Chapter 3 - Constructing a DSS

3.1 Building the DSS

In building a specific DSS (SDSS), the iterative design process seems to be the most appropriate because of the need for flexibility and the short development cycle needed by decisions and decision-makers (Sprague & Carlson 1982). Flexibility can be viewed as the ability of the SDSS to respond to changes in user decision-making processes as well as the ability to easily develop the specific DSS. An iterative design compresses the traditional levels of the system life cycle to generate repeated versions of the SDSS. It is facilitated in the use of a DSS generator, which reduces development time for the SDSS. The result of applying the iterative design process at all levels is an adaptive DSS capability.

The key aspects of iterative design are (Sprague & Carlson 1982):

- Focus on a sub-problem
- Focus on a small, but usable SDSS
- Plan for refinement/modification cycles, and
- Evaluate constantly

3.1.1 The user

This is the final component of a generic DSS that influences the way a final decision is reached. Differences exist in users' cognitive preferences and abilities and the way they arrive at a decision. The user is also known as the manager or decision-maker. Knowing who will use the DSS is important in the designing of it. Individuals can use a DSS for personal support, or they can individually use a DSS or a portion of a DSS in organisational or group support. A new dimension, namely how a group works together is introduced when decisions need to be made collectively. This complicated type of DSS is called a group DSS (GDSS).

3.1.2 Custom-made versus ready-made DSS

When a problem is non-routine and not structured, the DSS needs to be custom-made for the organisation. When similar functional problems exist in different organisations, a generic DSS can be built with the option of some modifications. Such a DSS is called a ready-made DSS. Most DSS are custom-made though.

3.1.3 DSS technology levels

Sprague & Carlson (1982) identified three levels of DSS technology: specific DSS (SDSS), DSS generators and DSS tools (also referenced by Turban 1995). DSS tools are used to construct DSS generators, which in turn are used to construct SDSS. The DSS tools may also be used to construct tools that are more complicated. Using a DSS generator saves time and money, making the DSS financially feasible.

♦ **Specific DSS**

Systems that actually accomplish the work are called a specific DSS (SDSS). A SDSS involves an application that allows a specific decision-maker or group of them to deal with specific sets of related problems. The case study (See Paragraph 4.4: p53) could be viewed as a SDSS, because it addresses a specific decision problem for a specific group of decision-makers.

♦ **DSS generators**

A generator is an integrated package of software that provides a set of capabilities to build a specific DSS quickly, inexpensively and easily (Turban 1995). A DSS generator is an integrated easy-to-use package with diverse capabilities ranging from modelling, report generation, graphical presentation to performing risk analysis. The ideal DSS generator may be a special-purpose language. The special-purpose language may be used to build a DSS application easily or as an integrated software system constructed around spreadsheet technology.

♦ **DSS tools**

This is lowest level also called the fundamental level of DSS technology and consists of software utilities or tools. These elements facilitate the development of a DSS generator or a SDSS. Examples include graphics, editors, query systems, random number generators and spreadsheets: elements used to build both SDSS and DSS generators. A specific DSS may be built directly from tools. DSS generators promise to create a platform from which SDSS can constantly be developed without much consumption of time and effort (Sprague & Carlson 1982).

3.1.4 Factors to consider when designing a DSS

According to Mallach (1994), one should consider the following before starting to design a DSS:

- One should first determine the purpose of the DSS in terms of the decision being made and the outputs it must supply
- One should determine any external sources that the DSS will communicate with and find any data flows to and from these sources
- Any internal data files needed should be determined. One should determine if the data in these files are obtained from external data sources and if it is, specify the external sources, and
- The major processes in the DSS should be determined. If one can understand all these considerations, you will understand your DSS as a system. One test of this understanding is being able to draw it as a flow diagram. A general schematic description of a DSS is shown as a data flow diagram in Figure 2-8 (p23).

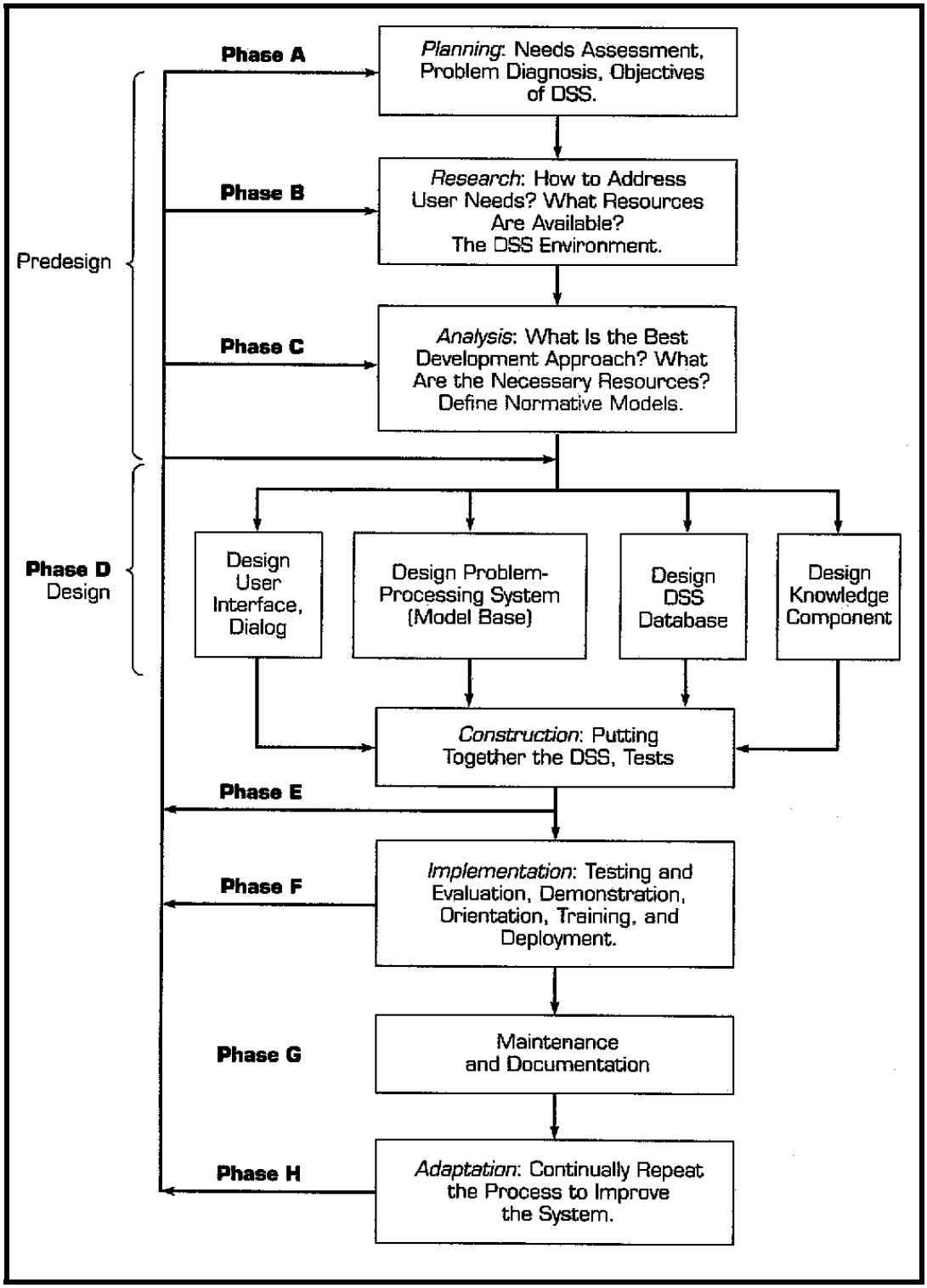

Figure 3-1 Phases in building a Decision Support System (Turban 1995)

3.1.5 Creating a DSS Generator

The DSS generator should have three general capabilities (Sprague & Carlson 1982):

- It should be easy to use:
  - The generator should create a SDSS that is easy and convenient for non-technical people to use in an active and controlling way, and
  - The generator should be easy and convenient for the builder to use for building and modifying the specific DSS
- The DSS generator should provide access to a wide variety of data sources in a way that supports problem solving and decision-making for a variety of users, problems and contexts, and
- The DSS generator should provide analysis to support problem solving and decision-making for a variety of users, problems and contexts. The generator should provide suggestions on request.

3.1.6 Classical development life-cycle versus prototyping

A classical DSS development process, including all activities necessary for the construction of a complex DSS, is given in Figure 3-1 (p34). As mentioned earlier, the iterative design process seems to be a better alternative because of the flexibility as well as the short development cycle needed by decisions and decision-makers (Sprague & Carlson 1982). A prototyping process often clears misconceptions between builders and end-users. Mutual learning occurs as the DSS develops.

Prototyping provides the following advantages (Turban 1995):

- Short development time
- Short user reaction time (feedback from the user)
- Improved user's understanding of the system, its information needs and its capabilities, and
- Low cost

3.1.7 The development process of a DSS constructed by end users

When end-users are allowed to modify a SDSS to suit their decision needs or the decision needs of the group that they represent, it is better to follow a construction process from the end-users point of view. A suitable construction process developed from an end-users point of view is given by Turban (1995):

- Phase one – *Choosing the project or problem to be solved*: Departments involved are committed to the process of finding a suitable solution
- Phase two – *Selecting software and hardware*: Select suitable DSS software and hardware
- Phase three – *Data acquisition and management*: Acquire and maintain data in the knowledge base
- Phase four – *Model subsystem acquisition and management*: Build the model base: acquire and include relevant models in the model base
- Phase five – *Dialogue subsystem and its management*: Develop the user interface

- Phase six – *Knowledge component*: Perform knowledge engineering
- Phase seven – *Packaging*: The various software components of the DSS will be put together for easy testing and usage
- Phase eight – *Testing, evaluation and improvement*: Test the DSS with sample input and validate it to prove that the DSS is reliable
- Phase nine – *User training*: Train users in using the DSS
- Phase 10 – *Documentation and maintenance*: Produce documentation and maintain the DSS, and
- Phase 11 – *Adaption:* Adapt DSS to suit user needs

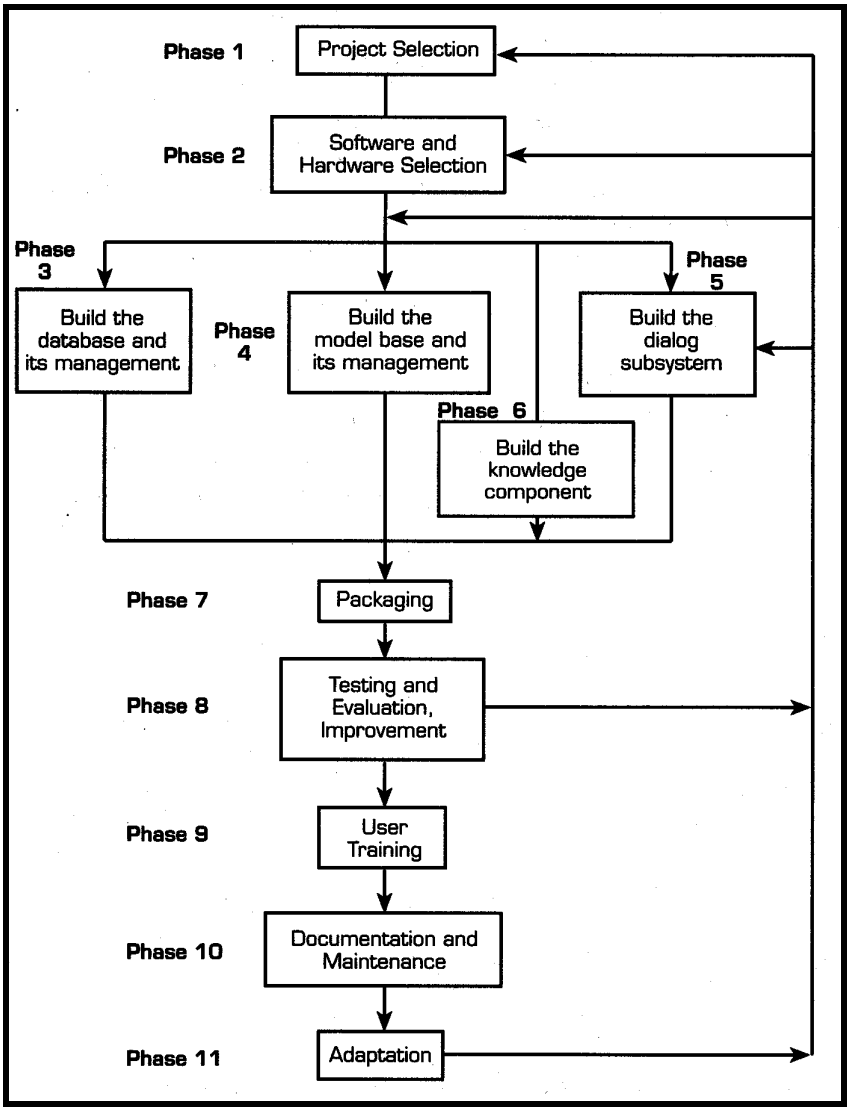

Figure 3-2 The development process of a DSS constructed by end-users (Turban 1995)

3.2 Building an ES

Before starting an ES project many factors should be studied such as the main goal of the system; its constraints; its available support facilities; availability of human experts; user-imposed reliability; maintainability; solution needs; and application needs. Updating and maintaining the knowledge base and enhancing the capability of the inference engine are central to a successful ES (Raggad & Gargano 1999). When the ES is used, the effectiveness of the ES should be continually evaluated and monitored to test if the problem domain has not changed. If the problem domain changes, it can invalidate the recommendations given by the ES. This last aspect will be explored in more detail in the section on knowledge validations (See Paragraph 7.2.1: p130).

3.3 Building a KB-DSS

Building a KB-DSS involves the capabilities, functionality and structures of both DSS and ES with the emphasis on the support of DSS. Factors discussed in Paragraphs 3.1 (p32), 5.5 (p101), 6.3 (p121) and 6.4 (p122) are of relevance. Klein & Methlie (1995) propose a methodology for the KB-DSS implementation process in Figure 3-3 (p40). This design methodology is related to the learning process of users and supports flexible design strategies. It presents the essential characteristics and allows the designer to use and combine a variety of design strategies. It is possible for the user to start with the models, the database, the knowledge base, or by defining the application with displays of results. Klein & Methlie (1995) suggest starting with the user interface and the global logic of the application and its associated variables, adding the other sources as the designer wishes as a function of the application. The methodology for Klein & Methlie's (1995) KB-DSS implementation process includes:

♦ **Understanding the user's goals**

The usual user goals are to

- Recognise a problem situation
- Diagnose a problem
- Generate alternatives
- Compute criteria
- Evaluate alternatives, and
- Select one alternative

These goals can more generally be to improve the way the problem is presently solved or facilitate communication between individuals to ease the reaching of a solution.

♦ **Understanding and defining the problem boundaries**

Understanding and defining the problem boundaries will identify:

- The decision-makers
- The relationship of the decision-maker with the decision structure of the organisation
- The fixed and controlled decision boundaries as accepted and challenged by the decision-maker

- The problem that will be solved if an alternative is chosen
- The willingness and ability of other decision-makers and experts that co-operate and provide inputs to the analysis, and
- The dominant culture in the organisation

This step should lead to the creation of alternatives.

♦ **Understanding and defining actual decision processes**

It is very unlikely that a large number of users will use exactly the same decision processes. The sub-steps for identifying the decision process are:

- Describe the general task within which the decision process occurs, and
- Describe the persons involved in the decision process and the sub-tasks they accomplish:
    - Domain knowledge used for studying the problem
    - Problem diagnosis methodology and task knowledge
    - Alternative generation
    - Facts and documents used to obtain criteria
    - Computation method of criteria
    - Presentation of criteria, and
    - Constraints to be taken into account

♦ **Define a normative decision process for the problem**

The task of the designer is to analyse the decision processes, make a diagnosis and define an improved process.

♦ **Defining changes in the decision process**

Many users will use the KB-DSS. The adoption of a DSS is a social process. Therefore, all the users should assimilate the decision process improvement.

♦ **Selecting which part of the decision process to support**

The starting environment of the user should be defined.

♦ **Functional analysis of the KB-DSS**

The purpose of this step is to define the main functions and overall architecture or conceptual model of the KB-DSS. Usually a first list of reports, decision models, data structures, input forms and knowledge bases are needed. A first layout of the application user interface should be defined to demonstrate the resources to be used during the problem solving process.

♦ **Selection of a development environment**

Klein & Methlie (1995) rendered five possible development environments for designing DSS, ES or Knowledge-Based Decision Support Systems (KB-DSS):

- Standard third and fourth generation programming languages such as VB, DELPHI, PASCAL, BASIC, C, Java
- AI languages: such as LISP PROLOG, or Smalltalk
- Expert System shells – no DSS function supplied
- DSS development environments – no expert function needed, and
- KB-DSS development environments – integrating DSS and ES

Choosing the development environment depend on the functionality needed (Klein & Methlie 1995)

- Graphical user interface environment: A graphical object orientated environment allowing the user to use icons, menus and other graphical components. It allows for a global logic interaction between the application user and the application resources.
- Report generator: Allows the user to define reports interactively integrating various objects
- Modelling language: Present a modelling formalism to assist the decision-maker
- Form definition: Input should be entered using a variety of controls and report the explanation provided by the intelligent part of the system
- Database management system: A link to the database when necessary
- Knowledge base Management System: The key issue here is to match the knowledge presentation method and the knowledge to be presented
- Toolbox: Algorithms that are useful for the planned application such as in finance, statistics and forecasting , and
- Communications interface, local area network (LAN) and client/server architecture: Matching the communication needed with the hardware required, and

♦ **Design and implementation of the initial KB-DSS**

The steps proposed by Klein & Methlie (1995) include:

- Data analysis and modelling
- Form definition and input verification
- Decision model design and testing
- Report definition
- Knowledge base modelling and testing, and
- Overall user interface design and global application logic definition

According to Sprague & Carlson (1982), the iterative design process seems to be the most appropriate because of the need for flexibility and the short development cycle needed by decisions and decision-makers. Prototyping the steps would support the iterative design process. It is essential to test the KB-DSS thoroughly. The decision models and the knowledge bases need to be tested on completion with the users that took part in the design. A methodology for verification and validation is presented in

Paragraph 7.3 (p134). It is good practise to write the user manual during the testing and evaluation phase. The education of the users is largely dependent on their implication in the design, their level of expertise and the purpose of the KB-DSS.

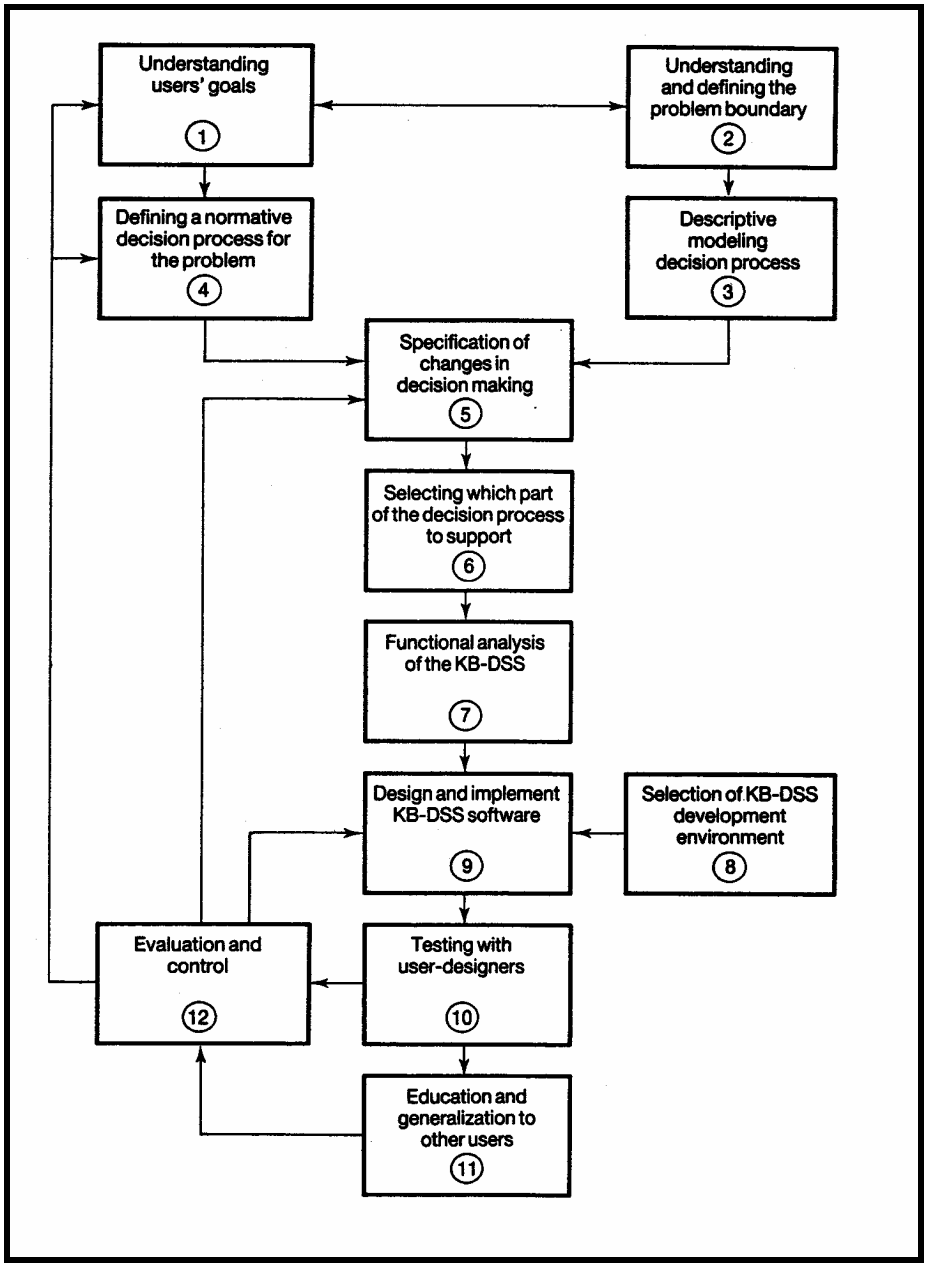

Figure 3-3 Methodology for the KB-DSS implementation process (Klein & Methlie 1995)