A MICROCOMPUTER-BASED DECISION SUPPORT SYSTEM

Antoni Olivé
Facultat d'Informàtica de la U.P.B.
Jordi Girona Salgado, 31.
Barcelona-34. Catalonia.

ABSTRACT

The last ten years have seen the emergence of some new ways of applying computers to organizations. This has been done by focusing attention on the support of the key decisions that must be made in organizations. Decision Support Systems (DSS) is the term that is used to designate systems specifically designed to be used by managers during their decision making. This paper addresses the problem of using microcomputer-based DSS in small firms. We analyze both the applicability of the DSS concept in small firms and the problems that arise in the implementation of DSS'S in microcomputers.

1. INTRODUCTION

1.1. The emergence of Decision Support Systems

Most computer-based information systems have been aimed at transaction processing, operational control or management control. However, the last ten years have seen the emergence of some new ways of applying computers to organizations. This has been done by focusing attention on the support of the key decisions that must be made in the organizations [10].

The reasons for this emergence are that improvements in decision making can result in high benefits, and that decision making can be improved through computer support.

The specific computer support best suited for a given decision depends on a variety of factors. Among these, there is the type of decision. H.A. Simon [11] distinguishes two polar types of decisions: structured (programmable) and unstructured (non-programmable). They are not really distinct types, but a whole continuum, with highly structured decisions at one end of that continuum and highly unstructured decision at the other end. Decisions not in the extremes have been referred to as "semistructured" decisions [6].

Computer support for decision making provided by traditional applications is mainly for structured decisions. For the other types of decisions these applications merely provide data to the decision makers, usually in the form of periodic reports.

However, many authors argue that precisely the semistructured and unstructured decisions are the ones of greatest concern to decision makers [5]. For these decisions, effective computer support must include capabilities such as interactive computer systems, data bases, decision models and others.

Decision Support Systems (DSS) is the term that has been proposed to designate systems specifically designed to be used, on a ongoing basis, by managers during the process of making those types of decisions.

1.2. The problem

Up to now most DSS have been designed for medium or large organizations, and they have been implemented in medium or large computers. While several exceptions exist, the point we want to stress is that few, if any, DSS have been designed to be used for small organizations with small computers.

Recent evolution of computer technology has made small computers accessible to the small firm. Today, many small firms use microcomputers for their transaction processing or operational control. They generally use the microcomputer in applications such as financial accounting, payroll and invoicing.

These applications do not aim at providing computer support for semistructured or unstructured decision making. However, small firms would probably welcome this support, if properly provided, as the larger organizations have.

This paper addresses the problem of using microcomputer-based DSS in small firms. We want to analyze both the applicability of the DSS concept in small firms and the problems that arise in the implementation of DSS'S in microcomputers.

In our opinion, the problem is important, at least for two reasons. First, there is a large number of small firms in our economy, so that computer hardware and software vendors may have an interest in the development of DSS'S for small firms.

The second reason is that DSS'S have high potential value for small firms, since DSS can be an effective means to improve the effectiveness of the firms as well as their efficiency.

1.3. Previous work

There is an abundant and growing literature on DSS. The literature describes a number of detailed case studies as well as frameworks, conceptual proposals, design methodologies, impact analysis, etc.

In spite of this, there is no a common agreement on what a DSS is, since DSS can be approached from different points of view. Alter [2] categorizes DSS in terms of the generic operation they perform, independent of type of problem, functional area, decision perspective, etc. From this viewpoint, he distinguishes seven distinct types: file drawer systems, data analysis systems, analysis information systems, accounting models, representational models, optimization models and suggestion models.
Other approaches define DSS in terms of the design characteristics of a DSS, which may require a different approach to its design than the traditional approaches to analysis and design have provided. This requires a more complex, more sophisticated approach to decision making, and because of the problems that this approach faces, it is often used by decision makers. A DSS needs to be built iteratively, following a set of guidelines, and the use of this new approach will change the way the user looks at the problem, and may even change the way he or she uses the system. As a result, the user's process of decision making is likely to change, and the user may have to make some new decisions.

1. A DSS provides support for decision making through computerized and structured data and decision models.

2. A DSS is an interactive, non-machine system.

3. During the decision making process, a DSS is accessible to users as an interactive, non-machine system.

4. A DSS is built to support a given class of decisions.

5. A DSS supports all phases of the decision making process.

For the purpose of this paper, we will focus only on two issues. First, we will consider the functions of a DSS. This means that these functions must be supported and the users must be able to understand them. Second, we will examine the techniques used to define, design, and develop a DSS. These approaches will be useful in developing and designing a DSS.

The functions and techniques for designing and developing a DSS can be divided into two categories: the design and the implementation. The design category includes the methods and techniques used to design a DSS. The implementation category includes the methods and techniques used to construct and maintain a DSS.

The design of a DSS involves selecting appropriate data and models, and determining the structure of the system. The implementation of a DSS involves determining the structure of the system, and the methods and techniques used to analyze and design the software.

The design of a DSS involves selecting appropriate data and models, and determining the structure of the system. The implementation of a DSS involves determining the structure of the system, and the methods and techniques used to analyze and design the software.

The design of a DSS involves selecting appropriate data and models, and determining the structure of the system. The implementation of a DSS involves determining the structure of the system, and the methods and techniques used to analyze and design the software.

The design of a DSS involves selecting appropriate data and models, and determining the structure of the system. The implementation of a DSS involves determining the structure of the system, and the methods and techniques used to analyze and design the software.

The design of a DSS involves selecting appropriate data and models, and determining the structure of the system. The implementation of a DSS involves determining the structure of the system, and the methods and techniques used to analyze and design the software.
that has a large impact on the objectives of the firm. Thus, they are the first candidates for computer support, since an improvement in these decisions is likely to imply an improvement in the effectiveness of the firm.

Four decisions were selected, which we briefly describe below. It must be noted that is was not easy to identify them. The reason for this is that both managers perceive their activity as being difficult for them the separability of the decision variables. This leads to a detailed analysis of subactivities rate or or to analyse it into smaller subactivities. However, after some discussion an agreement was reached.

1. **Inventory management.** This consists in the selection of an inventory policy concerning when to order an item and the amount to be ordered.

2. **Investment.** This decision must be made when a vendor offers a product at a lower-than-usual price or when the managers forecast a rising in vendor prices.

3. **Promotion.** It is a usual practice in the market to promote some products weekly. This is done by offering them at a lower price, usually with minimal order sizes. This decision selects the products to be promoted and determines their price and minimal order size.

4. **Elimination.** This decision attempts to forecast which products are expected to be obsolescent in the short term, in order to get rid of them as soon as possible.

In making each of the above decisions, the managers must keep many interrelated factors in mind. Moreover, some of these factors are not easily quantified, and in some cases they must be presented by a probability distribution.

For instance, in the inventory management decision the ideal policy should consider, for each item, at least the following factors:

1. The future demand for the item. It should be represented by a probability distribution. In many cases, different item demands are not independent of each other.

2. The lead time (or delivery lag) between the moment the replenishment action is initiated and the moment the item becomes available. This lead time can be deterministic or probabilistic, depending on the situation (vendor, items ordered, etc.).

3. The setup or reorder cost.

4. The inventory holding cost.

5. The penalty cost whenever an item is out of stock.

6. The quantity discounts offered by the vendor. These discounts can be offered taking as basis the quantity ordered of the item, the total amount of the items ordered in an order, or the total amount ordered in a year for all items supplied by the vendor.

The task of estimating reordering, holding, and penalty costs is quite difficult. Nevertheless, the problem is not insurmountable if only approximate and reasonable estimates are sought.

It is more difficult to deal with factors represented by probability distributions. Problems arise in our case because the managers ha-
distinct sets of values. This function presents the characteristics of the computed solutions to the managers.

7. Select an alternative. After the solutions, the managers can select one of them. This function allows to store the selected decision in the database, for use in the normal operation of inventory control and replenishment. The items stored are reorder point and reorder quantity.

8. Store the estimations. The quality of the decision depends on the quality of the estimations. For this reason, it is important to store the estimations and to compare them later with the actual values. If the differences are important then it may be necessary to update the policy for the product and perhaps to review the assumptions underlying estimations. This function does not exist in many published DSSs. However, as Ackoff [1] points out we consider it quite necessary in order to be able to adapt to changing situation and to learn from experience.

9. Help. This function includes several usual components. The most important is a program that managers may use as a teaching aid. The program allows them to compare the solution computed by the model with the managers' own solution. They can also analyze the change in the results when changing any value. This program was used by the managers in the early stages of design in order to fully understand the model, its principles and the influence of the variables in the result.

3. IMPLEMENTATION

3.1. Structure of a DSS

Figure 1 depicts the conceptual structure of a DSS [9, 12, 13]. This conceptual structure includes three main subsystems: data base, decision models and user-system interface. These three major subsystems provide a convenient scheme for identifying the technical capability which a DSS must have.

Figure 1. Conceptual structure of a DSS.

The data base subsystem is thought to be a well understood set of capabilities because of the rapid maturing technology related to data bases and their management. There are, however, some significant differences between the data base approach for traditional systems and those applicable for a DSS. First is the importance of a much richer set of data sources than are usually found in typical non-DSS applications, since typical oriented transaction data must be supplemented with non-transactional data. Another is the importance of the data manipulation process from this wider set of data bases, since it must be flexible enough to allow rapid additions and changes in response to unanticipated user requests [12].

The decision model subsystem consists of the decision models, standard models and model builder. Standard models consist of a set of modeling tools for analysis or modeling purposes. In general, they are not complete models, but rather, model building blocks in the form of management science type algorithms and routines. They can be used separately or jointly.

The model builder subsystem is the element of the DSS that is used to build or to refine the models. Its main component is a model definition language, such as a planning language, which is used originally to build the models and to refine them to correct deficiencies in the original design.

The user-system interface subsystem provides the mechanism for interaction between the user and the DSS. Actually, there should be at least two levels of interaction. At the first level is a "user language" which directs the operation of the major elements of the DSS. It is the language that is utilized when choosing which model to run, which data to examine, etc. The second interaction level is a subsystem level language consisting of the commands required to control the detailed operation of a particular system element [13].

3.2. Implementation in a microcomputer

The four DSSs described in the previous section have been implemented in a CBM 3032 microcomputer, with 32 Kbytes of main storage and 350 Kbytes of auxiliary storage (disk drive floppy disk). The DSSs have been programmed in BASIC. The system has not a data base management system (DBMS). However, the vendor provided some data management facilities, such as file definition, indexed file organization, file updating, sorting and report production.

The above hardware-software system is not the best suited for the intended applications, but it was the only one available.

Although the microcomputer can perform most of the same tasks that a larger computer handles, the limited storage, the lack of a DBMS and the language used make the task more difficult for the micro and cause it to take longer. In the following we discuss in some detail the main difficulties.

1. Limited auxiliary storage

With the 350 Kbytes available for auxiliary storage, it was not possible to keep on-line both the programs and the data. This is a problem because it difficults the change from a program to another. In our case this problem was not too serious since we have four DSSs and it is not very frequent to shift the files from one to another. However, in other cases it may be a strong limitation.
2. Lack of DBMS

There are some DSS's that do not use a DBMS. However, a DBMS is a very convenient element of a DSS, since it provides a flexible, extendable and easy to use way to the access and manipulation of data. The lack of a DBMS implies that the programmer must program the routines to access the data and that he must change them when the file structure is changed. This means more programming time.

3. Lack of an inquiry language

The data manipulation element of the database subsystem (figure 1) offers many capabilities to the decision maker. One of them is an inquiry language that enables the user to obtain any information of the data base. This is a very important capability since it allows the user to formulate any unanticipated query to obtain the information required in his decision making. The only way to partly compensate the absence of an inquiry language is to include in the program the routines required to answer some fixed set of queries. This is not the ideal solution, but it is the only one that can be provided.

4. Modularity

As indicated above, we used BASIC as programming language. Although BASIC has the properties of a high-level language, the version that was used does not have the capabilities required for modular programming. It is difficult to build an overlay structure since it is not directly possible to have a single main program and several subroutines. Moreover, it is not easy to pass parameters between modules. Both problems make it difficult to build modular programs, which is a necessary requirement for a DSS to be extendable and flexible. However, these problems and the storage limitation forced us to build a monolithic program instead of a modular one.

4. CONCLUSIONS

Experience gained from the development reported here makes clear that DSS is a meaningful concept for small firms and that DSS can be implemented in small computers. DSS theories and practices developed for larger firms and larger computers are also applicable here. However, there are some differences since in general managers of small firms are less professional than managers of large firms. Furthermore, it is more difficult to implement a DSS in a small computer than it is in larger ones.

REFERENCES


