Software Development for Computer Applications in Agriculture and Forestry

July 20-21, 1982

Atlanta, Georgia
Proceedings of

SOFTWARE DEVELOPMENT FOR COMPUTER APPLICATIONS
IN AGRICULTURE AND FORESTRY

Southern Association of Experiment Station Directors
Southern Research Development Committee

on

Computer Software Development

Prepared by the Southern Rural Development Center

Atlanta, Georgia
July 20-21, 1982
INTRODUCTION

Thirty-five persons attended the second meeting of the Southern Research Development Committee on Computer Software Development July 20-21, 1982, at the Capitol Airport Inn, Atlanta, Georgia. The first meeting of the group was held April 15, 1982, in Memphis, Tennessee, with the charge to consider the subject of computer software development in agriculture and forestry and to recommend a course of action to the Southern Association of Experiment Station Directors.

Although most state Extension Services and Experiment Stations are involved with computer applications, very little organized effort exists on a regional basis and no agreement has been reached concerning standards, needs and a definition of roles. As on-farm use of computers increases and this technology becomes available for widespread use, more regional coordination may be necessary in software development. One of the goals of this committee is to determine what portion of each state's efforts in this area can best be done on a regional basis and then to pull together a consensus of opinion concerning what kind of cooperative activity is necessary.

The second meeting of the committee featured five prepared papers which served as the basis for group discussion concerning such topics as present and future computer software needs of farmers; the role of land-grant researchers in software development; and the problems of software authorship, documentation, field testing, quality control, distribution and maintenance. This proceedings is a compilation of those papers and is prepared as part of the continuing effort of the committee to determine an appropriate course of action in a new age of on-farm computers.
TABLE OF CONTENTS

Introduction ................................................................. iii

Present and Future Computer Software Needs of
Farmers, Dr. Robert Kramer, The Kellogg
Foundation ................................................................. 1

Role of SAES Scientists in Developing Software,
Dr. Robert Sowell, North Carolina State
University ................................................................. 15

Role of SAES Scientists in Developing Software,
Dr. James McGrann, Texas A&M University ............... 21

Software Authorship, Documentation, and
Field Testing, Dr. Wallace Killcreas,
Mississippi State University ........................................... 37

Quality Control, Distribution and Maintenance of
Software, Dr. Steve Welch, North Central
Computer Institute ...................................................... 59

Workshop Program .......................................................... 107

Participants ................................................................. 109
The Future of Computers in American Agriculture

Robert C. Kramer
Program Director
W. K. Kellogg Foundation

A visitor to an American home in the last decade of this century might note several scenes not present in today’s homes. For example, while a six-year-old child might be seen playing with toys, these toys are different from those currently available because they are computer-controlled, and the child has programmed their actions. One toy is tracing a figure on a piece of paper spread on the floor. It is doing this without human intervention. The child is seen watching the pattern being drawn. If he decides that the figure is not to his liking, he walks over to a typewriter and keys his corrections to the program controlling the toy’s movements.

In another part of the house two teenagers are seen playing Space-War, a game played on a TV-like screen in a fashion similar to Ping-Pong. In Space-War, each side has a rocket with a number of missiles. The game is more complex and interesting than Ping-Pong because the rockets move in curvilinear fashion owing to the presence of a gravitational field generated by a star depicted by a bright spot in the center of the screen.

Later one of the teenagers is seen reading a book. The pages of the book are displayed on the face of a hand-held screen and are turned by pressing buttons on the console attached to the screen. On a nearby screen the other teenager is comparing the current batting statistics of his favorite baseball team with statistics taken at a similar point in the schedule a year earlier. The father appears to be reading the newspaper
on yet another screen. He uses the console to obtain more detailed information on a news item of particular interest to him. The mother is seen paying this month's telephone bill. No checks are visible. Rather, the bill is viewed on the screen, and various keys are punched on the console informing the bank to pay the telephone company the full amount.¹

The W. K. Kellogg Foundation's interest in the utilization of computers has covered our three programming fields--agriculture, education, and health. Several higher education institutions and academic health centers have received grants so that computer applications could be utilized to improve the educational and health delivery processes. More than 15 years ago the Kellogg Foundation provided a grant to Michigan State University to start a computer program which is called TELFARM. A few years after TELFARM began operating, the Kellogg Foundation provided another grant to Michigan State University to start the program called TELPLAN. Five years ago the Kellogg Foundation provided monies to Purdue University to start the Indiana state system called FACTS.

In the past 18 months the Kellogg Foundation has funded the following state projects:

<table>
<thead>
<tr>
<th>RECIPIENT</th>
<th>PURPOSE</th>
<th>PROJECT DIRECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michigan State U.</td>
<td>Maximize animal production by computer evaluation of cost benefits of animal health care</td>
<td>Dr. Edward Mather Chairman Dept. of Large Animal Surgery</td>
</tr>
<tr>
<td>U. of Florida</td>
<td>Establish a computerized agricultural diagnosis, consultation, and management system and computerized crop and pest control recommendations</td>
<td>Dr. F. A. Johnson Assoc. Professor and Extension Entomologist</td>
</tr>
</tbody>
</table>

Cornell U  Develop a computerized program to improve the transfer of agricultural technology to New York dairy farmers  Dr. Wayne A. Knoblauch  
Dept. of Economics

U. of Illinois  Assist Illinois farmers to use mini and microcomputer technology to improve their farming operations  Dr. Delmar F. Wilken  
Prof. of Farm Management

Clemson U.  Improve the transfer of agricultural technology to agricultural producers in South Carolina by installing microcomputers on campus and promoting their usage statewide  Dr. W. A. Tinsley, Dept. of Ag. Economics & Rural Sociology

Calif. State U., Chico  Enable faculty members and NOWCASTING, a specialized computer weather forecasting service, professional staff to assist Sacramento Valley ranchers in improving farming operations  Dr. Rolland K. Hauser, NOWCASTING

Purdue U.  Indiana Farmer Survey  Dr. Howard G. Diesslin  
Director, Indiana Cooperative Extension Service

U. of California Berkeley  Enable agricultural economics faculty to convert a large computer-cost-of-production program into a software package for microcomputers  Dr. Gordon Rowe

Penn. State U.  Northeast region computer feasibility study  Dr. G. Art Hussey, Jr.  
State Leadership Computer Services

U. of Arizona  Western region computer feasibility study  Dr. Roy Rauschkolb  
Director of Cooperative Extension Service

U. of Arkansas  Beef evaluation  Dr. Hayden Brown, Jr.  
Professor, Dept. of Animal Sciences

In the past three years I have had an opportunity to meet with all of the directors or associate directors of the Cooperative Extension Services in the
United States. I have also had an opportunity to meet with Extension specialists and faculty members in agriculture from many states in this nation. They report that farmers are increasingly using computers and the computer networks which are operated by the University of Nebraska, Michigan State University, and Virginia Tech. These networks are called AGNET, TELPLAN, and CMN. Extension agents are using terminals to access university main frame computers via these networks. A few farmers are using terminals to access large computers—owned by universities or private companies. Farmers are buying their own Radio Shack, Apple, or other small computers. (I include both the micro and minicomputers in the definition "small.") Feed companies, farm management associations, banks, and governmental agencies such as the Farmers Home Administration are also increasing the use of computers and providing farmers with computer information.

There have been large increases in the utilization of computers by members of the U. S. society. International reports likewise indicate that computers are expanding in use in many, many foreign countries. An article in the Wall Street Journal, Tuesday, June 3, 1980, says that about 2.8 million U. S. students in about 6 percent of the public elementary and high schools already do some of their work on computers, up from 1.5 million students in 4 percent of the schools four years ago. That number will grow over the next four years to 5 million students in 9 percent of the schools. About 13 percent of the 998,000 microcomputers sold in 1982 will be used by schools, up from 11.5 percent of the 315,000 sold in 1979. Students will come to colleges with many computer experiences in the 1980s.

It is estimated that there are now nearly two million personal computers owned by U. S. families. These are microcomputers and many farmers own micros.
In a recent survey of Indiana's farmers, 13 percent of the 354 respondents presently own an on-farm computer. The growth rate of on-farm computers in this sample group is accelerating each year. In 1979 the number of computers increased 2 percent, in 1980 it grew 3 percent, in 1981 7 percent of the group bought their own system. The projection for 1982 is 10 percent, and it is possible that 24 percent of the sample will have their own system by the end of the year. Since there are 87,000 farms in Indiana, the potential number of on-farm computers during the next few years is in the tens of thousands. If this sample is representative of what is occurring in other states, and if the growth trend continues, there could be over one million computers in use on farms by 1985.

Today a single silicon chip can store 10,000 words; very soon it will be able to encompass an entire novel; by the end of the 80s, a whole set of books--and at a tenth of the production cost.\(^2\)

Fifteen years ago experts said no computer could be programmed to play a decent game of chess; today there are computers which can beat 99.5 percent of the world's chess players, and very soon they will reach the International Master level.\(^2\)

Thirty years ago a computer with the same number of functions as a human brain would have had to be the size of New York City and would have used more power than the subway system; today that computer would be the size of a TV set; by the end of the 80s, it will be as small as a human brain and will run on a transistor radio battery.\(^2\)

Information obtained from the media, from meetings, from advisory committees and from computer experts have led the W. K. Kellogg Foundation

to conclude that it wishes to help institutions develop and deliver agri-
cultural computer programs in this decade. Listed below are the summary
points that have arisen from this study and from these experts:

1. Computers are being developed with increasing capabilities and
   the hardware is decreasing in cost. If the efficiency and cheap-
   ness of the car had improved at the same rate as the computer's
   over the last two decades, a Rolls-Royce would cost about $3.00
   and would get three million miles to the gallon.

2. Additional computer programs (software) are needed in U. S.
   agriculture.

3. The cost to develop and maintain application programs can often
   be higher than the cost of the hardware on which it runs. If each
   individual farmer is required to pay for initial development, enhance-
   ments, and maintenance of his own software, the costs could be up
   to eight times more expensive than his hardware.

4. The clientele for agricultural computer information includes
   farmers, extension personnel, agribusiness executives, research
   scientists, college instructors, high school teachers, farm manage-
   ment consultants, and USDA action agency personnel.

5. There are three voluntary, agriculturally oriented regional
   computer systems in the United States. These are TELPLAN, AGENT,
   and CMN. In addition to the Purdue FACTS computer network,
   there are other state plans—for example, WISPLAN—that provide
   information to selected farmers in their respective states.

6. All states either have or will soon have computer programs for
   their agricultural producers.
7. Hand-held programmable calculators are available at reasonable costs. They can help farmers and ranchers with less-complicated problems, and they are most suitable for daily or frequent use. The main limitation of the hand-held calculator is its lack of data storage. Today's farm records require several thousand to several million characters of storage capacity. It requires a micro or mini with that capacity if the farmer is going to store his production and financial data in an on-farm system.

8. Land-grant universities have faculty members with expertise to develop applicable programs for U. S. agriculture. Private businesses, generally speaking, do not have the agricultural expertise to develop the agricultural computer programs. (There are some exceptions to these statements, but generally, they do not.)

9. Interdepartmental cooperation in universities in computer program development is insufficient.

10. Agricultural administrators in colleges of agriculture have not supported computer program development and application as much as is needed. The administrators in the North Central Region have supported the North Central Computer Institute recently begun at the University of Wisconsin-Madison.

11. Poor transmission capability of computer data to farmers and from farmers to computer centers is an impediment to greater usage. There are telephone quality problems and inadequate and insufficient transmission lines in many U. S. rural communities. This situation requires some creative efforts to develop local and national data bases and a cost effective network facility.
12. The large number of computer manufacturers and the varieties of hardware available create problems such as linking computers, training individuals, and standardizing computer programs.

13. Farmers and ranchers should be consulted about their computer needs as programs are being developed and tested by university faculty members.

I want to share with you what I expect the situation will be in 1990. I will predict what will be the situation on U. S. farms—particularly mid-size farms. I will share with you what I think will be the developments in the county and state Cooperative Extension Service programs. I will forecast what I see at the regional level. And, lastly, I will indicate what I think will happen on the national scene.

Three-fourths of all of the mid-size farmers in the United States in 1990 will utilize computer software in helping make management decisions on their farms. A large proportion of the mid-size farms (I am speaking about those farms that have gross farm sales ranging between $20,000 and $150,000 per year) will utilize their own computers. They will store their individual farm information in their computers and will have access to larger computers which can download data and programs into their on-farm systems.

Many of the farming operations will be computerized and controlled by electronic instrumentation, thereby automating many standard tasks.

There will be intelligent computer terminals in 90 percent of the county extension offices in the United States. There will be communications between farm offices and county extension offices by telephone, by FM signals, or by hard wire. There will be an expert on computers in each of the multi-agent county extension offices who will help to educate farmers and assist
the other agents in the separate programming fields. There will be computers in the district or area offices of the extension service. Among the district agents or area agents there will be a computer specialist who will assist his colleagues in extension programming at the area level. All of the agricultural administrators will be much more sophisticated about computers and computer programming. There will be at least one computer specialist on the state staff of the Cooperative Extension Service. This individual will serve the extension computer needs of the Schools or Colleges of Agriculture and Home Economics.

Virtually all of the departments in the Colleges of Agriculture and Home Economics will have intelligent terminals and/or small computers. On-campus interdisciplinary cooperation will increase significantly in the decade of the 80s. Subject matter specialists and researchers will cooperate in developing software that can be utilized by extension professionals and farmers to help agricultural producers with decision-making. Academic personnel policies will be modified so that specialists and researchers will be given credit for their contributions to computer applications. Credit will be given for the development of computer programs and credit will be given for the development of educational and decision-making tools which will be utilized by agricultural producers.

U. S. agriculture probably has the honor of being the top industry in the United States where the end-users (farmers) know more about computer programs and utilize computers more frequently than their counterparts in other industries. In 1990, the gap will widen and even more end-users in agriculture will be sophisticated and will be using advanced computer technologies.
Computer marketing or electronic marketing will be commonplace in 1990. Many of the marketing decisions for agricultural products will be made at the farm, farmer co-op, or county level utilizing sophisticated computer marketing techniques.

There will be knowledgeable and active computer committees on each of the land-grant university campuses. These computer committees will have a definite role and will provide the latest information so that the land-grant university can provide timely and usable information to their clientele for decision-making.

In each of the four geographic regions of the United States, there will be agricultural computer centers or institutes. There is one in the North Central Region. In my crystal ball, it is difficult to see whether there will be one institute or center for teaching, research, or extension or a separate center for each. These centers or institutes will help with the development of regional programs, maintain an up-to-date inventory of software, help with the standardization of computer languages, develop methods whereby computer equipment manufactured by several different companies can be interlinked via electronic communications, and advise interested campus administrators how to have a cost-effective computer system in each of the states.

Regional sharing of software, regional sharing of specialists, and regional computer networking will be commonplace in 1990.

Regional committees will work with farm organizations, agricultural banks, agricultural cooperatives, and private agri-businesses in the sharing of data and data bases which will be used by commercial farmers in each of the regions.
At the national level, the U. S. Department of Agriculture will have an effective computer committee that will span the different parts of the U. S. Department of Agriculture and will also tie to the Department of Commerce, Department of Labor, Department of Education, and other governmental agencies.

In the action agencies of the U. S. Department of Agriculture, there will be a committee that will coordinate the different programs that are developed for computers so that U. S. farmers can be better served. In the other USDA agencies (ESS, FAS, FCS, etc.), there will be a coordinated development of computer software. These programs such as OASIS will be placed on an on-line, interactive information retrieval service. Individual state specialists, researchers, and administrators will be able to access the on-line service and get the latest market information, the latest statistics on U. S. and world agriculture. The Situation and Outlook Board will release its lock-up reports to an on-line interactive information retrieval service and 10 or 15 minutes after the reports are released, this information will be available to all those who have made the necessary preparations to access this information.

There will be a national inventory of agricultural software which will be developed with the assistance of specialists and faculty members in each state, regional computer institutes, and a national computer committee. This national inventory of agricultural software will be on an on-line, interactive information retrieval service. By using TYMNET or TELENET, an inquirer can learn quickly what programs are available and for which hardware they are written.

There will be a national computer policy committee which will be effective in making recommendations about effectively using computers to assist agricultural producers.
There will be a national computer center located either on one of the land-grant university campuses or in an off-campus location. In this national computer center will be the latest hardware, expertise, and training programs so that agents, specialists, researchers, and administrators in the land-grant system can learn about computers and their agricultural applications.

With specific reference to on-farm computers, I would like to share some of my thoughts concerning present and future computer software needs of farmers.

1. Farmers want education and training in the use of computers. They need this background and knowledge in order to accurately use and interpret what comes out of a computer.

2. Farmers want impartial advice about both software and hardware.

3. Farmers want good, economical software.

4. Farmers want decentralized computing. They want it in their own hands.

5. Farmers want updated and revised programs. This will require extra effort to inform users of availability of revisions and to encourage farmers and other users to incorporate these revisions into their computer software.

6. Farmers want to know how to locate and how to acquire software.

7. Farmers want adequate documentation of software. They want guidelines for obtaining data needed to operate the program, particularly programs from land-grant universities. Perhaps an agricultural computer cooperative is needed to work with the Cooperative Extension Service in order to take over the function of transferring software systems.
8. Farmers want software programs on mixed cropping practices. We develop software packages on soybeans, rice or other single commodities. What about programs for the whole farm with crop rotations? What are the effects when one crop is harvested and the other is planted? Farmers want software programs to choose crop varieties for their farm when they have to plant on a certain day, plan to harvest on a certain day and have specific soils by type and by Ph. Perhaps their fields lie a certain way and they have either natural or irrigation moisture. In addition, they want software for selecting pesticides for specific situations, especially if they have more than one insect. Farmers will want software showing the chemical structure and class, hazards involved with particular pesticides, precautions to be taken, symptoms of poisoning, and medical treatment in case of poisoning.

I believe that farmers will have two types of computers in the future. One will be a general purpose computer system used to maintain, analyze, and summarize financial and production records. This system will use an integrated set of software to deal with management problems of farm operation. It will monitor, collect, and communicate physical and financial information and use a refined data base management subsystem.

The second type of computer will be a special purpose system, a micro-computer system with a micro processor and a memory. It will be fed by monitors within fields, cribs or units on the farm and by external data from the general purpose on-farm computer. Hardware or short range radio transmission will provide the needed data. In this way all moisture and land humidity will be monitored in order to predict drought stress and activate an irrigation system.
I hope you and your colleagues in extension and teaching can jointly develop and deliver computer software and teach students, farmers, extension personnel, research workers and agribusiness leaders how to use it. Agriculture in the South will benefit if this happens.
The Role of the SAES Scientist in Developing Software for Farmers

Robert S. Sowell
Department of Biological and Agricultural Engineering
North Carolina State University

Before addressing the question of the role of the research scientist in developing software for farmers, let us briefly examine the role of the land-grant institution in providing direction for effective utilization of computer technology in production agriculture. Through our instructional component we have the responsibility to encourage, if not direct, all students in schools and colleges of agriculture to take courses that will give them some knowledge of computers and their applications in agriculture. Extension has the responsibility of keeping its clientele educated on new innovations, such as micro-computers, which have potential for improving the efficiency of production agriculture. (I also believe that Extension has an opportunity to utilize computers to enhance its educational delivery process. In the process of developing extension educational software, software will be available that has direct on-the-farm application.) The role of the research scientist then is to develop new and innovative ways to utilize computer technology for the betterment of agriculture. Such new and innovative uses of computers in agriculture make take the form of hardware, software, or both. This brings us to our theme for this session, that of the researcher's role in developing software.

I see the role of the researcher in developing software for farmers as falling into two categories.
1. Work cooperatively with the extension service by providing support for the development of software for client educational programs and for client use.

2. Develop software which results from new methods, procedures, information, and technology brought about by research and utilize computers for delivery of the end product.

Most public development software used by farmers today has been developed to run on state computer networks and microcomputers being placed in county extension offices. Extension is generally taking the lead in these developments and most of the software is being developed by extension professionals. I realize that in some states researchers are playing a key role in software development, perhaps even taking the lead in some cases. However, I believe this to be the exception. Reviews of listings of agricultural software available indicates that the technology, methodology, and information used in the development of software is currently available and in most cases has been available for some time. The computer is simply automating the process of solving many different types of problems. I am thinking of the software programs that appear to be currently of most interest to farmers, e.g., financial statements, enterprise budgeting, financial records, etc. I believe that extension will and should continue to take the primary responsibility for development of this type of software. However, I do also believe that the research scientist has a support role in this process. As microcomputers increase in capability (speed and storage capacity), so will the sophistication of software that will run on them. Many researcher developed software packages which currently run on large computers and have applications at the farm level will have the capability of running on micros. There will be a need for the research
scientist to work with the extension specialist to adapt or convert existing mainframe software to run on microcomputers, such as the least cost feed ration program which will utilize linear programming. These are available on mainframe computers at most of the land-grant campuses. More software will be developed to take advantage of increased microcomputer capability, and I believe the research scientist will be more active on the development of this type of software. Let me cite one example of this at North Carolina State University. In 1978 a project was initiated to develop a model for determining optimal mixes of crops in greenhouse farms. From the outset one of the objectives of the project was an end product that could be used by greenhouse operators in planning their annual production schedules. It was anticipated that by the time the model was available there would be terminals available in county extension offices so that growers could access the software on the university's mainframe computer. It is now feasible that the program or a modified version of the program will run on a 16-bit microcomputer with 256+K of memory. More research input will be necessary before this is possible.

The second role of the researcher in developing software for farmers concerns the use of new research results as in the development of research projects which will have end products that depend on the use of computers. I will discuss this role by referring to some specific examples. The first example is one that I mentioned to this committee at our first meeting in Memphis in April of this year. In the late 1960's, under the direction of Dr. George Kriz, new procedures were developed at North Carolina State University for designing fields for land farming. The practical implementation of the procedure required a computer because of the large number of interactive procedures used. This software was made available to the
public. It has been used by public agencies and engineering consulting firms around the world for designing fields for land farming. This program was designed to run on the mainframe computer. In 1978 a greatly reduced version was adapted to run on the 8-bit micro, but with the advent of the 16-bit micro it may be possible to adapt the original program in its entirety giving it much more widespread distribution and making it available in county extension offices and on farm computers.

Another broad area of potential software is that resulting from crop modeling research. After more than a decade of research activity in plant and crop modeling, the resulting software is now being made available to farmers. This modeling activity was originally justified on the basis that it could help guide research but is now valued for its application in management decision making. Most of this software still resides on mainframe computers and is accessed through remote terminals. It generally requires access to large data bases available on the mainframe. With the advances in hard disks and other main storage devices, it is likely that the required data bases can soon be available on county computers and on farm computers. Jim Jones, Wayne Mishoe and others at the University of Florida are making significant progress in the development of interactive soybean production and management models. Similar systems are sure to be developed for other enterprises, and research scientists will have the primary responsibility for this development.

One last example of the researcher utilizing computers to develop end-products for farmers is in the area of controls. Agricultural engineers are currently doing research on the use of microcomputers in many control applications, e.g., ventilation and heating of animal structures and greenhouses, grain drying, field machinery, etc. While the end product of this
research involves hardware development, it also includes a software component. Again with the rapidly changing technology in microcomputers, multi-user systems will soon be used on farms. The computer can be continuously monitoring and controlling on-farm processes while the farmer is updating a record or analyzing an enterprise budget.
Role of the Southern Agricultural Experiment Station Scientist in Developing Software for Farmers and Ranchers

Dr. James McGrann
Department of Agricultural Economics
Texas A&M University

Recent advancements in microcomputer technology present unprecedented opportunities to the research scientist to facilitate the use of powerful analytical tools and research by agricultural decision makers. This new, low cost technology has put a computer of considerable capacity in the hands of the decision maker at a time when greater public credibility of agricultural research is sought and farmers are in the worst economic situation since the Depression. Taking advantage of this opportunity requires software development and educational support for farmers and ranchers.

Research scientists have an important role in defining software needs, designing models, and specifying analytical procedures for software designs. Researchers have the responsibility to support interdisciplinary efforts necessary to develop user software. It is increasingly important for researchers to encourage professional peer review of research that is used for software development and user software which will lead to greater recognition of the scientific and creative activity of software development for farmers and ranchers. This review activity would improve the final product for the decision maker. Administrators must strive to acquire the resources and support development of incentives necessary for an efficient participation of research scientists in software development.
This paper briefly reviews the current status of agricultural software, the specific role of the agriculture research scientist and the necessary research support for successful software development for farmers and ranchers.

**Current Status of Software Development**

On-farm or on-ranch microcomputer systems can be purchased at four to ten thousand dollars and are being acquired by commercial farmers and ranchers at an accelerating rate. Private software vendors and land grant institutions have begun to provide agriculture software to the growing number of farmers and ranchers with microcomputers. This software is also being used for computer user education in extension and in the classroom.

Doane-Western has identified more than seventy firms that sell agricultural software for microcomputers. This nationwide activity is centered in the Midwest. Strain and Fieser have identified thirty land grant institutions with microcomputer software. Policy on making the software available to the public is highly variable between institutions. Texas A&M University and Mississippi State University are two of a growing number of land grant institutions with an organized support unit to distribute and support software for microcomputers.

The private sector has utilized public supported research and extension work for most of its biological and engineering relationships and analytical procedures used in the software currently available. Decision aid software available from Successful Farmer, Harris Technical Service and AG COM, for example, heavily utilized the programmable calculator (TI-59) programs developed and distributed by the land grant institutions. The private sector has largely provided a re-packaging,
distribution and limited support service for agriculture software. Research to develop and refine methods of analysis and the basic biological and engineering research to define functional relationships for software will have to continue to be supported by the public sector.

Most of the agriculture software presently available focuses on decision aids, record keeping, and accounting packages. This software fits the available methodology and data, and knowledge levels of users. Much of the initial work has been done under the leadership of professionals trained in agricultural economics. Opportunities exist to accelerate the software development activities in the biological sciences and agricultural engineering.

The logical step beyond the decision aid-accounting type software is a whole-farm integrated approach that would tie decision components together. Purdue University has used such a computerized tool on a main frame computers in their Top Farmers program. Considerable research and support will be required to develop user-oriented whole-farm decision tools for the complex Southern agriculture. A great deal of supporting education will also be necessary for these tools to be used by farmers, ranchers or agricultural consultants. This area of software development would be in the experimental or research state until researchers can assemble the necessary information model and prepare a user oriented analytical tool to capture the complexities of the whole-farm decision environment.

The urgency to get software out to the user, the lack of standardizations, and the lack of a formal, professional review process have resulted in some rather poor quality software being distributed for agricultural use. This is true for both the private and public sectors.
Many programs are poorly documented in terms of references and explanation of analytical procedures, data requirements and interpretation of the computer solution. Users are often faced with the "black box" situation where data is fed in and results come out of the computer. They are left with little understanding of what went on inside the computer program. This situation can lead to decision makers accepting the results from the computer "because that is what the computer said" and be worse off in terms of decision making information than before the computer software existed.

Poor quality software problems are compounded by noncompatibility of computer hardware, languages and operating systems which lead to duplicating of efforts and higher cost of software development and distribution.

As a consequence of these problems, software selection and evaluation is an increasingly important activity of researchers when identifying software needs and reviewing existing software to minimize duplication of effort.

The farmer and rancher clientele of the experiment station and extension service have been vocal in their requests for accelerated involvement of the land grant institutions in on-farm computer applications. They desire the objectivity, accountability, and support they have learned to expect from the land grant institutions. They expect the public supported activity to reduce their risk and cost in adopting the new technology. One hundred percent of the participants indicated that they wanted Texas A&M to continue development, distribution, and support of agricultural software in response to a questionnaire evaluating a Texas A&M seminar on computer application (McGrann).
The recent economic crisis has only accelerated the interest in more complete and powerful computerized analytical tools by farmers and ranchers.

Southern states have lagged the Midwest historically in computer applications in agriculture. The relatively low cost microcomputer technology can allow institutions to quickly catch up in "hardware computing power." It will take a much greater acknowledged effort and cost to develop software for a much more diverse and complex agriculture. Demands are great for computer education for research and extension staff and users on how to effectively utilize the technology. Any efforts between states to exchange research results, software, and education materials could be most cost effective through reduction in duplicative efforts.

**Interface of Software Development Activities**

Figure 1 can be used to summarize the interface of the land grant system research and extension activities with the private sector and finally the user clientele. Most all research phases of software development are still supported by the public sector and private supporters such as the Kellogg Foundation which has made a substantial increase in support of software development. Little support is coming from the private software or hardware vendors for software development research activities.

Research in the biological and engineering areas and activities that add the economic and management component will not necessarily change with the utilization of microcomputer technology. It is true, however, that the opportunity to use analytical tools and research at the decision level expands. This technology does broaden the necessary involvement
of the staff in the applied extension-research user software development area.

Figure 1 identifies the interrelationship between software development, distribution, and educational support activities. There are few activities in the land grant institutions that require a stronger tie between basic research, applied research, extension, and clientele than successful user software development for farmers and ranchers. Useful software must communicate effectively with decision maker. Communication with clientele is a primary responsibility of extension in the land grant system. Need for the communication skill in developing user friendly software is an important justification for extension involvement in the problem definition, design, verification and documentation phases of software development with the researchers.
Figure 1. Interface of Research, Extension and the Private Sector in Computer Software Development and Distribution

RESEARCH
- Biological
- Engineering

RESEARCH
- Economic Component
- Management Framework

PRIVATE SECTOR
- Software development
- Software distribution
- Hardware & software support
- User education

RESEARCH
- Software development

EXTENSION
- Software development

EXTENSION
- Education and training
- Computer use
- Software distribution and support

CLIENTELE
- Farmers - ranchers
- Home
- Agribusiness
- Education institutions
Though strong links between research and extension, the researchers can strengthen the potential for recognition and credibility of their work in the eyes of the clientele through software development. This also will speed the utilization of research by farmers, ranchers, educators and the private computer application activities.

Specific Role of the Research Scientist

The arrival of the microcomputer as a viable tool that can be put in the hand of the decision maker still does not change the economic, finance, marketing and production problems the decision makers must face. The microcomputer does, however, reduce the previous restrictions that kept certain analytical tools only in the hands of the research scientists. It also must be recognized that the research scientist has often hidden behind the computerized model avoiding real world problems because the farmers and ranchers did not have the "computer power" to use the tools anyway. The microcomputer is going to present a tremendous challenge to both the user who must increase his knowledge to effectively use these more powerful tools and the researcher who must incorporate more of the real world reality into computerized models so that these models will be useful in solving producers' problems.

For less than $8,000, a current state-of-arts, 16 bit microcomputer with appropriate software can handle any analytical tool used in farm or ranch level analysis in the agricultural economics profession. Today's computer use limitation is not memory capacity but the scientist's ability to develop models which will solve real world problems. The primary obstacle is the lack of information on the behavior of biological relationships and the ability to predict environmental and economic changes.
As noted, agriculture software presently distributed and under development draws heavily from previous research done in the experiment stations. Private software developers are not investing in creative research activities to discover analytical tools or biological and engineering relationships. This means that the responsibility of the experiment station to do the agriculture research will not change. Demands for research will expand due to the increased analytical capability of the decision makers using the computer, to the desire for more data, and to the biological and engineering relationships expressed in the computer algorithms.

The researcher's role is particularly important in the problem definition and program design phase of software development. It is in these two phases that the review of previous research, available software, and clientele needs is defined.

User oriented software for farmers and ranchers must be packaged to be comprehensive and requires interdisciplinary involvement. In today's agriculture, the economic and financial components must be included in the management framework in any decision tools if they are to be useful to agricultural producers. The virtues of interdisciplinary applied research are often expounded upon in the land grant system. In practice it is difficult in an environment with increased specialization and departmentalization of activities. Many scientists even question the professional payoff from interdisciplinary applied research activities.

---

1 The six steps in software development are as follows: problem definition, program design, program coding, program verification, program documentation, and program maintenance.
Encouragement of close links among researchers, subject matter disciplines and the extension service via such arrangements as the joint research-extension appointment are complementary, if not necessary for development of producer software. Much of the software development at Michigan State University, for example, is from the extension-research joint appointment staff in an environment conducive to interdisciplinary activities. Researchers have an important role in finding ways to support interdisciplinary-extension-research software development activity. Interdisciplinary research will always depend on the individual scientist and his willingness to work with other disciplines.

A paper by Busse and Vickers suggests that the role of the scientist should involve "...definition of requirements and prototyping of new programs..." These designs could then be provided to the private sector. The private sector would then fulfill as it sees fit the development, distribution, and training role. The land grant university would not be involved in development, distribution, maintenance and support of software. This approach would exclude the applied extension-research interaction phase of software development. No doubt some of the present public software activity operates under this philosophy.

Personal experience indicates that the most rewarding activity in software development is in the actual finishing phases of software development. Field testing and revising software may be one of the more important activities in software development. Little personal satisfaction can be gained if one cannot see the model and design operational. Many scientists would be reluctant to release designs to the private sector when they could not easily control revisions and use of the final product.
Lack of professional recognition for software development would likely be more serious if the researchers were not directly involved in publishing the final software. Current experience suggests the contribution of the experiment station research in software development will likely go unheard if distributed exclusively by the private sector. Distribution, maintenance and support of software by the land grant institutions will be at least as important an activity as the present publication distribution activity. The activity cannot be turned over exclusively to the private sector that necessarily has motivations other than meeting public needs.

The land grant research, extension and education work is justified as "public good" or the idea that society's gain is much greater than any individual could potentially justify from a profitable investment standpoint. This same situation exists today in computer software development distribution and education. Researchers must be selective in the software they choose to develop to insure it does not duplicate efforts in the private sector. All software should be made available to the public which includes the private sector involved in computer applications. Enhancing programs to individual needs must be a responsibility of the private sector, not the public supported researchers or extension staff members.

Supporting Computer Software Development

Computer software development, distribution and maintenance is an extremely costly activity. It consumes tremendous quantities of manpower in all phases. Too often farmers, ranchers, and land grant institution administrators consider that "you are in the computer business" when
the hardware is purchased. Reality is that few users or land grant institutions have sufficient manpower and resources to acquire software to fully utilize existing hardware no matter how limited it is.

Scientists have an important role in educating administrators and fellow scientists on the true cost of computer applications and the necessity of continually seeking the necessary resources to complement hardware acquisitions. Professional programmer support is extremely important for successful development of user friendly software. There is too much poor quality software available from land grant institutions developed by the "computer program hackers" who are really agricultural scientists, not programmers. The professional, trained programmers are necessary to incorporate the "arts" in user friendly software. Providing professional programmer support is an important incentive to get agricultural scientists involved in software development. It is not very cost effective when agricultural scientists or unskilled students are used as computer programmers.

Only a limited number of land grant institutions have support units to distribute and maintain software for the public once it is ready for distribution. Researchers should take an active role in developing these support units so they do not spend all their time supporting software after it is completed, thereby limiting new development activities. Lack of efficient distribution procedure has also encouraged some land grant staffs to use the public supported activity for personal gain by selling software directly or indirectly through the private sector. As noted before, enhancement of software for individual needs should be done in the private sector.
Professional Incentive

Professionally organized review and publication procedures are well established for the biological, engineering and economic research work. The professional recognition and publication of scientific and creative contributions in user software development are, however, severely limited. Until the professions catch up to the realities of computer technology, one cannot expect administrative and peer recognition of the activity. The research scientists have the responsibility to support efforts within professions to raise the status of software to journal articles, a step which will lead to increased incentives for researcher participation. Researchers need to work within professions to establish the evaluation criteria, a mechanism for peer review and a media for dissemination of reviews in a timely and professional manner.

Software does provide a media of communication than can improve the credibility of research in the eyes of the clientele. This activity does, however, require adjustment for many researchers in their present means to establish credibility for creative contribution. The individual scientist must view the potential benefits as greater than the costs to be motivated to support software development. Each scientist has a role to insure that benefits are greater than costs.

Summary

Recent advancements in microcomputer technology present an unprecedented opportunity to the research scientist to facilitate the use of powerful analytical tools and research by agricultural decision makers. Taking advantage of this opportunity requires user software development and educational support for farmers and ranchers.
The biological, engineering, and economic research will continue to be done by the experiment station scientist. The private sector involved in software development and distribution will depend on the public supported research. The private sector will play an important role in meeting individual needs that cannot be served by public institutions. The applied researcher has an important role in defining software needs, designing models, specifying analytical procedures, and facilitating interdisciplinary research-extension involvement in user software development.

User software development and support is a costly activity and requires tremendous quantities of manpower. One of the important incentives to encourage research involvement in software development is providing professional programmer support and a means to distribute software once it is completed by the scientist. In most situations, computer hardware capacity exceeds the manpower and software development resources to efficiently use the hardware. Scientists must strive to get a balance between hardware and software support needs.

The development of professional peer review and publication procedures for software has lagged the advancement in the technology. Recognition of the scientific and creative contribution will require researchers to support efforts to develop evaluation criteria, a mechanism for peer review and dissemination of reviews by the professional societies. Software does provide a media of communication that can increase the credibility of research. Scientists have a role to insure that professional incentives are sufficient to encourage participation. Researchers should encourage efforts to facilitate the regional exchange of research software and education materials to reduce the duplication of efforts.
References


McGrann, James M. "Microcomputers in Farm and Ranch Management Workshop Evaluation," unpublished data, Department of Agricultural Economics, Texas A&M University, College Station, TX, April, 1982.


Software Authorship, Documentation, and Field Testing

Dr. Wallace E. Kilcreas
Agricultural Economics Department
Mississippi Agricultural and Forestry Experiment Station

Preface

A general treatment is given of three broad areas related to agricultural computer software development. All ideas stated should be regarded as opinions of this author.

Introduction

Three interrelated topics are addressed in this paper: authorship concerns who should develop agricultural software as well as who should receive credit for its development; documentation is word communications of what a particular piece of software is about and how to use it; and field testing is an integral part of developing software that will do a good job for its intended audience--i.e., a test of practicality.

Authorship

SHOULD EXPERIMENT STATION PERSONNEL BE INVOLVED IN AGRICULTURAL SOFTWARE DEVELOPMENT?

Historically, experiment stations have participated in areas where (1) there is a grassroots demand for work to be done, (2) the commercial sector is not filling the need, and (3) potential payoffs are high.

Certainly there is a grassroots demand for agricultural software. Those of us involved in software development efforts know this from telephone traffic. In Mississippi alone, an estimated 500 microcomputers have been purchased by farmers. The Mississippi Cooperative Extension
Service (MCES) acting on their own behalf as well as on the behalf of the Mississippi Agricultural and Forestry Experiment Station (MAFES) have distributed over 2,000 orders\(^1\) for programs (many orders are for 5-10 programs) during the past year.

The commercial sector is currently providing some agricultural software, but the targeted markets thus far have been larger machines, micros and minicomputers costing $10,000 or more. Some help is coming from custom programmers, but their services are very expensive (custom development of one program generally costs from $1,000 to $20,000) for small operations. Also, for many applications, Experiment Station personnel are the only people qualified to develop the needed software.

Potential payoffs are, in my view, very high. Good record keeping, as well as complex analytical techniques and "spot" problem solving, are all easily "handled" on microcomputers. Also, current research findings can be packaged and transferred to the farmer owning a micro as simply as mailing him a diskette. This ultimately means increased profits for the agricultural sector.

I, therefore, believe that agricultural software should be developed by the Agricultural Experiment Stations because (1) it has grassroots support and it needs doing, (2) such development represents introduction of new technology, a classic research role, (3) many programs needed are too complex and lengthy for Extension Personnel who also have clientele responsibilities, and (4) the Experiment Stations have the scientists and the computing expertise to get the job done.

\(^1\) A copy of the program library currently being distributed by MCES is included in the Appendix.
Successful development of agricultural software by an Experiment Station is generally multi-disciplinary. The minimum number of disciplines involved include a technical discipline such as agronomy, plus the computer science discipline. In many cases, more than two disciplines are needed. For example, an animal management software package which monitors the production process, then determines profitability of a specific animal management system, necessarily embraces the animal science, computer science, and agricultural economics disciplines. In addition, help from other groups might be needed in the development effort: animal producers who would help in both planning and testing phases and Experiment Station animal management specialist whose general applications knowledge is sometimes invaluable to the development effort.

In the "brainstorming" phase of software planning, the agricultural technical disciplines, Extension specialists and farmers/farm managers should all be involved. Plans need to be carefully reviewed by computing personnel to estimate the resources needed to develop the software and to clarify tradeoffs between planned software features and computing and people resources needed to program the software system. After agreement is reached on software design, a careful computer systems analysis should be done. Further communication with technical disciplines is frequently needed during this phase. It is only at this point that coding should begin. It is sometimes very efficient for much of the coding to be done by less experienced personnel rather than the computer system designer.

Inter-disciplinary communications are needed throughout program development. Meetings of the program developers should be held on an ad hoc basis as coding is done.
Once coding is completed and the program is functioning as designed, follow-up inter-disciplinary meetings are needed to identify changes or additions needed to make the program practical. When this phase is complete, the program should be ready for field testing.

PRIOR TO ITS DEVELOPMENT, A CONSCIENTIOUS EFFORT SHOULD BE MADE TO DETERMINE IF THE SOFTWARE IS NEEDED.

First, the potential user group should be defined. This group would likely be farmers or agribusinessmen, but it might include other researchers, Extension specialists, administrators, or government workers.

Second, potential benefits from the software should be determined. Feedback from potential users (Extension directed meetings, direct contacts with producers, meetings with Extension personnel, regional meetings such as this one, communication with administrators) is particularly useful during this phase of program development. Exact determination of potential benefits is usually impossible when new technology is being developed; but general criteria such as the economic importance in your state of the subject area for which the program is being developed, the number of producers involved, and the potential payoffs for a given producer are all good general criteria for estimating potential benefits.

Third, some time should be spent in examining the software available. Considering the lengthy and expensive development period for good agricultural software, there is no logical reason for duplicating software already in existence. The only way to determine if existing software packages will satisfy perceived needs is to obtain and test them. Duplication of subject areas is sometimes needed because existing packages are not germane to the problem(s) at hand.
Finally, the defined need should be articulated and a short proposal written and discussed with all groups concerned (administrators, researchers, and potential users) prior to detailed planning and program development.

**CREDIT SHOULD BE GIVEN TO WHOM CREDIT IS DUE**

Those making major inputs into the planning, coding, and documentation of the software package should be included as authors. Others contributing to the effort should be included in acknowledgements.

Special situations may arise where those outside the Experiment Station "umbrella" should be included as authors. Such personnel might include Extension specialists and farm managers.

**Personnel having both knowledge in the subject area being programmed and computing expertise are extremely valuable.** It is essential that theoretical subject knowledge, computing expertise and detailed applications knowledge all be integrated when developing software for usage by farmers and agribusinesses.

**HOW IS PRIDE OF AUTHORSHIP PROTECTED?**

Well, there ain't no sure protection, folks. Sure, we can copyright the software and documentation. Current copyright laws offer little real protection against those who slightly modify software, then claim it.

A well written software publication is the authors' best protection against software piracy. If those of us in the Experiment Station system do a good job of publishing software documents (technical bulletins, etc.), then distribute the software free (distribution costs can be added if desired), we will minimize software piracy problems. Software may be
distributed via the Extension Service or via other means such as an editorial department.

The author's name and organization should be outputted (screen, if possible, or printer) at the beginning of each "run" of every program developed. Software "pirates" must consciously remove these lines (thereby creating evidence of intention to steal) to claim the program as their own.

Researchers doing software development should regard computer programs and documents to be just as important as any other publication. Publications should be regarded as the life blood of career development. We should strive to make program and documents as good as we can make them before we are willing to attach our names to them.

It is imperative that administrators who want researchers to participate in computer software development reward them by accepting computer programs and accompanying documentation as bona fide publications.

Documentation

Documentation, or words written to communicate what a piece of software is about, is absolutely essential for all software (I am using the word "program" in this section in lieu of "software systems" in the interests of brevity) that is to be used repetitively.

Documentation is needed by at least the following groups (the list is not exhaustive):

1. The program developers themselves. Even the best programmers forget details concerning large systems with 6 months of their development.
(2) Successors to the original developers. This group includes those who maintain programs as well as those charged with updating them. Documentation is absolutely essential for continuity.

(3) Administrators. It is very helpful to know what your people are doing.

(4) Potential users. Without communications articulating the many strange and wonderful things that a program can do, it is impossible for others to know if it will solve their problems.

(5) Users. It is extremely important that enough details be provided to program users to allow smooth program operation.

**DOCUMENTATION SHOULD BE TAILORED TO THE AUDIENCE**

In order for documentation to be germane for each of the groups mentioned above, at least three levels of documentation are needed:

(1) A technical bulletin detailing why the program was written; its capabilities and limitations; technical information such as the language used; machine(s) that is available for; memory and disk space required for various sizes of operations (example: in animal management, we might specify the amount of storage needed for, say a 500 animal herd, a 1000 animal herd, and a 2000 animal herd); printer required, etc; and procedures for obtaining a program.

The technical bulletin is incomplete unless it details each feature of the program, explaining exactly what the program will do. For example, if it is a menu-driven interactive program, a detailed explanation of what each menu item is designed to do is essential. Most large programs are broken into several modules or subroutines. A discussion of the functions of each module is sometimes very useful.
Data needed to operate the program should be detailed along example reports. Administrators, other researchers, Extension specialists, prospective users, and current users of the program will all make good use of such a bulletin.

Background technical material that is needed for development or computer algorithms, published or unpublished, should be referenced and appropriate credit given. Such material might include "older" programs, technical bulletins, vendor publications and/or standards (an example of standards might be productivity indices for beef or swine).

(2) A user's guide which explains how to operate the program on each machine for which it is available.

The user's guide should include example runstreams detailing how the program performs. This information is easily generated on many microcomputers, via screen dumps and options which echo each line from the screen to a printer.

The amount of detail needed depends somewhat on the user group for which the program is developed. Programs to be operated by farmers and agribusinesses should contain a rich variety of operational examples.

A well written user's guide helps to introduce computer technology in new areas, minimizes frustration on the part of even seasoned computer users, and aids in the detection of errors in the program.

Program users and maintenance programmers assigned to the program can be greatly benefit from a user's guide.
(3) Documentation intended for aiding in program development and maintenance.

Flow charts, comments "sprinkled" throughout the program, definitions of important variables, detailed organization and structure of all files, program listings, and detailed descriptions of program modules/subroutines/procedures, all fall into this category.

Flow charts can be very helpful for complex system development if they are structured at several levels of detail. For example, one might have a general flow chart for an entire system. Subroutines defined only as "processes" in the general flow might each have separate, more detailed flow charts. As many levels of detail as needed to show both the "big picture" and to detail exact program "flow" can be used.

Comments are sometimes very helpful in helping program developers remember what they did after it gets "cold" and to assist maintenance programmers and those involved in reprogramming. A warning: Comments are very expensive when working in the interpretative languages that are so popular on today's microcomputers. These comments take up comparable memory space to "executable" code.

Definitions of variables are very helpful when making program modifications. In particular, subscripted variables should be defined along with the ranges of subscripts used. One can go overboard by defining every variable in a program. Usually, the definition of all subscripted variables together with definitions of all simple variables that are important from the standpoint of program control will suffice.
Specification of file structures is extremely important. A schematic (graphic presentations are sometimes helpful) of all components of each file are essential.

Up-to-date program listings are essential for program maintenance. Programmers are notorious for not maintaining up-to-date listings, but listings are very important.

Detailed descriptions of each program component allow "compartmentalized" program maintenance and debugging.

THE REVIEW PROCESS

Peer review should be the first step of the review process. Three to five reviews should be chosen. Care should be taken to include reviewers in each discipline involved, plus at least one generalist.

The review process should include a demonstration of the program developed. Also, the program should be made available to each reviewer so that he can make his own test runs to evaluate it.

Ideally, programs having potential regional utility should be reviewed outside the developing Experiment Station. This committee could develop a forum for this. A group of program developers/reviewers could be defined. Only those willing to do regional reviews would be allowed to submit programs for regional review. Two regional reviews in addition to the in-house review should suffice.

All reviewers should be recognized in presentations of the work and in acknowledgements. Also, lengthy reviews are frequently suitable publication material.
Field Testing

One of the most neglected areas of program development is that of field testing. This is particularly true of programs developed for usage by farmers and agribusinesses. Contrary to popular belief, researchers and programmers do not always know what is needed and what will work best in the field.

Successful farmers and farm managers should be contacted during the planning stage of the program. If possible, several levels (small, medium, large) of operations should be involved. A few (two or three) veteran producers can give a wealth of information when the program is being planned.

A program testing group made up of five to eight producers should be set up for the purpose of testing software after it has been developed. Usually, in a group of six selected producers, there will be two who lack either the time of the inclination to spend the many hours needed to give helpful feedback on the new program operations.

Involvement of Extension specialists and agents in the field testing phase of program development is absolutely essential. Extension personnel can aid in the selection of the test group, schedule meetings with them (at least 2-3 test group meetings are generally needed) and provide applications advice that only someone working with many production units can provide.

Test group participants should be carefully selected, since they must necessarily have access to large software systems which are unpublished. Their advice should be carefully heeded, but final decisions must rest with the developer.
Software should **not** be released for field testing until all known errors have been corrected. After field testing has been completed, all errors detected during field testing should be resolved **prior** to general program distribution.

A follow-up sheet should be attached to distributed software to give users the opportunity to suggest changes **after** using the program for a given period in the field and to "feedback" changing needs.

Extension and test group personnel not included as authors should be remembered in acknowledgements.
APPENDIX
AVAILABLE SOFTWARE FOR MICROCOMPUTERS

COMPUTER APPLICATIONS AND SERVICES DEPARTMENT
MISSISSIPPI COOPERATIVE EXTENSION SERVICE
JULY 1, 1982

The following microcomputer programs are available from the Computer Applications and Services Department, Mississippi Cooperative Extension Service. This listing gives the program title, memory required to execute the program, diskette storage requirements, and a short description of the program.

These programs were developed on TRS-80\textsuperscript{1} microcomputers using Basic or Fortran languages and are available on diskettes or as source listings at no charge to the requesting party. The party requesting these programs must supply a 5½" or 8" soft sectored diskette upon which the programs may be copied. Individuals having equipment other than that listed above may obtain source listings of these programs for conversion to their computers.

As a rule, more than one program will fit on each diskette. In order to more efficiently use diskette space, the user may determine the number of programs that will fit one diskette by summing the diskette storage requirements for all programs requested. Any combination of programs will fit on a diskette so long as the total diskette storage requirement (listed under each program name) does not exceed the appropriate diskette's storage capacity (45, 180, 90 storage units for TRS-80 Models I, II, and III, respectively). This rule should allow the user to determine the total number of diskettes to send with each request. A software request form is provided on the back page of this pamphlet.

Mississippi residents may obtain these computer programs by contacting their local county Extension office.

Residents of other states must request these programs through the state office of their Extension Service. Requests directly from individuals or county extension personnel will not be accepted.

\textbf{Timber Evaluation (TIMBER)}

\begin{tabular}{ll}
Diskette storage requirements & 10 units \\
Memory requirements & 11 K\textsuperscript{2} \\
\end{tabular}

A technical forestry program designed to perform an economic analysis based on timber cruise data. The program requires explicit input of cruise

\textsuperscript{1}TRS-80 is a registered trademark of Tandy Corp., Fort Worth, TX.

\textsuperscript{2}Actual program size, however, a disk system requires a minimum 32K machine.
and a product identification data which are used to compute the economic value of a given timber tract. Raw data and final reports are printed.

The program was written by Wallace Killcreas and Mike Argo, Mississippi Agricultural and Forestry Experiment Station.

**Linear Programming (LP)**

Diskette storage requirements 45 units  Memory requirements 32 K (min)

A program to optimize use of limited resources (i.e., minimum cost maximum profit problems). The program may be used for basic farm planning, feed mix, or fertilizer mix problems. This program is mathematically oriented and assumes the user has a basic knowledge of linear programming. The program is capable of evaluating a 41 x 41 matrix on a 48 K machine or a 30 x 30 matrix on a 32 K machine. This program is available for Model I only.

This program was written by Wallace Killcreas and Verner Hurt, Mississippi Agricultural and Forestry Experiment Station.

**Market Pack (MARKPACK)**

Diskette storage requirements 6 units  Memory requirements 6 K

A program which considers production cost, interest rates, storage cost, and opportunity cost to evaluate strategies of selling storable commodities at current market prices or storing and selling at a future date.

This program was written by Warren Couvillion and Richard Levins, Mississippi Agricultural and Forestry Experiment Station.

**Farm Record System (RECORDS)**

Diskette storage requirements 45 units  Memory requirements 32 K (min)

A record keeping system designed specifically for the farm business. The program allows the user to define records according to enterprise or transaction. The program also has provisions for capital investment records including a depreciation schedule compatible with IRS forms.

This program was written by Mike Argo and Wallace Killcreas, Mississippi Agricultural and Forestry Experiment Station.

**Discounting/Compounding (DISCOUNT)**

Diskette storage requirements 4 units  Memory requirements 6 K

This program computes discounted values of a future sum or compounded future values of a present sum. The program also computes internal rate of return.

The program was written by Wallace Killcreas and John Waldrop, Mississippi Agricultural and Forestry Experiment Station.
Net Return Equating (NET)
Diskette storage requirements 4 units  Memory requirements 4 K

This program calculates the break-even price for an alternative enterprise as a function of the price of the primary enterprise and the respective variable costs and yields of the two competing enterprises. The program prints two tables comparing each enterprise to the other over a range of prices and yields.

Cost and Returns of Major Crops (BUDGET)
Diskette storage requirements 45 units  Memory requirements 7 K

Estimated costs and returns per acre for primary Mississippi crops are stored as a data file. The user may select any of the crops to obtain an estimated cost and return for each. The user may then modify the cost or income to personalize the program. A printed output of the original or the modified budget is available for the producer's own use. This program is not currently available for Model II.

Loan Calculator No. 1 (LOAN)
Diskette storage requirements 5 units  Memory requirements 5 K

This program calculates principal, interest rate, length of loan, number of installments per year, or payment amount when all other variables are supplied. The total interest for the loan is computed and printed. The program provides an option of printing a schedule of payments.

This program was developed at Oklahoma State University.

Loan Calculator No. 2 (LOAN 2)
Diskette storage requirements 3 units  Memory requirements 2 K

This program calculates the principal and interest payments for a loan. Program entries include principal, interest rate, length of loan, and number of annual installments. This program assumes a constant principal payment amount. Output includes a payment schedule, total interest paid, and amount of principal paid monthly.

Budget Analysis (SPEEDY)
Diskette storage requirements 11 units  Memory requirements 14 K

This program allows a person to input basic data concerning personal finances and receive an estimated budget based on standardized coefficients.

The program was developed at Clemson University.
Auto Trade Analysis (CAR)
Diskette storage requirements 4 units Memory requirements 4 K

This program analyzes user supplied data on a presently owned car and a replacement car. Cost of gasoline, miles per gallon, and annual mileage are considered to develop a cost of operation for a specified period of time. Minimum cost of operation is the decision variable as to trade or keep the present car.

Foliar Fungicide Recommendations for Soybeans (FUNGI)
Diskette storage requirements 11 units Memory requirements 21 K

This program, based on the county point syste, is used to determine economic feasibility of a foliar applied fungicide for soybeans. The program prompts entry of rainfall data, yield estimates, yield history, and week control data. Based on these entries, the program makes a positive or negative statement regarding the economic feasibility of application of foliar fungicides.

Season Average Price—Cotton, 1950-1979 (CS)
Diskette storage requirements 3 units Memory requirements 1 K

This program will print average yearly cotton prices for Mississippi for all years 1950 through 1980 as well as an average of these prices over any specified range of years.

Sprayer Calibration (SPRAVER)
Diskette storage requirements 5 units Memory requirements 5 K

This program allows the user to select one of three common methods of sprayer calibration. The user supplies data regarding speed of sprayer, row or nozzle spacing, and volume of spray from each nozzle. The program also allows the user to determine the quantity of herbicide necessary for each filling of the sprayer tanks. The program is capable of handling multiple chemicals when a tank mix is to be used.

Muscadine Production (MUSK)
Diskette storage requirements 10 units Memory requirements 12 K

This program is based on an Extension publication, "Muscadines for the Home Vineyard," and guides the user through instructions in training in vines and soil fertility. This program is not available for a Model II.
Dairy Ration Balancer (DFOR)
Diskette storage requirements 9 units Memory requirements 11 K

This dairy ration program is designed to balance rations to meet a lactating cow's requirements for crude protein, TDN, calcium, phosphorus and minimum dry matter consumption. Using forage as a base, recommendations are made indicating the pounds of forage, pounds of supplement as well as the percent crude protein, TDN, calcium and phosphorus of the supplement. Multiple forages may be used with or without limits on consumption (at least one must be free choice). Inputs required include body weight, pounds of milk produced and the percent butterfat.

Housing Purchase (HOUSING)
Diskette storage requirements 16 units Memory requirement 19 K

This program considers a number of investment and expense items associated with home ownership. From this evaluation of personal income, purchase price, ownership expenses, interest rate, and tax rates, the computer prints a report concerning feasibility of ownership of a given home as well as the long-term summary of payments and investment benefits from owning this home.

This program was developed at North Carolina State University.

Land Leveling Cut and Fill Estimate (LANDFORM)
Diskette storage requirements 11 units Memory requirements 32 K

This program is designed to compute cut and fill estimates for on-farm land leveling projects. It requires the user to enter survey data and establish field parameters for slope and benchmark elevation. These parameters may then be varied, based upon previous results, to allow the user to estimate a suitable final field design. The program prints the results of cut and fill calculation in the form of a field map.

Crop Lease Break-even Analysis (LEASE)
Diskette storage requirements 13 units Memory requirements 18 K

This program analyzes the costs and returns of competing crops to evaluate a crop lease. The program computes costs and returns for one or two crops for the leasee and leasor. Detailed cost tables are produced to show the cost per acre, per unit costs and total costs for all specified inputs. A second group of tables will include net returns per acre for three yields and three prices. These net return tables are constructed for the leasee and leasor. A final set of tables will indicate the break-even conditions for each crop. The break-even point will be in terms of price and units of production.
Cotton Production, Ginning, and Sales Records (COTREC)
Diskette storage requirements 16 units Memory requirements 48 K

This program was designed to record, sort, and summarize cotton production, ginning, and sales records. The program is menu driven and allows the user to build and edit record files and enter sales data for an individual bale. Cotton sales price may be entered as a net price per cwt or as a gross price with net price computed on the basis of standard discounts for quality and grade factors.

Catfish Production Management Records (CATFISH)
Diskette storage requirements 45 units Memory requirements 48 K

A program designed to store, retrieve, and summarize catfish production data. The program maintains feeding records, stocking rates, stress information, and prints summaries for each of these on a pond-by-pond basis. The program projects feedings rates and expected harvest dates on the basis of user supplied information regarding conversion rates.

Reports include: Feed schedule (one week); Extended feed schedule; General pond information; Stress information; Year-to-date fish summary; Year-to-date feeding summary; Future harvest projections; and General producer information.

The program was written by Lee Fouche, Wallace Killcreas, and John Waldrop, Mississippi Agricultural and Forestry Experiment Station.

Planning Your Estate (ESTAX)
Diskette storage requirements 11 units Memory requirements 32 K

This program is designed to help a user plan his/her estate. The program solicits information concerning the appropriate estate and then calculates the Federal and Mississippi estate taxes due. All tax information used is based on current tax data. Report options include single estate (widow or widower), married with results showing the assumed first estate followed by the assumed second estate (ex: husband dies then wife dies) or married and with two sets of results, one set as described above and another with the order reversed (ex: husband dies then wife, wife dies then husband).

Agents Mailing List (AGLIST)
Diskette storage requirements 8 units Memory requirements 32 K

This program was designed to store names, addresses, and enterprise/interest data and use these data codes to print various mailing list/labels. The program allows the user to change sort categories or sub-categories to meet his individual needs.
Swine Production Management Records (SWINE)

Diskette storage requirements 45 units* Memory requirements 48 K

A set of microcomputer programs developed for swine production management data collection and analysis. These programs will interactively build, update, and summarize individual sow records and feeder group records. Reports produced should be useful for production and inventory management. Programs support data collection and manipulation for each phase of the hog production cycle including breeding, farrowing, weaning, and culling. Farrowing projections are developed for up to 8 weeks in the future. Target oriented statistics are reported which estimate the efficiency of breeding, farrowing and weaning. Sow productivity indices are reported for more effective culling. A sow culling report is also produced.

A group management program monitors feed consumption, death losses, and animals saved for replacement. Feeder pigs are monitored in up to 5 user defined production areas (examples: cages, nursery, grower and finisher). Inventory and marketing reports are produced for animals being produced for market. Feed analysis, including detailed reporting on quantities consumed by feed types within production areas, and calculation of feed conversion ratios, is also available.

The programs were developed and debugged through collection and processing of "live" herd information for a 1,200 sow herd over a two year period.

These programs were developed by Wallace Killcreas and Ryan Hickel, Mississippi Agricultural and Forestry Experiment Station with the cooperation and help of the Deerbrook Company, Brooksville, MS and the Mississippi Cooperative Extension Service.

*These programs require a 48 K Model III TRS-80 with two disk drives or a 64 K Model II TRS-80 with one disk drive. The programs are not currently available on a Model I TRS-80.

Rice Growth and Management Predictions--1982 (DD50 1982)

Diskette storage requirements 13 units Memory requirements 11 K

This program predicts the date of occurrence of certain critical events in the life cycle of the rice plant. These predictions may be based upon: (1) effective planting date; (2) date of emergence; (3) mid-season nitrogen application, or (4) the date at which 65 percent heading occurs.

The program uses an average of 30 years weather data (converted to DD50 heat units) to make these predictions. Weather data for the North, Central and South Mississippi Delta counties are contained in the program.
MISSISSIPPI COOPERATIVE EXTENSION SERVICE
Software Request Form

Return this completed request, along with your diskettes, to your local county Extension office.

I am enclosing __________________ diskettes. I would like to have copies of the programs that are circled in the following lists.

<table>
<thead>
<tr>
<th>TIMBER</th>
<th>SPEEDY</th>
<th>HOUSING</th>
<th>LOAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>CAR</td>
<td>LANDFORM</td>
<td>LOAN2</td>
</tr>
<tr>
<td>MARKPACK</td>
<td>FUNGI</td>
<td>LEASE</td>
<td>DD50 1982</td>
</tr>
<tr>
<td>RECORDS</td>
<td>CAS</td>
<td>COTREC</td>
<td></td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>SPRAYER</td>
<td>CATFISH</td>
<td></td>
</tr>
<tr>
<td>NET</td>
<td>MUSK</td>
<td>SWINE</td>
<td></td>
</tr>
<tr>
<td>BUDGET</td>
<td>DFOR</td>
<td>AGLIST</td>
<td></td>
</tr>
<tr>
<td>ESTAX</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I am requesting these programs subject to the following conditions:

(1) The Mississippi Cooperative Extension Service assumes no liability or responsibility to any person or entity for damage caused or alleged to be caused, either directly or indirectly, by computer programs furnished thereby. This waiver includes, but is not limited to, interruption of service, loss of business or anticipatory profits, indirect, special or consequential damages, or delayed delivery of this or any program.

(2) Programs will be copied onto as few diskettes as possible. Requests for one program per diskette or for selected combinations of programs to be placed on a specific diskette will be disregarded.

(3) Programs may be removed from the library without notice.

________________________________________
Signature

I own/operate the following microcomputer equipment:

Brand _________________________________

Model _________________________________

Type Printer ___________________________

Number Disk Drives _____________________

Please return this request to:

Name _________________________________

Address _______________________________

City, State, ZIP ________________________
Quality Control, Distribution and Maintenance of Software

Steve Welch
North Central Computer Institute

My prepared remarks refer to the issues of software quality, distribution and maintenance; however, I realize there is considerable interest in other aspects of the North Central Computer Institute as well. I want to start by referring to several handout materials. The first document (Appendix A) describes the documentation standards adopted by the North Central Computer Institute. An important point to recognize is that it is not sufficient to merely have a standard for documentation. There also must be an accompanying set of codified procedures concerning how that standard is going to be used. Otherwise, there is no particular incentive for people to follow this standard. If a person writes a set of documentations according to some standard, what is he going to do with it? What happens to the document? Does it get to anyone? We have talked about publishing software in the form of department reports, but we could have a department report with an excellent standard way of documenting it and still not have a mechanism for distributing that report. Then the standard has not helped at all. To deal with this problem, the North Central Institute has developed a procedure for how standard software documented in standard form is to be handled (Appendix B).

For distribution purposes the institute has created the North Central Computer Institute Software Series. In essence this is a peer review journal intended to be exactly the same as any other journal that you might have from a scientific society. The linkage between the documentation standard
and the software series is that once a piece of software is documented in standard form, it is then submitted as an article. It goes through a normal review process and appears as an article in this publication series. The distribution mechanism is the peer review journal.

Another document describes our program of regional software development (Appendix C). In many situations it is beneficial from a cost point of view to develop software that can meet the needs of more than one state. We are entertaining proposals from staff at our member institutions for regional software projects. As those projects come in, they are given a rough screening. The first step is to conduct a feasibility study to determine beforehand if there is sufficient reason to go to the effort and expense of developing this software. The study is conducted jointly by the NCCI and the project team making the proposal. The study considers the software audience and the expertise that will be needed to complete the project. If everything seems satisfactory, the next step is to secure funding. The NCCI does not completely fund regional software development projects. We fund travel and the expense of providing accommodations to bring together members of the project team. We make great use of electronic mail, conference calls and other types of teleconferencing media. We fund that kind of activity but we do not pay salaries, computer time and other main line expenses. We do work with the project team to secure funding, and we go to as many different sources as we have to in order to obtain those funds. Once that funding is secured, the project team goes forward and implements the software. The team also has input from an advisory committee. This is a device for increasing the input and breadth of representation through the design phase. Each state that is not directly participating in the software development project has the opportunity to designate one member or
representative who will assist or provide input as the project progresses. The only request that the NCCI makes relative to the output of the project is that the results be documented according to NCCI standards and distributed through the peer review process.

In terms of software quality, I suppose there are as many definitions of "quality" as there are definers. Software quality is basically an attribute of software that is proportional to the degree to which a user's experiences are free from unexpected and/or undesirable effects. This definition focuses on the user. If a user is operating a program and all of a sudden the program blows up and dumps out unintelligible garbage, then that is a good example of poor quality. There are other examples that also fall under this definition. If the program does something unexpected because the documentation is not clear and has led the user to believe something false, then this is equally low quality software. If he looks at the screen and says this program is not doing what it is supposed to, this error reduces the overall quality of his experience. The worst thing that can happen and the worst infringement of quality is when the user runs this program and gets a wrong answer, but he is not prepared to argue with the answer because he thinks it must be right. He does not know that the reason the answer is wrong is that the software is not up to date in terms of current theory. The software did not embody the best research results we have. This leads to unexpected outcomes well after the user has turned off his terminal and gone away because when he tries to use that answer, everything will go awry. Software quality is not something that stops with turning off the terminal. It carries through the use of the results of that program.

One of the specific components of quality is the significance of the subject material. If the program addresses a very narrow aspect of the
problem or applies to a narrow set of circumstances that is seldom if ever encountered, that tends to reduce the quality of the software (see Appendix B, Sec. V, #A2). As software programs are reviewed in the peer review process, specific significance of the subject material is one of the considerations of the reviewers.

A second component of quality is sound underlying theory (see Appendix B, Sec. V, #A3). Programs should apply up-to-date theory relative to the particular problem-solving situation. This is another issue that we consider in our review process before incorporating the software into our software series. There are two aspects to a software program: (1) the technical subject-matter scientific base for that program and (2) the coding and techniques that go into the production of the program. Computer science is a science and there is a lot of theory concerning how to write programs. One ought to make them modular and one ought to follow certain approaches and do certain things to contribute to the understandability of the program or the ease of writing or the ease of transferring. This is called software engineering, and it is the third factor we consider in the review process (see Appendix B, Sec. V, #A4 and #A5). Does the software involve not only the best updated subject matter material but also the best software engineering? Is it written in a way that enhances readability, transportability, etc.?

A fourth component of quality is good documentation. The programs need to have documentation relevant to the expert who has to oversee the use of the program (see Appendix A, pg. C-1, I and II) and to the prospective user of the program (see Appendix A, pg. C-1, III). If we are talking about research programs, these may be the same person. In extension it will typically be separate people. One will be the end client, and one may be
the technical specialist in that area. The program also should be documented
to meet the needs of the programmer who creates and maintains the program
(see Appendix A, pg. C-2, IV and V). The institute tends to concentrate
more on documenting for the overseer and the programmer than for the end-
user. The reason is that our task is not to serve the needs of the farmers
directly but to provide information and services that enhance the ability
of our member institutions to meet the needs of their clientele. If there
is a user's guide for a particular program, for example, this is not part of
the standard documentation. What is part of that standard documentation is
simply a notation that such a document exists and where it is available.
The user oriented information we include is much more oriented toward the
prospective programmer of that program in order that he will be aware of the
program capabilities in the form of sample runs.

If those are some of the elements of quality, what are some mechanisms
for insuring software quality? The key element is broad representation
during the design and implementation of the program (see Appendix C, II E).
This provides insurance against overly narrow problem definition or range
application. If we have representation not only across state lines but also
across groups (certainly technical specialists and users), we are going to
get a better and higher quality of software.

The second means of insuring software quality is thorough testing prior
to release of the program (see Appendix A, pg. C-2, IV; also Appendix B,
Sec. V, HA7). Complete testing is the only way to detect and eliminate
program errors. It is important to realize, however, that only the most
trivial programs will be error-free at the time of release. This is one reason
on-going maintenance is necessary.

The final mechanism for software quality control is external review
of the entire software development and documentation process (see Appendix B,
Sec. IV, V, and VI). First of all, just knowing that a review is in his future can often lead a software developer to do a better job and a more thorough job of testing. Secondly, a review can function as part of this testing process to detect problems that have gone unnoticed in spite of a developer's best efforts. It can also suggest enhancements for the program that the developer may not have considered. A final product of the review concerns authorship and giving credit to the people who do the work. At Kansas State University we had a College of Agriculture policy subcommittee on computers that included three associate deans and a selection of faculty from the associate professor level on up. We sat down one day and had a three-hour argument concerning the problem of receiving credit for authorship of a computer program in the same way that a researcher receives credit for a journal publication. What we finally came to was a realization on the side of the professional staff that we had not taken up the cause of developing vehicles for this peer review process. This is our fault. On the administrative side, however, we often do not have clear, concise statements affirming that this does count as a research publication. These are two mutually reinforcing defects because if an administrator does not have a specific process that he can point to and refer to as something good to do, it makes it difficult for him to say that this activity is worth the effort and expense. Kansas State is trying to remedy both of these situations by availing themselves of the NCCI software series and yet coupling it with a distinct, formal announcement through the administration that this is a vehicle for publishing. Other states have gone further than that. In Illinois, for example, they have announced as official policy that no piece of software will be incorporated into their delivery system unless it has been documented by NCCI standards. So there is variation from state to state,
but the point is that in each case we are establishing a procedure for review and dissemination of information accompanied by explicit administrative statements that this is something that is worthwhile and that the author will be given publishing credit. The review is, of course, a key element in that entire process.

Once the software is tested and reviewed, there are a number of different ways to distribute the software. First of all, the software may not even be distributed at all but users in other areas can access the program through long-distance, interactive time-sharing hookups. This is particularly useful for those programs that operate on large mainframe computers. It is less applicable to software that resides on micros. Systems like AGNET, for example, are serving a software distribution function because they are making that software available. An advantage is that there is no particular programming necessary at the receiving end, but there are some disadvantages. Users have to somehow be informed that a given piece of software exists and has relevance to them. Then remote computer accounts must be established for users and the university staff to be taught the idiosyncrasies of several different computers. This complicates training efforts.

A second method of software distribution is in the form of computer codes, usually with some type of accompanying documentation to describe how to use the program. The distributed materials are computer language programs on machine readable media (tapes, cards). This is the most frequently used mechanism available at the present time. An advantage of this method is that no programming is required to the recipient if the hardware and software match exactly. A disadvantage is that the program is in the vocabulary of the machine as opposed to the vocabulary of the subject-matter expert who has to evaluate it or adapt it. Software can seldom be transferred without
some form of modification. This is the major defect of this method of
distribution. Too many things have to match. If dialects of language
and all the different ways to run basic programs do not match then
the software will not transfer. Data bases are organized in different
ways. If these bases do not match, then the software will not transfer.
Mainframe computer systems are generally handcrafted, locally adapted,
modified, and enhanced types of systems that are almost unique. On
micros there are some standard operating systems that work on 8-bit
machines. One would hope that we could get some standardization on
16-bit machines, but that is not clear at the present time. Hardware dif-
f erences or the problems of disk incompatibility are the major impediment
of microcomputer software. If any of these things go wrong, the software
will not transfer and will have to be modified. Modification is an expensive
process because you have to start by figuring out what the program is
supposed to do. Since the program is not in the subject matter vocabu-
lar y but in the machine vocabulary, this can be a frustrating and difficult
process.

While transferring computer codes is perhaps the most commonly used
method of software distribution, it is also the most expensive. The NCCI
does not support this mechanism for software transfer. We are not in
the business of moving codes from one place to another. Instead we
support accessing the program through time-sharing hookups or trans-
ferring the basic methods and procedures (algorithms) of the program.
We do not move the code but we do distribute materials describing the
scientific background and exact procedures used by the program to
obtain its answers. This provides everything a person needs to know
to write that program from scratch. We believe this is cheaper in terms
of the total life cycle of the program than receiving a computer code with generally poor documentation that must first be deciphered and then modified. There is another advantage to this method of transfer. By conveying basic procedures, program logic, and algorithms, we have material that is more similar to the normal format of journal articles as opposed to just endless lists of computer codes.

The key to this entire system of regional software creation and distribution is the software standard of documentation. There is two elements to this documentation (see Appendix D). The first element is what we call a software inventory form or a software summary. This is a two-page form (see Appendix A for copy of inventory form) that gives a brief summary of the program. It records the fact that a given piece of software exists and who the contact people are. It gives information about hardware and software required to run the program. It tells what types of guides are available. This summary sheet can either stand alone as a document or serve as a cover sheet for a more detailed description of the program design, basic methods, equations, algorithms. This information is aimed at two specific audiences: (1) the technical specialist who will evaluate the program in terms of its scientific merits and its applicability and (2) the programmer who will have the task of implementing the program should it be deemed desirable to have on the local system.

In discussing distribution and documentation, we are actually concerned about two different bodies of software. First of all, there is the body of the software that exists at the present time; secondly, there is new software that becomes available as it is completed. We simply cannot expect that authors of existing software are going to go back and document all that material. On the other hand, we want to provide some breadth
of coverage in terms of letting people know what is available. By using the short inventory form or software summary, most of the existing software will at least be recorded. All software summaries or inventories will be stored in an on-line retrievable data base system regardless of whether or not they are accompanied by design documentation. This system is operational now. People who want to know what software is available can log in on the terminal—do a keyboard search—and find out what is available just as if they were to go to the library and do a computerized literature search. In order to give breadth of coverage, this software summary or inventory is not in any sense reviewed. It just goes into the system. This introduces an interesting point that should be made concerning the problem of trying to achieve both the breadth of coverage and quality assurance. In order to have high quality programs, the programs must be reviewed. This means that some are not going to pass and the result is a loss in breadth of coverage. The more stringent the review, the worse the problem. We feel at NCCI that by having both reviewed and unreviewed components in this system, we can address both ends of that spectrum. When detailed design documentation comes in, it is subject to review. Thus, through the inventory, we have a broad data base of available software plus a vehicle in the form of the software series to distribute information in detail about very high quality products. These are the two elements of documentation. We think the major source of "articles" in the software series will come from new software products that will then be peer reviewed and appear in the software series.
This software series is in every sense a professional journal. We have an editor and a complete review process (see Appendix B). The editor is Dr. Eldon Fredericks in the Agricultural Information Section at Purdue University. In fact, the only real difference between the type of society journal most researchers are used to and this journal is that the software series will be published in microfiche form in order to save on publication costs. This gives us a per page charge of 35 cents. It also gives us the flexibility to reproduce almost any type of material a person would want in describing a program except a dynamic screen display where the thing blinks and differences pieces come on and off.

Once software appears and is implemented at some receiving or destination site (see Appendix D), we have a mechanism for feedback either from the user of the software or a technical staff member. This mechanism is in the form of software series critiques that are similar to letters to the editor or short notes that appear in other types of journals. These critiques come to the editor, are given a cursory review and then appear in the next issue of the software series. The series is published on a six-month cycle. This pattern of critiques is very similar to the procedures of the Association of Computing Machinery in their software dissemination efforts. If a person writes NCCI for back issues and a particular program documentation, that information goes to them in either microfiche or hard copy form, depending on the request. All of the critiques that go with that piece of software will be sent with the back issue. In that way, a person does not have to request six different issues in order to get all of the critiques.
From the time a software program is a glimmer in somebody's eye until the day it is obsolete, the total life-cycle cost of the program can be divided into phases with cost percentages associated with each phase. The conceptualization, design, and coding of the software typically runs about 25 percent of the total life-cycle cost of that program. Testing prior to release if it is done thoroughly and properly can account for another 25 percent of the cost. Maintenance accounts for 50 percent of the life-cycle cost. What this means is that for every FTE that you are funded to develop and test and release, another FTE is required to maintain that software. If these costs are not paid in maintaining the program, then they are paid in the sense that the software is not used because it does not work. One way or another that cost is paid and it is an expensive item. Software maintenance consists of four primary activities. The first activity involves fixing any program errors detected (usually by the user) after the data is released. The second maintenance activity is addition of improvements to the program as new theories develop and more efficient ways to perform calculations evolve. The third area of maintenance involves keeping the program consistent with other evolving elements of the overall computer operation. The microcomputer or the mainframe computer operation is not the same as having a laboratory scale that just sits there in its present form until the day it is discarded. Computer systems are evolving systems. Whenever a change occurs in a system, the change often affects the programs on that system and sometimes makes them unable to run. These programs will require maintenance in order to keep consistent with the new method of operation. The final maintenance activity—and the most important one—is keeping documentation up to date with respect to any of the other three maintenance activities.
This is the first working version of the Program Design Documentation Standards developed for the North Central Computer Institute.

Please direct comments, suggestions and requests for additional information to:

Thomas J. Olscheske
North Central Computer Institute
662 WARF Building
610 Walnut Street
Madison, WI 53706
(608) 263-4839

This publication was developed by the NCCI Documentation Standards Committee:

Thomas J. Olscheske, Coordinator
John R. Schmidt
Sheila Fitzgerald Krein
Phil Beetley
Barbara Woods
Steve Welch
Steve Harsh
B. J. Conlin
Delmar Hatesohl
Tom Thompson
Douglas Anderson
Ken Wenner
Phil Plumart
Eulyn Jensen

NCCI
Director, NCCI
University of Illinois
Purdue University
Iowa State University
Kansas State University
Michigan State University
University of Minnesota
University of Missouri
University of Nebraska
North Dakota State University
Ohio State University
South Dakota State University
University of Wisconsin-
NCCI PROGRAM DESIGN DOCUMENTATION STANDARDS

The NCCI documentation standards were designed to facilitate software exchange across a broad spectrum of hardware and software systems. Software sharing can be facilitated at three levels:

1. By providing adequate information about the subject matter or scientific basis for the solution procedure. This is critical for the peer review process.

2. By providing adequate information about the implementation of the subject matter solution as a program. This is important for reviewing the software engineering aspects of a program. This could go as far as a discussion of detailed program design (algorithms, modularity, data structures) from a language-independent standpoint.

3. By providing implementation-specific details about the exact hardware/software environment of operational software. This would be especially helpful in the area of microcomputers, where there is considerable variety.

Documentation written for system independent software sharing is more than just program documentation. Therefore the NCCI standards are designated 'design' documentation. This design documentation intends to satisfy these three aspects of software exchange.

Point 2 above addresses the subject area specialist while point 3 concerns the computer specialist. Discussion of the computer procedure or algorithm is directed toward these persons in Sections II (Scientific and Technical Information) and V (Program Logic), respectively, of the Design Documentation Outline. These are the two most important sections of the outline. These two sections when done properly will greatly facilitate software exchange.

The design documentation consists of two parts. The first is the Software Inventory Form. This contains the basic identifying information and system requirements of the software as implemented by the author. This form will be put in the on-line software inventory data base developed by the Institute for access by other member institutions. The second part, the Program Design Documentation Outline, or body of the documentation, will be kept elsewhere in the Institute and be available upon request. Because we encourage the use of graphic methods of presentation, access to the body will probably not be directly accessible through an on-line system.

Because of the wide variety of documentation methods in the computer science field, the author will have the freedom to use his/her own form of expression as long as the information required by the outline is clearly conveyed. The Institute will maintain information on documenting methods and keep up with current program documenting standards.

Since this documentation applies to a wide variety of computer applications, form and content will vary. For example, in a microcomputer-based project where the program is small and straightforward, the description of the procedure or algorithm for the subject matter specialist (Section II) and computer specialist (Section V) might be combined if the presentation fills both needs.

Some documentation sections may not apply to a particular situation. For example, in documenting a utility program which transfers data files between different computer systems, there would be no need to fill out Section II where the subject matter is presented. There is no subject matter area specifically addressed since its application is just to transfer computer data independent of what that data or subject of the data is. It is desirable to have this type of application documented but only those sections which apply need be used.
THE SOFTWARE INVENTORY FORM

The software inventory form is intended to be used in two ways:

1. As a means of collecting information for the Software Inventory, a database of software that is available or of interest to member faculty and staff. The completed form describes the software in enough detail to give the reader a general idea of the program's purpose and implementation.

2. As a requirement for software project documentation to be accepted into the NCCI Program Design Documentation Library.

The database of existing software (part 1), called the Software Inventory, will be separate and independent from the Design Documentation Library. Member institutions' faculty and staff and other interested parties are encouraged to complete the software inventory form for any software they feel would be of benefit or interest to others in their field. NCCI will accept any reasonable response for this software database. NCCI does not endorse or make any claims to the reliability or quality of this software. It will act only as a collection point for software which is available to members.

The Program Design Documentation Library requires that this form be completed before any software project is accepted into it. However, if one doesn't wish to submit a documented software project but still wants to have the project listed in the Institute's data base of existing software, then only the Software Inventory Form need be completed and returned. This information will be available for on-line access.

INVENTORY FORM GUIDELINES

The inventory form is intended to uniquely identify each program in the NCCI Library, to identify the author(s) and/or other(s) who can provide additional expert information about the program, briefly describe what the program does, its scientific or subject matter basis and primary orientation, define the program status, assign keywords that correctly reflect subject matter and mode of operation, and indicate general system requirements. The Program Design Documentation Library requires that this form be completed before any software documentation project is accepted into it. This information will be available for on-line access from the Institute's Data Base access system.

Some additional explanation may clarify what is asked for in certain sections:

In section H, 'Technical Knowledge Needed to Interpret Results,' state any special training or knowledge in the subject matter area that is required to understand the results of the program. For example, if the subject matter category checked is 'crops/horticulture,' indicate whether the knowledge needed is practical farming methods or more theoretical or academic oriented information.

In section I, 'Hardware and Software Resources of Specific Implementation,' give a general description of the computer system that the current software is implemented on. If the software is implemented on several systems then these can be listed here as room permits. Configuration means that some indication of memory size or peripheral devices required, such as graphics plotters, should be given. A detailed configuration specification is not needed.

If very unusual hardware requirements are called for or unusual software enhancements are required then these may be listed under 'Other Hardware/Software Resources.'

Section J, 'Programming Methods,' asks for either the name of the solution or processing technique or a brief description of the method used. If a program uses a formula to calculate the results, then it can be written here. If the program does simple arithmetic or statistics then 'summary inputs' or 'calculating mean,' for example, is sufficient. For complex or hard to define situations brevity is important. A simplistic description is better than a confusing one.

The 'Data File Structures' section should contain just a name or a brief description of the structure type. Typical responses would be: Sequential file access, Random access, TOTAL Data Base, B-tree sorting structure, or Hash table look-up.

If any special programming techniques are used or if the software takes advantage of a special system specification then indicate it in the 'Special Programming Considerations' section.
NORTH CENTRAL COMPUTER INSTITUTE
SOFTWARE INVENTORY FORM

This form is used to enter descriptions of existing software in the North Central region into the Software Inventory, and is required for software documentation to be accepted into the NCCI Documentation Library. It should be filled out in general terms that a programmer as well as an area specialist can understand it. Complete all sections appropriate to your program.

A. TITLE Give a concise, descriptive name to the program (15 words or less).

B. ABSTRACT The abstract should concentrate on what the program does, its primary orientation, primary scientific or subject matter basis, scope or range of operation, intended audience and the use it addresses (10-12 lines or 150 word limit).

C. AUTHOR AND CONTACT PERSON

1. AUTHOR(S)

2. CONTACT PERSON: Name, address, institution, and phone number of person responsible for maintaining & communicating information about the program.

D. CLASSIFICATION OF THE SOFTWARE

1. Subject Matter Categories: Check the appropriate categories. If none fit, suggest an addition(s) to the list.
   _Accounting
   _Administration
   _Basic Research
   _Climate/Meteorology
   _Community Resources
   _Crops/Horticulture
   _Economics/Finance
   _Education
   _Engineering
   _Government
   _Home Economics
   _Human Resources
   _Information Delivery
   _Information Retrieval
   _Livestock/Poultry
   _Natural Resources
   _Office Management
   _Pests
   _Statistics
   _Utility Program
   _4-H
   _Other

2. Specific subject area keywords. Suggest possible keywords that will categorize the program in more detail than the subject matter categories. The keywords will be used for online searches (1 to 5 keywords).

E. PROGRAM USERS Give an indication of the type of user the program was written for and what kinds of people are actually (currently) using the program.

1. Intended Users (check those that are appropriate):
   _Administrators
   _Agribusiness
   _General Public
   _Families
   _Farmers
   _Field Staff
   _Government
   _Researchers
   _Specialists
   _Students
   _Youth
   _Other(s)

2. Current Users

   <List users who might be contacted for program evaluation.>

F. PROGRAM STATUS Date when the program: became operational

   was last revised revision is anticipated
G. PROGRAM SOURCE  Indicate if the program is an original implementation or is a modification or improvement of an existing program.

Check one:  ___Original  ___Revision of:______________________________

H. TECHNICAL KNOWLEDGE NEEDED TO INTERPRET RESULTS  If any special knowledge or training is needed to understand or use the program results, indicate what it is and how it applies.

I. HARDWARE AND SOFTWARE RESOURCES OF SPECIFIC IMPLEMENTATION

Machine Classification:  ___Main Frame  ___Mini  ___Micro  ___Other:______________________________

Language:__________________ Operating System:__________________

Brand Name & Configuration:

<Memory & Peripherals. In small systems this may need to be specified.>

Specific Utilities Used:

<List any system dependent utilities needed to run this program.>

Other Hardware/Software Resources:

<Explain any special or unusual requirements of the system or program.>

J. PROGRAMMING METHODS (Conceptual approach)

Solution Technique:

<A short description or technique name of the method(s) used.>

Data File Structure:

<State the type or give a description of important file structures.>

Special Programming Considerations:

<Indicate any unusual programming techniques or problems.>

K. RESOURCES AVAILABLE FROM CONTACT PERSON

<table>
<thead>
<tr>
<th>YES/NO</th>
<th>FORM OF AVAILABILITY:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HARD COPY</td>
</tr>
</tbody>
</table>

NCCI Program Documentation

User Guide

Source Code

Annotated Source Code
THE PROGRAM DESIGN DOCUMENTATION OUTLINE

The Program Design Documentation Outline is to be used in creating the body of the documentation which will be submitted to and undergo peer review procedures by NCCI and placed in the Program Design Documentation Library. This is the design documentation and represents the minimum required. The three main parts are:

1. statement of the problem,
2. statement of the solution in subject matter vocabulary, and
3. statement of the solution in computer terms to facilitate its implementation on an arbitrary computer system.

Additional sections have been added to make the documentation more complete and usable. These provide information for program users, test information indicating the validity of the implementation, aids for testing a new implementation and reference and acknowledgements where appropriate.

The spirit of these standards is to communicate information between the subject matter specialist and the computer specialist and to facilitate the exchange and implementation of software between different computer systems.

PROGRAM DESIGN DOCUMENTATION OUTLINE

I. General Information
   A. Statement of the Problem
      In a concise manner, describe the need for the project/software.
   B. Statement of the Objective and Scope
      State what this program/project does and how it solves the problem. Indicate relevant research if appropriate.

II. Scientific and Technical Information
   A. Subject Matter Basis: Literature Review and Methodology
      1. Review of Previous Work
         Cite and discuss significant research and/or published works related to the subject matter application. Note in particular any previously developed programs relating to this area, whether documented or not.
      2. Selected Method
         A concise statement of the overall problem solving model or methodology used, including technical justification.
   B. Problem Solving Procedure
      1. Identify each major component, its overall purpose, and the logical relationships among these components.
      2. Break each major component down into sub-components and discuss for each of them:
         a. The purpose or function of the component
         b. Variables and parameters
            1. Source (database, algorithm routines, user-supplied, etc.)
            2. Default values and/or range limits
            3. Units of measure
         c. Solution procedure (formulas, logical steps, etc.)
         d. What results or variables are determined or calculated. Those variables which are program outputs should be so stated.
         e. Discussion of scientific or technical rationale

III. User Oriented Information
   A. Annotated Sample Program Run(s)
      1. Demonstration of all major options
      2. Commentary on procedure and steps
      3. Discussion of responses and/or types of responses
   B. Interpretation of Results
      1. Use plain language to say what results mean
      2. Set proper perspective for a valid interpretation
      3. Discuss special or extreme cases
      4. Give support and/or alternate interpretations
   C. User Guide: If one exists, reference its availability and source (User’s Guide is not part of this documentation.)
IV. Program Testing
   A. Previously Conducted Tests
      1. For the author’s test (required) and each or any independent tests (previously conducted tests) state:
         a. Participants in the test
         b. Methods used—indicate limits, ranges, special cases used in the test
         c. Test data where available for other user tests, that is, where the reader could obtain the test data
   B. Means of Future Testing
      1. Description of procedures required for proper test
         a. Options that should be addressed and the interaction between options
         b. Ranges, limits, etc., to be tried—extreme and special cases
   C. Disclaimer of Program/Project as a Whole

V. Programming Logic
   The intended audience for this section is the project programmer. This is written primarily in computer terms to facilitate the program implementation. Reliance on subject matter vocabulary and knowledge should be minimal.
   A. Programmer Generated Documentation; use one or more methods:

1. ANSI Flow Chart
2. Pseudo-code
3. Structured English
4. Dijkstra Chart (D-chart)

B. Other Documentation; where to obtain the following, if available.
   1. Internal programmer documentation
   2. Vendor manuals
   3. Project manuals

VI. Acknowledgements
   Could include other software used, development of documentation, subject matter contributions, funding, etc.

VII. Bibliography Section
   A. List references for all previous work cited in this paper.
   B. List other articles, monographs, textbooks, etc., used in program logic section (i.e. other documentation).
   C. List other documentation used or referenced.
   D. List references alphabetically by author numbered consecutively. Refer by number in the text.
GUIDELINES TO THE PROGRAM DESIGN DOCUMENTATION OUTLINE

I. General Information

This section need be only a few lines. It should state the motivation for the program, say what the program does and how it solves the problem and indicate any relevant supporting research.

Much of this section could be satisfied by a repeat of the Abstract of the Inventory Form.

II. Scientific and Technical Information

In this section, parts A and B need to be distinct areas of discussion. Within each section the author has the option of using his/her own organizational rationale as long as all points are covered. Also, this discussion should be expressed in terms of subject matter vocabulary. Computer terminology should be avoided here. The place for the procedural discussion using computer terminology is Section V, Programming Logic, unless the nature of the program (small application) allows the combining of sections II and V. (To some extent this is left to the discretion of the author. For advice on presentation, NCCI should be contacted.)

In Section A, emphasize overview aspects. Brevity is important. Stress the 'overall' aspect. In A.2, the selected method statement should correspond to the top level of the algorithm hierarchy.

In Section B a hierarchical breakdown of the procedure is suggested. For simple programs a formula description may be sufficient. In more complex procedures, major components should be discussed first at a low level of detail or at the top of the hierarchy, then particulars or sub-components should be described in more detail as necessary. Complicated programs might even have sub-sub-components. This kind of approach is often referred to as 'top-down.' 'Top' refers to the highest (least detailed, most significant) description of the program's component structure and 'down' means 'getting down to the details' of the sub-components. Section II-B might then look something like:

B-1. The 'top' of the hierarchy is described, with a discussion of the relationships between hierarchical components.

B-2. A detailed breakdown of the process is discussed or for more hierarchical structures, each component is broken down. Note that simple programs may only have major components but that complex ones may require several levels of iteration. Provide examples. Note also that information flow may be represented graphically if desired. If this is done it should be less detailed than in the programmer section (Section V, Programming Logic).

To improve readability, formulas should not be run in the text but rather set out on separate lines.

In section B, the scientific rationale and supporting argument are given for the developer's assumptions. Also new or unique aspects should be noted along with expected benefits of this procedure. Collectively these discussions are the most important part of the document!

III. User Oriented Information

A. The purpose of an annotated sample run is to give a prospective programmer an idea of what the program is capable of and to give a current programmer a guide in checking the new program.

Methods of presentation can vary from an actual session listing with commentary to a slide format for programs requiring graphical output or interaction, along with some accompanying commentary.

B. A discussion of interpretations of the above results is required using non-technical language. The technical aspects should be covered in Section II and a scholarly interpretation should be reserved for another publication.
IV. Program Testing

This section establishes test data for verification of the author's program and gives a new programmer something solid to test the program against.

In part A the author is required to show that the program does indeed work and to demonstrate the appropriate limits, ranges, etc., which give the expected results. The author is not expected to show that all possible cases do work for the program, especially for large projects, but a reasonable sample is expected.

In part B, to help facilitate future implementations, the author should describe any non-obvious checks or situations in data-manipulation that another programmer needs to be aware of. A test procedure need not be outlined but important checks should be pointed out along with an idea of the kind of results to expect.

V. Programming Logic

This section is intended for the programmer/analyst having the responsibility of actually implementing the program or system. From the inventory form the programmer will have the information on which particular hardware/software system the author implemented the original program. From that information the programmer should be able to determine the system requirements for the new implementation.

The information needed to program the project should be in this section. The programmer might find it helpful to read Section II describing the algorithm in subject matter terms.

The author(s) must present the procedure in terms programmers can use. Current methods used in the computer science field are structured English and pseudo code. Other methods would include standard flow charting and Dykstra Chart (D-Chart). Because of the variety of methods and differences in familiarity with these methods, NCCI cannot require a specific method. However, it will encourage and provide information in the use of the current standards in the field. The presentation and method used will be subject to peer review for completeness and clarity before being accepted into the NCCI Software Design Documentation Library.

The Institute will not accept source code in fulfillment of this section. If the author feels that source code would indeed help some programmers it may be referenced. However, any requests for source code must be directed to the author, whose responsibility it is (if referenced) to maintain.

Sections VI and VII

Acknowledgements are optional as with any publication.

A bibliography is required for any works referenced in the body of the documentation. Source code availability is listed here if the author desires. References to any programming documentation such as project manuals, vendor manuals or internal programmer documentation should also be included.
NON-INCLUSIVE REFERENCE LIST
FOR PROGRAM DESIGN AND DOCUMENTATION

Books on Flowcharting

Shelly, Gary B., and Thomas J. Cashman. INTRODUCTION TO FLOWCHARTING AND

Books on Program Design

Yourdon, Ed. TECHNIQUES OF PROGRAM STRUCTURE AND DESIGN. Prentice-Hall, 1975.


Books on Documentation


Rubin, Martin L., ed. DOCUMENTATION STANDARDS AND PROCEDURES FOR ONLINE

Books on System Analysis

DeMarco, Tom. STRUCTURED ANALYSIS AND SYSTEM SPECIFICATION. Yourdon, 1971.

The grandfather of books on 'structured systems analysis.' Sets forth many of the
rules still followed today by leading practitioners. DeMarco is a very successful
consultant and executive who writes practically and well.

Page-Jones, Meiler. THE PRACTICAL GUIDE TO STRUCTURED SYSTEMS DESIGN. Your-
don, 1980.

As the title implies, more about the design phase of large programming systems.
Not a bad book and it comes from the people at Yourdon, who are the de facto
leaders in the use of structured methods. Not too much help for a first-time reader.


Very strong book for the first-time reader. Good treatment of data flow diagrams,
decision tables (condition-action), structured English. Follows the sequence of the
development phases, treating each one in turn. Glossary is helpful. Good introduc-
tion to the concepts of technical, organizational and economic feasibility.

Gane, Chris, and Trish Sarson. STRUCTURED SYSTEMS ANALYSIS: TOOLS AND TECH-

A slightly more modern treatment than the Weinberg book, but not enough to
replace Weinberg as a completely better product. Gane and Sarson explore data
dictionaries more (a means of capturing the data that a given variable, document or
entry, i.e., a data flow, contains) and discuss the transformation from logical system
model to structured system design. A good book that has developed a big following.
Mehlmann, Marilyn. WHEN PEOPLE USE COMPUTERS: AN APPROACH TO DEVELOPING AN INTERFACE. Prentice-Hall. 1981.

A 'must' book for anybody who is considering the development of a computer system that must, for its survival, meet the real needs of its users. Good treatment of terminal, dialogue, primary and secondary users, and the place of training and documentation in the system development life cycle. Has a small but very current list of books and articles for further reading.

Books on Program Language-Like Pseudocode

There are a lot of computer books that contain large portions of pseudocode. This is because many well-known computer science academicians routinely use pseudocode to express algorithms. Unfortunately, little or no explanation about how to effectively read or write pseudocode is included. This is a real problem, because with an understanding of such pseudocode, a near-universal means of expressing algorithms and program logic becomes available.

However, one book called STRUCTURED BASIC AND BEYOND by Professor Wayne Amsbury (Computer Science Press, 1980) contains a very good introduction to pseudocode with BASIC language statements to demonstrate a language-specific implementation. It is recommended to anyone who wants to learn more about pseudocode.

A Note on Structured vs. Other Approaches

Just because an approach isn't 'structured' doesn't in any way mean that it's not worthwhile. Some critics argue that structured methods are just what successful engineers, programmers and managers have been doing for years. It's also become very trendy to be seen as a structured-methods type. Some authors have edited previous books and articles to put the word 'structured' in every other sentence and let it go at that.

There is a new approach on the horizon called 'information engineering.' There will be a new book out in the spring of 1982, co-authored by James Martin (the 'biggie' among computer-application authors) and published by Prentice-Hall. Basically, information engineering states that one should not start with a logical model of an existing system or set of needs as the basis for an improved system.

If anyone is interested in acquiring a library of analysis and design books for reference, look to either Prentice-Hall or Yourdon Press. Most of the recognized authors publish through one or the other.
I. Series Design

A. Information in the NCCI Software Series comes in several forms. These include Software Inventory Forms (short forms containing concise abstracts of program characteristics) and Program Design Documents (comprehensive, reviewed publications describing software via a standardized format). Also included in the Series and Critiques described below in section III 6. The total compendium of all current and past information appearing in the Series will be referred to as the NCCI Software Library.

B. Methods of distribution

1. Issues will be published semi-annually on a subscription basis in microfiche form. Issues will include reviewed Program Design Documents (with their corresponding Software Inventory Forms), and Critiques. In addition, a list of unreviewed items received will be appended. These will generally consist of Software Inventory Forms submitted without corresponding Program Documents (see section III 7).

2. Titles of all submissions received (whether for review or not) will be listed in the NCCI Quarterly.

3. All Software Inventory Forms will be available via an on-line retrieval system. Retrieval may be by keywords or by specific issue.

4. Copies of Program Design Documents may be obtained from subscribing local libraries with reproduction equipment or by request to the NCCI. The NCCI will supply copies in either microfiche or paper form according to the fee schedule in section VIII.

II. Statement of Goal and Objectives

Series Goal: Provide a communication mechanism for the spread of software applicable to the food and agricultural and related industries.

Software is being developed by professionals at numerous sites having relevance and application to users at other locations across the region and nation. Two general types of users are visualized: (1) technically oriented professionals at member institutions who need to evaluate software for inclusion in their efforts and (2) application programmers with the task of implementing this software.
Objectives:

1. Assure a flow of acceptable software applicable to the food and agricultural industries and into the NCCI Software Library.

2. Provide effective access to the acquisitions of the NCCI Software Library.

3. Provide adequate professional recognition for contributions to the NCCI Software Library.

4. Gain improved academic acceptance of software development function.

III. Examples of Acceptable Contributions

1. New applications programs which computerize previously available theory or data while assuring that the proper scientific approach has been used.

2. Applications programs which enhance information delivery and/or clarity to users through modifications of previously developed software.

3. Updates, improvements or enhancements of existing software through changes in basic algorithms.

4. New documentation of existing software.

5. General utility programs which make significant contributions to the flexibility and ease of computer operation and/or application program design and development.

6. Critiques of existing software. These might include (1) statements that software works as stated, (2) evaluation of algorithms, (3) indications of system faults in specific applications, (4) suggestions for improved methods.

7. Software Inventory Forms may be submitted without accompanying Program Design Documents. Such submissions will be stored without review in the on-line retrieval system along with a notation of their unreviewed status. This option is offered by the NCCI solely as a service to its audience; such submissions are not viewed in any form or fashion as having the same weight as peer-reviewed Program Design Documents.
IV. Review Process for Program Design Documents and Critiques

1. No limitations are imposed on sources of articles.

2. Any contributions submitted (including Critiques) must have been subjected to any internal review procedures required for other published documents in the author(s) organization.

3. Author(s) will submit contributions to the Series Editor.

4. The Editor has the authority to accept for review or reject contributions. The Editor performs a cursory review of contributions both for proper format and for substance. He may consult other person(s) on matters of substance. In case of major format infractions, the Editor will return the contributions to the author(s) for correction of format. If the contribution is deemed of inconsequential substance, the Editor will reject the submission and return it to the author(s). In all other cases the Editor exercises his judgment relative to appropriate disposition. The Editor is expected to strive for timeliness in the review process.

5. The Editor selects the reviewers to cover subject matter content, software engineering and appropriateness of format. The number of reviewers selected will depend on the scope and breadth of subject matter covered by the contribution.

6. Submit article with review form to reviewers.

7. Review evaluations returned to Editor.

8. Editor actions:
   a. Accept as is for inclusion and notify author.
   b. Return article and reviewer comments to author for response with instructions.
   c. Reject contribution for inclusion and notify author. Author recourse is to submit a letter to the Editor explaining more fully why this is a significant work.
   d. Seek additional review to clarify conflicting information.

9. Editor receives response from author and takes appropriate action.

10. The Editor may request that a Critique be written based on significant reviewer comments that might be helpful to users.

11. Upon acceptance, the Editor forwards the contribution to the Director of the NCCI or his appointed representative for inclusion of the contribution into the NCCI Series Library.
V. Review Criteria

A. Program Design Documentation

1. Contribution is formatted according to the NCCI Documentation Standards and that the material so included is lucid and meaningful.

2. Problem Significance. The solution to the problem which the software addresses will result in significant benefit to the intended user.

3. Subject matter substance. The algorithm(s) complies with accepted theory and practice in the relevant disciplines included.

4. Software Engineering. Sufficient information presented to permit the conversion of the algorithm(s) into effective and efficient machine readable code.

5. Recognition of Prior Software Development. Reflects extent to which the contribution adds to the current body of documented software.

6. User Friendliness. Relative ease with which the software can be employed by end users.


B. Critiques

1. Lucidly written and provides significant additional information concerning existing acquisitions in the NCCI Series Library.

VI. Materials Transmitted to Reviewer

1. Contributions.

2. Instructions from the Editor. These include the specific criteria to which each reviewer is to respond, an indication that review comments will be transmitted to author(s) (separate comments may be submitted directly to the Editor), and other relevant instructions.

3. Review Form. See attached.
VII. Publication Procedures

1. When a Program Design Document has been accepted for inclusion, in the NCCI Software Series, microfiche or paper copies of the individual document shall be made available.

2. Semi-annually, Program Design Documents accepted and Software Inventory Forms received during the prior six months shall be combined and issued as a microfiche journal.

VIII. Cost and Price Policies

1. One microfiche copy of each accepted Program Design Document will be given to the author(s) without charge.

2. Microfiche copies of individual Program Design Documents may be purchased from NCCI at a price of $3.00 each.

3. Printed copies of individual Program Design Documents may be ordered from NCCI at 10 cents per page.

4. The annual subscription fee for the microfiche journal (issued semi-annually) will be $10.00. A free subscription will be provided to:

   a. The agricultural library of each land grant institution.
   b. Each member of the Technical Advisory Committee; and
   c. Each Campus Coordinator.
NCCI SERIES REVIEW FORM

I. Series Document
   Title and Number

II. Reviewer Recommendation
   1. Accepted as is
   2. Accepted with suggested modification(s)
   3. Return to author for revision (requires review)
   4. Reject

III. Specific Evaluation for each criteria

   Review following criteria:

   ( ) 1. Adheres to proper formatting
   ( ) 2. Problem significance
   ( ) 3. Subject matter substance
   ( ) 4. Software Engineering
   ( ) 5. Recognition of Prior Software Development
   ( ) 6. User Friendliness
   ( ) 7. Appropriateness/Adequacy of Testing Procedures
   ( ) 8. Special Criteria for Critiques

IV. General Reviewer Comments
Additional Comments

1. The Editor should be appointed by the Board on an annual basis, with the expectation that the assignment will not normally continue beyond three years.

2. Editor will develop a listing of potential reviewers over time.

3. Need to determine responsibility for maintenance of NCCI Series acquisitions with respect to currency.

4. Need for publicizing NCCI Series in appropriate journals and Series.

5. Need for long term effort in advocating inclusion of contributions in appropriate professional journals.

6. The Technical Advisory Committee will, at least at the outset, act as and Editorial Board for the Software Series.
PROJECT PROPOSAL GUIDELINES FOR
REGIONAL SOFTWARE DEVELOPMENT
June 4, 1982

The North Central Computer Institute (NCCI) is an organization concerned with enhancing the computer-based information services of the land-grant universities in the North Central Region directed toward farms and farm families. The NCCI is governed by a Board of Directors consisting of one representative from each of these North Central land-grant universities.

A primary objective of the Institute is to foster development of multi-state and multi-discipline computer software. Funding for this software development will be carried out on a project by project basis, with faculty/staff from several states collaborating in the effort.

The Institute will provide funding for travel and communication costs for the software project development team and a Project Advisory Committee. Benefits to collaborative multi-state efforts include access to additional subject matter and technical expertise, as well as a broader application for the finished product.

The Institute provides a unique mechanism for bringing research, extension and teaching personnel together from across state lines for the development of software. Submission of proposals to the NCCI will involve very little additional effort beyond preparation of the initial project proposal.

Basic project funding (for personnel, computer resources, supplies, etc.) is expected from conventional sources or non-Institute grants and contracts. Assistance in securing such outside funding may, however, be part of the study phase.
PROCEDURES FOR MULTI-STATE SOFTWARE PROJECTS FUNDED BY NCCI

I. Application Procedures

A. Proposals are submitted by an individual or group to the NCCI Director through the Campus Coordinator, following initial clearance by the appropriate institution administrator.

B. Project proposals should be patterned after your state's standard Agricultural Experiment Station format (see Section C).

C. Proposals will be accepted on a continuous basis. The Institute Director will forward all proposals to the Technical Advisory Committee (TAC) for consideration at their quarterly meetings. Proposals must be received by February 1, May 1, August 1, and November 1, to be acted on at these quarterly meetings.

II. Evaluation and Implementation Procedures

A. Project proposals will be evaluated for suitability and relative priority by TAC according to the following criteria:

Requirements:

1. All project proposals must plan to involve more than one state.

2. Proposals will only be accepted from individuals with appointments in Cooperative Extension, Experiment Station or resident instruction in the College of Agriculture. (These agencies are the funding bodies of the NCCI).
Proposals will be given priority (with decreasing emphasis) that:

1. Contain a clear statement of the problem, objectives, and proposed methodology.

2. Demonstrate a potential for broad application across the North Central Region.

3. Have a high social and/or economic benefit in relation to cost.

4. Plan to involve the necessary subject matter expertise.

5. Have a high potential to secure the necessary basic project funding from other than NCCI funds for personnel, computer resources, supplies, etc.

6. Plan to involve more than one discipline.

7. Have a clear plan for delivering or implementing the final product.

8. Stimulate new software development within particular disciplines or states.

B. Proposals receiving the highest priority will be scheduled, funded and staffed for a formal study. The study team will be appointed by the Director of NCCI. This team will consist of an Institute staff member, a TAC member, project proposal initiator(s), and others if needed. The Institute staff member will be responsible for coordinating activities of the study team.
C. This study team will have the following responsibilities:

1. Identify potential collaborators at other member institutions and ascertain their interest in participating in a regional software development project.

2. Review the project proposal, objectives and proposed methodology.

3. Review current funding status and provide assistance in identifying additional funding sources. (The project leaders are responsible for securing funds for software development as previously described.)

4. Provide assistance in identifying potential technical problems associated with delivering or implementing the final software product.

5. Develop a plan for an Advisory Committee.

6. Provide a written recommendation, revised project proposal, and proposed budget to TAC for further NCCI funding consideration.

D. The study team's recommendation will be reviewed by TAC. Projects recommended by TAC and approved by the Institute Director will be granted funding for necessary travel and living costs for the project team and Advisory Committee. Institute support personnel will be assigned to assist the project team in an advisory capacity.

E. Each member institution not actively involved in the project will have the opportunity to involve an appropriate representative on an Advisory Committee. Early in the development stages of the project, this Committee should assist the project team by making technical suggestions which will make the software more easily utilized by other states. This committee will function as outlined in the study team recommendation.

F. The software development project would then be executed by the project team, culminating in an installed prototype on at least one computer system, with complete program design documentation as per the Institute standards.

G. Following the documentation, it will be submitted for review for possible acceptance in the NCCI Software Series. The project team is also encouraged to hold a workshop to publicize the newly developed software.
Figure 1  FLOWCHART OF NCCI SOFTWARE PROPOSALS

* ————> LOCAL
    ————
    |
    |
    ————< FACULTY
    ————
    |
    |
    ————< CAMPUS
    ————
    |
    |
    ————< CAMPUS
    ————
    |
    |
    ————< T A C
    ————
    |
    |
    ————< T A C
    ————
    |
    |
    ————< N C C I
    ————
    |
    |
    ————< PROJECT
    ————
    |
    |
    ————< SOFTWARE
    ————
    |
    |
    ————< REVIEWED
    ————
    |
    |
    ————< DOCUMENTED
    ————
    |
    |
    ————< IN NCCI
    ————
    |
    |
    ————< LIBRARY
PROJECT PROPOSAL FORMAT
(3 to 4 pages)

1. Title of proposed project.

2. Project proposal team – name of member(s) and brief resume.

3. Cooperating agencies if any.

4. Project date of initiation and duration of project.

5. Brief description of project.

6. Importance – justify the project on the basis of its importance. Indicate the magnitude of the problem and give some indication as to potential benefits. Be sure to indicate who will be expected to use the finished product.

7. Project objectives – what is the purpose or the projects expected benefits.

8. Review of previous work – literature and existing programs.

9. Computer System – description of the type of computer system on which software will be installed.

10. Methodology, algorithms and data bases to be used – indicate if they exist or need to be developed.

11. Budget – estimate of developmental costs, and existing or potential project funding sources.

12. Clearance through appropriate institution administrator(s) noted.
APPENDIX D
EXISTING SOFTWARE

SOLUTION METHODS

NEW SOFTWARE PROJ. (ORIGIN)

PROJECT TEAM + ADVISORY COM.

SECURE FUNDING

FEASIBILITY STUDY (NCCI + PROJECT TEAM)

PROPOSALS FOR MULTI-STATE SOFTWARE

SOFTWARE IMPLEMENTATION (DESTINATION)

SOFTWARE SERIES (MICROFICHE)

SOFTWARE SER. CRITIQUES

PEER REVIEWED

NOT REVIEWED

SOFTWARE SUMMARY

STANDARIZED DOCUMENTS

ONLINE, KEYWORD DATABASE

REMOTE INQUIRY (DETERMINES EXISTANCE OF SOFTWARE)

USER

INTRA-STATE
Software Development for Computer Applications
in Agriculture and Forestry

Atlanta, Georgia

Tuesday, July 20, 1982

1:00 p.m. Present and Future Computer Software Needs of Farmers
Dr. Robert Kramer, The Kellogg Foundation

1:30 p.m. Discussion
Dr. J. R. Carpenter, Director, Mississippi Cooperative Extension Service - Moderator

3:00 p.m. Role of SAES Scientists in Developing Software
Dr. Robert Sowell, Department of Agricultural and Biological Engineering, North Carolina State

and

Dr. James McGrann, Department of Agricultural Economics, Texas A&M University

3:30 p.m. Discussion
Dr. E. B. Browne, Director, Georgia Agricultural Experiment Station - Moderator

Wednesday, July 21, 1982

8:00 a.m. Software Authorship, Documentation, and Field Testing
Dr. Wallace Killcreas, Department of Agricultural Economics, Mississippi State University

8:30 a.m. Discussion
Dr. Rodney Foil, Director, Mississippi Agricultural and Forestry Experiment Station - Moderator

10:00 a.m. Quality Control, Distribution and Maintenance of Software
Dr. Steve Welch, North Central Computer Institute

10:30 a.m. Discussion
Dr. William Linder, Director, Southern Rural Development Center, Mississippi State University - Moderator
SOFTWARE DEVELOPMENT FOR COMPUTER APPLICATIONS IN AGRICULTURE AND FORESTRY

July 20-21, 1982
Atlanta, Georgia

PARTICIPANTS

Dr. Durward F. Bateman, Director
North Carolina Agricultural Research Service
North Carolina State University
Raleigh, NC 27650

Dr. E. Broadus Browne, Director
Agricultural Experiment Station
University of Georgia
Room 107, Conner Hall
Athens, GA 30602

Dr. James R. Carpenter, Director
Cooperative Extension Service
Mississippi State University
Box 5446
Mississippi State, MS 39762

Dr. Neville P. Clark, Director
Texas Agricultural Experiment Station
Texas A & M University
College Station, TX 77843

Dr. Tom E. Corley
Agricultural Experiment Station
Auburn University
Auburn University, AL 36849

Ms. Patti Sue Foil
Computer Applications and Services
Cooperative Extension Service
Mississippi State University
Mississippi State, MS 39762

Dr. R. Rodney Foil, Director
Mississippi Agricultural and Forestry Experiment Station
Mississippi State University
Mississippi State, MS 39762

Dr. Maurice H. Frere
USDA--ARS
Box 53326
New Orleans, LA 70156

Dr. J. E. Halpin, Director-at-Large
Southern Association of Experiment Stati
Clemson University
104 Barre Hall
Clemson, SC 29631

Dr. W. E. Hardy, Jr.
Department of Agricultural Economics
Auburn University
Auburn University, AL 36849

Dr. D. T. Hill
Agricultural Engineering Department
Auburn University
Auburn University, AL 36849

Dr. Donald C. Huffman
Department of Agricultural Economics
Louisiana State University
Baton Rouge, LA 70803

Dr. Ronald R. Johnson, Associate Director
Agricultural Experiment Station
Oklahoma State University
Stillwater, OK 74073

Dr. William H. Judy
USAID/AFRICA Bureau
Washington, D.C.

Dr. Terry C. Keisling, Associate Professor
Agronomy Department
University of Arkansas
Box 391
Little Rock, AR 72203
The SRDC is one of four regional rural development centers in the nation. It coordinates cooperation between the Research (Experiment Station) and Extension (Cooperative Extension Service) staffs at land-grant institutions in the South to provide technical consultation, research, training, and evaluation services for rural development. This publication is one of several published by the Center on various needs, program thrusts, and research efforts in rural development. For more information about SRDC activities and publications, write to the Director.

Southern Rural Development Center
Box 5406
Mississippi State, MS 39762

The Southern Rural Development Center is an Equal Opportunity Organization providing research, educational information, and other services only to individuals and institutions that function without regard to race, color, sex or national origin. SRDC is an Equal Opportunity Employer.