Jurnal Pengurusan 3, Julai 1984, 3-23

THE CONCEPT OF AND DESIGN STRATEGY FOR DECISION SUPPORT SYSTEMS

Abdul Barri Md. Mokhtar
Universiti Kebangsaan Malaysia

SYNOPSIS

Decision Support Systems (DSS) have become an increasingly important type of computer-based information systems. Unlike Electronic Data Processing (EDP) or Management Information Systems (MIS), DSS is specifically oriented towards the kind of information processing activity that constitute a decision process. It serves to increase the effectiveness of the managers in making decisions pertaining to the management of an organisation. This article begins by examining the concepts of DSS, focussing on the various definitions given in the literature so that a unified view of what DSS really is and how it differes from other computer-based information systems, can be formulated. The paper then discusses briefly the hardware and software requirements of DSS. Following this, a generic framework for designing DSS is presented. Finally, the strategies for designing DSS are proposed and discussed. The paper argues that the appropriate strategy depends on the objective and decision situation a particular DSS is meant to support.

SINOPSIS

Sistem Sokongan Keputusan (SSK) telah mula diterima umum sebagai salah satu daripada sistem maklumat berkomputer yang terpenting di dalam organisasi perniagaan. Ianya berbeza daripada Sistem Pemprosesan Data Elektronik atau Sistem Maklumat Pengurusan kerana orientasi SSK adalah khusus kepada aktibiti pemprosesan maklumat bagi maksud membuat keputusan. Ianya bertujuan untuk meningkatkan lagi keberkesanan seseorang pengurus di dalam membuat keputusan-keputusan pengurusan organisasi perniagaan. Artikel ini bermula dengan meneliti konsep SSK dengan merujuk kepada definasi-definasi yang telah diberi oleh penyelidik-penyelidik di dalam bidang ini supaya suatu pandangan umum dapat dirumuskan mengenai apakah sebenarnya SSK dan bagaimanakah ianya berbeza jika dibandingkan dengan sistem-sistem maklumat berkomputer yang lain. Kemudian, satu rangkakerja umum bagi mereka-bentuk SSK dibincangkan. Artikel ini disudahi dengan membincangkan strategi reka bentuk yang sesuai dan releven bagi sesuatu SSK. Penulis berpendapat bahawa sesuatu strategi reka bentuk yang releven adalah bergantung kepada keadaan dan matlamat sistem berkenaan.

INTRODUCTION

The early 1970's witnessed the emergence of a new concept in the information field called decision support systems (DSS). Numerous reactions have been given to it ranging from "a major breakthrough" to "just another buzzword." A number of writers claimed DSS as an advancement of the information systems technology starting with electronic data processing (EDP) to management information systems

(MIS) to the current DSS thrust. Some claimed DSS as an important subset of management science served to complement the parts that MIS have failed; and there are others that claimed it to be an extension of management science techniques. The former sees DSS as providing managers and users access to data and the latter as giving them access to analytical models. There are the majority who view DSS as an integrated system giving managers and users access both to data and models. This is the view that shall be adopted in this paper. And finally, there are people at the other extreme who believe that DSS are nothing more than just another buzzword to enhance the slacking image of management scientist and computer specialist or just another ploy by the computer vendors to increase the sale of computers.

The paper begins by examining some of the definitions of DSS, focussing on the issues highlighted in each definition so that a general consensus on what DSS is can be formulated. This is important in view of the fact that various definitions of DSS has been given in the literature which could mislead the readers concerning what DSS really is and how it differs from other types of information systems. Then the hardware and software requirements of DSS is briefly reviewed. Following this, a generic framework for DSS design is presented. Finally, the design strategies for DSS is discussed.

DEFINITION

The concepts involved in DSS were first articulated in the early 1970s by Scott-Morton under the term 'management decision system' (Scott-Morton, 1971). In a follow-up article in the same year, Gorry and Scott-Morton used the term DSS to define information systems that support managerial decision makers in unstructured and semi-structured decision situations (Gorry and Scott-Morton, 1971). Structured decisions on the other hand "encompass almost what has been called MIS in the literature, an area that has had almost nothing to do with real managers or information but has been largely routine data processing" (Gorry and Scott-Morton, 1971, p. 61). The information systems warrant distinction because different types and characteristics of information are needed to support managers in solving the different types of organisational problems.

There are three key concepts in their definition of DSS. First, DSS is meant to be an adjunct to the decision maker. It serves as a tool to extend his capabilities, but not to replece his judgement.

Second, they are aimed to support the decision maker in solving semi-structured and some parts of unstructured decision situations. These are decisions whereby managerial judgement alone is inadequate to comprehend them, perhaps due to the size of the problems,

the computational complexities and precision needed to solve them. On the other hand, it cannot be entirely automated because due to our limited understanding of the problems, only some algorithm or decision procedures can be specified and hence, programmed into the computer. Under these conditions, the decision maker together with the computer can provide a more effective solution than either alone.

Third, though not specifically stated, they implied that the system should be an on-line computer system incorporating data retrieval and modelling capabilities.

Keen and Scott-Morton view DSS as representing "the role of the computer in the management decision making process. Decision support implies the use of computers to (i) assist managers in their decision processes in semi-structured tasks; (ii) support, rather than replace, managerial judgement; and (iii) improve the effectiveness of the decision makers rather than its efficiency" (Keen and Scott-Morton, 1978; p.1). In addition to these, Keen also applies the term DSS "to situations where a 'final' system can be developed only through an adaptive process of learning and evolution" (Keen 1980, p. 15). In other words, he views DSS as the product of a development process in which the DSS user, the DSS builder, and the DSS itself are all capable of influencing one another and resulting in the evolution of the system and the pattern of its use.

Alter (1980) contrasts DSS to the traditional EDP systems on five dimensions:

	EDP	DSS
1/ Use	Passive	Active
2/ User	Clerk	Line staff & Management
3/ Goal	Mechnical Efficiency	Overall Effectiveness
4/ Time horizon	Past	Present and Future
5/ Objective	Consistency	Flexibility

He then proposed a typology for categorising DSS as follows:

File Drawer Systems	Data Retrieval	Data Oriented
Data Analysis System	Data Analysis	
Analysis Info. Systems		
Accounting	Simulation	Model Oriented
Representational		
Optimising Models	Suggestion	
Suggestion Models		

FIGURE 1. Categories of DSS

Source: Alter (1977), p. 42.

Moore and Chang (1980) also distinguish DSS from MIS and EDP. They refer to DSS as systems which are:

1/ intrinsically dynamic in terms of design and implementation;
2/ extensible in order to match the decision makers revealed preference;
3/ used at irregular or unplanned intervals.

They further argued that the 'structuredness' concept, so much a part of the early definition of DSS, is confusing and not meaningful in general. This is because a problem can be viewed as structured or unstructured only with respect to a particular decision maker or group of decision makers.

Bonczek, Holsapple and Whinston (1981) defined DSS as a computer-based system consisting of three interacting components:

1/ a Language System (LS) — linguistic facilities to provide communication between the user and other components of DSS;
2/ a Knowledge System (KS) — the repository of knowledge, data procedures, and models to solve particular application problems; and
3/ a Problem Processing System (PPS) — consisting of software which controls the action taken by the DSS. It is the link between the other two components.

And finally, Ginzberg and Stohr (1982) defined DSS as "a computer-based information system used to support decision making activity in situations where it is not possible or not desirable to have an automated system perform the entire decision making process."

In summary, it is difficult to precisely define what DSS is. The term is used fairly loosely as to include any system that supports decision making. In other words, all information systems, with the exception of EDP, are DSS. On the other hand, there are others who restrict and narrow the term to include certain characteristics only. For example, Keen (1980) would exclude any system which is not developed through an evolutionary proces of learning and adaptation; whilst Moore and Chang (1980) would exclude information systems which are used at regular and planned intervals. Within the context of definitions given by Keen, Moore and Chang, and some others, few actual systems can be categorised as DSS.

The key concepts underlying many DSS definitions can be observed as follows:

TABLE 1. Key Concepts in DSS Definitions

Key Concepts	Authors
1. Systems capabilities	— Alter (1977, 1978) — Moore and Chang (1980)
2. Systems objectives	— Gorry and Scott-Morton (1971) — Keen and Scott-Morton (1978) — Alter (1980) — Ginzberg and Stohr (1982)
3. Usage Pattern	— Alter (1980) — Moore and Chang (1981)
4. Problem type	— Gorry and Scott-Morton (1971) — Keen and Scott Morton (1978) — Ginzberg and Stohr (1982)
5. Development Pattern	— Keen (1980) — Bonczek et al (1980)

In general, the following characteristics of DSS can be observed:

1/ DSS aim at the less structured underspecified problems faced by the middle and senior management.
2/ They combine the use of models and other analytical techniques with the traditional data access and retrieval functions.
3/ Systems design emphasise flexibility and adaptability. The systems development may follow an evolutionary process of learning and adaptation.
4/ DSS serve as a tool under the users control to support and enhance their capabilities in making the appropriate decisions. As a result, it may ultimately increase the efficiency and effectiveness of the organisation in general.
5/ Most systems provide on-line real-time interaction with the users though this is not seen as an essential and distinctive characteristic of DSS.

DSS VERSUS MIS

How does DSS differ from MIS so as to justify the use of a new term? DSS and MIS are both concerned with information processing within organisation. However, unlike MIS, DSS is always specifically oriented towards that kind of information processing activity that constitutes a decision process.

Keen and Scott-Morton (1978) argued that DSS differ from MIS in terms of area of impact on and the payoff to the organisation, and relevance to managers. These differences can be observed from Table 2:

TABLE 2. Difference between DSSs and MISs

Key Factors	MIS	DSS
1. Impact:	On structured tasks where standard operating procedures, decision rules, and information flows can be reliably defined.	On decision in which there is sufficient structure for computer and analytical aids to be of value but where manager's judgement is essential.
2. Payoff:	Improving efficiency by reducing cots, turn-around time, and so on.	Extending the range and capabilities of managers' decision processes to help them improve their effectiveness.
3. Relevance to Managers:	Effect on their decision making is indirect; for example by providing reports and access to data.	Serve as a supportive tool, under their control, which does not attempt to automate the decision process; pre-defined objectives; or impose solutions.

Source: Keen and Scott-Morton, 1978

One frequently-mentioned difference is that MIS are often report oriented, in the sense that data is extracted and summarised into predefined formats, often on a periodic basis, for purposes of managerial review in reporting exceptions, deviations from standards, and historial trends. In many situations, DSS subsumes portions of MIS, especially the data summarisation and extraction capabilities. In addition to these, however, DSS often provides ad hoc enquiry capabilities or a personalised system according to the users preference and experience. The main focus is on flexibility, adaptability, and quick response.

With regards to the technology that makes up a DSS, like many other computer-based systems, there is the hardware and the software. In terms of hardware, there is no difference from those of MIS or any other modern-based computer systems. There are no special hardware requirements in order for a system to qualify as a DSS. In

terms of software requirements, there are a number of distinctive features. The most important, drawing upon the various case studies and reports of DSS systems, is that most DSS include data extraction and manipulation capabilities (DBMS), or modelling capabilities, or both.

DSS HARDWARE AND SOFTWARE REQUIREMENTS

DSS HARDWARE CLASSIFICATION

Ther is no difference between DSS hardware and the hardware that supports other computer-based systems. No special hardware is required for DSS, nor are there any necessary hardware requirements to qualify a system as DSS (Ginzberg and Stohr,, 1982). However, further advances in hardware development will certainly contribute to better and more effective DSS.

The greatest possible impact on DSS will be the introduction of a new generation of computers to be known as the Fifth Generation computers which are being developed by the Japanese, Americans, and others on a nationally coordinated effort. Some of the hardware features of this generation of computers will be higher central processing unit (CPU) speed and larger memory. The Japanese believe that by using Very Large Scale Integrated (VLSI) technology, it will be possible in the near future to develop a current large size processor with a few thousand gates on a single chip. (Moto-oka, 1982). Such a development can help to extend the range of structurable decision situations by making more sophisticated heuristic and artificial intelligence applications possible; believed to be of greatest potential to DSSs developments in the future.

This generation of computers may also include microcomputers. Advances in VLSI technology would enable these generation of micros to have comparable performance and functions with present day large size computers. DSS, which in many situations, are characterised by personalised usage, can be more effective and may increase individual effectiveness. Also package systems of various functions which are impractical on present day micros will be possible.

DSS SOFTWARE CLASSIFICATION

DSS is very much inter-disciplinary in nature drawing upon the work of data base management, management science and modelling, language presentation and processing, artificial intelligence, and so forth. As such, the DSS software components are diverse. Each system may emphasise on one or more of the softwave components, depending on the objectives and the nature of the decision situation the system is designed to support.

Four principal software components can be identified: data base management system, modelling system, programming languages, DSS generator.

Data Base Management System (DBMS) DBMS are used to deal with storage and retrieval, and the way these data are represented. As such, it constitutes an important tool for building DSS, particularly data oriented DSS. Alter (1977), based on his studies of 56 DSS, dichotomised them into two main types: data-oriented DSS and model-oriented DSS. As implied by its name, the former mainly use DBMS or other data management techniques to support efficient retrieval, manipulation, and/or analysis of data; while the latter uses models — simulation, optimisation, and/or suggestion to support and aid the users in the decision making process.

The complexity of a DBMS depends on the emphasis for data storage and retrieval, and the way these data ar represented. In many DBMS, data is represented in terms of a logical structure called a data model. The main purpose for this is to spare users concern over the details of dealing with various kinds of hardware storage devices. The way a DBMS generally works is as follows: Logical data model is represented in a schema. Raw data is independently represented in various storage devices. A user states his request with the logical data in the schema which is then mapped by the DBMS into the physical data. In this respect, users need not have to know the details of how the physical data is actually stored. An additional advantage in using DBMS is what is known as 'physical data independence,' that is, the DSS builder can change the physical structure of the data base without effecting the logical structure perceived by the users.

Commercially available DBMS are based on some version of three main data models: the relational model, hierarchical model, and network model. Each of these models has certain strengths and weaknesses and its choice by a DSS builder should be considered within the objectives and requirements of the DSS.

Model Management System. The concept of a Model Management System (MMS) is an innovative product of DSS. Its purpose is to facilitate the development and management of various models. Like DBMS, the complexities and tools used may vary widely. Some MMS may incorporate model generators such as modelling packages (e.g. FCS, PROSPER, AS, ORACLE and others), spreadsheets (e.g. MULTIPLAN, LOTUS/SYMPHONY, SUPERCALC and others), and prefabricated programs for solving straight forward analytical problems relating to inventory control, Internal Rate of Return (IRR), and so forth. Some systems may contain simple management science models whilst others may incorporate corporate/financial model for corporate and financial planning purposes.

Programming Language. Programming languages such as APL, FORTRAN, or BASIC have commonly been used to build DSS. These languages are well suited for implementing quantitative algorithm but not suitable for representing qualitative knowledge. Not all aspects of a decision making process involve quantitative data. In fact, many aspects of the less structured problems, which most DSS are designed to support, use qualitiative data. In such situations, they involve making inference rather than numeric computation. Furthermore, models in some application areas are more concerned with logical inferences than numeric computations. This implies that programming languages oriented towards logical inference can be a valuable tool for building DSS. Two frequently mentioned languages are PROLOG and LISP.

PROLOG is a high level and general-purpose programming language based upon the procedural interperatation of definite clauses known as Horn clauses (Horn, 1951) of predicate logic (Kowalski, 1979; Bonczek et al 1981; and Coelho, 1983). PROLOG, in contrast with languages like BASIC or FORTRAN, does not have assignments as the basic underlying operation. Instead, a PROLOG program is composed of a collection of axioms, each of which is a Horn clause based on first-order predicate logic. Hence, the execution of a PROLOG program attempts to prove a theorem, and the way it is carried out is specified partly by the logical declarative semantics of PROLOG, partly by what new facts it can 'infer' from the given ones, and partly by explicit control information supplied by the programmer (Coelho, 1983; Bonczek et al, 1981).

DSSs Generator. DSS generator are used to facilitate the building, maintenance, and extension of DSS efficiently and effectively in the shortest possible time. It may consist of various tools such as compilers and editors, or prefabricated software such as spreadsheet, modelling systems, DBMS and so forth.

Decision situations, in general, consist of two types: institutional/recurring DSS and ad hoc DSS. (Donovan and Madnick, 1977). Institutional DSS deals with decisions of a recurring nature such as portfolio management system, inventory management system, etc., while ad hoc DSS deals with specific problems that are neither anticipated nor recurring. Their relationship can be viewed as in Figure 2.

The design of an ad hoc DSS must focus on responding quickly on a one-time basis. The concern to the designer must be on rapid assimilation and integration of seemingly incompatible computer programs and data series; whilst the concern in an operational DSS is on operational costs and timing. (Donovan and Madnick, 1977). In institutional DSS, the problem is well understood and the usage pat-

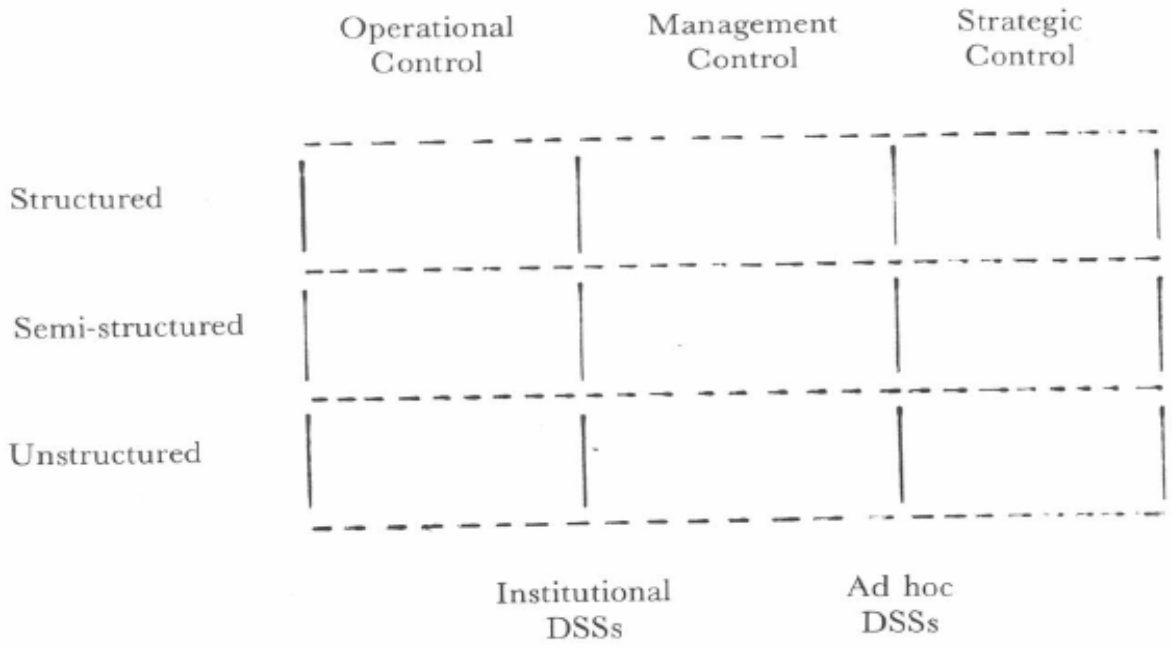

FIGURE 2. Institutional DSSs and Ad Hoc DSSs

Source: Donovan and Madnick, (1977)

tern known (perhaps through many years of using it or through better understanding of the problem algorithm); but in ad hoc DSS, these cannot be pre-determined. Hence, it becomes necessary to have a generalised DSS to facilitate building ad hos DSSs as quickly as possible. The generalised DSSs described by Donovan and Madnick (1977) is called Generalised Management Information System (GMIS). It has been designed to allow for the rapid assimilation and integration of incompatible computer programs and data series.

A classification scheme with similar implications is proposed by Sprague and Carlson (1982). They identified three levels of DSS hardware/software technology: specific DSS, DSS Generators, and DSS tools. Their relationship can be viewed as follows:

Specific DSS (conceptually similar to institutional DSS in Donovan and Madnick's classification) are built to support a particular application or tasks. Sprague and Carlson (1982) view DSS generator as a 'package' of related hardware and software which provides a set of capabilities to build specific DSS quickly and easily. The package may consists of a set of interpreters, and data creation utilities such as modelling system, spreadsheet, file handling package, statistical subroutines, etc. DSS tools are the hardware and software elements that facilitate the development of a specific DSS or to modify the DSS generators. This includes programming languages, special purpose

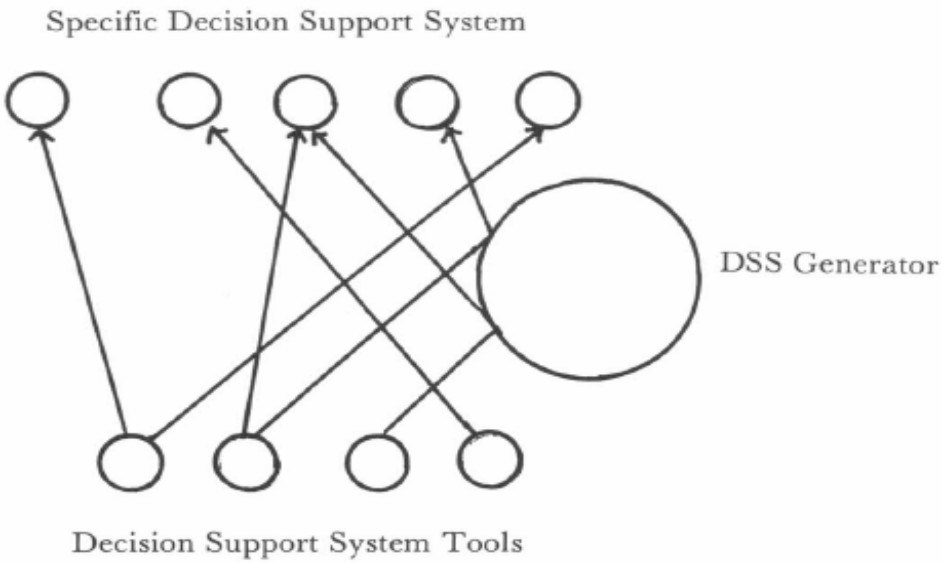

FIGURE 3. Levels of DSS Technology

Source: Sprague and Carlson, 1982; p. 12

interpreters, editors, and data creation utilities. (Sprague and Carlson, 1982).

Naylor and Schauland (1976) found a marked growth in the use of fabricated DSS generators in the U.S.A, particularly for building corporate financial planning models. Commercially available DSS generators include Executive Information System (EIS) by Boeing Computer Services; Interactive Financial Planning System (IFPS) by Execucom System of Austin, Texas; and SIMPLAN by SSI, EPS-FCS; and Lotus 1 – 2 – 3/Symphony.

An important consideration in using these prefabricated packages is the degree of integration with other software components of the DSS. As noted by Bonczek et al (1981), the degree of integration varies from one system to another. They cited MDBS's Knowledge Man as providing very high integration, incorporating speradsheet, relational data base management (with SQL ad hoc inquiry), structured programming language, screen management, and statistical analysis functionalities into a single system.

FRAMEWORK FOR DSSs DESIGN

Beyond the definitions, a generalised framework of DSS can serve as a useful basis for further understanding and research of the subject matter. However, like its definition, there seems to be little consensus among researches on a common generic framework of a DSS.

A common form, based on research and the various case studies reported by Alter (1977), Keen (1980), Sprague and Carlson (1982), Ginzberg and Stohr (1982) and others, is depicted in Figure 4:

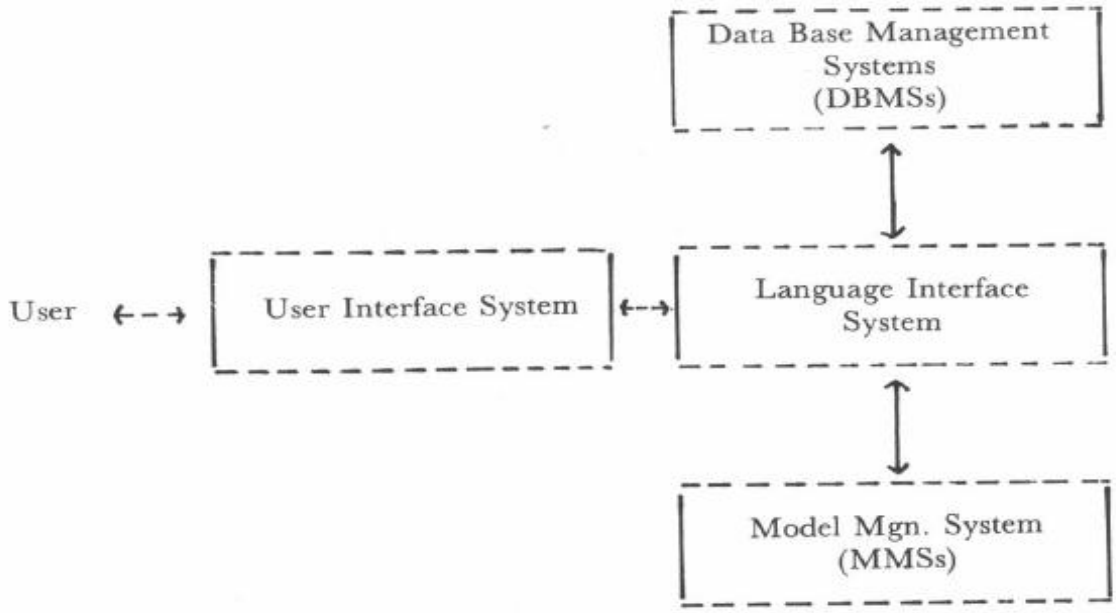

FIGURE 4: Generalised Framework of a DSS

A DSS is generally viewed as having four principal components: User Interface System, Data Base Management Systems, Model Management Systems and Language Interface System.

The use of a User Interface System as a separate layer of software between the end users and the DSS serves to eliminate the need for special application programming to generate displays and control dialogues; and to provide a variety of interface facilities that are familiar to the users.

The Language Interface System consists of the compilers and interpreters that translate the statements and commands used for specifying retrieval requests and defining models. An important consideration is its ability to provide integration between data representation in Data Base Management Systems and Model Management Systems so that the output of an analysis can be used as input to another. The language used in the system has an important effect on the ease of use of the system. Programming languages such as APL, BASIC, or FORTRAN can be well suited for implementing quantitative algorithm but not suitable for representing qualitative knowledge. On the other

hand, PROLOG or LISP are well suited for representing qualitative and heuristic knowledge but inefficient in performing mathematical computation or manipulation of large quantities of quatitative data.

As discussed above, DBMS are used to deal with the storage, maintenance, and retreival of large volumes of interrelated data whilst MMS is to facilitate the development and management of various models.

Another form of DSS framework stems from a growing interest in DSS research to incorpoate the techniques and methods of Artificial Intelligence, especially in the area of knowledge and expert systems. Drawing upon this background, Bonczek, and Whinston (1981) proposed a generic framework of DSS comprising of three principal components: a Language System (LS); a Knowledge System (KS); and a Problem Processing System (PPS). Their relationship can be viewed as follows:

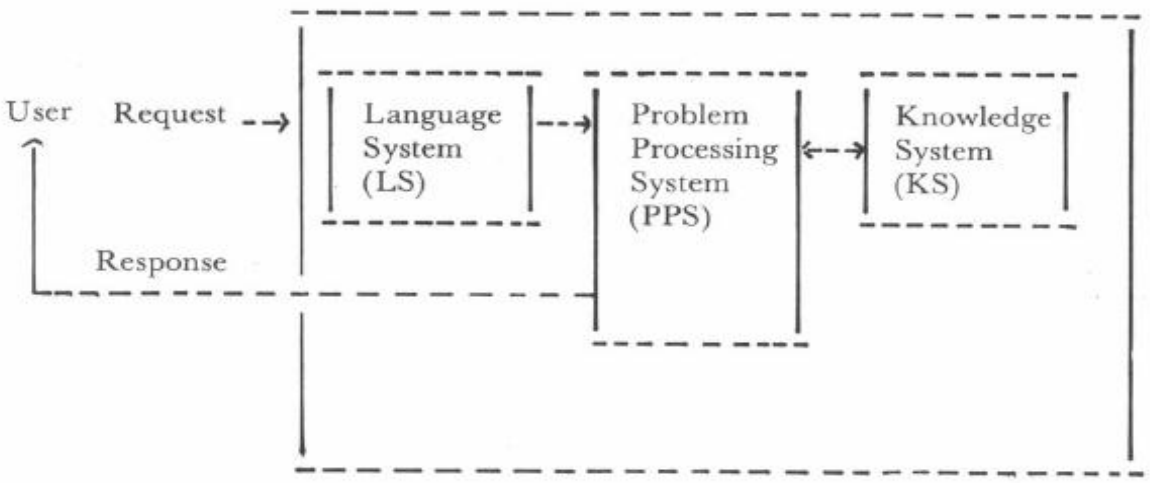

FIGURE 5. Generalised Framework of DSS Incorporating Knowledge-based Representation

Source: Bonczek et al, 1981.

A Language System (LS) is the total of all linguistic facilities made available to the user of a decision support system. (Bonczek et al, 1981). A user states requests in terms of the DSS's language system. They are typical requests for data retrieval and/or requests to carry out some computation (eg. model execution).

The complexities and orientation of the LS for data retrieval and model execution vary, depending on the decision situation the DSS serves to support. For data retrieval, the LS covers a range of two

extremes. At one extreme are languages in which a user explicitly states how the data is to be retrieved. In other words, retrieval is procedural. On the other extreme, a user retrives by merely stating what data is desired. The user is not required to know about logical or physical data organisation. Between these two extremes are those LS that allow a user to invoke one of a number of report generation programs, which have been predefined to meet the anticipated needs of users. By allowing conditional retrieval, such languages need not be entirely inflexible.

LS for directing numerical computations also covers range of two extremes. At one extreme are programming languages with which a user explicitly specifies all computations on a step-by-step basis. At the other extreme are LS with which a user merely states the problem to be solved in terms of the data desired. Between these two extremes are LS that enable a user to invoke a model or a number of models by name. Here, the models have been predefined to meet the anticipated needs of the users.

A Knowledge System (KS) is the DSS's body of knowledge about some problem domains. As pointed out by Bonczek et al, knowledge is the key factor in the performance of a knowledge-based DSS. Knowledge is of two types: the first is the facts of the problem domains which includes the model knowledge and the data used by these models (Bonczek et al, 1981); and the second type is called heuristic knowledge — the knowledge of good practice and judgement in the problem domain which is acquired and experienced through the years. (Feigenbaum, 1984).

The knowledge in the KS must be stored in some organised manner. The main consideration must be efficiency, integrity, and ease of design. In this respect, depending on the object of the DSS, the knowledge can be represented as relations or tables of data (flat files). Though a DSS builder can choose from various knowledge representation methods, the choice is largely determined by the tools used to build the DSS. (Bonczek et al, 1981). For instance, if a spreadsheet package is used as a tool for DSS construction, both empirical and formula knowledge are represented in the form of a spreadsheet. In this case, however, the resultant DSS contains no lingustic or representation method since all of these are incorporated in the spreadsheet. Apart from spreadsheet, other common knowledge representations include file structures, data base structures, and axiomatic systems. Thus, it is important to select tools that provide knowledge representation methods which are appropriate for the specific decision application to be supported.

At the heart of DSS is its Problem Processing System (PPS). Unlike the LS and the KS, which are systems of representatio, the PPS is a

dynamic system consisting of software that bridges the gap between the LS and the KS. (Bonczek et al, 1981). Given a problem (stated with the LS) and problem domain knowledge (represented in the KS), the PPS software attempts to derive a solution which supports (enhances, facilitates, or makes possible) a decision making process. Ideally, the PPS should possess data collection, problem recognition, automatic model formulation capabilities, and others but the extent of each ability of a given processor varies widely. As recognised by Bonczek et al and others, present day problem processors have limited capabilities particularly for problem recognition and automatic model formulation.

In considering this form of framework, the question arises as to the difference between a knowledge-based DSS which uses artificial intelligence type knowledge representation and an artificial intelligence expert system using a similar mechanism. The basic distinction is in the system's objectives and usage patterns. An expert system seeks to replicate, hence replace the abilities of a human expert in a specific problem domain. Its use in solving a particular problem does not require any knowledge from the human user (since any known knowledge of a problem domain will be stored in the KS). In addition to this, the system does not generate any learning process to the user and vice versa. A knowledge-based DSS on the other hand seeks to assist a human user by taking over the more structured parts of a larger, only partially formulised tasks, of a problem domain.

DSS DESIGN STRATEGY

There are three main agents that are integral to the development of a DSS:

1/ the client/initiator, who initially solicits the development of the system, specifies the overall performance criteria, and often pays or authorises payment for the system;

2/ the managers/users, who actually use the DSS and to whom the support aspects are tailored; and

3/ the designer/toolsmith, who specifies the detailed structures and capabilities of the system and is responsible for its implementation.

An addition, Bonczek et al argued the need for a DSS administrator to maintain the DSS especially in larger organisations. All these functions may be performed by the user if he has the expertise to do so.

This section shall look into DSS from the perspective of a prospective DSS designer or toolsmith. Compared with other aspects of DSS, little has been written about DSS design methodology. Three approaches have been proposed in designing DSS: the traditional ap-

proach, commonly known as the development life-cycle approach; phototyping; and evolutionary approach.

Development life-cycle approach. This approach was mainly used in designing MIS and other traditional computer-based systems. Under this approach, systems development follows an invariably standard pattern or procedure. It normally starts with an exhaustive analysis of users requirements, which is then specifically spelt out in the systems design, and finally implemented, often initially hand in hand with the existing systems. This approach has a number of fundamental deficiencies that makes it inadequate for the tasks of building succesful DSS.

Firstly, the development life-cycle approach presumes that the users can specifically spell out their requirements and the designer can understand their needs. As argued by many (Ackoff 1967; Moore and Chang 1980; and Keen 1980), this is an idealised abstraction. In practice, pre-specification of systems requirements are far more difficult to achieve. In many circumstances, the designer and/or users cannot provide functional specifications or are unwilling to do so. More important is the focus of DSS, that is, on the less structure tasks whereby pre-specification of functional requirements is uncertain or difficult to define.

Proponents of the development life-cycle approach argued that all systems evolve, during which various processes of iteration takes place, that is, the designer must cycle through feasibility assessment, information analysis, and systems design several times before converging to a final system. During these iterations, all the inconsistencies and ambiguities can be resolved; and permit the users to express their true preferences. It has been argued that even then, the final specification is difficult to achieve, particularly with regards to solving the less structured tasks, because the user may take too long to understand and have the knowledge to lay out procedures and requirements. Indeed, Keen argued that with most management problems, a final specification can never be made by the users. (Keen, 1980).

Secondly, directly related to the first point is what Moore and Chang termed as "problem migration." What this means is that the decision context within DSS, by virute of the problem type, modifies itself and is modified by the very attempt to obtain specification for its design. In the words of Moore and Chang, "... the decision process itself is migrating in the minds of both the decision makers and the designers" (Moore and Chang, 1980). Under these circumstances, the development life-cycle aproach would be ineffective because it implicitly assumes that decision making problems requiring the capabilities of a computer-based system is essentially static, and that

the process of iterations within the life cycle phases can finally lead to a convergent. In DSS, however, there is no guarantee that the process will be convergent.

Thirdly, understanding the decision rules of the less structured problems may take a long time and to create such understanding, the system must be capable of stimulating learning to the users which in turn stimulates new uses. The traditional life cycle approach even if iterated, has its ultimate goal of "freezing the spec" and to proceed with actual systems design in a procedural and efficient manner (Moore and Chang, 1980). As such, they argued that such an approach is not possible in an "unstructured setting without intimidating the decision makers themselves, forming premature closing on problem solving approaches, and inhibiting learning and search processes that are essetnial for the managers to undertake in addressing less 'structured' planning oriented tasks" (Moore and Chang, 1980; p. 11).

And finally, the approach is inappropriate in designing systems to support ad hoc usage patterns. The procedural nature of the approach inhibits a quick response in solving the many business problems that require urgent consideration.

Hence, two alternative approaches have been proposed for designing DSSs:

— Prototyping; and
— Evolutionary/Adaptive process.

Prototyping. Prototyping, like the life cycle approach, requires a systematic approach to designing, but prototyping avoids exhaustive analysis of users requirements and detailed design which is greatly emphasised in the life-cycle approach. "Efficiency goals are replaced by effectiveness goals". (Henderson and Ingraham, 1982). With prototyping, users interaction with the systems is provided at the earliest possible moment. This is in contrast with the life cycle approach where exhaustive efforts and resources are almost wholley concenrated in processes prior to users interaction/implementation. It is often mentioned that about 70% of the project effort and costs are spent before any actual usage by the users.

In prototyping, the process of iterations is also used but the feedback time is relatively fast. The focus is on timely feedback so that it can effectively support learning (Henderson et al, 1982). Encouraging learning is the key requirement, and as such prototyping greatly emphasise user(s) participation in the design and implementation processes to ensure success. Indeed, user participation is important regardless of the approach under which a system is designed.

Finally, with prototyping, the development costs of the project can be better controlled. In the life cycle approach, costs often ex-

ceeded budget because major errors are normally discovered during the last stage of the exercise, that is, during implementation when a considerable proportion of the budget has been spent. This can be overcome by prototyping.

Evolutionary/Adaptive Process. The evolutionary process can be represented as in Figure 6 below:

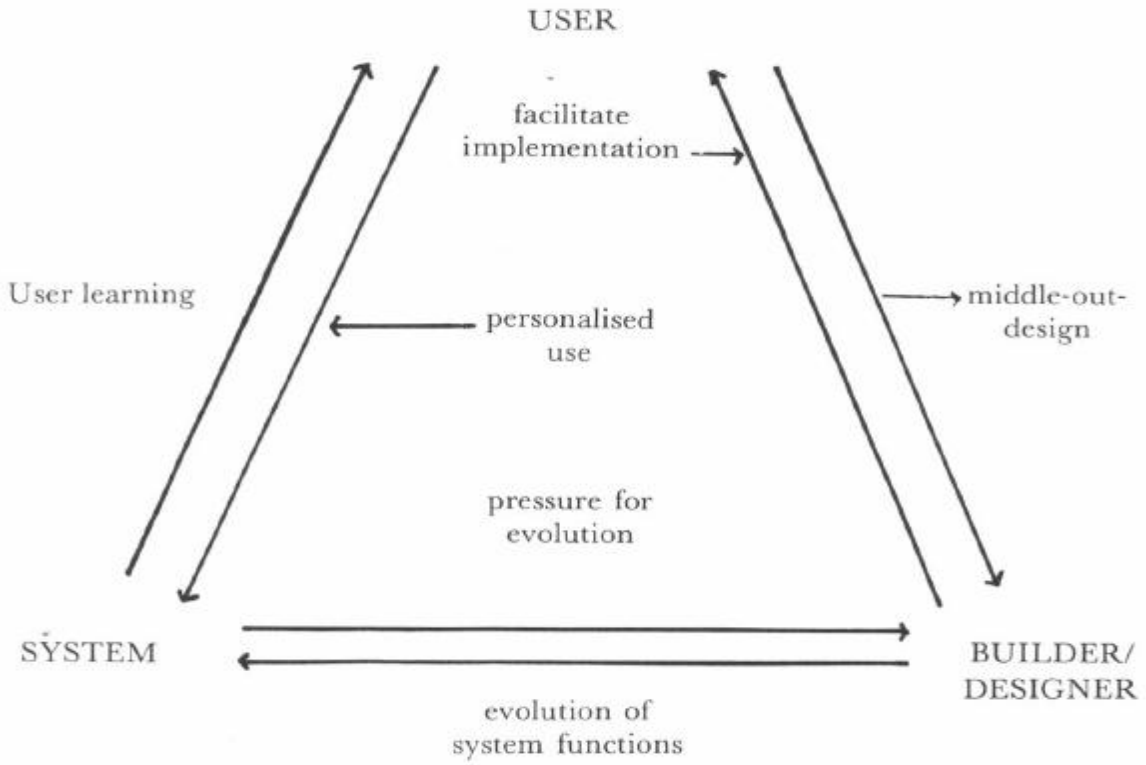

FIGURE 6. An Evolutionary Framework for DSS

Source: Keen, 1980

The arrows represent a direction of influence. For example, SYSTEM---→USER indicate that learning is stimulated by the DSS while USER-----→SYSTEM refers to the personalised, differenciated mode of use that evolves. Keen pointed out that the evolutionary/adaptive processes work together, that is, an effective DSS encourages users to explore new alternatives and approaches to the problem/task (SYSTEM----→USER). This in itself stimulates new uses of the system, often unanticipated and idiosyncratic (Keen, 1980; p. 16).

The arrows in Figure 6 are not merely conveniently schematic. According to Keen, they help clarify whether a particular system should be called a DSS or not. For example, an airline reservation system is technically similar to

many retrieval-based DSS. However, it is not intended to stimulate learning (SYSTEM --/--- USER), nor are there personalised modes of usage; there is a 'right' way to use the system and the user must adjust to it, not vice-versa. Hence, Keen argued that a system can only be called a DSS if it evolves from adaptive development.

Moore and Chang argued that an evolutionary process is often necessary because of what they called "problem migration", that is, the users problem or problem view is constantly changing in the minds of the users as well as the designer. Hence, the system must evolve to remain relevant and useful (Moore and Chang, 1980; p. 24). The design strategy they proposed is called "subset evolution"; an approach almost similar to prototyping. It involves developing expanding subsets of system capabilities based on an initial extensible nucleus. Using this method, the initial requirement analysis is minimal and geared towards providing sufficient information to build only the nucleus of the system in the shortest possible time. In Moore and Chang's word, "the implementation is the feasibility study" (Moore and Chang, 1980; p. 12).

The evolutionary approach is not without criticisms. Ginzberg and Stohr (1982) argued that any system that represents an initial effort is likely to evolve, both in form and usage pattern. This may result from learning or past experience. Hence, they argued, evolutionary design and usage does not appear distinct to DSS. "It fails to distinguish DSS from other computer-based systems, and ascribes to all DSS something which is characteristic only to some" (Ginzberg and Stohr, 1982; p. 21).

Despite this, the approach seems to be acceptable to many people because in situations where substantial uncertainty exists about users needs or probable usage patterns, an evolutionary design process may well be the most appropriate and effective method. But where litle uncertainty exist about users requirements and a fairly stable usage pattern can be projected, prototyping may well suit the situation. The key decision is to match and design approach to the needs of the situation.

CONCLUSION

DSS is designed to support the less structured decision situation usually faced by the management. It serves to extend the range and capabilities of the managers decision processes to help them improve their effectiveness in making decisions, financial or otherwise. It is intended to be a supportive tool under the manager's and other user's control, which does not attempt to automate the decision process or impose solutions.

The nature of a DSS depends on the type of decision situation it is meant to support. It may use extensive Data Base Management System for data-retrieval and data analysis — data oriented DSS, or use various forms of analytical and decision models — model oriented DSS.

Three main approaches have been proposed for designing DSS: the traditional development life cycle approach often used in designing management information systems; prototyping; and evolutionary approach. The relevant approach in a particular situation depends on the need and level of uncertainty (about user needs and usage pattern) of the situation.

Despite some early success of DSS, continued research and progress need to be made in important avenues such as knowledge representation to be made in important avenues such as knowledge representation and related areas of artificial intelligence, modelling techniques, user interface, and in other related areas, in order to further increase its effectiveness and impact on managerial functions. The introduction of a new generation of computers — the Fifth Generation, promise greater impact on DSS applications. However, advancement and development in computer hardware is not enough to ensure success in information systems development, be it DSS or other information systems. What is more important and relevant is the advancement in our understanding of business problems and decision situation which DSSs are design to support. Without these, we cannot reap the full potential of computing.

REFERENCES

Ackoff, R.L., 1967. Management Misinformation System. *Management Science,* Vol. 14, No. 4 Dec. pp. 147 – 156.

Alter, S.L., 1980. *Decision Support Systems: Current Practises and Continuing Challenges.* Addison Wesley.

Bonczek, R.H., Holsapple, C.W., and Whinston, A.B., 1981. *Foundation of Decision Support Systems.* Academic Press.

Coelho, H., 1983. Prolog; Tool For Logical Domain Modelling. *In* Sol, H.G. (ed.). *Process and Tools For Decision Support Systems.* North-Holland.

Donovan, J.J. and Madnick, S.E., 1977. Institutional and Ad Hoc DSS and Their Effective Use. *Data Base.* Vol. 8 No. 3 Winter.

Feigenbaum, E.A. and McCorduck, P., 1984. *The Fifth Generation.* London: Pan Books.

Ginzberg, M.J., and Stohr, E.A., 1982. Decision Support Systems: Issues and Perspective. *In* Ginzberg, M.J. et al. *Decision Support Systems: Proceedings of the NYU Symposium On DSS.* New York, May 21 – 22, 1981. North-Holland.

Gorry, G.A. and Scott-Morton, M.S., 1971. A Framework For Management Information Systems *Sloan Management Review*. Vol. 13 No. 1, Fall.

Henderson, J.C. and Ingraham, R.S., 1982. Prototyping For DSS: A Critical Appraisal. *In* Ginzberg, M.J. et al. *Decision Support Systems: Proceedings of the NYU Symposium on DSS*. N. York May 21 – 22, 1981. North-Holland.

Keen, P.G.W., 1980. Adaptive Design for DSS. *Data Base*. Vol. 12 No 1/2, Fall.

Keen, P.G.W. and Scott-Morton M.S., 1978. *Decision Support Systems: An Organisational Perspective*. Addison-Wesley.

Moore, J.H. and Chang, M.C., 1980. Design for Decision Support Systems. *Data Base*. Vol. 12 No. 1/2, Fall.

Moto-oka, T., 1982. *Fifth Generation Computer Systems: Proceedings of the the International Conference on Fifth Generation Computer Systems*. Tokyo. Oct. 19 – 22, 1981. North-Holland.

Naylor, T.H. and Schauland, H., 1976. A Survey of Users of Corporate Planning Models. *Management Science*. Vol. 22 No. 9, May.

Scott-Morton, M.S., 1971. *Management Decision Systems: Computer-Based Support for Decision Making*. Div. of Research, Harvard, Camb., Mass.

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