosine	338%	332%	192%
random-2	101-st to 200-th value from the same sequence	high.ord.stat	264%	250%	184%
		sine-cosine	330%	330%	190%
random-3	201-st to 300-th value from the same sequence	high.ord.stat	318%	242%	174%
		sine-cosine	236%	236%	188%
random-4	301-st to 400-th value from the same sequence	high.ord.stat	248%	226%	184%
		sine-cosine	338%	338%	192%
random-5	401-st to 500-th value from the same sequence	high.ord.stat	> 400%	242%	164%
		sine-cosine	> 400%	304%	184%

Table 3: Summary results - experiments with random files.

believe that the observed failure of our method on the given files is not accidental. It is very likely that the method would have the same problems with any irregular (non-smooth, non-redundant) file.

Some of the obtained results are illustrated in more detail by Figures 2 and 3. Both figures correspond to the same experiment described in Table 2, row 5. The chosen experiment uses the relatively complicated but still compressible "amplified sine" file, and applies the higher-order-statistics procedure for stimulus expansion. Figure 2 shows precisely how the mean reproduction error depends on the compression rate. Figure 3 compares the original file values with the reproduced values that are obtained after optimal compression (rate 34%).

Tables 1, 2 and 3 can also be used for evaluation of the two stimulus expansion procedures. Namely, we can compare the results of any two experiments that use the same file, and we can observe which procedure leads to a better compression rate within the same error threshold. Not surprisingly, the results obtained by two expansion procedures are never the same. As we can see from rows 7-8 of Table 1 or rows 3-4 and 9-10 of Table 2, the sine-cosine procedure can be better. But more often, the higher-order-statistics procedure is better.

It is an open question whether some new stimulus expansion procedure can be constructed, which would be generally better than higher order statistics. Obviously, each procedure is suitable for a certain limited class of files where file values happen to be in correlation with only few expanded stimulus elements. The problem is to find a procedure whose class covers as many files as possible.

6 Conclusion

In this paper we have considered the feasibility of a general data compression method based on holographic networks. The considered method interprets a given file as a set of stimulus-response associations. Each association corresponds to a particular value from the file, so that the stimulus identifies that value and the response contains the value itself. Through the process of training, the set of associations is stored (compressed) in the memory of a holographic neuron, and can later on be reproduced.

The described method includes stimulus expansion and memory optimization as subroutines. Stimulus expansion is necessary to assure accuracy in reproducing the file. Memory optimization takes into account special properties of the file (discovered through training) and assures optimal compression. Two stimulus expansion routines have been considered: the one based on higher order statistics, and the one that employs sines and cosines.

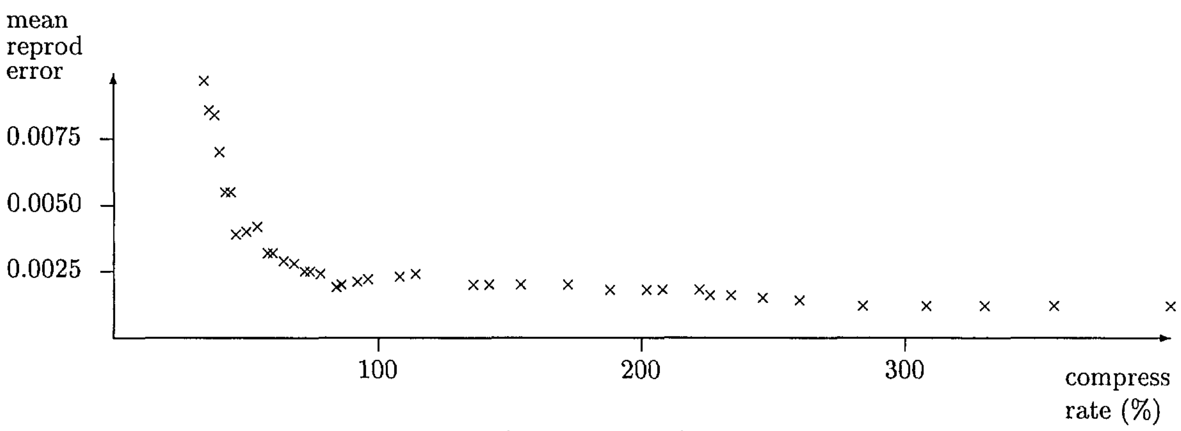


Figure 2: Detailed results for a chosen experiment.

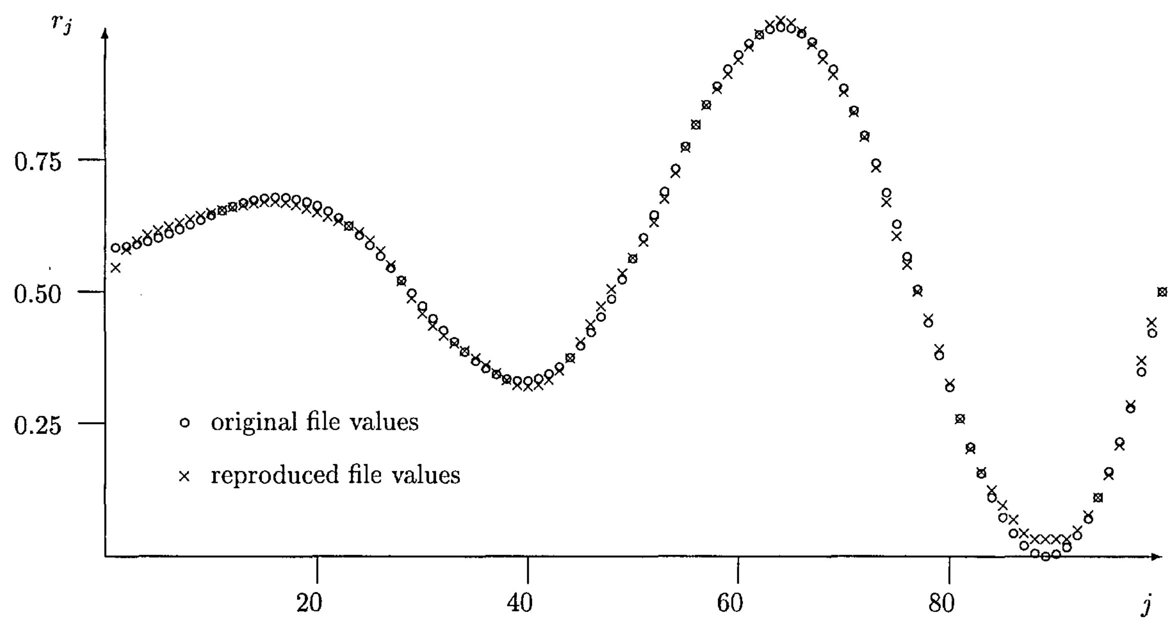


Figure 3: More details for the same experiment and a chosen compression rate.

Our experiments clearly indicate that the considered "holographic" compression method works well only if the following two conditions are met.

1. The file to be compressed is in some sense very "regular" (e.g. smooth, redundant, repeatable).

2. The required precision in reproducing the file is not too high (e.g. the reproduction error about 1% of full range can be tolerated).

Under these circumstances, we can expect moderate compression rates about 20-50%. Among the two stimulus expansion routines, higher order statistics seem to be more successful in general, although sines and cosines can be better in some special cases.

Our experiments also show that the considered method is not reliable enough if we deal with slightly more complicated files (e.g. oscilations, discontinuities). The method sometimes succeeds and sometimes fails for no obvious reason.

Finally, our experiments demonstrate that the considered method does not work satisfactorily with random (highly irregular and nonredundant) files. Namely, the stored version of a random file always turns out to be bigger than the original file, so that the whole effort degenerates into data "expansion".

References

[1] AND Corporation (1990), *HNeT Neural Development System - Version 1.0*, Hamilton, Ontario.

[2] AND America Limited (1993), *HNeT Discovery Package - Version 1.3 for Windows*, Oakville, Ontario.

[3] Hecht-Nielsen R. (1990), *Neurocomputing*, Addison-Wesley, Reading, Massachusetts.

[4] Manger R., Plantamura V.L. and Souček B. (1994), "Stimulus preprocessing for holographic neural networks", in *Frontier Decision Support Concepts*, ed. by V.L. Plantamura, B. Souček and G. Visaggio, John Wiley and Sons, New York, pp. 79-90.

[5] Press W.H. et al. (1988), *Numerical Recipes in C*, Cambridge University Press, Cambridge.

[6] Sutherland J.G. (1990), "Holographic model of memory, learning and expression", *International Journal of Neural Systems*, Vol. 1, No. 3, pp. 256-267.

[7] Sutherland J.G. (1994), "The holographic cell: a quantum perspective", in *Frontier Decision Support Concepts*, ed. by V.L. Plantamura, B. Souček and G. Visaggio, John Wiley and Sons, New York, pp. 65-78.

[8] Wallace G.K. (1991), "The JPEG still picture compression standard", *Communications of the ACM*, Vol. 34, No. 4, pp. 31-44.

Logic as the Language of Innate Order of Consciousness

Jeremy Horne
3128 West Reunion Drive
Phoenix, AZ 85027, USA
jhorne1@cris.com

Keywords: consciousness, information, logic, operators, order, Piaget

Edited by: Anton P. Železnikar
Received: April 10, 1997 **Revised:** July 28, 1997 **Accepted:** November 12, 1997

Basic logic and math textbooks usually present a standards for prioritizing operators in parenthesis-free expressions. The authors say that it is a scheme dictated by convention, but not clear is a description of circumstances under which one would ever find ungrouped expressions and need to prioritize operators. This paper argues that the circumstances generating the expressions dictate the prioritization. That is, if logic and mathematics at a basic level represent the real world, then ordering the operators would have a "natural" foundation. Logic and mathematics reflect a structure of human consciousness, as suggested by Jean Piaget. The operators, themselves, have degrees of intellectual complexity and an empirical foundation with major philosophical import.

Section One describes logical aggregation and its importance. Section Two briefly examines three examples suggesting that the ease of logical thinking depends upon ordering of operators. Cases found in human learning theory and Boolean neural networks suggest that each operator has a unique level of complexity. In Section Three, the author proposes a method for finding a natural order of operators that more closely fits the way in which humans think, and Section Four advances a procedure to analyze seemingly unordered phenomena. Section Five describes the philosophy upon which this proposed research is predicated. Binary logic's syntax displays semantics of order in human consciousness, as biophysical and cosmological research indicate. The syntax, itself, may be semantic expressed by a deeper structure linked to that of the cosmos, itself. Section Six suggests a direction in which research should proceed to understand how the source of our being might be communicating to us.

1 Introduction

Have you ever wondered about the reason for the convention that prioritizes or aggregates logical operators in parenthesis-free expressions and what the consequences would be if the order were different and had an empirical foundation? Questioning such an apparently mundane ground rule can lead to an upheaval of the way people have been thinking about a system. What if you learned that the logic and these operators had more significance than representing the structure of arguments, that they might represent the structure of the cosmos, itself? In this essay, I argue that the complexity of relations between/among entities should determine operator prioritization and that this complexity is the essence of the binary logic as the language of innate order in the universe.

Our logic has assumed paramount importance as the foundation of modern computer science and much of artificial intelligence. Binary logic usually is the first and most common logic students encounter, but they rarely encounter meaning beyond it being a mechanical convenience for analyzing mathematical relationships and attempting to analyze ordinary language arguments. Yet, exciting mathematical, neurophysiological, and psychological work recently done in binary logic suggests a connection between our consciousness and the cosmos.

This essay is conjectural in some parts and cross-disciplinary. It does not purport to be a deep analysis of competing ideas, but I wanted to tie together some crucial observations to create enough of a focal point for questioning present conventions, re-directing pedagogy, and proffering groundwork for a philosophy of binary logic and consciousness.

The second section describes logical aggregation and its importance. Section Three briefly examines three examples llustrating that the ease of logical thinking depends upon ordering of operators. Cases found in human learning theory and Boolean neural networks suggest that each operator has a unique level of complexity. In Section Four, I propose a method for finding a natural order of operators that more closely fits the way in which humans think, and Section Five advances a procedure to analyze a seemingly unordered phenomenon. The sixth section describes the philosophy upon which this proposed research scheme is predicated. Binary logic's syntax displays a semantics of order in the universe, as biophysical and cosmological data indicate. The syntax, itself, may be a semantics expressed by a deeper structure. Section Seven suggests a direction in which research should proceed to understand how the source of our being may be communicating to us.

2 Prioritization and Its Importance

In parenthesis-free expressions, such as $p \supset q \wedge r$, the truth value of $(p \supset q) \wedge r$ is different than $p \supset (q \wedge r)$. Using commonly accepted notation and by convention, the priority of operators is equivalence ($\equiv$), implication ($\supset$), or ($\vee$), and ($\wedge$), and negation ($\neg$) in descending order of scope, or precedence. That is, $\neg$ affects only the adjacent variable, $\wedge$ affects only the variables on either side, $\vee$ affects the $\wedge$ expression inside the parentheses and the first variable outside, and so forth. So, $p \equiv q \supset r \vee s \wedge \neg t$ would be grouped (parenthesized) $p \equiv (q \supset (r \vee (s \wedge \neg t)))$, and $p \vee \neg q \equiv r \supset s$ would be $(p \vee \neg q) \equiv (r \supset s)$, with $\equiv$ affecting every variable and $\neg$ affecting only one (e.g.: Stoll 1961, p. 60; Copi 1979, p. 219; Massey 1970, p. 34–62; Rosser 1978, p. 19–23).

The same occurs with arithmetic operators, as $9 + 5 \times 4 + 3$ would be $(9 + (5 \times 4)) + 3$. It is generally recognized that the prioritization of these relational operators in logic is patterned after mathematical ordering, conjunction being analogous to multiplication, disjunction resembling addition, and so forth. Which operator has a greater scope than another is determined merely by convention (Church 1956 p. 79–80; Exner 1959, p. 38–40; Rosser 1978 p. 19; Margaris 1967, p. 26; Copi 1979, p. 219).

As to the values that the variables may assume, there are 16 relationships generated from the four ways (00, 01,10, 11) the two elements in a basic linear order may be permuted. Each of these relationships may be seen as a way we describe how we know that the first element is related to the second. For example, the value of p as 0011 (the "0" traditionally regarded as a "false", and 1 as "true") can be related to the q value of 0101 as 0111 (0 or 0 = 0, 0 or 1 = 1, etc.), because the relationship is "or". In standard truth table form:

p	q	p or q
0	0	0
0	1	1
1	0	1
1	1	1

Prioritizing enters as a problem in parsing a sequence of variables connected by operators in a parenthesis-free expression. There are two cases. In the first case, the components of the idea are already known and determine the grouping. The second case is significant in this paper, where groupings of ideas are not known, but we group according to the convention described above. Logic texts, such those by Copi and Rosser will discuss the convention, but the reader if left wondering about how the problem of ungrouped expressions arises in the first place.

The convention serves convenient purposes, not the least of which is to preserve consistency in logical computation. For mathematics, it is easier from a visual perspective (because of the more closely spaced x and y) to multiply x times y in $xy+p$, rather than separate the x from the y and add y to p. Even while maintaining the visual preference for doing the xy calculation first, the xy just as easily could have meant "x plus y" and the plus "x times y", the order now being addition first, and multiplication second. Signs have not always meant the same thing throughout the ages, nor have they always been used. In the Bakhshali (Indian) system, the "+" meant negative. In European mathematics, the plus and minus appeared at the end of the 15th century (Cajori 1993, p. 77).

Aggregation did not assume much importance until the end of the 15th century, when Pacioli in his Summa found a need to compute roots in polynomial expressions (Cajori 1993, p. 385). Subsequent treatments of roots depended upon appropriate aggregation techniques. However, notation usually arose after the concepts were formulated and had the purpose of punctuation, or separating the symbols that represented concepts. Roman numerals ultimately were replaced by the Arabic system we see today. Long division by Roman numerals demonstrates the superiority of the present method.

A literature search has failed to yield a convincing philosophical rationale for the current prioritization convention. Quite to the contrary, as indicated above, standard logic texts refer to the ordering as a convention. Notations represent concepts, and the question at hand is why one operation should precede another. What we are looking for, of course, is an ordering of operators based on concepts rather than simple arbitrary agreement. Truth value or computational results obviously depend upon how operators are prioritized, but the importance of prioritization is greater than obtaining consistent calculation. Logical operators do not have the same degree of complexity, and the operators may be hierarchically ordered according to what constitutes complexity, as the following three briefly discussed examples indicate. The essence of the complexity may carry through to the result of the computation.

3 Consequences of Different Aggregation

Piaget wrote that the building block of a child's ideas of movement and speed is an awareness of serial order. A simple order is linear and "requires only a simple perceptual situation" (Piaget, Child's Conception 1971, p. 36). Through adulthood, people's cognitive abilities depend upon apprehension of operational complexity. Piaget and Inhelder demonstrated that children learn in logical stages, i.e., " ... memory is a function of operational developments" (Piaget and Inhelder 1973, p. 160). For example, a randomly selected five year old child will do conjunctive operations before disjunctive ones when given a task depending upon the conservation of length. The meaning of the addition ('or') function is apprehended more easily than the multiplication ('and') one, suggesting that each operator represents a level of intellectual complexity that affects the ability to memorize. More recent investigations confirm the same type of phenomenon in adults (Taylor 1987). While Piaget's and Inhelder's research methodology may be slighted, the general thrust of a decade or more of their work points to differences in operational complexity.

Research indicates that learning in Boolean neural nets depends upon the arrangement of operators. A network is designed to search for its own structure to solve a problem by accepting data and attempting to discover the rule governing the relationships among items in the data. The rule must be recognized with the least number of errors. Operators, as logic gates, are serial with no feedback or back propagation. Researchers seek to discover the ordering of operators based upon ascending energy levels required for a successful discovery to occur. Energy is defined as "the discrepancy between the correct result of the operation and the one obtained from the circuit averaged over the number of examples N_E shown to the system (which are chosen randomly in the beginning and kept fixed over the annealing) ... " (Patarnello 1989, p. 120). The difference between the correct result and the result obtained by the system yields the energy level required by that gate. Only when the result is zero can the correct gating be identified by the system. It is apparent that if the network is configured differently with the ordering of operations changed, the energy levels will differ as well. While the methods and schemas for Boolean neural network computation are quite complex, the results indicate that there is an optimum configuration of operators the net uses to learn a task.

Patarnello's work on performance energy of neural nets corresponding to configurations of input bits as a way of ordering operators is supported by Martland's work classifying network behavior corresponding to truth table elements. The complexity of operators in Boolean neural nets has not been studied extensively, but it seems that the density of truth outputs varies with the type of connective involved, thus suggesting a basis for classification (Martland 1989, p. 222-234). Kauffman has shown that when specific functions are forward fed into an element (propositional schema) specific patterns of function orders emerge. There are attraction points found within autonomous random Boolean network (BN) state-space (Kauffman 1993, Chapter 5). With respect to ordering in Boolean complexity, it can be demonstrated that numerous random couplings of operations result in patterns resembling those of cellular automatons (Wuensche 1993). If logical operators are randomly coupled to produce patterns, it would be interesting to see what patterns emerge if the operators were coupled according to an empirically determined scheme.

4 Discovering a Natural Aggregation

A method exists for determining how the concepts of operations are prioritized according to the way we think. Piaget and Inhelder built a foundation to show that operators have differing degrees of intellectual complexity. While it is beyond the scope of this paper to explain the details, suffice it to say that an approach exists to constrain the operator to a specific learning parameter (color, size, speed, and so forth) and make their experiments totally phenomenological, rather than having the testing rely upon words. One form of such an experiment I have created builds upon a Miller's Analogies-style of presenting information but pictorially represents the meaning of each operator, with the subject being asked to identify the meaning. Complexity is registered as a function of rapidity and accuracy of response. Each parameter may result in a different order or priority. For example, recognizing an operational concept (such as implication) weights may be more difficult than with sizes. The priority given the operator would depend upon the time taken to recognize the operation and the accuracy of recognition. Several orderings may emerge from these experiments. For an artificially intelligent device, a task involving so many parameters may have that many parameters may have that number of processors operating in parallel, each one configured to an ordering.

5 Applying Empirically Derived Ordering

How can one use an empirically derived ordering? The significance of ordering may be seen by comparing the computational outcomes of several different orderings of a bit stream. For example, take a sequence like $0 \vee 0 \supset 0 \wedge 1 \vee 0$. Normally, this would be $(0 \vee 0) \supset ((0 \vee 1) \wedge 0))$, the final result being 1. If the new grouping were $((0 \vee (0 \supset 0)) \vee 1) \wedge 0$, the result would be 0. Without knowing how the ideas are structured, what are we to do? What if the source meant: $(0 \vee 0) \supset (0 \vee (1 \wedge 0))$? For ordinary language, the problem is somewhat simplified when the person starts the sentence with "if", for we often know this is the main operator. However, what do we do with the second part? Does the source want us to use $\wedge$ or $\vee$ as the main operator? This does not mean that there is a universal way of parsing ordinary language utterances expressed in a parenthesis-free manner, but it does illustrate a problem of parsing a bit stream from what could be an intelligent source. As mentioned above, logic texts often raise the issue of unaggregated expressions but do not say when, where, or why we would encounter them. However, one can think of the binary digitizing of any phenomenon (seemingly non-repeating decimals, like $\sqrt{2}$ and π), cellular automata, electroencephalograms, or even data from the esoteric former Search for Extraterrestrial Life (SETI) project.

How could an empirically derived prioritization of operators be used to derive meaning from an ungrouped bit stream? The following example procedure is vastly oversimplified, possibly flawed technically, quite abstract, and merely suggests a general direction in which research might proceed in observing emerging order from such an ungrouped bit stream.

Taking 0s and 1s to represent truth values of statements (as opposed to operators):

A) Identify an experimentally-derived prioritization scheme, such as $\vee, \equiv, \supset$ (using the commonly accepted notation), where $\vee$ has the greatest scope of operation, $\equiv$ the next, and $\supset$ the least. (Other analyses may use different operators and orderings.) The present example might be the ordering found for persons doing logical operations involving sizes. That is, persons might find that it is easier to do logical operations involving size with $\equiv$ than with $\vee$ and $\supset$.

B) Divide the bit stream into $n + 1$ bit segments, where n is the number of operators in the prioritization scheme. With three operators, the segments would be four bits long. For 0110000111001100 ... , this would look like 0110–0001–1100–1100 ... .

C) Aggregate each segment according to the prioritization scheme by inserting the operators in between the bits. In the above example, in the first segment of 0110, the grouping would be $(0 \equiv 1) \vee (1 \supset 0)$. The next would be $(0 \equiv 0) \vee (0 \supset 1)$, and so forth.

D) Evaluate the segment. Evaluate $(1 \supset 0)$ first, to get 0. Next is $(0 \equiv 1)$, with 0 also being the result. Finally, $0 \vee 0$ is 0.

E) Do the rest of the four bit units the same way, the next being 0001, the aggregation being $(0 \equiv 0) \vee (0 \supset 1)$, and the value being 1.

F) Concatenate the results, 010 up to a length to be determined experimentally.

G) "Stack" the lengths on top of each other, such as:

```
010111011110001001
110001101010110001
110011000110001110
010100110001110000...
```

H) Observe the patterns and compare to those generated by cellular automata, or electroencephalograms, as suggested by Wuensche. (Wuensche 1993, p.11) These patterns, however, would be generated from logical operations based upon an empirically derived ordering of operators.

6 Philosophy of Aggregation

Why would a natural ordering scheme tell us more about how we think? Why should this research tell us about innate order? Part of the answer lies in rethinking how we view logic. Wuensche, Kauffman and others have demonstrated that patterns emerge from variously randomly coupled operators, each operator seeming to have a different degree of complexity. Two arguments may be advanced for regularity in the random coupling, each resting upon a view of causality. Either the patterns originate from something within the entity, itself, or something from outside the entity imparts that which is needed for the entity to generate the pattern. Modern philosophers refer to the first view as autopoiesis, or the theory of self-organization. How do systems self-organize? What "propels" organization? Artificial life and automata are two types of apparent self-organization that have current interest. Adherents of the second view of causality would argue that an external agent is responsible for the entity; entities don't arrange themselves. Patterns don't simply "happen". The first assumes independence, the latter interconnectedness. Not denying the first for now, I will focus on making a case for the second, which, in turn may help clarify the discussion for autonomy and the nature of the complexity.

According to Piaget,

There exist outline structures which are precursors of logical structures, ... It is not inconceivable that a general theory of structures will ... be worked out, which will permit the comparative analysis of structures characterizing the outline structures to the logical structures characteristic of the higher stages of development. The use of the logical calculus in the description of neural networks on the one hand, and in cybernetic models on the other, shows that such a programme is not out of the question (Piaget 1958, p. 48).

Other researchers hold that propositional logic reflects an order innate in the universe and human thinking. The arrangement in the universe is according to a "pregeometry as the calculus of propositions", such that " ... a machinery for the combination of yes-no or true-false elements does not have to be invented. It already exists" (Misner et al 1973, p. 1209). Everything is reducible literally to the *primordial*—first ordering. The binary structure may be a very natural expression of the way the universe exists in a fundamental and profound way. That is, the logic is a discovery more than a creation.

Our universe began from a singularity, or an undifferentiated phenomenon. Hesiod in the *Theogeny* and Lucretius in *The Nature of Things*, spoke of a chaos, or unordered condition prior to the beginning of the universe. It may be compared to Peirce's state of doubt, a feeling of undifferentiated or uniform energy. Bound up with the singularity was process; potential changed to kinetic, manifested by movement. Out of this "condensed chaos" came what we have in our dimension.

We see this emergence of being into our dimension today at both the infinitesimal and the infinite ends of the spectrum of our discernible world. At the infinitesimal end of existence, it has been found that there is a pressure exerted by elements within a space deemed to be a vacuum. In this space of zero point energy are particle density fluctuations as photons enter and exit this discernible vacuum space with no known reason (*Science*, "The Subtle Pull ... ", 1997, p. 58). At the infinite end of the spectrum, Stephen Hawking's latest research indicates that microscopic black holes ... eat one kind of particle and emit another (*Science*, "Visions ... ", 1997, p. 476). While particles entering the event horizon may be flattened with their primordial constituents scattered over the boundary layer, and ultimately the lost information may be ejected from the black hole, this does not say anything of the force inside the black hole (Susskind 1997, p. 52).

The dialectic between the discernible (what we see) and the unformed (bound up with the singularity) chaos and its other was the first and most basic of processes. Out of this process came and still does come that which exists in terms of what is not apparent, or what we do know in terms of what we don't. Order was born as the object of this dialectic process, allowing us to discern existence through existents (our world around us through the things in it). This order is expressed by the language of logic.

What is the nature of this binary logic? Minimally required for order is a set of two elements, and each operator establishes a relationship between these two elements. In binary logic, the two elements normally are semantically regarded as false and true (often symbolized by 0 and 1, respectively). What we say is true or false depends upon our knowledge. Logic displays a structure of existents that comprises a natural semantics, a structure standing as an ontology of knowing. (An example of such an ontological system may be found in Feibleman's *Ontology*.) A thing must exist in order for us to know it, knowing being a way of accounting for an assertion. Setting aside criterion of how we determine whether something is true or not, the primary existents for what I call an epistemological logic are two: that which is known, or measured, and that which not known. This is not the same as equating the unknown with false. Actually, false would fall into the category of known, for one knows in order to state something to be false. Likewise, true also is in the category of known. For the other existent, unknown, something can be true but unknown and exist, such as the actual number of stars in the universe. (While asserting a truth presumes its existence, existence of something does not mean that one has knowledge of its nature. Hence, my use of the binary logic is not as a truth-functional calculus of propositions (the former), but as a structure displaying epistemological relationships (the latter).

Symbolically, we can say 0 represents the unknown, (undifferentiated, etc.) and 1 represents the known; 0 is prior and 1 subsequent. The unknown becomes the known. Another way of expressing this is that 0 *contains* 1, for the universe of the unknown is larger than and precedes the known. Containment is the subject of deductive logic.

As a quantum semantics, 0 means a wave, and 1 the collapse of the wave function, or unity, where the observer apprehends the particle density fluctuation as information bounded by space-time. As a maximal expression of information, this would be regarded as an entropy, that which has emerged from chaos, or energy that has been expended (dissipated) to reveal information. Once the information has been expressed, it isn't expressed again from the same source in that space-time. In particle physics, the one is an absolute unit, a dimensionless number representing the speed of light, Planck's constant divided by 2π, and the gravitational constant (Hameroff 1996 p. 520). Wave function collapse apparently is a constituent of our consciousness, i.e., those 0s and 1s may very well physically represent our thought processes.

This wave function collapse to the value one is seen in the cytoskeleton, or microtubules, of neurons. Tubulin subunits make up the microtubule and are dimers (a bipolar entity that can assume either a positive or negative state), and these act as binary computational structures (Rasmussen et al. 1990, p. 428–449). When polarization occurs in the gigahertz range (10^9 to 10^{11} Hz) (Frohlich 1975, p. 1412) among groups of these dimers, the neuron assumes a shape that seems to modulate the neural pulse (Hameroff 1992, p. 517–518). However, the phenomenon seems to have cosmological correlates.

About 100 (10^{11} Hz) to 1000 GHz most clearly shows the uniformity of cosmic background radiation (CBR) at 2.73 K ($\pm.01$ with a 95% degree of confidence) (Smoot, 5), the same as black body radiation and about the same value as the natural logarithm e (2.718). Frohlich's upper boundary of 10^{11} is the lower boundary of CBR, or the unit measure of 1 mm. More than being simply a "true" in a semantics table, 1 very well may signify a resonance with CBR and what gave rise to it. As Penrose said: " ... there should vibrational effects within active cells which would resonate with microwave electromagnetic radiation, at 10^{11} Hz, as a result of biological quantum coherence phenomenon (Penrose 1994, p. 352). If the 10^{11} frequency is what "activates" consciousness, more support is given the view that the universe is, itself, conscious. What are the mechanics of the language and, more importantly, meaning?

Sixteen operators with four sets of relationships between placeholders for two entities (p and q) spatio-temporally relate the unknown to the known and the wave function (symbolized as 0) to its collapse (symbolized as 1). The conditional, so often the focal point of "paradoxes of material implication", consistently and faithfully describes the spatio-temporal nature of deduction, or the structure and processes of closed systems. One should note that the often used proper subset symbol ($p \supset q$, with $p \neq q$) for the traditional material implication "horseshoe" is incorrect, since it is the improper subset symbol, $\supseteq$ (horseshoe with bar underneath), denoting deduction, that says the first set can *contain* the second either totally or partially. With this correct use, there is intuitive, as well as logical sense of material implication. An element contains itself ($0 \supseteq 0, 1 \supseteq 1$). In popular logical parlance, the relationship is true or 1. Obviously $0 \supseteq 1$ is the case, and that leaves $1 \supseteq 0$ being false, or 0, hence completing the truth table for $\supseteq$. What is known consists of a smaller universe than the universe of the unknown. Quantitatively, the universe of the unknown contains what is known. (Or, in Kant's view, the appearance, or instance, is bounded by the reality of the whole.)

7 Where Do We Go From Here?

To bring the philosophical speculation and theoretical system constructs into the domain of being tangible, it would be useful to empirically demonstrate that the links exist between the 10^{11} frequency, the binary logic, and consciousness. While the technology may not be available now, the following offers one possible route of exploration for such a test.

Technology is such that simultaneous Positron Emission Tomography (PET), functional Magnetic Resonance Imaging (fMRI), and electroencephalogram (EEG) measurements can be taken (and mapped onto each other) of an individual doing a mental task, such as learning the meaning of a logical operator (Science "New Dynamic Duo" 1997, p. 1423). That is, the EEG can measure various structures exhibiting mental activity. With each logical operator, there should be two frequency ranges: that of the neural pulse (1–40 Hz) and the high GHz frequency that causes the microtubule to assume the shape that modulates the wave to produce the EEG matching the brain activity associated with processing a particular logical function. A confirmation that this approach has merit would be to re-introduce the measured electrical signals back into the brain structures to induce the subject to perform the mental task and possibly to report other thoughts that may be embedded in that code. While on the surface it may be that only the thought of an operator would emerge, there may be associated thoughts grabbed from other areas of the brain to create a more complete idea of what thoughts are associated with the random bit stream. For example, see Newman for how stimulating one area of the brain induces activity in other areas (Newman 1993, p. 267, 270–271). In principle, the object would be to correlate the EEG with the logical operator or series of logical operations done by the subject. Would it be farfetched to suggest that an extended truth table of 0s and 1s might pictorialize the EEG wave form or that the 0s and 1s could be mapped to the EEG?

A similar approach of correlating 0 and 1 patterns to EEGs may exist for the random concatenation of operations done by Kauffman and Wuensche. Wuensche suggests that his basins of attraction diagrams resulting from a random concatenation of logical operations may even indicate an " ... electroencephalogram (EEG) measure of the mean excitatory states of a path of neurons in the brain" (Wuensche 1993, p. 11).

A third area of investigation would be to correlate the patterns of conformational collapse on the surface of the microtubule to EEGs and the patterns exhibited in the work by Wuensche. If the display of the conformational collapse does correspond to the 1 in binary operations, and the 1 is symbolic of quantum collapse, then, this might bring us closer to showing that binary logic is the language of at least one form of consciousness.

8 Summary

I have presented the issue of aggregating logical operators in parenthesis-free expressions and discussed the importance of finding a method based on how we think. Three studies suggest that it is the complexity of the operators that determines the priority of operations in a parenthesis-free expression. If a string of 0s and 1s is generated by the complexity represented by operators, it would not be unreasonable to analyze such a bit stream using a prioritization that more closely resembles human thinking. Discovering such a natural order or orders is predicated upon a philosophy that is being borne out by emerging research in biophysics and cosmology. It gives new reason to an old logic.

Our logical thought processes, as expressed by 16 operators, are ordered according to a type of intellectual complexity. These processes are mappable to brain structures, and the frequency against which cosmic background radiation is measured drives these brain structures, thus making logic as a language of innate order in the universe. Consciousness as we know it is immanent in the universe.

References

[1] ______ 1996. New dynamic duo: PET, MRI, joined for the first time. Science 272:1423.

[2] ______ 1997. The subtle pull of emptiness. Science 275:158.

[3] ______ 1997. Visions of black holes. Science 275:476.

[4] CAJORI, F. 1993. A History of Mathematical Notations. Reprinted from the Open Court edition, 1928 and 1929. Dover Publications. New York.

[5] CHURCH, A. 1956. Introduction to Mathematical Logic. Part I. Princeton University Press. Princeton.

[6] COPI, I. 1979. Symbolic Logic. Macmillan PC. New York.

[7] EXNER, R.E. & M.F. ROSSKOPF. 1959. Logic in Elementary Mathematics. McGraw-Hill. New York.

[8] FEIBLEMAN, J.K. 1951. Ontology. The Johns Hopkins Press. Baltimore.

[9] FROHLICH, H. 1975. The extraordinary dielectric properties of biological materials and the action of enzymes. Proceedings of the National Academy of Sciences 72:4211–4215.

[10] HAMEROFF, S.R., et al. 1992. Conformational automata in the cytoskeleton. Computer 25:30–39.

[11] HAMEROFF, S.R. & R. PENROSE. 1996. Orchestrated reduction of quantum coherence in brain microtubules: A model for consciousness. Toward a Science of Consciousness. The First Tucson Discussions and Debates. *Qn* S.R. Hammeroff et al., Eds. 507–540. MIT Press. Cambridge, MA.

[12] KANT, I. 1963. Critique of Pure Reason. Trans. Norman Kemp Smith. McMillan, London.

[13] KAUFFMAN, S. 1993. The Origins of Order. Oxford University Press. New York.

[14] LUCRETIUS. 1959. The Nature of the Universe. Trans. R.E. Latham. Penguin Books. Baltimore, MD.

[15] MARGARIS, A. 1967. First Order Mathematical Logic. Blaisdell PC. Waltham, Massachusetts.

[16] MARTLAND, D. 1989. Dynamic behaviour of Boolean networks. Neural Computing Architectures. MIT Press. Cambridge.

[17] MASSEY, G.W. 1970. Understanding Symbolic Logic. Harper & Row. New York.

[18] MISNER, C.W., K.S. THORNE & J.A. WHEELER. 1973. Gravitation. W.H. Freeman. New York.

[19] NEWMAN, J. & B.J. BAARS. 1993 A neural attentional model for access to consciousness: A global workspace perspective. Concepts in Neuroscience 2:255–290.

[20] PATARNELLO, S. & P. CARNEVALI. 1989. Learning capabilities in Boolean networks. Neural Computing Architectures. MIT Press. Cambridge.

[21] PIAGET, J. 1971. The Child's Conception of Movement and Speed. Walden Edition. Ballantine Books. New York.

[22] PIAGET, J. 1958. Logic and Psychology. Basic Books. New York.

[23] PIAGET, J. & B. INHELDER. 1973. Memory and Intelligence. Routledge & Kegan Paul. London.

[24] RASMUSSEN, S. et al. 1990. Computational connectionism within neurons: A model of cytoskeletal automata subserving neural networks. Physica D 42:428–449. North-Holland.

[25] ROSSER, J.B. 1978. Logic for Mathematicians. 2nd. Ed. Chelsea PC. New York.

[26] SMOOT, G.F. 1995, 31 May. The cosmic background radiation. Lawrence Berkeley Laboratory and Department of Physics. University of California. Berkeley. Paper accepted for publication in Planetary and Space Sciences.

[27] STOLL, R.R. 1961. Sets, Logic, and Axiomatic Theories. W.H. Freeman. San Francisco.

[28] TAYLOR, B.W. 1987. An investigation of cognitive level and performance of total teaching duties of public school teachers. Psychological Reports 60:55–58.

[29] WUENSCHE, A. 1993. The ghost in the machine: Basins of attraction of random Boolean networks. Cognitive Science Research Papers CSRP 281. University of Sussex at Brighton.

Consciousness and Process

Andrej Ule
Philosophical Department,
Faculty of Arts,
University of Ljubljana,
Aškerčeva 2, 1000 Ljubljana, Slovenia

Keywords: consciousness, processes, characteristic qualities, modality, potentiality, Wittgenstein, patterns, sensitivity, information

Edited by: Rudi Murn
Received: April 24, 1997 **Revised:** November 3, 1997 **Accepted:** November 12, 1997

I wonder if is it possible to combine the view that consciousness is a flow of mental states with the view that it is not a process. I believe this combination is possible and necessary for a correct understanding and explanation of consciousness. I distinguish between a process (information-processing) and a non-process (at least not time-processed) side (aspect) of consciousness: The first consists of the neural information-process, the flow of sensations and the change of feelings. The second lies in the conception and representation of processing potentials of states of affairs, events and processes. The unity of process and non-process sides of consciousness lies in the awareness of time as an imaginary process that is the basis of all other processes.

1 Wittgenstein's challenge: consciousness is not an inner process.

The idea of consciousness as a kind of "inner process", a flow of "inner states" of an organism is very common in everyday's thinking, in science and philosophy. We often speak of the "flow of ideas", "flow of emotions", etc. We find this idea very often in philosophy, in psychology, in cognitive science, etc. James characterized consciousness as the continuous process of attention and intention (volition). He introduced the metaphor of the "stream of consciousness" (James 1950). Later, the idea of consciousness as an information process in the brain became the leading idea in the cognitive science.

The essence of this idea is: consciousness is a chain of inner states of a cognitive system of which the cognitive system is aware. This change is a permanent change of those inner states of the cognitive system which have the quality of awareness. Understanding this process depends on special philosophical standpoints.

For dualists, mental states are intrinsically inner states of each person, i. e., they are not penetrable by other people. The process of consciousness of a person is inaccessible to other people and is self-evident to the given person. For dualists, these facts indicate that inner states and the stream of consciousness are essentially immaterial. For materialist philosophers, mental states are certain states of neural structures in the brain. The process of consciousness is a neural process in the brain.

Díaz recently presented a semi-phenomenological theory of the stream of consciousness. He assumes a "neutral monist ontology according to which conscious phenomena are both brain- and mental- states of a peculiar information class; namely, patterned processes that call for meaningful correlations but not for mutual reduction" (Díaz 1996, p. 714). This is a middle standpoint between dualism and materialism.

If someone told me "Wait for me by the bank", and I asked: "Did you, as you were saying the word, mean this bank?" then I was referring to the time of speaking, not to an experience at that time. The question would be meaningless if it referred to a hidden state of meaning within us (Wittgenstein, 1976, p. 216f). The same would be true if someone asks me: "Did you intend to say such-and-such to him on your way to meet him?". The question refers to a definite time of walking but not to the experience (of intending) during that time (ibid.).[1]

[1] In connectionism the "inner states" of a neural net structure are diffuse and mainly implicit; the inner processes are holistic and synergetic (Rumelhart, McClelland, 1986). We cannot identify everyday mental states (i.e., states of believing, thinking, experiencing something, etc.) with some definite real inner states of neural nets in the brain. There is a difference between phenomenological inner states on the mental surface and their implementation in neural nets. Some proponents of connectionism deny the usual conceptions of mental phenomena, especially intentional mental states. They are part of a "radically false theory" (Churchland, 1981, p. 67). In the classical computational theory of mind we have a relatively closer connection of the manifest mental phenomena and their implementation in the

Wittgenstein drew the following conclusions from this apparently purely verbal reasoning:

"The intention with which one acts does not 'accompany' the action any more than the thought 'accompanies' speech. Thought and intention are neither 'articulated' nor 'non-articulated'; to be compared neither with a single note which sounds during the acting or speaking, nor with a tune."(ibid.)....

"Meaning is not a process which accompanies a word. For no process could have the consequences of meaning. (Similarly, I think, it could be said: a calculation is not an experiment, for no experiment could have the peculiar consequences of a multiplication)." (ibid., p. 218).

Thinking, intending, meaning something are not real states or processes, i. e., some states and processes that have some given duration in time. They do not "happen in the mind" (or in the brain).

We have to be cautious not to overemphasize Wittgenstein's criticism of some mental states and mental processes. Wittgenstein did not deny mental states or mental processes in general. He accepted emotional states and emotional processes, perceptual states and processes, pain states, etc. (Wittgenstein, ibid, p. 59, footnote), but he denied real processes of "higher mental acts": thinking, intending, believing, meaning something. Wittgenstein criticized also the idea of some "inner mechanisms in the brain" because this idea is subject to the so-called grammatical mistake of speaking about inner states and inner processes. Instead of that, he proposed to understand higher mental acts as some necessary "grammatical" parts of human life-form, namely of the life-form where language is the essential constituent.

"An intention is embedded in its situation, in human customs and institutions. If the technique of the game of chess did not exist, I could not intend to play a game of chess. In so far as I do intend the construction of a sentence in advance, that is made possible by the fact that I can speak the language in question." (Wittgenstein, 1976, §337).

Wittgenstein's criticism of the idea of private mental states and of mental processes is Wittgenstein's challenge to modern philosophy of mind, and to cognitive science, both of which depend especially on the idea of consciousness as a "mental process" (as the flow of "mental states"). To speak of inner-, mental-, brain states or processes leads us to some logico-grammatical paradoxes. This mode of speech indicates that a large portion of conceptual naVveté is bound up with the idea of mental states and mental processes.

I wonder if is it possible to combine the view that consciousness is a flow of mental states with the view that it is not a process. I believe this combination is possible and necessary for a correct understanding and explanation of consciousness. Consciousness has two aspects, one of which is process and the other is non-process. It is not only one of them. Here, I try to give only some first insights into this idea.

representational strcture of brain.

2 General characteristics of processes

It is very hard to give a clear notion of processes. I will give only an informal description of some properties of processes that are relevant from my point of view. Processes are usually conceived as some sequences of changes of states of a system (e.g. a system of some properties and relations of material objects, a system of mental phenomena) with a given pattern of changes. The first important property of processes is that at least some qualities (traits) of the states of a system that appear in processes change in time.Stationary states are events. We can in principle conceive of stationary states as elementary processes, too. Processes are then conceived of as ontologically fundamental (like Whitehead (1929)). Qualities of states that give rise recurrently and consistently to the pattern of change, are characteristic qualities of a process. The most general referential process of changes is the time-process that gives measure to all other real processes.[2]

The second important property of most real processes is the continuity of change of some characteristic qualities of the states. Each state could be presented by a valued state of affairs in a given moment of time. It could be formally described by some properties of the state (characteristic properties of the state). Material states are determined by the characteristic properties and relations of some material objects, mental states by some characteristic qualities of mental phenomena in a given time-interval. Sometimes time-intervals can be reduced to time-points (we get instant-states). Characteristic qualities and their changes in sciences are often given by some bounded and differentiable mathematical functions (of time and other variables). In the social sciences and humanities, we need some more descriptive and qualitative methods of finding the characteristic qualities of processes. In each case, we have to distinguish a process itself from its cognition by humans.

Commonly-encountered real processes are very complex. They consist of many simpler real processes and corresponding simpler real states. The measure of complexity for states and processes depends on the context, on the properties of the system of states and its connection with larger systems of states. However, concerning the state of a given system, we can speak

[2]A similar idea can be found in Aristotle's Physics (Book 4, 219B), where he defines time as the "number of motion in respect of 'before' and 'after'." Aristotle considered motion to be any process.

of simple states and simple processes. Simple states are determined by some finitely-many, well-determined characteristics or characteristic qualities of the system to which the states belong. They could be the properties or relations of some material things, fields, space-time points, etc. or the qualities of some psychological phenomena of a person at a given time.

The conceptual necessity of a referent system of states in conceiving a process is not only a mental, theoretical or grammatical means of the knowledge of processes, but also depicts a real dependency of the process on a larger context of events or states that give some identity or quality to a process. A process is not only a pure succession of changes of events, but has its own holistic character that differentiates the process from other real or possible processes of a similar kind. In the material world, the causal nets or lawlike connections among the successive states or successive phases of a process give the necessary holistic character to a physical process. We cannot find similar determinations of mental processes, because the appropriate causal links and laws of nature are not known (or not known yet) to us. However, we "feel" the individual qualities of different mental processes in us (e.g., the process of pains, a change of emotions) and their differences. I will not discuss here the question of what are the causes or grounds for this kind of feeling. I will consider the modal properties of the process that come out of its holistic determination or appearance.

Each process gives different possibilities for changes of states (of a system) according to the properties of the states and the nature of the process. Some processes are deterministic. The possibilities for changes of states depend only on some initial conditions of the original states. The process of rapping between billiard balls is described by a mechanical system of states. Each state and change to another state could be completely described by the relative positions, masses, changes in relative position, i.e., motions, velocities, etc. of all the balls that make up a system of billiard balls in a given moment of time. The causal character of this process could be given by the initial state of the system (by the initial impulses and initial velocities of all the billiard-balls). Here, the possibilities of a state depend on the initial conditions of the system. The same state (and the same change of state, hence "state-change") could be the result of different initial conditions. The states that follow a given state (or the state-change that follow a given state-change) are, at least in principle, completely determined by the given state (resp. state-change). Many physical processes are not completely fixed by their initial conditions. Each state depends not only on the previous state, but also on some undetermined additional conditions in the given time-moment or is unconditioned to some degree. For example, the microphysical states and processes in quantum mechanics are unconditioned. However, also in this case the possibilities for a state-change are given by the nature of the process. The possibilities for a state-change themselves change by process and through time. Each new state that appeared in a process bears some new set of possibilities (giving rise to new probabilities) for the change into another state of the system. I call this set of possibilities potential for change of a state. It depends generally on the characteristics of the given state, the initial conditions of a process, the characteristics of the whole process, the characteristics of the whole system of states, etc.[3]

The change of potentials of states is not a real process because the potentials are not real states of a higher system of states. They are only sets of possibilities of changes for different states. They form the unity of a process, and give its holistic character. We would enter an infinite regress of processes if we stated that each real process owns a higher real process. Sure, we can represent the pseudo-flow of possibilities as a higher formal process, but this is only a formal device for describing the change of possibility and not a reality.[4]

Information systems in nature and some artificial information systems (e.g, the neural nets) could be characterized by saying that such systems that are sensitive to other processes. The ability to get information from the environment (or from an input device) is first the ability to react to some significant processes, not to some significant states or simple changes of states. They react to some spatial and temporal pattern of signals, not to singular signals or the momentary inputs of signals to the system. My thesis is: the evolution of life proceeds from simple systems that react only to a narrow set of simple changes, to ever more complex systems that are sensitive to different complex processes. They became sensitive to the changes of potentials of states, not only to the changes themselves. Animals, especially animals with higher developed brains are "aware" of possibilities of movements

[3]Richard Feynman expressed this well in his discussion of the forces acting between two protons, when he said that these forces depend on as many parameters as possible. (Feynman Lectures, Vol. 2, lecture 37).

[4]Modern science, especially quantum mechanics, developed a new view of processes as intrinsically modal phenomena. Microprocesses could be interpreted as "interwoven" into a field of potentiality which is formally described by the complex function (of states. Schrödinger's equation is the central law of the temporal evolution of the quantum states. However, in the classical Copenhagen interpretation of the evolution of quantum states, they are only "mathematical devices" for calculating physical processes, thus the reference of the so called superposition of quantum states is purely formal, not ontological. There are some other, more ontologically oriented interpretations of quantum mechanics where the concept of quantum potential as a kind of information potential of microprocesses is emphasized (Bohm, Hiley, 1993). It is a very interesting question wheter the notion of the superposition of possibilities (of a quantum state) in quantum mechanics might be useful in the general theory of the modal characteristics of processes, too.

of their victims or their natural enemies, not only of their actual movements. This is necessary for survival in the dangerous and dynamic world. A lion, for example, is not attentive to the actual movement of a gazelle but tries to control its attempts to escape. This means the lion has to be sensitive to the possible movements of the gazelle, not only to the actual movements of the gazelle. Sure, the lion has no idea of possibility, he is not aware of possible movements, but it has somehow to be sensitive to it.

Humans and perhaps also some other highly developed animals (apes, dolphins, wales) also have the ability for conscious sensitivity to potential dimensions of some processes in the environment (and in the organisms themselves). They can really be aware of processes and their inner flows of possibilities. Humans possess a symbolic system (language) for representing processes and their potentials. Thus, we can think about them. We create ideas of possibilities, make expectations, plans, hypotheses. We can articulate our fears. This activity is frequently a conscious mental activity. Human consciousness is the highest achievement of the natural evolution of the sensitivity of some systems to processes.

3 Is consciousness a process?

It seems that consciousness is a process in a system that can represent other processes, especially their pattern of potential changes. Consciousness can represent some alternative to a given process, not only a fixed process. If the flow of mental states was only a kind of a process, then, by our reasoning, it would be a chain of changes of a certain system of mental states. There would also be a pseudo-process of changes of mental state(potentials that is internally "linked" to the process of consciousness. We could conceive of consciousness as a "system" which is incredibly sensitive to the "potential aspects" of the events in the world. We can comprehend this sensitivity as a similarity between the inner possibilities of change of mental states and the possibilities in the processes that a conscious being is aware of. A "conscious brain" must be enormously flexible to acquire similarity between its own inner processes and different outer processes simultaneously. Some PDP models and neural-net models of the brain could help us in modelling that kind of flexibility.

However, even the human ability of consciousness for depicting (some) potential aspects of processes has its non-process aspect. The ability of conscious beings to perceive or be aware of some potential aspects of processes is more than any process, more than any state (of brain or elsewhere). This ability of consciousness is not only sensitivity to potential aspects of events respectively to the information aspects of the events, but it also represents those possibilities to the brain and to the human being. The conscious representation of some possibilities to the brain, or better, to the human being, has some internally-logical aspects which are neither real states nor processes. They are no real parts of mental states or mental processes either.

It is not enough to find some appropriate "symbols" for this representation; one has to put them in an appropriate logical frame, where certain possibilia actually "appear" as possibilities of states-changes, and certain information as contents. This surpasses the powers of any state, and of any process. We became really aware of possibilia and information, not being merely sensible to them, only with the help of a complex language-system that knows different forms of logic frames of events. At this point, I see a deep meaning of Witggenstein's theses that language games are necessary to the human mind and consciousness, and the reason why he denied the process nature of some mental acts.

I conclude that consciousness is two-sided: process and non-process. The process (aspect) of consciousness lies in the neural information-processing, in the flow of sensations and the change of feelings. The non-process aspect of consciousness lies in the conceiving and representing of processes, especially in representing the potentials for change of states of affairs and events and in representing the holistic traits of processes. These "traits" lie over the real processes. We can get to them only through abstraction from states, from processes.

My last thesis is: the unity of process and non-process sides of consciousness lies in the awareness of the time. It is consistent with some theses in phenomenology on the time-consciousness (Husserl, 1928). We conceive time as an imaginary process that underlied all other processes. The "feeling" of pure succession of time-moments is not a real process. It is only the time-dimension for representating processes in consciousness. Our consciousness of events or processes is necessarily consciousness of some events (processes) in time. It includes necessarily a modal frame of identification and diferentiation of processes. A conscious presentation of processes refers explicitely or implicitely to some future processes which can happen. We refer also to some possible processes which could happen instead of those which actually happens or which have happend. Consciousness of time is thus the indication of our ability to be aware of processes in a modal sense, that means to be aware of their possibilia and to be aware of this ability itself.

References

[1] Aristotle, 1936. Physics, Sir D. Ross (ed.), Clarendon Press, Oxford:

[2] Bohm, D. & Hiley, B.J. 1993. The Undivided Universe. An Ontological Interpretation of Quantum theory. Routledge, London, New York

[3] Díaz, J.-L. 1996. The stream revisited: a process model of phenomenological consciousness. In S. R. Hameroff, A. W. Kasniak, A. C. Scott, Toward a Science of Consciousness. The First Tucson Discussions and Debates. MIT, Cambridge/MA

[4] Feynmann, R.P. 1964. The Feynmann Lectures in Physics, Vol. 2, Addison Wesley, Redwood City:

[5] Fodor, J. 1984. Psychosemantics. MIT Press, Cambridge/MA

[6] Hilmy, S. S.: Wittgenstein and behaviourism. In: Grazer Phil. Studien, vol. 33-34, 1989

[7] Husserl, E. 1928: Vorlesungen zur Phänomenologie des inneren Zeitbewußtseins (ed. by M. Heidegger), Niemayer Verl., Halle

[8] James, W. 1950. Principles of Psychology I, II. Dover, New York

[9] Ross, D. 1971. Aristotle. Methuen and Co., London

[10] Rumelhart, D. & McClelland, J., Eds. 1986. Parallel Distributed Processing: Exploration in the Microstructure of Cognition. MIT Press, Cambridge/MA

[11] Whitehead, A. N. 1929. Process and Reality. The Macmillan Co., Cambridge

[12] Wittgenstein, L. 1976. Philosophical Investigations. B. Blackwell, Oxford

Call for Conference Proposals (deadline January 30, 1998)

for the first international meta-conference on "New INFORMATION SOCIETY" (New IS Conference)

Ljubljana, Slovenia, Europe, October 21-23 (tentative) 1998

The meta-conference will consist of several stand-alone conferences. The working language is English.

The meta-conference has three main goals:

1. to contribute and promote knowledge about new information age and IT
2. to present interdisciplinary information technologies and sciences that are of substantial importance for further progress
3. to analyse and demonstrate essential changes facing politicians, regulators, the public, industry and society in the forthcoming years.

Conference proposals and a draft CFP should be sent to matjaz.gams@ijs.si. Deadline for submissions is January 30, 1998. Notification of acceptance will be emailed by February 28, 1998.

A typical paper will fall into the category of overview scientific papers, thus combining high scientific quality and presenting state-of-the art in a specific subfield. Classical technical papers are acceptable, yet not the most desired. Papers describing relevant applications are welcome.

The papers will be published in proceedings. Selected papers will be published in journals and in books. Each conference can publish papers on its own or in a joined way.

Each conference is entitled to one invited lecturer with covered expences for travel and housing. Invited lecturers can be proposed also independently of conference proposals by anyone. Invited speakers should be relevant scientists and experts in the information age or IT field. For these proposals, send tentative title, list of publications and other data to the above address.

A conference proposal should include the following sections:

- A brief technical description of the specific technical issues that the conference will address.
- The reasons why the conference is of interest this time.
- The names, postal address, phone and fax numbers and email addresses of the conference Organising Committee, which should consist of at least three people knowledgeable in the field but not all at the same institution.
- The name of one member of the conference Organising Committee who is designated the contact person.
- A list of previously-organized related events by any of the conference Organising Committee member.
- If possible, a list of potential attendees in case the proposal of the conference being accepted.
- A schedule for organising the conference and a preliminary agenda. Guidelines:
  - Call for papers - ready by end of February 1998
  - Distribution of the CFP - March 1998
  - Paper submission deadline - May 25, 1998
  - Notification to Authors - June 25, 1998
  - Camera-ready papers - July 25, 1998
  - Conference - October 21-23
  - Later publication in a book or in a special issue of a journal.
- A summary of the intended conference Call for Papers/Participation, showing how the organisers will encourage discussion.

Call for Papers

Working Group 8.3 of the International Federation for Information Processing invites you to participate in its 1998 working conference on

CONTEXT-SENSITIVE DECISION SUPPORT SYSTEMS

LOCATION: Bled, Slovenia

DATES: 13-15 July 1998 (opening reception on Sunday, July 12th)

ORGANIZED BY: IFIP Working Group 8.3 on Decision Support Systems; The University of Maribor, Faculty of Organizational Sciences and The Slovene Society Informatika

The focus of this conference is on issues related to developing context-sensitive decision support systems (DSS). There are a number of contexts that need to be taken into account (e.g., cultural, organizational, task-, role- or individual-related), depending on the purpose of the DSS and its target user(s). These contexts interact with and influence each other and an appreciation of their importance, in their totality, in design decisions can give rise to DSS which are adaptable to different environments and circumstances.

The adaptability of a decision support system should be considered along two different dimensions. On one hand, we have the horizontal dimension which considers changes though time within a particular context. That is, organizations and their practices change their requirements for decision support and technology continuously progresses to the effect that designers are given a much wider range of technical possibilities for providing decision support in a manner that could be more effective. Moreover, specific DSS face the problem where data, information and knowledge are continuously evolving. On the other hand, we have the vertical dimension of adaptability which relates to the transferability of the DSS to different (cultural or organizational) contexts. DSS, designed on the basis of an image of a generalizable task or role, irrespective of its context, will fail their purpose because they may be too general to support fully even the original intended user(s) because his/her cultural and organizational context has not been taken into account.

This conference aims to initiate a discussion on these issues which are of vital importance to designing effective and adaptable DSS. It will welcome contributions from all disciplines as the issues in question cut across different disciplines, each making its unique contribution to the topic.
The Conference Goal is:
How can we bring about a more useful, context-sensitive, generation of DSS?

The pivotal issues the conforence wishes to address are:
How one can understand the context within which one designs and implements a DSS?
How one can model, represent and use context in a DSS?
How may context-sensitivity improve the effectiveness of DSS?

We invite papers and panel proposals related to:

- design of DSS for reasoning about context
- design of a DSS that takes into account its context
- interaction of choice and context
- impact of context on DSS success
- role of context in the decision maker-DSS interaction/cooperation
- role of context in the decision maker-DSS complementarity
- role of context in the knowledge organization of the DSS and its reasoning
- context and alternatives (selection, argumentation, explanation)
- context and decision criteria
- change and adaptability of DSS within one organization

SUBMISSION DATES:

- Full abstract (500-600 words):1 October 1997
- Deadline for submission of paper: 9 January 1998
- Notification of acceptance:13 February 1998
- Camera-ready copy due:16 March 1998

INSTRUCTIONS TO AUTHORS:

Only original unpublished papers should be submitted. All submissions will be reviewed. Selection for presentation at the conference and publication in the proceedings will be based on originality, contribution to the field, and relevance to the conference theme. The conference book, published by Chapman and Hall, will be distributed at the conference and at least one author for each paper must register for the conference and present the paper.

Papers must not exceed 12-15 pages when single spaced. All submissions must include on the first page: title, author's name(s), affiliation, complete mailing address, phone number, fax number, and email address. An abstract of 100 words maximum and up to five keywords should be included before the body of the paper. Papers must be submitted in electronic form, using the Chapman and Hall Word template UKdoc.doc, which can be found on the web at `http://www.it-ch.com/itch/authors/macros.html`. This template contains detailed guidelines on how to format your paper. It is very important that you load the template and type within it using the automatic style guidelines it gives.

We would like to request that a 500-600 word abstract for your paper be submitted by 1 October 1997 for comments. Submissions can be sent by e-mail to: Dina Berkeley at the London School of Economics.

For further information, please contact program committee members:

George R. Widmeyer (Conference Chair)
University of Michigan Business School,
701 Tappan Street, Ann Arbor,
Michigan 48109-1234,USA
phone: +1 313 763 5808, fax:+1 313 764 3146
e-mail: `widmeyer@umich.edu`

Dina Berkeley
London School of Economics,
Dept. of Social Psychology, Houghton Str., London WC2A 2AE, UK
phone: +44 171 9557401, fax: +44 171 9163864
e-mail: `d.berkeley@lse.ac.uk`

Patrick Brezillon
LAFORIA - UPMC - Case 169, 4 place Jussieu - F-75252 Paris Cedex 05, France
phone: +33 1 44 27 70 08, fax: +33 1 44 27 70 00
e-mail: `patrick.brezillon@lip6.fr`

Vladislav Rajkovic (Organizing Chair)
Univerza v Mariboru,
Fakulteta za organizacijske vede,
Presernova 11, SI-4000 Kranj, Slovenia
phone: +386 61 1403301, fax: +386 61 1403301
e-mail: `vladislav.rajkovic@ijs.si`

Visit our web site at:
`http://www-personal.umich.edu/`
`~widmeyer/ifipwg83`

CC AI

The journal for the integrated study
of Artificial Intelligence, Cognitive Science
and Applied Epistemology.

CC-AI publishes articles and book reviews relating to the evolving principles and techniques of Artificial Intelligence as enriched by research in such fields as mathematics, linquistics, logic, epistemology, the cognitive sciences and biology.
CC-AI is also concerned with development in the areas of hard- and software and their applications within AI.

Editorial Board and Subscriptions

CC-AI, Blandijnberg 2, B-9000 Ghent, Belgium.
Tel.: (32) (9) 264.39.52,
Telex RUGENT 12.754
Telefax: (32) (9) 264.41.97
e-mail: `Carine.Vanbelleghem@RUG.AC.BE`

First Call for Papers
International Conference on Systems, Signals, Control, Computers (Sscc'98)
International Association for the Advancement of Methods for System Analysis and Design (Iaamsad) and Academy of Nonlinear Sciences (Ans)

Announce the International Conference on
Systems, Signals, Control, Computers (Sscc'98)
Durban, South Africa (September 22-24, 1998)

and Invite Potential Authors for Submission of Papers

A preliminary WEB home page can be accessed on
`http://nsys.ntech.ac.za/iaamsad/SSCC98test.html`
This home page will become public when International Programme Committee membership become confirmed.

Honorary Chairman: Academician V.M.Matrosov (Russia)
Conference Chairman: V.B.Bajic (South Africa)

Advisory Board:

V.B.Bajic (South Africa), J.Brzobohaty (Czech Republic), P.Daoutidis (USA), W.Hide (South Africa), C.Morabito (Italy), V.V.Kozlov (Russia), P.Leach (South Africa), P.C.Muller (Germany), L.Shaikhet (Ukraine), E.Rogers (UK), H.Szu (USA).

International Programme Committee:

V.Apanasovich (Belarus), V.B.Bajic (South Africa), C.Berger-Vachon (France), J.Brzobohaty (Czech Republic), M.Campolo (Italy), P.Daoutidis(USA), T.Fukuda(Japan), Z.Gajic (USA), M.Gams (Slovenia), J.Gil Aluja (Spain), Ly.T.Gruyitch (France), H.Hahn (Germany), M.Hajek (South Africa), R.Harley (South Africa), W.Hide (South Africa), M.Jamshidi (USA), V.Kecman (New Zealand), B.Kovacevic (Yugoslavia), V.Krasnoporoshin (Belarus), V.V.Kozlov (Russia), P.Leach (South Africa), L.K.Kuzmina (Russia), V.Milutinovic (Yugoslavia), C.Morabito (Italy), P.C.Muller (Germany), H.Nijmeijer (The Netherlands), D.H.Owens (UK), D.Petkov (South Africa), K.M.Przyluski (Poland), E.S.Pyatnitskii (Russia), E.Rogers (UK), L.Shaikhet (Ukraine), A.V.Savkin (Australia) H.Szu (USA), E.I.Verriest (USA), R.Vrba (Czech Republic), J.Ziska (Czech Republic).

Local Organizing Committee:

V.Bajic, P.Govender, R.Hacking, M.Hajek, M.McLeod, K.S.Moodley, R.Papa, C.Radhakishun, A.Singh.

Address Of The Conference Office:

Sacan, P.O.Box 1428, Link Hills 3652, Durban, South Africa Tel./Fax: (+27 31) 204-2560 e-mail: `bajic.v@umfolozi.ntech.ac.za`

Supporting Organizations:

SANBI - South African National Institute for Bioinformatics (South Africa)
SAICSIT - South African Institute for Computer Scientists and Information Technologists (South Africa)
CER - Centre for Engineering Research, Technikon Natal (South Africa)
M L Sultan Technikon (South Africa)

General Information

1998 year is the year of Science and Technology in South Africa. The intention of the Department of Arts, Culture, Science and Technology of South Africa is to make South Africans more aware of how Science and Technology affects them in every-day life. Such a national initiative is in a way a very good environment for a conference like this: one that has a broad scope and spans many different fields. At the same time an opportunity is given to the research community of

South Africa to interact more directly with overseas peers.

Aims And Scope

The Conference is broad in scope and will provide a forum for the exchange of the latest research results as applied to different branches of science and technology. The areas of interest include concepts, techniques and paradigms associated with systems, signals, control and/or computers.

Domains of application include informatics, biomedical technology, economics, management, diverse engineering and science fields and applied mathematics. Artificial intelligence techniques are of particular interest, as well as reports on industrial applications.

The conference will include several plenary and invited lectures from world renowned scientists and regular papers. A number of special and invited sessions will also be organised, dealing with focussed areas of interest. The proposals for these special sessions should be submitted at the same time as the abstracts. A special session cannot have less than three papers or more than six.

The official language of the conference is English.

Manuscript Submission And Review Process

Three copies of the extended abstract (at least two pages) should be sent to the Conference Office at the address given below. Full papers are preferred. Papers in Microsoft Word can be sent by e-mail. All submissions will be reviewed by members of the International Programme Committee; additional reviewers will be consulted if necessary. The submissions will be reviewed as soon as they arrive; the average review time is about four weeks. Authors of accepted papers will thereafter be informed (by e-mail if available) of the required format for camera-ready paper submissions. In order for reviewers to be able to assess the submissions, the extended abstract has to provide sufficient information about the background to the problem, the novelty of the obtained results and the results achieved, the conclusions drawn and some references. Up to five keywords should be provided. All submitted papers have to be original, unpublished and not submitted for publication elsewhere.

Proceedings

All accepted papers will be published in the Conference Proceedings, which will be issued by a renowned international publisher.

Important Notice

Although we expect that the authors of accepted papers will present the papers at this Conference, we recognize that circumstances may prevent authors from participation at the Conference. In such cases the accepted papers will be published if the authors inform organizers of their non-attendance at the Conference by 15th May 1998. However, conference fees according to established rules have to be pre-paid in order that papers appear in the Proceedings.

Conference Fees

The conference fee for one participant covers the publication of two papers (each with a maximum of five A4 pages in length) according to the required format; one volume of the Proceedings in which the paper(s) appear(s); refreshment during the conference; one lunch and a banquet. Additional volumes of the Proceedings can be purchased for US$ 55.00. Authors of multiple papers are to pay additional fees for extra papers according to the specified rule. Social programme and tourist visits will be provided at extra cost.

Reduced registration fee of US$ 280.00 (South Africans R 1120.00) is applicable for early received, reviewed and accepted papers for which fee is paid by February 25, 1998 - prospective authors are encouraged to take advantige of this convenience; otherwise the following rates apply:

Early registration fee: US$ 350.00 (South Africans R 1400.00)

Late and on-site registration fee: US$ 400.00 (South Africans R 1600.00)

Student fee: US$ 200.00 (South Africans R 800.00) - to qualify for the student scale of fees, all authors mentioned on the paper have to be current students; written proof has to be provided at the time of payment

Payment in South African rands is possible only when all authors of the papers are South African residents; written proof has to be provided at the time of payment.

Deadlines

Extended Abstracts and Special Session Proposals:
- submission by mail (15th February, 1998)
- submissions by e-mail (15th January, 1998)

Notification of acceptance (15th April, 1998)
Submission of papers in camera ready form (15th May, 1998)
Early payment of conference fees (15th May, 1998)
Late payment of conference fees (31 June, 1998)

Call for Papers

ICEC '98

International Conference on Electronic Commerce
April 6—9, 1998
Hotel Inter - Continental Seoul
Seoul, Korea

"Outstanding papers will be invited for publication in special issues of the following journals:International Journal of Electronic Commerce and Organizational Computing and Electronic Commerce"

Original paper submission due: December 1, 1997
Proposals for panels, tutorials and workshops submission due: December 1, 1997
Notification of acceptance: January 15, 1998
Final camera-ready copy for publication: February 15, 1998

Web site: `http://icec.net`

Organized by International Center for Electronic Commerce

Background

The advent of the Internet as a standard vehicle for communication has led to an acceleration of both the pace and vitality of economic activity. The global information infrastructure serves as the foundation for new modes of personal interaction and business transactions in a collective of activities known as electronic commerce. Electronic commerce represents one of the most promising directions for generating competitive advantage at the micro level of the organization and for increasing productivity at the macrolevel of the economy. In addition, the wealth of online information and the potential to facilitate diverse business transactions highlights the need for developing software tools to take full advantage of the potential for increasing productivity. This conference will serve as a forum for exploring future directions, business opportunities, government policies, and research avenues in the field of electronic commerce. The symposium is addressed to a diverse audience, ranging from researchers and technologists to policy makers and practitioners including developers, users, educators, system managers, and product vendors. This first annual conference ICEC 98 will be held in Seoul, and next ones ICEC 99 and ICEC 2000 will be held in Switzerland and the U.S.A. respectively.

Scope of the Conference

The goal of the conference is to bring together participants interested in the development of electronic commerce in all its aspects. The meeting will focus on substantial contributions in the form of original research, innovative concepts, path-breaking software, advanced applications, scholarly analysis, and incisive policy recommendations. The format of the conference includes invited speakers, technical papers, tutorial presentations, panel discussions, workshop sessions, commercial exhibits, and social activities.

Contributions to the conference may involve technological results, functional capabilities, innovative applications, or policy issues. Suggested topics for submissions to the conference include, but are not limited to the following areas.

1. **Technological Advances** Agent based commerce, Application program integration with web, Authoring environments for html files, Browsers and tools, Consistency and integrity maintenance, Content languages: HTML, VRML, Java, NetRexx, KIF, etc., Charging and payment, Directory and electronic catalogues, Integration of heterogeneous data, Knowledge discovery & Data mining, Multilingual access, Multimedia technology, virtual reality, Protocols: HTTP, IIOP, etc., Search techniques, Security and authorization

2. **Functional Capabilities**
Cyber financial systems, Digitized goods marketing, Direct marketing, EC strategy formulation, Electronic shopping malls, Electronic payment systems, Integration with CALS, Procurement reengineering

3. **Innovative Applications**
Business models, Corporate intranets, Case studies, Technology transfer issues, Validation of systems

4. **Policy Issues** Corporate technological initiatives, Government initiatives, International collaborative programs, International standards, Internet standards, Social impact

Call for Contribution

At the current stage of preparing for the conference, interested individuals are invited to submit papers as well as proposals on tutorials, workshops or panel discussions. All submissions should be in English and in electronic format.

For further information on the conference and electronic submission, refer to `http://icec.net`

Call for Papers

The technical sessions will begin on the third day of the conference. Authors are invited to submit original papers in any area relating to electronic commerce. Each submission should include the following items:

- A cover page listing the title of the paper as well as the name(s), affiliation(s), complete postal address(es), telephone, fax, and e-mail for the principal author.
- Title and Abstract Page.
- The body of the paper, not to exceed 15 pages,double spaced, including illustrations and references.

All text should be in 12 point font, except possibly for illustrations. Authors are asked to submit original papers only. All submitted papers will be refereed for correctness, originality, relevance to the conference, and quality of exposition. The articles will be published in the proceedings of the conference.

Call for Tutorials

Proposals for tutorials, whether for half day or full day, should cover timely topics of interest in electronic commerce. The tutorials will be

held during the first two days of the conference. The format for a tutorial proposal is as follows:
- Tutorial title, a 200 word abstract, and a brief outline. Information on the speaker(s) should include the names, affiliations, addresses (post, telephone, fax and e-mail), publications, and envisioned activities during the tutorial.

Call for Panels

The panels should address timely topics of interest in electronic commerce. The panels will run in parallel with the main conference. The panels will typically run for 1.5 hours, with 3 - 4 speakers. The format for a panel proposal is as follows:
- One page proposal indicating the proposed panel topic, panel chairperson, and likely panel participants. A 200 word abstract should describe the content of the panel and include the full names and addresses (post,telephone, fax and e-mail) of the panel chairperson.

Call for Workshops

Proposals for workshops, half day or full day, should cover timely topics of interest in electronic commerce. The workshops will run in parallel with the conference. The format for a workshop proposal is as follows:
- Workshop title, a 200 word abstract, and a brief outline. Information on the speaker(s) should include the names, affiliations, addresses (post, telephone, fax and e-mail), publications, and envisioned activities during the workshop.

Contact for Submissions

For questions regarding the technical program, please contact.
Steven H. Kim, Program Chair
Graduate School of Management
Korea Advanced Institute of Science & Technology
207-43, Cheongryang, Seoul 130-012, Korea
E-mail : `program@icec.net`
Tel. : +82-2-958-3616
Fax : +82-2-958-3604

Call for Exhibits

Hardware manufacturers, software companies, book publishers, universities and research groups with working programs are invited to present products and demonstrations during the conference.

Secretariat for ICEC'98 Exhibits

Yong Jung Kim
ICEC'98 Secretariat
4th Fl. Jisung Bldg., #645-20
Yeoksam 1-dong, Kangnam-gu
Seoul 135-081, Korea
E-mail : `intercom@soback.kornet.nm.kr`
tel. : +82-2-568-3208; 566-6339
Fax : +82-2-565-2434; 3452-7292

Headquarters

Jeong Yoon Chey
International Center for Electronic Commerce
60-97 Hoegi-dong, Dongdaemun-gu
Seoul 130-050, Korea
E-mail: `jychey@icec.net`
tel.: +82-2-3295-0620
fax: +82-2-3295-0623

Important Dates

Original paper submission due: December 1, 1997
Proposals for panels, tutorials and workshops submission due: December 1, 1997
Notification of acceptance: January 15, 1998
Final camera-ready copy for publication: February 15, 1998
Early registration due: February 10, 1998
Hotel reservation due: February 10, 1998

Upon acceptance of the paper, authors are expected to register and to present their papers. Papers will be published in the conference proceedings only if at least one of the authors is officially registered.

Conference Organization

Honorary Conference Chair:
Myong Oh, President
Dong a Ilbo, Daily Newspaper
Conference Chair:
Jae Kyu Lee, professor
Graduate School of Management
Korea Advanced Inst. of Science & Technology
207-43, Cheongryang, Seoul 130-012, Korea
E-mail: jklee@msd. kaist. ac. kr
Tel.: +82-2-958-3612
Fax: +82-2-960-2102
Conference Co-Chairs:
Andrew B. Whinston, professor
MSIS Department cBA 5.202
University of Texas at Austin
Austin, TX78712-1175, U.S.A.
e-mail: abw@uts.cc.utexas.edu
Beat Schmid, professor
Institute for information Management
University of St. Gallen
Dufourstrasse 50 CH - 9000
St. Gallen, Switzerland
e-mail: Beat.Schmid@iwi.unisg.ch

International Program Committee

Chair:
Steven H. Kim
(Korea Advanced Institute of Science and Technology, Korea)
Members:
Abersek, Boris (University of maribor, Slovenia) Bodendorf, Freimut (University of Erlangen-Nuremberg, Germany) Buhl, Hans Ulrich (Augsburg University, Germany) Bui, Tung (Naval Post-Graduate School, U.S.A.) Cardenosa, Jesus (University Politecnica de madrid, Spain) Drury, D.H. (McGill University, Canada) Dutta, Amitava (George Mason University, U.S.A.) Ertas, Atila (Texas Technology University, U.S.A.) Fourer, Robert (Northwestern University, U.S.A.) Gams, Matjaz (Jozef Stefan Institute, Slovenia) Glover, Fred (University of Colorado, U.S.A.) Gray, Paul (Claremont Graduate School, U.S.A.) Hanappi, Gerhard (University of Technology Vienna & Austrian Academy of Sciences, Austria) Holsapple, Clyde W. (University of Kentucky, U.S.A.) Iijima, Junichi (Tokyo Institute of Technology, Japan) Isakowitz, Tomas (New York University, U.S.A.) Jovanovic, Aleksandar (MPA Stuttgart, Germany) Kauffman, Robert J. (University of Minnesota, U.S.A.) Kim, Steven H. (Korea Advanced Institute of Science and Technology, Korea) Klein, Stefan (Koblenz-Landau University, Germany) Krishnan, Ramayya (Carnegie Mellon University, U.S.A.) Kusiak, Andrew (University of Iowa, U.S.A.) Lee, Heeseok (Korea Advanced Institute of Science and Technology, Korea) Lee, Ho Geun (Hong Kong University of Science and Technology, Hong Kong) Lee, Jae Kyu (Korea Advanced Institute of Science and Technology, Korea) Lee, Ronald M. (Erasmus University, the Netherlands) Lee, Young Hae (Hanyang University, Korea) Leong, G. Keong (Ohio State University, U.S.A.) Liang, Ting-Peng (sun-Yat Sun University, Taiwan) Liebowitz, Jay (George Washington University, U.S.A.) Milani, Alfredo (University of Perugia, Italy) Narasimhalu, Desai (National University of Singapore, Singapore) Narasimhan, Sridhar (Georgia Institute of Technology, U.S.A.) O'Keefe, Bob (Brunel University, England) O'leary, Daniel (University of Southern California, U.S.A.) Rolland, Erik (Univ. of California, U.S.A.) Schmid, Beat (St. Gallen University, Switzerland) Sharda, Ramesh (Oklahoma State University, U.S.A.) Sheng, Olivia R. Liu (University of Arizona, U.S.A.) Siong, Neo Boon (Nanyang Technological University, Singapore) Tam, Kar Yan (Hong Kong University of Science and Technology, Hong Kong) Tan, Margaret (National University of Singapore, Singapore) Tanik, Murat (New Jersey Institute of Technology, U.S.A.) Tung, Lai Lai (Nanyang Technological University, Singapore) Turban, Efraim (California State University, U.S.A.) Vogel, Doug (University of Arizona, U.S.A.) Whinston, Andrew B. (University of Texas at Austin, U.S.A.) Yu, Chien-Chih (National ChengChi University, Taiwan) Zwass, Vladimir (Fairleigh Dickinson University, U.S.A.)

Local Organizing Committee

Chair: Lee, Sang Kee Hanmoo Development Co. Ltd., Korea

Co-Chairs: Choi, Tae Young INTERCOM Convention Services, Korea

Lee, Seong-ha Computer World, Korea

Lee, Sang Goo International Center for Electronic Commerce, Korea

First Announcement and Call for Papers

JCKBSE'98

Third Joint Conference on
Knowledge-Based Software Engineering
Smolenice, Slovakia,
September 9-11, 1998

Sponsored by:

SIG on Knowledge-Based Software Engineering, Institute of Electronics, Information and Communication Engineers (EICE), Japan
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia
Russian Association for Artificial Intelligence
Bulgarian Artificial Intelligence Association

In cooperation with:

Japanese Society for Artificial Intelligence
Slovak Society for Informatics
The Institution of Electrical Engineers - Slovak Centre

Steering Committee:

Christo Dichev, IIT, Bulgarian Academy of Sciences Morio Nagata, Keio University Pavol Navrat, Slovak University of Technology Vadim L. Stefanuk, IITP, Russian Academy of Sciences Haruki Ueno, Tokyo Denki University

About the Conference

Joint Conference on Knowledge-Based Software Engineering aims to provide a forum for researchers and practitioners to discuss topics in knowledge engineering and in software engineering. Special emphasis is given to application of knowledge-based methods to software engineering problems. The conference originated from efforts to provide a suitable forum for contacts for scientists mainly from Japan, the CIS countries and the countries of the Central and Eastern Europe while always being open for participants from the whole world. JCKBSE'98 will continue in this tradition and expand for even greater international participation. Also, the scope of the conference as indicated by its topics is being updated to reflect the recent development in all three areas i.e.,

- knowledge engineering,
- software engineering,
- knowledge-based software engineering.

The conference will also include invited talks.

Topics (include, but are not limited to)

- Architecture of knowledge, software and information systems including collaborative, distributed, multi-agent and multimedia systems, internet and intranet
- Domain modelling
- Requirements engineering, formal and semiformal specifications
- Intelligent user interfaces and human machine interaction
- Knowledge acquisition and discovery, data mining
- Automating software design and synthesis
- Program understanding, programming knowledge
- Object-oriented and other programming paradigms, metaprogramming
- Reuse, re-engineering, reverse engineering
- Knowledge-based methods and tools for software engineering, including testing, verification and validation, process management, maintenance and evolution, CASE
- Decision support methods for software engineering
- Applied semiotics for knowledge-based software engineering
- Learning of programming, modelling programs and programmers
- Knowledge systems methodology, development tools and environments
- Software engineering and knowledge engineering education, distance learning, emergence of an information society

Program Committee

Zbigniew Banaszak Seiichi Komiya Technical Univ., Zielona Gora,PL IPA, Chair of SIG-KBSE, Co-chair, JPN Andras Benczur Behrouz H. Far Eotvos Lorand Univ.,Budapest,HU Saitama Univ., JPN Maria

Venue

Smolenice castle is a beautiful site renown for providing excellent environment for scientific conferences. It serves as a congress centre. It is situated approximately 60km from Bratislava in the surroundings of Small Carpathians mountains.

Proceedings

All accepted papers will be published in the conference proceedings and will be available at the conference. In addition, several of the highest quality papers will be selected for a special issue of IEICE Transactions on Information and Systems.

Language

The official language of the conference will be English.

Paper Submission

Full papers should not exceed 8 pages. Short papers should not exceed 4 pages. Papers will be reviewed according to: technical quality, originality, clarity, appropriateness to the conference focus, and adequacy of references to related work. Authors should submit the papers electronically. For details see conference web site. In addition, one copy of a manuscript should be sent, too. Each paper should contain the following information:

- Title of the paper.
- Name, affiliation, mailing address of the author, e-mail.
- Abstract of 100-200 words.
- The subject category (the topic) in which the paper should be reviewed.

Important Dates

Feb. 1, 1998 - Registration forms
March 1, 1998 - Paper submission deadline
May 1, 1998 - Notification of acceptance
May 20, 1998 - Camera-ready deadline
Sept. 9 - 11, 1998 - Conference dates

Correspondence Address

JCKBSE'98
Department of Computer Science and Engineering
Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, Slovakia
fax: +421 7 720 415,
e-mail: `jckbse98@dcs.elf.stuba.sk`

More information about JCKBSE'98 is available on the conference web site:
`http://www.dcs.elf.stuba.sk/jckbse98/`

Fee

Participants will pay for an integrated package comprising registration fee, board and lodging, proceedings and a half day excursion. Participants from CIS and Central and Eastern Europe are eligible for a substantially reduced fee.

Support For Students

There will be available a limited number of scholarships for students submitting papers to the conference to support partially their participation.

Second Call for Papers

PAKDD-98

The 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining

Melbourne Convention Centre, Melbourne, Australia,
15-17 April 1998

Invited Speakers:
Jiawei Han (ACSys Keynote Speaker, Simon Fraser U.); Chris Wallace (Monash U.)

The Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98) will provide an international forum for the sharing of original research results and practical development experiences among researchers and application developers from different KDD related areas such as machine learning, databases, statistics, knowledge acquisition, data visualization, software re-engineering, and knowledge-based systems. It will follow the success of PAKDD-97 held in Singapore in 1997 by bringing together participants from universities, industry and government.

Papers on all aspects of knowledge discovery and data mining are welcome. Areas of interest include, but are not limited to:
Data and Dimensionality Reduction
Data Mining Algorithms and Tools
Data Mining and Data Warehousing
Data Mining on the Internet
Data Mining Metrics
Data Preprocessing and Postprocessing
Data and Knowledge Visualization
Deduction and Induction in KDD
Discretisation of Continuous Data
Distributed Data Mining KDD Framework and Process
Knowledge Representation and Acquisition in KDD
Knowledge Reuse and Role of Domain Knowledge
Knowledge Acquisition in Software Re-Engineering and Software Information Systems
Induction of Rules and Decision Trees
Management Issues in KDD Machine Learning, Statistical and Visualization Aspects of KDD (including Neural Networks, Rough Set Theory and Inductive Logic Programming)
Mining in-the-large vs Mining in-the-small
Noise Handling
Security and Privacy Issues in KDD
Successful/Innovative KDD Applications in Science, Government, Business and Industry

Both research and applications papers are solicited. All submitted papers will be reviewed on the basis of technical quality, relevance to KDD, significance, and clarity. Accepted papers will be published in the conference proceedings by Springer-Verlag (in the LNAI series). A selected number of the accepted papers will be expanded and revised for inclusion in a special issue of an international journal.

All submissions should be limited to a maximum of 5,000 words. Four hardcopies should be forwarded to the following address:
Professor Ramamohanarao Kotagiri
(PAKDD '98)
Department of Computer Science,
The University of Melbourne
Parkville, VIC 3052, Australia

Please include a cover page containing the title, authors (names, postal and email addresses), a 200-word abstract and up to 5 keywords. This cover page must accompany the paper.

4 copies of full papers by:	October 16, 1997
Acceptance notices:	December 22, 1997
Camera-readies due by:	January 30, 1998
Conference:	April 15-17, 1998

CONFERENCE ORGANIZATION

Conference Chairs:

Ross Quinlan, Sydney University
Bala Srinivasan, Monash University

Program Chairs:

Xindong Wu, Monash University
Ramamohanarao Kotagiri, Melbourne Univ.

Organising Committee Chairs:

Kevin Korb, Monash University
Graham Williams, CSIRO, Australia

Publicity Chair:

Lipo Wang, Deakin University

Tutorial Chair:

Jon Oliver, Monash University

Treasurer:

Michelle Riseley, Monash University

Program Committee:

Grigoris Antoniou, Griffith University
James Boyce, Kings College, London
Ivan Bratko, Ljubljana University
Mike Cameron-Jones, Univ. of Tasmania
Arbee Chen, Natl. Tsing Hua U., Taiwan
David Cheung, Hong Kong University
Vic Ciesielski, RMIT
Honghua Dai, Monash University
John Debenham, Univ. of Tech., Sydney
Olivier de Vel, James Cook University
Tharam Dillon, La Trobe University
Guozhu Dong, University of Melbourne
Peter Eklund, University of Adelaide
Usama Fayyad, Microsoft Research, USA
Matjaz Gams, Jozef Stefan Institute
Yike Guo, Imperial College, London
David Hand, Open University, UK
Evan Harris, University of Melbourne
David Heckerman, Microsoft Research, USA
David Kemp, University of Melbourne
Masaru Kitsuregawa, Tokyo University
Kevin Korb, Monash University
Hingyan Lee, Japan Singapore AI Center
Jae-Kyu Lee, KAIST, Korea
Deyi Li, Beijing System Engineering Institute
T.Y. Lin, San Jose State University
Bing Liu, National University of Singapore
Huan Liu, National University of Singapore
Zhi-Qiang Liu, University of Melbourne
Hongjun Lu, National University of Singapore
Dickson Lukose, University of New England
Kia Makki, University of Nevada, Las Vegas
Heikki Mannila, University of Helsinki
Peter Milne, CSIRO, Australia
Shinichi Morishita, IBM Japan
Hiroshi Motoda, Osaka University
Hwee-Leng Ong, Japan Singapore AI Center
Jon Oliver, Monash University
Maria Orlowska, University of Queensland
Gregory Piatetsky-Shapiro,
Geneve Consulting Group, USA
Niki Pissinou, University of S/W Louisiana
Peter Ross, Edinburgh University
Claude Sammut, Univ. of New South Wales
Heinz Schmidt, Monash University S. Seshadri, IIT Bombay
Hayri Sever, Hacettepe University
Arun Sharma, University of New South Wales
Evangelos Simoudis, IBM, USA
Atsuhiro Takasu, NCSIS, Japan
Takao Terano, University of Tsukuba
Bhavani Thuraisingham, MITRE, USA
Kai Ming Ting, Waikato University
David Urpani, CSIRO, Australia
R. Uthurusamy, General Motors, USA
Lipo Wang, Deakin University
Geoff Webb, Deakin University
Graham Williams, CSIRO, Australia
Beat Wuthrich, Hong Kong Univ of Sci & Tech
Xin Yao, ADFA/Univ. of New South Wales
John Zeleznikow, La Trobe University
Diancheng Zhang, Hefei Univ. of Tech.
Ming Zhao, Telstra
Zijian Zheng, Deakin University
Ning Zhong, Yamaguchi University
Justin Zobel, RMIT
Dr Xindong Wu
Dept of Software Development, Monash Univ.
900 Dandenong Road
Caulfield East, Melbourne 3145 Australia
Phone: +61 3 9903 1025
Fax: +61 3 9903 1077
Email: `xindong@insect.sd.monash.edu.au`

PAKDD-98 Home Page:

`http://www.sd.monash.edu.au/pakdd-98`

Tenth IASTED International Conference on Parallel and Distributed Computing and Systems

Las Vegas, Nevada, U.S.A., October 28-31, 1998
Sponsored by International Association of Science and Technology for Development (IASTED)
`http://www.cps.udayton.edu/~pan/pdcs98`

Purpose

The International Conference on Parallel and Distributed Computing and systems, sponsored by IASTED, is a major annual forum for scientists, engineers, and practitioners throughout the world to present the latest research results, ideas, development, and applications in all areas of parallel and distributed processing. The 1997 conference attracted researchers from 30 countries. The 1998 meeting (PDCS '98) will be held in Las Vegas, Nevada, U.S.A., and will include keynote addresses, contributed papers, tutorials, and workshops.

Scope

The main focus of PDCS '98 will be parallel and distributed computing and systems viewed from the three perspectives of architecture and networking, software systems, and algorithms and applications. Topics include, but are not limited to, the following:

Architecture and Networking: SIMD/MIMD processors, various parallel/concurrent architecture styles, interconnection networks, memory systems and management, I/O in parallel processing, VLSI systems, optical computing, computer networks, communications and telecommunications, wireless networks and mobile computing.

Software Systems: operating systems, programming languages, various parallel programming paradigms, vectorization and program transformation, parallelizing compilers, tools and environments for software development, distributed data- and knowledge-base systems, modelling and simulation, performance evaluation and measurements, visualization.

Algorithms and Applications: parallel/distributed algorithms, resource allocation and management, load sharing and balancing, task mapping and job scheduling, network routing and communication algorithms, reliability and fault tolerance, signal and image processing, neural networks, high-performance scientific computing, application studies.

Paper Submission Guidelines

Papers reporting original and unpublished research results and experience are solicited. Papers will be selected based on their originality, timeliness, significance, relevance, and clarity of presentation. Accepted and presented papers will be published in the conference proceedings.

Please send four copies of a manuscript to the program committee chair at the following address by April 15, 1998: Professor Yi Pan, PDCS '98 Program Chair, Department of Computer Science, University of Dayton, Dayton, Ohio, 45469-2160, U.S.A. Phone: (937) 229-3807. Fax: (937) 229-4000. Email: pan@udcps.cps.udayton.edu.

A manuscript should not exceed 15 pages, including tables and figures. In the cover letter, please indicate the author for correspondence, and his/her complete postal address, phone and fax numbers, and email address (make sure the email address is current and working).

Tutorials/Workshops

Several workshops are being planned for PDCS '98. Each workshop will focus on a particular topic, and consists of several presentations and open discussion. A one-page abstract of each workshop presentation will be published in the conference proceedings. The proposal for a workshop should include the title, topics covered, proposed/invited speakers, and estimated length (hours) of the workshop. Anyone wishing to organize a workshop in connection with PDCS '98 should submit four copies of his/her proposal to the program chair at the address given above by April 15, 1998.

PDCS '98 will also offer half-day tutorials in parallel and distributed computing. Each tutorial proposal should provide the title, topics, targeted audiences, and instructor's biography. The proposal should be submitted to the tutorial chair Dr. Pradip K. Srimani via email at srimani@CS.Colostate.EDU.

IJPDSN

A special issue consisting of selected papers from PDCS '98 will be published in International Journal of Parallel and Distributed Systems and Networks. Other papers may be submitted to: Professor Marlin H. Mickle, Editor-In-Chief, IJPDSN, Department of

Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA.

Important Dates

Paper submission deadline: **April 15, 1998**
Author notification: June 15, 1998
Camera-ready version due: August 1, 1998
Workshop/tutorial proposal due: April 15, 1998
Conference: October 28-31, 1998

General Co-Chairs

Selim G. Akl, Queens University (Canada)
Keqin Li, State University of New York (USA)

Program Chair

Yi Pan, University of Dayton (USA)

Tutorial Chair

Pradip K. Srimani, Colorado State University (USA)

Local Arrangement Chair

Shahram Latifi, University of Nevada at Las Vegas (USA)

Program Committee

Ishfaq Ahmad, Hong Kong Univ. of Sci. & Tech. (H. K.)
Hamid R. Arabnia, University of Georgia (USA)
Mohammed Atiquzzaman, University of Dayton (USA)
Johnnie W. Baker, Kent State University (USA)
David A. Berson, Intel Corporation (USA)
Jingde Cheng, Kyushu University (Japan)
Kam-Hoi Cheng, University of Houston (USA)
Henry Chuang, University of Pittsburgh (USA)
Kuo-Liang Chung, Nat'l Taiwan U. of Sci. & Tech (Taiwan)
Bin Cong, California Polytechnic State University (USA)
Mark Cross, University of Greenwich (UK)
Sajal K. Das, University of North Texas (USA)
Brian J. d'Auriol, The University of Akron (USA)
Frank Dehne, Carleton University (Canada)
Eliezer Dekel, IBM - Haifa Research Lab. (Israel)
Ivan Dimov, Bulgarian Academy of Sciences (Bulgaria)
Erik H. D'Hollander, University of Ghent (Belgium)
Omer Egecioglu, Univ. of California at Santa Barbara (USA)
Hossam ElGindy, University of Newcastle (Australia)
Afonso Ferreira, CNRS-INRIA (France)
Rajiv Gupta, University of Pittsburgh (USA)
Mounir Hamdi, Hong Kong Univ. of Sci. & Tech. (H. K.)
Jack Jean, Wright State University (USA)
Weijia Jia, City University of Hong Kong (Hong Kong)
Shahram Latifi, University of Nevada (USA)
Yamin Li, University of Aizu (Japan)
Ahmed Louri, University of Arizona (USA)
Bruce Maggs, Carnegie Mellon University (USA)
Brian Malloy, Clemson University (USA)
Koji Nakano, Nagoya Institute of Technology (Japan)
William T. O'Connell, Lucent Bell Labs (USA)
Michael A. Palis, Rutgers University (USA)
Marcin Paprzycki, University of Southern Mississippi (USA)
Behrooz Parhami, Univ. of Calif. at Santa Barbara (USA)
Chan-Ik Park, Pohang Univ. of Science & Tech (Korea)
Lori Pollock, University of Delaware (USA)
Jerry L. Potter, Kent State University (USA)
Chunming Qiao, State University of New York (USA)
C. S. Raghavendra, The Aerospace Corporation (USA)
Sanguthevar Rajasekaran, University of Florida (USA)
Hesham El-Rewini, University of Nebraska at Omaha (USA)
Jose D. P. Rolim, University of Geneva (Switzerland)
Hemant Rotithor, Digital Equipment Corporation (USA)
Ponnuswamy Sadayappan, Ohio State University (USA)
Assaf Schsuter, Technion (Israel)
Edwin Sha, University of Notre Dame (USA)
Hong Shen, Griffith University (Australia)
Xiaojun Shen, University of Missouri at Kansas City (USA)
Gurdip Singh, Kansas State University (USA)
Mary Lou Soffa, University of Pittsburgh (USA)
Pradip K. Srimani, Colorado State University (USA)
Per Stenstrom, Chalmers Univ. of Technology (Sweden)
Ivan Stojmenović, University of Ottawa (Canada)
Hal Sudborough, University of Texas at Dallas (USA)
Jerry L. Trahan, Louisiana State University (USA)
Ramachandran Vaidyanathan, Louisiana State Univ. (USA)
Subbarayan Venkatesan, University of Texas at Dallas (USA)
Alan S. Wagner, University of British Columbia (Canada)
Yuanyuan Yang, University of Vermont (USA)

Steering Committee

Narsingh Deo, University of Central Florida (USA)
Ahmed Elmagarmid, Purdue University (USA)
Geoffrey Fox, Syracuse University (USA)
H. Scott Hinton, University of Colorado (USA)
D. Frank Hsu, Fordham University (USA)
Oscar Ibarra, Univ. of California at Santa Barbara (USA)
Joseph JáJá, University of Maryland (USA)
Lennart Johnsson, University of Houston (USA)
H. T. Kung, Harvard University (USA)
Tom Leighton, Massachusetts Inst. of Technology (USA)
Jane Liu, University of Illinois at Urbana Champaign (USA)
Marlin Mickle, University of Pittsburgh (USA)
Lionel Ni, Michigan State University (USA)
Stephan Olariu, Old Dominion University (USA)
Sartaj Sahni, University of Florida (USA)
Eugen Schenfeld, NEC Research Institute (USA)
Marc Snir, IBM - Thomas Watson Research Center (USA)
Hal Sudborough, University of Texas at Dallas (USA)
Si-Qing Zheng, Louisiana State University (USA)
Albert Y. Zomaya, Univ. of Western Australia (Australia)

Call for Papers

Third International Conference on Emergence

Helsinki, Finland
August 3-7, 1998

ECHO III
(Emergence, Complexity, Hierarchy, Organisation)
Under the auspices of the
International Society for the Study of Emergence
In Cooperation with the Helsinki University of Technology

Organizing Committee

Harald Atmanspacher
Max-Planck-Institut fuer extraterrestrische Physik
Garching, DE

Andrée Ehresmann
Editor
Cahiers de Topologie et de Géométrie Differentielle Catégoriques
Facultq'e de Mathématique et d'Informatique
Université de Picardie
Amiens, France

George Farre
Director
Interdisciplinary Program in Cognitive Science
New North Hall 223
Department of Philosophy
Georgetown University
Washington DC, USA
`farregl@gusun.georgetown.edu`

Rom Harré
Philosophy/Psychology
Georgetown University
Washington DC, USA

Jesper Hoffmeyer
Institute of Molecular Biology
The Biosemiotic Group
University of Copenhagen
Copenhagen, Danemark

Koichiro Matsuno
BioEngineering
Nagaoka University of Technology
Nagaoka, Japan

Paul Rapp
Clinical Research Center
Norristown State Hospital
Norristown PA, USA

Tarkko Oksala
Chair, Local Committee
Architecture
Helsinki University of Technology
Helsinki, Finland

Themes

Evolution, the defining characteristic of the observable universe (the big bang and all that) is punctuated by the emergence of complex systems. Narly all of these are hierarchies of distinct strata with different sets of characteristics, separated by conceptually and methodologically significant gaps or cuts.

Emergence so defined has many disciplinary dimensions, going from quantum physics all the way to the study of conscious organisms and the social structures they create by way of chemistry, biology, neurology, cognitive science, and including system theory, complexity theory and their philosophical dimensions. These disciplinary dimensions define the main foci of the conference, for which papers are hereby solicited.

The abstract must be received by Prof. George L. Farre by January 10, 1998. Authors will be informed of the decision of the review committee by the middle of March 1998 at the latest. The final draft of the paper (two hard copies and one IBM compatible diskette) should be received by Prof. Farre by the 15th of May, 1998 at the very latest. All the papers will be sent to all authors by e-mail in early July, to facilitate discussion during the conference. Each author will have 50 minutes, which may be segmented between presentation and discussion.

Conference Fees

The Conference fees are $ 200.00 if paid by April 15, 1998, $ 225.00 if after that date. They include a copy of the Proceedings, lunches, a Conference dinner/banquet, and a cocktail party. Additional activities will be announced in due time. Checks should be made payable to: ISSE (ECHO III) and sent to Prof. George L. Farre.

All correspondence relating to the papers should be sent to Prof. George L. Farre at the address indicated above. All correspondence relating to the arrangements in Helsinki should be sent to Prof. Tarkko Oksala, at the address indicated above.

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

Address: Slovenska 50, 1000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 1324 140.
WWW:http://www.mzt.si
Minister: Lojze Marinček, Ph.D.

The Ministry also includes:

The Standards and Metrology Institute of the Republic of Slovenia
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 314 882.

Slovenian Intellectual Property Office
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 318 983.

Office of the Slovenian National Commission for UNESCO
Address: Slovenska 50, 1000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 302 951.

Scientific, Research and Development Potential:

The Ministry of Science and Technology is responsible for the R&D policy in Slovenia, and for controlling the government R&D budget in compliance with the National Research Program and Law on Research Activities in Slovenia. The Ministry finances or co-finance research projects through public bidding, while it directly finance some fixed cost of the national research institutes.

According to the statistics, based on OECD (Frascati) standards, national expenditures on R&D raised from 1,6 % of GDP in 1994 to 1,71 % in 1995. Table 2 shows an income of R&D organisation in million USD.

Objectives of R&D policy in Slovenia:

- maintaining the high level and quality of scientific technological research activities;
- stimulation and support to collaboration between research organisations and business, public, and other sectors;
- stimulating and supporting of scientific and research disciplines that are relevant to Slovenian national authenticity;
- co-financing and tax exemption to enterprises engaged in technical development and other applied research projects;
- support to human resources development with emphasis on young researchers; involvement in international research and development projects;
- transfer of knowledge, technology and research achievements into all spheres of Slovenian society.

Table source: Slovene Statistical Office.

Total investments in R&D (% of GDP)	1,71
Number of R&D Organisations	297
Total number of employees in R&D	12.416
Number of researchers	6.094
Number of Ph.D.	2.155
Number of M.Sc.	1.527

Table 1: Some R&D indicators for 1995

	Ph.D.			M.Sc.		
	1993	1994	1995	1993	1994	1995
Bus. Ent.	51	93	102	196	327	330
Gov. Inst.	482	574	568	395	471	463
Priv. np Org.	10	14	24	12	25	23
High. Edu.	1022	1307	1461	426	772	711
TOTAL	1565	1988	2155	1029	1595	1527

Table 2: Number of employees with Ph.D. and M.Sc.

	Basic Research		Applied Research		Exp. Devel.		Total	
	1994	1995	1994	1995	1994	1995	1994	1995
Business Enterprises	6,6	9,7	48,8	62,4	45,8	49,6	101,3	121,7
Government Institutes	22,4	18,6	13,7	14,3	9.9	6,7	46,1	39,6
Private non-profit Organisations	0,3	0,7	0,9	0,8	0,2	0,2	1,4	1,7
Higher Education	17,4	24,4	13,7	17,4	8,0	5,7	39,1	47,5
TOTAL	46,9	53,4	77,1	94,9	63.9	62,2	187,9	210,5

Table 3: Incomes of R&D organisations by sectors in 1995 (in million USD)

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slovenia** (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1773 900, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

CONTENTS OF INFORMATICA Volume 21 (1997) pp. 1–705

Papers

MARCER, P.J. & W. SCHEMPP. 1997. Model of the Neuron Working by Quantum Holography. Informatica 21:517–532.

MIEL, G.J. & P.D. TURNBOUGH. 1997. Algorithms in the Method of Paired Comparisons. Informatica 21:293–297.

MIZZARO, S. 1997. Towards Recursive Models—A Computational Formalism for the Semantics of Temporal Presuppositions and Counterfactuals in Natural Language. Informatica 21:59–77.

PERUŠ, M. 1997. System-processual Backgrounds of Consciousness. Informatica 21:491–506.

PIDD, M. 1997. Model Development and HCI. Informatica 21:599–605.

RAKOVIĆ, D. 1997. Prospects for Conscious Brain-like Computers: Biophysical Arguments. Informatica 21:507–516.

RESLER, D. & D. CROOKES 1997. A Study in Generating Modula-2 from Prolog. Informatica 21:115–127.

ROSE, J.R. & C.M. EASTMAN. 1997. Hiearchical Classification as an Aid to Browsing. Informatica 21:49–57.

SCHEMPP, W. 1997. Quantum Holography and Magnetic Resonance Tomography: An Ensemble Quantum Computing Approach. Informatica 21:541–562.

SEGHROUCHNI, A.E.F. 1997. Multi-Agent Systems as a Paradigm for Intelligent System Design. Informatica 21:173–184.

SIEMER, J. 1997. Enhancing Simulation Education with Intelligent Tutoring Systems. Informatica 21:637–642.

STÅHL, I. 1997. Teaching the Fundamentals of Simulation in a Very Short Time. Informatica 21:627–635.

SUGIYAMA, S. 1997. A Basic Idea of Consciousness. Informatica 21:465–470.

SUN, Y. 1997. A Case Study in Non-monotonic Reasoning: an Alternative Analysis of the Yale Shooting Problem. Informatica 21:249–254.

ULE, A. 1997. Consciousness and Process. Informatica 21:683–687.

VENTER, F.J., G.D. OOSTHUIZEN & J.D. ROSS. 1997. Lattice-based Knowledge Discovery in Network Management Data. Informatica 21:227–238.

YUAN, S.-T. 1997. A Safe and Efficient Agent Architecture. Informatica 21:161–171.

ŽELEZNIKAR, A.P. 1997. Informational Graphs. Informatica 21:79–114.

ŽELEZNIKAR, A.P. 1997. Informational Theory of Consciousness. Informatica 21:345–368.

Editorials

CHEON, S.W. 1997. Toward More Human-like Systems Using the Techniques of Artificial Intelligence. Informatica 21:157–159.

PAUL, R.J. & S.J.E. TAYLOR. 1997. Simulation Modelling Methodology and Education. Informatica 21:579–586.

ŽELEZNIKAR, A.P. 1997. Informational Supervenience. Informatica 21:1–4.

ŽELEZNIKAR, A.P. 1997. Conceptualism of Consciousness. Informatica 21:333–344.

Profile

MARUYAMA, M. 1997. A Situational Informatical Dynamics: The Case of Situation-contextual and Time contextual Non-additive Influences. Informatica 21:5–18.

Journal and Book Overviews

ŽELEZNIKAR, A.P. 1997. Consciousness. Scientific Challenge of the 21st Century. Informatica 21:148.

JOSEPH, H.D. Intelligent Data Analysis—An International Journal. 1997. Informatica 21:151–153.

ŽELEZNIKAR, A.P. 1997. Grundlagenstudien aus Kybernetik und Geisteswissenschaft—grkg/Humankybernetik. Informatica 21:321–326.

Calls for Papers

Consciousness as Informational Phenomenalism. 1997. Informatica 21:147.

Parallel and Distributed Database Systems. 1997. Informatica 21:149.

Parallel Computing with Optical Interconnections. 1997. Informatica 21:150.

ERK '97—Electrotechnical and Computer Science Conference. 1997. Informatica 21:154, 327.

1997 IEEE Knowledge and Data Engineering Exchange Workshop. 1997. Informatica 21:328–329.

CC AI. 1997. Informatica 21:329, 691.

Context-sensitive Decision Support Systems. 1997. Informatica 21:563–564, 690–691.

Joint Conference on Knowledge-based Software Engineering. 1997. Informatica 21:365, 697–698.

PAKDD-98—The 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining. 1997. Informatica 21:566–567, 699–700.

GKPO '97—5th International Conference on Computer Graphics and Image Processing. 1997. Informatica 21:568–569.

Business Information Systems—BIS '98. 1997. Informatica 21:570–573.

4th International Conference on Numerical Methods ans Applications—NM&A '98. 1997. Informatica 21:574–575.

New INFORMATION SOCIETY. 1997. Informatica 21:689.

SSCC'98—Systems, Signals, Control, Computers. 1997. Informatica 21:692–693.

ICEC '98—International Conference on Electronic Commerce. 1997. Informatica 21:694–696.

IASTED International Conference on Parallel and Distributed Computing and Systems. 1997. Informatica 21:701–702.

Third International Conference on Emergence. 1997. Informatica 21:703.

Professional Societies

Jožef Stefan Institute. 1997. Informatica 21:156,331, 577.

The Ministry of Science and Technology of the Republic of Slovenia. 1997. Informatica 21:155,330,576.

Errata

AI in Eastern and Central Europe. 1997. Informatica 21:153.

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica LaTeX format and figures in .eps format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than five years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

Referees:

Witold Abramowicz, David Abramson, Kenneth Aizawa, Suad Alagić, Alan Aliu, Richard Amoroso, John Anderson, Catriel Beeri, Daniel Beech, Fevzi Belli, Istvan Berkeley, Azer Bestavros, Balaji Bharadwaj, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Jerzy Brzezinski, Marian Bubak, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Netiva Caftori, Ryszard Choras, Jason Ceddia, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, David Cliff, Travis Craig, Noel Craske, Tadeusz Czachorski, Milan Češka, Andrej Dobnikar, Sait Dogru, Georg Dorfner, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Jozo Dujmović, Pavol Ďuriš, Hesham El-Rewini, Pierre Flener, Terrence Forgarty, Hans Fraaije, Hugo de Garis, Eugeniusz Gatnar, James Geller, Michael Georgiopolus, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Inman Harvey, Elke Hochmueller, Rod Howell, Tomáš Hruška, Ryszard Jakubowski, Piotr Jedrzejowicz, Eric Johnson, Polina Jordanova, Sabhash Kak, Li-Shan Kang, Roland Kaschek, Jan Kniat, Stavros Kokkotos, Kevin Korb, Gilad Koren, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Phil Laplante, Bud Lawson, Ulrike Leopold-Wildburger, Joseph Y-T. Leung, Alexander Linkevich, Raymond Lister, Doug Locke, Matija Lokar, Jason Lowder, Peter Marcer, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Roland Mittermeir, Madhav Moganti, Tadeusz Morzy, Daniel Mossé, John Mueller, Hari Narayanan, Jaroslav Nieplocha, Jerzy Nogieć, Stefano Nolfi, Cherry Owen, Tadeusz Pankowski, Mitja Peruš, Warren Persons, Stephen Pike, Niki Pissinou, Gustav Pomberger, James Pomykalski, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Wolf Rauch, Peter Rechenberg, Felix Redmill, David Robertson, Marko Robnik, Ingrid Russel, A.S.M. Sajeev, Bo Sanden, Vivek Sarin, Iztok Savnik, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Denis Sever, William Spears, Hartmut Stadtler, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Tomasz Szmuc, Metod Škarja, Jiři Šlechta, Zahir Tari, Jurij Tasič, Piotr Teczynski, Ken Tindell, A Min Tjoa, Wieslaw Traczyk, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Kanonkluk Vanapipat, Alexander P. Vazhenin, Zyunt Vetulani, Olivier de Vel, John Weckert, Gerhard Widmer, Stefan Wrobel, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Robert Zorc

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 21 Number 4 November 1997 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: