Schmidt, Reinhart; Janowski, Wolfgang

Plasma II: An interactive modeling system for mathematical programming

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 73

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

NR. 73

PLASMA II
AN INTERACTIVE MODELING SYSTEM FOR MATHEMATICAL PROGRAMMING

REINHART SCHMIDT
AND
WOLFGANG JANOWSKI

PAPER PRESENTED AT NOAK '79 - THE 8TH NORDIC CONGRESS
ON OPERATIONS RESEARCH IN OSLO/NORWAY
ON SEPTEMBER 23-25, 1979

CONTENTS

1. Introduction

The main efficiency factors of a computerized planning system are:

- the possibility to solve different planning problems
- the easiness of system usage
- the speed of problem solution
- the planning costs.

Advances in planning theory, OR-methods, and EDP-software have promoted the implementation of such planning systems. But it seems that the benefits of these systems must be increased with respect to the flexibility of models, methods, and data. Therein we define (see [15]):

- models as portraitures of systems where the system structure can be manipulated
- methods as instructions of how to transform an input into an output under a given objective
- data as numerical or nonnumerical specifications of variables.

The demand for more flexibility with respect to those elements results from observations concerning implemented planning systems or offered software packages (see [17]):

1. The flexibility of models must be increased so that changes and connections of models can be brought about easily.
2. The flexibility of methods has to be improved so that different methods can be used.
3. The flexibility of data should become better by allowing to use different data sets.

These problems have been perceived by the literature and the software producers in the last years. And various attempts have been made to increase the benefits by establishing proper system characteristics. Therefore

we at first give a short review of these attempts. After that we describe and discuss the planning system for mathematical applications on a dialogue basis (PLASMA) which has been developed by the authors at the University of Kiel since 1976.

Looking at the literature one must pay attention to the fact that in many cases the authors develop conceptions which have not been realized and that the capabilities of realized systems are very different. The lines of development can be described by the main objectives of the systems in question as follows.

A first partial objective is a support on the way from model construction to the generation of numerical values. This is done by two approaches till now:

1. The algebraic language approach in case of optimization models. The origin of this approach lies in matrix generators (e.g. MAGEN) and it is tried to abstract from the numerical model specification by allowing a more or less general problem formulation (see [2], [3], [4], [7], [11], [18], [19], [20]).
2. The graphical approach in case of simulation models. Here the problem structure is portrayed by a network (see [10] and [13]).

A second partial objective is a support of the model solution process. This mainly concerns the choice of the solution algorithm by an adequate problem-oriented language (see [8], [18] and [19]). The implementation of different methods then leads to special problems concerning the EDP-organization of methods (see [1] and [6]). Finally such support also requires efficient procedures which test inconsistencies and restructure the data (see [14]).

A third partial objective is the construction of user-oriented reports - perhaps enabling some fundamental computations, too. Most of the software packages which are offered by computer producers address this objective.

An overall objective is the realization of man-machine-interaction (see [12] (1970, published in 1979)).

While the earlier approaches only gave attention to matrix generation or report generation, recent approaches stress upon the integrated process of problem solution. Therein model-based decision-making is based on modern organization theory. The planning process starts at the problem recognition phase and terminates with the feed-back after the control of results (see [16]). PLASMA has been drawn up with respect to this conception.

2. The Structure of PLASMA II

2.1. The Conception

Several requirements led to the development of PLASMA. They are related to research, teaching, and business applications:

- The research on corporate modeling shall be made easier by an instrument which allows a quick change of model structures, applied methods, and data sets.
- The teaching of operations research and planning can be improved if students can follow repeated processes of model construction and change, and if they can formulate larger models than can be solved by pencil and paper.
- Business applications of planning models may be encouraged if the modeling process can be made more comfortable and if EDP-use is possible without the knowledge of a programming language or a command language (see [5] for this problem).
- For all three areas of application it is desirable that several users can use the system in parallel.

The consequences of these requirements have been for PLASMA:

- the creation of user-oriented model construction and change - this requires a mode of model formulation closer to natural language than to matrix notation;
- the installation of several OR-methods like linear programming, goal programming, and multiobjective linear programming;
- the creation of data and file manipulation procedures;
- a dialogue system which allows for user-directed and programm-directed dialogues;
- a job control system which can handle several users and which is done by the system itself so that the user must not start programs.

2.2. The Elements of PLASMA II

PLASMA II is an extended and corrected version of a former version [17]. The system has been programmed in FORTRAN IV (with a few routines in the PDP assembler MACRO) and it is running on a PDP 10 at the computing centre of the University of Kiel. There is no procedure which needs more than 32 K words of core. The elements of the system can be understood from Figure 1.

The main procedure is PLASMA-MENU which is called up automatically after every other procedure and which allows a branching to different procedures (not: subroutines). In PLASMA II it is distinguished between system procedures and user files.

The system procedures belong to three categories:
- information [system information (SYS), printing (DRU), sending error messages from the user to the system (ERR), dialogue protocol (DIA)]
- methods [OR-methods (RS1, RS2, RVS, GOA, MOP), interpretation and translation of models (UEB), numerical preparation of a model (AUF), report generation (REP)]
- manager [model construction (MOK), data manipulation (DAT), goal manipulation (ZDF), report defining (RDF)]

The files of the users are structured hierarchically; a user file-directory ($USER_i$ FD) contains the following master file-directories (MFD):
- a model-MFD
- a data-MFD
- a goal-MFD
- a report-MFD.

In each of the master file-directories several file directories can be registered (only the R.MFD contains the names of report files directly). The user can decide how many file directories of one type he wants.

Each file-directory contains the names of the files which have been assigned to this directory by the user.

The hierarchical organization has the advantage that a file-manipulating command cannot only touch a single file, but also a lot of files.

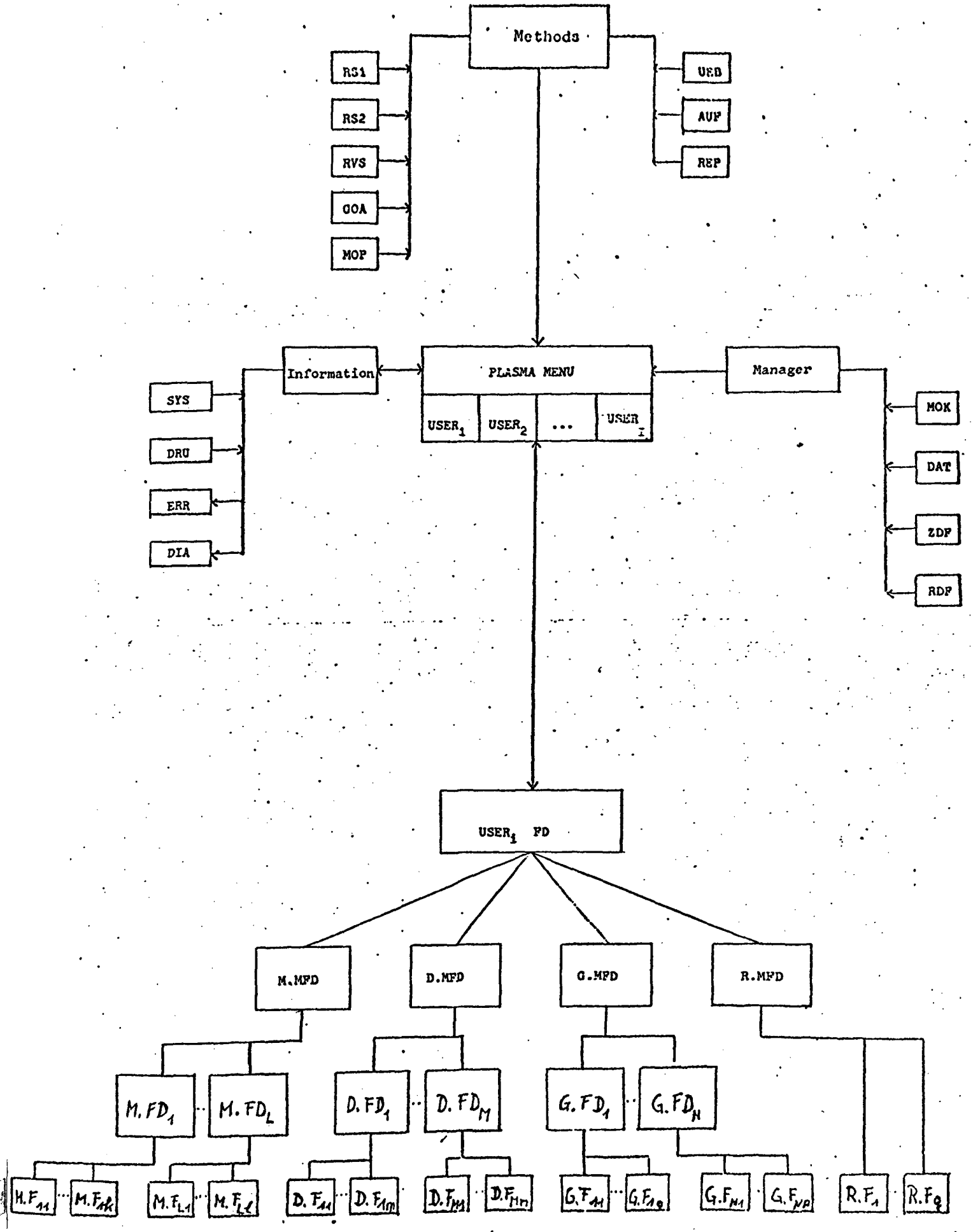

Figure 1

PLASMA system structure

2.3. Man-Machine-Communication

In case of PLASMA II it would be too restrictive to talk about the dialogue structure only. Rather it must be pointed out that there is some kind of machine-machine-communication. This is due to the automatic branching into different PLASMA-procedures - depending on the state of the system.

But the main sequence of steps and programs is done by the user - applying the menu-selection technique (in opposition to PLASMA I where we restricted ourselves to a single-answer mode). When the user tries to enter the PLASMA-MENU for the first time, it is examined whether he is allowed to use the system or not (perhaps one could allow specific parts only, too). If the user's project number is not registered in PLASMA-MENU the user will be automatically logged out and all files associated with PLASMA II will be destroyed. If the user is accepted then he gets the menu-list of Table 1 at the terminal.

Most of the procedures of Table 1 are explained in detail later on. At this point we only stress on the possibility to go to the monitor level of the PDP 10 by the command ML*.This procedure has been installed to allow that an expert of the PLASMA system can interrupt the automatic calling-up of procedures (perhaps the expert wants to do some special work, e.g. file manipulation or execution of other programs which do not belong to the PLASMA system).

A normal user can leave the system only by calling up the procedure LGT. The system then deletes all files which the user left in the temporary status.

TABLE 1: MENU LIST 1
OF PLASMA PROCEDURES

MOK	CONSTRUCTING A MODEL OR MANIPULATING A MODEL STRUCTURE
UEB	TRANSLATING A MODEL STRUCTURE INTO AN INTERNAL STRUCTURE
AUF	MATRIX GENERATION BY COMBINING A INTERNAL MODEL STRUCTURE AND DATA
DAT	MANIPULATING DATA (INSERTING, UPDATING, DELETING)
ZDF	DEFINING OBJECTIVES AND MANIPULATING GOAL-DATA
RDF	DEFINING ITEMS TO BE PRINTED IN A REPORT
REP	GENERATING REPORTS
GOA	EXECUTION OF A GOAL PROGRAM
RS1	EXECUTION OF A LINEAR PROGRAM BY ORDINARY SIMPLEX-METHOD
RS2	EXECUTION OF A LINEAR PROGRAM BY ORDINARY SIMPLEX-METHOD (USING EXTERNAL STORAGE)
RVS	EXECUTION OF A LINEAR PROGRAM BY REVISED SIMPLEX-METHOD
MOP	PERFORMING MULTIOBJECTIVE LINEAR PROGRAMMING (ZELENY)
SYS	PLASMA SYSTEMS INFORMATION AND INSTRUCTION
DRU	PRINTING THE RESULTS OF AN OR-MODEL JUST EXECUTED
DEL	DELETING FILES (PASSWORD)
MLX	LEAVING PLASMA AND RETURNING TO MONITOR-LEVEL (PASSWORD)
ERR	SENDING USER ERROR-MESSAGES OR USER REMARKS TO PLASMA SYSTEMS MANAGEMENT
LGT	END OF SESSION (LOGOUT)

The ways of the user through the PLASMA menu-lists are documented by a dialogue protocol. There also the duration of time with respect to specific procedures can be stated. Finally, it is possible to write down other user input and system output. Thus such a system can be used to investigate user behavior and to give the user the possibility of learning. These capabilities of the system contribute to empirical research and to teaching as well.

3. Modeling with PLASMA II

The modeling process with PLASMA II is characterized by a voluntary choice of a procedure which is to be executed. At the beginning of each procedure it is tested whether the conditions for a production run are fulfilled. If the conditions are not met, an error message is typed on the terminal and the program returns to PLASMA-MENU (see Table 1).

In the following we describe the phases of a normal modeling process which consists of model construction, model translation, data manipulation, numerical preparation of the model, model solution, and model output. Thereby we stress upon the details which characterize the PLASMA-procedures in question.

3.1. Model Construction

Establishing a new model requires the model construction procedure (MOK). This procedure allows for:

- creating a new model structure
- changing a given model structure
- combining several submodels
- combining parts of a model with other parts, i.e. selecting model sentences from different model files
- deleting model files
- listing of models.

In PLASMA I these operations were performed by a special procedure on a dialogue basis. The practical use of this procedure showed some dissatisfactions of the users concerning the stiffness of the change operations. Therefore in PLASMA II the normal text corrector (TECO) of the PDP 10 is used until a new editor will have been implemented by the authors. This TECO-system requires some basic knowledge of the PDP 10 command language; this disadvantage from the user's point of view will be removed after implementation of the new editor.

After having entered the MOK-procedure the user is requested to specify the names of the model file-directory and of the model in question. The program then checks whether the requested model file-directory has been registered in the user file-directory and whether the requested model is part of the specified model file-directory. If the names are known to the system the manipulation by TECO can start, the model structure can be stored after the manipulation. If the names of a model file-directory or of a model are unknown, the program asks the user if the file-names in question shall become valid names; a positive answer leads to a corresponding file creation.

The deletion of models causes a deletion of the model name from the relevant model file-directory and of the model file, too. If a model file-directory has been depleted by deletion of models, it will be deleted automatically.

The PLASMA-syntax has been created for algebraic operations and for the treatment of linear equation models. The language is based on logical records (model sentences), matrix notation is not used. We think that this

conception enables a stepwise approximation to the formulation of models by natural language. The length of the logical records is arbitrary, a record is terminated by a semicolon. The physical record length is fixed at 80 characters. The handling with the syntax is made easy by a minimal use of reserved expressions (only: "MODELL", "SATZ", "FUER", "SUMME"). Logical records may be:

- names of models (code: MODELL)
- names of logical records (code: SATZ)
- arithmetic expressions
- global loops within a given record or with validity for several records (code: FUER).

The syntax for arithmetic expressions is characterized by:

- a maximum length of 80 characters for the name of a word (the word may be a constant or a variable)
- up to 16 indices for one word
- no relevant restriction on the numerical value of an index
- allowance for leads and lags in index specifications
- loops for summing-up, for the multiple generation of records, and for a series of records.

An example of the capabilities of the syntax will be given below.

3.2. Model Translation

The source model which has been stored in a model file will at some time or other be prepared with data for execution. In PLASMA we do not generate the matrix by combining the source model with data, we rather translate the source model into an internal representation

of the model structure. Thus it is possible to save the time of plausibility checks and of compilation when one wants to run a given model with different data sets. The translation is done by the procedure UEB[1] which generates a program containing all the logical information of the source model. An example of this procedure is given in Figure 2.

```
MODELL 1; THIS A SIMPLE EXAMPLE FOR DEMONSTRATION
SATZ 1;                                   ARITHMETIC EXPRESSIONS
FUER I=LOW1, HIGH1,STEP1;
SATZ 2;
[SUMME (A1(1-5)*BB**2.)*VARIABLE(I1) : I1=1979,1988,2] +((A(1+2*J-1,K,L+K)
*5.0E-2)/7.0)*R(I,J,K)<= (BALANCE*0,05*(-MINIMUM + MAXIMAL POSSIBLE
VALUE)) * XYZ FUER K=1,3,J=1999,2001;  FUER MAY BE REPLACED BY "!"
000000   LOOP        I
000002        LOW    LOW1
000003       HIGH    HIGH1
000004       STEP    STEP1
000005   LOOP        J
000007        LOW    $1006
000010       HIGH    $1007
000011       STEP    *UNDEFINED
000012   LOOP        K
000014        LOW    $1004
000015       HIGH    $1005
000016       STEP    *UNDEFINED
000017   LOOP        I1
000021        LOW    $I001
000022       HIGH    $1002
000023       STEP    $1003
000024   ISUB        R = X - Y
000025        R =    $H001
000026        X =    I
000027        Y =    $I000
000030   ICOP        X = + Y ( I1, ... , IN )
000031        X =    $T000
000032        Y =    $H001
000033        N =     0
000034   RCOP        X = + Y ( I1, ... , IN )
000035        X =    $T001
000036        Y =    A1
000037        N =     1
000040        IND    $T000
000041   REXP        R = X ^ Y
000042        R =    $H002
000043        X =    BB
000044        Y =    $R000
000045   RMUL        R = X * Y
000046        R =    $H002
000047        X =    $T001
```

Figure 2:
Example of model translation by the PLASMA-procedure UEB

1) The UEB-procedure has been programmed by G.-F. Rueck at the institute of the authors.

In Figure 2 the model consists of the model name ("THIS IS A SIMPLE EXAMPLE ...") and two logical records. The first record ("SATZ 1") defines a loop for the following record(s), the second record ("SATZ 2") defines 9 inequalities (K=1,3 and J=1999,2001). Such a model equation at first is processed from the right until J and K have been interpreted. Then the processing continues from the left, at first interpreting the summing-up over I1 from 1979 to 1988 (step width 2).

Thus the translation into an internal structure - i.e. into a program which can be executed later on - does not require the numerical specification of indices, coefficients, and constants. On the other side it is possible to specify these values in the source model in advance - being dependent on the problem structure. For instance, numerical coefficients, which will be unchanged in their values for all problems, can be defined in the source model while data, which belong to special data constellations, may be stored in a special data file.

The output of the UEB-procedure also consists of a translation protocol, thus showing the places and kinds of model errors with respect to the PLASMA-syntax.

3.3. Data Manipulation

The manipulation of goal data and of other data is done by the procedures ZDF and DAT. The conception concerning file-handling is the same as in the procedure MOK. After having typed DAT the user gets presented a further (sub)menu-list which specifies the operations that can be started with respect to data manipulation (see Table 2).

In Table 2 the procedures are characterized as follows:

EIN means the inserting of new data into a file which must be specified by the user. Inserting thereby is possible from the terminal or from a data bank existing elsewhere. We did not look at the problem of data-bank retrieval because the mechanisms of access to such banks are very different. The specification of the data file by the user allows for different data sets even for the same model.

The data which shall be inserted must be described by their names, by index specifications (if any), and by the numerical value. It is also possible to define data, which are valid for a series of index values, by a single command.

TABLE 2: MENU LIST 2
OF PLASMA PROCEDURES
(WITHIN THE PROCEDURE DAT)

EIN	INSERTING NEW DATA
UEB	SUBSTITUTING OLD DATA BY NEW DATA
ANF	STORING THE RESULTS OF AN OR-MODEL JUST EXECUTED
LOE	DELETING DATA
ZUS	LINKING DATA-FILES
LST	LISTING DATA
END	END OF MANIPULATION
EOS	END OF MANIPULATION (LEAVING THE PROCEDURE WITHOUT STORING THE MANIPULATED DATA)

UEB allows for a change of the data values if the data alredy are in a data file. This operation mainly is requested if some parameters of a model run shall be changed.

ANF adds the results of a model run to an existing file which can be specified by the user. Thus it is possible to divide the execution of a model into several steps - as far as the model structure is separable. At this point PLASMA could be extended for simulation applications, too. Data of a model run can also be stored temporariliy.

LOE deletes data which - after file specification - must be described by name and index combination (if any).

ZUS allows the creation of a new data file by combining several other data files. Thereby a file must not be taken over in total, rather the user can decide to take over only certain data elements into the new file.

LST produces a file-listing on the terminal.

END terminates the work with the DAT-procedure. All data are stored, and it is checked that all files are registered in a file directory.

EOS - in opposition to END - allows for an interrupt of the operations so that the data are restored as they existed before the calling-up of DAT. This protects the user against data loss in case of erroneous data manipulations.

The distinction between goal data and other data is made because of more clearness. In case of goal definition there are some specialities which differ from usual data definition [see the (sub)menu-list of ZDF in Table 3].

The difference between DAT and ZDF can be demonstrated by the operation AEN which changes the mode of action for a given model. Modes of action can be:

- Approximizing (goal programming)
- Maximizing or minimizing (linear programming or multi-objective linear programming)
- Satisficing (inequalities of the "less than" or "greater than" type; the corresponding signs can also be defined in MOK - when constructing the model).

TABLE 3: MENU LIST 3
OF PLASMA PROCEDURES
(WITHIN THE PROCEDURE ZDF)

EIN	INSERTING NEW GOALS
AEN	CHANGING EXISTING MODES OF ACTION (OPTIMIZING, APPROXIMIZING, SATISFICING)
UEB	CHANGING EXISTING GOAL-VALUES
LOE	DROPPING EXISTING GOALS
ZUS	LINKING GOAL DATA-FILES
LST	LISTING MODES OF ACTION AND GOAL-VALUES
END	END OF MANIPULATION
EOS	END OF MANIPULATION (LEAVING THE PROCEDURE WITHOUT STORING THE MANIPULATED ITEMS)

Thus PLASMA enables the user to change the objectives of a given model structure interactively by (see [16]):

- changing rank and numerical values of goals in goal programming
- switching from one objective function to another function
- changing the mode of action.

It seems that these capabilities will be needed more and more for practical applications.

3.4. The Combination of a Model Structure with Data

After having established the model structure (by MOK) and after having prepared the data (by DAT and ZDF) the combination of model structure with data can take place by calling up the procedure AUF[1] in PLASMA-MENU.

The name of the translated model and the name of the data files (goal data and other data) are requested. Then it is checked whether the goal data are in accordance with the requirements of the corresponding OR-method. Further the system searches for model data in the data file. A search is successful if a numerical coefficient has been found, which has the same descriptions as the coefficient of the translated model structure: the system recognizes coefficients by the fact that they are standing before a variable. If there are several coefficients before a variable, algebraic computations are performed to get a single numerical value before the variable in question.

A search for data may be successful, too, if a model variable has the same descriptions as an element of the data file. In this case the system assumes that the

---

1) The AUF-procedure has been programmed by G.-F. Rueck at the institute of the authors.

variable shall become a numerical constant and adds the value to the value of the right-hand side. These capabilities are useful in case of multiperiod models where variables of the time-period before the planning period are treated as constants - though they have the same name as the decision variables.

If it is detected that the data file does not contain a certain coefficient, then the user is requested to insert the value directly at the terminal. After the list of coefficients and constants has been worked off, the matrix is generated. Thereby the procedure is as usual: row, column, and numerical value are written on a file. Also the type of equation (objective function, restriction of a special type) is written down. In case of goal programming each goal value is accompanied by the corresponding rank and weight of the goal.

Because the names of the decision variables are not elements of the generated matrix, these names must be written on a separate file so that the results of a model run can be presented with the proper names of the variables.

3.5. Model Solution

After the matrix generation the user can decide to run the model using a special algorithm. It would be an extension of PLASMA that the system itself has to decide on the proper algorithm within the same method. In PLASMA II the choice between several algorithms is possible only in case of linear programming at the moment:

- linear programming by the ordinary simplex-method
- linear programming by the ordinary simplex-method in case of larger problems (requiring external storage during the execution)
- linear programming by the revised simplex-method.

Though the system does several plausibility checks it would be desirable to extend such operations by restructuring the data or by decomposing the matrix automatically - due to the numerical problem structure.

The results of a model run are documented by a protocol of computation and by a file containing the names of the decision variables besides their optimal values.

The implementation of new algorithms or methods is possible by the fact that the matrix is the same, whatever the solution procedure may be. The rearrangement of the model data with respect to a special solution method is done by an additional bridge program, which is called up when the special solution method shall be applied. Thus we do adapt the algorithmic programs of other authors directly. The same is valid for the output of a strange program: the output may be changed to meet the requirements of a further handling by PLASMA.

3.6. Reporting the Results

Information about the results of a model run can be got by

- a computation protocol
- a listing of the results on the terminal whereby the algebraic abbreviations are used concerning the decision variables
- a special report allowing for a naming of the decision variables in natural language and enabling the construction of tables.

In order to construct such a report the procedure RDF must be called up. By this procedure the user can define

- the name of the headline (up to 80 characters)
- the names of the variables, the values of which shall be arranged besides the name (this is done by stating the algebraic abbreviation of the variables together with an existing index combination and by assigning a name to the variable in question).

Tables can be construced in case of indexed variables. One of the indices can be used to determine the columns of the table. This operation requires that the index in question is represented by a "9999" in each row where the index appears.

Figure 3 gives an example of report generation where the report shall use information from the report with number "1" (having been defined by RDF). Figure 3 replicates the input which has been specified by the user within the procedure REP (report generation). The wanted report is described by the input file, the output device, the output file, the number of years (index), the specification of the years, and the report scheme. The user then can decide if he wants to have the report at once or if the parameter input for the report shall be repeated.

```
*** PARAMETER FUER REPORT-GENERATOR ***

EINGABEFILE          : WJD1   .003
AUSGABEGERAET        : DSK
AUSGABEFILE          : BILANZ.AKT
ANZAHL JAHRE         :  3
JAHRE                : 1979,1980,1981,
REPORT               :  1

SIE KOENNEN JETZT UNTER FOLGENDEN AKTIONEN WAEHLEN :

1   ERSTELLEN DES (DER) REPORTS
2   WIEDERHOLUNG DER PARAMETER-EINGABE
3   RUECKKEHR ZUM MENU

AKTION : 1
```

Figure 3:
Report parameters

The execution of the report uses data from the data file and - if existing - the results of a previous model run. The output scheme perhaps is changed concerning the output device. Because of a smaller character number per row on the terminal the report scheme is abbreviated automatically in case of terminal output. Figure 4 gives an example of a constructed report.

BILANZPOSITIONEN (AKTIVA)

	1979	1980	1981
ANLAGEVERMOEGEN	100.00	120.00	130.00
UMLAUFVERMOEGEN	250.00	320.00	280.00
BILANZSUMME	350.00	440.00	410.00

Figure 4:
Example of a generated report

4. Conclusions

On the basis of PLASMA - planning system for mathematical applications on a dialogue basis - we have demonstrated some possibilities of how to construct a more user-oriented computerized planning system. The main point was to describe the capabilities of an existing system; the builder of such a system at a certain moment must put aside the multitude of theoretical ideas that could be realized, too.

PLASMA stresses upon flexibility with respect to models, methods, and data. Prevailing experiences in the system use by students and researchers show that the system is accepted. Critical remarks by users and the development of new methods will change and improve the system.

oOo

References:

(1) Barth, H., Ein graphentheoretisches Konzept zur Beschreibung von Methoden- und Modellbanksystemen, in: Mühlbacher, J. (ed.), Datenstrukturen, Graphen, Algorithmen (Applied Computer Science 8), München 1978, pp. 317-335.

(2) Bisschop, Johannes J., and Alexander Meeraus, GAMS - A General Algebraic Modelling System, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.

(3) Bisschop, Johannes J., and Alexander Meeraus, Towards the Development of a General Algebraic Modeling Language, Paper presented at the Joint National TIMS/ORSA Meeting, New York City May 1 - 3, 1978.

(4) Burger, Wilhelm, and K. Mani Chandy, A Language and Data Base for Modeling, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.

(5) Combs, David, On Choosing a Financial Modeling Package, Paper of Cuffs Planning and Models Ltd., New York 1977.

(6) Esprester, A.C., Die Entwicklung einer Methodenbank und einer Methodenbanksprache, in: Angewandte Informatik Vol. 20 (1978), pp. 203-206.

(7) Fourer, Robert, and Michael J. Harrison, A Modern Approach to Computer Systems for Linear Programming, Paper presented at the Joint National TIMS/ORSA Meeting, New York City May 1-3, 1978.

(8) Gernert, D., Benutzernahe Programmiersprachen, München 1976.

(9) Hirschfeld, David S., On the Current Technology in Commercial Mathematical Programming Systems, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.

(10) Hudetz, W., A Graphical Interactive System Simulation Program for Mini-Computers, Paper presented at the DECUS Sympcsium Europe, München 8.-10. Sept. 1976.

(11) Meeraus, Alexander, An Algebraic Approach to Mathematical Modeling, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.

(12) Mitchell, James G.,
The Design and Construction of Flexible and Efficient Interactive Programming Systems, New York 1979.

(13) Niemeyer, G., Simulationsstudien mit Hilfe des graphischen Dialogs, Discussionpaper, University of Regensburg, Fachbereich Wirtschaftswissenschaften, September 1978.

(14) Piepho, Melvin G.,
Linear Programming PRESOLVE, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.

(15) Schmidt, Reinhart,
Zur Verbindung von Modellen, Methoden und Daten bei Unternehmensplanung mit EDV, in: Müller-Merbach, H. (ed.), Quantitative Ansätze in der Betriebswirtschaftslehre, München 1978, pp. 95-109.

(16) Schmidt, Reinhart,
Flexibility of Corporate Planning Models in Case of Changing Objectives , Paper presented at the conference "Financial Management of Corporate Resource Allocations", Breukelen/Netherlands August 7-10, 1979.

(17) Schmidt, Reinhart, and Wolfgang Janowski,
Zur Gestaltung computergestützter Planungssysteme, in: Zeitschrift für Betriebswirtschaft, Vol. 47 (1977), pp. 417-436.

(18) Stark, Robert L.,
ALPS - A New User-Oriented Linear Programming System, Paper presented at the Joint National TIMS/ORSA Meeting, New York City May 1-3, 1978.

(19) Stark, Robert L.,
PROSE - A New Language for Optimization, Paper presented at the Joint National TIMS/ORSA Meeting, New York City May 1-3, 1978.

(20) Zisman, Michael D., and William D. Northup,
An Algebraic Modeling Language for Mathematical Programming, Paper presented at the ORSA/TIMS Joint National Meeting, Los Angeles November 13-15, 1978.