

AD-A251 869



NAVAL POSTGRADUATE SCHOOL
Monterey, California

2



DTIC
ELECTE
S D
JUN 17 1992

THESIS

A FRAMEWORK FOR SELECTION
OF DSS
DEVELOPMENT METHODOLOGY

by

Marcus G. Foote

March, 1992

Thesis Advisor:

Balasubramaniam Ramesh

Approved for public release; distribution is unlimited.

92-15787



92 6 16 148

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No 0704-0188	
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY Multiple Sources			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S)			5 MONITORING ORGANIZATION REPORT NUMBER(S)		
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b OFFICE SYMBOL (If applicable) 37	7a NAME OF MONITORING ORGANIZATION		
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000			7b ADDRESS (City, State, and ZIP Code)		
8a NAME OF FUNDING SPONSORING ORGANIZATION		8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c ADDRESS (City, State, and ZIP Code)			10 SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO	PROJECT NO	TASK NO
					WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) A FRAMEWORK FOR SELECTION OF DSS DEVELOPMENT METHODOLOGY (UNCLASSIFIED)					
12 PERSONAL AUTHOR(S) Foote, Marcus G.					
13a TYPE OF REPORT Master's Thesis		13b TIME COVERED FROM _____ TO _____		14 DATE OF REPORT (Year, Month, Day) March 1992	
15 PAGE COUNT 63					
16 SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17a CODES FIELD GROUP SUB-GROUP			17b SUBJECT TERMS (Continue on reverse if necessary and identify by block number) DSS Methodology, DSS development, DSS design process		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) Researchers in the area of Decision Support Systems (DSS) have focused for years on development of a single methodology for the development of DSS. Recent literature in this area suggests there is no single accepted definition or set of characteristics for all DSS; decision features vary with every situation; and DSS development should be based on the salient characteristics of a decision situation and the environment in which it exists. This thesis proposes a broad approach, recognizing different decision features (technology levels, participants, structure, environmental factors, and construction approach) require different approaches. An individual should be able to select the appropriate methodology based on known characteristics of DSS as well as its development and usage environment.					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a NAME OF RESPONSIBLE INDIVIDUAL Balasubramaniam Ramesh			22b TELEPHONE (Include Area Code) (408) 646-2439		22c OFFICE SYMBOL AS/RA

DD Form 1473, JUN 86

Previous editions are obsolete

S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Approved for public release; distribution is unlimited.

A Framework for Selection
of DSS
Development Methodology

by

Marcus G. Foote
Lieutenant, United States Navy
B.A., Baylor University, 1979

Submitted in partial fulfillment
of the requirements for the degree of

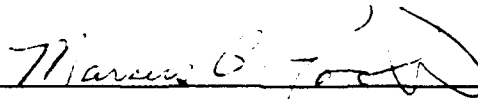
MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL

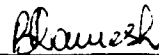
March 1992

Author:

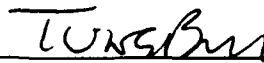


Marcus G. Foote

Approved by:



Balasubramaniam Ramesh, Thesis Advisor



Tung Bui, Second Reader



David R. Whipple, Chairman
Department of Administrative Sciences

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	PROCESS OF DSS DESIGN	4
	A. TECHNOLOGY LEVELS	8
	1. Specific DSS	8
	2. DSS Generator	8
	3. DSS Tools	9
	B. PARTICIPANTS	10
	1. Manager or user	10
	2. The Intermediary	12
	3. The DSS Builder	12
	4. The Technical Support Person	12
	5. The Toolsmith	13
	C. ENVIRONMENTAL FACTORS	13
III.	OVERVIEW OF APPROACHES TO DSS DEVELOPMENT	18
	A. TRADITIONAL (LIFE CYCLE) APPROACH	18
	B. ITERATIVE APPROACH	20
	1. Iterative	22
	2. Prototyping	22
	3. Adaptive	25

C.	KNOWLEDGE ANALYSIS AND ENGINEERING	27
D.	REPRESENTATIONS, OPERATIONS, MEMORY AIDS, CONTROL (ROMC)	29
IV.	SELECTING A DESIGN METHODOLOGY: A BROAD APPROACH . .	32
A.	CONSTRUCTION APPROACHES	33
1.	Quick-Hit	35
2.	Staged Development	36
3.	Complete DSS	37
B.	TECHNOLOGY LEVELS	38
C.	PARTICIPANTS	40
D.	ENVIRONMENTAL FACTORS	42
E.	DECISION SUPPORT SYSTEM CATEGORIES	44
V.	CONCLUSION	47
	LIST OF REFERENCES	50
	BIBLIOGRAPHY	52
	INITIAL DISTRIBUTION LIST	56

ABSTRACT

Researchers in the area of Decision Support Systems (DSS) have focused for years on development of a single methodology for the development of DSS. Recent literature in this area suggests there is no single accepted definition or set of characteristics for all DSS; decision features vary with every situation; and DSS development should be based on the salient characteristics of a decision situation and the environment in which it exists. This thesis proposes a broad approach, recognizing different decision features (technology levels, participants, structure, environmental factors, and construction approach) require different approaches. An individual should be able to select the appropriate methodology based on known characteristics of DSS as well as its development and usage environment.



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

I. INTRODUCTION

In 1980 Ralph H. Sprague, Jr. (1980) stated that we were "on the verge of another 'era' in the relentless advancement of computer based information systems in organizations." The topic of his article was a new approach in the evolution of information technology called Decision Support Systems (DSS). His examination of alternate views of Decision Support Systems and the subsequent framework he proposed has become a cornerstone of this new technology. A decade later, Decision Support Systems have flourished as computer technology has improved through larger memory, more processing power, and proliferation of micro-computers.

Throughout the intervening years, researchers in the area of DSS have focused on a single methodology based on the premise that there is or should be a single methodology to all DSS (Ariav and Ramesh, 1989). Yet with all the advances in technology, no single methodology for the development of DSS has gained acceptance as the methodology. Recently, among researchers, more credence has been given to the idea that DSS development should be based on the salient characteristics of a decision situation and the environment in which it exists.

The design of Decision Support Systems has been extensively studied and numerous methods for the analysis and design proposed over the past decade.

This thesis proposes a broad approach, recognizing different decision features (technology levels, participants, structure, environmental factors, and construction approach) require different approaches. An individual should be able to select the appropriate methodology based on known characteristics of DSS as well as its development and usage environment.

Chapter II discusses the background of DSS, its development from Electronic Data Processing (EDP) and Management Information Systems (MIS), and examines existing definitions and characteristics of DSS. Three technology levels for the construction of DSS (specific DSS, DSS generator, DSS tools) are discussed. Selection of the appropriate methodology and the construction of a DSS is accomplished by participants, either an individual or group. Five types of participants (manager/user, intermediary, DSS builder, technical support person, toolsmith) are identified. Environmental factors affecting DSS Design are identified as solicited, recurring, critical, and short or long-term. Finally, a discussion of the determination is made whether to develop a DSS through a single user or a team effort.

Chapter III is an overview of DSS development approaches. Three broad DSS categories (a) Traditional (Life Cycle), (b) Iterative, and (c) Representations, Operations, Memory Aids, Control (ROMC) are proposed as initial steps in identifying the appropriate DSS methodology for a decision situation. The

Iterative Approach, often called prototyping or evolutionary, can lead to confusion since other methodologies by the same names utilize the Iterative Approach or category in their DSS construction. Iterative, Prototyping (i.e. Evolutionary) and Adaptive methodologies are described clarifying subtle differences among these approaches.

Chapter IV proposes a broad approach for selecting a design methodology for DSS development, recognizing different decision features (construction approach, technology levels, participants, structure, and environmental factors) and based on three DSS techniques (Traditional, Iterative, ROMC) as underlying categories for all methodologies. Figure 1 in Chapter IV depicts a broad approach which recognizes seven areas which define different decision situations. Approaches to DSS construction are classified into three categories: (a) quick-hit, (b) staged development, and (c) development of a complete DSS. Their advantages and disadvantages are identified. Each level, and the subsequent elements, of the broad approach is reviewed as it pertains to the environment in which a decision situation exists.

Chapter V is the conclusion where implications of the proposed broad approach are made.

II. PROCESS OF DSS DESIGN

Electronic Data Processing (EDP) was the forerunner of information systems, and was initially utilized for automating paperwork within an organization. Its basic characteristics include:

- a focus on data, storage, processing, and flows at the operational level;
- efficient transaction processing;
- scheduled and optimized computer runs;
- integrated files for related jobs; and
- summary reports for management. (Sprague, 1980, p.2)

The study of Management Information Systems (MIS) began in the early '70s. "A management information system is a formal, computer-based system, intended to retrieve, extract and integrate data from various sources in order to provide timely information necessary for managerial decision making (Turban, 1990, p.6)." Most suited for routine, structured decisions, MIS have proven extremely successful for managing large quantities of detailed data utilized in transactional processing and emphasizing integration and planning. The characteristics of MIS include:

- an information focus, aimed at the middle managers;
- structured information flow;
- an integration of EDP jobs by business function, such as production MIS, marketing MIS, personnel MIS, etc.; and
- inquiry and report generation, usually with a database. (Sprague, 1980, p.4)

While highly successful in support of routine decision-making, MIS have not proven equally successful in support of complex decision-making. The development of DSS has bridged the gap between structured routine decision-making and unstructured complex decision-making. "The concepts involved in DSS were first articulated in the early '70's by Michael S. Scott Morton under the term 'management decision systems'." (Sprague, 1980, p.1) Characteristics of DSS include:

- decision focused, aimed at top managers and executive decision makers;
- emphasis on flexibility, adaptability, and quick response;
- user initiated and controlled;
- support for the personal decision making styles of individual managers. (Sprague, 1980, p.4)

This list of DSS characteristics proved too restrictive and evolved over the years to include a broader view. Researchers such as Alter and Keen have suggested that DSS characteristics also are aimed at less structured, underspecified problems

not typically handled by upper level managers. DSS "combine the use of models and analytic techniques with traditional data access and retrieval functions (Sprague, 1980, p.2)." DSS are highly flexible and adaptable to changing management environments, and are readily useable by non-computer experts. Turban (1990) expanded the characteristics described by Sprague into the following list:

- DSS provides support for decision makers mainly in semistructured and unstructured situations by bringing together human judgment and computerized information.
- Support is provided for various managerial levels, ranging from top executives to line managers.
- Support is provided to individuals as well as to groups.
- DSS provides support to several interdependent and/or sequential decisions.
- DSS supports all phases of the decision-making process: intelligence, design, choice, and implementation.
- DSS supports a variety of decision-making processes and styles.
- DSS must be adaptive over time.
- DSS should be easy to use.
- DSS attempts to improve the effectiveness of decision making (accuracy, timeliness, quality), rather than its efficiency (cost of making the decision, including the charges for computer time).
- The decision maker has complete control over all steps of the decision-making process in solving a problem.

- DSS leads to learning, which leads to new demands and the refinement of the system, . . . in a continuous process of developing and improving the DSS.
- DSS should be easy to construct.

There is no single definition for Decision Support Systems which all researchers agree upon. Definitions do not provide any clue to a methodology for DSS construction and seem to be as plentiful and varied as researchers on the subject of DSS. Efraim Turban states:

A DSS is an interactive, flexible, and adaptable CBIS that utilizes decision rules, models, and model base coupled with a comprehensive database and the decision maker's own insights, leading to specific, implementable decisions in solving problems that would not be amenable to management science optimization models per se. Thus, a DSS supports complex decision making and increases its effectiveness (Turban, 1990, p.109).

Ralph H. Sprague, Jr. (1980) characterized DSS "...as interactive computer based systems, which help decision makers utilize data and models to solve unstructured problems." Turban notes, "Because there is no consensus on what a DSS is, there is obviously no agreement on the characteristics and capabilities of DSS (Turban, p.109)."

Just as no two definitions are the same, the situation in which the decisions are developed varies with the given environment. Each situation must be examined for the salient characteristics in its specific environment. Salient features

which must be considered in creating any DSS are (a) technology, (b) participants, (c) environmental factors, and (d) approaches to construction (Sprague, 1980).

A. TECHNOLOGY LEVELS

Ralph H. Sprague, Jr. (1980) identifies three technology levels (specific DSS, DSS generators, DSS tools) for the construction of DSS. "They are used by people with different levels of technical capability, and vary in the nature and scope of task to which they can be applied." (Sprague, p. 6) These technology levels are expanded by Efraim Turban (1990) as a ". . . useful framework for understanding DSS construction."

1. Specific DSS

The Specific DSS is the final product of hardware/software which actually accomplishes the task and aides the decision-maker in resolving a specific set of factors. Turban (1990) provides a well-known example of a specific DSS in the police-beat allocation system implemented by IBM in San Jose, California. This system allowed police to display a map on a video terminal, recall various information, and manipulate the information into a variety of alternatives.

2. DSS Generator

"The DSS Generator is a 'package' of related hardware and software which provides a set of capabilities to quickly and easily build a Specific DSS

(Sprague, 1980, p. 6)." The Geodata Analysis and Display System (GADS) was used in the police-beat allocation system mentioned above. GADS can provide "... maps, data, a data dictionary, and statements for special procedures that can be changed from one application to another (Turban, 1990, p. 164)." Other examples of DSS generators include Lotus 1-2-3 for microcomputers, and the Interactive Financial Planning System (IFPS).

The development and use of DSS Generators promises to create a 'platform' or staging area from which Specific DSS can be constantly developed and modified with the cooperation of the user, and without heavy consumption of time and effort (Sprague, 1980, p. 7).

3. DSS Tools

Often called software utilities, tools are at the lowest level of DSS development. Tools, such as graphics (hardware and software), query systems, random number generators, spreadsheets, and programming languages, are used to develop both specific DSS and DSS generators.

This category of technology has seen the greatest amount of recent development, including new special purpose languages, improvements in operating systems to support conversational approaches, color graphics hardware and supporting software, etc (Sprague, 1980, p. 7).

B. PARTICIPANTS

An individual or group selects the appropriate methodology based on known characteristics of DSS as well as its development and usage environment. Five types of participants have been identified in the construction and operation of DSS. One individual may assume several of the roles or several individuals may fill one role (Turban, 1990).

1. Manager or user

An individual or a group, the Manager is the decision maker. This individual is responsible for solving the problem, taking the action, and assuming the consequences. The Manager is concerned with the DSS performance objectives--decision types and support types (Sprague, 1980).

The type of decision involved in the task includes numerous questions or capabilities:

- Does the DSS provide for a semistructured or unstructured situation?
- Does the DSS provide for only individuals, or groups, or both? Does it provide for various managerial levels?
- Does the DSS allow for several interdependent and/or sequential decisions?

A structured decision is one possessing relatively few parameters, situations, characteristics, or possible conclusions. While a structured decision may require

a DSS, today's DSS have evolved such that they are most advantageous for semistructured or unstructured decisions. A semistructured or unstructured decision has multiple parameters, numerous situations, characteristics, and possible conclusions. It is the complexity of the decision which the DSS addresses.

A structured decision normally has only one user. The semistructured or unstructured decision requires support for managers at all levels (individuals or groups). Due to the very complexity of decision-making in unstructured or semistructured situations, the DSS must provide support for interdependent and/or sequential decisions. The complexity of the decision alternatives require a DSS that is flexible, adaptable, and easy to use. The type of support in which a Manager is concerned in determining DSS performance objectives includes the following questions or capabilities:

- Are the phases of the decision-making process (intelligence, design, choice, and implementation) provided?
- Does the DSS provide a variety of decision-making processes and styles?
- Is the DSS adaptive and easy to used?

The DSS must identify the conditions of the decision, provide possible action, select and implement the best action.

2. The Intermediary

The Intermediary is someone who helps the Manager or User in the use of the DSS. This individual may be a "staff assistant" (Sprague, 1980) or a "staff analyst" (Turban, 1990) who assists by typing on a terminal, making suggestions, or interacting with the user. The Intermediary, also called a "chauffeur", was of particular help with early DSS which were not user friendly and often required additional assistance (Turban, 1990).

3. The DSS Builder

The DSS Builder, often called the Facilitator, makes the technical decisions about the DSS. This individual must be knowledgeable of the hardware and software, the problem areas, current DSS technology, and determine which tools and generators are to be used. Typically the DSS Builder is a member of an information center, but in any case, must be familiar with the problem area the DSS is intended to address. "The information center provides the end-user with appropriate education, technical support, usable tools, consulting (e.g., selection of DSS software), accessibility to databases, and convenient access to other computer systems (Turban, 1990, p. 166)."

4. The Technical Support Person

The Technical Support Person may be a programmer or computer scientist who provides support to the DSS Builder. While an extensive knowledge

of the problem area is not necessary, this individual must possess sufficient understanding to provide new databases, analysis models, and data display formats (Sprague, 1980).

5. The Toolsmith

The toolsmith's job is to improve the DSS package through better hardware and software. This individual incorporates new languages and technology into the DSS package, and develops new tools to improve efficiency and effectiveness. Concerned with the dialogue, data handling, and model handling, the toolsmith uses new technology to develop and improve the architecture of these basic tools (Sprague, 1980).

C. ENVIRONMENTAL FACTORS

DSS have evolved from aiding and supporting only top executive decision makers in very specific, structured problems, to supporting all managerial levels in nonspecific, unstructured problems. Currently, DSS normally support users in one major system or one of its parts. But as the use of DSS expands, "... future systems may well cut across the entire organization and go beyond" (Thierauf, 1988, p. 354). Future DSS will not only extend current DSS practices, but will be required to handle more difficult and complex problems. Such complex organizational problems will require involvement of all users (particularly top

managers and their staffs) to develop a system which can interact in countless variations. The system will be ". . . dependent on a highly integrated, total corporate planning model that interacts with all of the organization's major systems and their related parts" (Thierauf, 1988, p. 354). As the problems handled by DSS become more complex, DSS must further evolve from "problem solving" (which is solving an existing problem) to "problem finding" (which is searching the environment for new problems before they occur). Problems will be less structured and nonspecific, therefore greater care will be required in specifying the characteristics of the environment for which the DSS will be designed. Complex problems will require greater examination of the environment and the way organizations view a problem. (Thierauf, 1988) Mahmood, Courtney, and Burns (1983) identify the "environmental factors affecting DSS Design" as: solicited, recurring, critical, and short or long-term. Turban (1990) and Locander, Napier, and Scamell (1979) add team-developed vs. user-developed DSS to these environmental factors. The organizational environment and DSS design should first be categorized as solicited or unsolicited. Differences exist between the methodologies to be used in the development of a DSS which is solicited by upper management and one which is unsolicited. If solicited, a particular problem has already been identified and is under management's cognizance. Supported by upper management, a DSS is more likely to be well-received by lower echelons

within an organization. A known or suspected problem is addressed and an immediate fiscal incentive recognized. If unsolicited, the developer of the DSS must convince management that a problem exists and will be favorably affected or corrected by the application of the DSS. The developer may experience lack of support from upper management as an outsider and face an uphill battle winning support from lower echelons. Whether the DSS design environment is solicited or unsolicited, systems must be constructed to meet managerial styles and provide required support.

Once a DSS design is categorized as solicited/unsolicited, the decision incidence is determined as either recurring or non-recurring. A recurring decision occurs repeatedly, while a non-recurring decision occurs once or rarely. While a non-recurring decision will use a DSS once or at most a few times, a recurring decision will require a more complex DSS covering a much wider set of circumstances. As fourth generation languages are developed which allow "what if" questions, DSS must be capable of responding to an environment which cannot be specified in advance. "What if" questions increase management flexibility whether dealing with solicited/unsolicited or recurring/non-recurring problems. In the future, management must be able to perceive trends before they effect an organization. "What if" questions allow management to look ahead, anticipate,

and adjust long term strategic planning for future economic, political, and social trends. (Thierauf, 1988)

Next the decision is determined to be critical or non-critical. If non-critical, the decision has little effect on the organization and will have little urgency. If critical, the decision has numerous repercussions and has far greater urgency in the organization. Finally, the decision is determined to be either short-term or long-term. A short-term decision may be two weeks or one year and a long-term decision may be a couple of months or a couple of years. The period of time is determined by the organization, the situation, and the decision under consideration. (Mahmood, Courtney, and Burns, 1983) As managers seek to expand the use of DSS to more complex problems involving entire organizations, more DSS design will be long-term and strategic in nature. Today's business environment requires that management develop long-term planning to ensure competitiveness. Long-term planning must be designed as a continuous process. In other words, a long-term plan must be capable of translating into a medium-term and ultimately short-term plan. To meet the needs of all levels, DSS design must possess sufficient flexibility to cover the numerous environmental variations encountered between short-term and long-term plans. Future DSS must be forward-looking and designed to anticipate organizational problems in strategic planning.

With the environmental factors determined, the next step in the development of a DSS is to determine whether it will be developed by a team or a user. DSS during the 1970's and earlier 1980's generally were developed by teams for very large complex projects. The teams consisted of many participants (users, intermediaries, builders, technical support experts, and toolsmiths) and were usually costly, complex and difficult to manage. During the later portion of the 1980's, DSS were developed more frequently by users. This approach gained momentum due to the influx of micro computers, improvements in hardware and software, and developments in software tools and generators. Today, individual users, groups, or a mixture of both may develop DSS or new applications. Improvements in computer technology and evolution within DSS have lead to proliferation in the number of DSS and the processes by which they are developed. Chapter III is an overview of the major approaches to DSS development.

III. OVERVIEW OF APPROACHES TO DSS DEVELOPMENT

When researching types of DSS methodologies, one is immediately struck by the diversity of DSS development approaches discussed/proposed in the literature: Traditional, Iterative, Prototyping, Incremental, Decision Research, Knowledge Representation, and Heuristic to name only a few. Three broad DSS categories stand out as initial designations for all methodologies: (a) Traditional (Life Cycle), (b) Iterative, and (c) Knowledge Analysis and Engineering.

Representations, Operations, Memory Aids, Control (ROMC) is a framework for systems analysis that any category can use to identify characteristics and capabilities required in a DSS.

A. TRADITIONAL (LIFE CYCLE) APPROACH

The Traditional Approach is based on the assumption that the information requirements of a system can be predetermined. Traditional methodologies are typically used in situations where information requirements are reasonably well defined with detailed specifications. Translated into system designs, this approach is typically used for large, structured applications, requiring significant investments of time and resources to reach desired levels of detail (Meador and Rosenfeld, 1986). Involving very structured situations, information requirements are

determined by logical analysis. Though the System Development Life Cycle (SDLC) has many versions, it can be generalized into six basic phases (Turban, 1990):

- Systems analysis and planning - a conceptual design is developed with objectives, structure, resources, and controls.
- Design - a physical system (hardware and software) is developed from identified information requirements.
- Construction and testing - software is written, tested, and improved.
- Implementation - physical system and software is installed, operated, and training held.
- Operation and maintenance - develop maintenance, security, error detection, and backup system.
- Evaluation and control - monitor system for progress. Perform acceptance test, cost-benefit analysis and audits.

Meador and Rosenfeld (1986) identify numerous DSS methodologies which use the Traditional approach: IBM's Business Systems Planning, Yourdon's Structured Analysis Techniques, Softech's Structured Analysis and Design Techniques (SADT). Additional examples are provided by the Team Approach of Locander, Napier, Scamell (1979) and the Knowledge Representation methodology of Elam and Henderson (1983). The Traditional Approach is best suited for large structured applications and is unsuitable for small applications or large unstructured situations. Most DSS are developed for complex unstructured

situations in which the initial DSS will not work satisfactorily. In such cases, the design, implementation, and evaluation processes typically occur concurrently and are evolutionary in nature. The processes evolve as circumstances and user requirements change in the unstructured environment. Many application requirements cannot be prespecified, change frequently, and must be addressed quickly. Meador and Rosenfeld (1986) indicate these factors led many authors to develop a methodological approach which is evolutionary and uses prototyping and rapid development tools (e.g. fourth generation languages). Such evolutionary methods are categorized under the Iterative Approach.

B. ITERATIVE APPROACH

The Iterative Approach is well suited for unrestrictive, non-specific, and less structured decision requirements. Combining traditional data access and retrieval functions with models and analytic techniques, the Iterative Approach builds a DSS in a series of short steps (Sprague, 1980). This approach allows for quick and easy changes through the use of tools and DSS generators. "The iterative design process combines four major phases of the Traditional SDLC (analysis, design, construction, and implementation) into a single step that is repeated (Turbin, 1990, p.169)." Courbon et. al. (1980) describe four activities included in the Iterative Approach:

- Opening a line of communications between participants, the user and builder first select or identify important sections of a problem for which a DSS is constructed. The sections should be small and of value to the decision maker, yet provide the nature of the problem and required support.
- Emphasizing small scale and simple, a small usable system is constructed to assist the decision maker. Unlike the Traditional Approach of SDLC, no major systems analysis or feasibility analysis is completed. The various steps of the SDLC are accomplished on a quick, but small scale.
- Constantly evaluate the system. Evaluation is the control mechanism for the iterative design process and the means to control cost and effort of development. Evaluation is performed at the end of each cycle by the user and builder.
- The steps of SDLC are repeated in each cycle, allowing for refinement, expansion, and modification of the system.

These actions are repeated until a specific DSS is developed. Due to the constant reevaluation and subsequent refinement and expansion of the system, it is essential that all participants cooperate and interact throughout each successive cycle. The Iterative Approach is often called prototyping or evolutionary. This can lead to confusion since Iterative, Prototyping and Evolutionary are themselves methodologies which utilize the Iterative Approach. The Iterative Approach provides a short development time, short user reaction time, and improved user understanding (Turban, 1990). The following is an overview of DSS methodologies using the Iterative Approach:

1. Iterative

Sprague (1980) states the Iterative methodology is also known as "breadboarding" or "middle out". This methodology takes - analysis, design, construction, and implementation - from the Traditional Approach and combines them into one "iterative" step. The Iterative methodology resembles the Iterative Approach previously discussed. The user and builder identify a small portion of the problem, create a small working system, and refine it through several successive runs (usually three to six). This is continued until a relatively stable system is reached. Under the Iterative Methodology, the system is never complete, but constantly altered and modified by the changing environment. This approach requires a high level of management participation in its design and a conscious strategy on the part of the user and builder to fully utilize the flexibility of this method. The Iterative methodology differs from prototyping since the working system is not a "test pilot", but an actual continuously running system. A test may be performed, removed and examined while the original system continues to run. With the Iterative method, the system is actually running and evolving throughout each cycle.

2. Prototyping

Ginzberg and Ariav (1986) note that Prototyping is traditionally associated with DSS and still thought by many researchers to be an "ideal"

strategy. The terms Prototyping and DSS are synonymous and describe a variety of essentially similar approaches: middle-out design, evolutive approach, and adaptive design. The Linguistic Approach is a more rigorous prescription for the Prototyping approach to DSS design (Keen, 1983). A "good" (knowledgeable) user is first identified. This person must show interest, curiosity, and the initiative to resolve the decision problem. The focus is on the dialogue of the user. What the user says and sees becomes the conceptual model. Relevant verbs, which the user employs in the decision making process, are identified and then translated into systems commands. "The emphasis is on assuring that the system will correspond to the users' perception of the process (Ginzberg and Ariav, 1986, p.50)." Ginzberg and Ariav (1986) list four essential features of all prototyping methods:

- focus on a small piece of the decision problem
- focus on the dialogue and user interface
- follow closely the user's existing decision process when forming the initial process
- evolve and modify the system as changes demand.

Turban (1990) calls Prototyping a process of building a "quick and dirty" version of information systems. Two kinds of prototyping are recognized: throwaway and evolutionary.

The throwaway concept focuses on better understanding the system's performance and user's requirements. A test pilot is developed and when complete, a preliminary design constructed. The test pilot is then thrown away and the final system completed from the preliminary design. The evolutionary approach iteratively refines a minisystem over a long trial period with the following steps:

- Quickly identify user's information and operating requirements.
- Develop a working prototype.
- Test and evaluate the prototype.
- Modify and improve the system as necessary.

The last two steps are repeated numerous times to refine the system, thus resembling the Iterative Approach. Turban (1990) further identifies five distinct features of prototyping:

- By designing systems in an iterative fashion, learning is explicitly integrated into the design process. The prototype is evaluated and errors corrected in the next prototype.
- A prerequisite for effective learning is timely feedback. Prototyping allows for short intervals between tests, ensuring quick feedback.
- The user is involved in the development of each prototype test ensuring expertise in the design effort.
- The initial prototype is low cost. Each subsequent test may require additional funds as changes occur.

- By passing the SDLC stages, requirements evolve as experience is gained.

Often, a user may not be able to initially express a situation. Prototyping not only involves the user, but provides the user a means to visualize a problem and express his knowledge through test samples. Like the Prototyping method, the Evolutionary method also emphasizes learning, user involvement, and the ability to adapt to lessons through each iteration. The Evolutionary method allows the user to build descriptive models of decision making behavior and compare them to a normative model (Alavi and Henderson, 1981).

3. Adaptive

The Adaptive method combines the requirements analysis, design, development, and implementation into a single phase which is repeated in successive iterations (Sprague, 1980). Keen (1980) proposed a framework, for the Adaptive methodology, which includes participants (the builder and the user) with the technical system (the DSS). Through interaction, three adaptive links are established: (a) the user-system, (b) the user-builder, and (c) the builder-system. The user takes the action and is responsible for the consequences, but may not directly interact with the technical system. In such cases an intermediary interfaces between the user and the system. The builder creates the specific DSS,

used by the user or intermediary, and is knowledgeable of the technology and capability of the DSS.

The user-system link reflects the effects of the user's characteristics on the system. The user learns through the interaction with the system, enhancing their understanding and perception of the decision problem and possible solutions. The builder-system link is the means by which the builder adds new capabilities and functions to the system. It is important that the builder make changes based on input from the user, ensuring the system evolves and changes with the decision environment. The user-builder link is based on communication. The builder learns the requirements expected in the system and the user learns of the capabilities of the decision support system (Alavi and Napier, 1984).

Keen (1980) states the Adaptive method develops a DSS through an adaptive process of learning and evolution. An initial system is developed, ". . . the 'final' system must emerge through an adaptive process of design and usage (Keen, 1980, p.15)." Examples of the Adaptive method include the Task Representation by Hackathorn and Fetter (1981) and the Task Analysis method discussed by Bahl and Hunt (1984). The Adaptive process is effective in a number of situations:

- The designer or user cannot provide specific functional specifications. If a decision problem is unstructured or semi-structured and the requirements are

not known or cannot easily be expressed, then an initial system can be adapted and modified through a process of learning and evolution.

- The users do not know what they want and designers do not understand what is required. An initial system is developed to provide something concrete with which to work.
- The users concept of the decision task is enhanced by the DSS. The system is enhanced by new uses and functions as the users learn and develop new insights.
- Intended users are allowed personalized usage and computer support must be flexible.

A specific DSS provides a manager or user with flexibility to experiment with information requirements. Over time, as tasks or environment change, the specific DSS must adapt through reconfiguration of the DSS generator and evolution of tools into the final DSS (Sprague, 1980).

C. KNOWLEDGE ANALYSIS AND ENGINEERING

Knowledge engineering is the collaboration between human experts in a particular environment and knowledge engineers to collect the knowledge (or reasoning procedures) of the human expert and code this knowledge for others to use. As Turbin (1990, p. 454) writes, knowledge engineering is:

the art of bringing the principles and tools of AI research to bear on difficult applications problems requiring experts' knowledge for their solutions. The technical issues of acquiring this knowledge, representing it, and using it appropriately to construct and explain lines-of-reasoning are important problems in the design of knowledge-based systems. The art of constructing intelligent agents is both part of and an extension of the programming art.

It is the art of building complex computer programs that represent and reason with knowledge of the world.

Fox (1983) refers to knowledge engineering as knowledge representation research which ". . . is concerned with the identification, representation, and utilization of knowledge in problem solving. DSS methodologies which support management's decisions through the gathering of human expertise from a system fall into the category of Knowledge Analysis and Engineering. Using a Knowledge Analysis and Engineering approach, an expert support system is developed through the collection of rules of reason (knowledge) from experts and the transfer of this knowledge to a knowledge base or inference engine. Modular programs are constructed from the cooperation between a knowledge engineer and expert. Through discussion, procedures within a module are refined, and additions and deletions easily made. Knowledge engineering includes the following activities:

- Knowledge Acquisition (KA) - acquiring knowledge from human experts, books, documents, sensors, or computer files. This knowledge may be specific to a particular problem or general knowledge to a business.**
- Knowledge Representation (KR) - coding of the gathered knowledge for future inference.**
- Inference - designing software which allows inference to be made.**
- Explanation and Justification - designing a explanation capability to ask questions of "why" or "how". (Turban, 1990)**

The Knowledge Analysis and Engineering category is suitable for structured or unstructured problems. Construction of an expert system aids the human expert in expressing his/her knowledge. Often an expert has no clue how he/she accomplishes some tasks. As organizations expand their area of concern to "problem finding" within the entire organization, an approach to DSS development using a Knowledge Analysis approach will provide (a) management an ability to examine how or why a system functions and (b) an aid in strategic long-term planning.

D. REPRESENTATIONS, OPERATIONS, MEMORY AIDS, CONTROL (ROMC)

Developed by Sprague and Carlson (1982), ROMC is a framework for DSS systems requirement analysis whose objective is to identify the characteristics and capabilities that a specific DSS requires. Since most DSS are composed of unstructured information requirements, ROMC provides four user-oriented entities to overcome this difficulty (Turban, 1990):

- Representations provide the user the ability to conceptualize and communicate the problem, and interpret output or invoke operations.
- Operations provide the ability to analyze and manipulate the representations.
- Memory aids (fundamental learning aids) are used to link representations and operations.

- Control Mechanisms form a framework for integrating representations, operations, and memory aids into a useful decision-making system.

The ROMC approach is effective when decision makers have difficulties describing situations, but phases of intelligence, design, and choice apply to the DSS analysis. Memory aids such as reports, "split screen" displays, data files, indexes, mental rules, and analogies should be provided by a DSS. Since decision makers differ in style, skills, and knowledge, the DSS should help develop these characteristics (Turban, 1990). Examples of methodologies using the ROMC analysis are the Rep Test Methodology by Grudnitski (1981) and (1984), the Checkland's Systemic Method by Landry, Pascot, Birolat (1985), and the Representation-based approach by Carlson (1979).

The ROMC approach ". . . oscillates between design and analysis, with no clear delineations between the two (Ginzberg and Ariav, 1986, p.50)." Representations, operations, memory aids, and control define the elements of a check-list. A consistency check is provided by the relationships between the four elements. ROMC is a precursor to design and construction, and is user oriented. Showing little orientation to hardware or software resources, ROMC ". . . defines the scope of the decision process to be supported, but does not define the actual capabilities of the eventual system (Ginzberg and Ariav, 1986, p.51)." The

ROMC approach is of little assistance in translating functional requirements during DSS construction.

From the diverse DSS approaches in today's business and information technology markets, three broad techniques were examined as initial categories for all methodologies: (a) Traditional, (b) Iterative, and (c) Knowledge Analysis and Engineering. The Representations, Operations, Memory Aids, Control (ROMC) can be used by any category of DSS to identify required characteristics and capabilities through requirements analysis. The methods explained in Chapter II have similarities and differences which are examined in Chapter III. Chapter III will propose a broader approach for choosing a DSS methodology by considering construction approaches, technology levels, participants, structure, environmental factors, and DSS categories.

IV. SELECTING A DESIGN METHODOLOGY: A BROAD APPROACH

A focus of DSS research over the last decade has been the concept that a single methodology could be constructed which would encompass the development/design of all DSS. Yet no single methodology for the development of DSS has gained acceptance as the methodology. As shown in previous chapters there is no agreement on the characteristics and capabilities of DSS. Though design of DSS has been extensively studied and numerous methods for the analysis and design proposed over the past decade, more credence is given to the idea that DSS development should be based on the salient characteristics of a decision situation and the environment in which it exists. A broad approach, recognizing different aspects of decision situations (construction approach, technology levels, participants, structure, and environmental factors) and based on three DSS techniques (Traditional, Iterative, Knowledge Analysis and Engineering) as underlying categories for all methodologies is proposed for the selection of a design methodology.

When selecting a DSS design methodology, the user is confronted with an assortment of possibilities and the difficult task of identifying which method is most suited to the salient characteristics of the decision situation and the

environment in which it exists. A decade ago, DSS were basically constructed using the Traditional Approach using SDLC. At that time, computer technology and DSS development were time restrictive, not user friendly, and most suited for structured decision-making requiring little flexibility. Today's DSS have evolved to include flexibility, adaptability, and quick response to an ever changing decision environment. But the problem of identifying the most appropriate DSS methodology for a given situation has not been simplified by new technology or the plethora of DSS generators currently on the market.

Figure 1 recognizes seven areas which define different decision situations including approaches to construction, technology levels, participants, structure, environmental factors, and DSS categories. Each element within an area of the broad approach must be identified based on the environment and salient characteristics of the decision at hand.

A. CONSTRUCTION APPROACHES

Sprague and Carlson (1982) and Turban (1990) classify the numerous approaches to DSS construction into three categories: (a) quick-hit, (b) staged development, and (c) development of a complete DSS. Of the three approaches to DSS construction (quick-hit, staged development, complete DSS), Turban (1990, p. 168) states, "The approach to be selected will depend on the specific

APPROACHES TO CONSTRUCTION	(1) Quick-hit (2) Staged Development (3) Complete Development		
TECHNOLOGY LEVELS	(1) Specific DSS (2) DSS Generator (3) DSS Tools		
PARTICIPANTS	(1) Manager/User (2) Intermediary (3) Builder (4) Technical Support Person (5) Toolsmith		
STRUCTURE	(1) Structured (2) Unstructured		
ENVIRONMENTAL FACTORS	(1) Solicited/Unsolicited (2) Recurring/Non-recurring (3) Critical/Non-critical (4) Short Term/Long Term (5) Team Develop/User Develop		
DSS REQUIREMENTS ANALYSIS	ROMC Approach		
	(1) Representative (2) Operations (3) Memory Aids (4) Control		
DSS METHODOLOGY	Traditional Life Cycle	Iterative Approach	Knowledge Analysis and Engineering
	System Development Life Cycle (SDLC)	(1) Iterative Methodology (2) Prototyping Methodology (3) Evolutionary Methodology (4) Adaptive Methodology	Expert Support System

Figure 1. A Broad Approach

situation (i.e., the organization, purpose of the DSS, users, tasks, available tools, and builder)."

1. Quick-Hit

Many small DSS are constructed as a specific DSS if there is a recognized need, a high potential payoff, or an existing difficult problem. Using available generators, the quick-hit DSS is constructed relatively quickly, maintains low costs, with low risks, and utilizes the latest technology. Commercially available generators are a major advantage to the quick-hit approach since software updating and maintenance is done by software vendors, not the user's organization (Turban, 1990). A quick-hit approach is usually constructed for one individual or one purpose and seldom relates to other DSS. Since little experience from a quick-hit DSS can be carried over to the next system, the same procedures or steps must be repeated each time a DSS is developed using a quick-hit approach. Turban (1990) further states a quick-hit approach is appropriate when the decision situation has:

- Clear-cut goals - The information requirements is known from the beginning and not require any research.
- Clear-cut procedure - The DSS is constructed from existing technology (generators and tools), either off-the-shelf or existing DSS. Again, no research is necessary as procedures and calculations are well understood.
- Available data - All data are readily available.

- Fewer users - Crossing no organizational boundaries, the DSS requires little selling or training. The DSS is most effective for one or a few users with equivalent experience and goals.
- Independent system - Other than input data prepared by an outside system, the DSS operates independently.

Because existing off-the-shelf technology is used, it is easier to apply new advances in technology to the quick-hit DSS. Time and effort is minimized since the DSS is developed around existing technology and few new procedures are required. However, existing technology may also be a disadvantage, since the off-the-shelf technology is not created for a specific situation and any changes may require difficult modifications (Sprague and Carlson, 1982).

2. Staged Development

The Staged Development is an iterative construction approach which allows changes in direction and the ability to incorporate new technology as it becomes available. It is similar to the Quick-hit approach in the success and visibility which is obtained, yet unlike the Quick-hit approach in that prior planning is required to develop an initial system which may be used in future DSS. Though time consuming in development, the Staged Development approach constructs a specific DSS and often leads to the development of an in-house generator. Both the specific DSS and the generator can lead to development of future specific DSS. Prior planning requires some ". . . up-front development

costs and some delay in completion of the first specific DSS."(Sprague and Carlson, 1982, p. 61) This construction approach is appropriate for unstructured or semistructured decision situations.

3. Complete DSS

The Complete DSS most effectively utilizes the available architecture. "This approach requires the development of a full-service, large-scale, DSS generator; large-scale, specific DSS; and an organization unit to manage such a project."(Turban, 1990, p. 168) Composed of a generator and several specific DSS, the GADS system by IBM is an example of a complete DSS. Disadvantages of the complete DSS lie in its complexity. Several years may be necessary to develop a large scale process which integrates basic tools and generators. An inherent delay in development may risk technological obsolescence upon project completion, besides a long delay in ultimate project success. The very complexity and long development time also may lead to a "high risk of pitfalls".(Sprague and Carlson, 1982, p. 62) However, once the system is completely developed, it will reach full strength faster than the other approaches, develop the most efficient generator, and best integrate basic tools.

Organizations may use a combination of approaches - a large complete DSS for company-wide use and many quick-hit DSS for smaller unrelated jobs. With the advancement in commercial DSS generators and increased capabilities of

microcomputers, the quick-hit approach is likely to become the most frequently used. (Turban, 1990)

B. TECHNOLOGY LEVELS

The technology levels, including DSS tools, DSS generators, and specific DSS are reflected in level two of Figure 1. Current advances within software development have greatly increased the availability of DSS tools. Graphics, query systems, random number generators, spreadsheets and programming languages are more sophisticated and accessible to the general public. Powerful computer processors and compatible hardware are affordable to increasing numbers of people. Off-the-shelf software and larger more powerful personal computers increase the chance that a DSS tool of one kind or another is utilized in a DSS generator or specific DSS. Tools are advantageous for providing increased speed in information manipulation and visual expression. A query system allows a user to access data in the most advantageous format for a given situation.

The DSS generator is a composition of hardware and software which provides quick and easy capability to build a specific DSS. Though tools are not required in a DSS generator, they are becoming more frequent as the technology of tools improves. DSS generators (as with DSS tools) are becoming more commercially available. Commercial availability of DSS generators is advantageous to most DSS construction since the vendor, rather than the DSS

developer, is tasked with development time and cost. This is especially advantageous to the quick-hit construction approach to DSS. In this construction approach, research for generator construction should not be necessary because procedures and calculations are well understood. However, off-the-shelf generators may be disadvantageous since they are not constructed specifically for the characteristics of a given situation. Because a staged development construction is iterative in nature, a generator is normally created in-house and modified in stages. A complete DSS, requiring a full-service, large-scale DSS generator, necessitates a larger, more complex generator developed specifically for the salient characteristics involved.

Each construction approach contains a specific DSS, yet requires a very different technological approach to DSS construction. The quick-hit approach is likely to contain off-the-shelf tools and generators for their ease, availability and low cost. The staged development approach contains in-house generators designed for a specific situation. This allows for flexibility and future modification, yet may prevent use of the DSS in new and different situations. Finally, the complete DSS approach requires a full-service, large-scale DSS generator and large-scale specific DSS. The DSS is developed in-house over a long period of time (possibly several years) and is created for a specific project.

C. PARTICIPANTS

The participants, either an individual or group, must choose the appropriate technology level and approach to DSS construction based on the development and usage environment. Reflected in level three of Figure 1, the five types of participants are (a) manager or user, (b) intermediary, (c) builder, (d) technical support person, and (e) toolsmith. Since one individual may fill several roles or several individuals may fill one role, any number of combinations may evolve.

As the decision maker, the manager or user solves the problem, takes the action, and assumes the consequences. The manager identifies the problem, decides on the construction approach and its level of technology. He or she decides on the environmental factors, the specific audience, and whether the DSS is constructed for various managerial levels or for a single user. Having identified the problem, the audience (user of the DSS), environmental factors, construction approach, and requirements analysis, the manager then chooses an appropriate DSS category (Traditional (Life Cycle), Iterative, Knowledge Analysis and Engineering), and finally the DSS methodology (i.e. Prototyping, Adaptive, etc.).

The intermediary, perhaps a staff assistant or analyst, assists the manager in typing, user interaction, or merely making suggestions. A decade ago, the intermediary was necessitated by the cumbersomeness of developing a DSS with technology of that era. With today's technology, the intermediary's assistance is

advantageous in the staged development or complete DSS construction approach due to the complexity of the approach and involvement of more personnel in development. Since a quick-hit construction approach is appropriate for a DSS with few users, it is likely an intermediary will not be necessary. As the DSS becomes more complex (staged development or complete DSS), the services of the intermediary become more valuable to the manager and the project at hand.

The builder or facilitator makes the technical decisions about the DSS - which tools and generators are to be used. If a quick-hit approach to DSS construction is used, the positions of builder and manager may be held by the same individual. If a staged development or complete DSS is constructed, the position of builder may again be held by the manager, but likely will be filled separately. Typically the DSS builder is a member of an information center. In either case, the builder must be knowledgeable of current technology (including DSS tools, generators, software and hardware) and the problem the DSS is to address.

The technical support person provides support to the DSS builder through his or her knowledge of databases, analysis models, and data display formats. The toolsmith improves the DSS package through better hardware and software, increases efficiency and effectiveness through new languages and technology, and improves the overall architecture with the use of better tools. In smaller DSS, the position of technical support person and toolsmith would likely be performed by

a single individual. However, in a larger DSS (using the staged development or complete DSS approach to construction) these positions are filled as part of a larger organization responsible for control of the project.

The participants are significant in selecting a design methodology because they are both (a) part of the problem the DSS seeks to address and (b) the individual or group identifying the environmental factors, choosing the construction approach, and ultimately developing the DSS. The participants identify the composition (structured, semistructured, or unstructured) of the decision addressed by the DSS. A structured decision normally has only one user and possesses relatively few parameters or possible conclusions. The semistructured or unstructured decision requires support for managers at all levels (individuals or groups) and has multiple parameters and possible conclusions. A quick hit approach to construction of a specific DSS is best suited for few participants in a more structured situation, while a staged or complete development construction, using DSS tools and generators, is advantageous to a larger group and more complex, semistructured or unstructured problem.

D. ENVIRONMENTAL FACTORS

The environmental factors are identified in Chapter II as: (a) solicited/unsolicited, (b) recurring/non-recurring, (c) critical/non-critical, (d) short term/long term, and (e) team develop/user develop. Reflected in level five of

Figure 1, each factor will effect the methodology used in the development of a DSS and must be examined individually.

A solicited problem indicates management is cognizant and supports the development and implementation of a DSS. This does not indicate whether a problem is specific/non-specific, or simple/complex in nature. If the problem is unsolicited, management is not aware of the problem and must be convinced of the need for a DSS. In this situation, management must be convinced regardless of the technology level, structure, or construction approach.

The decision problem is next determined to be recurring or non-recurring. A quick-hit construction approach is best suited to a problem which seldom, if ever, recurs. The staged development or complete DSS approach is advantageous for a complex recurring problem covering a much wider set of circumstances.

Next, the decision is determined to be critical or non-critical. If the decision is critical, it will have numerous repercussions and urgency to the organization. A non-critical decision will have fewer repercussions and little urgency. After determining the decision to be critical/non-critical, it must then be determined as either short-term or long-term in duration, and finally, with the participants identified, a decision is made whether the DSS will be team developed or user developed. The quick-hit construction approach is best suited for user

development, and the staged development or complete DSS is usually team developed due to size and complexity of the DSS.

E. DECISION SUPPORT SYSTEM CATEGORIES

After determining a construction approach, technology levels, participants, structure, and environmental factors, an appropriate DSS category (Traditional (Life Cycle), Iterative, Knowledge Analysis and Engineering) and a DSS methodology (i.e. Prototyping, Adaptive, etc.) remain to be determined. A requirements analysis (ROMC - level six of Figure 1) can be performed on any DSS category to identify characteristics and capabilities required by the DSS. The DSS categories and methodologies are reflected in level seven of Figure 1. As discussed in Chapter III, current literature reveals a large diversity in DSS methodologies. Methodologies which fall in the Traditional (Life Cycle) Approach category are typically used in situations where information is reasonably well defined with detailed specifications. With the System Development Life Cycle (SDLC), these methodologies involve very structured situations, in which information requirements are determined by logical analysis. The Traditional Approach methodologies are best suited for large structured applications and are unsuitable for small applications or large unstructured situations.

Methodologies which fall in the Iterative Approach category combine four major phases of the Traditional SDLC (analysis, design, construction, and

implementation) into a single step. Repeating this step, these methodologies combine traditional data access and retrieval functions with models and analytic techniques. They allow for quick and easy changes through the use of tools and DSS generators.

Methodologies which fall in the Knowledge Analysis and Engineering category develop expert support systems through the collection of rules of reason (knowledge) from experts and transfers this knowledge to a knowledge base. They provide management an ability to examine how or why a system functions and can aid in strategic long term planning.

The Representations, Operations, Memory Aids, Control (ROMC) identifies the characteristics and capabilities that a specific DSS requires. ROMC provides four user-oriented entities to overcome the unstructured nature of most DSS. This approach is especially effective in situations where the user cannot clearly describe situations. User oriented and a precursor to design and construction, the ROMC defines the scope of the decision process to be supported.

After determining the DSS category, the specific methodology is decided upon. This leads to confusion since Iterative, Prototyping and Evolutionary are themselves methodologies which utilize the Iterative Approach. The Iterative methodology identifies a small portion of the problem, creates a small working system, and refines it through several successive runs. Iteration is continued until

a stable system is reached. The system is never complete, but continually modified and improved. Prototyping methodology is composed of two types: throwaway or evolutionary. The throwaway approach develops a test pilot which leads to a preliminary design. The test pilot is then thrown away and the final system completed from the preliminary design. The evolutionary approach iteratively refines a minisystem over a long trial period until a final system is reached. The Adaptive methodology develops a DSS through an adaptive process of learning and evolution. The user learns through the interaction with the system and the builder makes changes based on the input from the user. As the builder learns the requirements of the system, the user learns the capabilities of the Decision Support System. This procedure is repeated and adapted over time until a final system is completed.

V. CONCLUSION

Since the appearance of Decision Support Systems (DSS) more than a decade ago, DSS have evolved considerably from a "new approach" in the evolution of information technology. From the inception of DSS, researchers have focused on identifying the methodology based on the premise that there is or should be a single methodology for all DSS. DSS flourish as technology has improved computer memory, processing power, and the availability and use of micro-computers. To date, no single methodology for the development of DSS has gained acceptance as the methodology. This study investigates the premise of a single methodology as it relates to different decision features.

Existing methodologies and frameworks for the analysis and design of DSS were studied as background material for this thesis. The research confirmed there is no single accepted definition or set of characteristics for all DSS; decision features vary with every situation; and DSS development should be based on the salient characteristics of a decision situation and the environment in which it exists. This thesis proposes a broad approach to DSS development, recognizing different decision situations have different requirements. An individual should be able to select the appropriate methodology based on known characteristics of DSS as well

as its development and usage environment. Each level, and subsequent elements, of the broad approach is reviewed as it pertains to the environment in which a decision situation exists.

The plethora of DSS currently on the market make selection of a methodology for a given situation difficult. When selecting a DSS construction methodology, the user is confronted with an assortment of diverse DSS designs from which to choose and the difficult task of identifying which method is most suited to the salient characteristics of the decision situation and the environment in which it exists. Complicating the selection is the lack of a single accepted definition and set of characteristics for a DSS. Though called by many names, methodologies and frameworks fall into three broad categories. For example, the Iterative, Prototyping, Evolutionary and Adaptive methodologies use the same basic iterating pattern for DSS development, yet each is slightly different and applicable in different situations. The use of the broad approach simplifies the choice of DSS construction methodology by identifying the salient characteristics and environment of the decision situation.

The field of Decision Support Systems is made complicated by the lack of consensus on DSS definition, characteristics, overlapping terminology (i.e. methodology, approach, process, technique), and methodology names (i.e. Iterative, Prototyping, Evolutionary, Adaptive). The use and meaning of terms or

names are used in different context by different researchers. For example, the evolutionary type of prototyping is not to be confused with the evolutionary methodology, even though the both use an iterative approach. The broad approach is proposed, with its three categories, as an attempt to clarify the existing ambiguity and confusion through recognizing and defining seven areas of concern in the selection of a DSS methodology.

LIST OF REFERENCES

Alavi, Maryam and Napier, H. Albert, "An Experiment in Applying the Adaptive Design Approach to DSS Development," *Information and Management*, v. 7 (1984).

Ariav, Gad and Ramesh, Balasubramaniam, "Methodologies for DSS Analysis and Design: A Survey," paper for The Leonard N. Stern School of Business, New York University, June 1989.

Bahl, Harish C. and Hunt, Raymond G., "A Framework for Systems Analysis for Decision Support Systems," *Information and Management*, v. 7, 1984.

Carlson, Eric D., "An Approach for Designing Decision Support Systems," *Database*, Winter 1979.

Courbon, J. C., Grajew, J., and Tolovi, J., Jr., "Design and Implementation of Decision Support Systems by an Evolution Approach." Unpublished working paper, 1980.

Elam, Joyce J. and Henderson, John C., "Knowledge Engineering Concepts for Decision Support System Design and Implementation," *Information and Management*, v. 6, 1983.

Fox, J., "Formal and Knowledge-Based Methods in Decision Technology" - In: Proceeding of the 9th Conference of Subjective Probability, Utility and Decision Making," Groningen, 1983.

Ginzberg, Michael J. and Ariav, Gad, Methodologies for DSS Analysis and Design: A Contingency Approach to Their Application," Proceedings of the Seventh International Conference on Information Systems, San Diego, California, December 15-17, 1986.

Grudnitski, Gary, "Eliciting Decision-Makers' Information Requirements: Application of the Rep Test Methodology," *Journal of Management Information Systems*, v. I, n. 1, Summer 1984.

Grudnitski, Gary, "A Methodology for Eliciting Information Relevant to Decision Makers," Proceedings of the Second International Conference on Information Systems, Cambridge, Mass., 1981.

Hackathorn, R. D. and Fetter, R. A., "Toward A Formal Definition of Task Representation," Proceedings of the Second International Conference on Information Systems, Cambridge, Mass., 7-9 December 1981.

Keen, Peter G. W., "Adaptive Design for Decision Support Systems," *Data Base*, v. 12, n. 1, Fall 1980.

Landry, M., Pascot, D. and Birolat, D., "Can DSS Evolve Without Changing Our View of the Concept of 'Problem'?" *Decision Support Systems*, v. 1, 1985.

Locander, William B., Napier, Albert H. and Scamell, Richard W., "A Team Approach to Managing the Development of a Decision Support System," *MIS Quarterly*, March 1979.

Mahmood, Mo A., Courtney, James F. and Burns, James R., "Environmental Factors Affecting Decision Support System Design," *Data Base*, Summer 1983.

Meador, C. L. and Rosenfeld, W. L., "Decision Support Planning and Analysis: The Problems of Getting Large-Scale DSS Started," *MIS Quarterly*, June 1986.

Sprague, Ralph H., Jr., "A Framework for the Development of Decision Support Systems," *MIS Quarterly*, December 1980.

Sprague, R. H., Jr., and Carlson, E. D., *Building Effective Decision Support System*, Prentice-Hall, 1982.

Turban, Efraim, *Decision Support and Expert Systems: Management Support Systems*, 2d ed., Macmillan, 1990.

BIBLIOGRAPHY

Ahn, Taesik and Grudnitski, Gary, "Conceptual Perspectives on Key Factors in DSS Development: A Systems Approach," *Journal of Management Information Systems*, Summer 1985, v. II, n. 1.

Alavi, Maryam and Henderson, John C., "An Evolutionary Strategy for Implementing A Decision Support System," *Management Science*, v. 27, n. 11, November 1981.

Andriole, Stephen J., "The Design of Microcomputer-Based Personal Decision-Aiding Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-12, n. 4, July/August 1982.

Bahl, Harish C. and Hunt, Raymond G., "Decision-Making Theory and DSS Design," *Data Base*, Summer 1984.

Bahl, Harish C. and Hunt, Raymond G., "Problem-solving Strategies for DSS Design," *Information and Management*, v. 8, 1985.

Belardo, Salvatore, Karwan, Kirk R., and Wallace, William, "An Investigation of System Design Considerations for Emergency Management Decision Support," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14, n. 6, November/December 1984.

Belardo, Salvatore, Karwan, Kirk R., and Wallace, William, "DSS Component Design Through Field Experimentation: An Application to Emergency Management," Proceedings of the Third International Conference on Information Systems, Ann Arbor, Michigan, December 13-15, 1982.

Belardo, Salvatore and Pazer, Harold L., "Scope/Complexity: A Framework for the Classification and Analysis of Information-Decision Systems," *Journal of Management Information Systems*, v. II, n. 2, Fall 1985.

Ben-Bassat, Moshe and Freedy, Amos, "Knowledge Requirements and Management in Expert Decision Support Systems for (Military) Situation Assessment," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-12, n. 4, July/August 1982.

Bonczek, Robert H., Holsapple, Clyde W., and Whinston, Andrew B., "Future Directions for Developing Decision Support Systems," *Decision Sciences*, v. 11, 1980.

Buede, Dennis M., Yates, Gerald and Weaver, Carl A., "Concept Design of a Program Manager's Decision Support System," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15, n. 4, July/August 1985.

Cooper, Randolph B., "Identifying Appropriate MIS/DSS Support: A Cost Analysis Approach," Proceedings of the Sixth International Conference on Information Systems, Indianapolis, Indiana, December 16-18, 1985.

Davis, Richard K., "Strategic, Tactical, and Operational Planning and Budgeting: A Study of Decision Support System Evolution," *MIS Quarterly*, December 1979.

Elam, Joyce J. et al, "A Vision for Decision Support Systems," (A paper based on the outcome of a number of formal and informal meetings among researchers.).

Fuerst, William L. and Martin, Merle P., Effective Design and Use of Computer Decision Models," *MIS Quarterly*, March 1984.

Garnto, Carleen and Watson, Hugh J., "An Investigation of Database Requirements for Institutional and Ad Hoc DSS," *Data Base*, Summer 1985.

Hackathorn, Richard D. and Karimi, Jahangir, "A Framework for Comparing Information Engineering Methods," *MIS Quarterly*, June 1988.

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

Hogue, Jack T. and Watson, Hugh J., "Current Practices in the Development of Decision Support Systems," Proceedings of the Fifth International Conference on Information Systems, Tucson, Arizona, 28-30 November 1984.

Huber, George P., "The Nature of Organizational Decision Making and the Design of Decision Support Systems," *MIS Quarterly*, June 1981.

King, William R. and Rodriguez, Jamie I., "Participative Design of Strategic Decision Support Systems: An Empirical Assessment," *Management Science*, v. 27, n. 6, June 1981.

Klein, H. K. and Hirschheim, R., "Fundamental Issues of Decision Support Systems: A Consequentialist Perspective," *Decision Support Systems*, v. 1, 1985.

Lehner, P. E., Probus, M. A., and Donnell, M. L., "Building Decision Aids: Exploiting the Synergy Between Decision Analysis and Artificial Intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, v. SMC-15, n. 4, July/August 1985.

Liang, T. P., "Critical Success Factors of Decision Support Systems: An Experimental Study," *Data Base*, Winter 1986.

Mann, Robert I. and Watson, Hugh J., "A Contingency Model for User Involvement in DSS Development," *MIS Quarterly*, March 1984.

Moore, Jeffery H. and Chang, Michael G., "Design of Decision Support Systems," *Data Base*, v. 12, n. 1,2, Fall 1980.

Orman, Levent, "A Multilevel Design Architecture for Decision Support Systems," *Data Base*, Spring 1984.

Ramaprasad, Arkalgud, "Cognitive Process as a Basis for MIS and DSS Design," *Management Science*, v. 33, n. 2, February 1987.

Remus, W. and Kottemann, J. E., "Semi-Structured Recurring Decisions: An Experimental Study of Decision Making Models and Some Suggestions for DSS," *MIS Quarterly*, June 1987.

Sakthivel, S. and Tanniru, Mohan R., "Information System Verification and Validation during Requirement Analysis Using Petri Nets," *Journal of Management Information Systems*, v. 5, n. 3, Winter 1988-89.

Sen, Arun and Biswas, Gautam, "Decision Support Systems: An Expert Systems Approach," *Decision Support Systems*, v. 1 1985.

Szewczak, E. J., "Two Approaches to Strategic Database Design," *Data Base*, Winter 1983.

Wang, Michael Szu-Yuan and Courtney, James F. Jr, "A Conceptual Architecture for Generalized Decision Support System Software," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-14, n. 5, September/October 1984.

Watkins, Paul R., "A Measurement Approach to Cognitive Complexity and Perception of Information: Implications for information Systems Design," Proceedings of the Second International Conference on Information Systems, Cambridge, Mass., 7-9 December 1981.

• Watkins, Paul R., "Perceived Information Structure: Implications for Decision Support System Design," *Decision Sciences*, v. 13, 1982.

• Watkins, Paul R., "Preference Mapping of Perceived Information Structure: Implications for Decision Support Systems Design," *Decision Sciences*, v. 15, 1984.

INITIAL DISTRIBUTION LIST

- | | | |
|----|---|---|
| 1. | Defense Technical Information Center
Cameron Station
Alexandria, VA 22304-6145 | 2 |
| 2. | Library, Code 52
Naval Postgraduate School
Monterey, CA 93943-5002 | 2 |
| 3. | Professor B. Ramesh
Naval Postgraduate School
Code AS/RA
Monterey, CA 93943-5002 | 1 |
| 4. | Professor Tung Bui
Naval Postgraduate School
Code AS/BD
Monterey, CA 93943-5002 | 1 |