DOCUMENT RESUME

ED 276 822 CE 045 828

AUTHOR McGrann, James M.
TITLE Issues Related to Microcomputer Agricultural Software Design and Distribution in Developing Countries. Faculty Paper Series 85-6.
INSTITUTION Texas A and M Univ., College Station. Dept. of Agricultural Economics.
PUB DATE Dec 85
NOTE 17p.
PUB TYPE Viewpoints (120)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Agricultural Education; *Agriculture; *Computer Oriented Programs; *Computer Software; *Delivery Systems; *Developing Nations; Extension Education; Information Dissemination; *Microcomputers; Postsecondary Education; Program Development

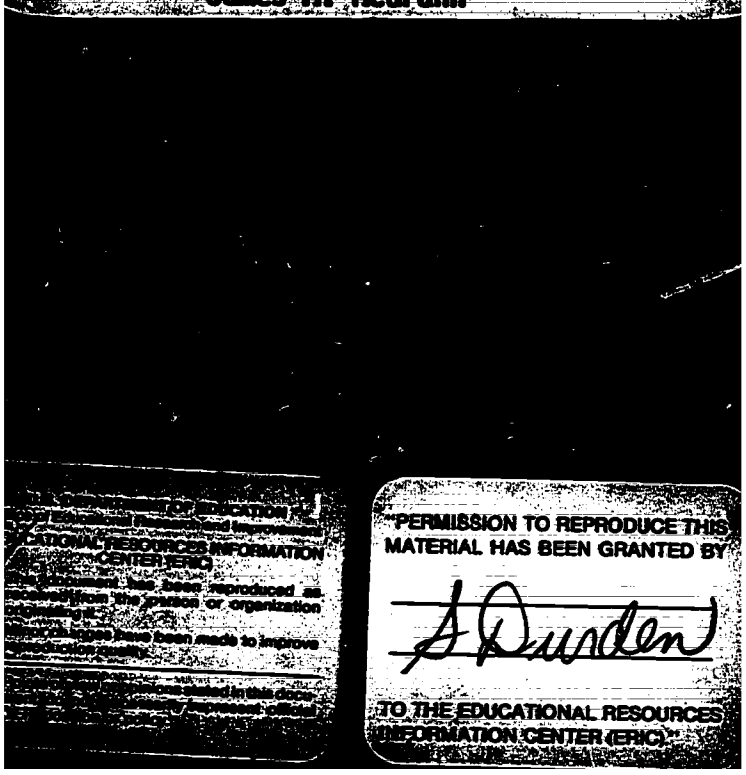
ABSTRACT

This paper deals with the issues related to the design and distribution of microcomputer software, focusing on agricultural software and its users in developing countries (researchers, Extension educators, planners, lenders). The paper is organized in four sections that discuss the following topics: defining the audience and software needs; software evaluation; steps in software development; and software distribution and maintenance support. The paper provides general information about these areas and offers some specific guidance related to software development and development of users' manuals and other consumer helps. Appendixes identify areas in which microcomputers may be used in developing countries for finance and economics and summarize the components of microcomputer software design. (KC)

FP 85-6 December 1985

Issues Related to Microcomputer Agricultural Software Design and Distribution in Developing Countries

James M. McGrann



Issues Related to Microcomputer Agricultural Software Design and Distribution in Developing Countries[1]

There is little doubt that microcomputer applications will continue to increase in developing countries. This tool allows for decentralization of computer use and puts the computer in the hands of the researcher, extension specialist and planner. The usefulness of the microcomputer, however, will always be limited by the availability of:

1. Appropriate software.
2. Good relevant data.
3. Subject matter knowledge to combine with the computer information.
4. A committment to learn how to effectively use the tool.

This paper will focus on the issues related to design and distribution of microcomputer software. The focus is on the users (researchers, Extension educators, planners, lenders, etc.) and agricultural software. Specifically, the paper discusses four issues which include:

1. Defining the audience and software needs.
2. Software evaluation.
3. Steps in software development.
4. Software distribution and maintenance support.

Appendix A, as defined by Salzburg Seminar Fellows in Session 240, identifies areas in which microcomputers may potentially be used in developing countries for finance and economics.

---

[1] Prepared for the Salzburg Seminar, Session 240, by Dr. James M. McGrann, Professor, Department of Agricultural Economics, Texas A&M University, College Station, Texas, April 1985.

I. Defining Audience and Needs

The microcomputer and its software are tools that serve as extensions of the human mind. Successful application of the tools requires identification of specific audience characteristics and their needs. In developing countries, for the most part, microcomputers will not be used for sometime by individual farmers. Instead, supporting individuals above the farm-level decision makers. Such as researchers, extension workers, policy makers, lenders, etc., that get involved in decision-making through technology selection and recommendations that are followed by individual farmers. To identify potential applications of the computer requires a knowledge of the farm decision environment, as well as an understanding of the user subject matter knowledge level.

To define needs, users should have some knowledge of the applications and limitations of the microcomputer; then systematically list activities throughout the year where a computer application would be appropriate.

There are two rules to follow in defining software needs for any specific audience. They are as follows:

1. Involve the end user in the process of defining needs;
2. Define the final information needs of users of the computer software before addressing issues related to data gathering and processing.

The goal of any microcomputer application effort is to improve the existing methods used to process data and generate information. Often, simple applications have greater positive impacts. The "KISS" approach has validity in microcomputer applications, which is -Keep it simple stupid-.

Priority selections are difficult, but essential, in that seldom are there sufficient resources to meet all needs. Again, involve the user in defining priorities.

II. Software Evaluation

Once the audience needs and software priorities are defined, a systematic effort should be made to evaluate existing software to see if it can meet, or be adapted to meet, those specific need(s).

Software evaluations should consider both the subject matter content and the programming evaluation. This is true for software developed using a specific programming code, software development using spreadsheets, or data base software.

The subject matter level of evaluation should consider:

1. Does it do what is needed?
2. Is the methodology used in the analysis appropriate?
3. Is the program well documented (coded if applicable) and the user's manual complete?

If the user's manual is not complete, it is often difficult to process the "calculation" procedures (algorithm) used in the program. Knowledge of the calculation procedures is essential in evaluating the software by the subject matter software.

The programming, or technical evaluation, deals with the presentation of inputs and outputs, and program operation. High quality software also has error checks on data entry and file management. Many of the features associated with "ease of use" and "on-line" user instructions are very helpful in the initial use of a program. The usefulness of these features diminish as one becomes more accustomed to the software. When possible, having the option to

override some of the operation instructions is helpful. Because of the importance of the knowledge of programming procedures, one should have a qualified programmer or system analyst involved in the technical review of software.

Quality software also has an artistic component that is a function of the programmer's capabilities. This is difficult to describe but easy to see once software is used.

After the audience and their needs are defined and available software evaluated, then if it is necessary to develop new software, the priority areas must be redefined. Software developers should return to the user to reconfirm needs and gain a committment for participation. The following section describes the steps in software development.

III. Steps in Software Development

Development of microcomputer software follows a procedure similar to any other scientific development work. This is true whether the program is going to be formatted into an electronic spreadsheet, or simply using a computer language. The basic steps to follow in software development can be summarized as follows:

1. Problem definition.
2. Research and software review.
3. Program design and manual preparation.
4. Program coding.
5. Program documentation and final manual completion.
6. Program review, verification, field testing and revision.

Following these steps can save the programmer time, an often scarce resource, and insure a quality product that meets clientele needs. Described are each of the previously mentioned steps in more detail.

Problem Definition

Problem definition is the single most important step in successful software development. Good problem definition, to identify needs, requires subject-matter knowledge of the problem environment and close communication with the potential end users. In order to define areas where software development can be useful, the designer must have knowledge of how the computer can be used to facilitate data generation, storage and processing. The designer must also be aware of how much knowledge the targeted end user has concerning the application being designed, as well as the type of data that is available to the user. It is extremely important that the designer consider end user input in defining the problems to be addressed in software development.

Research and Software Review

Not only do such reviews prevent a duplication of efforts, it also enhances the possibility of building on another person's research and software experience to make a higher quality product. Few software development endeavors begin as "completely new", on the contrary, most software packages build on previous research efforts and software development.

The identification and review of existing software is not an easy task. One problem is the process of actually gaining access to information on available software. A number of public and private

agricultural software, source lists, and computer newsletters are normally available from software vendors and software supplying institutions that can be useful in identifying available software. One can anticipate that there will be a growth in the publication of articles on software that will improve information flows, thus, making it easier to identify and review existing software packages.

Once existing software has been identified, it is almost a necessity to gain access to the software package for a hands-on review. It is very difficult to fully understand the capabilities and limitations of a program without actually using it on a computer. If the software cannot be acquired, the authors can be contacted directly to determine if the software addresses the identified needs.

Once the problem has been defined, the designer needs to review previous research relevant to the problem in question to determine if existing software is adequate.

A literature review of previous research should preceed existing software reviews to insure all relevant information is assembled prior to software evaluation. The research review will insure that relevant information was not overlooked by the designers of the existing software and also identify the potential need for additional research in other areas.

Software Design and Manual Preparation

Once the designer/researcher is satisfied that a new piece of software is necessary to solve a problem, it is time to begin the actual design. The software design is the organization and description of the software that is to be programmed. The communication link between the potential users of the software, the

designer, the programmer and subject matter specialist put the ideas together. The software design is the most creative part of software development.

Listed below are some components a designer might want to consider in putting together a software package for a programmer to follow:

1. A precise descriptive title.
2. A clearly stated objective of the software and the specific clientele it will serve.
3. A diagram(s) of the program structure and linkages (flow charts are useful tools).
4. Data entry screens and variable requirements--A data entry screen may consist of several questions for the user of the software to answer. Each question needs to contain such information as types of error checks, data ranges and help guides.
5. Mistake correction routine--This gives the user the opportunity to review and correct any erroneous data entered.
6. The calculation procedure or program algorithm.--All variables must be clearly labeled and formulas specified so they are easy to follow between data input and outputs. Notes to the programmer can be very helpful in establishing the clarity needed for this section of a design.
7. Specific layout(s) for output(s) or report(s).
8. Information dealing with disk storage and retrieval capabilities.
9. Example(s) worked out using the data entry screens, formulas and output reports for each option (working examples before designs are programmed is a <u>key</u> to efficient software development).
10. A preliminary user's manual for the software should be prepared before the design is given to the programmer. This can again increase the quality of the design and improve communication with the programmer(s) (Components of the users manual are discussed in detail later).
11. Subject matter and system design review--It is valuable to have design reviews by experienced professionals and users to incorporate these ideas into a final design.

12. An indepth review with the programmer(s)--The working relationship should be established for continued communication in the coding phase of software development.

Some of the tasks previously mentioned can be difficult for someone who has little or no computer experience. Therefore, it is wise to include a computer analyst, or programmer, to work with the subject matter specialist to design the software. The advantage of having a systems analyst is that he/she usually knows more about program structuring, data structuring, disk storage and retrieval requirements than a programmer. The systems analyst's knowledge might produce a better design, thus, reducing the amount of programming time.

The complexity of the program will dictate the need for a systems analyst.

A summary of the components of microcomputer software design is shown in Appendix B.

Program Coding

Once the design has been completed, it is given to the programmer to be coded. The communication lines between the designer and the programmer should have already been established. The programmer, therefore, should already be aware of the requirements necessary to code the software. During this stage of development, the interchange between the designer and the programmer become shorter and less frequent with the programmer consulting the designer only when a problem arises or when information is needed.

The programmer writing the code must take the responsibility to prepare the program code documentation that will facilitate program revision, correction and enhancement -- a burdensome task to most programmers. However, it is very important. A poorly documented code will make recoding activities very difficult and extremely costly. The program code should be reviewed by several other programmers to insure that the code can be followed and to help potential problem areas.

Final User's Manual

The final user's manual should build on the manual developed in the design step of software development and should be a joint effort of the software designer(s) and programmer(s). Specific components of a user's manual are:

1. Title page and liability statement.
2. Table of contents.
3. Operational instructions with loading and exit procedures.
4. Screen explanation, both data entry and outputs.
5. Formulas and calculation procedure.
6. Definitions used in the program and manual.
7. References and sources of supporting information.

Few programs are well documented and fewer have good, complete user's manuals. A program's quality and often usefulness, is a function of completing a documentation. Technical writers can be helpful in completing this hard-to-finish final step and add to the quality of the user's manual.

Program Review and Verification

The final step in program development is program review, verification, field testing and revision. The time required for this activity depends entirely on how well the design was thought through, communicated to the programmer and then executed into a program. At

this phase, numerous people should test the program's effectiveness by running the program, reviewing the documentation, examining the subject matter and algorithms, and checking out every aspect of the newly developed software. This activity should also be carried out by some of the anticipated users of the program. User input is sought throughout the development process, but real interest and useful advice is easier to attain after the user can run the program on a computer and look over a manual. It is often helpful to get unfamiliar users to evaluate the program in a field-testing environment to get positive feedback on ways of improving the program. The review process involves testing all the options of the program as well as making sure that proper error checking methods are employed to insure a user-friendly product. After completion of this step, and the reprogramming is completed, the software can be released for distribution.

IV. Software Distribution and Maintenance Support

When an individual or organization makes the decision to share software, the need for a distribution procedure, software support and maintenance begins. This is a costly activity which offers little opportunity for creativity or rewards associated with software development. Software, however, has only a limited value, if it is not distributed and supported for a wide audience.

The most efficient means to provide software distribution and support is through a centralized distribution and support unit. Individual authors provide the software to this unit, and users order software from the centralized unit. This takes the day-to-day

distribution activity out of the hands of developers and allows a more efficient use of limited resources. The distribution unit facilitates and encourages participants to develop and complete software for distribution. Much of the software support can also be handled by this unit, particularly in regard to computer and software operation issues. Subject matter support requirements most often must be met by software authors.

Since no software is perfect, software maintenance and enhancement capabilities are required. The dynamics of microcomputer technology has also required maintenance capability to respond to changing technology. The responsibility of software maintenance and enhancement must be a combination of the author(s) and the distribution unit.

Software that is well designed, rigorously reviewed, and supported by good users' manuals will require less education and maintenance support. For this reason, good design, documentation of software and indepth reviews should be encouraged.

Other issues that must be addressed for successful software distribution include:

1. Development of common standards and review procedures.
2. Setting distribution fees.
3. Software copywriting.
4. Software code release.

An efficient distribution unit can not only encourage more author participation but offers a tremendous opportunity for the successful use of microcomputers.

APPENDIX A

Potential Areas of Microcomputer Use in Finance And Economics in Developing Countries

I. **Farm Level** (to address extension agent or farm manager needs)

1. Financial statements
   - cash flow, income statement, balance sheet

2. Day to day funds management
   - cash flow
   - accounts payable and receivable
   - farm accounting
     * individual farm or cooperative

3. Enterprise budgeting
   * crops and livestock
   Partial budgeting
   * marginal analysis
   Whole farm budgeting
   * systems level

4. Use of budgets to evaluate
   - marketing alternatives
   - technologies
   - investment alternatives

5. Investment analysis
   - net present values analysis - capital budgeting
   - loan, payments, costs etc.
   - investment alternatives
     * financing terms, lease vs. buy etc.

6. Family support, cash flow projections and managements

7. Livestock
   - ration formulation
   - performance evaluation using economic criteria

8. Machinery economics and finance
   Cost:
   - capacity
   - efficiency
   Replacement
   Investment alternatives
   Records:
   - use level
   - maintenance
   - scheduling
   - machinery selection
   - matching equipment

9. Inventory management
   - inputs
   - livestock and crop

10. Extension teaching preparation

11. Economic analysis of input use
    - economic threshold for pesticides
    - production response for inputs
    - partial budget use for input use decisions

APPENDIX B

Summary of the Components of Microcomputer Software Design

I. **For Software Coded in Programming Language**

1. Objective statement

2. Diagram(s) of program structure and linkages (flow charts are useful)

3. Data input screen format
   - Columns 1 through 79 are used
   - Some lines are used for titles, messages, and borders leaving the rest for data entry.
   - Specify data range for each entry

4. Mistake correction routine specification

5. Output screen format
   - Two types of output
     * video
     * printer
   - Columns 1 through 79

6. Calculation procedure
   - Label each variable
   - Specify each equation

7. Manually worked out example for each data entry and output option

8. User manual with following components
   - Title screen
   - Operational instructions
   - Screen explanation (data input and output)
   - Definitions
   - Formulas and calculation procedure
   - References
   - Disclaimer

II. Specific Components of the Spreadsheet Design

1. Title
2. Objective
3. Data Input and Format
4. Formulas
5. Output and Format
6. Formula and Cell Protection Specification
7. Definitions
8. Guides on Operation Procedures
9. Examples Using All Formulas and Options
10. References
11. Disclaimer (see example associated with language coded programs)