Woman C.P.A.

Volume 51 | Issue 3 Article 6

7-1989

Bittersweet Spreadsheets: Application Development Needs Control

Ernst Goss

Tom Dillon

Jackie Kendrick

Elise G. Jancura

Follow this and additional works at: https://egrove.olemiss.edu/wcpa

 Part of the Accounting Commons, and the Women's Studies Commons

Recommended Citation

Goss, Ernst; Dillon, Tom; Kendrick, Jackie; and Jancura, Elise G. (1989) "Bittersweet Spreadsheets: Application Development Needs Control," *Woman C.P.A.*: Vol. 51 : Iss. 3 , Article 6.
Available at: https://egrove.olemiss.edu/wcpa/vol51/iss3/6

This Article is brought to you for free and open access by the Archival Digital Accounting Collection at eGrove. It has been accepted for inclusion in Woman C.P.A. by an authorized editor of eGrove. For more information, please contact egrove@olemiss.edu.

Bittersweet Spreadsheets

Application Development Needs Control

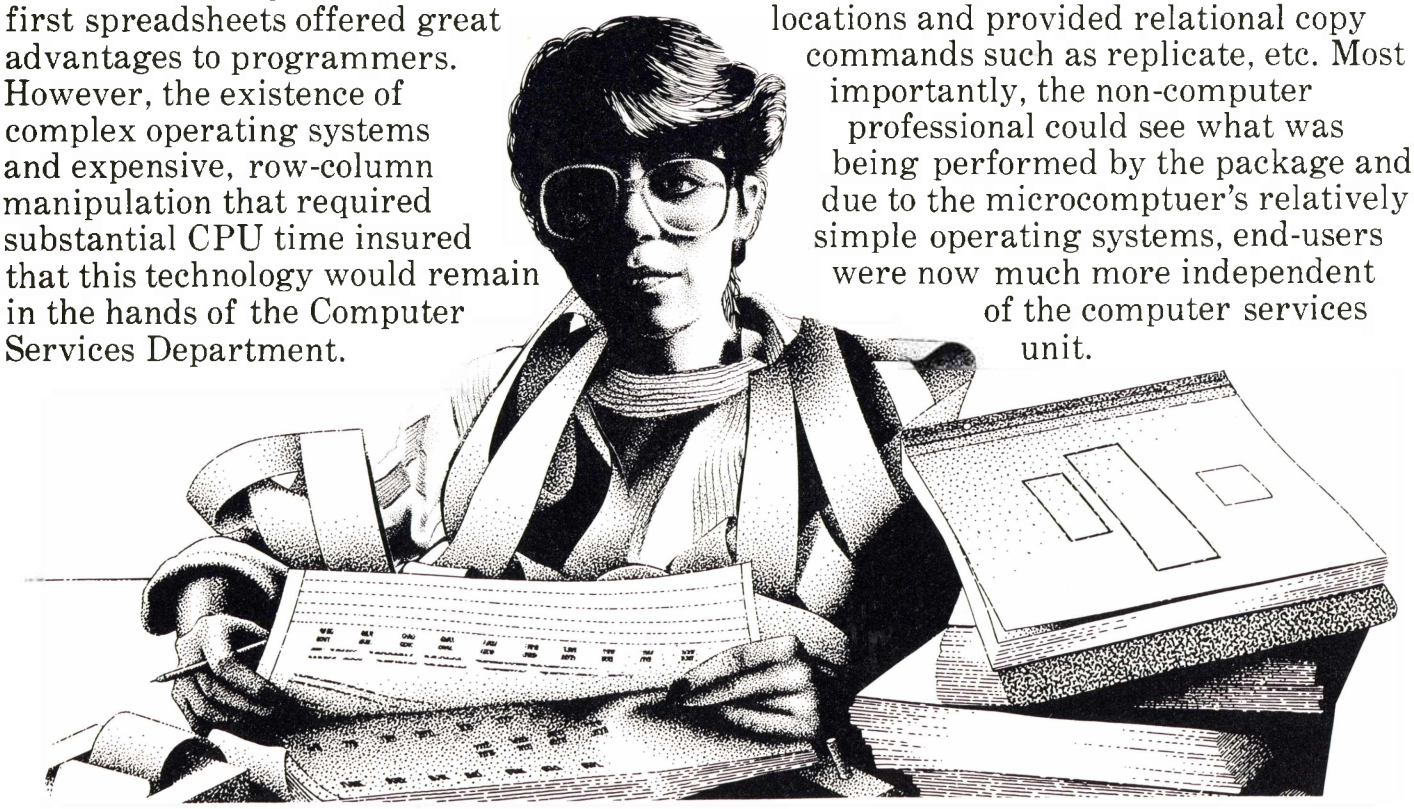
By Ernst Goss, Tom Dillon, and Jackie Kendrick

Editor: Elise G. Jancura, Cleveland State University, Cleveland, OH

Background

The first electronic spreadsheets were developed for large mainframe and/or minicomputers in the late 1970's. Developed for the time-sharing environment that became popular when row-column manipulation became available on computer systems, these first spreadsheets offered great advantages to programmers. However, the existence of complex operating systems and expensive, row-column manipulation that required substantial CPU time insured that this technology would remain in the hands of the Computer Services Department.

In 1979, the VisiCorp Company introduced VisiCalc, a microcomputer-based electronic spreadsheet. Due to its true "user friendliness" or window-screen communications, this package quickly became the *de facto* standard among non-computer professionals. VisiCalc allowed formulas to be placed directly in cell locations and provided relational copy commands such as replicate, etc. Most importantly, the non-computer professional could see what was being performed by the package and due to the microcomptuer's relatively simple operating systems, end-users were now much more independent of the computer services unit.



In 1982, the Lotus Corporation introduced the second generation of spreadsheet packages. Their product, Lotus 1-2-3, quickly replaced VisiCalc as the industry standard. Today, spreadsheet applications developed using Lotus 1-2-3 and other packages are the most popular type of applications in businesses. Spreadsheet applications have moved beyond mere financial analysis and have assumed importance as decision support tools. In a recent survey, Aggarwal and Oblak [1987] found spreadsheet packages to be the most important type of software employed for strategic decision making.

In view of the way that spreadsheet applications are now being put to use, it seems contradictory that the majority of the spreadsheet applications are developed by the end-user with little input from the computer professional or with little oversight from the manager. In many cases, from the assignment of the task until completion, a managerial oversight is virtually absent from the process [Kee and Mason, 1988]. This lack of managerial control presents several problems. Perhaps most importantly, the integrity of the application is endangered. With no consistent methodology for construction of the application, errors are much more likely to occur. It has been estimated that fully one-third of all spreadsheet applications contain errors [Creeth, 1985]. When errors are made or slight modifications incorporated, problems are compounded and troubleshooting becomes much more difficult. For the public accounting firm, error-laden or poorly designed applications can prompt malpractice suits [Yoder, 1986]. For the private firm, a lack of integrity in spreadsheets can result in poor decision making and suboptimal performance [Business Week, 1984].

The change in the focus of application development from the computer professional to the user has brought about a need to formalize techniques to insure the integrity of spreadsheets. This article presents a step-by-step procedure for the development of spreadsheet applications and provides a methodology for the manager to better control the development process so that efficiency and integrity are maximized.

Systems Development Life Cycle

For the manager who lacks expertise in structured design techniques, the Systems Development Life Cycle (SDLC) provides a basis for managerial control since the steps define the segments of the flow of work and specify the documents or deliverables to be constructed in each phase.

The number of steps in the various SDLC processes varies from seven to ten. Due to its wide adoption, the systems development life cycle approach suggested by Whitten, Bentley and Ho [1986, p. 142] is the approach recommended here. The process consists of the following seven steps:

1. Survey the Situation.
2. Analyze Specific Needs and Requirements.
3. Gather Available Information.
4. Logically Design on Paper the Application.
5. Construct the Spreadsheet Template or Model.
6. Test the Template or Model with Known-Output Data.
7. Implement the Spreadsheet.

Seven Step SDLC Approach to Spreadsheet Applications

Step 1. Survey the Situation.

The first decision the designer should make is how the problem is to be solved. The user/designer, along with the manager, should determine if a spreadsheet application is the best vehicle to attack the problem. Other alternatives such as database applications should be examined since each type of package provides its own set of advantages and disadvantages. For example, database packages provide a variety of input forms and output reports capabilities while spreadsheet packages are limited basically to the format that appears on the screen.

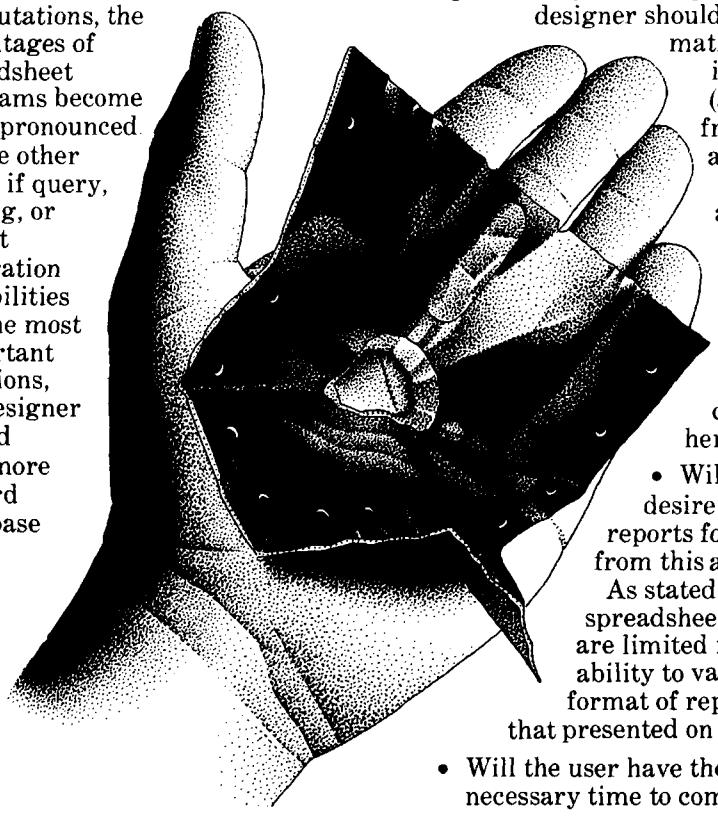
The key to selecting a spreadsheet application or some other type of software package centers on the visual appearance or the window-screen display. If an application will benefit from window-screen display, the software selection decision should lean toward the use of a spreadsheet package. Additionally, if the application calls for extensive computations, the advantages of spreadsheet programs become more pronounced. On the other hand, if query, sorting, or report generation capabilities are the most important functions, the designer should lean more toward database packages. Moreover, spreadsheet programs assume that the user's entire file will fit into memory. For this reason, spreadsheets are unable to support very large data sets.

During this step, the manager should meet with the user/designer along with an individual who is familiar with other types of packages. There may be, for example, an off-the-shelf application that can be purchased and that can accomplish the job at a lower cost and with greater integrity.

This is an important step in the development process, particularly for large jobs, and requires a good deal of involvement on the part of the manager. Unfortunately, this step is often omitted or given slight attention.

Step 2. Analyze Specific Needs and Requirements.

Is the needed data available at a reasonable cost? Will this data be available for subsequent use? Oftentimes, the user develops a very worthwhile application only to find that the data is provided sporadically or infrequently. The designer should produce a matrix of data inputs (along with frequency of availability) matched against the sources of the data. Other factors that should be considered here include:



- Will the user desire to prepare reports for outsiders from this application? As stated earlier, spreadsheet packages are limited in their ability to vary the format of reports from that presented on the screen.
- Will the user have the necessary time to complete the project? Here the manager should rely on experience and/or advice from others.
- Will graphics be needed? Certain spreadsheet packages have very poor graphics capability relative to other packages, such as statistical packages.
- Will the user need to integrate document preparation into the application?

Step 3. Gather Available Information.

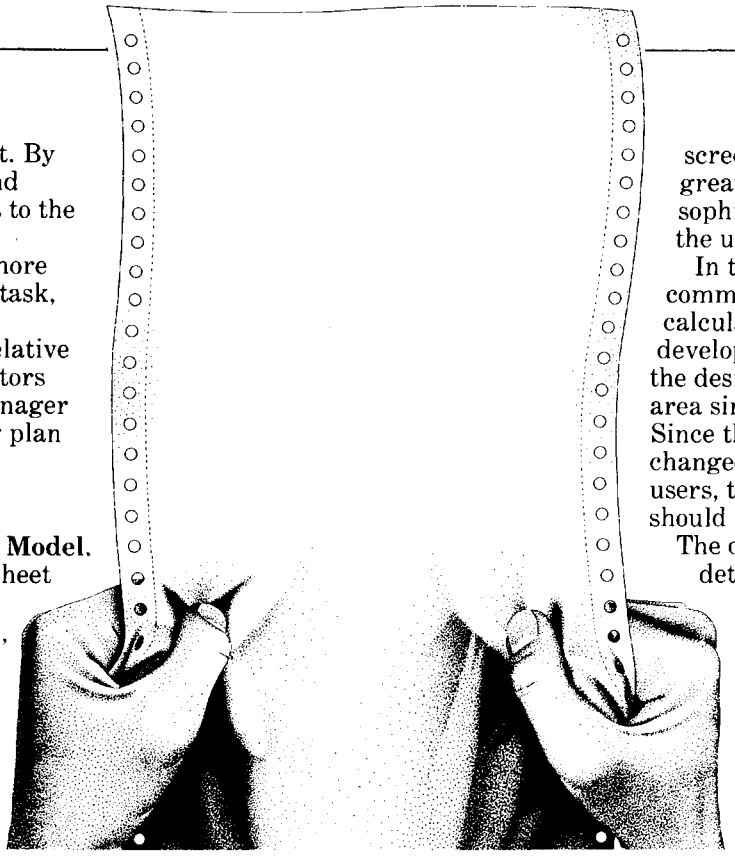
Prior to any hands-on computer work, the designer should define and list variable names, constant values, and equations. (The use of mnemonic names is encouraged since they are self-documenting when applied to file names, columns of data, or spreadsheet ranges.) In this step, it is important to consult with those who are most familiar with the input data since the designer must quantify all information for the purpose of estimating the size of the spreadsheet application. The designer must determine number size or column width. The designer must then estimate the number of rows, columns, and files that will likely be involved and combine this information with space available on floppy disks or hard disks in order to determine if space will be a problem. Often, due to security and portability demands, it is necessary that the application fit on floppy disks. This potential limitation must be given early consideration.

Step 4. Logically Design the Application.

The spreadsheet should be designed systematically in modules. Planning the spreadsheet in modules will make the spreadsheet easier to update if changes must be made or if re-creation is necessary. Also, modular design can build in multiple uses for multiple applications and allow several individuals to work on the same spreadsheet application at the same time.

For example, if graphs will be produced, one individual would be assigned the task of writing the commands, etc. necessary to

produce the required output. By modularizing the project and adding additional members to the project, internal control is enhanced. Of course, with more individuals working on the task, the problem of coordination becomes more acute. The relative importance of these two factors must be weighed by the manager and an appropriate staffing plan adopted.



Step 5. Construct the Spreadsheet Template or Model.

This phase of the spreadsheet SDLC calls for detailed documentation. In this step, the approach suggested by Bromley [1985] is recommended. The approach, as described by Bromley, calls for the application to be divided into the four divisions of the COBOL structure — Identification, Environment, Data, and Procedure. When applied to spreadsheets, the first three divisions, respectively, meet most documentation needs.

Identification Division. The Identification division lists the important attributes of the spreadsheet. The designer should include the following items in this section:

a. A heading.
b. Date of original design.
c. Date of last update of the application.
d. Version number, if there is more than one version.
e. Author's name, address and phone number for questions from other users.
f. Department, company, or installation name.
g. Security information (internal use only or internal/external use).
h. Table of contents with GO TO locations.
i. Problem domain of the spreadsheet and how it works.

In order to be found easily, the Identification division should be located in the HOME key area or cells A1 through H20.

Environment Division. This division should be sub-divided into a Configuration section and an Input-Output section. The Configuration section documents the type of system used in the design of the spreadsheet and describes any special names defined for variables and data value. The Input-Output section describes any special input or output information. For example, are graphs saved with the spreadsheet, or is sideways print necessary?

Data Division. Field or cell definitions should be placed in this division. Important information such as cell size or width, cell format, specific range names, and any other cell data is listed. The Data Division should also include information about other spreadsheets that will be used in conjunction with the current application such as file names, data descriptions, and the dates that other spreadsheet files used by the application were last updated.

Procedure Division. This division should be sub-divided into three sections: Input, Processing, and Output. In terms of the Input section, the developer must decide on the design of the input and help screens. This design will depend greatly on the technical sophistication and preferences of the ultimate user or users.

In the Processing section, macro commands and intermediate calculations should be stored. The developer must be meticulous in the design and construction of this area since most errors occur here. Since this area should not be changed, except by authorized users, the cell protect command should be invoked.

The destination of the output determines the design of the Output area. If the application provides output to the screen, an attractive screen should be designed as an output area. Or if the application produces output to the printer, the output should be designed as a printed report. Consideration should be given here for such options as reduced print, sideways print, and laser-print options.

Step 6. Test the Template or Model.

The test for spreadsheet applications provided by Simkin [1987] is a good one to use. First a visual inspection should be performed. Second, using prior years' or other known-output data, test to determine the accuracy of computations. If the spreadsheet application is very large, the designer should set the calculation mode of the spreadsheet to manual. This will increase data entry speed and computer response time. Third, copy the digit "1" into all of the cells in the range and use the summation function to determine if all of the cells are in the defined range. Fourth, multiply the number of columns by the number of rows and compare the calculation to the one obtained in the third test. Fifth, check output data or forecasts with graphs since this will sometimes result in the detection of missing data. Sixth, if time permits, develop the same application on an alternative package and compare the results. And seventh, if spreadsheet audit programs are available, perform

Planning the spreadsheet in modules will make the spreadsheet easier to update if changes must be made or if recreation is necessary.

the audit checks provided.

To insure that proper testing procedures have been followed, a test check sheet should be prepared and included with the design documentation. Keeping this record will encourage testing by template developers. Once testing has been completed, a backup copy of the spreadsheet application should be made and filed with the manager.

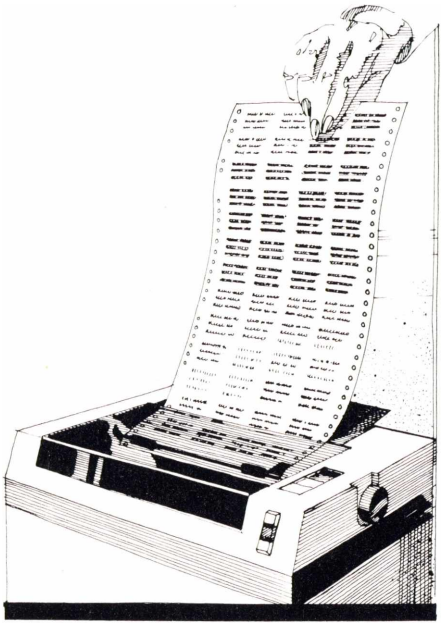
Step 7. Implement the Spreadsheet.

The application is now ready to bring on line. If the application is now being performed manually, the manual operation and the new application should be run parallel until the user/developer is assured that the spreadsheet application is performing as expected.

Each of the preceding steps should be documented properly with a paper trail to provide audit capability at a later date. Each step should be checked by a second person of equal or superior spreadsheet skill. Test steps should be performed with data that has a predetermined outcome so that final values can be checked for accuracy. Before actual implementation, the final product should be tested by the personnel that will be using the application to insure that it is truly useful in its present form. Prior to initiation of step 1, the manager should establish a method for feedback from the designer and provide a time budget for the completion of each step.

Conclusion

The approach recommended in this article provides a basis for managerial control since the steps define the segments of the flow of work and specify the documents or deliverables to be constructed in each phase. Nevertheless, the recommended approach does not *guarantee* the integrity of spreadsheet applications. However, it will insure greater consistency and integrity in the construction of spreadsheet applications and, if adopted, will significantly reduce errors and will provide the manager with a basis for control.



Ernst Goss, Ph.D., *is Associate Professor of Management Information Systems at the University of Southern Mississippi.*

Jackie Kendrick *is a partner in the firm of Business and Technical Services, a firm specializing in accounting, business and computer application services. She is the author of* ***Hands-On Database with Rbase.***

Thomas Dillon, M.S., *is a computer science faculty member in the Richard A. Henson School of Science at Salisbury State University in Salisbury, Maryland. Dillon also serves as a microcomputer consultant with MicroData Inc.*

Classifieds

Opportunities at all levels in Public Accounting ("Big 8's," Nationals, Top Locals) and positions in industry, nationwide. Resumes or call: BEN GRECO, Executive Search, 445 S. Figueroa Street, Ste. 2600, L.A., CA 90071, (213) 612-7766.

References

Aggarwal, A. K., and M. D. Oblak, "A Survey of Business Usage of Decision Support System Software," *Proceedings of the 1987 Annual Meeting of the Decision Sciences Institute*, Boston, MA (December 1987), pp. 205-207.

Amsterdam, J., "Build a Spreadsheet Program," *Byte* (July 1986), pp. 97-105.

Bissell, J. L., "Spreadsheet Planning and Design," *Journal of Accountancy* (May 1986), pp. 110-120.

Bromley, R. G., "Template Design and Review: How to Prevent Spreadsheet Disasters," *Journal of Accountancy* (December 1985), pp. 134-142.

Business Week, "How Personal Computers Can Trip Up Executives," Sept. 24, 1984.

Chew, R., "Transaction Processing Using LOTUS 1-2-3," *Journal of Systems Management* (January 1987), pp. 30-34.

Creeth, R., "Microcomputer Spreadsheets: Their Uses and Abuses," *Journal of Accountancy* (June 1985), pp. 90-93.

Er, M. C., "Principles of Program Documentation," *Journal of Systems Management* (July 1985), pp. 31-35.

Guillemette, R. A., "Application Software Documentation," *Journal of Systems Management* (May 1987), pp. 36-39.

Kee, R. C., and J. O. Mason, "Preventing Errors in Spreadsheets," *Internal Auditor* (Feb. 1988), pp. 42-47.

Simkin, Mark, "How to Validate Spreadsheets," *Journal of Accountancy* (November 1987), pp. 130-138.

Whitten, J. L., L. D. Bentley and T. I. M. Ho, *Systems Analysis and Design* (Times Mirror/Mosby College Publishing, 1986).

Yoder, S. E., "Designing Advanced Spreadsheet for Small Business Clients: An Engagement Report," *Journal of Accountancy* (August 1986), pp. 126-129.