SOME DESIGN CONSIDERATIONS
FOR COMMERCIAL EXPLOITATION OF
(INTERACTIVE) MICROCOMPUTER BASED
DECISION SUPPORT SYSTEMS

by
Scott Bleackley
B.Sc., Simon Fraser University, 1979

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the Department
of
Computing Science

© Scott Bleackley 1984
Simon Fraser University
December 1984

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without permission of the author.
APPROVAL

Name: Scott Bleackley

Degree: Master of Science

Title of Thesis: Some Design Considerations for Commercial Exploitation of (Interactive) Microcomputer based Decision Support Systems

Examining Committee:

____________________________________________________________________
Senior Supervisor

____________________________________________________________________

____________________________________________________________________

____________________________________________________________________

External Examiner

Date Approved: 17 December 1984
I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay


Author: [Signature]

[Name]

Dec. 13, 1984

(date)
Abstract

This thesis explores some design considerations for commercial exploitation of Decision Support systems for powerful, inexpensive microcomputers. We examine software packages which contain elements of Decision Support but are not sold as Decision Support software. From these packages we learn that the decision maker must have enough control to change the system so that the system reflects his or her world, rather than being fixed to the designer's concept of the decision maker's world.

We will look at the source of the knowledge in the Decision Support system. Public knowledge about the context and the source of the data allows decision makers to judge the credibility of the information used within a Decision Support system. We discuss the problems of turning private knowledge about the decision making domain into public knowledge and embedding this knowledge into a Decision Support system.

We also examine artificial intelligence programming in order to gain some understanding of existing techniques for dealing with uncertainty. We explore the possibility of using some of these techniques to deal with the inherent uncertainty of decision making.

During the design of a Decision Support system the product's market must be carefully considered. Otherwise, even a well designed and implemented system will not be commercially viable. We look at several factors which affect the market for Decision Support systems including the rapidly increasing power of microcomputers and the effect of existing software on buyers' expectations.

Taking all of the factors into account, we expect the trend in the future to be toward more powerful and intelligent Decision Support tools on microcomputers.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Trademarks</td>
<td>vii</td>
</tr>
</tbody>
</table>

Chapter 1: Introduction ........................................................................ 1
   Systems Review ............................................................................ 10
   Market Considerations .................................................................. 24

Chapter 2: Human Computer Interaction .............................................. 30

Chapter 3: Important Aspects of Unconventional Decision Support Systems ........................................................................................................ 45

Chapter 4: Design as a Process of Problem Solving ............................... 56

Chapter 5: Dealing with Uncertainty ..................................................... 73

Chapter 6: Knowledge in a Decision Support system: views and extendability .............................................................................. 80

Chapter 7: Communication and Interaction ........................................... 96

Chapter 8: Conclusions ....................................................................... 108

References ....................................................................................... 120
List of Tables

Table 1: Characteristics of some traditional and non-traditional Decision Support systems .................................................. 22

Table 2: Important aspects of Unconventional Decision Support systems ........................................................................ 46

Table 3: Techniques for Dealing with Uncertainty ..................................... 78

Table 4: Available Knowledge Development Tools .................................. 92
List of Figures

Figure 1: Traditional decision support concepts .............................................. 11
Figure 2: Programming by Rehearsal application: Clocks ............................. 37
Figure 3: Macintosh sets new standard for interaction: MacPaint .................. 43
Figure 4: Simplistic Decision Making ............................................................. 57
Figure 5: Linear Process of Design ................................................................. 61
Figure 6: Circular Process of Design ............................................................... 62
Figure 7: Feedback Process of Design ............................................................ 63
Figure 8: Branching Process of Design .......................................................... 63
Figure 9: Natural Process of Design ............................................................... 64
Figure 10: Project Chart .................................................................................. 68
Figure 11: Resource Chart ............................................................................. 69
Figure 12: Planning: Possible approaches / Project Plan ................................. 73
Figure 13: Existing low cost decision support: Odesta Helix ....................... 84
Figure 14: Major Components of a Contemporary Knowledge system ........ 91
Figure 15: Graphics for knowledge capture .................................................... 94
Figure 16: What aspects of Decision Support are Important to Whom .......... 109
Figure 17: Three Views of Decision Support .................................................. 110
List of Trademarks

Apple, Lisa, MacPaint & MacProject are trademarks of Apple Computer Inc.
Macintosh is a trademark licensed to Apple Computer, Inc.
VisiCalc and VisiOn are the trademarks of VisiCorp.
Thinktank is the trademark of Living Videotext, Inc.
Mindsight and Planning Laboratory are trademarks of Execucom System Corporation.
Conquer, Manager's Baseball and Money Trader are trademarks of Sydney Development Corporation.
Context MBA is the trademark of Context Management Systems.
Chapter One

Introduction

Decision Support is a term originally used within the business community to refer to any computer function that could provide some useful information to the decision maker. Generally, these functions are represented by: database query analysis for retrieving appropriate data, spreadsheet operations for manipulating the data, and charting for presenting the data in a meaningful form. In addition, project management systems like "Projacs" (IBM) and domain specific expert systems like "Explorer" (Lehnert 1982) are also being used to support decision making.

Decision Support is a complex, real-world problem that currently has only fragmented and not well understood solutions. As an area of study, Decision Support is another way of viewing problems that have been studied in artificial intelligence and cognitive science for many years.

The basic orientation of Decision Support is to provide an environment and tools that assist an individual's methods of making decisions in an ongoing fashion. Note that while expert systems may be used as a component of Decision Support, the basic orientation of the two disciplines are different. Expert systems attempt to empower the computer with enough knowledge to make decisions while Decision Support systems attempt to empower the individual with enough knowledge to make decisions.

Decision Support is much more than aiding a decision-maker through a series of isolated decisions. It encompasses the problem of integrating constantly changing
information into the decision making environment. The key element to understanding Decision Support is that decisions are interdependent, and often result in compromise. The decision maker is always attempting to balance the positive and negative results of a series of decisions while working towards some larger objective. It is useful to think of decision making as a process of search and refinement rather than think of decision making as a moment of sudden commitment. Focusing on the process that the decision maker goes through enables us to identify areas of decision making in which a computer may prove both useful and useable.

Since people are the decision makers, when we study Decision Support it is primarily the people and their methods that we study. Given that the goal of Decision Support is to use computers to assist in the process of decision making, our study takes into account several aspects of computer methodology including human-machine interaction, and knowledge based systems development. In addition to the study of human decision making and computer techniques for support, we will examine the design of Decision Support systems as they relate to creating a commercially viable software product.

This study is an informal review of Decision Support systems and a discussion of artificial intelligence techniques that may be applicable to the design of commercially viable Decision Support systems.

Systems Review follows the introduction in Chapter One. The topics reviewed include: current mainframe Decision Support systems, specialized expert systems, emerging directions, the impact of microcomputers on traditional Decision Support systems and recent personal Decision Support software.
Most traditional Decision Support systems are implemented on mainframe computers. We will review several of the most successful implementations to see what is viewed as achievable and saleable in this market sector. Specialized expert systems are beginning to make inroads into the traditional Decision Support market. We will examine these systems to see how they differ from more conventional approaches to Decision Support. There are several new systems under development by software and hardware manufacturers. They can provide us with some idea of the directions that are being considered for the evolution of traditional Decision Support systems. Microcomputer-based systems are rapidly becoming more accessible and powerful. These developments have encouraged software manufacturers to shift much of their Decision Support focus to these machines. The trend towards using microcomputers has been fueled by the introduction of VisiCalc, conversions of mainframe Decision Support systems, and a large number of follow-on software products. Finally, in chapter one, we will examine some of the new products on the market which are being sold as Decision Support systems. These products mark a turning point in the development of Decision Support software towards personal Decision Support systems.

Marketing Decision Support software and how the market forces shape the products is discussed in the Market Considerations section of Chapter One. Competition is the single driving force pushing Decision Support. In spite of considerable individual decision making skill, it remains that information must be gathered, sorted and related. As marketing pressures build, the information content of products and services can be seen to increase. This does not mean that more analysis will result in better information and better decisions. The
competitive edge will be provided by: organizing knowledge, having access to other people's knowledge, and having a method of recording and updating the decision maker's knowledge. This edge reflects the skill of the decision maker, not the speed of the software.

Chapter Two is an exploration of human-computer interaction. Communication between the software designer and the software user and how the technology that is used affects the end product are considered. In the last few years it has become increasingly clear that one of the primary prerequisites for a personal Decision Support system is clear and efficient communication. Too often software designers take the attitude that the system user has nothing to do with them. This approach to design allows the designer to remove responsibility from himself, and hide behind the software application. This results in impersonal software applications that communicate little. Rather than thinking of an individual interacting with his computer, it is instructive to think of this individual interacting with another person, the author of the software being used. This computer-mediated communication between people over time and geography is increasingly responsible for the success or failure of a Decision Support software product. We will begin by examining several projects done at Xerox's Palo Alto Research Centre (Xerox PARC). This work and the people who did it created a momentum which is only now beginning to appear in the commercial market. New machines and software have embodied many of the principles pioneered at Xerox PARC and developed by game manufacturers. Video games have captured the imagination and money of thousands of people over the last few years. We will investigate how successful game designers view communication with their
players. As human/computer communications improve, the problems of using knowledge become more prominent. The technology used to build knowledge-based systems can now be applied to further develop computer-mediated communication.

In Chapter Three unconventional Decision Support systems are introduced. We discuss what can be learned from them to improve traditional Decision Support systems. Decision Support systems are now developing from many different directions. Some are non-traditional when compared to the Decision Support market. This change in the larger Decision Support market is stimulated by the growing access to personal computers.

The view of Decision Support is limited when examined from the perspective of any of the contributing disciplines. One way of cutting across preconceived views is to examine software which is not billed as Decision Support but has some recognizable characteristics that do provide Decision Support to an individual. We will examine several software products for the purpose of isolating processes which are important to the people making decisions. These software products include a game based on international currency trading, and a course for teaching the Logo programming language. The processes which emerge include the differences between using algorithmic knowledge and heuristic knowledge, and the differences between learning from an example and having support in a real world situation, and finally, the problems of relating all the dimensions of Decision Support in a single system. While a variety of software products may succeed in providing Decision Support within the limited scope of traditional Decision Support, there remains the problems of portraying, conveying and capturing knowledge and moving it from one type of Decision Support mechanism to another.
This is comparable to the state that VisiCalc was in before the addition of VisiTrend and VisiPlot. However, the difference is of a larger scale with Decision Support.

We examine design as a process of problem-solving in Chapter Four. Decision Support software is a collection of several different and specialized problem solving mechanisms. The most difficult problem in many of these systems is taking into account the constantly changing information and goals. Design is a process of gradual definition, a converging of the problem and the solution at a point where all practical considerations are satisfied. The decision making process for a non-trivial decision is no different. We will examine some design strategies so that we have a basic map to which we can refer as we review the following two examples. First, Buggy, a project of Brown at PARC, provides an example of how to get inside of the decision process using a limited remediation approach to decision making. Second, we will examine a specialized type of Decision Support system, a project planning system, to see how the decision making process is formalized, where the system breaks down and how design as an approach can be applied to this type of Decision Support.

Uncertainty in the decision-making process is the topic of Chapter Five. Design as an approach to decision making opens up the complex problem of dealing with uncertainty. It is important to note that all decision making is dealing with uncertainty to reach a useful solution. Traditionally, decision makers have attempted to remove uncertainty right from the beginning by fixing the problem definition within a narrow context. This method has been useful for Decision Support software manufacturers since the complexity of tracking a problem is
reduced to a computable level. Allow the starting point to move and you have all kinds of problems. Managers of software development teams have learned this to their dismay as projects overrun their cost and time projections. The reason for planning problems is often an inflexible system of decision making combined with inflexible implementation techniques. Both planning and implementation may be very difficult to backtrack and adjust resulting in unexpected cost overruns.

We will examine why Artificial Intelligence (AI) development methods are appropriate to decision making and investigate how AI development methods could be applied to enhance the capabilities of Decision Support software. We will find that these methods can be applied to the function of Decision Support software and that they also illuminate the problem of gathering knowledge from individuals.

In Chapter Six we concentrate on knowledge. To create software which is both simple enough to build and sufficiently robust to be useful, we must organize the way we think about knowledge.

In this chapter we differentiate passthrough and process knowledge as well as public and private knowledge. We look at the importance of being able to extend the knowledge originally loaded into the system. This lead to a discussion of the various artificial intelligence techniques that can be used for ongoing knowledge capture. We examine how passthrough and process knowledge are made available in Decision Support systems. For the purposes of this discussion a computer can contain both passthrough knowledge and process knowledge. Passthrough knowledge is information that primarily conveys information to another person. This knowledge is not used by the computer because the knowledge only becomes useful in the context of sufficient world knowledge. Process knowledge is the
knowledge that is used by the computer for the purposes of computation or control.

We can also view knowledge from the point of view of the people who have it and use it. Public knowledge can be thought of as knowledge which is readily available and accepted. Public knowledge is often published and is usually substantial enough to point at in the form of a well-digested or abstracted definition or algorithm. Personal knowledge is the stuff of which expert systems are made; personal knowledge is used in day-to-day activity. Much of this knowledge has not been codified by an individual but may be much deeper and richer than any elucidated technique or reasoning. Personal knowledge need not be rigorous in its logic or even complete; partial knowledge abounds. Our personal knowledge is the basis of our decision making processes and our ability to communicate.

Any Decision Support system which will be useful on an ongoing basis must encompass expert knowledge as well as process knowledge embodied as models or other types of tools. It must also have a diary function which collects knowledge in terms of the preceding two components. A useful Decision Support system must allow ongoing addition and change. How knowledge enters the system, knowledge capture, becomes an important consideration. Initially, the knowledge, processes and presentation which are to be marketed as the Decision Support system must be assembled. We will examine some techniques used by commercial expert system builders to see how the knowledge engineering is done now.

Once a Decision Support system is in the hands of the decision maker we have the second and more difficult problem of ongoing knowledge capture in the field. Ongoing knowledge capture is more difficult than the initial knowledge
During the initial construction, we find an emphasis on the skill of the knowledge engineer. During the ongoing use of the system, there is no easy computerized method for knowledge capture. We will briefly examine some of the problems of integrating knowledge and consider a graphical method of representing relationships which may prove useful for both gathering and integrating information.

In Chapter Seven we discuss the role of communication in effective Decision Support. Decisions are not made in isolation. In addition to the original knowledge installed in a system and knowledge added to a system by its owner there is a requirement to share information and gather information from other people and other computer systems. We will examine current external information gathering requirements, review a discussion on the dynamics of information sharing within a Decision Support system, and comment on the possibilities and realities of information credibility and exchange.

In the conclusions we will look at the market for Decision Support software - who makes decisions that would benefit from the use of such a Decision Support system. In addition, we will examine characteristics of the market place with respect to available computers and distribution channels. Finally, we will briefly review what we have covered in this thesis with respect to market considerations. With an eye to the future, we will project directions for further consideration and possible outcomes for the near future for knowledge-based Decision Support systems.

By the end of this study we will have reviewed and established the importance of high-level human-computer interaction, identified some critical aspects in the
process of decision making, and proposed some methods for developing both useful and marketable Decision Support systems.

**Systems Review**

Traditional Decision Support systems such as Ramis II (Mathematica 1983), Focus (Information Builders 1983), Mantis (Cincom Systems 1982), Model 204 (Computer Corporation of America 1981) are often marketed for their Decision Support functions. Most of these fourth generation languages are characterized by a non-procedural form of interaction, an underlying database which is usually represented to the user as relational, the capacity to generate reports and possibly graphic output. The importance of non-procedural languages is described by Tesler as follows.

"...non-procedural, or descriptive languages are becoming increasingly important. A descriptive program states what result is wanted without specifying how to get it. The program sets forth relations rather than the flow of control, and so the programmer is relieved of responsibility for working out the steps of an algorithm and specifying their order." (Tesler 1984)

For the most part these fourth generation systems grew out of Management Information Systems (MIS), beginning as a database or database query system and evolved to meet growing requirements for user-initiated database access and report generation. Some of the common components of a traditional Decision Support system are shown in Figure 1. The primary function of the system is to provide a consistent user interface to database and supporting functions. Information can be extracted in the form of ad-hoc queries, reports and graphs.
Models, usually in the form of spreadsheets, can also be set up and the results passed through the user interface. Data files external to the database can usually be read by the traditional Decision Support systems. This allows information from other databases to be accessed through the same standard user interface.

![Diagram of Traditional Decision Support Concepts](image)

**Figure 1. Traditional Decision Support Concepts**

As computer costs decreased and more untrained people required access to the information contained within the database, these fourth generation languages became more acceptable to the computer centre staff. Generally, these languages use more computer resources than is strictly necessary to get the job done. The advantage to using them, from the manufacturers point of view, is that users with relatively little training can directly access database information and generate reports from this information. The amount of training is little relative to programming a solution. The amount of experience required to use one of these systems effectively is large enough that most decision makers have someone use the system for them. An example of the level of interaction provided by traditional Decision Support systems is shown below. Mantis' query language
enables end-users to request information in a free form language. A query is phrased as follows:

"PRINT CUSTOMER, ORDER AND QUANTITY TIMES COST IF ZIPCODE IS BETWEEN 90000 AND 99999 AND WHEN FINISHED PRINT TOTAL AND AVERAGE OF QUANTITY TIMES COST. “ (Cincom 1983)

The resulting report has headings, detail lines, a total and an average.

```
"12:15 P.M.  OCTOBER 2, 1978  PAGE 1
CUSTOMER       ORDER       QUANTITY TIMES COST
  01403          B4683       1,432.80
  09742          AA586       256.00
  11036          C9362       982.50
  45837          A3982       12,460.00
  94382          T4644       5,643.50

TOTAL            20,774.80
AVERAGE          4,154.96  “ (Cincom 1983)
```

This example shows that it is possible to extract information and have some calculations performed without writing detailed specifications stating how it is to be accomplished.

While these database systems have often evolved to include such functions as natural language interfaces, report generators, graph generators, data communication facilities and spread sheets they are still little more than a loose collection of tools bound only by the ability to access a common data format.

Although their operation is often difficult and expensive, these systems do provide the user with the beginnings of Decision Support because they give individuals
direct access to retrieve, change and manipulate the information that they must depend on during their own decision-making process. From the decision maker's point of view most traditional Decision Support systems are still too difficult to use. In addition, there are often problems with the credibility of the information provided and there are often problems with integrating current information with historical information stored in another format. The current best solution to these problems is to get someone else to get the information for you, and make them responsible for the quality of the information provided. In summary, people still provide the Decision Support function! The current Decision Support systems are not yet reaching their target audience.

As the traditional Decision Support systems described above continue to try to hold their market share by adding functions such as spread sheets and natural language front-ends to their systems, newer systems free from a commitment to thousands of lines of existing code begin to take advantage of the market which has been opened up. Some of the new manufacturers are focusing their efforts on improving the user’s control by adding better natural language front-ends to their systems. In addition to providing better interaction, these systems are combining basic inference capabilities with limited domain knowledge. In spite of the application of artificial intelligence techniques, many of these systems remain limited to keyboard and text screen interaction due to the installed base of terminals and other computer equipment. Some of the these new Decision Support systems have been preloaded with expert knowledge, making the process of their interaction and not the underlying user database the focus of their marketing thrust. For example, Cognitive Systems Inc. provides a number of products, some of which provide natural language interfaces to other systems, most of which are sold on the basis of the expert process that they contain. For example, "Explorer"
contains expert knowledge about oil exploration and mapping. The following is a sample of a request in "Explorer". The user's request is prefixed by asterisk. The request itself is English-like. The Explorer system then makes an assumption about a typing error and requests confirmation from the user. Finally, Explorer reiterates the request, making explicit the variables and values that have been selected.

"EXPLORER Version 02 9/23/81

Ready

*Show me a map of all tight wells drilled before May 1, 1980 but since May 1, 1970 by texaco that show oil deeper than 2000', are themselves deeper than 5000', are now operated by shell, are wildcat wells where the operator reported a drilling problem, and have mechnical log, drill stem tests, and a commercial oil analysis, that were drilled within the area defined by latitude 30 deg 20 min 30 sec to 31:20:30 and 80-81. scale 2000 fest

By FEST, do you mean FEET (Y or N) * y

User requests a POSTED map
latitude 30.34167 -31.34167
longitude 80 - 81
output medium: PAPER
output device: PHOTODOT
filters: DRILLING DEPTH > 5000 FEET
COMPLETION DATE >= 5/1/1970
COMPLETION DATE < 5/1/1980
OIL ANALYSIS AVAILABLE
DRILL STEM TESTS PERFORMED
MECHANICAL LOG FILE WELL
DRILLING PROBLEM
WILDCAT WELL
SHELL CURRENT OPERATOR
WELL SHOW OF OIL > 2000'
Note that these are not sold as expert systems because they appear to the user to be retrieving and manipulating user information stored in a conventional database.

In addition to this artificial intelligence offshoot from the traditional Decision Support systems there are other companies who see that the market for Decision Support systems is growing and they are providing their own alternative systems. Some examples are Conquer (Sydney 1984), Mindsight and Planning Laboratory (Wagner 1982), and Context MBA (Context Management Systems 1984). However, these new entries do not have the established market base nor the advertising power of the larger, established contenders. Many of the new contenders do not yet have their product on the market.

The commonly perceived scope of Decision Support has been limited predominance of traditional support systems in an otherwise empty market. The following extract from "DSS-82 call for papers, shows only a slight extent from the database model with its reference to providing insight into future possibilities.

"Decision Support systems provide decision makers with direct access to state-of-the-art computer based technology, providing them with essential information, data and insight on future possibilities to enable them to make the best possible decisions" (DSS-82 call for papers)

During the DSS-82 conference many presentations extended the scope of Decision Support defined by traditional Decision Support systems. One of the more interesting presentations in this regard was given by the president of Execucom Systems Corporation and discusses their design and use of Mindsight, a Decision Support system relative to Japanese management techniques. The importance of
their approach lies in its emphasis on the creation of common models and computer mediated communication between people. Wagner describes the quality of interaction between people that is facilitated by Mindsight. From his description one sees the excitement and potential of such a product.

"As ideas flow rapidly back and forth, and through models and Mindsight, an amazing process occurs. Not only do the participants find themselves searching their internal understandings and feelings for relevant contributions, but they also communicate these matters with an unprecedented depth and breadth. Numerous assumptions that would otherwise be overlooked are quickly expressed, tried out via a model, challenged and accepted or rejected. Enormous amounts of creativity are released. Gradually the individual minds present seem to meld into a collective mind containing an image made up of jointly held assumptions. The result is rarely unanimous agreement but is almost always a deeply satisfying consensus. Some issues naturally have to be resolved by the senior executive. But this process is enhanced by the explicit form in which alternatives are usually expressed. And having had a full opportunity to express themselves, the participants usually feel a strong commitment of the outcome." (Wagner 1982)

It should be noted that although usually no explicit reference is made to computer hardware, most papers make assumptions that are most appropriate for a mainframe time sharing environment.

Microcomputer systems have not played a large role in the market for Decision Support products until recently. The initial memory size, processing speed, mass storage, and communications limitations of microcomputers are no longer a major problem. Powerful microcomputers are now a cost effective solution for personal business computing. With an increasing share of the computing market now going to microcomputers, the traditional mainframe Decision Support manufacturers are being forced to migrate their mainframe systems to microcomputers, notably the IBM PC. Most of the microcomputer implementations of mainframe systems come with the capability to communicate with a host mainframe system. The addition of spreadsheets to the traditional Decision Support systems came as a direct
result of the pressure exerted on the mainframe market by the introduction of Visicalc in the microcomputer market. Visicalc was the first spreadsheet to be mass marketed for the microcomputer. Kay provides a succinct definition of a generic spreadsheet:

"A spreadsheet is an aggregate of concurrently active objects, usually organized into a rectangular array of cells similar to the paper spreadsheet used by an accountant. Each cell has a 'value rule' specifying how its value is to be determined. Every time a value is changed anywhere in the spreadsheet, all values dependent on it are recomputed instantly and the new values are displayed. A spreadsheet is a simulated pocket universe that continuously maintains its fabric; it is a kit for a surprising range of applications. Here the user illusion is simple, direct and powerful." (Kay 1984)

Following the introduction of Visicalc, spreadsheets of all sorts became the mainstay software of the microcomputer market. These spreadsheets provided the illusive what-if modelling capacity that the traditional Decision Support systems had not managed to provide. In addition to being easy to use, spreadsheets on a microcomputer provided an individual, for perhaps the first time, with complete control over both their computing resource and their information.

The drawback with a spreadsheet on a microcomputer was that you might run out of space to do everything that you wanted while this is not as much of a problem in a mainframe environment. In addition, and more importantly, on a microcomputer you had no database, report generator, word processor, graph utility, or communications facility that would take information from or send information to your spreadsheet program.

The obvious lack of this type of integration in a new, very lucrative, Decision Support market, established since the introduction of Visicalc, attracted manufacturers of integrated microcomputer software. One of the manufacturers who have successfully taken advantage of this market is Context Management
"Today's decision support systems employ many technologies to improve the elements of the decision analysis process; however, they focus only on the individual elements of the entire process. The hapless user must painstakingly integrate the results (drawings, charts, explanations and calculations), manually. Leonardo da Vinci knew no such constraints when he used pen and parchment to design new machines. He simply sketched a diagram on his worksheet, wrote some notes about the design, and worked out his calculations beside his drawing. This convenient juxtaposition of information was an important part of his creative process, a part denied today's managers by their systems. . . ." (Hoxie 1982)

We should note that none of the traditional Decision Support systems create an integrated environment, instead they use a non-procedural language as a means of interaction. The integration between components is only that they can read a common data format.

Context MBA and other more developed spreadsheet products, like the Multi-series and Lotus 1-2-3 continued the trend initiated in the market by Visicalc toward creating an interactive environment in place of a language. These products were all successful to some degree in achieving integration between the components contained within the product but, like many of the traditional Decision Support systems lacked the ability to be extended or the ability to communicate with other packages. Some manufacturers have added external communication functions to their products but in doing so have often compromised their integrated environment and have returned to procedural command languages.

As with the original Visicalc, the market was more than ready for any type of integration that could be provided to support the existing spreadsheet functions. The success in the marketplace of the first try at integrating Decision Support functions on a microcomputer has set the market for another round of competition. VisiOn, Symphony (by Lotus corp.) and the Apple 32 software provide a new dimension by providing all the preceding functions, and more, in an integrated
environment as the basis for other developers to build on. These packages all represent more of an operating environment than an operating system.

While it is not yet clear what will be marketed within the shell of Vision and Symphony, we can see two other examples built on the Apple Macintosh that extend the paradigm of Decision Support. In the case of Apple, the Apple-32 software is integrated with the design of the hardware and is resident in read only memory within the machine. First, ThinkTank is billed as an idea processor which allows a writer to scaffold his writing so that detailed and organizational work can proceed simultaneously. The concept behind ThinkTank is that a writer may move back and forth between the details of a paragraph and the general outline that he is creating. ThinkTank enables the writer to develop an outline and at any point elaborate on a subsection of the work and then return to the outline level of detail. The advertising describes ThinkTank as follows.

"Inspiration is fleeting, so just let your thoughts flow. The flexible format makes it easy to rearrange them later into headings and a basic outline. ... Use as many headings and as much text as you need to develop the outline fully. ... When you want to scope the Big Picture, a simple command drops out everything but the main headings. Subheadings and detailed text are stored for recall later" (Living Videotext 1984)

Second, Odesta Helix takes the integration of spreadsheet operation, report generation and database access to a new level of integration.

"With Odesta Helix, you can now manage information according to your own requirements, your own style, and your own imagination. Begin using Odesta Helix immediately, without having to set 'field parameters' or learn a programming language. You can type in a free form report or complicated calculations within your own easily created forms. You can even use pictures from MacPaint to illustrate your inventory or teach language skills. Odesta Helix is a new program, not an old program made to work on a new machine. With Odesta Helix, you experience the real power and flexibility of windows, pull-down menus, icons, and mouse interaction because it takes a new approach to information and knowledge.
Odesta Helix is an interactive information management system designed to take special advantage of the Macintosh windows, icons, pull-down menus and mouse. The program makes possible a flexible, 'give and take' interaction with the Macintosh for information and knowledge-based activities such as free-form data base management, text processing, statistical analysis, and a wide range of form-based input, query and reporting" (Odesta 1984)

Odesta Helix provides an astonishing degree of flexibility, mixing forms creation with data entry with database organization and spreadsheet capabilities. Odesta Helix and ThinkTank are both recent examples of the type of integrated software becoming available for personal computers.

Finally, we should note that currently the micros we are talking about typically contain a 32 bit processor, half a megabyte of resident memory, a ten megabyte hard disk drive, both network and mainframe communications, and for mass storage applications a one gigabyte video disk storage unit.

In this section we have reviewed a wide range of Decision Support systems. We began with the traditional Decision Support systems that grew out of a market for management information system. We looked further than the established mainframe Decision Support systems and pointed out other segments of the market place that are AI driven and microcomputer driven. The basic model for Decision Support began with database systems and database query systems. As the market began to diversify, the traditional Decision Support systems struggled to keep up. Traditional Decision Support manufacturers added spreadsheets, charting utilities, and natural language interfaces. In spite of the changes in the Decision Support market, the current model of Decision Support remains primarily a model based on information retrieval. The model includes components to present information in chart form, and to manipulate information in a spreadsheet. In addition, the current model of a Decision Support system includes some level of user interface.
Traditional Decision Support systems, like the management information systems that they grew from, have never reached the audience that they were originally intended for. Traditional Decision Support systems are generally used by support staff, not the decision makers themselves. The market for Decision Support systems is changing. Individuals are beginning to play a larger part in the Decision Support market place. As a result, Decision Support systems not only have to be inexpensive, and easy to use, they must provide more support to the decision making process instead of just improving the process of information retrieval.

Table 1 is a list of all the systems discussed in this section. Several general categories are listed across the top of the chart. Within each category are different characteristics that could be used to describe the Decision Support system. The categories used to classify these systems are: what type of DSS is it, what is the orientation of the Decision Support system, who is the user of the system, how well integrated is the system, what is the initial cost of the system, how well known is the product, and what type of interaction does the user have with the system.
**Table 1.** Characteristics of some traditional and non-traditional Decision Support systems

<table>
<thead>
<tr>
<th>Type DSS</th>
<th>Traditional</th>
<th>Extendable</th>
<th>New</th>
<th>Orientation</th>
<th>Database</th>
<th>AI</th>
<th>Interaction</th>
<th>User</th>
<th>Decision</th>
<th>Maker</th>
<th>Integration</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Initial Cost</th>
<th>Low</th>
<th>Established</th>
<th>High</th>
<th>New</th>
<th>Known</th>
<th>Product</th>
<th>New</th>
<th>Standard</th>
<th>Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ramis</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Focus</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mantis</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>IDMS</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Nomad</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mindsight</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Explorer</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Intellect</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>VisiCalc</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Multi-Series</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Context MBA</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Lotus 1-2-3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Visi-On</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Symphony</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Apple 32</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>ThinkTank</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Odesta Helix</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Money Trader</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Manager’s Baseball</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Logo Course</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

*... average *... better than average

**Note:** Money Trader, Manager’s Baseball and Logo Course are examples of unconventional Decision Support systems that are examined in Chapter three.

**Explanation of Characteristics**

**Type DSS**

- **Traditional** = fourth generation language; mainframe successor to MIS
- **Extendable** = new functional components can be added to the system
- **New** = the concepts and implementation are still new to the market
### Orientation

**Database**
- the core of the system is a database, the system is used primarily to retrieve data.

**AI**
- the system has expert knowledge or uses techniques from AI.

**Interaction**
- the user interacts with the environment, creating it, rather than primarily retrieving data.

### User

**Decision Maker**
- the decision maker is typically the user of the system.

### Integration

**High**
- the various functions of the system work very well together and information can be transferred from one function to another (e.g., spreadsheet data can be graphed without re-keying the data into the graph program).

**Medium**
- only some of the functions can share data or there are restrictions on how the data can be transferred between functions.

**Low**
- the functions are separate; the same database is accessed, but once data is processed by one function it cannot be input to another function.

### Initial Cost

**High**
- $20,000 - $150,000

**Low**
- less than $1,000 commonly less than $500
**Product**

Established = well known; existing for several years (since early-mid 70's); have an established market segment; accepted by data processing departments and their management; used by Fortune 500 companies.

Known = recognized in the consumer market because of large amounts of advertising and wide use in end-user departments.

New = products which are new and breaking into the market.

**Interaction**

Standard = Interaction only through standard keyboard, text screen and command language.

High = Interaction using advanced techniques either through the language interface, physical interaction or both.

**Market Considerations**

There is a large market for products which will assist decision makers. There has been, and will continue to be, an increase in the amount of knowledge required to make better and better decisions. The quality of the decisions must improve so that the decision maker can compete with others who are using the current technology to consider more information more carefully. Building a good knowledge based Decision Support system is difficult. Marketing such a system is even more difficult.

Several years ago, with the advent of an easy-to-use electronic spreadsheet, the consumer market reacted by making the electronic spreadsheet one of the most pervasive of microcomputer programs. The appeal of the spreadsheet is that it provides a flexible modelling environment. This desire for control has driven the development of many products that have added features and functions to the
original spreadsheet concept. Many companies, such as Visicorp and Lotus, which subsequently built more advanced spreadsheet products have been very successful. However, there have been few new products that have gone beyond the original modelling paradigm of the spreadsheet.

The marketplace has been both primed and wearied by the profusion of spreadsheets with databases and databases with spreadsheets. It has been "primed" in that Decision Support is now thought of as the spreadsheet - database combination. The market has been "wearied" in that it is very difficult to introduce a new product to a crowded marketplace. If we focus on the problem of making good decisions, a spreadsheet model is not always the answer. A product which can make headway in such a crowded market place is one that will more directly serve the needs of the decision maker. In a crowded market place people have a lot of preconceived ideas about what a Decision Support system is. Therefore, the way that Decision Support software is introduced to the market can make the difference between selling and failure. By understanding some of the ways that decisions are made we can both create a product which is useful and have a way of differentiating our product from others on the market.

"The power of a true DSS lies in its gestalt nature. A DSS is far more than a collection of technological pieces, and the nature and capabilities of those pieces are not sufficient for success. DSS is a dynamic organism. It evolves through interaction with its users to constantly adapt to their environment. Therefore, the successful DSS is not built on the specifics of the tools or technology. The critical area lies in the builders' and users' understanding of the decision-making environment. The successful DSS is built from a range of capabilities that won't constrain the evolution of that environment." (Chase Econometrics/Interactive Data Corp. 1982)

There are some simple ideas that are often ignored during the design of a commercial decision support system. An understanding of what is possible and a
high awareness of competing products often leads to a product design which attempts to compete by improving the features of a standard product rather than producing the simplest solution that is appropriate to the given task. In this case the emphasis for new Decision Support systems must be placed on ways of approaching problems that require decisions, on how people solve problems. This is different approach from the question of how do add function to a database. A product orientation that considers the decision making environment first will have the best chance of striking the chord of recognition in the consumer. Software that focuses on application rather than function is selling best. This recognition on the part of the consumer is critical if any product is to succeed. The best product will only succeed if its presentation on the market reflects the current awareness of the buyers. This sounds simple, but positioning in the market is possible the most difficult and critial aspect that contributes to the success of of consumer software products. The market population has a rapidly growing awareness of the Decision Support systems they would like to have. Part of this awareness arises from the limitations that are rapidly discovered through the process of using existing Decision Support products. It is this growing consumer awareness that will eventually mean the success or failure of any new Decision Support product. This future product will be knowledge-based to some degree because AI approaches and techniques are becoming practical on the microcomputers which are on the market now. Future products that are under development now by other software manufacturers are using techniques that are derived for AI. These products will be future competition.

The traditional Decision Support system manufacturers will continue to sell mainframe solutions because of their inherent advantages: shared use of an existing facility, control of data and user population, and the ability to handle
large amounts of online data. The major problem that the traditional makers of mainframe Decision Support systems are facing is one of compatibility. They have a great deal of effort and marketing invested in large, difficult-to-change systems which are already in place. Most of the changes that are being made are layered on top of these existing systems at a cost to the end-user of machine cycles and memory usage. However, the performance to the user often remains the same because of the increased speed of upgraded computers.

Marketing begins with the design of the product. The design in this case begins with an understanding of: how people make decisions in chapters 4 and 7, what type of products are on the market in the systems review section of this chapter, how these products affect the decision making process throughout the thesis, and approaches are available from other domains in chapters 2 through 6. Because any new software is likely to be based on new hardware, different software technologies and different views and approaches to problems it will be difficult for traditional Decision Support manufacturers to make the jump in software technology and change their approach to Decision Support in order to match some of the emerging Decision Support products. New products will provide varying degrees of Decision Support in the expert system development market and consumer Decision Support market places. However, the traditional makers of mainframe Decision Support systems, like the manufacturers of mainframe computers, are well entrenched within the fortune 1000 companies where most Decision Support systems are now sold. In order for other companies to get the market access and capital necessary to create a new generation of knowledge-based Decision Support systems, they must take one of three approaches. Either they must enter the same market place with a different enough appearance so they do not appear to be competition, as the expert systems are now
doing. Or, they must start as small microcomputer based systems aimed initially at a much broader consumer market base. A possible third option is to sell the knowledge-based Decision Support system through one of the traditional Decision Support system vendors. We should note that several of the traditional vendors has developed microcomputer software which allows access to and provides some of the function of the mainframe system. This final solution has the advantages of a large existing client base. It has the disadvantages of not being compatible with the old system and of needing a different approach to using and selling the new system. Other disadvantages to using a traditional Decision Support vendor is that the consumers are a different market segment than is optimal for personal decision support systems. There is also the problem of having to work within the traditional vendors existing concepts of Decision Support. Because of the nature of the interaction, most knowledge-based Decision Support systems are likely to run on powerful personal computers. The people who buy and use knowledge-based Decision Support systems will include many people who have never used a traditional Decision Support system, nor a computer for that matter.

We are seeing micro-computers with enough power to do the complex tasks found in expert systems or AI development environments. Many computer manufacturers are building equipment which is designed primarily as single user machines that can communicate on a network. Sun, Apple, IBM, Symbolics, AT&T and Apple, among others, are all producing such machines. The concepts embodied by these machines are new to the large markets in which they will be sold. The power and utility of machines like these have been used in research environments for many years. Software techniques developed in research environments can now be applied to developing products aimed at a sophisticated consumer market. Initially, we will see the most progress towards knowledge-based Decision
Support systems in the consumer market for machines like the Apple Macintosh. This trend is the result of a large installed base of machines and accessible distribution channels. The stated market for Macintosh and the implied market for many other machines is the knowledge worker, the person who makes decisions as a regular part of their day. Most Decision Support products that will be produced will only partially fit our description of a Decision Support system as it is characterized throughout the thesis. This is because the cost of production and marketing makes it necessary for all but the largest companies to begin a product line by defining a narrow focus of application and then elaborate the application as the product becomes successful. The initial focus could highlight a variety of factors presented in this thesis including communication, interaction, knowledge capture, and mechanisms for dealing with uncertainty. Whatever the initial focus all the other aspects should be taken into account when the decision is made. These products will likely be small, relatively inexpensive, easy to use and well integrated with their operation environment so that other products can be used with them. As the market shakes out, and a base of Decision Support components are available, then we will see products which come closer to satisfying the knowledge and Decision Support requirements of an individual decision maker.
Chapter Two
Human-Computer Interaction

For our discussion of human-computer interaction, we will use advanced programming environments as an example domain. We are examining programming environments because they were the first complex environments that have both problems with human-computer interaction, and have implemented some solutions which deal with these problems of interaction. We begin our discussion of human-computer interaction by looking at three related applications that were developed for the specific purpose of creating a personal, integrated, interactive programming environment: the Xerox Star system, SmallTalk-80 and Programming by Rehearsal.

Xerox Star

The Xerox 8010 Star Information System is a powerful single-user computer which was originally developed for research purposes, but is now available for commercial use. The Star consists of a processor, a large display, a keyboard, and a cursor-control device. It is intended for business professionals who handle information. The significance of this machine is that, for the first time, both the hardware and software were designed to create a working environment tailored to the way people work and think.

The Star user interface works because it adheres rigorously to a small set of principles designed to make the system seem friendly by simplifying the human-computer interface. The following main goals were used in designing the
Star user interface:

- familiar user’s conceptual model
- seeing and pointing versus remembering and typing
- what you see is what you get
- universal commands
- consistency
- simplicity
- modeless interaction
- user tailorability 
  
"(Smith 1982)

Since the introduction of the Star in April 1981, many other hardware and software manufacturers have improved their human-computer interaction by adopting one or more of the principles of human-computer interaction proposed by people working at Xerox PARC.

"One of the most troublesome and least understood aspects of interactive systems is the user interface. In the design of user interfaces, we are concerned with several issues: the provision of languages by which users can express their commands to the computer; the design of display representation that show the state of the systems to the user; and the more abstract issues that affect the user’s understanding of the system’s behavior." (Smith 1982)

**SmallTalk-80**

SmallTalk-80 is the result of many years of refinement and development. SmallTalk was developed as a research project at Xerox Palo Alto Research centre by PARC’s Software concepts Group. They placed their attention on developing an environment that would help individuals use a computer to solve easily varied and complex problems. This decade-long exploration has lead to a new generation of computer-based information tools.

Smalltalk is based on the premise that the power of a computer must be accessible and comprehensible to individuals. Dr. Adele Goldberg, manager of the PARC Software Concepts Groups states that:
"over the years we've trained ourselves not to start from performance requirements, but to start from user needs." (Goldberg 1983)

The result of this focus is that Smalltalk is based on a simple and integrated set of design principles that extend from the fundamental structure of the language all the way to the user interface.

Smalltalk is based on a set of classes of objects that communicate and interact by passing messages. An object consists of some private memory and a set of operations. What an object does depends on the kind of component it represents. For example, objects representing data structures store and retrieve information, objects representing positions and areas answer inquiries about their relation to other positions and areas.

"The real world is a world of objects, people and things communicating and interacting in one way or another. Smalltalk also provides an integrated environment; one that shares a fundamental structure throughout the system which allows a user to move effortlessly between different types of tasks. The high degree of integration found in Smalltalk increases productivity, but more importantly, encourages an experimental approach because changes can be made easily." (Goldberg 1983)

Some of the things that make Smalltalk so useful include the object orientation of the language combined with the use of inheritance, a rich and flexible development environment which includes browsers, and a fast high-resolution display system featuring windows, animation and a mouse pointing device. Inheritance enables us to define a new object by inheriting all the characteristics of that object and then just defining the differences. It is like pointing and saying "it's just like that existing object, except for these few differences". Browsers enable us to examine only the level of detail that we require, and to move easily between levels of detail. In the following example "FinancialHistorym" defines a class of objects that inherits the characteristics of the class "Object". In the
example "instance creation", "transaction recording", "inquiries", and "private" are message categories. In addition, all of the bold face type are message patterns. The lines following each message pattern are the statements. The example shows the SmallTalk code required to maintain a financial history. In this case an instance of the class "FinancialHistory", a particular financial history, is the object.

```
"class name     FinancialHistory
  superclass    Object
  instance variable names 
cashOnHand  incomes  expenditures

  class methods

  instance creation

    initialBalance: amount
      "super new setInitialBalance: amount
      new
        "super new setInitialBalance: 0

  instance methods

  Transaction recording

    receive: amount from: source
      incomes at: source
          put: (self totalReceivedFrom: source) + amount.
      cashOnHand <- cashOnHand + amount

    spend: amount for: reason
      expenditures at: reason
          put: (self totalSpentFor: reason) + amount.

  inquiries

    cashOnHand
      "cashOnHand
    totalReceivedFrom: source
```
The remainder of this example defines the protocols for using the SmallTalk code illustrated above. Notice how the message categories, and message patterns correspond to those in the SmallTalk example above. The text to the right is the functional specification of each message pattern.

```
FinancialHistory class protocol

class initialization
   initialBalance: amount
      Begin a financial history with amount as the amount of money on hand.

   new
      Begin a financial history with 0 as the amount of money on hand.

FinancialHistory instance protocol

transaction recording
   receive: amount from: source
      Remember that an amount of money, amount, has been received from source.

   spend: amount for: reason
      Remember that an amount of money, amount, has been spent for reason.

inquiries
   cashOnHand
      Answer the total amount of money currently on hand.

   totalReceivedFrom: source
      Answer the total amount received from source, so far.

   totalSpentFor: reason
      Answer the total amount spent for reason, so far. " (Goldberg 1983)
```
The importance of this example is that it demonstrates how a few very simple and well thought-out design principles can be employed to keep simple problems simple. By building on a set of consistent principles, SmallTalk can be extended to provide interactive graphics and to provide knowledge structures that have the same simplicity and extendability as the underlying language. SmallTalk is important to consider while designing decision support systems because it, more than any other reference, establishes the importance and effectiveness of creating a user interface as an integral part of every function in a system rather than layering the user interface on top of an existing functions. SmallTalk establishes a user interface that permeates the design of an entire system. SmallTalk creates an exciting environment that the user can reach into and change.

**Programming by Rehearsal**

Building on SmallTalk, Gould has created a system for curriculum designers. This system extends the concept of "Object" to an actor metaphor complete with stage, wings, scripts, productions and rehearsals. Gould provides an overview of her system "Programming by Rehearsal" in the following quote.

"'Programming by Rehearsal' is the name given to a new method of programming that allows curriculum designers to implement their own conceptions of interactive, graphical, educational activities, even if they've never programmed before. The process itself relies heavily on interactive graphics and allows designers to react immediately to their emerging product by showing them, at all stages of development, exactly what their potential users will see. The method is evolutionary, with design proceeding in an exploratory manner through a sequence of constructive experiments. The process is quick, easy, and fun to engage in; once they have been conceived, many interactive activities take less than half an hour to construct" Gould (1982)
Each performer has a predefined set of actions which it performs in response to an associated set of commands called cues. When we select a cue, the performer acts immediately, always causing some observable change to the screen. Once a performer is selected, we can discover the inherent capabilities of a performer by just watching what it does.

To start a new production, we can select the "stage" button from the bottom of the screen. This creates a new stage containing the "done" button. Once the stage is in place we can copy performers from "central casting" to the stage. The performers used on the "Clocks" stage are text actors and buttons. Once we have rehearsed our performers with the menus in "Central Casting", we understand what they can do. We want to assemble them into a production. We can specify the code by three methods: Paste (copy an already existing segment), Type or Watch. Watch is the simplest way to write code. All we do is instruct the "codebox" (were the code is stored) to watch while we show it what the desired action should be. The appropriate code is written for us in the code box as we show it what we want to happen by simply rehearsing the performers as before.

Clocks was made by Laura Gould. It leads the user to explore time expression in four different languages. The user can either let the clocks tick, or can set them to be particular times of interest.
A simple example of the user interface available in Programming by Rehearsal follows. In Figure 2 we see a clock application that has been constructed using the Rehearsal environment. In the following example if we want to discover what performers are currently available we can select the "Central Casting" button shown on the right side of Figure 2. This causes a menu of troupe names to appear. If a particular troupe member is of interest to us then we can select it from this menu, causing it to appear and display the individual performer that it Programming by Rehearsal is implemented in the Smalltalk-80 programming environment and runs on a Xerox 1100 series computer. It both provides an example of the sort of application that is written in Smalltalk and describes, in a more concrete form, some of the methods of interaction involved in creating a highly usable interactive environment. In the following quote Gould highlights
two of the key principles embodied by Programming by Rehearsal. These principles are to make things real or concrete for the user, and to allow the user to determine the course of events by their own actions.

"An important aspect of the Rehearsal World is that everything is made visible; only things that can be seen can be manipulated. Thus rather than thinking abstractly, as is necessary in most programming environments, a designer is always thinking concretely, selecting a particular performer, then a particular cue, then observing the cue's instant effect. ... The designer in the Rehearsal World has the task of constructing a production to be used for the edification of others, and thus assumes the roles of both director and playwright. The script, however, is not as fixed as is in a theatrical production, which is played to a more or less passive audience. Here, the production is different each time it is performed, since the course of events is determined in part by the actions of its user." (Gould 1982)

In the process of creating Programming by Rehearsal, Gould has established some important points of reference. By using a well known analogy, that of the stage, she quickly establishes the relative relationships between each of the other objects in the system. The requirement to convey these relationships clearly to the user is the important aspect to the design of Decision Support systems. In addition, Gould has succeeded in creating a system that involves the user in a process of exploration, creation and ultimately communication with other people. Some of the detailed interactions in the Programming by Rehearsal environment demonstrate the care that has been taken to create an environment which works well with people. For example, it is possible to say "watch me", and then perform some sequence of actions; the program captures what the user/developer has done and adds it to the environment that is evolving.

**Control**

Two things are noteable about "watch me". First, "watch me" establishes the
computer system as partner in communication because the system can now appear to listen and modify its behavior as a result of the information it has gathered. Although primitive, the concept of the computer paying attention in a general way is a powerful one, and it causes the user to focus on the task at hand. Few computer languages or environments are designed with the capacity to both produce actions from code and code from actions. The second notable thing about “watch me” or perhaps more properly the entire Programming by Rehearsal environment is that the boundary between the user and the developer is fuzzy. As the user becomes familiar with an application, they are also becoming familiar with the concepts of Programming by Rehearsal. With time, they can explore and extend the application since the “wings” and “props” are just as accessible to the user as they were to the developer. Ultimately, the interested user becomes a developer and the cycle of communication continues. This fuzziness between user and developer is important for Decision Support because people are different, and no matter how good an application may be, there will be parts of it that could always be adjusted or extended to satisfy an individual’s requirement. We should note that this aspect of accessibility to a system is central to the design of Smalltalk. It is implemented through a process of browsing which allows the user/designer to view only as much detail as is necessary to make the change or addition they require.

“Browsing is based on the psychological principle that recognition memory is different from recall memory - there is an added halo effect to help you out” (Kay, 1984).

We can see some of this potential for extension within the traditional Decision Support systems to the extent that the user can define the names of their fields within a database, and the colors and format of graphics and reports. These changes in traditional Decision Support systems are limited to a choice of
predefined combinations while Smalltalk and Programming by Rehearsal provide a much broader range of interaction primarily because they are both established on, and extended from a simple and consistent conceptual base.

Video games are particularly interesting in terms of Decision Support systems design because their popularity illustrates a tradeoff between detail and control.

"...You see a nice progression from live theater, which has an enormous amount of detail, to movies, which have less detail but are more accessible, to television, which has even less detail but you can turn it on and off in your own home, to video games, with even cruder detail but allowing you to control what is going on to some extent. I think control tends to dominate detail." (Kay, 1983)

Kay provided the original impetus behind Smalltalk. His comments reinforce the direction Smalltalk and Programming by Rehearsal have taken toward giving the user a high degree of control. Kay goes on to suggest that a good interface should be adaptable by the user.

"During the 10 years I was at Xerox, I realized that I really didn't want programming languages anymore because they were not at the right level. Even the strongest programming language I could conceive of is not at the right level of discourse for users ... I think kits are a much better way of doing things ... What you would like to have is a kit with which to build kits. ... In Smalltalk, the basic act is saying, 'This new thing I want is just like this old thing except ....' That is a very kit-like idea. You are starting off with something that already works and modify it creatively. It gives you new things where you only specify the differences. You can get much further this way because you don't have to do the thinking; somebody else has already done the thinking." (Kay, 1983)

**Available Technology**

New computers and software that embody many of the principles pioneered at PARC are becoming increasingly affordable to large numbers of people. Along with the use of high resolution graphics and mouse-type pointing devices, the amount of information that can be conveyed from computer to user and user to computer has
increased significantly. A teletype is a peephole compared to the window that has been opened by the consistent application of small set of design principles that have been tested in the Xerox Star, SmallTalk, and Programming by Rehearsal for human-computer interaction. The information carrying capacity is called bandwidth. With the bandwidth limitations now beginning to lift, there is no longer a physical limitation to what can be done with interaction. The additional machine resources bring the human problems of communication and using knowledge to the forefront. Given the resources to provide rich interaction and control, how do we take advantage of these capabilities? Certainly the lessons we can learn from the Star, Smalltalk and applications like Programming by Rehearsal will give us a starting point and some indication as to which direction to move in next.

With computers like the Apple Macintosh and Lisa, Apollo, Sun workstations, and Symbolics lisp machines we are already seeing just how closely these important aspects of interaction and control are being tied to the machine. It is only recently that personal computers have had enough computing capacity to support both a consistent user interface and operate the programs that users require. Consistency is one of the design principles established for the design of the Star interface. In order to maintain a consistent interface across many manufacturer's software, the computer manufacturers have created operating environments for their machines which bridge the gap between different software packages. Apple has gone the furthest by including much of the code that facilitates the Macintosh user interface in read only memory. By retaining the software basis of the interface in read only memory, the user interface becomes a fixed part of the computer. This demonstrates the importance Apple has attached to the integration of a strong user interface with application software.
Apple Macintosh

Some of the design decisions that make Macintosh a benchmark machine, in terms of the Decision Support market, include more than the high performance hardware and Star-like user interface. The importance of Macintosh lies in its positioning in the marketplace and the combination of its power and ease of use. Because of its price and performance, Macintosh sales are targeted at individual knowledge workers, the primary market for Decision Support systems. In addition, because Macintosh's design includes an extensive underlying user interface, Macintosh lends itself well to the complex interaction and integration required to support useful Decision Support systems.

Mitch Kapor, President & CEO of Lotus Development Corporation, a major manufacturer of business support software says that

"Macintosh is much more natural, intuitive and in line with how people think and work... this is going to change the way people think about personal computers. Macintosh sets a whole new standard, and we want to take advantage of this" (Kapor 1984)

The following diagram (figure 3) is a partial view of a Macintosh screen displaying an application call MacPaint. MacPaint illustrates many of the user interface facilities that are made available as an integral part of the Apple Macintosh computer. Included here is a brief explanation of the MacPaint environment.

Across the top of the screen is a list of pull-down menus: File, Edit, Goodies, Font, Fontsize and Style. When any of these menu headings are selected with the mouse pointing device, a set of menu options are displayed directly beneath the menu heading. The mouse can then be used to choose one of the options.
instance, under the menu heading "File", some of the options listed are OPEN, CLOSE, QUIT, SAVE, and PRINT the current file.

![Macintosh interface with menu bar and drawing window](image)

**Figure 3.** Macintosh sets new standard for Interaction: MacPaint Program

Below the menu bar is the standard MacPaint Screen. It consists of a palette of tools: the two columns of shapes and icons on the left side of the screen. The window labeled "Doc one" provides a canvas on which graphics can be created using the tools on the left. In this instance the drawing window contains both picture and text. This sample was chosen to display the level of detail that can be achieved and some of the variety of image and text that is possible.

The built-in Macintosh user interface facilities that MacPaint uses include: menu handling for the pop-down menus; window generation and handling (including sizing, movement and clipping); generation of rectangles, round-rectangle, circles and lines; text editing functions; font generation (including different fonts, sizes and enhancements); memory management for swapping fonts and portions of the
graphics; file handling; mouse, cursor and event handling; fast graphics generation; and high resolution bit-mapped printing.

The power of computer hardware that incorporates sophisticated user interaction provides a good base on which effective Decision Support systems can be built. More importantly, personal computers which provide good user interface facilities enable the designer of a Decision Support system to incorporate good user interaction at a minimal additional development cost.

Since computers such as the Apple Macintosh are now available on the market any decision support system which does not provide a similar ease of use will not sell competitively. The key to all the interaction we have examined in the chapter, including the principles established for the Xerox Star, is that the attention of the designer has shifted from what is possible given limited computer hardware, towards how people do things. If we take the people approach towards designing Decision Support systems we can address at least two market issues for Decision Support. We can create a system which will be used and we can create a system which stands out among traditional Decision Support systems.
Chapter Three
Important aspects of unconventional Decision Support systems

Unconventional Decision Support systems are systems which because of market or perceived use are not normally categorized as Decision Support systems. It is important to consider at least a few of these systems because as the market changes to reflect the growing access to powerful personal computers, both the market and techniques used to build Decision Support systems will also change.

It may be that, in the near future, few applications will be billed as Decision Support because their function will embody knowledge and interactions specific to a particular type of Decision Support or Decision Support within a specific domain. The goal of this chapter is to examine a few applications that are not considered by people who market them to be Decision Support but effectively provide some aspects of Decision Support during the course of their use. A good example of how a system that was not designed to be a Decision Support system impacted the design of Decision Support systems is Visicalc, the first electronic spreadsheet. When Dan Bricklin and Bob Frankston wrote Visicalc, they had no thought about forecasting; they thought that they were writing an accounting system. But Visicalc is used almost totally for forecasting. That is what it is good at. Visicalc is used primarily as a simulation system in which a decision maker can explore the future before his decisions make it a reality.

VisiCalc and other unconventional Decision Support systems illustrate several concepts which are important for Decision Support but are not considered in traditional Decision Support systems. The following list highlights the considerations illustrated by unconventional Decision Support systems.
• Decision Support in a Simulated environment
  (ie. the role of human learning in Decision Support systems),
• Real Time Simulation of a real world event
  (ie. how good does a simulation have to be for what-if questions),
• Tutorial Assistance
  (ie. tells you how what you are doing relates to the world)
• Extendability
  (ie. what contribution can the user make to the system)
• The Role of Knowledge
  (algorithmic and heuristic knowledge within a Decision Support system).

The following applications are not normally categorized as Decision Support systems by their manufacturer. While it is valuable to examine many applications that incorporate Decision Support components, the following three applications have been selected as examples to highlight the aspects of Decision Support noted above.

<table>
<thead>
<tr>
<th></th>
<th>Simulated Environment</th>
<th>Real-Time</th>
<th>Tutorial Assistance</th>
<th>Extendability</th>
<th>Domain Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Money Trader</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Manager's Baseball</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logo Course</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Important Aspects of Unconventional Decision Support Systems
The applications we examine are two knowledge based games, Manager's Baseball and International Finance (marketed as "Money Trader"), and a course of instruction in the Logo programming language. Table 2 provides an overview of how these considerations are represented in the examples.

"Baseball Manager: This product is based on a board game evolved over 14 years by baseball expert Guy Pilette. It lets you manage your team like a major league manager. You pick players on their abilities, not on their statistics, then manage their play, pitch-by-pitch." (Sydney 1984)

"Money Trader: This is a sophisticated game of international finance which will appeal ... to bankers, trust company personnel and a wide range of people interested in international money and commodity markets. ... Money Trader was developed for us by Dr. Maurice Levi, Bank of Montreal Professor of International Finance at the University of British Columbia, with the assistance of Martin Kupferman. Players make major financial decisions based on realistic events flashed on the screen and play against "the gnome of Zurich". (Sydney 1984)

I was involved in the design phase of each of these applications and the considerations that will be discussed are therefore from a design point of view. It is not my intention here, nor is it possible, to provide a complete description of the workings of a graphic and interactive application. We will outline the design considerations in each application to the extent that they apply to an increased understanding of the process of Decision Support. Each application has some characteristics that the others do not have. In addition to illustrating the importance of specific characteristics within each application, we will compare each application in order to highlight the problems of integrating all of the Decision Support characteristics in a single application.

Manager's Baseball

Manager's Baseball grew out of an extremely complex board game. The game
had been refined through extensive play over a period of fourteen years.

The game has two human players. The input these players provide to the game comes in the form of team selection and play strategies before and during the game. Since a ready-made team is available to any player that wants it, it is possible for novice players to begin playing immediately. Gradually, as the player gains experience at the hands of a more knowledgeable opponent, he begins to exercise the more complex options that are at his disposal. There is no tutorial component to this game. Only experience within a life-like simulation provides the understanding of the game that may be necessary at a later date to manage or bet on a team.

The opponent in this game is another human player. Therefore, as long as the simulation is accurate, the strategy and play can become as complex as during the real game.

The baseball players, within the game, are represented by a collection of abilities which are measured by a numerical degree of skill. The team players do not represent any real baseball player but can be configured to reflect any combination of skills.

The play of the game approximates the game of baseball from a manager's point of view. There is little or no real time action; baseball strategy is the active component of this game. In the board game, several dice are used to simulate results relative to the specified play. Subsequent interactions among the set of rules then perform the bulk of the resulting play. It is interesting to note that the player interactions are not based on statistical probabilities, but are the result of heuristic knowledge that has been adjusted over a long period of time to achieve a balanced and realistic game in most situations.

The computer implementation of this game serves the purpose of reducing the
complexity of the play to posting strategy and presenting outcome. The inherent
difficulties and time consumption involved in playing the board game are now
eliminated, allowing situations to be setup and “what if” questions to be played
out. The Manager’s Baseball simulation provides the decision makers with no
explicit instruction and provides no simulated opponent. Manager’s Baseball
provides Decision Support in the form of a learning and forecasting environment
for baseball managers and would-be baseball managers. What they learn while
using Manager’s Baseball may help them make better decisions in a real baseball
competition.

Money Trader

The International Money game also began as a board game and evolved over a
period of some years during the course of play and the writing of a text book on
international finance. The game deals with decisions that involve international
currency transactions. The player begins with a fixed amount of money and has
options of trading for currency and commodities. The play is driven by news of
events happening in the world which may have some impact on the player’s
position. It is up to the player to decide what to buy, sell, or hold in light of the
current world situation. There are some two hundred news events which are
interrelated to produce an impact relative to other events and the current
situation. After each transaction the player may, if he wishes to do so, receive an
explanation of the impact that the news had on the simulated world market, a
postmortem such as you might find in any financial newspaper the day after.

The game as described was a bit dull, something was missing. The information
that came in the initial news story was much better information than anyone
could get because the player got the information before the market had time to
react. In addition, the player had unlimited time to check his position and evaluate the implications of different strategies before he acted; not a very realistic simulation of a highly volatile marketplace. The only people in the world who may have this pre-information are the so called "gnomes of Zurich". We modified the play to include a market-wise opponent, the gnome, who would act some time after the news was released to the player. The action of the gnome represented the market adjusting to the new information and effectively reduced the value of the news to nothing. The addition of this component to the game provided motivation to act quickly, as is necessary in the market place. The introduction of the gnome has another value. To move fast enough to profit, the player must develop an intimate understanding of international financial transactions.

The purpose of this game is to teach the basics of a set of complex relationships that reflect the workings of international finance. To make decisions within this domain it is not enough to just understand the components. Because of the time constraints imposed by other people acting to normalize the market, the player in real life must understand the relationship between the components enough to extend the relationships to allow for new situations.

This game provides Decision Support for international or would-be international money managers by allowing them to gain experience and instruction in a relatively safe situation. This form of Decision Support does not extend to provide Decision Support in a real world environment because the system is not extendable by any user, even for their own purposes. The opponent in the game is the author of the game, and the events are contrived, although they are realistic due to the experience of the author. In the real life situation there is more than one opponent, more subtle variations concerning degree of involvement, and information is generally old by the time you receive it. You will not become a
successful money trader unless you understand the money market relationships well enough to synthesize the information. International Finance provides Decision Support primarily as a learning environment.

**The Logo Course**

The Logo course facilitates learning through a process of decision making within a real but limited environment. The Logo course is different from BaseBall or International Finance in that it deals with a real programming environment, rather than a simulated environment. The purpose of the course is to provide the user with a basic understanding of how they can use the Logo programming language. The course provides hands-on experience within a structured sequence of exercises. Like Programming by Rehearsal and Smalltalk, the use of the target environment is possible because the environment is completely computer-mediated. Using a real environment creates an additional set of problems that were not encountered in the games that used a simulated environment. In a simulated environment situations can be predicted and limited with respect to their outcome. This gives the designer a consistent and predictable situation to work within. When using a live Logo environment there are a number of problems that arise from developing the support functions within the target environment. First, the environment is extendable in a variety of ways. In fact, one of the goals of the Logo course is to convey this feature of the language. Unfortunately, unless the environment is completely simulated to allow monitoring by an external program there is no way of knowing just what the state of the Logo environment is. Even if the course author did know what the environment was like at any given time, the problem of responding is at least as difficult as creating another player for the Baseball game. Response to action
within the Logo environment is made more difficult because the environment is inherently constructive, enabling the student to extend the environment in any number of unforeseen directions.

This leaves the course author with a difficult choice. The author can simulate isolated aspects of Logo which can be monitored and then write a series of exercises within this environment to achieve mastery of a specified task. The other option for the course author is to leave the Logo environment intact, and provide an external course of action that encourages limited exploration while gradually introducing the language through a series of exercises. Neither alternative turned out to be very good. Both alternatives were used. As an introduction to the concepts of turtle graphics and the syntax of Logo, a small simulation was used that allowed single key-stroke interaction to create pictures on the screen. These pictures were saved as procedures and used later in the course. The creation of a viable simulation requires detailed knowledge of the target environment, and an attention to presentation detail. The remainder of the course was produced in the form of a combination of paper instructions and illustrations. The paper component was necessary because if the instructions were within Logo, the Logo environment would have to be a simulation as opposed to the real environment. More importantly, there was no room on the screen or in memory for both the user’s exploration and the instructions. Having an instruction manual required all information to be presented to a hypothetical average student. There would therefore always be either too much detail or not enough detail for most students. This meant that not enough instruction was available for some people while too much was available to others.

The Logo course provided Decision Support in a live environment, although in a contrived situation to people who wanted to learn how to use the Logo computer.
language effectively at a basic level. The course provided no opponent or feedback mechanism except the passive voice of the author and the active feedback from the Logo system in the form of graphics, animation and text. While this course has an instructional component, it does provide the facility for extensive "what if" type questions. Finally, the Logo environment is event driven. The only aspect of a real-time system that is applicable is the user's complete freedom to do what they wanted when it suited them.

Tradeoffs

These three examples illustrate the difficult and often consuming problems encountered when designing commercial software applications. Some of the problems that must be resolved are: where to put all that information, how to motivate someone in a decision making environment, and how to present information to the user in such a way that they can modify it easily. The question is how do we maintain consistency within the application while still reflecting a real environment.

The tradeoffs between a real and simulated environment often come down to the availability of a real environment and the complexity of dealing with unknowns. In order to deal with unknowns we must give the user enough control to adapt to the environment. These are a another set of interaction and knowledge representation problems in themselves. There is also a tradeoff between explicit instruction and experience as teacher. Experience can be far richer than explicit instruction unless the explicit instruction or explanation is generated by the system to suit the current situation. For any real environment this is only being done by specialized knowledge-based systems such as KEE of Intelligenetics corp.

Many applications on the market today can be examined in the light of Decision
Support processes. Most of these applications work because they have done something to reduce the overall complexity of the application to a manageable size. Either an environment can be simulated, or a selectively limited real domain can provide an environment which captures the aspects which are important to the decision-maker. BaseBall Manager eliminated a simulated opponent and made good simplifying assumptions to simulate the game. International Money limited the range of news stories and possible interactions to a manageable number. As a result, it was able to add explanations about what happened, as well as an opponent, since the best possible move was known. The Logo course used a real but small environment and provided structured instruction. It lacked an ability to monitor the environment and respond to the user. The lack of response was two fold. First, the environment would be different if a monitor were inserted. Second, as the rules of Logo provide an extendable environment, the domain would require a very complex facility to monitor and respond to a user's individual creativity.

Both programming environments like Smalltalk and Programming by Rehearsal and simulated environments like Money Trader and Manager's Baseball are complex environments which are characterized by a high degree of interaction and an ongoing process of making decisions. Smalltalk and Programming by Rehearsal both had a limited environment that was real and could be extended. The goal of both these projects, though not stated this way, was to provide a consistent mechanism that could deal with the problems of representing, portraying and collecting information relative to an existing context or model. The designers of Smalltalk and Programming by Rehearsal made these systems work by creating a system that is accessible and extendable by the user from the ground up, something that none of the three examples above can do. The primary reason that all of the above examples cannot deal with the complexity involved in giving the
user an accessible and extensible environment is the lack of a powerful and coherent set of concepts from which to build. The methods of implementation that were applied to these three applications were not designed to easily enable interplay and change between the established relationships within the system.

To provide a more concrete example of the lack of interplay and change between system components we can look at Visicalc and the other first generation spread sheets. They were viable components of a Decision Support system, but they originally had no way to move the results of their simulation to a charting program or an external database. There was no way to move information from one type of Decision Support mechanism to another. To solve the problem, a new generation of spreadsheet, charting and other applications was born that could communicate but were still only extendible in a predefined pattern.

In general, we can see that in order to bridge the gap between simulations and the real world of Decision Support, the user must have enough control to adapt the systems resources to reflect a real world environment which is different from the environment that the author of the system originally perceived. In addition, the systems must have some reasonable way of dealing with, representing and communicating, knowledge about both the simulated and real environment about which people make decisions.
Chapter Four
Design as a process of problem solving

Design like decision making is a process of gradual definition. In both cases we find a converging of the problem and the solution at a point where all practical considerations are satisfied. The problem may be refined and redefined depending on the certainty of the situation. The initial solution may be correct but careful consideration may cause the problem to be rethought.

Decision making has a curious instantaneous guise. Once a decision is made the solution is perceived by others as obvious or obviously wrong. Other people have little awareness of the time and effort that is often involved in the process of making a decision which will solve a problem. The perception that a decision is a simple choice among alternatives contributes to a simplistic model of decision making.

Decision making takes into account future consequences as well as immediate results. The basis for most decisions is a question which must be answered. In order to answer a question it is usually necessary to understand the context in which the question is asked. To provide the best answer it is necessary to understand the implications and side effects that your answer might have.

If we view decision making as it is sometimes practiced in a management hierarchy we can see a simplified model of decision making in action. If each level of activity is the responsibility of a different level of management then by the time the decision is made very little information about the problem is available. If at any level of management the concerns or considerations are different then the omission of information will compounded. Any misinformation
in the process of decision making ultimately results in the decisions being more and more difficult to make. The following diagram illustrates this simple model of decision making.

![Decision Making Diagram](image)

**Figure 4. Simplistic Decision Making**

A question represents one way of characterizing and limiting a problem as a means of moving towards a solution to the problem. A question may appear, on the surface, to be much simpler than "the full" problem because alternate answers may be available or immediately evident. Without the context of the problem, the question leads us to try to answer the question by applying perhaps insufficient or uncertain information.

An example of such a simple question is one that is often raised by friends and associates. "What computer should I buy?" This problem is often simple in the eyes of the person asking the question because they may only want to reaffirm their current assumption. The question becomes more complex as we discuss the
merits of particular computers. The context of the question finally begins to emerge when we find out why that individual wants a computer. In this example we move down in terms of Figure 4, clarifying and checking assumptions as we enlarge the context of the problem. The initial question serves only to guide the direction in which our search for an appropriate decision proceeds.

Correct decisions are only right when viewed within an appropriate limited context. They may be quite wrong outside this limited context. For instance: suppose an officer of a large Canadian company rightly opted for the lowest interest rate on a large loan to be paid off over a period of many years. To achieve the lowest rate the officer arranged the loan in US dollars. In the larger context of international currency speculation this transaction could be considered dangerous, unless the company hedged its bet by contracting to buy sufficient US dollars at the current rate of exchange to cover loan payment in the future. Without a hedge against fluctuations in relative currency values the savings achieved by the initially lower interest rate could easily be lost many times over.

Most decisions must be limited in scope by the nature of our ability to solve problems. As part of the context of the decision we have information and knowledge about how to use the information. However, both the information and our knowledge about using the information are probably partial, uncertain and changing. Incomplete information demands that the decision maker make informed assumptions in order to achieve good decisions. These assumptions are generally based on broad world knowledge and may be drawn from creative analogies with other similar but unlikely situations.

Decision Support systems should enhance the creative application of broad world knowledge by the decision maker. Currently, expert systems or other computer components that could be incorporated in a Decision Support system lack
enough world knowledge and reasoning ability to creatively solve problems.

What computer systems in general, and expert systems in particular, do well is to remember many relationships and perform rapid calculations and comparisons. Decision Support software is a collection of several different and specialized problem-solving mechanisms. The advantage that a decision maker has by using these tools is primarily the ability to handle more complex decisions and do a better job on simple decisions. Complex decisions will likely have much more information involved, and probably more uncertainty than a simple decision.

The most difficult function to implement in Decision Support systems is the ability to track constantly changing information and goals. While the computer can process increased amounts of information, the problem of resolving uncertainty is left to the experience and judgement of the decision maker.

We consider decision making as a process which may begin with several options, a question, or a problem, but continues by exploring the nature of the problem and by gradually generating an appropriate solution.

Viewing decision making as a design process reinstates the element of time, and focuses our attention more closely on the malleability of the problem itself.

"The DESIGN PROCESS is a sequence of events that demands creative behaviour from its participants. The activity of design is to improve existing conditions and to find clear paths out of dilemmas." (Koberg 1981)

In order to talk about any specific examples of Decision Support as a design process, we need to establish an outline which describes a basic process of design. Before we can review processes of design we need to establish some basic components. The following quotation establishes seven basic steps. These same steps can be found in almost any planning methodology. In other methods the names of the steps may be different and there may be additional steps, but a similar functional breakdown will be found.
The logical general sequence of events included in the design process is:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept situation</td>
<td>To find reasons for going on: To state initial intentions; to accept the problem as a challenge; to give up personal autonomy; to allow the problem to become the generator of process.</td>
</tr>
<tr>
<td>analyse</td>
<td>To get the facts and feeling: To get to know about the ins and outs of the problem; to discover what the “world of the problem” contains.</td>
</tr>
<tr>
<td>define</td>
<td>To determine the essential ingredients: To decide what we believe to be the main issues of the problem; to conceptualize and to clarify those goals concerning the problem situation.</td>
</tr>
<tr>
<td>ideate</td>
<td>To generate options for achieving the essential goal(s): To search out all possible ways of realizing the major goals; to search for the means to achieve the determined ends.</td>
</tr>
<tr>
<td>select</td>
<td>To choose from the options: To compare our defined goals with the possible ways of getting there; to determine the best ways to go.</td>
</tr>
<tr>
<td>implement</td>
<td>To take action (or plan to act): To give physical form to our selected “best ways;“ to make real or “realize“ what we intend.</td>
</tr>
<tr>
<td>evaluate</td>
<td>To review and plan again: To determine the effects or ramifications as well as the degree of progress of our design activity; to self-improve.” (Koberg 1981)</td>
</tr>
</tbody>
</table>

It is possible to impose a number of different views to the order of the design elements. Orders such as linear, circular, feedback and branching have been proposed and used successfully. A linear approach is similar to the simple model
illustrated in Figure 4. Problem, Question, Possible answers, Decision, and implement in Figure 4 correspond to Analysis, Define, Ideate, Select and Implement in Figure 5. The simple model represented by Figure 4 does not include the first and last elements found in Figure 5 because the simple model for decision making is generally used in a reactive environment. The steps of Accepting the situation and Evaluating results stem from an awareness of decision making as a process which can be improved with feedback and practice. A linear process of decision making takes for granted that first impressions are always correct and that all applicable factors are taken into account from the beginning. If the decision is simple enough relative to the experience of the decision maker then the linear approach may be appropriate.

![Linear Process Of Design](adapted from Koberg 1981)

The circular approach to the process of decision making is characterized by the possibility of correcting errors once the decision has been implemented. This approach is different from the Linear model because it extends the responsibility of the decision maker to include and correct the consequences of his decision.
Feedback as a process of decision making has the advantage of eliminating many misconceptions without the costs associated with the multiple implementations that would be the result of using a Circular, or Linear process of decision making. The process of Feedback as illustrated in Figure 7 may take more time, and is more complex to administer than the other processes described above but may result in a more considered decision.
The Branching process of decision making can be accomplished in many different ways. Branching provides some of the advantages of the Feedback process in Figure 7 but without the attendant complexity. The Branching process is more easily divided among several decision makers because the communication from one stage to the next is minimized when compared to the Feedback process.
The limiting assumption implicit in the processes illustrated in Figures 5 through 7 is that only one step can be done at a time. The Branching process in Figure 8 illustrates two concurrent pathways but still retains a step-by-step orientation.

Another assumption implicit in all of the above processes is that the illustrated process should guide the process of decision. The other option is to allow the subject of the decision to effect the importance of each step and the requirements for feedback. For example, the certainty of assumptions made at any point in any of the processes may change the requirement for feedback.

For the purposes of decision making the best way to think about decision making is to imagine all of the steps proceeding at the same time, with communication between steps anytime it is required. At any time during the process one step may be proceeding in advance of the others, but no step is
excluded. Each step moves forward as the decision maker with a decision to be made requires. Moving forward towards a final decision requires give and take between the steps. The Natural model illustrated in Figure 9 can emulate any of the previous models but differs primarily because the approach to decision making is problem driven and considers that several steps can be in process concurrently.

Now we are somewhat prepared to examine two examples which will illustrate some of the advantages and disadvantages to using a computer-based decision support system in conjunction with a process of design.

**Buggy**

The first example is illustrated by "Buggy", a program designed by John Brown while at Xerox PARC. Buggy represents an attempt to understand the decision making process of grade two and grade three students as they learned, or mislearned, arithmetic through the practice of arithmetic. Buggy is interesting in the context of problem solving and decision support because Buggy deals with a very limited knowledge base, and examines the basis on which successive decisions are made to solve a problem.

The project was named Buggy because the approach taken towards understanding the decision making process was to identify typical bugs, or misconceptions in the work of the students. These hypothesized bugs were tested by constructing a model of the student's behavior which included the bugs and various methods of performing arithmetic problems. The model reproduced the student's decision making behavior with reasonable accuracy. Currently, the model is used to show teachers, by interacting with the student model, how to identify and correct the bugs and combinations of bugs that an average student might have acquired.
Look at Buggy from the student’s point of view. The student is the decision maker. He has a problem to solve but he does not have enough information or knowledge to use the information, and what he does know is uncertain. Much of the knowledge that the student does have has been gained through experience with similar situations, although some of the knowledge that the student has is known only by rote (it works but he does not know why). The student’s task of design, like any decision making process, is not limited to solving a single problem. Once the first problem has been dealt with there will be more. The design, in the student’s case, is to develop a set of problem solving strategies that will enable the student to achieve his goal of getting an answer (which like the results of our own decisions, it may not be the right answer). For example, in the process of learning to add numbers together a student may have decided that the process of adding two numbers means that you should take the successor to one of the numbers. Using this rule a student might solve a set of addition questions as follows.

\[
\begin{align*}
1 + 1 & = 2 \\
2 + 3 & = 4 \\
1 + 2 & = 3 \\
1 + 4 & = 5
\end{align*}
\]

This decision results in answers which are correct, most of the time, for the examples so far encountered. This is not a serious problem if counter examples force another hypothesis. However, decisions are cumulative, we build our experience from one decision to the next. If the original concept is retained even if modified, it could have serious consequences on the student’s concept of multiplication, since multiplication is usually addressed in terms of addition. Imagine what would happen if a corporate decision maker were to act on similar incorrect decisions about the stock market. The process of making decisions involves learning and relearning. Learning provides the decision maker with
communication between the concurrent steps that we have illustrated earlier.

The advantage to "Buggy" over other possible reports on student performance is that learning is viewed as a process. The elements of the design process are also elements of the student's decision making process. Furthermore, the outcome and process of one decision making process are shown to provide much of the information and knowledge on which subsequent decisions are made.

This combination of decision making, design and learning reinforce the requirement for extendible systems shown earlier. As the user continues to make decisions, their knowledge and world view changes to accommodate the situations they have encountered. Unless a knowledge-based decision support system can grow with the user, it will not be used further because the user has outgrown the system.

Finally, we will examine a specialized type of decision support system, a project planning system, to see how the decision making process is formalized and where the system breaks down.

**Project Planning**

Project planning involves two elements which make it useful in the context of this discussion. First, there are many operational computer systems which are used to manage projects. Second, because the process of planning a project usually involves several people, the process of decision making is visible.

Typically, a project begins with a goal, such as producing a software product, and then proceeds linearly through several levels of planning. Each planning session establishes estimates and relates the tasks to be accomplished available resources and the final goal. At each phase of planning the project planning system, such as Projacs or MacPlan, calculates a close to optimal schedule given
task and resource dependencies. Figure 10 illustrates a section of a much larger project plan. The project chart shows the order in which things must be done. The numbers above and below the activity boxes denote the start date and completion date for each activity. The boxes which are drawn with dark lines represent the critical path. Any delay of the activities on the critical path will delay the whole project. The project plan is similar to the Linear, Feedback and Branching models because the project plan is defined by what is known about the problem before the project begins.

Figure 10. Project Chart

The resource chart in Figure 11 corresponds to the plan in Figure 10. It shows the activities for each person or resource assigned to the project; only the first of several resources are shown. The resource chart and other charts which are available, such as cost charts, are simply other ways of organizing and presenting the information so that different aspects of the information are recognizable by
the decision maker. This type of reformating of information to help the decision maker easily grasp the important aspects is one of the things that will characterize almost any type of new decision support product.

<table>
<thead>
<tr>
<th>8/10/84</th>
<th>9/7/84</th>
<th>10/5/84</th>
<th>11/2/84</th>
<th>11/30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marketing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review Market Areas</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basic Agreement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prelim. Marketing Plan</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Architect</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cash Flow Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Begin Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 11. Resource Chart**

The "Project Chart" and the "Resource Chart" shown above were generated by MacProject and can be updated by directly altering the graphics displayed on the project chart. This ease of change makes MacProject a useful project planning system.

This linear system of planning everything before you start, works fairly well if the people who make the estimates are experienced with the work involved, and if the project is well defined with respect to existing implementation techniques. In addition, since resource and time data can be altered, the planning tool can be used to answer complex "what if type questions" about changes to resources and schedules.
In practice, during the course of a project, the project plan acts primarily as a communications device between members of a project team. The existence of a complete plan allows minor adjustments to be made during the course of the project in order to keep the project on track.

When you face a real-life decision the knowledge you have available is often uncertain. There is an absolute requirement that the plan be changed frequently as the problem and solution converge to a point where all practical considerations are satisfied. Estimating future resource use is not an easy exercise. Generally, there is great resistance to major changes once a plan is in place, even if the plan is not quite right. Therefore, the plan may no longer be reflecting reality and so the decision maker is faced with uncertain information.

Project Planning and Design are not the same thing but it is instructive to examine the problems of attempting to make design decisions using a project planning model, since this is often what happens. This situation arises because a solution is proposed when the problem has not been clearly identified. Additional factors usually include a limited time budget and resource budget, and an implementation concept and style (e.g., new machines, new language, new design philosophy) with which there is little or no experience. The tendency is to begin immediately and finish as soon as possible. What should properly be a process of design resulting in a product, is reduced to a process of implementation.

The differences between the two approaches can be summarized by two key points. Imagine two groups; one that accepts the uncertainty and plans for change and the other which uses a project management methodology and assumes they will do it right the first time. We will call the first group the design group and the other group the project group. The attention of the design group is on the problem to be solved while the attention of the project group is on the product.
The project group must fit changes into the project plan which was defined when they knew least about the product. The design group maintains a flexibility which allows changes to evolve easily. This strategy may result in a product which is different from the one that was originally conceptualized as a solution. They may have even changed their view of the problem. The result of a design process, by an aware group, usually comes closer to meeting the actual requirements of the market than the result of an implementation. An aware group is a critical element because if people do not care about the decisions that they make, it would be better to use a more structured approach that forces at least minimal checks and communication. The Apple Macintosh computer is a good example of the result that can be achieved by remaining flexible throughout a long process of decision making. A rigid system of preplanning would have resulted in the loss of inumerable subtle but important design decisions that were made during the course of the project.

It should be noted that the implementation or project planning process is used as a decision making structure in many organizations. This is primarily because the design alternatives involve many changes to a great deal of information and are difficult to track. Normally, the changes are noted at a high level on a project plan, but are held within the design team during the course of the project. Until recently there was little opportunity to interactively modify and restructure a plan on a minute-by-minute basis. The advantage of interactive planning tools is that they return real change control to the people who need it, the decision makers (ie. the designers). As more computerized decision support becomes available that supports free-form design, including elements such as time and resources, then the structure of decision making will shift towards the design approach which results in individual's growth and learning as well as better decisions.
In summary, this chapter brings out several factors that should be considered important to the design of a new decision support system. The concept of decision making being a cumulative learning process, that involves possibly many concurrent and interrelated tasks. That the problem and the way in which the decision maker decides to view the problem should determine the strategy for problem solving. A problem solving method should not be imposed. Multiple views of the same or extended information help people to conceptualize different elements of the same problem. Finally, the design of a Decision Support system must facilitate the decision making process. Up to now most Decision Support systems have provided information and left the remainder of the decision making process to the user. We can go further than that by providing software applications that facilitate the development of a good decision after the information has been retrieved by addressing, in software, the decision making concepts which have been examined above.

The target market for effective Decision Support systems are the decision makers who is now becoming good at making small decisions because they care about what they are doing. These people will become better decision makers as they gain experience and when they have access to an effective Decision Support system.
Chapter Five
Dealing with Uncertainty

If the art of decision making lies in the ability to remain flexible while at the same time appearing firm in your convictions, then it must be important to examine some ways of accomplishing this.

The strategy typical in a planning situation is to prune the tree of possibilities from the trunk upwards.

![Possible approaches/Project Plan](image)

**Figure 12.** Possible approaches / Project Plan

This strategy results in the elimination of all but a single path of action. From the planning stage on, events are usually made to fit the plan, rather than face the complexity of further decisions. This strategy works well if the task domain is well understood by the planners.

If the decisions are more difficult, this usually reflects more uncertainty in the available information, or more uncertainty in how to deal with the available information, or both.
In this section we will examine two approaches to software implementation. The first approach is the standard planned approach to building structured applications, the second is the more flexible approach typical of artificial intelligence applications. More attention will be drawn to the AI approach as it is more interesting for the purposes of decision support. We will examine the AI approach to programming with an eye to how decisions are made, particularly how flexibility is maintained while progress is made.

If you don’t know where you are going, you will probably start out in the wrong direction. It will therefore be important to be able to change direction quickly. Most complex decisions begin this way. The options are to live and work with the uncertainty and gradually design a solution, or to eliminate uncertainty by making assumptions about what you are doing at the beginning, when you know least about the problem. Assumptions can be thought of as decisions that are made early, then proven wrong at a later time when they are deeply entrenched within a project. The latter approach of pruning the problem domain has been adopted by programmers of commercial software. This approach to implementation is currently used by the developers of contract software. The difference between the commercial programming environment and the contract programming environment is the degree of certainty. The contract programmer begins with a set of program specifications. Their job is one of technical implementation which can be managed with some degree of certainty, if it has been done before. In contrast, the commercial software developer is developing software which must be sold to a largely unknown and changing market. These people have the additional problem of defining what it is that they should build: what will sell in which market, for which machine, when. The effect of making initial planning assumptions for this group can be devastating. Here, we are approaching the level
of complexity involved in many AI projects and many real world decisions. The
commercial software developer faces the design problems associated with a
changing market, technology, and financing as well as the more usual resourcing
and scheduling problems. Together these factors raise the complexity of the
decisions to the point where either a new approach to development and decision
making must be used, or the initial assumptions must be right.

We are looking to the programming technologies that have been developed for
use in Artificial Intelligence because a tremendous amount of effort has been put
into developing strategies that help people deal with the problem of developing a
solution when there is remains a great deal of uncertainty in the specification of a
problem and its method of solution.

Basically, if you do not know, or are uncertain about something, you must find
out. Usually, the only way to find out is to try something and see what works.
Unless you are lucky, you do not usually get the right answer when you try
something. Instead, you get the answer "not quite right". So you must expect to
be in a continual state of change. The focus of any personal decision support
system must be on how to best manage the changing knowledge of the decision
taker.

In terms of programming, this exploratory approach is completely orthogonal to
conventional programming methodology.

"There, one begins by writing down the design, checking it until you're sure you
got it right, and then locking the coding process into that design so you implement
exactly what the design specifies and nothing else. That's absolutely not what we
want here. We want to decide what we want at the end of the implementation
process, not at the beginning when we know almost nothing about the problem.
That only works when you have perfect foresight, which we most certainly don't
have in AI. As a result, most conventional programming methodologies can be ( and
are, in AI practice ) discarded immediately. They don't work." (Sheil 1983)
Some of the ways that AI software development techniques make progress while still providing the necessary flexibility are illustrated by the characteristics of the languages, environments and programming techniques which have evolved for this particular style of programming.

Most AI programming languages have two important features. The first idea often embodied in AI programming languages is that they enable the programmer to defer commitment as long as possible. A decision that has not been cast in code does not have to be recast when it is changed.

Techniques for deferring commitment make experimentation easier by minimizing how much you have to do when you back out and change your mind. The second idea often embodied by AI programming languages is the concept of supporting embedded languages and metalanguages. This is important because if you have an idea about what is important, you build your idea into the language and now you think in terms of your idea. This is important because it enables you to start thinking in your own terms. Using a system built in your own image is enormously seductive, and should be a major consideration in the design of any knowledge based decision support system. (In effect enabling the user to build themselves into the system.,) This concept of building beyond the initial knowledge base to incorporate new knowledge can be considered system maintenance but should be handled as a function of normal, productive interactions.

AI environments use language-specific programming knowledge to provide exactly those control and bookkeeping functions that are the greatest drain on the programmer during rapid systems development.

The complexity of information collection, organization and management can be managed to some degree by a Decision Support system. These support functions
are usually performed for a corporate decision maker by an assistant or staff. In the context of a knowledge-based decision support system it is important to, where possible, support the knowledge management transparently. This enables the decision maker to focus on the components of the decision at hand rather than the supporting tool. There are other programming techniques that have been developed to facilitate AI programming. The goal of these techniques is primarily the management of large complex environments that need frequent, local changes. In AI the environment is a programming environment, but the same techniques hold true for any Decision Making environment. For example rule based systems represents a general technique for structuring large environments or systems so that small pieces can be changed or added to the environment without having to understand the entire environment. The ability of the decision-maker to easily make local changes increases the complexity of the decision making model that the decision-maker can successfully maintain.

These approaches are important to consider when designing a decision support system. We can consider them from two perspectives. First, decision making in any complex environment is similar and may benefit from similar support systems. Second, these technologies could themselves be used to implement more general knowledge-based decision support systems. In both cases AI software technology will not make the uncertainty disappear, but it at least holds out the promise of being able to adapt decisions fast enough so that the uncertainty can be mapped and the range of possibilities can be understood.

"Al is programming under terminal uncertainty; programming under uncertainty by definition. Conventional data processing occupies the other end. Until recently, there has been only one cost-effective application development technology — one that assumes that everything is fixed and decided and certain. As a result, the application spectrum was dichotomized and everything that was considered doable (which did not include AI) was deemed to be completely understood because the
application methodology demanded it. The availability of a cost effective technology for dealing with design uncertainty removes the need for such an artificial barrier and suggests that we should revisit many applications that have had little or no success using conventional development methods. For example, an enormous variety of specialized, professional information support systems now seem to be realizable given AI's software tools to develop them." (Sheil 1983)

AI development techniques provide us with some useful insights into how flexibility can be maintained while progress is being made in a complex decision making environment. By incorporating some of the software technology used in the field of Artificial Intelligence into the development of commercial decision support systems we can achieve two things. First, the tools a designer uses are often reflected in the final product. Second, if the concepts and tools are carried through the design of the decision support system to the decision maker, we will have a product which makes dealing with uncertainty in decision making a productive activity.

Table 3 highlights several characteristics that have been derived from techniques that are used to deal with the problems of uncertainty in Artificial Intelligence programming environments.

Table 3. Techniques for dealing with Uncertainty
predefined problem: The specification is well understood in terms of existing tools and techniques.

changing environment: Changes must be made during the process of development due to new knowledge or findings.

exploratory approach: When problems are not well understood it is worthwhile to try an approach even if it may not work or will require additional work.

defer commitment: Continue development pressing commitments into the future when a decision can not yet be made.

knowledge management: If a system has knowledge about knowledge (meta-knowledge) it can organize knowledge for the user, removing a lot of otherwise necessary distractions.

incremental extension: The system can be extended where and when it becomes necessary a little at a time, rather than through a process of redesign and rewrite.

integrated tools: The knowledge manipulation and other tools can make use of each other providing the facility to construct compound tools.

transparent tools: The tools work smoothly enough that the decision maker can focus on the decision making process instead of the software tools that make the process more effective.
Chapter Six  
Knowledge in a Decision Support System:  
Views and Extendability

Knowledge provides the basis for making decisions. In general we can view knowledge as an aggregation of relationships between things, people, concepts and relationships.

Making a decision is often a process of balancing many interrelated factors to achieve a goal without sacrificing other less tangible concerns. Knowledge, specific to the immediate situation yet broad enough to cover unlikely but possible contingencies, is the heart of decision making. The process of applying this knowledge is called judgement. The intent of Decision Support software is to support the decision maker in the process of making judgements.

In order to create software which is both simple enough to build and sufficiently robust to be useful we must organize the way we think about knowledge. We will first discuss two types of knowledge and how they relate to the author of the Decision Support system, the user of the Decision Support system and the computer. For the purposes of this discussion we can view knowledge that is embodied in a computer as either passthrough-knowledge or process-knowledge. Later in the chapter, we will discuss ongoing knowledge capture in a Decision Support system. We will examine this critical function and look at some of the factors necessary for a successful implementation of processes that capture knowledge.

Passthrough-knowledge refers to information stored within the computer that is used primarily to communicate between two people, it is not used to control the activities of the computer. Passthrough-knowledge is not used by the computer
because passthrough-knowledge only becomes useful in the context of sufficient world knowledge. For example, in the "International Money" game the tutorial component is entirely made up of passthrough-knowledge. The author's intent is to convey an understanding of why, for example, the price of gold rises with destabilizing influences in the world economy. The simulation within "International Money" may parallel the passthrough-knowledge presented here but does not use the knowledge directly. The advantage of passthrough-knowledge within a Decision Support system over traditional book presentation is that the computer enables the knowledge to be keyed to the process at hand. Images from video disk and recording from audio disk that are manipulated by the software application can be considered passthrough-knowledge for the purposes of this discussion, since the application has no direct access to the knowledge presented to the user.

The problem of providing passthrough information is not a simple one. For the system designer the problem is two fold. First, how should the information be organized and presented; will it include graphics, sound, or animations? Second, what will be the content of the information presented? How can I provide the level of detail which is appropriate for each user? Remember, this passthrough information will usually be available in conjunction with a more interactive component of the Decision Support system. The intent is to provide information about real world concepts and how they relate to each other. The interactive tools provided within the Decision Support system mirror the real world concepts and relationships that are presented as passthrough-knowledge to the user. There is an important distinction between what computer users commonly see as help screens and the type of passthrough-knowledge discussed above. "Help screens", provide information about a tool. The passthrough-knowledge that is important to
Decision Support provides knowledge about the world.

*Process-knowledge* is the knowledge that is used by the computer to control its actions. It is commonly embodied within the interactive simulations or other Decision Support tools provided with the system. A good example where we can find two distinct levels of process-knowledge is in the form of a computer spreadsheet. The first level of process-knowledge is represented by the computer implementation of the spreadsheet itself. The knowledge of how rows and columns are located and how formulae are calculated and propagated combine with the calculation power of the computer to achieve an automatic spreadsheet. The second level of process-knowledge for this example is the use of a spreadsheet template. A template is a set of labels and calculated relationships implemented on a computer spreadsheet. Spreadsheet templates usually apply to a specific domain of knowledge like home budgeting, dairy milk production, or stock analysis. The template embodies the author's understanding of the important relationships. These relationships are provided to the user in a functional way. The user is free to simply input his own data to be manipulated by the provided functions, or the user may modify these functions and add his own functions. Often the passthrough-knowledge that explains why things were calculated the way they were, is minimal and in the form of an accompanying manual.

If we borrow some implementation strategies from the world of expert systems, we can imagine that the subject matter expert has available some form of authoring system (perhaps like Programming By Rehearsal) which will handle the representation, manipulation and presentation of both passthrough and process-knowledge. The subject matter expert preloads a knowledge representation and manipulation system with a base of well organized knowledge about a specific subject. The user can then use the same facilities for knowledge
manipulation plus the extended facilities that embody the expert's process-knowledge, to extend the system with his own knowledge and data. The real power of the system comes from the user's ability to extend an already established base of both process and passthrough-knowledge.

Currently there are no consumer products which provide knowledge manipulation facilities in addition to an initially useful knowledge base. The closest we come to that goal now are database systems with integrated spreadsheet facilities. There are commercial expert systems builders such as Teknowledge and IntelliGenetics, who have systems available which will support the continued development of larger knowledge bases involving both passthrough and process-knowledge.

In order to build an extensible knowledge based Decision Support system it is important to make a distinction between the vehicle (knowledge based authoring system) and authors. In general, the vehicle provides a framework and data manipulation facilities, while the author initializes the vehicle with application specific information and models. If a system is extensible, that is the user is given the same manipulation facilities as the original author, then the user can also be thought of as an author. This aspect of user adaptation and extension is critical to the commercial success of a personal Decision Support system.

Currently, the only consumer product which appears to provide a broad enough base to support both the authoring and the knowledge representation aspects of the problem, (at least to a limited extent) is a product called Odesta Helix.

"Set up your own forms (or customize provided forms) and then input information only once. Here is just one example: Enter an order, and generate an invoice, adjust inventory, update customer records, and keep track of receivables. Everything you enter can be easily modified, redefined, and interrelated. There are no barriers placed between you and the way you want to do things. ... Once you have the form put together, choose 'Lock', and the form is ready to use. Of course, if you
want to re-design or edit the form, you can always do so. ... No complicated 'data
definition language' in Odesta Helix. Just open the icon for any piece of
information, and click it into the format you want. You can set up text, numbers,
dates and even flags to appear in different ways in different forms. ... Tiles act
as arithmetic, text, Boolean and date operators, functions and values. ... Visual
building blocks let you set up even the most complicated statement or calculation
and use it whenever you want. You 'flow' the information from one block into
another - just by pointing. ... Odesta Helix automatically creates icons to represent
every piece of your information. Make them appear if you wish, and rearrange them
within or between forms. Define and restrict them if you wish, for data checking
and security, But most of the time, just let them stay hidden 'behind' the form. ...

No special commands to learn in order to search through your 'ocean' of
information. Simply ask a question by pointing to a spot on a form, or by filling in
spaces with your own questions." (Odesta 1984)

The Figure 13 shows a user defined form within Odesta Helix. This example
shows a standard data retrieval function. The power of Odesta Helix lies in its
consistency and variety of function. This example demonstrates one aspect of the
Odesta Helix's function. The query "find all books published before 1940, that are
first editions costing more than $13.00, and that are about World War I and are
illustrated'.

<table>
<thead>
<tr>
<th>LIBRARY MANAGEMENT</th>
<th>REF. #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DATE</td>
</tr>
</tbody>
</table>

| AUTHOR: ____________________________ |
| TITLE: ____________________________ |
| DATE: 1 | < | 1940 | PLACE: ____________________________ |
| PUBLISHER: ________________________ | EDITION: FIRST |
| PHYSICAL DESCRIPTION: ____________________________ |
| CONDITION: ____________________________ |
| PRICE PAID: $ ___________ | RETAIL PRICE: $ | > | 13.00 |
| DATE PURCHASED: ____________________________ | LOCATION: ____________________________ |
| CATEGORY (S): World War I and illustrations | NOTES: ____________________________ |

Figure 13. Existing low cost Decision Support : Odesta Helix

In addition to viewing knowledge within a Decision Support system as having
passthrough and process-knowledge, we can view knowledge from the point of view of the people who have it and use it.

**Personal and Private Knowledge**

"Knowledge in any specialty is usually of two sorts: public and private. Public knowledge includes the published definitions, facts, and theories of which textbooks and references in the domain of study are typically composed. But expertise usually involves more than just this public knowledge. Human experts generally possess private knowledge that has not found its way into the published literature. This private knowledge consists largely of rules of thumb that have come to be called heuristics. Heuristics enable the human expert to make educated guesses when necessary, to recognize promising approaches to problems, and to deal effectively with errorful or incomplete data." (Hayes-Roth 1982)

Our private or personal knowledge is ultimately what our decisions are based on. This personal knowledge may be enriched as a result of communications with other people but ultimately our decisions are our own. People can communicate their knowledge by a variety of methods ranging from personal instruction to published works. An Decision Support system should facilitate rapid diffusion of knowledge and interaction to help refine the knowledge that people put out. Computer conferencing systems demonstrate one component of communication that could be incorporated in a Decision Support system. Decision Support systems can begin to bridge the gap of effort involved in making our personal knowledge public so that other people can explore that knowledge, work with it to achieve goals and extend it with their own knowledge. One of the things that we have not seen in Decision Support is a recognition that more than one person is usually involved in the process of making any decision. As a result we should consider one function of a Decision Support system to be a medium for communication among people who have knowledge to contribute to one another.
"Knowledge is a scarce resource whose refinement and reproduction creates wealth. Traditionally the transmission of knowledge from human expert to trainee has required education and internship years long. Extracting knowledge from humans and putting it in computable forms can greatly reduce the costs of knowledge reproduction and exploitation. At the same time, the process of knowledge refinement can be speeded up by making private knowledge available for public test and evaluation." (Hayes-Roth 1982)

Another aspect that should be considered with respect to the transmission of knowledge from an expert to a trainee is that in any medium including the computer a great deal of knowledge is omitted due to the limitations of the communication medium. This results in loss of knowledge if we wait until the expert is finished expressing the knowledge before the trainee gains access to the work. However, if the trainee or trainees are in a position to question and contribute to the process of expressing knowledge from the beginning to the end of the process then the process of learning and communication reflects the traditional relationship between master and apprentice. Computerized Decision Support systems are in a position to, and should provide facilities that encourage this type of communication as an integral part of the Decision Making process.

**Knowledge Capture**

Computer mediated communication between people can take place using only pass-through knowledge, the computer acts only to transfer knowledge. In a Decision Support system it is important to capture at least some knowledge in computer usable form. If knowledge is accessible to the computer then some of the tasks of organizing, relating, retrieving and using knowledge can be done by the Decision Support system, rather than leaving the tasks to the Decision Maker.

Some processes for creating knowledge that the computer can use are available now. The orientation of products which encompass these processes differs from
the concerns of ongoing communication that we have for Decision Support systems. Most of the companies that are now involved in accumulating personal knowledge into a computer are doing so to create expert systems. Expert systems differ from Decision Support systems in several ways. The expert system is designed to make decisions, not support the process of human decision making. Building an expert system is like publishing a book, all the material goes in and is polished before the book is released for publication. Decision Support systems should focus on the development of the knowledge as part of decision making the resulting knowledge base is then a useful starting point for further development. The process of capturing knowledge is the common ground between expert system development and the use of Decision Support systems.

Teknowledge, one of the few companies focusing exclusively on Knowledge Systems and Services, describes "Knowledge Engineering", the process of building expert systems, as follows:

"Knowledge engineering/nal-ij en-je-‘ni(e)r-ing,n. 1. the engineering discipline whereby knowledge is integrated into computer systems in order to solve complex problems normally requiring a high level of human expertise. 2. The subfield of artificial intelligence concerned with the acquisition, representation, and application of knowledge, including inexact, heuristic and subjective knowledge. 3. The activity of constructing a computer representation of a body of knowledge (knowledge base) and the inference procedures required to interpret that body of knowledge." (Teknowledge 1984)

We will be primarily concerned with the second element of knowledge engineering: the acquisition and application of knowledge, including inexact, heuristic and subjective knowledge. We will pay special attention to the human element of knowledge acquisition, or knowledge capture.

We may be able to make use of expert system technology in two ways. One, to assist the decision maker with the tasks involved in knowledge engineering. Two,
to learn from the ways that knowledge engineering is done now, and from the tools that are used to do it, to incorporate limited knowledge engineering facilities into a knowledge based Decision Support system. The primary goal of providing knowledge engineering within the framework of a Decision Support system is to enable the system to keep up with the growing experience of the decision maker. The key to using knowledge for decision making is to make it explicit, allow both the human and the machine to use and understand the same knowledge format.

Knowledge capture is particularly important for Decision Support systems because it keeps the system up to date with respect to the real world of the decision maker. This correspondence between collected knowledge and the actual environment will determine the degree to which a Decision Support system can be used to assist in the process of making decisions. A Decision Support system will only be used if the knowledge represented within it is up to date. The knowledge within the system will only be up to date if the process of incorporating knowledge is not an extra burden on the decision maker. Therefore, the best solution would be to capture new knowledge during the normal process of Decision Making, and this includes communication with other people.

We are concerned with two elements of knowledge capture for commercial Decision Support systems. The first element is the knowledge that is integrated into the application before it reaches the market place. The second and more difficult element is ongoing knowledge capture in the field.

Knowledge engineering is primarily concerned with the initial component: capturing expert knowledge to sell to an expert system market. By examining current knowledge engineering systems and the ways in which they are used, we can gain an insight in to how knowledge capture can be achieved. These systems are worth examining because they represent the best available experience in
capturing human perceptions of complex decision making environments.

While part of the continuum of knowledge capture, the ongoing problems of end-user knowledge manipulation are more difficult than the initial knowledge work. This is because the initial work often requires the skill of a human knowledge engineer. There is currently no mechanism for knowledge capture that works as well as a human mediated system.

According to Teknowledge, one of the key elements typical of knowledge based systems is explicit representation. The following quote expresses the viewpoint of Teknowledge.

"the explicit representation of knowledge in a knowledge base permits the building of systems that can assimilate and exploit quantities of knowledge orders of magnitude larger than could ever be practically incorporated into standard computer programs. Consequently, problems that require the application of large amounts of knowledge become practical with this technology. In addition, the explicit representation of knowledge facilitates maintenance and debugging of knowledge systems. In contrast to a 'decision tree' approach, where the entire process must be analytically determined in advance and coded into a data structure, a knowledge base normally contains only independent 'packets' of knowledge, each of which can be examined, debugged, updated or altered separately. Furthermore, the actual protocol or reasoning process of the systems is synthesized dynamically using the knowledge base, and so a single change may be reflected throughout the entire system." (Teknowledge 1984)

This separation of knowledge from the mechanics of the Decision Support system allows any part of the knowledge to be modified or extended. Such modification of the knowledge base is essential because the process of accumulating a consistent knowledge base tends to be fragmented and incremental.

Building a knowledge base from scratch requires attention to detail and the structuring skills of a knowledge engineer and the experience of a subject matter expert. Once a Decision Support system has been engineered: the basic
organization of knowledge and interaction, the decision maker must continue the process by adding knowledge to the system and modifying the existing knowledge to keep it consistent with the decision making environment.

In the environment of a knowledge-based Decision Support system the decision maker is both the knowledge engineer and expert. The Decision Support system developer provides the basic knowledge and a framework for change and addition. The decision maker molds the system of knowledge to conform with a changing decision environment. We have to be careful here, knowledge engineering is not a well defined science, and as a result when we talk of a decision maker continuing the process of knowledge engineering it must be in terms of a limited and very well defined knowledge engineering environment.

An important consideration in terms of knowledge capture is how to allow the decision maker to manipulate knowledge without being distracted by the way in which the knowledge must be represented. Knowledge based systems make use of several different techniques for representing knowledge often mixing the forms of representation.

One method of supporting the decision maker is to make the inference engine (the logic that uses the knowledge) itself an expert on its own operations and the representations used to store the knowledge it operates with.

"The inference engine uses metaknowledge (knowledge about the contents of the knowledge base and about reasoning strategies) to choose an appropriate approach to solving the current problem. Such introspective systems promise to be more flexible - and hence more powerful - than their unreflective predecessors. " (Kinnucan 1984)

The Figure 14 illustrates the major components of a contemporary knowledge engineering systems. One difference between this illustration and a Decision Support system is that the knowledge development tools must be made available
to the decision maker as part of the process of Decision Making while most of the current knowledge engineering systems are used to create an nonextendable system.

**Figure 14.** Major components of a contemporary knowledge system adapted from (Hayes-Roth 1984)

**Commercial Knowledge Development Systems**

In a commercial market place it is important to examine the cost of using these technologies today. Some private firms are beginning to offer commercial systems for knowledge engineering. Some of the most popular systems are listed in table 4.

Most of the systems in table 4 are expensive and operate on relatively expensive computers. In addition to the large knowledge engineering systems there are other offerings that will make knowledge engineering more cost
effective. Teknowledge is among the first to announce micro computer versions of its knowledge engineering systems. On June 11, 1984, tekknowledge released two software products for knowledge engineering: "M.1" and "S.1". M.1 is used to design, build and run stand-alone knowledge systems on an IBM Personal Computer. The price is $12,500 and includes a training course. S.1 currently runs on a Symbolics lisp machine and includes a two week training course and costs $50,000. Future versions of S.1 will run on the DEC VAX 11/750 and 11/780 under VMS.

<table>
<thead>
<tr>
<th>Company</th>
<th>Name</th>
<th>Cost</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teknowledge</td>
<td>M.1</td>
<td>$12,500</td>
<td>IBM PC</td>
</tr>
<tr>
<td>Teknowledge</td>
<td>S.1</td>
<td>$50,000</td>
<td>Symbolics &amp; VAX</td>
</tr>
<tr>
<td>Intelligenetics</td>
<td>KEE</td>
<td>$30,000</td>
<td>Symbolics</td>
</tr>
<tr>
<td>Inference Corp.</td>
<td>Art</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software A &amp; E</td>
<td>KES</td>
<td>$16,000</td>
<td>DEC</td>
</tr>
<tr>
<td>Prologica</td>
<td>Prolog</td>
<td>$3,500</td>
<td></td>
</tr>
<tr>
<td>Xerox</td>
<td>Loops</td>
<td>*</td>
<td>Xerox 1100</td>
</tr>
</tbody>
</table>

* available on an unsupported basis for a nominal duplication fee

Table 4. Available Knowledge Development Tools

It is important to note that in spite of the availability of these products, the process of knowledge engineering remains people intensive. This is also reflected in the duration of the training course which comes with the Teknowledge products. These systems represent only a fragmented approach to knowledge capture, however what they do represent is the state of the art techniques of what to do with the knowledge once you have it.

Knowledge engineering techniques and existing systems can be used in several ways to build a new Decision Support system. The systems noted above can be used to assemble and organized knowledge that is to be built into the Decision
Support system. Some of the techniques that are used to manipulate and structure knowledge can be adapted to the development of facilities within the Decision Support system that allow for ongoing knowledge capture.

Other Approaches to Knowledge Capture

It is not enough to examine knowledge-base development system in order to understand how knowledge can be captured and used in a Decision Support system. To get more ideas about how to approach the process of knowledge capture, it is sometimes useful to examine the inverse process. In this case the inverse of Knowledge capture could be considered to be knowledge presentation. The following example highlights how a simple non-computing concept might be applied to create a limited system of knowledge capture. While the following example can be viewed as depicting knowledge, it can also be use to capture knowledge if the decision-maker can easily create and manipulate the graphic representation so that it corresponds to their understanding of the relationships that are portrayed.

Recently at a conference for professional technical writers' there was a paper presented title "The Four Dimensional Page: a Workshop in Visual Creativity" (Horton 1984). This paper discussed the advantages of presenting printed information using the several methods of three dimensional illusion. By adding a third or fourth dimension to a two dimensional presentation system more relationships could be depicted clearly.

The main problem of mechanical knowledge capture is the problem of relating, presenting and manipulating complex relationships between multiple objects. Communication of knowledge between human and computer in ways that are comfortable to the decision maker and possible for the computer is very important
if we are to capture knowledge once the Decision Support product has been sold.

Writers must deal with the problems of communicating complex concepts effectively. Writing is only one example of a profession where we can find existing techniques that can be applied to make knowledge capture easier and more effective. To create knowledge based Decision Support systems, designers must be creative and draw on tested methods of communications to create computer models for knowledge communication.

As an example of applying well known techniques for representing special relations. We take a lead from the writers' conference. We find that by manipulating graphic representations of relationships in three dimensional space, we can achieve a limited but effective system of knowledge capture.

In the example below, the size, labeling, and relative position front to back can be manipulated. This picture represents a fragment of a knowledge capture system designed to portray and capture estimated degree of relationship (defined by overlap), relative ordering front to back and bottom to top, and the relative size of the circle is used to denote both importance, or complexity, and as a positioning cue.

![Figure 15. Graphics for Knowledge Capture](image-url)
Spacial relationships have recently been applied in the form of overlapping windows, on a screen, to deal with activation control and the lack of display area on a screen. The ideas that can be applied need not be new, but they should have some significant advantage over existing techniques.

**Summary**

To conclude, we have seen that knowledge capture is an important concern for both the construction and use of a knowledge based Decision Support system. Existing techniques for building expert systems can be useful for building knowledge based Decision Support systems and may be extended or adapted to provide facilities for ongoing knowledge capture in a Decision Support system.

The conclusions that we can draw from this discussion of knowledge as a base for Decision Support are that it is useful to treat the computer as a medium of communication among people, knowledge should be separated from the authoring framework on which it rests, and that Decision Support systems should be extendable through use as a basic component of their normal operation.

Knowledge capture remains a difficult problem but this does not mean that knowledge based Decision Support is not practical. It does mean that a pragmatic constructive approach to designing such a system must be employed. A Decision Support system has the advantage over an expert system that it has a support function not a decision making function. A Decision Support system has the added advantage that a human can do most of the very difficult (in terms of world knowledge) tasks. The role of a Decision Support system therefore allows solutions to hard problems like knowledge capture, that are not available to systems which are not designed to be evolved by their users.
Chapter Seven

Communication & Interaction

Communication and interaction between people through and around a Decision Support system can have the effect of accumulating, and verifying an accurate model for the decisions at hand.

Communication in this context extends beyond the scope of knowledge capture and information display. By communication we mean people sharing understanding and knowledge with other people. This factor can be a problem when a Decision Support system is used outside of the mainstream of interpersonal interaction. When designing computer software it is easy to think only about the computer, its interface to the user and perhaps how the decision-maker does his job. Rarely does the software designer look beyond the immediate application to access to broader social impact of the system they are developing. In the case of decision support systems people must be involved, convinced, shown. They must be allowed to search, explore, rummage, investigate, and interact with other decision makers. Decisions are not made in isolation.

Decisions are generally made to accommodate change or to bring about change in the real world. A great deal of information is available to us from other people and through a variety of media. Much of this information has been predigested to suit some target audience. As a result, contextual information is often lost. Only the experience of people is available to assemble this fragmented information. Only people can make judgements about its validity and fill in the gaps. The task of creating a meaningful whole from changing fragmented information can be facilitated with the experience of several different people at once.
The database is central to traditional mainframe Decision Support systems. Often the database is extended by allowing access to external databases. While access to the information is important, it is more important that the decision-maker understand the context in which the information is presented. Unfortunately, information stored in a complex database loses much of its context. This problem can be overcome if the individual using the database is completely familiar with the domain of interaction. The lack of context becomes increasingly misleading as the decision-maker enters unfamiliar areas of information.

**Communication Media**

A database has a tendency to be anonymous. Information is fed into the system but it is unknown who did it, when and why. The decision-maker must evaluate if relevant information has been misrepresented or if it has been skewed to the advantage of some special interest group.

The only source of information for any decision-maker is ultimately their own experience, a significant part of which involves interactions with other people. We can gain experience vicariously from the experience of other people. Much of the information that is used for making decisions is of this sort because we can not be everywhere at once. Most people have a high level of skill with regard to vicarious learning; they have used this skill most of their lives. The people the decision-maker learns from may be close by, in the office, or they may be people at a distance who are writing books or articles. People have the most experience dealing directly with other people. As the media becomes less familiar to the decision-maker less experience is available to be applied to interpreting the
credibility of the information that is presented.

The medium is important because in the absence of personal contact, the regulations and reviews placed upon a communicator in each media are different and places different interpretations on the information received.

For example, the Wall Street Journal database implies a degree of credibility corresponding with the reporting quality of the Wall Street Journal. There are hundreds of commercial databases available to today's decision-maker. There remains a dual problem of finding where the information is stored, how to access the information and most importantly, how to access the credibility and completeness of the information provided. For some decision makers, the solution to the complexity and uncertainty inherent in using commercially provided information is to create their own database.

Creating a database involves establishing and maintaining the integrity of the information within the database. In addition, at some point in time the decision-maker will want to manipulate his information and compare it with commercially provided information. Traditional Decision Support systems provide these functions. Once the decision maker's information requirements become large and more complex a traditional data base tends to become impersonal. This is because the volume of data defies attempts to individually characterize each piece of data relative to the existing data except in terms of existing categories. The communication between the information providers and the information users (both people) becomes more limited as the overall mass of the stored information increases. In addition, with a large database the original organization of the information acts to artificially shape relationships in the information that is added. In short, a traditional Decision Support system does not encourage people to interact and shape common knowledge and models because people do not exist,
as contributors, in the system. The opinions and knowledge of individuals is not usually identified. As a result interaction between people about common problems cannot be mediated by a traditional Decision Support system. A traditional Decision Support system is missing a number of elements needed for people to meaningfully share information.

An example where information sharing can take place is when multiple players assume roles within a predefined model. There are many examples of such models used in roles including economic competitions between universities and dog fights between simulated fighter planes. In the case of a financial model each decision-maker defines a set of strategies and assumptions against a market driven model. Assuming that the model is well understood by the players, the actions and assumptions of each decision-maker contribute to the creation of complex interactions within the model.

The advantage of such interactions is that oversights and strategic flaws are noticed and exploited by the other players. This rich and rapid feedback emulates real world interactions because in both cases people are making all the decisions. This "micro world" concept with multiple decision makers points to the power in pooling individual understanding. The pooling of information is not the only important aspect of a "micro world" with multiple decision makers. A "micro world" focuses the attention of every decision-maker onto the same topic. The resulting communication is therefore more effective than a non-computer mediated meeting. The key to using simulated environments effectively is to keep the environment simple and accurate. Allow the decision makers and other contributors to provide the information and interaction which makes such communication vital.

The value of a simulation for decision makers has a lot to do with the amount
and detail of control they have within the model. A model can only approximate a
live situation, but like any artistic rendering, if the characteristics that are used
are chosen well then model can provide access information and relations that
would not otherwise be immediately available. Models can range from simple video
games to sophisticated inflight control systems. For such a flight control system
(for example fighter or shuttle flight surface controls) the data acquisition, user
interaction, and control functions operate in real time. Models can also be
realtime or preplanned. The realtime model has several advantages such as
animation for maintaining the illusion of a real environment. The realtime
environment also plays against the player's ability to think on their feet. A
preplanned environment is sometimes more realistic because it emulates the
process of delegation that often occurs within a company. Both approaches can be
useful in a multiple player situation.

**Truckin': a multiple-user-model**

A simple framework that illustrates a preplanned environment is found in a
discussion of an experimental course conducted in early 1983 at Xerox PARC.
"Loops" is a system for building knowledge-based systems that was developed at
Xerox PARC. A course was conducted in order to find out how novice users reacted
to working with knowledge using Loops. The following example focuses on the
course itself but it is also useful to note the speed with which novice users
adapted to using a knowledge-based construction system. Fifty people took the
course, and during the course they extended and debugged small knowledge
systems in a simulated economics domain called Truckin'.

"... At the end of the course a knowledge competition was run so that the
strategies used in the different systems could be compared. The punchline to this
story is that almost everyone learned enough about Loops to complete a small knowledge system in only three days. ..." (Stefik 1983)

The "Truckin" environment used during this course simulates roadways, fuel stops, suppliers, unions, buyers, etc. In short, this environment includes those things which are critical to the decision making process of an independent trucker. Note that the difference between the "decision-maker" and the "player" is that the player is the small expert system that is built by the decision maker to act independently in the Truckin' environment.

In the Truckin' environment, during a competition the decision makers have an opportunity to test their strategies and expectations against those of other decision makers. Two groups had a private playoff just before the competition.

"... and discovered that when both players were in the same game, the inventory of luxury goods on the game board became exhausted before the end of play. Neither player was able to cope with this situation." (Stefik 1983)

This example shows that the interactions between multiple participants creates the most useful characteristics of the model. The unpredicted and complex interactions between people and their decisions are difficult to simulate but can be found in practice when several people are making decisions in a limited environment. This is real world competition in a limited and familiar environment. Decision Support systems should have some mechanism for creating micro worlds and allowing their use by multiple decision makers on an ongoing basis. If the use of micro worlds is monitored by the Decision Support system then we have a good mechanism for capturing a knowledge base relating to the micro worlds that are used.

In Truckin', the strategies are prepared entirely ahead of time. During the competition there is no interaction between the decision makers and the player
they have created. As a result it is the silliness of the ill-fated move that all the observers, of the competition, appreciate almost immediately. The perception of silliness demonstrates the immediacy with which all the decision makers recognize and acknowledge the situation. Part of the reason for this speed of recognition is that the decision makers recognize themselves relative to a real world context. The recognition of previous unrecognized factors is important for continuing communication among decision makers because it creates a common ground for discussion.

For example, the following episodes highlight the sort of difficulties that are commonly encountered in the Truckin' environment. The following points provide a context to make the examples easier to understand. Alice's Restaurant has significance as a goal for the end of the competition. Players must buy and sell the goods that they carry to make a profit, but the more trips they can make, the more potential profit they can make. In addition to buying and selling goods, the decision makers must consider the day to day problems of operating a truck.

"A player may be racing to Alice's Restaurant. One move before the game ends it is unable to resist a business 'opportunity' and doesn't make it to Alice's.

A player may go to the closest place to sell some goods, even if it happens to be the City Dump which unfortunately pays a 'negative price'.

A player may become focused on a tight producer/consumer loop. Making money faster than any other player on the board. If it is programmed to only buy fuel from stations along its route. But there is no gas station in the tight loop, the team will watch anxiously as the fuel gauge drops lower and lower." (Stefik 1983)

The "Truckin" environment illustrates two things. Techniques currently exist that allow decision makers to communicate constructively, by creating small knowledge based systems. This communication does not imply a watchdog approach
to decision making, instead provides focused medium for exploring the rational
and implications of possible decisions with other people. A preplanned strategy
can bring into sharp relief problems which would be immediately corrected and
perhaps not noticed in a realtime control environment. This sort of ongoing
learning is very important for decisions that are being made in a rapidly changing
environment for which there are no simple answers. The preplanned simulation
has a less sophisticated parallel in real world project planning systems.

Other Multiple-User Models

In traditional Project Management systems, there is an attempt to build a
model which will reflect the world in terms of available resources and progress
against an initial specification. In most project management systems the people
involved in the project report hours against progress toward planned objectives.

The concept of many interacting and interdependent resources within the
project management model is similar to the Truckin' environment. The project
management model lacks the immediate decision making mechanisms which are
represented by small expert systems in Truckin'. The result in the project
management model is that unless all the people involved are online all the time (are
notified whenever someone else updates the plan so that they can respond if
necessary) the resulting simulation is patchy and degrades quickly. This occurs
because changes to a plan, (the decision making model in this case) are dependent
on up to date information. The planning system suffers from a co-ordination
problem.

If each person named in the plan could make changes to the plan to reflect their
situation then the project plan would be current. However, the real situation is that changes and reactions to these changes are constantly lagging and old information is propagated through the project plan causing degradation of the model.

Communications are the key to tracking progress relative to the model. However feedback, and reaction time must be taken into account when applying the model. The project management system works if there are few changes to the plan and if the changes occur within the update cycle of the model. The project management model is not an effective decision support system when there are many changes within the update cycle of the model.

With the uncertainty inherent in making decisions, communication must be effective and timely relative to the change or update cycle of the model. If the decision support system is to be more than a game, the model must reflect real world events in real world time.

From the project management example we can see that it is important in a multiple player (decision maker) environment to maintain continuing interaction and attention, either directly or through a "proxy". The small expert system that acted as the player in the Truckin' environment is an example of a proxy.

In the International Money Game the "gnomes of Zurich" are proxies for the system designer. The gnomes are an opponent to the decision maker using the system. An opponent was introduced to create a game which more closely modelled the realities of money markets. Information effecting money markets is quickly assimilated into currency pricing by other people in the market place. There is little time to think about the pros and cons of buying or selling. The International Money Game is a small example of communication (by proxy) and of interaction in a decision making environment. It is a step towards multiple-user
models exploring real problems. It helps to achieve a consensus among decision-makers about concrete approaches and market strategies.

This shared development of models among several people through a process of contention and interaction is a method of building or extending a knowledge base. This knowledge base changes when necessary to fit the decision-makers' understanding of their problem and its environment. The multiple-user model also has a synergistic effect creating new knowledge as a result of interactions between contributors on the system.

**Building Consensus**

This method of evolving models results in knowledge bases which have a context and credibility. The context is both internal to the model, and relates the model to the external environment. The credibility comes from understanding the process of building the knowledge.

There are other important advantages to working with multiple-player models. This approach has the potential to achieve informed agreement among decision-makers who must work together towards overall goals. Too often, there is misunderstanding and misinformation between people when understanding and shared knowledge are critical to making the best possible decision.

Currently there are a few decision support systems that allow multiple decision-maker interaction within the context of a single model. These systems use only direct interaction with no expert system element. The models are created using Decision Support system techniques such as spreadsheets. Communication through mutual interaction as part of the decision making process is important
even in such a limited environment.

"Among the side benefits, this process of consensus-development takes place much more rapidly than would be possible without DSS tools. Use of these systems focuses attention on the subject at hand and provides results very quickly. Thus, we get the benefits of collective and eclectic decision-making without the stultifying slowness with which these methods would otherwise be practiced. Furthermore, we often find ourselves startled at what our colleagues are really thinking and feeling; without these processes, we would not understand one another nearly so well. With them, we make the most of humans' natural inclination to work in support with one another." (Wagner 1982)

**Market Factors**

In market terms, the technology and cost of microcomputers has reached a level where multiple-computer interaction among decision-makers is feasible. The computing power is available on these machines to create high-level human computer interfaces and drive moderately complex decision support environments. Network technology provides the communication between machines and people. Affordable networking is, or will soon be, available for the same machines. These machines include Sun, Apollo, Symbolics, Apple, and AT&T personal workstations.

The impact of a good network design is that it allows the system designer to facilitate rapid and rich communication and interaction among several decision makers operating their own software on their own machine. Available networks and powerful microcomputers allow the creation of a cohesive interactive decision support environment. This enables interaction between people and their proxies through interaction between machine models. The availability of powerful personal computers with useful and affordable network facilities moves the
design of decision support systems away from the limited model of single person, single machine interaction, towards many people interacting in a common decision making environment.

Summary

In this chapter we have focused on the importance of placing a decision support system in the mainstream of communication between people. The effect of doing this is three fold. One, to focus and improve communication among decision makers. Two, to gather or capture the knowledge that is important to making these decisions so that the available information can be validated by several experts. Three, so that the information has a context and makes sense, at least, to the group that develops and works with it.

The concepts of ease of use, effectiveness of human / machine interface, types of knowledge and representation, uncertainty and methods dealing with it, are all critical when we begin to share our knowledge with other people. Decision-making often affects other people and is often done in consultation with other people. This will not change. What we can consider is how we can create a decision support system that can take advantage of the situation and provide some leverage to decision makers by enabling them to share knowledge as well as responsibility.
Chapter Eight
Conclusion

Target Audience for a Microcomputer Decision Support system

The recommended target audience are those individuals who are making difficult decisions that have a large measure of uncertainty, and whose decisions will have a significant impact on an organization. Middle managers fit this role. Their involvement within a small decision making group of other managers allows us to design a Decision Support system that takes advantage of communication within that group for the purpose of knowledge capture. Figure 16 illustrates that the decision making group has the largest vested interest in the result of their decision relative to the other groups shown. In addition, the individual decision maker has only a little more time and a little less responsibility than the decision making group as a whole. His major concerns relative to a Decision Support system relate to modelling and manipulation of knowledge. Within a decision making group the individual turns to functions that will facilitate communication with other members of the group. The groups represented by DP and Office Support staff have been included in this illustration to highlight the concerns of the groups that have been responsible for selecting Decision Support systems in the past. Their concerns are reasonably addressed by the functions provided by traditional Decision Support systems. With the increasing ease of use and decreasing cost of software and computers, middle managers are buying computers and software from their own discretionary budgets for their own use. Therefore, the considerations appropriate to the individual decision maker and the decision making group should apply as the major focus for any new microcomputer-based
Who Cares: Who has the largest vested interest in the result of decisions which involve Decision Support systems.

Time Available: Who has the greatest time available to use a difficult system.

Uncertainty: Who has the largest amount of uncertainty, or risk involved in their decisions.

Differentiation

In order to differentiate our target Decision Support system from other Decision Support products in the general market we need to look at how Decision Support manufacturers are attempting to position their products in the market. Figure 17 provides three characterizations of how Decision Support manufacturers view their products. The manufacturer's view of their product does not always accurately reflect their product's final implementation and subsequent use. The manufacturer's view represents how the manufacturer wants his product to be perceived by the customer.
Figure 17. Three Views of Decision Support Systems

**Top View:** Corresponds to how makers of Traditional Decision Support systems see themselves. The emphasis is on access to information and presenting the retrieved information in a variety of forms. Interaction and other factors become important as competition increases.

**Middle View:** Corresponds to how makers of Project Management systems, Educational systems, some Expert systems and a variety of other makers of non-traditional Decision Support see themselves. The factors in the top view remain, but the emphasis has changed. Human-computer communication as well as computer-computer and human-human communication in the form of electronic mail and multiple player games and models, has become a major selling point.
Depending on the specific application, knowledge manipulation may be just as important as communication. Explicit knowledge enables the same system to operate in a variety of domains. Simulation is important as a method of answering "what if" type questions.

**Bottom View:**
Again, this view incorporates the factors noted in the middle and top views, but the emphasis has changed. The bottom view corresponds to our target Decision Support system. With the use of knowledge in the middle view came the requirement to extend and alter prepackaged knowledge (Knowledge Capture).

Knowledge Use refers to the application of knowledge for the purposes of problem solving, creating simulations or proxies, or for other processes of information collection and processing.

Knowledge Sharing refers to interpersonal communication through the medium of the Decision Support system. This communication between decision makers represents not only pass through knowledge but includes interactions that can be captured and represented as knowledge for later use.

All three of the above views of Decision Support reflect how each manufacturer envisions their product. The implementation of a particular product in any of the views will not always reflect the initial goals.

The views in Figure 17 represent a starting point for the design of a Decision Support system for each of the three levels of decision-maker involvement.
Our target Decision Support system should include the factors that are characterized in Figure 17 in the top and middle views. It should focus on knowledge capture, use and sharing among decision makers as characterized in the bottom view.

**Recommended Approach to Designing a DSS for a Commercial market**

The best way to approach the design of a new micro-computer Decision Support system in order to achieve the focus illustrated in the bottom view is to center the design of the product around interpersonal communication functions. Traditionally, Decision Support systems have be built around information retrieval functions. Focusing the design of our target Decision Support system on interpersonal communication enables the manufacturer to avoid the startup costs associated with building a traditional Decision Support system that can then be extended to the level of function illustrated in Figure 17. Using interpersonal communication as a basis for developing a Decision Support system also avoids the problem of direct competition with established manufacturers since the perceived function of the product is different from existing Decision Support systems. This approach allows for the initial development of very simple products, such as software that presents sequences of still graphics that have been prepared by other software applications. While simple, this type of application begins a process of development that has its roots in the process of mediating communication between people. If we view our target Decision Support system as a line of developing products, the next range of products would likely involve products which extended and complemented the presentation products by providing a multiple player environment that included both listening and acting proxies. From this point the manufacture is in a good position to introduce knowledge
capture as a basis for knowledge use within an environment of interpersonal interaction and communication. By focusing first on interpersonal communication rather than information retrieval we have been able to envision a sequence of development that places knowledge capture in a dominant position for the system user rather than in a position secondary to the use of predefined and preorganized knowledge as is the case typical of expert systems.

If we apply this view of a Decision Support system based on interpersonal communications to the preceding chapters in this thesis we can see that each chapter represents an important stage towards motivating a basic shift in the concept of decision support away from an system centered on information retrieval towards a system based on interactions between people. The existing market provided our initial motivation both to take advantage of a growing market segment and to find a way to deal with the competition. Examples of highly interactive systems showed that we could do better than current Decision Support systems, but that interaction was not something that could be layered on top of a preconceived system. To be effective interaction had to permeate the design of the entire system.

Unconventional systems which provide some aspects of decision support are useful and will continue to be useful. They highlight problems and provide some solutions to problems that are not generally considered within the design of a Decision Support system but should be. In Chapter three unconventional Decision Support systems introduced us to the concepts of multiple player, proxy interaction, and learning as important considerations for a Decision Support system.

Design, uncertainty and the use of knowledge come together to motivate a concept of Decision Support which involves time and interaction with other people
and generally characterizes decision-making as an ongoing process which requires continual development. Chapter seven on communication and interaction provides the motivation for a Decision Support system that includes interpersonal communication. It is a function that will provide a basis knowledge capture which in turn facilitates ongoing development of the Decision Support system by its users.

To facilitate the development of a Decision Support system which satisfies market considerations and defines a relationship between the seemingly diverse range of considerations noted above requires a strategy. Such a strategy is to shift the central focus of the Decision Support system from its traditional role of information retrieval to the process of interpersonal communication.

Our recommended strategy for approaching the Decision Support market is to design small applications that implement a single useful function within the context a line of products centered on interpersonal communication as a basis for a Decision Support system. For example, a simple presentation system as noted above may be a good beginning. Design the first application so that it will enhance existing Decision Support products but develop subsequent applications so that together the applications create an increasingly useful Decision Support environment.

**Contributions**

In this thesis we have shifted the concept of Decision Support from a market ploy to a basis for the design of a system which actively supports a complex and ongoing process of human decision making. We have brought together market considerations with concepts for interplay between people and the knowledge that can be captured, contained and manipulated using available and emerging technologies. We have drawn from unconventional sources in an effort to highlight
design considerations which are important and are not currently part of the mainstream of Decision Support products. These factors provide us with the view that will help to differentiate our target Decision Support system in the market.

We noted how knowledge is used and characterized by people and showed through a variety of examples how knowledge could be used. The problem of updating knowledge (knowledge capture) could be solved by our requirement for communication and sharing of information between people through the mechanism of simulations and other devices within the Decision Support system.

We made recommendations as to where to focus design effort and how to approach the market place with a new product that may for the first time deal directly in the process of human decision making.

Summary

Traditional Decision Support systems began the process of providing information directly to the decision-maker. They served to increase the awareness of the decision makers: an awareness of the value stored within their computer systems. Traditional Decision Support systems provided the awareness but did not completely satisfy the decision makers' requirements for access and control of credible information.

Micro-computers brought with them a degree of independence from the traditional data processing departments. More importantly, micro-computers provided the CPU cycles needed to begin improvements to the user interface. Beginning with Visicalc, spreadsheets made limited memory micro-computers useful for decision makers. The modelling power and ease of use soon made spreadsheets the best selling business software for micro-computers.

Traditional Decision Support systems were enhanced to include spreadsheets
and modified to operate, in part, on popular micro-computers. Still, traditional Decision Support systems have not gained a larger market share, nor greatly improved their acceptability to decision makers.

This research began as work towards a better traditional Decision Support system, Conquer (Sydney 1984), and continued, as an awareness of Decision Support, through the design of several other products. These products which included "Money Trader", "Manager's Baseball", and several educational products, contained important elements of Decision Support that were lacking in the traditional Decision Support systems.

Other computer systems, including those first developed at Xerox PARC, served to distinguish elements that appear important to the strong interaction required of a Decision Support system. In addition, the work at PARC was constructive providing several methods for the development of complex systems that could be easily controlled by the user of the system.

Control of a computer system is important but does not create a Decision Support system. In order to make use of the available technology to create a viable Decision Support system we must examine, in concert, how decisions are made, and what existing hardware and software technology can be applied. The goal in this case is to identify the critical aspects of design which must be considered during the design and implementation of a useful knowledge-based Decision Support system. In order to be useful, a commercial Decision Support system must meet at least two overriding criteria. It must be accessible to the decision-maker and it must sell well enough for a software manufacturer to consider building it at all.

The traditional Decision Support systems were based on a facility for accessing stored information. The approach is correct but decision makers often
require more than just information. They require context, and procedural knowledge not normally contained within a database. They also require focused communication with other people who have special knowledge, and with other people who share responsibility for the decisions. These requirements have no technical solutions but can benefit from the solutions to similar types of problems in other areas of software application and development. To this end we examined decision making as an ongoing process of problem solving. The critical factor in this process was the uncertainty that was often involved in each step of the process. By focusing on the types of knowledge involved we began to examine ways of dealing with uncertainty. The decision-maker could benefit by extending the knowledge embedded in the Decision Support system to more clearly reflect his own working environment. Without the decision makers ongoing involvement in defining the knowledge within the Decision Support system, the system would become more and more of a learning game rather than an appropriate tool for making real decisions.

Real decisions require interaction to gain knowledge and benefit from the support and experience of other people. If a computer system is to work in such an environment it must have some painless method of capturing some of this knowledge. The best way for a Decision Support system to capture knowledge has other less tangible but equally important benefits. By using the Decision Support system as a multiple-player modelling tool, the parameters of decisions and the knowledge involved in their execution remain accessible within the Decision Support system. In addition, working through a multiple-player-model the decision makers can act to focus communication and knowledge constructively on topics of common importance. Computer mediated interactions can have the effect of improving people communication because in a concrete environment solutions
that once seemed obvious can be seen by all to have weaknesses. With a common understanding of the problems and uncertainties involved in making decisions, a group of decision makers may begin to direct the process of solving problems and creating new direction, rather than simply setting goals and reacting to crisis.

We are entering a time when computers are becoming powerful and inexpensive enough to be useful as decision making tools. New machines do not mean only new software technologies. Powerful approaches to solving difficult problems have been developed in research environments for many years. These techniques and others which will be developed will form the basis for consumer software for the next few years. Decision Support systems, whether they are labeled as such or not, will be important components of the software industry. Their market will be oriented towards individuals who control knowledge within a decision making group. Knowledge-based Decision Support systems are just beginning to emerge into the market place, as a result the information contained in this work may be useful to those people who are now in the process of designing systems which are effectively knowledge-based Decision Support systems.

In terms of future directions, this work raises more questions than it answers. It is intended to distinguish problems and approaches to solutions which must be addressed during the design of any Decision Support system. Most commercial implementations will make simplifying assumptions to reduce the complexity and cost. However, if the designer has considered the aspects of Decision Support addressed in this work they may make useful simplifications.

The emerging problems and benefits of computer-mediated environments and communications between people are among the most interesting aspects for further study. How knowledge is organized or who organizes the knowledge remains a question. Knowledge could potentially be collected as a result of
continuing and detailed workshops. What are the problems of mediating knowledge exchange? Should it be controlled and how can it be facilitated outside of a corporate environment? This area of people networking seems to have the greatest impact on the way people make decisions, and yet it is an area that is often invisible. How can the concepts of communication that have been discussed here be extended to involve large numbers of people on a casual basis, or small groups of people intimately? How can we capture knowledge more effectively and make it available in a way which is most usable? These questions and more remain and are the subject of future work.
References


Intelligenetics. (1984)
Personal Communication


Markoff, John. (1982). "In Focus: Smalltalk", in InfoWorld


Odesta, "Odesta Helix", Odesta, Northbrook. IL, in Macworld, September/October, 1984, Vol. 1 No. 4, pp. 16


Sheil, Beau. (1983a). "The artificial intelligence tool box (Draft)", Xerox Special Information Systems, in Walter Reitman (ed), Proceeding of the NYU Symposium on Artificial Intelligence and business, ABLEX


