Working Paper

Selected issues of design and implementation of Decision Support Systems

Marek Makowski

WP-91-16
June 1991

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Selected issues of design and implementation of Decision Support Systems

Marek Makowski

WP-91-16
June 1991

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

This Working Paper discusses the problems related to the design and implementation of Decision Support Systems (DSS) based on the experience of the Methodology of Decision Analysis Project. It summarizes selected research activities and discusses different approaches to the development of DSS. The methodology developed and used in various applications made both at IIASA and by the collaborating institutions is presented and the selected problem of designing and implementation of DSS are discussed. Short descriptions of the software for DSS developed in cooperation with the MDA Project for various types of problems are given. The paper also provides information about recent activities of the MDA Project.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

Abstract

The paper presents selected issues related to design and implementation of model based Decision Support Systems (DSS). For over ten years the SDS Program has been involved in cooperation with various projects at IIASA and in collaborating research institutes. This cooperation has resulted in the development of many DSS, which in turn stimulated research on the theory and methodology of decision analysis. An overview of selected DDS developed within the cooperation with IIASA is presented. Different concepts of DSS are briefly discussed and one specific type of DSS, namely model based, aspiration–led DSS is characterized. Finally, selected problems of designing and implementation of a DSS are discussed in more detail. A short description of software packages developed within the cooperation with the MDA Project is provided. The paper also gives a short summary of recent activities of the Methodology of Decision Analysis Project and of the DSS software available from the MDA Project.

Contents

Selected issues of design and implementation of Decision Support Systems

Marek Makowski

1 Introduction

The paper contains a summary of selected methodological issues and of experiences from applications and is intended to stimulate a discussion between the scientists that have different approaches to Methodology of Decision Analysis as well as various experiences with actual applications of this methodology. This paper discusses the following issues related to the design and implementation of model based Decision Support Systems (DSS):

Methodology of Decision Analysis project at IIASA: A short overview of activities of the MDA Project at IIASA will be given. The MDA activities have been strongly related to the development of theory, methodology and software for DSS by many researchers from several countries.

Concepts of DSS: The concept of DSS is widely used but no established definition exists. Therefore the characteristics of DSS, rather than a strict definition, will be presented. These characteristics are selected from the point of view of real-life applications in different areas.

Model based, aspiration–led DSS: The integral part of this type of DSS is a mathematical model which can be used for predicting the consequences of decisions either proposed by a Decision Maker or computed by DSS. It is also assumed that decision making has a quantitative character, i.e. different decisions can be compared via criteria values. The concept of aspiration level (reference point) for criteria values is used because practical experiments with this approach indicates that the *language of aspiration levels* coincides very well with the actual decision making processes.

Selected applications: For over ten years the SDS Program has been involved in cooperation with various projects at IIASA as well as at collaborating research institutes. This cooperation has resulted in the development of many DSS, which in turn stimulated research on the theory and methodology of decision analysis. An overview of selected DDS developed within the cooperation with IIASA will be given. The experiences from this research will be briefly summarized.

Designing and implementation of a DSS: Based on lessons learned from selected applications, an approach for the designing and implementation of DSS will be proposed. Functions and architecture of the model based, aspiration–led DSS will be discussed. Organization of the development process of DSS will also be briefly summarized.

DSS software: A number of software packages developed within the cooperation with the MDA Project are available for non-commercial research purposes. Short descriptions of these packages is provided.

2 MDA Project at IIASA

Decision making is one of the most complex human activities. In most important decision-making situations it is necessary to take a number of competing and contradictory factors into account; moreover, there are usually complicated links and relationships between various alternatives. Frequently decisions have to be made without sufficient background information in the face of considerable uncertainty, which needs to be taken into account. One way of assisting the decision maker is to provide him with computerized tools capable of evaluating the various alternatives. These tools are known as decision support systems. On the one hand, such tools consist of standard mathematical algorithms, and on the other hand, they also depend on the needs of the tools to create the necessary models. Although the methodology of decision support systems is already well advanced, various extensions related to theoretical and methodological issues require further analysis. An especially important issue relates to the methodology and tools for computer implementation of decision support systems.

The Methodology of Decision Analysis Project is an application-oriented project which focuses on computer-aided decision making and is devoted to developing methodology, software and applications of decision support systems concentrated primarily around interactive systems for data analysis, interpretation and multiobjective decision making, including uncertainty analysis and multicomputer group decision making. The project's results are applied at IIASA and utilized by other institutions in the NMO (National Member Organization) countries of IIASA.

The objective of the MDA Project is not only to advance theories and methodologies of decision analysis but also to convert them into usable tools that could easily be used by decision makers in solving real-life problems. An important goal is to develop tools that are simple to use, user friendly and robust. The project requires considerable effort and cooperation on the part of scientists representing the quite different fields of interest. The only way to successfully perform this task, therefore, is by creating a rather strong international network of collaborating institutions. The greatest part of the work to date has been done within this network, in particular, by the Bulgarian Academy of Sciences, the Japan Institute of Systems Research and the Polish Academy of Sciences.

Although the MDA Project has officially existed for only a few years, related activities have been an important part of the SDS Program since the very beginning of IIASA. A substantial part of the SDS activities have been following the principle saying that theoretical and methodological research should be strongly connected to applications to sufficiently complicated, real-life examples. This resulted in feed-back between converting existing theories and methodologies into tools also usable for solving real-life problems and stimulation of the development of DSS methodology. In the past, many projects at IIASA have been decision-oriented, while many today remain so. This provides more opportunities for applications and development of theory. Let us mention just few examples: the Food and Agriculture Program, the Energy Program, Integrated Regional Development, the Computer Integrated Manufacturing Project, and the Transboundary Air Pollution Project. Later on, we will discuss selected applications that were also implemented within cooperation with these programs and projects.

Conferences play an important role in promoting discussions and innovative results, as well as further topics of research. The MDA Project participates in the organization of Workshops that are devoted to problems of Decision Analysis. For the past few years, small workshops on *"Advances in Decision Analysis"* have been organized during the summer at IIASA. The next Workshop in this series is scheduled for July, 1991. On September 9–13, 1991, *The IIASA Workshop on User-Oriented Methodology and Techniques of Decision Analysis and Support* will take place in Warsaw. This Workshop is aimed not only at stimulating scientific discussion and cooperation on the related topics. Its primary goal is to disseminate the results obtained so far within the cooperation with the Polish Academy of Sciences and to obtain feedback from decision and policy makers in business and administration.

The MDA Project attempts to provide software developers with the possibility of discussion and exchanging experiences. A meeting for this purpose was organized in April 1990. Another activity of this type is *The International Competition in Decision Support Systems* proposed and coordinated by an executive committee led by Prof. van Hee. A specific type of a decision situation (constrained resource scheduling problem) has been selected for this exercise. Each participating team has developed a DSS for this class of problem and has provided a real-life example of this class. According to the agreed specifications, each DSS should be capable of solving examples provided by all of the teams. Finally, eight research teams from five countries took part in this exercise. The exercise will be summarized in a special issue of the *European Journal of Operational Research.* The exercise was considered by all participants to be very useful. Therefore we plan to organize more exercises of this type.

The last MDA activity we are going to present is the distribution of the software developed within the cooperation with research teams in different countries. So far, the most advanced are the results of the project on *Theory, Software and Testing Examples in Decision Support Systems* – developed by the team organized by Prof. A. Wierzbicki in Poland. The basic topics of this research were: converting existing and developing new DSS software (robust, efficient and user friendly), providing applications to real-life problems, and development of methodology. The book summarizing the theoretical and methodological developments of this activity was published by Springer [L5]. Several DSS software packages (developed for IBM compatible PC's for different classes of problems) are distributed free-of-charge for non-commercial research usage to institutions in countries which are members of IIASA (cf Section 7 for details). The last report on the on-going research is presented in [R4].

3 Concepts of DSS

The concept of *a Decision Support System* (DSS) is widely used in both research and in many different applications, but it is not yet uniquely defined (cf e.g. [A2, D1, H2, L1, T1]).

The acceptance of probably the broadest definition of DSS (*"DSS is anything that supports decision making"*) results in the already famous inference that a cup of coffee or an efficient secretary are also DSS (cf [L1]).

Many authors have expressed concern (cf e.g. summary given by Davis in [D1]) that the misunderstanding and misuse of the concept of DSS may eventually result in its demise as a distinctive management tool. On the other hand, there are many successful applications of DSS in fields that differ remarkably (e.g. management and engineering design). It would be very difficult to give a representative list of applications of DSS

in different fields, therefore we have selected only a few publications that can serve as examples of both the development of DSS methodology and its applications: [A2, D1, E1, F1, G6, G7, H1, H2, K4, K6, L1, L2, L5, M11, M13, N2, N3, S2].

The concise overview of concepts and definitions of DSS is given by Lewandowski and Wierzbicki in [L4]. Instead of discussing the possible definitions of a DSS (and its relations to related concepts such as Management Information Systems, Data Management Systems, Expert Systems, Artificial Intelligence, Intelligent Decision Support, etc.) let us quote the following statement by Davis [D1]:

> *Differences in our professional experience and backgrounds can cause the same concept to be discovered and applied in many areas only by different names. It takes a long time to realize that a "Rose is a Rose" by any name.*
>
> *The manager is rarely concerned with the title used to classify an automated system. The important criteria is whether the system provides the features and capabilities necessary to support a particular process.*

Therefore we will not contribute to the discussion on the possible definitions of DSS. Let us also simplify further the considerations by limiting them in this paper to decision situations with a single person that makes a decision.

To avoid possible misunderstandings it is necessary to present the basic characteristics and features of the class of DSS we will be dealing with. Let us start with a brief discussion of the environment in which a DSS may be used. The key person in this consideration is an individual who uses a DSS in real-life situations. By convention such a person is called *a Decision Maker* (DM). By this term we mean a person who makes real decisions (depending on the application it may be a manager or an engineer or an operator) or an expert or an advisor. Decisions are taken within a *Decision Making Process* (DMP), which, in situations that justify the usage of a DSS, is a complex sequence of distinctive stages preparing and evaluating decisions. Making a rational decision often requires access to and the processing of a large amount of data and logical relations which (due to the nature of the problem) cannot be replaced by intuition. In many situations it is not a small task to examine even the possible range of feasible alternatives. We assume that a decision is finally to be made by the DM and a DSS does not replace or control a DM. In other words, DSS is not aimed at the automatic selection of decisions.

The following characteristics of DSS implies a class of DSS we will be dealing with:

- A DSS is a supportive tool for the management and processing of large amounts of information and logical relations which helps the DM to extend his habitual domain (cf [Y2]). The design of a DSS should be consistent with actual decision making contexts since a DSS is a tool which must fit to an existing environment of a DMP. The purpose of a DSS is not to change this environment substantively, therefore a DSS uses and provides only that information that is recognized by a DM as important and available during a DMP. Obviously the introduction of a DSS changes this environment qualitatively because it is aimed at assisting in making faster and more accurate decisions.

- A DSS is a problem dedicated system which is designed for a specific decision making problem. A DSS is usually designed for a specific DMP environment and it is often tuned for a specific DM.

- A DSS is composed of appropriate hardware and several mutually linked software modules, which usually include user interface, data base, software for formulation and modification solution of a related mathematical programming problem.

- A DSS should not be a *black box type* tool which asks a series of questions, then checks the consistency of answers and then finally suggests a solution which the designers of the DSS considered to be the best one. The structure and functioning of a DSS must be such that a DM understands and accepts them.

- The user interface of a DSS should be designed in such a way that a DM may obtain from the DSS information and answers for questions that he considers important for a DMP. A DSS does not only serve to help in reaching a single decision, but it also helps a user during an entire DMP. A DSS should be designed in a way that justifies its use. It should not only be easy to use, but should also provide information consistent with a DMP, should not restrict (without explicit decision of a DM) a possible range of decisions, should allow for examination of consequences of any decision, accept changes in the DM's preferences, etc.

- A DSS can be considered as a tool which, under full control of a DM, performs the cumbersome task of data processing and provides relevant information that enables a DM to concentrate on this part of the DMP which cannot be formalized and automatized. The data processing may mean quite different tasks depending on a particular DMP or its selected phase, e.g. it may be relatively simple data retrieval and analysis or solution of a complicated optimization problem. A DSS brings together human judgment (expressed by a DM during interaction with a DSS) and computerized information for improving the quality of the final decision (e.g. when evaluation of alternatives or determination of decision variables values requires processing of data and logical relations which cannot be replaced by intuition).

Hopple [H2] suggests that *"The human-machine symbiosis is a hallmark of a genuine decision support system"*. The above listed characteristics are the necessary main conditions for such a symbiosis. By no means are they sufficient conditions. There is no general specification of sufficient conditions for the implementation of a DSS since this obviously depends on a particular environment of a DMP. The key element of this environment is habitual domain of a DM (cf [Y2]). Recognition and understanding by a DSS developer of the DM's habitual domain is essential for design and implementation of a DSS.

One of the possible approaches is to design a DSS in which part may or may not be used depending on the evaluation of the current situation by a DM, who uses another part of the DSS for this evaluation. An example of an industrial application of such a hybrid DSS is given by Otsuka et al. in [O2].

Another important issue for understanding the DSS concept is the idea of *rational decision*. However, this is to broad of a topic to be discussed in this paper. There is a large bibliography which covers this problem. A good overview of the different concepts is given e.g. by Lewandowski and Wierzbicki in [L4], by Keeney and Raiffa in [K5], by Rapoport in [R2] and by Yu in [Y2].

Finally, one should be aware of the limitations of computer based tools applications to real-life decision making. On the one hand, there are many successful implementations of DSS in many quite different areas. On the other hand, there are serious arguments against being too optimistic in proposing wide application of such tools in possibly all areas of decision making processes. A good discussion of the related problems is given by Dreyfus & Dreyfus (cf [D2]). Instead of hopelessly trying to summarize the problems discussed in this book let us extract only one quotation:

The computer literacy is not just knowing how to make use of computers and computational ideas. It is knowing when it is appropriate to do so.

Seymour Papert [P1]

4 Model based, aspiration–led DSS

For the discussion of the model based, aspiration–led DSS concept, let us assume the following decision making situation:

- A well-defined part of a DMP (for which a DSS is to be implemented) can be represented in the form of a mathematical programming model. Decisions have quantitative characters and therefore can be represented by a set of the model variables, hereafter referred to as decisions[1] $x \in E_x$, where E_x denotes a space of decisions.

- The model defines (usually implicitly) a set of admissible decisions $X_0 \subseteq E_x$. Therefore x is admissible, iff $x \in X_0$. The set X_0 is usually not fixed since a DM can change values of constraints and/or parameters[2]

- The model can be used for predicting the consequences of decisions proposed by a DM or computed by DSS. The feasibility of decisions given by a DM should be assessed (DSS provides with $x \in X_0$, if a feasible solution exists). The prediction of the consequences can usually be represented by a mapping $y = f(x) \in E_y$, where E_y is a space of possible consequences (or outcomes) of the decisions.

- The consequences of different decisions x can be evaluated by values of criteria $q \in E_q$, where E_q is a space of criteria (sometimes referred to as outcomes, goals, objectives, performance indices, attributes, etc.). A partial preordering in E_q is usually implied by the decision problem and has obvious interpretations, such as the minimization of costs competing with the minimization of pollution. However, a complete preordering in E_q cannot usually be given within the context of a mathematical programming model. Usually we can assume that E_q is a subspace of E_y, that is, that the DM might select some criteria q_i between various outcomes y_j. The problem of selecting criteria is discussed in more details in Section 6.4.

The decision problem boils down to a selection of *the best* admissible decision x. The key problem here is to understand what *the best* means for a DM who actually makes a decision. This problem will be briefly discussed in Section 6.4.

The first natural approach has been based on the application of optimization models. However, it has become obvious that the specification of a single-objective function, which adequately reflects preferences of a model user is perhaps the major unresolved difficulty in solving many practical problems as a relevant optimization problem. This issue is even more difficult in the case of collective decision making. Multiobjective optimization approaches make this problem less difficult, particularly if they allow for an interactive redefinition of the problem.

The problem of a rational choice of a decision has been extensively discussed in a number of publications. A discussion of different approaches to this problem is given e.g.

[1] For the sake of brevity we call decision variables simply decisions.

[2] Some authors consider such a change as the definition of a new model.

by Wierzbicki [W4], Lewandowski and Wierzbicki [L4], Yu [Y2]. We will therefore only briefly comment on the most strongly established rationality framework which is based on the concept of *the value function* (cf e.g. [K5, Y1]).

The concept of a value function assumes that it is possible to construct a function which maps elements of the criteria set E_q into R^1 in such a way, that a larger number corresponds to the stronger preference. There are many fundamental and technical difficulties related to the identification of the value function which adequately reflects the preferences of a DM. But an even more important argument against the application of the value function concept was given in 1957 by Simon [S3], who pointed out, against all traditional economic concepts, that people look for satisfying solutions instead of one which maximizes the expected utility.

Simon formulated [S4] another rationality framework, called *bounded rationality* or *satisficing decision making.* This framework has been extented further by many researchers (cf e.g. a summary given by Lewandowski and Wierzbicki in [L4]). One of the main directions in that field was set by Wierzbicki [W3], who formulated the *principle of reference point optimization* in multiobjective optimization and decision support. That principle has been extended by Wierzbicki (cf [W4, W5, W6]) to *principles of quasisatisficing decision making* and has been extensively used both in research and in applications (cf [L5, R4]). Parallelly, Nakayama also developed a similar method called the satisficing trade-off method (cf [S1]). Similar approaches and their extensions have also been elaborated and applied by many other researchers (cf e.g. [I1, K7, K8, K9, L2, N1, N3, S8]).

For an illustration of the methodology applied in the model based, aspiration–led DSS, let us formulate a simple example of a linear programming type problem. Assume the following decision problem related to an improvement of environment quality (this is a simplification of the real-life problem made for the illustration of the methodological approach). There are n sources of air pollution each discharging an amount x_i of a pollutant[3]. Therefore the decision variable is $x \in R^n$. The air pollution transport model (cf e.g. [A1]) may be applied for the calculation of deposition of pollution in different places $y \in R^m$, where m is the number of places in which the pollution concentration is measured. Let us first consider the problem of finding the minimum cost strategy of emissions that would result in the concentration of pollution y that is not greater than the given admissible level $\bar{y}$. Therefore the corresponding model may be formulated as follows:

$$\min f(x) \tag{1}$$

$$Ax = y \tag{2}$$

$$l \leq x \leq u \tag{3}$$

$$y \leq \bar{y} \tag{4}$$

where $f(x)$ is a linear[4] function which gives the costs associated with keeping emissions at level x and constraint (3) corresponds to technological or economical constraints, which imply lower and upper boundaries for emission values. Application of classical LP optimization would obviously result in a solution which satisfies the constraints $y = \bar{y}$ (because in practical situations other constraints are "less binding"). However, in practice the resulting costs are too high to be covered. Therefore the classical approach offers the

[3]For the sake of simplicity we assume only one type of pollution.

[4]This function is in fact piece-wise linear. Minimization of such a function can be easily transformed to an LP problem, therefore LP formulation was assumed here for simplification of the discussion.

formulation and solution of the related problem in which the goal function (1) is replaced by

$$\min \max_{k=1,\ldots,m} y_k \tag{5}$$

with the constraint (4) replaced by:

$$f(x) \leq \bar{c} \tag{6}$$

where $\bar{c}$ is a given value of a budget for decreasing emissions. Solutions of both related LP problems for different values of $\bar{c}$ and $\bar{y}$, respectively, may help to find out the acceptable emission levels and corresponding costs. Although the solution of the series of LP problems sounds simple, it is actually a time consuming and cumbersome process.

This example illustrates the typical decision situation in which one has to deal with more than one criterion. Since classical LP formulation allows for the formulation of only one criterion (goal function), the main objective is selected as the performance index whereas other objectives are converted into constraints whose values are treated as a parameters. For such problems it is natural to formulate and deal with multiobjective optimization.

Now let us formulate the corresponding aspiration–led model. The model is composed of equations (2) and (3) which correspond to the model of air pollution transport and to the technological constraints on emissions, respectively. Those two sets of constraints define the set of admissible decisions X_0. We also define two criteria: first, corresponding to costs (1) and second, representing the environment pollution (5). Assuming that these two criteria properly reflect the goals of a DM, we may assume that a rational decision would be one of the proper Pareto-optimal solution (i.e. such a solution that there is no other solution for which one can improve the value of any criterion without worsening the value of at least one other criterion).[5]

To illustrate the problem of the analysis of Pareto solutions let us briefly discuss the problem in the criteria space (cf Figure 1).

First, one can solve the corresponding optimization problem for each criterion separately. More precisely, we use here an achievement function in which all other criteria enter with a small weight to assure that a Pareto solution is obtained (otherwise only weakly Pareto-optimal points (e.g. point B' in Fig. 1) can be guaranteed - cf e.g. [M8]). This gives the initial, and important, information about the possible range of criteria values. In our case, one should expect that for any rational decision the amount of cost will be between C_{min} and C_{max} and that the pollution level will be between P_{min} and P_{max}. If this is not acceptable one should verify the model assumptions (most probably[6] the constraints (3)).

After the two single-criterion optimization runs we have obtained two solutions represented by points $\{A, B\}$. We can also define in the criterion space the utopia point U composed of the best (obtained for respective single-criterion optimization) values of criteria. The point is named *utopia* because in practical situations it is not attainable. However, the set of Pareto solutions (in the criteria space) depicted as segments between points A and B is not known. In order to provide more information about the possible range of values (for Pareto-optimal solutions) for each criterion, we can also define the nadir point N composed of the worst values of criteria obtained for a series of single-criterion

[5]For the sake of brevity we will refer to properly Pareto-optimal solutions as to Pareto solutions (unless otherwise mentioned).

[6]Assuming that the transport equations are properly estimated.

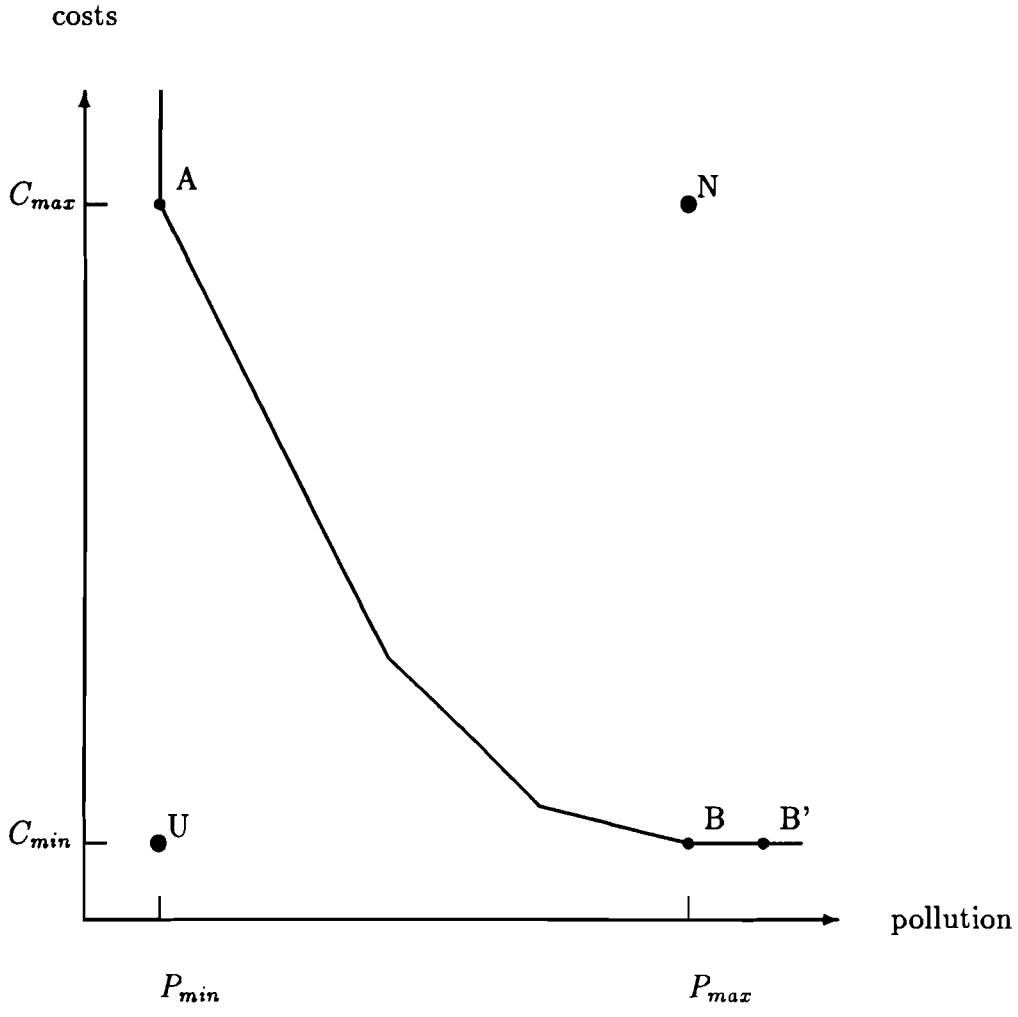

Figure 1: The illustration of the analysis of the Pareto surface.

optimization runs for each criterion[7]. The utopia and nadir (or its approximation) points can also be used for the automatic scaling of criteria, which is especially important in cases for which ranges of criteria values are of different magnitude for different criteria.

The set of all properly Pareto-optimal solutions is usually difficult to determine. Even the determination of Pareto solutions that are on vertices is a complex task. Steuer (cf [S9]) has developed a set of programs which serves for the computation of all Pareto solutions that are on vertices for LP problems. However, usually the determination of all Pareto solutions is not necessary. A DM usually prefers to analyse solutions that have values in a certain part of the criteria space. There are many methods for the scanning and analysis of solutions on the Pareto surface. The most natural method which best corresponds to a real-life DMP seems to be the method based on the *aspiration level* (sometimes referred to as *reference point*) concept (cf [L4, W1, W3]).

[7]There are some technical difficulties for the determination of the nadir point, if there are more than two criteria and if nadir point is defined as the point composed of the worst (over a set of all Pareto-efficient solutions) values for each criterion. Therefore for practical applications the definition applied above seems to be justified for the sake of analysis of a Pareto set and for the optional definition of scaling factors for the achievement function.

This approach may be summarized in the form of the following stages:

1. The DM specifies a number of criteria (objectives). In typical applications there are 2–7 criteria. For an LP problem a criterion is often a linear combination of variables but criteria may also have a form specific for an actual application (cf below).

2. The DM specifies an aspiration level $\bar{q} = \{\bar{q}_1, \ldots, \bar{q}_{NC}\}$, where $\bar{q}_i$ are desired values for each criterion and NC is a number of criteria. In some applications the DM may additionally specify the reservation level, which is composed of the worst values of criteria that a DM would like to accept.

3. The problem is formulated as minimization[8] of a (piece-wise linear) *achievement function* that can be interprated as an ad-hoc non-stationary approximation of the DM's value function depending on the currently selected aspiration levels. Then the problem is transformed into an auxiliary parametric LP problem. Its solution gives a Pareto-optimal point. If a specified aspiration level $\bar{q}$ is not attainable, then the Pareto-optimal point is the nearest (in the sense of a Chebyshev weighted norm) to the aspiration level. If the aspiration level is attainable, then the Pareto-optimal point is uniformly better than $\bar{q}$. Therefore this approach may be considered as an extension of the goal programming. Properties of the Pareto-optimal point depend on the localization of the reference point (aspiration level) and on (optional) weights associated with criteria. Some applications offer the option of computing weights based on utopia and nadir points, which usually provide good scaling in the criteria space.

4. The DM explores various Pareto-optimal points by changing either the aspiration level $\bar{q}$ or/and weights attached to criteria or/and other parameters related to the definition of the multicriteria problem.

5. The procedures described in points 3 and 4 are repeated until a satisfactory solution is found. Additionally, the user can temporarily remove a criterion (or a number of criteria) from analysis. This option results in the computation of a Pareto optimal point in respect to remaining "active" criteria, but values of criteria that are not active are also available for review.

The utopia and nadir points help to define reference points in the procedure outlined above because it is reasonable to expect values of each criterion to lie between utopia and nadir point.

To give a more formal presentation, let us introduce following notations:

NC is the number of criteria

q_i is the i-th criterion

$\bar{q}_i$ is the aspiration level for i-th criterion

w_i is a weight optionally associated with i-th criterion

ε_m is a given non-negative parameter.

[8]It can be also formulated as maximization problem, depending on the interpretation of the achievement function.

A Pareto-optimal solution can be found by the minimization of the achievement scalarizing function in the form

$$\max_{i=1,\ldots,NC}(w_i(q_i - \bar{q}_i)) + \varepsilon_m \sum_{i=1}^{NC} w_i q_i \rightarrow \min \tag{7}$$

This form of achievement function is a slight modification of a form suggested by Wierzbicki [W1]. Note that for $\varepsilon_m = 0$ only weakly Pareto-optimal points can be guaranteed as minimal points of this function. The use of a very small ε_m results (except in situations in which reference point has some specific properties) in a properly Pareto-optimal solution with trade-off coefficients bounded approximately by $\varepsilon_m NC$ and $1/\varepsilon_m NC$. If ε_m is very small, these properly efficient solutions might differ very little from weakly efficient (Pareto optimal). On the other hand, too big values of ε_m could drastically change properties associated with the first part of the scalarizing function.

A user may define any number of criteria. To facilitate the definition of different types of criteria some multi-objective optimization packages provide pre-defined types of criteria. For example, for a user of the Hybrid package [M8], six types of criteria are available. Two types of criteria are a simple linear combination of variables. Four other types of criteria correspond to various possible performance indices often used for dynamic problems.

A k-th criterion q_k is defined in one of following ways:

Type MIN

$$q_k = \sum_{t=1}^{T}\sum_{i=1}^{n} a_{it}x_{it} \rightarrow \min \tag{8}$$

where n is the number of (state and control) variables, T is the number of periods, if a dynamic problem is considered, ($T = 1$ for a static problem);

Type MAX

$$q_k = \sum_{t=1}^{T}\sum_{i=1}^{n} a_{it}x_{it} \rightarrow max \tag{9}$$

Type SUP

$$q_k = \max_{t=1,\ldots,T}(x_{it} - \bar{x}_{it}) \rightarrow \min \tag{10}$$

where x_i is a selected variable, $\bar{x}_i$ —its reference trajectory

Type INF

$$q_k = \min_{t=1,\ldots,T}(x_{it} - \bar{x}_{it}) \rightarrow \max \tag{11}$$

Type FOL

$$q_k = \max_{t=1,\ldots,T}(abs(x_{it} - \bar{x}_{it})) \rightarrow \min \tag{12}$$

where *abs* stands for a function returning the absolute value of its argument.

Type DER

$$q_k = \max_{t=1,\ldots,T}(abs(x_{it} - x_{it-1})) \rightarrow \min \tag{13}$$

For the sake of illustration let us consider x_{it} as being the flow of a river in point i at time period t, $\bar{x}_{it}$ would be a reference trajectory for the criterion of type FOL or a minimal and maximal reference trajectories for criteria of types INF and SUP, respectively. The criterion of type DER may be used to account for decreasing fluctuations of flows which is sometimes undesirable. Obviously the criteria given above may be easily modified (e.g. by introduction of weights) whenever a modification fits to a real-life DMP. Other types of criteria may also be predefined for a specific problem.

The conversion of the multiobjective problem which uses the types of criteria listed above into the corresponding single-criterion LP problem is given in [M8].

5 Selected applications

The applications discussed here have been selected in order to illustrate the usability of the DSS methodology outlined in the previous section to different types of problems and in different decision making environments. It should, however, be stressed that these examples illustrate applications of only one of the available approaches, namely the model-based aspiration-led methodology. This selection is not aimed at implying that this approach is the only one recommended for any possible type of DMP. This short survey is also by no means an attempt to give even a brief overview of all the representative applications. Such an attempt would be beyond the scope and the aim of this paper.

As it has been pointed out earlier, the methodology and applications of DSS have been developed, partly independently, by many different scientific communities. In many approaches there are similarities, but obviously many methodologies differ substantially. This overview is not aimed at suggesting that the selected examples are better applications than those which are not presented here. On the contrary, the examples have been chosen to a large extent to illustrate what elements of applied approaches should be changed.

The first group of presented applications consists of those made several years ago. These applications do not conform to the assumptions for the DSS design and implementation which are discussed in the next section. The main reason for being so had been the lack of the appropriate computer hardware and software at that time, which enables the implementation of the proper user interface. Another reason was due to the lack of experience in real-life applications of DSS. It is, however, worth briefly discussing this group of applications for two reasons. Firstly, they have provided the basis for the development of both the theoretical background and the methodology of designing DSS. Secondly, those old implementations well illustrate the differences between old and modern user interfaces.

One of the first implementations of the model-based, aspiration-led DSS was the forecasting and planning of the development of the Finnish forestry and forest industry sectors [K2]. This implementation was done at IIASA in cooperation with the appropriate Finnish partners and was based on a linear dynamic model of interaction between those two sectors. At the same time, the cooperation between the SDS and Food and Agriculture Programs at IIASA, and several institutions in Poland, has resulted in applications of DSS to planning and controlling agriculture production in Poland (cf e.g. [M4, M6]). Another activity organized by the SDS Program and the Integrated Regional Development Project at IIASA, with the cooperation of Bulgarian, Polish and Swedish scientists, has resulted in case studies in those countries on modelling the expansion and management of water systems (cf e.g. [M3]). Later, a similar methodological approach was applied within the cooperation of the SDS and Energy Programs at IIASA for the planning of energy supply strategies [G3], which, in turn, led to other applications in the analysis

of future relations between the energy sector and the rest of the economy [S10] and of future gas trade in Europe [M11]. Applications have also been made for regional investment allocation in Hungary [M1], selected issues of macroeconomic planning [G4] and for problems of environmental protection of ground water quality [K1]. Two other activities were started in the early 1980s, namely applications to chemical industry planning [G2] and to flood control [K10]. Both of these applications have been implemented in Poland in cooperation with the SDS Program and were the basis for the second type of the DSS discussed here.

All of the above mentioned applications can be characterized by the following features:

- Each application was problem oriented and was developed for a real-life application.
- For each application a corresponding mathematical programming model and specialized software were developed.
- The user interface was batch oriented for most applications, i.e. the off-line analysis of results had to be made, then a data file had to be edited and a new problem was generated. For most applications, separate modules were developed which served first as preprocessor which generated input files for stand alone solvers (e.g. LP packages) and then as a postprocessor which processed data provided by the solver to the form suitable for the analysis. Mainly general purpose packages have been used as solvers, which implies the necessity of providing input in the specific formats. The problem of providing data for a solver was easier in few cases when either the access to the part of source code that handle input data or the internal data format was available to software developers. For some problems a specialized solver was implemented. Most of these DSS provided a kind of shell which facilitated the usage of the software, but due to the software design and the hardware capabilities no real-time interactive utilization for real-life application was possible.
- The most limiting feature of these applications was the fact that their utilization require good skills in operating computers and in most cases additional knowledge of modelling and numerical techniques. Therefore, they can hardly be used by a DM alone[9] and these DSS may basically be used only within the constant cooperation between their developers and a DM.

Despite all the stated reservations, most of these DSS have been used in practice and have provided a good basis for the design of other decision support systems.

The experience gained by these early applications has been matched with the rapid development of computer hardware and of operating systems which provide environment and tools for creating user friendly interfaces. This resulted in many DSS specialized for particular applications. However, due to the scope of this paper, only a limited number of examples can be presented.

A good example of applications in engineering design is given in [E1]. This field is very specific for designing DSS since a DM is usually a person with a good background in modelling and numerical methods. Therefore, we will not discuss this field of DSS despite the fact that it provides an excellent justification of DSS applications.

We will briefly discuss selected DSS developed by scientists from several different institutes in Poland because as the result of the scientific cooperation of the authors,

[9] Unless a DM is trained in modelling and has good experience with not-so-user-friendly computer operating systems, which is rather a rare case.

similar methodologies have been applied. In all of the presented examples the specialized software packages have been developed for personal computers compatible with the IBM PC. Although each package has been designed for a specific application, it also provides a tutorial example which illustrates both the methodology applied to the specific class of decision problems and the capabilities of the particular DSS.

The above mentioned activity on the development of a DSS for flood control is being continued (at the Institute of Automatic Control of the Warsaw University of Technology) and the current stage of research is described by Karbowski in [K3]. The system under study consists of a main river and three of its reservoirs. The DSS is aimed at assisting the system operators in deciding on the water releases from the multipurpose reservoirs during flood periods. The corresponding mathematical programming models have been developed based on both deterministic and stochastic approaches. A specialized (by adding functions that make it easier to interprate results for this particular problem) version of DIDAS-L package (cf [R3]) has been implemented. Despite the high scientific standards of the research and the lasting cooperation between the future users and the researchers, the DSS is still not considered to be robust enough for real-time applications. One of the reasons is due to legal problems which can be summarized as follows: in practice, the DM is supposed to follow specific decisions rules for each reservoir which determine the amount of water released for each state of reservoir and forecasted inflows; the operator is not responsible for any flood damages if those rules are followed, but he could be responsible for damages resulted by the application of a DSS even if a DSS would help to avoid losses in most cases. Similar legal problems occurred in the application of a DSS for Lake Como (cf [G7]). However, after several years of testing, the DSS for Lake Como was eventually approved by the authorities as the official tool for controlling the outflow from the lake. In the future, a similar legal procedure will most likely have to be passed for the Vistula river reservoirs. The same team has also developed DIDAS-N - Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models (cf [K11]). DIDAS-N is equipped with a nonlinear model generator and an editor which ease the generation, edition and symbolic differentiation of nonlinear models. One of the applications of DIDAS-N was the analysis of acid deposition in forest soil.

A set of DSS for farm management has been developed by the scientific team organized at the Poznań Technical University (Poland). The FLIP (cf [C2]) is a DSS for searching for the best structure of production activities on a typical Polish farm. The DSS is based on the Multiobjective Fuzzy Linear Programming approach and is used by consultants to farmers. The MPS (cf [S5]) is the Decision Support System for Multiobjective Project Scheduling under multiple-category resource constraints. It handles quite a general class of nonpreemptive scheduling problems with renewable, nonrenewable and doubly constrained resources, multiple performing modes of activities, precedence constraints in the form of activity network and multiple project performance criteria. The DSS is used for scheduling farm operations but it can also be used for other problems which fall within the above mentioned class. Another DSS is being developed by the same team for the Two Dimensional Cutting Problem (cf [B2]). This DSS is still under development and its current version may be used for the design of cutting rectangular material into pieces of arbitrary shapes.

Several applications have made for the problem of facility locations by the team from the Institute of Informatics of Warsaw University. These activities have resulted in Dynamic Interactive Network Analysis System DINAS (cf [O1]). This DSS is aimed at assisting in solving various multiobjective trans-shipment problems with facility locations

and has been applied to several different decision problems in Poland. One of these applications was made for supporting the decision regarding location of new hospitals in Warsaw, another application was made for solving transportation problems relating to the construction of the underground in Warsaw.

For over ten years, the team from the Academy of Mining and Metallurgy in Kraków (Poland), has been successfully implementing decision support systems for programming the development of the chemical industry and for selected management problems in this industry. This activity is documented in various publications and its overview is given in [K6]. The team (which is also composed of specialists in the chemical industry) addresses specific problems by the development of a specialized DSS, e.g. for planning the development of selected chemistry subsectors in Poland, Algeria, China, Iraq and for the SADCC[10] countries. A DSS has also been developed for management problems, e.g. the MIPS (Decision Support Systems for Multiobjective Interactive Project Scheduling - cf [R6]) or a DSS for Multiobjective Design of Multi-Purpose Batch Plant (cf [R5]). Some of these activities have been organized in cooperation with UNESCO.

Hybrid-FMS - an element of DSS for designing Flexible Manufacturing Systems (FMS) was developed within the cooperation between the SDS Program and the Computer Integrated Manufacturing Project at IIASA and the Systems Research Institute of the Polish Academy of Sciences (cf [M9]). Since the design of FMS (cf [R1]) is not only multicriteria but also a multilevel task, the Hybrid-FMS is only an element of the required DSS. Hybrid-FMS is composed of a specialized interface to the data base (which contains data on different FMS) and of a specialized editor for updating input data. Additionally, the efficient solver has been implemented for solving the resulting bilinear optimization problem.

The above discussed applications have been made for IBM compatible computers running under DOS (except for some of the applications made for the chemical industry which were made for the UNIX-based minicomputers). This implies obvious limitations on the speed of computations, size of the problem and for the quality of the user interface. There are, however, good examples of implementations for computer workstations, which provide both a higher computation speed and a better user interface. We will briefly present two of them. First, *The Shanxi Province DSS*, has been developed within the context of a regional case study on Shanxi Province, the People's Republic of China (cf [F2]). This project was carried out by IIASA's Advanced Computer Applications (ACA) group[11] which has been engaged in the development and implementation of an expert system for regional development planning. The DSS is designed as a hybrid system combining classical data processing methodology and the methods of operations research and systems analysis with concepts and techniques of artificial intelligence. Conceptually, the main functional elements of the integrated software system are:

- an *Intelligent User Interface*, which provides a largely menu-driven conversational guide to the system's usage and a number of display and report styles, including color graphics and linguistic interpretation of numerical data;
- an *Information System*, which includes the system's Knowledge and Databases, as well as the Inference and Database Management Systems;

[10]SADCC stands for Southern African Development Coordination Conference. Its nine member countries are: Angola, Botswana, Lesotho, Malawi, Mozambique, Swaziland, Tanzania, Zambia, Zimbabwe.

[11]Some elements of the theory and software developed within the cooperation of the MDA Project with the Polish Academy of Sciences have been also used in this project.

- the *Model System*, which consists of a set of models (simulation, optimization) which describe individual processes that are elements of a problem situation, perform risk and sensitivity analysis on the relationship between control and management options and criteria for evaluation, or optimize plans and policies in terms of their control variables given information about the user's goals and preferences;

- the *Decision Support System*, which assists in the interpretation and multi-objective evaluation of model results, and provides tools for the selection of optimal alternatives with interactively defined preferences and aspirations.

After the completion of the Shanxi project, its DSS, which is based mainly on the multicriteria alternative selection tool DISCRET (cf [M2, Z1]) has been used as the basis for the development of the hybrid discrete analysis system HYDAS (cf [W9]), which combines numerical, symbolic, graphical, and statistical methods to support the decision maker when exploring the solution space and enables the user to arrive at a well-informed decision. In HYDAS the user has the option of using rule-based techniques (eg., logical filters for alternative selection), traditional numerical type multicriteria optimization, or an effective combination of both. Currently ACA is engaged in the development of a DSS for air pollution control in China, which will use HYDAS as its decision support component.

The second group of the presented applications differs substantially from the early applications characterized above. The main similarities are the first two features specified for the first group, i.e. that each application is problem oriented and that for each one a corresponding mathematical programming model and a specialized software was developed. The main differences are as follows:

- Each DSS is one program which is easily used. Obviously many of the DSS are composed of mutually linked modules which use the sophisticated structure of data management but from the user point of view, a DSS is just one integrated program.

- A DSS has a user-friendly interface and is usually operated by the commonly known menu system. Most DSS are equipped with the context sensitive help, which provides the DM with information relevant to the current status of the program and helps use the program with limited knowledge of an operating system.

- The user interface eases the problem of data management (i.e. retrieval of selected data or results of a scenario analysis) and also substantially decreases the probability of making mistakes in data management (like overwriting data files). Data forms and spreadsheet are also easily used for inputing and updating data[12]. The results are presented in a form most suitable for a specific application, often in a graphical form.

- Most of the technical details are hidden from a user, who is navigated through the options in the proper sequence. The diagnostics are much less cryptic than those provided by commercial packages.

- Solvers used for optimization problems are much more robust than those that have been available for over the past ten years, but in this field a lot still remains to be done.

[12]The difference is especially clear for those who have experience with formatted Fortran input.

However, despite the remarkable improvements in the design and implementation of DSS, a lot of shortcomings still remain to be fixed. They will be discussed in the next section.

6 Designing and implementation of a DSS

The problem of DSS design and application has been well covered by many conferences (cf e.g. [G3, G4, G5, K4, K7, L2, L3, M13, S2]), in many text books (cf .e.g. [A2, D1, E1, H1, H2, L5, S2, S7, T1]) and in many articles (cf e.g. [A3, B1, B3, G1, K8, L1, N2, N3, W5, W6]). The topic itself is far too broad and complex to be fully presented and discussed within a paper. Therefore, only some elements of the design and application procedures have been selected for this presentation. Two types of those elements have been selected. First are those critical for any implementation of a DSS. Second are those which are not considered enough in practice.

Both the selection of topics and the evaluation of the DSS which are referred to are very subjective. The criticism implied by this section of the paper is definitely not aimed at the underevaluation of the applications which serve for the illustration of the considered problems. The author has enough personal experience in both programming and trials of DSS applications to appreciate the amount of work which lies behind both software development and contact with software users. Interaction with real decision makers obviously results in an increase of both the time and effort required for the development of any application.

There are many approaches for the definition of a DSS structure, (many of them can be found in the books and articles cited above) but the classical structure of a DSS proposed by Minch (cf [M12]) probably serves as the best starting point for structuring the discussion on design and implementation of a DSS. The discussion is divided into subsections dealing with problems related to:

User : The identification of a DM who really needs a DSS, of a decision problem and of the DMP environment, is probably the most critical issue for any real-life application. The related problems include cooperation with a DM during the DSS development and implementation process.

DMS - Dialog Management System : The design and implementation of a user interface is often the second most important issue. Although the rapid (in last few years) development of hardware and software tools ease the implementation of a proper interface, none of the known DSS applications provide the user with an interface which meets all justifiable[13] requirements.

PPS - Problem Processing System : This software component coordinates the cooperation of the other three software modules (i.e. DMS, MMS and DBMS). Implementation of the PPS is purely a technical problem, but still few rules related to the whole DSS should be considered.

MMS - Model Management System : This component has perhaps the best theoretical background based on modelling techniques and numerical methods. There

[13]This means providing easily-used options and/or tools to retrieve any information considered by a DM as valuable and to start any requested action (like generating a scenario, setting or changing values of decision variables, generating different reports, etc).

are, however, few issues specific for models used as a part of a DSS which should be considered. For the aspiration-led DSS the MMS also contains a solver module.

DBMS - Data Base Management System : This is possibly the easiest component of a DSS. There is a lot of experience and related software which can easily be adopted for a particular implementation.

DSS implementation : There are critical issues of a DSS design and implementation procedures, which do not belong to any of the above listed subsections but are, nevertheless, worth discussing.

6.1 User

As Andriole (cf [A2]) points out, there is no more important, yet more neglected, element of DSS design than requirements analysis. One can consider this problem as composed of the three related elements all of which are critical for the design and implementation of a DSS:

DMP : An understanding by a DSS designer of the DMP environment is usually not only the starting point for proper identification of both a DM and a decision problem which will be the topic for the DSS. The DSS must fit the DMP environment, which includes mainly procedures of making decisions and the structure of an organization.

Decision problem : This is a part of the DMP which can and should be formalized to become a kernel problem for which the DSS will be used. The key question to be answered for justification of the decision problem selection may be formulated as follows: to what extent is it possible to specify, verify and solve a mathematical programming model that is good enough to replace at least a part of the DM's experience and intuition for this part of a DMP ?

DM : Understanding of and cooperation with the future user of the DSS is obviously a critical and necessary condition for any implementation. Numerous authors (cf e.g. survey by Benbasant and Nault [B1]) agree that many users who are involved in the DSS development perceive the DSS as more worthwhile than those who simply have the system delivered to them. This is most probably because involvement of the DM in the DSS design and implementation results in:

- better understanding by the DSS designers of the DMP and the decision problem,
- the user interface of the DSS better fits the needs of the DM,
- the DM better understands the functioning and capabilities of the DSS.

The decision makers are very different persons and the problem of user profiling (cf e.g. [A2]) is also difficult. Yu (cf [Y2]) gives excellent and thorough coverage of related problems. Andriole [A2] presents a short discussion of this problem and also a checklist of questions and answers which help to profile a user.

Without the clear identification of all three components of the requirement analysis, even the best DSS could, at the most, satisfy the needs of the designers, but not the intended users and therefore will never actually be used in a real-life DMP.

Identification of the three user related components usually takes at least several months. It is therefore a part of the iterative procedure composed of design, implementation and examination of a series of DSS prototypes. Prototyping is a controversial issue because it was considered to be a result of the wrong approach to software design and implementation. However, prototyping seems to be both necessary and desirable for a DSS design and implementation (cf e.g. [G1]). Moreover, many modern software tools facilitate fast prototyping. Therefore, application prototyping has become popular over the past few years. Andriole [A2] summarizes the following advantages of prototyping: it supports modular software engineering, permits user participation in the design process, and protects project resources from tunnel programming. Most importantly, the DSS prototyping keeps the requirement analysis process alive during the DSS implementation.

6.2 DMS - Dialog Management System

The appropriate design and implementation of the user interface is probably the most difficult part of the DSS design. This question is directly linked to the requirement analysis issues, especially with the problem of meeting the expectations and needs of the DM. It obviously supports arguments for prototyping. None of the above mentioned DSS (with the exception of the two made by ACA for the UNIX based workstations) have an interface that can be evaluated as at least good.

There are two extreme approaches to the DMS. First is to apply a programming language like interface that allows for all possible tasks (from coding data base and model details through selection and execution of various solvers to sophisticated formatting of reports writers). This approach results in a very flexible software system, which can, however, hardly be used without digging into hundreds of pages full of computer jargon. Therefore it would never be used by any real DM. Second is a *push button*-like system. Such a system is very easy to be used, but is extremely inflexible, therefore it can be useful only in very specific situations. Obviously both extremes should be avoided. The DMS design should conform to good software engineering rules and to requirements specific for a DSS. These requirements also imply that the DMS has to be both problem and user specific.

The complexity involved in the design and implementation of a DSS is often overwhelming for a typical user who is not trained in modelling and computer techniques. This complexity should be hidden from the user who should understand only those details of the DSS operation which are necessary to: first, identify which part of the DMP is covered and supported by the DSS and second, the capabilities and limitations of the DSS. Both elements are critical to give the DM confidence in results supplied by the DSS, and to make him comfortable with how well the DSS reflects reality and provides a useful analysis.

A good justification for the above statement on complexity follows from an analysis of the usage of spreadsheet packages, which are one of the most popular software used on personal computers. Spreadsheet packages are commonly used for modelling accounting and financial problems and are considered to be software which is reliable and easy to learn and use. However, Brown and Gould [B3] have discovered that in forty-four percent of the cases, even experienced users of spreadsheet packages make programming errors. There is no doubt that a proper update and analysis of optimization results[14] of an average LP model requires more modelling and computer experience than formulation and solution of an accounting problem with a spreadsheet package. While complexity should and can

[14]We assume that formulation and generation of the model is done by a system analyst.

be hidden, the DMS should be designed in such a way that a DSS would not provide misleading information.

Andriole [A2] gives a good summary of the guideliness for design requirements of a DMP. This summary results from a critical compilation of a number of publications. The following list does not contain most of the guidelines that can be found in [A2]. Instead, some additional suggestions which result from analysis of the DSS presented in the previous sections are given. These requirements and features for the DMS can be briefly summarized as follows:

1. Give the user full control over the package while retaining robustness. The DMS should provide a DM with easy access to and usage of, all capabilities of the DSS while conforming to all principles which lie behind the DSS design. It should be as easy as possible to use. The event driven type of interface should be implemented whenever possible. The user should be clearly informed about which actions are being performed, what his choices are at each moment (this includes also informative *please wait* type messages), what information is available and what action is necessary for getting specific information.
2. The DMS should accept possibly any action of a DM (e.g. no assumptions should be made that a user enters only valid input; all input should be verified as much as possible and a relevant diagnostics for possible errors should be provided). It should be flexible enough to allow for the identification of the state of the problem, to input questions and set tasks (like a generation of scenarios), examine results without regard for the problem size and complexity, comparing alternatives with respect to criteria and so on. In another words, it should provide all options that were identified as being useful during the iterative design process.
3. A DMS should not restrict the DM, who usually changes his mind and evaluations (e.g. he may temporary disregard some criteria or may want to convert a constraint into a criterion or examine values of selected[15] variables and/or constraints). A DMS should not allow for any implied changes in the problem formulation (e.g. restriction of alternative's set) without being explicitly instructed to do so.
4. One of the main functions of a DMS (of the model based, aspiration-led DSS) is to facilitate the examination of various Pareto optimal solutions. Therefore, a DMS should allow for not only easy specification of aspiration levels for objectives, but also at the same time, to provide all information which the DM considers as useful at this point. In addition to this, it should be possible and easy to generate a scenario with values of variables (all or selected) being set by the DM. The DM may want to examine the feasibility of certain scenarios and also to evaluate them using some additional indices. Therefore an option of interactive definition and computation of indices (using any model's variables and/or constraints) should be available.
5. The DMS should provide structuring and aggregation of data and results. For example, allowing a problem to be handled as a dynamic one which results in an easy way of formulating criteria and interpreting results, since one may refer to one variable trajectory contrary to a "static" formulation of dynamic problems which involves separate variables for each time period.
6. The DM should never feel inundated with information, but all needed information should be easily available. The context sensitive and instructive help is a must. The hypertext technique is a very good tool for implementing the help system.
7. The DMS should be reliable, permissive, helpful and friendly. Although there is

[15]This implies that an easy-to-use interactive selection should be available.

probably no bug-free software, the DMS should be tested well enough to keep the probability of *crash type* behaviour close to zero. Errors should never punish the user (e.g. it should be very difficult to destroy files by mistake without having a back-up). A good DMS gives the user a chance to undo his mistake (i.e. to return to the situation before the mistake occurred).

8. Messages should be clear, concise and courteous. They should also be directly and immediately useful (e.g. error messages can provide a selection of the most probable correction options, one of which may be an *undo* option) and also instructive.
9. A DMS should possess a variable flexibility (be adaptive for users with different levels of experience). At the beginning, a DM might find flexibility burdensome, therefore a DMS should provide a mode of operation which gives more guidance and fewer choices (at least available at the first choice level). Typically, after gaining more experience, a DM both prefers and benefits from more flexibility of the system. Therefore a DM would probably like to gradually use more options and features of the DSS along with getting more experience.
10. Potential benefits from graphics in DMS are usually underevaluated. The old saying *"a picture is worth a thousand words"* summarizes very well the possible profits. However, inappropriate usage of graphics may also result in providing misleading information. A good overview of related problems and a description of BIPLOT (the graphical interface for the selection of aspiration level) is given by Lewandowski in [L1]. Another idea of an interactive graphical approach in proposed by Ng [N4].

Both the above list of requirements and the guidelines given in many textbooks, (cf e.g [A2, D1, H2, T1]) sound simple enough to be fulfilled. However, there is probably not one software product whatsoever (which for many types of applications should be easier to implement than a DSS) that conforms to all of these requirements. The reason for this is not the ignorance of the software developers, but is due to the fact that the design of such a user interface is a really difficult problem.

6.3 PPS - Problem Processing System

A PPS is the software module that is responsible for the proper coordination of cooperation of all the other software components. It is traditionally called a driver.

The driver performs three main tasks:

- Processing requests generated by the DMS. This is self-explanatory, although any efficient coordination of cooperation between sometimes complicated software components is not easy to be designed and implemented.
- Providing status information of any activity within the DSS. This includes answering questions generated by the DMS and generating informative and error messages according to the current status of any related process. This is an easy task for a single task operating system[16], but its proper implementation is quite complicated on systems with server-client architectures.
- To work quietly and efficiently as a trouble shooter for all problems that a casual user would not like to know about (like time limits, limits in storage space, making file back-up, preparation for and performing of recovery action in the case of an unexpected shutdown of the system, etc).

[16] Although one should also take care of such problems as clean handling of user interrupts or numerical exceptions.

The design and implementation of a PPS is purely a technical problem. This does not mean that it is an easy task, but there is a broad and well-covered area of software engineering that provides enough specific knowledge for software design and implementation. Therefore we have only briefly summarized above the problems which are directly related to DSS-oriented functions of a PSS.

6.4 MMS - Model Management System

The Model Management System[17] facilitates a model management approach to the computerized support of decision making which is conceptually distinct from the more traditional data- or language-oriented perspectives of DSS. The MMS is involved in all activities associated with the use of mathematical models in DSS, including the building, implementation, testing, validation, execution, maintenance and interfacing of models. For the model based, aspiration-led DSS, models are actually algorithms for solving mathematical programming models, but for other types of DSS models may do nothing more than translate or reformat data.

Formulation of a mathematical programming model is a complex task[18] and this paper is not devoted to discussing this issue in detail. Therefore this section is aimed at providing only a short summary of a recommended approach.

The first stage consists of the model specification. First, a set of variables that sufficiently describes the problem – for the sake of the desired analysis – should be selected. It is desired – however, not necessary – to define the model in such a way as to possibly exploit the problem structure which will ease the implementation of the DMS and of the solver[19]. Secondly, a set of constraints which define a set of admissible (i.e. acceptable or recognized as feasible by a decision maker) solutions should be defined. There are basically two classes of constraints. First are the so-called "hard" constraints, i.e. the constraints that must hold at any price. The second class of constraints are the so-called "soft" constraints which values are very often in practice decision variables. Therefore one should consider converting such constraints into criteria. Finally, a set of criteria which could serve for the selection of an admissible solution should be defined.

It should be stressed that the specification of a complex model usually requires the cooperation of a specialist – one who knows the nature and background of the problem to be solved – with a system analyst who can advise on a suitable way of formal definition. It should be clearly pointed out that a proper definition can substantially improve the use of any computational technique. For small problems used for the illustration of the method, it is fairly easy to define a model. However, for real life problems, this stage requires close cooperation between the DM and a team of analysts, as well as a substantial amount of time and resources.

An optimal (meaning the best for the DM) solution is the one that maximizes the DM's utility or satisfaction. These cannot be adequately well-formalized therefore they are represented by a set of criteria. A definition of the set of criteria that could adequately reflect the DM's preferences may be in some situations quite a simple task. In more complex cases it is recommended to follow the guidelines of the classical textbook by Keeney and Raiffa [K5], who specify five desirable properties of the criteria set:

- completeness (should cover all aspects of a decision problem, thus indicating the degree to which the overall objective is met),

[17] We follow the definition of MMS given by Minch [M12].

[18] Examples that illustrate the complexity of this problem can be found e.g. in [H3] and in [W7].

[19] Cf e.g. the discussion of dynamic LP models in [M8].

- operational (should help to understand the implications of the alternatives),
- decomposable (to allow for decomposition into parts of small dimensionality in the criteria space),
- nonredundant (to avoid the problem of double counting of consequences),
- minimum size (according to the classical recommendation the number of objectives should not exceed seven).

Keeney and Raiffa also give guidelines and examples for the analysis of conflicting objectives and structuring of objectives. Examples of different types of criteria are given in Section 4.

There are many specialized tools called model generators which can facilitate the implementation of a mathematical programming model (cf e.g. LPL Modelling Language [H4]). However, in many situations it is more reasonable[20] to implement a specialized model generator, which can easily be written by a system analyst.

The next stage, after the model has been specified and implemented, is the model verification. This stage is crucial for the real application of any mathematical programming model. Model verification usually results in the necessity of corrections of the model definition and/or implementation. Most DSS allow for modification of the problem formulation at any stage of analysis. This is a useful option, but it should be used with care. The DM should be informed that any modification that results in a change in the set of admissible solutions may result in a change of the set of Pareto-optimal solutions. Therefore comparisons of the Pareto solutions computed for different sets of admissible solutions may provide misleading results. Thus it is recommended that both utopia and nadir points should be computed again for any change in the model formulation.

After the model is well-defined and verified and no longer requires changes in parameters that define the set of admissible (acceptable) solutions, the main activity (outlined in Section 4) may be started.

The separate part of MMS (called a solver) is dedicated to solving the related numerical problems. The solver is problem-specific and its implementation requires the involvement of good specialists in numerical methods since the solver has to be both robust and efficient. For the aspiration-led DSS the conversion of the multiple criteria optimization problem to an equivalent single objective problem is made by a specialized preprocessor. The resulting single objective problem can solved using one of the standard or a specialized mathematical programming techniques. However, it should be stressed that the solver must be robust and accept input as well as provide information in a way acceptable for the particular implementation of the DSS. Another problem is related to non-uniqueness of an optimal solution. Although this is theoretically rare, in practice many problems (especially of LP type) have actually a large set of widely varying solutions for which the objective values differ very little. A more detailed discussion of this problem and the possible ways to deal with it is given by Sosnowski e.g. in [S6]. These specific requirements for solvers applied in the DSS software very often excludes the usage of stand-alone commercial packages.

The solution obtained is converted by a specialized postprocessor to its original, multiple objective formulation, providing the DMS with information about objectives, at-

[20]Taking into account the time needed for mastering a general purpose model generator to the extent needed for more complex models and problems related to the model verification and update within the integrated environment of the DSS.

tainability of aspirations, feasibility, etc. An example of such a solver integrated with the driver and two specialized preprocessors, is the Hybrid package (cf [M8, M9]).

It is generally agreed upon that the choice of an appropriate scaling of a problem being solved is a critical issue for numerical stability. There are obviously two approaches to deal with this problem. First, suggested by Tomlin [T2], assume that an experienced model builder, who uses sensible units, may avoid unnecessarily large or small matrix elements. This is true, but requires a lot of time-consuming preparations, which are a reliable source of frustrating bugs. This implies that "manual" scaling should be avoided. Therefore, the second approach suggested by Curtis and Reid [C1] for solving the scaling problem is recommended. One of its possible implementations (for the LP type of a model) is discussed in detail in [M5]. For this formulation of the scaling problem, it was possible to design a specialized algorithm based on the conjugate gradient method. Since excessive accuracy of minimization is not required (because for the numerical reasons the scaling coefficients should be anyway rounded), the scaling algorithm is very efficient (usually it takes less then ten iterations regardless of dimension of a problem). Therefore, the scaling option should not be suppressed except if special requirements apply.

6.5 DBMS - Data Base Management System

The design and implementation of data bases is one of the best-developed areas of software enginering. There is a large collection of commercial implementations of stand-alone data base systems that provide interfaces to other applications. There are also many software tools that make it easy for the implementation of a customized data base. Therefore only two comments are given in this subsection.

First, one should make sure that only original data is stored in data base and that the appropriate procedures are implemented for protection and verification of the contents of the data base. The famous saying *"garbage in garbage out"* implies that the problem of data should not be underevaluated. A user need not worry about the possible range of quantities (which usually has an impact on computational problems) because this should be accounted for by the MMS. Nowadays, proper data handling and protection is routine on any installation that is used for data collection, therefore no further comment is necessary. An additional element of a data base should be the solution data base. The usual utilization of DSS results in generating a vast amount of solutions. Therefore the proper organization of the solution data base is not a trivial task.

A second comment is related to usage of different spreadsheets as interfaces for data input and storage. In most cases, this seems to be the wrong practice not only because of the surprisingly high probability of making errors while using spreadsheets (cf [B3]), but also because the spreadsheet approach is practically limited to problems of very small dimensions and can hardly support the proper user interface (i.e. consistent with the rest of DSS and providing with all support by DMS). This limitation is now also recognized by DSS authors who implemented the spreadsheet as the only way of inputing data (cf [K3]). The data storage capabilities offered by spreadsheet packages obviously do not conform to the requirements of DSS. Therefore the programming of data input using the same tools which are used for user interface in the other components of a DSS is the only reasonable solution.

6.6 DSS implementation

The process of DSS implementation is specific to each particular application. But nevertheless, there are a few issues that seem to be common for most applications and therefore are worth considering.

In very few and rare situations will there exist a *ready from the shelf* DSS that can be easily adopted. In most cases such a DSS is not available. A specific DSS may be designed and implemented by a highly skilled team which is capable working efficiently, fulfilling the following requirements which are critical for any successful application of a DSS:

- Recognition of this part of a DMP (which includes data and procedures) for which the DSS is desired.
- Close cooperation with the future user during the design and implementation of a DSS.
- Design of a DSS which meets the requirements of a good DSS. The critical part is an the appropriate user interface.
- Building and verification of a mathematical programming model which reflects the chosen part of a DMP.
- Development of the appropriate software (this may partly include adaptation of some existing software). The crucial point is the robustness of software (it is assumed that a user may not have any knowledge about an operating system, numerical methods, etc). Based on the analysis of many software packages developed for various applications, the Guidelines for software development (cf [M10]) have been proposed.

As already argued, the involvement of the DM in the design, prototyping and implementation of the DSS (cf [B1]) is essential for any successful implementation. One should make sure that the DM understands the function of the DSS and accepts its limitations. Such cooperation with the DM also results in the possibly of the best design and implementation of the user interface. It also provides the team of software developers with a better understanding of the DMP and the role of the DSS in this environment.

One of the most important problems is the appropriate selection of the computer hardware. This selection should be made by taking into account both the computational requirements[21] and the software tools provided by specific computer hardware. So far, most applications have been made for personal computers running single-user single-task operation systems. Since prices for computer hardware are rapidly decreasing, and computer hardware capabilities are rapidly increasing, it is likely that most of the new applications will be implemented on UNIX-based installations running in server-client architecture. Additionally, the use of distributing computation and parallel processing is recommended whenever the resulting numerical problem is complicated enough and the relevant hardware possibilities are provided. This decision is crucial, but it is easily reached by an experienced software developer.

All DSS software components should have a modular structure and should be developed preferably by using object-oriented programming languages. The advantages of

[21] This should definitely include inevitable extensions of requirements for DSS functions; therefore, one should make sure that the requirements are not underestimated.

the application of object-oriented languages are hardly questioned, and therefore there is no need to give any arguments here (some basic arguments are however summarized in [M10]). The consequences of selecting a programming language usually lasts for years. However, it should be pointed out that software and data structures which are designed properly make it possible to use different programming languages even in one program. A more detailed discussion of topics related to the development of software for DSS is far beyond the scope of this paper. Therefore Guidelines for development of such software are proposed in the separate paper (cf [M10]).

There is probably not one bug-free software package, therefore it is important to use software engineering techniques which ease the software development and result in better end products like the modular structure of each DSS component and the application of object oriented programming languages, as well as utilization of well tested software tools. Very often the testing phase of software development is underevaluated despite the commonly known fact that there is probably never enough software testing. One should also remember the DSS project scheduling and the related *5% syndrome*[22], which implies that either the software is not ready on time or that an inadequately tested version of the software is released.

There are two key principles that allow for the re-use of software. First, the software should have a modular structure and should be well-tested. This implies that the popular *"quick and dirty"* programming technique should never be used in DSS development (no matter how far behind schedule the project is). Second, the software should be developed in such a way that it is portable between many possible architectures. This implies that one should prefer both the programming languages and tools that results in a portable software.

7 DSS software available from the MDA Project

The following software packages have been developed for IBM compatible personal computers and are available from the MDA Project:

DIDAS-L – is an implementation of the aspiration-led methodology for LP problems. Special attention has been paid to the simplicity of the user interface, tools for problem definition and analysis of results.

DIDAS-N – is an implementation of the aspiration-led methodology for nonlinear problems. Highly efficient and robust nonlinear solver algebraic manipulation tools which allow model analysis and differentiation. The model can be formulated in terms of algebraic equations;

DINAS – is an implementation of the aspiration-led methodology for transportation and location problems. Highly efficient solver. Problems can be defined in terms of a graph and the specialized network editor can be used for this purpose.

DISCRET – supports analysis of finite (but large) set of alternatives: supports a selection of nondominated alternatives according to the additional constraints given, computation of approximation of Pareto set and aspiration-based analysis of alternatives.

[22]Which says that the last 5% of the scheduled development time often takes about 50% of the real development time.

FLIP – is an implementation of the aspiration-led methodology for LP problems with imprecise coefficients. The user friendly interface also provides information on the implemented methodology.

HYBRID – is a DSS framework with non-simplex algorithm for multicriteria analysis of LP systems based on the aspiration-led methodology. Due to the solution technique Hybrid is capable of solving relatively large problems.

HYBRID-FMS – is a specialized version of Hybrid for the FMS model. The user interface allows for the modification of data that defines the model. The package can generate and solve two versions of the corresponding model: the linearized FMS model and the bilinear model.

MCBARG – is the experimental implementation of aspiration-led methodology for analysis of a decision situation and for mediation in multicriteria bargaining processes.

MIDA – is a demonstration package for supporting spatial allocation of resources in the programming of development of the chemical industry.

MIPS – is a DSS for multicriteria project scheduling processes in real-life applications in the planning of the development of the chemical industry.

MPS – is a DSS for solving a general class of precedence and resources constrained project scheduling problems. The DSS either computes the exact solution (if sufficient time is available) or use one of the three built-in heuristics to compute a suboptimal solution.

ROZKROJ – is a DSS for solving a two-dimensional irregular cutting problem (elements can be of arbitrary shapes and a sheet of rectangular material is assumed to have a constant width).

The software listed above has been developed by the group of researches in Poland within the cooperation between the System and Decision Sciences Program and the Polish Academy of Sciences. Within the next few weeks several other packages currently being developed by the group of scientists organized by the Bulgarian Academy of Sciences, will be available.

The distributable versions of the software are usually tailored for the illustration of methodology and of possible applications. However, for most of these software packages there exists a version made for a specific application and it is possible to modify each software package for a specific real-life application (if the corresponding mathematical programming model is of the type for which a particular package has been designed).

All software developed within the scientific cooperation mentioned above is available either at distribution cost or free of charge for scientific non-commercial usage by institutions and individuals from the countries which are Members of IIASA. Inquiries about more detailed information and requests for the software should be addressed to the Leader of the MDA Project.

8 Conclusion

In very few situations may a DSS be reduced to a simple tool (like data base, spreadsheet, classical optimization software or even a calculator). But in more typical situations, a

DMP includes an examination of the many alternatives and the DMP is a complex task for two reasons. First, evaluation of a decision alternative requires processing data and logical relations which (due to the nature of the problem) cannot be replaced by intuition. Second, it is often desired to have a supporting tool which not only answers *what - if...* type questions but also *what to do - for achievement of...* questions. In many practical situations a DM has to consider more than one criterion. At the same time, it is not trivial to examine the possible range of feasible alternatives. Additionally, a DM often changes his preferences and assumptions during a DMP. In such situations a DSS could be a helpful supportive tool provided that it fits to the context of the DMP.

The arguments summarized above show that there is no way to provide a *ready from the shelf* DSS which could be applied to any real life problem. In particular, a DSS should not be just a collection of available software.

The discussion presented in this paper should not imply that the theoretical and methodological research related to DSS development is an already closed area and it remains only to develop good DSS software – even if modern software development is in itself a challenging research task.

The development of DSS also creates new challenges in many fields - mathematical programming, game theory, decision theory, psychology, computer science, etc. There are many theoretical and methodological problems in those scientific fields that are stimulated by the development of DSS for real-life applications. Many of these problems still remain to be solved, by scientists collaborating in these fields.

The development of computer hardware has been very rapid during the last ten years and this trend will most probably continue. For example, the SUN Microsystem Company claims that they will double the capabilities of their computers every 12-18 months (cf [V1]). In addition, the rapid development of software tools provides technical support for the fast and good design of decision support systems. Hence, the methodology of DSS design and implementation seems to be the main bottleneck in the broader applications of DSS. Therefore the cooperation between different scientific communities working in this field may contribute remarkably to successful future applications of DSS to real-life problems.

Acknowledgement

A large part of the paper deals with the results of many different scientific activities which have been organized by the System and Decision Sciences Program of IIASA both at IIASA and in cooperation with many scientific communities in different countries. References are given whenever practical, but the author would like to make it clear that most of the results presented in this Working Paper have been obtained due to the work of many outstanding scientists.

The first version of this paper was originally prepared for the Symposium on *Decision Support Systems* organized by the Japan Institute of Systems Research in Tokyo on March 12–13, 1991. Discussions with Japanese scientists during the series of lectures organized by the Japan Institute of Systems Research in Tokyo, Kyoto and Kobe resulted in modifications of this first draft.

The author would like also to acknowledge the detailed comments on the draft of this paper given by Andrzej P. Wierzbicki.

References

[A1] Alcamo J. and R. Shaw and L. Hordijk, The rains model of acidification: Science and strategies in Europe, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1990.

[A2] Andriole, S.J., Handbook of Decision Support Systems, TAB Professional and Reference Books, Blue Ridge Summit, PA, 1989.

[A3] Ariav G. and M.J. Ginzberg, DSS Design: A systematic view of decision support, Communications of the ACM, Vol. 28, pp. 1045-1052, October 1985.

[B1] Benbasat I. and B.R. Nault, An Evaluation of Empirical Research in Managerial Support Systems, Decision Support Systems Journal, 6 (1990), pp. 203–226.

[B2] Błażewicz J. and M. Drozdowski and B. Soniewicki and R. Walkowiak, Two-Dimensional Cutting Problem, In [R4].

[B3] Brown, P.S. and J.D. Gould, An Experimental Study of People Creating Spreadsheets, ACM Trans. Office Information Systems 5 (1987), pp. 258–272.

[C1] Curtis, A.R. and J.K. Reid, On the automatic scaling of matrices for Gaussian elimination. *Journal of Mathematics and its applications*, No. 10, pp. 118–124, 1972.

[C2] Czyżak P. and R. Słowiński, FLIP - Multiobjective Fuzzy Linear Programming Package, In [R4].

[D1] Davis, M., Applied Decision Support, Prentice Hall, 1988.

[D2] Dreyfus H.L. and S.E. Dreyfus, Mind over Machine. The Power of Human Intuition and Expertise in the Era of the Computer, The Free Press, Macmillan. New York, 1986.

[E1] Eschenauer, H. and J. Koski and A. Osyczka Eds., Multicriteria Design Optimization, Procedures and Applications, Springer Verlag, 1990.

[F1] Fandel, G. and M. Grauer and A.B. Kurzhanski and A.P. Wierzbicki Eds, Large-Scale Modelling and Interactive Decision Analysis, Lecture Notes in Economics and Mathematical Systems, vol. 273, Springer Verlag, 1986.

[F2] Fedra, K., and Z. Li and Z. Wang and C. Zhao, C., Expert Systems for Integrated Development: A Case Study of Shanxi Province, The People's Republic of China. SR-87-1, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1987.

[G1] Gerlach, J. and F.Y. Kuo, An Approach to Dialog Management for Presentation and Manipulation of Composite Models in Decision Support Systems, Decision Support Systems Journal, 6 (1990), pp. 227–242.

[G2] Górecki H. and J. Kopytowski and T. Ryś and M. Żebrowski, A Multiobjective procedure for Project Formulation-Design of a Chemical Installation, In [G6].

[G3] Grauer, M. and A. Lewandowski and L. Schrattenholzer, Use of the reference level approach for the generation of efficient energy supply strategies, WP-82-19, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[G4] Grauer, M. and E. Zalai, A Reference Point Approach to Nonlinear Macroeconomic Planning, WP-82-134, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[G5] Grauer, M. and M. Thompson and A.P. Wierzbicki, Eds., Plural Rationality and Interactive Decision Processes, Lecture Notes in Economics and Mathematical Systems, vol. 248, Springer Verlag, 1984.

[G6] Grauer, M. and A.P. Wierzbicki Eds., Interactive Decision Analysis, Lecture Notes in Economics and Mathematical Systems, vol. 229, Springer Verlag, 1984.

[G7] Guariso, G. and S. Rinaldi and R. Soncini-Sessa, A Decision Support System for the Management of Como Lake, Paper presented at the CISM-IIASA Summer School on Decision Support Systems, CISM, Udine, Italy, 1990 (to be published by Springer).

[H1] Haimes, Y.Y. and V. Chankong, Eds., Decision Making with Multiple Objectives, Methodology and Software for Interactive Decision Support, Lecture Notes in Economics and Mathematical Systems, vol. 242, Springer Verlag, 1985.

[H2] Hopple, G.W., The State of the Art in Decision Support Systems, QED Information Sciences, Inc., Wellesley, Massachusetts, 1988

[H3] Huntley, I.D. and D.J.G. James, Mathematical Modelling, A Source Book of Case Studies, Oxford University Press, 1990.

[H4] Hürlimann, T., Reference Manual for the LPL Modelling Language (version 3.1), Institute for Automation and Operations Research, University of Fribourg, Switzerland, October, 1989

[I1] Insua, D.R., Sensitivity Analysis in Multiobjective Decision Making, Lecture Notes in Economics and Mathematical Systems, vol. 347, Springer Verlag, 1990.

[K1] Kaden S. and T. Kreglewski, Decision Support System MINE – Problem solver for Nonlinear Multi-criteria Analysis, CP-86-1986, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1986.

[K2] Kallio, M. and A. Lewandowski and W. Orchard-Hays, An implementation of the reference point approach for multiobjective optimization. WP-80-35, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.

[K3] Karbowski A., Application of IAC-DIDAS-L2 to Optimization of Water Releases from a Retention Reservior during a Flood, In [R4].

[K4] Karpak, B. and S. Zionts Eds, Multiple Criteria Decision Making and Risk Analysis Using Microcompouters, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 56, Springer Verlag, 1989

[K5] Keeney, R.L. and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, Willey, 1976.

[K6] Kopytowski, J. and M. Zebrowski, MIDA – Experience in Theory, Software and Application of DSS in Chemical Industry. In [L5].

[K7] Korhonen, P. and A. Lewandowski and J. Wallenius, Proc. of the MCDM Conference in Helsinki, 1989 (to be published by Springer in 1991).

[K8] Korhonen, P. and J. Wallenius, A Multiple Objective Linear Programming Decision Support System, Decision Support Systems Journal, 6 (1990), pp. 243–251.

[K9] Korhonen, P. and J. Wallenius, Multiple Criteria Decision Support, Theory, Applications and Computer Implementation, A Project Report, Helsinki School of Economics, Helsinki, 1991

[K10] Kreglewski, T. and A. Lewandowski and T. Rogowski Dynamic Extension of the DIDAS system and its Application in Flood Control, In [G5].

[K11] Kreglewski, T. and J. Granat and A.P. Wierzbicki, IAC-DIDAS-N - Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models, Version 4.0, In [R4].

[L1] Lewandowski, A., Decision Support Systems and Multiple-Criteria Optimization, Paper presented at the CISM-IIASA Summer School on Decision Support Systems, CISM, Udine, Italy, 1990 (to be published by Springer).

[L2] Lewandowski, A. and I. Stanchev Eds., Methodology and Software for Interactive Decision Support, Lecture Notes in Economics and Mathematical Systems, vol. 337, Springer Verlag, 1989.

[L3] Lewandowski, A. and V. Volkovitch, Proc. of the MCDM Conference in Yalta, 1988 (to be published by Springer in 1991).

[L4] Lewandowski, A. and A.P. Wierzbicki, Decision Support Systems Using Reference Point Optimization. In [L5].

[L5] Lewandowski, A. and A.P. Wierzbicki Eds., Aspiration Based Decision Support Systems. Theory, Software and Applications. Lecture Notes in Economics and Mathematical Systems, vol. 331, Springer Verlag, 1989.

[M1] Majchrzak J., The implementation of the multicriteria reference point optimization approach to the Hungarian Regional Investment Allocation Model, WP-81-154, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.

[M2] Majchrzak J., DISCRET–A Package for Multicriteria Optimization and Decision Problems with Discrete Alternatives. In [G5].

[M3] Makowski, M., Modeling the Expansion of the Water System in the Upper Notec Region, CP-80-09, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.

[M4] Makowski, M. and J.S. Sosnowski, Coordination of Sectoral Production Planning Using Prices and Quotas: a Case Study for the Polish Agricultural Model. CP-81-38, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.

[M5] Makowski, M. and J.S. Sosnowski, Implementation of an algorithm for scaling matrices and other programs useful in linear programming, CP-81-37, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.

[M6] Makowski, M. and J.S. Sosnowski, A decision support system for planning and controlling agricultural production with a decentralized management structure. In [G5].

[M7] Makowski, M. and J.S. Sosnowski, User Guide to a Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID Version 3.1, WP-88-111, International Institute for Applied Systems Analysis, Laxenburg, Austria, December 1988.

[M8] Makowski, M. and J.S. Sosnowski Mathematical Programming Package Hybrid. In [L5].

[M9] Makowski, M. and J.S. Sosnowski, Hybrid-FMS: an element of DSS for designing Flexible Manufacturing Systems, In [K7].

[M10] Makowski, M. Guidelines for Software Development for Decision Support Systems, WP-91-15, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1991.

[M11] Messner, S., Natural gas trade in Europe and interactive decision analysis, In [F1].

[M12] Minch R. and J.R. Burn, Conceptual Design of Decision Support Systems Utilizing Management Science Models, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, no. 4, pp. 549–557, 1983.

[M13] Mitra, G., Ed., Mathematical Models for Decision Support, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 48, Springer Verlag, 1988.

[N1] Nakayama, H., Satisficing Trade-off Method for Interactive Multiobjective Programming Methods, In [G6].

[N2] Nakayama, H. and J. Nomura and K. Sawada and R. Nakajima, An application of Satisficing Trade-off Method to a Blending Probelm of Industrial Materials, In [F1].

[N3] Nakayama, H., Interactive Multi-objective Programming and its Applications, Paper presented at the CISM-IIASA Summer School on Decision Support Systems, CISM, Udine, Italy, 1990 (to be published by Springer).

[N4] Ng W.Y., An interactive descriptive graphical approach to data analysis for trade-off decisions in multi-objective programming, Information and Decision Technologies, 17 (1991) 133-149.

[O1] Ogryczak W. and K. Studziński and K. Zorychta, Dynamic Interactive Network Analysis System - DINAS, version 3.0 (1990), Users's Manual, In [R4].

[O2] Otsuka K. and Y. Matoba, Y. Kajiwara, M. Kojima, M. Yoshida, A Hybrid Expert System Combined with a Mathematical Model for Blast Furnace Operation, ISIJ International, Vol. 30 (1990), No. 2, pp. 118–127

[P1] Papert S., Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, Inc., 1980.

[R1] Ranta J. and A. Alabian, Interactive Analysis of FMS Productivity and Flexibility, WP-88-098, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1988.

[R2] Rapoport A., Decision Theory and Decision Behaviour, Normative and Descriptive Approaches, Theory and Decision Library, Kluwer, 1989

[R3] Rogowski T. and J. Sobczyk and A.P. Wierzbicki, IAC-DIDAS-L Dynamic Interactive Decision Analysis and Support System Linear Version, In [L2].

[R4] Ruszczyński, A. and T. Rogowski and A.P. Wierzbicki, Eds., Contributions to Methodology and Techniques of Decision Analysis (First Stage), CP-90-008, International Institute for Applied Systems Analysis, Laxenburg, Austria, November 1990.

[R5] Ryś T., Multiobjective Design of Multi-Purpose Batch Plant, In [R4].

[R6] Ryś T. and R. Stanek and W. Ziembla, MIPS: a DSS for Multiobjective Interactive Project Scheduling, In [R4].

[S1] Sawaragi, Y. and H. Nakayama and T. Tanino, Theory of Multiobjective Optimization, Mathematics in Science and Engineering, Vol. 176, Academic Press, 1985

[S2] Sawaragi, Y. and K. Inoue and H. Nakayama Eds., Toward Interactive and Intelligent Decision Support Systems, Lecture Notes in Economics and Mathematical Systems, vol. 286, Springer Verlag, 1987.

[S3] Simon, H.A., Models of Man, Willey, New York, 1957.

[S4] Simon, H.A., Administrative Behavior, Macmillan, New York, 1958.

[S5] Słowiński R. and B. Soniewicki and J. Weglarz, MPS - Decision Support System for Multiobjective Project Scheduling, In [R4].

[S6] Sosnowski, J.S., Linear Programming via augmented Lagrangian and conjugate gradient methods. In: Methods of Mathematical Programming, Proceedings of 1977 Conference in Zakopane, S. Walukiewicz and A.P. Wierzbicki (Eds.), Polish Scientific Publishers, Warsaw, 1981.

[S7] Steuer, R.E., Multiple Criteria Optimization: Theory, Computation, and Application, John Wiley & Sons, 1986.

[S8] Steuer, R.E., Concepts of the Reference Point Class of Methods of Interactive Multiple Objective Programming, Paper presented at the CISM-IIASA Summer School on Decision Support Systems, CISM, Udine, Italy, 1990 (to be published by Springer).

[S9] Steuer, R.E., ADBASE Operating Manual (version 9/90), Department of Management Science and Information Technology, Univeristy of Georgia, Athens, Georgia, 1990.

[S10] Strubegger M., An Approach for Integrated Energy-Economy Decision Analysis: the case of Austria, In [F1].

[T1] Thierauf, R.J., User–Oriented Decision Support Systems: Accent On Problem Solving. Prentice Hall, 1988.

[T2] Tomlin, J.A., On scaling linear programming problems. *Mathematical Programming Study 4*, North Holland Publishing Company, Amsterdam, 1972.

[V1] Vollmuth J., Linenzen sichern Vorsprung, CHIP, Februar 1991, pp. 28–29.

[W1] Wierzbicki, A.P., A methodological guide to multiobjective decision making, WP-79-122, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1979.

[W2] Wierzbicki, A., A mathematical basis for satisficing decision making. WP-80-90, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.

[W3] Wierzbicki, A.P., The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, eds., Multiple Criteria Decision Making, Theory and Applications, Springer Verlag, Heidelberg 1980.

[W4] Wierzbicki, A.P., A mathematical basis for satisficing decision making. Mathematical Modelling, 3, pp. 391–405, 1982.

[W5] Wierzbicki, A.P., Critical essay on the methodology of multiobjective analysis, Regional Science and Urban Economics, Vol. 13, pp. 5–29, 1983.

[W6] Wierzbicki, A.P., Interactive decision analysis and interpretative computer intelligence, In [G6]

[W7] Wierzbicki, A.P., Models and Sensitivity of Control Systems, Elsevier/WNT, 1984.

[W8] Wierzbicki, A.P., On the completeness and constructiveness of parametric characterizations to vector optimization problems, OR-Spectrum, Vol 8, pp. 73–87, 1986.

[W9] Winkelbauer, L. and S. Markstrom, Integration of AI with Quantitative Methods for Decision Support, Expert systems with Applications: AI International Journal, Vol. 1, No. 4, 1990.

[Y1] Yu, P.L., Multiple-criteria Decision Making, Concepts, Techniques, and Extensions, Plenum Press, 1985.

[Y2] Yu, P.L., Forming Winning Strategies, An integrated Theory of Habitual Domains, Springer Verlag, 1990.

[Z1] Zhao, C., and L. Winkelbauer and K. Fedra, Advanced Decision-oriented Software for the Management of Hazardous Substances, Part VI: The Interactive Decision-Support Module, CP-85-50, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1985.