Specifications for the development of a generalized data base planning system

by J. F. NUNAMAKER, JR., D. E. SWENSON and A. B. WHINSTON

Purdue University
West Lafayette, Indiana

INTRODUCTION

Societal problems and industrial problems, if they are to be studied analytically, will require two broad components: first, a structure or model to analyze the interaction of the variables; and second, a large scale data base. The data base may be used in various ways, such as validation of the model’s parameters, input data for actual runs of the model, and simply providing the data itself for other interests. While the ability to devise meaningful models with appropriate supporting data is of primary importance for the advancement of our capacity to serve societal and industrial needs, the possibility of integrating the data base handling techniques with techniques of simulation and optimization will greatly facilitate this work.

With the advent of large scale computers and complex operating systems during the last six years, the capacity exists for the processing of large scale applications. In addition, much work has been done with respect to the development of Generalized Data Management Systems (GDMS), which were designed to handle and manage large data bases. Typically, such systems have been used by management of large industrial and military organizations. These systems are used for handling inventory, receivables, customer accounting and billing, quality control and other administrative tasks.

Generalized Data Management Systems have contributed to increased use of computers, but a major void still exists. The problem of many application programs interacting automatically with a single data base remains a major barrier to general use of an information system as a planning system.

The use of an information system as a planning system will come about only when a methodology exists for automatically creating the data files needed by the many application programs and answering general queries of the data base at the same time. What is needed is a software system that, as a result of a specific query from a user, can: (1) retrieve the data necessary to answer the query; and/or (2) set up the application program or model that must be run to answer the query. This requires that the data be automatically retrieved and arranged in the proper format required by the model.

The value of such a software system is based upon: (1) the efficiency of storing and retrieving data; and (2) the range of services provided through the interactive query system.

The planning system must be designed so that the user is freed from the mundane task of data preparation, which can be tedious and fraught with human errors, in order to run a model. Often, the user is not familiar with the problems and procedures of data handling, and, in most cases, would prefer not being bothered with the data handling at all.

A user who has just queried a data base will have gained very little if he must further select, re-arrange, reformat, and punch his retrieved data for input to an application program or model. Thus what is needed is a Generalized Data Base Planning System (GPLAN), which results from the extension of a Generalized Data Management System to handle the automatic setup of models from a Data Base as instructed by the user through the Query Language. GPLAN is a natural extension of GDMS and represents the next generation of GDMS’s.

In many areas, there has been considerable work on development of simulation and optimization packages, with the end result that these packages are not really used by the people who should be using them. The reasons for this huge investment in application packages with little resultant usefulness are simple. The packages are so difficult to use, requiring either very much technical knowledge in the particular mathematical programming technique and/or complex file setup and manipulation steps, that few people are able to use the package without relying heavily on technical help or considerable education in the specifics of each particular package.

It is obvious that to increase the usefulness of simulation and optimization packages, the nontechnical and/or management personnel who are knowledgeable in the general area of endeavor should be able to easily use these packages.

What is proposed here is to generalize data base planning systems. For a specific area of human endeavor needing such a system, this would mean that it would be easy to set up a data base, link it with application packages, and query it in a meaningful manner—in short, it
would be easy to set up a generalized planning system. This would mean that ad hoc solutions to specific areas would be replaced with a generalized approach comprehensive enough to solve many of the problems occurring in setting up specific planning systems.

In order to take advantage of the knowledge that can be gained from the consideration of specific systems, a regional water pollution control planning system is described briefly and used for some examples throughout the rest of the paper. This system is discussed in order to develop the proper motivation for a Generalized Planning System.

WATER POLLUTION CONTROL PLANNING SYSTEM

The purpose of the water pollution control planning system is to develop a plan that minimizes the cost of pollution abatement structures while satisfying a set of water quality goals throughout an entire river basin. This planning system uses the most prevalent measure of water quality in use today, the level of dissolved oxygen concentration.

The constraints of this model are constructed by dividing the river into sections and constraining the water quality, interpreted as the dissolved oxygen deficit level, to be met at the end of each section. Starting from the headwaters of the river, new sections are defined whenever the river parameters change significantly, such as effluent flow entering the river, incremental flow entering the river (tributary flow, ground water, etc.), the flow in the main channel being augmented or diverted, or the parameters describing the river changing gradually over a longer distance.

The quality constraints are sequentially dependent in that the quality in each section is a function of the quality in the preceding section. But the possibility of tributaries, flow augmentation, and incremental and effluent flows entering at downstream points, complicates the relationship between the constraints.

Three possible treatment techniques are allowed for in the model: (1) by-pass piping; (2) regional and on-site treatment plants; and (3) flow augmentation. Thus piping flows are allowed from each polluter to each river section, from each polluter to each treatment plant, and from each treatment plant to each river section.

In addition to quality constraints, flow conservation constraints are needed for both the polluters and the treatment plants.

The solution technique of the model is the use of a general purpose non-linear algorithm adapted for this model. The major problem involved in adapting the algorithm was the calculation of the partial derivatives of both the constraints and the objective function. These partials were necessary to set up a local Linear Programming problem to determine the direction of search in the stepwise nonlinear problem. Starting with a point in the domain of the objective function, a new point is calculated from it by making a step to either reduce the value of the objective function, if the original point is a feasible solution to the nonlinear programming problem, or obtain a "more feasible" solution if the point is an infeasible solution ("more feasible" by reducing the value of the most infeasible constraint).

The cost function essentially involves the costs of new or upgraded treatment plants, reservoirs for flow augmentation, and various costs of new pipes to or from the polluters, treatment plants, and river sections.

DIFFICULTY WITH THE PRESENT APPROACH

Let us take an example of solving a mathematical programming problem (although any complex application problem would be similar), such as the nonlinear programming model discussed in the water pollution control planning system. A programmer with knowledge of mathematical programming theory, or a group of people which together has the required knowledge, would write and check out a program to solve the specific problem. Then data would be gathered and stored in the format necessary for the program. To check out the data, separate programs would have to be written to test each part of the input data file for which testing was needed. Several sets of corrections to the data file would probably need to be made.

The programmer(user) would then have to fill out some control cards giving various parameters of his data. Obtaining these parameters might involve some manual calculations and may require running some other programs on the original data.

Finally, the mathematical model could be run and, with several iterations and interpretations by the knowledgeable mathematical programming person, the problem could be solved. A report would have to be written explaining the problem and its computer solution. To solve a similar problem would take almost the same steps, except that a lot of the existing programs could probably be used, although major revisions are also possible.

The existing data on a file for solving this problem is most likely usable only for this application, even though parts of it may be usable for other applications.

The time lag between problem recognition and problem solution will be too long and the cost will be expensive. It is possible that, without countless hours of detailed documentation, the program written is unusable except for one or a few people.

If we consider the water pollution control planning system, set up as an application program with a file, the information on treatment plant cost data and the parameters of the rivers cannot be used easily by anyone other than the developers of the system, unless the new user knows the specific format of the data and how it is stored on auxiliary memory.
MOTIVATION FOR THE DEVELOPMENT OF
GENERALIZED DATA BASE PLANNING
SYSTEMS

We must define what a Generalized Data Management System is, before we define the characteristics of a Generalized Data Base Planning System.

Groner and Goel\(^1\) define and characterize a Generalized Data Management System as follows:

"A GDMS consists of data, structure, and a set of algorithms for manipulating the data and the structure. It acts as a communication channel between the user community and the data base. Minker\(^1\) characterizes a GDMS as follows: 'A data management system is considered generalized when it permits the manipulation of newly defined files and data with the existing programs and systems.' [A GDMS] facilitates reference to data by name and not by physical location,' and, '[it] facilitates the expression of logical relations among data items.' A well designed GDMS permits users to access and manipulate elements of the data base in a way that is both natural and convenient for them and efficient in terms of its system utilization.

"Much of the benefit derived from a GDMS results from the insulation of the people and programs from the data. In conventional systems the structure of the data must be explicitly embodied in each program accessing the data. This limits the application of programs to data whose structure has been defined to them. It also limits the ability to restructure data in response to new needs without modifying every program embodying the old structure. These limitations in applicability of programs and the flexibility of the data base impose a rigidity upon conventional data processing systems. While this rigidity is not serious in repetitive well defined tasks such as payroll or inventory control, it is a decided obstacle to the successful performance of systems that must respond to continually changing requirements.

"GDMS systems evolved from sequential formatted file systems. This evolution has been in the direction of more complex logical data structures and more complex operations upon them."\(^1\)

The work on Generalized Data Management Systems until now has focused on two related, but distinctly different approaches to the design of data management systems.\(^3\) The first approach involved the design of a special query system for retrieving data from a data base. This approach permitted new programmers to readily access data and to ask sophisticated questions of the data base. The query capability made this approach very popular, but it has a serious drawback. Namely, that it is extremely difficult to process other applications written in FORTRAN, COBOL, PL/1, etc., against the data base. Examples of the first approach are Informatics Mark IV and GIS of IBM. As a result of this deficiency, many people preferred the second approach which involved extensions to host languages (FORTRAN, COBOL, etc.) that gave the programmer some general file handling capability. However, the non-programming user found this approach undesirable, since, if he wanted a query answered, he had to write the program(s) in the host language to retrieve the data. Examples of the second approach are General Electric's IDS (Integrated Data Store) and Burrough's Disk FORTE (Disk File Organization Technique).

Now the CODASYL Data Base Task Group\(^6\) has proposed a solution to the problem, but in its specifications has given first priority to the host language approach, with COBOL as the first language. To interface between a host programming language and a data base, a specific subschema Data Description Language and an appropriate Data Manipulation Language need to be provided. When an interface is provided, applications still must be implemented in the specific host language system. (While this may not be a problem when and if the standardized implementation of CODASYL DBTG concepts occurs throughout the industry for the major higher level languages, it is quite a drawback for several years to come.) Even if CODASYL's concepts were all implemented, we are still without a system that a planner or manager could easily use. GPLAN is proposed as such a system.

GPLAN is a synthesis of components from other systems. There are a number of systems that exhibit some, but not all, of the features of GPLAN. Examples of some of these systems are:

1. NAPSS: Numerical Analysis Problem Solving System.\(^7\)
2. SODA: Systems Optimization and Design Algorithm.\(^8\)
3. GDMS: Generalized Data Management Systems.\(^4,5,6\)
   (a) System 2000—MRI\(^1\)
   (b) RAMIS—Mathematica, Inc.\(^10\)
   (c) IMS—IBM\(^11\)
   (d) DISK FORTE—Burroughs Corporation\(^12\)
4. OPTIMA.\(^13\)

NUMERICAL ANALYSIS PROBLEM SOLVING
SYSTEM (NAPSS)

A system that is a special case of GPLAN is the Numerical Analysis Problem Solving System (NAPSS) designed and implemented at Purdue University.

A long recognized goal of Computer Science has been to facilitate the stating of problems in languages appropriate to the specific fields in which the problems exist, and then to provide for this solution without the services of highly trained programmers and analysts. These systems are "problem solving systems" and their languages are problem-oriented languages. Thus the aim of the NAPSS project is to make the computer behave as if it had some of the knowledge, ability, and insight of a professional numerical analyst.
A user unskilled in numerical analysis can describe relatively complex problems in a simple mathematical language. Then the system selects algorithms, performs analyses, and gives diagnostics of possible difficulties and meaningless results.

The problem-oriented language of NAPSS uses some of the applicable notation of Fortran, Algol, and PL-1, since they have quite similar facilities for describing computational algorithms. But it goes beyond these languages to include mathematical concepts such an integration, differentiation, algebraic and differential equations, and approximation as part of the basic language.

The basic approach to the system design is through the development of polyalgorithms which become the numerical analysis packages that are the essential elements of the problem solving system. "A polyalgorithm is formed by the synthesis of a group of numerical methods and a logical structure into an integrated procedure for solving a specific type of mathematical problem." The goal of a polyalgorithm is to combine a number of algorithms (corresponding to numerical methods) with a strategy for their selection, and use a procedure which is relatively efficient and very reliable.

The NAPSS system exists as an extension of a procedure-oriented language in an environment, permitting both on-line and remote use of the system. While NAPSS operates most frequently in an online time-sharing environment, it will accept programs submitted for batch processing. Parameter values that are needed by NAPSS can be entered at a terminal in conversational mode or be present on a standard or user-named input file for either conversational or remote mode.

In Figures 1a and 1b we can see how NAPSS can be put into a more general structure with a renaming of its components.

SODA (SYSTEMS OPTIMIZATION AND DESIGN ALGORITHM)

SODA is a computer-assisted decision making system for the design of information processing systems. SODA generates a complete information systems design, along with cost/performance projections of how the designed system will perform on a specified hardware/software configuration. SODA consists of four major components:

SSL: SODA STATEMENT LANGUAGE
SSA: SODA STATEMENT ANALYZER
SGA: SODA GENERATOR OF ALTERNATIVES
SPE: SODA PERFORMANCE EVALUATOR

SSA is a computer program that analyzes the requirements of an information processing system stated in SSL. The Statement Analyzer also provides feedback information to the user to assist him in achieving a better problem statement.

SSA also produces a number of networks which record the interrelationships of processes and data and passes the networks on to SGA and SPE.

Each type of input and output is specified in terms of the data involved, the transformation needed to produce output from input and stored data. Time and volume requirements are also stated. SSA analyzes the statement of the problem to determine whether the required output can be produced from the available inputs. The problem statement stored in machine readable form is processed by SSA which:

1. checks for consistency in the Problem Statement (PS) and checks syntax in accordance with SSL; i.e., verifies that the PS satisfies SSL rules and is consistent, unambiguous, and complete.
2. prepares summary analyses and error comments to aid the problem definer in correcting, modifying and extending his PS.
3. prepares data to pass the PS onto SGA, and
4. prepares a number of matrices that express the interrelationship of Processes and Data Sets.

SGA is a procedure for the selection of a Computer System (CPU, core size, auxiliary memory devices) and the specification of alternative designs of program structure and file structure. SGA constructs a configuration of equipment in order to evaluate performance of the system. A number of models are used (to compute timing estimates) that select timing factors for alternative hard-
ware/software configurations from a data file. SGA simu­
lates the jobstream as it would be processed on the
selected configuration, and, using the factors from the
hardware/software library, SGA and SPE produce
detailed cost/performance projection reports so that the
user can evaluate the final design.

There are a number of systems similar to some aspects
of SODA, such as SCERTI4,15 and CASE.16

In Figures 2a and 2b it is shown how SODA fits into a
more general structure.

GENERALIZED DATA MANAGEMENT SYSTEMS

SYSTEM 2000

SYSTEM 2000, developed by MRI Systems Corpora­
tion, is a general-purpose data base management system.
The basic system provides a comprehensive set of data
base management capabilities, including the ability to
define new data bases, modify the definition of existing
data bases, and retrieve and update values in these data
bases.

In SYSTEM 2000, the basic components of data base
definitions are data elements and repeating groups. Val­
ues are stored in data elements. Repeating groups
describe a structure for storing multiple sets of data val­
ues (data sets) and also serve to link hierarchical levels of
the definition.

Values for each element and logical entry (record) may
vary in length. The user may specify without restriction
which elements in the data base are to be inverted and
become key fields, and what hierarchical relationship an
element will have with other elements in the data base.
Data security is maintained by password control to the
data base and additional password control to each compo­
nent.

The Procedural Language feature of SYSTEM 2000
enables users to manipulate data in a SYSTEM 2000
data base from COBOL or FORTRAN. This feature pro­
vides the mechanism to address any part of the data base
of interest to the procedural program, to retrieve data in a
sequence and format suitable for procedural processing,
and to update the data base from the program.

RAMIS: RANDOM ACCESS MANAGEMENT
INFORMATION SYSTEM

RAMIS, developed by Mathematica, Inc., is a data
base management system which permits a user to
describe and build data bases, maintain the data in the
data bases through updates, additions, and deletions,
retrieve information from the data bases and display it in
meaningful report formats, or pass the information to
other processing programs.4

RAMIS is both a report generator and a data manage­
ment system, since it has a simple and logical English­
like language, which permits the user to both request
information from data bases, and, at the same time, pro­
cess it into finished reports.

RAMIS organizes the physical placement of data into
tree structures on random access devices by exploiting the
hierarchical relationships of the data fields. The user has
to supply only some minimal information about these
relationships. User written programs in Fortran, Cobol,
Assembler, or PL/1 can also be linked directly into
RAMIS.

IMS: INFORMATION MANAGEMENT SYSTEM

The Information Management System (IMS) is a sys­
tem designed to facilitate the implementation of medium
to large common data bases in a multi-application envi­
ronment.2 This environment is created to accommodate
both online message processing and conventional batch
processing, either separately or concurrently. The system
permits the evolutionary expansion of data processing
applications from a batch-only to a teleprocessing envi­
ronment.

The data base processing capabilities of IMS are pro­
vided by a facility called Data Language/I. The data
base functions supported are definition, creation, access,
and maintenance. The full data base capabilities of Data
Language/I can be used in the IMS batch processing or
teleprocessing environment.

From the collection of the Computer History Museum (www.computerhistory.org)
Data communication capabilities are characterized by the use of input/output terminals in remote and local environments, connected to the computer, which provide the user with access to the data base. IMS also has extensive message scheduling, checkpoint, and restart facilities.

**DISK FORTE**

Disk FORTE, the Burroughs manufacturer system, is programmer-oriented at the most basic level. Nearly all features and capabilities of other data management systems must be programmed in Disk FORTE. Yet, it permits both hierarchic and network data structures (user-programmed, of course) which make possible more complex associations among data.

Disk FORTE makes its data management capabilities available through extensions to COBOL which are handled by a pre-compiler.

Figure 3 shows the generalized structure of SYSTEM 2000, RAMIS, IMS and DISK FORTE.

**OPTIMA**

OPTIMA is an advanced mathematical programming system for the CDC 6000 series computers. It includes advanced algorithms and techniques in addition to algorithms for standard linear programming formulations. User-controlled data and storage management features are also provided.

"The basis of OPTIMA is a revised product-form, composite, bounded variable, separable, multipricing, simplex, linear programming algorithm." Some of the advanced features that OPTIMA provides are: the capability to form a nontrivial starting basis; the ability to start a solution using a previous basis; dynamic control of the frequency of inversion of the basis matrix; provision for partial and multiple pricing; the use of maps to exclude or include specified vectors in the basis; and elaborate recovery procedures. A dual optimization algorithm is available for those problems in which its use might be advantageous; and postoptimal analysis of a problem can be accomplished as an integral part of OPTIMA.

Through the use of the Applications Control Language (ACL), OPTIMA allows dynamic control of the progress and execution of the program. ACL has logic and computational capability and provides verbs and phrases for modifying various parameters and controlling the progress of the solution.

An ACL program must be written for any study. This program defines the data files to be used and the operations to be performed on these files, sets any parameters and controls necessary, and calls various routines required to carry out the study.

Two other languages, the Matrix Generator Language (MGL) and the Report Generator Language (RGL) operate within control of the ACL. MGL provides capabilities for generating a problem matrix automatically. RGL provides the capability for generating reports in any desired format, and permits computer-generated solutions to be used for further arithmetic and logical computation.

**SUMMARY OF COMPARISONS**

Figure 4 shows the structures of GPLAN. In considering the four information processing systems (1) NAPSS,
TABLE 1—Comparison of information processing system

<table>
<thead>
<tr>
<th>CHARACTERISTICS</th>
<th>DISK FORTE</th>
<th>CPL/LS</th>
<th>IMS</th>
<th>SODA</th>
<th>IMS/DB</th>
<th>OPTIMA</th>
<th>RAN5</th>
<th>SYSTEM 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mnemonic value</td>
<td>DISK File</td>
<td>CPL/LS</td>
<td>IMS</td>
<td>SODA</td>
<td>IMS/DB</td>
<td>OPTIMA</td>
<td>RAN5</td>
<td>SYSTEM 2000</td>
</tr>
<tr>
<td>Implementers</td>
<td>Burroughs</td>
<td>IBM</td>
<td>Burroughs</td>
<td>University of Michigan</td>
<td>Control Data Corporation</td>
<td>Mathematics, MRI Systems Corporation</td>
<td>IBM</td>
<td>Burroughs</td>
</tr>
<tr>
<td>Hardware</td>
<td>8200, 8300</td>
<td>CDC 6600, IBM 360, IBM 270</td>
<td>CDC 6600, IBM 360, IBM 270, IBM 6500, CDC 6500</td>
<td>CDC 6600, IBM 360, IBM 270</td>
<td>IBM 360, 370</td>
<td>IBM 360, 370</td>
<td>IBM 360, 370, University of Michigan, CDC 6600, CDC 6500</td>
<td></td>
</tr>
<tr>
<td>What is the system?</td>
<td>Burroughs</td>
<td>IBM</td>
<td>Burroughs</td>
<td>University of Michigan</td>
<td>Control Data Corporation</td>
<td>Mathematics, MRI Systems Corporation</td>
<td>IBM</td>
<td>Burroughs</td>
</tr>
<tr>
<td>System implementor's DBMS</td>
<td>Burroughs</td>
<td>IBM</td>
<td>Burroughs</td>
<td>University of Michigan</td>
<td>Control Data Corporation</td>
<td>Mathematics, MRI Systems Corporation</td>
<td>IBM</td>
<td>Burroughs</td>
</tr>
<tr>
<td>User Language</td>
<td>COBOL</td>
<td>FORTRAN</td>
<td>BAL</td>
<td>FORTRAN</td>
<td>BAL</td>
<td>FORTRAN</td>
<td>BAL</td>
<td>FORTRAN</td>
</tr>
<tr>
<td></td>
<td>extensions</td>
<td>own</td>
<td>own</td>
<td>own</td>
<td>own</td>
<td>own</td>
<td>own</td>
<td>own</td>
</tr>
<tr>
<td>File Organization Structure</td>
<td>Hierarchical</td>
<td>Hierarchical</td>
<td>Sequential</td>
<td>Sequential</td>
<td>Sequential</td>
<td>Special</td>
<td>Special</td>
<td>Special</td>
</tr>
<tr>
<td>(Storage Structure - under user control if visible to user)</td>
<td>or Network</td>
<td>or Network</td>
<td>or Network</td>
<td>or Network</td>
<td>or Network</td>
<td>File</td>
<td>File</td>
<td>File</td>
</tr>
<tr>
<td>Data Structures</td>
<td>Hierarchical or Network</td>
<td>Hierarchical or Network</td>
<td>Sequential</td>
<td>Sequential</td>
<td>Sequential</td>
<td>Trees</td>
<td>Hierarchical</td>
<td></td>
</tr>
<tr>
<td>Allowed Users:</td>
<td>Nonprogramming</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Programming</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>User Language Classifications:</td>
<td>Tabular</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Command</td>
<td>Free</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Procedural</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Nonprocedural</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Query Language and Analyzer:</td>
<td>Data Base Queries Allowed? (user programmed)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Application Package Queries Allowed</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Problem Oriented Language</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Application Packages or Models:</td>
<td>Mathematical and/or optimization models</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Statistical Packages</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Simulations</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Any Program or Group of Interconnected Programs:</td>
<td>COBOL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

(2) SODA, (3) GDMS, and (4) OPTIMA, it is observed that we must have at least a query language, a query analyzer and a Generalized Data Management System. NAPSS and SODA are missing the data management capabilities; the GDMS's are missing those components needed to readily interface models; and OPTIMA is missing the data management and query capabilities. These differences are elaborated on in Table I.

From the collection of the Computer History Museum (www.computerhistory.org)
A synthesis of the previous systems results in the definition of the following nine components of a Generalized Data Base Planning System:

- A Generalized Data Management System (GDMS)
- Raw Data for the Data Base
- A Query Language
- A Query Analyzer
- A Collection of Application Packages or Models
- Administrative Report Module
- User’s Interface
- Extraction Files
- Users

Each of these components is discussed in more detail in the following sections. An overview of GPLAN for Water Pollution Control is shown in Figure 5 and Figure 6.

**A generalized data management system (GDMS)**

A Generalized Data Base Management System that is implemented at a particular installation under a specified operating system must be available. The GDMS must meet minimum requirements as to data structure definition, data base loading, data base updating, and data base retrieval, and it must satisfy some minimum set of queries, as specified in the following section.

Six general functions must be provided by the GDMS:

- **Input**—the system accepts data values or information about data structures.

- **Search**—the system searches the data base by examining the descriptions of data structures and storage structures to ascertain the existence and location of certain data values.

- **Storage**—the system accesses a data base to add, insert, modify, or delete data values.

- **Maintenance**—the system generates or modifies descriptions of data, data structures, and storage structures to adapt to change.

- **Retrieval**—the system accesses a data base to obtain data values previously stored.

- **Output**—the system exhibits data values or information about data structures and storage structures.

**Raw data for a data base**

The raw data for the planning system must be available in whatever form it can be collected. A Data Input Module is used to convert the raw data into a form necessary to be loaded by the GDMS into the data base.

To refer to the data in the data base, the following terms, adapted from the CODASYL Data Base Task Group Report, are defined:

- **DDL**—Data Description Language. A language for defining data and their relationships. The DDL is divided into schema and sub-schema.

- **Schema**—That part of the DDL which defines the “universal” data base.

- **Sub-schema**—That part of the DDL which describes the data known to each application program or model.

A schema describing the data on the data base must be prepared. (Note that the DDL is not necessarily the one defined in the DBTG report.)

While preliminary Data Input Modules would be dependent on the format of the raw data and dependent on the GDMS, it is hoped that some measure of independence can be achieved in the same manner as the data transformations used in implementing the user interface mentioned later.
A query language

A Basic Query Language (BQL) defines all those queries that can be handled by the GDMS alone and allows the user to request that the DBMS display certain data or answer questions about the data. The BQL is a comparative and computationally oriented language used to compare item values with other item values, or with constants, or with results of computations. Arithmetic operators define the computations to be performed, and logical operators combine simple expressions into compound expressions. The BQL can be extended as application packages are added to include questions that are answered by the new application packages, i.e., each package adds a set of new query components to the BQL. The BQL, together with all the query components from the applications packages, makes up the Query Language (QL).

General capabilities provided by the QL are:

- **Selective retrieval**—the user specifies the selection conditions to be satisfied in retrieving the desired data.
- **Nonselective retrieval**—the user specifies unconditional retrieval of data.
- **Conditional retrieval**—the user employs verbs such as IF . . . ELSE to test items for some qualifying values in determining alternative courses of action.
- **Statistical retrieval**—the user may query the system about data. Statistical computations for all the instances of one item, for example, would include maximum value, minimum value, mean value, median value, mode value, standard deviation, and total number of instances.

The QL thus should provide the capability for easily asking questions of the data base and asking questions that can be handled by the various application packages. By including optimization models in the application packages, the policymaker is able to move efficiently beyond the "What if" to the "What's best" question. Moreover, much additional research needs to be done on the query language (and its associated analyzer), since its applications packages, makes up the Query Language Analyzer (QA).

As an example of the power needed in the query language, consider the added components of the planning model from the Regional Water Pollution Control Planning System. It is able to handle two distinct types of planning problems. First, it is able to select a least-cost combination of treatment methods, given water quality goals and economical, political, and water quality information from the river basin. The second type of question which can be handled is directed toward individual projects. Types of individual planning problems that could be analyzed are:

- **What is the least cost solution for towns X and Y to handle their effluent? Should they combine to construct and operate a joint treatment plant?**
- **What would be the least cost solution if consideration is given to the political constraints that may become operative?**
- **What is the optimal plan for capacity expansion giving consideration to the growth and shift of population and industrial growth in the basin?**
- **What are the least cost and optimal treatment plans that correspond to the task of providing water of high enough quality for certain recreation activities?**
- **What is the sensitivity of the optimal pollution control plan to costs and constraints?**
- **What is the optimal tradeoff between water quality, flow and alternative costs? For example, what would the difference in costs be if a plan were to permit the violation of a water quality standard once in 25 years as compared to once in 50 years?**

GPLAN is a methodology for obtaining answers and responses to the above type of questions.

A query analyzer (QA)

A Query Language Analyzer must be able to analyze the BQL and as many application package query components as are available. A user enters his query in the QL, and the Query Analyzer analyzes the question and provides the user diagnostics to help him reformulate his question, if necessary. The query stored in machine readable form is processed by the QA which:

1. Checks for consistency in the Query and checks syntax in accordance with the Query Language; i.e., verifies the QL rules and is consistent, unambiguous, and complete.
2. Prepares error comments to aid the user in correcting, modifying and extending his Query.
3. Decides whether to pass the Query to the GDMS or one of the application packages.
4. May request additional information from the user if the action to be initiated requires it.

A collection of application packages or models

Application packages are simulation and optimization models, statistical packages, and other self-contained systems currently functioning under a specific computer and operating system.

We want to make it as easy as possible to add application packages, so we generalize the process by describing how we add a specific package.

We will make several assumptions about application packages: First, they are already running as batch jobs rather than interactive jobs. They may have quite long running times and making them interactive may simply mean waiting at the terminal; Second, they require user preparation to get the data ready; and third, they require other programs to run before the input data is complete.
We must know certain characteristics of an application package or model before we can consider tying it into GPLAN:

A. Input
1. For each data input, we must know what kind it is (pure data or commands) and the associated types and formats.
2. For each data input, we must know what kind of device it is assigned to (sequential or random-access).
3. We must know the passes and correct logic steps and transformations to go from the data base to each input.
4. For each data input, what must be included in system queries to the GDMS for 3 above?

B. Output
1. Is the output self-explanatory or does it require minor explanation in the form of good documentation in the Query Language description?
2. Does the output require much technical knowledge to interpret the results? If yes, an output interpretation module is required (e.g., nonlinear river basin model solution).

C. What query components can it add to the Basic Query Language?

D. Minimal Documentation to be used by:
   1. Systems personnel
   2. Non-programmer researcher

Administrative report module

The Administrative Report Module will produce standard reports that must be completed and filed on a routine basis. These reports may be automatically triggered by queries or specifically requested from the console.

The standard reports will be supplied with data from the extraction file structure.

User's interface

Each interface component required for each part of an applications package must be defined:

A. Simple input linkage—direct to the Data Base.
B. Phased inputs—including self-started and analyzed Data Base retrievals.
C. Any combination of A. and B.
D. Simple output—direct from package.
E. Output interpretation module needed.

Extraction files

Between the data base and the set of application packages and the Administrative Report Module is a set of extraction files containing those items from the data base that are used for the packages and reports. Questions to be answered on extraction files are:

A. How many should there be?
B. What items should they contain?
C. What should their data structure and storage structure be?
D. How often should they be updated?
E. If a new application package or report is added, what changes should be made in the extraction file structure?
F. Should some application packages bypass extraction files entirely?

Users

There are two types of users connected with GPLAN: the technical systems personnel and the nontechnical administrator or manager.

A data administrator and his systems staff are responsible for: all original data input; updating of data; restructuring the Data Base and extraction files as necessary; changing machines and operating systems; adding new packages, standard reports (Administrative Report Module), and other additions or improvements. These systems users, taken as a group, must understand fairly well every component of GPLAN. They possibly could get by with not being familiar with some of the application packages, but then would have to get consultation to patch up or improve on these.

The major group of users are non-programming administrators. These are users who don't know how to program, probably don't want to learn, and definitely shouldn't have to learn. They have a good understanding of the area for which the planning system was designed, or will have to have some training in this area before using GPLAN. Most of the details of the GPLAN implementation should be transparent to these users, and they should not notice any changes in the system, except the addition of new capabilities (possibly requested by them), new efficiency, or new packages. The success or failure of GPLAN depends on how well these users are able to carry out their querying of the data base and interaction with the application packages using only the query language and its documentation.

RFMS and RAMIS easily meet and/or exceed the minimal requirements for a GDMS as specified above. Thus all software being developed for GPLAN is being implemented on the CDC 6500 and the IBM 370/155.

Research is proceeding on the Query Language—Query Analyzer components in three areas. One area of investigation is the relationship between the QL and QA and the SODA Statement Language and SODA Statement Analyzer as used in the SODA project. Second, research on QL and QA is proceeding as a result of the development of the water pollution control models. Finally, the state of the art in artificial intelligence is being investigated for incorporation into the query language and analyzer.
STATUS OF GPLAN

There are two major efforts under way with respect to the development of GPLAN:

- Development and construction of the software for GPLAN.
- Work on a real world planning system (Water Pollution Control) and development of user training aids.

Development of software for GPLAN

Two different GDMS systems are being evaluated in parallel with the construction of GPLAN. Development is proceeding using RAMIS and RFMS (Remote File Management System). RFMS is a version of SYSTEM 2000 that was originally developed at the University of Texas at Austin. RFMS has been converted to run under the Purdue MACE Operating System, and substantial improvements have been added to the original version.

The two most difficult areas in the development of software for GPLAN are the User's Interface and the Extraction Files. Thus, a Data Description Language schema for data base description, and the Data Description Language subschema for the description of application package and administrative report data requirements, have been defined. Research is proceeding on the automatic mapping between the data base schema and an application package's subschema. Included in this mapping is a set of extraction files to be composed of subsets of items from the data base. An integer programming model has been defined which relates the data items of the data base to those on the extraction files as required by the application programs. Also, a cost function representing the cost of operating GPLAN has been defined. An important aspect of the problem is the optimization of the extraction files by solving for the extraction file arrangement which minimizes operating costs.

Work on a real application

Data is presently available to us from a previous study on the West Fork of the White River in Indiana for the development of a demonstration project concerning the Water Pollution Data Base Planning System. The insight achieved through the development of a specific planning system has already proved to be a tremendous aid in the accomplishment of the major goal of having a truly easy-to-use system.

The query system is being implemented in two modes:

1. Standard 80 column teletype
2. Graphics Terminal

The usefulness of GPLAN is enhanced considerably through the effective use of a graphical display. The graphics terminal offers the obvious advantage of being able to output designs, graphs, etc., in a more visible and appealing form. But the main advantage of interactive graphics is that it offers the user the capability of complete user interaction with the planning system.

Consider, for example, the river basin planning system. The optimization models output a diagram of the optimal solution to a specific water pollution control problem, showing the actual location of treatment plants and cooling towers on a computerized representation of the river basin, i.e., the actual solution is illustrated on a map of the river basin. The user may not have much confidence in the results, but he can at least relate to the output in this form. However, if he is given the opportunity to improve the solution by making adjustments to the design, or by changing the location or capacity of a treatment plant through the graphical query system, he finds that he is part of the decision making or planning process. Now, we can let the user input his own design and then compare the value of the objective function for his design with the optimal design. GPLAN can then indicate whether or not his design is even feasible. The user can also be given the opportunity to experiment with the values of the constraints and try different water quality goals. The result of this interaction is that the user has increased confidence in the planning system with a unique appreciation of the special talents and capabilities of man and machine.

The user can only be convinced that the mathematical solution is "good" if he can't improve on it himself. This experience was also supported by observations from a project concerned with the location of a major highway in southern California. Interactive graphics allows the user to utilize insight that often can't be built into models. This man-machine interaction enhances the planning system and brings the user into the decision making process.

REFERENCES

1. Pingry, D., Whinston, A., A Survey of Planning Techniques for Pollution Control, Krannert School of Industrial Administration, Purdue University, August 1972.
2. Pingry, D., Whinston, A., A Multi-Goal Water Quality Planning Model, Krannert School of Industrial Administration, Purdue University, August 1972.
10. *RAMIS Information Bulletin—Description and Specifications*,