processes and tools for decision support

edited by
h.g. sol

IFIP

north-holland

PROCESSES AND TOOLS FOR DECISION SUPPORT

Joint IFIP WG 8.3 / IIASA Working Conference on
Processes and Tools for Decision Support
Schloss Laxenburg, Austria, 19-21 July, 1982

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

PROCESSES AND TOOLS FOR DECISION SUPPORT

Proceedings of the Joint IFIP WG 8.3 / IIASA Working Conference on Processes and Tools for Decision Support
Schloss Laxenburg, Austria, 19-21 July, 1982

edited by

Henk G. SOL
University of Groningen
The Netherlands

1983

NORTH-HOLLAND PUBLISHING COMPANY
AMSTERDAM • NEW YORK • OXFORD

ISBN: 0 444 86569 1

Published by:

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

Sole distributors for the U.S.A. and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY, INC.
52 Vanderbilt Avenue
New York, N.Y. 10017

PRINTED IN THE NETHERLANDS

PREFACE

IFIP Technical Committee 8, dealing with the subject of 'Information Systems', was founded in 1975. It now has three Working Groups, established in 1976, 1977 and 1981 respectively.

> WG 8.1 – Design and Evaluation of Information Systems
> WG 8.2 – The Interaction of the Information System and the Organization
> WG 8.3 – Decision Support Systems

The Working Group on Decision Support Systems has as SCOPE

> Development of approaches for applying information systems technology to increase the effectiveness of decision-makers in situations where the computer can support and enhance human judgement in the performance of tasks that have elements which cannot be specified in advance.

The AIMS of the Working Group are

> To improve ways of synthesizing and applying relevant work from reference disciplines to practical implementations of systems that enhance decision support capability. Reference disciplines include information technology, artificial intelligence, cognitive psychology, decision theory, organizational theory, operational research and modeling.

Initiatives to this Working Group were taken during a three day meeting on Decision Support Systems, held at the International Institute for Applied Systems Analysis (IIASA) from 23-24 June, 1980. The framework for research on decision support systems, which came out of that meeting, was pursued by the Management and Technology Area of IIASA and IFIP WG 8.3 in the preparation of a joint Working Conference on Processes and Tools for Decision Support. This Conference took place at IIASA, Schloss Laxenburg, Austria from 19-21 July, 1982.

The Programme Committee consisted of:

J. Hawgood (Chairman)	UK
R.M. Lee	Austria
R. Scheinin	USSR
H.G. Sol	Netherlands
R.H. Sprague	USA
A. Vari	Hungary
G.R. Wagner	USA

Local arrangements were under excellent control of the Programme Secretary, Ron M. Lee, assisted by Miyoko Yamada, whereas Alec M. Lee acted as General Chairman.

The main objective of the Conference was to bring together the reference disciplines of DSS and the reference experience of people who have actually designed and implemented DSS for real managers, or expert systems for other types of knowledge workers. This theme is approached in the organization of the book by dividing the contributed papers in the four main categories:

1. Those concerned with underlying theory and concepts.
2. Those analysing the DSS-design problem in the abstract.
3. Descriptions of non-specific DSS-generators.
4. Case studies of specific applications.

In the first paper we discuss the various contributions and we address the possible inferences which may be drawn from these.

We hope that this book provides a stimulus to users, builders and toolsmiths who want support for decision support.

HENK G. SOL
Information Systems Research Group
University of Groningen

TABLE OF CONTENTS

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

PROCESSES AND TOOLS FOR DECISION SUPPORT

INFERENCES FOR FUTURE DEVELOPMENTS

Dr.Henk G.Sol

Information Systems Research Group
University of Groningen
P.O.Box 800, 9700 AV Groningen
The Netherlands

1 INTRODUCTION

The change in description of Decision Support Systems (DSS) from 'concept' through 'movement' to 'bandwagon' clearly illustrates the growing interest in the managerial as well as in the research field for decision support systems in their different manifestations. One may expect that a conference on processes and tools for decision support brings together people from practice and research, whose experiences give insight in the direction the bandwagon is likely to go. This brings us to the question how expertise on processes and tools for decision support can be stored in such a way that one can get advice when developing future DSS. This asks for the definition and construction of a knowledge base, into which the expertise can be brought.
In section 2 we address possible frameworks for such a knowledge base. In section 3 we place the papers presented in this book into the chosen framework. Finally, we try to make inferences from the contributed papers.

2 A KNOWLEDGE BASE FRAMEWORK

A useful framework for research on decison support systems is introduced in Sprague[1980]. He discusses the perspective of the end-user, the builder and the toolsmith from which a DSS can be viewed. In accordance with this distinction the concept of a DSS-generator is put forward to bridge the gap between general tools and specific DSS.
Sprague distinguishes as the main components of a DSS a data base, a model base, and an intermediate software system which interfaces the DSS with the user.
Bonczek et al.[1980] introduce a generic framework for DSS where the components mentioned are replaced by the concepts of a language system, a knowledge system and a problem processing system. The language system is the sum of all linguistic facilities made available to the decision-maker by a DSS. A knowledge system is a DSS's body of knowledge about a problem domain. The problem processing system is the mediating mechanism between expressions of knowledge in the knowledge system and expressions of problems in the language system.

In Sol[1982] we argue that it is enlightening to look upon the process of developing a DSS as an ill-structured problem. For problem-solving in general we may identify the activities of conceptualization, problem specification, solution finding and implementation. Especially ill-structured problems demand specific emphasis on the activities of conceptualization and problem specification, leading to a good correspondence between the problem situation and the empirically supported model of it. This model is the frame of reference for creating and evaluating alternative courses of action. The efficiency of the design process and the effectiveness of the design are depending on:

- a paradigm or Weltanschauung governing the conceptualization and the problem specification,
- a construct paradigm or modelcycle, expressing in broad terms the order of activities,
- a methodology, as an actual sequence of activities in view of a problem situation, telling what to do in which activity,
- a theory, contributing to the actualization of the modelcycle and the methodology in terms of how the activity is to be performed.

It should be clear that a methodology cannot be discussed and evaluated apart from the Weltanschauung, the construct paradigm and the possible theories involved.
In order to be able to describe the distinction between tools, specific DSS and DSS-generators, the logical components of a DSS, as well as the process of decsion-making, we present Figure 1, which integrates the framework of Sprague, the one of Bonczek et al. and our one.

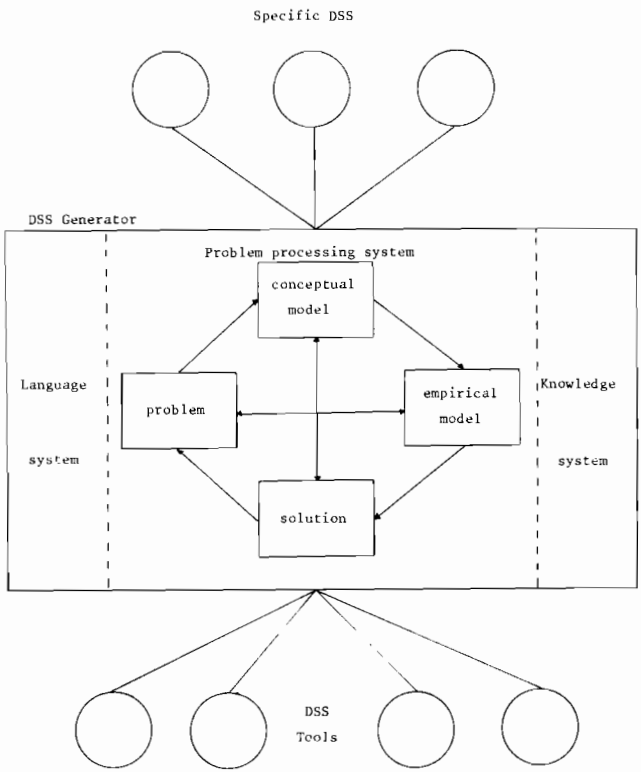


Figure 1

3 GETTING EVIDENCE

We use the framework above to discuss the various papers contributing to the theme 'Processes and tools for decision support'.

THEORETICAL PAPERS

Ćećez-Kecmanović addresses the value of information in decison-making under uncertainty, not only about outcomes, but also about goals. She models the learning process for successive decisions, making explicit the way that this may work against the organization's interests if its goals are not correctly perceived by the decision-maker. The paradigm used implicitly is that reduction of uncertainty leads to better decision-making. Interesting in this paper is that through the concept of structural uncertainty the co-ordination of decision-makers is looked into. The conclusions of her work may turn out to be valuable for the construction of knowledge bases.
In her opinion a specific DSS is mainly used for the provision of information on previous stages in the process of decision-making, especially with regard to the activity of problem-finding.

Lee uses an artificial intelligence standpoint to show the role of logical domain models for generating qualitative inferences. He correctly remarks that semi-structured problems are ones that are not covered by a single decision model. This calls for flexibility as to the language systems and the knowledge system. He argues that procedural as well as non-procedural vehicles are important for the description of the logics of problem domains. This contribution, influenced by developments from the artificial intelligence field, is also recognizable in some recent methodologies for developing information systems, see e.g. Olle, Sol and Verrijn-Stuart[1982].

Coelho supports Lee's paper by outlining the usefulness of the language PROLOG as a big step towards 'programming in logic' and the construction of knowledge bases. He demonstrates the importance of combining descriptive and prescriptive facilities for system description.

ANALYTICAL PAPERS

Young addresses in a tentative style the design of right-brained DSS dealing with the more creative and ill-structured assistance required in scenario-building, problem definition and policy definition. He argues that the right-brained assistance is still an underdeveloped area in top-management. A prerequisite for the facilities which might support these activities is the presence of appropriate data. In his opinion this is not a big problem. Clearly his approach has important consequences for the education of right-brained decision-makers.

Bonczek, Holsapple and Whinston discuss the capabilities of a language to capture expert's knowledge about model usage. They introduce the notion of an interactive DSS development system. Data

structures and models are described in a generic way. They specifically address the problem recognition capability of a problem processing system to select and formulate automatically a model. Therefore, they introduce the concept of an evaluation operator, in addition to the relational algebra they put forward. They focus on features of a problem processing system in specific situations. They argue that the generalizability of these features is still questionable.

Bosman argues that there is no hard core for research on decision processes in organizations. He doubts wether the paradigm of loosely coupled systems or nearly decomposability as applied in many DSS-contributions can play the role of a generic mechanism for the construction of theories on organizational decision-making. He gives some evidence that co-ordination problems in organizations cannot be solved through aggregation of data. He therefore makes a plea for the application of inquiry systems, which can keep track of the various stages in the process of problem solving. He concludes that for the construction of DSS, not only a Weltanschauung, a construct paradigm and a methodology are to be considered, but that a development philosophy is at least as important. He sketches an outline of an expert system to deal with co-ordination problems in organizations.

Methlie lookes into various ways to structure the components of DSS in an organization. He concludes that the effectiveness of DSS is determined to a large extent by the technological infrastructure and the socio-political environments of the decision-makers. His conclusions are useful for the development of future DSS-generators.

NON-SPECIFIC DSS-GENERATORS

Fox describes the Carnegie-Mellon Robotics Institute's 'Intelligent Management System' to explore ill-structured problems by means of heuristic problem-solving techniques. Using contributions from artificial intelligence and management science, he introduces a modeling system with a model of an organization in its core. His modeling language is closely related to the area of process simulation. He models application domains in a system description language. He connects simulation tools with the processing of data about ongoing operations on a workfloor.

Brookes is using office automation technology as a DSS-generator, paying particular attention to the use of soft information contained in managers' letters, messages and conversations. He introduces a design principle for a corporate intelligence system. He argues that in storing and dealing with information we should take care of the qualification aspects or the contextual constraints. His paper relates to the work on logical domain modeling. The efficiency problems of dealing with large data bases of soft information are still to be looked into.

Maes, Vanthienen and Verhelst describe their experiences with the decision-table generator PRODEMO. They mainly use this package as a tool to describe procedural decision-making in well-structured situations. Their work contribute to the elicitation of decision

rules in the construction of DSS.

Dickson addresses the replacement by 1990 of the standard telephone in the manager's office by a management support facility. He distinguishes the universally needed features of such a facility. Next to the functions of problem finding and problem solving, he identifies the much neglected one of conveying a decision. He clearly points out that our knowledge of the nature of decision-making processes is still underdeveloped.

Klein and Manteau describe the notion of a decision support center supported by their DSS-generator OPTRANS. They present the concepts of Optrans tailored to a great many organizational functions at various hierarchical levels. Interesting is their choice for the entity-relationship description form for the specification of applications. In their modeling tool they have flexibility to apply predicates in functions. Above, they are able to define a specific organizational context. The key feature of Optrans is that the user really assembles his own system from the modules provided.

Bergquist and McLean show that the programming language APL can be the basis for a powerful and flexible DSS-generator, also allowing implementation and modification by the user himself. This IDAMS system is very complete with regard to data base and model base use. Also the meta-definitions of data are easily accessible and modifiable.

CASE STUDIES

Humphreys, Larichev, Vari and Vecsenyi give a report on a multi-national project comparing the development histories of four systems designed to support research and development decisions. They describe the rounds and stages in the different development paths and the roles played by the various participants. The DSS-tools applied are mainly build around decison tree analysis. They point at the discontinuity between functions of DSS at various levels in the organization.

Bohanec, Bratko and Rajkovic explain their expert system for decison-making in the Jugoslav self-management context. They relate the choice of alternatives to the calculation of utilities. They apply fuzzy functions derived from the users' subjective opinions expressed in words. They show that implementation of expert systems requires a new way of thinking in many decision-making situations. They give a nice example of an iterative strategy for developing such an expert system.

Takkenberg introduces a system generator combining linear programming and simulation features. He shows the importance of describing the actual situation as a first step in the development of a DSS, because this is the only frame of reference for accepting new alternatives. He addresses the elicitation of the expertise of decison-makers before discussing the possible tools that might support them.

4 CONCLUSIONS

What inferences may be drawn from these papers?

1. As to the paradigm or Weltanschauung applied, we may conclude that the one of loosely-coupled systems is still prevailing. It is striking how little thought is given to the elicitation of assumptions governing the construction of DSS. Only Bosman and Takkenberg discuss this explicitly.
2. The modelcycle behind most contributions can be characterized as a Singerian one, trying to integrate scientific, ethical and esthetic modes of thought in a synthetic, interdisciplinary way. However, most approaches are starting from the premise that more and better information will also lead to better decisions. The availability of data and appropriateness of data for decision-making in organizations is not much questioned, which might be in contradiction with the Singerian point of departure.
3. As to a methodology for developing DSS, it is difficult to identify a generic framework. It is clear that an evolutionary or iterative approach is prevailing. The case studies give some evidence on how DSS possibly should evolve. We still need more insight in the process of actually developing DSS.
4. As to the possible theories for constructing DSS, we observe that co-ordination of decison-making processes is still a neglected topic.
   Little attention is given to the process of problem-solving and to possible ways of keeping track of it. One may expect that knowledge bases and expert systems can be made operational to realize this.

What we still need is an expert system to store evidence on the various activities in developing DSS, in order to be able to support builders, users and toolsmiths in the construction of future DSS.

REFERENCES

[1] Bonczek, R.H., Holsapple, C.W., Whinston, A.B., Foundations of Decision Support Systems, Academic Press, 1981.
[2] Olle, T.W., Sol, H.G., Verrijn-Stuart, A.A.(eds.), Information Systems Design Methodologies: A comparative Review, North-Holland, Amsterdam, 1982.
[3] Sol, H.G., Simulation in Information Systems Development, PhD Thesis University of Groningen, 1982.
[4] Sprague, R.H., A Framework for Research on Decision Support Systems, in: Fick, G., Sprague, R.H.(eds.), Decision Support Systems: Issues and Challenges, Pergamon Press, Oxford, 1980.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

THE VALUE OF INFORMATION IN DECISION-MAKING

Dubravka Ćećez-Kecmanović

Elektrotehnički fakultet
Univerzitet u Sarajevu
Yugoslavia

Our research is devoted to the analysis of information in decision-making processes. For this purpose we develop descriptive models of decision-making with various types of uncertainties.Then we can examine the influence of information on decision-makers' behavior in different situations. Decision support systems with various focuses of information are observed within two experiments and the resulting impact of information on the decision-makers' performance are measured.

1.INTRODUCTION

Before developing or improving methods for creation and evaluation of decision support systems it seems reasonable to investigate the very nature of the role of information in decision-making processes. We believe that the more we know about information usage and effects and their possible values in decision-making, the better methods we can create. More specifically, developing a model of information value in decision-making could contribute to the development of better methods for creation of decision support systems and their evaluation.

Our main concern in this paper is descriptive modelling of decision-making processes. We start from a rather simple model of decision-making and expand it to make it more realistic. The restrictions built in the model of decision-making processes directly limit the extent to which information usage and effects can be examined. Although the model in its present stage of development still contains some simplified assumptions, it enables us to experiment with different decision situations and examine information effects and values. The model is open to further refinement and extensions.

In a decision-making process decision-makers have to determine what courses of action are available to them and they have to predict the likely outcome of these actions. Furthermore, to be able to choose the best action, they have to evaluate possible contributions to their goals and desires. In this process, decision-makers behave according to their knowledge of the decision situation.

There are many uncertainties involved in a decision situation. For a particular state of nature, there is uncertainty as to the available set of actions and likely outcomes of these actions and as to the probabilities of outcomes after the execution of an action, respectively called structural and relational executional uncertainty. The model of decision-making with relational executional uncertainty was studied extensively by Yovits, Rose, Abilock, Gavin and Whittemore at Ohio State University [4], [5], [6]. Their model is briefly described in section 3.

When decision-makers do not know precisely the desires and related goals they want to achieve, nor how action-outcome pairs are related to these goals and desires, there exists goal uncertainty. Modelling decision-making with goal uncertainty and related influence of information is the main topic of our research , which is reported in section 4.

The absence of the uncertainty of the external uncontrollable conditions (called the state of nature

uncertainty) in our model of decision-making is one of its restrictions. However, some considerations on the possible impact of state of nature uncertainty on decision-making are discussed.

The only way to learn about a decision situation is to be informed about the results of decisions and about possible future decision situations. Increased knowledge of a decision situation reduces uncertainties in decision-making. In this way information is related to the reduction of different uncertainties. Furthermore, the reduction of uncertainties results in better decisions and thus improves decision-makers' performance. The change in decision makers' performance depends on the information value. In order to measure the information impact on decision-making, i.e. the change of uncertainties and the resulting change of performance, we apply the measures, section 5., developed by Yovits et al.

On the basis of the theoretical model of decision-making processes with executional and goal uncertainty we develop a simulation model. It permits the examination of information effects and value in different decision situations, see section 6.

2.DECISION-MAKING SITUATION

A part of an organization is called a decision-making entity if there exists:

a) a set of alternative courses of action $A^s = \{a_1,..,a_m\}$ for a state of nature s,
b) a set of possible outcomes of these actions $O^s = \{o_1,..,o_n\}$ for a state of nature s,
c) relations between actions and outcomes: what consequences will follow from one course of action and with what probability, for a state of nature s,
d) a group of people that chooses one course of action at a time, according to their desires and goals, based on their knowledge of a decision situation, and
e) a decision support system providing information to the decision-makers about a decision situation.[1)]

The relations between the courses of action and outcomes, for a state of nature s, perceived by decision-makers are described by a probability matrix of a size m by n:

$$W^s = \begin{bmatrix} w_{11} & \cdot & \cdot & w_{1n} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ w_{m1} & \cdot & \cdot & w_{mn} \end{bmatrix} \qquad (2.1)$$

where w_{ij} denotes the decision-makers' probability that execution of action a_i will result in outcome o_j.

A course of action and its likely outcome may have some value to the decision-makers. Their perceived value of each action-oucome pair, for a state of nature s, is described by a value matrix of size m by n:

$$V^s = \begin{bmatrix} v_{11} & \cdot & \cdot & v_{1n} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ v_{m1} & \cdot & \cdot & v_{mn} \end{bmatrix} \qquad (2.2)$$

For one state of nature s, the matrices W^s and V^s represent the decision-makers' model of a decision situation. This model does not necessarily correspond to reality. The real decision situation, in a state of nature s, is described by the actual matrices $W^{s\star}$ and $V^{s\star}$: $w^{\star}_{ij}$ is the actual probability of occurrence of outcome o_j after the execution of action a_i, and $v^{\star}_{ij}$ is the actual value achieved when execution af action a_i resulted in outcome o_j. Note that the matrices perceived by decision-makers may differ from the actual ones in size and content.

In a given period of time $[t_1,t_2]$ a number of states of nature can occur. The set of possible states of nature for that period of time is denoted by $S = \{s_k\}$, k=1,..,Q, where each state of nature has some probability of occurrence. For the given period of time three-dimensional matrices $W^{\star} = \{w^{\star}_{ijk}\}$ and $V^{\star} = \{v^{\star}_{ijk}\}$ describe an actual decision situation. This model of a decision-making situation enables explicite consideration of various uncertainties. Since we limit our examination of information impact on decision-making processes to executional and goal uncertainty, we assume a single state of nature.

3.INFORMATION RELATED TO EXECUTIONAL UNCERTAINTY IN DECISION–MAKING

The uncertainty of available courses of action and possible outcomes of these actions for a particular state of nature is called structural executional uncertainty. The uncertainty of outcome of a given course of action is called relational executional uncertainty. We start in this section from a simple model of decision-making, assuming only relational executional uncertainty, see Yovits at al.,[4],[5],[6].

3.1. Selection of actions

In order to describe decision-makers' behavior Yovits at al. used the expected value model. Decision-makers estimate the expected value, EV, for the i-th course of action as the sum of the outcome values –resulting from a course of action– weighted by the probability of that outcome:

$$EV_i(t) = \sum_{j=1}^{n} w_{ij}\, v_{ij} \qquad i=1,..,m \tag{3.1}$$

If one knows the actual matrices for a state of nature s , $W^{s\star}$ and $V^{s\star}$, one can calculate the actual expected value for the i-th course of action:

$$EV_i^{\star}(t) = \sum_{j=1}^{n} w_{ij}^{\star}\, v_{ij}^{\star} \qquad i=1,..,m \tag{3.2}$$

Decision-makers in general do not know the actual expected values.Their expected value estimates depend on their knowledge about a decision situation. Rational decision-makers, starting from initial estimates, will learn from information, and eventually approach the actual ones, after an ample period of time:

$$EV_i(t) \longrightarrow EV_i^{\star}(t) \quad \text{as} \quad t \longrightarrow \infty \tag{3.3}$$

Now we come to the question of how decision-makers choose a course of action based on their estimates of expected values.

Unlike "classical" expected value decision theory, where the decision-makers choose the action with the greatest expected value, the autors suggest that the selection of actions depend not only on expected value, but also on the decision-makers' confidence in their estimates. The probability of selection of an action is defined as:

$$P(a_i) = \frac{EV_i^{C}}{\sum_{k=1}^{m} (EV_k)^{C}} \tag{3.4}$$

where C is a confidence function representing the decision-makers' confidence in their knowledge of a decision situation. The confidence will increase with experience,i.e. with the number of similar decisions made:

$$C(\tau) = k_c\, \tau \tag{3.5}$$

k_c is a confidence factor,constant for a group of decision-makers ($0 \leqslant k_c \leqslant 1$) and τ is the number of trials. A trial is a compound execution of all actions executed in proportion to their probability of selection.

3.2. Learning process

After the execution of a selected course of action a_k,at time t, the actual outcome o_j occurs. Since in this model decision-makers do not face structural executional uncertainty (they know A^S and O^S with certainty) and the state of nature does not change, the decision-making process is repetitive. Making similar types of decisions (choosing one course of action among a set of actions A^S), decision-makers learn from the results of their earlier decisions. Comparing the resulting (actual) value of action a_k and outcome o_j that occurred,i.e. $v_{kj}^{\star}$, with their expected value of action a_k, decision-makers update their knowledge of the decision situation.

The learning process within the execution of one course of action k, at time t, resulting in outcome j, is modelled as follows:

$$EV_k(t_1) = [\,1 - \lambda_k(t)\,]\,EV_k(t) + \lambda_k(t)\,v^{\star}_{kj}(t) \qquad k=1,..,m \qquad (3.6)$$

$$0 \leqslant \lambda_k \leqslant 1$$

where λ_k is the learning function dependent on the number of trials:

$$\lambda_k = \frac{1}{k_\lambda \tau + 1} \qquad k=1,..,m \qquad (3.7)$$

k_λ is the learning factor $(0 \leqslant k_\lambda \leqslant 1)$, constant for a group of decision-makers. The more experienced decision -makers become, the less they will learn.

The average learning rule describes the learning process within a trial:

$$EV_k(\tau+1) = [1 - P(a_k)\lambda(\tau)\,]\,EV_k(\tau) + P(a_k)\,\lambda(\tau)\,EV^{\star}_k(\tau) \qquad k=1,..,m \qquad (3.8)$$

Instead of a specific $v^{\star}_{kj}$ for a single execution of action k in (3.6) we have the average value $EV^{\star}_k$ for a trial. Learning is also proportional to the probability of selection.

(3.4) and (3.8) describe the average behavior of the decision-makers in a particular decision situation. For these relationships there is a unique initial point. Assuming that at the beginning the decision-makers have no knowledge of the situation and hence assign equal probabilities to outcomes, $w_{ij}=1/n$, it follows from (3.1) .

$$EV_i(0) = \frac{1}{n}\sum_{j=1}^{n} v_{ij} \qquad i=1,..,m \qquad (3.9)$$

Since the decision-makers with no knowledge have confidence equal 0, the probability of selection of all actions are equal:

$$P(a_i) = \frac{1}{m} \qquad i=1,..,m \qquad (3.10)$$

The initial point is characterized by random choice.

EXAMPLE . In a production unit of an organization different alternatives for increasing the production rate are considered:

- a_1 — increase the number of production hours by 20% overtime work paid extra,
- a_2 — increase the number of production hours by 40% overtime work paid extra,
- a_3 — increase the number of production hours by 20% with new temporary employed workers,
- a_4 — increase the number of production hours by 40% with new temporary employed workers.

The possible outcomes of these actions are:

- o_1 — no increase of production rate
- o_2 — 10% increase of production rate
- o_3 — 20% increase of production rate
- o_4 — 30% increase of production rate
- o_5 — 40% increase of production rate
- o_6 — 50% increase of production rate

The actual probability matrix is:

$$W^{\star} = \begin{bmatrix} 0.05 & 0.40 & 0.50 & 0.05 & 0 \\ 0 & 0.10 & 0.20 & 0.50 & 0 \\ 0 & 0.10 & 0.70 & 0.15 & 0 \\ 0 & 0 & 0.05 & 0.60 & 0.05 \end{bmatrix}$$

Decision-makers do not know the actual probability matrix. Since at the starting point they have no experience, they predict all w_{ij} to be equal 1/6. Decision-makers assign values to action-outcome pairs according to the attainment of their operational goal "to increase the production rate by 20%" and

their open goal " to keep production cost as low as possible". If the production rate goal is attained (outcomes o_3, o_4 , o_5 and o_6) the resulting value for decision-makers is 100 units (in utilities), otherwise the value is 0 (outcomes o_1 and o_2). Production cost depends on both the actions and the outcomes. Assuming that there is no goal uncertainty, decision-makers will calculate actual values of action-outcome pairs as : the value dependent on production rate goal attainment minus the increase of production cost plus a constant. For this example we have :

$$V = V^{\star} = \begin{bmatrix} 50 & 40 & 130 & 122 & 115 & 109 \\ 30 & 20 & 110 & 102 & 95 & 89 \\ 40 & 30 & 120 & 112 & 105 & 99 \\ 10 & 0 & 90 & 82 & 75 & 69 \end{bmatrix}$$

A simulation of the decision-making process is performed with a confidence factor k_c=0.2 and the learning factor k_λ=0.1. Some results are illustrated in table 1. The actual expected values of actions are given in the first column. For τ=0 decision-makers start with random choice (the probability of selection aof all actions is equal 0.25). In the next trial they learn about the actual expected value of actions and change their estimates of the expected values. The probability of selecting all actions is also changed, since decision-makers' estimates are changed and their confidence is increased. Finally, after 45 trials, they learn to choose the best action (tho probability of selection of the best action approaches 1).

		τ = 0		τ =1			τ =45	
a_i	$EV_i^{\star}$	$P(a_i)$	EV_i	$P(a_i)$	EV_i		$P(a_i)$	EV_i
a_1	89.6	0.25	94.33	0.2577	93.15		0.0095	91.1
a_2	94	0.25	74.33	0.2495	79.25		0.0193	93.2
a_3	109.05	0.25	84.33	0.2563	90.51		0.9705	109.01
a_4	79.65	0.25	54.33	0.2365	60.66		0.0007	75.38

Table 1.

4. INFORMATION RELATED TO GOAL UNCERTAINTY IN DECISION–MAKING

The existence of goals is one of the basic characteristics of organizations. Goals form motivational factors to perform organizational tasks. Goals are also a means for steering organizational activities towards the desired results. In any decision-making process the criterion for choice depends on explicitly or implicitly stated goals.

In order to develop a model of the decision-making process under goal uncertainty, we assume that:

a) decision-makers bring their values and goals into the organization and change them when properly influenced,
b) there exists a pluralism of interests in the organization and hence incomplete parallellism of group, intergroup and common organizational goals, and
c) decision-makers behave rationally relative to their ability to recognize a particular goal, to perceive relations between actions and goal satisfaction and their motivation to contribute to the particular goal.

4.1. Existence of goals and evaluation of actions

In order to understand why decision-makers choose a specific action among the available set of actions, one needs to investigate the values assigned to actions. A search process starts from the immediate goals decision-makers want to achieve, in a given period of time, e.g. "to keep production costs below a certain level", "to increase production rate by 20%", "to increase productivity by 10%", etc. Why do they want to achieve these goals? Because they want to reach some higher goals, e.g. "to increase sales by 20%", "to maximize income", "to improve working conditions", etc. This analysis leads us towards higher,more general and less precisely defined goals and desires. Finally we may end up with the question of the ultimate purpose of the organization. Here we can notice many different kinds of

goals, the implicit recognition of the levels of goals (lower/higher goals) and the kind of relationships between the goals (some goals are means for reaching other,higher goals). From this "system" of goals and desires the values of actions are derived. Now we face the questions: what are the goals and desires in an organization and its decision-making entities? How are the goals related to each other ? What are the values of actions relative to the attainement of goals ?

Let us first examine the very existence of goals and desires. The organization is supposed to function on the basis of the goal–assignment process through organization entities. Starting from the highest or ultimate desirables (Langefors [2]) of the organization as a whole,the first precedents, i.e. the desira–bles or goals that contribute to the ultimate ones, are determined. Subsequently, the next level of desirables/goals precedents are determined, and so on. At the lowest level, so called operative goals are identified. This deduction process of goal formation is actually a top down creation of goal structure. In real organization this process is rarely systematic and consistent. There exists another process of goal formation, in parallel. The members of the organization enter their values and goals into the organiza–tion entities. Integration of individual goals and desires within a group results in group goals, intergroup goals and desires, etc. This is an inductive goal formation process.

A decentralized decision-making process in an organization consisting of distinct organization entities, raises the question of how to evaluate actions. Decision-makers in an entity tend to evaluate the action outcome pairs with respect to the contribution to their goals even when these are in conflict with the goals of other entities. Since the value of an action-outcome pair can be different when derived from contributions to different goals, the main question of the evaluation process are what goals or desira–bles are considered and how are the contributions measured ?

4.2. Goal uncertainty

When there is uncertainty as to what are the desirables of an organization and what are the goals to be attained in a given period of time, then the decision-making process contains structural goal uncerainty . The uncertainty of relationships between the actions and their contributions to goals and desirables is called relational goal uncertainty. The existence of structural and relational goal uncerta–inty causes the difference between the decision-makers value matrix and the actual value matrix within the organization.

The desired result of an organization is expressed by the organizational desirables (e.g. survival, growth, long term profit) and goals (e.g. specific market share, income, productivity). Any action-outcome pair, generated from some organizational entity, will in general influence the goal and desirable structure of the organization. If an organization develops one criterion (an ultimate desirable or goal variable) to evaluate actions and outcomes than it is possible to have a unique value matrix for the whole organi–zation V^*.[2)]

Decision-makers evaluate actions and outcomes according to their criteria, based on their goal structure. Decision-makers' value matrix V is, in general, different from the organizational value matrix V^*:

$$V \neq V^* \tag{4.1}$$

since either:

- they do not explicitly recognize the ultimate organizational desirable or goal, against which action-outcome pairs are evaluated, or
- they do recognize that desirable (or goal), but they do not know the actual action-outcome contribution to that desirable or goal (their perceived relationships between the two do not correspond to reality), or
- they do recognize the ultimate desirable (or goal) and they know the actual action-outcome contribution to that desirable (or goal) but they are not willing to contribute to that desirable or goal.

These are possible reasons for ultimate desirable (or goal) uncertainty in an organization. We can extend this reasoning to the other desirables and goals in the organization goal structure.

Decision-makers can change their goal structure when they are informed about the actual results of

their actions : achievements of goals and contributions to desirables. In this respect decision-makers can learn about organizational desirables and goals. This learning process is of quite a different nature than learning to choose better actions (according to defined criteria). Learning about desirables and goals is much slower and it does not automaticlly bring the decision-makers value matrix closer to the actual value matrix. Assuming no other causes for the change of the decision-makers goal structure [3] we shall investigate learning about desirables and goals as a result of information supply.

From this brief discussion of goal uncertainty we can conclude that modelling of goal uncertainty in decision-making has to comprise the description of the goal/desirable structure of the organization and that of the decision-makers, as well as the way decision-makers learn about the goals and change their goal structure.

4.3. Model of goal structure

The model of goal structure intends to describe goals and desirables and their interrelationship without consideration or reference to the process of goal formation. The model has to serve two purposes : 1) to be the basis for calculations of the elements of value matrices and 2) to provide the modelling for goal learning.

From our previous work,[1], we find it convenient to model the goal structure with a graph [4] using goals and/or gesirables as elements and their relations as branches between the elements. The goal structure of the organization is described by a simple graph:

$$S = (G,R) \tag{4.2}$$

which is antisymetric and has no loops (closed paths). G is a set of organization goals and desirables:

$$G = \{G_k \mid G_k \text{ is k-th goal/desirable of organization}, k=1,..,K\} \tag{4.3}$$

R is a set of all relations between the goals and the desirables – elements of the set G:

$$R = \{r_{kl} \mid r_{kl}=(G_k,G_l), G_k, G_l \in G\} \tag{4.4}$$

where the relation:

$$r_{kl} = (G_k, G_l) \qquad \begin{matrix} k=1,..,K \\ l=1,..,K \end{matrix} \quad k \neq l \tag{4.5}$$

represents the direct contribution of the goal/desirable G_k to the goal/desirable G_l (in this case it is common to say that G_k is "lower" and G_l "higher" goal/desirable). If r_{kl} exists than r_{lk} can not exist, hence the graph is antisymetric.

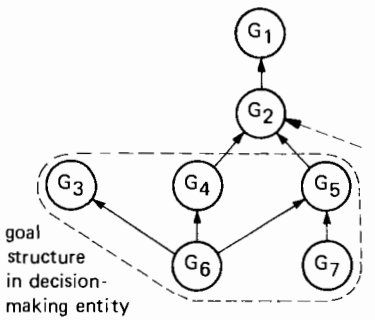
In the model of goal structure, (4.2), (4.3), (4.4), some goals and desirables have special roles,dependent on their place in goal hierarchy. The highest elements in a goal structure – the desirables that are wanted for their own sake – are acalled ultimate desirables . At the lowest level the goals have no precedent, and are usually called operative goals. The attainment of operative goals depend directly on the actions executed and the outcome obtained. The attainment of the intermediate goals depend on the attainment of the precedence goals, and finally on operative goals.

EXAMPLE: In an organization profit is defined as the ultimate desirable, G_1. The desirable income G_2 contributes to the profit. In order to increase income, the goals of an organization entity are defined, for a certain period of time:
– to increase sales (sales variable $\hat{G}_4$ and sales value K_4)
– to keep costs below the level K_5 (cost variable $\hat{G}_5$, cost variable value K_5)
According to these goals, operative goals are defined:
– to attain production rate K_6 (production rate variable $\hat{G}_6$ and production rate value K_6), and
– to keep administrative cost below K_7 (administrative cost variable $\hat{G}_7$ and administrative cost value K_7)
Furthermore, in this organization entity decision-makers have their own local desirable "personal income per worker" G_3, which is dependent on the production rate. The goals of the other entities are not considered.

The goals are defined for a certain period of time, in which production rate should be K_6 in order to

attain sales goal $K_4 = k_{6,4}K_6$, and administrative cost should be below K_7 in order to attain the goal G_5 i.e. to keep total cost below $K_5 = k_{6,5}K_6 + K_7$. If the production rate goal is not attained i.e. $\hat{G}_6 < K_6$ than the sales goal can not be attained and, since the sales contract is broken, the sales goal variable becomes 0, see figure 1.

Recalling the example from section 3. we see that the values of operative goals depend on action-out-come pairs. Nonoperative goals attain values through contributions from lower goals to higher ones, through the goal structure.



$$\hat{G}_1 = k_{2,1}\hat{G}_2$$

$$\hat{G}_2 = k_{4,2}\hat{G}_4 - k_{5,2}\hat{G}_5 + \cdots$$

$$\hat{G}_3 = \begin{cases} K_3 + k_{6,3}K_6 & \text{for } \hat{G}_6 \geqslant K_6 \\ K_3 & \text{for } \hat{G}_6 < K_6 \end{cases}$$

$$\hat{G}_4 = \begin{cases} 0 & \text{for } \hat{G}_6 < K_6 \\ k_{6,4}K_6 & \text{for } \hat{G}_6 \geqslant K_6 \end{cases}$$

$$\hat{G}_5 = \begin{cases} \hat{G}_7 + k_{6,5}\hat{G}_6 & \text{for } \hat{G}_7 \leqslant K_7 \\ \hat{G}_7(1+k_7) + k_{6,5}\hat{G}_6 & \text{for } \hat{G}_7 > K_7 \end{cases}$$

Figure 1. Example of goal structure
(end of example)

Let us describe more formally the calculation of goal variable values in the goal structure. The subset of goals which directly contribute to the goal G_l is given as:

$$g_l = \{G_k \mid r_{kl} \in R,\ G_k \in G,\ G_l \in G,\ k=k1,k2,..,kl, k \neq l\} \tag{4.6}$$

The goal variable $\hat{G}_l$ of the goal G_l is the function of the variables of goals belonging to the subset g_l:

$$\hat{G}_l = f(\hat{G}_{k1}, \hat{G}_{k2}, .., \hat{G}_{kl}) \qquad l=1,2,..,K-B \tag{4.7}$$

(B is the number of operative goals). This goal function exists (whether we know it or not) for each non-operative goal. The goal variable values attained due to the action-outcome pair i,j will be denoted:

$$\hat{G}^{ij}_k \qquad k=1,..,K \tag{4.8}$$

To be able to calculate contributions of an action-outcome pair along the goal structure, (4.2), it is necessary to know a) the goal functions for non-operative goals, (4.7), and b) action-outcome contribu-tions to operative goals.

4.4. Value functions

The actual value of an action-outcome pair is measured by its contribution to the goals and desirables of the organization. If there is one ultimate desirable, let it be G_1, the actual value is equal to the contribution to that desirable value:

$$v^*_{ij} = \hat{G}^{ij}_1 \qquad \begin{matrix} i=1,..,m \\ j=1,..,n \end{matrix} \tag{4.9}$$

If there is more than one ultimate desirable, there is more than one actual value of an action-outcome pair. Avoiding the discussion of the possible reduction of a number of ultimate desirables, we assume, for sake of clarity of further treatment, that is is possible to derive one criterion for the unique actual value assignment, in the organization (the elements of the actual value matrix are calculated from (4.9)).

In order to calculate the elements of the decision-makers' value matrix it is necessary to describe the decision-makers' goals and desirables. Let decision-makers' goals and desirables belong to a set GD :

$$GD = \{G_{d1}, G_{d2}, \ldots, G_{dD}\} \tag{4.10}$$

These can be their personal and group goals and desirables, and some perceived organizational goal and /or desirable. The value of an action-outcome pair for decision-mekers is measured by its contribution to goals and desirables in the set GD.

We assume that decision-makers are rational and share, in one entity, common goals and desires as well as preferences between them. If they have the means to calculate an action-outcome contribution to their goals and desirables, they compute the unique value of this action-outcome pair as the sum of these contributions weighted by their preference factors:

$$v_{ij}(t) = q_{d1}(t)\,\hat{G}^{ij}_{d1} + q_{d2}(t)\,\hat{G}^{ij}_{d2} + \cdots + q_{dD}(t)\,\hat{G}^{ij}_{dD} \qquad \begin{matrix} i=1,..,m \\ j=1,..,n \end{matrix} \tag{4.11}$$

where $q_d(t)$ is the decision-makers' preference function that describes the relative importance of the goal/desirable G_d, at some time t, compared to the other goals/desirables in GD. Preference functions for the elements of GD are different from 0:

$$q_d(t) \neq 0 \quad d=d1,d2,..,dD \tag{4.12}$$

and

$$\sum_{d=d1}^{dD} q_d(t) = 1 \tag{4.13}$$

When decision-makers' goals and desirables are not commensurable, which is usually the case, we assume that decision-makers apply utility functions for goal variables. (4.11) describes the decision-makers' value function, where $\hat{G}^{ij}_d$ can be either the goal variable or the utility function of the goal variable.

Goal uncertainty in decision-making can be identified by analysing the decision-makers' value function. Suppose there is only one goal (or desirable) in the set GD, G_d, so that:

$$q_d(t) = 1$$

in the decision-makers' value function. While making decisions in a large enough number of trials, decision-makers will learn to attain that goal and the uncertainty of that goal approaches 0. In the opposite case, when the goal G_d is not explicitly recognized, i.e. :

$$G_d \notin GD \quad \text{and} \quad q_d(t) = 0$$

its uncertainty is maximal. For any situation between the two previous cases, when G_d belongs to a set GD, $0<q_d<1$, the structural uncertainty of that goal is eliminated, but the relational component of uncertainty still remains.

4.5. Goal learning process

One important reason as to why decision-makers prefer to use lower goals rather than higher, more common ,ones in their criterion, is that the lower goals are perceived as operational and the higher ones as nonoperational (March and Simon (1958) p.156). If decision-makers were informed not only about their goal variable values, but also , about organizational goal variable values, they would be able to use them as operational. For the particular purpose of this work we shall describe how decision-makers learn about goals and change their criterion (value function) when they are properly informed, leaving aside the discussion about the motivational, cognitive and other factors involved.

Consider a decision-making process within the period of time [0,T] in which the number of trials is executed. Let that number of trials be large enough for decision-makers to learn to choose the action with maximum expected value. The probability of selection for that action is equal to 1, and for all other actions is equal 0.

$$P(a_i,T) = \begin{cases} 1 \text{ for that i for which } EV_i(T) = \underset{k}{MAX}\,\{EV_k(T)\} \\ 0 \text{ for all other i} \end{cases} \qquad i=1,..,m \tag{4.14}$$

Here the expected value for decision-makers is calculated from the actual probability matrix and the decision-makers value matrix:

$$EV_i(T) = \sum_{j=1}^{n} w^{\star}_{ij}\, v_{ij}(T) \qquad i=1,..,m \tag{4.15}$$

Replacing $v_{ij}(T)$ with the value function (4.11) for t=T, we obtain:

$$EV_i(T) = \sum_{j=1}^{n} w_{ij}\,[\,q_{d1}(T)\,\hat{G}^{ij}_{d1} + \cdots + q_{dD}(T)\,\hat{G}^{ij}_{dD}\,] = q_{d1}(T)\,EV_i^{d1} + \cdots + q_{dD}(T)\,EV_i^{dD} \tag{4.16}$$

where EV_i^d stands for expected value of goal variable $\hat{G}_d$ for an action i. q(T) denotes the preference function which is constant in the period [0,T]. Since during that period, decision-makers learn to choose the action with maximum expected value, the information about the attained results, at the end of the period, will contain:

$$EV^{d1}, \ldots, EV^{dD}$$

These values resulted from the execution of action i for which $EV(T) = \underset{i}{MAX}\,\{EV_i(T)\}$ They represent the average goal variables attained at the end of the period.

The decision-makers succeed in maximizing their value function, but the question is whether they really want those particular results, or whether they need to revise their preferences in order to get different balances between the goals. If e.g. the decision-makers would like in the next period to attain the average value of goal variable $EV^{d'}$ instead of EV^d, than they will change their preference function for that goal. The change of preference will be proportional to the differences between the desired and obtained average value of the goal variable:

$$\bar{q}_d(2T) = q_d(T) + K_d \frac{EV^{d'} - EV^d(T)}{EV^d(T)} \qquad q_d(2T) = \frac{\bar{q}_d(2T)}{\sum_{d=d1}^{dD} \bar{q}_d(2T)} \tag{4.17}$$

$d=d1,..,dD$

where K_d (0 K_d 1) is the factor describing the decision-makers' willingness to change their preferences. This factor depends on the decision-makers, as well as on other circumstances in the organization (not discussed here). The expression (4.17) describes a kind of learning process : in order to improve the overall result of decisions, decision-makers change their preferences on the basis of information about attained average values of goal variables. This model of goal learning includes learning about a new goal, when $q_d(T)=0$ in (4.17).

Like any other goal, the ultimate organization goals or desirable G_1 may be the subject of learning. If decision-makers change their preference functions so that q_1 attains higher values, the decision-makers' value matrix becomes closer to the actual organization value matrix. When :

$$q_1(t) \to 1 \quad \text{then} \quad v_{ij}(t) \to v^{\star}_{ij}(t) \tag{4.18}$$

If q_1 becomes 1 , then the uncertainty of the organizational ultimate goal/desirable is eliminated and $v_{ij}(t) = v^{\star}_{ij}(t)$,(not a realistic possibility).

The learning process concerning executional uncertainty (described by (3.8)) increases the decision - makers' ability to behave according to the set of their goals. Learning about goals (described by (4.17)) enables them to change the set of goals and their preferences between the goals. Decision-makers learn from information. Here we find a partial answer to the question : how does information become valuable ?

5. MEASURES OF INFORMATION AND DECISION–MAKERS PERFORMANCE

In order to examine the effects of information in decision-making Yovits at al. developed quantitative measures of information quantity, decision-makers' performance and effectiveness as well as information value and effectiveness, [4],[5],[6]. Having defined these measures, we can discuss possible information effects in decision-making processes with various uncertainties.

5.1. Quantity of information

The quantity of information relates information to the uncertainty of choosing the appropriate action.

The quantity of information at time t, I(t), is equal to the ratio of the mean square variance of the probabilities of selection to the square of the mean of these probabilities $\mu(P)^2$:

$$I(t) = \frac{\sum_{i=1}^{m} [P(a_i) - \mu(P)]^2 / m}{\mu(P)^2} = m \sum_{i=1}^{m} P(a_i)^2 - 1 \qquad (5.1)$$

I(t) ranges from 0 to a maximum of m−1. This measure depends only on the decision-makers' probability of selection. When decision-makers learn about the decision situation and change their probability of selection, the quantity of information is changed. The change of the quantity of information due to the receipt of data is named the amount of information in a set of data D :

$$QI(D,t_1) = I(t_1) - I(t) \qquad (5.2)$$

5.2. Decision-makers performance and effectiveness

The quantitative measure which characterizes the performance of decision-makers is the average performance for a trial:

$$AP = \sum_{j=1}^{m} P(a_i)\, EV_i^{\star} \qquad (5.3)$$

This measure normalized with the maximum average performance (obtained for $P(a_i)=1$ for the i-th action for which $EV_i^{\star}=EV_{max}^{\star}$) is called decision-makers' effectiveness :

$$DME = \sum_{i=1}^{m} \frac{P(a_i)}{(EV_k)_{max}} \frac{EV_i^{\star}}{} \qquad (5.4)$$

This measure has no dimensions and ranges from 0 to 1 (for nonnegative $EV_i^{\star}$). At the initial point we have :

$$DME = \frac{1}{m} \sum_{j=1}^{m} \frac{EV_i^{\star}}{(EV_i^{\star})_{max}} \qquad (5.5)$$

The average performance and effectiveness can be measured on different levels e.g. in the decision-makers' entity and on the level of organization.

5.3. Value of information

While the quantity of information is dependent on the decision-makers' current probabilities, the value of information has to depend on their performance. The value of information in a given set of data D, at time t, is defined as a change in the decision-makers' average performance, due to receipt of D :

$$VI(D,t_1) = AP(t_1) - AP(t) \qquad (5.6)$$

Normalized value of information, called the effectiveness of information, in a given set of data D, is the change of decision-makers' effectiveness, due to the receipt of D:

$$EI(D,t_1) = DME(t_1) - DME(t) \qquad (5.7)$$

If decision-makers learn from information, they will improve their performance and effectiveness. Hence the value and effectiveness of information will be positive.

5.4. Information effects in decision-making : some typical curves

Information affects decision-making through the leanning process causing a decrease of uncertainty in choosing the best action. This is measured by the amount of information in a decision situation. A decrease of the uncertainty in choosing the best action results in improvement of the decision-makers' performance and effectiveness. For a simple model of a decision-making process, with relational executional uncertainty, typical curves, depicting information quantity, decision-makers' performance and effectiveness, are illustrated in Figure 2. After a sufficient number of trials the quantity of information is maximum (I=m−1) and the average performance of the decision-makers reaches its maximum and their effectiveness reaches 1.

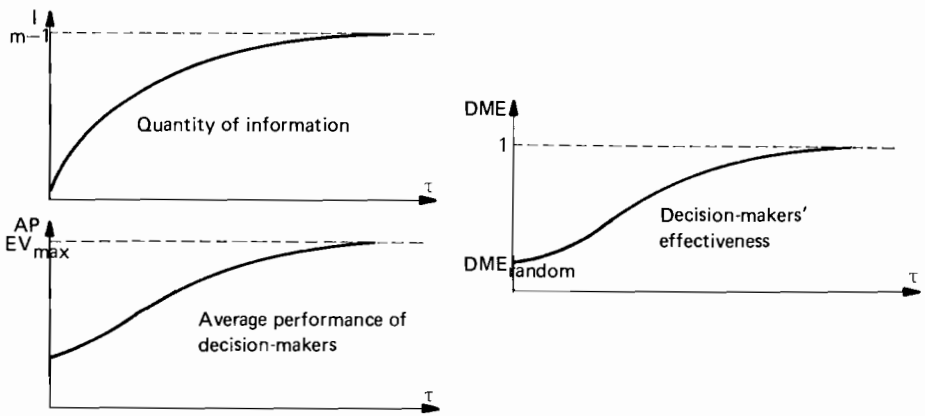


Figure 2. Typical curves for decision-making processes with relational executional uncertainty

The learning process within this model of decision-making is based on information about the results of prior decisions. Instead of using information about past events directly, decision-makers can be supplied with predictions of a decision -making situation. Predictions are derived from the decision-makers'past experience (internal information) as well as from other peoples' experience (external information). The prediction information accelerates the learning process, so their effect is equivalent to a number of trials in (3.8). It is illustrated in Figure 3.

If structural executional uncertainty is entered into the model, which is necessary for non—repetitive decision-making models, the learning process may take some other forms. Structural uncertainty degrades decision-makers ' performance. Searching for new actions can be urged by unsatisfactory results (past in—formation) and predictions about future possibilities. As a result of learning, structural executional uncer-tainty can be removed (positive derivative of learning curve), which is illustrated in Figure 4.

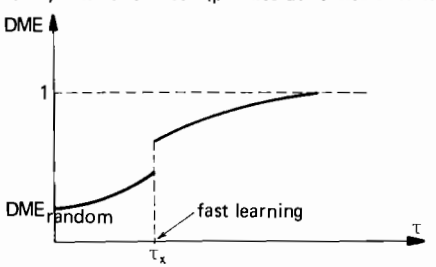


Figure 3. The effect of predictive information

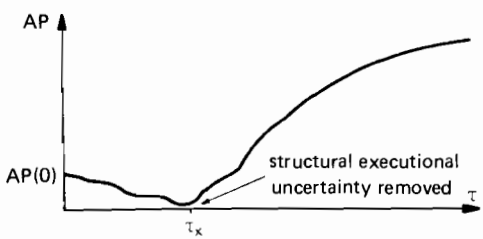


Figure 4. The effect of structural executional uncertainty

If the state of nature does not change for a long enough period of time, the decision-makers will hopeful-ly learn an available set of actions and likely outcomes. But when the state of nature changes, the action and output sets are likely to change as well. The change of these sets causes a negative step in decision-makers' performance, (Figure 5a). The more frequent the change of the state of nature, the more unpre—dictable and creative decision-making will be. Only accelerated learning can cope with a fast changing environment, (Figure 5b).

Goal uncertainty in decision-making causes the difference between the decision-makers' value matrix and the actual organization's value matrix. As a result, a system supplying information for decision-making processes can have different values from different points of view. In Figure 6a the value for decision-mak-ers in one entity is positive, and at the same time the value for the organization is negative (measured by the change of organization performance). The other example, in Figure 6b also shows the difference between these two values, but here both values are positive.

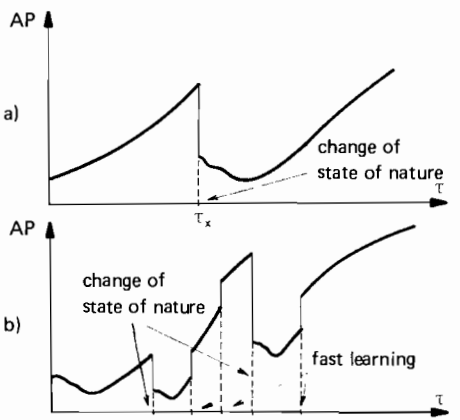


Figure 5. Effects of the change of state of nature and learning processes.

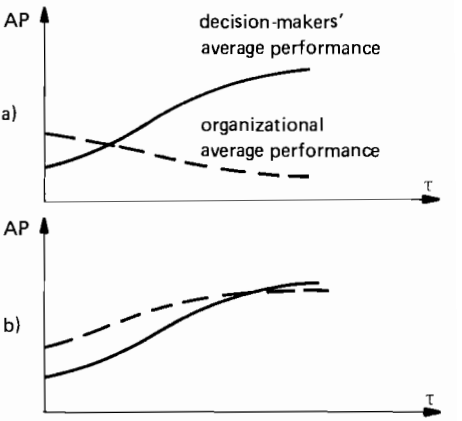


Figure 6.Effects of goal uncertainty and goal learning process

6. SIMULATION MODEL AND EXPERIMENTS

The descriptive model of a decision-making process enables us to analyse the effects and value of decision support systems. Therefore we have developed a simulation model which permits experimentation with different situations on decision-making and observation of the resulting information effects and value. This model can also be used to test how various characteristics and restrictions in decision-making processes limit possible information value, and how much the value of a decision support system depends on certain characteristics of information.

6.1. Simulation model of decision-making

The simulation model in its present stage is shown in Figure 7. It includes a decision-making process with executional and goal uncertainty for one state of nature, together with the possibility to change the state of nature. In an organization entity decisions are made (selections of actions a_i) based on information about outcomes and attained and predicted results, which are supplied by the decision support system. In a stationary situation (no change of state of nature) this information causes the decrease of executional uncertainty and improves decision-makers' performance. If decision-makers are not satisfied with the results – the actual and predicted attainment of their goals – they need to investigate either the goals themselves or the means to reach them. If the goals are satisfactory, then probably structural executional uncertainty is the cause of failure. Decision-makers have to search for new actions and predict likely outcomes of these actions. As a result, new sets A^S and O^S are entering the block of selection of actions.

If the goals are found to be unsatisfactory, than the goal learning process is activated. This process can also be activated by other goal setting processes in the organization. Goal learning results in the change of the decision-makers' set of goals GD and/or preferences between these goals . In this way the information influence on goal uncertainty is simulated.

The block of executing actions contains the actual probability matrix , which is changed when the state of nature changes. Information about actual outcomes of actions goes to the decision support system, where the average results for a trial and decision-makers' performance and effectiveness are calculated. The decision support system produces predictions about future results based on stored information and estimation models. Since the state of nature affects calculations, the decision support system also acquires information about the state of nature.

Experiments can be made with different levels of information support:

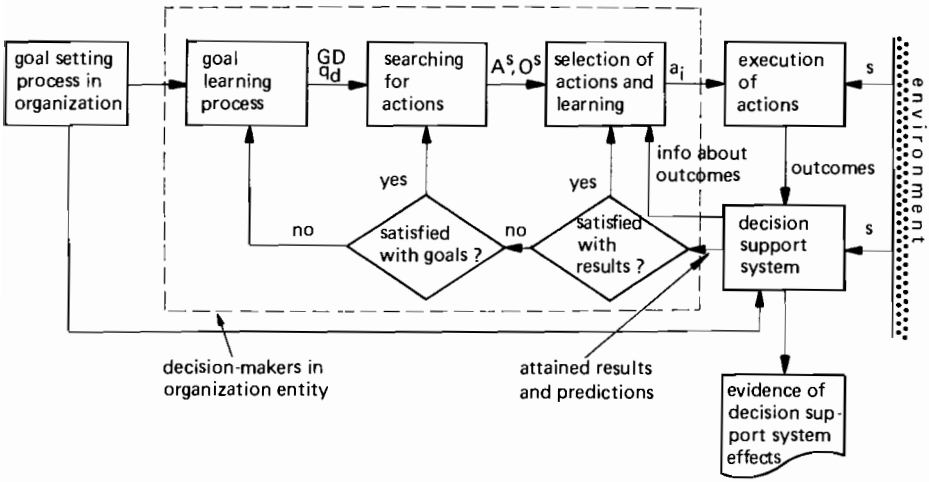


Figure 7. Simulation model of decision-making process

1. level – information about outcomes,
2. level – information about the attainment of goals and the contribution to desirables
3. level – information about predicted future results.

At the levels 2. and 3. different focuses of information can be designed : A decision support system can inform about the attainment of operative goals and/or higher goals of decision-makers and/or higher goals of the organization etc.

The quality of information supplied by a decision support system can be the subject of change. E.g. the effect of information delay or, the effect of information age or of timeliness and accuracy on decision-making can be examined. This general simulation model can be adapted to describe any real decision-making entity or organization by specifying its elements.

Note that state of nature enters our model only in terms of change of actual sets $A^{s\star}$ and $O^{s\star}$ and the size and content of matrices $W^{s\star}$ and $V^{s\star}$. State of nature can be observed, data gathered and predictions of future states performed. State of nature uncertainty and the related role of information are straightforward extensions of this model.

6.2. Simulation experiments

For our simulation experiments we used the example of an organization with a simple goal structure , described in section 4. (see Figure 1.). Decision-makers in the organizational entity do not face structural executional uncertainty and, for the given period, they consider 4 alternative courses of action and 6 likely outcomes. The effect of information on decision-making with relational executional uncertainty and goal uncertainty are examined in two simulation experiments.

In the first experiment the decision-makers' value function is changing, as shown in Table 2. In case a) decision-makers do not know the organization goal structure. They make their decisions according to their particular desirable G_3–personal income. The simulation results are shown in Figure 8a. The curve of decision-makers' effectiveness shows how they learn to maximize their desirable during the period [0,50]. At the same time they affect the organization desirable G_1–profit of the organization, in the opposite direction, without knowing it (no information about it). The resulting effectiveness of the information

1. EXPERIMENT	GD	PREFERENCE FUNCTIONS
case a)	G_3	$q_3 = 1$
case b)	G_3, G_4	$q_3 = 0.6 \quad q_4 = 0.4$
case c)	G_3, G_4, G_2	$q_3 = 0.4 \quad q_4 = 0.3 \quad q_2 = 0.3$

Table 2.

supplied by the decision support system in the period [0,50] for the decision-makers is:

$$EI_{DM}(DSS,50) = DME(50) - DME(0) > 0$$

and for the whole organization is:

$$EI_{OR}(DSS,50) = ORE(50) - ORE(0) < 0$$

In case b) decision-makers include G_4, which is their local goal that contributes to the organization desirable G_2, in their preferred set of goals. They are also informed about the attainment of this goal. For specific preference functions (see Table 2.) the resulting effectiveness curves are illustrated in Figure 8b. Although organization effectiveness is higher than in case a), it is still less than the starting effectiveness, so the effectiveness of information is negative from the organization point of view. In case c) the decision-makers recognize the organization desirable G_2. They are informed about the contribution to this desirable. Their value function contains G_3, G_4 and G_2 with preferences given on Table 2. In this case the change of effectiveness for the decision-makers and for the organization is positive, but different (see Figure 8c).

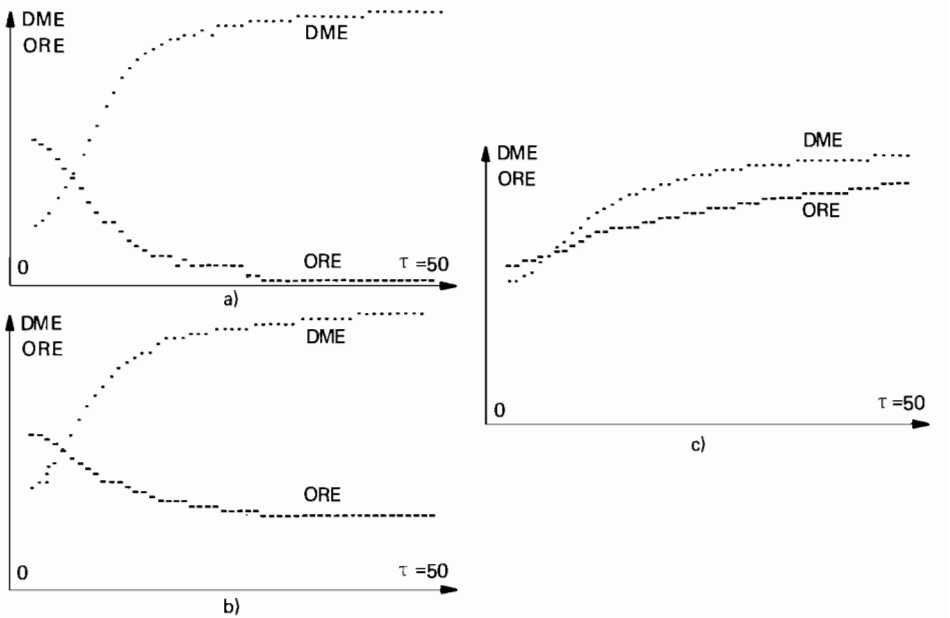


Figure 8. The results of the 1. experiment

In the second experiment a decision-making process with a changing focus of information is simulated. Within the same simulation run, the decision support system supplies information about the attained values of different goals and desirables, according to Table 3. In this example the information influences the goal learning process, which in turn, causes the change of the decision-makers' value function. Figure 9. shows the resulting change in the average performance of decision-makers and in the average performance of the organization.

2. EXPERIMENT	focus of	PREFERENCE FUNCTIONS
period	information	
1. $0 \leqslant \tau < 20$	$\hat{G}_3$	$q_3 = 1$
2. $20 \leqslant \tau < 40$	$\hat{G}_3, \hat{G}_4$	$q_3 = 0.6 \quad q_4 = 0.4$
3. $40 \leqslant \tau < 60$	$\hat{G}_3$	$q_3 = 1$
4. $60 \leqslant \tau < 80$	$\hat{G}_3, \hat{G}_4, \hat{G}_2$	$q_3 = 0.3 \quad q_4 = 0.5 \quad q_2 = 0.2$
5. $80 \leqslant \tau \leqslant 100$	$\hat{G}_3, \hat{G}_4, \hat{G}_2$	$q_2 = 1$

Table 3.

In the first period the decision-makers' value function contains only their desirable G_3. During this period the decision-makers' performance increases (as a result of executional learning) but the organization performance decreases. In the second period the decision support system informes about the attained values of G_3 and G_4. As a result of the goal learning process decision-makers change their function so that it includes both G_3 and G_4. We see that in this period the quantity of information is increased, the decision-makers' performance is highly improved and the organization performance is slightly improved. In the third period the decision support system ceased to supply information about the attainment of goal G_4. Although the decision-makers are willing to include G_3 and G_4 in their prefered set of goals, they can not take G_4 into their value function because of the lack of information. This causes a decrease in the quantity of information and also degrades the decision-makers' performance and the organization performance. In the next two periods new goal learning processes take place in such a manner that finally, the decision-makers' value function equals the organization value function. This means that the uncertainty of organization goal G_1 is eliminated.

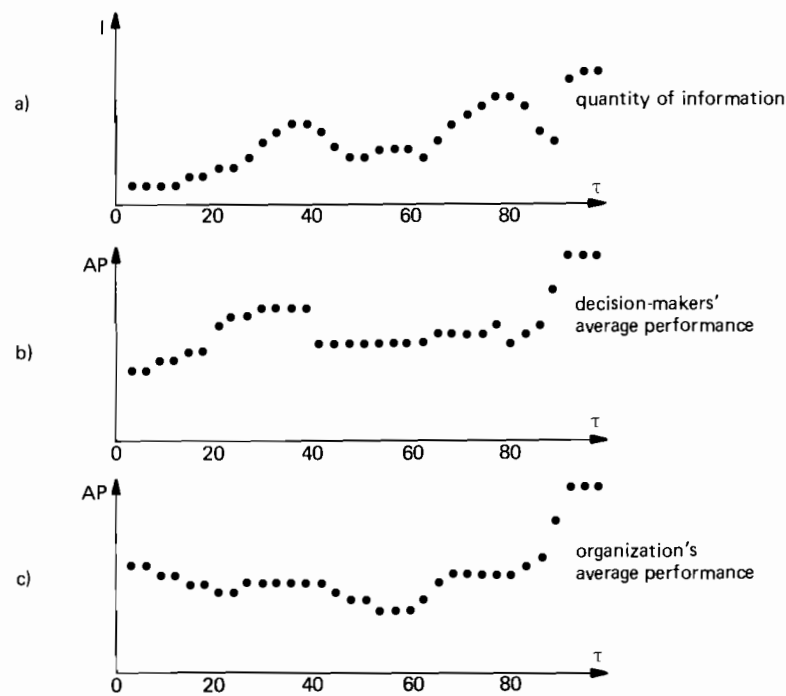


Figure 9. The results of the second experiment

7. CONCLUDING REMARKS

The presented model of decision-making assumes two types of uncertainties; executional uncertainty and goal uncertainty. Through the learning processes information decreases uncertainties in decision-making (the change measured by the amount of information). The decrease of uncertainty results in the improvement of decision-makers' average performance. The change of performance is the measure of information value. Information usage and value within the model of decision-making is analysed.

The simulation model of a decision-making and a decision support system is developed. The model in its present stage permits the examination of information influence on decision-making with executional and goal uncertainty. The model of a decision support system can produce different kinds of information : about outcomes and attained results (past information) and predictions, with a different focus of information. Hence it is possible to simulate different decision situations and observe the resulting quality of the decision-making process.

Although the uncertainty of state of nature is not included in the model, it is possible to experiment with the change of state of nature and examine the effects on the decision-making process. In this way a non–repetitive decision-making process can be simulated. It would be of interest to extend the model with the state of nature uncertainty and related information influence.

The presented model aims to describe the decision-making process and information usage. Different hypotheses about possible decision support can be tested, and likely locations of information value can be identified. The value of information from different points of view can be measured.

FOOTNOTES

1) Some underlaying assumptions are not mentioned, like:
   – whenever the choice among a set of actions is available, the decision-makers are capable of choice,and
   – A^S and O^S are finite sets, the courses of action and outcomes are not "given", they are conceptual constructs, which decision-makers investigate, create and learn about, for each state of nature.

2) In the case of multiple ultimate desirables or goals an organizational value matrix would not be unique. Nevertheless, if it is possible to judge whether an action is more or less valuable than the other, some implicit criterion is applied. This criterion may be some utility function dependent on the number of ultimate desirables or goals.

3) March and Simon mention, among others, communication within the subgroups, division of labor, focus of information, etc. March and Simon (1958),p. 154.

4) We found the graph theory to be convenient but not necessary. The model can be described in any other formal language.

REFERENCES

[1] Ćećez-Kecmanović, D., Model evaluacije informationih sistema, Ph.D. Thesis, Fakulteta za elektrotehniko, Univerza v Ljubljani (1979).

[2] Langefors, B., Desirables, Values and Goals. Goal Analysis, Royal Inst. of Technology, University of Stockholm, TRITA-IBADB 1038, (1974/75).

[3] March, J.G., and Simon, H.A., Organizations, (John Wiley and Sons, 1958).

[4] Yovits, M.C., Rose, L.L. and Abilock, J.G., Development of a Theory of Information Flow and Analysis, in: Weiss, E.C.,(ed.), The Many Facets of Information Science, (Westview Press,Col. 1977).

[5] Yovits, M.C., and Rose, L.L., Information Flow and Analysis . Theory, Simulation and Examples, Tech. Report OSU-CISRC-TR-78-5, Ohio State University (1978).

[6] Whittemore, B.J. and Yovits, M.C., A Generalized Conceptual Development for the Analysis and Flow of Information, JASIS, Vol. 24, No. 3, (may-june 1973).

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

EPISTEMOLOGICAL ASPECTS OF KNOWLEDGE-BASED DECISION SULPORT SYSTEMS

Ronald M. Lee

International Institute for Applied Systems Analysis
Laxenburg, Austria

Knowledge-based decision support applications differ from those typical of artificial intelligence expert systems in their open-ended, evolutionary character and need to coordinate with other systems resources, such as organizational databases and quantitative analysis routines. While knowledge representation machinery is becoming available, the corresponding formalization of managerial/administrative knowledge needed for DSS application is still lacking.

This entails problems of an epistemological nature, identifying the foundational concepts of business. An abstract framework based on formal languages and denotational semantics is proposed, and ontological issues are identified.

Keywords: decision support systems, knowledge representation, knowledge-based systems, applied epistemology, denotational semantics

INTRODUCTION

The influence of artificial intelligence (AI) in decision support systems research has now become an identifiable trend. This draws mainly from AI work in knowledge representation and expert systems. The book by Bonczek, Holsapple and Whinston (1981) provides a good background reference.

The question arises as to the difference between a 'knowledge-based' DSS which uses AI type knowledge representations and an AI expert system using similar mechanisms. The basic distinction is in the system's objectives. An expert system seeks to replicate, hence replace the abilities of a human expert in specific problem domain. A knowledge-based DSS on the other hand seeks to assist a human (manager) by taking over the more structured parts of a larger, only partially formalizable, problem domain.

It is here that the basic concerns of this paper arise. Expert systems typically involve a closed-world assumption; the problem domain is circumscribed, and the system's performance is confined within those boundaries. In DSS contexts, on the other hand, the world is open. A knowledge-based DSS must be adaptable and extendable to meet the evolving needs of the user and changing conditions in the environment.

More importantly, it is clear that DSS's oriented towards individual users are only a special case of the much broader problem of aiding *organizational* decision

processes. This raises the important problem of interactions between knowledge representations, an aspect largely ignored in AI.

Why is this a problem? The reason is that knowledge representation schemes, e.g., semantic net formalisms, various forms of predicate calculus, have been designed to be *general purpose*, be applicable in any variety of subject areas. Thus, each new effort at knowledge base construction must essentially start from scratch, and the semantic elements chosen tend to be ad hoc, specific to the immediate problem at hand. Consequently, efforts to extend or modify the knowledge-base for changes in the problem scope or definition, and attempts to interface the knowledge-base to other knowledge-bases, databases, etc. are usually frustrated by semantic incompatibilities. (Similar criticisms apply to the design of databases, leading to the semantic difficulties of database translation.)

On the other hand, the contention here is that managerial applications do have certain commonalities (they must or business schools would have nothing to teach), and that these commonalities, properly formalized, can guide and discipline the design of knowledge bases in managerial domains. The issue becomes one of *epistemology* — seeking the basic semantic foundations upon which managerial knowledge can be constructed.

In the sections to follow, the potential role and character of a knowledge-base in a DSS is discussed. The use of knowledge bases in DSS applications poses two types of problems not typically in artificial intelligence contexts: one, a broader, open-ended and evolving problem domain; and two, interactions with other system resources (databases, quantitative routines). In order to focus on the theoretical issues involved, an abstracted view of a DSS as a formal language is proposed. This highlights the fundamental role of a uniform semantic foundation (ontology) for the various DSS components. Using this perspective, various issues in philosophical semantics are described as they apply to managerial DSS applications.

STRUCTURE OF A KNOWLEDGE-BASED DSS

Sprague (1980) characterizes a DSS as having two basic types of problem oriented resources:

1. databases — which contain *facts* about the environment
2. models — which enable *inferences* to be made.

Practically speaking, the models are almost always quantitative algorithms, typically providing optimization or statistical inferences.

A *knowledge- based* DSS adds an additional component, the so-called 'knowledge-base.' The formalisms employed fall roughly into two general categories: semantic net and predicate logic formalisms. The pros and cons of each are much debated, the general objectives are similar: the *declarative* representation of (mainly) *qualitative* knowledge.

Databases, of course, contain both qualitative and quantitative data. Where as operations research models may provide inferences on the quantitative data, a knowledge base provides structures of inference of a qualitative sort.

The more important aspect is that these are *declarative*, as opposed to *procedural*, structures. That is, the problem-oriented information is represented as independent, axiomatic rules which are searched heuristically. While this is computationally less efficient, it is correspondingly more flexible in that a complex

network of potential inference paths is represented. It is this aspect which warrants the comparison to human knowledge, capable of being applied in various directions and forms, rather than limited to a single deductive path as are normal (procedural) computer programs.

However, declarative representations are computationally practical only for a limited number of primitive qualities.

Quantities, regarded as qualities mapped onto a linear (ordinal, interval, ratio) scale represent large families of qualities. Thus, represented as declarative axioms, arithmetic becomes terribly cumbersome computationally. This is why such declarative languages as PROLOG have so much difficulty incorporating arithmetic operations and, correspondingly, why quantitative inference is nearly always represented procedurally.

The potential of a knowledge base in a DSS is to provide a unifying framework of higher level abstractions of the qualitative facts in databases as well as incorporating the specialized inferences of quantitative models where appropriate. The knowledge base would thus provide an conceptual map of the user's problem domain allowing flexible and adaptive integration of system resources.

On the other hand, while artificial intelligence research is providing the mechanisms for building knowledge bases, the successful application of these tools depends on a formal understanding of managerial problem domains. This is so far lacking. The need is for an applied epistemology of the knowledge typical in business environments.

DSS AS A FORMAL LANGUAGE

The issue here for DSS, as we see it, is to find a representational perspective that somehow avoids computational preoccupations and focuses on the conceptual organization of the DSS in modeling managerial problem domains.

A useful approach is that used in logic for comparing and evaluating logical representations (e.g., van Fraasen 1971). This is to regard each as an instance of a *formal language*, consisting of:

a. *syntax* comprising
   i. a *vocabulary* of elementary symbols
   ii. *formation rules* which define well formed expressions in the language.

b. *transformation rules* — which define truth preserving substitutions between expressions

c. *semantics* indicating what the symbols and expressions of the language denote.

Thus, various logics are compared based on differences in their syntax, inferential power (transformations), and semantics. A similar concept of formal languages is also familiar in theoretical computer science. Turing's concept of abstract automata is as a recognizer of formal languages of varying degrees of syntactic complexity. This view is almost entirely syntactic however. (See, e.g., Hopcroft and Ullman 1974).

When semantics is discussed with respect to computer languages, what is usually intended is *computational semantics*. the machine operations and data structures corresponding to each high level expression. (In human terms, this would be analogous to the neurophysiological representation of our spoken sentences.)

Logicians and linguists are on the other hand concerned with *denotational semantics*, the objects or sets of objects which symbolic expressions signify in the real world. It is this latter concept of semantics which is of concern here.

Earlier we categorized the internal resources of a knowledge-based DSS as:

- databases of quantitative and qualitative facts
- procedural routines for quantitative inference
- declarative structures for qualitative inferencing

In principle, these various components should each contribute to aiding the user's understanding of a certain problem domain. But how do these components interact? A way of examining the problem abstractly is to regard them as various interacting formal languages, or indeed as different aspects of a single formal language.

Clearly the syntactic compatibility of these aspects will be important, though this is mainly an engineering problem. The deeper problems are semantic: how the symbolic expressions of the various DSS components refer to the phenomena in the user problem domain.

MODELS OF FORMAL LANGUAGES

While we normally consider the semantics of a language to be something fixed, it is clear that the association of an arbitrary symbol to the object it signifies is a matter of convention ("a rose by any other name would smell as sweet"). In the perspective of formal languages, this convention is made explicit in the concept of a *model*, which is an assignment of interpretations to the basic symbols of the language. Note that this use of the term 'model' is slightly different than the colloquial usage in the DSS literature. Most of what are there called 'models' would here be called an algorithm that is, they are procedures for performing a sequence of deductions. For instance, a multiple regression routine, in itself, would be an algorithm. However, when an interpretation is given to its terms, e.g., as sales, advertising costs, disposable income, it is then a model in the formal language sense; i.e., it models or is an abstraction from some real world situation.

This usage also differs from that in database management, e.g., the relational or network models. In the formal language sense these would only be models when used to describe some actual organizational environment.

Despite the confusion it may create in terminology, we believe that this formal sense of the term 'model' represents a central issue for DSS research: *that is, to develop a theory which defines families of models (interpretations of formal languages) common to administrative contexts and their variations in specific situations.*

The contributing disciplines of DSS — e.g., database management, operations research, statistics, artificial intelligence, logic, etc. — can be viewed as offering various types of uninterpreted formal languages. These are normally interpreted

in specific, isolated situations, for instance, a database design for a bank, an OR model of traffic flows, a regression forecast of sales in a particular market area. ***Modeling*** (the interpretation of these formal languages) is not itself formalized in these disciplines and remains the art of the technical analyst.

The contention here is that while the phenomena of managerial environments varies widely from one situation to another, there are nonetheless commonalities which can be organized to guide and discipline the modeling process. This organization would no doubt take the form of similarity hierarchies where situations are compared at varying levels of abstraction. Strong evidence for this possibility is the long success of the practice of accounting in providing abstract measures of business activity; e.g., the comparability of financial statements. Accounting however is mainly concerned with measurement, based on monetary valuation, and leaves the underlying phenomena to be informally understood (for instance, few accountants can give a formal definition of an "asset") whereas it is these latter aspects that are the focus here.

ONTOLOGY AND THE IDENTIFICATION OF INDIVIDUALS

Ontology refers to the nature of the primitive entities which the expressions of a (formal) language denote; i.e., what basic conceptual constructs are used to define the sets of the objects which form a model of the language.

The purpose of an ontology is to *clarify*, through reduction of informal description to a smaller set of more sharply defined terms. The inferences made in the language can only be as sound as the underlying ontology. (This is a philosophical version of 'Garbage-In-Garbage-Out).

An ontology can only clarify if the sets it comprises (the denotations of the language) are clearly understood and whose elements are clearly distinguishable by the users of the formal language. Thus the adequacy of an ontology is a matter of *consensus;* but it is a consensus that must be carefully scrutinized, since the value of further definitions and inferences in the language depends on the soundness of this foundation.

Since sets consist of discrete individual elements, the central issue in most ontological debates is the identification of individuals. That is, what are the sorts of things (individuals) which form the sets our concepts refer to? An intuitive test for the consensual recognition of individuals is whether the parties involved agree that two individuals are the ***same***.

Discrete physical objects, for instance, seldom give rise to confusion, and it is noteworthy that most operations research models apply to ontologies of this type; e.g., involving employees, machines, or physical inventories.

Transformations on physical individuals can however give rise to potential confusions (which gives some insights to the difficulties in dynamic modeling). A delightful example (Brachman, personal conversation) is that of a wooden boat and we replace one of its planks with a new one. Is the modified boat the same individual as the original? Most people would agree. Suppose we continued to systematically replace planks in the boat with new planks until all parts of the boat were now replaced. Is this individual the same as the original? Some, though perhaps not all would agree. Now, suppose we collected the planks we removed

and constructed another boat in the design of the original boat. Is it now the same as the original?

Austin (1970) summarized the matter by observing that *similarity* is a property of nature whereas *sameness* is a matter of linguistic usage. The boundaries of individuation, in short, depend on the consensus of the user group or population.

Time spans – e.g., days, weeks, months, years – tend also to be relatively unproblematic in ordinary situations. Few people disagree about the temporal boundaries of 7 December, 1941, for example, despite the minor problems created by different time zones. (Among theoretical physicists, however, the ontology of time is quite different and more open to dispute.)

The ordinary language use of "same" has another, apparently separate sense. "I drive the same car as John," may mean that there is one individual vehicle that we share, or that we drive the same type of car. These are sometimes distinguished as sameness of individuals vs sameness of type. In a logical notation, the latter involves a predicate variable, i.e.,

$$\exists X \text{ drive(me,X) \& drive(john,X)}$$

vs

$$\exists \text{TYPE } \exists X \exists Y \quad \text{drive(me,X) \& drive(john,Y) \& TYPE(X) \& TYPE(Y).}$$

(Here and throughout, constants are lower case, variables upper case.)

As we move out of the domain of discrete physical objects, individuation becomes less clear. For example, in a hospital if a doctor declares that patient Smith has the same disease as patient Jones, it is apparently meant that the two diseases are of the same type, e.g., that the bacteria are of the same species. On the other hand, it may mean that the two diseases are from the same bacterial pool. The difference matters where contagion is of concern. Again it depends on the needs of the user group.

Abstract objects are notoriously difficult to individuate, essentially because there are no lowest level 'atoms' (molecules, cells, etc.) to which one can take recourse. For instance, to say that X independently had the same idea as Y, or that X plagarized or stole Y's idea is extremely difficult to pin down; is this sameness of individuals or sameness of type?

Strawson (1959) asserts that the only reliable basis for individuation is to locate the individual in a spatial temporal framework. In this way, ideas might be identified to the mental activities of a certain person throughout a certain period in time.

These aspects of individuation are of central importance to the development of knowledge-based DSS since, in most cases, these have ambitions to include expertise beyond the ontologically safe domains of discrete physical objects.

ONTOLOGIES INCLUDING NUMBERS

Pure mathematics usually adapts some abstract set of numbers in their ontology, e.g., the integers, real numbers, rational numbers, etc. Applied mathematics, on the other hand, usually includes a broader ontology, namely that the numbers involved are ***measures of some scalable properties***. The type of scale involved,

e.g., ordinal, interval, ratio, determines the algebraic flexibility of the inferencing. Typically left implicit or informally described, are the individual objects to which these measures are applied. As observed earlier, these are typically straightforward from an ontological standpoint, so little confusion arises.

However, when measures are applied to less obvious phenomena, the summary statistics generated from these measures can become quite ambiguous to the people using them. This has become a serious problem in accounting where monetary valuations are applied to a wide range of disparate phenomena (with subsequent allocations, prorations, amortizations, price level adjustments, etc. applied to them) so that the final results are only vaguely meaningful. For example, an occasional student exercise in financial accounting is to revise a company's net income 100% entirely through adjustments conforming to Generally Accepted Accounting Principles. Knowledge-Based DSS's applied to such domains would be prone to similar difficulties. The suggestion is to expand the ontology to explicitly recognize the types of underlying entities being measured.

ONTOLOGY OF DATABASES: DATA VS OBJECT

In the architecture for a knowledge-based DSS presented earlier, current ***facts*** about the environment are recorded in (one or more) databases. Since these provide the basis for higher level inferences, the ontology they assume plays a fundamental role.

Codd's (1970) Relational Data Model ('model' in the database sense) is often regarded as a useful, mathematically abstracted prototype of database systems. The relations involved are tuples of elements drawn from sets of ***data*** items (in relational terminology called domains), such as single characters, character strings, integer numbers, floating point numbers, etc. The operation of these systems depends only on the symbolic shape of these items, not on their significance to the users of the system. This is similar to the use/mention distinction in natural language semantics. E.g., the teacher's question

Can you spell "can"?

first uses the word "can," then mentions it (as was done again in this sentence). Database designs present a syntax of data but no denotational semantics. Hence, databases have no explicit real world ontology. However, they often, implicitly, reflect a certain ontology in the definition of relations. For instance, a database

EMPLOYEE(E-NAME,EMP-ID,AGE,...)
DEPARTMENT(D-NAME,DEPT-ID,LOCATION,...)
WORKS-FOR(EMP-ID,DEPT-ID)

implicitly recognizes employees and departments as individuals, with "WORKS-FOR" as a two place predicate relating them. The existential implication is that for each tuple in the EMPLOYEE relation there is an actual employee in the company, and for each tuple in the DEPARTMENT relation there is a department in the company. Such existential presuppositions of certain database relations are the basis of Chen's Entity-Relationship Model (1976).

GRANULAR AND LIQUID OBJECTS, MASS OBJECTS AND PROBLEMS OF INDIVIDUALIZATION

The world, according to Quine (1960), consists of middle size objects. Problems of individuation arise when we consider granular objects, such as corn, wheat and liquid objects, e.g., water, oil, etc. The problem is the same in both cases: to discretize these objects and assign names to them, it is impractical to go to their lowest level elements (grains or molecules).

While this poses a difficult theoretical problem (logics over continuous domains, paradoxes arising from axioms of choice), in commercial practice, the problem is typically avoided through the simple device of a *container*. That is, these substances are normally conveyed in a (middle sized) container which is easily individuated and named. The contents of the container become properties of (predicates applied to) the container. Emptying one container into another involves changes of properties of the two containers (see temporal aspects, below).

Note that whether something is to be treated as a granular substance or as discretely identifiable objects depends on the interests of the potential users of the language. For instance, rock and gravel companies would no doubt regard stones beneath a certain diameter as granular. A rock collector, on the other hand, would regard them as individually identifiable specimens.

Mass objects are an intermediate class sharing properties of discrete individuals and liquid objects. Examples are planks of lumber, bars of steel, etc. These can be divided into increasingly smaller units of the same substance. These can of course be treated as individual objects. Divisions of the object cause the destruction of the original and the creation of two new individuals. Alternatively, these are often regarded in a way similar to liquid objects, where the container is some specified inventory location, section of a warehouse, etc. In this case, e.g., lumber is treated as so many board feet without regard for how many individual pieces the inventory contains. The choice, again, depends on the intended usage of the formal language.

AN ONTOLOGY INCLUDING TIME

Time, which is so central in commercial environments has, oddly enough, had relatively little development in the concept of formal, especially logical, languages. Principle works on temporal logic are by Prior (1967) and Rescher and Urguhart (1971).

The implicit conception of time in commercial environments seems to be a continuous dimension of time points. This would normally cause the same logical problems as liquid objects except that the reference to time is inevitably with reference to *time spans*, which have a similar ontological status as containers to liquids.

Examples of individual time spans are:

The year: 1984
The month: January, 1981
The day: 7 December, 1941
The minute: 11:59 a.m., 2 July, 1982, Central European Time

An ontology of time might alternatively assume a time line of discrete units of some minimal size. Such is the perspective in digital watches and computer clocks. Time, taken as discrete or continuous, is regarded as linearly ordered. This is the basis for concepts of change and of precedence in changes. By including time in the ontology, the truth of a predicate becomes dependent on time. This amounts to adding a temporal sort to the language and adding a time place to each predicate.

POSSIBLE WORLDS SEMANTICS

No doubt the most seductive yet controversial concept introduced in ontological theories this century is that of a ***possible world***. Intuitively speaking, a possible world is like a formalized gedanken experiment: it is an imaginary locus to which truth values can be attached. The world we know is a priviledged possible world: the actual world.

Debate over possible worlds centers on whether the concept can be consensually understood sufficiently well by the users of a formal language whose semantics depend on it. (In this regard it is like the utile in economics: theoretically very useful but ontologically rather questionable).

A principal motivation for the concept of possible world is to give a denotational semantics to generic concepts. We would like to consider the denotation of a predicate as the set of things of which it is true. However, those things existing in the actual world are typically not enough. This denotation in many cases must be extended to possible worlds as well.

For example, consider the denotation of the concept: person? Is the property of personhood equivalent to elementhood in the set of all people currently alive? or the set of all people who have ever lived? or the set of all people who ever lived or will live? Normally, even this last set is considered incomplete, for it refers only to actually existing persons in the past or future. The essence of the concept person (called its ***intension***) is however the denotation (or ***extension***) of human individuals in all times in all possible worlds.

Further, of perhaps more practical consequence, the concept of possible worlds permits the formal definition of concepts of ***action*** and ***responsibility***. Various conceptions of action are possible, depending on the purpose of the formalization. One, due to von Wright, distinguishes action from a simple change in state in that it is brought about by some (human, organizational) agent. This contains an implicit counter-factual: that if it were not for the agent's intercession, the change would not have taken place. Thus, while a concept of change can be described in terms of transitions in states of the actual world from one time to the next, a concept of action requires the notion of another, possible world to express the state of affairs were it not for the agent's intercession. Thus, by asserting someone responsible for a particular state of affairs, we allude to some alternative state that would exist had that person's influence not been present.

PREDICTIONS, PLANS AND PROMISES

In discourse relating to administration, finance and commerce, it is only statements concerning the past and present that are considered factual. For instance,

that company X sold company Y a piece of equipment Z on date D, is either true or false if D is in the past. However, if D is a date in the future, the statement is not regarded as either true or false, but rather one of conjecture or speculation.

Three principal types of conjectures or attitudes in these contexts are predictions, plans and promises. In their semantic formulation, each of these makes an assertion about some possible world in the future, with the additional claim that the actual world will eventually match this possible world.

A *prediction* is simply a description of such a future possible world with the assertion that the course of events in the actual world will eventually lead to this state.

A *plan* is a prediction augmented with intentions of action. The assertion is that the future possible world described in the plan would not normally come about, except for the intended actions of the planner.

A *promise* is a plan augmented with a *commitment* to another party. Implicit in the notion of commitment is some penalty for not carrying out the plan. This penalty may be a vague moral reproach, some type of legal recourse or perhaps definite consequences such as foreclosure or siezure of assets.

A promise is the act of incurring an *obligation*. Obligation is one of several operators in a so-called *deontic logic* (von Wright 1968). Others are permission and prohibition. Each involves two parties and an action. Symbolically,

obliged(X,Y,A)	=	X is obliged to Y to do A.
permits(X,Y,A)	=	X permits Y to do A.
prohibits(X,Y,A)	=	X prohibits Y to do A.

These are inter-definable: to be permitted to do something is to not be prohibited from doing it and vice versa; to be obliged to do something is to not be permitted not to do it and vice versa.

A *contract* is a relationship of mutual obligation. A *contingent obligation* is one where the obligation depends on the occurrence of some event. A familiar example is insurance.

Interestingly, the deontic relationships of obligation, permission and prohibition are, in commercial contexts, often reified to the status of *objects*. Examples of deontic objects based on obligation, are notes, various types of bonds, and with a real but less definitely described obligation, the various types of preferred and common stock. Insurance policies are examples of contingent obligations. Examples of deontic objects based on permission are licenses, easements, etc. whereas examples of deontic objects based on prohibitions are: copyrights and patents.

A formal device to accomplish this reification to objecthood is the intension operator, $^{\wedge}$, due to Montague (best explained in Dowty (1981)). (This is essentially a lambda abstraction on time/possible world pairs, serving to make intensions extensional. In our case this operator would be applied to deontic expressions.)

These deontic objects constitute *assets*, that is they are *owned*, by one of the parties involved. For instance a bank owns its notes outstanding; investors own their stocks and bonds. Likewise insurance policies, licenses, copyrights and patents are owned. In the case of promissory objects (deontic objects based on obligation), the object represents a *claim on assets* to the other party.

CONCLUDING REMARKS

In the foregoing we have argued that an important theoretical problem for knowledge-based DSS in organizations involves the epistemology of management: identifying the foundational concepts of managerial knowledge. In this paper we sketched an approach using denotational semantics, and suggested several basic types of individuals: physical objects, numbers, time and possible worlds. We stressed that these basic entities are not to be considered as 'essential' in that no other bases are possible. Rather as Goodman (1978) points out in *Ways of Worldmaking*, all such conceptual systems are a matter of consensus and utility to its user population. However, this does not mean that no generally useful conceptual foundations are possible for managerial domains. Indeed, the widely accepted terminology of accounting provides informal evidence that this is possible. For more detailed discussion of these issues, see Lee (1981a).

ACKNOWLEDGMENTS

The author wishes to acknowledge useful interactions with Helder Coelho, Steven Kimbrough, and Amilcar Sarnadas on these topics.

REFERENCES

Austin, J.L. 1970. *Philosophical Papers*. Second Edition. (Section 5. Truth, pp.117-133.) Oxford: Oxford University Press.

Bonczek, R.H., C.W. Holsapple and A.B. Whinston. 1981. *Foundations of Decision Support Systems*. New York: Academic Press.

Chen, P.P. 1976. The Entity-Relationship Model — Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(March):9-36.

Codd, E.F. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(June):377-387.

Dowty, D.R., R.E. Wall and S. Peters. 1981. *Introduction to Montague Semantics*. Boston: D. Reidel Publishing Company.

Goodman, N. 1978. *Ways of Worldmaking*. Indianapolis, IN: Hackett Publishing Company.

Ijiri, Y. 1967. *The Foundation of Accounting Measurement*. Englewood Cliffs, NJ: Prentice-Hall. Reprint ed., Houston: Accounting Classics Series, Scholars Book Co., 1978.

Hopcroft, J.E. and J.D. Ullman. 1969. *Formal Languages and their Relation to Automata*. Reading, Mass: Addison Wesley.

Lee, R.M. 1981a. CANDID Description of Commercial and Financial Concepts: a Formal Semantics Approach to Knowledge Representation. WP-81-162. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Lee, R.M. 1981b. Relational Databases, Logical Databases and the Entity-Relationships Approach. WP-81-164. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Prior, A. 1967. *Past Present and Future*. Oxford: Oxford Press.

Quine, W.V.O. 1960. *Word and Object*. Chambridge, MA: MIT Press.

Rescher, N. and A. Urguhart. 1971. *Temporal Logic*. Vienna: Springer-Verlag.

Sprague, R.Jr. and G. Fick. 1980. *Decision Support Systems. Issues and Challenges*. Oxford: Pergamon Press.

Strawson, P.F. 1959. *Individuals*. Methuen & Co. Ltd. Reprint, Anchor Books, Doubleday & Company, Inc., New York in 1963.

van Fraasen, B.C. 1971. *Formal Semantics and Logic*. New York: The Macmillan Company.

von Wright, G.H. 1968. An Essay in Deontic Logic and the General Theory of Action. *Acta Philosophica Fennica* Fasc. XVIII. Helsinki. pp. 293-301.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

PROLOG: A PROGRAMMING TOOL FOR LOGICAL DOMAIN MODELING

by

Helder Coelho

Centro de Informática
Laboratório Nacional de Engenharia Civil (LNEC)
101, Av. do Brasil, 1799 Lisboa CODEX, Portugal

Recent research in decision support systems shows a growing interest in knowledge representation and knowledge-based systems. Yet, implementations of DSS continue to be dominated by quantitative, algorithmic perspectives of such languages as APL, BASIC, FORTRAN, etc. This paper presents an alternative approach. It uses the language Prolog.

INTRODUCTION

There is a growing interest in Decision Support System (DSS) research to incorporate techniques and methods from Artificial Intelligence (AI), especially in the area of so-called "knowledge-based" or "expert" systems.

An AI expert system may be distinguished from a DSS in that the former seeks to replace some specialized function of a human expert, whereas a DSS seeks to assist that person. A DSS therefore addresses problem domains that are less well structured, rather than the more formalized problem areas typically addressed by AI expert systems, eg. medicine, engineering.

Much of the research debate has focused on the relative advantages of various knowledge representation schemas, eg. semantic nets, frames, predicate calculus, for describing the problem domains of a DSS.

For implementating DSS, programming languages such as APL, BASIC or FORTRAN are frequently used, because they are well suited for implementing quantitative algorithms. However, they are badly suited for describing the sorts of qualitative knowledge and heuristic control structures presupposed by the knowledge representation research.

The purpose of this paper is to propose a candidate tool for bridging the gap between DSS-theory and practice: the language Prolog and the technique of "logic programming" as a means of implementing knowledge-based DSS.

PROBLEM DOMAIN REPRESENTATIONS

DSS problem domains are, in many cases, developed directly in an implementation oriented language. The following difficulties may then arise:

1) the description of environmental characteristics with programs and data files tends to be ad hoc;

2) the degrees of freedom for problem domain representation are limited by computational considerations, and

3) the accessibility of information on the logical model of the problem domain is difficult.

The value of an abstract representation using logical formalisms is to separate and clarify the modeling aspects from computation oriented considerations.

WHAT IS LOGIC PROGRAMMING

Logic programming is the use of the clausal form of logic as a practical computer programming language. It is based upon the following thesis: predicate logic is a useful language for representing knowledge. It is useful for stating problems and it is useful for representing the pragmatic information necessary for effective problem solving [12].

ARGUMENTS FOR USING PROLOG

Prolog is a very high level and general-purpose programming language based upon the procedural interpretation of definite clauses (the so-called Horn clauses [11] of predicate logic).

It can be viewed as an extension of pure Lisp, or as an extension of a relative database query language. It was first conceived around 1972 by Colmerauer [6]. There are now many other implementations of the language, interpreters and compilers, for a large range of computers, including mini and microcomputers [3,5]. Since 1972 it has been used, mainly in Europe, for a wide variety of applications, including natural language processing, algebraic symbol manipulation, compiler writing, architecture design, and expert systems [4].

Prolog is not the ultimate logic programming language but a first step towards it. In fact, current investigation carried out in Europe, Japan and USA put forward other more powerful languages à la Prolog (eg. the hungarian T-Prolog, as very high level discrete simulation system, competes against SIMULA).

PROLOG COMPARED WITH OTHER LANGUAGES

Prolog is different from most programming languages, in that it does not presuppose a von Neumann architecture and does not have assignmemt as the basic underlying operation.

For applications requiring an easy-to-use and transparent language for symbol processing, Prolog offers significant advantages over Lisp [18]. Lisp has not an easy reading syntax and variable binding mechanism. The major barrier to its readability is the size and degree of nesting of typical function definitions. Prolog allows a program to be built into small modules, each having a natural reading. In addition it gives the programmer generalised record structures with an elegant mechanism for manipulating them. Experiments carried out by [18] showed that Prolog was more efficient than Lisp. Also, for applications requiring interactive programming, such as those in CAAD, Prolog offers real advantages over FORTRAN [14]. For query processing applications it was shown that a Prolog database system compares well to System R and Ingres [19].

PROLOG PROGRAMMING STYLE

Prolog is a step towards a more declarative style of programming and can be viewed as a descriptive language as well as a prescriptive language [8,9]. The Prolog approach is rather to describe known facts and relationships about a problem than to provide the sequence of steps taken by a computer to solve the problem. When a computer is programmed in Prolog the actual way the computer carries out the computation is specified partly by the logical declarative semantics of Prolog, partly

by what new facts Prolog can "infer" from the given ones, and only partly by explicit control information supplied by the control programmer.

A Prolog program consists of a set of statements which can be read declaratively as well as procedurally [1,15,16,17].

Example 1:

```
work_with_at (X,Y,Z):-      colleague (X,Y), employee (X,Z),
                            (employee (Y,Z); friends (X,Y))
(declarative reading (Prolog semantics):

           (For any X, Y and Z,)

              X works with Y at Z if
              X is colleague of Y and
              X is employee at Z and
              Y is employee at Z or
              X is friend of Y).

(Procedural reading (Prolog pragmatics):

           To find whether X works with Y at Z
              find whether X is a colleague of Y and
              .
              .
              .
```

A Prolog programmer may declare that certain predicate and function name will be written as operators with a particular precedence.

Example 2:

```
:-op(700,xfy,'likes').
João likes Maria.
Helder likes Maria.
Maria likes.
```

Therefore, a possible query would be written as:

```
              ? - X likes Y, Y likes X.
```

with the reading: "Who does like some one and is corresponded?".

When a Prolog program is to be executed in order to attain some goal or to solve some problem. the objective of the machinery behind the language is to verify if the goal statement is true or false in all possible interpretations, through a proof of its validity or inconsistency. The Prolog proof procedure is composed of an inference system and a search strategy. The inference system specifies (what to do) by means of axioms and rules of inference: the search space of all admissible derivations. The search strategy determines (how it is done) the sequence in which derivations in the search space are generated in the search for a refutation. The proof can also be interpreted as a computation.

FEATURES USEFUL FOR DSS

Prolog is a tool for modeling decision making behaviour and data management facilities. It supports the development of evolving and adapting systems.

- Generally, a DSS may be able to contain a large body of knowledge which is subject to frequent change [10]. The incremental nature of constructing a Prolog program allows its human developer to determine the immediate effect of adding or deleting a new clause. Also, databases may be increased on decreased during an interaction with users.

- One frequent application of addition of clauses is the lemma generation. It occurs, for example, when it is advisable to record sucessful evaluation of queries: the new assertion added corresponds to the derived substitution instance [2].

- A DSS needs a explanation module [10]. It provides the following facilities:

    a) displaying on demand the rule currently beins invoked,

    b) the association of specific rules with specific events and explanation of why and how each occurred, and

    c) searching the knowledge base for a specific type of rule in answer to queries from the user.

In Prolog it is easy to implement such a capability because the use of a general pattern directed invocation of procedures improves parameter passing and there is a Prolog built-in predicate called "ancestor".

- Generation and manipulation of sets of answers to a query are handled by an in-built Prolog predicate, which may be used to define other aggregation relations [13].

- Set construction can be formalized either in first-order set theory or in an amalgamation of object language and metalanguage [13], similar to the language FOL. Amalgamating the two levels of language is very advisable in the case of databases. For example, it is necessary both to describe and query databases in the object language, and to construct and manipulate databases in the metalanguage.

- Interactions between a DSS and its user are implemented through a grammar of dialogues [3], written with Prolog grammar rules [7]. A grammar cascaded with a natural language understanding program defines the front-end of a DSS. The control structure can also be implemented in Prolog by writing an interpreter as a set of meta rules. These rules allow the appropriate choice of search strategies, separating the logic from control.

CONCLUSION

Prolog is a simple, clear and economical (in terms of source code size and programmer's productivity) programming language, based on predicate logic. Its main features are:

1) A declarative semantics inherited from logic in addition to the usual procedural semantics.

2) Program and data are expressed in the same way (clauses).

3) All data objects have a machine-independent, readable representation.

4) Procedures are multi-purpose. They may be called with many different input-output paterns. They may generate, through backtracking, a sequence of alternative results.

5) Structured data is treated in an easy way: pattern matching, instead of selector and constructor functions, no declaration of record types and no type restrictions.

6) Procedures can return partially defined results (ie. containing variables) which are completed by other procedures -- the "logical variable".

7) Unification generalize the pattern matching: unification = pattern matching + logical variable.

8) The effect of executing a Prolog program is completly defined.

The principal applications to date, after 12 years of existence, cover problem domains such as natural language understanding, algebraic symbol manipulation, pharmaceutical design aids, modelling of machine parts, architectural design aids, specifications support, medical decision making, and knowledge engineering (a more comprehensive list is provided in appendix 1). The door is now opened do DSS applications.

ACKNOWLEDGEMENT

The author wish to express his appreciation to Ronald Lee for fruitful discussions and suggestions that helped to refine an early version. My recognition goes also to Henk Sol for the criticism he made regarding the final version, more oriented to the concerns and questions of the DSS community.

REFERENCES

[1] Byrd, L. et al.
A guide to version 3 of DEC-10 Prolog
LNEC, 1978

[2] Clark, K.; McCabe, F.
Prolog: a language for implementing expert systems
Imperial College, 1980

[3] Coelho, H. et al.
How to solve it with Prolog
LNEC, 3nd. edition, 1982

[4] Coelho, H.; Cotta, J. C.
Prolog -- a tool for logic programming and problem solving
DECUS Symposium, 1981

[5] Coelho, H.
Logic programming bibliography
LNEC, 1982

[6] Colmerauer, A. et al.
Un systéme de communication homme-machine en Français
Univ. d'Aix-Marseille, 1973

[7] Colmerauer, A.
Les grammaires de metamorphose
Univ. d'Aix-Marseille, 1975

[8] Emden, M. van.
The semantics of predicate logic as a programming language
Univ. of Edinburgh, 1974

[9] Emden, M. van.
Programming with resolution logic
Univ. of Waterloo, 1975

[10] Hammond, P.
Logic Programming for expert systems
Technical Report DOC 82/4
Imperial College, 1980

[11] Horn, A.
On sentences which are true of direct unions of algebras
J. Symbolic Logic, 16, 14-21, 1951

[12] Kowalski, R.
Logic for problem solving
North-Holland, 1979

[13] Kowalski, R.
Logic as a database language
Imperial College, 1981

[14] Swinson, P. S. G.
Logic programming: a computing tool for the architect of the future
Proc. of the Logic Programming Workshop, 1981

[15] Warren, D. et al.
Implementing Prolog - compiling predicate logic programs
Univ. of Edinburgh, 1977

[16] Warren, D. H. D.
Logic programming and compiler writing
Univ. of Edinburgh, 1977

[17] Warren, D. H. D.
Prolog on the DEC-system-10
Univ. of Edinburgh, 1979

[18] Warren, D. H. D. et al.
Prolog - the language and its implementation compared with LISP
LNEC, 1977 and SIGART/SIGPLAN Symposium, 1977

[19] Warren, D. H. D.
Efficient processing of interactive relational database queries expressed in logic
Univ. of Edinburgh, 1981

APPENDIX 1

SELECTION OF PROLOG APPLICATIONS

PROBLEM DOMAIN	AUTHORS	YEAR
I) PURE RESEARCH		
Plane geometry	Welham	1976
	Coelho & Pereira	1976
Mechanics	Bundy et al	1979
Symbolic calculus	Kanoui	1973
	Bergman & Kanoui	1975
	Kanoui	1975
Natural language understanding	Colmerauer	1971
	Pasero	1972
	Colmerauer & Kanoui	1973
	Roussel & Pasero	1973
	Colmerauer	1974
	Colmerauer	1975
	Guizol	1975
	Pasero	1976
	Dahl	1977
	Pique	1978
	Mellish	1978
	Pereira & Warren	1979
	Milne	1979
	Coelho	1979
	Mc Cord	1980
	Mc Cord	1981
	Mellish	1981
	Pereira & Warren	1981
	Pique & Sabatier	1982
	Colmerauer et al.	1982
Speech understanding	Batani & Meloni	1975
Learning	Brazdil	1978
Knowledge engineering: Chess	Emden	1980
	Bratko	1982
Robotics	Warren	1974
	Giannesini	1978
Database management	Pereira & Martins	1976
	Coelho	1976

II) PRACTICAL PROGRAMS

Compiler writing	Colmerauer	1975
	Warren	1977
Interpreter writing	Pereira & Porto	1979
	Byrd	1979
	Pereira	1982
Distributed logic interpreter	Monteiro	1982
Computer utilities	Battani & Meloni	1975
Travel agent problem	Mellish	1976
	Silva & Cotta	1978
Computer catalogue	Dahl	1976
Plotter programs generation	Darvas	1976
Pollution control	Darvas	1976
Pesticide information	Darvas	1976
Statistics	Darvas	1976
Flat design	Markusz	1977
Building design	Markusz	1980
		1981
Architecture	Rodriguez	1978
	Swinson	1980
		1981
Civil engineering legislation	Cotta & Silva	1978
Drug design aids	Darvas	1978
Drug interaction prediction	Darvas & Futo' & Szeredi	1978
Carcinogenic activity	Darvas	1978
GT-42 Picture book	Santos	1979
Distribution of portuguese families through a scale of income	Pereira	1979
Pre-Registration appointments	Townshend	1979
Library manager	Coelho	1979
Intelligent analyst	Dwiggins	1979
List of equipment	Simões	1980
Program writing	Gaspar	1980

Algebra of relational composition	Brainbridge & Skuce	1980
Fault finder system	Hammond	1980
Gasheater faults		
Car engine faults		
Modeling machine parts	Molnar & Markus	1981
Geographical information system	Warren & Pereira	1981
Electronic CAD	Pasztome-Varga	1981
Production control system	Markus, Molnar & Szelke	1981
Fixture design	Farkas et al.	1982
Specification support system	Farkas et al.	1982
Environmental resources evaluation	Pereira et al.	1982
Migration decision-making	Roach	1982
Database system	Pique & Sabatier	1982
French public administration	Giraud et al.	1982
Translator Edinburgh Prolog form into Micro-Prolog form	Townsend	1982

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

COMPUTER SUPPORT FOR CREATIVE DECISION - MAKING: RIGHT - BRAINED DSS

Lawrence F. Young

Department of Management and
Organizational Sciences
College of Business and Administration
Drexel University
Philadelphia, PA 19104, U.S.A.

Interactive computer-based Decision Support Systems are supposed to enable managers to more closely follow their own behavioral process and apply judgement and creativity in decision-making. However, an examination of actual applications of DSS indicates that they are largely mathematical-model based, and quantitatively analytical in their approach. This kind of process had been called "left-brained" because it is associated with what are thought to be functions of the left hemisphere of the human brain. More qualitative approaches toward creativity in dealing with "open" decision problems have been applied in some human group processes such as brainstorming and synectics. These "Right-brained" approaches have been examined in order to formulate a starting point for functionally specifying a set of computer modules for a new kind of DSS. This new type of system, called "Right-brained DSS", is decribed in this work. A total DSS, utilizing an interface between Right and left-brained approaches, is also described. Selected modules are currently being experimentally developed by the author.

DECISION SUPPORT SYSTEMS: LEFT VS RIGHT BRAINED

Decision Support Systems (DSS) have been described as computer-based aids for management decision-makers dealing with semi-structured problems [Keen and Scott Morton, 1978]. DSS are differentiated from other MIS components in that they seek to establish a symbiosis of human mind and computer by allowing for a high degree of human-computer interaction and by enabling the manager-user to maintain direct control over the computer's tasks and their outcomes.

The first stage of support is to assist the manager in problem exploration and definition. The second stage aids in formulating alternative solutions, and the third and final stage in selecting a strategy or plan. These stages correspond to Simon's [1960] description of the three major stages of the decision process as Intelligence, Design, and Choice.

Simon recognized that many aspects of decision-making, and many decision problems in their entirety, were not "programmable." These unstructured or "open" problem tasks were characterized as requiring human judgement and creativity. Keen and Scott Morton [1978] point out that Simon had expected an increasing number of management decisions previously found to be unprogrammable would rapidly become structured and programmed for computer solution, but that this has not happened. The computer, they claim, "has had minimal impact on tasks involving judgement, ambiguity, creativity, and volatility of environment." The new, DSS approach needed to increase the computer's impact on such tasks has been evolved by a

number of practitioners [Little, 1970].

The DSS approach is supposed to succeed where previous efforts failed because it does not require either the complete automation of decision-making or the computerization of an isolated task unconnected with the human processing that must follow it. It instead breaks the decision process into a menu of selectable modules, each of which is understood by the user, adjusted and controlled by the user, and inter-woven into the decision-maker's own step by step human processing sequence.

The broad charter of DSS is to support the more creative and intuitive aspects of decision-making as well as related structured, analytical tasks. While we do not yet have a precise definition of "structured" vs "unstructured," it has been postulated that structured analytical tasks requiring computation are associated with functions performed by the left side of the human brain, while those tasks of a more creative, unstructured and qualitative nature, involving preceiving patterns, have been attributed to the right side of the brain [Restak, 1979]. Recently, some doubt has arisen as to whether or not the brain is actually differentiated along these lines, but for our purposes it is useful to maintain the dichotomy as a convenience to categorize varieties of DSS functions as Left-brained vs Right-brained. Attributes of each of these types of decision support are summarized in Exhibit I.

Considering these attributes, the DSS applications in use seem to be mostly left-brained. All six cases described by Keen and Scott Morton [1978] and summarized in Exhibit II are essentially left-brained in both their analysis techniques and in the nature of the problems themselves. Other decision problems, such as formulating general policy, determining methods and processes to influence individual and group behavior, conceptualizing alternative new products, etc., are essentially qualitative in nature. These types of problems often have a much greater impact on organizations and societies than those that more naturally lend themselves to quantitative analysis.

Some efforts to structure these qualitative problems for computerized analysis have been made. An example is the Yale University simulation called POLITICS, which attempts to model national reactions to international events [Restak, 1979]. But efforts such as these are experimental and can be likened to earlier management science efforts to model and computerized an entire solution process. These efforts have not and are unlikely to produce automated replacements for human judgement.

The DSS approach, however, does not require replacement of human judgement and therefore its application need not await break-throughs in totally modelling all of the complexities in either the external phenomena or the internal human thought processes involved.

But new approaches are needed to enhance the right-brained capabilities of DSS before we can extend their scope of application to open, qualitative problems. We propose that these new approaches can be based on non-computer protocols that have existed for some time, and have been demonstrated to be useful. They are methods to aid creative thinking and problem-solving by redefinition, restructuring, and group dynamics such as the brain-storming and synectics procedures. A succinct presentation of the full range of such methods has been provided by Rickards [1974] and is summarized in Exhibit III.

Based on a selection of the non-computerized techniques listed by Rickards and described by Nystrom [1979], Osborne [1957], Prince [1970], Allen [1952], De Bono [1970], and Barker [1968], in the following section of this paper we will describe a set of computerized modules that would comprise a generalized right-brained DSS.

EXHIBIT I LEFT VS RIGHT - BRAINED DSS FUNCTIONAL CHARACTERISTICS

Level of DSS Function*	"Left-Brained"	"Right-Brained"
Level 1		
Information Retrieval	Deals mainly with numeric data bases	Deals with alphabetic (words, phrases, statements) data bases
Level 2		
Filtering and Pattern Recognition	Numerical summarizing, graphing, statistical data reduction, analysis of variance and covariance, time series analysis	Qualitative similarity analysis, taxonomy, formulation of non-numeric concepts and relationships, content analysis
Level 3		
Extrapolation, Inference and Logical Comparision	Simple numerical computation, numerical comparison and ranking	Combinatorial generation, restructuring, and ordering of qualitative elements
Level 4		
Modeling	Heuristics, optimization, and simulation dealing with quantitive outcomes	Aiding scenario building. simulation, and evaluation of qualitative outcomes
Model Features:		
a) Nature of objectives	pre-defined variables, numerically measurable	often not known at outset
b) Nature of constraints	Relatively "closed", allows for variable numerical parameters of pre-defined dimensions	Qualitatively described, relatively open, categorical parameters of dynamically changeable dimensions

* Levels of DSS support are based on those given by Keen and Scott Morton (1978), p. 97.

EXHIBIT II — CATEGORIZATION OF CASES

DSS Case	Main Characteristics: Left-Brained	Right-Brained
1. "Portfolio Management System" (PMS)	Large, mainly numerical data base treated with graphical analysis and a variety of simple computational modes.	very limited qualitative info retrieval (generally item identifying info)
2. "Projector"	Corporate financial planning system utilizes analytical tools such as regression, exponential smoothing, evaluation of variety of quantitative indices.	"
3. "Capacity Information System" (CIS)	Interactive graphics used to assess impact of production plan changes. Main impact is help identify production bottlenecks. Forecasts, analyzes plans, does a capacity analysis using LP.	"
4. "Brandaid"	User calibrates numerical marketing response curves and simulates numerical aggregate outcomes.	"
5. "Geodata Analysis and Display System" (GADS)	Inter-active graphics that draws maps (allocates space) by coordinate analysis of quantitative criteria selected by users.	"
6. Generalized Management Information System (GMIS)	Allows user to combine otherwise stand-alone analytical computer systems. Thereby enables user to construct an ad hoc (non-recurring) analytical decision support system.	"

*Cases were described by Keen & Scott Morton [1978] (Ch. 5 pp 99-166) but characterized here by the author.

EXHIBIT III NON-COMPUTER TECHNIQUES AND SUB-ROUTINES FOR CREATIVE ANALYSIS *

Techniques	Sub-Routines	
1. Restructuring	a. morphological analysis	
	b. relevance systems	
	c. attribute listing	
	d. research planning diagrams	
2. Decision Aids	a. weighting procedures	
	b. checklists	
3. Redefinition	a. goal orientation	
	b. successive abstractions	
	c. analogy procedures	
	d. wishful thinking	
	e. nonlogical stimuli	
	f. boundary examinations	
	g. reversals	
4. Brainstorming	a. Osborne's methods	
	b. trigger sessions	
	c. recorded round robin	
	d. wildest idea	
	e. reverse brainstorming	
	f. individual brainstorming	
5. Synectics	a. active listening (constructive group behavior)	d. changed meeting roles and analogy
	b. goal orientation	e. excursion through speculation and analogy
	c. itemization	f. individual synectics

* Taken from Summary by T. Rickards [1974].

FUNCTIONAL SPECIFICATIONS FOR A GENERALIZED RIGHT-BRAINED DSS

General characteristics of a right-brained DSS at each level of support were summarized in Exhibit I. A more specific list of computerized functional modules intended to serve each of the support levels is presented in Exhibit IV.

These modules are tentatively offered as prototypes which can be operationalized in a variety of ways. A description of the functions to be performed by each module is presented in the following discussion. The basis and relationship of these functions to the non-computer techniques presented in Exhibit III is also given where appropriate.

While it would clearly be a large task to develop even a single version of all of these prototype modules, we can work toward this goal in an evolutionary manner by selecting particular modules, creating at least scaled-down versions of them, and maintaining the capability of ultimately linking them into a single system.

Several of the modules described require large, structured data bases. Unit storage costs continue to decrease and access and retrieval speeds continue to increase, thereby making the use of very large data bases more feasible. The problem of categorizing, encoding and capturing all the material needed, however, remains a formidable undertaking. On the other hand, much of the revelant material would be generally applicable to a wide population of users and therefore qualitative data bases could be centrally or cooperatively constructed and maintained. Such data bases have not generally been constructed, we believe, because the software to process them has neither been defined nor constructed. Most new data bases are quantitative rather than qualitative because we know how to do arithmetic and perform basic methods of quantitative analysis. If we can demonstrate that we have useful, generalizable methods of qualitative analysis, the building of the required data bases will follow.

LEVEL 1 MODULES

The "FETCH" family of modules goes beyond mere retrieval by aiding the user's search in a semi-structured manner, and by retrieving more than the traditional forms of published reference material. In different situations, a user may want to recapture his/her own past case experiences as well as those of others, and the system should accomodate the storage and retrieval of these. Beyond retrieving references and experiences, the user will also want to retrieve hypothesized causal linkages, theories and methods derived from these, which we have called "knowledge". Beyond knowledge, the general policies or prescriptions for individual and collective behavior based on knowledge should be accessible. They are referred to here as "wisdom". Each of these modules are described below.

1.1 FETCH REFERENCES

This module retrieves and displays references based on the key words or phrases entered by the user, but does so in a structured, categorical, hierarchical sampling manner, rather than as an exhaustive, uncategorized listing. By means of an interactive user-system dialogue, the system can continue retrieval within a given category or branch to another. The search and dialogue operates in a hierarchical pattern, moving from more general to more specific categories. In this manner the user is spared the burden of being too specific prematurely in the search process and the system and the user are saved time by avoiding an oversampling of less relevant material.

EXHIBIT IV PROPOSED RIGHT-BRAINED DSS MODULES

Level of DSS Function	Right-Brained Modules
1. Information Retrieval	1.1 FETCH REFERENCES
	1.2 FETCH EXPERIENCES
	1.3 FETCH KNOWLEDGE
	1.4 FETCH WISDOM
2. Filtering and Pattern Recognition	2.1 DIMENSIONALIZE
	2.2 CLASSIFY
	2.3 ANALOGIZE
3. Extrapolation, Inference and Logical Comparison	3.1 COMBINE
	3.2 COMPARE
	3.3 GENERALIZE
4. Modeling	4.1 BUILD SCENARIO
	4.2 BUILD POLICY

This interactive DSS approach, although not identified as such, has been the experimental thrust and expectation for advances in library-type information retrieval for some time. The SMART system [Salton, 1971] is one such example of experimental work started in the 1960's. Trends noted at a 1973 ASIS conference [Stevens, 1974] included: on-line interaction and dialog, on-line search aids, machine aided and automatic categorization, and computer-assisted instruction for indexers and searchers.

Categorization, coding and indexing problems notwithstanding, a right-brained DSS type of FETCH REFERENCE module should be created and used at least within a limited area of user interest both for its immediate benefits, as well as for its value as a prototype for developing similar or wider scope modules.

1.2 FETCH EXPERIENCES

This module retrieves profiles of historical cases which resemble a situation of current interest. Cases are screened and selected for display according to both user-specified categorical designations and according to relative similarity measures. Categorical selection criteria are used as absolute or hierarchical "go - no go" filters. Similarity measures are computed according to a variety of subjectively controlled "distance" metrics applied to any mix of qualitative dichotomous attributes or quantitative scaled dimensions. The relative importance of each dimension in assessing similarity is under user control through the assignment of numerical weights.

Case profiles are created, stored, and displayed in a standard format which identifies and distinguishes between the following descriptive elements:

a. active human elements involved
   1. individuals
   2. groups

b. limited resource elements available and used
   1. financial
   2. material
   3. informational

c. functional elements
   1. methods, techniques used
   2. processes followed

d. timing elements
   1. event regularities, frequency
   2. event sequence, priority, dependency
   3. dated events

e. objectives, motives
   1. multiple objectives
   2. conflicting objectives
   3. explicit and implicit motives (informal objectives)

f. constraints
   1. policy constraints
   2. capability constraints
   3. violations of constraints

g. outcomes
   1. financial
   2. material
   3. human
   4. organizational
   5. environmental

h. structural, background
   1. organizational attributes (size, industry, structure, etc.)
   2. organizational history, culture

In the process of examining and weighing the relevant dimensions of similarity between a current problem situation and historical cases, the user is led through consideration of each of the factors enumerated above. In specifying which of these are relevant and to what degree, the user is performing a partial morphological analysis (1a. in Exhibit III), attribute listing (1e), and using the goal orientation method (3a, 5b). The process of specifying relevant experience is thereby itself of value to the decision-maker as well as what may be gained from reviewing the occurrences and outcomes of those similar cases which are displayed.

1.3 FETCH KNOWLEDGE

This module retrieves portions of a highly summarized, organized body of theory and methods in a structured form according to the initial request and subsequent responses of the user. Material would be organized in a manner similar to methods used in programmed instruction [Fry, 1963] and could be created through computerized procedures similar to the automatic course generators used in

computer aided instruction (CAI) [Osin, 1974]. The major difference between the approach used here and that of most CAI Systems is that the DSS principle of user control would be paramount here.

The user needs a rapid briefing by an expert, such as is given in the initial stage of a synectics group meeting (5. in Exhibit III) prior to allowing the group to structure and define the problem and to generate ideas. The computer is used here as a controlled, automated expert, allowing the user to select both the subject matter and level of detail required for the briefing.

The ad hoc nature of the user's need for rapid knowledge makes it necessary for the material to be structured in a manner that makes an efficient search and selection possible. This is an essentially different design requirement from the more general educational aims of most CAI. The hierarchical organization required for this purpose may be a simpler design task than the need to predetermine sequencing and branching for a CAI system. In FETCH KNOWLEDGE it is the user who dynamically controls sequence and any branching, not the system designer.

1.4 FETCH WISDOM

This module retrieves general policies and principles that guide behavior in a situation. Where displays of "experience" show what happened and "knowledge" speaks of how things work, the "wisdom" data base contains prescriptions for how one ought to act based on other observers' and experts' distillation of both experience and knowledge. For example, an entry in a wisdom data base might state: "When a consumer product becomes a commodity in a shrinking market, the appropriate strategy is to milk remaining profits through pricing strategy but not to invest in promotion at the expense of growth products; source: Young, 1982." Another entry on the same subject might contain an alternative, possibly opposing or possibly supplementary normative statement. The categorization and selection of wisdom should operate in a similar manner to that of experience and knowledge. The user selects the subject area and controls a hierarchical selection of material, moving from the general to the more specific. Conflicting wisdom is presented where it exists and general underlying foundations and assumptions accompany each set of prescribed wisdom. The use of this routine would not be aimed at finding specific strategies so much as determining broad parameters that are accepted by the user as guidelines within which one can refine a specific strategy. This module can thus be seen as an extension of the expert briefing provided by FETCH KNOWLEDGE and can be used in the preliminary stages of problem definition. Alternatively, it may be better used in the later stage of evaluating a proposed approach or policy for its compatibility with general wisdom. This delayed evaluation of wisdom may be less likely to limit creative approaches.

A body of modifiable user wisdom could also be captured and retrieved by the system as the user may desire. By this device, a user can track and develop a personal set of policies and guidelines for private use. The system thus supplements and supports the users own internal memory, feedback and learning mechanisms.

LEVEL 2 MODULES

Given the qualitative background data with which to work by some of the Level 1 FETCH routines, the user may then begin to break the problem into its parts and to perceive general patterns. Level 2 modules aid in this process by supporting several of the restructuring (1. in Exhibit III) and redefinition (3. in Exhibit III) techniques.

In some cases the user may cycle back and forth between Level 1 and Level 2 modules. For example, the user may start working in a general area without a clear idea of the dimensions of the problem. First some references might be examined using 1.1 and some knowledge might be sought using 1.3 Then the user may apply 2.1 in order to help define the dimensions of the problem and 2.2 in order to categorize the problem in terms of its dimensions. The user may then return to Level 1 in order to examine similar cases with 1.2. In some instances, the user may want to examine cases that are not similar in a direct manner, but bear certain metaphorical similarities that may help in redefining the problem along entirely new lines. 2.3 aids the user in this respect.

The three Level 2 modules are described below.

2.1 DIMENSIONALIZE

This module aids the user in structuring the problem by defining its relevant dimensions, as in the techniques of attribute listing (1c in Exhibit III) and morphological analysis (1 a in Exhibit III).

For example, if the objective is to select a design for a system to transport people, the following dimensions of a system alternative must be considered:

Dimension 1: a power source
Dimension 2: a "people container"
Dimension 3: a control sub-system (velocity, start-stop, steering etc.)
Dimension 4: a pathway or routing sub-system

The alternatives within each of these dimensions must eventually be identified and combined (see 3.1 COMBINE) in order to completely specify a system design.

The module aids the user in two alternative ways, according to the user's choice:

a. Responsive Mode

   By prompting the user to specify dimensions and acting as a secretary in recording them, displaying them, and modifying them, as the user directs.

b. Initiating Mode

   By suggesting potential dimensions for the user's consideration either by selecting them from a standardized set of dimensions or by retrieving the dimensions applicable to a similar case the user has examined by means of 1.2.

If the user begins with the Initiating Mode, he/she would often want to take the computer's offerings as a starting point and then refine his or her own dimension definitions by switching over to the Responsive Mode.

In support of the Initiating Mode, a data base of standard dimensions could be constructed containing alternative definitional structures such as the "Synopsis of Categories" developed by Roget [1977 edition], or much more limited dimensional definitions such as Nadler's [1970] dimensions of a system.

2.2 CLASSIFY

This module aids the user in formulating taxonomies of objects or concepts based on dimensions previously defined. A similarity metric would be used in order to classify items according to their attributes.

For example, suppose the following three alternatives have been defined:

	Power	Container	Control	Routing
Alt. 1:	Electric	50-person benches on wheeled cars	chauffeur driven	road network
Alt. 2:	Electric	Single person lounge chairs on open conveyor	user initiated	fixed routes feeders
Alt. 3:	Nuclear	Single person lounge chairs on open conveyor	user initiated on-off	fixed routes, local on-off feeders

"Distance" between alternatives could be subjectively assessed as a measure of similarity on a scale such as 0 to 10, where the lower the number, the "closer" or more similar the pair. The three alternatives given above might be assessed as follows:

Alts.	Distance
1 vs. 2	8 (very dissimilar)
1 vs. 3	10 (extremely dissimilar)
2 vs. 3	3 (similar)

This assessment could classify alternatives 2 and 3 in the same category and 1 as being in another category.

The classification or clustering here would differ from many existing computer clustering routines in that it deals mainly with qualitative attributes rather than numerically measured variables and that the user's role of interactive control of the clustering results would be paramount, as in other DSS.

Where other clustering methods are validated by some type of objectively measured cluster variance minimization, the "variance" governing a DSS classification should be subjectively appealing above any other criterion. The CLASSIFY module could be used in a general form for objects or concepts or, in a more specialized form to group similar historical "cases" needed as part of the FETCH EXPERIENCES module.

2.3 ANALOGIZE

This module supports the formation of metaphors and analogies, a process commonly recognized as key to creativity [Yukawa, 1973; Prince, 1970]. It is a basic ingredient in both group creativity processes (4 and 5 in Exhibit III) and in individual techniques (3c in Exhibit III).

For example, assume we are trying to find alternative new uses for powdered milk, thinking along the lines of making a cheap bulk product more valuable by making it available on a timely basis when needed. We may seek analogies for the statement: "timely packages satisfy".

The following sequence could be generated by a brain-storming or synectics group:

1) "timely packages satisfy"
2) "prompt pieces please"
3) "regular intervals please"
4) "measured sugar-packages sweeten"
5) "regular breaks relax"
6) "coffee breaks relax"
7) "picnic occasions satisfy"

Statements 4), 6) and 7) could suggest ways to use powdered milk (pre-packed in small envelopes like powdered sugar-packs, requiring no refrigation) to accompany coffee breaks and picnics in places where refrigerated fresh milk is less convenient.

The ANALOGIZE module would operate by accepting a simple user statement as input and producing other simple statements as outputs, each of which is semantically linked to the input statement by virtue of the degree of similarity of their respective predicates. The input statement "Ships cut through water" might produce as output, for example, the following:

a) "knives slice bread"
b) "carriages run through the park"
c) "Chairmen facilitate discussion"

Each of these three output statements are retrievable through linking different semantic senses of the term "cut through". These linkages could be obtained by a data base of word lists and pointers such as are included in a good thesaurus ["Roget's", 1977], and a linked set of categorized "statement" lists. A statement would consist of two or three parts (arguments):

1) a subject noun or pronoun,
2) verb,
3) an object (sometimes omitted).

Statements could be categorized and grouped into lists according to their general fields of reference, which we will call "worlds" as in "the world of sports," "the world of finance", etc. (The notion of "worlds" is used in the synectics technique in order to aid people to produce metaphorical examples from a different sphere of concerns than the immediate problem being examined.) A relatively long list of statements about conditions and/or activities of a widely varying nature would comprise what we can all a "rich world", while a relatively short statement list would represent a "poor world". This analogy could be carried further by calling a system which contains many different "worlds" a "rich universe" and one with only a few worlds, a "poor universe". Precisely how "rich" a world or universe might have to be in order to generate "interesting" analogies is a question left open for subsequent experimentation.

The user could request analogous statements to his input statement from specific worlds or leave everything open to computer selection from any variety of worlds. The user could alternatively ask the system to generate metaphors by leaving the analogy linkage open and entering as input only a subject noun or only a predicate rather than a full statement, thereby allowing the system to find statements which contain these arguments and subsequently link them to the arguments of other statements. Thus, an input entry of "ships" could product outputs of

"knives", "carriages", and "chairmen", (assuming the statements previously mentioned are listed in some "worlds" and the verbs are linked by Thesaurus lists and pointers). The user could ask the system for the full statements behind the metaphorical connections or use his/her own imagination. The ANALOGIZE module, as described, would act as a simulated participant in a brain-storming or synectic session, "originating" metaphors and analogies and thereby stimulating and expanding the thinking of the user. In this manner, some of the advantages of group activity can be gained by a lone user using a DSS, without also obtaining some of the potential negative aspects of group activities.

LEVEL 3 MODULES

After having at least partially structured a problem and identifying some of its most significant features and dimensions, the user may be concerned with possible ways to restructure and redefine the problem. This is done by making further comparisons to other problems, by shuffling and recombining the parts of the problem, and by examining varying generalizations of the problem. The user may often want to cycle back and forth between using Level 3 and Level 2 modules. The level 3 modules are described below.

3.1 COMBINE

This module aids in restructuring the problem by carrying out the generation of alternative combinations as is done in morphological analysis (1a. in Exhibit III). For each dimension previously defined as being relevant, a set of alternative forms are selected or defined interactively with the user.

In the example given for 2.2 CLASSIFY, the alternative forms for two particular dimensions may be as follows:

	Dimensions	
	A.	B.
	Power Source	Control
1.	electric motor	1. chauffer driven
2.	gasoline engine	2. user driven
3.	nuclear engine	3. automatic
4.	steam engine	4. semi-automatic (user-monitored)
5.	solar	

If we considered only these two dimensions, there are 20 possible combinations: any one of the five types of power sources can be combined with any of the four types of control. If we consider other dimensions and their alternatives, the number of possible combinations that comprise a design alternative increases. One way to define a "creative" solution, is that it represents an "unusual", as well as useful, combination of known elements. A means of searching through a large number of possible combinations should therefore support creativity.

The computer could be programmed to exhaustively generate all combinations, taking one alternative from each dimension at a time. These could be arranged in groups for display and evaluation by the user. The user could specify which combinations are to be rejected immediately as not being of enough interest to

save for further evaluation, which appear to merit first priority in subsequent assessment, and which are to be saved but not presented as being of primary interest. If the number of combinations is too large to evaluate each one after it is generated, the system could enable the user to select dimensions to be combined in stages. The user could eliminate some partial combinations at the end of a stage before bringing the next dimension of interest into consideration. This inter-active staged combination generation (similar to a dynamic programming approach) would require user control throughout the process, but could result in a net saving of user time compared to having to examine all combinations at the end of computer processing.

3.2 COMPARE

This module would facilitate user comparison of two alternatives at a time by displaying juxtaposed descriptions of respective dimensions.

For example, alternatives 2 and 3 previously described under 2.2 CLASSIFY can be seen to be alike with respect to their dimensions of Container type, Control, and Routing. They differ only in the Power dimension, for which alternative 2 uses a conventional electric motor and alternative 3 uses a nuclear engine.

Upon examination, the user may want to exchange or otherwise modify selected dimensions in either of the paired alternatives being compared. The user would begin by specifying, either explicitly, or implicitly (by describing some of its characteristics), the two alternatives to be compared. The system would then retrieve these paired alternatives for user review and possible modification. The user may finally want to save some alternatives for later additional comparison and assessment and eliminate others. This comparison analysis would facilitate techniques 1a, 1b, 1c, 3a and 3f in Exhibit III.

3.3 GENERALIZE

This module would assist the user to formulate successive generalizations or abstractions of a particular entity (as is done in 3b in Exhibit III). For example, a generalizing sequence that could be generated starting with "pencils" is: "pencils - pencils and pens - hand-held writing instruments - typewriters - visual print display devices - written long-distance communication systems - two way written communications systems".

The capability of formulating successively more general boundaries often helps to eliminate a too restrictive view of the problem. This technique has been applied to defining the scope of a business organization and thereby redirect definitions of goals and objectives. In the above example, someone starting out to think of themselves as being a pencil manufacturer could end up by searching for new product lines that would come under the broader heading of "visual print display devices."

The user of this module may alter the sequence generated by requesting that the system return to any point and regenerate a sequence along one more different dimensions. For example, the sequence previously given follows the dimension of "manually created written communication". An alternative could be to follow another path based on a different method of capturing a message, such as electronic voice recording, and an alternative means of sensing the message, such as aurally rather than visually.

The DIMENSIONALIZE and CLASSIFY modules can be utilized in creating the entity network needed to support the GENERALIZE module.

LEVEL 4 MODULES

The fourth and highest level aids the user in tying together a full view of the problem area, its essential functional characteristics, and constellations of

decisions and action guides called policies. The modules that support these modeling functions are described below.

4.1 BUILD SCENARIO

This module helps in constructing a scenario by:

a) prompting the user and leading him/her through the consideration of outcomes, controllable and uncontrollable impact factors, and alternative event sequences; and

b) acting as a secretary in recording and displaying back to the user the scenario as it progresses and in its totality, for modification and ultimate user approval.

The user may want to cycle back to 1.2 FETCH EXPERIENCE and 1.3 FETCH KNOWLEDGE in the early stages of building a scenario.

Independently constructed scenarios can also be placed in a central data base so that other scenario builders can inter-actively display them, assess them, and take them into account in building their own scenario. Used in this manner the computer system can act as a neutral communication device. An option that could be included in this communication device would request that a user's assessment and modification of a scenario be accompanied by a short rationale stating the reasons for the assessment or modification (this need not consist of proof). This assessment and commentary could then be displayed by the originator of the scenario for his/her consideration. This procedure would facilitate an anonymous Delphi method of building scenarios.

The structured, summarized format required for expressing scenarios would also impose a beneficial discipline on scenario writers which would facilitate understanding.

4.2 BUILD POLICY

This module would aid in building a policy in a manner similar to that of BUILD SCENARIO. That is, it would both lead and prompt the user as well as act as a recording secretary to record, display and modify a policy in different stages of completion. The user may want to work first with or cycle back to 1.4 FETCH WISDOM and 4.1 BUILD SCENARIO.

The independent construction of policies, cross-comparison and Delphi-type option discussed above for inter-actively building scenarios could also be features of this module. The succinct and standardized format imposed on policy authors would also be a benefit.

A common benefit of all of the FETCH modules, as well as the BUILD SCENARIO and BUILD POLICY modules would be that they would result in establishing a common retrievable organizational memory of total context material. In this way, misunderstandings, loss of learning due to personnel changes, duplication in problem solving, and duplicated construction and maintainance of files and reports, are all minimized.

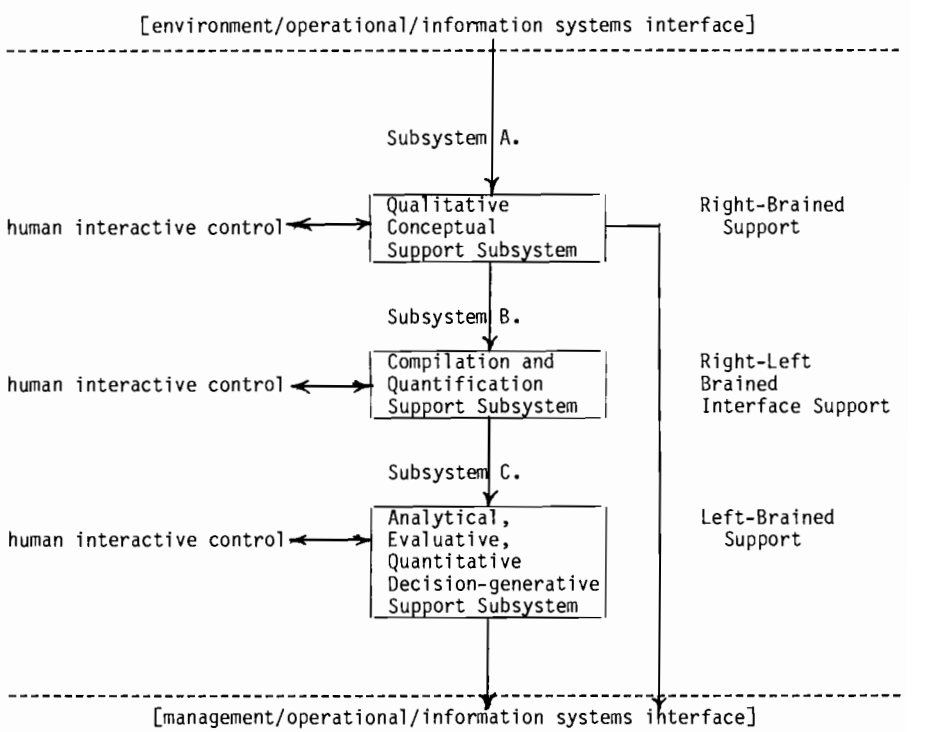
THE RIGHT-LEFT BRAIN INTERFACE AND TOTAL DDS

With advances in understanding complex social systems, problem areas once categorized as open and unstructured may become more structured and quantifiable. In such cases, the bridge between right and left brained support will often lie outside the scope of a DSS. In these cases, the user will apply what has been learned from various sources and, if the user desires, begin to work with the more quantitative, analytical tools available within a left-brained DSS.

In other cases, users of right-brained DSS will themselves begin to impose structure and quantitative thinking on initially open, qualitative problems. Once they reach this point, they may be further aided by an interactive dialogue with another type of computerized support system that is designed to aid in the transition of qualitative decision problems into quantitative decision problems. This type of DSS can be characterized as the Right-Left Brain Interface. A Total DSS (TDSS) would then provide: 1) user access to the type of right-brained modules previously described, 2) the more standard leftbrained type of decision support, and 3) modules to support the interface between them. (See Exhibit V)

We will not discuss functional specifications of the Right-Left Brain Interface in detail in this paper. Some of its functions however, should probably include the support of scaling methods, the automatic translation of qualitative descriptions into partially complete mathematical expressions, and the automatic translation of verbally expressed scenarios into partially complete mathematical or symbolic models. Some of these functions would utilize the Decision Aids listed under Technique 2 in Exhibit III.

EXHIBIT V TOTAL DECISION SUPPORT SYSTEM



CONCLUSION

Is building such an extensive, generalized right-brained DSS justifiable? It has been observed that existing left-brained DSS' are often not used to really solve problems and make decisions so much as to document prior decisions [Alter, 1977]. But even if current management practice only rarely solves problems using computer support, the new mind set of the next generations of managers and the environment they will work in will inevitably lead to the extensive use of computer supported decision processes.

Qualitative decision-making, probably much more significant in its impact, will ultimately be served no less than quantitative decision-making. The mundane harbinger called "word processing" will lead ultimately toward a variety of advanced computerized methods of qualitative processing. We suspect that right-brained DSS and their accompanying qualitative data bases will burst upon the scene in a future generation of computer applications that will far overshadow current numerically-based usage. The practical orientation of DSS, its principle of user control, and its conceptual foundation in a behaviorally described multi-stage decision process, makes it the most suitable unifying framework for the advancement of qualitative computer processing. DSS design criteria will facilitate the difficult tasks already being faced by those working in information retrieval, CAI, artificial intelligence, and other related areas.

The survival of human civilization appears to require rapid mental adaptation in order to better manage volatility and complexity. The speed of the adaptation required obviates depending upon either evolution of our species or traditional forms of training.

An evolving symbiosis between humans and computers is the prerequisite for the conceptual break-throughs needed. In human-computer decision systems, humans will not relinquish control and create a dictatorship of robots. It is not in our nature to do so by intent and only the most unlikely folly could create such a futuristic nightmare.

The danger is not so much that we will produce what we have previously [Young, 1978] called Type III ("machine-master") interaction, but that we will not produce Type I, "creative interaction systems", quickly enough. From where we currently find ourselves, building extensive right-brained DSS or Total DSS is a large step. But we need to enlarge our vision of DSS and begin in that direction now. In the race to manage the survival and re-creation of desirable social systems we must speed up our use of computers not to supplant, but to support human creativity.

REFERENCES

Allen, M. "Morphological Creativity", Prentice Hall-Hall, 1952

Alter, S. "Why is Man-Computer Interaction Important for Decision Support Systems?" Interfaces, Vol. 7 No. 2, Feb. 1977.

Barker, W. "Brain Storms; A Study of Human Spontaneity", Grove Press, 1968

De Bono, E. "Lateral Thinking; Creativity Step by Step", Harper and Row, 1970

Fry, E. "Teaching Machines and Programmed Instruction", McGraw Hill, 1963

Keen, P.G.W.; Scott Morton, M.S., "Decision Support Systems: An Organizational Perspective", Addison-Wesley, 1978

Little, J.D.C., "Models and Managers: The Concept of a Decision Calculus" Management Science, vol. 16, no. 8, pp 466-485, April 1970

Nadler, G. "Work Design: A Systems Concept", Richard D. Irwin, 1970

Nystrom, H., "Creativity and Innovation", John Wiley and Sons, 1979

Osin, L. "A System to Produce CAI courses from Logically Structured Educational Material". Doctoral Dissertation, Israel Institute of Technology (Technion), June, 1974

Osborne, A. "Applied Imagination", C. Scribner & Sons, 1957

Prince, G.M. "The Practice of Creativity", Harper and Row, 1970

Restak, R. M. "The Brain, The Last Frontier", Doubleday and Co., 1979

Rickards, T. "Problem Solving Through Creative Analysis", John Wiley and Sons, 1974

"Roget's International Thesaurus" Fourth Edition, Harper and Row, 1977

Salton, G. (editor) "The SMART Retrieval System: Experiments in Automatic Document Processing", Prentice - Hall, 1971

Simon, H.A. "The New Science of Management", Harper and Bros., 1960

Stevens, M.E. "Strategies for Organizing and Searching" in "Changing Patterns in Information Retrieval", Tenth Annual National Information Retrieval Colloquium, May 3-4, 1973, Philadelphia, PA, ASIS, 1974

Young, L.F. "Another Look at Man-Computer Interaction", Interfaces, Vol. 8, No. 2, pp 69-71, Feb. 1978.

Yukawa, H. "Creativity and Intuition; a Physicist Looks at East and West", Harper and Row, 1973

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

SPECIFICATION OF MODELING KNOWLEDGE IN DECISION SUPPORT SYSTEMS

Robert H. Bonczek
Clyde W. Holsapple
Andrew B. Whinston

Krannert Graduate School of Management
Purdue University
West Lafayette, IN 47907
U.S.A.

The DSS literature is plagued with muddled descriptions of implementations. Without a framework for comparison, discussion would be futile. This paper presents a theoretical framework for analyzing a DSS. A multilevel approach is employed to reflect the objectives and constraints of the many different DSS users. A design mechanism, based on data base management, is motivated by this framework.

INTRODUCTION

Due to the newness of the decision support field, development tools specifically oriented toward building decision support systems are practically nonexistent. Tools do exist that can be usefully applied to the task of building a decision support system (DSS). Examples include programming languages and generalized data base management systems. However, these were not initially conceived as tools for DSS construction and therefore are not ideally suited for use by a DSS developer. For instance, a data base management system does not aid a developer in managing models and their interactions with each other and with data.

Recent works have examined the issue of DSS development tools. In [1] and [2], desirable properties for such tools are discussed. A knowledge system definition language is outlined in [3]. The purpose of this article is to extend those earlier ideas, with a special focus on tools for the collection and utilization of knowledge-system contents. These include a language (and processor) for gathering experts' knowledge about the utilization of models and data. Also introduced is the notion of an interactive DSS development system. The tools explored here are general, in that they are not oriented toward building decision support systems for a particular area of decision making. They are equally useful for building a financial planning DSS, a pollution control DSS, or a marketing DSS. Given that these tools exist, we then introduce an approach for operationalizing the problem solving process based on knowledge system contents. The approach involves an extension of data base management to model management.

In particular, we focus on the problem processing framework in this paper and on the task of the DSS designer. This in no way is meant to minimize the importance of the other aspects of the problems. This is especially true of the interface between the DSS and the manager who uses the system. However the DSS designer needs the appropriate set of processing tools, in order to construct a viable system before managers can become users.

DSS FRAMEWORK

An examination of DSS development tools must occur within the context of a framework for understanding the constitution of decision support systems. A generic framework for decision support systems [1] maintains that any decision support

system can be viewed as having three components. As shown in Figure 1, these are its language system (LS), knowledge system (KS), and problem processing system (PPS). A user states problems for a DSS to solve by using a language system. A decision support system's KS holds facts about an application area that are relevant to solving problems arising for that application. The problem processor lies at the heart of a DSS, accepting problems represented with the LS and utilizing application-specific knowledge represented in the KS in order to generate information for decision support.

The LS and KS are representation systems. The PPS is the dynamic DSS component; it is a system that displays some behavior. The syntactic and semantic rules of a LS determine the permissible problem statements that can be posed to a DSS. Similarly, a KS is characterized by the facilities it furnishes for the representation and organization of knowledge.

Language systems vary in terms of the level of procedurality that they require for expressing a problem [1]. At one extreme are procedural languages that allow a user to state a problem by specifying the procedural steps (involving retrieval and/or computations) to be used in solving that problem. At the opposite extreme are nonprocedural languages that permit a problem to be specified by merely stating the characteristics of that problem's solution. For example, DISPLAY EARNINGS-PER-SHARE FOR YEAR = 1983, GROWTH-RATE = .7 AND SHARES = 583400 is a nonprocedural indication of the characteristics of a problem.

Knowledge systems vary not only in terms of the knowledge they contain, but also utilize differing approaches to knowledge representation and organization. If a data base is employed, it may offer hierarchical, network, or extended-network [4] constructs. Other reasonable knowledge representation methods include those from the artificial intelligence area, such as the predicate calculus and production system approaches. Of course, it may be desirable to integrate two or more of these approaches for use in a single knowledge system [5]. The focus of the latter portion of this presentation is upon the treatment of modeling knowledge.

As the software component of a decision support system, a PPS is the formal specification of a DSS's behavior patterns. Clearly, the coding of a PPS depends on the natures of the LS and KS with which the PPS is associated. A problem processor may be more or less general and it may display rudimentary or extensive problem solving abilities. As a minimum any PPS must have the abilities to gather information from a user (expressed via the LS) and from a knowledge system. The latter involves manipulation of data in the KS, while the former ability involves the manipulation of LS expressions.

A PPS must also possess the ability to explicitly recognize problems by transforming problem statements into appropriate executable plans of action. A problem processor's problem recognition ability is comparable to the familiar notion of compilation (though it could also be viewed from an interpretive standpoint). A PPS has explicitly recognized a problem when a problem statement has been converted into a detailed procedural specification which, when executed, yields an answer to the problem. For language systems that require a procedural problem statement, the PPS problem recognition ability is at most rudimentary. Nonprocedural problem statements may necessitate a more sophisticated problem recognition ability.

The explicit procedural specification resulting from problem recognition may involve a computational model, as well as retrieval (be it traditional or inferential). If a user directly specifies or selects a model through the LS, then there is no need for the PPS to recognize the modeling problem; the user has already done so. A highly sophisticated problem recognition ability may be required if a PPS must itself select or formulate a model. In the case of model building, the PPS must have a capability beyond information collection and problem recognition abilities. It must have the ability to formulate models, as a service

to the problem recognition ability

Where models are involved, another vital PPS ability is that of analysis. Analysis is the process or interfacing models with data in order to generate some beliefs, facts or expectations. When a PPS has arrived at the explicit recognition of a model and the data that are to be used by it, an analysis mechanism starts and controls the execution. This may merely involve a jump, or it may entail activities such as module loadings and parameter initializations. The precise nature of an analysis mechanism depends largely on how models (modules) are maintained within the DSS. For instance, are they embedded in the PPS code or are they held in the KS?

If a PPS is not oriented toward a particular application or application area, we shall refer to it as a generalized problem processing system (a GPPS). GPPS code is invariant to changes within an application area and to differences across application areas. In other words, a GPPS can be used to build decision support systems in a variety of application areas, without necessitating modifications to the GPPS software. It is this trait that makes a GPPS valuable for the production of customized decision support systems. Problem processor generality implies that the associated KS and LS must also be general in some respects.

For instance, if the grammatical structure of a LS is application-specific, changes are likely to be needed in the associated PPS whenever changes are made in the LS grammatical structure to handle a different application. Similarly, the knowledge representation and organization techniques of the KS associated with a GPPS cannot vary from one application to another. Different data manipulation and organization. Since these facilities are a part of the GPPS, they cannot be changed from one application to another.

The foregoing is a brief outline of the generic DSS framework of [1], examining the nature of the three DSS components. Various types of knowledge and language systems were identified. The PPS of a decision support system has several abilities that are useful in problem solving and it must, of course, be consistent with the DSS's language and knowledge systems. This background allows us to consider the notion of a generalized PPS, as a tool for DSS construction, and to subsequently focus on the issue of model management in the context of a DSS built from a GPPS.

It must be stressed that this exposition is that of a design framework. An implementation method is not being suggested at this time. It is most likely that the tools described herein (e.g. Horn clauses) would not be used internally; the actual implementation technique must be computationally effective and viable, from both the standpoints of computer science and artificial intelligence.

A GPPS is perhaps the central tool for DSS development, but there are a number of necessary allied tools. These are illustrated in Figure 2: KSDL (Knowledge System Definition Language), EKL (Expert Knowledge Language), and DSSDS (DSS Development System). They are aids for collecting information for a decision support system's KS. The KSDL has been examined in [3] and receives no further attention here. The EKL and DSSDS are explored.

As shown in [1], Horn clauses provide a very convenient approach to representing the knowledge of an expert, about how to judiciously use models and data. This convenience is with respect to conceptualizing the processing that utilizes an expert's knowledge. Horn clauses do not offer a particularly convenient form for initially capturing the expert's knowledge. The purpose of an Expert Knowledge Language is to provide an English-like mechanism for capturing expert knowledge, in a form that can easily be translated into Horn clauses or equivalents. At most, one model predicate is permitted per Horn clause. The Horn clauses are stored in the Knowledge System (e.g., as occurrences in the data structure). The processor that performs the transformation and storage, could also analyze EKL

statements for consistency.

Fragments of knowledge gathered with the EKL are used to determine the flow of processing needed in order to respond to a DSS user's request. In theory this is accomplished by having the GPPS perform resolution (at least conceptually) on the set of Horn clause axioms, as explained in [1]. However, combinational problems make resolution an unattractive approach in practice. As an alternative, a DSS developer could utilize a DSS development system (DSSDS), in order to effectively select the Horn clauses (i.e., resolution path) needed to correctly answer a given DSS problem or class of problems. The DSSDS would be interactive, accepting as initial input a statement of the problem to be solved [1]. This problem statement would be given by the DSS developer, not the DSS user. The DSSDS responds by determining the first step for each of the potential solution (i.e., resolution) paths. This determination is made by consulting the Horn clauses, which have been stored according to a data base structure that expedites the determination process. The first steps are displayed to the DSS developer for selection. There are several options for the nature of this display, including the display of the model name for each alternative step or the display of the EKL form of each alternative. If one of the alternative steps does not involve a model, then the initial problem may be solved without model execution.

When a selection is made the process is repeated. This iteration continued until a solution methods has been determined. The result is that a resolution path has been interactively determined for a given problem statement. The resolution path is stored in the knowledge system, together with the original problem statement. When a DSS user states the problem, the GPPS uses the stored path to generate the answer. It is of value to make use of a macro definition facility similar to those available in sophisticated data base query systems (e.g., [6]). This simplifies the problem statement and allows it to be easily parameterized. Macro definitions are, of course, stored in the KS. Another value of macros is that two (or more) different macro names can be assigned to a single problem statement if two (or more) different solution paths for the same problem statement need to be stored. Thus the DSS developer uses the DSSDS to customize a DSS for a particular user's or users' needs.

The alternative to an interactive DSSDS are a GPPS that performs resolution, or a problem statement language with which the DSS user can specify a resolution path. The first corresponds to a generalized query system for network data bases that somehow selects one of the many paths which can be used in answering a retrieval request. Different paths can yield different answers and this alternative offers no way to distinguish among them. In the case of a DSS, this type of drawback is overcome with meta-axioms [1]. The second alternative corresponds to a generalized query system that allows a path clause in the problem statement. The path clause is very simple to use in the data base retrieval case. In the case of a DSS language system it would need to include model sets, as described in [3].

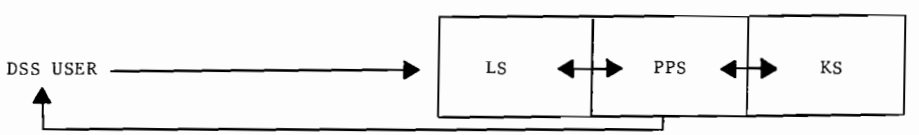


Figure 1
DSS Framework

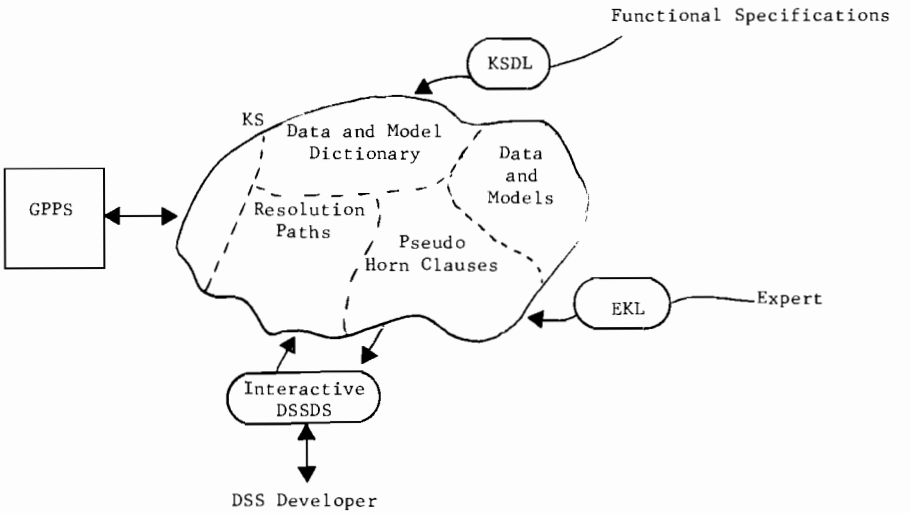


Figure 2
Tools for Constructing a Knowledge System

PROBLEM PROCESSING

Once a resolution path has been determined for a DSS user's problem, it must be used by the GPPS to generate a solution. We shall conceptualize this generation using modified relational data base terminology. The modification consists of adding one more command to the traditional set of relational algebra commands (e.g., PROJECT, JOIN, etc.). This new command is called EVAL. It has two relations and a model as arguments:

EVAL (rel-1, model, rel-2)

The tuples of rel-1 are to be used as input to the model. EVAL has the effect of executing the model with these inputs with the result forming tuples in rel-2. EVAL assumes that a model's inputs can be expressed in a single first normal form (1NF) relation and that its outputs are written to a single 1NF relation. This is not a particularly limiting assumption for a very large class of models. If the data needed by a model exist in two or more relations, the relational JOIN, PROJECT and DIFFERENCE operators can be used to derive the needed input relation before EVAL is involved. Similarly these three traditional operators can be used to operate on the relation generated by EVAL, yielding a new relation for a report or for later use as input to another EVAL command.

This simple modification to the relational data base management approach renders it much more suitable to addressing the issues involved in DSS knowledge representation and problem processing. Given the resolution path (and hence a sequence of models), the GPPS must be able to determine the appropriate JOIN, PROJECT and DIFFERENCE operators (or the appropriate SELECT statement) to invoke prior to each EVAL. This is easily deduced from the left hand side of the Horn clause. The precondition arguments of the model predicate give the form of the input relation. The right hand side gives the form of the relation that results from EVAL. While the modified relational approach is useful in conceptualizing the model execution process of a GPPS, it has a number of drawbacks from the operational standpoint.

Chief among these are the relatively poor storage and processing performances that typify attempts to operationalize the relational approach. In addition there are its relatively cumbersome (user-unfriendly) method for representing relationships and its inherent data integrity difficulties. One possibility for overcoming these problems is adopting an extended network approach that has been modified to include model sets as a part of its data description language [3] and EVAL as part of its data manipulation language. The remainder of this paper explores such an approach.

DESIGNING HORN CLAUSES: AN EXAMPLE

As an example of the design of a DSS, we will consider a user who, in the course of a DSS session, needs to forecast future inventories of one (or more) items. Part of this forecast will involve the Economic Order Quantity (EOQ). This data must be the result of the execution of several models, since the calculation of EOQ is a function of demand, ordering and holding costs, and these values must themselves be predicted for the future period(s) in the forecast. However, the DSS user would not necessarily be aware of all this; he or she would simply need to enter a query expression such as

PREDICT EOQ FOR ITEM i in PERIOD p.

This query explicitly references the output of a model in the KS of the DSS, namely, EOQ. Let's now develop a Horn clause representation of that model, and all associated models. (This syntax of this and future query expressions is derived from the query system QRS [6].)

We can (internally) refer to the output of the EOQ model as O. Thus we can state the following high level, Horn clause form description of the EOQ modelling process:

$$OC(c_0, i, p) \wedge DMD(d, i, p) \wedge HC(c_h, i) \wedge EOQ(c_0, d, c_h, q) \rightarrow O(q, i, p) \quad (1)$$

where OC is the predicate defining the ordering cost c_0 for item i in period p (recall that i and p are parameter values stated in the user's query, i.e., derived from the LS of the DSS). The DMD predicate specifies the expected demand d for item i in period p, while the predicate HC calculates the average holding cost c_h for item i. The model itself is represented by the predicate EOQ; the first three arguments are the input values to EOQ, the fourth (q) is the output, i.e., the optimal reorder quantity. The predicate O then states that q is the optimal reorder quantity for item i in period p.

In this predicate expression, variables other than i and p are implicitly assumed to be universally quantified. Further, they represent single values, i.e., not arrays or relations.

We now must specify how the input values are computed. Since this is a prediction problem, these values will themselves be the output of some model. Hence, we must develop Horn clauses with the predicates OC, DMD and HC appearing on the right-hand side.

We will predict the ordering cost c_0 by using linear regression over the past orders for the specified item. If the prediction model is REGRESS(T, C, p, c_0) where T is an array of times (the independent variable), C is the corresponding array of historical costs (the dependent variable), p is again the time period for which the forecast is desired (independent variable) and c_0 is the ordering cost value, then a Horn clause can be formulated as:

$$\text{ORDERS}(i, C, T) \wedge \text{REGRESS}(T, C, p, c_0) \rightarrow \text{OC}(c_0, i, p) \quad (2)$$

The ORDERS predicate refers to the historical record of ordering information for item i. Assuming that this information is available in the data base of the KS, this predicate is defining a retrieval operation from the data base, i.e., a query. We shall return to this issue below.

Naturally, the correlation coefficient of the regression should be tested before using the computed values. This kind of performance issue is discussed in [1].

The forecast of expected demand has a parallel development:

$$\text{SALES}(i, Q, T) \wedge \text{REGRESS}(T, Q, p, d) \rightarrow \text{DMD}(d, i, p) \quad (3)$$

Here T is again the time period array, Q represents the historical quantity sold. The REGRESS model is performed with independent variable T and dependent variable Q to predict in period p the demand d. As above, SALES represents information that must be retrieved from the KS.

We shall assume that a special model, AHC, exists for computing the average holding cost c_h. The form of this expression might read:

$$\text{IC}(i, H, T) \wedge \text{AHC}(H, T, c_h) \rightarrow \text{HC}(c_h, i) \quad (4)$$

where IC retrieves the inventory costs stored in the KS for item i.

The definition of the EOQ model is thus given by expressions (1) - (4). These would exist within the KS of the DSS, and be referenced implicitly through user queries. The expressions (2) - (4) are in no way subordinate to (1); in fact, they can be referenced in queries without regard to EOQ:

PREDICT DEMAND FOR ITEM i IN PERIOD p

is such an example.

Moreover, some of these expressions define queries for performing retrieval from the KS. Expressions (2) - (4) would give rise to the following queries, respectively:

LIST ORDERING-COST, TIME FOR ITEM i (2q)

LIST DEMAND-QUANTITIES, TIME FOR ITEM i (3q)

LIST ACTUAL-HOLDING-COST, TIME FOR ITEM i (4q)

The operation of these queries would, of course, depend upon the underlying structure of the data base within the KS.

The processing of the user's original request will then take the form: identify the relevant Horn clause for the referenced model; substitute the parameter values specified in the request; for each input predicate, either perform retrieval from the KS, or locate a Horn clause defining that input (this step is of course recursive); then use EVAL to execute the stored model.

Three further points are worth mentioning here. One is that, in the specification of the predicate expressions, all variables are implicitly universally quantified. Thus, if the user's original query does not explicitly reference the item or period, then the Horn clause formulation is still valid; the model will be evaluated for all elements.

In the same vein, the user can specify values for other predicate variables as

well. This leads to natural processing of "what if" type queries, e.g.,

LIST EOQ FOR ITEM i IN PERIOD p IF DEMAND IS $\hat{d}$

In processing this request, the $\hat{d}$ value is substituted for d in expression (1); since all values of this predicate are specified, there is no need to even reference expression (3). It is a simple task to incorporate a memory feature to support a posteriori "what if" processing as well (i.e., to permit the use of past values of d for determining what the ordering value should have been compared to the valued actually used).

Finally, there is no requirement that says the data base of the KS must be designed before the modeling information is incorporated. Thus the predicates defining the retrieval operation in (2) - (4) can also be used to design the underlying data structure required to support the model. This can be achieved through a normalization process on the retireval predicates, for example.

This presentation has developed a set of predicate expressions for specifying modeling knowledge independently of any data structure. In [3], a similar set of predicates is derived after the data structure has been developed. Queries (2q) - (4q) developed above can be depicted as in Figure 3. Here at the top is the data base structure of [3]; the PC and PQ elements refer to the input relations for the regression models; links are made from the appropriate record types to the appropriate models. In [3] an alternate syntax is developed for specifying the relationships among models and data.

THE EVAL COMMAND

To fully understand the operation of a GPPS that processes data and models symmetrically, it is first valuable to consider the correspondence between the data manipulation language (DML) of a data base management system and to develop a parallel model manipulation language (MML) for processing the models.

Consider first the model defined by the predicate expression

$$A(x) \wedge Q(x, y) \rightarrow M(y)$$

where A is a data base record type including data item x, Q is a model computing y as a function of x, and M is a parametric predicate with argument y. This expression is depicted in Figure 4. Q is shown (properly) as a 1:1 set.

If M represented a data base record type, and Q were implemented not as a model within a DSS but as a subroutine of a DML program, then we could perform the following typical DML processing to compute a value for y:

find the appropriate occurrence of A

perform the computation Q (execute the model)

create and store the computed value as a new occurrence of M

associate the A and M occurrences through the set relationship Q.

The actual commands for performing these operations are shown in Figure 5. The command forms are adapted from the extended network DML of MDBS [7].

An alternative ordering of these commands appears in Figure 6. Here the new occurrence of M is first created; then this occurrence is associated with a record occurrence of A; and finally the subprogram Q is executed. The

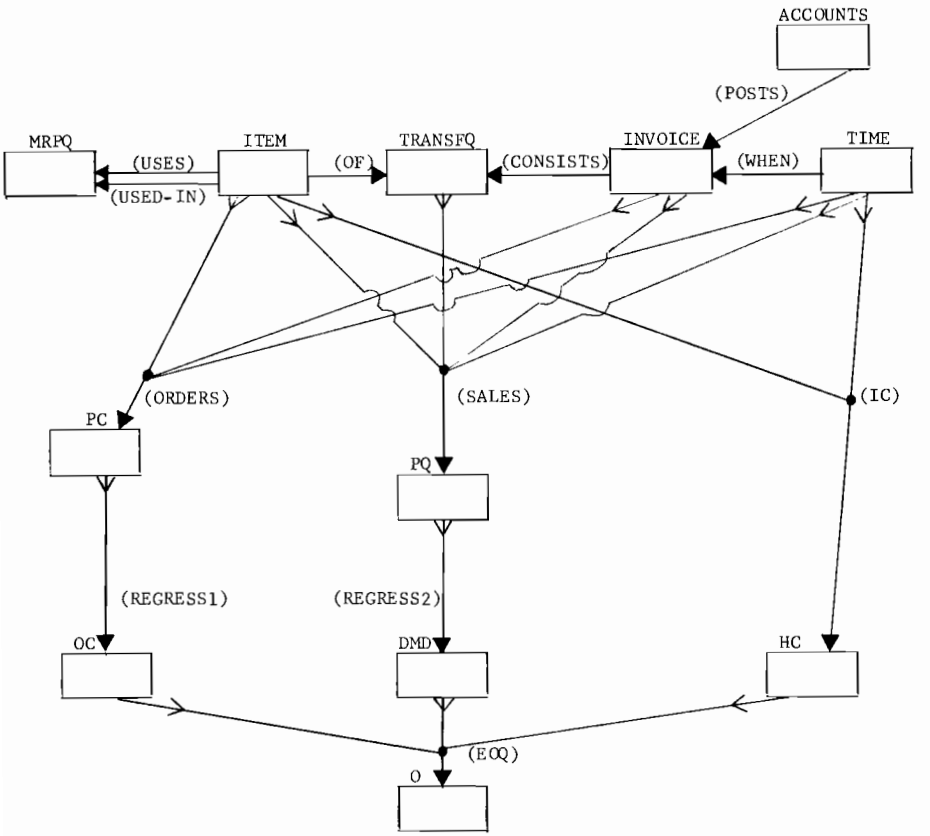


Figure 3
Sample Logical Structure with Model Sets

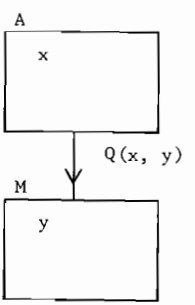


Figure 4
Defining Model Q

resultant value is stored (put) into the new occurrence of M. While not as elegant as the first solution, this second approach is also correct.

The value of the second approach becomes apparent if we consider the situation of a model with two input parameters. We will refer to this predicate expression:

$$A(x) \wedge B(y) \wedge R(x, y, z) \rightarrow N(z)$$

```
input x;
frk(A, x);                          /*FIND THE APPROPRIATE OCCURRENCE OF A*/
perform Q(x, y);
soc(Q);                             /*MAKE THE A OCCURRENCE THE OWNER OF
                                        SET Q*/
crs(M, y);                          /*CREATE AND STORE*/
ims(Q);                             /*INSERT THE NEWLY CREATED OCCURRENCE AS
                                        A MEMBER OF SET Q*/
```

Figure 5
Computing and Storing Q(x, y)

```
crs(M, ϕ);                          /*CREATE M WITH A NULL VALUE*/
smc(Q);                             /*MAKE THE M OCCURRENCE THE MEMBER
                                        OF Q*/
input x;
frk(A, x);                          /*FIND THE APPROPRIATE OCCURRENCE OF A*/
ios(Q);                             /*INSERT THE OCCURRENCE FOUND AS AN
                                        OWNER OF THE SET Q*/
perform Q(x, y);
putm(Q, y);                         /*STORE THE COMPUTER VALUE*/
```

Figure 6
Alternate but Equivalent Computation of Q

Here A and B represent data base record types, R is a model that computes z as a function of x and y, and N the parametric predicate. This is depicted in Figure 7. Note that the model set R is a multiple-owner set; it is also an N:M set. That is, each occurrence of N will be associated with two owner occurrences (an A occurrence and a B occurrence), while each A occurrence can be used in conjunction with several B occurrences to compute many R values.

Assume once again that R is available as a subprogram. Using the second form of processing as our template, we can compute the z value of N in the following way: build the N occurrence; find the A occurrence, and associate it with the N occurrence; find the B occurrence, and associate it with the N occurrence; then perform the computation, and store the value in the new N occurrence. The appropriate DML code is shown in Figure 8.

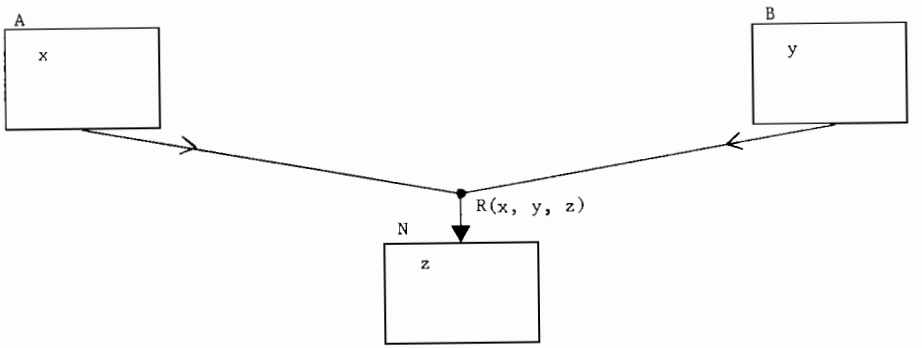


Figure 7
Defining the Model R

```
crs(N, null);
smc(R);
input x;                              /*input A*/
frk(A, x);
ios(R);
input y;                              /*input B*/
frk(B, y);
ios(R);
perform R(x, y, z);
putm(R, z);
```

Figure 8
Computing R

Note the necessity of first creating the N occurrence. Since this occurrence is to be associated with several other occurrences, we first build the parametric (output) occurrence, and then iteratively associate the date (input) occurrences. If we were to attempt to first locate the data (input) occurrences, we would be unable to keep track of the first such occurrence once the second has been found.

It is readily apparent that this procedure is extendable to three or more input predicates, simply by adding the appropriate sequences of code that will find the particular record occurrence, and then insert it as an owner of the model set.

Now drop the assumption that Q is a subprogram of a host language DML program. Instead, regard Q as a model to be invoked by a DSS developer. To do this we incorporate a new command into the MML for performing model evaluation. We refer to this command as EVAL. The single argument of the EVAL command is the name of the model to be evaluated. EVAL computes and stores a value for the current member occurrence of the specified model set, as a function of that occurrence's owners in the set. It is necessary for all of the owners to be associated with the member occurrence before the EVAL command is invoked. The processing of the previous two examples, with EVAL incorporated, is given in Figure 9.

```
crs(M, null);
smc(Q);

input x;
frk(A, x);
ios(Q);

eval(Q);
getm(Q, y);
```

```
crs(N, null);
smc(R);

input x;
frk(A, x);
ios(R);

input y;
frk(B, y);
ios(R);

eval(R);
getm(Q, z);
```

Figure 9
Using the EVAL Command

A more extensive example of MML logic, using the logical structure of Figure 3 is given in Figure 10.

```
frk(ITEM, i)                          /*Find item i*/
soc(USES)
For all desired orders  y  associated with item i
     {  crs(PC, null)                 /*Create a PC occurrence*/
        smc(ORDERS)
        sco(USES)                     /*Associate item i*/
        ios(ORDERS)
        frk(INVOICE, y)               /*Associate order y*/
        ios(ORDERS)
        smc(WHEN)                     /*Associate time*/
        ios(ORDERS)
        eval(ORDERS)
        }
crs(OC, null)
smc(REGRESS1)
soo(ORDERS, USES)
For all newly created PC occurrences
     {  ios(REGRESS1)
        fnm(ORDERS)
        }
eval(REGRESS1)
:
:    /*Create DMD and HC occurrences similarly*/
:
crs(O, null)
smc(EOQ)
scm(REGRESS1)
ios(EOQ)
scm(REGRESS2)
ios(EOQ)
scm(HOLD)
ios(EOQ)
eval(EOQ)
```

Figure 10
Computing EOQ

This is the approach taken by a DSS developer using the MML, in the absence of a high level GPPS and a DSSDS. We shall examine how to automate this within a GPPS in the next section. First there is another aspect of EVAL that must be addressed. As the foregoing example illustrates, EVAL must use one or more occurrences of one or more record types in order to execute a model. The associations of record occurrences through a model set clearly indicate which occurrences are involved. However, the occurrence structure (in general) provides no information about which occurrence is to be used first (second and so forth) and which occurrences are to be used in conjunction with other occurrences. The MML user should not be expected to be concerned about such details every time a model set is used.

AUTOMATING MODEL EVALUATION WITHIN A GPPS

The examples of the last section show that the approach to using the MML is very routine and repetitive. This suggests that this processing can be automated within a GPPS. Three main problems to be solved in order to accomplish this automation are:

1. determining the sequence of model sets to be evaluated for a given problem statement.

2. generalizing the processing logic for a model set.

3. specifying conditional input for an evaluation process.

The model set sequence is determined through the DSSDS as described earlier. The general processing logic for a model set consists of the following steps:

a. creating a null occurrence of the member record type.

b. finding appropriate occurrences of owning record types and connecting them to the null occurrence.

c. evaluate the model set for the null occurrence.

Step b in this processing logic is related to the problem of specifying conditional input for an evaluation process. That is, to which occurrences is it "appropriate" to connect the null occurrence. The mapping stated for the model set partially determines the appropriate occurrences. For instance, it rules out all INVOICE occurrences that are not related to the ITEM occurrence that is being connected to the null occurrence. Even more selectivity would be allowed if we extend the QRS mapping statement introduced above to include the QRS conditional clause. For example,

LIST ITEM, INVOICE, TIME FOR TIME $\preceq$ xxx, TIME, $\preceq$ yyy

The times over which the average order for an item are to be computed are indicated by the parameters xxx and yyy. These values are specified by a DSS end user through the DSS LS and substituted into the QRS map when the ORDER model set is evaluated by the GPPS.

A final topic related to conditioning the input for evaluation involves control over how many occurrences of the model set's member record type are to be created (i.e., how many times the GPPS is to invoke EVAL for a given model set). This too can be controlled by the QRS mapping statement:

LIST ITEM, INVOICE, TIME BY ITEM FOR TIME $\preceq$ xxx, T $\preceq$ yyy,
ITEM IN $[z_1, z_2, \ldots, z_\ell]$

The BY ITEM clause specifies a control break, indicating to the GPPS that it should perform an EVAL(ORDER) every time the value of ITEM changes (in the flat file resulting from the QRS statement). In the conditional clause, the ITEM IN$[z_1, z_2, \ldots, z_\ell]$ indicates that the EVAL is to be performed ℓ times, once for each of the items $z_1, z_2, \ldots, z_\ell$.

CONCLUSION

This papaer discussed the need for an Expert Knowledge Language which captures expert's knowledge about model usage in a form that can easily be converted into Horn clauses. The EKL syntax was not described and remains a topic for future research. The notion of an interactive DSS development system was introduced. A DSSDS enables a developer to effectively identify resolution paths that are of DSSDS enables a developer to effectively identify resolution paths that are of interest. This is comparable to selecting a path through a logical data base structure prior to a retrieval query.

A new type of command (EVAL) was introduced as a necessity for extending relational algebra so that it could, in theory, provide model manipulation as well as the usual data manipulation capabilities. It was shown how a comparable EVAL command could be used in an extended-network context. A DSS developer could use EVAL together with conventional data manipulation commands to specify a procedure for responding to a problem statement of a DSS user. However, this is not in the spirit of a DSS based on a generalized PPS. The paper concluded by outlining the operation of a GPPS that uses EVAL and data manipulation commands to respond to a DSS user's problem statement. That is, the GPPS, in conjunction with the DSS developer, devises the procedural form of a problem solution. With this type of GPPS, the developer has only to identify pertinent resolution paths during DSS design (or revision).

While this paper does show a reasonable approach to GPPS implementation, there remain a number of research issues. These include the precise constitution of the DSSDS from a developer's viewpoint, a general method for deriving model sets from Horn clauses, the internal representation of Horn clauses and resolution paths, and the flexibility of query parameterization associated with a model set.

REFERENCES

[1] Bonczek, R. H., Holsapple, C. W., and Whinston, A. B., Foundations of Decision Support Systems (Academic Press, 1981).

[2] _____, Development Tools for Decision Support Systems, Computerworld, Vol. XV, No. 37 (September 14, 1981).

[3] _____, The Evolution from MIS to DSS: Extension of Data Management to Model Management (May 1981).

[4] MDBS Inc., Lafayette, IN, MDBS Data Base Design Reference Manual (1981).

[5] Bonczek, R. H., Holsapple, C. W., and Whinston, A. B., The Integration of Data Base Management, Information Systems, Vol. 4, No. 2 (1979).

[6] MDBS Inc., Lafayette, IN, MDBS Query Retrieval Reference Manual (1981).

[7] _____, MDBS Application Programming Reference Manual (1981).

The authors are indebted to the National Science Foundation for a grant, Proposal No. IST-8108519, and also the Army Research Office, Contract No. DA79C0154.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

DECISION SUPPORT SYSTEMS, PROBLEM PROCESSING AND CO-ORDINATION

A. Bosman

Information Systems Research Group
University of Groningen
P.O.Box 800, 9700 AV Groningen
The Netherlands

ABSTRACT

There is a lot of discussion about the status and the future of decision support systems (DSS). In this discussion two points relevant for the development of DSS do not get the attention they deserve. The first one is the fact that in various sciences studying the topic of decision processes in organizations a hard core for research is missing. It is questionable whether DSS, even in the form of a theory, can deliver such a hard core. The second point is that too much emphasis is placed on the possibilities to generate action alternatives. This can result in a situation where it is difficult to find a solution for the co-ordination problem with the usually applied instruments. Therefore the two aspects of decision making in organizations, namely the generation of action alternatives and finding a solution of the co-ordination problem, should be regarded simultaneously.

1. INTRODUCTION

Two points relevant for the development of decision support systems (DSS) do not get the attention they deserve. These two points are:
- the lack of a hard core for research on decision processes in organizations;
- too much emphasis on the generation of action alternatives, neglecting the solution of the co-ordination problem.

We discuss the first point in sections 3 and 4. DSS are supposed to assist decision makers in solving decision problems, especially ill-structured decision problems. In section 3 we conclude that it is not possible to define a theory of DSS without considering the methodological problems of defining theories of organizational decision processes. DSS "theories" should not be regarded as a substitute for theories of organizational decision processes. In section 4 a concept for the construction of a DSS that can describe and solve ill-structured problems is presented, using the concept of Bonczek a.o. [1981] of a problem processing system.

The second point is treated in sections 5 and 6. In section 5 it is shown that the usually applied instruments for the solution of the co-ordination problem rest on two assumptions, viz. nearly decomposability of the organization as a system and, although generally implicitly, the paradigm of unbounded rationality. The paradigm of bounded rationality and the feature of nearly decomposability can lead to contradicting results as far as a solution of the co-ordination problem is concerned. In section 6 we try to formulate a solution of this problem.

2. DEFINITIONS

In this section a number of definitions are formulated. A system is a set of elements and of relations between these elements. If the elements are concrete, we name the system a concrete system, if the elements are abstract, the system

is an abstract system. Abstract systems are models of an observed reality. This reality is referred to as an object system. In the process of decision making a decision maker has to choose from a set of alternatives regarding possible ways to allocate means; these alternatives are named action alternatives, see Ansoff and Hayes [1974]. We define, see Bosman and Sol [1982], a problem as well-structured if the following conditions are met:

1) The set of action alternatives is finite and identifiable.
2) The solution is consistently derived from a model that shows a good correspondence.
3) The effectiveness or the efficiency of the action alternatives can be numerically evaluated.

Problems that do no fulfill these requirements are defined as ill-structured. A problem is named well-defined if the set of action alternatives is identifiable. In principle such a problem can be programmed.

3. DECISION SUPPORT SYSTEMS AS A GENERATIVE MECHANISM

The word system in DSS is object system oriented, see figure 1. In this figure the parts of the system are grouped into three subsystems, viz. a method bank, a data base and a model bank. The computer, its operating system, data base management systems and programs in the method bank are the core for the development of DSS. The purpose of DSS, the support of ill-structured management decision making processes, is only implicitly taken into account. In the last few years one can detect in the literature on DSS various attempts to construct a DSS theory taken explicitly into account the purpose of DSS, see Sprague [1980], Keen [1981] and Bonczek a.o. [1981]. We think that the results of these attempts are not satisfactory. There are several reasons for drawing this conclusion. The main one is a lack of a hard core [Lakatos, 1978] in research on decision processes. These processes are studied by various sciences, as economics, management sciences, operations research, organization theories, information systems, social psychology and informatics. These sciences have different subject matters and different methods of research. The whole of organizational studies, including management information systems (MIS) and DSS, are confronted with this lack of a hard core. We assume that until a hard core is defined, it will be impossible to answer the question whether a DSS theory can be constructed.

Several attempts were and are made to define a hard core for organizational studies. We refer to the construction of different paradigms, especially the differences between unbounded and bounded rationality [Simon, 1965], the use of different metaphors [Weick,1979], inquiring systems [Churchman, 1971], metatheories [Mitroff a.o., 1974] and methods of research, as systems theory, econometrics, statistics and operations research. The quest for a generally accepted concept, or a hard core, is not met by these various approaches.
We think it is impossible to meet this quest, because reality is not uniquely defined. Pondy and Mitroff [1979, p. 33] distinguish four types of reality for organizational research:

1 Empirical reality or the world of actual behaviour.
2 Explicit and formal assertions about the rules that govern behaviour (e.g. standard operating procedures).
3 Myths, rituals, metaphors, and other statements, explicit or implicit, about organizational values and processes.
4 Generative mechanisms of a fundamental sort that produce behaviour of all kinds.

With a generative mechanism it should be possible as Pondy and Mitroff [1979, p. 29] remark, to regard an organization as: "not just groups of people; they are sets of organizing rules. And "explaining" organizations is not merely establishing empirical regularities across a set of organizations, it is discovering those deeper organizing rules in each case, and only then comparing across organizations". These four realities can, from a methodological point of view,

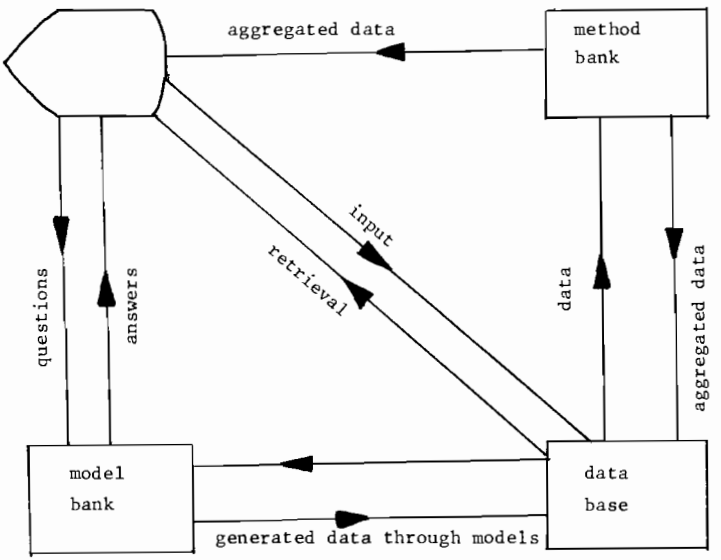


Figure 1 A first generation DSS.

be related in different ways as Bosman and Sol [1982] have shown. We prefer the sequence depicted in figure 2.

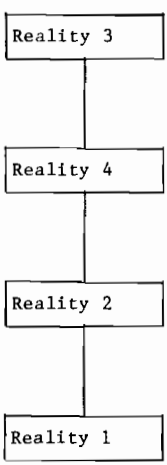


Figure 2 Different images of reality.

A discussion on a DSS theory should start with reality 4, or the construction of a generative mechanism. A survey of literature reveals that most publications refer to differences between well- and ill-structured problems, between unprogrammed and programmed decision processes and between strategic, administrative and operational levels of decision making. These distinctions do not produce a generative mechanism, they are dealing with reality 1 or 2. An interesting and rather unique approach that opens possibilities to formulate a generative mechanism for a DSS theory is given by Bonczek, Holsapple and Whinston [Bonczek a.o., 1981].

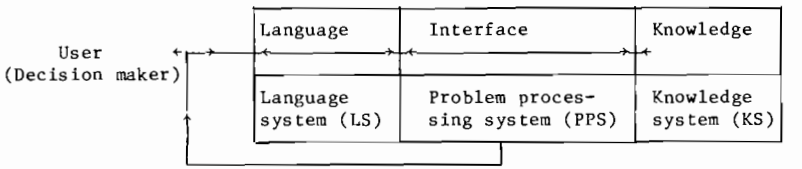


Figure 3 Functions of DSS.

The three principal components for a generic description of a DSS are according to Bonczek a.o. [1981, p.69] depicted in figure 3. The authors [p. 70] circumscribe a LS "as the total of all linguistic facilities made available to the decision maker by a decision support system. A language system may encompass either or both retrieval languages and computational languages. A language system is not concerned with the interfacing of models and data."
The KS is specified [p. 70, 71] as: "This knowledge system typically includes large volumes of facts that the decision maker has neither the time, nor the inclination, nor the opportunity to absorb into his own memory. However, certain subsets of this volume of facts are important to a reasonable or good decision in the face of a particular problem from the problem domain."
A PPS is described [p. 71] as: "The main function of a decision support system is to take strings of symbols organized according to LS syntax (problems) and to take strings of symbols organized according to KS representation rules (problems domain knowledge) and to produce information that supports (enhances or make possible) a decision process. To do so there must be an interfacing mechanism between expressions of knowledge in the KS and expressions of problems in the LS. We refer to this interfacing mechanism as the problem processor or the problem-processing systems (PPS)."

A comparison between figure 1 and figure 3 learns that figure 1 depicts some of the instruments implementing the function of DSS, as sketched in figure 3. The KS consists of the data base and parts of the method- and model bank. Other parts of these banks can be incorporated in the PPS. Not defined in figure 1 is the PPS and the LS. The last one is available, the first one is not specified and may be not even recognized as a function. This is in accordance with the different kind of realities that can be distinguished looking at organizational decision processes.

4. PROBLEM PROCESSING SYSTEMS AND GENERATIVE MECHANISM

Bonczek a.o. [1981, chapter 5] present three different approaches dealing with the contents of a PPS. Emphasis in these approaches is on the generic character of a DSS. A PPS should not be domain oriented. Instead, the main instruments for the construction of a PPS are derived from data base management and artificial intelligence. It is our opinion that PPS's constructed with these instruments do not deliver a generative mechanism in the sense of Pondy and Mitroff.

The main reason for this conclusion is the fact that only parts of the process of decision making can be handled by the PPS's just described. Emphasis lies on problem solution, not on problem finding.
Decision processes can be described in various ways. We use an approach related to a schema adopted from Bonczek a.o. [1981, p. 39], see figure 4. Figure 3 and 4 are similar in construction. The interface function of a PPS is given a content in figure 4. The content of column b is the one of the well known modelcycle in which three different parts can be distinguished. The first part is the one of problem recognition, resulting in an empirical model formulation, the stages 1, 2, 3 and 4. The second part is the one of finding a solution, the stages 5, 6, 7 and 8. The last part is the one of finding the solution, the stage of design, and the implementation of this design.

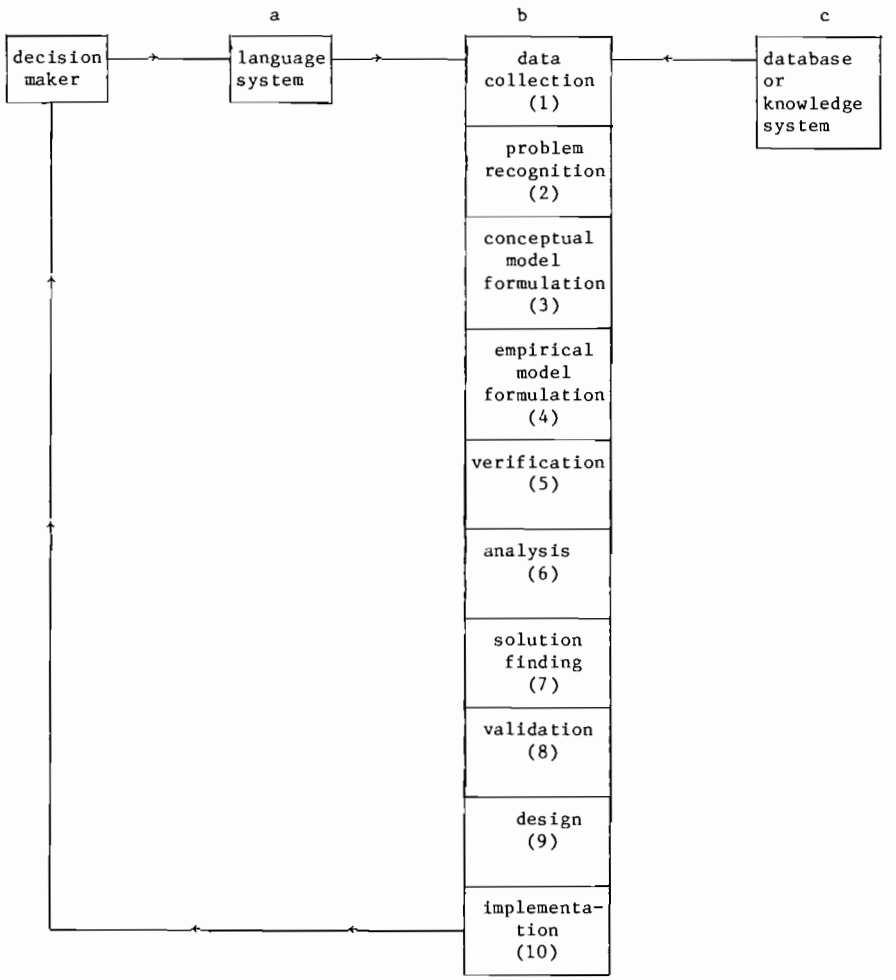


Figure 4 A specification of a problem processing procedure.

Depending on the type of problem and the type of modelcycle [Bosman, 1977] various stages in figure 4 can be omitted. In the case of a well-structured problem one could omit the stages 2, 8 and 9. In the case of a normative axiomatic approach, rather common in organization literature, one can forget stages 1, 2, 4, 5, 6, 8 and 10. In the approach of Bonczek a.o., special attention is paid to stage 1, 4 and 7.

Regarding the purpose of DSS two features are relevant and necessary from a point of view of decision analysis and methodology.

1 Problems are ill-structured, meaning that a set of action alternatives must be generated and some kind of analysis of these action alternatives must be conducted. An ill-structured problem should be transformed into a well defined one. To realize this, the stages 2, 3, 4, 6, 7 and 9 in figure 4 have to be specified. These are also the stages generally used in programs of operations research. The PPS's discussed by Bonczek a.o. could probably also deal with these stages with an extension to stage 1.

2 Problems are formulated using data. Special emphasis is on the combination and interrelation of the stages 1, 2 and 4 with an extension to stages 5 and 7. Methods of systems analysis and system identification, e.g. econometrics, can be used in this field.

Techniques of econometrics and operations research are generally available in method banks. However, this availability is not regarded as a sufficient condition for the construction of a PPS. What is necessary is the possibility to integrate methods of analysis, as mentioned under point 1, with those of problem recognition. Problem analysis is a combination of problem recognition, problem formulation and solution finding. For a non data-void analysis of a problem, as DSS suggest to do, these three phases cannot be formulated independently. For a solution corresponding to the three stages of problem analysis just mentioned, see figure 4.

We endorse the idea of a PPS as depicted in figure 3. Such a PPS should have facilities to solve problems, especially organizational problems, as sketched in column b of figure 4. Emphasis should be on facilities to describe and solve problems. In our research group we, therefore, developed a PPS in the form of a set of techniques to recognize, formulate and analyse problems [Bosman and Sol 1981, 1982]. This set has a number of special features [Sol, 1982].

a Different stages of problem formulation are recognized explicitly, the main difference being a conceptual and empirical formulation. The empirical formulation can be conducted at different levels of aggregation, see section 6.

b The different models mentioned under point a are interrelated. The empirical models are an extension of the conceptual one and in some cases extensions of each other.

c Simulation is used as the main technique for analysis, see section 6. Econometrics is used to aggregate the outcomes and to find a solution.

d Facilities for storing, retrieving and aggregation of output data of simulation runs are available. The program package, called an inquiry system, is user friendly.

It is rather remarkable that the inquiry system approach of a PPS is uncommon in literature. In our opinion it has a number of advantages.

1 It does not hinder, as most available PPS do, the decision maker with a number of restrictions while formulating his problems. Thus, from a cognitive or personal point of view [Mitroff, [1974], Kilmann, Pondy and Slevin [1976]] different decision makers can use the same instruments. In this way different aspects of decision making can be implemented, contributing to multi-disciplinary research.

2 It directs the problem solver through a number of stages necessary to formulate a problem. This can lead to a learning process. From an organizational point of view, see also section 5, it can result in a kind of standardiza-

tion of decision processes, especially when also the data base is standardized. The decision processes can then be compared better and differences can be traced to the way these processes are started and conducted.

3 Research in the sphere of artificial intelligence, data base management and operations research can be directed at different stages in figure 4. It is then perhaps possible to integrate these different results of research for organizational research purposes.

Can an inquiry system be regarded as a generative mechanism? It is our opinion that this question must be answered in a negative sense. If it were a generative mechanisms, it should loose a part of its generality, excluding other generative mechanism. The inquiry system as sketched should, therefore, have facilities to incorporate any paradigm of organizational behaviour. Stage 3 in figure 4 should make this possible. Pondy and Mitroff [1979, p. 18] stress the same point as they remark: "The performance programs proposed by March and Simon (1958) are such "generative mechanism" that produce organizational behavior, and the task of inquiry is to infer the nature of those programs. It is curious that organization theory should have drawn so heavily from parts of March and Simon, but largely missed this very central point of the book". This is the result of two facts.

1 The wrong hypothesis that the paradigm of bounded rationality should result in one theory, see Bosman [1977].

2 The lack of research tools to describe different theories of bounded rationality. Although we have various metaphors [Weick, 1979] related to bounded rationality, different theories specifying these metaphors are not available. An inquiry system as proposed could be an instrument to define these theories.

Generative mechanisms for theory construction and a PPS for DSS are different subjects. A PPS should help a decision maker to specify sets of action alternatives, it should help a researcher to formulate theories. If these demands on a PPS should be met, it must be subject matter free and it should therefore have descriptive and analytic features, as the proposed inquiry system.

5. PROBLEM PROCESSING SYSTEMS AND THE CO-ORDINATION PROBLEM

A PPS in most of the literature on DSS, is decision maker oriented. Its main purpose is to support the decision maker in recognizing his problem so that he is able to formulate action alternatives for the solution of ill-structured decision problems. In organizations the problem of decision making has two aspects. The first one is that relevant action alternatives should be considered. The second one is that decision making in an organization is an interrelated process. The final outcome, profit or least costs, depends on the way the various and numerous decision processes are interrelated. If one regards the organization as an entity, these aspects are the two sides of the same medal. In organizational studie these two sides are generally regarded as independent or nearly independent entities, see e.g. the distinction between macro and micro organizational research. This distinction can also be detected in DSS literature. Nearly all of the literature is directed at the aspect of generating action alternatives. An exception is a publication of Keen and Hackathorn [Keen, 1981]. They distinguish:

- "Personal Support Systems (PSS), for use by individuals in tasks that do not involve interdependencies, so that the user can indeed make a decision.
- Group Support Systems (GSS), for tasks with "pooled" interdependencies which then require substantial face-to-face discussion and communication; and
- Organizational Support Systems (OSS), for tasks that involve "sequential" interdependencies.

A PSS may thus support a manager's own budget decision; a GSS the budget negotiation; and an OSS, the organizational budget process".

The distinction made by Keen and Hackathorn is rather remarkable, for two reasons.

1. The distinction in three different processes explicitly recognizes differences between degrees of interdependence, it even assumes independence. It is, therefore, in that case not possible to find a solution for the co-ordination problem without taken explicitly into account the interdependence between decision processes. Keen and Hackathorn, therefore, correctly distinguish different decision support systems, the differences depending on the nature of the dependencies between decision processes.
2. In economics we are dealing with the problem of allocating scarce resources. In financial terms these resources are in the same pool and not in different ones on a basis of different functions in an organization.

The interdependence between decisions can be strong or weak, they cannot be non existent. Keen and Hackathorn implicitly refer to a mechanism in organization literature that is named the mechanism of nearly decomposable systems [Simon, 1969] or loosely coupled systems [Weick, 1979]. This mechanism assumes that the co-ordination problem can be solved best by regarding the organization as a set of nearly independent parts or subsystems. These subsystems are created by a division in two directions. A horizontal division in levels of the organization, as strategic, administrative and operational, a vertical division in functions, as marketing, production, personnel and finance, see figure 5. Nearly decomposability and loosely coupling both assume that there are a limited number of interrelations. One can ask what kind of interrelations these are? From literature (e.g. Mesarovic a.o. [1970], Dirickx and Jennergren [1979], Ritzman a.o. [1979], Plötzeneder [1976]) it becomes clear that one type of interrelation is considered. It is the relation connecting the different horizontal levels through aggregation, where at the strategic (top) level the co-ordination problem is solved. The generative mechanism behind this construct is the one of unbounded rationality, assuming one decision maker and perfect knowledge. The co-ordination problem can be specified by a set of simultaneous equations, the well known Walras equilibrium system in general economics, see e.g. Naylor and Vernon [1969]. In organizational literature this construct is known as the holistic approach. If this approach should be valid, DSS only have to solve well-defined problems, meaning the translation of the solution of the co-ordination problem at the top of the organization to the lower levels of management in that organization. Research on bounded rationality, however, shows that there is an interrelation between levels of decision making upwards and downwards. These interrelations depend heavily on aggregation of data. This means that lower levels in the organizational structure deliver information for the formulation of the co-ordination problem. This information must, by definition, depend on the way decision making problems at the lower levels are solved. A solution for the co-ordination problem can then only be found if the number of action alternatives generated is small. Then co-ordination is reached by a number of organizational instruments, such as standardization, standard procedures, a certain style of decision making, personal influence of management through a number of informal procedures, a not well-defined structured set of goals and rules of thumbs as decision procedures, see e.g. Kieser and Kubicek [1975, 1977]. All these instruments work "best" in nearly decomposable systems. It is incorrect to assume that nearly decomposability, in the sense sketched in figure 5, should be a general characteristic of organizational system design, as generally stated. Nearly decomposability is the result of applying a certain paradigm, and a resulting procedure to solve the co-ordination problem. It does not guarantee that the best solution for the co-ordination problem is found.

If our hypothesis on the structure of organizational systems is correct, it is possible that the introduction of DSS is going to deliver additional problems for the solution of the co-ordination problem. Because if more action alternatives at different levels of the organization are generated it is due to become

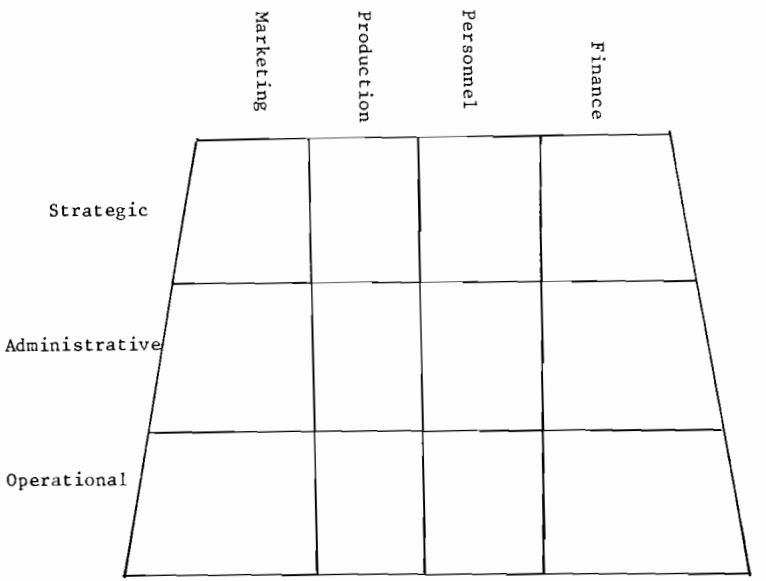


Figure 5: The structure in organizations

harder to solve the co-ordination problem with "normal" instruments. It is, therefore, possible that gains resulting from the introduction of a DSS are only illusions, because they produce losses in the field of co-ordination. This problem is comparable to the one of sub-optimalization in operations research. If the hypothesis is correct, a PPS of DSS can from an organizational point of view only be useful if one takes into account both sides of the allocation medal, viz. the generation of action alternatives and the solution of the co-ordination problem. We doubt, see the next section, whether in such an approach the golden rule of design, viz. nearly decomposable systems, can be maintained.

As standard solutions for the co-ordination problem are not available, a solution can only be found if a generative mechanism is explicitly formulated. In the next section we discuss some properties of problem precessing systems for the solution of the co-ordination problem.

6. CO-ORDINATION AND DISAGGREGATION

In literature two approaches for the solution of the co-ordination problem can be distinguished.

1. The planning approach, with two variants.
   - The monolithic approach, where the co-ordination problem and the generation of action alternatives are formulated simultaneously, see Graves [1982]. If the generation of action alternatives is regarded as a well-structured or a well-defined problem, the main difficulty with this approach is the size of the problem specification. Mixed-integer linear programming techniques can be used to find a solution. As stated in section 5, the generation of

action alternatives is an ill-structured problem that can only be solved correctly if one takes the interdependencies between decision processes into account when formulating the problem.

- The multilevel approach. The main features of this approach are according to Dirickx and Jennergren [1979, p. 2]: "In the multilevel methods of modeling and solving a decision problem, a complete problem representation is put together from subproblems, where each subproblem refers to some part of the whole problem situation. The subproblems are to some extent independent of one another, but not totally. There are certain ties between them. That is, the total model complex is constructed out of a set of building blocks, each subproblem constitutioning one such block". Decomposition methods can be used to specify the interrelations between the different blocks. A well-known structure is the one of a hierarchy used by Mesarovic a.o. [1970] and by Hax and Meal [1975] in the case of production planning. The multilevel approach offers facilities to define the co-ordination problem as a problem consisting of parts and to find a solution. It does not pay much attention to the way the subproblems should be formulated, taken into account the possibility of different connections between interdependent parts. As stated in section 5 aggregation is generally used to specify interdependencies.

2. The second approach can be named the data approach. Here, also two variants can be distinguished. The first one is the data base approach. We regard this approach as not relevant for the solution of the co-ordination problem. The second one can be named the corporate model approach. It uses econometric techniques to reduce the loss of information while aggregating. In some corporate models this approach is chosen. For instance Rozenkrans and Pellegrini [1976] define a corporate model as a : "description of a complete firm and its development or activity in time and at different locations. This description is achieved by the specification, estimation, and solution of equations to reproduce and predict the behaviour of a firm in the areas of finance, production, and marketing, assuming different environments and courses of action". The main features of this approach are the formulation of one integral model in which the interdependencies are, implicitly, represented in the estimated parameters of the variables. Recent research, see Reuyl [1982] and Sol [1982], has shown that aggregation in econometric models creates independence between the different parts of an organizational structure. Models of different levels of the organization describe problems that cannot be related explicitly. What remains is the possibility for co-ordination at the same level. This is a relevant feature, but it is not what aggregation is supposed to do, viz. the coupling of problems at different levels of the organization. Besides, econometric model construction has other disadvantages, see Bosman [1980].
   1. It is status quo oriented. It can only describe what has happened, related to the action alternatives that are chosen. It does not give a description of the set of action alternatives that were considered, but not chosen, nor about the arguments for not choosing these alternatives. Therefore, it is hardly possible to use econometric models for the generation of action alternatives.
   2. Econometric models define problems describing average actual behaviour. Average actual behaviour is used to describe what should be done. This is only possible on a short term basis assuming that average actual behaviour gives an acceptable description. Econometric models, therefore, can in our opinion not be used at a strategic level of decision making, as is sometimes suggested, see Naylor [1979]. Applications are possible, as shown by Dikkers [1982], at an administrative level of the organization, especially with emphasis on the numerical evaluation of action alternatives.

With the exception of multilevel methods the mentioned approaches try to find a solution for the co-ordination problem through a simultaneous specification. When one uses bounded rationality as a paradigm for describing decision proces-

ses in organizations the generation of action alternatives and the solution of the co-ordination problem must be distinguished as two different, though related, processes. The main question is, how this relationship should be specified?

In the union of DSS and organization literature different approaches to find a solution for the co-ordination problem can be distinghuished. We think that the difference between these approaches are of importance for a correct interpretation of DSS and the possibilities to apply DSS for a solution of the co-ordination problem. In figure 3 we depicted three functions of DSS. For our purpose relevant is the relation between PPS and KS. This relation is of vital importance for the problem of applying computers for the solution of ill-structured problems. To illustrate this importance we quote Michon [1982]: "Listen, for instance, to someone speaking an unknown language, or look at a text in Linear-A-Minoan script: you can hear, or see, that there must be something to it but you can't tell what it is. You lack an adequate representation, because you lack the appropriate access structure. The briefest possible summary of this point of view was recently formulated by Newell [1982], who presented the following compact functional equation:

Representation = Knowledge + Access.

Perhaps you will agree with me to the following anology. Libraries ar bodies of uninterpreted knowledge, the (physical) embodiments of the objects in a Popperesque "Third World". Only if there is an access structure - such as is normally described in an instruction manual, a grammar or a book of recipes - that knowledge can be given a meaning. This is what we call a representation"

Acces and PPS have the same pupose resulting in a representation, a model or a design specification. There are, however, different PPS's possible and therefore different representations. We discuss some of these PPS's using figure 4 as a yardstick.

a. The techniques mentioned at the beginning of this section are related with stages 6 and 7 of figure 4. The PPS delivers algorithmic properties to find a solution in a well-defined knowledge system. The algorithmic properties assume a definition of action alternatives according to rather restrict specifications. For well-structured problems such a PPS is feasible. For ill-structurend problems assuming possibilities to generate action alternatives such a PPS is infeasible.
b. Action alternatives as specified in the stages 1 and 2 of figure 4. In a number of group decision support systems (GDSS) facilities are available to assist the process of action alternatives generation, see Wagner [1982]. These facilities are in the field of data gathering, data presentation and data processing. Data processing can be done with help of methods as nominal group techniques and the Delphi technique, see Huber [1982]. Emphasis in this case is on the construction of a well-defined knowledge system. A PPS, in the sense of Bonczek a.o., is not defined, there is not a relation between the stages 1 and 2 with the stages 3 and 4 in figure 4.
c. Three different PPS's will be discussed, each one of them using another relation between on the one hand the stages 1 and 2 and on the other hand the stages 3 and 4.

1. Though not defined as a DSS a specification of a relation that can be used as a PPS is given by Mason and Mitroff [1981]. Emphasis is on the generation of action alternatives in a group process. The conceptual model is formulated using a philosophy of pragmatism. Emphasis is on a group dialectic debate. In the empirical model the outcome of the discussion in groups can be ranked using methods as mentioned under (b). This PPS has a weak content. Emphasis is on the construction of a well-defined knowledge system using as a relation between the different stages in figure 4 a specification of a conceptual model. However, this specification is directed at the formation of groups for the purpose of generating action alternatives. An empirical model is

lacking and it is, therefore, not possible to define the other stages in figure 4.

2. It is possible to integrate stages 3 and 4 in figure 4 in one model. Such an integration is named an expert system. An expert system is defined, following Feigenbaum [1980,p.1], as: "A program that achieves a high level of performance on problems that are difficult enough to require significant human expertise for their solution. The more common strategy of AI research is to choose a highly simplified problem - sometimes called a "toy problem" -and exploit the toy problem in depth. In contrast, the problem we choose require the expertise of an M.D., or a Ph.D., or at least a very highly trained specialist in a field, to solve. An expert system of this type consists of only two things: a knowledge base, and an inference procedure. The knowledge base contains the facts and heuristics; the inference procedure consists of the processes that work over the knowledge base to infer solutions to problems, to do analyses, to form hypothesis, etc."
   Emphasis in the case is not on the generation of action alternatives but on the possibility to transform an ill-structured problem in a well-structured one using expert knowledge. In a certain sense this approach is comparable with the one of an econometric corporate model. One can calculate the consequences of different scenarios. The result of the integration of the stages 3 and 4 is that the other stages in figure 4 are no longer relevant. This approach is from a methodological point of view efficient. Lacking, however, is any guarantee that the relevant action alternatives are generated, nor that the expert knowledge is outdated.
3. From a methodological point of view the PPS we formulated in figure 4 can be regarded as an expert system. This PPS could be used to integrate some of the positive points of the two other PPS's we discussed to find a solution for the co-ordination problem. These positive points are:
   - The generation of action alternatives makes it possible to use disaggregation to formulate action alternatives.
   - Expert systems deliver a more efficient way than artificial intelligence procedures to reduce search processes.

   We suggest to start with stage 3 of figure 4 and define it as an expert system for the solution of the co-ordination problem in the organization concerned. It schould be followed by the stages 1 and 2 to generate and to evaluate action alternatives. For the evaluation an inquiry system, see Sol [1982], and simulation can be used to specify an empirical model. In some cases the methods discussed at the beginning of this section can be used in the stages 6 and 7. The other stages can be maintained as discussed.
   The concept of an expert system is not frequently used in DSS literature. We assume that the main reason for this fact is that in this literature emphasis is on defining instruments to assist individual decision makers. Emphasis is not on instrument to replace them, as the content of an expert system suggests to do. As already stated, decision making processes in organizations result in a co-ordination problem, that can only be solved taking into account the interactions between decision makers. To specify these interactions PPS's of a different nature than the ones for individual decision makers are necessary. We believe that for the specification of these PPS's the basic ideas in the construction of an expert system can play an central role. These basic ideas can be considered as a generative mechanism, see section 3, and as a specification of a conceptual model. Further research how this can be done in relation with bounded rationality and the co-ordination problem is necessary.

Literature

H.I. Ansoff and R.L. Hayes (1974), "Roles of models in corporate decision making", Systems and Management Annual 1974, ed. R.L. Ackoff, New York, Petrocelli.

R.H. Bonczek, C.H. Holsapple and A.B. Whinston (1981), Foundations of Decision Support Systems, New York, Academic Press.

A. Bosman (1977), Een metatheorie over het gedrag van organisaties, Leiden, Stenfert Kroese.

A. Bosman (1980), "Long-term forecasting - a paradox?" Prospects of Economic Growth, eds. S.K. Kuipers and G.J. Lanjouw, Amsterdam, North-Holland.

A. Bosman and H.G. Sol (1981), "A simulation based inquiry system for the development of information systems", The Information Systems Environment, eds. H.C. Lucas et a., Amsterdam, North-Holland.

A. Bosman and H.G. Sol (1982), "Evolutionary development of informations systems", Evolutionary Information Systems ed. J. Hawgood, Amsterdam, North-Holland.

C.W. Churchman (1971), The Design of Inquiring Systems, New York, Basic Books.

G.J.O.D. Dikkers (1982), Development and Introduction of a Decision Support System, A Case Study of an Econometric Corporate Financial Model, Groningen, doctoral thesis.

Y.M.I. Dirickx and L.P. Jennergren (1979), System Analysis by Multilevel Methods: With Applications to Economics and Management, Chichester, John Wiley.

E.A. Feigenbaum (1980), "Expert systems: looking back and looking ahead", GI-10 Jahrestagung, ed. R. von Wilhelm, Berlin, Springer-Verlag.

S.C. Graves (1982), "Using lagrangean techniques to solve hierarchical production planning problems", Management Science, vol. 28, nr. 3.

A.C. Hax and H.C. Meal (1975), "Hierarchical integration of production planning and scheduling", Logistics, ed. M.A. Geisler, Amsterdam, North-Holland.

G.P. Huber (1982),"Group decision support systems as aids in the use of structured group management techniques", DSS-82 Transactions, ed. G.W. Dickson, San Francisco.

P.G.W. Keen (1981), "Information systems and organizational change", Communications of the ACM, vol. 24, nr. 1.

A. Kieser and H. Kubicek (1975), "An organizational concept for the design of management information systems", Information Systems and Organizational Structure, eds. E. Grochla and N. Szyperski, Berlin, De Gruyter.

A. Kieser and H. Kubicek (1977), Organisation, Berlin, De Gruyter.

R.H. Kilmann, L.R. Pondy and D.P. Slevin (1976), The Management of Organization Design, two vol., New York, North-Holland.

I. Lakatos (1978), The Methodology of Scientific Research Programmes, Cambridge, Cambridge University Press.

R.O. Mason and I.I. Mitroff (1981), Challenging Strategic Planning Assumptions, New York, John Wiley.

M.D. Mesarovic, D. Macko and Y. Takahara (1970), Theory of Hierarchical, Multilevel, Systems, New York, Academic Press.

J.A. Michon (1982), "How to connect a library with a mind",

I.I. Mitroff (1974), The Subjective Side of Science, Amsterdam, Elsevier.

I.I. Mitroff et al.(1974), "On managing science in the system age: Two schemes for the study of science as a whole systems phenomenon", TIMS Interfaces, vol. 4, nr. 3.

Th. H. Naylor and J.M. Vernon (1969), Microeconomics and Decision Models of the Firm, New York, Harcourt, Brace and World.

Th. H. Naylor (1979), Corporate Planning Models, Reading, Addison - Wesley.

A. Newell (1982), "The knowledge level", Artificial Intelligence, vol.18, pp. 87-127.

H.D. Plötzeneder (1976), Computer Assisted Corporate Planning, Chicago, Science Research Associates.

L.R. Pondy and I.I. Mitroff (1979), "Beyond open system models of organization", Research in Organizational Behaviour, ed. B.M. Staw, Greenwich, JAI Press.

J.C. Reuyl (1982), On the Determination of Advertising Effectiveness: An Empirical Study of the German Cigarette Market, Leiden, Stenfert Kroese.

L.P. Ritzman et al.(1979), Disaggregation, Problems in Manufacturing and Service Organization, Boston, Martinus Nijhoff.

F. Rosenkranz and S. Pellegrini (1976), "Corporate modelling Methodology and computer-based model design procedure", Angewandte Informatik, vol.6, pp. 259 - 267.

H.A. Simon (1965), Administrative Behaviour, New York, The Free Press.

H.A. Simon (1969), The Sciences of the Artificial, Cambridge, MIT Press.

H.G. Sol (1982), Simulation in Information Systems Development, Groningen, doctoral thesis.

R.H. Sprague, Jr. (1980), A framework for research on decision support systems", Decision Support Systems: Issues and Challenges, eds. G. Fick and R.H. Sprague, Jr., Oxford, Pergamon Press.

G.R. Wagner (1982), "Beyond theory Z with DSS", DSS-82 Transactions, ed. G.W. Dickson, San Francisco.

K.E. Weick (1979), The Social Psychology of Organizing, Reading, Addison-Wesley.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

ORGANIZATIONAL VARIABLES INFLUENCING DSS-IMPLEMENTATION

Leif B. Methlie

Institute for Information Systems Research
Norwegian School of Economics and Business Administration
Bergen, Norway

The general problem dealt with in this paper is how the infrastructure set up by an organization, on the one hand, will give several economic benefits, and, on the other hand, will impose constraints on individuals, and, thus affect socio-political behavior. Both kinds of variables are assumed to influence overall effectiveness of specific DSSs. A proposition set forward and analyzed in the paper, is that the more complex the infrastructure is, the less discretion will the decision-maker exercise in the job. Rational-economic thinking may support sharing of a range of resources in the organization. However, socio-political issues may counteract these expected economic benefits. The ultimate interest is how potential dysfunctional consequences of socio-political aspects impact on effectiveness and efficiency of task performance.

INTRODUCTION

Decision Support Systems deal with ill-structured problem situations or decision processes. Ill-structured situations refer to decision processes that are not known prior to the decision-making and therefore have no predetermined set of responses or outcomes. Several different labels have been used to describe these situations: unstructured [10], [20], semi-structured [13], less well structured [24] and ill-structured [4], [28]. Newell and Simon's experiments with human problem solving show that a person deals with unstructured situations by factoring it into familiar, structurable elements [21]. Mintzberg [20] observed such basic logic or structure underlying managerial decision-making. The structureness of the decision process to be supported by a computer system has impacts on system design and architecture, as well as on system development and implementation. Ill-structured problems are solved by incremental steps [14]. Therefore, system development and implementation must be an open-ended process of design, usage and learning. Due to ever changing problems the decision-maker is engaged in continuous restructuring of data, a process which may include needs for new data inputs, new alternatives to be considered, and altered goals. Therefore, a specific DSS has a short life time. To be able to respond quickly to evolving demands, new capabilities are continuously added. High-level, user-oriented and open-ended design tools are required. The organization may set up facilities such as time sharing systems, databases, model bases and information networks, to facilititate easy and quick building of specific DSSs. A network of computer and communication resources on which the specific DSSs resides is here called the technical infrastructure of the organization.

Another feature of DSS is the kind of users it supports. These are broadly thought of as knowledge workers such as managers and professionals. These people normally exercise high discretion in their jobs, i.e. they exercise control on how and when to perform a specific task, and to some extent whether or not to perform the task. Discretion also means that a decision-maker can choose the tools he or she finds

appropriate in a given situation. Therefore, for a DSS to be used, it must not only fit the information needs of a user, but should also fit into the socio-political behavior of the decision-maker. Discretion is one of several socio-political factors dealt with in this paper. Other factors are influence, control, complexity, filtering etc. They determine the power base of an individual and the power distribution among individuals within an organization. Changes in power may be caused by introduction of new tools to decision makers, e.g. a DSS. Socio-political factors have no absolute values. They are related to perceiptions of the individuals. Therefore, we can not say that high discretion is generally desirable. "Individuals exercise discretion whenever they believe it is to their advantage to do so and seek to evade discretion on other occasions" [26].

The general problem dealt with in this paper is how the infrastructure set up by an organization, on the one hand, will give several economic benefits, but, on the other hand, will impose constraints on individuals, and thus, affect socio-political behavior. Both these two factors are assumed to influence overall effectiveness of specific DSSs. A proposition set forward and examined in this paper is that the more complex the infrastructure is, the less discretion will the decision-maker exercise in the job. Rational-economic thinking may support sharing of a range of resources in the organization. However, socio-political issues may counteract these expected economic benefits. The ultimate interest is how potential dysfuntional consequences of socio-political aspects impact on effectiveness and efficiency of task performance. In section 2 the components of an infrastructure and the supporting facilities of an organization are described, and in section 3 some socio-political variables are introduced. These are followed by a simple framework of organizational support in section 4. Section 5 will relate rational-economic variables and socio-political variables to the levels of support in this framework. Finally, in a concluding section some DSS implementation problems are discussed to avoid counteractions due to dysfunctional consequences of socio-political behavior.

DSS-INFRASTRUCTURES

A specific DSS is often depicted as a system consisting of three components: a database, a modelbase, and a software system comprised of three sets of capabilities: data management, model management and a user-system interface [24]. It is also recognized that the information needs of managers come from a variety of different data sources, internal as well as external to the company. Furthermore, modeling needs may require access to a variety of modeling tools and pre-programmed (canned) models. Model management must be capable of integrating these models or modules into a specific DSS. Quick and easy building of specific DSSs is determined by the facilities available. Therefore, the organization may set up an underlying structure, a technical infrastructure, of resources such as computer processing resources, communication resources and databases. In the following a description of the components of this technical infrastructure is given.

a. The DSS-Generator

Packaging data management, model management and display management into a single tool has become known as a DSS-generator [24]. It provides end-users with high-level, open-ended design tools which shorten the time for DSS implementation considerably as compared to the use of more general programming languages. The software that might be used as generators is evolving from special purpose languages, better known as modeling tools or modeling languages. These languages have strong modeling capabilities and provide users with high-level, non-procedural interactions with the computer. Sprague and Carlson [25] have described desirable properties of DSS-generators. Future DSS-generators are expected to have more enhanced capabilities for data management and display management.

b. Computer Resources

Computer resources may range from independent micros, through time-sharing systems, to interconnected networks of personal work stations. A micro will normally support a limited set of related tasks. They may also be used to access and retrieve data from corporate databases. Time-sharing computers may be set up in-house or by external time-sharing services. Users are connected to these systems by terminals. A personal work station is intended to provided most of a person's information processing needs through a single, multi-functional device [7], [23].

c. Data Management Facilities

Data management can be performed on several levels in an organization. On the lowest level, data are highly personalized with one individual responsible for capturing and entering the data into the system. The data are stored for personal use only. We shall call this personal data management. Personal data management is typical in early stages of DSS implementation where data files are model-specific and usually small.

Some specific DSSs may be designed with enhanced data retrieval capabilities that can take advantage of data already captured and stored on files inside or outside the company. Access may be provided to data files or other application systems, for instance accounting information systems, to corporate databases, and to external on-line databases. Accessing, aggregating and reformatting data to fit individual models use a technique known as data extraction [5], [17-18]. Data entered into a DSS by means of data extraction techniques we shall call external data management.

Data extracted from other sources may be pooled into a single database and shared among end-users. These data are managed by a DSS-database management system and made available to both users and models by an interface language [3]. We shall call this DSS-data base management.

Data stored in databases are normally structured and formatted into fields, records and files. Much of the information a manager needs is not in this form. It is frequently in the form of free text contained in documents. But documents can also be stored in files according to content and searched for by key-words or context-words. Also maps, pictures, diagrams and figures are representations of information that a manager frequently use. We shall use the generic term information management to include storing and retrieval of any kind of information representation. It constitutes the most advanced information management capability in the technical DSS-infrastructure.

d. Model Management Facilities

As for data management we can distinguish various facilities for model management. We shall describe these facilities in the order of increasing interdependence with other components of the organization.

The simplest model management facility provided is a modeling language or a modeling tool by which an end-user can develop a range of separate models. Next, the user may want to use several modeling tools, software programs such as SPSS, IFPS, TROLL etc., to develop classes of models. The organization may provide mechanisms for linking together models written in different languages (model-linkage). The organization may also set up a library of company-specific model-building blocks to be used in multiple DSSs and provide facilities to share models among DSS-users. We shall call this model base management.

e. Information Networks

The infrastructure described above provides primarily processing and storing faci-

lities. However, an important activity in decision-making is communication, and communication facilities should be added to the infrastructure. Communication within buildings can now be provided by special local area networks (eg. Xerox's Ethernet), while long distance communication is provided by several alternative transmission links: telephone networks, special trunk networks, satellites, or computer networks (so-called value-added services [16]). The communication services set up on these networks may range from electronic message systems to teleconferencing systems. These systems offer communication in either synchronous or asynchronous modes; and thus, impose varying degrees of dependency among the users.

SOCIO-POLITICAL VARIABLES OF INTEREST

When the concept of a Decision Support System was introduced it added a behavioral component to the management sciences and MIS approaches. This behavioral component was concerned with cognitive elements or styles of decision making. The purpose was to solve the cognitive dissonance problem between OR/MS-models and real decision-making. However, there is another component of behavior of importance to DSS design and implementation that has not been recognized until recently. This is the socio-political behavior of decision making. Socio-political behavior is concerned with power bases of individuals and power distribution among individuals within an organization. By adding this behavioral component, problems of power dissonance can be explicitly dealt with in implementing DSS. We shall use some power concepts as found in political sciencesand applied to computer systems implementation by Bjørn Andersen and Pedersen [2]. Two concepts are basic; influence and discretion.

Influence refers to the power of changing behavior or outcome of a decision-making process. The means with which this can be done, range from disseminating factual information (being a gatekeeper (see section 5.d)), to giving advices, exercising control and issuing orders. Also the channels used can vary among personal contacts. formal meetings, and formal reports and memos.

Discretion refers to the power to influence one's own task or absence of influence from everybody else. Attributes that will characterize discretion are timing of actions, methods of problem solving, and dependency on other people. Discretion and influence are determined by the power base which consists of resources available to an individual. It has two major components: the formal position of the decision maker in the organization and the knowledge (skill) possessed by the individual. In addition, some personality factors and organizational factors also influence discretion. For instance, centralization and formalization will constrain individual actions in the organization.

A DSS can impact on the power base and power distribution in several ways dependent on the characteristics of the DSS. We shall therefore first look at DSS-characteristics in terms of the infrastructure set up by the organization.

ORGANIZATIONAL SUPPORT LEVELS

In section 2, we have described software and hardware facilities of varying degree of complexity that an organization may provide to facilitate quick and easy building of specific DSSs. The information technology applied by an organization to set up the infrastructure will influence the interdependence among its units [9]. We shall now develop a simple framework, based on interdependence imposed on organizational units, to discuss rational-economic benefits as well as organizational and socio-political impacts. Both factors are assumed to influence system effectiveness, and, therefore, objects of interest in DSS-implementation. Works done by [15] and later [26] are found particularly useful to develop this framework although we shall tamper with their labels.

Our framework consists of four types of infrastructures, here called organizational support levels. These levels are organized into a hierarchy, such that each higher level imposes additional dependency constraints on the users. The lowest level is where each individual decision-maker has full control of the facilities, and the highest level is where decision-makers interact with each other through a DSS. Interdependencies in an organization are resolved by coordination, and each level will have its own mechanism to achieve sufficient coordination. Figure 1 may help to understand the interdependencies created by the infrastructure.

Level 1: Independent Facilities - Here, each specific DSS is built on its own facilities, completely separated from other systems in the organization, (eg. a micro with a software package such as VisiCalc). The user has full control and no coordination of the usage of these facilities among the decision-makers is required.

Level 2: Pooled Facilities - This situation is described by pooled resources on which multiple DSSs may be built. A specific DSS does not contribute to the pooled facilities; it is only supported. Examples of pooled facilities are: time-sharing systems, corporate databases, on-line external databases, and model-building blocks. Each DSS operates independently of each other. Coordination of the usage is achieved by standardization of interfaces between the DSSs and the pooled facilities, for instance, by data definitions and data structures.

Level 3: Shared DSS-Facilities - In this case a specific DSS contributes to the infrastructure, that is, it supports other DSSs. However, there is no interactions between multiple DSSs. A model developed by one user may be available by means of a model base for others. Likewise, data may be entered from one DSS into a DSS-database which can later be interrrogated by other DSSs. This may be the case where a DSS supports sales forecasting and sales budgeting. The sales budget may be input to several other DSSs, for instance financial planning, production planning, cost budgeting, etc. This kind of interdependence is in organizational literature called sequential interdependence [26]. Coordination is usually achieved by plans. Plans may be necessary in a DSS-environment where several DSSs are developed in sequence such that one is an input-element to another. However, as is the case for databases, these very often function as buffers, and coordination by standardization may therefore be sufficient.

Level 4: Interdependent (Reciprocal) Facilities - If decision-makers interact through facilities, as in an electronic message system, or through a DSS designed to support an organizational task involving several actors [11], the outputs of each decision-maker become the input for others. This is sometimes referred to as reciprocal interdependence. Examples are electronic mail, computer conferencing, DSSs for organizational budgeting and market planning. Coordination in this type of environment is usually obtained by mutual adjustment [26]. Here, the difficulty lies in making sure that the learning levels of all users are reasonably synchronized.

The above discussion can be summarized in Table 1.

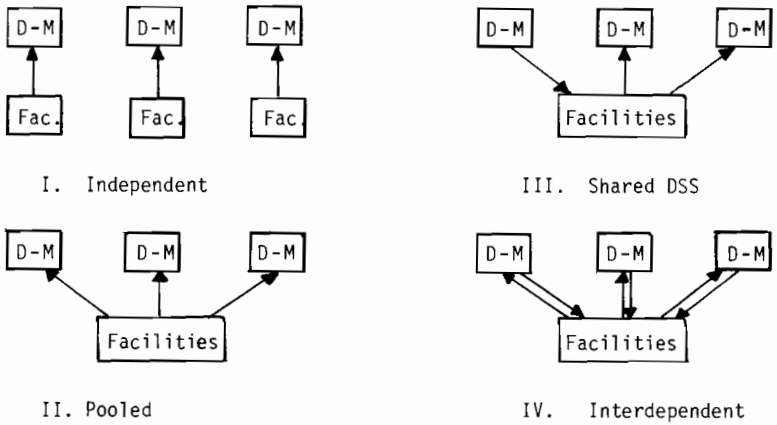


Figure 1
Interdependencies Created by the Infrastructure

Support Levels	Sample Facilities	Coordination Mechanisms	Organizational Impacts
I. Independence	Independent micros	none required	Decreased Discretion ↓ / Increased Coordination ↓
II. Pooled	Time-sharing Data input devices (corporate databases)	Standardization	
III. Shared DSS	Data sharing Model sharing	Standardization/ Plans	
IV. Interdependent (Reciprocal)	Electronic Mail Organiz. Task-DSS	Mutual adjustments	

Table 1
Relationships between Support Levels and Organizational Impacts

DISSONANCE FACTORS IN DSS IMPLEMENTATION

The purpose of this section is to examine potential conflicts between rational-economic thinking and socio-political behavior. Conflict variables are often referred to as dissonance factors in the organizational literature [12]. Dissonance factors may be created by DSS-facilitators, DSS-designers or other information specialists focusing too much on technical aspects of organizational effectiveness or by the decision-makers themselves being very reluctant and resistant to changes in decision-making behavior. In a manager's highly descretionary job environment acceptance is equally important as technical quality of a DSS. Acceptance is to a large extent governed by socio-political aspects. Therefore, rational-economic benefits may sometimes have to be traded with socio-political values. Socio-political values are not absolute, but are related to perceptions by individuals. They may, therefore, be influenced by, for instance, training and learning about the system, and knowledge of the economic benefits of it.

It is necessary to understand how dissonance between rational-economic values and socio-political values may impact on system effectiveness. We shall in this section relate potential conflicts and dissonance to the four levels of organizational support described above. The findings must be interpreted such that dissonance factors specified on one level, have all higher levels' dissonance factors present as well (Guttoman-type scale [26]). Fields of dissonance are summarized in Table 2 at the end of this section.

a. Independent Facilities

The basic assumption behind information systems development is that increased information processing capacity will increase performance at a given level of uncertainty. Thus, by giving a decision-maker enhanced information processing capabilities, his or her effectiveness is assumed to be increased. Many researchers have challenged this direct relationship between information systems and performance. Mintzberg [19] found in his studies of managers' activities that information systems do not play a central role in decision-making. Human information processing in experimental and relies on simplifications [13]. Under pressure, the decision-maker may even discard information, avoid bringing in experts and exploring new alternatives [29]. Enhanced information processing capacity may be used to explore more alternatives or to better and more comprehensively predict consequences of a decision by taking more valiables into consideration. Thus, we may expect that complexity in a decision situation increases as the information processing capabilities increase. Eason [6] in a study of impacts of computer based information systems on managers, reports on an example from a hospital case-study. A doctor used to base his decisions to admit patients almost exclusively on clinical criteria. Access to information about the bed-state of the entire hospital, changed the decision-making complexity by also considering consequences on the hospital staff.

Bringing in micros which are used independently enhances the information processing capacity of individuals. The current DSSs residing on micros are primarily used to substitute processes which are highly analytical and involve much calculations, eg. financial analysis. These systems do not introduce dependencies on other resources in the organization. A DSS is used as a personal tool, and full descretion on task performance is retained. Therefore, no dysfunctional consequences from socio-political aspects are expected in this environment. Increased efficiency as a result of more efficient tools may be converted into increased control and influence on other people, the economic benefits, however, may be fairly limited.

b. Pooled Facilities

The rational for sharing of resources is economy of scale. Data are an example of one such resource. Capturing and filing data are expensive, but using them involves very small costs. Therefore, if the data can be used in multiple situa-

tions, the costs of collecting, storing and maintaining them can be shared. This normally requires a database with access from terminals. In a database approach, users are relieved from the technical tasks of personal data management. Furthermore, data capture at the source enhances the values of several other attributes such as timeliness (data can be accessed at any time), consistency (data redundancy eliminated or reduced), and objectivity (no filtering due to passing of information through multiple subordinate-superior pairs). Furthermore, the amount of information used in a specific situation is not only a function of uncertainty, but on time and availability as well. Under norms of rationality the decision-maker is expected to increase the amount of information used in decision making with access to data from a database.

In conflict with some of the rationales of pooled facilities may be the socio-political values often associated with having possession of resources. Data, for instance, are a resource which is as much political as economic, and whose re-distribution affects interests of particular groups [13]. Pooling facilities must take this into consideration. It may redistribute power relationships both vertically and horizontally. Vertically, it may give a superior access to unfiltered operational data and thus better monitor a subordinate's performance. Pooled facilities, such as corporate databases, may also lead to horizontally more evenly distributed objective information (unless technical restrictions with respect to access are put on the system, eg. passwords). Shared access to planning and operating data among managers of sub-units in an organization "should minimize potentially dysfunctional gaming strategies and tend to equalize power" [1]. Pooled facilities will probably lead to a reduction in dependencies among people on the same level.

c. Shared DSS-Facilities

When a DSS becomes an input component in a greater network not only operational data, but also managerial information, and analytical models and procedures can be shared. A database and a model-base may be established within the DSS-environment to accomodate this [25].

The rationales for sharing resources are much the same as that of pooled facilities. In this case, however, managerial information is shared, and the decision-makers operating in a shared information environment must be willing to do so. Thus, we may expect socio-political factors to play a more dominant role here than in the case of pooled facilities. With pooled facilities we expected that sharing of information would have an equalizing effect. Empirical research, however, has shown that persons dominating the input media may gain increased influence on other peoples work environment. Therefore, a decision-maker controlling the input DSS is in a gatekeeper position [22] between the system and other users of this same information. Bjørn-Andersen and Pedersen [2] found that these gatekeepers increased their influence over other people. Another factor, likely to influence socio-political values, is that entering information into a shared computer environment, probably reduces opportunities for concious distortion, filtering and channeling of information.

The rationales of sharing can only be obtained by standardizations on data elements and model elements. Data have to conform to certain standards and grouped in certain ways to be handled by the specific data management system used. Likewise, models must conform to certain interface standards. These contraints put the user in a more dependent position of DSS-facilitators or information specialists.

d. Interdependent (Reciprocal) Facilities

When facilities are used in a reciprocal mode, exchange of information, as well as processing, can be made by means of the infrastructure. A DSS built on reciprocal facilities will support organizational interdependent decision making, eg. planning or budgeting. Several people interacting with each other, will use the system

to communicate, store and process information. Thus the system can also support communication between groups participating in the decision-making. A special case is electronic message systems, where interaction between groups takes place through a computer network. Faster decision can be obtained by more efficient communication and sharing of information. However, a DSS which supports organizational interdependent tasks where several persons are involved, formalizes the interaction between actors. Rules will have to be established on how the system can be used, and the system implicitly establishes contracts among actors. These contracts impose restrictions on what an actor can do, and when it can be done. Furthermore, use of this system requires a syncronized learning process among all its users. Reciprocal facilities increase organizational control at the expense of individual control and discretion. It also imposes greater interdependency among decision-makers. Examples on increased organizational control and their socio-political effects have been given in a recent issue of Business Week (March 29, 1982). Use of "open" electronic calendars, for instance, has lead to people filling in dummy meetings (to look busy and retain discretion). Electronic message systems allow easy filing and record-keeping of messages passing in the system. Thus, the organization can keep track of all correspondence, and, if necessary, present a correspondence-record in a particular matter. Furthermore, people can not claim that a particular message never was received.

Support Levels	Rational-Economic Variables	Socio-Political Variables
I. Independent	- Reduced uncertainty by increased info.proc. - Increased efficiency	- Increased factual knowledge (i+) - Increased complexity (d-) - Increased span of control (i+)
II. Pooled	- Economy of scale - Better timing - Consistency - Objectivity	- Power shift upwards (d-) - Power equalization horizontally (i-)
III. Shared DSS	- More managerial information available - More analytical aids available	- Gatekeepers controlling info. flows (i+) - Less opportunity for filtering (i-) - More standardization (d-) - Depending on DSS-facilitation (d-)
IV. Interdependent (Reciprocal)	- Faster decisions - More efficient communication	- Timing constraints (d-) - Increased expert power (d-) - Increased formalization (d-)

i = influence, d = discretion, + = gain, - = loss

Table 2
Fields of Dissonance

DSS IMPLEMENTATION

The discussion above, indicates two mechanisms through which overall system effectiveness can be affected. These are 1) the degree to which the system performs according to a rational-economic criteria (technical quality), and 2) the degree to which the system matches the socio-political profile of the user. The latter will to a great extent determine acceptance of a new tool. Low fit in either of the two fitness criteria results in low overall effectiveness. The above emphasizes the importance of using both mechanisms in implementing a DSS.

Complexity in the decision-making environment is found to be a major counteractive factor in technology transfer in organizations [8]. The framework developed above with new complexity constraints added as we move from lower levels to higher levels, may indicate that there is a learning curve, and a growth curve associated with these levels. This may explain the market penetration in the managerial area of personal computers with simple software packages such as VisiCalc (see for instance MIS Weekly of February 17, 1982). It also explains why so many DSSs being implemented as data processing systems on interdependent facilities or a least pooled facilities, have failed in implementation. It seems that the introduction of DSS to managers, at least first time users, is accepted more easily on level 1 with independent facilities than on higher levels.

The framework developed suggests that the socio-political profile of the decision maker must be identified before a DSS is designed and implemented. The appropriate level of implementation must be determined and any "power dissonance" that can be expected from the DSS must be reduced by training and learning.

Several strategies for organizational change taking into consideration behavioral aspects have been presented in the literature [12], [13], [27]. None of these studies have, however, related socio-political variables to the complexity-level of the technical system on which they are based. Socio-political issues as discussed in this paper should be built into a more comprehensive strategy for DSS implementation.

CONCLUSION

The DSS-movement was created to deal with the cognitive dissonance problems that were so apparent in many management science models and in MIS. Recently the problem of power dissonance in computerized information systems have been recognized. This paper deals specifically with power dissonance in DSS. The approach of this paper has been to analyze works in the area of computer impacts on individuals and infer from these findings consequences to be expected in DSS implementation. More specifically, the dissonance between rational-economic values and socio-political values are related to the complexity level of the technical infrastructure on which the DSS is built. This framework can be used to explain and predict socio-political behavior. Furthermore, the levels of complexity may indicate a stage-of-growth hypothesis of DSSs in organizations. The main argument presented in this paper is that the more complex the infrastructure for DSS-implementation is, the less discretion is left for the decision-makers to exercise. Choice of infrastructure and organizational support must be determined not only from rational-economic criteria, but from socio-political factors as well, since dissonance between them will greatly affect overall effectiveness.

REFERENCES

[1] Bariff, M. L. & Galbraith, J. R., Intraorganizational Power Considerations for Designing Information Systems, Acounting, Organizations and Society, Vol. 3, No. 1, Pergamont Press (1978).

[2] Bjørn-Andersen, N. & Pedersen, P.H., Computer Facilitated Changes in the Management Power Structure, Accounting, Organization and Society, Vol. 5, No. 2, Pergamont Press (1980).

[3] Bonezek, R.H., Holsapple, C.W., Whinston, A.B., Future Directions for Developing Decision Support Systems, Decision Sciences, Vol. 11, (1980).

[4] Bosman, A., Decision Support Systems, Problem Processing and Co-ordination, to be presented at IFIP/IIASA conference on Process and Tools for Decision Support, IIASA, Austria, 1982.

[5] Carlson, E.D., Using Large Data Bases for Interactive Problem Solving, Proc. International Conference on Very Large Data Bases, ACM, New York, (1976).

[6] Eason, K.D., Computer Information Systems and Managerial Tasks, in: The Human Side of Information Processing, Ed. Bjørn-Andersen, North-Holland, Amsterdam, 1980.

[7] EDP Analyzer, Get Ready for Managerial Work-Stations, EDP Analyzer, Vol. 18, No. 1, (January 1980).

[8] EDP Analyzer, The Coming Impact of New Technology, EDP Analyzer, Vol. 19, No. 1, (January 1981).

[9] Gailbraith, J.R., Organizational Design, Addison-Wesley, Reading, Mass., 1977.

[10] Gorry, G.A. & Scott Morton, M.S., A Framework for Management Information Systems, Sloan Management Review, Vol. 13, No. 1 (Fall 1971).

[11] Hackathorn, R.D. & Keen, P.G.W., Organizational Strategies for Personal Computing in Decision Support Systems, MIS Quarterly, Vol. 5, No. 3, (September 1981).

[12] Den Hertog, J.F., The Role of Information and Control Systems in the Process of Organizational Renewal, Accounting, Organizations and Society, Vol. 3, No. 1, (1978).

[13] Keen, P.G.W., Information Systems and Organizational Change, Communications of the ACM, Vol. 24, No. 1, (January 1981).

[14] Lindblom, C.E., The Science of Muddling Through, Public Administration Review, Vol. 19, No. 2, (Spring 1959).

[15] March, J.G. & Simon, H.A., Organizations, John Wiley & Sons, Inc., New York, 1958.

[16] Martin, J., Future Developments in Telecommunications, (2nd Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1977.

[17] Methlie, L.B., Data Management for Decision Support Systems, Data Base, Vol. 12, No. 1-2, (Fall 1980).

[18] Methlie, L.B., Data Management Rquirements for Decision Systems, Working Paper, Norwegian School of Economics and Business Adm., (1981).

[19] Mintzberg, H., The Nature of Managerial Work, Harper and Row, New York, 1973.

[20] Mintzberg, H., Raisinghandl, D. & Theoret, A., The Structure of "Unstructured" Decision Processes, Administrative Science Quarterly, Vol. 21, No. 2, (1976).

[21] Newell, A., & Simon, H.A., Human Problem Solving, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[22] Pettigrew, A.M., The Politics of Organizational Decision-Making, Tavistock, Londong, 1973.

[23] Schoichet, S., Personal Work Stations: A Concept Evolves into a Booming Industry, MINI-MICRO SYSTEMS, (April 1981).

[24] Sprague, R.H., A Framework for the Development of Decision Support Systems, MIS-Quarterly, Vol. 4, No. 4, (December 1980).

[25] Sprague, R.H. & Carlson, E.D., Building Effective Decision Support Systems, Prentice-Hall, Englewood Cliffs, N.J., 1982.

[26] Thompson, J.O., Organization in Action, McGraw-Hill, N.Y., 1967.

[27] Uhlig, R.P., Farber, D.J., Blair, J.H., The Office of the Future, North-Holland, Amsterdam, 1979.

[28] Ungson, G.R., Braunstein, D.N., Hall, P.D., Managerial Information Processing: A Research Review, Administrative Science Quarterly, Vol. 26, (March 1981).

[29] Wilensky, H.L., Organizational Intelligence: Knowledge and Policy in Government and Industry, Basic Books, New York, 1967.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

The Intelligent Management System: An Overview

Mark S. Fox

The Robotics Institute[1]
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

This paper describes the Intelligent Management System (IMS) project, which is part of the Factory of the Future project in the Robotics Institute of Carnegie-Mellon University. IMS is a long term project concerned with applying artificial intelligence techniques in aiding professionals and managers in their day to day tasks. This report discusses both the long term goals of IMS, and current research. It describes research in the modeling of organizations, constraint-based job-shop scheduling, organization simulation, user interfaces, and system architecture. Examples of working systems are provided.

1. Introduction

Classical research in the area of factory automation has been concerned more with production processes than with management. Yet, it has been observed that in many small batch-size factories, white collar labor accounts for a large fraction of total labor cost, and in some cases exceeds 50%. And that small batch-size production accounts for 50%-75% of the dollar value of durable goods produced in the United States. In metal-cutting, job-shop production environments, it has been found that only 20% of the time an order is in a factory, is it actually mounted on a machine. And during only 5%-10% of its time on the machine, are value-adding operations being performed (Am. Mach., 1980). There are (at least) two approaches to dealing with this problem. The first is to discover new methods of producing products that do not suffer from these inefficiencies. The second approach is to increase the effectiveness of professional and managerial personnel. The project described herein is concerned with the latter.

In the summer of 1980 we began the design and construction of what we call an *Intelligent Management System* (IMS). Research in IMS flows in two directions. The first is concerned with creating theory and systems whose functionality will aid professional and managerial personnel in their day to day decision making. These systems must be more effective in the tasks they perform. They must integrate and communicate the knowledge and skill of the whole organization, making them available for management decisions. More importantly, they must aid professionals and managers in carrying out tasks. Management systems must become more *intelligent*.

The second direction of our research is concerned with reducing the cost of creating, maintaining, and adding new functionality. It is not sufficient to increase the effectiveness of one part of an organization, e.g., managerial, by increasing costs in another, namely programming and system support. Yet much of the software constructed today, while providing increased functionality, also requires increased programming support. Research and development must be concerned not only with functionality but with adaptability.

The methodological focus of IMS is not the creation of an information system to support decision making as typically found in decision support research (Alter, 1979), but to explore more ill-structured problems (Simon, 1960) by means of heuristic problem-solving techniques. Ten years ago the promise of artificial intelligence techniques was unclear, but advances in the last decade have resulted in systems that display surprising capabilities and in some cases perform better than experts in the field. Examples can be found in Computer Layout: R1 (McDermott, 1980); Medical Diagnosis: Mycin (Shortliffe, 1976), Internist (Pople,

[1]This research was supported by the Robotics Institute, Carnegie- Mellon University, and, in part, by the Westinghouse Corporation.

1977); Geological Analysis: Prospector (Duda et al., 1978); Speech Understanding: Hearsay-II (Erman et al., 1980), Harpy (Lowerre, 1977). It is our goal to explore the application of artificial intelligence to managerial and professional problems.

This paper describes both the goals of the IMS research, and the research performed to date in organization modeling, operations control, management, and analysis, and in interfaces provided to the user. AI knowledge representation techniques form the basis of our modeling system. They provide both flexibility and richness in representation, resulting in a single model which can support a variety of functions. Constraint-directed and rule-based problem-solving architectures are used to perform organization control, management, and analysis. These architectures easily incorporate constraints/heuristics in the performance of control and management tasks such as job-shop scheduling and trend analysis. And natural language parsers and rule-based architectures are used to construct flexible user-interfaces.

The next two sections of the paper describe both the functional and architectural goals of IMS. Following them are sections describing research in organization modeling, analysis, management, and in user-interfaces and system architecture.

2. Functional Goals

The broad functional goals of the Intelligent Management System Project include:

1. Providing expert assistance in the accomplishment of professional and managerial tasks, and

2. Integrating and coordinating the management of the organization.

These goals were refined by analyzing professional and managerial needs. Over 30 managers from three plants in the Westinghouse Corporation were asked to record the types of questions they frequently faced during the performance of their jobs. Some of the questions were concerned with the status of activities. But the majority were in the area of "decision support", where the decisions spanned the areas of operations control to advance planning. The following is just a few of their replies:

- What is the effect of changes in engineering specifications: designs, materials, process specifications, etc?

- What is the proper inventory level at various levels, i.e., raw, work in progress, finished parts, etc?

- What if the jig grinder goes down with maintenance problems?

- Based on downtime, cost of maintenance and cost of replacement parts, how do I determine when to buy a new piece of production equipment?

- Charlotte moves one (1) BB72 rotor two (2) months ahead of schedule and Lester moves two (2) BB22 rotors back in schedule.

    1. How are all promised dates for next six months affected?

    2. What manpower changes are needed by cost center to accomplish changes?

    3. What material delivery changes need to be made?

- How do I select the process to be used? What if the shop changes the plan and needs to perform a particular job on a different machine from the standard?

- Not enough information concerning the current state of production on the floor, orders in process, problems etc. is communicated quickly and succinctly to interested parties within the plant.

- Changes to products or tools by the engineering division are not adequately communicated and coordinated with changes taking place in the factory.

- If I have a database containing lamp prices, sales volume, ics (internal product cost) transportation cost, overhead allotment, .. for all lamp types. What will the profit be if I change transportation, or change the source of manufacturing.

- Predict machine failures and prescribe preventative maintenance by sound vibration, machine timing, shrinkage trends, and end of normal part life.

- Determine correlations between gas, fill, temperature, stack up, and shrinkage.

In order to solve these problems, the Intelligent Management System must:

- *Sense*: Automatically acquire state data. Sense the location of objects, state of machines and status of activities both on the plant floor and in supervisory departments.

- *Model*: Model the organization at many levels of abstraction. For example, machines, people, materials, orders, departments, need to be modeled in detail from both an attribute and a process view, including their interactions, authority, and communication.

- *Operate*: Provide expert assistance in the accomplishment of complex professional tasks within the organization.

- *Manage*:

    - Analyze and manipulate the model to schedule production and resource utilization, and answer short and long term state and planning questions. The system in this role is *passive* in that it responds to user initiated queries.

    - *Actively* monitor the organization and inform responsible personnel when important events occur. For example, when a machine break down occurs, not only is the foreman informed, but also maintenance, and the salesman who must inform the client that the order will be delayed.

- *Analyze-Optimize*: Analyze how the structure and the processing of the organization should be changed to further optimize some criteria such as cost, throughput, and quality.

The above goals are concerned more with "what" functionalism is provided than "how" it is provided. Included in the functional goals of IMS is the need for better user interfaces. The following scenarios depict how the user interface should be integrated with the rest of the system.

"Tell me when ...": The marketing manager is under heavy pressure to get a rather large order out of the factory. He wants to be informed the minute it is shipped. He turns to his terminal and types the message: "Inform me when order X is shipped." His User Interface Process (UIP) translates the request into a rule "IF order X is Shipped THEN send message to manager Y", and sends it to the shipping Task Management Process (TMP). The TMP monitors the system to determine when the rule condition occurs. When the order is shipped, the TMP interprets the rule, resulting in the shipping message being sent.

"You've got problems ...": The milling machine breaks a tool while cutting a high priority order. The machine and order are damaged. The machine's sensors transmit the information to its Machine

Supervisory Process (MSP). The MSP analyzes the problem, and shuts down the machine. It then informs the floor supervisor and the scheduler TMP of the breakdown; the scheduling TMP re-routes orders. A message is also sent to the maintenance TMP which allocates a maintenance person to fix the machine. Lastly, the MSP checks the importance of the order, and informs marketing and other personnel of the problem, if it affects their tasks.

"What if ...": The manager in charge of production is considering the problem of a continually large back-log of orders. Should another machine be bought, or should the orders be subcontracted? He/she turns to his/her terminal and types in: "What are the effects on orders over the next six months if we buy another machine X?" The system then enters into a dialogue with the manager determining other information required to analyze the question. The UIP then scans the system wide functions that may help answer the problem. It finds a simulation module that can analyze structural changes in an organization. It gathers the initialization data and alters the factory model. It then runs the simulation, analyzes the output and provides the manager with the answer, and further explanations.

Obviously, creating a system that satisfies these goals will require many years of research and development. Never the less, it is important to understand the long-term goals so that short-term research is properly directed.

3. Architectural Goals

A necessary characteristic of any management system is that it provide the requisite functionality. But in most cases this is not sufficient. Unless the organization and its environment is static, system functions will quickly become inappropriate and fall into disuse. When the software maintenance is quoted to be 90% of software costs, this is due more to software adaptation than to software errors. We believe it is just as, if not more important, to attend to architectural goals as to functional goals. Then if the provided functionality is not applicable, the architecture will reduce the complexity of alteration.

In IMS, three levels of architecture are distinguished:

Hardware: The hardware level encompasses both the number and types of processors, and their networking.

Software: The software level covers the process architecture, process description languages, protocols for communication, module functionality, and databases.

User Interface: The user interface level covers the means by which the user can communicate with the system functions.

As an organizations changes, the hardware must also change, reliably providing services at the point of need. While timesharing systems are *accessible*, i.e., can provide services at the point of need, their *reliability* and *adaptability* are limited. (Some) Distributed systems connected by general communication networks can expand on demand, provide service at the point of need, and also shift processing loads when nodes fail.

Software must be *general* in the sense that modules, i.e. programs, can be used in more than one situation or application. For example, a factory modeling system can support more than one application such as simulation and scheduling. If the software does not provide the functionality it should not be difficult to *extend*. It should be extendible in the sense of program functionality, and in the sense that new modules and processes can be added to the system without requiring alterations to existing software. Currently, software systems are tightly coupled, requiring extensive re-programming whenever changes are required.

Lastly, the user interface must have the following characteristics:

Accessibility: Interfaces to computer systems are usually idiosyncratic and difficult to learn and use. Also, systems that change require that their users be continually re-educated. Our goal for IMS is to enable all personnel to meaningfully communicate with it. The interface will gracefully interact with the user and provide guidance and help in deciding what the user needs.

Accountability: A major obstacle to computer acceptance is that users are unable to question how and why output was generated. Our goal is to construct an explanation system which will allow IMS to explain its actions at various levels of detail.

Adaptability: As the user's needs change, the interface must be able to alter its processing and responses to fit the changes.

The results of our research have been targeted to run in a distributed, multi-processor and process environment (figure 3-1). Employee's will have a User Interface Process (UIP) that will act as an intelligent "aide". A UIP is composed of a personal computer, graphics display, keyboard, microphone, and network interface (e.g., a SPICE machine (CMU-CSD, 1979)). The UIP will have either voice or typed natural language input. It will act as an "aide" in the sense that it will interpret and implement user requests and queries. All UIPs will be inter-connected via a communication network allowing them to cooperatively interact to solve problems and communicate information. The UIP will also carry out many of the employees well-structured tasks automatically. Each machine will have a Machine Supervisory Process (MSP) which monitors and controls it. It is also connected to the network, and can reply to queries and commands initiated by other MSPs or UIPs on the network. Lastly, there are Task Management Processes (TMP). A TMP provides the focus for task management. It does more of the mundane task monitoring and control, freeing managers to do the more complex decision making tasks.

4. Organization Modelling

4.1. Introduction

The purpose of organization modeling is to provide the information base upon which intelligent processes rely. What the content of a model should be, and how it is represented is dependent upon the processes that use it. It is safe to say that an "Intelligent Management System" will require at least as much information as humans. The richness and variety of this information cannot be found in the databases of current management information systems. Nor is the form of the organization model related to current notions of database structures.

For example, a simulation system requires knowledge of existing processes including process times, resource requirements, and its structural (routing) relation to other processes. It must also know when routings for products are static, or are determined by a decision process such as a scheduler. In the latter case, it must know when and where to integrate the scheduler into the simulation. If IMS is to generate the sequence of events to produce a new product, it must have knowledge of processes (e.g., machines) which includes the type of processing it can do, its operating constraints, the resources it consumes, and its operating tolerances. If data is to be changed in an interactive, possibly natural language mode, IMS must have knowledge of generic processes such as machines, tasks, and departments if it is to understand the interaction. It must also know what information is important and how it relates to other information in order to detect missing information and inconsistencies. Hence, the organizational model must be able to represent object and process descriptions (structural and behavioral), and functional, communication and authority interactions and dependencies. It must represent individual machines, tools, materials, and people, and also more abstract concepts such as departments, tasks, and goals.

Consider a process model of a fluorescent lamp production machine group. The purpose of a process model is to represent each physical process and the causal relations which link it with other processes. For

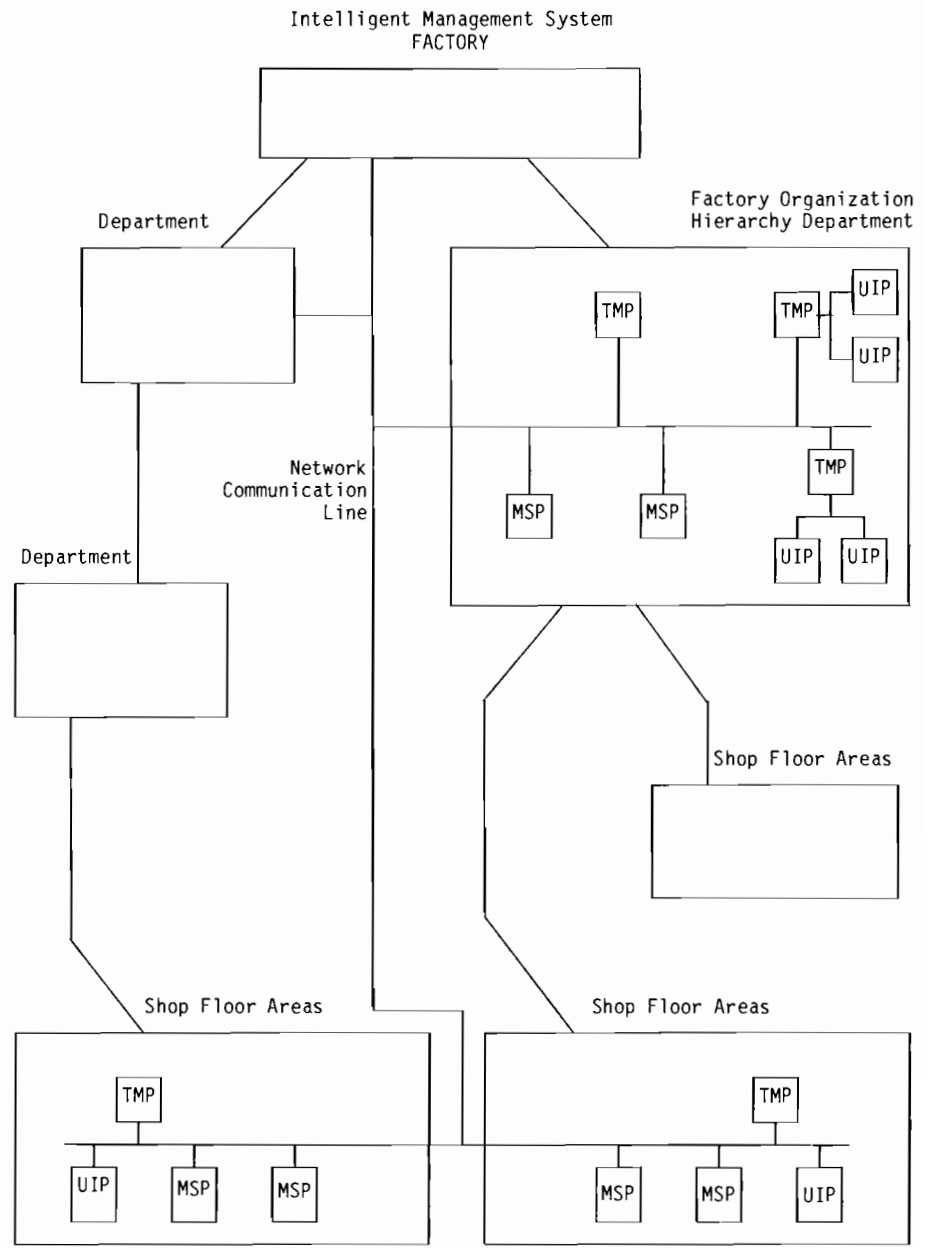


Figure 3-1: IMS System Architecture

example, the Lehr[2] process comprises of hundreds of subprocesses concerned with bulb grasping, positioning, heating, cooling, etc. Each is sequentially related with others in time. The performance of each is related to the performance of previous subprocesses. Each subprocess is described in terms of how it physically performs, its three dimensional movement, what and how it may grasp and heat an object, what object it expects, operating constraints, etc. The scope of the description is limited only by the uses to be made of it, and the information available.

One use of such a model is for machine diagnosis. Engineers spend a great deal of time watching a malfunctioning machine to determine what has gone wrong and what variables to alter to improve performance. One of the major stumbling blocks in the systematic analysis of such systems is the unavailability of process instrumentation for data collection. Yet the availability of the data solves only half of the problem. The other is the automatic analysis of data to find relations between system parameters and productivity. Statistical correlations are only a small part of the analysis. Statistics alone can only *suggest* relations. *Understanding* whether the relation is valid requires a thorough knowledge of the process itself and all its interactions. This cannot be performed without a sufficiently rich model of the process describing physical, functional, and time relations.

Another use of a model is to provide process cost analysis. Given a model, questions about the resource consumption and production of each process and subprocess can be answered. The individual process descriptions can be integrated across the group to provide summary cost information. But more importantly, because the model represents not only the process but the effects and relations among processes, questions related to process alterations can also be analyzed. For example, what will the effect on cost be of changing a Lehr process to use microwave heating. And from a diagnosis point of view, we would want the system to determine whether the change would have any significant effect on shrinkage[3]. For example, the temperature of the glass may be lower after microwaving, hence the following process, end sealing, may not have glass at a high enough temperature.

In summary, the modeling system should provide:

1. A rich source of information.

2. A context within which all IMS modules can interact.

But if it is to be used in a changing environment and by non-programmers, it should also provide the following features:

Accessibility: the model user should be able to easily peruse the model to determine its attributes and structure.

Extensibility: the model should be alterable by its users.

Consistency: when a model is created or extended, the modeling system should be able to determine when the model contains inconsistent information.

4.2. Basic Modelling System

The above discussion presents a case for a modeling system that supports a variety of functionality such as simulation, cost analysis, and process diagnosis, and a flexible user interface. Therein lies the question: does such a modeling system exist? Traditional, computer-based modeling systems fall into four categories:

---

[2] Baking lacquer out of lamp phosphor.

[3] loss through defects.

1. Mathematical.
2. Discrete event, facility based.
3. Continuous.
4. Arbitrary program.

In looking at these modeling systems, we found that:

- They are too problem specific, hence inflexible.
- The modeling techniques are difficult to learn by managers and engineers who are the prime users.
- Alteration may require substantial change to the model and related systems.
- Once constructed, the models are difficult to understand, peruse and verify.

The Intelligent Management System has at its core, a model of the organization. This model is shared by all subsystems to achieve their tasks. The modeling system provides the following features:

- The model is composed of declarative objects and relations which match the users conceptual model of the organization.
- The modeling system provides a library of objects and relations which the user may use, alter, and/or extend in their application.
- The model incorporates a variety of representational techniques allowing a wide variety of organizations to be modeled (continuous and discrete). And it is extensible, allowing the incorporation of new modeling techniques.
- The user interactively defines, alters, and peruses the model.
- The model can be easily instrumented. For example, the model can be diagrammatically displayed on a color graphics monitor at different levels of abstraction. The complete organization, or parts thereof, can be viewed with summaries (e.g., queue lengths, state).
- The modeling system is simple to learn to use because the modeling tools match the concepts people use to think about problems.

The modeling system is based on the knowledge representation system SRL: Schema Representation Language (Fox, 1979a; 1982). SRL has its basis in schemata (Bartlett, 1932), which have come to been known as frames (Minsky, 1975), Concepts (Lenat, 1976), and Units (Bobrow & Winograd, 1978).

The basic unit for representing objects, processes, ideas, etc. is the **Schema**. Physically, a schema is composed of a schema name (printed in the bold font) and a set of slots (printed in small caps). A schema is always enclosed by double braces with the schema name appearing at the top.

```
{{ Machine

        CAPACITY:
        QUEUE:
        OPERATOR:
        CONTENTS:
        LOAD:
        UNLOAD:
   }}
```

Figure 4-1: Machine Schema

The **Machine** schema (figure 4-1) contains six slots, some which define physical limitations of the machine, i.e., CAPACITY, some which define its current status, i.e., OPERATOR, and some which define event behavior, i.e., LOAD. Slots can have simple values (figure 4-2).

```
{{ Machine

        CAPACITY: 3
        OPERATOR: joe
        CONTENTS: lot-29
        LOAD:
        UNLOAD:
   }}
```

Figure 4-2: Machine Schema with values

Schemata can be more complex. Each slot has a set of associated facets (printed in italics) (figure 4-3). The *Restriction* facet restricts the type of values that may fill the slot. The *Default* facet defines the value of the slot if it is not present.

{{ **Machine**

CAPACITY:
Value: 3
OPERATOR:
Value: joe
CONTENTS:
Restriction: (TYPE is-a product)
LOAD:
Restriction: (SET (TYPE is-a rule))
Default: load-rule
UNLOAD:
Restriction: (SET (TYPE is-a rule))
Default: unload-rule
}}

Figure 4-3: Machine Schema with facets

And each filler of a *facet* may have one or more pieces of meta-information termed characters (printed underlined) (figure 4-4). The Filler character defines the value of the facet. Creator defines who created the filler, and Creation-Date defines when the filler was created.

{{ **Machine**

CAPACITY:
Value:
Filler: 3
Creator: shop-supervisor
Creation-Date: 22-OCT-79
OPERATOR:
Value:
Filler: joe
CONTENTS:
Restriction:
Filler: (TYPE is-a product)
LOAD:
Restriction:
Filler: (SET (TYPE is-a rule))
Default:
Filler: load-rule
UNLOAD:
Restriction:
Filler: (SET (TYPE is-a rule))
Default:
Filler: unload-rule
}}

Figure 4-4: Machine Schema with characters

An important aspect of SRL is that schemata may form networks. Each slot in a schema may act as a relation tying the schema to others. The schema may *inherit* slots and their fillers along these relations. Consider the schema for a **Continuous-machine**.

```
{{ Continuous-Machine
   { IS-A Machine
      USED-CAPACITY:
      LOAD: { INSTANCE # rule
               IF: (< USED-CAPACITY CAPACITY)
               THEN: (fill USED-CAPACITY ( + 1 USED-CAPACITY))
                     (add object CONTENTS)
               ELSE: (add object QUEUE) }
      }
   }}
```

Figure 4-5: Continuous-Machine Schema

Figure 4-5 defines a CONTINUOUS-MACHINE which works much like a pizza oven, it can be continuously filled up to capacity. A **Continuous-Machine** IS-A **Machine**. The IS-A relation between the two schemata allows **Continuous-Machine** to inherit attributes (slot names) and their values from the **Machine** schema. The LOAD slot defines the behavior of the machine when a load event occurs. The loading rule tests whether the machine has capacity, if so the object is placed in the machine, otherwise it is queued.

SRL provides the model builder with the ability to define new schemata and slots, and to define the inheritance semantics of slots which act as relations. This includes defining what information, i.e., slots and their values, is inherited, not inherited, and altered when inherited.

The **Machine** and **Continuous-Machine** schemata are generic schemata that form part of the basic modeling system. The system contains a variety of basic schemata which the model builder can use to model an individual organization. An organization model is constructed by instantiating the basic schemata with appropriate attribute values, e.g., capacity, and possibly, new behavioral rules. Schemata are structured (linked) through a user extendible set of relations: IS-A, INSTANCE, PART-OF, etc. For example, a circuit board baking oven (figure 4-6) can be instanced as an INSTANCE of a **Continuous-Machine**. It has a CAPACITY of 10, and inherits its loading rule from **Continuous-Machine**. It is also part of a **Work-Area** in the plant called the **Baking-Shop**. The PART-OF relation allows the inheritance of locational information.

```
{{ PcOven

   { INSTANCE Continuous-Machine
      CAPACITY: 10     }

   { PART-OF Baking-Shop  }

   }}
```

Figure 4-6: PcOven Schema

Another type of schema used in project modeling and management is the **activity** schema (figure 4-7).

{{ **activity**
PRE-CONDITION:
SUSTAINED-CONDITION:
POST-CONDITION:
SUB-ACTIVITY: }}

Figure 4-7: Activity Schema

An **activity** is defined as having four slots (attributes). The PRE-CONDITION defines what must be true before the activity is to take place, e.g., materials available, previous activities finished. The SUSTAINED-CONDITION defines what must be true throughout the activity, e.g., water-level in machine is to be maintained above a certain level. The POST-CONDITION defines what must be true when the activity is finished, e.g., the temperature of the part is above 100. Finally, the SUB-ACTIVITY points to the sub-activities which comprise the activity. Activity schemata can be constructed into a network to define both parallel and sequential precedence, and hierarchical to describe activities to many levels of detail (decomposition).

4.3. Model Accessibility and Extensibility

Current management information systems require the use of programmers when changes are made to the organization model. While such an approach is fine for domains where the model changes seldomly, the dynamics of a factory organization require continual updating of the model; both parametric and structural changes are constantly occurring. In lieu of employing a brigade of programmers to implement changes, the alternative is to construct a model acquisition system that allows the person initiating the change, to directly inform IMS of the modifications.

The IMS modeling system currently provides the following functionality:

- ***Accessibility***: The user may interactively view each schema in the model using a variety of schema printing functions. Relational hierarchies can also be displayed. Color graphic display of the model is supported at various levels of abstraction.

- ***Extensibility***: An interactive schema editor allows the user to create and alter schemata in the model.

- ***Consistency***: A model consistency language and checker has been developed (Reddy & Fox, 1982). The consistency checker uses the consistency specifications to check a model and reports inconsistencies to the user.

Though the above mechanisms provide a reasonable user interface to the modeling system, our ultimate goal is to allow managers to alter the model directly. By means of a natural language interface, the manager should be able to describe to IMS changes that have occurred in his area of responsibility. To achieve this, the modeling system interface must:

1. Determine what part of the factory model is affected by the new information.

2. Check to see if the new information is consistent with what it already knows about the factory.

3. Determine the effect of the change on other parts of the model, and make the appropriate changes.

4. Query the manager when inconsistencies appear in the reconciliation of the new information

with the existing model.

A natural language understanding and discourse modeling system to support model acquisition, factory layout, and job-shop scheduling is currently under development.

4.4. Flow-Shop Modelling

A complete model of a printed-wire-board (PWB) bareboard production plant[4] has been constructed to the machine level (completed August, 1980). The factory consists of a number of areas (work areas, service areas, offices etc.) where different activities take place. Different machines are located in work areas and perform individual operations. A circuit-board is produced by performing a series of operations on the raw material. All work pieces waiting for an operation wait in a queue in front of a collection of machines or in a centralized in-process storage. The flow of work is controlled by the "operation-lineup" associated with the product being manufactured. The operation-lineup specifies the sequence of operations which will be used to schedule the next operation to be performed on the work-piece. The factory is configured such that "work-pieces" flow to various work-areas on a centralized conveyor system. If there is no space for a work-piece in a given work-area, it is stored in a centralized in-process storage from which it could be recalled when needed. The model contains 17 work-areas, 48 machines (both discrete and continuous), 34 queues, and 30 different operations.

Figure 4-8 provides a glimpse of part of the relational structure of the model without each schema's slots. You will note that information on how to simulate and display the factory is embedded in the model. The model also includes schemata for operations, process sequences (lineups), products, personnel and others. The model has been used directly by our simulation system and to display the factory on a color graphics monitor. Figure 4-9 displays the complete layout of the plant. Each work-area is color coded according to type of processing. Under each name is the number of orders in process and queued for the work-area. At 1:15 there are 9 orders in the inspection area. The UIP allows the user to specify what part of the plant to display. Figure 4-10 shows a blow up of the inspection area. At 1:36 there are 6 orders in the inspect1 station and 8 orders in the touchup station.

Process flow is defined by a production (operation) lineup defined for each product type. When an order is unloaded from an object such as a machine, work-area, etc. the next operation is determined by information inherited by the object. For some objects access to a scheduling system is inherited via the PART-OF relation, for others the next operation is defined directly due to physical coupling of operations.

4.5. Job-Shop Modelling

A model of part of a turbine component production plant[5] has also been constructed to support simulation (section 5.2) and job-shop scheduling functions (section 6.2). The model contains information about machines, products, tools, work-centers, labor and cost data, and factory layout. Much of the schemata used to model the circuit-board plant were used in the turbine plant model. In some cases additional slots (attributes) were specified, e.g., cost data, and operation sequences were expanded to operation graphs to include alternate processing routes in the plant.

4.6. Machine Modelling

Our third modeling project, currently underway, is to construct a model of a fluorescent lamp production line; including enough detail to support machine process diagnosis (section 6.4). This model focuses on the pre and post-conditions of individual machines in the production process, and on the causal relations that exist between and within the machines.

---

[4] under design by the Westinghouse Corporation.

[5] Westinghouse Turbine Component Plant, Winston-Salem NC.

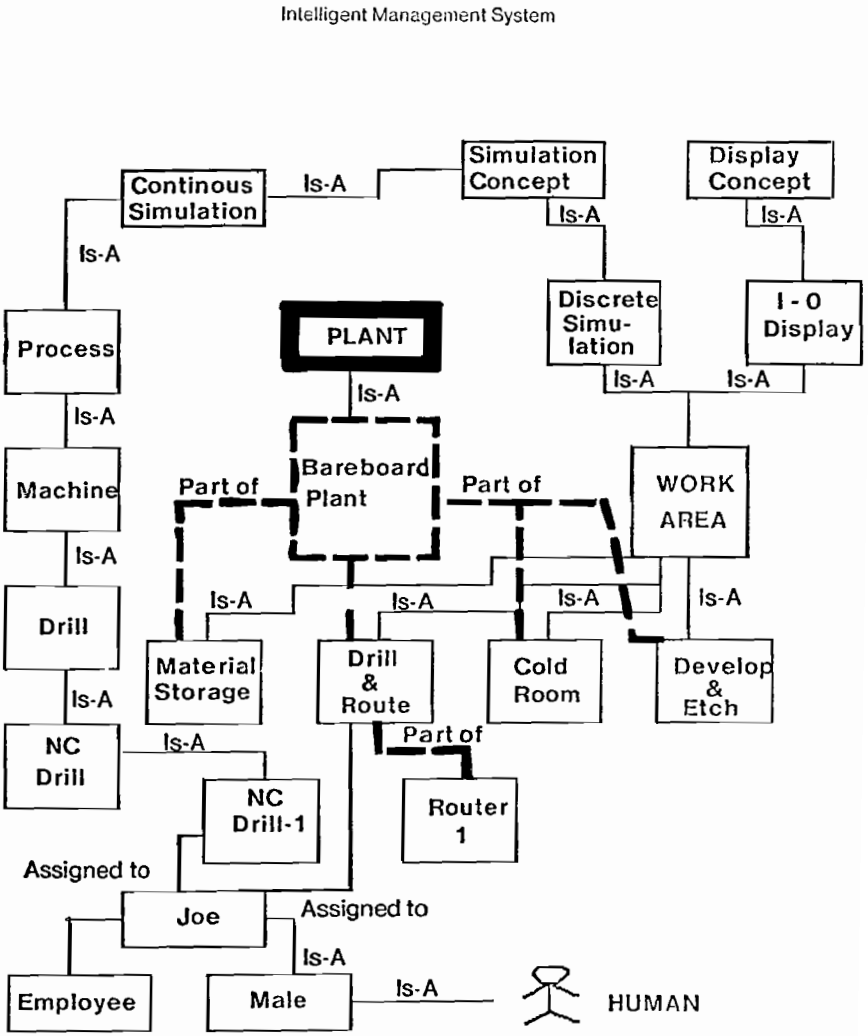


Figure 4-8: Flow-Shop Partial Model

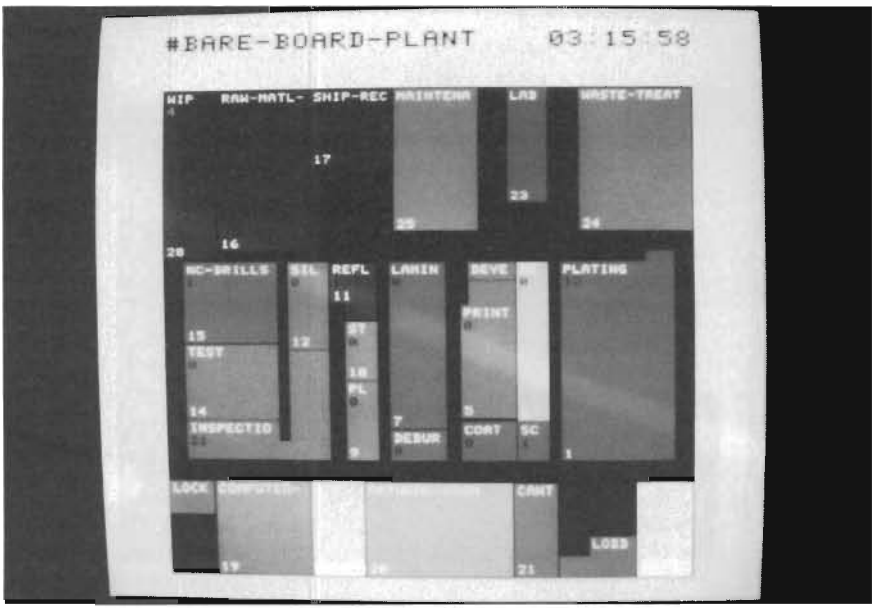


Figure 4-9: Monitoring the Factory

5. Organization Analysis

5.1. Introduction

The effectiveness of an organizational is determined by its ability to deal with environmental uncertainty and complexity. Environmental uncertainty and complexity coupled with organizational uncertainty and complexity results in sub-optimal, and even sub-satisfying organizational behavior. To produce requisite behavior, an organization must analyze its environment and adapt. But it is too often the case that management lacks the time or ability to carry out the analysis; hence the organization internalizes more rigid behavior. One of the most important but least understood aspects of an organization is its ability to adapt. Much of Organization Theory has been concerned with determining environmental and task characteristics that affect organization structure (March & Simon, 1957; Galbraith, 1973; Williamson, 1975). At best the results are descriptive.

Tools do exist that aid in the analysis of organizations. In particular, a simulation system allows one to test structure and processes. But the cost of constructing a simulation can be prohibitive, with the resulting system being of little use except for running the same or similar simulations again. Simple "what if" questions cannot be answered readily. A manager requires an intermediary, such as a system analyst, to answer the question. There is a definite need for more sophisticated tools for analyzing organizations, and for providing usable tools directly to the managers and professionals. The following describes some of the research in IMS towards these goals.

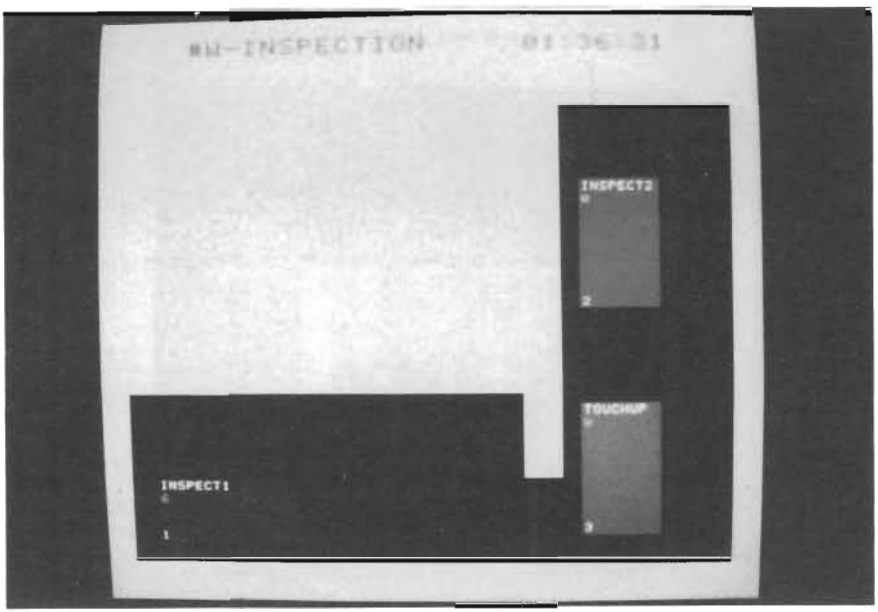


Figure 4-10: Monitoring the Inspection Area

5.2. Simulation

Many of the "what if" questions encountered in our analysis were concerned with the understanding of structural changes in the factory. For example, the decision to buy a more flexible but more expensive machine depends on its effect on the performance of the factory. In a job-shop, it is difficult to answer question analytically due to the complexity of the model. Simulations are used to predict the performance of complex systems. Hence they are a potentially useful tool to provide managers with. Why do research in this area when systems like GPSS, Simscript, etc. already exist? The reasons are many:

- Current systems are difficult to learn to use and require extensive computer programming skill.
- They suffer from the modeling problems described in the modeling section.
- Output is limited to the analysis provided by the system.
- Most systems are not interactive, and cannot be monitored easily while they run.

A discrete simulation system has been constructed that interprets the organization model directly (Reddy & Fox, 1982). The model represents the complete organization, not just the information necessary to perform a simulation. The simulation uses as little or as much of the model as it requires. Hence the user only has to modify the general model without having to alter the simulation system. Some of the characteristics of the simulation system are:

- It is interactive, allowing the user to start, halt, and resume the simulation.

- The user can query the state of entities in the simulation as the simulation proceeds.
- The organization is displayed on a color graphics device with pertinent data (e.g., queue sizes) changing as the simulation proceeds. The user may view the organization at several levels of abstraction as the simulation runs.
- The user can interactively alter the model while the simulation is in progress and the simulation will continue with the alterations.
- The user can display a variety of analysis at the terminal.
- It accesses the *same* organization model that other functions use. Hence, only one model is maintained in IMS.

The simulation system proper is small. Its sole purpose is to manage the event queue. How events are interpreted is defined in the model. For example, a load event of the **Pcoven** (figure 4-6) would be accomplished by evaluating the contents of the **Pcoven**'s LOAD slot, which is inherited via the IS-A relation from the **Continuous-Machine** schema (figure 4-5). The contents of an event slot is a set of rules defining the event's behavior.[6]

Data gathering for analysis is accomplished by associating data gathering rules with the appropriate slots in the model. In particular, rules can be specified to be evaluated anytime the contents of a slot are changed.

Simulation monitoring is handled in the same manner as data gathering. Display rules are associated with slots in the model. When a slot value is changed, the associated rules are evaluated, resulting in display changes.

Figure 5-1 shows one of the interfaces. The upper right window displays each event as it occurs. It is maintained by display rules in the event scheduler. The lower right window monitors the queue of a user specified facility. The lower left window follows an order through the factory, printing each event as it occurs. The upper left window displays the available commands. Additionally, the user can monitor the factory via a graphics display (figures 4-9 and 4-10). The graphics display gives factory wide or close-up information, e.g., queue lengths, machine status, etc.

The simulation system is part of a general analysis system to be used by managers directly. Research is continuing to extend its capabilities (e.g., multi-level simulation, continuous system simulation, etc.) and making the interface simpler to use by managers.

5.3. Structure Analysis

Whether generating a more optimal process, or diagnosing an existing one, the organization must be analyzed. Analysis methodologies fall into two categories: analytical and experimental. The latter approach includes simulation, and is used when analytical techniques do not exist, are intractable, or the required information is missing. But analytical approaches are still very useful, especially in a local setting. Many organizational changes can be measured by analyzing the causal relations that exist "around" the proposed change. By following causal links and measuring their change effects, most generated changes can be adequately tested. The generation of changes is dependent upon the existence of causal or dependency information that can denote high expense areas measured by some criteria.

In the laboratory, we are extending the organization modeling research to include organization structure

---

[6] Klahr & Fought (1980) use a rule-based approach to simulation.

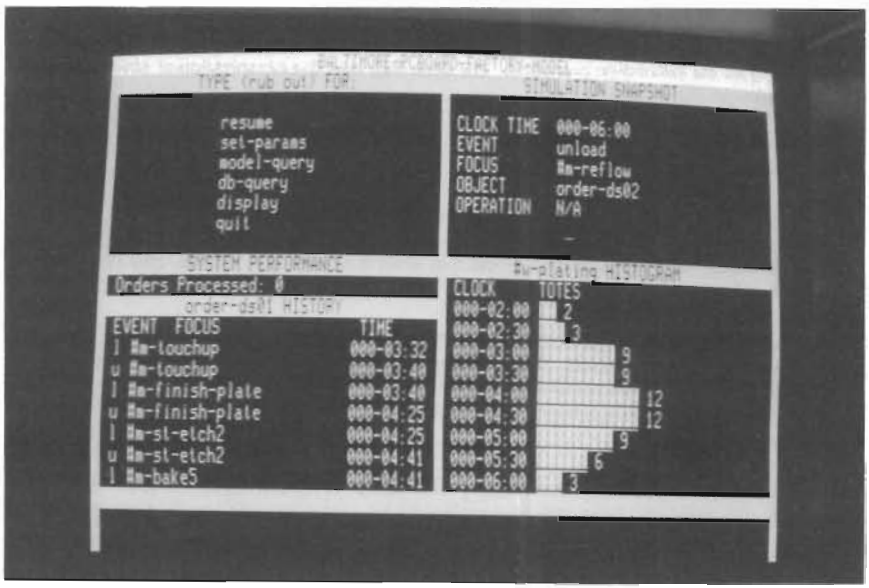


Figure 5-1: User Interface Display for Simulation

analysis. Our goal is to construct a system that can analyze an organization model and suggest structural changes in order to improve performance as measured against some well defined criteria such as profit, reaction time, and quality of output.

6. Organization Operation and Management

6.1. Introduction

The purpose of our research in organization operation and management is to provide intelligent aids to managerial, professional and salaried staff. While the primary goal is to provide expert level functionality, other factors must be considered. First, the aid must be interactive to allow access in a timely fashion. It must be integrated with the rest of IMS so that it can access available information and functions, communicate with other users, their UIPS, TMPs, and MSPs. Just as important, the aid must be constructed in a manner that its declarative and procedural knowledge can be easily altered and expanded. In order to achieve these goals research from artificial intelligence, computer science, and management science have been used in the construction of these aids.

6.2. Job-Shop Scheduling

One of the most important functions in a factory is scheduling. Simply, scheduling is a resource assignment problem which is constrained by resource availability of one form or another and organizational goals in a job-shop. Scheduling in a job-shop is a problem because there is more than one way of

producing an item, and that all orders are competing for the same resources (machines, personnel, materials, tools, etc.). In the factories we are working with, there are several thousand *distinct* product models. At any given time, there could be between one and several hundred duplicate units of a particular product model on the floor. There are many alternative routes that each product model could travel through the factory, depending on the availability of physical production resources, other active orders on the floor, and on the relative importance of time, cost and other cost constraints.

Job-shop flexibility is only part of the problem, dynamically changing factory status is another. Although the physical number of machines in the plant, or the number of operators employed does not change too often, the set of machines, tools, and workers that are *actually* available at a given moment is constantly changing. Machines break down often, at unpredictable times. A worker may get hurt and have to leave the shop floor, leaving the machine unattended. Small tools may disappear because the operator forgot to return them to the tool crib, or lent them to someone else without recording it.

Existing job-shop scheduling systems, for the most part, are inflexible. They are based on a "static" model of critical production resources, based on the number of machines, tools and workers which are available under "ideal" conditions, and **not** on the set of production resources which are *actually* available at a given moment. These "static" models can not be modified at the rate at which changes occur. As a result, these systems are limited in their usefulness, since there are few times when the "static" model inside the computer matches actual operating conditions. Many factories which currently have these types of systems *still* have to schedule manually because of the frequent changes in operating conditions.

Another, more important, problem with existing scheduling systems is that they do not account for *all* the constraints that human schedulers use in manually constructing schedules. Interviews with schedulers and other factory personnel has shown that they receive information from more than 20 sources in the plant, such as forging, tooling, NC programming, and materials, marketing, and forecasting. Much of the scheduler's time (80%) is spent gathering these constraints, and then constructing a schedule that satisfies as many of the constraints as possible. Hence, the problem of scheduling can be viewed as the problem of constraint satisfaction.

An interactive, real-time, job-shop scheduling/operations control system has been constructed which interprets the factory model directly. It is called ISIS: Intelligent Scheduling and Information System[7]. It schedules orders on machines for complex job-shop environments where orders require a large number of operations, and there are many different ways of producing an order on machines under high contention. It allows the user to enter orders and specify scheduling constraints. Existing constraints include:

- start and due dates
- operations costs
- operation alternatives
- machine alternatives
- machine down-time and maintenance
- order priorities
- order lottings
- lot priorities
- work in process levels
- production goals
- shop stability requirements
- machine productivity
- shifts available
- resource availability

[7] ISIS-I was demonstrated in December 1980 at CMU, a second version, ISIS-II was demonstrated in December 1981.

- machine attributes
- operation requirements

The system will generate a schedule that attempts to satisfy the constraints. The system uses a constraint-based heuristic search to construct forward (from start date) or backward (from due date) planned schedules. Constraints are not built into the algorithm, but are part of the factory model. The user can add, alter, and remove constraints. ISIS dynamically resolve what constraints to use and where to use them during the scheduling process. The scheduler is a factory floor level system which updates its factory model (e.g., machine, order, personnel, resource status) either by user input or through computer messages. And the system does a minimal rescheduling in reaction to any state change in the factory. Figure 6-1 depicts the scheduling system after it has constructed a schedule. The right window is the operation sequence chosen, the left is the machine reservations for the lot.

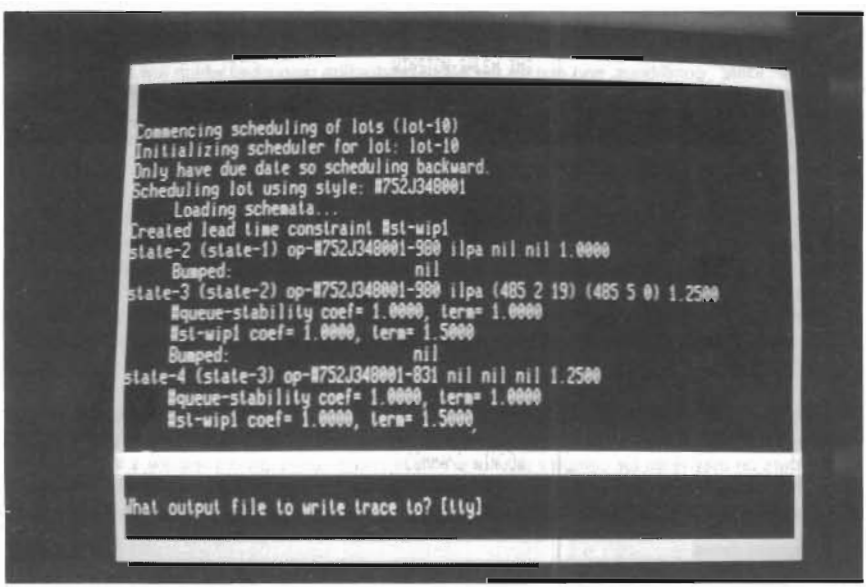


Figure 6-1: Scheduling System

It is often the case that many of the constraints cannot be met. Under these conditions the human scheduler bargains with the constraint sources to have them altered. It follows that a scheduling system should not only take into account the large number of constraints, but should consider the conditions under which the constraints can and should be *relaxed* and/or *strengthened*. For example, if there are normally five workers available to perform a task on a given shift, and there is a rush order to get out, then the system should recognize the workers available constraint can be relaxed so that more workers are put on the the job. From a problem solving point of view, in addition to knowing what operators can achieve a goal (e.g., machinists), you must know what and how operators can be added, removed or modified. Hence the system must:

- Consider all constraints that may impact a scheduling system.

- Attempt to satisfy them according to importance.
- Decide *when* constraints should be relaxed.
- Choose *what* constraints should be relaxed.
- Decide *how* to relax the constraint.

ISIS-II can and does selectively relax a subset of constraints. An new version, ISIS-III, is under development. It will have the capability to relax any constraint. The research emphasis of this system is to extend the general representation and utilization of constraints, and on how these constraints can be dynamically relaxed.

6.3. Factory Monitoring

As discussed in the section on simulation, the display facilities used to monitor the simulation are attached to the model. Hence, if the model is updated from real-time messages from the factory floor, as opposed to the simulation, figures 4-9, 4-10, and 5-1 can be used to monitor the real-time operation of an organization.

6.4. Process Diagnosis

Complex production environments may introduce a variety of defects into a product. The defect may not appear until later in the production process, where cumulative processing compounds the earlier induced defect to a point where signs of its manifestation appear. The purpose of process diagnosis is to understand each step of the production process to determine what type of problems can occur and the causal (cumulative) relations that can exist amongst sequential steps in the production process. The goal is to construct a system that monitors the production process, and when defects occur, analyze the problem to determine the possible points in the production process which could have caused it. A machine group[8], monitoring, signalling, and control system has been designed, and the prototype simulation has been demonstrated. Our next step is to construct a training simulator for new operators, before completing the diagnosis system. The rational for this multi-step approach is both pragmatic and theoretical. The group does not contain adequate sensoring to support diagnosis at present, but new senors will be added over the life of the project. The simulator project, when completed, will provide the group model for the diagnosis project.

7. User Interfaces

7.1. Introduction

An obstacle preventing the introduction of intelligent systems into organizations is the lack of reasonable interfaces. The keyword here is reasonable. The reasonableness of an interface is not absolute. It depends upon the person accessing the system: knowledge of the system, expectations, time constraints, etc.; the style of the interface: menu, touch screen, voice, typing, interactive, batch, and the functionality of the system. Successful systems tend to constrain the expectations of the user, and then optimize the interface for the constrained task (e.g., ZOG (Robertson et al., 1979), DAISY (Buneman et al., 1977)). This is sufficient when the task itself is naturally constrained, e.g., sales order entry, inventory control. But many of the ill-structured tasks that managers do, e.g., planning, forecasting, are not well enough understood, let alone highly constrained. If the goal is to introduce intelligent aids at the professional and managerial level, then the interface's acceptability will increase as the interface closely approximates typical human interactions.

---

[8] A machine group is a set of machine integrated into a automated flow shop. The machine group produces fluorescent bulbs.

The purpose of our user interface research is to explore the problem of interfacing people with machines by incorporating dialogue and problem-solving capabilities in the UIP. During the first year of the project, research focused on modeling, managing and analysis functionality. Little was done in the area of user interfaces. Since then our emphasis has shifted to include interfaces. The following describes both our accomplishments and goals.

7.2. Information Display

In a multiple process environment, monitoring more than one process can be difficult when using only a single display. Previous solutions to the problem entailed programming each process to cooperate in the use of the display, essentially hard-wiring their interaction. Changes in processes, information to be displayed, etc. required each process's display code to be altered. We have solved the problem by providing a communication and display system that allows processes to communicate information for display, independent of who is to receive it and the device upon which it is to be displayed. This allows a process to dynamically communicate with any other process and have the communication appear on a screen in conjunction with information from other processes, without interference or recoding (see figure 5-1).

7.3. Natural Language Interface

The goal of *accessibility* can only be achieved if users are able to communicate with computers in a language that is natural to use. An obvious candidate is English. Over the last 20 years, researchers in artificial intelligence have constructed natural language understanding (NLU) systems that allow users to type English statements and questions into the computer. The state of natural language understanding research does not allow the computer to understand arbitrary sentences; they must be restricted to a particular task or domain. But, research continues to expand the capabilities of such systems.

In IMS, simple, constrained, English input is being experimented with. Two tasks have been chosen, model building and scheduling, to explore the use of natural language interfaces. Using the parsing system of Carbonell & Hayes (1981), a grammar for scheduling has been constructed and is being tested.

7.4. Explanation

Much of the failure of computer systems can be ascribed to their inability to *account* for their results. Once the output is produced, the user cannot delve into how results were produced without reading the computer program. Since many end users are not programmers, this is intolerable.

An explanation facility provides the means by which a user can investigate how results were derived, why a particular action was executed, or what the current state of the system is. The explanation facility is a natural language generating system which is able to maintain a discourse centered around analyzing a particular action or state. The purpose of NLU is to translate English sentences into a representation that the machine understands. The explanation system describes the machine's reasoning processes upon request.

7.5. Discourse Modelling

The act of describing an organization can be quite complex from a natural language understanding point of view. Not only must each sentence be understood, but ideas communicated earlier are continually referred to directly and indirectly. Consider the following description:

There is an inspection area.
It seats 12 inspectors.
They have a desk, magnifying lamp and a terminal.
The room is next to the cold room

In order for the UIP to carry out a conversation, it must track the flow of ideas, bind referents, watch for and signal inconsistencies, etc. This ability is referred to as discourse modeling.

7.6. Planning

The true mark of intelligence in management systems is the ability to use the best information and apply the best methods in answering a user request. Consider the task of determining the effect of adding or changing a machine in a job shop. There may be many ways of doing this such as statistical analysis, simulation, and looking at similar situations. Deciding exactly how to measure the effect is a planning process, where a sequence of tests and actions must be constructed to arrive at an answer. This sequence is chosen from a set of competing methods. Planning requires knowledge of what tools are available, how effective they are, and the conditions under which they are to be used. Using a tool may require planning the use of other tools to provide the required conditions. For example, before doing a simulation, a model must be constructed, and initialization data be gathered.

If the user interface is to act as an intelligent aid, it must be able to extend the user's capabilities by determining the most appropriate means to implement the user's request or command. The planning mechanism examines the data and functions known to it and constructs a plan whose execution accomplishes the task.

7.7. Personalization

The user interface process can be viewed as an extension of the user. But users differ in a variety of ways: task prescription and language of communication for example. The UIP must *know* the user in order to amplify or restrict the user's capabilities according to their position in the organization. Consider the machine operator. He should be able to communicate with his UIP using the vocabulary of his task, he should be able to provide status information, and request scheduling information, but he should be restricted from finding out the plant manager's salary.

8. Integrating Distributed Systems

Research is proceeding in the *flexible integration* of the multiple functions of the Intelligent Management System. As described in section 2, IMS will be comprised of processes of type UIP, MSP and TMP[9]. These processes will run on processors spread throughout the plant and are connected by a communication network (figure 3-1). Any process may dynamically spawn other processes according to its problem-solving needs. For example, a scheduler's UIP may have to analyze the effect of alternative priority ratings on orders over the next year. The UIP first determines how the question can be answered. It communicates with a module librarian process (TMP) to see if there are any modules in IMS that may help in answering the question. There is a module called "simulation" that can simulate a discrete-event model over time. It also finds that there are modules that can instrument a model to gather data over time. And a module that can analyze time-series data. The UIP acquires the module definitions from the librarian and spawns as many processes as needed to execute them. Knowledge of what modules are available to solve a problem or answer a question is not programmed into a module, but is part of a module's goal description which is accessible by other modules in the system. This is in the tradition of stimulus-response frames of Hearsay-II (Hayes-Roth & Lesser, 1977).

The key points in this organization of processes are:

1. Modules have been created that provide *generic* functionality.

2. Modules are described in a machine interpretable manner.

---

[9]Versions of IMS have been created with multiple MSPs being simulated, and communicating with a UIP. Also UIPs for scheduling and organization analysis have been created.

3. Processes have problem-solving capabilities that allow them to determine what modules are appropriate to solve a problem or answer a question.

4. Processes can communicate needs and information.

A language for describing both software system architectures and individual modules has been created (Fox, 1979b). ODL (Organization Design Language) can be used to define module capabilities, resource usage, goals, and the system architecture. By describing IMS tasks and tools (modules) with ODL, processes can reason about module and system applicability to problem situations. Additionally, a simple information protocol has been developed to allow processes to request and provide information.

In addition to function availability, data accessibility is important to maintaining system flexibility. When a module is instanced as a process, a process schema (description) is created, defining who created the process and what rights it has in the current environment. A process has its own local data space which contains the information necessary to carry out it's task. When a process requires information not defined locally, it uses its process schema and the module descriptions of other processes to determine where the information may reside. Once having determined accessible processes, it sends messages requesting the information. Upon receipt of the information, it is cached in the process's local data space. This technique has been used by the simulation system to acquire only the parts of the factory model it needs for a simulation.

9. Conclusion

Most of the costs of producing products in complex organizations are attributed to overhead, much of which is comprised of managerial, professional, and salaried personnel. If significant productivity gains are to be made in this decade, more attention must be paid to aiding both the professional and the manager. The Intelligent Management System is a step towards this goal. By combining artificial intelligence, computer science, and management science techniques, more intelligent aids and solutions for the operation and management of complex organizations can be found.

This article provides an overview to the Intelligent Management System. It provides a glimpse into a goals and the systems we are creating and have created to date. In the area of organization modeling we have completed a second version of an interactive organization design system and have applied it to the modeling of three plants. In organization operations, we have created the second version of a constraint-based job-shop scheduling and control system and our about to install it in a test site. And in organization analysis, we have created an interactive simulation system which uses the same model as scheduling. Each of these working applications represent a piece of an evolving Intelligent Management System.

10. Acknowledgments

IMS is the result of the efforts of many people in the Intelligent Systems Laboratory. They include Brad Allen, Don Cohen, Bryan Griffin, Jenny Hood, Carol Johnson, Joe Mattis, Drew Mendler, Steve Miller, Tom Morton, Ramana Reddy, Gary Strohm, Ari Vepsalainen, and Mark Wright. I also wish to thank Jaime Carbonell, Bill Ferguson, and John McDermott for their comments on earlier drafts of this report.

11. References

Alter S.L., (1980), *Decision Support Systems: Current Practice and Continuing Challenges*, Reading, MA: Addison-Wesley Pub. Co.

American Machinist, (1980), "Machine Tool Technology", *American Machinist*, Vol. 124, No. 10, Oct. 1980, pp. 105-128.

Bartlett F.C., (1932), *Remembering*, Cambridge: Cambridge University Press.

Bobrow D., and T. Winograd, (1977), "KRL: Knowledge Representation Language," *Cognitive Science*. Vol 1, No. 1, 1977.

Buneman O.P., H.L. Morgan, and M.D. Zisman, (1977), "Display Facilities for DSS Support: The DAISY Approach", *Proceedings of a Conference on Decision Support Systems*, SIGBDP, Vol. 8, No. 3, 1977.

Carbonell J., and Phil Hayes, (1981), "Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update", *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver B.C., Canada, August 1981.

CMU-CSD, (1979), "Proposal for a Joint Effort in Personal Scientific Computing", Computer Science Department, Carnegie-Mellon University, Pittsburgh PA.

Duda R.O., P.E. Hart, P. Barrett, J.G. Gaschnig, K. Konolige, R. Reboh, and J. Slocum, (1978), "Development of the Prospector Consultation System for Mineral Exploration: Final Report", Tech. Rep., SRI International, Menlo Park CA, Oct. 1978.

Erman L.D., F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, (1980), "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, June 1980, pp. 213-253.

Fox M.S., (1979a), "On Inheritance in Knowledge Representation", *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo Japan.

Fox M.S., (1979b), "Organization Structuring: Designing Large, Complex Software", Technical Report CMU-CS-79-155, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

Fox M.S., (1982), "SRL: Schema Representation Language", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Galbraith Jay, (1973), *Designing Complex Organizations*, Addison-Wesley.

Hayes-Roth F., and V. Lesser, (1977), "Focus of Attention in the HEARSAY-II Speech Understand System," *Fifth Int. Joint Conf. on Artificial Intelligence*, Cambridge, MA, Aug. 1977.

Klahr P., and W. Faught, (1980), "Knowledge-Based Simulation", *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford CA, Aug. 1980, pp. 181-183.

Lenat, D., (1976), "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," (Ph.D. Thesis) Computer Science Dept., Stanford University.

Lowerre B., (1976), The HARPY Speech Recognition System, (Ph.D. Thesis), Tech. Rep., Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

March J.G., and H.A. Simon, with H. Guetzkow, (1958), *Organizations*, New York: John Wiley and Sons.

McDermott J., (1980), "R1: an Expert in the Computer Systems Domain", *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford University, Aug. 1980, pp. 269-271.

Minsky M., (1975), "A Framework for Representing Knowledge", In *The Psychology of Computer Vision*", P. Winston (Ed.), New York: McGraw-Hill.

Pople H., (1977), The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Aug. 1977.

Rashid R.F., (1980), "An interprocess communication facility for UNIX", Technical Report CMU-CS-80-124, Computer Science Department, Carnegie-Mellon University, Pittsburgh PA, February 1980.

Reddy Y.V., and M.S. Fox, (1982), "Knowledge-Based Simulation", Technical Report, The Robotics Institute, Carnegie-Mellon University, Pittsburgh PA.

Robertson G., D. McCracken, and A. Newell, (1981), "The ZOG Approach to Man-Machine Communication", *International Journal of Man-Machine Studies*, Vol. 14, pp. 461-488.

Shortliffe E.H., (1976), *Computer-Based Medical Consultations: MYCIN*, New York: American Elsevier.

Simon H.A., (1960), *The New Science of Management Decision*, New York: Harper and Row.

Williamson O.E, (1975), *Markets and Hierarchies: A Transactional and Antitrust Analysis of the Firm*, New York NY: The Free Press.

I. System Particulars

IMS is programmed in Franz-lisp on a VAX-11/780 running the UNIX operating system. Processes communicate using the Interprocess Communication Facility (Rashid, 1980). The color display is a Grinnell system.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

TEXT PROCESSING AS A TOOL FOR DSS DESIGN

CYRIL H.P. BROOKES

Department of Information Systems
University of New South Wales
P.O. Box 1, Kensington 2033
AUSTRALIA

The hardware and software environment which is increasing the sophistication of mechanised office and text processing systems can also play a role in the design of decision support systems. It is clear that managers make considerable use of so-called 'soft information' in their work, and this is almost always encoded in textual or natural language form. The manner in which text processing technology might be used to increase effectiveness is examined. The design basis for a message system which can fulfil a DSS role by reporting text data is described.

INTRODUCTION

Text processing possibly represents one of the last "frontiers" for computer applications. The collection, storage, processing, and reporting of numeric data is so well established within computing culture that it often comes as quite a shock when a systems designer is faced with the task of performing even simple manipulations of data encoded in text. Apart from rather esoteric fields such as artificial intelligence research, most people now associate text processing with office automation (OA) type systems.

OA is usually regarded as comprising the hardware and software which perform word-processing, document distribution, electronic mail and other forms of message handling. However, it can be considered to cover also new types of hardware devices such as 'clever' copiers, i.e. printer/copier/facsimile combinations, photo-typesetters, computerised PBX, store and forward voice communication, micro-film systems, optical disc, etc. Within the past twenty-four months there has been an explosion of interest in local networks, for example the Ethernet and Cambridge Ring systems and the introduction of 'professional' terminals, for example XEROX STAR with its 'mouse' and others which have touch sensitive screens. A major function of those networks will be the assistance with office related procedures.

Most commentators don't seem to include speed-reading techniques as part of the technology though possibly this might have more impact on some manager's efficiency!

It is apparent that the major marketing thrust of OA suppliers is the increased efficiency of the operational aspects of office activity e.g. typing, filing, document editing, and electronic mail. There have been surveys of the time spent on different categories of tasks at various levels of an office hierarchy. One of these, carried out at Xerox Corporation, and published in March 1981, reported the results shown in Table 1 [1].

The figures reported by Terrie have been re-classified so as to separate out creative and knowledge related tasks (writing, reading, proofreading, meetings etc)

from operational activities.

TABLE 1

OFFICE ACTIVITY AS PERCENT OF TIME

Activity	Managers & Professionals	Other Staff	Total
Creating text:	35%	17%	20%
Reading & proof-reading:	16%	8%	10%
Interactive communications (phone, meetings, instructions):	27%	18%	20%
Sub-total	78%	43%	50%
Conventional office automation target areas (searching, filing, distributing, copying, operating equipment):	14%	34%	30%
Travel and Other:	8%	23%	20%
	100%	100%	100%

Hence, although the managerial environment is being impacted by current OA systems, the 78% of managers' and professionals' actual work activity which could be described as creative or 'knowledge-based' is largely unaffected by it. (Note: the 5% of time spent on the phone might be subject to assistance through store/forward voice and other messaging systems.)

There appears, therefore, considerable scope for study as to how the emerging OA technologies, especially text processing) can be used in a more positive (i.e. less mechanical) way to contribute to management effectiveness. Employment of these technologies as a basis for DSS design is one major possibility.

This paper firstly establishes the central role soft information does play within a decision-making environment (and therefore the importance of text processing, since soft information is usually expressed in words). Differences between num--eric and text reporting are then highlighted and finally the design criteria of a DSS oriented message system are presented.

DECISION SUPPORT SYSTEMS AND OFFICE AUTOMATION

The decision support system field is as ill-defined as OA, but the general object--ive of these systems is to improve managerial decision making (however hard that improvement may be to define) through the provision of information. Just what information, when it is provided, and in what manner is the crux of the DSS designer's problem.

The emphasis on decision support systems research has focussed on a number of fields including:

* The cognitive style of individual managers with special reference to differ-ent approaches in the information gathering and information processing tasks

* Database organisation structure and query interfaces

* Basic DSS tools, for example financial modelling, simulation, risk analysis etc

* More sophisticated DSS generators

* Specialised output techniques, for example graphics and voice.

Nearly all the above fields relate to the ways that formatted, factual and numerically encoded information can be provided to the manager. Relatively little attention has been paid to the role that information encoded in text or natural language plays within decision making activities.

Of particular importance in this field is the 'soft' component of text encoded data since a number of studies have shown that managers pay particular attention to peoples' opinions, explanations, rumour and other forms of soft or non-factual information [2].

To demonstrate this, consider the tabular output drawn from an order entry database of a rural manufacturing company as shown below.

COUNTRY ARTIFACTS LTD

SALES REPORT JUNE 1982 - $000's

	Magpie Wackers			Possum Rings		
	Actual	Budget	Last Year	Actual	Budget	Last Year
March	1185	1000	951	392	700	654
April	1191	1000	1131	385	700	581
May	1275	1100	1083	495	800	699

A reasonable conclusion to be drawn from the above figures is that the department responsible for fabricating magpie wackers could be regarded as a 'cash cow' and that the possum nose ring department is a lame, if not dead, 'duck'. Any manager asked to make a decision about priorities in advertising etc. would tend to favour the magpie wacker department. Yet it is only likely to take a single memo note from a trusted lieutenant, for example the chief accountant, to effect a change in policy. Such is the power of soft information.

> "Over lunch yesterday Bill Smith from Rural Stores Limited told me that there is a huge plague of possums out west this winter and we can expect an incredible upsurge in demand for nose rings. All our competitors have stopped making the device, apparently because of poor sales, and we ought to be able to make a killing."

In two previous articles [3,4] dealing with this subject, I have nominated two main sets of findings as follows:

(a) The reporting, for DSS purposes, of soft information contained within text will be significantly different from reports prepared from a base of factual numeric information.

Looking firstly at factual data, the following points are self evident:

(i) It is normally stored and reported in a highly structured form. Links between records usually follow a pre-determined schema and will be created automatically. Access to specific types of data is therefore easy.

(ii) Its existence tends to be well publicised and notice may even be included in a formal data dictionary.

(iii) The designers of systems which report factual information generally assume that the identities of sources are of no consequence, hence they are normally not recorded or reported.

(iv) The meaning of the data stored is also assumed by systems designers as being self-evident. In addition, it is not anticipated that there will be any ambiguity.

(v) Users of the reports assume they are based on accurate data.

(vi) Security and access control are governed by universal rules, usually based on the level of a would-be user within the organisation or membership of a particular unit.

(vii) The life of factual information is generally considered to be months or years, thus justifying the creation of archiving systems etc.

Referring to soft data and the problems of reporting information derived from it to a decision-maker, the following assumptions appear valid:

(i) Soft data is usually stored in an unstructured format, apart from any structure imposed by the constraints of natural language. Cross-reference links may need to be created or destroyed as a result of author or user directions since the system cannot easily use a schema or code to establish inter-relationships automatically.

(ii) The existence of any piece of soft data is not usually well known. Thus there is a problem for a source of this type of information because it is frequently difficult to determine who in the organisation should be told. Conversely, a person requiring soft information often has an incomplete idea as to who in the organisation might be an appropriate source.

(iii) The meaning and implications of the soft data stored in a database will often not be self-evident from the text as created by the author, and often some ambiguity will be perceived.

(iv) The user will often be concerned about the accuracy of the soft information, and any judgment on accuracy is likely to hinge very much on the identity of the source.

(v) Usually the author will wish to exert some control over the use of the data and its further dissemination and, consequently, the extent of co-operation will depend upon the identity of the user.

(vi) The value of the soft information may only be significant for a matter of minutes or hours - e.g. a horse-racing tip.

Reviewing these differences between the two reporting situations it becomes clear that, in the case of factual data, it is valid for attention to be focussed on timeliness, ease-of-use and presentation. On the other hand with soft information the main priority ought to be the need for rapid 'match-making' - that is the bringing together of authors and users. This will facilitate communication of the meaning of the text, the user will be able to validate its accuracy, and the author will be able to monitor security.

(b) A study of the interaction between the mental processes of decision makers and their external environment has led to the conclusion that soft information can be classified into one of five categories. These are based on the treatment given the subject matter, rather than the identity of the subject itself. It is proposed that the use of this, or a similar, classification

can form a useful basis for improving the effectiveness of text database searching using keywords. The five categories are as follows:

(i) Qualification soft information adds meaning to the numeric data which describes real events, processes, entities or relationships. Thus the memorandum from the chief accountant mentioned earlier qualifies the sales figures and adds meaning to these figures.

(ii) Structural soft information gives details about potential changes in the relationships between important variables within the decision maker's sphere of influence. For example, a perceived potential change in the consumer response rate to an advertising program will influence decisions in the assignment of funds to different promotional exercises.

(iii) Uncertainty soft information informs the manager about potential variability of critical external parameters - particularly inputs to the organisation (e.g. customer ordering profiles, the security of supply of raw materials etc) or external control variables (e.g. inflation rates, overall corporate policy on inventory etc).

(iv) Strategy soft information will affect the manner in which a decision maker formulates the objectives which govern decisions in the area he controls. For example the opinions of peers will have a big impact in determining how a funds manager directs the current investment program.

(v) Reference soft information relates to the identity, potential accuracy and knowledge range of extra sources of information, both factual and soft.

DESIGN PRINCIPLES FOR A CORPORATE INTELLIGENCE SYSTEM

The design of a system which facilitates the exchange of soft information within an organisation would involve the incorporation of the reporting principles described earlier within a hardware/software environment which provided for the capture, storage, retrieval and transmission of text based data. In effect, we are examining the ways that OA or text processing technology can provide the "vehicle" for implementation of decision support systems. Hence, text processing must be a basic tool for DSS systems design development where soft data is to be processed. A prototype local network system to provide this type of decision support is being implemented at the University of New South Wales. Apart from conventional factors such as security, reliability, etc, the design features of this system include:

(a) Objective: The system is being designed to serve executives in a manufacturing/competitive marketing and distribution organistion. The Corporate Intelligence System (or CIS) is directed towards the rapid communication of soft information about customers, products, services, government policy, market trends etc, between persons who gain access to this intelligence and those who are in a position to best make use of it.

(b) Personal Interest Profile: A key part of the system is the creation of a set of interest profiles for each person who is a user of the system, in a manner similar to that adopted within some computer conferencing and library distribution systems [5]. These profiles form the basis of the content addressing and match-making procedures. The initial forms are keywords covering customer names, product groups, competitor names etc.

(c) Local Text Database: This database is the main storage location for soft data entered by users of the system. A mixture of fixed and free formats are employed, the fixed part of each record containing "standard" information including author, source identity, date and time received, life-span, and soft information classification, with space for cross-references to related

data. The soft data is then expressed in text in the free format section. the free format area may simply contain keywords appropriate to the subject matter of the intelligence.

(d) Interface with other Text Databases: Provision is being made for an interface with other text databases created as a result of word processing, document distribution and electronic mail facilities. This is being provided for two reasons:

1. A "scan" of these documents may produce relevant intelligence, and

2. It allows the CIS terminals to be used in a "conventional" word processing, electronic mail manner via the local network and its gateways.

(e) Interface with Conventional Data-processing Databases: The CIS local network is being set up permitting database inquiry on the organisation's EDP files. In particular, this will be necessary to permit access to numeric data items cross-referenced with qualification type soft information entries.

(f) The User Interface: The system is being designed to permit the following kinds of CIS type user access:

1. To support a potential receiver of intelligence the system will have these functions:

(i) Scan the local and remote database seeking relevant entries using nominated keywords and other information such as author, source, date range, soft information classification etc.

(ii) Scan the interest profiles of the system's users to determine those who are interested in the topic of current interest and, therefore, to identify persons who may be potential sources.

(iii) Set up an interest monitor which will cause the system to examine messages entering the database to determine if they satisfy search criteria.

(iv) Permit the receiver to follow cross-referencing chains which link related records (some of which may be comments on the accuracy or meaning of other records).

2. To support authors of information the system will provide these functions:

(i) Allow the author to enter a message with relative ease. In this mode of operation the author will be relying on the content addressing capability of the system to determine potential receivers. The message may comprise either text which expresses (in the opinion of the author, anyway) the intelligence or it may comprise only a set of keywords. In the latter case, the receiver will need to contact the source to obtain the details.

(ii) Allow comments to be entered which are linked automatically with related messages and referred to the original author, and other commentators.

(iii) Seek the identities of persons likely to be interested in the intelligence by an examination of the interest profiles; the author would then contact some or all of these people directly.

(iv) Establish new cross-reference links or comment on the validity of existing links. This may follow a review of comments made by others

on data previously entered.

(g) Searching the CIS Text: Two basic techniques for text searching using keywords are available to the system designer:

1. Full or partial indexation
2. Content Addressing (or associative memory).

Effectively, indexing requires that effort be put into the "front end" of processing in return for reducing the effort at enquiry time, and content addressing shifts almost all processing to the enquiry phase.

Because the CIS local text database is likely to be relatively small - 30 million characters is assumed an upper limit in this exercise - content addressing is being used since it is easier to implement. The initial form of searching used is based on the "database stack" concept [6]. However, full indexation would provide exactly the same functionality in practice.

Of more importance in this context is the real utility of the keyword style of searching. Researchers in the linguistic AI field have long recognised the deficiencies that keyword type searching have in an application such as this. The enquirer has a "concept" in mind and often finds it difficult to embody the meaning in keywords. It then transpires that authors in different environments use different words [7].

Our preliminary experience has shown that a restricted user group, keen to use a system such as this, will tend to use particular words to describe particular situations. Hence keyword searching is a reasonable basis for the prototype.

Greater difficulties exist when it comes to searching larger text databases - such as will be found in conventional word processing/electronic mail environments. Here automatic indexation [8] or high-speed associative memory devices become essential components of the searching system.

(h) DSS capability: The character which distinguishes this design approach from other "message" systems is the underlying "DSS" objective. It is an attempt to bring text data within the ambit of information resource management principles usually confined to numeric data. The differences between reporting the two types of data have already been stated, but the creation of a DSS environment for this reporting requires further consideration.

The DSS objective means that assistance should be provided in these functional areas:

1) identifying potential problems
2) understanding the problem area
3) assessing the implications of alternative options for action
4) monitoring the actual consequences of earlier decisions.

In turn these functions devolve to a process of either:

(i) Attenuating a large amount of data so as to highlight for the decision-maker situations which are unexpected, i.e. 1) and 4) above, or

(ii) Amplifying the information supply so as to assist with 2) and 3).

To provide these functions from a text database is the key objective.

Progress to date has shown that managers want from this system mainly attenuation type information. Interviews have shown that once a problem is ident-

ified the existing set of DSS aids (manual and computer based) are perceived adequate. Hence our initial design focus is on determining how best to communicate text soft information which:

(i) Summarises or assesses a situation
(ii) Alerts to significant events
(iii)Compares current status with past experience in important areas
iv) Monitors previous problem situations.

As yet, an AI approach to assessing meaning or significance is not being followed, rather an internal set of "operating guidelines" is being postulated which will allow the author or a commentator to indicate importance.

CONCLUSION

The application of office automation technology within a decision support environment needs special care on the part of the designer because of the nature of soft information reporting.

Contrary to the factual reporting situation, assumptions cannot be made about adequate accuracy, self-evident meaning and lengthy life-span of data entries. It therefore becomes important to bring together the sources and receivers of information so that they can establish effective communication on a person-to-person basis. It follows that the major function of the system is to facilitate the matching of these people and to assist the communication process. Design of the match-making function is itself a complex task. To introduce a DSS quality to text reporting it is necessary to focus on both the "attenuation" of the data flow, seeking messages likely to identify problem situations, and the "amplification" of available detail about the problem and options for action. Attenuation reporting is deemed to be of most value in this context.

REFERENCES

[1] Terrie, D., (ed.), Office Automation Reporting Service, Internationa Data Corporation, Waltham, Ma. U.S.A. (March 1981)

[2] Mintzberg, H., The Manager's Job: Folklore and Fact", Harvard Business Review (July/August 1975).

[3] Brookes, C.H.P., Incorporating Text Based Information Within a DSS, DSS-81 Transactions, Proceedings First International Conference on DSS, Atlanta (June 1981). Execucom Systems Corporation, Austin, Texas.

[4] Brookes, C.H.P., Effective Soft Information Communication - A Key to Decision Support, Information Systems Forum report, School of Accountancy, University of New South Wales (1981).

[5] Hiltz, S.R. & Turoff, M., The Evolution of User Behaviour in a Computerised Conferencing System, Comm. A.C.M., (Nov. 1981).

[6] Brookes, C.H.P., A Database Structure for Natural Language Data Processing, Proc. Eighth Australian Computer Conference, Canberra (August 1978). Australian Computer Society.

[7] Smith, L.C., Representation Issues in Information Retrieval Systems Design, Proc. 4th International Conference on Information Storage and Retrieval, Oakland, Ca. U.S.A. ACM (1981).

[8] Salton, G., A Comparison of Search Term Weighting: Term Relevance v's Inverse Document Frequency, Proc. Fourth International Conference on Information Storage and Retrieval, Oakland, Ca. U.S.A. ACM (1981).

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

PRACTICAL EXPERIENCES WITH THE PROCEDURAL DECISION MODELING SYSTEM

R. Maes
University of Amsterdam, The Netherlands

J. Vanthienen
&
M. Verhelst
Catholic University of Leuven, Belgium

The PROcedural DEcision MOdeling (PRODEMO) system was developed to enhance the application of decision tables as a general management technique. This shift of focus with relation to the usefulness of decision tables is briefly indicated in the paper, after which the proper PRODEMO system is presented. Then attention is given to practical experiences with the PRODEMO system in multiple organizations. To this end, three typical case studies are reviewed and empirical results obtained from many other cases are reported on.

1. INTRODUCTION

Originally, decision tables were developed as handy tools for computer programmers, who were confronted with logically complex representation problems. Despite this very practical issue, a great part of the 'classical' literature on decision tables is devoted to the single problem of optimally converting decision tables into programs, whereas their practical applicability is greatly neglected. This distortion was accused for the first time in [1].

Looking for the reasons of this anomaly, we struck upon the following:

1. The applicability of decision tables as a *programming* technique is indeed rather limited, as was investigated in [3].
2. *Constructing* decision tables starting from conventional texts and/or human reasoning processes is the real key problem. Only by the development of adequate methods for constructing decision tables (see [8], chapter 2), the introduction of the computer into this modeling process became possible.

With regard to the first cause, the conviction that decision tables should be used outside the proper programming sphere, is more and more gaining ground. In [7], it was shown that decision tables are useful as a general management technique. Their potential value during the preliminary, users-oriented phases of the information systems life cycle is demonstrated in [3] (chapter 4).

The aim of this paper is threefold:

1. To demonstrate by real-life examples, the usefulness of decision tables as a decision support technique. (Sections 2 and 4).
2. To present the PRODEMO system (Section 3). This system enables the computer supported construction and manipulation of decision tables and in this way really is a significant step towards a non-trivial DSS generator.
3. To report empirical results obtained from a great lot of field research on the applicability of the PRODEMO system (Section 5).

2. DECISION TABLES AND PROCEDURAL DECISION MAKING

2.1. Procedural decision making

A large part of decisions to be made in organizations are of a procedural nature. They require the application of rules, regulations, laws, policies, etc.. Very often, such rules are of the following type: 'If condition 1 and condition 2 and ..... and condition n are met, consequence 1 and ..... and consequence m applies'. Even when decisions are made on an intuitive basis, it can frequently be shown that strict procedures are followed, albeit unconsciously.

If procedural decisions should be made by other persons than the ones who prescribed the underlying procedures, these procedures must be documented. This is usually done by means of narrative. We will call this: *a-priori structured* procedural decision situations.

If decisions are made by the same person who made the rules, then documentation may or may not exist. If it does not exist, the rules (conscious or unconscious) are in the head of the decision-maker. This latter case will be called: *a-priori unstructured* decision situations.

In the 'a-priori structured' case, several problems appear:

(1) it takes time for the decision-maker to study the procedures before being able to make the decision or to draw the conclusion;
(2) very often, the text is unclear and can be misunderstood, leading to wrong decisions or conclusions;
(3) very often, the text is not exhaustive (i.e. does not treat all possible cases) and is hiddenly contradictory in many places, which again entails poor decision making.

Problem (1) is inherent to the fact that natural language is used; problems (2) and (3) can be avoided if the person who made the procedures is an extremely intelligent one (but only few procedures are made by persons of that type). It follows that in actual practice all three problems are frequently experienced, leading to poor decision making. Moreover, they grow very fast as a function of the complexity of the decision situation. They can all be avoided by using decision tables instead of narrative as a means of documenting the decision. Decision tables can also be very useful in the a-priori unstructured case. They can become a vehicle for modeling the decision process such that the process gains in objectivity, logical consistency and efficiency (for an example, see subsection 4.1).

2.2. Decision tables

For clarifying the power of decision tables in the context of procedural decision situations, let us consider the case of a-priori structured decisions, i.e. procedural decisions documented by means of natural language.
Standard methods exist ([8]) for translating narrative to decision tables. They will be presented briefly in subsection 2.4.

We will now illustrate how decision tables can dispel the above-mentioned problems by comparing a typical narrative (figure 1) with the decision table derived from it in such a standard way (figure 2).

ORDER HANDLING

1. Discount

   Only wholesalers are granted discount, provided that they order a quantity of at least 10 units. The discount rates are 10 %, 5 % and 2 %: 10 % for wholesalers ordering at least 15 units, or living at a distance of less than 50 km and ordering at least 10 units; 5 % for wholesalers ordering at least 10 but less than 15 units and living at a distance of at least 50 km but less than 100 km; 2 % for wholesalers ordering at least 10 but less than 15 units and living at a distance of at least 100 km.

2. Way of transportation

   We transport by railway if the order is not from a wholesaler or if a wholesaler orders at least 15 units. In all other cases transportation is by road.

3. Type of invoice

   The normal type is A. Exceptionally, an invoice type B should be made, viz. for a wholesaler who orders at least 15 units.

Figure 1

ORDER HANDLING	R1	R2	R3	R4	R5	R6
C1. Customer = wholesaler	Yes	Yes	Yes	Yes	Yes	No
C2. Quantity ordered (Q)	Q<10	10≤Q<15	10≤Q<15	10≤Q<15	Q≥15	-
C3. Distance between warehouse and place of delivery (D)	-	D<50	50≤D<100	D≥100	-	-
A1. Discount in %	0	10	5	2	10	0
A2. Transportation by railway	-	-	-	-	X	X
A3. Road transportation	X	X	X	X	-	-
A4. Type of invoice	A	A	A	A	B	A

Figure 2

Let us now review the contribution of decision tables for solving the three problems mentioned above.

Problem 1: If decisions are made using the decision table, a maximum of 3 questions must be answered, sometimes only 2 questions (in case R1 or R5 applies) and sometimes only 1 question (in case R6 applies). If all rules occur equally frequently, this makes an average of 2.33 questions. With the text of figure 1, an average of 7.66 questions is needed. Therefore with the decision table, decision speed is increased by a factor of 3.3. The underlying reason is that, by their nature, texts are action-oriented and decision tables are condition-oriented.

Problem 2: Making use of the decision table, fewer mistakes will be made because (1) fewer questions must be answered and therefore the probability of wrong answers is lower and (2) the table does not contains words or

expressions which are easily and unconsciously misunderstood. Examples of such words appearing in the text are: 'only', 'provided that', 'or', 'in all other cases', 'the normal type', 'exceptionally', 'viz', 'but'.

Problem 3: The text of figure 1 is complete and does not contain any contradictions. We will now show how decision tables can be used for finding omissions and contradictions. Suppose that the first paragraph of the text of figure 1 were:

1. Discount

Only wholesalers are granted discount.
The discount rates are 10 %, 5 % and 2 %: 10 % for wholesalers ordering at least 15 units, or living at a distance of less than 50 km and ordering at least 10 units; 5 % for wholesalers ordering at least 10 but less than 15 units and living at a distance of at least 50 km; 2 % for wholesalers ordering at least 10 but less than 15 units and living at a distance of at least 100 km.

Applying a standard method for translating the new text into a decision table with the purpose of finding omissions and contradictions, we get the table of figure 3.
By inspecting the action part of the table, we see that for R1 no discount appears (not even 0 %) which is an omission. Furthermore R4 shows a contradiction: discount = 2 % and discount = 5 %. The advantage of the decision table here is, that such omissions and contradictions are made visible, whereas in the text they remain hidden.

ORDER HANDLING	R1	R2	R3	R4	R5	R6
C1. Customer = wholesaler	Y	Y	Y	Y	Y	N
C2. Quantity ordered (Q)	Q<10	10≤Q<15	10≤Q<15	10≤Q<15	Q≥15	-
C3. Distance between warehouse and place of delivery (D)	-	D<50	50≤D<100	D≥100	-	-
A.1.1. Discount = 0 %	-	-	-	-	-	X
A.1.2. Discount = 2 %	-	-	-	X	-	-
A.1.3. Discount = 5 %	-	-	X	X	-	-
A.1.4. Discount = 10 %	-	X	-	-	X	-
A.2.1. Transportation by railway	-	-	-	-	X	X
A.2.2. Road transportation	X	X	X	X	-	-
A.3.1. Type of invoice=A	X	X	X	X	-	X
A.3.2. Type of invoice=B	-	-	-	-	X	-

Figure 3

2.3. Audience

A very often heard remark concerning procedural decisions is that this category of (repetitive) decisions is located at the lowest level of the organizational hierarchy. One of the basic themes of this paper, however, is that, by properly applying the decision table technique (e.g. by using the PRODEMO system), one

should be able also to "structure" a whole range of decisions traditionally dealt with by tactical management. In this respect, the decision table can be considered as a real structuring tool. Some striking examples will be given in Section 4.

In addition, anyone who is confronted with prescriptions, laws, procedures, ... can (irrespective of any organizational context) benefit from the technique proposed in this paper. The following schematic classification of the audience we were working with in the past may be clarifying in this respect:

- Managers who want to design new procedures or to analyze and correct existing ones used in their organization (or even by themselves).
- Lawyers who are confronted with logically chaotic and even incomplete and/or contradictory laws.
- Legislators who have to design new laws adapted to complex modern life.
- Anyone involved in the design of new regulations and prescriptions in general.
- Systems engineers who have to grasp complex organizational and infological problems.
- Teachers looking for a clear and unambiguous tool for representing complex material.

2.4. The crucial problem: how to construct effective decision tables ?

The crucial problem, commonly neglected in the traditional literature on decision tables, is how to construct tables given a more or less explicit narrative or written decision description. In our opinion this common lack of knowledge is also the main reason why decision tables failed to get accepted as a *practical* technique.

As far as the systematic construction of decision tables is concerned, experience has shown that different methods are needed for the 'a-priori structured' case and for the 'a-priori unstructured' case:

a. The conversion of an a-priori structured decision into decision tables (in order to examine the correctness of the existing representation) involves the application of so-called 'direct' construction methods:
   - if the starting description is rather simple, one can immediately enter the appropriate entries (including don't care's) in the decision table, using any variant of the progressive rule development method (see, e.g. [2]).
   - in any other case, a two steps method can be applied:
     1. the original text is translated into equivalent logical expressions (originally proposed in [6]).
     2. these expressions are used to fill in the action entries of the completely expanded decision table.

b. When dealing with a-priori unstructured situations, decision tables can very well be used as structuring tools. In this case, so-called search methods for constructing decision tables have to be applied (see [8]).

All these methods will be illustrated in Section 3, when dealing with the interactive PRODEMO system for constructing and manipulating decision tables. It will be demonstrated that this system enables us to use the different methods in any arbitrary combination. This feature has proven to be very useful.

3. THE PRODEMO SYSTEM

The PRODEMO (PROcedural DEcision MOdeling) system is a computer program for constructing and subsequently using decision tables. In the following, we successively deal with:

- the reasons for computer introduction into the decision table manipulation process (3.1)
- the philosophy behind the PRODEMO system (3.2)

- the PRODEMO modeling methodology (3.3)
- PRODEMO and decision *making* (3.4)
- PRODEMO and the application of decision table *structures* (3.5)

Finally, in 3.6, some considerations are given on the actual implementation.

3.1. A rationale for computer introduction

In Section 2, straightforward manual procedures for constructing decision tables were mentioned. However, in a high number of cases, the complexity becomes overwhelming; then, introducing the computer into the construction process is the obvious means to enlarge the applicability of the decision table technique. The following reasons can be put forward:

1. Combining different construction methods is hardly possible without the assistance of the computer. Besides, an interactive computer program, like the one presented in this paper, can give valuable indications about the desirable method.
2. The automatic generation of condition entries guarantees the completeness by enumerating all possible combinations of condition ranges.
3. A lot of administrative and clerical work that inherently accompanies the use of the decision table technique can be taken over by the computer.
4. Some manipulations of the resulting decision table can very easily be automated.
   Ex.: - the contraction of the decision table using various criteria;
   - reordering the conditions and actions.
5. Some other manipulations lend themselves very well to the use of interactive problems solving techniques.
   Ex.: splitting up a decision table.
6. The resulting decision tables can, in a further stage, form the basis of computer based decision making (see below).

3.2. The PRODEMO philosophy

As was outlined, the main purpose of the PRODEMO program is to guide and support the user during decision modeling as well as during decision making by giving suggestions and feedback, by checking for incompleteness and inconsistencies and by executing all of the administrative routine tasks and the combersome drawings.

No special knowledge is required in order to use the PRODEMO system. The interactive environment, in which it has been conceived, guarantees an extended and effective user support when this is required and is able to create a highly flexible and well controlled use of the system.

The PRODEMO system is able to operate in one of two modes:

- Command mode : the user takes control of the program and determines his path through the modeling process.
  This is achieved by grouping all important functions on a central index page: the PRODEMO menu (see figure 4 below). From this page the user can choose which option he wants to take, execute it and then return to the menu to choose another option. One of the options is to load previously constructed decision tables in memory and to use these tables.
- Response mode : the computer controls the program and leads the user through the modeling process.
  The construction of the decision table is accomplished through suggestions of the PRODEMO software, after deducing underlying rules and constraints.

3.3. PRODEMO modeling methodology

Since any procedural decision is translated into its equivalent decision table, PRODEMO in order to start the modeling process needs at least:

- a table name
- some prevailing conditions and their states

- some actions
- some relations between conditions and actions (in the form of logical expressions).

The supply of all these elements is grouped around the MENU-page (figure 4). From this page the decision description is gradually built up, while fully exploiting the advantages of the decision table scheme.

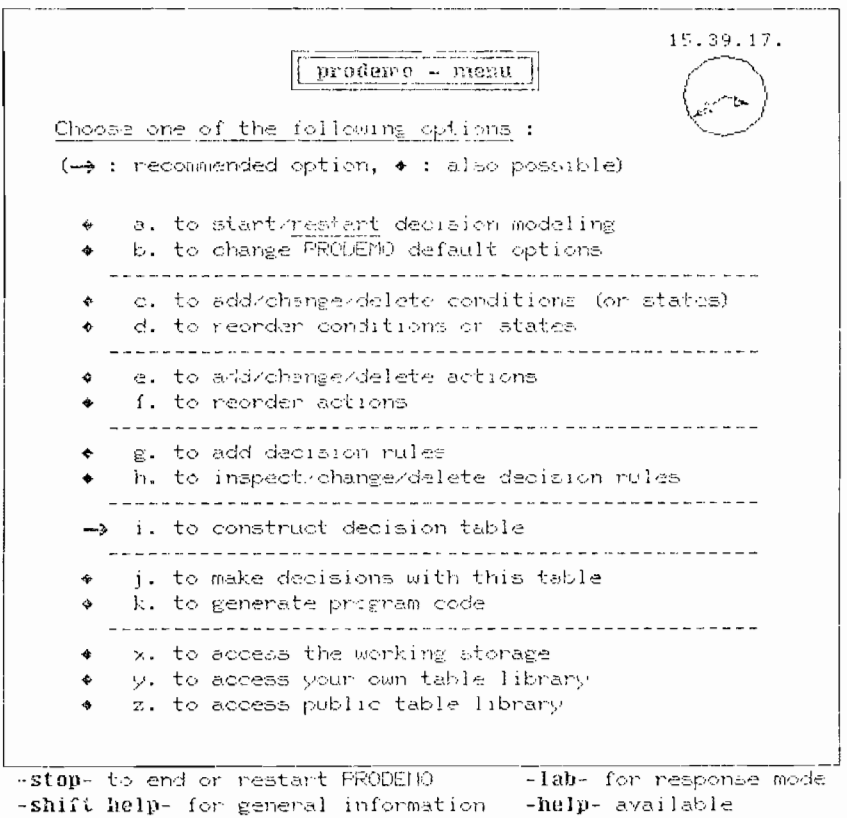


Figure 4

The general methodology to be followed by the PRODEMO user is very straightforward: at any given moment he enters all the information he has about the decision; the most obvious way to do that is by entering all conditions (with their states) and all actions he has in mind (see, e.g. [7]). Then he can enter the decision logic. Using PRODEMO, this can be done from the Decision Input page (see figure 5).

On this page, sequence numbers are used for referring to conditions and actions

and letters are used for referring to condition states. The decision logic is expressed by relating condition states and actions by means of a straightforward syntax. Suppose e.g. that the following sentence is part of the order processing example treated in figure 5: 'An order should be put on a waiting list if stock is not sufficient and if either the credit limit is not exceeded or if the credit limit is exceeded and the customer is important'.
This sentence results in the following logical expression (cfr. fig. 5):

3 ← 2b and (1b or (1a and 3a))

This means that action number 3 ('Put order on waiting list') should be executed each time that condition 2 ('Sufficient stock ?') has its second value ('No') and either condition 1 ('Credit limit exceeded ?') is 'No' (1b) or condition 1 is 'Yes' (1a) and condition 3 is 'Yes' (3a).
Notice also that the number of mutually exclusive states a condition can have is not limited to two. If more than two states exist, they are indicated by a, b, c, d, etc. ...

At any chosen time, the user can ask the PRODEMO system to construct and display the decision table (option i in figure 4).

Constructing a table implies matching the logical expressions to an expanded table, contracting the table, checking for errors and displaying the table on the screen.

When contracting a table, the user can choose between:

- a contraction with the given condition order;
- a contraction with optimal condition order.

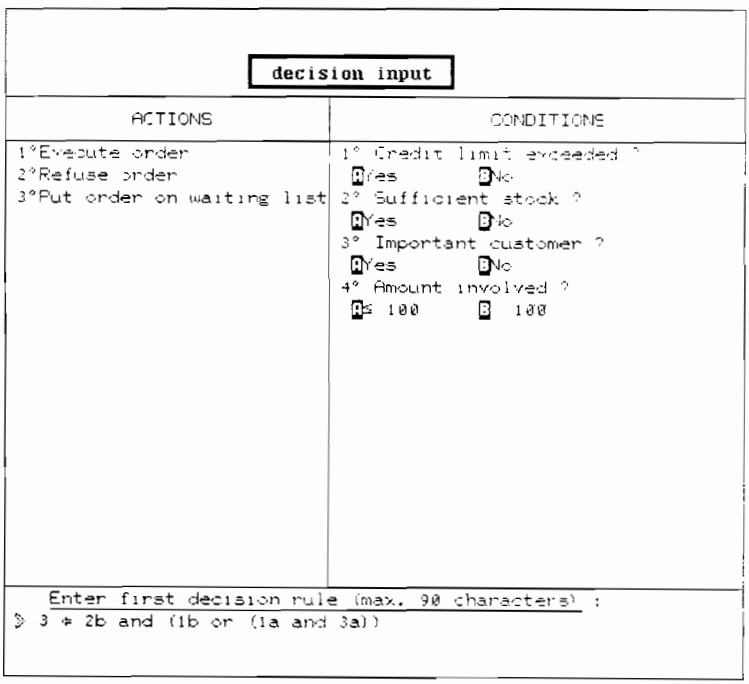


Figure 5

The latter option minimizes the table length (number of columns) by reaching an optimal condition order, which improves both the efficiency of automatic decision making and the clarity and ease of use by a human decision maker. The condition order can also be subjected to precedence constraints.

Suppose now that, at a certain moment, the developing decision table looks as in figure 6. A simple table diagnosis (provided by PRODEMO) reveals that column 3 contains contradictory actions ('Execute order' and 'Refuse order') and that column 5 has no executable action. Now the user can either

- switch to touch submode and adapt the table

  or

- switch to response mode and ask PRODEMO what to do next

(this latter option is more convenient for elaborate tables involving lots of conditions).

ORDER TREATMENT

Credit limit exceeded ?	Yes						No	
Sufficient stock ?	Yes			No			Yes	No
Important customer ?	Yes	No		Yes	No		-	-
Amount involved ?	-	≤ 100	> 100	-	≤ 100	> 100	-	-
Execute order	x	x	x	-	-	-	x	-
Refuse order	-	-	x	-	-	x	-	-
Put order on waiting list	-	-	-	x	-	-	-	x

Figure 6

Suppose the user wants to add that in the case of column 5 the order should be put on a waiting list. He enters the search submode by pressing a function key and adds the new action entry by simply touching the screen. By inspecting the table, he detects that, in the case of the credit limit being exceeded, the order should be refused when the customer is not important and the amount involved exceeds 100 (columns 3 and 6). He can then correct column 3 either by changing the appropriate logical expressions (cf. supra) or he can immediately adjust the decision table via the touch submode. The end result of this very simple modeling exercise is shown in figure 7 (screen output).

The rest of the PRODEMO menu-page (figure 4) is rather self-explanatory. However, the following conventions/and or comments should be kept in mind:

- the user can impose and change the order in which the conditions, condition states and actions should appear in the table (option d and f in figure 4).
- once a table has been constructed, it is possible to make decisions on an interactive basis (option j in figure 4). This option is further treated in section 3.4.
- although the PRODEMO-system was designed as a modeling tool for the interactive construction of decision tables, it can also be used to translate the resulting decision table(s) in executable source code (option k in figure 4). The resulting code is a non-optimized straightforward translation of the decision table.
- in order to save or reload a decision table or a system of decision tables, one can use a working storage, with rather limited protecting mechanisms (option x),

or have access to the general public library (option z). The latter option is more complex and should only be used for (almost) finished decision descriptions. Any user can also create his fully protected private library (option y).

ORDER TREATMENT

Credit limit exceeded ?	Yes					No	
Sufficient stock ?	Yes			No		Yes	No
Important customer ?	Yes	No		Yes	No	-	-
Amount involved ?	-	≤ 100	> 100	-	-	-	-
Execute order	X	X	-	-	-	X	-
Refuse order	-	-	X	-	X	-	-
Put order on waiting list	-	-	-	X	-	-	X

Figure 7

3.4. Making decisions with the use of PRODEMO

Making decisions is nothing but an option to be taken at the PRODEMO menu page (see figure 4). Actually, this can be accomplished in one of the following two ways:

a. one can display the "active" decision table and simply use it as a versatile and very compact directory for procedural decision making
b. one can ask PRODEMO to "interpret" the decision table(s) (option j in figure 4). In this case, PRODEMO goes through the contracted decision table(s), while confronting the decision maker with the successive relevant condition tests. Practical experience has shown that this "interpretive" mode leads to very fast decision making due to the fact that all redundant information is disregarded and all irrelevant condition tests are avoided.

Moreover, the PRODEMO system automatically links interrelated decision tables when making decisions (see below).

3.5. Decision table structures

PRODEMO is able to deal with structures of decision tables.

A table structure is a collection of interrelated decision tables, concerning the same decision situation. The relations are formed by the fact that some tables are a further elaboration of a condition or of an action of another table.

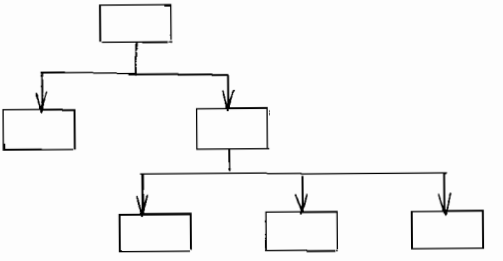
In most cases, all tables within a system can be combined into one single table with the same logic. Such a table, however, would be so large that it would be completely useless. Moreover, the construction of small and related tables, containing coherent decision information, offers some important advantages. It enables the designer to focus on only the relevant aspects of the decision situation part by part and to keep the decision structure in mind. It therefore adds to the modularity and the top-down approach of the problem description.

The relation between tables in a system can take two forms, because of the distinction between action and condition subtables. An action subtable is a table which refines an action of another table. A condition subtable refines a condition of another table, by indicating which condition states are satisfied in the various decision situations.

Only tree structures are allowed. Tree structures consist of one head table with underlying levels of subtables, so that each subtable has one and only one higher

(parent) table. A table can refer to various (lower level) subtables, but it can only be referred itself by one (higher level) table.

The structure of a system might e.g. look like this:



PRODEMO deals with these system relations.
During the decision making process, e.g., the successive condition tests of all relevant tables are automatically presented to the decision maker in their correct order and, depending upon his answers, the actions to be executed are displayed.

3.6. Some notes on the actual implementation

PRODEMO was first implemented on the CDC Plato system. This rather unusual environment was chosen by the fact that

- PRODEMO was created in corporation with the Management and Development Training Department of a Belgian bank. There, the Plato system was extensively used as a training tool.
- Plato had wide facilities for creating visually attractive displays and a touch-sensitive screen which is useful for manipulating decision tables.
- Processing speed and response time had proven very satisfactory even for performing lengthy and complex decision table manipulations.

4. CASE STUDIES

In the following, three real-life case studies are presented, two of which start from written documents (so-called 'a priori structured' situations) whereas the first case study typically deals with an 'a priori unstructured' situation

4.1. Case study 1: Credit granting in a bank

The subject of this analysis was the procedure concerning the approval of cash credit requests. Because of its highly risky nature, this credit judgement has traditionally been dealt with by complex and casual decision procedures. The main concern of the study therefore was

(a) to standardize the procedure as far as possible, and
(b) to create more objective decision criteria, which could also lead to a (partial) computer support.

Unlike the studies of the following paragraphs this analysis is based upon an a priori unstructured problem situation, i.e. the procedure did not confine itself to strict and written operating rules.
Therefore it was necessary to take the following steps, when performing the analysis:

(a) Interviews combined with document analysis, enabled the detection and separation of various judgement criteria. This lead to a mathematical formulation or to a decision table explicitation of each criterion. In the decision tables, the criteria were detailed into measurable or qualitative components (conditions) applying the 'search method' (cfr. [8], § 2.3).
A rough subdivision of these qualitative components into positive (+) average (±) and negative (-) seemed sufficient at first glance.

(b) Due to the absence of written operating rules, the resulting decision tables had to be discussed in detail with the responsable management representative.

(c) Only at this stage the modified decision tables were combined into a hierarchical structure of tables. The cooperation of the related department and the growing insight concerning the structure of the problem have proven very helpful in this respect. (1)

(d) Approval by higher management.

(e) The global and rough value distinctions of the qualitative criteria were translated into operational terms, which together with the measurable criteria added up to a list of 19 distinct elements to be filled in by the bank's credit agent(s) when handling a customer' request.

Though the study was meant to reach an end here, the following, more speculative, extensions have been added:

(f) Based upon the obtained criteria and their relative weightings (implicitly enclosed in the decision tables), tentative 'scores' have been constructed. These can lead, after a precise mutual fine-tuning to the (semi-) automated processing of a number of routine requests.

The contribution of the decision table technique to this study can be summarized as follows:

1. The syntactical scheme of the decision table proved to be an extraordinary tool for detecting all relevant criteria (conditions) of the credit granting procedure.
2. Structuring and relating the various criteria has shown very confusing without the aid of decision tables; these latter were advantageously used for structuring the processes of thought.
3. The resulting decision tables need not be considered as the final step. A more elaborate scoring and weighting could emanate from the obtained decision table structure.

Specific problems in this project concerned the degree of complexity and especially the degree of subjectivity of the problem situation. Constructing an evaluation scale for every criterion, e.g., was a tough and indistinct operation. Other, more general problem areas are indicated in Section 5.

4.2. Case study 2: Technical specifications in a manufacturing firm

A metal construction firm, which produces and assembles various types of pumps, is confronted with a detailed description of specific pump requirements (i.e. the so called API norms).

For every requirement, concerning e.g. material, size, construction, tests, delivery, ..., it is indicated when and for which pump type it applies. In order to satisfy all requirements, the entire standard has to be examined for every pump construction.

Like most procedures and regulations, this kind of standard enumeration is strictly action-oriented, i.e. for every action (requirement) a description is given of when and where it applies. The actual decision process, the detection of the

relevant requirements, is essentially condition-oriented (pump type, operation specifications, ...). 'Translation'of the standards into the condition-oriented decision table format seemed therefore appropriate.

The use of the decision table technique served a double objective:

1. Checking the original text for contradictions and shortcomings, which can be performed very easily due to the specific decision table structure (this is also an automatic function of the PRODEMO system).
2. Simplifying the decision making process, in order to determine the required specifications in a fast and correct way.

In a few days and by extensively using the PRODEMO facilities, the entire document has been rewritten in a number of simple decision tables, an example of which can be found in figure 8.

pressure casing 02/20/81

pumping temperature	<401°F								≥401°F	
thermal shock probable	Y		N						-	
flammable or toxic liquid	-		Y					N	-	
specific gravity at pumping temperature	-		<0.7		≥0.7			-	-	
rated discharge pressure	-		-		≤1000 psig		>1000 psig	-	-	
approval of purchaser for axially split case pumps	Y	N	Y	N	-	Y	N	-	Y	N
radially split case pumps required [2.2.1.]	x	x	x	x	-	x	x	-	x	x
axially split case pumps may be furnished [2.2.1]	x	-	x	-	-	x	-	-	x	-
no special recommendations	-	-	-	-	x	-	-	x	-	-

Figure 8

As most of the standards deal with only a limited number of pump type conditions, there is no need to examine the complete set of requirements everytime a decision has to be made. Pointing out the required specifications can be performed in a straightforward manner, with a minimum of (mis)interpretations and a fair guarantee of correctness and completeness.

4.3. Case study 3: Analysis of a Collective Agreement

The automation of payroll in a large manufacturing firm was troubled by
(a) the complex nature of the collective agreement with the unions
(b) the casual, often unknown, discrepancies between the text of the agreement and the applied operating rules.

Use of the decision table technique succeeded in
- restructuring logically complex articles
- pinpointing some of the ambiguities and discrepancies, one of which was found to have rather important (financial) consequences.

The applied method can be summarized as follows:

(a) In cooperation with personnel management, a number of articles qualifying for further analysis, were selected. Selection criteria were a.o.: logical complexity, experienced interpretation problems and possible advantage of decision table analysis. A short presentation on the use and limitations of decision tables preceeded this selection step.

(b) Analysis of the retained articles with the use of PRODEMO.

(c) Instruction of the responsable personnel management executives.

(d) Confrontation of the results obtained sub (b) and the concrete operation rules. Here again, as in the first case study, a bottom-up approach was preferred, which improved the motivation of the responsible end users.

This comparison often showed substantial deviations, in which case separate decision tables had to be constructed in order to deal with these discrepancies between the text and the real implementation of some calculation procedures.

(e) Final report.

Perhaps more advantageous than the achievement of the specific objectives was the introduction of the decision table as a useful technique to the responsable management.
This project also thoroughly illustrated the power of the technique as a means of communication between different management levels and executives.
These and other common experiences are briefly summarized in Section 5.

5. EMPIRICAL CONCLUSIONS

In our opinion the foregoing examples, taken from a vaster lot of empirical studies, fully illustrate both the applicability of decision tables as a decision support technique and the contribution of PRODEMO as a decision support system.

As is demonstrated by case studies 1 and 3, the computer-supported use of decision tables is rated at its greatest value during the administrative/organizational analysis phase, leading towards a more valuable information system. Notice that this phase not necessarily implies the development of an *automated* information system.

In what follows, we mainly evaluate the use of decision tables and of the PRODEMO system from this perspective. We successively deal with

- some empirical notes on the use of decision tables as a decision support technique (5.1)
- the repercussion of practical experiences on the further development of the PRODEMO system (5.2).

5.1. Decision tables as a decision support technique

As was indicated in Section 2, only a well-defined class of *procedural* decisions can be influenced in a positive way by the systematic application of decision tables. Practical studies, however, revealed that this class covers a wide range of (certainly not always well-structured) decisions at each level of the organizational hierarchy. In what follows, we try to summarize our experiences obtained in application studies in different fields.

5.1.1. The changing role of the decision/information analyst

a. The very simple, computer-independent structure of the decision table enables the users to analyse *themselves* their activities and/or decision processes. By the exhaustive enumeration of the successive cases in the condition part of the decision table and by the numerous PRODEMO features for manipulating the decision description at hand, the users are stimulated to proceed into the direction of a highly *autonomous* design of better procedures.

b. Due to the fact that activity/decision procedures are *documented* in a more understandable way, the decision/information analyst becomes a more replaceable person. Instead of being responsible for the final content of the information system under development, he is assigned the role of catalyst.

5.1.2. The changing users' attitude

a. By many users of the PRODEMO system, the decision table scheme and the facilities offered by the interactive system for dealing with *structures* of decision tables were appreciated as a handy tool for maintaining an overview of their activities. Although no long term experiments are carried through, the use of decision tables seems to become a very natural practice.

b. In some cases, the resulting decision tables were extensively used as means of communication. This feature was especially mentioned by tactical management people, trying to pass down their directives to the operational management level.

c. Initially more amazing were these cases where decision tables were introduced as a tool for detecting divergences between directives (laws, regulations, ...) and their realization in practice. Especially striking were the nature and the dimension of the detected anomalies.

5.1.3. Recommendations

a. Whereas the use of the PRODEMO system itself didn't add any additional problems, the effective and efficient use of decision tables for structuring and/or analyzing procedural decisions was a rather tough problem. Initial *training* by means of case studies incorporating a growing degree of reality is strongly recommended.

b. In order to minimize any *psychological resistance*, the introduction of both the decision table technique and the use of PRODEMO shouldn't start with the computer department. Besides, in a single case better acquaintance with decision tables was used as an interpersonal weapon in the organization under study. Once more, this event underlines the importance of *a priori* training.

c. During this initial training period, the inherent *limitations* of the decision table technique should be discussed. In general, an introduction to decision tables leads to *over-enthousiasm*, which strongly reduces the ultimate effectiveness of both decision tables and the PRODEMO support.

5.2. The PRODEMO system in perspective

Systematic use of the PRODEMO system in its turn influences the further development of the interactive system itself. The following is only a very partial selection out of this mutual influence:

a. *Users' friendliness* of the system is reduced by the increased number of implemented functions. This imminent problem was dealt with by introducing many new on-line help functions. Getting acquainted with the PRODEMO system was made much more simple by the preparation of a programmed-instruction-like text ([5]).

b. Feedback from different PRODEMO-users, together with personal experiences, induced a high number of small changes and improvements, particularly enhancing the ease and speed of use.

c. Unanimously, the availability of PRODEMO is considered to form a fundamental contribution to the practical applicability of decision tables. Functions like

the 'touch mode', changing condition order a.o. are really appreciated in real-life studies, where proceeding in a straight forward and irreversible way is almost utopian.

The extension of the available PRODEMO capabilities should be looked for in two directions:

a. A more powerful and above all more differentiated interactive specification language should replace the current one. A first proposal was elaborated in [3].

b. By incorporating higher intelligence in PRODEMO, one could enter functions like the automatic detection of 'trends' during the decision specification phase and the automatic splitting of too complex decision tables into a hierarchy of smaller subtables.

Final Note

Parts of this paper, especially sections 2 and 3, previously appeared in a slightly different form in [4].

(1) This so called 'bottom up' approach also enlarges the user's trust in the capabilities of the decision table technique.

REFERENCES

[1] Chvalovsky, V., Problems with decision tables, Comm. of the ACM,19,12 (December 1976) 705-707.

[2] London, K.R., Decision tables (Auerbach Publ., Princeton, 1972).

[3] Maes, R., Bijdrage tot een kritische herwaardering van de beslissingstabellen-techniek, Catholic University of Leuven, Fac. of Applied Sciences, Doctoral Thesis, 1981.

[4] Maes, R., Vanthienen, J. & Verhelst, M., Procedural decision support through the use of PRODEMO, Proc. 2nd Int. Conf. on Information Systems, Cambridge (Mass), (Dec. 7-9, 1981) 135-152.

[5] Vanthienen, J., PRODEMO getting started, Catholic University of Leuven, Dept. of Applied Economics, Internal Report, 1980.

[6] Verhelst, M., A technique for constructing decision tables, IAG Quarterly Journal, Vol. 2, No. 1, March (1969) 27-36.

[7] Verhelst, M., De beslissingstabel : een nuttige techniek voor management, Tijdschrift voor Economie en Management, 20,3 (1975) 393-412.

[8] Verhelst, M., De praktijk van beslissingstabellen (Kluwer, Deventer/Antwerpen, 1980).

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

REQUISITE FUNCTIONS FOR A MANAGEMENT SUPPORT FACILITY

Gary W. Dickson

Management Information Systems Research Center
University of Minnesota
Minneapolis, Minnesota
U.S.A.

The equivalent of today's black dial telephone in the early 1990's will be a management support facility. These devices will cost, relative to managerial wage rates, roughly four times what today's telephone costs. The important issue concerns the functions these devices will perform. This paper defines the functionality of the MSF based upon managerial and professional task requirements plus a forecast of the MSF's technical attributes.

BACKGROUND

In an earlier paper written by this author, discussion was presented concerning the technical attributes and the functions of a device termed a "management support facility," or MSF [1]. The timeframe was roughly ten years in the future, the early 1990s. A management support facility was suggested as the evolution of today's personal computer. It was projected to consist of a CRT display, a keyboard, a printing device, direct access secondary storage, and being capable of data communication.

One of the most startling aspects of such a management support facility will be its pervasiveness. Given the development of a manufacturing and distribution infrastructure, these devices will be very widespread in the office and at home by the early 1990s. One of the reasons for proliferation will be the economics associated with the MSF. In the referenced paper it was stated:

> Currently a telephone costs about $100.00. A professional manager is paid, including fringe benefits, about $50,000 annually. The ratio is .002. In 1990, a fully functional management supporting facility will cost about $10,000 in 1990 dollars. Our average professional manager will be making about $132,500 per year if an inflation rate of 10% is assumed. This ratio is .0075. This means that, relative to managerial wage rates, the management support facility available early in the 1990s will cost 3.75 times what today's telephone does ($375.00) [1, p. 2.].

Moreover, it has become apparent that the managerial support functions of the facility are not dependent upon an extremely accurate forecast of the unit's technical attributes. For example, whether the graphics provided by the unit are in color or are black and white appear not to have a major impact on what one does with the graphics. In other words, arguments can occur as to the finer points of the attributes of the unit but the ultimate form taken by these attributes will not be awfully influential upon the functionality of the device.

The purpose of this paper is to expand the previous discussion of the management support functions to be provided by the unit. Concentration will be upon those functions that are most meaningful and which must undergo the greatest enhancement

over what is available today. In order to provide the reader with some perspective, it is useful to very briefly review some forecasts (by the author) as to the technical nature of 1990's MSF.

THE TECHNICAL NATURE OF THE MSF

Within a ten year timeframe, we have a good notion of the characteristics of basic computer technology [7]. In the ten year period, no new basic technologies are forecast to be in use, even in large scale systems. The attributes of the managerial support facility will be enhancements based upon contemporary personal computers and the technologies employed therein.

The projections which follow are based upon a system price of about $10,000 (U.S. dollars will be used throughout the paper) in 1990 dollars. Any special enhanced features which are mentioned will not increase the total cost over $20,000. The only item that is difficult to forecast is the cost to be included to cover the software which will make the system attractive to managers.

An estimate of software costs included in the management support facility is that they will be from three to five thousand dollars per unit. These figures are not at all unreasonable when one considers that the currently available Micro-DSS/Finance software costs $1500.00 per installation and provides only one partial subsystem of a complete modelling capability and does not perform at all a number of all functions to be introduced below. These software costs are included in the $10,000 figure.

ATTRIBUTES

Figure 1 summarizes the likely technological attributes of the MSF.

	Basic System	Available
CPU		
Memory	512K/1Mbyte	5Mbytes
Wordsize	32 bits	----
Secondary Storage		
Rigid Disk	5-30 Mbytes	100-500Mbytes
Floppy Disk	1Mbyte	----
Input/Output		
	Keyboard	Touch Sensitivity Logistics Light Pens Limited Voice
	Matrix Printer	Other Printing Techniques Plotters
High Resolution	High Resolution B & W CRT	Color CRT
Communications		
	9600 bps local 1200 bps long distance	5M-50Mbps local calls & long distance

Figure 1
Capabilities of basic MSF and features available as extra cost options

IMPLICATIONS

Before leaving the technology section, some of its implications on functionality

should be noted. First, recognize that the realitively large memories will commonly not need to employ multiprogramming and thus a great deal of capacity will be available for a user application program. Couple this condition with the higher available processor speeds which will be forthcoming and one can see that some very interesting uses are possible.

Second, fairly high disk capacities and transfer rates will be available at the MSF. Couple this capability with the fact that communication network speeds (either baseband or broadband) will be available to transmit relatively large data volumes. The implication of these characteristics is that MSFs can be used to manipulate meaningful local databases and "subdatabases" drawn from other systems.

As will be seen, these two conditions have important implications concerning the functionality of the MSF. Before going into a projection of functions, however, it is useful to provide a bit of discussion concerning the environment in which the MSF will operate. The available technology and the requsite tasks in the user environment are what determine the MSF's requsite functions.

ASSUMPTIONS AND CONSTRAINTS

A MSF will support managers and professional workers. This section contains some qualifying statments about the environment in which support will take place and about the nature of the work tasks to be supported by the facility.

THE ENVIRONMENT

The scenario being projecting is one in which many managerial and professional positions will be supported by the Management Support Facility (MSF). In many organizations which are heavily professional, virtually everyone will have an MSF. In addition, each secretary will also have a device communicating with each MSF. Realize that we are speaking of a time period only eight to ten years away and a cost of $10,000 per unit which amounts to a large investment in a short time period. Since such a total investment by most organizations is not realistic, one must assume that complete utilization will be spotty in large organizations. In some cases, one department may be supported and others not. In other cases, some people in a function may have units but not others. Personal speculation is that the former installation will be much more effective than the latter.

One can also forecast that for many of these persons, two MSFs may be necessary to provide professional work support. One will be at the office workspace and the other would be located at home. Two implications stand out. One is that the market for MSFs is vast. The other is that the "computer in the home" will find as a primary task the extension of the professional workplace. In a sense, the coupling of MSF's in home and office creates an "electronic briefcase."

Returning now to the environment, qualify the comments about the MSF by noting that we are speaking primarily about personal units and personal support in contrast to group support as being recently considered by several authors [2,5,10]. This is not to say that the MSF could not be utilized in a group decision process (in which members of the group are all physically present at the same time). The intention is that individual support is the fundamental concern of this paper. In addition, the workplace in which the MSF will have most impact is one that is heavily professional or managerial. This qualification means that we are not talking about support for an office that is primarily clerical in function. Managers, staff personnel, engineers, educators, and the like are the audience addressed in the following discussion.

So, from an environmental point of view, one should envision a number of professional workers each of which has at their desk (and perhaps at home) an MSF. The MSFs can communicate with devices utilized by the clerical staff. In other

words, the facility supporting the clerical worker may be different from that used by the professional worker. Although it is conceivable that the clerical worker may utilize a MSF, it is more likely that their unit will much more like today's full function word processor. The manager or professional worker will prepare and/or edit text and the clerical person will take care of what can be termed as the paper's 'final cosmetics.'

These MSFs can also communicate with each other on a local network or with others at remote locations over long distance communications facilities. They also can communicate with large scale computer networks in either a "dumb" or "smart" mode. Since acting as a dumb terminal to a computer system is an obvious function of an MSF, it will be mentioned here but nothing else will be said about this function. The smart functions are much more interesting and worthwhile to speculate about.

SUPPORTING THE PROFESSIONAL

There are several issues that need to be discussed in relation to professional work and the MSF. First, the assumption is that the MSF will be used by managers at all levels and by professionals that, technically, are not managers in the strict sense of the word. Thus, professionals undoubtedly spend their time doing different tasks than do managers (at least in terms of percentage of time spent). Take as an example university professors and managers. In contrast to a manager, professors spend a considerable percentage of time in writing. Further, the type of writing done by a professor is atypical of that done by a manager. Another type of professional, say an engineer, may spend a disproportionate amount of time in calculating. Nevertheless, the MSF should support all of these tasks.

Regarding the manager, one may envision the MSF serving all levels in more or less the same form. The higher level manager, because of budget (ability to add features) or task, may have an expanded facility in terms of technical attributes, but the basic functions ought to be available to all users.

In most instances, the professional person would be the direct user of the facility. Some managers, especially those at the higher levels may utilize an intermediary. As time goes by and new people move into the higher management levels, the intermediary function may diminish! Personal use of an embryo MSF leads the author to feel that to interpose another person between the user and the machine would be so awkward as to make the concept of a MSF unworkable except in regard to a few of the functions that will be presented.

Much has been heard about how managers, especially those at the top of the organization will not interact with a device that is at all unfriendly or requires keyboard input which may be seen as secretarial work. This writer does not agree with these views. In the first place, at no level of work will the MSF be utilized more than about 20% of the user's time (even less at higher managerial levels). In the second place, many people (even managers) can type and find the keyboard not all that difficult. Others are learning and can learn to type. Use of an Apple computer system has already caused a personal faculty colleague to do this.

If the MSFs are as popular as many of us think they will be, the peer pressure to use them will be tremendous. There is nothing like a very senior executive user demanding interaction by machine on the part of subordinates to get these people using the systems. Finally, virtually all entry level professionals entering the workplace are coming out of educational programs in which they have terminal interaction with a computer system. These persons will readily adapt (some would say stampede) to the use of MSFs.

As a concluding point, consider that the MSF system under discussion will be very user friendly. In the words used by Sprague, the MSF will, "focus on features which make them easy to use by noncomputer people in an interactive mode and they emphasize flexibility and adaptability to accommodate changes in the environment

and the decision making approach of the user" [9, p. 2.].

FUNCTIONS

Defining the functions of a MSF is difficult because not a great deal is known about what managers and professionals do and, more importantly, how they do what they do. Despite the work by Mintzberg and others (e. g., Ives and Olson, 1981), our insights into managerial work are based upon very limited samples and are at a level of abstraction too high to be of much help in designing a facility to support managerial work.

One fact does appear to be clear. Observations lead us to the conclusion that individual differences are very great concerning how people perform work. A few simple examples will illustrate this point. One person may prefer to pick up the telephone to confer with a subordinate, whereas another may go to the person's office to have the same conversation. Frequently we find some of our academic colleagues who love to work at the computer terminal writing programs in APL or VISICALC. Other individuals may delegate this programming to a graduate assistant and still others may not want anything to do with these sorts of applications, preferring to work on problems in a different manner entirely. Finally, consider the writing of reports. Some use the writing pad approach, a few may use a typewriter or word processor, and others use a dictation approach.

These observations support the conclusion that different managers and professionals will use a MSF in very different ways. Some will utilize many features heavily, others may use very little of what the facility offers. Finally, because of differing job tasks, some people may have facility functions that are extra cost features and are not available with the basic facilities. Color graphics as a technical attribute is one example. A software system to do "problem finding" is an example of a programmed function that could fall into this category.

Let us begin the discussion of MSF functions begin by presenting some that are rather obvious and have been discussed for quite some time. The discussions will then go on to some more interesting functions, some of which have not been discussed elsewhere to any great degree.

WRITING

Managers and professionals all "write." That is, they prepare memos, letters, reports, and in a few cases manuscripts. Thus, the MSF will support text preparation functions. The modes of operation will allow the "author" to enter text, edit text, finalize text, and distribute final copy. In some cases, the author will perform all these functions depending on individual style. In other cases, a clerical support person may do the entry (with the author still working with pen and paper!) and the author will edit on the MSF. In the opinion of this observer, a common mode of operation will be that the author will perform the first two functions and the clerical person will "finalize" the text. This is the function referred to previously as getting the final cosmetics just right. Here is an example of an instance in which the facility available to the clerical support person will have a superset of special purpose functions. Our author will transmit edited text to the clerical station where the cosmetics will be taken care of and the text sent back to the MSF for final approval and distribution.

It is not very worthwhile to get into details concerning the type of text processing support needed by professional from the MSF. The text processor used by this writer is a good example of what a manager needs. It is very simple and uses no special characters or text marking. Functions are available for entering text, editing, and moving text around. Its best feature is its simplicity. It should be added that from a secretarial point of view and from the perspective of performing all the functions listed above (especially final cosmetics), the author's system suffers somewhat. But, as always, there is a tradeoff. Who are we serving, the

professional/manager or the secretary?

One final observation here concerns the support of different types of work tasks on the same MSF. An academic preparing manuscripts, has different text processing needs from those of a manager writing memos, letters, and reports. Handling footnotes at the bottom of a page is one example of differing needs. The question is, who is supported? One answer is to build in the superset of system/program features but make only a basic subset functional on the standard system. If the special features are required, they can be "requested" by the user needing them. Thus, in the author's case, I get the bare bones text processing but if I want to make the extra effort to learn how to use the "manuscript preparation features," they are available to me.

COMMUNICATING

Another function that is pretty obvious is that of communication. The user of the MSF must have the ability to transmit text and files to and from other MSFs and, in some cases, to and from large scale computers. Clearly, common files will exist on the network and special purpose subfiles will also be utilized. Point to point transmission and various types of distribution lists are obvious applications. The communications function can be briefly characterized as having an easy way of specifying what is to be transmitted and where.

INDIVIDUAL PROCESSING

Almost everyone, whether they are a professional or a manager, will have a number of individual applications they will want to access periodically. The engineer, for example, may have an estimating program; the professor will have a grading program. These types of calculating and report generation applications will vary by person and job, but represent a diverse function that will be present in almost every case.

These personal applications may be written by the user in a commonly available language that runs on the MSF (e. g., APL, PASCAL, BASIC) or written in the MSF's higher level language as a special purpose procedure. Another source of the applications may be the general program marketplace. The notion is that these are individually oriented, self-contained types of applications whose data, if any, will be local. Each MSF user will have a portfolio of such applications.

MANAGING DATA

This is a function that begins to get more interesting. Professionals and managers do work with data. Alter [1] gives us one perspective of the managerial use of data. He states that functions involve: (1) retrieving a single item of information; (2) ad hoc data analysis; and (3) prespecified report generation. He goes on to couple data functions with analysis functions, but this topic will be deferred until later in this paper.

The proposed MSF will allow all these types of data functions to be performed on data. What is interesting with the MSF network concept is that the data <u>does not have to be purely local data.</u> It is very clear that the way the MSF concept will work with data is to be able to establish local, "working databases," which may be drawn from central databases or distributed databases. Keep in mind the fact that the MSF will be capable of storing from 5 to several hundred megabytes of packed data. The function must exist to abstract data from remote databases and to work on the data at the MSF.

Given that data exists at the local level, the MSF can easily perform all the functions described by Alter. Simple commands such as "FIND" coupled with logical operators will perform functions (1) and (2). After all, (1) is a subset of (2). Report generation is similarly easy. Even more important is the fact that the MSF

works locally with subdatabases. In other words, one can select a subset of the data on one attribute, work with this subset, and then generate sub-subsets and work on these.

An additional attribute of the MSF that applies not only to data management but other functions as well is that all the functions must be integrated. As an example, the data management and text management functions on the MSF should be totally integrated. For example, one should be able to create letters or reports utilizing the text management function and insert data into the letters or reports from the data management function. This type of integration will be key as will be seen in subsequent discussion.

The most important thing to recognize about data handling by the MSF is that it is imperative that the function exist to create local MSF databases which are abstracted from central systems. Although it is too early to tell, one may surmise that the local databases may not be too volatile and may need to be created only monthly or weekly. Even daily creation with the transmission rates mentioned in the technology section is not too much of a problem. This is an area into which research needs to be conducted to determine the nature of applications and the consequent technical considerations.

MAKING DECISIONS

Alter [1] also helps us in describing the functions involved in this area. He suggests that one must be able to propose decisions and estimate the consequences of proposed decisons. He goes on to describe several types of models which ought to be able to assist in these tasks. His concept is much the same as suggested by Sprague [9] in which a model database is a subsystem for DSS. Alter simply provides more detail.

One thing is obvious regarding this area. This is that the model subsystem must be integrated with the data management subsystem within the MSF. In other words, it must be possible to perform logical analysis on a local database, and then to execute models using the data from the resulting subdatabase.

Several types of "modelling" functions immediately come to mind as being required. First is the ability to do basic statistical functions on data. Descriptive measures, tests of differences, relational analyses, and projection are obvious basic functions. Second, projection of the type exemplified by current financial planning models should be available. One should be able to prepare either tabular or graphic reports.

Personally, the author would not suggest putting too much effort into doing the same thing for optimization models. Conceptually, it is not too difficult to be able to transmit data from the data management subsystem to an optimization model, but there is not likely to be a great deal of real demand for this feature. The feature will probably be available on the MSF because it is not too hard to do and offers another feature to support salability of the system.

The integration of the decision supporting subsystem and the data management subsystem is the area that is currently least developed and needs the most work. The most difficult problem appears to be generalizing the modelling functions without making the interface so difficult that no one will use these capabilities.

DECISION CONVEYING

An important part of any professional's or manager's job is "selling" ideas. Statements are often heard to the effect that, "It's not so much what one says as it is how one says it that is important in gaining acceptance of ideas." This statement implies the notion of conveying a recommendation or decision to others. The MSF will find use in performing this function. Peter Keen, in responding to a

previous version of this paper, recognized this very important function which has been added to the paper.

The use of graphics and, perhaps, especially, color graphics will be the technical attribute of great importance in supporting the decision conveying function. One can expect line graphs, bar graphs, and pie charts to find frequent use when the requirement exists to convey one's view of a situation, a recommendation, or a decision.

PROBLEM FINDING

This function is one that has received little attention in recent years. One of the most important managerial functions is finding problems. Typical enhanced by exception reports. Given the technological attributes of the MSF of a fast processor, a large memory, and access to relatively large data bases, this function can be expected to find use. Why not use the analytical capabilities of the system to bring potential problems to the attention of the manager?

SOME ADDITIONAL SPECULATION

Fox [4] describes some of the types of user interactions which he forsees being supported by an "intelligent management system." Among these interactions are: "Tell me when...; You've got problems...; and What if....". These general categories translate into specific questions such as:

* What should be done if the milling machine goes down?
* Why is productivity dropping?
* When is it time to buy a new piece of equipment?

It seems to this writer that many of these types of questions can be dealt with using the MSF. The joint functions highlighted here are those of data management coupled with analysis and modelling. These joint functions, it appears are the ones which the research community ought to pursue most vigorously. In concluding this paper, a few comments regarding these functions are in order.

First, let us examine the problem finding function. What is required here is to have a sensing function which, because of its size, may be associated with what we know today as the large scale, general purpose computer. Using one or more approaches, problems can be sensed by the system. Statistical analysis of historical data or heuristics based upon "expert" judgment are two approaches to problem finding which come to mind. In any event, assume that a potential problem is called to the attention of a particular manager supported by a MSF. Either the system, the manager, or a combination of both may select the data necessary for analysis of the problem situation and communicate the data to the manager's MSF.

At this point, the manager uses the MSF to explore the nature of the problem and attempt to solve it. One can envision a "search" function in which the files of the subdatabase are scanned to obtain information. Beyond simple queries such as "find," it is likely that relational conditions will be employed. An example is reflected in a request such as, "Find to what extent the sales of Department X in the month of March correlate with advertising expenditures in January."

Following this type of search, the manager may wish to project data in a "what if" mode. This may require model building on the spot and entry into the data management subsystem to obtain data to feed the model. A recursive process may then occur in which the manager "looks backward" to the data to explore relationships to be employed in model projections.

The point of this discussion is to demonstrate that the least well understood functions to be supported by the MSF are those in which the data management

subsystem is integrated with the modelling subsystem. The understanding of the integration between the data management subsystem and the "communication" subsystem appears to be much easier to deal with.

It has been postulated in this paper that the MSF will have technical attributes that will allow communication, data management, and modelling to be supported in a pervasive manner. Further, on a gross level, the management supporting functions of the facility have been described. There appear to be two areas which call for immediate attention in order that we may realize the highest level capabilities of the device which we have been considering. One is primarily a technical problem whereas the other needs a great deal of intellectual effort.

The former area is the one dealing with selecting data from a large transactions processing oriented system and transmitting subfiles to the MSF. In other words, one needs an easy way of creating files on the MSF from a host database management system. This is an area that ought to be addressed by those persons responsible for the development of database management systems. These systems ought to be developed keeping in mind that there will be a requirement to strip out data to feed MSFs.

The latter area is the one dealing with the integration at the MSF level between the data management subsystem and the modelling subsystem. Modelling with interactive computer support is still in its infancy. Most modelling support currently available either takes a form dictated by an algorithm, e.g., LP, or is some designer's best guess as to what ought to be available and how the user ought to interface with it, e.g., statistical packages and planning languages. In the database management area, we have at least a primitive understanding of what type of queries to expect and how to state them. We need, as a beginning, at least this same level of understanding with regard to modelling. Precisely, what kind of data analysis will managers be doing, and therefore what modelling functions must be provided?

To date our understanding of this question has been gained more or less by the market test of providing a particular tool (capability) and finding out if anyone will use it. It is the author's contention that a more cerebral approach to this problem is called for. Given the technology that is coming, we will be very inefficient in utilizing it unless we do a better job of exploring the requisite functions required for the support of managerial problem solving.

REFERENCES

[1] Alter, S., A Taxonomy of Decision Support Systems, Sloan Management Review, 6 (1977) 39 - 55.

[2] Couger, D. (ed), Decision Rooms Facilitate Both Teaching and Learning, Computing Newsletter, 3 (1981), 1 - 2.

[3] Dickson G. W., Toward A Definition of the Attributes and Functions of the Basic Management Suport Facility of the 1990's, 15th Hawaiian Information Systems Conference, Honolulu, Hawaii (January 1982).

[4] Fox, M. S., The Intelligent Management System: An Overview, Intelligent Systems Laboratory, The Robotics Institute, Carnegie-Mellon Univ. (August, 1981).

[5] Gray, P., Berry, N., Aronofsky, J., Helmer, O., Kane, T., and Perkins, T. The SMU Decision Room Project, in Keen, P. and Young, D. (eds.), DSS-81 Transactions (Execucom Systems Corporation, Austin, Texas, 1981), pp. 122-129.

[6] Ives, B. and Olson, M., Manager or Technician? The Nature of the Information Systems Manager's Job, MIS Quarterly, 4 (1981), 49 - 64.

[7] Little, A. D., Future Information Processing Technology, A Report to The Defense Intelligence Agency (unclassified), (November, 1980).

[8] Mintzberg, H., The Nature of Managerial Work (Harper and Row, New York, 1973).

[9] Sprague, R. Jr., Framework for the Development of Decision Support Systems, MIS Quarterly, 4 (1980) 1 - 25.

[10] Wagner, G.,DSS: Dealing With Executive Assumptions In The Office Of The Future, in Keen, P. and Young, D. (eds.), DSS-81 Transactions (Execucom Systems Corporation, Austin, Texas, 1981), pp. 113 - 121.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

OPTRANS : A TOOL FOR IMPLEMENTATION OF DECISION SUPPORT CENTERS

Michel Klein(1), Alain Manteau(2)

(1) CESA, Dept Informatique, 78350 Jouy-en-Josas FRANCE
(2) SIG, 4bis, Rue de la Libération, 78350 Jouy-en-Josas FRANCE

This paper :
- introduces the concept of a Decision Support Center (DSC) and of a Decision Support System Generator (DSSG)
- gives an example of the implementation of a DSS using OPTRANS
- points out some important concepts of OPTRANS and give some examples of some of the language constructs available in OPTRANS to use these concepts
- describes our experience with the system

1. THE CONCEPT OF A DECISION SUPPORT CENTER (DSC)

1.1. The concept

A DSC aims at providing a large population of users with :
1. a tool to develop their DSS for a wide spectrum of applications, such as :
- financial analysis and engineering,
- investment analysis and portfolio management,
- financial planning,
- management control,
- planning and budgeting,
- market analysis and marketing using market data bases,
- cost accounting,
- production planning,
- consolidation,
- statistical analysis.

2. the capacity to create and maintain different data bases in an evolutionary way.

3. the capacity to have their DSS accessing several data bases.

4. the capacity to support group communication and decisions through a teleconferencing system.

Having developped DSS in the mentioned application areas, we felt the need to develop a tool to automate the generation of DSS and, as a consequence, to decrease the investment to implement them.

1.2. Example of a situation where a DSC is relevant

A typical DSC implemented in a bank may comprise the following DSS.

DSS needed at head office level

a) DSS for management control. The controller and his assistants have to follow for all branches the variance between actual and budgeted results on a monthly

basis, period by period, cumulative in various currencies for the balance sheet and income statement.
They have to compare the performance of branches, to consolidate them comparing financial aggregates and to make simple statistical analysis.

b) DSS for financial planning

The main goal is to compute the financial consequences of management decisions, on a routine basis, eg. the impact of change of the interest rate structure on the financial result of the bank, the impact of the latest forecasts of the branches on the year-end result.

c) DSS for cash management

d) DSS for risk analysis. This DSS is used to check the proposed loans to client companies made at the branch level.

e) DSS for the financial engineering group

This group needs a DSS to enter, modify, save and retrieve financial data on a compagny. It uses deterministic medium to long-range financial planning models dealing with activity levels, expenses, investment and depreciation policies, working capital requirements, financing policies.

f) DSS for the investment department

This department uses a DSS for the management of the bank investment in stocks, bonds a well as the management of portfolio for bank clients.
The data base of this DSS contains prices and dividends for a few hundreds stocks. This information together with other important economic indicators are kept on a monthly basis.

g) DSS for the economic and statistical department

The economic department uses a DSS to make statistical analysis on a large data base of economic and financial series.

DSS needed at branch office level

h) DSS for branch planning

This DSS helps the branch manager to prepare his budget given a set of objectives, growth rate for the accounts of the balance sheet, and the interest rate structure. The DSS also helps him to follow his results, make his own variance analysis and to test hypotheses.
This DSS is the same as the one used by the controller at the head office.

i) DSS for credit analysis...

This DSS is used by credit analysts to enter the financial data (balance sheet, income statement) and other non numerical information on companies applying for loans.

Given the administrative rules of the bank, all loans above a certain amount will have to be agreed by the "head office". We have here a typical sequential decision process where decision makers are geographically scattered.

In this banking organisation seven different DSS may be applied sharing four main data bases. For an average size bank the numbers of users may be estimated as follows :

for the credit DSS : roughly 200 persons,
for the control DSS : 45 persons,
for the financial planning DSS : 5 persons,
for the economic DSS : 3 persons.

1.3. The lack of adequate software to meet this need

1.3.1. Inadequacy of current DBMS

The situation above mentioned may be approached through the application of data bases because :
. numerous users have to share information,
. integrity of data has to be preserved,
. the system has to be evolutionary and expendable,
. access control to different sub-classes of data has to be preserved.

In Klein, Tixier [1] , we defined some of the specific characteristics of data base oriented DSS. More recently, Methlie [4] pointed out DSS specific requirement in data base management. Manola [2] identifies eleven requirements for a DSS, which we shortly recall.
1. Ability to construct accurate models of an enterprise, including entities of interest in the enterprise, their interrelationships with one another and consistency constraints which govern their interaction.
2. Ability to retrieve data from a data base using a relatively high level and easy to use query facility.
3. Ability to apply various types of special purpose software packages to data obtained from the data base.
4. Ability to specify the format of the output of an analysis or retrieval so that it can meet various user requirements.
5. Ability for users to easily determine what data and software tools are available in their environment.
6. Ability to produce output, not only in hard copy form such as reports, but also using charts, diagrams, pictures, video-display.
7. Ability for software to provide human-engineered computer interface for decision-makers.
8. Ability of systems to provide "triggers" or "alerting" capabilities.
9. Ability to provide access by users to other large data bases or libraries.
10. Ability to provide effective communication facilities among decision-makers when more than one person is involved in a decision process.
11. Ability of software tools to realise efficient performance.
It is an experience that no commercially available system provides such a set of capacities needed to implement DSS.
Concerning point 2, very few DBMS provide a modeling capacity for the casual user. Most query languages do not let the user transform data. They are merely retrieval tools, with no capacity to define and retrieve new objects.
Concerning the capacity to apply special purpose software packages (point 3) to data in the data base, this is possible in some specialized systems such as EXPRESS [9] or FOCUS [10] .
Concerning point 4, interactive report generator are usually not available in a DBMS environment.
Concerning point 5, it is usually difficult to explore a data dictionary and obtain semantical information on data.
Concerning point 7, very few DBMS have a DML and DDL which use constructs at the conceptual level. Shipman in [12] presents a DAPLEX system not yet implemented.
Concerning point 8, this characteristic is to our knowledge not found in commercially available DBMS.
Concerning point 9, the ability to transfer automatically a large amount of data from one data base to another, is not an usual DBMS-feature.
Point 10, Klein [6] reports on the use of a teleconferencing system in conjunction with DSS. The communication support aspect of DSS is getting more and more interest among researchers [23] .

Point 11, most DBMS need a large amount of resources (central memory, processor speed) and usually are much too complex to operate efficiently for most DSS situations.
We point out in the next sections how these problems are dealt with in OPTRANS.

2. THE CONCEPT OF A DECISION SUPPORT SYSTEM

For the notion DSS, we refer to Klein [1] , Scott Morton [16] , Sprague [23] . We adress here the class of business problems that might be solved with DSS and the functional requirements of such systems.

2.1. The class of problems relevant to a DSS approach

Problem solving using DSS may be characterised as heuristic problem solving and learning using an evolving data base.
Characteristics of the problems involved are :
. the problem is badly structured,
. a decision has to be made in limited time,
. the problem is not well defined,
. the criteria are numerous and changing with the user,
. intuition, judgment and experience are crucial factors in the decision,
. the possible paths toward a solution are numerous and for each of these paths these is no algorithm, but a blend of hypothesizing and computing.
- Another characteristic of this set of problems is a data base. In all these problems the first step towards a solution is fast access to relevant data.
- There are, usually several data bases under the control of different executives and computations have to be done on data from several data bases. Last but not least, much of the data usually exist in data bases of the data processing department. The problem is then to perform a data extraction process, as pointed out by Methlie in [4] .
- There is a computer assisted-instruction aspect (see [1]). The reason is that, while quantitative methods are developing rapidly, nevertheless, intuition, judgment and experience remain essential factors, in the process of exploration and search for adequate tools. This implies that the key to success is not to developp better or more quantitative techniques, but to improve the search ability of the user, to help him acquiring better heuristics, to improve his knowledge of the limits and applicability of the tools he has at his disposal.

2.2. Functional requirements

The above set of problems imposes on the DSS design a set of functionnal requirements. These requirements were listed in Klein and Tixier [1] . We recall them :

2.2.1. Convenience, since users are not computer scientists

This bears upon :
- linguistic aspects : vocabulary and syntax, which must be simple and natural to the user,
- handling of error cases and abnormal conditions.

2.2.2. The system has to be evolutionary and extensible, in the following respects :

- languages and associated semantics. It must be possible to introduce new objects, new syntax forms, new models, without disturbing day to day system use.
- subsystems. It is necessary that a tree of subsystems can be created. These subsystems are developed independently. The have to be protected as related applications may share the same data base functions.
- developing private libraries and data bases. This is the so called "computer utility" problem.

2.2.3. The user must be close to the machine, using a time-sharing mode.

3. FUNCTIONAL DEFINITION OF OPTRANS

In this section we give a user's view of OPTRANS and make explicit :
- some important concepts of the data base subsystem,
- the modelling language,
- the report generator and graphical subsystem,
- and give some indication on the specialized subsystems available today and on the assistance provided to the user.

3.1. Description of the system

The structure of OPTRANS is represented in fig (1) and (2).
There is a structure with a nucleus and several specialised subsystems.
The nucleus is to be used in the generation of any DSS, the other subsystems can be developed according to the problem area.

3.1.1. Description of the nucleus

The nucleus consists of 4 subsystems :
- a specialised DBMS,
- an information display subsystem,
- a modeling subsystem,
- a monitor.

3.1.1.1. The DBMS Function

This subsystem includes a data definition language (DDL) and a data manipulation language (DML) and other usual DBMS functions such as :
- multi-access to data,
- control of access,
- ability to allow concurrent update and retrieval,
- no redundant data,
- physical data independence.

The OPTRANS DBMS has a number of features specific for DSS.
- OPTRANS has specific features to deal with numerical data structures.
- The fundamental goal of OPTRANS is to provide a "conceptually natural" data base interface language for the class of users defined in section 2.1.
  The OPTRANS constructs used to model decision making situations are intended to match closely the concepts a human might use when he thinks about these situations.
- A report generator and graphical output capabilities.
- A data dictionary and commands to obtain semantic definitions of data, as well as assistance on algorithms of analyses available.
- The ability to transfer data from an OPTRANS data base to another OPTRANS data base and a data aggregation function.
  The basic constructs of OPTRANS are the entity and the relationship.

3.1.1.2. The information display subsystem

This subsystem includes a report generator and simple graphical capabilities for :
- drawing curves,
- scatter diagrams,
- histograms,
- bar charts.

The report generator includes a non-procedural language to create, save, retrieve and modify reports and a language to define reports.

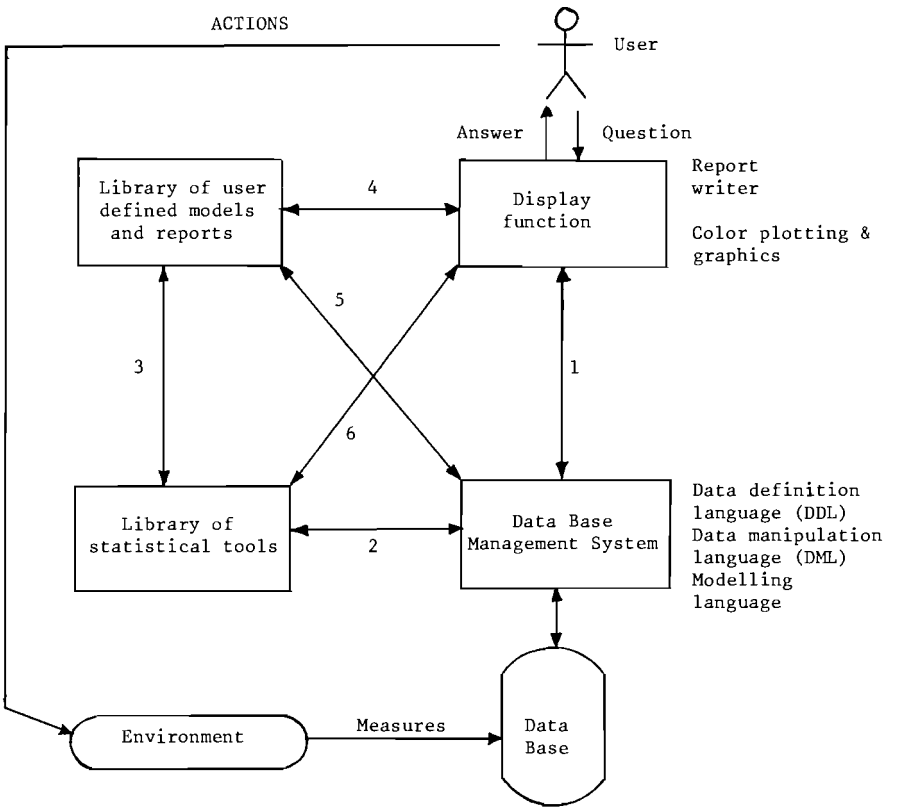


Fig. 1. The main functions of OPTRANS

Functional description of a DSS developed by OPTRANS.

(1) retrieving of information from the DB to be displayed in a report, a graphical presentation on CRT or colored graphics or plotter,
(2) use of statistical or optimalisation programs accessing data in the DB,
(3) use of statistical programs to analyze data,
(4) model interrogation without accessing the DB,
(5) model interrogation accessing the DB,
(6) use of statistical tools accessing the DB.

3.1.1.3. The modelling function

This subsystem includes a non-procedural language to create, save, unsave, evaluate and modify models, and a language with some procedural capabilities to define relations of the models.

3.1.1.4. The monitor

This is the system's executive, it enables the user to list and select available subsystems.
It also enables users to immediately access and execute models and reports.

3.1.2. Assistance to the user

Quality of user-support is a very important feature in a DSS. In OPTRANS several mechanisms are implemented :
- automatic diagnosis of syntax errors,
- possibility to request the syntax at any point,
- error and assistance messages,
- capacity to request the system to describe the concepts existing in the DB.

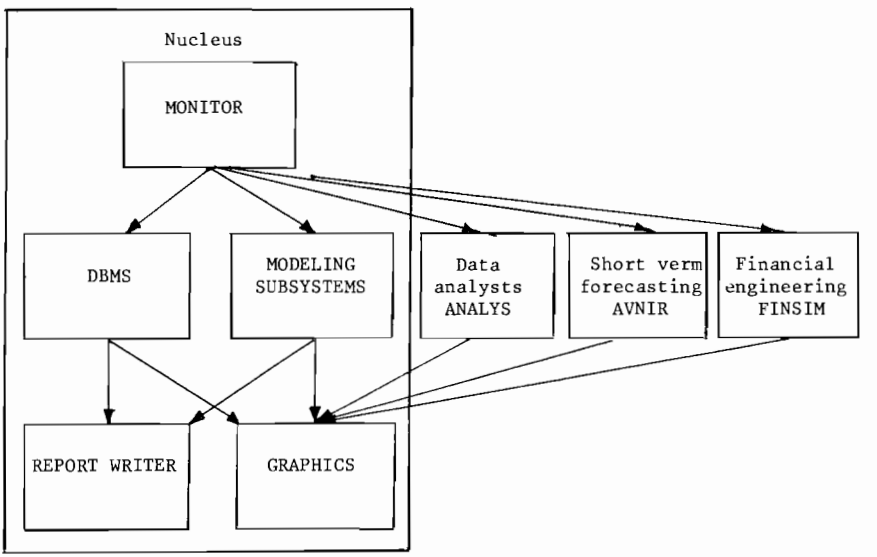


Fig. 2. The subsystems and the nucleus of OPTRANS

3.1.3. Specialized subsystems

For some problem areas specific DSS may be developed : financial analysis [6] and planning [28] , marketing [14] , management control [17] , portfolio management [25] , interactive use of economic and financial data base [1] .
For these domains the user expects the DSS to give him access to tools which have become standard.
It is the goal of the specialised subsystems to provide him with an access to these tools.

The concept of subsystem be clarified. A subsystem corresponds to a class of problems for which it has been possible to identify :
- a common vocabulary and syntax,
- a set of frequently used algorithms,
- data structures associated with these algorithms,
- a set of models corresponding to usual ways of formalising problems of this class.

We do not present the existing subsystems in detail, we refer to Klein [1] .

3.1.3.1. ANALYS (Data Analysis)

The goal of the ANALYS subsystem is to provide the user with a set of algorithms to study the data in the DB with statistical methods as well as data analysis techniques.

This subsystem may deal with various types of data : quantitative and qualitative in various scales.

With respect to statistical methods the available techniques are : segmentation, multiple regression, stepwise multiple regression.
With respect to descriptive methods the available techniques are : factorial analysis, correspondence analysis, discrimant analysis, typology, multidimensional scaling, chi 2, correlation, elementary statistics (mean, range, standard error...)

Description of this subsystem are given in Manteau [20] , [21] . This subsystem is usually used with OPTRANS in the definition of Marketing DSS, such as the ones described in Little [13] [14] .

AVNIR (short term forecasting)

The goal of this subsystem is to provide the user with most frequently used endogeneous models : trend adjustment or exponential smoothing, methods for non seasonal series (additive scheme), methods for seasonal series (multiplicative scheme).

Financial Analysis (FINSIM)

A library of interactive financial models known under the name of FINSIM is available,for more details see Klein [6] [15] . These models are supporting financial diagnosis and financial forecast for investment or credit analysts.

4. EXAMPLE OF USING OPTRANS FOR DEFINING A DSS

We discuss the DSS definition for the bank referred to in 1.2. We show the stages in solving the problem with OPTRANS. The problem has been reduced in size, for the sake of simplification.

4.1. The problem

The control department of a bank receives information on branches from the accounting department every term or month. In our example these informations have been limited to accounts from balance sheets and income statements of branches.

The manager of the control department wishes to develop a DSS to enable him to :
- store and update financial information on branches,
- question the data base, as to :
  - value of one account at a given time,
  - evolution of an account over time,
  - evolution of an account deflated,
  - financial ratios to measure profitability and performance,
  - comparisons between branches,
  - branches meeting certain criteria.
- consolidate accounts of balance sheets and income statements at branch level to obtain the balance sheet and income statement of the bank,
- compute forecasted income statements of branches according to the objectives which have been assigned to them,
- simulate the impact of a change in the term structure of interest rate on a branch.

4.2. Developing the DSS with OPTRANS : the stages

To implement a first version of the DSS the following steps are taken :
a) definition of entity-type names and relationship between entities in accordance with the model introduced by Chen [25] ,
b) definition of the concepts needed by the user, derivable from elementary entities,
c) loading the data base (DB),
d) defining models,
e) defining presentations of data.

4.3. Defining the DB

A user can distinguish four types of entities :
- the type variables (which will include three subtypes : account, growthrate, interest rate),
- the type date (time),
- the type branche,
- the type currency.

It is clear that some entities (accounts, growthrates...) of the first type are related to dates, branches and currency, since we can assign a numerical value to an account at a certain date for a given branch and a given currency.
In other words we shall have to create the four types of entities, define the entity names and then define the relationship between the entity types.
Then it is possible to ask questions such as : what is the value of the account "cash", date january 1980, for branch AG14, in French currency ?

This structure can be represented in the following schema, where circles are a symbolic representation of types of entities.

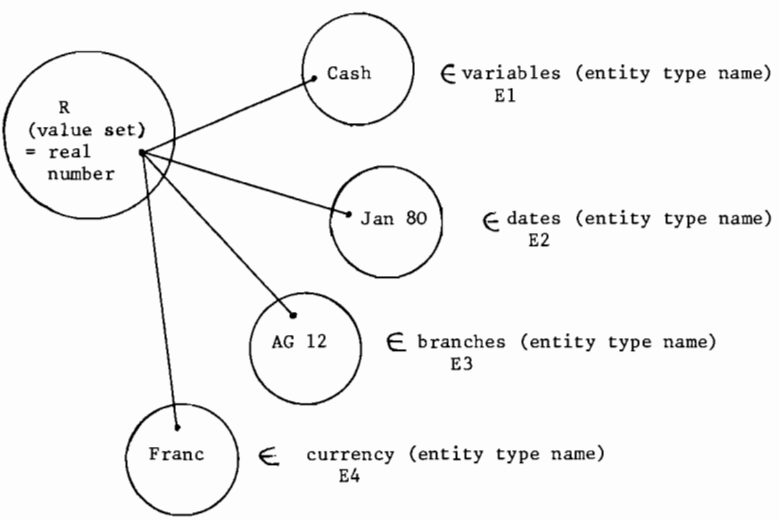


Fig. 3. The logical view of data in OPTRANS

Clearly each tuple of entities E1, E2, E3, E4 is a relationship. Note that the value set is not unique in OPTRANS since a value can be denoted as : estimated, certain, non available.

4.3.1. Example of the creation of entity types and entity names

Fig. 4 shows the definition of entity names using the DEFINIR (define) command.

```
BASE  ? DEFINIR SERIE CAISSE

BASE  ? DEFINIR SERIE 'BDF-CCP'

BASE  ? DEFINIR SERIE 'BANQUE ET CORRESPONDANTS'

BASE  ? DEF SERIE 'CREDIT A LA CLIENTELE'
```

Fig. 4. Example of definition of entity names

In this example one can see that we have two classes of entities : elementary entities and others. An entity which is non elementary is called an aggregate. Aggregates are in fact virtual data since they are not stored physically, but evaluated when needed.
As can be noticed on Fig. 5 functions such as !SOM can be used in the definition of a new entity.
Once a new concept has been defined it can be used as any other concept.

In the example (Fig. 5) we show the definition of two ratios using entity names which are number names. An entity name can be a list of symbols or a number (such as the account or budget item number).

```
BASE  ? DEFINIR SERIE 'TOTAL ACTIF'= !SOM CAISSE  A 'DEBITEUR ET REGULARISATION'

BASE  ? DEFINIR SERIE RATIO = 'CREDIT A LA CLIENTELE' / 'TOTAL ACTIF'

BASE  ? DEF 1 'RATIO 2' = :140 / :200
```

Fig. 5. Example of definition of aggregates (full name or numerical name)

```
BASE  ? CREER AGENCES

BASE  ? DEFINIR AGENCE AG12
```

Fig. 6. Example of defining entity type names

4.3.2. Type of variables in a define statement

The user can define new concepts using the define command.
For instance, if he wishes to define a moving average of the account "cash" over the last three periods he will define :

```
BASE  ? DEF SERIE 'MOYENNE MOBILE' = (CAISSE($N) + CAISSE($N-1) + CAISSE ($N-2)) / 3
```

To compute an index of value 100 in 1970 for the account "cash" the user will define :

```
BASE  ? DEF SERIE INDICE1 = CAISSE / CAISSE ($ AN 70)*100
```

To compute an index of value 100 in 1970 for the account "cash" taking "Agence 12" (branch 12) as a reference the user will define :

```
BASE  ? DEF SERIE INDICE2 = CAISSE / CAISSE ($ AN 70 AGENCE AG12)*100
```

The number "1" after the DEF (define) command is the reference number of the entity type it could have been its name.

4.3.3. Defining predicates

It is possible to define predicates, i.e. aggregates which take a value TRUE OR FALSE. Some examples are given below :

```
BASE  ? DEFINIR SERIE BPS = BENEFIT / 'NUMBER OF SHARES'

BASE  ? DEF SERIE DPS = DIVIDENDS / 'NUMBER OF SHARES'

BASE  ? DEF SERIE PE = PRICE / BPS

BASE  ? DEF SERIE PREDICATE.1 = BPS > 0

BASE  ? DEF SERIE PREDICATE.2 = PE > 20

BASE  ? DEFINIR SERIE PREDICATE.3  = (BPS > 0 ) ET (PE>20 )

BASE  ? DEFINIR SERIE PREDICATE.4 = (BPS > 0 ) OU PREDICATE.2
```

Fig. 7. Defining predicates

These predicates can then be used after a "TEL QUE" (Such That) modifier in a print statement.

4.3.4. Printing the list of entities names

It is possible to print the list of entity names, of entity names of the same type, any subset of a set. Entities with asterisk (*) are aggregates.

```
BASE  ? LISTER SERIE


DIMENSION : SERIE

 :100  CAISSE
 :120  BDF-CCP
 :130  BANQUES ET CORRESPONDANTS
 :140  CREDIT A LA CLENTELE
 :150  COMPTES DEBITEURS CLIENT.
 :160  CREANCES DOUTEUSES
 :170  EFFETS RECUS EN RECOUVREMENT
 :180  EFFETS A ENCAISSEMENT
 :190  SIEGE ACTIF
 :200  DEBITEURS ET REGULARISATIONS
*:210  TOTAL ACTIF
*:212  COMPTES CREDITEURS CLIENTS
```

Fig. 8. The list of entity names belonging to the entity type "serie"

4.3.5. Adding a comment to an entity name

It may be useful to be able to relate any entity of any type name with a text. Such a text can be the semantic definition of a concept, the source of the information. With this capacity OPTRANS can handle non-numerical data. This information can then be retrieved.

4.3.6. Establishing relationship between entities

In our example it is clear that the account and growth rate variables are related with time, branch and currency, whereas other variables such as interest rate are related with time only.

As a consequence the user will want to establish a relationship set between "account" entities of the variable type and entities of the date, branch and currency type and to establish a relationship between "growth rate" and entities of the data type. The syntax is :

```
BASE  ? AFFECTER SERIE CAISSE A DEBITEURS POUR TEMPS 1 AGENCE 40 'NATURE DES FONDS' 2

        DATE DE REFERENCE ?T4 80

        REFERENCE POUR LA DIMENSION (AGENCES) ? AG12

        REFERENCE POUR LA DIMENSION (NATURE DES FONDS) ? FRANC
```

Fig. 9. Establishing relationship and disc space allocation

The establishement of a relationship also gives the opportunity to reserve storage space for values. In the above example we save storage for 4 terms 40 branches and 2 currencies.

The user can then display the relationship using the command LISTER.

4.3.7. How to retrieve the definition of a concept in OPTRANS

A user of the system can obtain the definition of any aggregate using the DECRIRE (describe) command :

```
BASE  ? DECRIRE  SERIES PUB.EFFICACE PART.MARCHE

       DIMENSION : SERIE

       *:320  PUB.EFFICACE =  'EFFET MEDIA'*'EFFET MESSAGE'*'DEPENSE PUBLICITAIRE'

PART.MARCHE = 'MIN'+(('MAX'-'MIN')*(('PUB.EFFICACE'**'GAMMA')/'DELTA')+('PUB'**'GAMMA'))
```

Fig. 10. Using the describe command to obtain the definition of a concept

4.3.8. The concept of context in OPTRANS

Once the conceptual structure of a DB has been defined, it is possible to define a part of this structure. The context can be considered as the intersection of any subset of the entity name sets. After the definition of such a context, the user can apply the usual operations on it.

4.4. Entering values and marking of values

After a context has been defined the user can enter data with a command such as :

```
ENTER  Type 1   Type 2   Type 3   Type 4
       TEMPS    AGENCE  VARIABLE  DEVISE  (NATURE DES FONDS)
```

The order in which the type of entities are given in the command, determine the sequence of the prompts from the machine.

It is possible to add the information that a value is known, estimated or not available : by typing a figure followed by a "<" or "?". One important consequence of this capacity is that any aggregate computed using an elementary entity marked as estimated will be estimated and known as such by the user anywhere in the system.

```
BASE  ? ENTRER TEMPS AGENCES SERIE / 'NATURE DES FONDS'
 T 4 80
   AG12
     CAISSE
       FRANC ? 624
       C.VF DEVISES ? 29
     BDF-CCP
       FRANC ? 78
       C.VF DEVISES ? 0
     BANQUE ET CORRESPONDANTS
       FRANC ? 2065 0
     CREDIT A LA CLIENTELE
       FRANC ? 21336 4271
     COMPTES DEBITEURS CLIENTS
       FRANC ? 6353 113
```

Fig. 11. The interactive data collection command (ENTER)

4.5. Questioning of the DB

4.5.1. Principle

After a user has defined a context, it is possible to ask for the printing or plotting of values. The plotting can be done on a CRT or KSR terminal, or on a graphic terminal or plotter.

In the following example, the user wants to retrieve all accounts of the right hand side of the balance sheet on branches 12 and 14 (AG12 and AG14), for the 4th term of 1980 ; French currency as well as in other currencies french equivalent. The user can modify parts of this context and retrieve information using the IMPRIMER (print) command. In our case he wishes also to print the list of branches of a certain region by decreasing order of their total assets.

```
BASE  ? CONTEXTE AGENCES AG12 AG13

BASE  ? CONTEXTE SERIE CAISSE A 'TOTAL ACTIF'

BASE  ? CONTEXTE TEMPS T 4 80

BASE  ? CONTEXTE 'NATURE DES FONDS'
```

```
BASE  ? IMPRIMER AGENCES TEMPS SERIE / 'NATURE DES FONDS' TRIE PAR TOTAL DECROISSANT



(AGENCES)
  (TEMPS)
    (SERIE)

 (NATURE DES FONDS)     =    FRANCS  C.V F/DE     TOTAL
                           --------  --------  --------

AG12
 T 4 80
  *TOTAL ACTIF             16000<     3897<     19897<
   SIEGE ACTIF              6072<     2680<      8752<
   CREDIT A LA CLENTELE     5334<     1068<      6402<
   DEBITEURS ET REGULARIS   2066<        0<      2066<
   COMPTES DEBITEURS CLIE   1588<       28<      1617<
   BANQUES ET CORRESPONDA    516<        0<       516<
   EFFETS RECUS EN RECOUV    150<      106<       255<
   CAISSE                    156<        7<       163<
   EFFETS A ENCAISSEMENT      92<        8<       100<
   BDF-CCP                    20<        0<        20<
   CREANCES DOUTEUSES          6<        0<         6<

AG13
 T 4 80
  *TOTAL ACTIF             19579<     5913<     25491<
   SIEGE ACTIF              7752<     3770<     11522<
   CREDIT A LA CLENTELE     6417<     1960<      8377<
   DEBITEURS ET REGULARIS   2469<        0<      2469<
   COMPTES DEBITEURS CLIE   2033<       58<      2091<
   BANQUES ET CORRESPONDA    479<        0<       479<
   EFFETS RECUS EN RECOUV    113<      106<       219<
   CAISSE                    178<        9<       187<
   EFFETS A ENCAISSEMENT     106<       11<       117<
   BDF-CCP                    22<        0<        22<
   CREANCES DOUTEUSES         11<        0<        11<
```

Fig. 12. Interrogation of the DB with a SORT (TRIE) option.
The nesting of entity type is obtained by the order of the entity type names in the IMPRIMER command.

Exception reporting can be performed using the TELQUE (such that) operator.
Example :
Print the accounts in french currency for branches AG12 and AG13 such that CV/F devises >0.

```
BASE  ?

IMP AGENCES TEMPS SERIE/'NATURE DES FONDS'TRIE PAR TOTAL DECR TEL QUE 'C.V F/DEVISE' >0
```

(NATURE DES FONDS) =	FRANCS	C.V F/DEVISE	TOTAL
AG12			
T 4 80			
*TOTAL ACTIF	16000<	3897<	19897<
SIEGE ACTIF	6072<	2680<	8752<
CREDIT A LA CLENTELE	5334<	1068<	6402<
COMPTES DEBITEURS CLIENT.	1588<	28<	1617<
EFFETS RECUS EN RECOUVREMENT	150<	106<	255<
CAISSE	156<	7<	163<
EFFETS A ENCAISSEMENT	92<	8<	100<
AG13			
T 4 80			
*TOTAL ACTIF	19579<	5913<	25491<
SIEGE ACTIF	7752<	3770<	11522<
CREDIT A LA CLENTELE	6417<	1960<	8377<
COMPTES DEBITEURS CLIENT.	2033<	58<	2091<
EFFETS RECUS EN RECOUVREMENT	113<	106<	219<
CAISSE	178<	9<	187<
EFFETS A ENCAISSEMENT	106<	11<	117<

Fig. 13. An interrogation of the DB with a and a sort.

Any level of consolidation can be performed since a consolidation is an aggregate of branches. For instance a consolidation of all branches will be defined as :

```
DEFINE TOT.BRANCHE = !SOM AG1 A AG40
```

4.5.2. Relation between OPTRANS and predicate calculus

In OPTRANS, data are looked upon as functions. For instance, if the user has defined four entity types
- entity type series (the set of names being denoted as S)
- entity type date (the set of names being denoted as T)
- entity type product (the set of names being denoted P)
- entity type distributors (the set of names being denoted D)

We have a function :

$v = DATA\ (s,t,p,d)$ with $s \in S \quad t \in T \quad p \in P \quad d \in D$

We can then use the system to request, for instance, all products whose margin is above 20 and sales below 100. The corresponding syntax in OPTRANS would be

```
BASE ? CONTEXT SERIE MARGIN, PRICE, QUANTITY
       CONTEXT TIME JAN 80
       CONTEXT PRODUCT
       CONTEXT DISTRIBUTOR HYPER
BASE ? PRINT TIME DISTRIBUTOR PRODUCT/SERIE SUCH AS MARGIN > 20 AND SALES < 100
```

Fig. 14. Interogation of the DB with a predicate.

which is equivalent in the usual predicate calculus form to :

$$\exists p : DATA\ (s,t,p,d) > 20 \wedge DATA\ (s,t,p,d) < 100$$

4.6. The problem of the evaluation of aggregates

This problem is due to the fact that certain aggregates are defined with entities from different entity types. In the process of their evaluation the "order" in which the entity type is taken into account is important.
This problem is difficult to solve if we wish to have a convenient system.
We can imagine to modify the system so as to imbed an axiom base in the system (and deduction theorem), in such a way that the axiom base translates the economic principles within which we wish to work. The trouble is that since we have a system with any user can name and define the concepts by himself, this is not feasible. The only way is to have the user introduce the axiom base himself. In the case where we have introduced the notion of private and public data the solution is clear : the axiom based is implemented for public data since the name of entities are the same for every one. For private data the user has to implement is own axiom base if different from the public (or system) one.

4.7. Modeling with OPTRANS

As indicated carlier the goal of the DSS in our example is to provide the branch manager with a tool to :
- model the computation of the income statement (IS),
- compute the forecasted value of the IS on a quarterly basis for the coming year,
- test the impact of different objectives of growth on the IS,
- test the impact of the quarterly structure of interest rate at branch level.

and the controler at HQ with a tool to :
- model the consolidation of all IS and balance sheets (BS) at branch level,
- compute the consolidation of IS and BS on actual results,
- model the consolidation of the forecasts over all branches,
- test the impact of a change in the interest rate structure on the consolidated results.

4.7.1. Model solution

It is possible, using a CALCULER command, to feed data from the DB and solve models. A model being solved, the user can request using the IMPRIMER command any value of the variables computed or used in the model :
The CALCULER (solve) command implies two actions :
- feeding the model work-space with the data as requested in the CONTEXT command,
- solving the model.

Once a first CALCULER command has been used it is possible to use a "CALCULER LOCAL" command, (for instance after a modification of the relations in the model). In that case it is not needed to feed the work-space.
Figure 15 gives an example.

```
 MODELE  ? CALCULER
QUELLE AGENCE ? AG12
DATE DU RAPPORT ? 11 JUIN 1982
CHOISISSEZ VOTRE PROGRESSION BLEU ? 0.15

CHOISISSEZ LES TAUX DE PROGRESSION  CREDIT CLIENT : FRANCS, DEVISES ? 0.12 0.11

CHOISISSEZ LES TAUX DE PROGRESSION  DEBITS CLIENTS : FRANCS,DEVISES ? 0.18 0.22

CHOISISSEZ LES TAUX DE PROGRESSION  COMPTES A VUE : FRANCS,DEVISES ? 0.19 0.15

CHOISISSEZ LES TAUX DE PROGRESSION  COMPTES A TERME :FRANCS,DEVISES ? 0.17 0.17

CHOISISSEZ LES TAUX DE PROGRESSION  COMPTE EPARGNE :FRANCS,DEVISES ? 0.13 0.13

CHOISISSEZ LE TAUX DE PROGRESSION ENGAGEMENT SIGNATURE ? 0.10
TAUX MOYEN DU MARCHE MONETAIRE ? 0.16
TAUX DE BASE BANCAIRE ? 0.18
```

```
TAUX D'EPARGNE ? 0.09
MARGE OPERATION DEVISE ? 0.30
TAUX DE PROGRESSION DU BENEFICE DE CHANGE ? 0.12

TAUX DE PROGRESSION DES COMMISSIONS ? 0.16

TAUX DE PROGRESSION FRAIS DE PERSONNEL ? 0.18

TAUX DE PROGRESSION FRAIS GENERAUX ? 0.16

TAUX DE PROGRESSION FRAIS ENTRE AGENCES ? 0.15

MONTANT DES AMORTISSEMENTS PREVUS ? 200
QUELS SONT LES FRAIS DIVERS PREVUS ? 150
```

Fig. 15. A run of the branch model with conversational selection of the context.

In figure 16 we show how a model is evoked in a specific context.

Instruction 10 means that the answer to the question concerning the name of the branch will be the CONTEXT of the entity type branch.

Instruction 55 means that the date given by the user will be the CONTEXT of the entity type TEMPS (TIME).

As a consequence it is possible to use this model for the same branch for another year, or for the same year for anothe branch...
The user will just change the context during the execution of the model.

```
1 BASE SIG2>BANQUE
5 QUESTION QUELLE AGENCE
10 REPONSE CONTEXTE 3
15 * CONSERVATION DU NOM DE L'AGENCE DANS &3
20 &3 CONTEXTE 3 =1
25 QUESTION DATE DU RAPPORT
35 REPONSE &1
36 ALLER 57
40 * DEMANDE DU CONTEXTE TEMPS
45 TEXTE POUR QUELLE ANNEE
50 QUESTION AN 80 AN 81
55 REPONSE CONTEXTE TEMPS
56 &2 CONTEXTE   3 =1
```

Fig. 16. How to evoke a model (showing the CONTEXT selection).

Editing and displaying in OPTRANS

The user can either use an instruction at the model level (automatic printing of the report) or use a print command at the report level.
An important aspect is that name of the branch (ie the name of the entity selected in the CONTEXT of the entity type BRANCH), and the date of the report are automaticaly transmitted from the model to the report (instruction 20 and 56 in fig. 16).
It is also possible to plot curves on alphanumeric or graphic terminal. The type of terminal being used is specified in a TERMINAL command.

```
CONTEXTE SERIE IPAP
CONTEXTE TEMPS JAN73 A DEC80
CONTEXTE PRODUIT SEIGLE
TERMINAL ECRAN 4025
TRACER POSTE PRODUIT/TEMPS
```

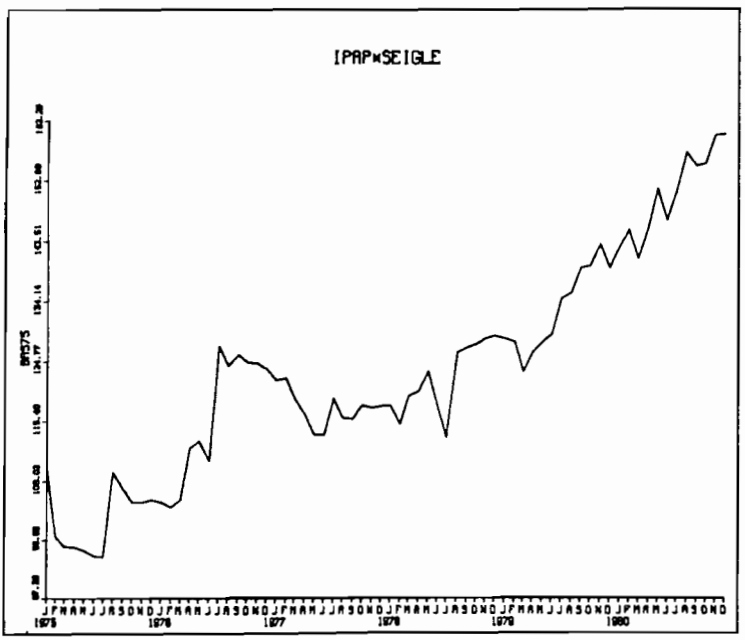


Fig. 17. Using a graphical terminal with the tracer command.

5. MAIN CHARACTERISTICS OF A DSS GENERATOR IN THE MANAGEMENT FIELD

5.1. The nucleus of a DSS generator

From the user's point of view, our conclusion is that a DSS generator in an organization should have the following capacities :

1) A data base approach.

We believe that if we have to face the problem of implementing a DSC (as described in §2), only a Data Base approach is able to fulfill the requirement. Our experience has been that a specialized system is needed for DSS applications ; such system sould in a multi-user environnement :

- provide : . a DDL at a conceptual level, and
  . a DML with the capacity to modify the syntax without reprogramming and use the concept of virtual data.
- in the spectrum of applications we have discussed in this paper, be mainly oriented to deal with numerical data (this helps in keeping the system on a modest size compared with a general DBMS).
- provide : . the capability to handle private as well as public information
  . an easy way to transfert information from different DB developed by different user groups (using the same DSS generator)
  . a good access control for each user (control by context)
  . a tool easily to implement specialized data capture interface

. a capacity for easily sorting (alphabetical or numerical)
. a capacity for filtering (first order predicate calculus)

2) An interactive report generator
Many DSS starting environments are made up of a Data Base with a good high level interactive tool to present information.
- the same report generator should be accessible from the DB as well as within the modelling subsystem,
- the report generator should be able to handle a mixture of graphical and tabular types of Data.

3) Graphical subsystems
- graphical subsystems for DSS seem to us rather simple (compared to what is needed in CAO),
- ability to handle a wide variety of plot and graphic terminals including color graphics seems useful.

4) A modelling subsystem
Our experience has been that two types of modelling languages are needed :
- a procedural "electronic sheet" type language,
- a non procedural (economic type) language.

It is a fact that if most financial and control managers like the "electronic sheet" approach to modelling, this approach is of no use with econometricians or management scientists as can be found in marketing, economics, or planning department.

5) A communication subsystem
This subsystem enables the users to create user groups (a so called electronic conference) with commands to send and read private as well as public messages.
The reading capability enables users to retrieve messages according to criteria such as : number, date, sender, period, content...
The service command of the system enable the user to accept a system such as OPTRANS and share it among them. In other way, OPTRANS is then imbedded in a communication system as described in Klein [26] . We call the nucleus of a DSS generator a software with the above 1 to 6 components.

5.2. The application subsystem

Experience and theory shows that the nucleus of a DSS generator must have a "plug in" capability for specialized software for each field of application.
The Marketing man expects to find readily available knowledge about his field under the form of specialized algorithms or models, so does the financial manager.
The fields we have identified are the following :
- statistical tools (explicative and descriptive methods),
- short term forecasting,
- financial models,
- management control models (consolidation algorithms)
- production models.

6. STRUCTURE OF THE SYSTEM AND DESIGN

6.1. Hierarchical structure of the nucleus

This structure is best presented on fig. 18.
The structure is simplified since the nucleus is made of 80 programs which constitutes a series of layers of software. The approach to derive this hierarchical structure is sketched in Klein, Tixier [1] .
The main layers are the following : the lowest layer comprises a series of primitives, such as :
- scanner, analyzer, input routine (NEWLINE), output routine (IMPLIGN), interrupt handler, data access function (EVALUATION), context function, message access function.

The second layer consists of a REPORT WRITER, a data base subsystem and a modelling subsystem.

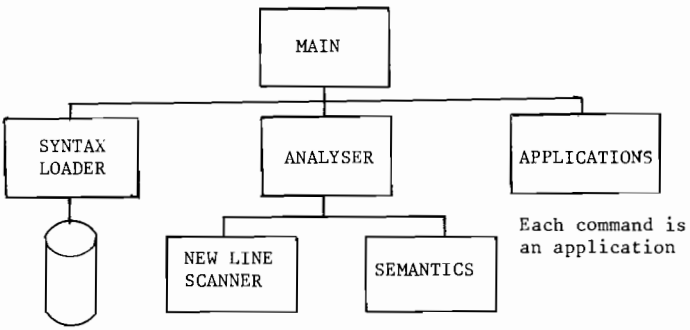


Fig. 18. Control structure of a layer in OPTRANS.

6.2. Design strategy

OPTRANS is designed around up three separate subsystems :
- a specialized DBMS,
- a modeling subsystem,
- a report generator (with graphical capabilities).

The question may be asked why it was decided to have a modeling subsystem related to the DBMS through a work space on disc, rather than a modeling capability integrated with the DBMS software.
The reasons are the following :
(1) we think that it is essential to be able to access several DB's from a model.
(2) we have not been able to introduce a change in context within the evaluation of an aggregate.
(3) although it is theoretically feasible to design an evaluation function for the set of relations that make up a model (including if needed the algorithm for solving a system of equations), it has seemed too complex to include such algorithms in the evaluation function.
(4) some of the evaluations of the aggregates are so complex that it is too time consuming to restart the evaluation whenever the model is interrogated or evaluated.

7. PRESENT STATE OF THE SYSTEM

The first version of OPTRANS was developed in 1978. The formal definition of this first version is given in [7] .
The present state of the nucleus of the system is described in [8] . The system is written in FORTRAN 77 and is available on PRIME computers under PRIMOS (virtual memory operating system) on any machine of the 50 series having a minimum of 512 k of central memory and on HB-66 biprocessor under DTSS, the latter computer being connected to TRANSPAC, EURONET and TYMNET. The nucleus has a size of 20 000 statements and runs in 256 k under DTSS (with overlays).
The system has been in use in organizations since the beginning of 1978 for version 1 and since June 1979 for version 2.

7.1. Implementation of in House DSC

OPTRANS has been used to implement DSC in several large organizations since 1979. In the banking industry several DSS's have been implemented and have led to improvements.

In the industrial sector, Marketing DSS have been developed as well as planning DSS's. In multinational organizations the control of subsidiaries frequently requires DSS for the purpose of consolidation and variance analysis at different levels.
In the public administration DSS's have been implemented for supporting the budgeting process of "municipalities". In the Education sector, OPTRANS is being used at C.E.S.A. and in several other Business Schools in a DSS-course.
Among the main problems found we point out :
- extraction of data from data processing files, (mainly in commercial applications),
- need for a more general relationship definition,
- improvement in performance,
- alternative syntax,
- improved control of access for users (access control by context),
- compatibility between different FORTRAN 77 compilers.

Most of these implementations have been done on 32 bits minicomputers such as PRIME with virtual memory. An attempt will be made however to transfer OPTRANS on a 16 bits minicomputer under the UNIX operating system in the future.

7.2. Network access and multilanguage usage

OPTRANS is being used on a host (HB 66 under DTSS) connected to TRANSPAC, EURONET, TYMNET and TELENET. The system has been used mostly to implement DSS's applying large economic, financial or marketing data bases. Due to the different countries of origin (PORTUGAL, US, GERMANY) of the users we have been led to adapt the command language and assistance messages to the national language of the users. This adaptation is all the more simple as the syntax analyzer is table driven and assistance messages are not in programs but in outside files on disc. The following extensions are the major current developments.

7.3. Present development

7.3.1. Graphics

Currently it is possible to use an external graphic package using standard alphanumeric terminals (DISSPLAS). The work in process is aiming at developing a specialized graphical subsystem for DSS applications. The first graphics terminal to supported is the tektronix 4025, 4010 and the corresponding plotter.

7.3.2. Group support : a computer-assisted teleconferencing function

We are currently facing two problems with respect to communication :
- some of the decision problems are problems involving people geographically dispersed,
- user groups of a size of 10 to 20 persons are now using the same DSS and feel the need to exchange their experience.

We have already experimented the use of computer-assisted teleconferencing in such situations (see [6]). Such a component will in the near future be an important part of a DSS generator. This function can be implementated in relation with the access control mechanism.
A user of the teleconferencing system can easely access OPTRANS. Conversely he can leave OPTRANS to enter the teleconferencing system.

7.3.3. Improvement in performance

Some OPTRANS applications lead to the development of large data bases. For example, a marketing DB for 1000 products(1), 10 elementary variables per product and 100 dealers, all information being kept for 5 years on a monthly basis will include 60 millions of numerical values or 240 Mo.

The access time to an elementary piece of information is independent of the number of products and is very fast. However the evaluation of an aggregate referring to a product, with a condition predicate computed over even a subset of the products can be very resource-demanding on the computer and still be very simple to express. We are now doing research on how to minimize the cpu time in such situations.

8. ACKNOWLEDGMENTS

We would like to thank Mr. RUGGIU for his advice on the predicate calculus aspect of OPTRANS, Mr. POUSSARD for his help in translating a french version of the paper, Mrs SAILLANT, BOEFFARD and RICO in typing the manuscript.

BIBLIOGRAPHY

[1] Michel KLEIN, Vincent TIXIER, SCARABEE : A data and model bank for financial engineering and research IFIP 1971.

[2] Frank A. MANOLA, Data base technology in decision support systems : An overview. Paper presented at the task force meeting on decision support systems, June 23.25, 1980, IIASA, Luxembourg, Austria.

[3] Eric D. CARLSON, "An approach for designing decision support systems", data basen (10,13), Winles 1979.

[4] Leif B. METHLIE, Data management for decision support systems data base, Vol. 12, Number 1/2 Fall 1980.

[5] Francis GIRAULT, Michel KLEIN, Le projet SCARABEE de banque de données et de modèles pour la recherche en analyse financière et en gestion de portefeuille. ANALYSE FINANCIERE 3ème trimestre 1971.

[6] Michel KLEIN, FINSIM : A decision support system for financial planning and engineering. Information processing, Gilghrist Ed. North Holland 1977.

[7] Philippe GROSJEAN, Michel KLEIN, Alain MANTEAU, OPTRANS : Un langage pour la définition et la conception de systèmes d'aide à la décision. Unpublished document 1978. Prepared for the IFIP working conference on formal models and practical tools for information system design.

[8] Michel KLEIN, Alain MANTEAU, Michel GINESTE, OPTRANS user manual, SIG, 4bis, rue de la libération, 78350 Jouy-en-Josas, June 1981.

[9] EXPRESS User manual, Management decision systems, Inc. Riverside Roal Weston Massachusetts, 02193, U.S.A.

[10] FOCUS User manual, Tymshare INC.

[11] IFPS, Tutorial Execucom P.O. Box 9758, Austin Texas 78766 U.S.A.

[12] David W. SHIPMAN, The functional data model and the data language daplex, TODS (Mars 1981) volumes 6, number 7.

[13] John D.C. LITTLE, Models for managers a calculus of decision, Management Science, vol 16, n° 8, Avril 1970.

[14] John D.C. LITTLE, Decision support systems for marketing managers, Journal of marketing, vol 43 (summer 1979) p. 9-26.

[15] Michel KLEIN, Alain MANTEAU, Joseph RAPHAEL, An investment and financial planning system for the road Transport Industry. Engineering and process economics, 4 (1979) p. 193-210.

[16] Peter G.W. KEEN, Michael S. SCOTT MORTON, Decision support systems an organizational perspective, Addison-Wesley 1978.

[17] Dave KAHL, J.C. PERRIN, Michel KLEIN, Alain MANTEAU, "Un système d'information et d'aide à la décision pour le contrôle financier des sociétés multinationales" - Editions Hommes et Techniques, Actes du Congrès Modélisation et Maîtrise des Systèmes, tome 2 p. 178-188.

[18] Michel KLEIN, Systèmes QR et aide à la décision en analyse financière, International seminar on intellignet question-auswering and data base systems, Bonas (Gers) June 21st-30th, 1977 IRIA.

[19] Michel KLEIN, J.P. LEVY, SCARABEE : Un langage d'aide à l'analyse financière, 01-Informatique Novembre 1974.

[20] Michel KLEIN, Alain MANTEAU, ANALYS : Un système intéractif pour l'analyse des données, Colloque IRIA, Analyse des données et Informatique.

[21] Alain MANTEAU, ANALYS : User manual. SIG, 4bis, rue de la libération, 78350 Jouy-en-Josas.

[22] David B. MONTGOMERY and Glen L. URBAN, Management Science in Marketing, Prentice Hall, 1969, p. 18.

[23] Ralph H. SPRAGUE Jr, Overview of decision support systems, IIASA, Laxenburg Austria, June 23-25 1980, Task Force Meeting on Decision Support Systems.

[24] Michel KLEIN, Jean-Pierre LEVY, SCARABEE : Un langage pour l'aide à l'analyse financière, 01-Informatique n° , 1974.

[25] Peter CHEN, The entity-relationship model - Toward a unified view of data. ACM Transaction on Data Base Systems, vol. 1, n° 1, March 1976.

[26] Alain MANTEAU, Michel KLEIN, Using a teleconferencing system in connection with a DSS in decentralized groups, unpublished paper.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

INTEGRATED DATA ANALYSIS AND MANAGEMENT SYSTEM: An APL-Based Decision Support System

J. William Bergquist
Los Angeles Scientific Center
IBM Corporation
Los Angeles, California

Ephraim R. McLean
Graduate School of Management
University of California
Los Angeles, California

The Integrated Data Analysis and Management System (IDAMS) is an APL-based system designed to support business and scientific end users. Although there are a number of references to the use of APL as a DSS *tool*, IDAMS is one of the first examples of an APL-based DSS *generator*. It was developed at the IBM Scientific Center in Heidelberg, Germany, and is now undergoing testing and further development in the United States in a joint project between the IBM Scientific Center in Los Angeles and the University of California, Los Angeles (UCLA). This paper describes the design objectives and structure of IDAMS and illustrates its use with a sample terminal session.

INTRODUCTION

Recently, distinctions have been made by Sprague (1981) and others between decision support systems (DSS) and the tools and languages used to build such systems. Because the end user must be considered an integral part of a DSS, the computer-based component can only be just that -- a component. By this definition, decision support systems cannot be generic; they will always be user specific. From this perspective, therefore, Execucom's IFPS (Interactive Financial Planning Systems), a widely used computer-based planning and modelling package, is not a DSS *per se*, but a sophisticated tool -- or "DSS generator" in Sprague's terms -- from which many DSSs can be built. IFPS, in turn, is itself constructed in FORTRAN, a "DSS tool."

It would be possible, of course, to build a DSS directly in FORTRAN -- possible, but certainly costly and time-consuming. The use of any third-generation high-level language (e.g., FORTRAN, PL/1, Pascal), with the corresponding need to specify the problem to be solved in step-by-step procedures, places a very high threshold cost on the user in undertaking the development of a DSS. Considerable technical expertise is usually required; and, in most cases, this calls for enlisting the aid of outside technical specialists. With the introduction of these programmer intermediaries comes all of the problems of communication, interpretation, mutual understanding, and so forth (see McLean (1979) for a further discussion of these issues). Of course, such systems *have* been built. The early work of Scott Morton (1971) and Gerrity (1971) attest to the fact that it is possible to construct decision support systems directly from third generation procedural languages. But these DSS projects were massive "labors of love," undertaken by their builders as subjects for their Ph.D. dissertations. If the huge time commitments that these systems required were still the norm, it is doubtful that much progress in DSS would have been made to date. Fortunately, such large expenditures of effort are no longer necessary.

Among the developments of the last decade are the introduction of more powerful tools to aid in the development process. For instance, Keen (1976), McLean and Riesing (1981), and others have suggested that the programming language APL offers significant advantages over other procedural languages in the rapid development of decision support systems. On the other hand, Gerry Wagner, the developer of IFPS, has argued that fourth-generation, non-procedural tools like IFPS (or FOCUS or VisiCalc) are much better vehicles for building DSSs.

Of course, comparing APL to IFPS, as to which is better for building decision support systems, is a little like comparing apples and oranges. To use Sprague's terminology, APL, like FORTRAN, is a DSS *tool*; IFPS is a DSS *generator*. The appropriate comparison would be between FORTRAN and APL, as tools to build generators, and between FORTRAN-based IFPS and an equivalant APL-based DSS generator. On the former question, concerning DSS tools, most would agree that APL has a clear advantage over FORTRAN; but what of the latter? *Is* there an APL-based DSS generator to compare with other DSS generators? It could be argued that ADRS (A Departmental Reporting System) or APL-DI (APL Data Interface), two widely used IBM program products, might qualify for this distinction. However, their functionality is somewhat limited. Also, they were not designed to work together; and most users want data access and report generation to be integrated together under a common architecture.

In recognition of the need for an integrated approach, the IBM Scientific Center in Heidelberg, Germany, has been working on IDAMS, an experimental APL-based end-user-oriented system, for several years (1980). IDAMS, standing for Intergrated Data Analysis and Management Systems and designed to support both scientific and business end users, has been tested in several governmental and university sites throughout Germany. In the spring of 1981, a version of IDAMS was brought to the University of California, Los Angeles, for further testing and development under a joint study agreement between IBM and UCLA. As of this date, this two-and-a-half-year study, staffed by technical experts from both organizations, is approximately one-third complete. This paper is intended to provide a description of the features of IDAMS, some discussion of its design philosophy, and an illustration of a typical session at the terminal.

AN OVERVIEW OF IDAMS

Large quantities of data are being accumulated in both science and industry. These data may be needed for a variety of applications, whether purely scientific, like experiments in physics, engineering, or medicine, or business oriented, such as in forecasting, resource planning, or financial analysis. The detailed analysis and non-routine usage of such data by technical or business end users, most of whom have little or no programming skills, creates heavy demands for

- a powerful interactive language
- a flexible data base management system
- comfortable and versatile report formatting
- easy embedding of application-specific algorithms
- rich facilities for graphical display of data
- application-oriented descriptions of data and programs
- on-line usage information about the system, the programs, and the data.

IDAMS combines several components to address these requirements. As discussed earlier, it has been largely implemented in APL. It provides an interactive problem solving environment suitable for programmers as well as for non-programmers. End users must be knowledgeable in their application; but they need not necessarily be knowledgeable about computers, for special programming skill is not required to use IDAMS. However, some familiarity with APL, particularly desk calculator mode, will be helpful. The virtues of APL for interactive problem solving have been enhanced by giving access to a data base management system, a program library, and an information component, the latter being used to guide the user in the selection and usage of the data and programs and to aid him in using the system.

In IDAMS, the user views his data as a collection of interrelated tables, a data base with a relational data model as its external view. The menus and descriptions of the data and programs, what is called the "inventory data" in IDAMS, consist of tabular (formatted) and non-tabular (unformatted) information. The formatted part serves primarily for the identification and definition of these tables, while the unformatted part supports the selection of the data and programs.

These inventory data appear as an information network with facilities to guide the user to those nodes that contain the information he is seeking. The nodes of the network carry the

actual information, while the arcs express an order relationship, associating one node to the nodes that succeed it. This allows the user to navigate from general menus to more specific ones. By narrowing down the search context through successive selections of menus, one arrives at the node containing the information the user is seeking.

Figure 1 provides a diagrammatic overview of IDAMS. Typically, a user will set up a "query," i.e., his specific data analysis or manipulation requirement, by using the guidance front end to identify the tables and programs required to perform his processing; and then he will use the data analysis front end to set up and execute this query. On-line training is provided for new users to acquaint them with the system's facilities. The user guidance features make it possible to develop an extensive repertoire of user-tailored solution steps customized to the needs of specific applications. The system is thus not application dependent in any way. Instead, it offers facilities for the inclusion of application-specific information to achieve customization. Information about the system, as well as about data and programs, may be consulted within a uniform language interface.

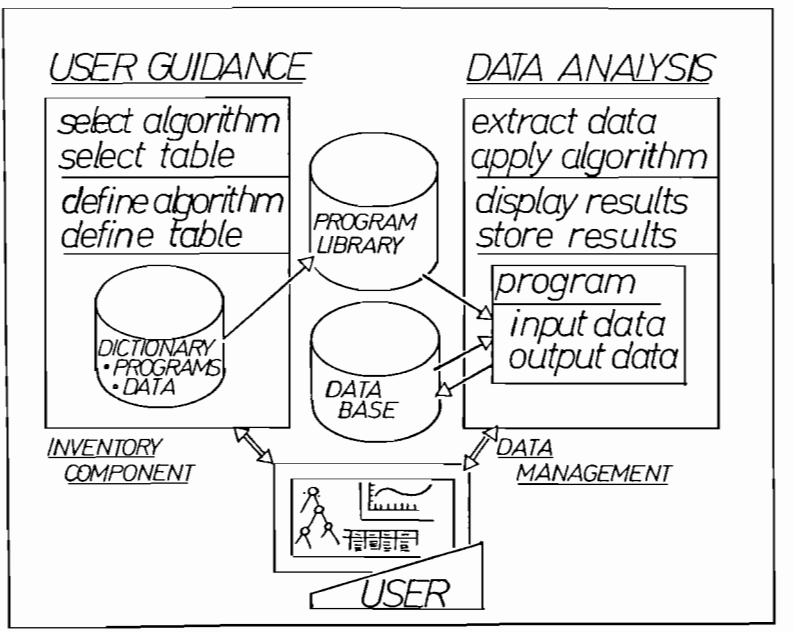


Figure 1

A problem solver's needs for information are illustrated in Figure 2. Let's take the example of a business planner. When this planner wishes to use some forecasting techniques, he may first need some general information about forecasting techniques in order to guide his selection from the set of available programs. He then needs assistance on how to use the particular program of his choice. He must also understand the nature of the data available to him on which he wants to base his forecast. To be effective, all this information must be maintained by the system and available to him on request. To do this, IDAMS maintains three interrelated sets of data:

- Algorithms (the program library, e.g. APL, FORTRAN, PL/1, or Assembler procedures)
- Problem data (raw data as well as extraction data)
- "Inventory data" (descriptive information about data and programs)

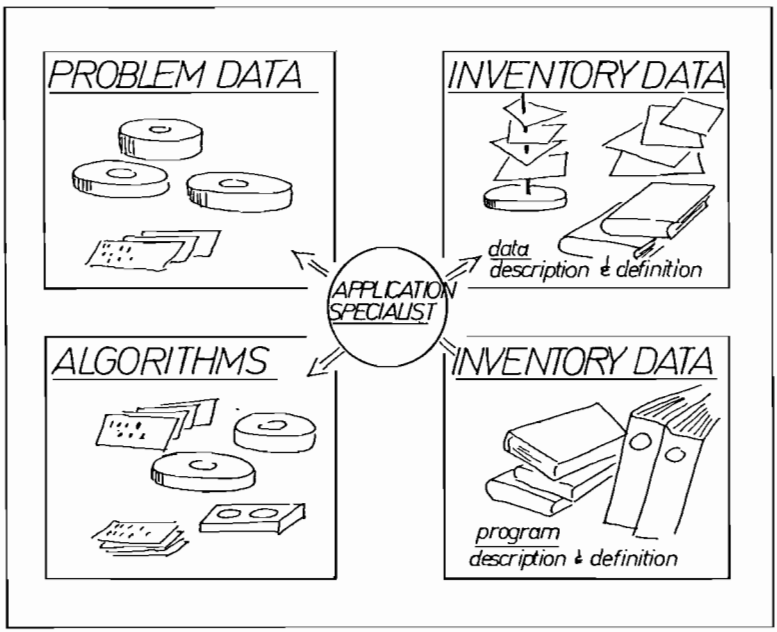


Figure 2

Tables, views, and programs as used in IDAMS need some definition, that is, a description of the names and types of columns or arguments. These definitions (i.e., the formatted part of the inventory data) form the so-called data dictionary. For adaptation to a user's specific needs, the IDAMS objects (i.e., tables, views, and programs) may be put into the user's active environment. If this is done, it means that he will find them automatically accessible in every session. In addition to the user's active environment, a directory is also maintained which describes the objects he owns privately and the objects to which he has been granted access. This is shown in Figure 3.

Classification of IDAMS objects is by ownership, by authorization, and by permanence. In this regard, IDAMS objects may be either temporary or permanent. Temporary objects get lost at the end of a session, if they are not changed into permanent objects. A user's active workspace may encompass both temporary and permanent objects. Each of these will have an entry in the active dictionary. For temporary objects, this entry is generated during the session; while for permanent objects, it is copied from the permanent dictionary. The difference between permanent and temporary objects is transparent within a session as far as the syntax of the data analysis is concerned. User guidance, however, is only provided for permanent objects. Insertion of a permanent object therefore implies that the object becomes embedded into the information network of the inventory component.

The documentation of an object may involve text insertions at various points of the information network, describing the object and its role in various levels of detail. The format of the description is only partially free form, in order to support specific needs; e.g., the description of a table must give information about the table and each of the table's columns. However, in addition to the necessary syntactically oriented information, descriptive information is also possible.

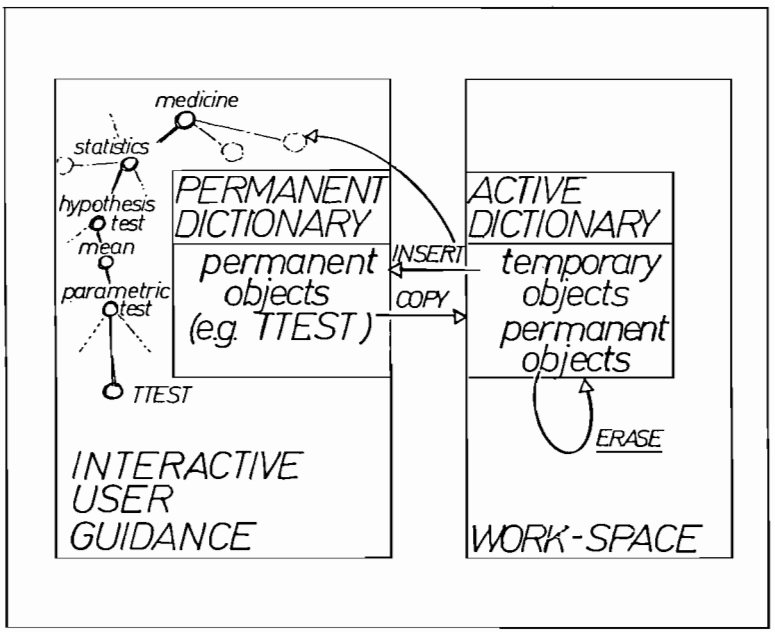


Figure 3

Figure 4 further illustrates that the integration process of developing application programs involves both the update of the program library and of the inventory data component. In addition to APL programs, non-APL programs may also be inserted into the library by using the particular compiler required. However, this task must be done by an application programmer, not by the end user. This person would also be responsible for writing a description of the program's call interface; i.e., the specification of the input and output parameters.

To support the identification, selection, and usage of a particular program, some descriptive information about the program must be inserted in the information network. This would also be done by an application specialist who would know about the potential usefulness of the program for various applications and about the information needs of the prospective end users. The use of the program by end users is then based on this syntactical description in the call interface, supplemented by textual description where necessary.

The system's architecture, shown in Figure 5, illustrates that the interface between VSAPL, in which IDAMS is written, and PL/1, Assembler, or FORTRAN programs, plays a central role in the system. To provide this interface, two previously developed programs, Interactive User Guidance (IUG) and High Level Query (HLQ), are used; however, other interfaces built upon the existing ones are also possible. More experienced users may use the menu level and the command level of IDAMS interchangeably. A variety of IDAMS commands can be used directly instead of accessing them through the menus. This is particularly important when computer response time may be sluggish or a large number of menu screens have to be paged through until the desired one is obtained.

The system can be extended by introducing user-generated commands by adding the pertinent commands in APL. To do this, however, requires a knowledge of both APL and IDAMS. This

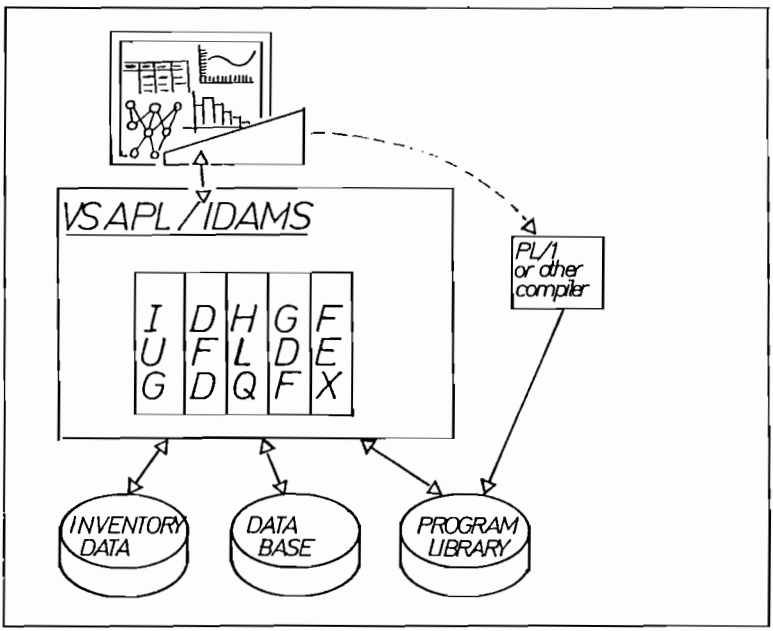


Figure 4

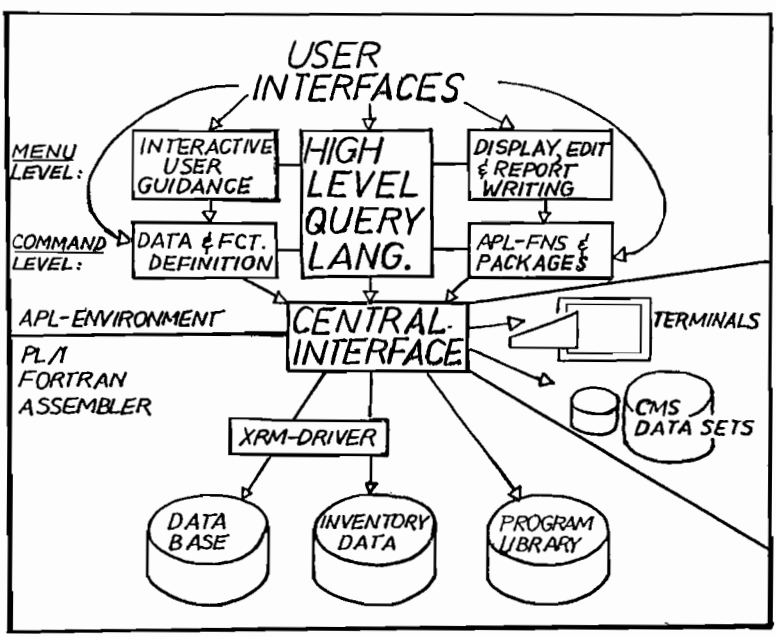


Figure 5

extensibility principle also applies to the high level query language. Special application-specific data manipulation requirements, beyond such simple things as SUM, AVERAGE, COUNT, etc., can be made available through specially written APL functions. The interface to such functions is either the standard APL call procedures or a table-like function presentation.

Although IDAMS has been implemented largely in APL, the data base management system resides in a non-APL environment. Currently, two data base management systems are supported: Extended Relational Management System (XRM) (1974) or System R (1976). As with the programs written in PL/1 or other non-APL languages, the interface to the data base is carefully managed so that users can conveniently move back and forth between the two environments.

To provide the reader with some feeling for using IDAMS, a sample terminal session is included in the Appendix. It shows how a user with little or no knowledge of IDAMS might obtain plots of stock quotations of firms listed on the German Stock Exchange. The displays shown in the Appendix are taken directly from a Tektronix display device which is used in conjunction with an IBM 3277 terminal. On the right side of each display is a description of the activity shown on the left.

IDAMS is currently being used in several applications in Germany for actual problem solving. Logging of usage patterns is built into the system to allow for subsequent evaluation. The analysis of these usage patterns in real-life applications may help not only to pinpoint possible future improvements in IDAMS but also to improve the general understanding of human problem solving. The "human factors" are a critical aspect of any decision support system and it is here that most of the current developmental work in IDAMS is being done.

SUMMARY

IDAMS is an experimental system; and, as such, it is still undergoing development. Although designed for use by non-programmers, it is an even more powerful tool in the hands of an APL programmer. In a way, IDAMS may be viewed as an enhancement to APL in several respects:

- It provides access to an underlying non-APL data base system (currently XRM, but this could be replaced by System R without changing the user interface).
- It provides efficient access to non-APL programs and thus opens new opportunities for the use of APL for interactive data analysis in combination with existing libraries of compiled programs.
- It provides an easy-to-use means of data analysis and management for non-programmers via its non-procedural language, which is translated into executable APL.

The major differences between IDAMS and many other query languages and implementations are:

- IDAMS supports a data base with a tabular data model, supporting array structures on the field level; i.e., a table entry may be a scalar, vector, matrix, or even an array of higher dimensions, whatever is most appropriate.
- IDAMS offers a guidance component which enlarges the facilities of the data dictionary through descriptively oriented information.
- The high level query language of IDAMS derives its power from the possibility to draw on the full data manipulation facilities of APL within a query.
- IDAMS allows existing program libraries to be embedded into the system, thus providing for customization of program libraries as well as of the high level query language.

IDAMS is not a perfect DSS generator. Its many features, some of which are quite powerful, are not all uniformly easy to use. Similarly, although a knowledge of APL is not needed to use IDAMS, there are many occasions when such knowledge is of considerable advantage. Finally, there are many specific applications which a user may wish to undertake which are not possible without the direct intervention and assistance of a skilled IDAMS systems expert.

However, all of these issues are currently being investigated; and, by the end of the 30-month IBM-UCLA joint study, it is expected that a number of changes and improvements will have occurred.

ACKNOWLEDGEMENTS

Many people have contributed ot the development of IDAMS, especially the members of the staff at the Heidelberg Scientific Center. In particular, the authors would like to acknowledge the contributions of A. Blaser, L. Dumke, H. Eberle, R. Erbe, R. Hartwig, H. Lehmann, G. Mueller, U. Schauer, and H. Schmutz at Heidelberg; and D. Cummings at UCLA.

REFERENCES

1. Chamberlin, D., Relational data base systems, *Computing Surveys*, 8, (March 1976), 43-66.

2. Erbe, R., *et al.*, Integrated data analysis and management for the problem solving environment, *Information Systems*, 5, (1980), 273-285.

3. Gerrity, Thomas, P., Jr., The design of man-machine decision systems: An application to portfolio management, *Sloan Management Review*, 12, (Winter 1971), 59-75.

4. Keen, Peter G.W., Computer systems for top managers: A modest proposal, *Sloan Management Review*, 18, (Fall 1976), 1-17.

5. Lorie, R.A., XRM: An extended (n-ary) relational memory, *IBM Technical Report*, 320-2096, (January 1974).

6. McLean, Ephraim, R., End users as application developers, *MIS Quarterly*, 3, (December 1979), 37-46.

7. McLean, Ephraim R. and Riesing, Thomas F., Installing a decision system: Implications for research, *Decision Support Systems: Issues and Challenges*, International Institute for Applied Systems Analysis Proceedings, Pergamon Press, Oxford, (1981), 127-141.

8. Morton, M.S. Scott, *Management Decision Systems: Computer Based Support for Decision Making*, Division of Research, Graduate School of Business Administration, Harvard University, Boston, Mass., (1971).

9. Sprague, Ralph H., Jr., A framework for research on decision support systems, *Decision Support Systems: Issues and Challenges*, International Institute for Applied Systems Analysis Proceedings, Pergamon Press, Oxford, (1981), 5-22.

A Sample Terminal Session

```
                                                    DISP1
OLD SELECTION(S) =
NEW SELECTION ==> ?4
 I D A M S - M E N U - INTERFACE
 1.MENU       - START OR RESUME MENU MODE
 2.INIT       - GOTO THE INITIAL MENU
 3.INDEX      - DISPLAY THE GLOSSARY (INDEX)
 4.SET        - CHANGE THE WORKING ENVIRONMENT
 5.TRAIN      - USE THE ONLINE TRAINING COMPONENT*
```

Your previously stored profile causes you to enter the IDAMS master menu after log on and password specification. You make your selection by entering a number or an unique character string. The beginning user would probably start with the training component to get used to the system (Option 5). If you want more information on a specific menu item, IDAMS provides a HELP option, activated by typing a question mark followed by the menu number or letter.

```
                                                    DISP1
OLD SELECTION(S) =
NEW SELECTION ==>
                        THIS PATH IS OF INTEREST
 1.MENU                 ONLY IF THE ENVIRONMENT HAS
 2.INIT                 TO BE CHANGED (E.G. FROM
 3.INDEX                TERMINAL TO TYPEWRITER).
 4.SET
 5.TRAIN
```

```
                                                    DISP1
OLD SELECTION(S) =
NEW SELECTION ==> M
 I D A M S - M E N U - INTERFACE
 1.MENU       - START OR RESUME MENU MODE
 2.INIT       - GOTO THE INITIAL MENU
 3.INDEX      - DISPLAY THE GLOSSARY (INDEX)
 4.SET        - CHANGE THE WORKING ENVIRONMENT
 5.TRAIN      - USE THE ONLINE TRAINING COMPONENT*
```

Assuming we have some familiarity with IDAMS, we won't use the training component (Option 5) and start instead with Option 1 to create a query.

```
                                                                DISP2
OLD SELECTION(S) =
NEW SELECTION ==> G
 I D A M S - M A I N - MENU
 1.SELECT     - UNIT, SKELETON, OR QUERY (PREPARATION)
 2.FILLIN     - MANIPULATE THE CURRENT/LOADED QUERY IN MENU-FORM
 3.EXECUTE    - COMPILE THE CURRENT/LOADED QUERY IN LINEAR -FORM
 4.TRACE      - MANIPULATE THE LOADED QUERY IN TRACE - FORM
 5.GUIDE      - UTILIZE THE GUIDANCE SYSTEM *
```

This advances us to the IDAMS main menu where we select Option 5 (GUIDE) to start searching for objects (data, methods) which may be useful. The guided tour through the information network starts and ends here and we will return to this point after identifying all objects of interest.

```
                                                                DISP3
OLD SELECTION(S) =GUIDE
NEW SELECTION ==> S
 STARTING POINT
 1.START      - OFFERS ROOT NODE TO START THE GUIDED TOUR IN NETWORK "IDAMS"
 2.INPUT NODE - ENTER NAME OF THE NODE TO ACCESS DIRECTLY IN NETWORK "IDAMS"
```

This is the root of the information network. You can fetch information directly by entering a node name (Option 2) or by starting a guided tour through the net (Option 1), which is what we have chosen.

```
                                                                DISP4
OLD SELECTION(S) =GUIDE.START
NEW SELECTION ==> S

 1.STEPWISE   - SEARCH FOR INFORMATION BY A GUIDED TOUR THROUGH THE NETWORK
 2.NODE TEXT  - DISPLAY COMPLETE TEXT OF THE CURRENT NODE
 3.NETWORK    - TO OFFER AN OUTLOOK, AND/OR TO PICK UP INFORMATION RANDOMLY
```

Every node may contain several pages of description text (Option 2). A quick search is possible by looking at the display of the whole net (Option 3).

```
-----------------------------------------DISP5--
OLD SELECTION(S) =GUIDE.START.STEPWISE
NEW SELECTION ==> S
 SELECT SUCCESSORS OF CURRENT NODE:     2 IDAMS
 1.SEARCH-NODES
 2.TECHNICAL-NODES
-------------------------------------------------
```

Note that the node names and their relationships are application dependent. IUGS (Interactive User Guidance System) provides a facility for the application developer to enter node names, texts, and the relationships between nodes; IUGS then generates menus with that information.

```
-----------------------------------------------DISP6--
OLD SELECTION(S) =GUIDE.START.STEPWISE
NEW SELECTION ==> 1
 SELECT SUCCESSORS OF CURRENT NODE:     3 SEARCH-NODES
 1.IDAMS-DATA
 2.IDAMS-FUNCTIONS
-------------------------------------------------
```

Here we wish to find out about data currently available in IDAMS. Because the minimum unique character string would require typing 7 letters, we will simple use the option number.

```
-----------------------------------------------DISP7--
OLD SELECTION(S) =GUIDE.SEARCH.STEPWISE
NEW SELECTION ==> 1
 SELECT SUCCESSORS OF CURRENT NODE:    10 IDAMS-DATA
 1.PORTFOLIO-DATA
 2.PERFORMANCE-DATA
 3.HORSE-RACE-DATA
 4.OPTION-MARKET-DATA
-------------------------------------------------
```

There are 4 data bases currently available to this user of the system and we select the one on stock portfolios.

```
-----------------------------------------------DISP8--
OLD SELECTION(S) =GUIDE.START.STEPWISE
NEW SELECTION ==> 2
 SELECT SUCCESSORS OF CURRENT NODE:    42 PORTFOLIO-DATA
 1.NAME
 2.STOCK
 3.DATE
-------------------------------------------------
```

Here we further specify the stock option.

```
                                                              DISP9
** STOCK ** PAGE 1 OF 1 /PF7=↓, PF8=↑, PF3=SCROLLEND

STOCK:
  STOCKQUOTATIONS FROM 1971 UP TO 1976.
  THE VALUES OF ONE COMPANY ARE GIVEN
  IN FORM OF A TIME SERIES I.E. IN A
  SEQUENCE OF SUCCESSIVE QUOTATIONS.
  ADJUSTED AND RAW RATES CORRESPOND TO
  EACH OTHER ELEMENTWISE.
SHORTΔNAME:
  THREE CHARACTER ID OF COMPANY, E.G.'BAY'
RAWΔRATES:
  ACTUAL STOCK QUOTATIONS (TIME SERIES)
ADJUSTEDΔRATES:
  ADJUSTED QUOTATIONS TAKING INTO ACCOUNT
  DIVIDENDS, SPLITS ETC. DURING 1971 .. 1976
```

This is a display of an end node. The end nodes are object nodes containing information about data (see example) and methods. Other nodes are navigation nodes. This information is important in helping the user combine data and methods in subsequent analysis.

```
                                                                                DISP10
OLD SELECTION(S) =GUIDE.START.STEPWISE
NEW SELECTION ==> 7
 END NODE REACHED:  STOCK !
 1.MAIN OPTION- GO BACK TO THE IUGS- OR IDAMS-MAIN-MENU
 2.NODE TEXT  - DISPLAY COMPLETE TEXT OF THE CURRENT NODE
 3.SEARCH PATH- DISPLAY THE SEARCH PATH CREATED BY A GUIDED TOUR
 4.ALL PATHS  - ALL PATHS INCLUDING SEARCH PATH LEADING TO THE CURRENT NODE
 5.NETWORK    - TO OFFER AN OUTLOOK, AND/OR TO PICK UP INFORMATION RANDOMLY
 6.INPUT NODE - ENTER NAME OF THE NODE TO ACCESS DIRECTLY IN NETWORK
 7.IDAMS EXIT - DIRECT EXIT TO THE IDAMS-PROCESSING-COMPONENT
```

There are several other options which could be used to continue the search for useful objects (Options 1, 5, & 6). We will leave the information network with Option 7 to get back to the main menu.

```
                                                              DISP2
OLD SELECTION(S) =
NEW SELECTION ==> S
 I D A M S - M A I N - MENU
 1.SELECT     - UNIT, SKELETON, OR QUERY (PREPARATION)
 2.FILLIN     - MANIPULATE THE CURRENT/LOADED QUERY IN MENU-FORM
 3.EXECUTE    - COMPILE THE CURRENT/LOADED QUERY IN LINEAR -FORM
 4.TRACE      - MANIPULATE THE LOADED QUERY IN TRACE - FORM
 5.GUIDE      - UTILIZE THE GUIDANCE SYSTEM *
```

The next step is to set up the current query, the SELECT Option. We will select a subset of the objects, which have been identified via IUGS.

```
                                    DISP11
OLD SELECTION(S) =SELECT
NEW SELECTION ==> S

 1.UNIT       - DISPLAY AND/OR DEFINITION
 2.SKELETON   - SELECTION AND/OR DEFINITION
 3.QUERY      - SELECTION AND/OR EDITING
```

The SKELETON Option is used to define the full query.

```
                                                              DISP12
OLD SELECTION(S) =SELECT.SKELETON
NEW SELECTION ==> T

 1.TAKE       - INTO THE CURRENT SELECTION (FROM ACTIVE)
 2.DROP       - FROM THE CURRENT SELECTION
 3.INSERT     - INTO THE ACTIVE DICTIONARY
 4.ERASE      - FROM THE ACTIVE DICTIONARY
 5.COPY       - INTO THE ACTIVE DICTIONARY (FROM PERMANENT)
 6.SAVE       - INTO THE PERMANENT DICTIONARY (FROM ACTIVE)
 7.FILLIN     - ENTER QUERY FILLIN-MODE WITH CURRENT SELECTION
```

The TAKE Option shows the currently available data tables and methods.

```
                                                                                DISP13
OLD SELECTION(S) =SELECT.SKELETON.TAKE
NEW SELECTION ==> 2 3 9

  1. DENSITY |FUNCTION  |MATRIX
  2. NAME    |SHORT     |LONG       |INDUSTRY∆CODE
  3. STOCK   |SHORT∆NAME|RAW∆RATES  |ADJUSTED∆RATES
  4. NORML   |SIGMA     |X          |→PDF
  5. LOGIT   |SIGMA     |X          |→PDF
  6. VEC     |X         |→R
  7. WALD    |SIGMA     |X          |→PDF
  8. PRINT   |OUTLINE
  9. HPLOT   |HNUM      |HTYP       |HGRAF          |HLAB|HTIT|HTICS|HLOGA|HSCAL|HSIZE
 10. HMPLOT  |HNUM      |HTYP       |HGRAF          |HCV |HLF |HHOR |HVER |HTIT
```

To plot the raw and adjusted growth rates for all companies, we select two tables (2 & 3) and a plot function (9).

```
                                                                                DISP13
OLD SELECTION(S) =SELECT.SKELETON.TAKE
NEW SELECTION ==>

  1. DENSITY |FUNCTION  |MATRIX
  2.×NAME    |SHORT     |LONG       |INDUSTRY∆CODE
  3.×STOCK   |SHORT∆NAME|RAW∆RATES  |ADJUSTED∆RATES
  4. NORML   |SIGMA     |X          |→PDF
  5. LOGIT   |SIGMA     |X          |→PDF
  6. VEC     |X         |→R
  7. WALD    |SIGMA     |X          |→PDF
  8. PRINT   |OUTLINE
  9.×HPLOT   |HNUM      |HTYP       |HGRAF          |HLAB|HTIT|HTICS|HLOGA|HSCAL|HSIZE
 10. HMPLOT  |HNUM      |HTYP       |HGRAF          |HCV |HLF |HHOR |HVER |HTIT
```

The system marks the selections with a X on the left, which means they are ready to be used.

```
                                                              DISP12
OLD SELECTION(S) =SELECT.SKELETON
NEW SELECTION ==> F

 1.TAKE        - INTO THE CURRENT SELECTION (FROM ACTIVE)
 2.DROP        - FROM THE CURRENT SELECTION
 3.INSERT      - INTO THE ACTIVE DICTIONARY
 4.ERASE       - FROM THE ACTIVE DICTIONARY
 5.COPY        - INTO THE ACTIVE DICTIONARY (FROM PERMANENT)
 6.SAVE        - INTO THE PERMANENT DICTIONARY (FROM ACTIVE)
 7.FILLIN      - ENTER QUERY FILLIN-MODE WITH CURRENT SELECTION
```

We move back one menu level and request the "Programming" mode of IDAMS, called FILLIN by selecting Option 7.

```
                                                              DISP14
FILLIN:
  1|APL   |EXPRESSIONS|
 --|-0-   |-1---------|
 *  |*     |*         |

  2|NAME  |SHORT |LONG  |INDUSTRY∆CODE|
 --|-0--  |-1--- |-2--  |-3-----------|
 *  |*     |*     |*     |*            |

  3|STOCK |SHORT∆NAME|RAW∆RATES|ADJUSTED∆RATES|
 --|-0--- |-1--------|-2-------|-3------------|
 *  |*     |*         |*        |*             |

  4|HPLOT |HNUM |HTYP |HGRAF |HLAB |HTIT |HTICS |HLOGA |HSCAL |HSIZE|
 --|-0--- |-1-- |-2-- |-3--- |-4-- |-5-- |-6--- |-7--- |-8--- |-9---|
 *  |*     |0    |1    |*     |''   |''   |0     |''    |10    |10   |
```

The system shows the current object selection in skeleton form. The user's job is to name the columns he wants to use in order to combine these objects into a program. At this point, the descriptive information from IUGS is very helpful in knowing how to fill in the skeletons. For example, the use of a forecasting routine like Box-Jenkins requires the filling out of 14 columns (our example with HPLOT requires 9). The variable names are the user's choice; IDAMS has no explicit declaration statements.

```
                                                                DISP15
FILLIN:
 1 |APL |EXPRESSIONS
---|-0- |-1----------
1  |*   |LA='.TIME.DM'
2  |*   |GR=RR AND AR
3  |*   |HRO 5 2

 2 |NAME |SHORT |LONG |INDUSTRYΔCODE
---|-0-- |-1--- |-2-- |-3----------
1  |*    |SN    |LN   |*

 3 |STOCK |SHORTΔNAME |RAWΔRATES |ADJUSTEDΔRATES
---|-0--- |-1-------- |-2------- |-3------------
1  |*     |SN         |RR        |AR

 4 |HPLOT |HNUM |HTYP |HGRAF |HLAB |HTIT |HTICS |HLOGA |HSCAL |HSIZE
---|-0--- |-1-- |-2-- |-3--- |-4-- |-5-- |-6--- |-7--- |-8--- |-9---
1  |*     |1    |1    |GR    |LA   |LN   |0     |''    |10    |10
```

The data types are implicitly defined by the table columns that are used (RR and AR are vectors; SN and LN, character strings) or by assignment (LA and GR). HRO 5 2 is a command to format the graphic screen into five rows and two columns (10 pictures). The query is read as follows: Get the first row of the name table and store the short and the long name of that company into the variables SN and LN. Look for that specific name (SN) in the stock table for the corresponding raw and adjusted rates and store those in RR and AR. Combine the two vectors (RR and AR into GR) and use the plot function (HPLOT) with the long name (LN) of the company as the title of that picture and a specific label (LA). Repeat that for every short name which is in the NAME table and in the STOCK table.

```
                                                                DISP15
FILLIN: ?HPLOT.HLAB
 1 |APL |EXPRESSIONS
---|-0- |-1----------
1  |*   |LA='.TIME.DM'    DESCRIPTION OF AXES IN LISTFORM.
2  |*   |GR=RR AND AR     THE FIRST CHARACTER IS USED TO
3  |*   |HRO 5 2          SEPARATE SUCCESSIVE TEXTSTRINGS
                           FOR X, Y, Z, AXIS (IN THIS SEQUENCE)
 2 |NAME |SHORT |LONG |    E.G.:   '.X-STR.Y-STR.Z-STR'
---|-0-- |-1--- |-2-- |-
1  |*    |SN    |LN   |*

 3 |STOCK |SHORTΔNAME |RAWΔRATES |ADJUSTEDΔRATES
---|-0--- |-1-------- |-2------- |-3------------
1  |*     |SN         |RR        |AR

 4 |HPLOT |HNUM |HTYP |HGRAF |HLAB |HTIT |HTICS |HLOGA |HSCAL |HSIZE
---|-0--- |-1-- |-2-- |-3--- |-4-- |-5-- |-6--- |-7--- |-8--- |-9---
1  |*     |1    |1    |GR    |LA   |LN   |0     |''    |10    |10
```

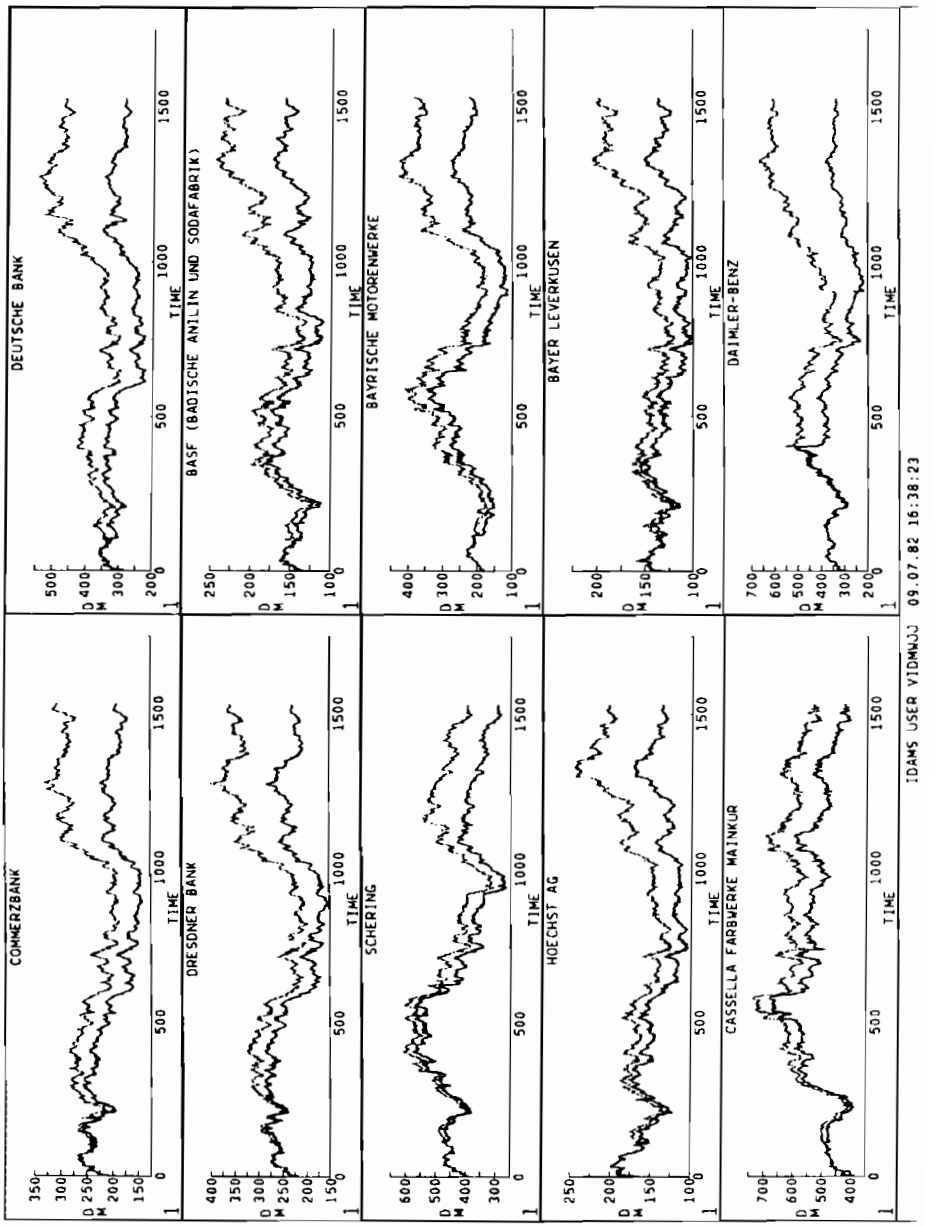
This display is an example of how you would ask for help to fill in the query skeletons.

```
                                                    DISP16
OLD SELECTION(S) =FILLIN
NEW SELECTION ==> 3

 1.FILLIN      -CONTINUE FILLIN MODE (EDITING)
 2.CLEAR       -DELETE ALL TABLE-ENTRIES
 3.EXECUTE     -EXECUTE SPECIFIED QUERY
 4.SAVE        -SAVE QUERY FOR REPEATED USAGE
 5.MODIFY      -MODIFY SKELETON SELECTION
```

With the skeleton complete, we select the EXECUTE Option to compile the filled in query skeletons into running code.

```
                                                    DISP16
OLD SELECTION(S) =FILLIN
ENTER QUERY NAME: EXAMPLE

 1.FILLIN      -CONTINUE FILLIN MODE (EDITING)
 2.CLEAR       -DELETE ALL TABLE-ENTRIES
 3.EXECUTE     -EXECUTE SPECIFIED QUERY
 4.SAVE        -SAVE QUERY FOR REPEATED USAGE
 5.MODIFY      -MODIFY SKELETON SELECTION
```

The last step is entering the name by which this query is to be identified. We have chosen EXAMPLE. The result is shown on the next page.


COMMERZBANK
DEUTSCHE BANK
DRESDNER BANK
BASF (BADISCHE ANILIN UND SODAFABRIK)
SCHERING
BAYRISCHE MOTORENWERKE
HOECHST AG
BAYER LEVERKUSEN
CASSELLA FARBWERKE MAINKUR
DAIMLER-BENZ
TIME
IDAMS USER VIDMAJJ 09.07.82 16:38:23

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

COMPARATIVE ANALYSIS OF USE OF DECISION SUPPORT SYSTEMS IN R & D DECISIONS

Patrick Humphreys
Decision Analysis Unit
London School of Economics and Political Science
Oleg I. Larichev
Institute for Systems Studies, Moscow
Anna Vari and Janos Vecsenyi
State Office for Technical Development, Budapest

Issues determining the success of applications of Decision Support Systems in ill-structured situations are examined through four case studies of R & D decision making. These concern (1) introduction of a new product where the R & D decision is taken at the company board level; (2) the determination of the product mix for a medium-sized manufacturing enterprise where more than one level of decision making is involved; (3) R & D decision making at the branch level within a three-level planning hierarchy; (4) the use of DSS in "top-level" decision making involving selection between proposals covering a wide range of R & D activities. In each case the context of each round in the decision making process is identified, together with the roles, motivation and responsibilities of the participants, and the level in the organizational process at which the DSS is implemented; factors which are shown to vitally influence the nature and success of the DSS usage. Pitfalls for DSS designers are uncovered through analysis and comparison of the cases, and ways of avoiding them are proposed.

INTRODUCTION

This paper examines through the use of case studies some factors which facilitate or limit the effectiveness of Decision Support Systems introduced into R & D decision making processes[1]. There is as yet no formal theory of exactly what constitutes decision support, and "Decision Support Systems" (DSS) is partly a rallying cry (Keen and Hackathorn, 1979). Here we adopt a very general view of a DSS. The system may involve the systematic use of tools, techniques, methods, etc., which support activities like the generation of decision alternatives, the elicitation and representation of information (values, premises, uncertainties) within decision models, the estimation of consequences of possible decisions, and the ranking of the alternatives in order of acceptability. However, while elements of these activities may be computer-based, in our view the system as a whole involves procedures carried out by individuals in interaction with others within an organizational context.

While successful implementations have been documented within R & D planning contexts (Boichenko et al, 1978; Mansfield, 1978; Souder, 1978), it is more common to find that the role actually occupied by the DSS in the overall decision making process was much more limited and quite often at variance with that anticipated by its designers or by the personnel who introduced the DSS into the decision making process (von Winterfeldt, 1982). One of the reasons

for this lies in the nature of the decision problems which have to be addressed: R & D planning problems tend to be relatively unstructured (McCosh & Scott Morton, 1978, Larichev, 1982). The decision problems faced in the four cases studied here were all of this type.

In Case 1, the directors of a medium sized company in Great Britain were quite uncertain about the structure of an R & D problem: should they replace an old, but currently successful product (a marine engine) with a new one involving a change to a more advanced technology in which they, as yet, have little experience. The new engine may be better able to meet as yet unspecified stricter environmental polution regulations, should they be introduced (and how should one estimate the likelihood of their being introduced in the next few years?)

In Case 2, the future of a Hungarian chemical works was uncertain. The works was producing plastic articles, pesticides, intermediaries used in the pharmaceutical industry and a variety of other organic and non-organic chemicals. Recently its rate of development had decreased, it had economic troubles and the ministry wanted to reduce its autonomy by fusing it with a larger enterprise. In what was seen as a last chance for the chemical works, new top managers were invited to help in solving the company's problems and to formulate a strategy for its development. But what should this strategy be - what criteria does it have to meet?

In Case 3, decision makers within a Hungarian State authority responsible for a sector of services at the national level were facing the problem of budget allocation among R & D projects. Because of the heterogenity of R & D activities in the field, the projects, as well as the phases in decision making were arranged in a three level hierarchical system comprising main areas (first level), programs (second level) and tasks (third level). However, each second level program comprised a set of tasks which were not rigidly defined, and each first level area comprised programs which were not rigidly pre-determined, and so decisions arrived at sequentially would not necessarily be consistent. How can harmony, rather than conflict, be ensured between the decisions taken at the three levels?

In Case 4, located at the highest level in an organizational hierarchy, a decision maker in the USSR had the task of evaluating approximately 1000 research and development projects. The decisions had to be made as the individual proposals came in. Yet the criteria for choice of projects had to be stable and set a priori. Moreover, these criteria, and values on them were all expressed in verbal terms. How can the decision maker decide how to structure them in the absence of a data base, that is, before the projects arrive?

In facing decision problems like these, there is likely to be confusion in the decision maker's mind at the outset concerning the nature of the problem domain. In some cases, parts of the decision problem may be well modelled a priori. For example, in Case 1 the company had a marketing department which felt confident that it could predict market share for a product with a given price and specification, defined in terms of advantages and disadvantages over its competitors. But other parts of the same problem may be less easy to structure on the basis of information already gained about the problem domain. In anticipation of the possible effects of changed environmental pollution regulations, does one have to consider the possibility of a change of

government in the next ten years and predict the changed policy that would go with it?

In providing decision support for ill-structured problems like these, it is our belief that the "knowledge system" incorporated in the DSS (Bonczek et al, 1983) must be of a fundamentally different character to that which can profitably be developed in supporting the analysis of well structured problems and repeated decisions. A corollary of a problem being ill-structured is that at the outset the domain of facts which may be relevant is potentially infinite. Given such a problem, and the severe time constraints which typify R & D decision making, we consider that it is usually better to leave the knowledge base in the minds of the participants. Computer-based resources can then be focussed on developing methods for (i) structuring the problem, (ii) accessing the knowledge base existing in the participants' minds, (iii) simulating alternatives under various assumptions and perspectives and (iv) performing interactive sensitivity analyses to provide an informed basis for choice.

DEVELOPING THE METHODOLOGY OF DESCRIPTION AND EVALUATION OF DSS USAGE IN R & D PLANNING

R & D planning in real life is a continuous process with sequential variety in the pattern of activities and participants involved. The conceptual framework used here requires that we first divide up the process into interconnected segments which can be separately modelled, together with the specification of linkages between these segments. This involves identifying a sequence of rounds, and stages within each round, in the planning process as well as specifying the level (or levels) of the decision making activities within each round.

ROUNDS AND STAGES

Our definition of a round within the decision making process follows that proposed by Kunreuther (1982) as

> "A round is simply a convenient device to illustrate a change in the focus of discussion either because (1) a key decision was taken (or a stalemate reached due to conflicts among parties) or (2) a change occurred in the context of the discussions due to an exogenous event, entrance of a new party or new evidence to the debate ... no matter how a round is initiated it is characterized by a unique problem formulation which is presented in the form of a set of attributes."

Within each of our case studies of R & D decision making we identify a set of stages within the rounds studied. Each stage is located in terms of those stages which precede and follow it. Its inputs and outputs are usually well defined. The outputs from a stage may serve as inputs to the immediately following stage in the round, or to any defined subsequent stage in the round. The converse holds for inputs to a stage. Inputs and outputs between rounds are generally less well defined as a boundary between rounds generally represents an untheorized discontinuity in the planning process. At the start of a new round outputs from previous rounds tend to be picked up and interpreted as inputs in ways unanticipated during the previous round.

At each stage within a round the "unique problem formulation" to which the round is addressed is represented in a different form. Where a DSS is employed, it will be important to examine whether the problem formulation to which the DSS is addressed is "requisite". Phillips describes the relevant requirements as follows:

> "To develop a requisite model, it is necessary to involve all those who are in some way responsible for aspects of the decision in the development of the requisite model. The process of building the model is iterative and consultative, and when no new intuitions emerge about the problem, the model is considered to be 'requisite'. In requisite modelling, it is expected that people will change their view of the problem during the development of the model; that is why the process has to be iterative." (1982, p.304)

This ideal way of constructing a requisite model is rarely achieved in practice but it gives us some clues about questions to ask in examining the degree of "requisiteness" extant in actual applications supported by DSS, viz: Are all those who are in some way responsible for currently modelled aspects of the decision involved in the development of the model? Are intuitions emerging about the decision in personnel currently involved or responsible for subsequent actions which are not incorporated in the model? Is the modelling process iterative in a way that can encompass changing or different views?

LEVELS

R & D policymaking usually progresses at several levels. These may be bureaucratically determined, where different strata within an organization are charged with policies with different scopes and time horizons. A particular R & D planning process may involve a departmental management stratum dealing with the evaluation of the characteristics of a particular product; a general enterprise management stratum dealing with problems of introduction of positively evaluated new products; a corporate or sector management stratum dealing with the future of the enterprise within a wider plan, and so forth. In general, the way in which a R & D planning problem is represented within a DSS will, if requisite, be specific to the organizational management stratum whose activities are being supported by the system (Jaques, 1976). Within any stage of the decision process "officially" located at the level of a defined stratum we may, however, find participants operating at different levels of problem conceptualization.

ROLES OF PARTICIPANTS IN THE ROUND

Within any stage in a round, individual participants can be classified as **decision makers** (defined as those who have the executive power to define the use of outputs from the round); **proposers** (those who have power only to make recommendations on this); **experts** (those whose primary function is to supply inputs to the currently modelled problem structure); **consultants** or decision analysts (those who advise on methods of problem representation) and **facilitators** (those who do not take any direct role in the decision making process, but who are in a position to facilitate the collaboration of experts, the transmission of the results within and/between rounds, and so on). A participant may act in different roles within rounds located at different levels, or even within different stages in a particular round. Therefore, in

categorizing participants, it is important that their roles be defined in relation to the state of problem representation and DSS in use at each stage in a round. Participants may also serve as links between stages or rounds, carrying certain information with them, but this is a process which can be studied separately.

MOTIVATION OF PARTICIPANTS IN THE ROUND

In the case studies outlined below we will be looking primarily at rounds in the R & D planning process where a new form of DSS was introduced. We shall see that it is very important to understand the differing motivations of the participants in the round, as this will affect the results they expect from the DSS, and how they view their significance (Berkeley & Humphreys, 1982).

A decision maker may be strongly motivated to apply decision analytic methods implementing DSS in situations involving many complex R & D proposals. In such cases a decision maker has practically no avenue of influence on decision processes except through the utilization of decision rules superimposed on expert evaluations. The support he is likely to seek from a DSS is that which will help him to increase the centralization of decision making through exercising influence in this way.

A proposer may wish to employ a DSS to get proof of support of experts, while already having some idea of what will make the project acceptable to those who will consider his or her recommendations. Some of the support which he is likely to seek from a DSS has to do with the possibilities of manipulation which can lead to a particular interest in a DSS having simulation capabilities under alternative scenarios (Humphreys & McFadden, 1980). It is often proposers who introduces consultants as this can serve their interest in increasing the probability of the acceptance of their proposals.

A consultant's principal motivation, as an outsider, is usually concerned with the acceptance of the procedures he or she introduces, which in the case studies discussed here were linked to DSS. However, doing this means penetrating an organizational culture (Handy,1981) and taking on a temporary role within that culture. The nature of this role sometimes attracts other motivations potentially in conflict with the principal motivation. These can sometimes reflect a desire for power ("behind the throne"), status ("consorting with the great"), social beliefs (promoting the "decision culture"; improving "organizational democracy") or self image (to be a "helping person")

The motivation of a participant in the round determines his or her view of the function of expected results of DSS usage. However, the extent to which this will lead to a positive orientation towards the DSS will depend upon how its effects are preceived at the outset of the round. Some of the relations involved here are shown in Figure 1.

RESPONSIBILITY

The relationship between motivation and perceived effects of DSS depends also on the responsibility a participant holds or wishes to assume. A high level decision maker with responsibility for implementation of policy may use the report from a decision analysis as justification for the policy. In effect

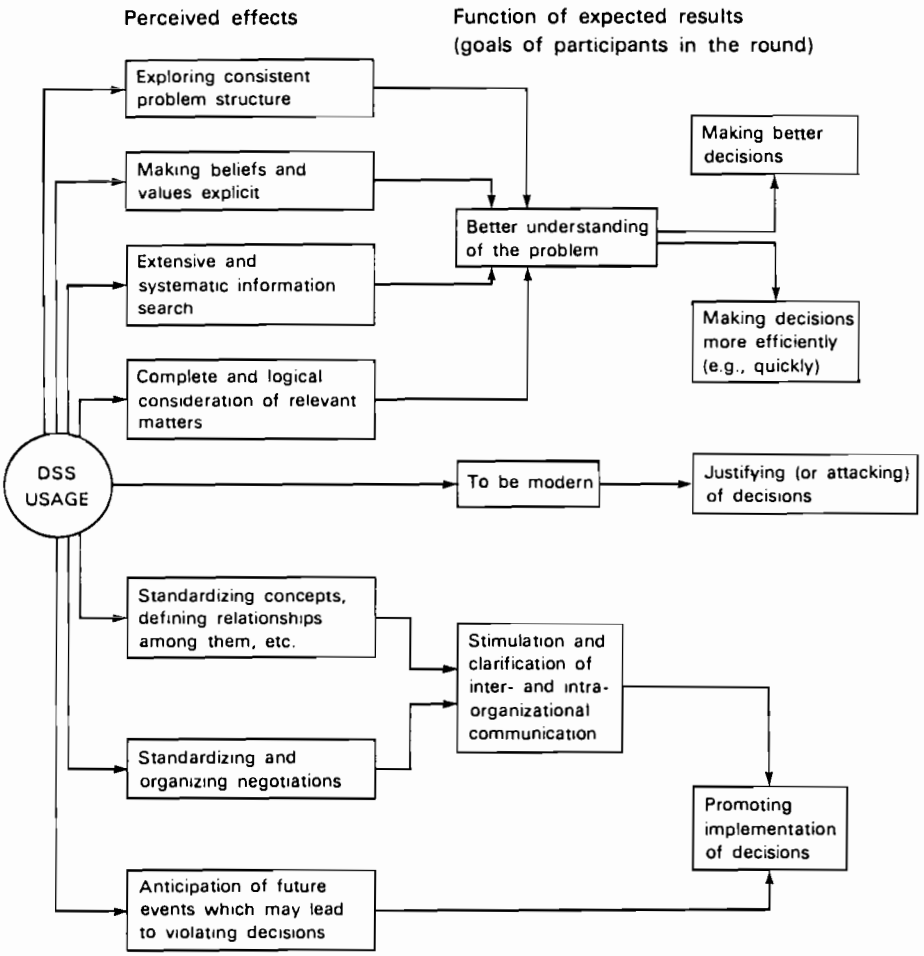


Figure 1 Some relations between perceived effects of DSS usage and goals of a participant in a round

this shifts responsibility in the case of failure onto the report and its creators and where a DSS has been explicitly involved it often ends up collecting a large share of the blame. Proposers may attempt to structure a problem to fit the preferences that they believe held by those with executive responsibility to whom they report. They may be more sensitive to their own position and career prospects than to an effective outcome, and it is with regard to these prospects that they may examine the adequacy of a problem representation constructed through the use of the DSS. In view of possibilities like these, we hypothesize that motivation and responsibility will interact in determining DSS acceptability in the way shown in Figure 2.

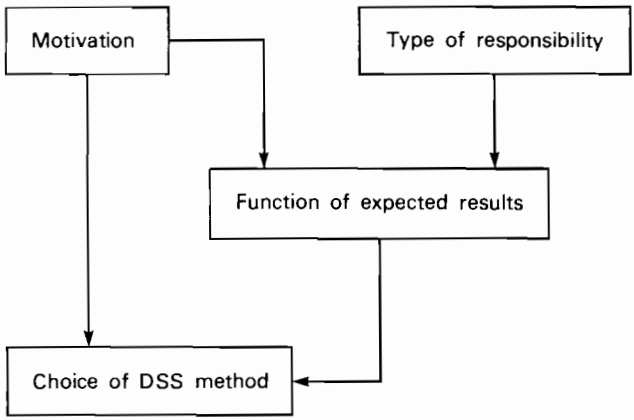


Figure 2 Hypothesized effects of motivation and responsibility on choice of DSS

A CHECK LIST FOR DEVELOPING CASE STUDIES

The issues raised in the previous section imply that, in comparing R & D/DSS case studies, one should develop a framework whose components are connected with:

(a) the organization and procedure of R & D planning.
(b) the goals that participants in the planning process hope will be achieved by using DSS within the context.
(c) the expected and the actual role of the consultants and of the other participants in stages of the decision process within the round.
(d) the expected and real function of inter- and intra-organizational communication within and across stages in the round (e.g., group negotiations).
(e) the requirements for information (e.g., the required number of alternatives, attributes, and scenarios regarded simultaneously), and its mode of availability.
(f) the way of handling uncertainties.
(g) the way divergent views are reconciled.

From this framework we developed a checklist of some items to be used in analysing the case studies described here. This check list, and details of its use, are published in Humphreys, Vari and Vecsenyi(1982). Here we will highlight some of the findings which were gained from applying the checklist retrospectively to four case studies, and provide references to the papers where the case analyses are described in detail. This will be followed by a review of findings from some comparisons we were able to make across the various case studies.

CASE 1: INTRODUCTION OF A NEW PRODUCT: MARITIME ENGINES AND MOTORS (MEM)

This study is located at the board level of MEM, a British company manufacturing outboard motors and small maritime engines (details are given in Phillips, 1982; the name of the firm and its product have been changed to maintain confidentiality). A single R & D decision had to be taken between continuing to manufacture an old product that might in the near future be banned by the government for failing to meet exhaust emission standards or introducing an improved product that would beat the ban but might lose market share to competing products using micro-chip technology. The company was unable to move directly to a chip-based product as it did not, as yet, have the required technology. Hence any product introducted in the next few years would have to rely on improvements in conventional technology.

MEM ROUND 1 (NO DSS)

The participants in the decision making process comprised MEM's board of directors. The managing director was the decision maker with executive power, but the board, meeting as a whole, had to agree on the action taken on the basis of the information participants have about the problem. The last time MEM considered introducing a new product, a report had been written recommending approval by the board of directors. Directors on the board took exception to certain assumptions made in the report and asked for it to be done over. The revised report was submitted to the board where participants now took exception to other assumptions, and so this process continued for eleven revisions over twelve months at the end of which no decision was taken.

MEM Round 2 (introduction of DSS)

After attending a university management programme course in which decision analysis was introduced, MEM's Managing Director sought the help of the Decision Analysis Unit at Brunel University to see if decision technology could be applied to the problem. This led to the start of a new round where a DSS was introduced in modelling the R & D problem outlined above. The stages and participants in this round are shown in Table 1.

Stage 1 defined the terms of reference for the subsequent stages. The decision makers who participated were MEM's managing director, business planning, finance and production managers and a consultant from the Decision Analysis Unit. Major uncertainties about act-event sequences were identified, company objectives were discussed, a time horizon (10 years) was set for evaluating possible effects of consequences when modelling the decision, and the major characteristics of the financial model to be employed were agreed.

A decision tree was developed in Stage 2 by the consultant through showing and discussing it in individual meetings with a variety of experts (e.g., sales, finance and production managers and some of their staff) and decision makers. In Stage 3, MEM's business planning manager, in the role of an expert, supplied the financial model which was married by the consultant to the decision tree. This formed the basis for a DSS generated by the Decision Analysis Unit, a process which included the use of generic software for building and analysing decision trees.

Stages in the Round \ Participants		Decision makers	Consultants	Experts	Computer
Stage 1: Define terms of reference		* (meeting together)	*		
Stage 2: Develop decision tree		* (meeting singly with consultant)	*	*	*
Stage 3: Develop DSS incorporating tree and functional model		* (meeting singly with consultant)	*	*	*
Stage 4: Decision making	Explicate model	* (meeting together)	*		*
	Conduct sensitivity analysis: final decision	* (meeting together)			*
Stage 5: Defining conditions for implementation of decision		* (acting alone)			

Table 1

Stages and participants in Round 1 of Case 1:
Introduction of new product: Maritime Engines and Motors

In Stage 4, the financial and decision models were presented to MEM's board of directors. Participants in this stage (MEM directors) expressed disagreements about assumptions in the model, and the DSS was used interactively by the decision makers themselves to test the sensitivity of the decision to their differing expressed views on assumptions (the IBM 5110 portable computer implementing the DSS had been brought to the meeting). This process continued iteratively until all participants in the stage were agreed that the patterns of assumptions which would be needed to overturn the

desirability of one of the policies modelled in the decision tree ("introduce improved product now") were unreasonable. The decision makers implicitly located their discussion of alternatives within the problem representation modelled in the DSS. Agreement was reached on policy without the requirement that the decision makers also reach agreement on assumptions about all the values in the model (although the general **structure** of the model had been accepted by consent in stages 1 and 2). The sensitivity analysis had demonstrated to them that each could agree the policy while maintaining his own set of assumptions.

At this point the consultants considered the round to be completed. However, the Managing Director subsequently initiated a further stage within the round. Noting that the R & D decision model accepted in stage 4 indicated that there was a considerable difference between the expected value of having a "clear introduction of the new product" as decided upon in stage 4 and "continuing with the old product", he decided that it would be well worth while trying to improve on the probability (0.6) which had been agreed by the Board as their estimate of getting a clear introduction. This involved spending money on improving the aesthetic design of the product and improving MEM's marketing methods before introducing it. The Managing Director took responsibility for deciding on this expenditure, justifying it as being considerably less (<10%) than that which would be saved through the resulting increase of the probability of a clear introduction to around 0.8.

REASONS FOR SUCCESS

This case study can be judged a success in that the introduction of DSS provided a framework for thinking about the problem which allowed the participants in Stage 4 of the round to reach a unanimous decision and to understand the nature of that decision in relation to their uncertainty about future events and conflicting assumptions. However, in comparing this result with the output of MEM Round one, one can see that nearly every aspect of the decision making process was ripe for improvement through the introduction of a DSS. We can identify some of the reasons which contribute to this success by contrasting this case with those described below, where improving on the previous (non DSS) baseline was generally not so straightforward a task.

CASE 2: DETERMINING A PRODUCT-MIX DEVELOPMENT STRATEGY MAKING BY A HUNGARIAN CHEMICAL WORKS (CW).

The context of this R & D is more complex than that for MEM in that (i) the decision concerned the product mix constituting the entire output of a medium size (3,000 employee) enterprise, rather than just a single product, (ii) while the rounds investigated here were located at top management level within the company, the resulting strategic decision making took place outside at a higher level.

CW ROUND 1

The first round we studied started with the introduction of new top managers at the chemical works to provide the 'last chance' for the works which we described in the introduction to this paper. One of these managers initiated

the analysis of the problem by decision analytic tools. He had been introduced to practical applications of multiattribute utility theory in a post-graduate management science course, and believed that this new method would be better than the traditional cost-effectiveness, market position evaluation. However, the actual method used for the decision analysis (and the supporting computer software) was developed during the round by a team of consultants from the Bureau for Systems Analysis of the Hungarian State Office for Technical Development and the Technical University, Budapest.

R & D strategy making is usually based on the assessment of proposals comprising ideas for new products together with the evaluation of previous R & D results. However, in the situation facing CW, R & D strategy had a different character, requiring pruning of the existing product mix rather than a consideration of possible new additions. The problem for analysis was defined at the outset of the round as the evaluation of the product currently being fabricated, revealing their weak points and requirements for development. This was seen as forming an important input to the overall R & D policy making, which was to determine the development and production strategy for the next one to five years.

Consequently the procedure used within the round did not focus on the assessment of alternative R & D proposals as such but on their components: preferences between products for development and production. This does not mean that assessment of the overall proposals should be omitted from strategy making, only that the DSS applied here was not required to support that activity.

CW ROUND 2

Two years later there was a second round in the process. The mode of initiation of the analysis, the definition of the problem and the method of use of DSS remained the same. However, at this time the situation of CW had improved. In the period between the rounds the firm had gradually started to develop, its economic state had stabilized, and its independence had been assured. Consequently, the motivation of the participants in the round for DSS had changed. In Round 1, decision makers perceived the use of DSS as one of the tools of survival but in Round 2 DSS was perceived by decision makers only as a useful aid in re-evaluating the R & D strategy based on the results of using the DSS in Round 1.

The participants in the round also changed. In Round 1 representatives of state authorities and of related organizations (foreign trade companies, association of chemical enterprises, etc.) were also involved in the analysis supported by the DSS. However, in Round 2 only internal experts were involved. In Round 1, securing the involvement of external representatives was one of the ways of getting their benevolent support in helping CW to survive.. In effect, they had played the role of facilitators in the decision making processes and in round 2 there was no longer a need for such explicit participation of external facilitators.

Here we shall concentrate on CW Round 1, making parallels and contrasts with the MEM study where appropriate. A detailed account of CW Rounds 1 and 2 is given in Vecsenyi (1982).

RESPONSIBILITY AND MOTIVATION OF DECISION MAKERS

As in the case of MEM, the board of top level managers were responsible for the determination of company strategy. However, they knew that they also had to "set an example" to decision makers at a higher level than their own if the company was to survive its independence. At this higher level, they acted as **proposers**, recommending their methods of analysing the problems of the company as the evidence of their own capacity for good strategic planning (as against the alternative of being fused into another strategic plan). Hence their motivation was more complex than that for MEM decision makers, being oriented towards three goals:

(i) rationalizing their decisions by basing them on more reliable information (as in MEM, see the discussion of MEM stages 4 and 5).
(ii) getting the collaboration of lower level managers in carrying out the strategy (this was hardly a problem in MEM, which was hierarchically organized with a board of established authority).
(iii) having a tool for convincing higher level authorities by setting an example to them to solve the problem of company by using up-to-date tools (this served the decision makers in their **proposer** role, which did not exist in the MEM case).

STAGES IN THE ANALYSIS

The DSS used in round 1 was seen by both decision makers and consultants (analysts) as a procedure generating multiattribute utilities of products in the mix on the basis of preferences expressed by individual participants grouped in various ways. This DSS supported the first four of the five stages shown in Table 2.

In the first stage in the analysis, a list of ten main criteria and 36 subcriteria, initially prepared for evaluating the products by postgraduate students in industrial engineering was discussed and modified by 30 leading executives of the company. This resulted in 70 subcriteria, and some changes in the interpretation of the main criteria. In the second stage CW managers determined company objectives and requirements related to criteria, so that attributes of products related to these criteria could be evaluated. In support of this, the consultants (the analysts who designed the DSS) organized a training course for the participants on the methods of weighting attributes and assessments of the products and on the procedure of DSS[5].

In weighting the attributes (stage 3), separate vectors of weights were provided for all 63 participants in this stage in the round (five top level executives, 38 medium level executives and 20 experts)[6]. The director of CW also asked 15 external "higher level" experts (members of the supervising committee and representatives of their respective supervisory committee at Ministry level) to determine importance weights for the principal criteria. The consultants used clustering techniques to compute pooled vectors of weights for three groups of participants in the round: (i) top decision makers within CW, (ii) CW company experts, and (iii) external (higher level) experts. The executives of the company discussed the similarities and

Stages in the Round \ Participants	Proposers	Decision makers	Consultants	Experts	Computer
Stage 1: Exploration of criteria	*	*	*	*	*
Stage 2: Weighting of attributes	*	*		*	
Stage 3: Assessment of alternative products	*	*		*	
Stage 4: Computation of multi-attributed utilities of products in the mix			*		
Stage 5: Strategy making		*			

Table 2

Stages and participants in Round 1 of Case 2:
Determining at product-mix development strategy by a chemical works

differences between the results for the various groups, and agreed that the model should be simulated using (separately) the vectors of weights from these three groups: (i) top executives of the company, (ii) the group whose individual weights demonstrated the highest degree of concordance, and (iii) the weights for the group of all 63 evaluators.

Assessments of 46 alternative CW products were made in stage 3 by the same 63 participants as in Stage 2 (no external experts were involved). The procedure was taught to them in the training course arranged by the consultants so that they could express both valuation and uncertainty on the attributes identified

in Stage 1 in a format appropriate for computing the multiattributed utilities of the products in the in the mix (stage 4). This computation was carried out by the consultants using a multicriteria simulation model developed by Kiss et al (Kiss, 1978; Kiss & Torok, 1979) from a procedure proposed by Kahne (1975). Three separate sets of rankings were computed for the products, one for each of the groups whose attribute weighting vectors were assessed in Stage 3. The consultants reported the evaluation of each product in terms of how each of the three groups viewed it according to the simulation model.

Strategy making (stage 5) was not covered by the DSS as the decision makers did not wish the support of the consultants in this stage and, on the other hand, consultants had no adequate method for strategy making in this case. Thus the problem defined at the start of the round by CW's director as that to be addressed by the DSS was complete at Stage 4, but it provided simulation outputs, not strategic prescriptions. In this sense the DSS supported **proposals**, rather than decisions. Vari and Vecsenyi (1982) discuss this as a pitfall for decision analysis: where the domain of the problem is greater than the domain of the decision analysis. In order to make decisions about the actual development strategies, additional criteria were used in stage 5 by the decisions makers, (e.g., those relating to governmental programs, costs required for development, capacity constraints, etc.). Excluding these from explicit consideration within the DSS meant that only part of the decision makers' values and preferences had to be made explicit and subjected to formal analysis while implicit values could be taken into consideration intuitively by the decision makers during the actual decision which was taken at this stage.

SUCCESS OF THE DSS AS A PROPOSAL SUPPORT SYSTEM (PSS)

Despite the apparent pitfall for DSS in Stage 5, two years later the director of CW initiated a second round in the R & D planning process with the introduction of similar DSS to that used in Round 1. Hence, in some sense the DSS had been found useful. But for supporting what? Von Winterfeldt (1980) has discussed how Multiattribute Utility Theory (MAUT) based systems like that underlying the DSS used here are inappropriate for **decision** analysis in resource allocation problems, like that facing CW. However, given the motivation of CW's managers, discussed in our account of Round 1, it appears that here the DSS met the goals of these decision makers through being perceived as a **proposal** support system (hence the emphasis on its simulation capability) rather than as a decision support system, which is, by contrast, the appropriate characterization for the DSS employed in the MEM case. Understanding the role of the DSS as a PSS sidesteps von Winterfeldt's criticism of the use of MAUT since here it is being used is a system with capabilities for proposal analysis rather than decision analysis. This in our opinion is what provided the key to its success.

CASE 3: R & D PLANNING AT THE BRANCH LEVEL

Case 2 illustrated how a decision maker's differing roles in a R & D decision making process in which he is involved at more than one level can affect his perception and use of DSS. We can explore the nature of differences between levels further by looking at a case of budget allocation among R & D programs.

In the Hungarian sector in which this case is located, programs - as well as the phases of the usual decision making process - are arranged in a 3-level hierarchy in the way that we outlined in the introduction to this paper.

The problem considered in Round 1 comprised budget allocation across ten 3rd level programs. Analysis of the problem by decision analytic tools was initiated by a lower-level decision maker responsible for making proposals for financing R & D programs in his field. The method for the decision analysis and the supporting computer software were developed by consultants (a team of decision analysts from the Bureau of Systems Analysis of the State Office for Technical Development and the Institute of Psychology).

In Round 2 the scope of the problem expanded to the consistent allocation of the budget on all the three levels, which means allocation across 4 areas, involving about 20 2nd level programs and about 50 3rd level programs. The analysis of the problem - by using the method and procedures developed in Round 1 - was initiated by higher level decision makers responsible for the whole R & D planning. Due to the greater complexity of the problem and the extended circle of experts involved some minor modifications were made in the method. The work is in progress, and so here we shall be mainly concerned with Round 1, and our preliminary experiences of Round 2.

RESPONSIBILITY AND MOTIVATION OF DECISION MAKERS AND PROPOSERS

Higher level decision makers were responsible in this case for the budget allocation among all candidate R & D programs. The programs covered all of the main areas of R & D activity and, of course, higher level decision makers cannot be competent in each of these areas. In using a DSS, decision makers were motivated by the opportunities it provided for (i) rationalizing their decisions by basing them on more reliable information, (ii) having a tool for explaining (justifying) their decisions to their subordinates (to the managers of the competing areas programs, etc.) and (iii) modernizing their decision making practice.

Proposers (in this case lower level decision makers within each of the main areas) were responsible for making suggestions to higher level decision makers as to which R & D program of their particular area should be supported. They were more competent in their area than higher level decision makers, although they needed the help of experts who were familiar with the details of the R & D programs. In using DSS proposers were motivated by the possibility of (i) influencing the decision makers by using more efficient tools, (ii) eliciting information from experts, and (iii) learning new methods for modernizing their own decision making practice.

FUNCTION OF RESULTS EXPECTED FROM THE DSS

Because of the difference in responsibilities and motivation between the decision makers and the proposers, we would expect from Figure 2 that the function of the results these two classes of users expected from the DSS used would be quite different. We found that decision makers expected to use the results for (i) prescriptions for action (e.g., rank order of R & D programs in terms of cost-effectiveness) which would at least partly transfer the responsibility for such actions from themselves to the DSS, and (ii) as rationalization for actions they might wish to take.

On the other hand, proposers expected to use the results for (i) gaining a better insight into the decision problem (e.g., simulation of the consequences of the possible choices, multiple criteria analysis of the options, etc.), and (ii) communication of information. Some of the reasons behind the differences between the expectations of the proposers and the decision makers were those we discussed for Case 2, above. Others will be examined when we consider some additional pitfalls below.

STAGES IN THE ROUND

At the outset of the round the consultants proposed that the problem should be conceptualized as one of budget allocation, and therefore the stages in the round would be those required to compute, as a basis for decision, the subjective expected utility (SEU) for each R & D project or, in the case of project-interdependencies, each project-combination. SEU was accepted as a basis for "decision making" by the proposers who were responsible for initiating the use of the DSS, but at the end of the round, they more or less neglected SEU in forming the final proposal.

The consultants (decision analysts) proposed direct optimization algorithms for budget allocation on the basis of the SEU of projects, but this was refused by the proposers (reasons why the proposers acted in this way will be discussed below). In view of this, the eight stages shown in Table 3 were agreed to constitute the round. The procedures used in the above stages are described in detail in Vari an David (1982). Table 3 also summarizes the involvement of the various types of participant across the stages in the round.

In round 1 the proposers played the following three roles: (i) they controlled the whole decision analysis process, determining the output of the crucial stages (e.g., alternatives, criteria, weights, probabilities, utilities) and they wrote the final report (including proposals and explanations, etc.), defining the output from the round; (ii) they mediated between decision makers and all other parties, thus determining the way that the outputs from Round 1 served as inputs to higher level decision making; (iii) they acted as experts, participating in the exploration of alternatives on criteria, evaluation of weights, probabilities and utilities. In Round 2 the role of the proposers was somewhat different. The overall decision analysis was controlled by the consultants and the formal proposal was formulated by the consultants as well. The participants who had played the role of proposers in round 1 acted only as experts throughout all the stages of round 2.

The experts' participation in the exploration of decision alternatives and criteria was focussed on determining criteria weights and those probabilities and utilities related to the projects within the field of their experience. The consultants (i) organised the decision analysis and the interactions between the participants involved in making this analysis, (ii) designed and implemented the computer programs supporting stages 2-7, and (iii) elicited data from participants and explained the results of the computer procedures to them. In round 2 they also formulated the final proposal for presentation to higher level decision makers.

Stages in the Round \ Participants		Proposers	Decision makers	Consultants	Experts	Computer
Stage 1: Definition of the set of projects to be evaluated		*			*	
Stage 2: Exploration of the evaluation criteria		*		*	*	*
Stage 3: Determination of weights of criteria		*		*	*	*
Stage 4: Definition of utility functions		*		*	*	*
Stage 5: Estimation of the uncertainties		*		*	*	*
Stage 6: Evaluation of the projects		*		*	*	*
Stage 7: Multi-criteria aggregation				*		*
Stage 8: Selection of the projects	Forming the proposal (reporting)	*		*		
	Final decision		*			

Table 3

Stages and participants in Round 1 of Case 3:
R & D planning at the Branch level

PITFALLS LIMITING THE SUCCESS AND SCOPE OF DSS FOR R & D PLANNING AT THE BRANCH LEVEL

In a general sense, the introduction of DSS into R & D planning at the Branch level in Round 1 of Case 3 was successful, as it led to Round 2 where higher level decision makers initiated the use of similar DSS in evaluating a much wider range of programs, located at three levels. However, when we examine **how** the DSS was used at the Branch level and compare this with the perspective and expectations of the consultants who designed the DSS at the start of the round (c.f. Czako and Vari, 1980), we can identify some pitfalls.

PITFALLS OF MISUNDERSTANDING THE ROLE OF THE PARTIES INVOLVED.

The first group of pitfalls the consultants had to face during the round was related to misunderstanding of the roles of the parties involved, which led to wrong definition for some of the communication interfaces:

Communication of the proposers with other parties. Proposers intended to play a more active role than the consultants originally expected. They wanted to participate in the analysis as experts as well as mediators of the, (supposed) preferences of the higher level decision makers. On the other hand, they wanted to influence the decision makers and to this end they wanted to have the freedom to manipulate the results. Therefore, they preferred having an insight into the consequences of the possible actions (choices) instead of receiving a direct prescription from the DSS. Thus, as in case 2, a system designed in support of **proposers'** activities was found to need to be centered on rather different capabilities than would be the case in supporting actual decision makers. Failure to recognise this distinction at the design stage constitutes a possible pitfall for DSS implementation. Note also in this respect that the proposers preferred to consider only a certain part of the whole problem structure (e.g., some criteria) in the course of the formal decision analysis process and to take the other components into consideration intuitively while making proposals (see the discussion of Case 2, above, and Vari & Vecsenyi, 1982). One way of limiting the extent of this problem is to make the formal analysis less public (as when, in subsequent rounds, proposers learn the method and replace the consultants). However, owing to the organizational character of decision analysis as a collective activity, publicity cannot be wholly eliminated.

Communication of the decision makers with other parties. Decision makers played a much more passive role than the consultants expected. Originally, the plan was to involve them in the analysis, particularly in determining and weighting the criteria, but they refused to participate. They required only a one-way channel which would serve to communicate to them the essence of the experts' views.

A conclusion to be drawn from this is that the DSS supported the proposers, rather than the decision makers. In other words, while implemented within a two-level organisational system, it was appropriate in providing support only at the **lower** level where it served as a proposal support system (PSS) in a way analogous to that described for the CW case. The lack of information available at the lower level about the real criteria and values of the decision makers also supports the advantages of simulation-type methods over direct optimization algorithms, and the importance of sensitivity analysis within PSS.

A comprehensive DSS supporting both levels would need to include a PSS at the lower level together with a system with rather different characteristics supporting the higher level where decisions are actually taken. Some of these characteristics are explored in the fourth case study which describes the development of a DSS for top-level decision making.

PITFALLS STEMMING FROM HIERARCHICAL DIVISIONS IN ROLES AND RESPONSIBILITIES IN R & D PLANNING.

The second group of the pitfalls is connected with the sequential character of decision making which, in this case, is the consequence of the hierarchical division of roles and responsibilities between participants.

Selection between programs on different levels. The usual practice in the context of Case 3, is for higher level decision makers to allocate resources between the main areas (1st level), and programs (2nd level), while lower level decision makers do so between tasks (3rd level). The result of this procedure is usually suboptimal: it may happen that an area which is declared to be "very important" does not include as many realistic R & D programs as do other, not so "important" areas. An optimal solution would require the simultaneous comparison of all 3rd level tasks and the selection of the tasks to receive support on the basis of this comparison.

Hence, in Round 2 of this case, the consultants carried out multiple criteria comparisons of the areas and programs at all three levels and put together the results gained on the different levels. The method employed (details of which are given in Vari & David, 1982) proved to be useful, although the consultants had to face serious methodological problems related to the comparability of the evaluations given by different experts. None of the experts could evaluate **all** the 3rd level projects. Each evaluated a subset, and so evaluations on different projects by different experts had to be compared.
Another potential pitfall stemmed from the decision makers' assumption that, given appropriate DSS methodology and experts, the large number of R & D themes which characterized the third level tasks could be analysed directly (using in effect an aggregated PSS as the DSS). This unrealistic assumption ignores the necessary discontuity between the requirements for top level DSS and lower level PSS.

Discounting for conditions of implementation. Another consequence of the division of roles is that decision on "what?" and "how?" are usually not considered simultaneously. The consultants in this case, motivated (i) to develop and test new methods in the field of R & D planning and (ii) to make the decision process more coherent, democratic and better organized, strived to ensure the use of an evaluation method which could take the conditions for successful research and implementation into consideration. For this reason they proposed calculating SEU of projects as the basis for choice between them since this took into account the probabilities of conditions permitting successful implementation, as well as the costs and benefits of the results.

However, on reaching the final stage (8) of Round 1, the proposers neglected the SEU of the R & D projects as a basis for choice. Post-hoc analysis of the choices actually proposed indicated that choice was governed by the **maximum feasible utility**, calculated for each project as that which could be

obtained in the case of total success in all aspects of research, implementation and application (Vari & Vecsenyi, 1982). In other words, probabilities of failure in implementation were discounted.

The consultants informed the proposers of the results of this analysis, the proposers utilized these results by putting forth suggestions designed to alter the social **context** of implementation of the programs in such a way that uncertainty about future events would be reduced. They alerted decision makers to the fact that in certain programs "great attention should be paid to promoting the implementation" **without any further analysis** of the possible alternatives to promotion.

CASE 4: USE OF DSS IN TOP LEVEL DECISION MAKING ON R & D PROPOSALS

Case 3, while situated in the context of multi-level R & D planning concentrated through force of circumstances on the use of DSS at the lower, branch level. In this final case study, we investigate the appropriate use of DSS at the top level, approaching the problem from the stand point of a large interdisciplinary research institute in the USSR and from the point of view of planning office heading a number of research institutes. Each situation had the same general features, notably:

(i) Individuals or organizations submitted proposals on R & D. These proposers were potential executants or clients, interested in gaining R & D results.
(ii) the decision maker reponsible for the choice of the best R & D alternatives was located at the top level: head of a planning office or the chief executive officer of an organization. The decision maker followed a policy in choosing among the R & D proposals realized through a set of **his** criteria (Zuev et al, 1979; Larichev, Zuev and Gnedenko, 1979).

A special feature of decision making at this level is that the decision rule had to be developed before any of the R & D proposals were submitted, so that the decision maker could assess the proposals as they reached him. Because of this, the decision maker had no opportunity to employ the characteristics of the R & D proposals which were actually submitted in formulating the structure of the R & D plan and thus determining the decision rule. Instead, he had to fix the concepts of his policy before the proposals started arriving and merely adjust it soon afterwards.

Another feature of the problem which had to be taken into account is that in this case there were **no** rigid limits on the resources necessary for conducting the R & D. This means that the problem studied here is not consistent with the general problems of portfolio optimization (Francis & Archer, 1971) or program selection within budget constraints (e.g., Buede & Peterson, 1977). The position here was that the authors of proposals would be able to secure the required resources (e.g., from state budget organizations) should the decision maker approve their R & D proposals. Rejection of a proposal was expected to force its proposers to formulate new approaches.

The decision maker's first task was to make a choice of a set of the best alternatives to be integrated into the R & D plan. His second task was to

compare both the accepted and rejected proposals in order to define the merits of the proposal developers. Hence, the decision maker was interested in ranking the R & D alternatives with respect to their utility. The introduction of DSS was a consequence of the desire of the decision makers to exert a stronger influence on the process of selecting the best R & D proposals.

STAGES IN THE ROUND

Each round in the decision making process involved the stages shown in Table 4. The proposers formulated the proposals so as to emphasize their merits (c.f. the discussions of cases 2 and 3, above). Inasmuch as the proposals were quite different and multidisciplinary by nature, they were too complex for the decision maker to be able to evaluate them directly. Consequently, in order to evaluate the alternatives he had to resort to the assistance of experts. These experts, however, were not required to make a general evaluation of the proposals but to answer explicit questions reflecting some or other aspects of the decision maker's scientific policy. So the need arose to develop a decision rule integrating the decision maker's scientific policy and the experts' judgements, and the method developed was supposed to be utilized by the decision maker.

The problem under study in each round constituted a choice of the best R & D alternatives to be included in the 3-5 year plan (one decision). The decision maker, the proposers, experts and consultants all contributed to the elaboration of this plan. The information concerning the set of criteria to be employed (see below) was available to everybody. The decision rule was developed by the consultants and the decision maker for the latter's use. The decision maker expected the consultants to submit explicit verifiable recommendations consistent with his policy. This placed specific constraints on the decision rule elaboration technique.

The traditional process of R & D formulation had previously involved the following three stages constituting a round:

Stage 1. R & D proposal formulation (by proposers)
Stage 2. Proposal evaluation (by experts)
Stage 3. Decision making (by the decision maker)

The new plan formulation procedure differed from the old one in that Stages 2 and 3 in the round were changed to the way shown in Table 4. The experts would now receive a special questionnaire and the decision maker would take decisions on the basis of the formulated decision rule.

As far as the proposers were concerned, the old and new procedures did not differ. The decision maker was the participant in the round who was most affected by the new procedure, as it qualitatively changed the entire style of his work. In practice the number of proposals considered in a single round ranged from several hundreds to several thousands. They comprised R & D alternatives largely representing applied research (i.e., they were oriented towards the solution of specific problems). The number of criteria employed by the decision maker did not exceed ten in any of the rounds and usually amounted to between five and seven.

It is important to emphasize the nature of these criteria. The choice among the R & D alternatives was considerably affected by hardly formalized factors such as "scale of R & D", "scientific backup", "versatility of expected results", "skill of potential researchers and developers", etc. In a word, the criteria were **qualitative** in nature.

Stages in the Round \ Participants	Proposers	Decision makers	Consultants	Experts	Computer
Stage 1: R & D proposal formulation	*				
Stage 2: Elaboration of set of criteria		*	*	*	
Stage 3: Estimation of projects				*	
Stage 4: Construction of decision rule		*	*		*
Stage 5: Construction of quasi-order of projects					*
Stage 6: Final decision		*			

Table 4
Stages and participants in rounds of Case 4 following the new procedure

(Note: Stage 1 is located at a lower level than stages 2 to 6, which constitute the top level decision making activities. Stage 1 activities continued at the lower level while the other stages were being implemented at top level.)

NECESSARY CHARACTERISTICS FOR DSS SUPPORTING TOP LEVEL R & D DECISION MAKING

The description given above of the requirements for a DSS in the rounds in this case indicates that the nature of this "top level" DSS must be qualitatively different from the lower level PSS's of Cases 2 and 3 or the single lower level DSS of Case 1 if we are to expect any success in its usage. Hence the consultants in this case developed a DSS for use by the decision maker which was distinguished by a specific way of describing the R & D choice situation in entirely qualitative terms on criteria, and a special way of obtaining a general estimate of the quality of R & D proposals through evaluating them on these multiple criteria. Details of this methodology are given in Larichev (1982). Here we present only a snynopsis of some principles underlying its design. These were:

(i) The description of the R & D choice problem was exercised in a language that allows one to structure many real-life problems through the use of verbal estimates of all degrees of quality on those scales on the criteria scales.
(ii) The DSS formulation could take account of the uncertainty brought about by the incomplete knowledge of decision implications at the time of decision making.
(iii) The description of the situation provided in the DSS through qualitative criteria constituted a verbal decision model reflecting the actual quality grades which the decision maker took account of in decision making and represented a language for communication between the decision maker and experts typical of that used in their work environment.
(iv) The set of criteria employed in the DSS was defined on the basis of the decision maker's desire to emphasize particular qualities which he considered substantial for a comprehensive evaluation of R & D projects. (The formulation of grades of quality on each of the criteria were developed by the consultants with the decision maker's assistance).

The description of a decision situation in the decision maker's usual language considerably increases his or her trust in the outcome of the decision analysis (Humphreys & McFadden, 1980). To maintain the trust, it is necessary to use this language throughout the decision rule formulation. Larichev (1982) describes how this was achieved in this case through treating the R & D general utility model as a rule according to which every combination of criteria estimates which might characterise a project is consistent with a certain class of quality which can be assigned to that project.

The resulting DSS procedures were designed to (i) provide for verification of the decision maker's preferences for stability and consistency, (ii) involve primarily those questions where the probability of obtaining reliable information was the greatest, and (iii) employ a method of preference elicitation from the decision maker which matched his "natural" way of making comparisons between characteristics of alternatives.

The principles outlined above are not in themselves a specific characteristic of DSS located at the "top" level of decision making. What is **specific** is the utilisation of the decision maker's language at every stage of the method, without any quantitive scales, scores, lotteries and so on; without any

transformation of the decision maker's preferences into numbers. What is also specific is how these principles are used to structure the DSS characteristics and interface with the decision maker in way that takes into account the language, motivation and responsibility of the decision maker, as well as the level of the organizational structure within which he works, and the way in which information relevant to R & D policymaking is communicated between levels.

EVALUATION OF DSS USAGE

In the rounds in Case 4 involving the DSS, decision makers trusted the results it provided in implementing their R & D policies. The two characteristics of the DSS which seemed especially important in generating this trust were: (i) all the resulting estimates were directly based on the decision maker's verbal information without any transformation thereof, and (ii) it was possible for the DSS to define a relative position of any pair of alternatives directly on the basis of data obtained from the decision maker.

We also have evidence from one of the rounds on the success of the system in terms of forecasting ability in proposal evaluation. At the end of this particular round, the recommendations obtained for a group of 700 proposals with the help of the DSS were, for a number of reasons, not implemented. Retrospective examination of the actual results of the R & D proposals which were subsequently chosen revealed that estimates provided by the DSS-based method were correct for 80% of the R & D proposals in the round.

SOME COMPARISONS

Table 5 summarizes some comparisons between the four case studies on the basis of the following major characteristic features:

- the number of alternatives taken into consideration in the DSS
- the number of criteria used in evaluating the alternatives
- the formal goals of the analysis, accepted by the decision maker
- the analytical tools used or constructed for solving the choice problem
- the participation, or otherwise, of decision maker in the analysis supported by the DSS
- whether there was identity between the criteria represented in the DSS and the real criteria controlling the decisions

The study of this table allows us to come to the following conclusions:

1. The choice of analytical tools used in the cases was related to the number of criteria and alternatives and to formal goals. In Case 1, use of decision tree methodology allowed the creation and evaluation of scenarios involving a small number of alternatives and the use of sensitivity analysis in selecting a best one. Cases 2 and 3 appear identical when viewed on these three factors and they are characterised by the same analytical tool - multiattribute utility assessment. In Case 4, the existence of a large number of alternatives defines an approach expressing the decision rule in terms of combinations of criteria estimates.

2. The last two factors in table 5 shed light on a major cause of success or failure of DSS implementation. Only the direct participation of the decision maker and precise correspondence of his expressed criteria with the real criteria controlling decision can provide a chance for the real implementation of a DSS aimed at supporting that decision maker's actual choice making. Systems supporting people who can hope only to influence these choices (as in cases 2 and 3) need to be aimed differently in order to ensure successful implementation.

Factors / Level	Number of alternatives	Number of Criteria	Formal Goal	Analytic tools	Did decision-maker participate in analysis?	Criteria in DSS = Criteria controlling decision?
Case 1:						
Corporate (UK)	small	small	Choice of best alternatives	Decision tree	Yes	Yes
Case 2:						
Corporate (Hungary)	medium	large	Select best projects	M.A.U. assessment	Yes	No
Case 3:						
Branch of industry (Hungary)	medium	large	Select best projects	M.A.U. assessment	No	No
Case 4:						
Planning for a number of research institutes (USSR)	large	large	order projects	Descriptive-normative multicriteria method	Yes	Yes

Table 5

Comparisons between four cases of R & D planning supported by DSS

DECISION SUPPORT SYSTEMS, ORGANIZATIONAL SYSTEMS AND DECISION METHODS

It is very important, in regard to the applicability of any particular DSS, that the users for whom it is intended be ready to apply it. Of course, a DSS comprising a more reliable and methodologically validated technique for comparison of decision alternatives has a greater chance of success in application. However, as we have seen above, reasons for succss or failure do not lie only in the merits or shortcomings of a procedure or a method.

At the outset, new DSS methods and procedures must be tuned to the existing organizational structure and to traditional ways of gathering and considering the inputs to the R & D planning system at the level at which the DSS is to be located. In penetrating such systems through successful long-term usage, DSS serve to change their essence sharply increasing the rationality and centralization of decision making.

There are also problems of a psychological nature in applying new methods and procedures involving DSS usage. R & D decision makers at all levels tend to share a number of old-fashioned views which hamper improvements to their traditional ways of working. One of them is a belief that a great number of R & D themes (up to several thousands) can well be analyzed directly. When faced with a variety of complex and different R & D proposals (as in Cases 3 and 4), such notions are far from realistic. Another notion is that having to choose can be avoided either through proportional allocation of resources to all the options, or by securing additional resources. Experience has shown that this unrealistic assumption can result in dissipation of resources. The third notion holds that the application of new methods and procedures must lead to a reduction of the decision makers' influence on decision making (see the discussions of proposers' expectations of DSS usage in Cases 2 and 3). Quite the reverse occurs where DSS methods adequately match the context within which they are employed.

R & D planning is characterized by complex problems which tolerate neither an approach which is too simple nor extreme formalization. The practical utility of DSS consists of its assistance to planners and only when we understand what is involved in providing such assistance will the new methods become a useful tool for improving existing systems of long range R & D planning.

Footnotes

1. These case studies formed part of a project of collaborate research coordinated through the International Institute of Applied Systems Analysis (IIASA), Laxenburg, Austria. Detailed descriptions of the cases, and of the methodology used in preparing the case studies were published by IIASA in 1982 as a collaborative paper series on Comparative analysis on application of Decision Support Systems in R & D decisions. Laslo David and Lawrence D. Phillips also contributed to the analysis of the cases described here.

2. These questions address **pre-requisites** for requisite modelling. Berkeley and Humphreys (1982) describe and discuss such types of uncertainty, all of which must be resolved in structuring a decision problem given that these pre-requisites are met. A 'requisite' DSS must also address these types of uncertainty in an adequate way.

3. In relatively unstructured situations like those typically found at the commencement of rounds in R & D planning, these typically determine the initial attitudes towards the DSS.

4. The development of the financial model was a stage in the round in itself, but conducted at a lower level than that examined here. Hence we cite only the output from this stage. Note that the business planning manager plays the role of expert at the higher level, but the role of proposer at the level of the team developing the financial model.

5. This followed from the consultants' goals from the round, which were quite different from the motivations of the decision makers, viz: (i) developing and testing new methods for real life problem solving, and (ii) proving that the information of the managers and experts can be effectively used in an organized communication process compatible with DSS.

6. This is in contrast with MEM where a single set of parameters was input to the model, and then varied interactively to take into account decision makers' expressed differences (MEM, Round 2, Stage 4).

7. Recall that at this higher level the director acted as **proposer**, rather than decision maker.

References

[1] Berkeley, D. and Humphreys, P.C., Structuring decision problems and the 'bias heuristic', Acta Psychologica 50 (1982) 201-252.

[2] Boichenko, V.S., Larichev, O.I., Moshovich, H.M. and Septalova, L.P., Hierarchical methods for goal-oriented planning of scientific research VNIISI, Moscow, (1978), in Russian.

[3] Bonczek, R.H., Holsapple, C.W. and Whinston, A.B. Specification of modelling knowledge in DSS (this volume).

[4] Buede, D.M. and Peterson, C.R., An application of cost-benefit analysis to the USMC Program Objectives Memorandum (POM), Technical Report 77-8-72, Decisions and Designs, Inc., McLean, Va. (1977).

[5] Czako, A. and Vari, A., A method for evaluation of R & D programs, Tudomanyszervezesi Tajekoztato, 1 (1980) in Hungarian.

[6] Francis, J.C. and Archer, S.H., Portfolio Analysis (Prentice-Hall, Englewood Cliffs, New Jersey, 1971).

[7] Handy, C.B., Understanding Organizations, (Second edition, Penguin,London,1981).

[8] Humphreys, P.C. and McFadden, W., Experiences with MAUD: Aiding decision structuring versus bootstrapping the decision maker, Acta Psychologica 45 (1980) 51-70.

[9] Humphreys, P.C., Vari, A. and Vecsenyi, J., Methods for analysing the effects of application of Decision Support Systems in R & D decisions. IIASA collaborative paper (1982)*

[10] Jaques, E., A general theory of bureaucracy, (Heinemann, London, 1976).
[11] Kahne, S., A procedure for optimizing development decisions, Automatica 11 (1975) 261-269.
[12] Keen, P.G.W. and Hackathorn, R.D., Decision support systems and personal computing, Technical Report 79-01-03, Department of Decision Sciences, Wharton School, University of Pennsylvania (1979).
[13] Kiss, R., The organization of open decision making, Bureau for Systems Analysis of the State Committee for Technical Development (1978) in Hungarian.
[14] Kiss, R. and Torok, L., A model and procedure for the analysis of complex systems on the basis of technical-economic criteria, Szigma 1 and 2 (1979) in Hungarian.
[15] Kunreuther, H., A multi-attribute multi-party model of choice: Descriptive and prescriptive considerations, in: Humphreys, P.C., Svenson, O. and Vari, A. (eds.) Analysing and Aiding Decision Processes (North Holland, Amsterdam, 1982).
[16] Larichev, O.I., A method for evaluating R & D proposals in large research organizations, IIASA collaborative paper (1982).*
[17] Larichev, O.I., Systems analysis and decision making. In Humphreys, P.C., Svenson, O., and Vari, A. (Eds.) Analysing and Aiding Decision Processes (North Holland, Amsterdam, 1982).
[18] Larichev, O.I., Zuev, Ju.A. and Gnedenko, L.S., Metod ZAPROS (ZAmkutyje PRotsedury u Opornyh Situatsii) reshenija slabostruk-surizovannyh problem vybora pri mnogih kriterijah, VNIISI (1979) in Russian.
[19] Mansfield, D.B., How to select successful R & D projects, Research Management 12 (1978).
[20] McCosh,A.M. and Scott Morton, M.S., Management Decision Support Systems (MacMillan, London, 1978).
[21] Phillips, L.D., Requisite decision modelling: A case study, Jrnl. Operations Research Soc. 33 (1982) 303-311.
[22] Souder, W.E., A system for using R & D project evaluation, Methods, Research Management Journal (1978).
[23] Vari, A. and David, L., R & D planning involving multicriteria decision analytic methods at the branch level, IIASA collaborative paper (1982).*
[24] Vari, A. and Vecsenyi, J., Decision Analysis of industrial R & D problems: Pitfalls and lessons, in: Humphreys, P.C., Svenson, O., and Vari, A. (eds.), Analysing and Aiding Decision Processes (North Holland, Amsterdam, 1982).
[25] Vecsenyi, J., Product mix development strategy making at the enterprise level, IIASA collaborative paper (1982).*
[26] von Winterfeldt, D., Structuring decision problems for decision analysis, Acta Psychologica 45 (1980) 71-94.
[27] von Winterfeldt, D., Pitfalls of Decision Analysis, in: Humphreys, P.C., Svenson, O. and Vari, Anna (eds.) Analysing and Aiding Decision Processes (North Holland, Amsterdam, 1982).
[28] Zuev, Ju.A., Larichev, O.I., Filippov, V.A. and Chujev, Ju.V., Problemy otsenki predlozhenii po provendeniju nauchnyh issledovanii, VestnikAkademii nauk SSSR (1979) in Russian.

* These papers may be obtained from: Distribution Section, Office of Communications, IIASA, A-2361 Laxenburg, Austria.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

AN EXPERT SYSTEM FOR DECISION MAKING

M. Bohanec (1), I. Bratko (1,2), V. Rajkovic (1,3)

(1) "J. Stefan" Institute, Jamova 39, Ljubljana Yugoslavia
(2) Faculty of Electrical Engineering, "E. Kardelj University, Ljubljana, Yugoslavia
(3) School of Organisational Sciences, Kranj University of Maribor, Yugoslavia

A new decision support system, developed as an expert system, is presented. The method is formally described and discussed. Its distinguishing feature is its human orientation which is mainly reflected in the system's ability to *explain* utility calculation. A corresponding computer implementation is presented together with a practical application in decision making.

1. INTRODUCTION

Expert systems are intelligent information systems that behave, in a certain sense, as a human expert in the application domain (e.g. Michie (1979)). A major new feature introduced by the methodology of expert systems is the system's ability to *explain* its decision in user understandable terms.

Expert systems are typically composed of two modules:

(1) a knowledge–base,
(2) an inference machine.

The knowledge–base contains the knowledge about a particular problem domain. The inference machine (a) solves problems stated by the user by using the knowledge–base, and (b) generates user–oriented explanations of the solutions.

The decision making process (DMP) can be treated as the selection of a particular alternative from a given set of alternatives so as to best satisfy some given aims or goals. The problem to be solved is to evaluate alternatives, e.g. to calculate their utilities. This can be done on the basis of the decision, or utility, knowledge which a decision maker or a decision system has (White (1978)). An expert system for DM has to establish an appropriate knowledge base and use it for utility calculation. In addition to this, it has to explain the way the utility was calculated.

The explanation of utility calculation is especially interesting because DM knowledge is subjectively defined, it offers different interpretations and has some degree of uncertainty. This kind of knowledge is usually referred to as "soft knowledge" (e.g. Expert Systems (1980)). Typically, soft knowledge is also poorly formalised, nonsystemised and often changes with time. When we are dealing with soft knowledge the explanation of results seems to be the only effective way of verifying them.

We believe that the main shortcoming of the existing DM techniques is their black–box behaviour. Usually this is accompanied by a complicated and inadequate aggregation of partial utilities and numerical coding of DM information (Alter (1980)). This leads to minimal possibilities for discussing the credibility of the final results which is fundamental not only for the verification of decisions but also for negotiation among different DM groups. As a consequence, it is usually difficult to handle changes in the DMP and there are no means for dealing with uncertain or incomplete information.

It seems that these problems can be resolved by a better fit between the decision maker and the DM method, using the approach of expert systems. In this paper a formal model of a novel DM method, DECMAK, is presented. The main emphasis is on the human factor. A computer implementation of the method is presented together with an analysis of its use in the DMP and a discussion of its practical applications.

2. A FORMAL MODEL OF THE "DECMAK" METHOD

The DECMAK method was gradually developed and tested on several practical decision making situations (Efstathiou, Rajkovic (1979) ; Rajkovic, Bohanec (1980)). The method is based on the following formal model. A *semantic tree* is a triple

$$(X, F, E)$$

where:

X is a set of *performance variables* $x_1, x_2, ..., x_n$ whose domains are $D_1, ..., D_n$;

F is a set of functions $f_1, ..., f_n$, $(m < n)$ from tuples of performance variables into performance variables:

E is a set of equations $e_1, ..., e_m$ of the form:

$$x_i = f_i (x_{i1}, x_{i2}, ...)$$

The set E satisfies the following conditions:

(1) there is exactly one variable which never appears as an argument of any of the functions; this variable is called the *root–variable* (or "overall utility"), all other variables are non–root variables;

(2) each non–root variable appears as an argument of the functions in the equations exactly once;

(3) each variable appears in the left–hand side of the equations at most once; the variables that never appear in the left–hand side are called the *leaves.*

The leaves are also called *basic variables.* All other variables are *aggregate variables.*

Note that the above constraints ensure that the equation set E can be represented as a tree as illustrated in Fig. 1.

The domains $D_1, ..., D_n$ of performance variables are discrete and finite. They typically consist of a few $(2 - 5)$ values. The values can be numerical or "descriptive", e.g.:

(poor, satisfactory, good)

$X = \{x_1, ..., x_{10}\}$
$x_1 = f_1(x_2, x_3)$
$x_2 = f_2(x_5, x_6)$
$x_3 = f_3(x_4, x_7)$
$x_4 = f_4(x_8, x_9, x_{10})$

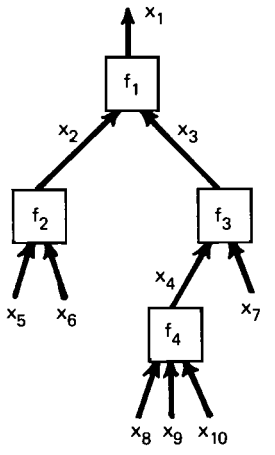


Figure 1.
An example of semantic tree. x_1 is the root variable (overall utility); $x_5,...,x_{10}$ are basic variables

All the domains are assumed to be ordered. Descriptive values are introduced primarily to facilitate a more direct representation of the user's understanding of the problem. So the user is not forced to transform his usual terms into numerical values. There is no explicit intention to use numerical values as absolute measures and descriptive values as relative measures.

The constraints imposed upon the set of equations E facilitate the evaluation of variables. Any variable x can be easily evaluated by implementing the following rule. If x is an aggregate variable then

$$x_i = f_i(x_{i1}, x_{i2}, ...)$$

If x_i is a basic variable then its value has to be supplied by the user.

In terms of DM this model is interpreted as follows: a decision alternative is specified by an instantiation of the basic variables. This is sufficient to compute the values of all other variables, including the root–variable which represents the "overall utility" of this alternative. Alternatives are compared through their overall utilities. Thus the root variable is the overall criterion for the final ranking of alternatives. The set of functions F and equations E define how the basic descriptive features of alternatives are combined into a single utility measure. The semantic tree has to be designed by the user–expert. We call this tree "semantic" because it defines the concepts of the problem–domain and the relations among them. The tree is in fact a representation of the expert's decision knowledge about the given decision problem.

The above formal model was motivated by the assumption that such a semantic tree is a natural form for representing decision knowledge and provides a suitable framework for the expert for systematically formalising his or her decision expertise. This assumption was confirmed by the applications of the method outlined in the sequel. Advantageous features of the model are: the variables are entirely defined by the user, and the user may use descriptive rather than numerical values. The tree structure facilitates a *gradual* aggregation of the basic–

variable values through aggregate variables. As the expert defines each function in F independently of other functions, he can focus his attention entirely on this local problem. In this way the whole problem of formalising the decision knowledge is decomposed into a set of subproblems (i.e. defining particular functions in F). Each of these can be kept reasonably small by limiting the number of arguments of the functions to a few variables only. In practical applications the number of arguments was typically between two and four.

The expert defines the functions in F by specifying the function value for some chosen argument values. This is done through interaction with the program which supports this process, as described in the following section. For the undefined points in the argument–domain, the system computes the function values by a simple interpolation method.

The semantic tree model has been extended for handling unreliable and/or incomplete information, for the case that the user does not have a complete specification of alternatives. Instead of dealing with precise (single) variable–values, the model can thereby deal with distributions of values. Distributions are defined by a list of values, each of them associated with its corresponding "certainty factor". Thus for example, if some basic variable describes the documentation of a technical system and the user is not sure about the quality of the documentation of some available alternative system, we can specify a "fuzzy" estimate: good with certainty factor 0.4, satisfactory with factor 0.2, poor with factor 0.1. The certainty factors can be normalised so that their sum becomes 1, and can then be treated as probabilities.

If the values of variables $x_{i1}, x_{i2}, \ldots$ are specified as distributions and

$$x_i = f_i(x_{i1}, x_{i2}, \ldots)$$

this results in a value distribution for x_i. In DECMAK, the certainty factors propagate through the semantic tree by two alternative rules for mapping the distributions of the values of the arguments $x_{i1}, x_{i2}, \ldots$ into the distribution for x_i. The first rule is borrowed from the theory of fuzzy sets (Zadeh (1965)), and the second from probability theory.

Let V_u be the set of all vectors $\vec{v} = (v_1, v_2, \ldots)$ such that if $x_{i1} = v_1, x_{i2} = v_2, \ldots$, then

$$f(x_{i1}, x_{i2}, \ldots) = u$$

Thus for any combination $\vec{v} \in V_u$ of argument values, the variable $x_i = u$. Let us denote the certainty factors of $v_1, v_2, \ldots$ by $c(v_1), c(v_2), \ldots$ and the certainty factor of $\vec{v} = (v_1, v_2, \ldots)$ by $c(\vec{v})$. Then by the first rule (borrowed from fuzzy sets):

$$c(\vec{v}) = \min_i(c(v_i))$$

By the probabilistic rule:

$$c(\vec{v}) = \prod_i c(v_i)$$

The certainty factor of $x_i = u$ is then by the "fuzzy rule":

$$c(u) = \max_{\vec{v} \in V} (c(\vec{v}))$$

By the probabilistic rule:

$$c(u) = \sum_{\vec{v} \in V_u} c(\vec{v})$$

In the present implementation of DECMAK the user can choose between both principles for certainty–factor propagation in the semantic trees. For an example see graphical illustrations to a practical example in section 4 where the min/max rules were used.

As for comparison of the two types of rules, the probabilistic rules are mathematically better justified. However, the fuzzy rules turned out to have practical advantages for the following reason. If the number of values is high the user may find it difficult to specify the probability distributions which would properly reflect his intuitive image, specially when the values are not independent.

3. THE COMPUTER IMPLEMENTATION OF THE MODEL

The main goal of the computer implementation of the DECMAK method was an appropriate man–machine communication, which makes the model alive in the sense of a creative partnership between man and computer. This is a distinguishing new quality in comparison with traditional decision support systems, where the role of the computer is mainly in transferring the documentation and calculation burdens from decision maker to computer.

In this section the achievement of the goal mentioned by the DECMAK program (Bohanec (1980)) will be explained.

3.1. Decision knowledge–base construction

Every decision subproblem, i.e. every function in F, can be treated separately. The decision maker (user) starts with a subproblem which he or she wants to solve. There is no prescribed order of dealing with decision subproblems. Let us take function f_3 (Fig. 1) as the first subproblem. Fig. 1 can represent a car selection decision situation. The overall utility of a car is x_1, x_2 can be the price and x_3 technical characteristics.

After user identification, the names of performance variables x_3, x_4, x_7 and corresponding domains D_3, D_4, D_7 have to be entered. In our case

x_3 = TECHNICAL–CHARACTERISTICS; D_3 = (poor, satisfactory, good, very good)
x_4 = COMFORT; D_4 = (bad, acceptable, good, very good)
x_7 = SECURITY; D_7 = (low, medium, high)

If values are descriptive, i.e. words, the corresponding compatibility functions can be entered (Zadeh (1975)); Efstathiou et al. (1979)). In the present system, compatibility functions are only used for graphical representation of results.

After this, a decision knowledge construction for our decision subproblem "TECHNICAL–CHARACTERISTICS" can start. There are three possibilities:

(1) The user states the values of arguments and the function value. This can be interpreted as formulating logical statements, or *rules,* such as:

> if SECURITY is high and COMFORT is very good
> then TECHNICAL–CHARACTERISTICS are very good.

(2) The program generates combinations of arguments and the user states corresponding values. This can be interpreted as questions:

> What is value of TECHNICAL–CHARACTERISTICS
> if SECURITY is low and COMFORT is very good?

(3) The user asks questions of the above type and the program calculates the answers using the linear multidimensional interpolation formula.

This method of knowledge–base construction can take place when some knowledge already exists in the base. In any case the user has to confirm a calculated value before it is entered as a new piece of knowledge. This is so not only because the simple interpolation formula may be inappropriate, but mainly because of handling possible discontinuities in the functions from F.

When knowledge for all the decision subproblems is so defined, the program builds up the whole semantic tree. The tree can be simply reviewed and revised whenever desired.

3.2. Evaluation of alternatives

Once a decision knowledge–base has been constructed, it can be used for the evaluation of alternatives. First the user enters the name of an alternative. Then the program asks for the values of all leaves – basic variables. In our case these are x_5, x_6, x_7, x_8, x_9 and x_{10}.

If the user is not certain about values of variables with respect to the alternative being evaluated he can put several values together with appropriate certainty or probability factors. For example:

SECURITY: high/0.8, medium/0.4

When all the leaf variables have been defined, evaluation can start. The program calculates the overall utilities as a single value. For example:

x_1 : OVERALLCARUTILITY: acceptable

If the data was of a fuzzy or a probabilistic nature, the overall utility is expressed as a distribution of values.

3.3. Explanation of evaluation

Once the final overall utility has been calculated, the usual question is: how and why was the utility obtained? The user can follow the utility calculation along the semantic tree. He follows the decomposition of overall utility into partial utilities. Every partial utility can be examined separately. This is done by displaying rules (points of a function f) which were taken into account during the utility calculation. Whether a rule already existed or it was calculated during the evaluation is also displayed.

Such an explanation is especially useful in group decision making where negotiation among different interests takes place. In this case the negotiation moves from overall utilities to distinguishing features of alternatives in corresponding nodes of the semantic tree.

3.4. Some technical data about the DECMAK program

The program is implemented on PDP–11 (under RT–11 or RSX–11 operating systems) and DEC–10 (under TOPS–10) computers. It is written in Pascal and can be easily transferred to other machines. Its size is 3500 lines (about 100 subprograms).

A special point of the implementation is its user orientation. It has a HELP system, and was characterised as a "friendly system" by its users.

4. A PRACTICAL EXAMPLE

4.1. A decision problem

One real decision problem, where the DECMAK method was used, was:

In a factory with about 2000 employees, a purchase of a computer system was planned to be used in their administration and research work. The change analysis showed that they needed a computer with about 140 interactive terminals, 10 printers and 700 Mbytes of disk storage. The decision problem was to choose a computer among alternative offers.

This problem was solved in the following steps:

(1) establishing a decision making group,
(2) change analysis,
(3) identifying alternatives,
(4) identification of performance variables and semantic tree construction.
(5) definition of the decision–knowledge functions,
(6) analysis of the alternatives and evaluation,
(7) explaining the results of evaluation,
(8) implementation of the chosen alternative.

Steps 1, 2, 3 and 8 are highly dependent on the problem environment. As they are not directly related to the DECMAK method itself they will not be discussed here. Steps 4 to 7 using the DECMAK method will be further discussed in more detail.

4.2. Performance variable identification and semantic tree construction

The performance variable SYSTEM (x_1) represents the quality of the computer (overall utility). Its domain is:

$$D_1 = \text{(unacceptable, acceptable, good, very good)}$$

The quality of a computer depends on economic conditions, technical features and personnel. So three new performance variables are introduced:

x_2 = ECONOM; D_2 = (unacceptable, acceptable, good, very good)
x_3 = TECHNICAL; D_3 = (bad, acceptable, good, very good)
x_4 = PERSONNEL; D_4 = (bad, acceptable, good, excellent)

and

$$\text{SYSTEM} = f_1 \text{ (ECONOM, TECHNICAL, PERSONNEL)}$$

Each of these variables depends on other characteristics, e.g. TECHNICAL depends on hardware and software.

By this top–down approach a semantic tree with 9 nodes and 20 leaf variables was eventually designed.

A semantic tree is the final result of the analysis of the decision problem (steps 4 and 5). If our knowledge about the problem is good enough, we can construct semantic trees of this size quickly – typically in 2 or 3 hours.

4.3. Definition of decision–knowledge functions

In our case we defined all the 9 decision–knowledge functions interactively using the DECMAK program. The usual heuristic for defining a function is:

(1) Enter rules which seem to be "realistic", that means that we expect the situation the rule fits. For example:

if ECONOM = acceptable
and TECHNICAL = good
and PERSONNEL = acceptable
then SYSTEM = acceptable.

(2) Let the program ask some questions which are close to the rule we entered.

(3) Test the quality of the current knowledge–base with some questions which we expect to occur during evaluation. Bad answers mean that we have to refine our knowledge–base by iterating this heuristic once more.

For the definition of all 9 functions in our case we spent 2 times 3 hours of interactive work with the DECMAK system. The function f_1, one of the 9 functions in our semantic tree, was thus specified as shown in Table 1.

Table 1.

An excerpt from the rule–defined function f, where SYSTEM = f (ECONOM, TECHNICAL, PERSONNEL).

ECONOM	TECHNICAL	PERSONNEL	SYSTEM
unacceptable	bad	bad	unacceptable
unacceptable	bad	acceptable	unacceptable
...	...	...	...
acceptable	good	good	acceptable
acceptable	good	excellent	good
acceptable	very good	bad	acceptable
acceptable	very good	good	good
acceptable	very good	very good	good
...	...	...	...
very good	very good	acceptable	good
very good	very good	excellent	very good

4.4. Analysis of the alternatives and evaluation

The next step in solving our decision problem was to determine the values of the leaf variables for all alternatives (5 computer systems in our case) and to evaluate them. When all the data are present, this is simple and straight–forward. But in some of the offers we could not find all the data that were needed for precise and certain evaluation. In such cases we had to define the value of a variable as a distribution of values that, in our judgement, best fitted the real situation.

An example: for one of the computers the capacity of the disk storage per unit was not explicitly specified. As we know nothing about its disk storage, initially we let the values for this parameter be:

DISKS : 0–100 Mb, 100 – 300 Mb, 300 – 600 Mb, more than 600 Mb

But on second thoughts we decided that the first and the last values were surely not possible, and that the most certain value was 100 – 300 Mb, but the size was possibly in the range 300 – 600 Mb, too. So we stated for the parameter DISKS the more precise value:

DISKS: 100 – 300 Mb/0.9, 300 – 600 Mb/0.4

The analysis of the alternatives and their evaluation (which was repeated with different values for the same alternative in order to check the sensitivity of the evaluation) took 2 times 2 hours of discussion and interactive work with the system.

4.5. Explaining the results of the evaluation

The evaluation of the computer systems gave us the following (note that the "fuzzy" rule for certainty–factor propagation was used):

COMPUTER	VALUE	CERTAINTY FACTOR
A	acceptable	1.0
B	unacceptable	1.0
C	acceptable	0.3
	good	0.7
D	acceptable	0.6
	good	0.15
E	good	0.5

Which of the systems is the best and why?

The first step is to answer the question: do the above results agree with our intuitive expectations? If they do not, we can go through the tree and the rules and check if the knowledge functions work properly. If not, we have to modify a function in question and to repeat the evaluation. In our case the results agreed with our expectations.

In the results we find that computer E got the highest single value (although with the relatively small certainty factor 0.5). Alternative C was also evaluated as *good* with an even greater certainty factor of 0.7, but it also got the value *acceptable,* which involves a greater degree of risk in choosing the alternative C. Graphical illustration of the situation is in Fig. 2. The vertical axis corresponds to SYSTEM overall utility and the horizontal to the grade of membership. The left diagram shows the fuzzy utility of computer E where "good" is defined by a fuzzy distribution of overall system utility values in the interval between 0 and 1. The certainty factor is taken into account by using the "min" rule (Zadeh (1975), Efstathiou et al. (1979)). The corresponding diagram for computer C is on the right side of Fig. 2.

As this situation does not look clear enough, further analysis is required.

We must look at the other utility values derived for the alternatives:

NODE	COMPUTER	VALUE	CERTAINTY FACTOR
ECONOM	C	good	1.0
	E	acceptable	1.0
TECHNICAL	C	good	0.7
	E	very good	0.6
PERSONNEL	C	acceptable	0.3
		good	0.7
	E	good	0.5
		very good	0.5

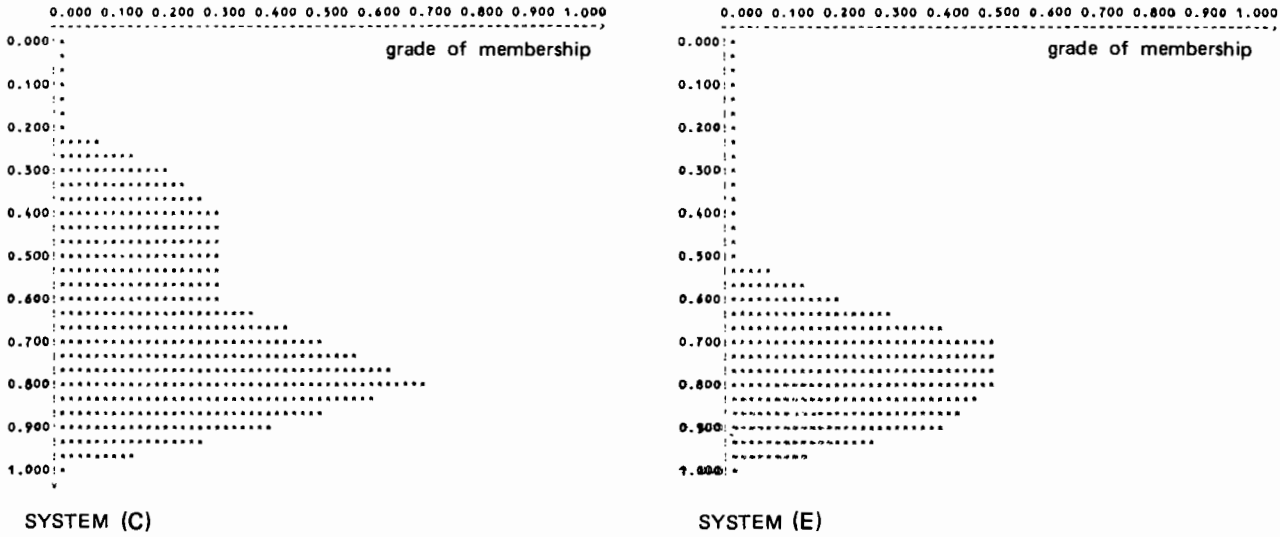


Fig. 2. Graphical illustration of overall utilities for systems C and E

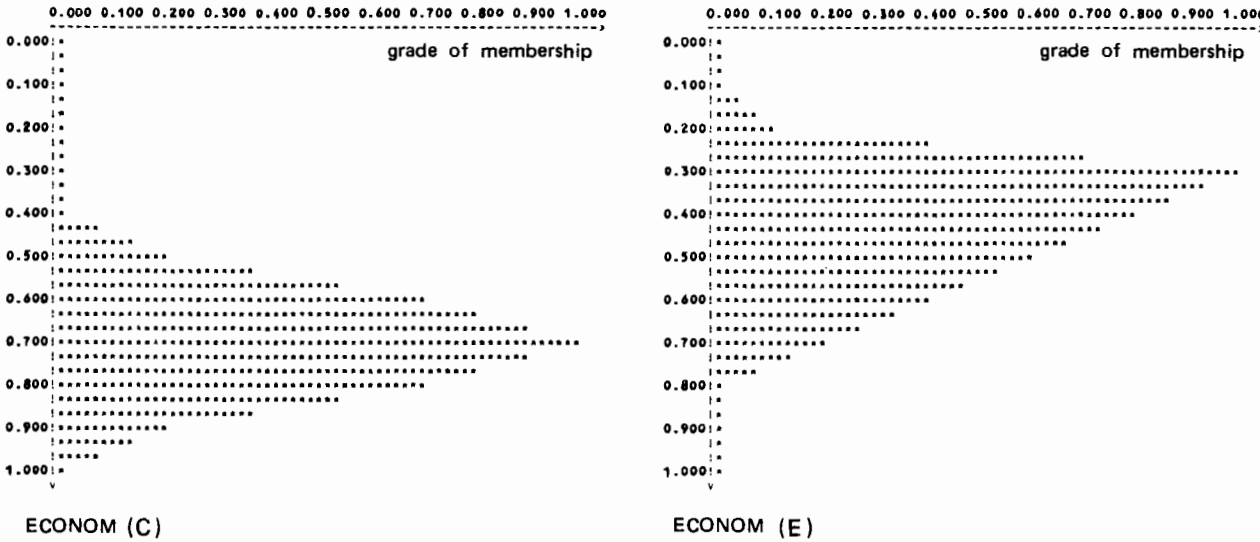


Fig. 3. Graphical illustration of economic conditions for systems C and E

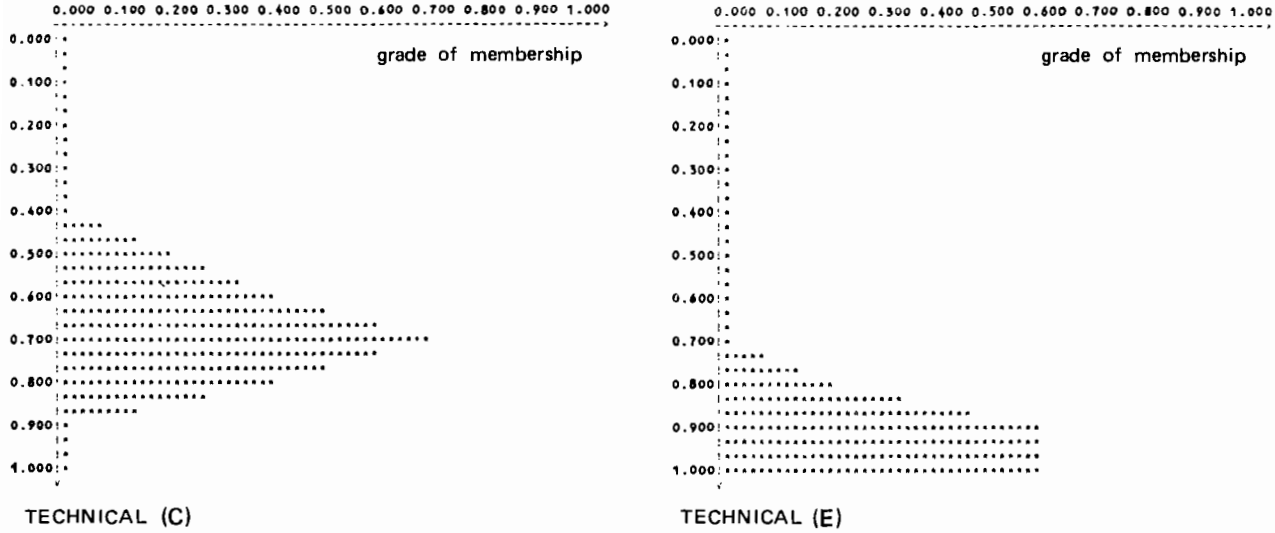
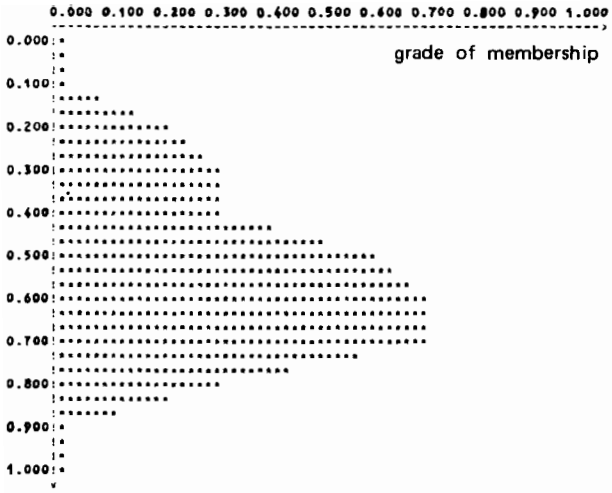


Fig. 4. Graphical illustration of technical characteristics of systems C and E



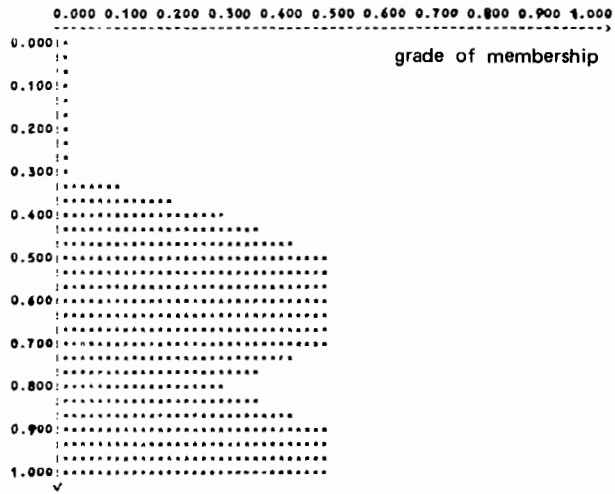


Fig. 5. Graphical illustration of personnel requirements for systems C and E

So, in technical characteristics and personnel, E is better and in economic criteria C is better. The same conclusion can be drawn from Figs. 3, 4 and 5 which show the fuzzy distributions of values of ECONOM, TECHNICAL, PERSONNEL for both computers. If we agree that technical characteristics and personnel of the above values are more important than economic criteria, then E should be chosen, otherwise C. If we still cannot decide, further similar top–down comparison can be done.

5. CONCLUSION

The main advantage of the DECMAK system, working as an expert system, is in its user (decision maker) orientation. The method is fully transparent for ordinary users. This can be explained by the following features of the method:

- direct aggregation of utilities,
- possible usage of descriptive variables and values,
- interactive use of the DECMAK program,
- utilities can be expressed as distributions of values,
- easy change handling,
- participation of different interest groups,
- explanation of utility calculations.

For implementation of expert systems a new way of thinking is needed. This should be different from the traditional black–box decision making where a computer is needed primarily for number crunching and represents a barrier behind which a user's opinion may be manipulated.

The DECMAK system was used in several decision making situations such as computer hardware and software selection. The above mentioned features are particularly important in group decision making. In these cases, decision problems should be well structured and documented. Explanation of the decisions is extremely useful in negotiation among different decision interests. The possibility of forgeting important things is reduced. Crucial points are recognised which are essential in the implementation phase of the alternative chosen.

REFERENCES:

[1] Alter, S., Decision Support Systems: Current Practice and Continuing Challenges (Addison Wesley, 1980).

[2] Bohanec, M., DECMAK – Program Documentation and User's Guide, IJS Report DP-2266, J. Stefan Institute, Ljubljana (1980).

[3] Efstathiou, J., Hawgood, J., Rajkovic, V., Verbal Measures and Inseparable Multidimensional Utility in System Evaluation, in: Schneider, H.–J. (ed.), Formal Models and Practical Tools for Information Systems Design, Holland, 1979)

[4] Efstathiou, J., Rajkovic, V., Multiattribute Decisionmaking Using a Fuzzy Heuristic Approach, IEEE Trans. on Systems, Man and Cybernetics, 9 (1979) 326–333.

[5] Expert Systems, Proc. 69th Infotech State of the Art Conference (London, 1980)

[6] Michie, D. (Ed.), Expert Systems in the Microelectronic Age (Edinburgh, University Press, 1979)

[7] Rajkovic, V., Bohanec, M., A Cybernetic Model of the Computer Aided Decision Making Process, Proc. of 9. Int. Congress on Cybernetics, Namur (1980), 185–189.

[8] White, J.D., Decision Methodology (J.Wiley & Sons, 1978)

[9] Zadeh, L.A., Fuzzy Sets, Information and Control, 8 (1965) 338–353.

[10] Zadeh, L.A., The Concept of a Linguistic Variable and Its Application to Approximate Reasoning – I, II, III, Information Sciences, 8 and 9 (1975), 199–251, 301–357, 43–80.

Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

CAP: A DECISION SUPPORT SYSTEM FOR THE PLANNING OF PRODUCTION LEVELS

C.A.Th. Takkenberg

Faculty of Economics
University of Groningen
The Netherlands

CAP (Computer Aided Planning) is a DSS for the planning of production levels. The major tools are linear programming and model experimentation. This paper presents a case history and evaluates CAP as a DSS. The major recommendation is to start the development of DSS with the construction of a descriptive model.

1. INTRODUCTION

This paper describes a case study concerning the improvement of decision processes related to the planning of capacity and aggregate production in a Dutch factory.
We think that the failures and successes encountered in this project contribute to the improvement of a methodological framework and a strategy for development of DSS.
This paper consists of three parts. Firstly the case history will be told without "touch up" or rearranging of the several phases. It takes us through the steps actual taken, delaying the stimulus for comments to the next part. Finally, we make some recommendations in view of lessons learned.

2. A CASE STUDY

The major events and processes of the case study are described briefly in the following subsections.

2.1. The first trigger

When the development of CAP started, the author was a member of a small team with special interest in planning and control of operations. We had adopted a particular view of planning systems: with the emphasis on the hierarchy between long term planning, medium term planning and short term planning. We considered the technique of mixed integer linear programming as superior to other techniques in formulating and solving planning problems. Undoubtedly our view was operations research flavoured.
The plea for our ideas during the discussion with the industry in our region was convincing enough to result in a contact with the manager of the Central Planning Department (CPD) of the product division "small domestic appliances" in the summer of 1975. This manager was responsible for the planning of sales as well as the

planning of the manufacturing departments. He stimulated the sales department to produce realistic forecasts of sales and controlled the level of production in such a way that inventory was kept within certain limits. His staff suggested a possible improvement in the planning process concerning the level of production.
We made an arrangement to perform a pilot study in one of the factories.

2.2. Description of organizational procedures

The planning of production levels is performed for production families. Each month the sales department supplies a sales forecast for each product family with a horizon varying between 13 and 24 months and sub-periods of one month. The varying horizon is due to the procedure to make a forecast for the total amount to be sold in the current year and the total amount for the following year. With the aid of historical data and using commercial insight, the total for a year is broken up into monthly data. The sales department also establishes the desired level of inventory for each product family at the end of the current and the following year. This desired inventory is expressed as a percentage of yearly sales. It should be reached by December 31.
It is very simple to calculate the desired total production for a year if the starting inventory is known. Each month the desired total production for the current year and the following year is calculated for all product families. It will be clear that the sales department determines the amount of production.
The production planner has the task to level production in such a way that the targets will be reached. We did not pay attention to the way this was performed and invested much time in formulating a Mixed Integer Linear Progamming (MILP) model.

2.3. The MILP approach

A broad class of problems -including certain non linear and boolean relationships- can be solved by using the MILP-technique. Nowadays the mathematical formulation and generation of the computer input is no longer a problem. Sophisticated algorithms and large computers give us the opportunity to handle problems with e.g. 130000 variables and 8300 constraints utilizing the APEX-III package on a CDC 170/760 main frame. The impact of improved technology and software on the computer time required was tremendous during the last decade. We generate and solve models we did not even consider in our most sanguine dreams five years ago.
However, MILP algorithms are notorious for their unpredictability of the required computer time. In general, the cpu-time consumed is relatively long and as a rule of thumb 100 integer or binary variables is the upper bound for a problem requiring a reasonable throughput time. The throughput time of the developed MILP-model on a CDC 7416 turned out to be a severe constraint, observing that the planning staff always felt the need to explore other alternatives. At that time (76-77) our computer and software facilities were inadequate to tackle this kind of planning problems and even now the throughput time for a MILP-model is too long if a prompt response is required.
Another frustrating feature of LP as well as MILP is the disaster caused by an infeasible solution. Improved software gives a good trace of infeasibilities for an expert in OR, but for a member of a planning staff it will require a lot of training to find the origin of the infeasible solution. Moreover, planners have no interest in these matters and consider them to be outside their province.

2.4. The LP approach

The main argument for using MILP was the consideration that each family groop was manufactured on production lines with a fixed output over a certain time period. The number of production lines had to be an integer value. This production system required frequent switching of capacity between family groups and caused "shocks" in production levels. From 1976 this situation slowly changed into a more flexible one. The production manager introduced special production lines with a variable output per time period. The optimal solution of the continuous part of the LP-model already gave the planning staff a rough indication of the alternatives to be explored and the introduction of the flexible production system gave opportunities to approximate the continuous solution. Therefore the MILP-problem was reduced to a LP-problem. However, in most cases the planning staff had arguments to modify the LP-based plans. A lot of their comments led to the improvement of the LP-model. In general the LP-generated plan turned out to be a preliminary plan, it was mostly modified by the planning staff. Therefore the staff felt the need to acquire a tool for evaluating the effect of alternative plans, a simple tool calculating the aggregate production, inventory and service level for each month with the number of production lines being the input. This led to the introduction of a facility for evaluation of alternatives.

2.5. The model experimentation approach

The process of using models for the generation and evaluation of alternatives is called "model experimentation". Evaluation means the judgement of alternative plans. The input for this process is the specification of a plan concerning the utilisation of capacity i.e. the number of production lines and the production rates of this lines in each period. The calculation of the aggregate production in a certain period is based on the number of production lines, the production rates of the lines, and the amount of hours to be consumed in this period. A starting inventory at the beginning of the period and a forecast of sales for this period leads to the calculation of the expected value of the inventory at the end of the period. This inventory is compared with a desired level of inventory, reflecting a required service level. In the evaluation of alternative plans, the planners gave much attention to the service level to be expected by the execution of the plans. Up to that time the LP-alternatives got the qualification "nice to have". It did not have an important impact on decision making. The evaluation of the effect of a self specified plan required less occupation of the core memory than LP and produced the reports for the whole factory within one minute throughput time. The priority rules for the job scheduling of the CYBER 7416 we used at that time caused a throughput time of one day for LP, although the consumed cpu-time of LP was about two minutes.
The first software developed was written in SIMULA 67. This is a very powerful language for modelling complex processes. The model experimentation required simple calculations and reporting. A conversion to PASCAL was made, reducing the response time during calculating and reporting to some seconds. A planning procedure as sketched in figure 1 got established. The start was always the solution of the LP-model and the analysis of the LP-report. After that several "simulation" trials were made until a satisfactory solution was found. The new facility evoked a great enthousiasm and the introduction was a major event in the project. The monthly planning process turned out to be an interesting game. The fast

response to "what if" questions stimulated the generation of new alternatives and the parties involved -material management manager, production manager and the planning staff- got consensus within some hours.

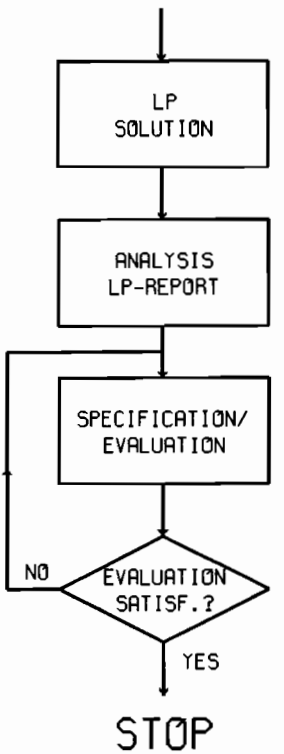


Figure 1 Planning procedure

2.6. Computer system design

In the mean time we write the year 1978. The two systems developed -generation/solving/reporting of LP and calculating/reporting of model experimentation- were quite different. Especially the LP approach required a lot of preparation to adapt the model monthly to new data, for a great part the same data as required for the evaluation. The idea rose to create a small special data base, feeding both the LP-run and the model experimentation process. Moreover, the system for model experimentation and reporting was extended with a procedure generating the mathematical formulation of the LP model. The extended program got two modes of operation, the generation of the current LP-model as well as the calculation and reporting of the effect of plans.

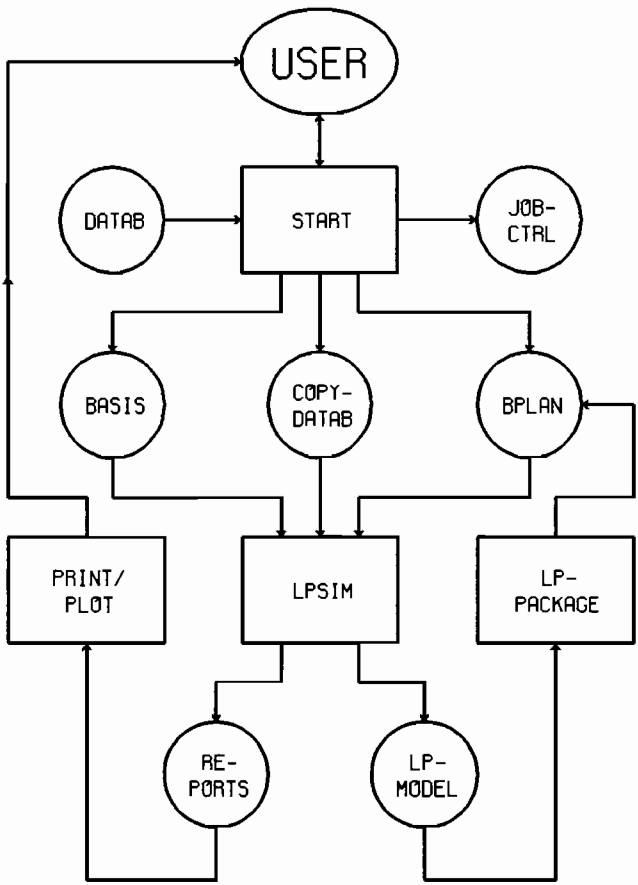


Figure 2 Interconnection of programs and files of CAP

Figure 2 gives an outline of the organization of programs and files. A program is represented by a rectangle and a file by a circle. In both cases the reading of the data base (DATAB) is the first activity. The interface software (START) discriminates between the experienced and unexperienced planners with an adapted dialogue from the program START. By this program the user specifies the mode of operation (LP or SIM) and selects the product families and reports required. This information is written on the file BASIS. The sequence of the directives for the CYBER is written on the file JOBCTRL. A copy of the data base -possibly a modified copy- is created and written on the file COPYDATAB. After a dialogue with the program START the system performs the directives in the sequence established by START. The user is unaware of the execution of complex programs and handles the system in figure 2 like a black

box. When LP is selected, the program LPSIM reads the files BASIS and COPYDATAB and generates the mathematical formulation of the actual LP-model and writes this information on the file LPMODEL, being the input for the LP-package. This package includes a matrix generator [1], the APEX-III package of CDC and a reportgenerator [2]. The LP-model is solved and the relevant output -the number of production lines as suggested by the LP-model- is written on the file BPLAN. The program LPSIM is triggered for the second time to calculate and report the effect of the LP based BPLAN. The preparation for printing and plotting is performed by special packages. The prints and plots are subject of analysis by the user. When SIM is selected the user has to specify BPLAN in a dialogue with START and the remainder of the procedure is the same as the second phase of the use of LP.
Up to now a terminal with a print facility is used to print reports and the plotting is performed on an off-line plotting installation. The installation of a CDC 170/760 main frame in 1981 improved the system with respect to response time and interactive facilities. An LP-model with about 2000 variables and 1000 constraints is solved in the interactive mode within some minutes throughput time, (including generation of the model and reporting) and the model experimentation mode gives a response within some seconds.

2.7. Trial for evaluation

Attempts were made to prove that the new planning procedures as sketched in figure 1 were better than the application of the old procedures. Historical data were gathered for the years '78 and '79. The sets of data of eight planning situations contained the exogeneous variables -being the input for the planning process- as well as the endogeneous variables -being the output of the planning process, i.e. the plan.
Trials were made to compare the value of the targetfunction of the optimal LP-solution with the value of the same function when evaluating the historical plans. However, it is impossible to compare both systems, because of different horizons. LP uses a horizon of 13 to 24 months and historical data cover a horizon of 3 to 12 months.
This problem could be solved by opening of the black box of the former planning process to uncover the procedures of the historical planning process. In acquiring an algorithm describing this process a comparison between the new and old procedures comes within reach as will be shown.

2.8. The descriptive model

A model was constructed describing the process of the planner to arrive at a satisfactory solution of this planning problem. Fortunately, there was one particular person who had performed the planning for many years and we hoped his experience had lead to a stable procedure.
The interviews with the planner did not bring about rules or procedures applied. However, some hypothesis governing the "thinking" process could be formulated. The main consideration in the search for rules of the planner was the conviction that according to the findings of Simon [3] decision making of man is simple minded. The differences between an actual state and a desired state will be reduced by applying instruments guided by priority rules to parts of the problem. When all parts are solved and there is no dependency between the sub-problems the complete problem is solved. The hypotheses were the following:

a. The family group to be considered first is the largest one (with respect to the amount of total production in a year) with a shortage of capacity.

b. A change of levels is performed with a lot of instruments, like changing the output of a production line, utilisation of capacity in reserve and switching capacity between family groups. In the search of the planner the simple tools will be considered first.

c. The planner maintains current levels of production as long as possible, but at December 31 the desired targets must be reached.

It required much programming and many computer runs to build a model of the decision process of the planner. This model is a detailed description of the decision making by a computer program. It was striking that simple rules generated complicated time series.
The descriptive model was validated in two ways. The first one was a simple counting of turning points, the change of levels in the time series. About 70% of the turning points occurred at the same time and in the same direction as in reality. Moreover about 50% of all timeseries showed an exact similarity with reality. The second way was an expert -or Turing- test with a team of two experts with many years of experience in factory planning. They tried to classify reports of 8 planning situations. For each planning situation, two reports were generated, one reflecting the actual decisions taken and one based on the outcomes of the descriptive model. The experts tried to classify the genuine and simulated documents. It was quite a surprise that in 6 out of 8 situations the simulated document was seen as genuine. This supports a hypothesis of Bowman and Kunreuther [4], [5]. They state that management assessments suffer from inconsistency due to the influence of environmental clues. The experts were supplied with the starting points of the planning process, but without environmental clues. In practice, there have been truly special circumstances that influenced the plans. The descriptive model does not take special circumstances into account. The experts recognized more of their own solution in the simulated documents than in the real documents, not being disturbed by special information.

2.9. Comparison between new and old procedures

The descriptive model gave us the opportunity of a quantitative comparison between new and old procedures. By applying the old procedures twice -for the current year and for the next year- the horizons of new and old procedures are both between 13 and 24 months. We believe that the "hard" figures are only a small part of the improvement. Firstly a quantitative comparison is made and secondly a brief discussion of qualitative improvements is given.

2.9.1. Comparison of the value of a goal function

The LP-model is designed as a goal programming device. The values of the penalty categories of the goal function can be compared with the values of the corresponding descriptive model.
The data of 1978 and 1979 were used to generate the LP-solution as well as the solution of the descriptive model for 8 situations. To compensate for differences in inventory at the end of the planning period a kind of "cost price" was calculated, i.e. the total penalty over the planning horizon divided by the total amount of production. The LP-based "cost price" was taken as a standard and the improvement of LP over the results of the descriptive model was calculated as a percentage of the LP-based "cost price".

This figures are tabulated in Table 1 for 8 cases in 1978 and 1979. In order to show the improvement by implementing a longer horizon comparison was made with "cost prices" calculated with penalties accumulated up to the end of the current year (the old horizon) and penalties accumulated till the end of the next year.

Table 1 Improvement of LP over the descriptive model

HORIZON->	CURRENT YEAR	NEXT YEAR
PLANNING DATE	%	%
Jan 1978	7.4	10.2
Feb 1978	9.0	8.2
Sep 1978	-3.8	6.8
Jan 1979	1.3	4.9
Feb 1979	2.3	5.9
Mar 1979	2.4	5.0
May 1979	5.0	8.0
Sep 1979	0.4	6.7
Average	3.0	7.0

Almost every experiment shows the improvement of LP over the descriptive model. The impact of the extended horizon is very clear. A more detailed analysis points out that the improvement lies for a great part in the lower investment in inventory and a reduction of lost sales. The major shortcomings of the procedures in the descriptive model can be stated now with more evidential force.

-the old procedures gave insufficient attention to the inventory levels between the moment of planning and the end of the year. The planner just implicitly aims at the desired inventory level of December 31 by meeting the requirement of the desired total production in the actual year,

-the planning horizon in the old procedures is obviously too short in some situations. Again the syndrome of December 31 is the culprit.

2.9.2. Some qualitative improvements

(1) Supply of reports.

The planning staff considers the reports as very valuable as they are a subject of discussion during the planning meeting. The reports visualize matters that stayed invisible before, evokes comments and stimulates the search for new alternatives.

(2) Improvement of co-ordination.

In the old situation the sales department dictates the total production without a weighing of production possibilities. In the new situation the amount of production is established in such a way that a compromise is found between desired inventory levels and the possibilities of the production unit.

(3) Sensitivity analysis and avoidance of uncertainty.

In the old situation the desired inventory at the end of the year was expressed as a percentage of sales in the same year. This percentage is set partly arbitrarily and is influenced by the uncertainty of future sales. The new system is not able to reduce the uncertainty, but enables the planners to perform a sensitivity analysis visualizing the impact of optimistic and pessimistic views.

An improved version of the LP-model contains an option to generate plans assuming an attitude of expectation. Instead of the immediate but often small changes of the level of production after the "frozen" periods, the new option creates a tendency to delay the response to exogeneous disturbances thus avoiding unnecessary changes of the production level.
(4) Support the planning of personnel.
Work measurement determines the relation between the output of production lines and the need for personnel. Special reports visualise the need for manpower, and thereby support the planning of personnel.

3. EVALUATION OF CAP IN THE LIGHT OF DSS

When evaluating CAP one may question its degree of membership to the set of DSS.
We think CAP is a DSS because of the following aspects:

- -it applies information systems technology,
- -human judgement is necessary,
- -the emphasis is on supporting rather than on replacing the decision maker,
- -the support concerns the solution of ill-defined problems,
- -the improvement with CAP lies in better plans. Although the planning process is performed faster, the emphasis is on improving effectiveness rather than efficiency.

The author recognized a certain degree of conformity with the view, the design proposal and the tools, as recommended by Keen and Scott Morton [6].
Keen and Scott Morton present a broad perspective integrating several views. We endorse the statement that the DSS-designer will not restrict himself to one view and has to recognize the impact of:

- -the rational view,
- -the satisficing view,
- -the organizational procedures view,
- -the political view,
- -the individual difference perspective.

We also consider the meshing of the descriptive and the prescriptive perspective in the design process as being of particular importance. In the development of CAP a wrong modelling sequence was chosen. In conformity with the design proposal of Keen and Scott Morton every DSS-project must start with descriptive modelling. The design of prescriptive models is the next step.
Sprague [7] considers three levels from which a DSS can be viewed, namely the level of the specific DSS, the level of the DSS generators and the level of DSS-tools.
When viewing CAP in the light of these levels, it obviously is a specific DSS and applies DSS-tools.
In our opinion, the level of DSS generators is very broad. One can imagine a DSS generator to build all kinds of DSS, a general purpose

DSS generator. One can also imagine a DSS generator capable of building a small class of special DSS.
In the design of CAP it was foreseen that it should be usefull for more than one factory. The structure of the data base was chosen in such a way that it reflects the structure of family groups. Moreover, the data base contains "knobs" to tune the LP and model experimentation to a wide variety of situations. By specifying the data base, the software generates the suitable models. Therefore in our opinion CAP is both a specific DSS and "a specific DSS generator".

4. RECOMMENDATIONS DUE TO LESSONS LEARNED

When reviewing this project to gather matters with a potential impact on future development of DSS we consider the following being worth to be subject of discussion.

4.1. The necessity of descriptive modelling

We recommend that every DSS-project must start with the development of a descriptive model. OR experts tend to start with prescriptive models generated behind their desks instead of penetrating the confusing battle field where the real decisions are taken.
The descriptive model is the base for

- -the understanding of the problem,
- -the definition of objectives to be used in prescriptive laws,
- -demonstration of possible improvements,
- -the facilitation of the process of change.

4.2. The importance of validation

The design proposal of Keen and Scott Morton may be improved by inserting a validation of the descriptive model. It gives the model a kind of quality label and raises the scientific level of statements concerning possible improvements when using the model as a reference.

4.3. The importance of removing cognitive constraints

We consider the visualizing of matters that stayed invisible as the major improvement of CAP. This visualizing stimulates the search for better alternatives and accelerates the process for acquiring consensus in group decision making. We believe this "visualizing" being a general aspect of DSS and recommend an emphasis on research to facilitate man-machine communication.

4.4. The power of model experimentation

Major DSS-tools in CAP are LP and model experimentation. The continuous solution of LP gives a sufficient base for the search for a satisficing solution. We believe that it will never be possible to solve ill-defined problems with LP alone.
Organizational decision making often has to deal with an uncertain and political environment. We consider model experimentation as a more powerful tool to meet these aspects.

REFERENCES

[1] Tijssen, G.A., Beschrijving MATGEN, Unpublished report, University of Groningen. (1981).

[2] Bink, A.J., Beschrijving REPGEN, Unpublished report, University of Groningen. (1981).

[3] Simon, H.A., Administrative behavior (Macmillan, New York, 1976).

[4] Bowman, E.H., Consistency and Optimality in Managerial Decision Making, Management Science 9(1963) 310-321.

[5] Kunreuther, H., Extensions of Bowman's theory on managerial Decision Making, Management Science 15(1969) b415-b439.

[6] Keen, P.G.W. and Scott Morton, M.S. Decision Support Systems, (Addison Wesley, Reading, Mass., 1978).

[7] Sprague, R.H. Jr, A framework for research on DSS, In: Fick, G. and Sprague, R.H. Jr. (eds), Decision Support Systems: Issues and challenges (pergamon Press, Oxford, 1980).

ISBN 0 444 86569 1