Edgar H. Sibley
Panel Editor

Although spreadsheet programs and microcomputers have revolutionized information processing in organizations, a significant number of serious errors have been reported through the misuse of this technology. This article discusses several different contexts for the development of spreadsheet models and presents structured design techniques for these models.

SPREADSHEET ANALYSIS AND DESIGN

BOAZ RONEN, MICHAEL A. PALLEY, and HENRY C. LUCAS, JR.

Spreadsheet programs have become extremely popular with microcomputer users. Spreadsheets are heavily used by both end users and systems professionals. In fact, spreadsheet software has contributed greatly to the popularity of personal computers. These packages present the user with a general purpose problem solving tool.

A spreadsheet can be viewed as a large matrix in which columns are typically designated by letters and rows by numbers. The intersection of a row and column defines a cell; a cell can contain a number, label, or formula that relates it to other cells in the spreadsheet. The ability to relate cells with formulae is what provides spreadsheets with their tremendous power. If the spreadsheet model is constructed with formulae, a change in one or two numbers is immediately reflected throughout the spreadsheet [9].

As an example, consider Figure 1 which is the printout of a simple spreadsheet model. Figure 1 is a salary calculation model; the decision maker enters an increase amount, and the percentage of the 1987 wage the increase represents is calculated in the third column. Total increases and the percentage of total 1987 compensation they represent are computed in the "Totals" row.

Suppose that the decision maker is told that wage increases in his department can average eight percent for 1987. This model allows the decision maker to enter different increases for individuals, observe the individual's percentage increase, and at the same time, view the impact of all increases on the bottom line represented by the totals. In this instance, the decision maker has stopped since a departmental aggregate wage increase of 8.03 percent has been reached.

Once this model has been constructed, the only input needed by the decision maker is to enter dollar amounts in the column labeled "Increase." The alternative to a spreadsheet model is pencil, paper, and calculator; each time a new salary increase is entered, the totals must be manually recomputed. One of the authors used to use a calculator for this process and developed the model in Figure 1 as his first spreadsheet application. He reports that the time needed to figure raises fell by a factor of at least five using the model.

Spreadsheets offer a tremendous amount of analytical power. Most spreadsheet users develop the models themselves. Another attraction of both microcomputers as well as this type of software is that the user is no longer dependent on the information services staff. In particular, for quick, infrequently used applications, such as the one in Figure 1, it would take far too long to create an application on a mainframe or mini through the information services department.

Overview

This article recommends a structured approach to the design of spreadsheet models. Topics covered include the differences between spreadsheets and traditional information processing applications. We then suggest that different types of design contexts influence the degree of structure appropriate in spreadsheet design.

It is recognized that one of the major advantages of spreadsheets is their ease of use. A decision maker can develop a model quickly without waiting for completion of the steps required during traditional systems analysis and design. Advocating more formal approaches to spreadsheet design may be viewed by some as a step backwards. Rather than advocate that the de-

Identification
  Owner: John Johnson
  Developer: Peter Green
  User: Betty Larson
  Date: Jan 30, 1987   Revised: Feb. 12 1987

File: Wages

Map of Model
  Identification
  Map
  Parameters and assumptions
  Model

Parameters and Assumptions
1. Current salaries (Input Vector) are given in second column. I
2. Salary increase (third column) is the Decision & the Parameter Vector. D
3. The recommended salary (fifth column) presents the Output Vector. O

*****SALARY ANALYSIS*****

Name	Current 1987	Increase	Percent	Recommend 1988
Jane Doe	$60,000	$4,000	6.67%	$64,000
Mary Roe	$50,000	$4,000	8.00%	$54,000
Sam Smith	$55,000	$3,000	5.45%	$58,000
Peter Jones	$70,000	$5,500	7.86%	$75,500
Harold White	$65,000	$4,000	6.15%	$69,000
Steve Black	$50,000	$5,000	10.00%	$55,000
Marvin Gardens	$45,000	$3,500	7.78%	$48,500
Betty Boop	$56,000	$7,200	12.86%	$63,200
Totals	$451,000	$36,200	8.03%	$487,200
	I	D		O

FIGURE 1. Salary Computation Model

signer follow *all* of its suggestions for *every* spreadsheet, it is important to determine which method to employ contingent on the nature of the design problem.

In particular, the argument here is that when a model is designed for use by someone other than the developer, more formal methods are needed, both to improve the design process itself and to reduce errors. On the other hand, some of the suggestions made in this article on format and style are appropriate for almost all types of spreadsheets.

DESIGN METHODOLOGY

The Problem

Spreadsheet packages and micros have extended computing to vast numbers of individuals. For many users, the spreadsheet program represents the first *hands on* experience with a computing device, programming, and documentation. In general, these users have not been trained in systems analysis and tend to overlook the concerns of the professional systems analyst in designing a system, such as reliability, auditability, and control. In fact, the spreadsheet user is often happy to avoid systems professionals.

Unfortunately, user independence comes at a potentially high cost. The practitioner literature has discussed a number of problems with spreadsheet construction [4, 8, 11]. Examples of frequently cited errors are mistakes in logic, incorrect ranges in formulae, incorrect cell references, confused range names, incorrectly copied formulae, incorrect use of formats and column widths, accidentally overwritten formulae, and misuse of built-in functions. Table I describes some of the problems found with spreadsheets.

Design Objectives

To minimize the probability and severity of the problems in Table I, the designer of a spreadsheet should be concerned with the following issues: 1) A spreadsheet should produce reliable results; the output it generates should be correct and consistent. 2) A spreadsheet should be capable of being audited; the user should be able to retrace the steps followed to generate different outputs from the model in order to understand

TABLE I. Problems Associated with the Lack of a Design Methodology

Problem	Description
Reliability	The degree that the spreadsheet generated output is correct–impact on degree of confidence user places in the model
Auditability	The ability to retrace the steps followed in the generation of spreadsheet results
Modifiability	The ability to change or enhance the spreadsheet to meet dynamic user requirements

the model and to verify findings. 3) A spreadsheet should be capable of being modified easily without introducing errors.

A final issue impacts the three listed above: comprehensibility. The designer and user should be able to easily understand the model and its assumptions as represented in the spreadsheet.

Spreadsheets and Traditional IS

Spreadsheets are a type of information system, though they are most often not developed by information systems professionals. The IS professional is (or should be) concerned with the issues described above in designing any information system. A few reasons for lengthy development times for multiuser systems designed by professionals are concerns about data integrity, input editing, and error checking.

Since the spreadsheet developer is often a user who is basically unfamiliar with the design principles above, ad hoc design is common. There tends to be little concern with formal analysis or documentation. Often the spreadsheet model is a one–time exercise or an infrequently used decision support system (DSS). Formal design methods slow progress on *quick* systems.

A structured approach to spreadsheet design can help reduce the probability and severity of problems with spreadsheets. To be useful, a spreadsheet design approach must help achieve the objectives for these models described above, must achieve results quickly, and must be suited to the style of the end user. Table II describes the characteristics of spreadsheet applications and the implications of these characteristics for design.

Because of the wide variety of use for spreadsheet models, it is very difficult to construct a design methodology that will be generally applicable. This article proposes a structured approach to design, the use of which is contingent on the type of model being developed.

SPREADSHEET ANALYSIS AND DESIGN

Types of Applications

Spreadsheets are distinguished by the variety of applications for which they have been used. The most frequent use of spreadsheets is for decision support and personal productivity, but there have also been many spreadsheet applications that could be considered mainstream information systems applications [2]. For example, a firm might do all of its financial statement consolidation for its subsidiaries using a spreadsheet package. Before this software was available, the firm might have written COBOL programs or purchased a dedicated package to produce financial statements.

Figure 2 depicts the design context for spreadsheets. If the spreadsheet is to be used for a traditional transactions processing or information processing task, then one could argue that it should be designed using normal information services methodologies such as structured design and data flow diagrams [10]. It is unlikely, however, that such a formal design will be used. The techniques presented next are alternatives to completely informal, ad hoc design for systems professionals as well as end users.

In the DSS environment, it is important to consider whether or not the developer is also the user. If the user develops his or her own model for a one-shot decision, then informal design procedures are probably satisfactory. It is also unlikely that anyone can convince the user in these circumstances to adopt any other approach. If the one-shot decision is crucial to the organization, it might be useful to at least have someone other than the designer audit the model.

If a user is developing his or her own application and plans to use it frequently, then the analysis approach suggested later in this article should be considered.

When the developer is not the primary user, it is recommended that a formal design methodology be employed. Even if the system is to be used for a one-shot decision, the spreadsheet being designed for someone else suggests that the decision is either very important or the model is complex; either condition warrants the use of a formal approach to design.

Applications that will be used frequently by many users are candidates for formal design, macros, and menus. If users are experienced and understand the

TABLE II. Characteristics of Spreadsheet Analysis and Design

Attribute	Spreadsheets	Implications
Users	Non professional users often are designers of spreadsheets	User does not realize problems with ad hoc approaches; assumes no need for a methodology
Development	Relatively Short	Attitude of no need or no time for a formal approach
Modifiability	Easy	May convince user to disregard formal analysis
Life cycle	Short	Formal techniques seen as useless
Context	Variety of situations	Difficult to build a methodology suitable to all contexts

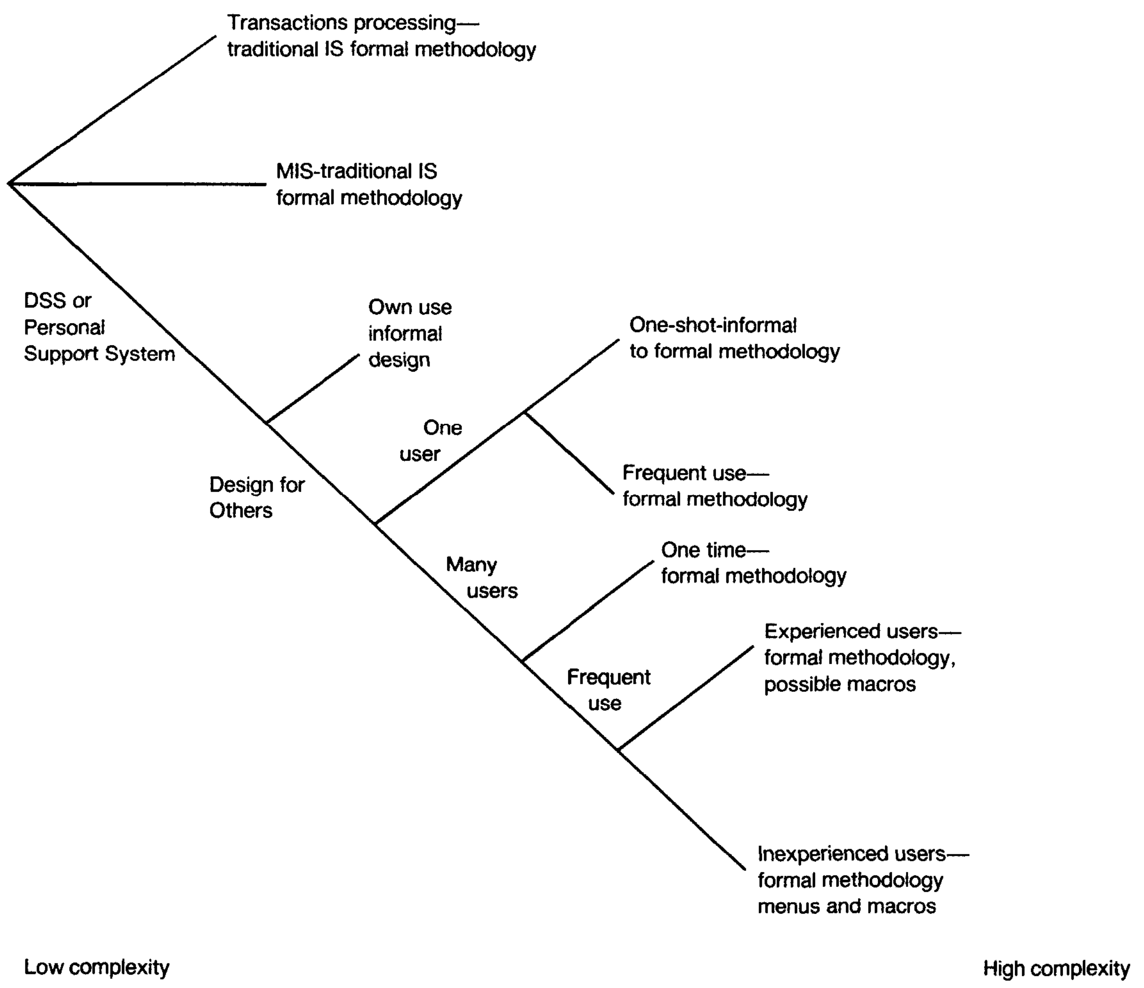
Transactions processing—traditional IS formal methodology
MIS-traditional IS formal methodology
DSS or Personal Support System
Own use informal design
Design for Others
One user
One-shot-informal to formal methodology
Frequent use—formal methodology
Many users
One time—formal methodology
Frequent use
Experienced users—formal methodology, possible macros
Inexperienced users—formal methodology menus and macros
Low complexity
High complexity

FIGURE 2. Different Design Contexts and Recommended Approaches for Spreadsheets

package, macros and menus may be an extra option. If users are inexperienced, however, menus and macros are needed to help the user execute the model and to protect the spreadsheet.

The Spreadsheet Development Life Cycle

Texts on systems analysis and design usually include a systems development life cycle [1, 10]. It is possible to develop a similar life cycle for spreadsheet applications, though the life cycle must be flexible to reflect the different contexts for spreadsheet design described above as shown in Figure 3.

Problem Identification. The designer defines the nature of the problem to be solved. How is the problem currently solved if at all? What are the performance bottlenecks? This stage is similar to the "existing systems study" of the systems development life cycle. How will a spreadsheet model help solve the problem? What are the sources of information?

a. Cost benefit analysis. What is a reasonable time frame for model development? What kind of application is this? (e.g., is it a one-shot decision model or a model that will be used many times?) What resources should be put into development?
b. "Make or buy" analysis. The designer should conduct research to determine if an existing template can be purchased for this application. There are a number of such templates for income tax calculations, rental analysis, real estate investments, etc.

Definition of model outcome/decision variables. The spreadsheet is usually developed to produce results such as a net present value for an investment, a yearly estimate of profits, the deviation of actual from budgeted expenses, or a set of financial statements. The outcome variable needs to be defined. The designer may have variables that represent a decision such as the dollar raises granted to each employee in Figure 1.

a. Define how the outcome is generated. This part of the model represents the calculations which are undertaken in the model.

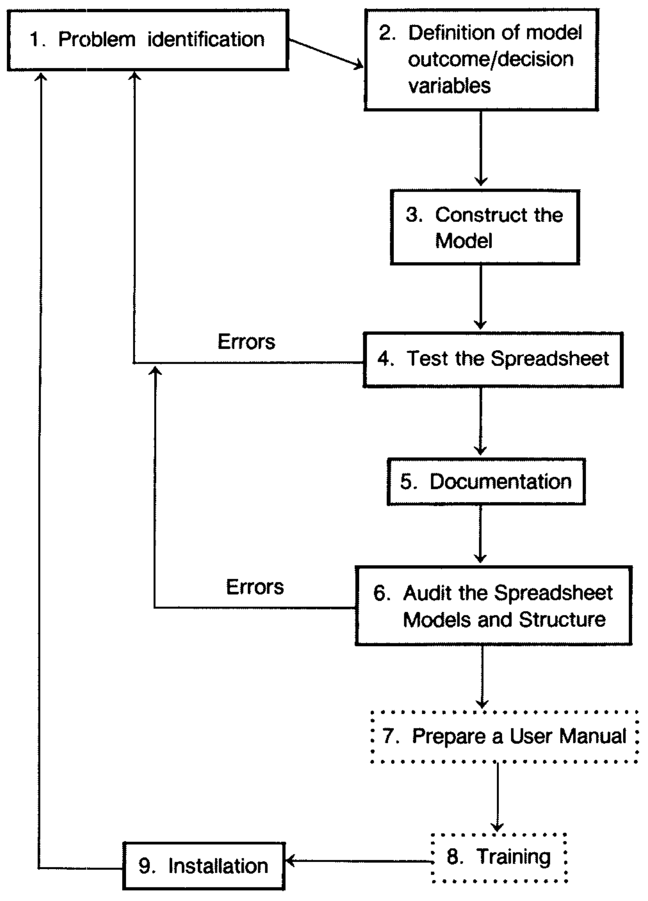

FIGURE 3. The Spreadsheet Development Life Cycle

b. Define the block structure of the model. A recommended model structure is presented later in this article.
c. Define menus/macros (optional). If the model will have menus and macros, the designer should describe their function at this point.

Construct the model. This stage corresponds to the traditional notion of programming. Using the various commands of the spreadsheet language, build the model. When the model is large, use a top down approach in which building blocks are constructed and then filled in with details.

a. Build macros and menus (optional). Depending on the type of applications, the developer may want to include menus and macros.

Test. Carefully test the results of the model. Print a hard copy of the model and the cell formulae. Check all calculations independently from the spreadsheet. Test both historical and extreme input data. Examine the spreadsheet to see if there is an audit trail; can someone follow through the assumptions to determine how a cell's value was determined, or is too much hidden in formulae? Look for formatting errors that might result in the unintentional rounding of percentages or multipliers.

Documentation. Document the spreadsheet on the spreadsheet itself. That is, include text on the spreadsheet that explains the model as shown later in the paper.

Audit. Review the model and its structure. Consider the use of audit packages to trace through formulae.

Prepare a user manual (optional). For systems designed for others to use, a manual is a necessity. For applications created by the user, a manual is valuable if the application is to be used more than once.

Training (optional). If the model is to be used by others, they may need to be trained prior to installation.

Installation. Prepare the spreadsheet for use, for example, by installing it on a user's computer so that the model loads whenever the spreadsheet program is started.

Model Format

Figure 4 presents a recommended structure for a spreadsheet. Similar designs have been proposed in [3, 7]. The purpose of the structure is to separate parts of a spreadsheet into blocks to reduce the potential for errors. A well structured spreadsheet also clarifies the assumptions of the model to users.

Figure 4 contains a number of blocks which, when taken together, comprise the spreadsheet model. The identification block presents the name of the developer, user, and model. It also contains a list of revision dates and the name of the spreadsheet file.

To the right of the identification block one finds the macros/menus block. Macros and menus must be iso-

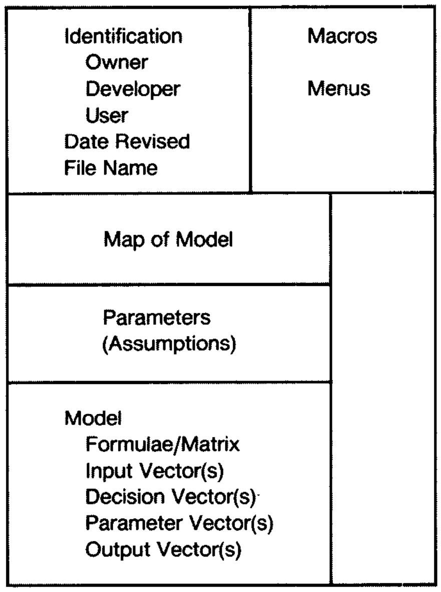

FIGURE 4. Recommended Spreadsheet Structure

lated from parameters and formulae because the insertion or deletion of a row in a model could delete a line or insert a blank line in a macro causing quite anomalous behavior when the macro is executed.

Immediately below the identification block is a map or index to the spreadsheet. It contains a description of where the various blocks may be found and acts as a table of contents for the model.

The large documentation block allows the spreadsheet developer to describe in general terms how the model works and to annotate various rows in the model.

The parameter block contains variables that are used in the formulae. For example, one would place interest rates, assumptions about sales growth rates, and so on in the parameter block. A good rule to follow is that no formula should contain a number; there should only be references to parameters or cells in the worksheet. If this rule is followed, there is little danger of misinterpretation because a crucial parameter is hidden in a formula. Looking over multiple runs of the model, the user will be able to see what assumptions pertain to each run. For example, the interest rate used on each model run will be clear by examining the parameter block for the interest rate parameter.

The final block in the spreadsheet is the model itself. The spreadsheet packages with their row and column references suggest a view of the model as a matrix. Certain columns or rows of the matrix (possibly a single cell) might be interpreted as input or output vectors.

Spreadsheet Flow Diagrams

The notion of Data Flow Diagrams (DFD) has proven popular in traditional systems analysis and design as a way to encourage structured, top-down design and to reduce complexity [5, 6]. This article proposes the use of Spreadsheet Flow Diagrams (SFD) for the same purpose with spreadsheets. For many spreadsheets there will be no need to follow a top-down approach because the model will be simple. For large spreadsheets, however, the notation allows and encourages the use of top-down design approaches.

Why not just use existing DFD notation? DFDs were designed for transactions processing systems; they are excellent for showing sources, flows, and processing of data. The spreadsheet is less concerned with the flow of data than it is with modeling relationships. Even a low level DFD might only indicate Post Receivables to denote a process that most designers and users would understand. Spreadsheet analysis and design needs a notation that shows the *algorithm* or the underlying formulae of the model.

Figure 5 shows the basic symbols of SFDs. A simple rectangle is used to represent input vectors, output vectors, decision vectors, and parameters. (Any of these vectors can be a scalar if it contains only one value.) The type of vector is designated by a 45 degree line on one corner of the rectangle as shown in Figure 5. A rounded rectangle represents the formulae in the model.

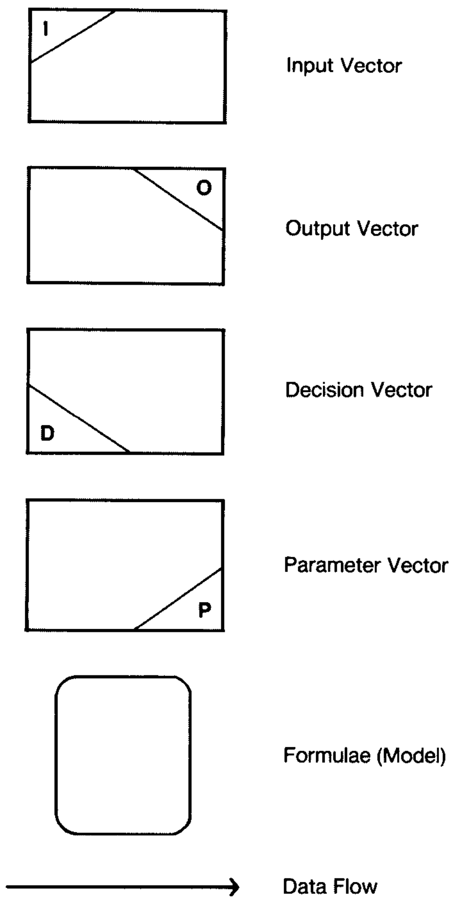

FIGURE 5. Spreadsheet Flow Diagram Symbols

It should be noted that the input, decision, and outcome vectors are subsets of the model matrix in most instances. Some examples will show these unique characteristics of spreadsheet models and will also demonstrate how the SFDs can be used in design and documentation.

In Figure 1, the input, decision, and outcome vectors are highlighted. They constitute three of the four columns in the matrix that is the model. An SFD for the salary increase model in Figure 1 may be found in Figure 6a. The SFD shows clearly that the increase is the decision variable and that the outcome is to be a salary figure for 1988. While this decision vector may seem obvious, approximately 50 percent of the students assigned this problem ended up using the percentages as the decision vector and awarding raises like $2432.

Figure 7 presents a typical spreadsheet application for predicting sales of a new product, and Figure 6b is a high level SFD for the model. The input vector has a subscript *t* to show that the model is predicting values over a time period. Because it is easier to understand, the model diagram shows profits as being sales less expenses. The model box (labeled 1) in Figure 6b is exploded into a lower level of detail in Figure 6c.

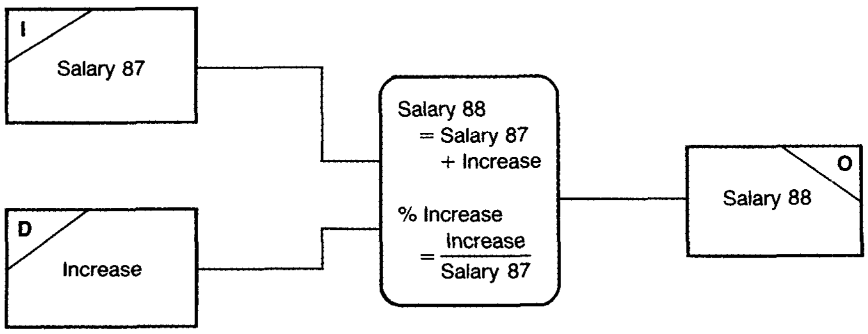

FIGURE 6a. SFD Problem 1

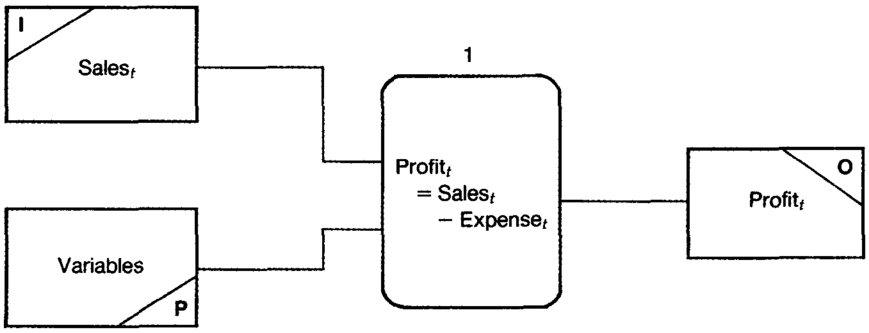

FIGURE 6b. SFD Problem 2

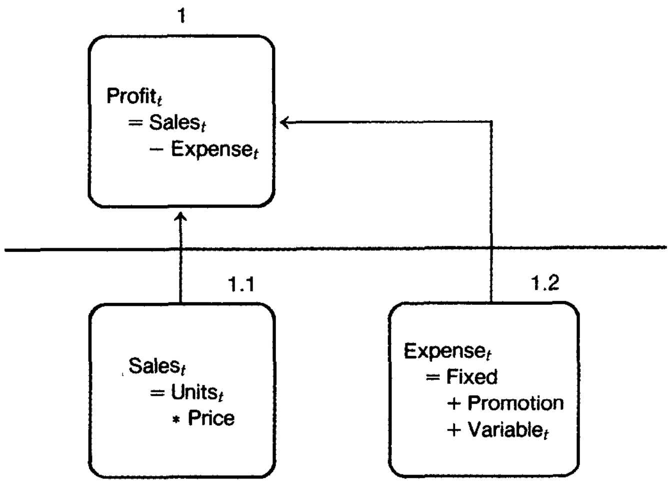

FIGURE 6c. Level 2 SFD for Problem 2

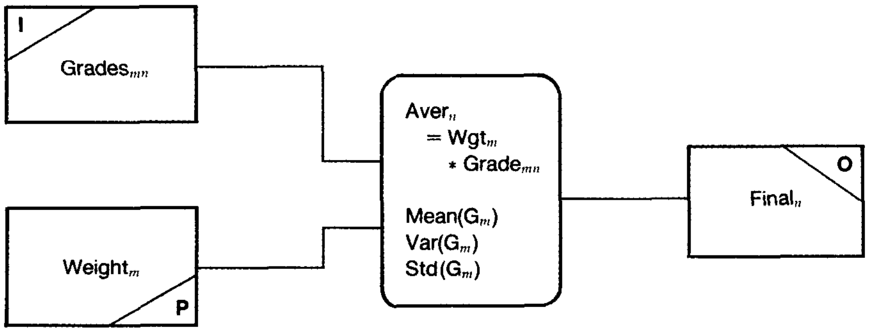

FIGURE 6d. SFD Problem 3

ALL PURPOSE WORKSTATION SOFTWARE CO.

Identification
 Owner: All Purpose Workstation Software Co.
 Developer: Anne Smith
 User: Jerry Brown
 Date: 2/1/87 Revised: 4/1/87

File: Forecast

Map of Model
 Identification
 Map of Model
 Parameters and assumptions
 Variables
 Model

Assumptions
1. Sales (in units) for 1988 is the Input Vector.
2. The Output Vector is the Net Profit and Net Present Value.
3. The Model Parameters are presented in the Parameters Box.
4. Sales Growth is considered the same in each year.
5. The interest rate (cost of capital) is fixed during all the period.
6. Variable costs are proportionally linear to the number of units sold.
7. Tax rate is flat for all profits.

Parameters

Sales Growth	45%	
Interest	12%	
Price	350	P
Variable costs	75	
Tax rate	35%	
Assets deprec	500,000	

The Model

NEW PRODUCT PROJECTIONS*

	1988		1989	1990
Sales (units)	5,000	I	7,250	10,513
Revenue	$1,750,000		$2,537,500	$3,679,375
Variable cost	$375,000		$543,750	$788,438
Fixed cost	$900,000		$900,000	$900,000
Promotion	$1,000,000		$1,000,000	$1,000,000
Gross Profit	($525,000)		$93,750	$990,938
Depreciation	$25,000		$25,000	$25,000
Taxes	$0		$24,063	$338,078
Net Profit	($550,000)		$44,688	$627,859
			NPV $540,425	O

FIGURE 7. New Product Projections Computation Model

Figure 8 (see next page) contains the final example, a grading program; the corresponding SFD may be found in Figure 6d. Here there are *m* students with *n* grades. The model weights the grades to provide a final average and grade for each student. The instructor also is interested in the mean, variance, and standard deviation of the scores on each assignment. The subscripts *m* and *n* used in the SFD clarify the requirements of the model.

The advantage of using a structured notation for spreadsheets is the same as the advantage of using a notation such as DFDs for traditional systems analysis and design. SFDs help the designer structure the design of the problem. They assist in communicating the structure of a model to others and they serve as documentation when it is necessary to audit or modify the spreadsheet.

Menu Design

Many popular spreadsheet packages allow the model developer to define macros (programs consisting of keystrokes) and menus. Figure 9 is an example of a state-transition diagram for a menu for the grading program in Figure 8. Each keystroke selection from the main menu moves the user to a state represented by one of the circles. In Figure 9, the menu appears as the spreadsheet is loaded. The user sees a menu that contains the following commands: ADD, DELETE, MODIFY, PRINT, SAVE, and QUIT. Typing the letter A results in the execution of a spreadsheet macro to add a student to the roster.

If there were submenus reached from this first level, they could be shown in a similar fashion. For example, suppose that *modify* allows the user to choose what is to be modified, a name or a grade. Entering *N* results in positioning the cursor in the name column while a grade change places the cursor at the first assignment from which the user can move it to any assignment for any student. The designer has made a decision to return to the modify menu at the completion of a change in anticipation of another change. Alternatively, the designer could choose to return to the main menu.

Another alternative to state-transition diagrams is the menu trees often printed for menu-driven packages. These menu trees look like organization charts and communicate the same information found in Figure 9. They are less clear about the return point from each command issued than the state-transition diagram though.

SUMMARY

This article has argued for greater structure and care in the design of spreadsheet information systems. It first explored some of the problems with spreadsheet design and the differences between the design of traditional information processing applications and spreadsheet models.

The major contributions of this article are the following:

Identification
Owner: Jo Little
Developer: Harry Brown
User: Dr. Dolittle
Date: 2/3/87 Revised: 3/3/87

File: Grade F88

Map of the Spreadsheet
Identification | Macros
Map |
Assumptions
Parameters
Model

Assumptions and Parameters
1. Each assignment is weight as per Table 1.
2. Grades are assigned per Table 2.
To use the Model:
Alt M starts the macro execution and is the name of XM
BEGIN is the first student name
START is a range name for the menu
GRADE is a range name for the grading table

TABLE 1: WEIGHTS

Assignment 1	25%	
Assignment 2	35%	P
Assignment 3	40%	

TABLE 2: GRADES

	−1	F
Below	0.01	D
1 std below	2.20	C
Mean	2.90	B
1 std above	3.60	A
	4.00	A

The Model

*****Student Grading Program*****

Name		Assign 1	Assign 2	Assign 3	Course	Final	
					0.00		
Boop, Betty	I	2	2	2.3	2.12	D	
Caroline, Princess		2	1.7	3.3	2.42	C	
Carson, Kit		3.7	3.3	4	3.68	A	
Doe, John		2	2.3	2.7	2.39	C	
Dupris, Jean		3	2	3.3	2.77	C	
Field, Marshall		4	3.7	3.3	3.62	A	
Jones, Betty		3	4	3.7	3.63	A	
McDonald, Ronald		3	2	4	3.05	B	
Newman, Alfred		4	4	3.7	3.88	A	
Pan, Peter		3.7	3.7	3.7	3.70	A	O
Roe, Mary		1.7	2	2.3	2.05	D	
Rogers, Roy		2.7	3.3	3.7	3.31	B	
Smith, Sam		2	3.3	2.7	2.74	C	
Square, Harold		1	1	2.3	1.52	D	
VanPelt, Lucy		3	2.7	2.3	2.62	C	
Mean		2.72	2.73	3.15	2.90		
Variance		0.77	0.83	0.40	0.49	O	
Standard Deviat		0.88	0.91	0.63	0.70		

FIGURE 8. Student Grading Computation Model

(1) Guidelines for differing levels of formal design and the appropriate use of menus and macros in spreadsheet design.
(2) A suggested systems development life cycle for spreadsheets.
(3) A block-structured format for spreadsheet models.
(4) Spreadsheet Flow Diagrams.
(5) Menu and macro design aids.

Spreadsheet packages have made a major contribution to analysis and problem solving. The importance of the decisions entrusted to spreadsheet modeling and the pervasive use of these packages suggests that end users and developers of these models need to be concerned with good practice in spreadsheet analysis and design.

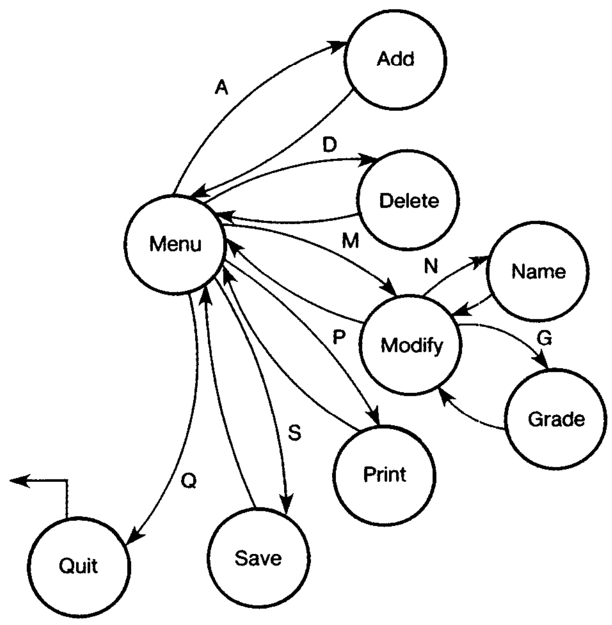

FIGURE 9. State-Transition Diagram for Menu Design

REFERENCES
1. Ahituv, N., and Neumann, S. *Information Systems for Management.* 2d ed. Wm. C. Browne, Dubuque, Iowa, 1985.
2. Ariav, G., and Ginsberg, M. DSS design: A systematic view of decision support. *Commun. ACM 28,* 10 (Oct. 1985), 1045–1052.
3. Berry, T. How to structure spreadsheets. *Bus. Softw.* (Oct. 1986), 56–58.
4. Bryan, M. Are your spreadsheets lying to you? *Bus. Softw.* (Oct. 1986), 59–64.
5. Demarco, T. *Structured Analysis and System Specification.* Prentice-Hall, Englewood Cliffs, N.J., 1979.
6. Gane, C., and Sarson, T. *Structured Systems Analysis.* Prentice-Hall, Englewood Cliffs, N.J., 1979.
7. Grupe, F. Tips for better worksheet documentation. *Lotus* (Aug. 1985), 68–70.
8. Grushcow, J. Avoid these common spreadsheet errors. *Lotus* (July 1985), 59–62.
9. LeBlond, G., and Cobb, D. *Using 1-2-3.* 2d ed. Que Corporation, Ind., 1986.
10. Lucas, H.C., Jr. *The Analysis, Design, and Implementation of Information Systems.* 3d ed. McGraw-Hill, New York, 1985.
11. McGrath, J. PC's can't add—but you can teach them to. *Bus. Softw.* (Sept. 1986), 58–59.

CR Categories and Subject Descriptors: D.1.m [**Programming Techniques**]: Miscellaneous; H.1.m [**Models and Principles**]: Miscellaneous; H.4.2 [**Information Systems Applications**]: Types of Systems; K.6.1 [**Management of Computing and Information**]: Project and People Management; K.6.3 [**Management of Computing and Information Systems**]: Software Management
General Terms: Design, Documentation, Management
Additional Key Words and Phrases: Documentation, reliability, spreadsheets, systems analysis and design

ABOUT THE AUTHORS:

BOAZ RONEN is a visiting assistant professor of information systems at New York University's Graduate School of Business Administration. His research interests include end user computing, decision support systems, and MIS in operations management. He formerly worked for more than 10 years in the Israeli hi-tech industry as a project manager. Author's present address: Leon Recanati Graduate School of Business Administration, Tel Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel.

MICHAEL A. PALLEY is an assistant professor of computer information systems in the School of Business Administration at Baruch College, City University of New York. His research interests include management information systems, systems analysis and design, and statistical databases. Author's present address: Baruch College, CUNY, School of Business and Public Administration, 17 Lexington Avenue, Box 513, New York, NY 10010.

HENRY C. LUCAS, JR. is research professor of information systems in the Leonard N. Stern School of Business at New York University. He is the author of several books including *Managing Information Science.* His research interests include implementation, systems analysis, expert systems, and the management of information processing. Author's present address: Leonard N. Stern School of Business, New York University, 100 Trinity Place, New York, NY 10006.