Project demonstrating excellence "operations management decision support system"

Walsh, T. Joseph, Ph.D.

The Union Institute, 1994

PROJECT DEMONSTRATING EXCELLENCE

"OPERATIONS MANAGEMENT DECISION SUPPORT SYSTEM"

T. JOSEPH WALSH

THE GRADUATE SCHOOL OF THE UNION INSTITUTE

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE:

DOCTOR OF PHILOSOPHY

IN

MANAGEMENT INFORMATION SYSTEMS / OPERATIONS MANAGEMENT

CORE FACULTY ADVISOR: BARRY C. HEERMANN, Ph.D.

APRIL 30, 1994

ABSTRACT

This Project Demonstrating Excellence consists of two parts. The first part includes personal computer software on two disks and an accompanying text. The second part is a contextual piece that contains a review of the relevant scholarly literature and reflections on the methodology used in the software development and the student and faculty testing.

The purpose of this Project Demonstrating Excellence was to develop and demonstrate a general purpose, user-friendly, decision support system for operations management. This system is called: "Operations Management Decision Support System." It includes the following modules: Time Series Forecasting Analysis, Regression Analysis, Facilities Layout Analysis, Location Analysis, Linear Programming, Transportation Model, Assignment Model, Line Balancing, Inventory Analysis, Aggregate Production Planning, Material Requirements Planning, Statistical Process Control, Gantt Charts, Critical Path Method, and Program Evaluation and Review Technique. The system design incorporates Object Oriented Programming and a Graphical User Interface to produce a high-level of user friendliness. The system provides the ability to display and print graphics, where appropriate.

The software has the appearance of a Window's application but runs from DOS. There are many context-sensitive help windows. Example problems are set up and tied to the text.

The text provides both examples and problems. High quality screen prints and sample reports are inserted throughout, where appropriate. The first chapter is an overview of the system. Each subsequent chapter covers one of the system modules.

The contextual piece is organized into four chapters. It begins with a review of the relevant scholarly literature. The literature review is divided into two parts. Chapter one reviews management information systems as sociotechnical systems. The technical approaches of management science, computer science, and operations research and the behavioral approaches of psychology, political science, and sociology are discussed. Chapter two looks at the thinking of several important scholars with respect to management information systems teaching theory. Chapter three traces the methodology used in the software development. The fourth chapter reflects on the student and faculty testing of the software.

PREFACE

This Project Demonstrating Excellence applies management information systems theory using the latest personal computer technologies for the development of decision support systems as applied to operations management issues.

The software and text developed as part of this PDE serve two groups. First, they serve as a teaching and learning resource for both undergraduate and graduate operations management courses. Use of the Decision Support System will facilitate the student understanding of operations management decision tools. The software and text will be used in classrooms to supplement traditional operations management texts. Second, they serve as management tools. Herbert Simon proposed the theory that managers do not look for optimum solutions when making decisions. They generally look for one good solution that works, make the decision, and then move on to another problem. Simon called this "satisficing." Simon also proposed that managers have a *limited* or *bounded* rationality in that they are not capable of considering all possible alternatives. This Decision Support System for Operations Management helps overcome the limitations of human beings as processors of information. Human-machine synergies will be gained. Managers will have an increased capability of moving their decision making process from "satisficing" to "optimizing." Use of the Decision Support System will permit operations managers to make better decisions in a more timely manner.

The foundation upon which the development of the software and text was based encompassed a variety of theoretical and practical considerations. Resources included industrial seminars, industry manuals, commercial texts, software development tools, graduate coursework, and scholarly works on humanized information systems, decision support systems, and expert systems. Technological and behavioral approaches were applied. This Project Demonstrating Excellence represents a balance of theory and practice.

TABLE OF CONTENTS

CHAPTER ONE
REVIEW OF THE CURRENT LITERATURE: MANAGEMENT INFORMATION SYSTEMS AS SOCIOTECHNICAL SYSTEMS

Chapter one begins with a discussion of the study of management information systems. This is followed by a review of the differences between electronic data processing, management information systems and decision support systems. The balance of the chapter is organized into examinations of several important behavioral approaches to management information system and decision support system design.

The study of management information systems (MIS) began in the 1960s. It was during this decade that the MIS discipline and the first academic departments began appearing in business schools across the United States. The focus was on computer-based information systems aimed at managers. Technical approaches to information systems dominated the field in its early years.

According to Loudon and Loudon, the study of management information systems has come a long way since then. Today, it is divided into technical and behavioral approaches. The Loudons tell us that: "Information systems are sociotechnical systems. Though they are composed of machines, devices, and 'hard' physical technology, they require substantial social, organizational, and intellectual investments to make them work properly" (Loudon, 1994, p. 20).

Loudon and Loudon go on to define the technical approaches as:

-	**Management Science**	emphasizes the development of models for decision making and management practices.
-	**Computer Science**	concerned with establishing theories of computability, methods of computation, and methods of efficient data storage and access.
-	**Operations Research**	focuses on mathematical techniques for optimizing selected parameters of organizations such as transportation costs, inventory control, and transaction costs.

and the behavioral approaches as:

-	**Psychology**	concerned with individual responses to information systems and cognitive models of human reasoning.

-	**Political Science**	investigates the political impacts and uses of information systems.
-	**Sociology**	focuses on the impact of information systems on groups, organizations, and society.

Simon differentiates between the terms 'operations research' and 'management science'. He says that these terms "are nowadays used almost interchangeably to refer to the application of orderly analytic methods, often involving sophisticated mathematical tools, to management decision making, and particularly to programed decision making. At a more philosophic level, operations research may be viewed as the application of scientific method to management problems, and in this sense simply to represent a continuation of the earlier scientific management movement." He goes on to say that:

> Historically, operations research and management science did not in fact emerge out of scientific management or industrial engineering. As a sociological movement, operations research, emerging out of the military needs of World War II, brought the decision-making problems of management within the range of interests of large numbers of natural scientists, and particularly of mathematicians and

> statisticians. The operations researchers soon joined forces with mathematical economists who had come into the same area - to the mutual benefit of both groups. And there was soon widespread fraternization between these exponents of the "new" scientific management and men trained in the earlier traditions of scientific management and industrial engineering. No meaningful line can be drawn any more to demarcate operations research or management science from the two older fields. (Simon, 1977, p. 55)

The technical approaches emphasize mathematically based normative models and the physical technology and formal capabilities of computer systems. Many system utilization, implementation, and creative design problems cannot be solved by application of the technical approaches alone. They can only be solved by also including a consideration of the behavioral approaches. Both technical and behavioral approaches are discussed throughout the rest of this contextual piece.

Most people are confused by terms like EDP, MIS, and DSS. They don't know the exact definitions of each and where one leaves off and the next picks up. Sprague tells us that much of the "difficulty and controversy with these terms can be traced to the difference between an academic or theoretical definition and 'connotational' definition. The former is carefully articulated by people who write textbooks and articles in journals. The latter evolves from what actually is developed and used in practice, and is heavily influenced by the personal experiences that the user of the term has had with the subject" (Sprague, 1986, p. 9).

According to Sprague, the characteristics of the "connotational" definitions are:

ELECTRONIC DATA PROCESSING:

- a focus on data, storage, processing, and flows at the operational level;
- efficient transaction processing;
- scheduled and optimized computer runs;
- integrated files for related jobs; and
- summary reports for management.

MANAGEMENT INFORMATION SYSTEMS:

- an information focus, aimed at the middle managers;
- structured information flow;
- an integration of EDP jobs by business function, such as production MIS, marketing MIS, personnel MIS, *etc.*; and
- inquiry and report generation, usually with a database.

DECISION SUPPORT SYSTEMS:

- decision focused, aimed at top managers and executive decision makers;

- emphasis on flexibility, adaptability, and quick response;
- user initiated and controlled; and
- support for the personal decision making styles of individual managers.

Sprague believes this "connotational" view has some serious deficiencies:

- It implies that *decision support* is needed only at the top levels. In fact, *decision support* is required at all levels of management in the organization.
- The decision making which occurs at several levels frequently must be coordinated. Therefore, an important dimension of *decision support* is the communication and coordination between decision makers across organizational levels, as well as at the same level.
- It implies that *decision support* is the only thing top managers need from the information system. In fact, decision making is only one of the activities of managers that benefits from information systems support.

Sprague does not accept the narrow connotational view. He only describes it. He argues that MIS should refer to "the entire set of systems and activities required to manage, process, and use information as a resource in the organization" (Sprague, 1986,

p. 10).

According to Sprague, the characteristics of the "theoretical" definition of management information systems are:

- Dedicated to improving the performance of knowledge workers in organizations through the application of information technology.
- Improving the performance is the ultimate objective of information systems - not the storage of data, the production of reports, or even 'getting the right information to the right person at the right time.' The ultimate objective must be viewed in terms of the ability of information systems to support the improved performance of people in organizations.
- Knowledge workers are the clientele. This group includes managers, professionals, staff analysts, and clerical workers whose primary job responsibility is the handling of information in some form.
- Organizations are the context. The focus is on information handling in goal seeking organizations of all kinds.
- The application of information technology is the challenge and opportunity facing the information systems professional for the purposes and in the contexts given above.

Sprague tells us that "a DSS is not merely an evolutionary advancement of EDP and MIS, and it will certainly not replace either. Nor is it merely a type of information system aimed exclusively at top management, where other information systems seem to have failed. A DSS is a class of information system that draws on transaction processing systems and interacts with the other parts of the overall information system to support the decision making activities of managers and other knowledge workers in the organizations" (Sprague, 1986, p. 11).

Parker reports: "The term management information system was first coined to distinguish the systems that merely processed transaction data from systems whose primary purpose was to provide information" (Parker, 1989, p. 432).

The Loudons expand on Sprague's theoretical definition characteristics by telling us that:

> Managers use decision support systems (DSS) to assist them in making decisions that are semistructured, unique or rapidly changing, and not easily specified far in advance. DSS differ from MIS in several ways. DSS have more advanced analytical capabilities that permit the user to employ several different models to analyze information. These systems draw on internal information from TPS [transaction processing systems] and MIS, and they often bring in information from external sources . . . DSS tend to be more interactive, providing users with easy access to data

> and analytical models through user-friendly computer instructions. (Loudon, 1994, p. 20)

The concepts involved in DSS were first described in the early 1970s by Michael S. Scott Morton under the term *management decision systems* (Morton, 1971, pp. 1-6). Morton depicted such systems as:

- interactive computer-based systems
- that help decision makers utilize *data* and *models*
- to solve unstructured problems

In 1978, Keen and Morton defined Decision Support Systems to be: "computer-based support for management decision makers who are dealing with semistructured problems" (Keen, 1978, p. 97).

O'Brien included both unstructured and semistructured problems:

> Decision support systems help managers solve the *semistructured* and *unstructured* problems typically faced by decision makers in the real world. They are quick-response systems that are user-initiated and controlled and support the personal decision-making style of a manager. DSS are designed to use decision models and a decision maker's own insights and judgments in an interactive computer-based process leading up to a specific decision. (O'Brien, 1988, p. 344)

Parker expands on this and defines a Decision Support System as: "a system that provides tools to managers to assist them in solving semistructured and unstructured problems in their own, somewhat personalized, way." Parker goes on to say: "Often, some type of modeling environment - perhaps a very simple environment such as the one accompanying a spreadsheet package - is involved. A DSS is not intended to make decisions for managers, but rather to provide them with a set of capabilities to enable them to generate the information they feel they need to make decisions. In other words, DSSs support the human decision-making process rather than [provide] a means to replace it" (Parker, 1989, p. 432).

Perhaps the best definition is provided by Davis and Olson:

> **"A decision support system (DSS) is an information system application that assists decision making"** (Davis and Olson, 1985, p. 11).

What are the major characteristics of Decision Support Systems?

Turban offers the following four:

- DSS incorporate both data and models.
- They are designed to *assist* managers in their decision processes in *semi-structured* (or *unstructured*) tasks.
- They *support*, rather than *replace*, managerial judgement.

- The objective of DSS is to improve the *effectiveness* of the decisions, not the *efficiency* with which decisions are being made.

(Turban, 1988, p. 7)

Loudon and Loudon believe there are five:

- DSS offer users flexibility, adaptability, and a quick response.
- DSS allow users to initiate and control the input and output.
- DSS operate with little or no assistance from professional programmers.
- DSS provide support for decisions and problems whose solutions cannot be specified in advance.
- DSS use sophisticated analysis and modeling tools.

(Loudon, 1994, p. 48)

Awad tells us there are three:

- A focus on *decisions* and use of the computer to *support* judgment.
- Applicability to all levels of management for solving *semistructured* problems or ones where part of the information is computer based, but where the manager's judgment is needed to provide a solution.
- An individualized, manager-oriented, user-friendly, and interative environment that makes it possible to explore a problem situation using a combination of the

> computer's analytic model and the manager's judgment. The DSS must be easy to create, easy to understand, and easy to use.
>
> (Awad, 1988, p. 264)

The three views are not inconsistent.

Most experts agree that a Decision Support System has three basic components: a database, a model base, and a software system. The Loudons describe these components as follows:

> The **DSS database** is a collection of current or historical data from a number of applications or groups, organized for easy access by a range of applications. The DSS database management system protects the integrity of the data while controlling the processing that keeps the data current . . .
>
> A **model base** is a collection of mathematical and analytical models that easily can be made accessible to the DSS user. A **model** is an abstract representation that illustrates the components or relationships of a phenomenon. . . . Each decision support system is built for a specific set of purposes and will make different collections of models available depending upon those purposes . . .
>
> The third component of DSS is the **DSS software system.** The DSS software system permits easy interaction between

> users of the system and the DSS database and model base. The DSS software system manages the creation, storage, and retrieval of models in the model base and integrates them with the data in the DSS database. The DSS software system also provides a graphic, easy to use, flexible user interface that supports the dialogue between the user and the DSS. DSS users are usually corporate executives or managers, persons with well developed working styles and individual preferences. Often they have no patience for learning to use a complex tool, so the interface must be relatively intuitive. In addition, what works for one may not work for another. Many executives, offered only one way of working (a way not to their liking) will simply not use the system. In order to mimic a typical way of working, a good user interface should allow the manager to move back and forth between activities at will. (Loudon, 1994, p. 558)

There are significant differences between MIS system design and DSS system design.

In traditional MIS system design, the design starts with an end user specifying what he or she wants the final output of the system to look like. At the beginning of a DSS system design, the end user generally has no idea of what the final output will be. "All the vital features of the system that are decided up front in the traditional life-cycle methodology are decided at the end in DSS design. DSS must therefore use a changing, evolving method that is iterative. Iterative design utilizing prototyping is recommended" (Loudon, 1994, p. 559).

In traditional MIS system design, the last development stage is an implementation stage. This is where documentation and training typically occur. In a DSS system design, the users start using the system as soon as the first prototype is available and they continue to use the system as it develops. Implementation is spread out through the entire development process and it is not a separate stage. Documentation and training have to occur simultaneously with development.

THE HUMAN FACTOR

According to Samson, "The strength of a DSS is the integration of man and machine, the synergistic coupling of human problem-solving potential with computerized analytic models and data processing/memory." Samson goes on to say: "The ability of computers to apply rules of optimization, simulation, evaluation, and analysis to data sets with speed and accuracy makes them valuable as decision support systems. Further, computers can help people overcome some of the limitations inherent in their unaided decision behavior, including forgetfulness and capriciousness, through providing a variety of memory aids and rational modeling approaches" (Samson, 1988, p. 538).

Systems designers need to be concerned about the human factors. Rubinstein offers the following definition:

> **"Human factors is the study of how people and machines interact. It is a technology for creating designs that work well in human terms. Without knowledge of people and how they use systems, design is only a hit-or-miss process"** (Rubinstein, 1984, p. 5).

In *Humanized Information Systems Analysis and Design*, Kozar states:

> Excellent organizations lead from their strengths. We must use "people strengths" in designing systems. These strengths include being adaptive and creative, being intuitive and having feelings, handling unstructured situations, using judgment, and dealing with other humans in a sensitive manner.
>
> Along with the strengths of people come very different needs. People need to be motivated and recognized for their contributions. They tire easily, get bored, are slower calculators and processors than machines, have smaller short-term memories, and are often imprecise. We must design systems to use people strengths and support human weaknesses. (Kozar, 1989, p. 276)

Systems designers have not always done a good job of using people strengths and supporting human weaknesses. Not only do they fail to make the person a part of the system; they also fail to consider whether the person will even accept the system.

When designers first consider developing an information system, they typically start off with a feasibility study. Kozar recognizes three types of feasibility studies:

1.	*Technical*	-	whether something is possible with current technology.
2.	*Economic*	-	whether something makes sense from a cost/benefit view.
3.	*Operational*	-	whether something will be accepted and used.

According to Kozar, "Historically, we have been very conscious of technical feasibility and have tried to economically justify systems. However, some of our technical successes were never used. A system that is not used - not operationally feasible - provides no benefits. This destroys the proposed economic feasibility." Many information systems have been designed, developed, implemented, and then, shelved. (Kozar, 1989, p. 276)

End-user acceptance must be gained starting with the time of the first user contact. Constant feedback and appropriate responses to that feedback must occur throughout the development of the system. Kozar claims "there is no magic to developing accepted systems." He reports that he has found common themes running through all successful people subsystem development work.

These common themes include:

- *Empathy for the User* - the developer must constantly think about how the user perceives the system. Perceptions count! The user must never be put in the position of thinking about the developer, "You really do not understand me or my job or needs."
- *Action-Orientation* - the developer must tell the user what steps are needed to accomplish tasks that must be performed. If actions are required, the developer must specify those actions. The developer should not inform the user about topics and then expect the user to ferret out what he or she must do.
- *Testing of All Developments and Documentation* - the developer must dispel a user's fear of new activities by making sure the performer will not get into trouble and will not experience unnecessary anxiety. The developer must make sure the new procedures, manuals and training get the job done by actual performance evaluation, not armchair speculation.

(Kozar, 1989, p. 276)

Systems designers must <u>integrate</u> man and machine. They must build better interfaces.

USER INTERFACES

According to Saiedian:

> User interface design has become increasingly important as computer use grows. In fact, it has become a distinct design activity during the development of many modern applications. Its goal is to establish the layout and interaction mechanisms through which a user interacts with the computer. If the interface is easy to use, intuitive and forgiving, the user will be inclined to make good use of the application. . . .The user interface can be defined as everything that the user interacts with when operating the computer.
>
> Feedback should be given after input is accepted to acknowledge the input. Guidance should be given before input is expected explaining what input is required. Error messages should be given when the user makes a mistake. Status information should be given at the start of task sequences and at subtask boundaries to inform the user about what is happening. . . . Good error messages are very important if the interface is to be intuitive.
> (Saiedian, 1993, p. 17)

Kozar reminds us:

> The objective of systems development is to have human-machine *symbiosis* - two dissimilar systems dependent on each other, existing in a common space, to achieve a

> mutually advantageous goal. Each subsystem needs the other to achieve the best results . . .
>
> . . . people have the abilities to be adaptive, creative, and intuitive; they have feelings, handle unstructured situations, use judgement, and deal with other humans in a sensitive manner. These abilities must be supported with machine strengths. These strengths include the machines' abilities to read and write certain documents quickly, store vast amounts of data, do calculations at a rapid rate, follow and execute instructions tirelessly and without question, and be very accurate. But to have success and use these strengths, these unlike entities must support each other.
>
> Much of this support is possible when these two unlike entities can *communicate*. Lately, much attention has been given to these communications, often called *interfaces* or *interactions*, to encourage wider use of machines and automation. Since systems development is a relatively new discipline, we have borrowed from other disciplines to understand communications. This includes the fields of video game design, graphic arts, filmmaking, psychology, sales, and even magic. (Kozar, 1989, p. 306)

Heckel has researched these other disciplines. He developed a listing of the elements of what he considers to be friendly software design:

- Know your subject.
- Know your audience.
- Maintain the user's interest.
- Communicate visually.
- Level the user's knowledge.
- Speak the user's language.
- Communicate with metaphors.
- Focus the user's attention.
- Anticipate problems in the user's perception.
- If you can't communicate it, don't do it.
- Reduce the user's defensiveness.
- Give the user control.
- Support the problem-solving process.
- Avoid frustrating the user.
- Help the user cope.
- Respond to the user's actions.
- Don't let the user focus on mechanics.
- Help the user to crystallize his thoughts.
- Involve the user.
- Communicate in specifics, not generalities.
- Orient the user in the world.
- Structure the user's interface.
- Make your product reliable.
- Serve both the novice and the experienced user.
- Develop and maintain user rapport.
- Consider the first impression.

- Build a model in the user's mind.
- Make your design simple...
- But not too simple.
- You need vision.

(Heckel, 1984, pp. 21-101).

Kozar recognizes that people interact with machines in different ways. People must:

- Provide data on forms for conversion to machine-readable mediums.
- Provide data directly to machines for processing through online interaction.
- Generate queries where the machine will respond to a specific question.
- Carry on dialogs with machines to do analysis.
- Receive processed data in the form of documents such as paychecks and bills or informative reports.

If any of these interactions is poorly designed, people will become frustrated with a machine. This frustration is the result of human characteristics, including getting bored with tedious tasks, becoming physically and mentally tired, and seeking better ways of doing things. The person-machine interface designer should attempt to come up with solutions that allow the person and the machine to team up without causing errors, boredom, frustration, and even hate. Above all, a properly designed system will allow the user to focus on his or her task, not on using a machine. (Kozar, 1989, p. 306).

Many times we hear the phrase 'user friendly.' Just what does 'user friendly' really mean?

Kozar has thought about this. He started by considering the characteristics of a good friend. He decided a good friend would be someone who:

- Lets me be me.
- Trusts me and allows me to trust him or her.
- Is predictable when I need stability.
- Shows me respect.
- Adapts to me if I need to change.
- Pushes me if I need a push.
- Helps me define my goals.
- Helps me reach my goals.
- Helps me stay out of trouble.
- Provides aid if I do get into trouble.
- Lets me make mistakes and learn from them.
- Is natural and not phony.
- Doesn't bore me.
- Does not greet or treat me like a stranger unless I am one.
- Learns from interactions with me.

- Adapts to my style of doing things.

> These are all things a good human friend would do if harmoniously working together with us. It is important to realize that when people use language to interact with a machine, they behave as if the rules for person-to-person communications apply (Rubinstein and Hersh, 1984, p. 63). For example, friendly persons are perceived as cooperative and not confrontative, not aloof, interrupting only with care, saying enough but not too much, and not creating new rules (like making me use commas where I normally leave a space). This is the key for building friendly interfaces. (Kozar, 1989, p. 307)

Kozar believes that:

> When dealing with humans, machines must be perceived to be tolerant. The strategy of humanized dialog design consists of analyzing the user's environment to establish natural communication patterns, determining and establishing natural sequences and codes, and then allowing for normal variations (including shortcuts). Consideration must be given to:
>
> 1. *Sequence Tolerance* - Allowing natural sequences (not last name first!).
> 2. *Content Tolerance*
>   a. Allowing synonyms . . .

b. Accommodating learning humans with flexible command identification ("not knowing" frustration changing to "overexplanation" frustration as learning takes place).

c. Allowing for spelling errors . . .

d. Not forcing the filling of fields such as middle initial . . .

3. *Procedure Tolerance*

a. Allowing experienced users to use a different order for multiple commands. ([Withdraw, $50, checking account, end] or [$50, checking account, withdraw, end]).

b. Not self-destructing or damaging data due to operator error if the machine can automatically build backups, allow recoveries, or permit "undoing" the last command. There should be no opportunity for "self-destructs," and audit trails should be kept for recovery.

c. Composing alternative "noncomprehension" messages to allow multiple explanations if an operator does not understand an error or "help" message. ("Let me explain it another way.")

4. *Pacing*

a. Duplicating natural conversation paces by not being too slow, which provokes impatience, or being too fast which pushes the person and provokes anxiety. The range of response times is more critical than the average. Examining distributions of response times to determine acceptability must be done.

b. Providing interim responses just like a human would do ("Hang on, I'm thinking").

c. Keeping track of mistakes and "suggesting" that the operator take a rest break if too many errors occur.

The designer must use judgement in specifying interactions. He or she should avoid dialogs that are so condescending that they are degrading, or so machinelike that they make people feel "robotic." (Kozar, 1989, p. 308)

PERSON-MACHINE DIALOG

A good user interface can not exist without an appropriate person-machine dialog:

Kozar describes a general process for developing one:

- Determine where dialogs are needed.
- Analyze dialog requirements.
- Specify dialog design.
- Specify input/output layouts.

Kozar asks us to: "Remember that development is not a linear process. Reviews and tests will reveal weaknesses that will cause more analysis and redesign. A prototyping philosophy often is required. These reviews and tests should take place throughout the development process. Constant interaction with the user will assure that the interfaces will fit the needs of user groups and be used once developed. Participation promotes both system quality and acceptance" (Kozar, 1989, p. 309).

Kozar admonishes:

> The developer must not only understand what the user does when things go right, but also when they go wrong. This will aid in designing strategies for backups, recoveries, and audit trails. Unreasonable stress should be avoided. This could be caused by the user feeling that:
>
> - Control has been lost.
> - Unexpected events have confused me.
> - Novelty or levity is used when seriousness is needed, or vice versa.
> - Responses are inappropriate, such as using codes when an explanation should have been used.
> - Hope is lost that recovery is possible.
>
> (Kozar, 1989, p. 310)

An appropriate dialog process is very important. Bennett divides the dialog process into three parts: action language, display, and knowledge base. (Bennett, 1977, p. 3)

Turban defines these three parts as follows:

- The action language is what the user can do to communicate with the system. It includes such data input options as the use of a regular keyboard, function keys, touch panels, mouse, joystick, voice activation, and an optical reader for typed or written material.
- The display or presentation language is what the user sees or hears. The display language includes options such as a character or line printer, a display screen, graphics, plotters, and audio output. A variety of studies have shown that the type of output provided has an impact on the quality of the decisions made and on the user's perception of the system (Dickson et al., 1977, pp. 913,923). It is important to provide output that is appropriate for the users of the systems and for the decisions being supported.
- The knowledge base includes the information that the user must know. The knowledge base consists of what the user needs to bring to the session with the system in order to use it effectively. The knowledge may be in the user's head, on a reference card or instruction sheet, in a user manual, or in a series of "help" outputs available upon request. (Turban, 1988, p. 86)

The dialog component of a DSS is the software and hardware that provide the user interface for DSS. The term *user interface* covers all aspects of the communications between a user and the DSS. It includes not only the hardware and software, but also factors that deal with ease of use, accessibility, and human-machine interactions. Sprague and Carlson feel:

> Much of the power, flexibility, and usability characteristics of a decision support system derive from capabilities in the interaction between the system and the user, which we call the dialog subsystem . . . For years there have been systems with strong computational algorithms or excellent data access routines but whose effectiveness was limited because they were difficult to use. In fact, from the DSS user's point of view, *the Dialog is the System*. All the capabilities of the system must be articulated and implemented through the Dialog.
> An inconvenient user interface is one of the major reasons why managers have not used computers and quantitative analyses to the extent that these technologies have been available. (Sprague and Carlson, 1982, p. 29)

Turban talks about "Dialog Styles":

> The effectiveness of the interface depends on the strength and variety of capabilities in each of the dialog elements presented above. Combinations of these elements comprise what might be called a *dialog style*. Dialog styles determine how the DSS is directed and what the DSS requires as an input and provides as output. A DSS can be directed

in several different ways. One approach is to have a menu-driven system. The user simply selects the desired options from series of menus. This approach provides more flexibility but requires the user to be more familiar with the system. Recently, other approaches have been developed. For example, some systems use an input form displayed on a CRT, and the user fills in the form with a light pen or by pointing a finger, or by using the mouse. These inputs are then processed by the DSS and the results are displayed. The concept of windows enables a simultaneous display of several files or computer screens that may enhance the dialog process. It is usually accompanied by a mouse device. Finally, a DSS can be directed by voice. This is an ideal means of communication and is most natural to the user. Unfortunately, it is still in the development stage. (Turban, 1988, p. 87)

Siegfried Treu describes characteristics of an ideal "User-System Interaction" design that focuses on the paramount need for users to be able to communicate with the system across the physical interface:

- The interaction language should be grammatically or syntactically structured to be easily learned and to enable it to be a logical and therefore effective means for communication. It should also exhibit an incident-specific contextualness which, perhaps through visually or logically apparent prompting or association, enables a user, at any given point in time, to determine what can or should be done next. In addition, the language should also facilitate the patterning of successive interactions into well-defined sequences or scenarios.

- People may enjoy a certain amount of variety and unpredictability in communicating with other people. However, when people deal with computers as tools, they need and want predictability. Firstly, the user wants to have uniformity in available actions contingent on presented action contexts. That is to say, if a user is able to or must take certain action under one set of graphic circumstances, it should be possible to do likewise under repeated identical or similar circumstances.
- A user should be able to expect identical system responses in action contexts which are identical in the perception of the user. The printed messages themselves can be made to vary interestingly by some innovative designer, but the essential meaning and location of the substantive response must be the same.

(Treu, 1977, pp. 66-67)

Davis and Olson believe that *consistency* in relating to users' general or semantic knowledge is a key design feature of user interfaces. They say "A user faced with a request for a response in a dialog with a computer and an unclear set of choices may borrow from the experience of other computer systems with similar interfaces to make 'an educated guess'" (Davis, 1985, p. 531).

Davis and Olson tell us that a "good screen design is clear, uncluttered, and free of irrelevant information." They offer two useful guidelines for deciding what information should be put on a single screen:

- Provide only information that is essential to making a decision or performing an action.
- Provide all data related to one task on a single screen. The user should not have to remember data from one screen to the next.

They also offer guidelines for placement of text and data based on human information processing limitations and culture-based habits for processing:

- Place items together that logically belong together (name and address).
- Place items in customary processing order (city followed by state followed by zip code).
- Position most important items at left side and arrange in importance from top to bottom.
- Leave sufficient space so that items do not get confused.
- Position items across close enough that the eye does not change rows in moving across. Use guide strips of lines, dots, or dashes if necessary.

(Davis, 1985, p. 534)

A basic objective of good user interface design is to minimize the errors of the person in the person-machine dialog. A fundamental part of "user-friendly" is having adequate error control. Davis and Olson say "there should be a system response not only to every correct input but also to every incorrect input."

They go on to tell us that a well-designed user interface should have four distinct dimensions of error control:

- *Error prevention.* As much as possible, the system should provide specific instructions (e.g., prompts, help facilities) so that the user knows exactly what to do and avoids making errors.
- *Error detection.* When an error is made, the system should identify it clearly and explicitly. "Polite" error messages are favored. Notification that an error has been detected should occur immediately. The user should be able to easily identify the error . . . no error should cause the system to terminate abnormally.
- *Error correction.* Correction should be straightforward and should require rekeying only the portion of the data in error.
- *Error recovery.* Once a command has been accepted, it may still be in error and cause an incorrect action. An important feature of a well-designed system is the ability to "undo" something which has been done.

(Davis, 1985, p. 540)

The response time is part of the dialog. Davis and Olson also offer some useful guidelines for appropriate response time:

> - Frequent, simple commands should take less than a second.
> - For a given command type, response time should be as consistent as possible, even if it takes slightly longer. A maximum response time deviation of no more than 20 percent from the mean has been suggested.
> - If a response will take a long time (say greater than 10 seconds) a message should be issued within 1 or 2 seconds to give the user feedback. A novice user, in particular, may become frustrated or anxious if the system appears to be idle and waiting for an action.
>
> (Davis, 1985, p. 540)

Rubinstein tells us "when a computer needs a noticeable amount of time to accomplish a task, immediate acknowledgment of input tells users that they've successfully done their part" (Rubinstein, 1984, p. 147).

Rubinstein says:

> Some feedback is required for every user action. After two to four seconds, people start to wonder if what they have done has worked, or they begin to think about other things. The two-to-four second rule follows from observations of conversations. When one party in a dialogue stops talking, the other will usually commence. If there is an

> unanticipated silence in the conversation, the interchange quickly becomes very awkward. . . . When the response time is much longer than two to four seconds . . . the system should inform users of a rule change for the conversation. Options include a graphic indicator such as a timer clock, hourglass, or bar chart showing actual progress, or at a minimum a statement that it will take some time to finish what was requested. A user not adequately informed that the system is processing will start to think that something is wrong. A little feedback can go a long way toward improving the perception of good response without actually improving performance. . . . Irregular response times make a system seem unpredictable and can cause frustration. (Rubinstein, 1984, pp. 147-148)

Saiedian tells us, "when planning an interface that would adapt to the capabilities and limitations of people, the designers must consider the sociological context in which the software will be used (e.g., the users skills and background) as well as the psychological aspects (e.g., brightness, loudness) and physical variables (e.g., intensity)" (Saiedian, 1993, p. 23).

THE GRAPHICAL-USER INTERFACE

When designing a person-machine dialog, designers must decide whether to use commands that specify actions, to use menus from which the user chooses an action, or to develop a graphical user interface. Experienced users may prefer commands. Occasional users may have difficulty remembering commands and feel more at ease with a menuing system. Most everyone feels comfortable with a graphical-user interface.

All early microcomputers and most microcomputers today are command driven and use a **character-user interface.** The standard screen of today can display twenty-five rows of eighty characters each. Commands are typed in on the keyboard and appear on the screen. The operating system responds by executing the command and displaying the result of that execution in characters on the screen.

Schultheis and Sumner comment on this:

> Learning, remembering, and using many different keyboard commands can be intimidating to the user. To provide an easier method to talk to the operating system, some software firms have developed other operating environment software that provides an easier-to-use interface. For example, one type of operating environment uses a **graphical-user interface**, or **GUI** (pronounced gooey), to allow the user to interface with the operating system . . . A GUI environment employs dialogue boxes, drop-down menus, buttons, icons, scroll bars, and pointers instead of commands . . . Users move a pointer around the screen with a mouse to activate programs, data files, or features. Users 'point and shoot;' that is, they move the pointer to an icon and click a button on a mouse instead of keyboarding commands. Users on some

systems may also point to icons using a light pen or even their fingers, with a touchscreen system. Moving a pointer to an icon of a program to be run and clicking a mouse button is a lot simpler for the manager than learning and keyboarding a series of commands. Furthermore, in the near future, voice synthesizing capabilities may allow you to use your voice to activate the GUI icons. Instead of pointing to a screen feature such as an icon and clicking the mouse, you would simply say the word that represents the icon.

Operating environment software may also allow you to run more than one program and view more than one data file at once by splitting your display screen into parts called **windows**. You display may show a document you are editing using one program in one window and a budget you are creating using another program in another window . . .

Some operating environments offer windowing but do not use a graphical-user interface. For example, the Desqview operating environment allows you to window several programs and data files on a single screen . . . Desqview, like Windows/386, uses virtual memory to handle more than one program and data file. (Schultheis, 1992, pp. 132-133)

Loudon and Loudon agree:

Proponents of the graphical user interface claim that graphical user interfaces save learning time because

> computing novices do not have to learn different arcane commands for each application. Common functions such as getting help, saving files, or printing output are performed the same way. A complex series of commands can be issued simply by linking icons. Commands are standardized from one program to the next, so that using new programs is often possible without additional training or use of reference manuals . . . Graphical user interfaces also promote superior screen and print output communicated through graphics. (Loudon, 1994, p. 212)

The graphical-user interface is gaining momentum. In the October 13, 1992 edition of PC Magazine, Joel Dreyfuss states:

> "It has been the year of the graphical user interface. Both Windows and OS/2 have dominated the industry dialog as they jockeyed for hearts, minds, and CPUs to run on. Following the laws of momentum, software vendors have been feverishly shipping products for Windows - and to a far lesser extent, OS/2. DOS fanatics have felt neglected, and not without reason."
>
> "However, economic reality is now setting in. Windows may be sexy, but DOS is where most of the work gets done. Microsoft claims that 10 million copies of Windows are now in circulation and IBM says 1 million copies of OS/2 have now been shipped. Assume that 75% of these are in use most of the time and you get 8.25 million computer users who

> favor a graphical user interface. With an estimated 100 million PCs in use worldwide, some 92 million users are absolutely, positively non-GUI. The software community has finally realized the potential of that mature marketplace. Windows may be the fastest-growing software market, but DOS is still the biggest."

Dreyfuss goes on to report that several major vendors such as Microsoft, Lotus, WordPerfect, and Norton are updating old applications for DOS and developing new ones that "borrow liberally from the GUI concept without sacrificing the speed that endears text-based programs to many of us" (Dreyfuss, 1992, p. 4).

DOS is where most of the work gets done. A character-user interface is difficult. A graphical-user interface is fun. The best of both worlds would be to have a graphical-user interface that runs under DOS. Because of the extra system management layer that Windows loads under DOS, a program that can run under either DOS or Windows will run much faster under DOS.

OBJECT-ORIENTATION

Object Oriented Programming is becoming a "buzz" word for the nineties. Just what is "object oriented programming?" Just who is it that's saying it's "the wave of the future?"

Sheriff says: "Take all the rules you ever knew about structured programming, turn them inside out, and you've got object-oriented programming" (Sheriff, 1992).

Hawkins and Schaffhauser report: "According to some observers, object-oriented programming will soon be what everybody does. Object-oriented databases might become what everyone uses. So now might be a good time for you to find out what object orientation is all about" (Hawkins, 1992, p. 87).

Traditional programming utilizes a procedural language methodology. The data is kept separate from the instructions. It is called procedural because the program is written to tell the computer exactly what to do, one step after another. In object-oriented programming, the program and the data are joined together into something called an object. The object is a self-contained entity. A separate program can send a message to an object and that object will perform a procedure that is already embedded into it in accordance with parameters in the message. The same message can be sent to many different objects and each object may do something different. This is called "polymorphism." What an object does is dependent upon the contents of the message <u>and</u> the instructions contained within its own procedure. When I move a mouse so that the pointer points to a close button on a window and click on a mouse button, a message is sent to the window object that invokes that window object's close procedure and the window closes itself. The window disappears from the screen. The program that interpreted the mouse click and sent the message to the window object is called the "event handler." An event handler is always present in a graphical-user interface. It just sits there and waits for a user to do something.

An object's procedures and data are hidden from the other parts of a program. This is called "encapsulation." An object's data can only be manipulated from inside the object. An object's procedures are called its "methods." An object's methods can be changed internally without affecting the other parts of a program. Each object is independent and can be used in many different systems without changing its program code.

Object-oriented programming is based on the concepts of "class" and "inheritance." Classes are general categories of similar objects. A class is never used directly in a program. Objects are. A class is used for creating objects which are "instances" of that class. Objects belonging to a certain class "inherit" all the structures and behaviors of that class. Each object can be modified by adding variables and behaviors unique to the object. If I want to display something in a window, I first create a window object by declaring it to be an instance of the window class. Then I add variables and behaviors that are unique (such as background color, title bar, size, location, and what it is to contain) to the window object. New classes of objects can be created by choosing an existing class and specifying how the new class differs from the existing class.

Object-oriented programming is expected to reduce the time and cost of writing software by producing objects that can be reused in other systems. The productivity gains should be considerable. How considerable? Loudon and Loudon feel that, as of 1994, the experience in object-oriented software development is still too limited and it's too early to properly evaluate the methodology. They do tell us that:

> Nonetheless, early studies have been promising. Experience has shown that programming productivity gains of better than 10:1 are possible. Electronic Data Systems Corp. (EDS) studied the benefits by building a maintenance management system twice, once using structured techniques and once using object-oriented programming. EDS equalized the skill level of the two project teams and had them work from the same specifications. They found productivity improvement of 14:1 using object-oriented programming (IDC White Paper, 1992). Maintenance costs are also lowered by reducing multiple maintenance changes. For example, when the U.S. postal system changed the zip code from five to nine digits, each program within a company had to be changed. If a company's programs were object-oriented, the programmer would only have had to modify the code within the object, and the change would be reflected in all the programs using that object. (Loudon and Loudon, 1994, pp. 453-454)

OOP languages have been around since the 1960s. The reason they have recently become popular is that they seem to be especially effective for developing graphical-user interface applications. It's only been recently that personal computer hardware has become powerful enough to display a graphical user interface fast enough that the user isn't sitting there waiting for a screen to refresh so long that he or she notices the wait.

GRAPHICS

Turban says: "Graphics enable the presentation of information in a way that more clearly conveys to many managers the meaning of data and that permits managers to visualize relationships" (Turban, 1988, p. 162).

Awad reports: "The advent of graphics is having a significant impact on decision support in the organization. Communicating effectively through graphics means reducing ambiguities, crystallizing thought, and promoting consensus in decision making" (Awad, 1988, p. 312).

Kozar tells us: "Pie charts are best for showing parts of a whole. Line graphs show changes in a variable over time and the relationship between two variables. Bar charts can be used to show changes over time, parts of a whole, relationships between two variables, and comparisons between several items" (Kozar, 1989, p. 326).

Shore advises:

> When computer graphics are the right choice for output, there are many benefits to be achieved, such as
>
> - More effective conversion of data to information.
> - Easier recognition of relationships and trends.
> - Shortening of the time needed to make decisions.
> - Better organization of output.
> - Ability to focus attention on important issues.
> - Capability of presenting ideas in an attractive format that may more readily receive attention.
>
> Behind these benefits lies the fact that the mind can absorb information more rapidly from an effective picture than it can from words or numbers. If they are used when appropriate, computer graphics can bridge the gap between computer data and the human mind. To the business professional this means more information in less time. (Shore, 1988, pp. 294, 295)

There is a lot to producing a good graph. It is not easy. Kozar warns: "Care should be taken to ensure appropriate titles, labels, and scales. Keys should be carefully designed. All items on the graphs should be carefully located to avoid confusion, misunderstanding, or misinformation" (Kozar, 1989, p. 326).

Ives has developed a very complete set of guidelines for graphic design based on empirical research. He suggests the following:

Content

1. Display no more data than the user needs.
2. Labels should be consistent from display to display.
3. Labels should be readily distinguishable from each other.
4. Labels should be made up from common terms that are familiar to the target user population or from standard lists of terms developed for the particular user population.
5. Avoid humor in accompanying text.
6. Use whole words rather than abbreviations where space permits.
7. If abbreviations must be used, do not include abbreviation punctuation.
8. Avoid contractions such as 'won't' in labels and text.
9. Each display should contain enough title material to be comprehensible.
10. If the same chart form is being used multiple times (for example, to compare trends) then use the same set of scales for each instance of the chart.
11. Include zero on scales in which values are being compared; if only a small nonzero part of the scale is used, the chart may be misleading.
12. Values on grid scales should be selected so that interpolation is easy to do.
13. Avoid using two different scales on the same axis.
14. If two scales are used on the same axis, clearly mark

(perhaps with an arrow) which scale goes with which curve.

Fonts

1. Generally, a single type face should be employed for a presentation, although different forms of that type face (e.g., bold, medium, or light; condensed, regular, or expanded) may be employed to distinguish among various levels of labels.
2. Such changes should be limited to two or three levels.
3. Proportional variations in type font sizes should be simple and obvious. Primary versus secondary labels, for example, might differ by ratios of 2:1 or 3:1.
4. Use the same family of type styles across multiple charts being used in the same presentation.
5. Simple sans-serif type fonts are preferable...
6. Avoid italicized, stenciled, or Old English script type fonts.

Position and Size

1. Line curves should be directly labeled if there is room rather than relying on a key.
2. Make scales on line chart axis large enough to easily be read.
3. If room permits, labels should be placed within the colored or shaded areas they represent rather than in a separate key.
4. Use size or color coding to help differentiate between levels of label importance.
5. Use capital letters for labels and short instructions

because they can be read at a greater distance than capital and lowercase letters.

6. All labels should read from left to right. Avoid labels that read from top to bottom, that are on their sides, or that go around corners.

Displaying Text

1. Multiple lines of text should be displayed in lines of approximately 40 to 60 characters.
2. Unless the display device is capable of proportional spacing, justified right margins should not be used as they impede reading speed.
3. Normally long lines of text should be displayed with normal use of upper and lower case. This will improve readability as short words of mixed case can be recognized rapidly by their shape.
4. Avoid breaking words by hyphenation between lines.

Color Usage

1. For tasks involving item identification, color is a better coding strategy than anything except alphanumeric codes.
2. If a screen full of alphanumeric characters is broken into groups, blocks, or columns, then each can be made more discrete by separately coloring them.
3. Use color to differentiate two or more lines plotted on the same graph.
4. Color can be used as a prompt to indicate which information should be entered next.

5. Color is a good coding strategy for tasks involving screen searching and counting.

Color Choice

1. Data to be highlighted should be colored with bright colors, while 'normal' data should be colored in less noticeable colors. For some IBM devices, colors from most bright to least bright are white, turquoise, yellow, pink, green, blue, red.
2. Use similar colors for similar situations (e.g., red for severe problems, pink for cautionary situations).
3. Color frequently used fields differently from the rest of the data.
4. If important categories must be clearly differentiated, then assign strongly contrasting colors to them first, and then assign colors to the remaining code categories.
5. Avoid red-green, blue-yellow, green-blue, and red-blue color pairs.
6. The color green is often appropriate for representing normal data.
7. Use the color red for error messages, identifying problems, and indicating significant conditions.
8. Use high color contrast for character/background pairs.

Color Cautions

1. Irrelevant colors interfere with an individual's ability to perform a task involving item identification.

2. Use no more than four colors for novice users. Seven colors may be used effectively with experienced long-term users.
3. Red and green are not readily visible in the periphery of the eye. Material to be identified in this area should be coded as white.

Display of Icons/Symbols

1. An excellent coding scheme can be developed using up to 20 unique shapes.
2. A particular shape can be varied to indicate some change in an attribute in the object being represented (for example, the wake of a ship can be used to represent both direction of travel and relative speed).
3. The meaning assigned to symbols should remain consistent across displays presented together.
4. Special symbols such as asterisks or arrows can serve to draw attention to exceptional conditions.
5. Limit displays to two or three different sizes of symbols to represent intensity.
6. If multiple sizes are employed for the same symbol, the next larger symbol should be at least 1.5 times as big as the previous smaller one.

Display of Lines, Grids, and Axes

1. A coding scheme involving line widths should employ no more than three different line widths.
2. A coding scheme should consist of no more than four line types (i.e., solid versus varieties of dashed).

3. In plotting data curves, use dotted or dashed lines only for representing projections or extensions.

(Ives, 1982, pp. 15-47)

Ives provides a very comprehensive list of graphics guidelines.

The first step in the development of graphics, however, is to remember that the user is a person. Empathy for the user <u>and</u> the use of Ives' guidelines should result in the creation of graphical displays that are informative, easy to interpret, and beautiful to behold.

HELP SCREENS

If we want user interfaces to be intuitive, it is essential that we build a good help system. The best help system is one that is context sensitive. A context sensitive help system provides help to the user based on what the user is currently doing. (Saiedian, 1993, p. 22) Many people feel some anxiety in using computer-based systems. Without an effective help system, they may give up and turn back to pencil and paper methods. (Schultheis, 1992, p. 562)

Rubinstein advises:

On-line help is particularly difficult to design into systems, especially large, general-purpose ones. Which information to provide under various circumstances depends intimately on the user's knowledge, situation, and abilities. Ineffective help is expensive for developers and users alike and is worse than no help at all. Users encounter help facilities at a sensitive time. They are admitting that they need assistance and are actively seeking it. To be frustrated at this point by an ineffective help facility will permanently discourage many users.

If help is to be provided, it should be uniformly available. The user must be able to get help without backing up, retyping, losing text, or changing context. In other words, the help provided must not disrupt the task or put a strain on the user's short-term memory. The more disruption that occurs in the process of seeking help and returning to the original task, the less likely it will be used. (Rubinstein, 1984, p. 103)

In a graphical-user interface environment, the most appropriate help system would be one that consisted of context-sensitive help windows that would pop up at the click of a mouse, could be dragged around the screen so they wouldn't cover up what the user is looking for help for, and are accessible from anywhere in the program that they might me needed.

PROTOTYPING

Loudon and Loudon recommended prototyping for developing Decision

Support Systems (page 17). Kozar said that a prototyping philosophy was often required in developing effective person-machine dialogs (page 29). What is *prototyping*?

Jenkins tells us:

> Prototyping is a method for developing systems that requires a new philosophy or set of beliefs about systems development. Systems are built rapidly and are enhanced and changed as they are used until the system satisfies users. The method resembles engineering approaches to systems design, where a physical model of a system is built to allow user reaction. The user exercises versions of the system to understand and refine his or her needs. Prototyping calls for a less rigorous requirements definition. Since users may not be able to define requirements, the assumption is made that by giving them some physical results to react to, they will be able [to] 'zero in' on what they would find useful. (Jenkins, 1985, p. 2)

Sprague and Carlson advise:

> DSS need to be built with short, rapid feedback from users to ensure that development is proceeding correctly. They must be developed to permit change quickly and easily. The result is that the most important four steps in the typical systems development process (analysis, design, construction, implementation) are combined into a single step which is iteratively repeated. The essence of the approach is that

the manager and builder agree on a small but significant subproblem, then design and develop an initial system to support the decision making that it requires. After a short period of use (a few weeks), the system is evaluated, modified, and incrementally expanded. This cycle is repeated three to six times over the course of a few months until a *relatively* stable system is evolved which supports decision making for a cluster of tasks. (Sprague and Carlson, 1982, p. 15)

Kozar adds:

Prototyping allows reality testing. In essence, prototyping is based on the premise that people have trouble putting themselves into a "future state" and trying to imagine what the future would be like.

One should be sure what is meant when hearing the word 'prototyping' since it can mean different things to different people. There are different degrees or levels of prototyping. One degree is that the *output only* is created in a mock-up screen or report. For example, screens are shown to users who can react to them, but the screens were not created with actual data from a database. This approach also is called *heuristic development*.

Another level of prototyping is using a fourth generation language to actually create a working system using

> representative and 'live' data. This type of prototyping leverages modern software tools and decreasing costs of hardware. (Kozar, 1989, p. 427)

According to Johnson:

> Prototyping advocates make several assumptions about the development of computer systems:
>
> - It is difficult to do something right the first time.
> - It is easier to modify or improve something than to start from scratch.
> - Learning is gradual, iterative, and accomplished by example.
> - Users cannot visualize systems accurately until they see working models.
> - The experience and insight of people are the most important factor in creating new systems - not a prescribed methodology.
>
> (Johnson, 1990, p. 208)

Davis and Olson say: "The prototyping methodology . . . has several significant advantages in development of applications having high uncertainty as to requirements:

- Ability to 'try out' ideas without incurring large costs
- Lower overall development costs when requirements change frequently
- The ability to get a functioning system into the hands

of the user quickly

- Effective division of labor between the user professional and the MIS professional
- Reduced application development time to achieve a functioning system
- Effective utilization of scarce (human) resources"

(Davis and Olson, 1985, p. 570)

The development of a Decision Support System should be an iterative process and that is what the prototyping methodology supports. Management of a prototype development may be more difficult than management of a traditional development. There will be frequent changes.

There can also be a tendency on the part of the end-user to want to accept a prototype as the final product before it is.

DEVELOPING AN ACCOMPANYING TEXT

Integrating instruction, documentation, help, and error messages with the software design can only improve the software design.

Rubinstein tells us:

- Dealing early with these issues reduces the likelihood of discrepancies later. Because each component of the system relates to the whole in an important way, all parts will benefit from early attention.

- User documentation should be done in parallel with design . . . there is much to learn from documentation before the system is wholly designed, much less built.
- [If these things] are added later or developed separately they may not relate well to the whole. Many 'help' systems are barely used because they are seldom integrated with the rest of the product.
- The design of user documentation (such as primers, tutorials, reference manuals, and user's guides) should occupy an important place in the development process because documentation supports the user and is part of the human interface of the system.
- It is impossible to overstate the importance of examples. Users need concrete examples of how things work. They need scenarios that show how to use features and options. Users need concrete procedures that they can readily apply, detailed examples that can serve as the basis for more general learning. Examples are little pieces of procedure that people can pick up and use. (Rubinstein, 1984, p. 96-102)

Kozar tells us that, before we write an accompanying text, we should have answers to the following questions:

Why am I writing this?

Who is going to be using it?

What tasks will users want to accomplish by reading this?

What don't readers care about that I might be tempted to include?

What is the *minimum introduction* readers need?

What tasks are done by the user most frequently?

(Kozar, 1989, p. 294).

Answering these questions will give the writer the basic material he/she needs to write an accompanying text. Empathy for the user comes first. Then the writer can address the mechanics.

Kozar addresses the mechanics:

- use functionally cohesive modules to facilitate changes and updates. Functionally cohesive modules are 'chunks' of writing that address one and only one task. There is one task or function to be accomplished, not several of them. This will facilitate changes since you can 'unplug' one module and replace it.
- Keep sections, paragraphs, sentences, and words simple. Remember the letters KISS - Keep It Super Simple. (but not so simple it would offend someone).
- Include adequate referencing, indexes, and, if needed, a glossary.
- Write for content, not bulk. Be concise, but also repeat materials to eliminate the 'four-fingered bookmark problem' where the reader must 'see this section' or 'go to page xx.'...
- Utilize process logic tools such as decision trees,

> decision tables, or 'tight English' [a form of pseudo-code] if appropriate. Include illustrations, pictures, checklists, or tables if they help clarify.
>
> - Avoid jargon, abbreviations, theory, technical discussions, complexity, and items of poor taste such as puns or dialect.
> - Test the manual with practical exercises with representative users. Your own peers may not be representative of future users. Very few computer programs work on the first draft. Very few computer programs are completely debugged. About the same number of user manuals work on the first draft and are completely debugged. So, test any manual you write.
> - Before developing the manual, be aware of organization rules, guidelines, outlines, formats, and other standards that exist for reference manuals. The standards may include help in deciding on spellings, content, order, and other decisions with which the developer may be confronted. (Kozar, 1989, p. 295)

The help windows, screen dialogs, and the accompanying text are all part of the system design. They need to be developed as the system is developed. Making one work right can show deficiencies in another. The deficiencies can be corrected. The system can be revised. If prototyping is used, the end-user will have many opportunities to see if all these system components are in agreement. If they're not, the end-user will let the system developer know. Appropriate fixes can be made in the next prototyping cycle.

Proper Decision Support System design requires a truly "sociotechnical" approach.

CHAPTER TWO
REVIEW OF THE CURRENT LITERATURE: THE THEORY OF TEACHING MANAGEMENT INFORMATION SYSTEMS

Chapter two begins with a review of Nathaniel Borenstein's critique of college computer science courses. This is followed by a discussion of Fredric Margolis' Androgogy in Action, Malcolm Knowles thoughts on teaching adults, and Stephen Brookfield's characterizations of adult learners. The chapter closes with a statement on teaching Management Information Systems.

Nathaniel S. Borenstein (1992, p. B3) says, in industry:

> it has long been taken for granted that universities provide little practical training for computer programmers. Programmers often regard their formal training, if they have had any, as little more than a bad joke. Statements such as 'I learned more in four months on the job than in four years of college' are so common as to lead almost inescapably to the conclusion that something is seriously wrong with the way programmers are educated in our universities today.

Borenstein goes on to say: "The principal problem is that the skills taught in university computing programs are astonishingly irrelevant to the tasks of programmers in the real world."

Borenstein tells us that students learn about data structures, analysis of algorithms, program verification, program syntheses, and the mathematical theory of computation. He argues that:

> while it is fascinating from a mathematical viewpoint to be able to characterize the intrinsic computational complexity of a problem, it is considerably more rewarding for the programmer simply to solve it in the best possible manner - where 'best' often refers at least as much to the speed with which the solution can be implemented as to the ultimate performance of the overall program.

(Borenstein, 1992, p. B3)

Computer science has its origins in two very distinct academic disciplines, mathematics and electrical engineering. The age of computing was born when Alan Turing and John von Neumann had a series of insights into the nature of problems that can be characterized formally and solved via formal processes. The computing era began when mathematicians formally proved that relatively simple machinery could solve any computable problem (Borenstein, 1992, p. B3).

It turns out that, in order to solve <u>any</u> computable problem, these machines would have to have infinite time and memory. In the real world, the value of a computing machine is directly related to the size of its memory and the speed of its operations. Improving the practical capabilities of real machines quickly became the province of electrical engineers.

The engineers were very practical people. They avoided formalisms and promoted whatever worked well and quickly. Although their methods worked well for computer hardware, they were less successful with software. Software, by default, became the domain of the mathematicians (Borenstein, 1992, p. B3).

Borenstein has zeroed in on the fundamental problem with the way computer science courses are taught. Computer Science is generally taught by mathematicians and, in fact, in many universities, the Computer Science program is found in the math department. At Capital University in Columbus, Ohio, for example, the department is named MATH/COMPUTER SCIENCE/PHYSICS. Capital offers approximately twenty computer science courses.

There is a related discipline. It is called management information systems. Management information systems are systems that produce information that can be used in decision making. This information is used to assist managers in performing their basic functions of planning, organizing, leading, and controlling. The overall purpose is to provide the right information to the right manager or decision maker at the right time. The objective of an MIS is to improve the efficiency of an organization. The focus of management information systems development is on "practical application."

Management Information Systems courses are generally found in the business department. That is, if they exist at all. At Capital

University there is one undergraduate MIS course. The emphasis is on "using the personal computer as a business tool." The course deals specifically with wordprocessing, spreadsheet analysis, graphics, and data base management. This hardly scratches the surface of what MIS can do. However, the course is practical application oriented.

In Androgogy in Action, a collection of case studies, Fredric H. Margolis (1984, p. 46) recounts:

> . . . simply teaching the prescribed knowledge is not sufficient. If you are involved in information-based training programs long enough, you begin to hear revealing comments from managers and participants:
>
> > "These people have learned the principles, but they don't know how to apply them."
> >
> > "This training program doesn't reflect the real world."
> >
> > "Some of the information was useful, but most of the session was over my head."
> >
> > "I knew most of that stuff before I came to the course."
> >
> > "Too many lectures and slides. After a while, it all seemed the same."

Margolis sounds a lot like Borenstein. There appears to be a difference, though. Borenstein addresses the very specific problem of teaching computer science. Margolis is talking about adult training programs. But, is there really a difference? The comments found in the Borenstein article were not made by students in a university setting. They were made by students after they had graduated, were working in the real world, and were trying to apply what they had learned in school. They were now adults. They had an adult perspective.

Knowles tells us that: "all of the great teachers of ancient times - Confucius and Lao Tse of China/ the Hebrew prophets and Jesus in Biblical times; Aristotle, Socrates, and Plato in ancient Greece; and Cicero, Evelid, and Quintillian in ancient Rome - were all teachers of adults, not of children" (Knowles, 1990, p. 27).

Knowles describes how the ancient teachers taught:

> They perceived learning to be a process of active inquiry, not passive reception of transmitted content. Accordingly, they invented techniques for actively engaging learners in inquiry. The ancient Chinese and Hebrews invented what we would now call the case method, in which the leader or one of the group members would describe a situation, often in the form of a parable, and jointly they would explore its characteristics and possible resolutions. The Greeks invented what we now call the Socratic dialogue, in which

> the leader or a group member would pose a question or dilemma and the group members would pool their thinking and experience in seeking an answer or solution. The Romans were more confrontational: They used challenges that forced group members to state positions and then defend them. (Knowles, 1990, p. 27)

According to Knowles (1990, p. 28), starting in the seventh century in Europe, schools began being organized for teaching children. The primary purpose of these schools was to prepare young boys for the priesthood. "Since the teachers in these schools had as their principal mission the indoctrination of students in the beliefs, faith, and rituals of the Church, they evolved a set of assumptions about learning and strategies for teaching that came to be labeled 'pedagogy' - literally meaning 'the art and science of teaching children.'" This model of education replaced that of the ancient teachers. It is the basis of organization of our entire educational system today.

Shortly after World War I, educators in both North America and Europe started reconsidering the unique characteristics of adults as learners. The American Association for Adult Education was founded in 1926. Edward L. Thorndike published *Adult Learning* in 1928 and *Adult Interests* in 1935. Herbert Sorenson followed with *Adult Abilities* in 1938. By the onset of World War II, adult educators had scientific evidence that adults could learn and that they possessed interests and abilities that were different from those of children (Knowles, 1990, p. 28).

Knowles credits Sigmund Freud with the identification of the influence of the subconscious mind on behavior. Some of Freud's concepts such as anxiety, repression, fixation, regression, aggression, defense mechanism, projection, and transference have had to be taken into account by learning theorists. Freud saw the human being as a dynamic animal which grows and develops through the interaction of biological forces, goals, purposes, conscious and unconscious drives, and environmental influences (Knowles, 1990, p. 38).

Knowles considers Carl Jung to have advanced a more holistic conception of human consciousness. Jung introduced the notion that the human consciousness possesses four functions - or four ways of extracting information from experience and achieving internalized understanding - sensation, thought, emotion, and intuition. Jung's plea for the development and utilization of all four functions in balance laid the groundwork for the concepts of the balanced personality and the balanced curriculum (Knowles, 1990, p. 38).

Knowles (1990, p. 39) tells us that Eric Erikson conceived the concept of the "eight ages of man." The last three ages occur during the adult years and provide a framework for understanding the stages of personality development. The eight ages of man are:

1. Oral-sensory, in which the basic issue is trust vs. mistrust.
2. Muscular-anal, in which the basic issue is autonomy vs. shame.
3. Locomotion-genital, in which the basic issue is initiative vs. guilt.
4. Latency, in which the basic issue is industry vs. inferiority.
5. Puberty and adolescence, in which the basic issue is identity vs. role confusion.
6. Young adulthood, in which the basic issue is intimacy vs. isolation.
7. Adulthood, in which the basic issue is generativity vs. stagnation.
8. The final stage, in which the basic issue is integrity vs. despair.

It is the clinical psychologists, especially those who identify themselves as *humanistic*, who have concerned themselves most deeply with the problems of adult learning. The humanistic psychologists are concerned with the study and development of *fully functioning persons* (to use Rogers' term) or *self-actualizing persons* (to use Maslow's) (Knowles, 1990, p. 39).

Whereas "pedagogy" means "the art and science of teaching children," "andragogy" refers to "the art and science of teaching adults."

In Andragogy in Action (1984, p. 9), Knowles describes the assumptions inherent in the andragogical model:

1. *Regarding the concept of the learner*: The learner is self-directing. In fact, the psychological definition of adult is 'One who has arrived at a self-concept of being responsible for one's own life, of being self-directing.' When we have arrived at that point, we develop a deep psychological need to be perceived by others, and treated by others, as capable of taking responsibility for ourselves. And when we find ourselves in situations where we feel that others are imposing their wills on us without our participating in making decisions affecting us, we experience a feeling, often subconsciously, of resentment and resistance.
2. *Regarding the role of the learner's experience*: The andragogical model assumes that adults enter into an educational activity with both a greater volume and a different quality of experience from youth. The greater volume is self-evident; the longer we live, the more experience we accumulate, at least in normal lives. The difference in quality of experience occurs because adults perform different roles from young people, such as the roles of full-time worker, spouse, parent, and voting citizen.
3. *Regarding readiness to learn*: The andragogical model assumes that adults become ready to learn when they experience a need to know or do something in order to perform more effectively in some aspect of their lives. Chief sources of

readiness are the developmental tasks associated with moving from one stage of development to another; but any change - birth of children, loss of job, divorce, death of a friend or relative, change of residence - is likely to trigger a readiness to learn. But we don't need to wait for readiness to develop naturally; there are things we can do to induce it, such as exposing learners to more effective role models, engaging them in career planning, and providing them with diagnostic experiences in which they can assess the gaps between where they are now and where they want and need to be.

4. *Regarding orientation to learning*: Because adults are motivated to learn after they experience a need in their life situation, they enter an educational activity with a life-centered, task-centered, or problem-centered orientation to learning. For the most part, adults do not learn for the sake of learning; they learn in order to be able to perform a task, solve a problem or live in a more satisfying way. The chief implication of this assumption is the importance of organizing learning experiences (the curriculum) around life situations rather than according to subject matter units . . . Another implication is the importance of making clear at the outset of a learning experience what its relevance is to the learner's life tasks or problems.

5. *Regarding motivation to learn*: Although it acknowledges that adults will respond to some external motivators - a better

job, a salary increase, and the like - the andragogical model predicates that the more potent motivators are internal - self-esteem, recognition, better quality of life, greater self-confidence, self-actualization, and the like.

Brookfield says:

> [There are] four characteristics of adult learners - their special orientation to learning, their experiential base, their particular developmental changes and tasks, and their anxiety regarding learning - generate . . . certain conditions for learning. Adults learn best when they feel the need to learn and when they have a sense of responsibility for what, why, and how they learn. Adults use experience as a resource in learning so the learning content and process must bear a perceived and meaningful relationship to past experience. What is to be learned should be related to the individual's developmental changes and life tasks. The learning method used will foster, to different degrees, the adult's exercise of autonomy. Adults will, however, generally learn best in an atmosphere that is nonthreatening and supportive of experimentation and in which different learning styles are recognized. (Brookfield, 1986, p. 30)

Knowles reflects on different atmospheres of learning:

> I have often wondered why, among the many kinds of

> organizations I work with, those that seem most consistently open to new ideas and approaches are business and industry - especially large corporations. My guess is that the answer is competition. In order to be competitive, corporations must keep up with, if not get ahead of, their rivals in the marketplace. And in the case of productivity and sales, the effectiveness of personnel is at least as critical as machinery and advertising. For this reason, corporations have been among the most innovative in applying modern concepts of adult learning in their human resources development programs. (Knowles, 1984, p. 23)

One modern concept of adult learning is to balance theory and practical application. Brookfield explains:

> One of the most frequently offered criticisms of programs of professional preparation by graduates who subsequently inhabit the 'real world' of practice is that such programs are strong on theory but weak on practical application. It is not unusual to hear practitioners declare that their first few months of practice were spent unlearning the lessons of graduate training programs. Indeed, the low esteem and credibility granted to programs of professional educator preparation is one of the most significant commonalities of practitioner's attitudes that I have noticed across the three countries in which I have worked as an educator and trainer of adults (Britain, America, and Canada). A common view is that one must attend university-

sponsored programs of professional preparation and in-service development because of licensing requirements and for the possibilities of promotion and salary increases this brings. The idea that one might become a more insightful or effective practitioner as a result of attendance at such courses is greeted with an amused skepticism.

Nowhere is this theory-practice disjunction more evident than in the realm of program development for adult learners. As a professor who has taught many program development courses to educators and trainers of adults, I can attest to the frequency with which participants in these courses (who are mostly practitioners with several years of experience) state that they "break the rules of good practice" or "disregard theory for the *real* world of practice." It is as if such participants will tacitly agree to attend courses, conferences, and workshops on program development for the various career rewards this brings, but sit through such experiences only with a sustained suspension of belief. They give the appearance of battle-weary veterans of trench warfare; the skepticism with which they view neat textbook models of program development is grounded in their years of experience dealing with organizations in which personality conflicts, political factors, and budgetary constraints constantly alter neatly conceived plans of action. (Brookfield, 1986, p. 202)

Brookfield sounds a lot like Margolis who sounds a lot like Borenstein.

Knowles (1984, p. 48) summarizes an andragogical approach to teaching. He suggests that an andragogical approach:

- Emphasizes the skills of analysis and decision making through a series of job-related cases or problems.
- Establishes a learning approach rather than a teaching approach by a series of planned structured activities enabling the learner to acquire the appropriate knowledge.
- Is a practical, job-based approach which keeps the learners constantly aware of the value of the training program to them and their work. (Knowles, 1984, p. 48)

The development of management information systems should be application focused and end-user driven. The delivery of upper level undergraduate and graduate coursework in management information systems should include a strong practical application focus. Good teaching involves application of both pedagogical and andragogical principles. The authors' works discussed in this chapter reflect a need for the balance of theory and practice.

CHAPTER 3
METHODOLOGY USED IN THE SOFTWARE DEVELOPMENT

Chapter three begins with definitions of operations management and decision support systems. This is followed by a discussion of end-user satisfaction criteria and a listing of the reasons for choosing Clipper as a programming language. The balance of the chapter is a chronological review of the system development from the personal perspective of the system developer.

Operations Management is the systematic direction and control of the processes that transform resources into finished goods and services.

Decision Support Systems are computer-based systems that help decision makers confront ill-structured problems through direct interaction with data and analysis models. The major impact is on both structured tasks and unstructured tasks that require managerial judgment. The objective of a DSS is to improve the effectiveness of managerial decision making.

The purpose of this Project Demonstrating Excellence was to develop and demonstrate a general purpose, user-friendly, decision support system for operations management.

The first step was to define end-user satisfaction criteria:

1. The software should be executable from DOS. The user should not have to load a spreadsheet analyzer such as Lotus or an interpreter such as BASIC before he/she has access to the system.
2. The software should be "user-friendly" in accordance with Kozar's description of "user-friendly" (see pages 25-26).
3. The software should use a graphical-user-interface as opposed to a character-user-interface (see pages 37-41).
4. The software should use graphics and color in accordance with the guidelines set out by Ives (see pages 45-52).
5. The software should be able to display <u>and</u> print <u>high-resolution</u> graphics where appropriate.
6. The software should display a window full of data records as opposed to one data record at a time.
7. Once a data file has been loaded, the user should be able to display, modify, print, and solve without having to load the data file again, each time.
8. The software should not crash if the printer is not turned on and on line.
9. Manually entered data should be saved automatically. It should not be left to the user to have to remember to save.
10. The number of messages like "Correct Y/N (Y)?:_" should be kept to a minimum. The software should make it easy to go back and change the contents of a field.
11. The examples in the accompanying text should match what is seen on the screen.

The following Operations Management Decision Support System software packages were reviewed:

Attaran, Mohsen. Operations Management Information Systems. New York: John Wiley & Sons, Inc., 1992.

Chang, Yih-Long and Robert S. Sullivan. Quantitative Systems for Business Plus. version 2.0. Englewood Cliffs, NJ: Prentice Hall, 1991.

Finch, Byron J. and Richard L. Luebbe. Spreadsheet Applications for Production and Operations Management: An End User Approach to Problem Solving, Modeling, and What-If Analysis. Homewood, IL: Irwin, 1990.

Gupta, Sushil K., Stelios H. Znakis, and Tomislav Mandakovic. POMS: Production and Operations Management Software. Boston, MA: Allyn and Bacon, 1988.

Lotfi, Vahid and C. Carl Pegels. Decision Support Systems for Production and Operations Management. 2nd ed. Homewood, IL: Irwin, 1991.

Render, Barry, Ralph M. Stair, Jr., Mohsen Attaran, and William Foeller. Microcomputer Software for Management Science and Operations Management. Needham Heights, MA: Allyn and Bacon, 1989.

Each software package was found to be lacking in one or more of the satisfaction criteria. For example, none of the packages had a graphical user interface.

Clipper was chosen as the programming language for the following reasons:

1. Clipper is a high-level language that was developed for the efficient and effective design of data base management information systems.
2. Clipper programs can be compiled and linked into machine executable .EXE files.
3. Clipper has an open architecture and is supported by more third-party add-on libraries than any other X-Base language.
4. Clipper has consistently received very good to excellent reviews in trade journals and magazines such as PC Magazine and The Data Based Advisor.
5. While Clipper is not an object-oriented language, it does have the hooks necessary for third party libraries to be added on which would provide the ability to do object-oriented programming.
6. Clipper is supported by third-party libraries which could provide the ability to develop a graphical-user interface and display and print graphical solutions.
7. Clipper has a TBrowse() function which improves the appearance of database maintenance screens.
8. Clipper has the ability to pass code blocks as parameters to

functions which facilitates an object-oriented programming style.

9. Clipper is written in C. Sub-routines written in C or Assembler can be linked into the final executable system.

The following modules are typically included in operations management decision support systems. They are also typical of the tools that are taught in an operations management course.

Time Series Forecasting Analysis
Regression Analysis
Facilities Layout Analysis
Location Analysis
Linear Programming
Transportation Model
Assignment Model
Line Balancing
Inventory Analysis
Aggregate Production Planning
Material Requirements Planning
Statistical Process Control
Gantt Charts
Critical Path Method
Program Evaluation and Review Technique

It was decided to include these modules in the new decision support system.

An on-line database search was made of doctoral dissertations on Operations Management and Decision Support Systems over the last ten years. The search came up with one hit. The hit was a 1987 dissertation on A Proposed Methodology for the Design of Decision Support Systems in Operations Management by Ahmet E. Serpen at Brunel University in Great Britain. The dissertation was ordered and reviewed. The primary functional difference between Serpen's work and this Project Demonstrating Excellence was that Serpen "proposed" a methodology for a decision support system in operations management and the purpose of this PDE is to actually develop one. This was a case of "practice" being overwhelmed by "theory."

The January/February 1993 issue of The Journal of The Decision Sciences Institute had a couple of good related articles and what looked like several excellent dissertation references. One of these was inter-library loaned. The other was unpublished and not available. The one received and reviewed was a 1985 dissertation on "A Knowledge-Based Decision Support System for Managerial Problem Recognition" by Nassar H. Ata Mohamed at Texas Tech University. Mohamed's dissertation focused on the early phases of decision making (i.e., problem finding and problem diagnosis). This PDE focuses on a later phase of decision making (finding a solution).

The technical approaches to systems design were already understood. A research effort was undertaken to locate and review textbooks and scholarly articles on the behavioral approaches.

Work began on developing the graphical user interface. This activity rapidly became an iterative process. Code would be written, it would be tested, modified, and tested again. This is what is called "prototyping." "**Prototyping** is a method for developing systems that requires a new philosophy or set of beliefs about systems development" (see pages 53-57). Prototyping is well suited for developing a graphical user interface.

Agonizing choices had to be made: the right color for a push button, the right word to use in a screen display dialog. Push buttons, radio buttons, check boxes, menus, sub-menus, windows, groups, dialog boxes, help screens, scroll bars, title bars, and close buttons were incorporated. The use of minimize and maximize buttons and icons were considered. A clock, a calendar, and a calculator were developed. A set of utilities that would let a user personalize the system with his/her own name and zap all files was designed. Various foreground colors and background colors, window size and shape, and different window characteristics were experimented with. The interface was to look a lot like Windows but, at the same time, not have all the features of Windows.

For example, a user should not be able to pop up an overlaid window. If the user could do this it could become too confusing in this application. The logic of what appeared where had to be controlled. The user had to be restricted to going forwards or backwards. That's all. - And the user shouldn't be able to size or move any module system windows either. What displayed where had to be totally controlled. Otherwise, it would be too confusing to the user and (worse yet) wouldn't agree with what was in the accompanying text.

The first two months were filled with design and experimentation, design and experimentation. A good 200 hours were invested in developing the initial graphical user interface.

The opening pages and the first chapter of the accompanying text were written. The first chapter was called "Overview". It described computer system requirements, system installation, and how to get started. Chapter one walked a user through the menuing system, the help screens, the utilities, and how the graphical user interface works. Three different ways of incorporating screen prints into the text were experimented with. - And then the opening pages and the first chapter were rewritten. - And then rewritten again. Again, agonizing choices were made over the use of the exact right word.

Then, design began on the first module: "Time Series Forecasting." Technological approaches were employed in determining how to calculate solutions. Behavioral approaches were applied to determine how best to interface with the user.

Different ways of displaying and printing graphical solutions were experimented with. It was decided to provide ten increments on each axis and have the program calculate the scale of each axis and the number at which each axis would begin based on the data being used. The actual and forecast lines had to look different on both the display screen and the printer. A solid red line was chosen for the actual and a dashed green line for the forecast. The different colors stood out quite nicely from each other on the screen. The solid and dashed lines stood out quite nicely from each other on the printer, even when using a black only dot-matrix. White text and axis lines on a black background was used for the display, and black text and axis lines on a white background for the printer. The printed graph looked just like the displayed graph. It was clean. It looked good.

Three different forecasting techniques were allowed for: Moving Averages, Single Exponential Smoothing, and Double exponential Smoothing. A user could choose a three-month moving average forecast and look at the graph. He or she could then go back, pick a four-month moving average forecast, and look at the graph again. He/she could visually see which was the better predictor. The choice of a solution became interactive.

The ability to print out the original data and the problem solution was added. A second person ran the system. She looked at the problem solution screen and said: "I don't know what 'error' means. Did I do something wrong?" The system developer knew what it meant. Anyone who knew something about time series analysis would know what it meant. But, the intended end-users would be students who were in the learning stage and they could become confused. "Error" was changed to "forecast error." This second person became a major reviewer as work progressed through the system design.

Chapter two of the text was written.

Work moved on to the second module: "Regression Analysis." Simple linear regression analysis and multiple regression analysis with one or two independent variables could be handled quite easily. Multiple regression analysis with more than two independent variables required the use of matrix math. Clipper did not have matrix math functions. Very few programming languages did. Clipper did, however, have array functions. It was realized that a mathematical matrix was an array. Work was undertaken to represent mathematical matrices as numerical arrays and then to write the code necessary to create mathematical matrix functions. It worked. The Regression Analysis module handled up to six independent variables and up to ninety-nine observations.

The behavioral approaches to systems design continued to be of paramount importance. It was important to achieve a great degree of consistency between modules. A couple of pieces of code in

Time Series Forecasting were modified to agree with a better way of doing something that had been discovered in the work on Regression Analysis. A graphical solution capability was added to Regression Analysis following the same design parameters that had been developed in the work on Time Series Forecasting. This was, again, "prototyping."

Chapter three of the text was written.

The third module, "Facilities Layout Analysis," was developed. Portions of this module were real number crunching intensive. There were long periods of screen inactivity. In accordance with what Kozar had said about interim responses (see page 28), dialog messages were designed to appear on the screen and describe what part of the solution was being evaluated and what the intermediate results were as the computer worked its way through each step of the code. A full screen graphical solution was not appropriate for Facilities Layout Analysis. However, since a graphical user interface was being employed, it was possible to develop a combination of a text and graphics display that, in effect, carved out three little windows within the problem solution window in which were displayed the beginning layout, the layout being evaluated, and the best layout. The flashing of different departmental assignments as they were being evaluated was expected to become a real "crowd pleaser."

The first and second modules were revised in accordance with what had been learned in designing the third. The system was coming together and it was looking nice.

Chapter four of the text was written.

The fourth module was "Location Analysis." Technological problems started to develop. This module had been originally laid out to handle 16 facilities and 16 locations. As the code was developed, conventional memory kept being exhausted and the system would crash. The module was cut back to 14 and 14 and it worked. Then more code was written and it crashed. The module was cut back to 12 and 12, then 10 and 10. In mid-July it was at 4 and 4. The module was losing its effectiveness. Sixteen and 16 could be run with a character user interface but a lot of the attractiveness and uniqueness of the DSS is in its graphical user interface. Graphical user interfaces gobble up a lot of memory. The design was stuck. It couldn't do what it was intended to do.

Blinker was being used to link the Clipper compiled code. It was, by far, the best linker around. Blinker had too many special features that were being used to even consider using another linker.

Then, in mid-July, Blinker announced version 3.0 which they planned on shipping in August. Blinker 3.0 was to include a DOS extender that would permit the Decision Support System to run in "protected" mode on a 286 or higher machine. This would let the

DSS run in 256K of conventional memory! The memory problems would be resolved. All bets were placed on Blinker 3.0. The Time Series Forecasting and Regression Analysis modules were disconnected. Development continued "in the dark." Code was written for 16 facilities and 16 locations, but only parts of the module could be tested at one time. The whole module couldn't be tested at once without the system crashing.

A call was placed to Blinker during the last week in July. Blinker advised that 3.0 would ship in September. An order was placed and the money sent in. Work continued on the development of modules and the writing of text chapters. As each module was worked on, all the others had to be disconnected. Memory problems plagued the system development throughout July, August, September, and October. Modules were developed and chapters written for Linear Programming, Transportation Model, Assignment Model, Line Balancing, Inventory Analysis, Aggregate Production Planning, Material Requirements Planning, Statistical Process Control, Gantt Charts, and Critical Path Method.

In September, Blinker advised: "early Fall is the release date." Blinker sent out a letter the last week in September saying that they were shipping "the last week in September." A call was placed the first week in October. They now said they were shipping about 10/22/93. A call was placed on 10/22/93. Blinker said it would be a few more weeks.

The fifteenth module: "Program Evaluation and Review Technique" was developed. The sixteenth chapter of the text was written. The first draft was finished. There was concern about Blinker. Some of the smaller modules had been kept together, but the whole system could still not be run as a single executable file. The executable file was divided into four parts and each part would run separately. That meant there would be three or four modules in each. It really wasn't one complete system. But it was better than having to run each module on its own.

Texts by Knowles and Brookfield on Andragogy and a text on Synergogy by Blake and Mouton were reviewed. The system design was modified. The user interface was improved. A few more utilities were added such as the ability to change the default data disk drive and select a graphics printer.

The system design was reviewed a third time. A check was made to insure choices of color patterns were consistent. The system was checked for consistent and parallel structures in each of the modules. Necessary changes were made. The review of reference materials continued and some small code modifications to improve the user interface were made in accordance with the behavioral approaches to systems design.

A major technological breakthrough was realized during Thanksgiving week. The Decision Support System opened with a color screen that included graphical representations of several modules from within the system. Experimentations had been carried out last May on getting this to work properly. The only way it would work was to swap out the main program, call in a second program to display the screen, and then swap out the second program and call back in the main program. There had to be a simpler way. Experimentations continued. A new way was found. The start-up screen could display without leaving the main program. This meant the Blinker swap command was no longer needed. The source code was recompiled and relinked. The load size was 27K smaller! That was unexpected. Thirteen of the fifteen modules could now be fit into 1 executable file. Two executable files could now be shipped to the evaluators instead of four. Three disks instead of five.

Five copies of the text and the software were made, evaluation sheets were prepared, packets were assembled, and the packets were mailed out to the evaluators. They had the system in their hands before the end of November. They were requested to return their evaluations by December 31.

Blinker shipped on November 29th.

There had been concern that, just because Blinker 3.0 would permit the linked Clipper code to run in protected mode, there was no guarantee that the current releases of the third party libraries that had been used to do graphics, generate random numbers, and provide a graphical user interface would be compatible. Vendor bulletin boards were accessed. They were all working on Blinker 3.0 compatibility. No one would promise a shipping date.

In the mean time, it needed to be determined if Blinker 3.0 had a chance of providing what was needed: more available memory. A little test program was written that would display a message telling how much memory was available from within a program. The test program was compiled and linked under Blinker 2.1. The test program was run. It said there was 321K available. The test program was relinked under Blinker 3.0 and run again. This time, there was 2,980K available! That's almost three megabytes! It worked! What had been banked on since July still held promise.

The entire Operations Management Decision Support System was then pulled together, recompiled, and relinked under Blinker 3.0. The link was successful. The executable file size was 1,061,230 bytes. That would fit on one 3-1/2" high-density disk. All the other system files could be put on a second 3-1/2" high-density disk. That's exactly what was needed. The system would ship on two disks. That was practical.

Then an attempt was made to run the recompiled and relinked system. The system loaded but would not execute. It was expected that this would happen. An inspection of the hexadecimal code traced the error to a dGE Graphics function call. The current version of the dGE Graphics library would not work. That, too, was expected. A call was placed to the vendor. They said they didn't know when their Blinker 3.0 compatible package would ship, but an order could be placed for version 5.0c now and the compatibility upgrade should be downloadable and free. 5.0c was ordered. Five days later 5.0d was available. It was downloaded from their Electronic Bulletin Board, patched in, recompiled, relinked, and then an attempt was made to run the system.

This time, the start-up screen displayed. - And then, the system hung up. Nothing further would display and the system returned to DOS. The hexadecimal code was again inspected and it was determined that, this time, the error was a Visual Interface function call. Visual Interface is the add on library that gives the ability to provide a graphical user interface. The vendor was called. The vendor would give no promise on the shipping date for the compatibility upgrade. The vendor would not accept a prepayment but did agree to call as soon as the upgrade shipped.

The random number generator that is used in two of the modules comes from another third party library: Funcky. A test program incorporating the Funcky random number generator was written, compiled, and linked under Blinker 3.0. It worked. There was no compatibility problem.

Visual Interface 2.0 shipped on February 11, 1994.

All the development tool upgrades that were needed to run the Operations Management Decision Support System in "protected" mode were now in hand. Work began on pulling it all together. It was discovered that not only were the development tools upgraded to be compatible with Blinker 3.0, but that they were also enhanced in other ways. Each one of the new software tools interfaced with each of the other tools in a slightly different manner than they had before. Each one of the 41 separate programs in the Decision Support System had to be revised. The revisions were made.

By March 16th, another problem was realized. The system would run in "protected" mode all right, but only with 14 of the 15 modules. When the 15th module was pulled in, the system would crash with a "General Protection Fault" error message. One Blinker command that should have solved the problem would not work. A call was placed to the vendor. It was found out that this command was "documented but not incorporated into the current release." Scratch Blinker.

A competitive "protected" mode linker had recently been introduced. It was ordered. ExoSpace arrived on March 22nd. Three telephone calls had to be made to the vendor to solve problems not covered in the manual. ExoSpace worked.

A final "protected mode" product exists. There is one executable file. The entire system, including all files, fits on one disk.

CHAPTER FOUR

STUDENT AND FACULTY TESTING OF THE SOFTWARE AND TEXT

Chapter four begins with a sketch of the decision support system evaluators and a rational for why they were chosen. This is followed by a description of the evaluation form and a review of the overall evaluation scores. Most of the balance of the chapter is devoted to a summary of the evaluator's suggestions for improvement and the author's responses. The chapter closes with a summary of evaluator comments and a statement on implications for future research and study.

The Operations Management Decision Support System evaluators:

- **Dr. Robert C. Banasik:** Adjunct Professor, member of the Doctoral Committee, The Union Institute and Associate Professor in the Graduate School of Administration at Capital University.
- **Gayle DeGennaro:** MBA student in the Graduate School of Administration at Capital University.
- **Scott Roseberry:** undergraduate student in the College of Arts and Sciences at Capital University.

These individuals were chosen because they represent undergraduate and graduate level students, teaching faculty, and the Doctoral Committee. All are familiar with the Operations Management discipline. All have used an Operations Management

Decision Support System before.

The evaluation form consists of three pages. Page 1 is an evaluation sheet listing eight parameters that are considered to be important measures of a successful Operations Management Decision Support System. Each parameter can be rated from 1 (poor) to 10 (excellent). Page 2 provides a place for evaluator suggestions for system improvements. Page 3 gives the evaluator the opportunity to comment on anything he/she doesn't think was adequately covered on page 1 or page 2.

The complete evaluations can be found in the Appendix.

The following table shows the average of the ratings of the three evaluators for each of the parameters:

PAGE 1, EVALUATIONS:	Avg Rating
APPROPRIATENESS FOR SUPPLEMENTING AN OPERATIONS MANAGEMENT CLASS	9.7
EASY TO FOLLOW INSTRUCTIONS	8.3
COORDINATION OF TEXT WITH SOFTWARE	9.0
USER-FRIENDLINESS	9.7
CONSISTENCY FROM MODULE TO MODULE	9.3
INTERESTING, COLORFUL USER INTERFACE	9.0
ERROR-FREE	7.3
COMPARISON TO OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE	9.7
OVERALL COMPOSITE RATING	**9.0**

Any rating below a 9.0 is considered to be not satisfactory and requires an explanation. There are two:

Easy to follow instructions: The undergraduate student awarded an eight, the graduate student a seven, and the professor a ten. The graduate student has evaluated software before and has developed a strong preference for a certain style of instruction. It is believed that her assignment of a seven reflects her preference for this different style of instruction rather than the ease with which this system's instructions can be followed.

Error-Free: The undergraduate student awarded a seven, the graduate student a seven, and the professor an eight. All three experienced memory problems which caused the system to abort. In every case, they were able to recover and go on. The memory problems would not have occurred if the system had been running in protected mode. This solution was not available at the time of the evaluation. This was explained to all the evaluators but it is suspected that they marked down for it anyway.

PAGE 2, SUGGESTIONS FOR IMPROVEMENT:

	Suggestion	Response
Banasik:	EOQ - need to consider Total Annual Cost at bottom of feasible region.	This is an enhancement to the system. Other competitive products do not do this. Later. After the Ph.D.
	Linear Programming - start with a word problem.	Added second problem at end of chapter.
	Need a help screen on "Access a File."	Added instructions to window.
	When hit SETUP and no file picked, just goes back to previous screen. Need to display a message.	Message added to help window.
	Consider overlaying main menu title bar so user can't attempt to click on the wrong help window.	Considered, and will not be done. The screen is balanced the way it is. No harm is done or confusion added if user does

	Suggestion	Response
		click on Main Menu help window. Controls only work within the current window. That is true throughout the system.
DeGennaro:	I would suggest changing the menu names to make it easier to remember where each of the choices are located. For example: Menu_A Analysis Menu_B Modeling Menu_C Planning Menu_D Charting	This has been attempted. It was not possible to come up with an arrange-ment of modules and appropriate menu names that would work. The suggested examples do not work. Analysis and Modeling are part of every module. All modules are used as part of planning. Change will not be made.
	Facilities Layout. Allow the user to type characters in lower case and then convert it to upper case.	This is an enhancement to the system. Later. After the Ph.D.

Suggestion	Response
Zap all files is too dangerous. User should be warned of impending danger. [ENTER] key should be least destructive mode. Maybe a "Are you sure?"	System changed to require three overt actions to Zap all files. Enter key is no longer default.
Would be nice if mouse click would exit user from graph display (in addition to [ENTER] key).	Can't be done with present technology. Mouse is off during graphics display.
Time Series. Might be nice if Analyze screen remembered last entry.	This is an enhancement to system. Later. After the Ph.D.
Would like opportunity to overwrite filename.	Can not overwrite actual filename. Technology does not permit changing the name of a file that is open.

	Suggestion	Response
	Facilities Layout. "Novice User Warning" When user (i.e., me) entering the numbers in flow matrix, I would type a number then press [ENTER]. I continued doing this and found that all my numbers were off because I typed 100 and pressed [ENTER].	This is an enhancement to the system. Later. After the Ph.D.
Roseberry:	Page 2, paragraph 2: Beginning with "But, Dos 6.x will tell..." This sentence is confusing to read unless you have had extensive use with computers.	Sentence cleaned up.

Suggestion	Response
When moving the calendar around, after it is moved the highlight on the current day doesn't appear again. It does after an option (i.e., Last Month) is selected.	This may be very difficult and time consuming to fix and may not be fixable at all. It is not a significant part of Decision Support System. Later. After the Ph.D.
When using the calculator is it possible to let it have exponents? Most of the time you won't have numbers that big but who knows. The screen on the calculator changes to *s after # entered exceeds the screen size. This could result in a frustration problem.	The calculator was designed so that it would completely display the largest number that would need to be generated while using this DSS system. There is not a need for exponential numbers. There should be no frustration problem.

Suggestion	Response
Need to explain there are subsequent pages in the chapter that look exactly like or similar to the ones you (the user) has just printed out. Throughout the book.	This quickly becomes self-evident. A well-written text uses as few words as possible. Change will not be made.
Page 23, paragraph 2: "this time its..." The its in this sentence should have an apostrophe.	"its" has been changed to "it's."
Screen for Regression Analysis isn't big enough. If enter in a group of #s example taken from text from our operations mgt class pg 53. When you select X1, X2, X3 or etc. Then if you make an error the error message can't	Program modified so that error messages can be seen.

Suggestion	Response
be totally seen on the screen. The bottom is cut off.	
Page 66: "minus sigh" should be "minus sign."	Now reads "minus sign."
Chapter 10: In the Example, when you press analyze and it says the Total Cost for the Feasible EOQ is _____. Is there a way to change the # to include a $ sign and commas separating thousands of dollars?	The system should be able to handle any currency. It can, as long as there is no dollar sign. The display is a character string. It would take a signifi- amount of code to parse the character string and add commas. Change is not necessary. Change not made.
Chapter 11: Same as above.	Same as above.

Suggestion	Response
Chapter 13: When you display graphs is there any way to put on the screen/printer which graph it is displaying at the top of the page. Also is there any way you could label the graph (i.e., X and Y axis) tell what the increments represent. This is for the X-bar and R, NP, P, and C charts.	If any more text is added to the graph, it will become too "busy." There are already five lines in the top title. If you look at the graph, it is self-evident which graph it is. There are no X and Y axes on Statistical Process Control charts. The horizontal increments are samples and that is uniform across all control charts. The vertical increments are dependent on the individual problem.
Chapter 14: Same as above.	Same as above.

PAGE 3, COMMENTS:

Banasik: This is fun to use.

DeGennaro: Also, as I already mentioned, I feel that this is an excellent tool and could be marketed outside of the educational environment. The software was easy to use and easy to understand. The printouts were clear and concise. I especially liked the graphing capability. I have always felt one good graph said more than a page full of numbers. Great job!!!

Roseberry: Overall this is an excellent program. I enjoyed working with it and seeing quick results to the problem. This was more enjoyable also, because it eliminated the lag and frustration of the other programs we used in class. This program definitely is a step up from all others of this caliber, with the windows appearance screen and the options it has there will be a lot of people changing to this program.

It is obvious that the evaluators spent a considerable amount of time digging in, understanding, and evaluating the Decision Support System. Their suggestions for improvement and comments reflect deep and serious thinking. Those suggestions that needed immediate incorporation into the system have been incorporated.

IMPLICATIONS FOR FUTURE RESEARCH AND STUDY:

A future Operations Management Decision Support System could be developed and tested which might include additional modules such as:

- Queuing Theory
- Monte Carlo Simulation
- Line of Balance Method
- Shift Scheduling
- Learning Curve Planning
- Break-Even Analysis
- Mark or Buy Decisions
- Group Replacement Analysis
- Standby Equipment Analysis

Research could be conducted on developing a literature module that would allow for the entry, change, and deletion of research references, citations, etc. These entries could then be printed out under a variety of different formats.

Research and development could be undertaken to incorporate additional new technologies such as multimedia. Perhaps a video disk could present a case and an integrated decision support system could assist in case analysis and decision making.

This Operations Management Decision Support System could be tested in various teaching and learning modalities such as undergraduate, graduate, case method, and independent study.

Decision Support Systems and Graphical User Interfaces go together well. Similar systems could be developed for disciplines other than operations management. One good candidate would be the field of marketing.

REFERENCES

Attaran, Mohsen. Operations Management Information Systems. New York: John Wiley & Sons, Inc., 1992.

Awad, Elias M. Management Information Systems: Concepts, Structure, and Applications. Menlo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1988.

Borenstein, Nathaniel S. "Colleges Need to Fix the Bugs in Computer-Science Courses." The Chronicle of Higher Education, July 15, 1992: B3.

Brookfield, Stephen D. Understanding and Facilitating Adult Learning. San Francisco: Jossey-Bass Publishers, 1986.

Chang, Yih-Long and Robert S. Sullivan. Quantitative Systems for Business Plus. version 2.0. Englewood Cliffs, NJ: Prentice Hall, 1991.

Davis, Gordon B. and Margrethe H. Olson. Management Information Systems: Conceptual Foundations, Structure, and Development. 2nd ed. New York: McGraw-Hill Book Company, 1985.

Dickson, Gary W., James A. Senn and Norman L. Chervany. "Research in Information Systems: The Minnesota Experiments." Management Science, May, 1977: 913-923.

Dreyfuss, Joel. "Inside." PC Magazine, October 13, 1992: 4.

Fandt, Patricia M. Management Skills: Practice and Experience. Minneapolis/St. Paul: West Publishing Company, 1994.

Finch, Byron J. and Richard L. Luebbe. Spreadsheet Applications for Production and Operations Management: An End User Approach to Problem Solving, Modeling, and What-If Analysis. Homewood, IL: Irwin, 1990.

Gupta, Sushil K., Stelios H. Znakis, and Tomislav Mandakovic. POMS: Production and Operations Management Software. Boston, MA: Allyn and Bacon, 1988.

Hawkins, John L. and Dian Schafferhauser, "Experts Speak on Object-Oriented Development!" Data Based Advisor, April, 1992: 82.

Heckel, Paul. The Elements of Friendly Software Design. New York: Warner Books, 1984.

International Data Corporation, "Object Technology: A Key Software Technology for the '90s." Computerworld, May 11, 1992.

Ives, Blake. "Graphical User Interfaces for Business Information Systems."MIS Quarterly/ Special Issue, 1982: 15-42.

Jenkins, A. Milton. "Prototyping: A Methodology for the Design and Development of Applications Systems." Spectrum 2, April, 1985.

Johnson, G. Vaughn. Information Systems: A Strategic Approach. Omaha, Nebraska: Mountain Top Publishing, 1990.

Keen, Peter G. W. and M. S. Morton Decision Support Systems: An Organizational Perspective. Reading, MA: Addison - Wesley, 1978.

Kemper, Robert E. and Joseph Yehudai. Experiencing Operations Management: A Walk-through. Boston: PWS-Kent Publishing Company, 1991.

Knowles, Malcolm. The Adult Learner: A Neglected Species. 4th ed. Houston: Gulf Publishing Company, 1990.

Knowles, Malcolm S. and Associates. Andragogy in Action. San Francisco: Jossey-Bass Publishers, 1984.

Kozar, Kenneth A. Humanized Information Systems Analysis and Design: People Building Systems for People. New York: McGraw-Hill Book Company, 1989.

Laudon, Kenneth C. and Jane P. Laudon. Management Information Systems, Organization and Technology. 3rd ed. New York: Macmillan Publishing Company, 1994.

Lotfi, Vahid and C. Carl Pegels. Decision Support Systems for Production and Operations Management. 2nd ed. Homewood, IL: Irwin, 1991.

Margolis, Fredric H. "Teaching Technical Skills in a National Accounting Firm." Andragogy in Action. Ed. Malcolm S. Knowles. San Francisco: Jossey-Bass Publishers, 1984. 45-54.

Mohamed, Nassar H. "A Knowledge-Based Decision Support System for Managerial Problem Recognition." Diss. Texas Tech University, 1985.

Morton, Michael S. Scott. Management Decision Systems: Computer-Based Support for Decision Making. Boston: Division of Research, Harvard University, 1971.

O'Brien, James A. Information Systems in Business Management: with Software and BASIC Tutorials. 5th ed. Homewood, Illinois: Irwin, 1988.

Parker, Charles S. Management Information Systems: Strategy and Action. New York: McGraw-Hill Publishing Company, 1989.

Render, Barry, Ralph M. Stair, Jr., Mohsen Attaran, and William Foeller. Microcomputer Software for Management Science and Operations Management. Needham Heights, MA: Allyn and Bacon, 1989.

Rubinstein, Richard and Harry M. Hersh. The Human Factor: Designing Computer Systems for People. U.S.A.: Digital Press, 1984.

Samson, Danny. Managerial Decision Analysis. Homewood, IL: Irwin, 1988.

Serpen, Ahmet E. "A Proposed Methodology for the Design of Decision Support Systems in Operations Management." Diss. Brunel University, Great Britain, 1987.

Schultheis, Robert and Mary Sumner. Management Information Systems: The Manager's View. 2nd ed. Homewood, IL: Irwin, 1992.

Sheriff, Paul. Seminar Leader, Compass for Clipper, Master Seminar Series, Terrace Hilton Hotel, Cincinnati, Ohio, March 4, 1992.

Shore, Barry. Introduction to Computer Information Systems. New York: Holt, Rinehart, and Winston, Inc., 1988.

Simon, Herbert A. The New Science of Management Decision. New York: Harper & Brothers Publishers, 1960.

Simon, Herbert A. the new science of management decision. Revised edition. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.

Sprague, Ralph H., Jr. "A Framework for the Development of Decision Support Systems." Decision Support Systems: Putting Theory into Practice. Ed. Ralph H. Sprague, Jr. and Hugh J. Watson. Englewood Cliffs, New Jersey: Prentice-Hall, 1986. 7-32.

Sprague, Ralph H., Jr., and Eric D. Carlson, Building Effective Decision Support Systems. Englewood Cliffs, NJ: Prentice-Hall, 1982.

Treu, Siegfried. "A Framework of Characteristics Applicable to Graphical User-Computer Interaction" in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

Turban, Efraim. Decision support and Expert Systems: Managerial Perspectives. New York: Macmillan Publishing Company, 1988.

BIBLIOGRAPHY

Ackoff, Russell L. "Management Misinformation Systems." Management Science, December, 1967: B147-B156.

Adam, Everette E., Jr. and Ronald J. Ebert. Production and Operations Managment: Concepts, Models, and behavior. 5th ed. Englewood Cliffs, NJ: Prentice Hall, 1992.

Ahituv, Niv and Seev Neumann. Principles of Information Systems for Management. 3rd ed. Dubuque, IA: Wm. C. Brown Publishers, 1990.

Allen, Brandt R. and Andrew C. Boynton. "Information Architecture: In Search of Efficient Flexibility." MIS Quarterly, December, 1991: 435-445.

Anderson, David R., Dennis J. Sweeney, and Thomas A. Williams. Statistics for Business and Economics. 2nd ed. St. Paul: West Publishing Company, 1984.

Attaran, Mohsen. Operations Management Information Systems. New York: John Wiley & Sons, Inc., 1992.

Awad, Elias M. Management Information Systems: Concepts, Structure, and Applications. Menlo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1988.

Borenstein, Nathaniel S. "Colleges Need to Fix the Bugs in Computer-Science Courses." The Chronicle of Higher Education, July 15, 1992: B3.

Bower, Joseph L. The Craft of General Management. Boston: Harvard Business School Publications, 1991.

Brookfield, Stephen D. Understanding and Facilitating Adult Learning. San Francisco: Jossey-Bass Publishers, 1986.

Burch, John G. and Gary Grudnitski. Information Systems: Theory and Practice. 5th ed. New York: John Wiley & Sons, 1989.

Burns, James MacGregor. Leadership. New York: Harper & Row, 1978.

Buss, Martin D. J. "Managing International Information Systems." Harvard Business Review, September, 1982: 153-162.

Cash, James I., F. Warren McFarlan, James L. McKenney, and Lynda M. Applegate. Corporate Information Systems Management. 3rd ed. Homewood, IL: Irwin, 1992.

Chang, Yih-Long and Robert S. Sullivan. Quantitative Systems for Business Plus. version 2.0. Englewood Cliffs, NJ: Prentice Hall, 1991.

Chase, Richard B. and Nicholas J. Aquilano. Production & Operations Management: A Life Cycle Approach. 6th ed. Homewood, IL: Irwin, 1992.

Cook, Gary J. "An Empirical Investigation of Information Search Strategies with Implications for Decision Support System Design." Decision Sciences, May/June, 1993: 683-697.

Davis, Gordon B. and Margrethe H. Olson. Management Information Systems: Conceptual Foundations, Structure, and Development. 2nd ed. New York: McGraw-Hill Book Company, 1985.

Dickson, Gary W., James A. Senn and Norman L. Chervany. "Research in Information Systems: The Minnesota Experiments." Management Science, May, 1977: 913-923.

Dilworth, James B. Operations Management: Design, Planning, and Control for Manufacturing and Services. New York: McGraw-Hill, Inc., 1992.

Dreyfuss, Joel. "Inside." PC Magazine, October 13, 1992: 4.

Drucker, Peter F. "The Coming of the New Organization." Harvard Business Review, January-February, 1988: 45-53.

Edwards, Alex and N. A. D. Connell. Expert Systems in Accounting. London: Prentice Hall International (UK) Ltd., 1989.

Evans, James R. Applied Production and Operations Management. 4th ed. Minneapolis/St. Paul: West Publishing Company, 1993.

Fandt, Patricia M. Management Skills: Practice and Experience. Minneapolis/St. Paul: West Publishing Company, 1994.

Finch, Byron J. and Richard L. Luebbe. Spreadsheet Applications for Production and Operations Management: An End User Approach to Problem Solving, Modeling, and What-If Analysis. Homewood, IL: Irwin, 1990.

Forester, Tom. High-Tech Society. Cambridge, Massachusetts: The MIT Press, 1987.

Gaither, Norman. Production and Operations Management: A Problem-Solving and Decision-Making Approach. 3rd ed. Chicago: The Dryden Press, 1987.

Gould, F. J., G. D. Eppen, and C. P. Schmidt. Introductory Management Science. 3rd. ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

Gorry, G. A., and Michael S. Scott Morton. A Framework for Management Information Systems." Sloan Management Review, Fall, 1971: 55-70.

Gremillion, Lee L. and Philip J. Pyburn. Computers and Information Systems in Business: An Introduction. New York: McGraw-Hill Book Company, 1988.

Gupta, Sushil K., Stelios H. Znakis, and Tomislav Mandakovic. POMS: Production and Operations Management Software. Boston, MA: Allyn and Bacon, 1988.

Harrison, E. Frank. The Managerial Decision-Making Process. 3rd ed. Boston: Houghton Mifflin Company, 1987.

Hawkins, John L. and Dian Schafferhauser. "Experts Speak on Object-Oriented Development!" Data Based Advisor, April, 1992: 82.

Heckel, Paul. The Elements of Friendly Software Design. New York: Warner Books, 1984.

Heizer, Jay and Barry Render. Production and Operations Management: Strategies and Tactics. 3rd ed. Boston: Allyn and Bacon, 1993.

Henderson, John C., and David A. Schilling. "Design and Implementation of Decision Support Systems in the Public Sector." MIS Quarterly, June, 1985: 157-169.

Hicks, James O., Jr. Management Information Systems: A User Perspective. 3rd ed. Minneapolis/St. Paul: West Publishing Company, 1993.

Huber, George P. "The Nature and Design of Post-Industrial Organizations." Management Science, August, 1984: 928-951.

Hunsaker, Phillip L. and Curtis W. Cook. Managing Organizational Behavior. Reading, MA: Addison-Wesley Publishing Company, 1986.

Hussain, Donna and K. M. Hussain. Managing Computer Resources. 2nd ed. Homewood, IL: Irwin, 1988.

International Data Corporation. "Object Technology: A Key Software Technology for the '90s." Computerworld, May 11, 1992.

Ives, Blake. "Graphical User Interfaces for Business Information Systems. "MIS Quarterly/ Special Issue, 1982: 15-42.

Jenkins, A. Milton. "Prototyping: A Methodology for the Design and Development of Applications Systems." Spectrum 2, April, 1985.

Johnson, G. Vaughn. Information Systems: A Strategic Approach. Omaha, Nebraska: Mountain Top Publishing, 1990.

Keen, Peter G. W. and M. S. Morton. Decision Support Systems: An Organizational Perspective. Reading, MA: Addison - Wesley, 1978.

Keen, Peter G. S. and Lynda W. Woodman. "What to do with all those micros." Harvard Business Review, September-October, 1984: 142-150.

Kemper, Robert E. and Joseph Yehudai. Experiencing Operations Management: A Walk-through. Boston: PWS-Kent Publishing Company, 1991.

Knowles, Malcolm. The Adult Learner: A Neglected Species. 4th ed. Houston: Gulf Publishing Company, 1990.

Knowles, Malcolm S. and Associates. Andragogy in Action. San Francisco: Jossey-Bass Publishers, 1984.

Koory, Jerry L. and Don B. Medley. Management Information Systems: Planning and Decision Making. Cincinnati: South-Western Publishing Co., 1987.

Kozar, Kenneth A. Humanized Information Systems Analysis and Design: People Building Systems for People. New York: McGraw-Hill Book Company, 1989.

Kroeber, Donald W. and Hugh J. Watson. Computer Based Information Systems: A Management Approach. 2nd ed. New York: Macmillan Publishing Company, 1987.

Kroenke, David M. Management Information Systems. 2nd ed. New York: Mitchell McGraw-Hill, 1992.

Laudon, Kenneth C. and Jane P. Laudon. Management Information Systems, Organization and Technology. 3rd ed. New York: Macmillan Publishing Company, 1994.

Leavitt, Harold J. and Thomas L. Whisler. "Management in the 1980's." Harvard Business Review, November-December, 1958: 41-48.

Leifer, Richard. "Matching Computer-Based Information Systems with Organizational Structures." MIS Quarterly, March, 1988: 63-73.

Liberatore, Matthew J. and Anthony C. Stylianou. "The Development Manager's Advisory System: A Knowledge-Based DSS Tool for Project Assessment." Decision Sciences, September/October, 1993: 953-976.

Licklider, J. C. R. "User-Oriented Interactive Computer Graphics." in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

Liker, Jeffrey K., David B. Roitman, and Ethel Roskies. "Changing Everything All at Once: Work Life and Technological change." Sloan Management Review, Summer, 1987: 29-47.

Long, Larry and Nancy Long. Computers. 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1993.

Lorette, Richard J. and H. Charles Walton. Cases in the Management of Information Systems and Information Technology. Homewood, IL: Irwin, 1990.

Lotfi, Vahid and C. Carl Pegels. Decision Support Systems for Production and Operations Management. 2nd ed. Homewood, IL: Irwin, 1991.

Margolis, Fredric H. "Teaching Technical Skills in a National Accounting Firm." Andragogy in Action. Ed. Malcolm S. Knowles. San Francisco: Jossey-Bass Publishers, 1984. 45-54.

McKenney, James L. and Peter G. W. Keen. "How manager's minds work." Harvard Business Review, May-June, 1974: 79-90.

Miller, Robert B. "The Human Task as Reference for System Interface Design." in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

Mohamed, Nassar H. "A Knowledge-Based Decision Support System for Managerial Problem Recognition." Diss. Texas Tech University, 1985.

Morton, Michael S. Scott. Management Decision Systems: Computer-Based Support for Decision Making. Boston: Division of Research, Harvard University, 1971.

Mouton, Jane Srygley and Robert R. Blake. Synergogy: A New Strategy for Education, Training, and Development. San Francisco: Jossey-Bass Publishers, 1984.

Nahmias, Steven. Production and Operations Analysis. 2nd ed. Homewood, IL: Irwin, 1993.

Negroponte, Nicholas. "An Idiosyncratic Systems Approach to Interactive Graphics." in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

Neiderman, Fred, James C. Brancheau, and James C. Witherbe. "Information Systems Management Issues for the 1990s." MIS Quarterly, December, 1991.

Nickerson, R. S. "On Conversational Interaction with Computers. in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

O'Brien, James A. Information Systems in Business Management: with Software and BASIC Tutorials. 5th ed. Homewood, Illinois: Irwin, 1988.

O'Brien, James A. Management Information Systems: A Managerial End User Perspective. Homewood, IL: Irwin, 1990.

Parker, Charles S. Management Information Systems: Strategy and Action. New York: McGraw-Hill Publishing Company, 1989.

Render, Barry, Ralph M. Stair, Jr., Mohsen Attaran, and William Foeller. Microcomputer Software for Management Science and Operations Management. Needham Heights, MA: Allyn and Bacon, 1989.

Roche, Edward M. Managing Information Technology in Multinational Corporations. New York: Macmillan Publishing Company, 1992.

Rochester, Jack B. Computers: Tools for Knowledge Workers. Homewood, IL: Irwin, 1993.

Rubinstein, Richard and Harry M. Hersh. The Human Factor: Designing Computer Systems for People. U.S.A.: Digital Press, 1984.

Saiedian, Hossein and Scott Ames. "On the design of intuitive user interfaces." IBS Computing Quarterly, Spring 1993: 17-25.

Saldarini, Robert A. Analysis and Design of Business Information Systems. New York: MacMillan Publishing Company, 1989.

Samson, Danny. Managerial Decision Analysis. Homewood, IL: Irwin, 1988.

Schonberger, Richard J. and Edward M. Knod, Jr. Operations Management: Improving Customer Service. Homewood, IL: Irwin, 1991.

Sen, James A. Analysis and Design of Information Systems. 2nd ed. New York: McGraw-Hill Publishing Company, 1989.

Serpen, Ahmet E. "A Proposed Methodology for the Design of Decision Support Systems in Operations Management." Diss. Brunel University, Great Britain, 1987.

Schultheis, Robert and Mary Sumner. Management Information Systems: The Manager's View. 2nd ed. Homewood, IL: Irwin, 1992.

Sethi, Vijay and Shawn Carraher. "Developing Measures for Assessing the Organizational Impact of Information Technology: A Comment on Mahmood and Soon's Paper." Decision Sciences, July/August, 1993: 867-877.

Sheriff, Paul. Seminar Leader, Compass for Clipper, Master Seminar Series, Terrace Hilton Hotel, Cincinnati, Ohio, March 4, 1992.

Shore, Barry. Introduction to Computer Information Systems. New York: Holt, Rinehart, and Winston, Inc., 1988.

Showalter, Michael J. "Integration of P/OM and MIS Research." Decision Line, July, 1992: 16.

Simon, Herbert A. The New Science of Management Decision. New York: Harper & Brothers Publishers, 1960.

Simon, Herbert A. the new science of management decision. Revised edition. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977.

Smith, Allen N. and Donald B. Medley. Information Resource Management. Cincinnati: South-Western Publishing Co., 1987.

Sprague, Ralph H., Jr. "A Framework for the Development of Decision Support Systems." Decision Support Systems: Putting Theory into Practice. Ed. Ralph H. Sprague, Jr. and Hugh J. Watson. Englewood Cliffs, New Jersey: Prentice-Hall, 1986. 7-32.

Sprague, Ralph H., Jr., and Eric D. Carlson, Building Effective Decision Support Systems. Englewood Cliffs, NJ: Prentice-Hall, 1982.

Stair, Ralph M., Jr. Principles of Data Processing: Concepts, Applications, and Cases. Rev. Ed. Homewood, IL: Richard D. Irwin, Inc., 1984.

Stair, Ralph M., Principles of Information Systems: A Managerial Approach. Boston: Boyd & Fraser Publishing Company, 1992.

Stevenson, William J. Production/Operations Management. 4th ed. Homewood, IL: Irwin, 1993.

Summers, Edward Lee. Accounting Information Systems. Boston: Houghton Mifflin Company, 1989.

Thierauf, Robert J. Effective Management Information Systems: Accent on Current Practices. 2nd ed. Columbus, Ohio: Merrill Publishing Company, 1987.

Toffler, Alvin. Future Shock. New York: Random House, 1970.

Treu, Siegfried. "A Framework of Characteristics Applicable to Graphical User-Computer Interaction" in User-Oriented Design of Interactive Graphic Systems. Siegfried Treu ed. New York: Association for Computing Machinery, Inc., 1977.

Turban, Efraim. Decision support and Expert Systems: Managerial Perspectives. New York: Macmillan Publishing Company, 1988.

Watson, Hugh J., Archie B. Carroll, and Robert I. Mann. Information Systems for Management: A Book of Readings. 3rd ed. Plano, Texas: Business Publications, Inc., 1987.

Whitten, Jeffrey L., Lonnie D. Bentley, and Victor M. Barlow. Systems Analysis & Design Methods 2nd. ed. Homewood, IL: Irwin, 1989.

APPENDIX

DECISION SUPPORT SYSTEM for MANAGEMENT

EVALUATION OF SOFTWARE AND TEXT

	POOR 1	2	3	4	5	6	7	8	9	EXCELLE 10
APPROPRIATENESS FOR SUPPLEMENTING AN OPERATIONS MANAGEMENT CLASS	O	O	O	O	O	O	O	O	O	●
EASY TO FOLLOW INSTRUCTIONS	O	O	O	O	O	O	O	O	O	●
COORDINATION OF TEXT WITH SOFTWARE	O	O	O	O	O	O	O	O	O	●
USER-FRIENDLINESS	O	O	O	O	O	O	O	O	●	O
CONSISTENCY FROM MODULE TO MODULE	O	O	O	O	O	O	O	O	●	O
INTERESTING, COLORFUL USER INTERFACE	O	O	O	O	O	O	O	O	●	O
ERROR-FREE Ease of computer utilities	O	O	O	O	O	O	O	●	O	O
COMPARISON TO OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE	O	O	O	O	O	O	O	O	●	O

NAME OF OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE COMPARED TO Todd, & [illegible]

[illegible]
SIGNATURE

1/10/94
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

SUGGESTIONS FOR IMPROVEMENT

- EOQ - need to consider Total Annual Cost at bottom of feasible region.
- Linear Programming - start with a word problem.
- Need a help screen on Access a File.
- When hit SETUP and no file picked, just goes back to previous screen. Need to display a message.
- Consider overlaying entire main menu title bar so people can't attempt to click on the wrong help window.

[illegible]
SIGNATURE

1/10/94
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

COMMENTS

This is fun to use.

[illegible]
SIGNATURE

1/11/94
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

EVALUATION OF SOFTWARE AND TEXT

	POOR									EXCELLE
	1	2	3	4	5	6	7	8	9	10
APPROPRIATENESS FOR SUPPLEMENTING AN OPERATIONS MANAGEMENT CLASS	O	O	O	O	O	O	O	O	●	O
EASY TO FOLLOW INSTRUCTIONS	O	O	O	O	O	O	●	O	O	O
COORDINATION OF TEXT WITH SOFTWARE	O	O	O	O	O	O	O	●	O	O
USER-FRIENDLINESS	O	O	O	O	O	O	O	O	●	O
CONSISTENCY FROM MODULE TO MODULE	O	O	O	O	O	O	O	O	O	●
INTERESTING, COLORFUL USER INTERFACE	O	O	O	O	O	O	O	O	●	O
ERROR-FREE	O	O	O	O	O	O	●	O	O	O
COMPARISON TO OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE	O	O	O	O	O	O	O	O	O	●

NAME OF OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE COMPARED TO LoTFE

Gayle DeGennaro
SIGNATURE

1/14/94
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

SUGGESTIONS FOR IMPROVEMENT

- see attached

Gayle DeGennaro
SIGNATURE

1/14/94
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

COMMENTS

- See attached

Also, as I already mentioned, I feel that this is an excellant tool and could be marketed outside of the educational environment. The software was easy to use and easy to understand. The printouts were clear and concise. I especially liked the graphing capability. I have always felt one good graph said more than a page full of numbers.

Great job!!!

Gayle DeGennaro
SIGNATURE

1/14/94
DATE

Suggestions

- I would suggest changing the menu names to make it easier to remember where each of the choices are located.
  For example:

Menu-A	Analysis
Menu-B	Modeling
Menu-C	Planning
Menu-D	Charting

- pg 49 prob 1 (b)
  Allow user to type characters in lower case and then convert it to upper case

- pg 11
  Zap all files is too dangerous. User should be warned of of impending danger. [ENTER] key should be least distructive mode. Maybe a "Are you sure?"

- Would be nice if mouse click would exit user from graph display (in addition to [ENTER] key.)

Suggestions (cont.) -2-

- pg. 26
  Might be nice if Analyze screen remembered last entry.
- pg. 62
  Would like opportunity to overwrite filename.
- pg 62 "Novice User Warning"
  When user (ie me) was entering the numbers in flow matrix, I would type a number then press [ENTER]. I continued doing this and found that all my numbers were off because I typed 100 and pressed [ENTER].

DECISION SUPPORT SYSTEM for MANAGEMENT

EVALUATION OF SOFTWARE AND TEXT

	POOR									EXCELLE
	1	2	3	4	5	6	7	8	9	10
APPROPRIATENESS FOR SUPPLEMENTING AN OPERATIONS MANAGEMENT CLASS	o	o	o	o	o	o	o	o	●	o
EASY TO FOLLOW INSTRUCTIONS	o	o	o	o	o	o	o	●	o	o
COORDINATION OF TEXT WITH SOFTWARE	o	o	o	o	o	o	o	o	●	o
USER-FRIENDLINESS	o	o	o	o	o	o	o	o	o	●
CONSISTENCY FROM MODULE TO MODULE	o	o	o	o	o	o	o	o	●	o
INTERESTING, COLORFUL USER INTERFACE	o	o	o	o	o	o	o	o	●	o
ERROR-FREE	o	o	o	o	o	o	●	o	o	o
COMPARISON TO OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE	o	o	o	o	o	o	o	o	●	o

NAME OF OTHER OPERATIONS MANAGEMENT DECISION SUPPORT SOFTWARE COMPARED TO

Decision Support System For Production and operations Mgt.
2nd Ed.
by LOTFI and Pegels

Scott Roseberry
SIGNATURE

12-28-93
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

SUGGESTIONS FOR IMPROVEMENT

- Pg 2 ¶2 Beginning with Bat, Dos 6.x will etc.
  This sentense is confusing to read unless you have had extensive use with computers.

- When moving the calender around, after it is moved the highlight on the current day doesn't appear again. It does after an option (ie last month) is selected.

- When using the calculator is it possible to let it have exponents. Most of the Time you won't have #'s that big but who knows. the screen on the calculator changes to #'s after # entered exceeds the screen size. This could result in a frustration problem.

- Need to explain there are subsequent pages in the chapter that look exactly like or similer to the ones you (the user) has just printed out. throughout the book.

- Pg 23 ¶2 this ~~Timeits~~ Time its, The its in this sentence should have an apostrophe.

- Screen for Regression Analysis isn't big enough. If enter in a group of #'s example taken from Text DSS from our operations Mgt class pg 53. When you select X1, X2, X3 or etc. ~~and~~ Then if you make an error the error message can't be totally seen on the screen. The bottom is cut off

Scott Roseberry
SIGNATURE

12-28-93
DATE

DECISION SUPPORT SYSTEM for MANAGEMENT

COMMENTS

- Pg 66 ch. 6. in Brackets the word should be minus sign.

- Ch. 10 In the Example, when you press analyze and it says the Total cost for the Feasible EOQ is ____ (Dollar). Is there a way to change the # to include a $ sign and commas seperating Thousands of Dollars.

- Ch. 11 Same as above.

- ch 13 when you display graphs is there any way to put on the screen/printer which graph it is displaying At the Top of the page. Also is there anyway you could Label the graph ie x and y axis and tell what the increments represent. this is for XBar and R, NP, P, and c Charts.

- Ch 14 Same as above

* Overall this is an excellent program. I enjoyed working with it and seeing quick results to the problem. This was more enjoyable also, because it eliminated the lag and frustration of the other programs we used in class. This program definitely is a step up from all others of this caliber, with the windows appearence screen and the options it has there will be a lot of people changing to this program.

Scott Roseberry
SIGNATURE

12-28-93
DATE