Proceedings of the 1984
Winter Simulation Conference
S. Sheppard, U. Pooch, D. Pegden (eds.)

GENERALIZED MODEL BUILDING WITHIN AN INTEGRATED DECISION SUPPORT SYSTEM

David P. Yancey, Ph.D.
Michael Sale
Pritsker & Associates, Inc.

and

Lt. Robert A. Carringer
ICAM Program Office

This paper describes of a Generalized Model Builder (GMB) for use within a framework for integrating multiple analytic tools, operating from an integrated model database. This system provides a uniform interface for a variety of applications, including: simulation, optimization, financial analysis, steady-state network analysis, scheduling-balancing heuristics, and project planning. Graphical, prompt, and spreadsheet-like interfaces are described.

BACKGROUND

Introduction

The Integrated Computer Aided Manufacturing (ICAM) Program, sponsored by the United States Air Force, was established in 1977 to improve productivity in the aerospace industry through the systematic implementation of computer technology [1]. As part of the ICAM Program, several efforts have been initiated in the design and development of decision support systems. These efforts have included development of prototypes [2, 6, 7] and a long-range integrated decision support system (IDSS Build 1).

The IDSS Build 1 project was initiated in June, 1982. One of the outcomes of this project is an IDSS which provides a framework for integrating a variety of analytic tools, operating from an integrated model database with a consistent problem solving interface. Analytic tools include: simulation, optimization, financial analysis, steady-state network analysis, scheduling-balancing heuristics, project planning, and so on.

IDSS Build 1 is planned to provide decision support to the aerospace manufacturing industry for operations planning and control. IDSS will be implemented in the ICAM Integrated Sheet Metal Center (ISMC) at Boeing Military Airplane Company as part of the ICAM plan to demonstrate gains in productivity from Computer Integrated Manufacturing systems.

Decision Support Requirements

When the IDSS Build 1 project was initiated, research was performed on the needs for decision support in aerospace manufacturing [3]. It became clear that the IDSS requirements could be divided into two areas. The first area related to the need for performing specific analyses related to given problems in aerospace manufacturing. The second area related to requirements for effective use of the system in performing these analyses.

It was observed that problems and analyses of interest and significance varied widely among the aerospace manufacturers polled. Even in the cases where DSS requirements were similar, there was the need to "tailor" the user interface such that prompts, messages, and reports were contextual to the installed environment. Other conclusions regarding the design of IDSS are summarized in the list below.

- IDSS should allow the user to interact with the system using an easily learned set of commands.
- IDSS should support a wide variety of analysis tools and permit them to be combined for analysis of a problem.
- IDSS should provide a uniform user interface.
- IDSS must be extensible; analytic tools must easily be added, replaced, or deleted from the system.
- IDSS must accommodate not only the different user roles but also, within each role, various levels of experience in using the system. Depending upon user role, various IDSS functions should be available; depending upon experience, various levels of on-line help and explanations should be available.

- IDSS should be able to support a large number of users, each of which may have different analysis requirements.
- IDSS must be designed to support contextualized communication with the user. Prompts, help messages, analyses and so on should be tailorable for a specific installation.
- IDSS databases must be accessible via methods comprehensible by non-computer experts. IDSS application databases should be available for inquiry and update.
- IDSS must provide a mechanism to preprogram analyses that can be used by experienced users to support less experienced users.
- IDSS must provide an easy means of creating reports, invoking analyses, and managing applications.
- IDSS must provide an efficient means for Model Builders to tailor one or more analyses for use by Decision Makers.
- IDSS must support parameterization of an analysis in order to efficiently support the analysis of multiple scenarios or experiments.

A conceptual view of the IDSS Build 1 system, from the perspective of a model builder/analyst, is illustrated in Figure 1. As illustrated, IDSS provides a "shell" within which analyses, analytic techniques, spreadsheet software, model builders, and other software may be implemented. The system command language (DSL - Decision Support Language), functions, and support are available as user type and experience prescribe [4].

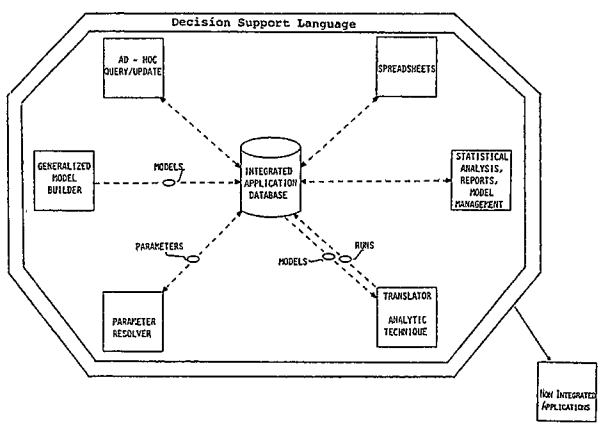

Figure 1. The IDSS Build 1 Shell - A Model Builder/Analyst View

A complimentary view of the system from the perspective of a system installer/maintainer is illustrated in Figure 2. IDSS is driven by a data model of itself (CDM - Common Data Model). Tools are provided to alter this data model (schema generation), change the command language (parser generator), introduce new applications and functions, and otherwise alter the system configuration and behavior.

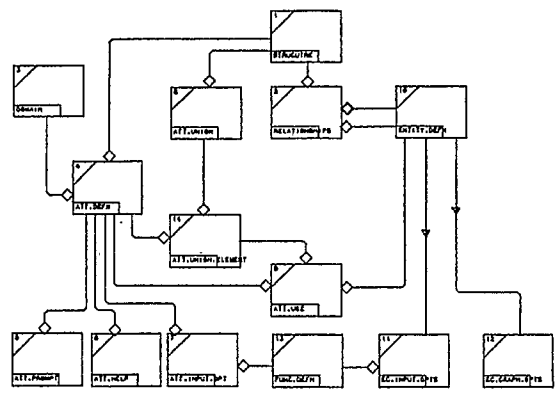

Figure 2: IDSS Common Data Model

GENERALIZED MODEL BUILDING

One of the greatest challenges of the IDSS Build 1 system was the design and development of a generalized model builder (GMB). The model builder is required to support a wide variety of analytic techniques possessing widely different input requirements; some models are graphical (e.g., PERT, GERT-E, network simulation); others are tabular (e.g., linear programming, statistical analyses).

The GMB is designed to be data driven. The same model build software serves many applications and users; however, different application "meta data" is loaded depending upon the characteristics of a selected analytic technique. Meta data is composed of an information model of the application technique and its associated user interaction specifications (e.g., prompts, response constraints, help and error messages, definition of graphical symbols). The installation of new applications is greatly simplified with this design.

As previously noted, application installation is supported by a number of support tools. A schema generator component updates the physical schema of the end user's application database; a symbol sketch facility (see Figure 3) is used to define new icons; other components are provided to populate the application database with metadata.

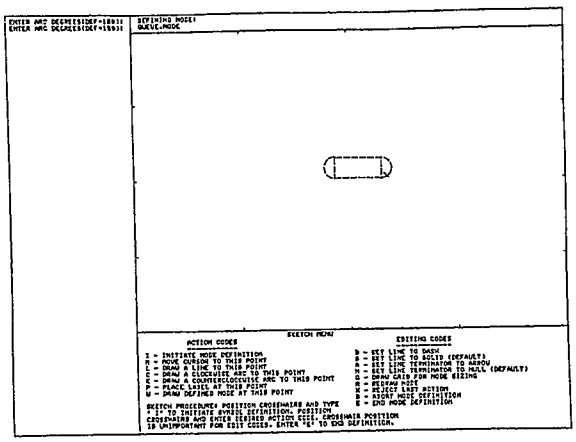

Figure 3: IDSS Symbol Definition Function

Information Structures

Figure 4 is an example application input data structure. This is an information model of an application as it might be specified to the system by an application installer [5]. In this figure, each block represents an entity class (i.e., a relation). Entity classes may be viewed as tables which possess columns (attributes) and rows (occurrences).

Information structures may be declared to support applications which are entirely nongraphical, partly graphical, or entirely graphical. Data structures which support graphical applications must contain a page entity which is the parent of symbol entities. An example graphical data structure is illustrated in Figure 4.

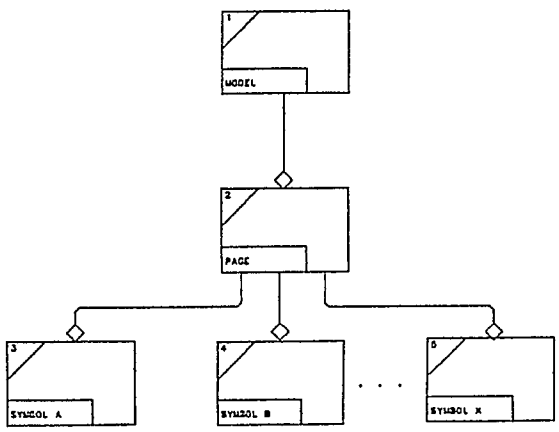

Figure 4. An Example Graphical Application Data Structure

Figure 5 illustrates a data structure in which both graphical and nongraphical entity classes exist. Entity classes themselves can be declared to be addressable by the GMB graphically, nongraphically, or both. All entity classes, regardless of type, are addressable by ad-hoc queries in DSL (see Figure 1).

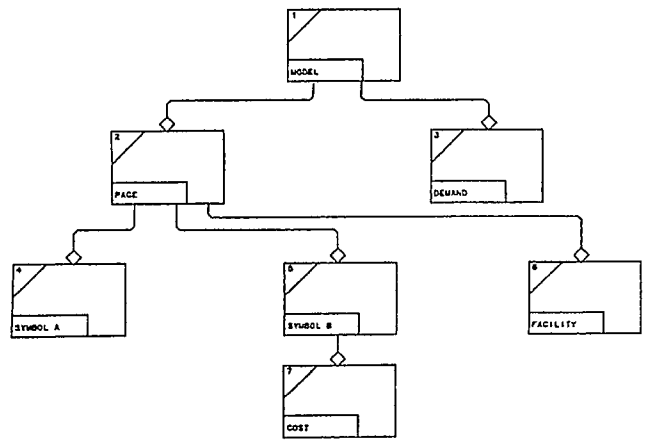

Figure 5. An Example Application Data Structure

In Figure 5, occurrences are added to DEMAND, FACILITY, and COST much like one would add rows to a table. Occurrences are added to SYMBOL A and SYMBOL B much like one would add a new symbol (e.g., a QUEUE in IDEF2) to a page.

In some applications, the structure of information appropriate for building a model may vary depending upon input options or analysis conditions. In IDSS Build 1, three forms of conditional information have been identified: conditional entity classes, conditional entity class occurrences, and conditional attributes.

Consider Figure 5. In this structure, for example, either DEMAND or FACILITY might be populated, but not both. The proper selection might be based on a previously specified attribute value in the MODEL entity class. If this were the case, the user would be warned if an unnecessary entity class were selected for population.

Attributes may have conditionally different interpretations and restrictions, or in some cases, not used. Consider, for example, parameters for a random distribution. Depending upon the distribution selected, a different number of parameters are required, with different constraints and interpretations.

Conditional specifications may relate to prompts, help messages, constraint checking, and functional options. Constraint checks on attributes may involve logical operations. For example, part numbers must be 9 digits and begin with the letters PN when a given model option is specified.

Model Build User Interface

In building or editing a model non-graphically, entity classes are addressed one at a time. The user must select an entity class to be operated upon. Depending upon the entity selected and characteristics of the user's terminal, functions are provided to ADD, EDIT, DELETE, COPY, or LIST occurrences. The user is also invited to specify a mode of data entry. Options include: QUICK, PROMPT, TABLE, and GRAPHICAL. Depending upon the mode selected additional commands may be available to direct processing.

At all points in the GMB, certain universally recognized special commands are provided to support user responses to prompts and processing. Following are special commands and their function.

ABORT	Terminate this command or current action.
END	Terminate selections from this menu, go to the next highest level or menu.
HELP	Provide context specific information about this prompt.
HELP/keyword	Provide detailed information about this command, mode, or application.
%EXPLAIN	Explain the last error message.
%ATTRIBUTE	Describe the meta data characterizating this attribute.
%STRUCTURE	Provide a brief structural diagram of entity classes in this application.
%CLASS	Provide detailed descriptions of this and adjacent entity classes.

The ADD Function

The ADD function supports the introduction of new entity class occurrences. If a previously existing occurrence is named (i.e., non-unique key attributes) the user is informed of the error.

In PROMPT input mode the user is prompted for attributes of the entity, one at a time. The QUICK input mode is like the PROMPT mode; however, the prompts are shorter, as might be desired by an expert user. The user is prompted for additional entity class occurrences until he chooses to enter the END command.

An illustration of a screen for the TABLE input mode is provided in Figure 6. At the top of the screen, the user is reminded of the current application technique (GERTE), the entity class (BRANCH), the function mode (ADD) and the keypad mode (SELECT a field for input). Each column represents an entity class occurrence. Each row in the table is an attribute. In this figure, prompts followed by the > symbol signify key attributes, those followed by a colon are non key. Attribute prompts presented in reverse video possess conditional interpretation. On the second line, messages regarding the current field (column 1, row 7) are presented in reverse video.

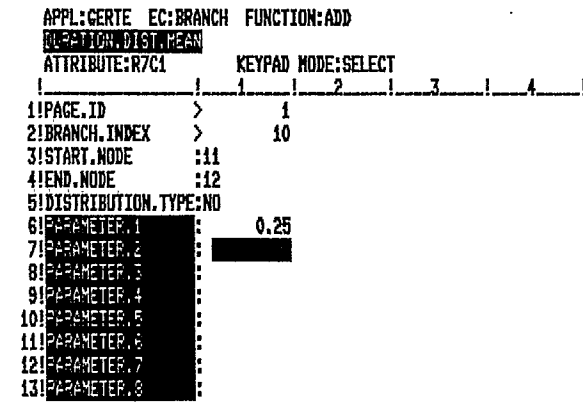

Figure 6. ADD TABLE function

The user may enter an indefinite number of occurrences. (As in a spreadsheet, the user views a window of a much larger table). Occurrences are not immediately stored in the database as they are entered. Storage occurs on command. A user may move the cursor to previous or later columns in order to correct input errors prior to issuing a database storage command.

Special commands which are available in the ADD TABLE mode are listed below. If the parameters of a command are incompletely specified, the user is prompted appropriately.

COPY	Copy an area of the table to another area
DELETE	Delete one or more columns
END	Store the table in the database and terminate the ADD TABLE function
EXPLAIN	Explain the most recent error message
GOTO	Move the cursor to a cell location in the table
HELP	Display a screen of help information
INSERT	Insert a column in the table

QUIT	Terminate the ADD TABLE function without storing the table in the database
RESUME	Recover the table as it existed at the last SUSPEND command
SET	Set the cursor movement direction following a carriage return
SUSPEND	Terminate the ADD TABLE function; do not store the table in the database, but save the table for possible later use

The graphical mode is available when the user requests that a "page" entity class is to be addressed, and the session is being conducted on a graphics terminal. Once initiated, a new graphical page occurrence is defined. The screen will subsequently be cleared and window outlines will be drawn, as illustrated in Figure 7. Prompts will appear in the command window (left of screen) and a menu of symbols (if requested) appears at the bottom of the screen. The available edit commands are:

ADD	Add a symbol to this page and position it on the screen
CHANGE	Change the attributes of a symbol on the screen
DELETE	Delete a symbol from the page
END	Terminate the GRAPHICAL mode
GRID	Toggle the screen grid on or off
HELP	Provide command help
MENU	Display a menu of symbols for this application technique
MOVE	Relocate a symbol on the screen
REDRAW	Refresh the screen

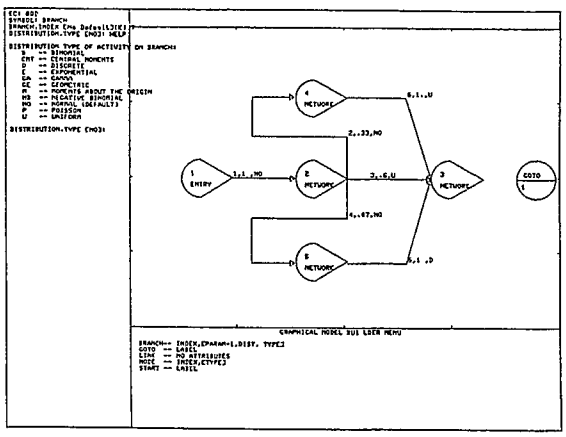

Figure 7: ADD GRAPHICAL function

As symbols are added to the page, the user is prompted with crosshairs to locate the symbol on the screen. Symbols are one of two types: nodes or arcs (e.g., in IDEF2, a QUEUE is a node and an ACTIVITY is an arc). The arcs generally represent connectors between nodes. The node and arc linkages are automatically determined through proximity analysis.

If the user is at a nongraphical terminal, pages and symbols may be added using one of the other nongraphical GMB functions to ADD an occurrence to the appropriate entity class. If a symbol is added nongraphically, the user is not prompted for location; a (0,0) location is assumed. The user will be prompted for connecting start and end nodes for arcs in lieu of proximity analysis.

Attribute Input Options

The GMB provides the modeler a variety of options for responding to prompts for attribute values. In addition to providing a numeric or alphanumeric response, the user may enter any of a variety of special commands to discontinue processing or provide help (ABORT, END, HELP, EXPLAIN, STRUCTURE, ATTRIBUTE, and CLASS were previously described). See, for example, Figure 8.

```
SELECT ENTITY CLASS NAME > PRO
Current entity class is PROJECT
SELECT ACTION > ADD PROMPT

PROJECT NAME [No Default]<K> %ATTRIBUTE
*************************************************
*  CHARACTERISTIC          VALUE                *
*  --------------          -----                *
*  ATTRIBUTE NAME          PROJECT.NAME         *
*  DATA TYPE               CHARACTER LEN= 20    *
*  DEFAULT                 NONE AVAILABLE       *
*  SPECIAL CHECKING        NONE                 *
*     *** SPECIAL RESPONSE PERMISSION ***       *
*  CALCULATOR              NOT PERMITTED        *
*  DSL VARIABLE ACCESS     YES                  *
*  DEFERRED RESPONSE       YES                  *
*************************************************
PROJECT NAME [No Default]<K> ■
```

Figure 8. Attribute Description Option

Other options for responding to attribute prompts include algebraic expressions, deferred parameters, or database variable references. Algebraic expressions may contain constants, arithmetic operators, parentheses, standard functions as found in FORTRAN, references to variable values found in the database, or deferred parameters.

An example of a deferred parameter reference is illustrated in Figure 9. The question mark signals a deferred parameter. The value of this parameter is determined at application run time via a prompt to the user. Either the standard or a model-specific prompt may be issued. This facilitates parameterization of the analysis; multiple analyses could be run without rebuilding or editing the model.

```
Occurrence being proccessed.
PROJECT.NAME = CELL1
INVESTMENT.YEAR =          1
INVESTMENT.AMOUNT = ?
Deferred attribute = INVESTMENT.AMOUNT
=?
 ^
SPECIFICATION OPTION >HELP
Your options are:
     STANDARD - Standard prompt at resolve time
     TAILORED - User specified prompt at resolve time
SPECIFICATION OPTION > STANDARD
```

Figure 9. Deferred Parameter Input

Other Model Builder Functions

In addition to ADD, other functions are provided to manage entity class occurrences, such as EDIT, DELETE, LIST, and COPY. The EDIT function supports the update of entity class occurrences which already exist. For the QUICK and PROMPT options of the EDIT function, the user identifies a single entity class occurrence by specifying its key attributes. Prompts are then issued for new values for the non-key attributes. The current attribute value is indicated in the prompt. If a carriage return is entered to a non-key attribute prompt, the current value is unchanged. If a new value is entered, it replaces the old value. If a semicolon is entered, it terminates attribute prompting for this occurrence.

In TABLE mode EDIT, the user first builds an edit selection which specifies those entity class occurrences to be considered. The user may indicate a Boolean selection expression for any attribute value using relational operators EQ, NE, LT, LE, GT, and GE and the conditional operators AND, OR, NOT. For character attributes, the special symbols "*" (match on any zero or more characters) and "?" match on any one character) may be used.

Once the selection has been built, the table is loaded with those entity class occurrences which meet the specified selection criteria. Arrow keys may be used to position the cursor to the attribute values to be edited.

The DELETE function is used to remove entity class occurrences from the model. Descendants are automatically deleted. Descendents are occurrences of entities lower in the information structure hierarchy which inherit key attributes from the selected entity class occurrence.

The nongraphical LIST is initiated by building a LIST selection like that for the EDIT function. The selection is used to select the occurrences to be included in the LIST report. If the user requests that the entity class and all of its descendants are to be listed, an entity class join is performed. The report itself is in the form illustrated in Figure 10.

```
APPLICATION TECHNIQUE: NETSOL
MODEL NAME           : BMAC.CABLE.C
ENTITY CLASS         : MODEL
DATE                 : 8/6/84
```

ATTRIBUTE	1	2	3
MODEL.NAME	BMAC.CABLE.C	BMAC.CABLE.C	BMAC.CABLE.C
OPEN OR CLOSED	CLOSED	CLOSED	CLOSED
BRANCH OR VISIT	BRANCH	BRANCH	BRANCH
OUTPUT.REPORT	YES	YES	YES
DATABASE.STORAGE	YES	YES	YES
MODEL.DATE	8/2/84	8/2/84	8/2/84
DEVICE.NUMBER	1	1	2
DEVICE.NAME	SPOOL.TWST	SPOOL.TWST	16.BRAID
NUMBER.OF.SERVERS	2	2	1
SERVICE.DISCIPLINE	FCFS	FCFS	FCFS
DEVICE.SERVICE.TIME	300.	300.	332.
END.DEVICE.NUMBER		2	2
BRANCH.PROBABILIT		0.430	0.430
CLASS.NUMBER	2		
CLASS.NAME	16.SPOOL		

Figure 10. LIST function

The GRAPHICAL LIST creates a graphical display for the selected page. This function is available for graphically oriented applications only. However, if a user is at a nongraphical terminal, the user could initiate one of the nongraphical functions to observe, add, and edit the symbolic data for a graphical model in a tabular form.

The COPY function is used to copy one entity class occurrence (and all its descendants) within a given model. A source and destination entity class occurrence is identified by uniquely specifying a value for each key attribute. (In the model builder, all operations are limited to processing one model at a time; however, the model management functions available in the shell accomodates inter-model, inter-application, and inter-database data transfer).

DEVELOPMENT STATUS

At the time of this writing, IDSS Build 1 remains under development. The model builder has been coded and tested, though system testing will continue. It is planned that this system will be available for distribution by the end of the year.

IDSS Build 1 was developed under the Air Force Integrated Computer Aided Manufacturing (ICAM) Program. It is the policy of this program to distribute this and other software without charge to

American Industry. Wider distribution is being encouraged by the ICAM Program Office; it is hoped that this will improve the long-term competence, efficiency, and responsiveness of American Aerospace and related industries. Further details regarding this program may be obtained from AFWAL/MLTC, Wright-Patterson AFB, Ohio, 45433, or from Pritsker & Associates.

REFERENCES

[1] ICAM Program Office, ICAM Program Overview, PRO131020000, AFWAL/MLTC, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, November, 1983.

[2] Miner, et.al., "Decision Support for Manufacturing". 1981 Winter Simulation Conference Proceedings, pp 543-549.

[3] Pritsker and Associates, Inc., IDSS Build 1 System Requirements Document, SRD820540000, October, 1982.

[4] Pritsker and Associates, Inc., IDSS Build 1 Seventh Interim Technical Report, ITR82054007W, April 1984.

[5] Softech Inc., ICAM Architecture Part II, Volume V: Information Modeling Manual (IDEF1), AFWAL-TR-81-4023, June, 1981.

[6] Yancey, David P. and Miner, Robin J., "The Use of IDEF2 for Analyzing Material Handling Problems", 1981 Fall Industrial Engineering Conference Proceedings, pp 396-403.

[7] Yancey, David P. and Phillips, Don T., "Simulation Output Analysis within IDSS Prototype (2.0)". 1983 Annual Industrial Engineering Conference Proceedings, pp 471-480.