The use of system development methodologies in the development of decision support systems: an interpretive study

J.P.S. Ellis

12331368

Dissertation submitted in partial fulfilment of the requirements for the degree *Master of Science* at the Potchefstroom campus of the North-West University.

Supervisor: **Professor H.M. Huisman**

November 2010

Acknowledgements

I would like to sincerely thank the following people:

- Rynetta and Catherine for their support, understanding, help and sacrifices they made throughout this study.
- Prof. Huisman for her guidance and patience.
- My family for their support, help and encouragement.
- My friends for their support.

Thank you Jesus, for I shall not want.

Abstract

The world we live in today demands systems that make our lives easier and help us make the right choices on time. There exists a growing need for quality products that help us in our day to day activities. Easy-to-use computer-based decision support systems apply all available and applicable data with the correct model, knowledge and skill of decision makers to support the user to choose the best solution. It is therefore important to develop decision support systems correctly to be of value to the user. Looking at other information system developments, the author tries to suggest ways to develop decision support systems. System development methodologies are investigated to determine if they are able to address the development of the very important decision support system components. Five methodologies were discussed and researched for their theoretical suitability to address the development of decision support systems. The author performed qualitative research using case studies and semi-structured interviews to assess the use or non-use of system development methodologies in the development of decision support systems in a South African context. Content and cross-case analyses were used to achieve results that are discussed to broaden the knowledge on the development of decision support systems. The author provides some explanations to why system development methodologies were not used in the development of the case studies. This research not only contributes to the academic body of knowledge about using system development methodologies in the development of decision support systems, but could also be useful to developers embarking on a new decision support system development.

Key terms: System development methodology, decision support system, interpretive study.

Opsomming

Ons hedendaagse lewens vereis stelsels wat ons lewens vergemaklik en ons help om betyds die regte keuses te maak. ’n Groeiende behoefte bestaan vir kwaliteit produkte om ons in ons daaglikse take te ondersteun. Besluitsteunstelsels is rekenaarprogramme wat maklik is om te gebruik. Dit pas die korrekte model, kennis en vaardighede van besluitnemers toe op alle beskikbare en toepaslike data. Daardeur help dit die gebruiker om die regte keuse te maak. Besluitsteunstelsel moet reg ontwerp en ontwikkel word om van waarde te wees. Die skrywer stel ontwikkelingstrategieë vir besluitsteunstelsels voor deur na die ontwikkeling van ander inligting stelsels te kyk. Stelselontwikkelingmetodologieë word ondersoek vir hul bruikbaarheid in die ontwikkeling van die belangrike komponente van besluitsteunstelsels. Vyf metodologieë word bespreek en ondersoek vir hul geskiktheid om gebruik te word in die ontwikkelingsproses van besluitsteunstelsels. Die skrywer het kwalitatiewe navorsing gedoen en van gevalle studies en semi-gestruktureerde onderhoude gebruik maak om die gebruik van stelselontwikkelingsmetodologieë in ’n Suid-Afrikaanse konteks te ondersoek. Resultate is verkry deur die inhoud van die onderhoude te analiseer asook deur ’n vergelyking van die gevalle studies. Hierdie resultate dra by tot die kennis oor die ontwikkeling van besluitsteunstelsels. Die skrywer gee moontlike verduidelikings waarom stelselontwikkelingsmetodologieë nie werklik gebruik was in die ontwikkeling van die gevalle studies nie. Hierdie studie dra nie net by tot die akademiese kennis oor die gebruik van stelselontwikkelingsmetodologieë in die ontwikkeling van besluitsteunstelsels nie, maar kan ook van waarde wees vir ontwikkelaars van nuwe besluitsteunstelsels.

Sleutelterme: Stelselontwikkelingsmetodologie, besluitsteunstelsel, interpretatiewe studie.

Contents:

Contents (continued)

Contents (continued)

Chapter 1: Introduction

In the first chapter the author introduces the reader to the study on the use of system development methodologies in the development of decision support systems. The chapter starts by stating the problem as well as the reason for the study. The reader will be briefly introduced to the topics of decision support systems and system development methodologies and conclude why it is necessary to do research on this subject. The author states in this chapter what the aim of the study was as well as the way the research was conducted. The planned contributions to the body of knowledge are stated and the author gives a brief outline of the following chapters.

1.1 Problem description and motivation

Our lives as humans are filled with decisions. We live in the era of the information super highway. We have the ability to use information to achieve our goals or improve our lives. Not only do we need information to make decisions, but we need to interpret the information, and use it in the best way possible. The problem is that we first need to make sense of it all.

With the technological advancements, we are faced with new decisions everyday, decisions we as humans did not need to make a few years ago. Nowadays we are faced with decisions like the power setting of a microwave oven for a specific dish or the time needed for our laundry in the washing machine. Some of us need to decide which stocks to invest in on the stock exchange to maximise our profits and so, ensure our retirement. We have to choose a cell phone provider, a cell phone package and also a cell phone that should best fit our life style. These decisions are all examples of choices and decisions our ancestors did not need to make a few years back. What makes it more difficult, is the fact that it seems that the number of choices grows exponentially.

Decisions become more complex as the factors to consider during decision making increase, or the impact of the decision grows. Sometimes there is so much data available we don't know how to use it in decision making. In these cases, we need a tool or program to help us decide.

Decision Support Systems (DSS) were first developed in the 1970's. Decision Support Systems will be fully defined in Chapter 2, but for now, it will be described as easy to use computer programs that help people make decisions by using the available data and applying specific models to the data. These models can be mathematical equations, heuristics or knowledge and skill of experts on similar decision problems. The goal of decision support systems is to help the decision maker in making the best decision in a short time using the available data. DSS are usually computer systems people can use.

Examples of DSS include scheduling problems in the transport industry. DSS are helpful in determining the shortest route for a delivery truck or the optimal scheduling or passenger flights for an airline. DSS are helpful in the retail sector and can determine the optimal number of service points under the constraint of optimising profit, that is lowering cost of operational service points, and keeping clients happy, that is keeping queues short. These are just some examples how DSS are being used to improving our daily lives.

It seems almost obvious that DSS should be part of our lives. In fact, we might have expected to be more aware of DSS in our lives. Why don't we use DSS all of the time for all of our decisions? Why wouldn't DSS be the best discovery and most useful invention? One answer might be that DSS are only developed to assist the decision maker in making the best decision. The user is still the decision maker and as long as we don't have perfect information about our universe, we are destined to make wrong decisions.

Another reason might be that DSS are developed for problems that do not have only one solution. How to solve the problem might not really be known, and the problem that needs solving might change before a recommendation to a solution is given. How to achieve peace in a conflict region is a problem without a simple one word answer. The situation might change before all the variables have been identified, never mind entered into a computer program. There would possibly be no historical events or data applicable to the current situation.

Lastly, another reason why some might think that DSS are not useful is because DSS are computer programs that need to be developed. Software are notorious for either not being user friendly, not suited to all the user requirements, being very expensive and are more times than not, delivered late and developed beyond budget.

To summarise: DSS operate in complex environments. It might therefore be difficult to build a decision support system for a specific problem. The development process itself can create a system that users find difficult to use, not completely applicable to their problem or just too expensive to justify acquiring the system.

Can we find the best way to develop DSS to ensure the DSS are used as they should be by the intended users? Can we suggest the best way to develop DSS to ensure that the development stays within the budget and the timeframe but still producing a quality product?

1.2 Research aims

This research sets out to find out how DSS are developed and why they are developed in such a way. It seems that the size of the organisation and the specific problem the decision support system is developed for, might determine the type of decision support system that will be developed. Some DSS are integrated into organizations and used with their existing information system. Some DSS are developed only to solve a problem once and then the decision support system becomes obsolete. Some DSS might be developed as small systems designed to grow with the organisation and its problems. Each one of these scenarios might suggest a different way to develop the decision support system. Factors like available time and budget might also play a role in how the DSS are developed.

It is therefore clear that not all DSS are developed in the same way. Before we look at how to develop a decision support system, the author looks at what to develop. DSS components are identified in Chapter 2 as consisting of users, user interface, hardware, data, models and an intelligence component. The DSS characteristics are also listed in Chapter 2. The author suggests that the way DSS are developed and the development itself should address these components and characteristics.

Systems Development Methodologies (SDM) can be used in the development of information systems. SDM are fully defined in Chapter 2. Huisman (2000) describes how SDM entail using system development techniques in the development of the system. System development techniques form part of system development methods that also consists of guidelines, activities and tools. These are based upon a particular philosophy on systems development. The philosophy forms part of the systems development approach that is principals and underlying beliefs, goals and fundamental concepts to system development. That means certain beliefs or thoughts on how systems should be developed. The systems development process model is the sequence of stages through which development takes place. In other words: When to do what. The philosophy determines the applicable model.

In other words, SDM provide developers with ways to develop systems. The steps, techniques and methods to be used in the development are included based upon a belief or philosophy on how information systems should be developed. The aim of using SDM is that SDM should help developers address all the necessary aspects of system development. SDM were developed to improve development and user acceptance of final products. Then surely SDM should be used in the development of DSS?

The author sets out to find out if SDM can be used in the development of DSS. If SDM can be used, which one of the many SDM would be the best to use? Do real developers actually use SDM in the development of DSS? If they do use SDM, does it improve the quality of the DSS? Why would developers not use SDM?

Currently not much is known about the use and the effectiveness of SDM in the development of especially DSS. Even less is known about the use, or non-use of SDM in DSS development in a South African context. Through this study, the aim is to better understand the development process of DSS.

The author identified possible SDM approaches from Avison and Fitzgerald (2003). The author looked for SDM to address the development of each component of DSS. The approaches and five SDM chosen are described in Chapter 2. The author suggested Soft Systems Methodology, Extreme Programming, Information Engineering, Structured Systems Analysis and Design Method and CommonKADS.

The aim of the research is to:

- Evaluate the selection of SDM for their theoretical suitability to develop DSS.
- Describe the use, if any, of SDM during DSS development.
- Explain the reasons for the use or non-use of SDM during DSS development.
- Evaluate the influence of SDM on the quality of DSS.

1.3 Method of research

The author needed to find the best way to research the development of DSS. Research is based upon a paradigm or a way of thinking about the world. Probably the most popular and well known research paradigm would be positivism. Positivistic research wishes to make generalisations about the world regardless of the situation or actors involved, Oates (2006).

The assumptions that the world is regular and ordered with no random events and that we can investigate the world objectively makes this philosophy not very suited to this research. Other philosophies exist because not all researchers believe that problems can be reduced into smaller parts, that actions are predictable, and that generalisations are not always possible.

Interpretive research tries to understand the social context. The aim is more to describe behaviour than to explain behaviour. This implies that there might be more than one perception or interpretation. Knowledge is transferred through social constructs and the interpretations thereof may differ. The research setting may also differ each time and could influence the results.

Critical research is another research paradigm that additionally focuses on power relations and conflicts and aims to empower people.

These paradigms and their characteristics are defined and described in Chapter 3. The chapter also distinguishes between qualitative and quantitative research. The research in this study is based on the interpretive paradigm and therefore uses qualitative data.

Oates (2006) suggests different research strategies that can be used for each paradigm. The author discusses in Chapter 3 the case study as research strategy. Attention is also given to interviewing as a data collection method. This research used semi-structured interviews to obtain data that was transcribed, coded and analysed in tables. The process is described in Chapter 3 and the tables are available in appendix A and appendix B.

The interpretations of the author are given in Chapter 4. Each interview was coded and tabled. The author looked for similarities and contradictions in the answers of the interviewees. The results and propositions from these interpretations are presented in Chapter 5.

Chapter 4 contains insights on how the developers see DSS. The developers explain how they think DSS differ from other information systems, what the characteristics of DSS are and what the DSS components are. The author compares this to the literature study in Chapter 2.

Following this is interpretations on the way DSS were developed. What part of the development was difficult, what would be possible approaches to DSS development, the development process and techniques and tools used in the development? How long the development took is described and what role documentation played in the process. Developers included the importance of clients and experts in the discussions. The author describes factors that may determine how DSS are developed.

Chapter 4 also contains a discussion on the use of in-house developed tools, the life span of DSS, the testing within the development process and the use or non-use of methodologies.

Chapter 5 concludes with the findings of the research. The theoretical suitability of SDM to develop DSS, the actual use or non-use of SDM in DSS development, the reasons therefore and the influence of SDM on the quality of DSS are discussed.

The author states in Chapter 5 possible limitations and the contributions to the field and makes suggestions for future work.

1.4 Planned contributions

DSS were developed more than 30 years ago. Yet, so many DSS are never used, or never used as was intended. The author reviewed the literature on DSS, IS development and SDM to find possible ways to develop DSS and make suggestions regarding the use of SDM in the development of DSS. The author not only aims to contribute to knowledge by suggesting possible development strategies but also by providing information regarding the development of DSS in practice.

1.5 Contents of dissertation

The author included the following chapters in the study:

- **Chapter 1: Introduction**
  The chapter briefly introduces the reader to the topics of decision support systems and systems development methodologies. The problem is stated as well as the research objectives. The method to research the topic is briefly discussed and the planned contributions listed.
- **Chapter 2: Literature study**
  This chapter represents the knowledge gained by reviewing the literature on DSS, DSS development, and system development methodologies. Five methodologies are discussed and evaluated for their suitability for DSS development.
- **Chapter 3: Research design**
  The chapter describes the philosophy on which the research was based as well as the research approach. The author explains what type of data was used in the research and how the data was collected and analysed.
- **Chapter 4: Results**
  The results of the data analysis are given in Chapter 4. The author explains and motivates propositions formed by analysing the data.
- **Chapter 5: Conclusion**
  The chapter lists the research objectives and the corresponding findings. The author explains the connections between the propositions of Chapter 4 and the findings of Chapter 5. The contributions to the field are listed together with the limitations of the study. The author also makes suggestions for future work.
- **References**
  References used in the study and discussions are listed in the back.

1.6 Summary

The author outlined the study in Chapter 1. Some theoretical background on DSS, DSS development, and SDM were presented in this chapter. The author stated the research objectives of the study and explained what research method would be followed. The planned contributions are listed, as well as an outline of chapters to come. In the following chapter, the author will explain in detail the information collected by reviewing the literature on DSS, DDS development, and IS development using SDM.

Chapter 2: Literature study

This chapter presents the information discovered by reviewing the literature on DSS and system development in a quest to determine what would be, according to the literature, the best way to develop DSS.

It is necessary to first define DSS. By defining DSS, the components and characteristics that make DSS different from other information systems, are identified. The author wanted to know if these components and characteristics would imply that the development of DSS then would be different from the development of other systems.

The development strategies of systems development and System Development Methodologies (SDM) were studied to learn more about the development of information systems.

Literature was researched to find methodologies that could address the development of these specific components and characteristics. These results are given in the last part of the chapter.

The following paragraph gives the reader insight into why DSS exist, and the type of decisions DSS were developed to support.

2.1 Decision Support Systems (DSS)

DSS were developed after Management Information Systems (MIS). MIS produce summary reports and in this way increase the efficiency of managers. These reports are based upon structured information collected or generated by the data processing system. MIS are still popular in organizations and its biggest benefit lies in its ability to cope with routine functions and structured problems (Sauter, 1997). Unfortunately, not all decisions are based upon routine reports and not all problems are structured.

DSS are effective in semi-structured or unstructured problems and in situations where the outcomes are unsure. Well-structured processes refer to decision making processes that are routine and repetitive. Less structured problems have alternative solution methods. The solutions may not be equivalent. The solution methods for completely unstructured problems are not known or are too many to evaluate each one efficiently (Newell and Simon, 1972; Simon,1973; Adams *et al.*,1993).

This implies that MIS focus on providing information while DSS aims to help with making decisions (Sauter, 1997).

DSS are especially useful in cases where a fixed goal exists without an algorithmic solution. There might be many ways to reach a solution and the way to reach a solution may depend on the user. The goal of DSS is to help the decision maker to evaluate and choose the best solution (Klein and Methlie, 1995).

Our lives are filled with everyday decisions. The complexity of the decision depends on the number of factors to consider during decision making, the impact of the decision, or a combination of both. DSS aims to help the decision maker by utilizing as much as possible information, using the correct and appropriate models to help the decision maker choose the best alternative according to predetermined goals.

Fazlollahi *et al.* (1997) state that in crisis situations humans tend to only consider a few alternatives and then quickly reach a decision. The quality of the decision can be reduced by limited analysis. Poor decisions can be the result of rejecting the correct course of action, accepting a wrong solution to the problem, solving the wrong problem or solving the right problem but too late.

Some problems are not well-structured. There might be more than one way to reach a solution. Choosing between solutions can be complex due to the number of factors to consider, or the impact a decision can have. Decision makers tend to consider only a few options and even the wrong way to analyse the options. For these reasons, DSS are developed to aid the user to make the best decision for a particular situation in the limited time available. The following paragraph defines these systems called DSS.

2.1.1 Defining DSS

It is not clear who developed the first DSS; therefore, there are no clear definitions of DSS. Currently there exist a number of definitions of DSS. Some of these definitions are included in table 2.1.

Table 2.1: DSS definitions

"A computer program that provides information in a given domain of application by means of analytical decision models and access to databases, in order to support a decision maker in making decisions effectively in complex and ill-structured (non-programmable) tasks".	Klein and Methlie (1995:112)
"A decision support system is an interactive system that provides the user with easy access to decision models and data in order to support semistructured and unstructured decision-making tasks". It is recommended that management support and users is included in the development process. The development process should be evolutionary and iterative.	Watson HJ *et al.* (1997:263)
"DSS are software products that help users apply analytical and scientific methods to decision making. They work by using models and algorithms from disciplines such as decision analysis, mathematical programming and optimization, stochastic modelling, simulation, and logic modelling. DSS products can execute, interpret, visualize, and interactively analyze these models over multiple scenarios."	Bhargava (1999:31)
DSS are computer-based systems that bring together information from a variety of sources, assisting in the organization and analysis of information and facilitating the evaluation of assumptions underlying the use of specific models. Information is available from outside and inside the organization. The models may be simple summarizations of sophisticated mathematical models.	Sauter (1997)
DSS are information systems, designed to support decision makers by applying decision models to large collections of data.	Zwass (1998)

"The term DSS applies to information systems designed to help managers solve problems in relatively unstructured decision-making environments."	Meador *et al*. (1984:117)
DSS imply computer programs that: • Assist individuals or groups of individuals in their decision process; • Support rather than replaces judgements of individuals; and • Improve the effectiveness rather than the efficiency of a decision process.	Janssen (1992)
"A decision support system is an interactive system that provides the user with easy access to decision models and data in order to support semistructured and unstructured decision-making tasks".	Mann and Watson (1984:27)
DSS became characterized as: "interactive computer based systems, which help decision makers utilize data and models to solve unstructured problems".	Sprague (1980:1)

By looking at these definitions, the author would define DSS as easy-to-use computer programs that use data, models, knowledge and skill of decision makers to aid the user in decision making in complex and ill-structured problems.

The author used the term "computer program" in the definition because of the size of organizations and the amount of data and models available. Making the best decision in time would mostly imply using a computer. The data could come from inside or outside the organization, or even private data. To solve the problem the decision support system can use any sort of model: probably analytical, but also heuristics, skill and experience of experts in the field. In some cases, the user might be the expert.

DSS need not be scary super-computer programs that only a select few understand and can operate. The author describes the factors leading to the increased use of DSS in the following paragraph.

2.1.2 The increased use of DSS

A few decades ago, the thought of personal computers in family homes was far-fetched. Today, many homes have more than one computer. It is possible nowadays to travel with handheld devices that, with the aid of satellites, can direct us to any destination. Some models can even determine the route according to our preferences, for example the quickest or most scenic route. The application of DSS grows with the complexity of our world and our lives. It only makes sense that the use of DSS increases with the number and size of our problems. Sauter (1997) demonstrates in figure 2.1 the factors leading to the increased use of DSS.

Friendlier packages of information systems in general have led to greater user satisfaction and less computer anxiety. Users are prepared to try and use new developments in the information technology field.

Uran and Janssen (2003) claim that the reasons for the popularity of DSS can partly be found in the technological development, which makes it possible to install and use the systems on PCs, and partly in the need to manage large amount of often complicated data that play a role in the decision making process. Technology is part of most people’s lives. People are getting used to the idea of microwave ovens that can automatically determine the cooking time of the dish or cellular phones that act as navigation devices, music players, cameras, video recorders, game consoles all in one whilst still being the device that allows you to make phone calls, send messages and data, and surf the world wide web.

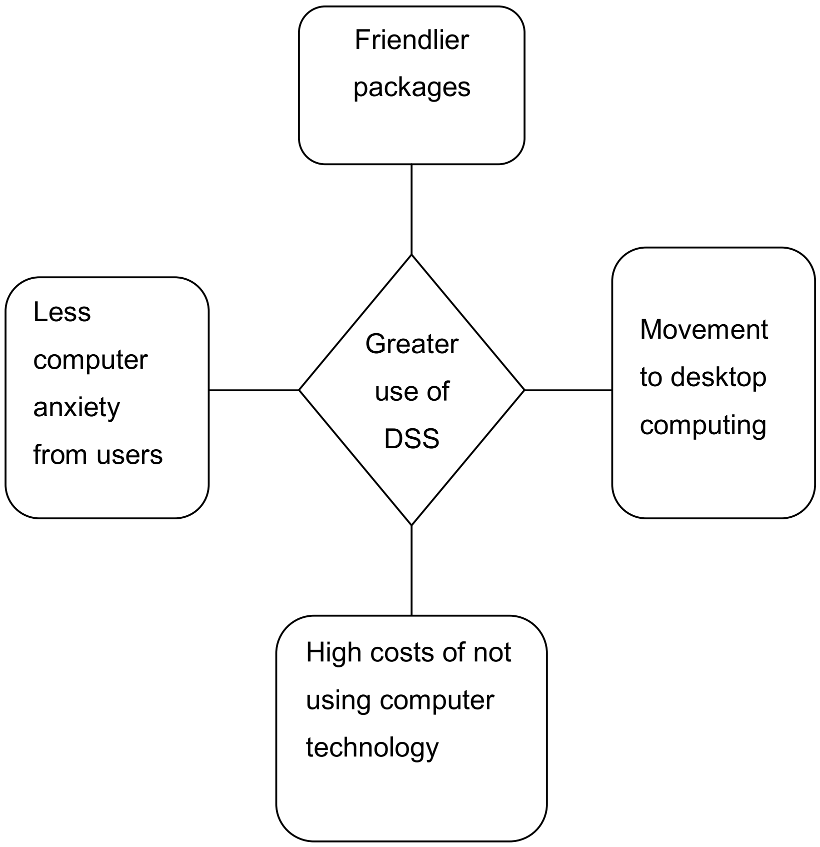
Friendlier packages
Less computer anxiety from users
Greater use of DSS
Movement to desktop computing
High costs of not using computer technology

Figure 2.1: Factors leading to increased use of DSS. (Sauter, 1997)

Mainframe computers are no longer a prerequisite for DSS or management support systems. Decision makers can use desktops and laptops to operate DSS. Meador *et al*. (1984) states that managers are more proactive in their interest and open to the idea that they can be hands-on themselves. The Internet makes it possible to communicate objectives over the globe. Computer hardware is not only getting smaller and more powerful, the technology is becoming more affordable. We now have hand held devices with more processing power and memory than some mainframe computers had a few years ago. This means that it is not just the most powerful and wealthy organizations who can afford the technology that are able to use DSS, but actually anyone with a personal computer.

Organizations are realizing how important it is to use computers. Fick and Sprague (1980) state that managers seek help from information technology because of the constant pressure to improve the efficiency and effectiveness of their organizations.

Operations are still possible without computers, but to compete without the aid of computers is to no avail. Meador *et al*. (1984) also state that the increased popularity of the DSS is partly because the economic climate is increasingly becoming more competitive, complex and uncertain. Technology makes it possible to communicate with almost anyone, anywhere, anytime. Data can be sent across networks and files can be shared on the Internet. Organizations that use the right technology can at anytime know how many products have been sold during a specific time, how many items are still in stock in any one of the warehouses, or even when to reorder and how many items to reorder from a supplier to minimize cost and maximize profits.

According to Bhargava (1999) the growing popularity of online analytical processing, data warehousing, and supply chain management has led to an increased interest in the development of DSS. Naturally, it would still be possible to run an organization without technology, or make important decisions without DSS, but will the organization still be as successful as it could be? Would the right decision, based upon all available data, using the most applicable model, still be made in time to be relevant? Shouldn't we be using DSS for all our decisions? We first look at the components and characteristics of DSS in the next section.

2.1.3 Components and characteristics of DSS

The components or building blocks of DSS are also components of other information systems. It is, however, the combination of these components that form DSS.

Decisions are based upon data. Decision makers use logic or models to make sense of the data in the form of information. In our world today, we can assume large DSS are computer based due to the vast number of data available and the limited time available to make and communicate decisions. This would mean that there is at least one user of the decision support system. It is important that the decision support system is easy to use and the user understands the components of the DSS. Making decisions does entail intelligence. Knowing what model to use and why one solution is better than others.

Therefore, we know that DSS should at least consist of the following components:

- **Data or Database management system**. Users should be able to access the data through the database management system (DBMS) without programming effort (Sauter, 1997; Turban, 1988; Duffy and Hough, 1985;

Shim *et al.,* 2002; Lu *et al.,* 2001; Westmacott, 2001). The data could come from inside or outside the organization. The Internet, Web and emails facilitate data transfer and data accessibility. The amount of data might be too overwhelming for other information systems or the management of this data may not be sufficient in other systems.

- **Models or Model Management system.** The modelbase management system (MBMS) acts in the same way as the DBMS but only for the models in the decision support system. The MBMS tracks all the possible models and the controls for running these models (Sauter, 1997; Turban, 1988; Duffy and Hough, 1985; Shim *et al.,* 2002; Lu *et al.,* 2001; Westmacott, 2001). The MBMS might contain modules for Linear Programming problems, for example modules for queuing problems, optimized cutting problems, and scheduling problems. The models can be mathematical or even simple heuristics.
- **User-friendly graphical interface for the users.** The user interface handles the input to system and output from the system (Sauter, 1997; Duffy and Hough, 1985; Shim *et al.,* 2002; Westmacott, 2001). The user interface is extremely important as it is the portal to the decision support system and sometimes the only component the user might use. The user's satisfaction with the interface will determine the willingness of the user to use the decision support system.

Kloditz *et al*. (2000) state that the user interface, if designed and used carefully, can guide the user through the various steps of the analytical procedure. The interface should not just present the results but rather help the user to achieve the results and understanding why steps are taken in the process. In other words the interface, like DSS itself, should help the decision maker, even in understanding why specific suggestions are made.

Moreau (2006) claims that if users believe a product is easy to use, they will be more open to the idea that the product can help them in the duties.

Keil *et al*. (1995) distinguish between perceived usefulness and ease of use. Perceived usefulness would be how much a user believes that using a product would enhance their job performance. Ease of use would indicate the product could be used without effort. Usability is not referring to if a user could use the tool, but if the tool would actually help the user in a task or doing a job.

- **Hardware.** The hardware applies the models to the data, does the calculations and also displays the results. According to Eierman, Niederman and Adams (1995), the DSS should structure a decision in a way that analytical tools can generate solutions. These tools may even be used in combination. Furthermore, the DSS should manipulate, retrieve and display the data. As stated in the definition of a decision support system, the decision support system should be computer-based and operate using at least the minimum required technology.
- **Intelligence or expert component**. The know-how to assist the decision maker. Preferences, judgments and experience of decision makers are essential. Decision makers are sometimes seen as experts in these specific problems. As much knowledge and experience of these experts as possible should be built into the DSS (Klein and Tixier, 1971).

The author has firstly explained why DSS are developed. DSS were then defined and the components or building blocks of DSS identified in the previous section. As mentioned, it is the combination of these components that makes DSS. Mann and Watson (1984) state that definitions can sometimes fail to tell the whole story and that it is indeed the case with DSS. For this reason, some prefer to discuss DSS characteristics rather than definitions.

The characteristics of DSS are important to identify and define. Quality DSS should address the characteristics and the problem scenarios that make the DSS environment complex. The author describes the following as characteristic of DSS:

- Support and not replace decision makers or judgement, Turban (1988), Duffy and Hough (1985), Alavi and Henderson (1981). It is important that the DSS should not be seen as a replacement for the decision maker but rather assist in the decision making. People still run organizations and only people can make decisions.
- Support semi-structured and unstructured problems, Turban (1988), Duffy and Hough (1985), Sprague (1980). The DSS should facilitate the decision maker through problems with multiple solution paths. DSS should be effective in situations where routine actions will not necessarily lead to a solution.

- Computer based, Duffy and Hough (1985). The complexity of the decisions, together with the numerous models and enormous data sets make the use of computers in DSS almost paramount.
- Flexible, Duffy and Hough (1985), Sprague (1980), Meador *et al*. (1984). DSS should be able to adapt to our changing world that leads to different, changing and new problems.
- Used for rapidly evolving problems. As the world and technologies change, problems evolve with it. Because the problem evolves and changes, it is not possible to formulate a solution once and then apply it to future problems, Klein and Tixier (1971); Mann and Watson (1984).
- Try to combine the use of models or analytical techniques with data, Sprague (1980).

It is clear that DSS should support the decision maker in making a decision in time while the solutions can still be relevant. This would imply that a computer will be used and that the use and interpretation of the results should be in a user-friendly way. The decision support system uses data that can be in a database. Solutions are reached by using models that can be mathematical or just heuristics. The decision support system should be flexible because it is used for semi-structured or unstructured problems and because the problem can easily evolve. How the decision is reached, depends heavily on the user or expert. By taking the components and characteristics mentioned, we define the decision support system components as:

- Users – end-users as well as experts.
- Hardware.
- Data.
- Model.
- User interface.
- Intelligence component.

In the next paragraph the author describes how DSS are developed. Now that the components are defined, the DSS should be developed to make sure all components are included.

2.1.4 Developing DSS

Lu *et al.* (2001) state that decision makers do not benefit from DSS that are never used. What steps should be taken during development to ensure that completed DSS are accepted by its intended users? Some research indicates that millions of dollars have been spent on DSS that have never been used (Fuerst and Cheney, 1982; Cheney and Dickson, 1982; Robey, 1979). How should DSS be developed to prevent this?

Fick and Sprague (1980) argue that DSS need a development strategy that is different from the traditional analysis-design-programming-implementation routine and that the one-shot systems analysis strategy is not applicable to the development of DSS.

By defining the components of the DSS in the previous paragraphs, we know what to build. We now need to know how to build it. Surely, the size of the decision support system, the project team and budget plays a role in the way the DSS are developed. Consider the following scenarios:

- Organizations may feel it is necessary to develop a complete DSS using a lengthy, thorough and possibly expensive development method. This can ensure that all relevant stakeholders are consulted, the necessary resources are used and the DSS fulfil expectations. Fick and Sprague (1980) warn that not all people involved will understand the decision problems of the DSS. A single systems analysis will not solve these problems. DSS professionals should be open to learn and redefine the system boundaries, structures and methods.
- In some cases, the organization might want to develop a decision support system for only a part of the decision process. Other DSS can be added later and possibly linked to one another. Meador *et al*. (1984) state that the development and use of the DSS should continue to be managed instead of just installing the system and exiting. In this way, the decision support system can stay responsive over time.
- Some problems are unique and help is needed with a specific decision as soon as possible. The DSS will most probably never be used again and this means the system need not be implemented or incorporated with other systems. As soon as the results are achieved, the system becomes obsolete. The likelihood of a decision support system succeeding and having the

desired organizational impact strongly depends on understanding how to manage DSS development and use over time. Meador *et al*. (1984) state that to stay useful, a decision support system must respond to changes.

The way we develop DSS will differ for all three these cases. It evens differs for the size of the decision support system or the budget or the developers. Turban (1993) describes three types of DSS developed namely complete, staged and quick-hit. The following discussion of these types is based on the work of Turban (1993).

Complete DSS.
A full-service, large-scale decision support system generator or specific decision support system is developed. An organizational unit to manage such a project is needed. Because the development may take several years the success of the development might not be visible. Another problem is that the technology may become out of date due to the long development time.

The construction of large DSS is especially a complicated process. Some DSS may require a lengthy and well-documented development process. Turban (1993) describes the following phases for this type of DSS development as:

Phase A: Planning
Planning should determine the needs of the users and organization as well as defining the problem and goals of the decision support. The most important might be determining the key decisions and the method of critical success factors is recommended.

Phase B: Research
The best way to address user needs with available resources should be determined. Possible resources include hardware, software, people, experience, etc.

Phase C: Analysis
The analysis phase determines the best way and use of resources to implement the system. It could be seen as a conceptual design followed by a feasibility study.

Phase D: Design
Each component of the decision support system can be designed. The design determines the detail of the components of the system, the structure and the features of the system.

Phase E: Construction

The decision support system is designed according to the tools used and the design. Continuous testing can lead to regular improvements to the system.

Phase F: Implementation

The implementation phase includes the following:

- Testing how the system performs compared to the design specifications.
- Evaluating how the system should meet the user's needs.
- Demonstrating the system when it is fully operational to the users.
- Managerial users are trained in basic capabilities.
- Users are trained in the system structure, its functions and how to maintain it.
- The system is deployed to all users.

Phase G: Maintenance and documentation

Maintenance is the ongoing support for the system and users. Documentation is developed for the use and maintaining of the system.

Phase H: Adaptation.

Adaptation involves repeating the steps above on a regular basis. This is done to respond to the changing user needs.

The phases above described by Turban (1993) need not follow a linear fashion. Loops and cycles are possible. The phases are illustrated in figure 2.2.

The development process described above corresponds with the DSS life cycle Meador *et al*. (1984) suggest. Meador *et al*. (1984) state that DSS development should follow a plan and that adaptive development does not mean ad-hoc development. The plan states what should be done, when and in which order. The plan must be cyclical because DSS evolve. By redoing all or most of the tasks at regular intervals keeps the system responsive to the changing needs of users. Fazlollahi *et al.* (1997) state that the "effectiveness of DSS are enhanced through dynamic adaptation of support to the needs of the decision maker, to the problem, and to the decision context".

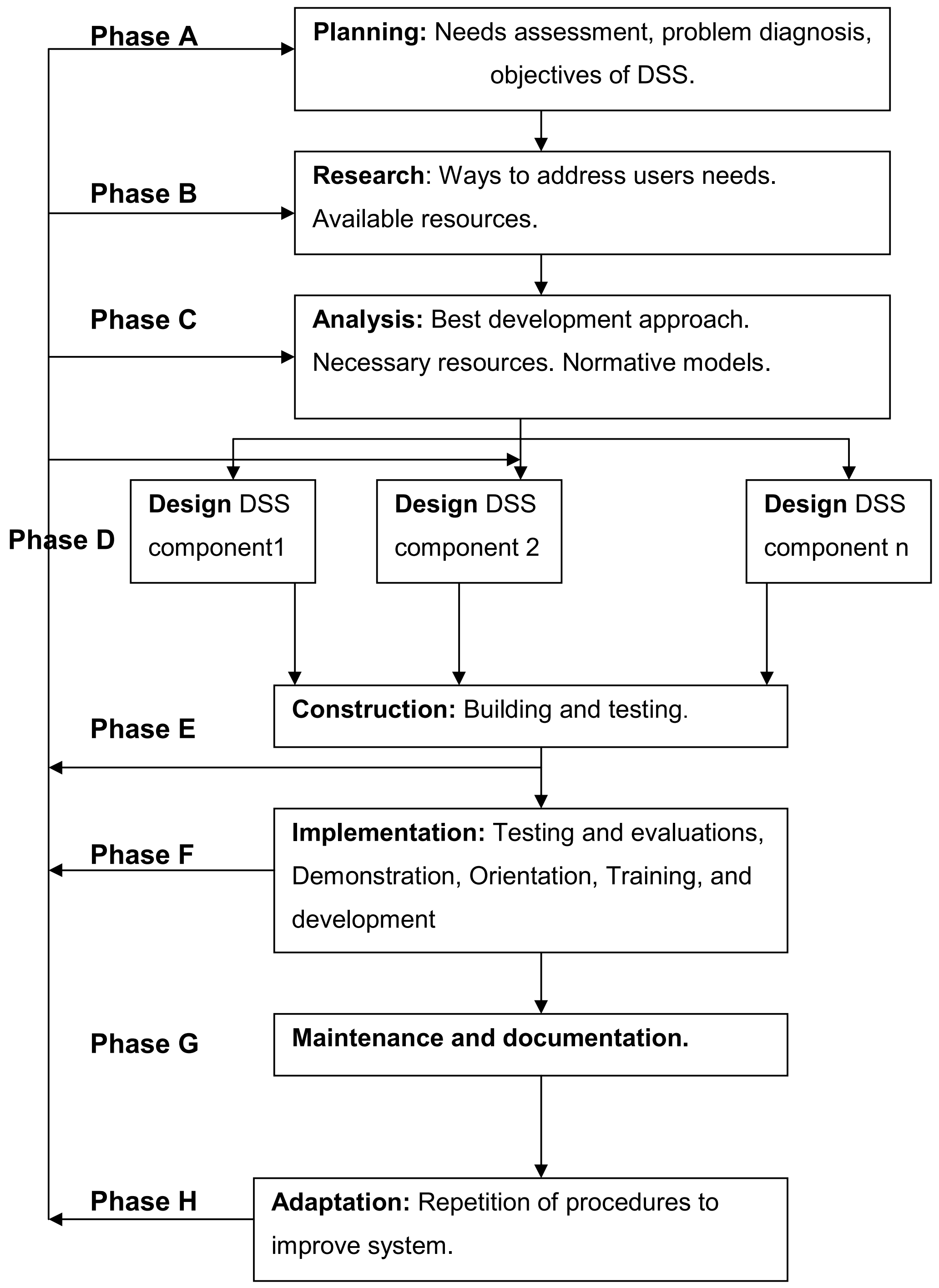
Phase A
Planning: Needs assessment, problem diagnosis, objectives of DSS.
Phase B
Research: Ways to address users needs. Available resources.
Phase C
Analysis: Best development approach. Necessary resources. Normative models.
Phase D
Design DSS component1
Design DSS component 2
Design DSS component n
Phase E
Construction: Building and testing.
Phase F
Implementation: Testing and evaluations, Demonstration, Orientation, Training, and development
Phase G
Maintenance and documentation.
Phase H
Adaptation: Repetition of procedures to improve system.

Figure 2.2: Phases in Building a DSS – Turban (1993)

Meador *et al*. (1984) state that time and money required for DSS development can often not be predicted, and managers cannot easily estimate the return on this investment beyond some vague promise of better decisions. This means that the development of DSS may not get the go-ahead from management if a lengthy and very costly development is involved for a system for which its benefit cannot precisely be identified and quantified.

The developers might be unsure of how to initiate and develop a decision support system or how to manage the development process as it evolves. A distinct feature of DSS development relative to other information systems is the use of an adaptive design and development strategy. This calls for small-scale prototype systems, continued incremental development and responding to the changing needs of users (Keen, 1980).

For this reason it is clear that it is not always possible to use a lengthy and costly construction approach. The problem may change before the development is complete making the decision support system obsolete. This is why other development orientations are used. Quick-hit and staged development are examples of other development approaches.

Quick-hit

Turban (1993) states that these DSS are constructed when a need is recognized with a high potential payoff or where a difficult problem exists. Costs and risks should be low and the decision support system should be developed in a short time. It uses commercially available generators. A decision support system developed with a Quick-hit approach is normally developed for a single user or purpose, and does not relate to other DSS. Limited experience is carried over to the next DSS.

In the following cases a Quick-hit approach is appropriate (Turban, 1993):

- **Clear-cut goals:** No research should be necessary beforehand.
- **Clear-cut procedure:** The decision support system should be developed based on existing types of well understood procedures and calculations. No research should be necessary.
- **Available data:** The needed data should be readily available.

- **Few users:** The decision support system should be for one or a few highly motivated users with common goals and concerns. The decision support system should not cross organizational boundaries.
- **Independent system:** The decision support system may use input data from other systems, but should operate independently of all other systems once the data has been retrieved.

Staged development

The decision support system is constructed with advanced planning. This is done to facilitate the use of part of the effort in future DSS. This approach can lead to the development of an in-house DSS generator.

The approach used to develop DSS depends on the specific situation. This includes the organization, purpose of the decision support system, users, tasks, available tools, and builder. A combination of the approaches may also be used. Kivijärvi and Zmud (1993) further state in their findings that implementing DSS involve employees that possess scarce talents and whose time is already over-committed. By knowing which implementation activities are the most important for success, the resources can be assigned optimally. This can minimize the impact on the organizations.

Geraghty (1993) lists three main drawbacks in the development of DSS:

- Long development time involved (1-5 person years).
- Acquiring knowledge can bottleneck and transferring knowledge from the expert to the computer can be difficult leading to more bottlenecks.
- Unlike an expert, the decision support system do not know when it has reached the limits of its expertise. A person will say when a problem cannot be solved with his or her knowledge. A computer could provide misleading results.

Even though these drawbacks exist, Geraghty (1993) still recognizes the benefit of expert systems. The type of DSS that are developed and the way it is developed should be in a way that the drawbacks are minimized. The development time should be according to the type of DSS and the people, especially the experts, involved should be minimally disrupted.

Limited resources impacts DSS implementations. Efforts to improve the quality of development activities tend to be resource intensive. Knowing which activities are the most important can therefore be extremely useful (Kivijärvi and Zmud, 1993).

Can the development of DSS be based upon the development of other Information Systems? Are there any existing tools, techniques, methods, or sequence of steps to follow to develop Information Systems and more, could they be applied to the development of DSS? How can the developers be sure they include all the components of the decision support system into a system that is accepted by its intended users? The author tries to answer these questions in the following paragraphs.

2.2 Systems development methodologies

New systems are developed to increase revenue, avoid cost and improve service (Gane and Sarson, 1977). Even though it was stated more than 30 years ago, it is still relevant and should be kept in mind when developing systems. Systems should be developed to perform in the way it was intended to, within the development time and budget framework. There exist a number of beliefs regarding the development of Information Systems. The system development life cycle is an example of this.

From the 1970's developers tried to solve development problems. Some believe the development should rigorously follow a sequence of steps with step-by-step instructions and deliverables. This would ensure everything is agreed upon and developed, with the extensive documentation as assurance in the case where users are not completely satisfied with the end result. Other developers believe a number of factors determine the way the system is developed. Many System Development Methodologies (SDM) have been developed to address some of the problems associated with the development of Information Systems.

Developers can use SDM to develop systems, use a pick-and-mix approach where different development methods are used that may be applicable to the problem, or no methods may be used at all. The author describes in the following paragraphs exactly what are SDM, how SDM differ from other development methods and describes the grouping of SDM according to a structure described in Avison and Fitzgerald (2003).

To construct something, one needs a plan. This plan would indicate what should be constructed, how it should be constructed, the tools, materials, and other inputs that are needed. It might even be specified what sequence the construction steps should have. In other words, what should be built first, what parts can be built simultaneously, what parts should be completed before a new part starts. When constructing an IS, the developers need to know the same. Developers need a plan indicating the steps of development and what is needed at each step.

The system development life cycle (SDLC) is an example of such a plan. The SDLC is a waterfall approach to development. This means a step should be completed before the next step is started. Once a step is completed, it is not changed again. This makes the SDLC rigid and not very suitable for the development of DSS. The system development life cycle includes the following steps:

- System analysis and planning.
- Design.
- Construction and testing.
- Implementation.
- Operation and maintenance.
- Evaluation and control.

The SDLC, however, might not address all the development aspects of the DSS. Furthermore may it be necessary for incremental or iterative development. Sprague (1980) states that because DSS differ from other systems, a different design technique from traditional batch, or online transaction processing systems is needed. One reason why traditional methods are inadequate is because of the rapid changing conditions which decision makers face.

Mann and Watson (1984) point out that many practitioners and scholars have stressed that a decision support system requires a somewhat unique development approach. Names such as heuristic (Berrisford and Wetherbe, 1979), iterative (Sprague, 1980; Kivijärvi and Zmud, 1993) and evolutionary (Courbon *et al*., 1979; Boland, 1978; Kivijärvi and Zmud, 1993) have been suggested for the development of DSS. The people that suggest these different approaches points out how difficult, if not impossible, it is to establish DSS specifications without a trial and error process. Mann and Watson (1984) further suggest that high levels of user involvement are commonly mentioned as being important to this approach.

Parker *et al*. (1995) warn that not applying the models and tools correctly can lead to unrealistic and misleading outputs. Westmacott (2001) claims that much of the dangers can be overcome through careful design of the system. Appropriate and sufficient information should be given to the decision makers about the model and its limitations. Results are not always more reliable when the models become more complex. Geraghty (1993) states that a decision support system can provide misleading results if it is used for problems outside the limits of its expertise. Making the model more complex will most likely not help. Developers should therefore be very careful of how they develop DSS.

Could the use of SDM ensure that the best DSS are developed that address all the components and characteristics of the DSS? Could the use of SDM ensure greater user satisfaction that will lead to greater and more effective use of the DSS? Could SDM develop DSS that make decision making easier? These are some of the questions we would like to answer through the planned study. In the following paragraphs, the author introduces the reader to SDM by defining SDM, discussing some approaches to SDM and discussing some SDM in detail.

2.2.1. Defining System Development Methodologies

Avison and Fitzgerald (2003) define a systems development methodology as a "recommended means to achieve the development, or part of the development, of information systems based on a set of rationales and an underlying philosophy that supports, justifies and makes coherent such a recommendation for a particular context. The recommended means usually include the identification of phases, procedures, tasks, rules, techniques, guidelines, documentation and tools. They might also include recommendations concerning the management and the organization of the approach and the identification and training of the participants".

This implies that SDM should be of assistance in the development of information systems. It could be used for a part of the information system or the whole system's development. A philosophy is a way of thinking about the world or in this case development of information systems. These thinking frameworks determine the way development is done, and why it is done in such a manner. The SDM will usually state what should be done, when it should be completed, by whom, and why it should be done.

According to Huisman (2000) a system development methodology is a combination of:

- **Systems development approach(es)**
  It is the set of guiding principles and underlying beliefs, goals, fundamental concepts and principles of the system. Examples include the structured approach, data driven approach and people-orientated approach.
- **Systems development process model(s)**
  A process model can be seen as the sequence of stages through which the system evolves. Examples include the linear life cycle model and spiral model.
- **System development method(s)**
  A method consists of a set of guidelines, activities, techniques and tools. These are based on a particular philosophy of system development and the target system. The method should be a systematic approach to conducting at least one phase of system development. Examples include Information Engineering (IE) and Soft Systems Methodology (SSM).
- **Systems development technique(s):**
  A systems development technique is a procedure to perform a development activity. The procedure could possibly have a prescribed notation. Examples include entity relationship diagrams or rich picture diagrams.

This implies that a system development methodology has a development approach to guide the development. What needs to be included in the development and why it is important is underpinned in the approach. The process models are the order in which development steps are carried out. Why the steps are performed in this order depends on the development approach. The development method is what needs to be done to develop systems according to the approach. The techniques are a way to do these development steps. The combination of all this is a methodology. This idea is better demonstrated in figure 2.3.

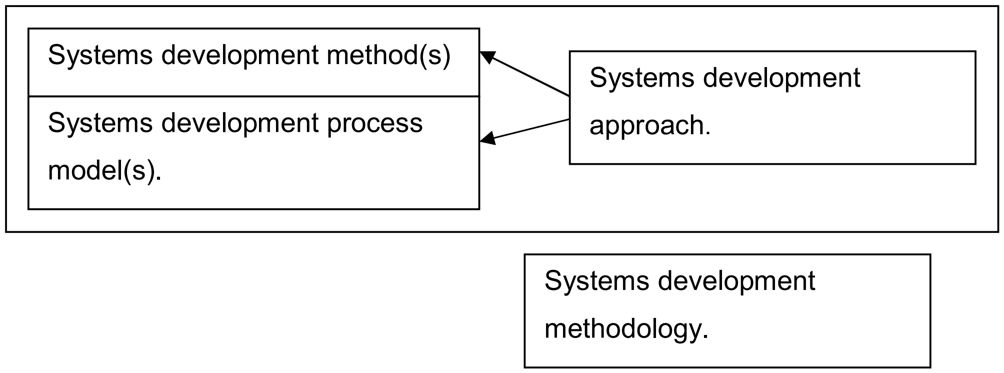
Systems development method(s)
Systems development process model(s).
Systems development approach.
Systems development methodology.

Figure 2.3: Systems development methodology, Huisman (2000).

2.2.2. Systems Development Methodology approaches

Avison and Fitzgerald (2003) define a number of approaches in systems development of information systems. The approaches the author felt applicable to the development of DSS are:

Organizational-orientated methodologies

According to Avison and Fitzgerald (2003), organizational systems are not predictable because humans are involved. Humans do not follow or operate as a piece of software would. Even interpretations and how problems are viewed differ from one person to another. The author included a methodology that is organizational-orientated because DSS are developed to support human decision makers. All humans are unique and even two experts in the same field might solve the same problem differently. The author chose to research the Soft Systems Methodology (SSM). SSM is known for not reducing a problem in smaller units of investigation. SSM stresses systems thinking and views the problem as part of a bigger system. Emphasis is placed on the opinions of all the stakeholders of the system. In this way, not only are the user component of the DSS addressed, but using the SSM might be very useful to connect the expert to the DSS and extract knowledge needed to support the user.

Blended methodologies

Blended methodologies are known for not only focusing on the data elements of system development, but also the processes involved. Although they view data as the building blocks of systems, they recognize the importance of processes in the development. The author suggest Structured systems analysis and design method (SSADM) and Information Engineering (IE) as methodologies to address the development of DSS and in specific, the data and model components of DSS. The author chose to suggest two SDM from this approach because two DSS components are addressed through these methodologies. The methodologies are more structured and, if used, may most likely lead to longer development times. Some may feel that these methodologies, especially SSADM, are too strict and rigid for the development of DSS. As mentioned earlier, the waterfall approach is most likely not suitable for DSS development. The methodologies, however, have evolved and it is sometimes possible to do a pick-and-mix approach or use it with other methodologies like SSM. In situations where decisions need to be made that will have a big impact, or may affect many people, or a combination of both, it is likely that a thorough investigation and in depth study of the problem is necessary. In this case, short development times and minimum documentation may not be appropriate. An example would be if 100 years after a decision was made, people would like to know why the specific decision was made. Documentation and records would be the only answer available. If a decision was made that had a big impact on people, for example, leading to the loss of lives, documentation would provide some answers why it went wrong. The well documented process may be very appealing to managers that have to approve budgets and developments where the immediate benefit might not always be obvious.

Rapid development methodologies

Rapid development methodologies are known for reducing the time to develop systems. They have high user involvement and use prototypes to determine the requirements. Extreme programming (XP) is researched as a rapid development methodology. XP is known for using paired programming and architectural spikes in the development process. XP will address the user component as well as the user interface that should lead the user to making a decision and thus controlling the usage of the system. Rapid development methodologies were also chosen due to the fact that DSS should be developed in a short time to accommodate the evolving problems of DSS.

People-orientated methodologies

People-orientated methodologies stress the importance of the users and other strategic persons in the development of systems. For example, CommonKADS, a methodology specially designed for the development of expert systems, relates to a wide domain of knowledge management. This will address the Intelligence component of the DSS.

Using these approaches, the author linked methodologies that could address the development of the specific components of DSS. Table 2.2 illustrates how the approach and SDM researched fits in to address the DSS components identified earlier in the chapter.

Table 2.2: Approaches to address DSS components.

DSS components	**Development approach**	**SDM example**
Users: end-users as well as experts.	People-orientated and Organizational-orientated methodologies	SSM
Hardware		
Data	Blended methodologies	SSADM IE
Model	Blended methodologies	SSADM IE
User interface	Rapid development methodologies	Extreme programming
Intelligence component	People-orientated and organizational-orientated methodologies	CommonKADS

Many developers may feel SDM are old and outdated. It is claimed that only parts, if any, of SDM are used. SDM also have problems. Avison and Fitzgerald (2003) mention the era of methodology reassessment. Problems like complexity of SDM, inflexible SDM, difficult to use tools, goal displacement, and significant skills required are some of the problems associated with using SDM. Although there exist problems with using SDM, and there is an era of reassessment, it does not mean that SDM cannot offer anything positive. One positive aspect of using SDM may outweigh all the negative aspects. IS development involves the development of systems for humans by humans. Whenever humans are involved, something can go wrong. It is the opinion of the author that there is much good in SDM that should not be discarded just because critics of SDM can be found.

Chae *et al.* (2005) ask if there should be ethical issues involved in the development of DSS. They argue that the order of the impact and the number of people affected by the decision should guide the developers to the ethical aspects involved. Chae *et al*. (2005) stress the importance of solving the right problem and the involvement of as many stakeholders as possible in the development. The use of SDM may address these issues Chae *et al*. (2005) mention by involving stakeholders and using a structured approach of developing systems based upon a philosophic view and beliefs. The use of SDM facilitates the documentation of the development, and thus serves as reference why the system concludes certain decisions. There is a responsibility to develop good quality systems. SDM were developed to improve the development.

The author discusses in the following paragraphs each of the SDM approaches as well as the individual SDM chosen with the approach to address the development of the DSS components.

2.2.2.1 Organizational orientated methodologies

System thinking is concerned with the interactions between a system and its environment, and not only with how the system works. One of the corner stones of systems theory concerns Aristotle's dictum that the whole is greater than the sum of the parts, Avison and Fitzgerald (2003). It may be difficult to determine all the parts used in a decision making process. In terms of DSS it can be equally difficult to explain the whole in terms of its parts. Experts might find it difficult to explain exactly how they reach a decision. Another challenge would be to use all the elements of decision making effectively in one product called a decision support system.

User acceptance and efficient use of the final product is very important. This is also crucial for the efficient use of DSS. It is thus important that the users of the DSS are comfortable in using it, and pleased with the system. It is thus not possible to separate the human aspect from the system because experts in the field are consulted during the development of the systems.

In order to get the best possible picture of the problem (decision making), the author suggests using an organizational orientated methodology. These methodologies should address the human and expert component in DSS. Alavi and Joachimsthaler (1992) state that DSS are voluntary and the users may choose not to work with the system. One of the biggest goals should be that DSS are used correctly and efficiently. This can only be achieved if the users are completely satisfied with the system and the system addresses the right problem. The organizational orientated methodology discussed is Soft Systems Methodology.

2.2.2.1.1 Soft Systems Methodology

Why SSM?

According to Lea *et al.* (1998), SSM was developed to be applied to a wide range of problem situations, particularly endemic and ill-structured problems. For these problems it might be impossible to find a solution that everyone agrees on. This makes SSM useful for the development of DSS. The author stated that one of the characteristics of DSS were the ill-structured problems it is developed for.

According to Checkland and Scholes (1990), SSM grew because systems engineering failed in messy complex problem situations. Systems engineering looks at ‘how to do it’ when the ‘what to do’ is known. In ill-structured problems, the ‘what to do’ is most often not known. SSM was developed to address this.

Checkland and Scholes (1990) also describe jobs that can never be completed because the problem always changes or evolves. This is also a characteristic of the problems DSS are developed for. They state that SSM was developed to assist managers to cope with problems like this and that SSM is an organized way to tackle messy situations. Fitzgerald *et al.* (2002) state that the SSM approach acknowledges how important people are in an organization and attempts to make sense of complex human activity systems.

According to Avison and Fitzgerald (2003) one of the underlying assumptions of SSM is that it is necessary to develop systems for the organization as a whole, and not only for the individual parts of the organization. Humans may behave differently in isolation than in the organization. According to DSS definitions, DSS should support the decision maker. These decisions can affect the whole organization and therefore it is important to understand the whole organization as well as knowing what the impact on the whole organization would be.

By including people in the model and not only data and processes, it is easier to understand the whole and notice soft problems that might not be obvious. It is important to view the organization in its environment, because humans are unique, experiments are not repeatable and between people there exist conflicts, different ideas, and attitudes. Other methodologies that break up problems into solvable parts may not be effective.

The methodology

Checkland (1981) uses action research where ideas are tested out in organizations. Analysts would not prescribe or observe but rather participate. Conclusions can be drawn from experience and the central focus of Checkland's research was to obtain a certain view (or views). The worldview is a central theme through the methodology.

The stages of SSM that Avison and Fitzgerald (2003) describe are more a framework than a detailed set of rules because analysts might work on several stages at any time, and backtracking and iteration is essential. The discussion of SSM is largely described according to Avison and Fitzgerald (2003).

The seven stages of SSM:

a) The problem situation: unstructured.

According to Lane and Oliva (1998), the purpose of stage 1 is to form the richest picture possible of the problem situation. A wide selection of viewpoints should be explored. There will be many different views on the problem. The first two stages will try to gather as much as possible information regarding the problem. It is not likely that all the participants will see eye to eye on the problems. Some of the participants will include the problem owners, the actors and the stakeholders.

The analysts need to study the problem situation in terms of physical layout, the reporting structures, formal and informal communications patterns, and how these activities are controlled. The relationship between structure and process can be very informative.

b) The problem situation: expressed.

This stage expresses the informal view of the problem situation in a more formal way. The goal is to find out more about the problem situation. Although Checkland (1981) does not prescribe an approach, many users draw rich picture diagrams.

Rich picture diagrams are a communication tool between analyst and users as it helps the problem solver to understand the problem better. Rich pictures should encourage in-depth discussions from all parties to give developers as much information as possible. Rich picture diagrams will usually show people involved, problem areas, controlling bodies, and sources of conflict. Rich pictures should be used to identify problems and not to recommend solutions.

c) Root definitions of relevant systems.

In this stage, relevant systems are named according to the problem themes. Debate is paramount as several different relevant systems could be named and changed. The problem owner and problem solver should decide how to describe their relevant system.

A root definition is created for the relevant system. A root definition is a theory about the relevant system and possible ways to help the problem situation. The root definition is created by using the CATWOE checklist.

The mnemonic CATWOE is formed by the six parts:

- Client is the ‘whom’ (affected by the activities).
- Actor is the ‘who’ (the agent of change).
- Transformation is the ‘what’ (the change taking place).
- Weltanschauung (worldview) is the ‘assumptions’ that makes the change meaningful in context.
- Owner is the ‘answerable’ (sponsor or controller).
- Environment is the ‘environment’ (the wider system).

d) Building conceptual models.

The root definition is used to create the conceptual model when everyone is satisfied with the root definition. The conceptual model describes what will be performed by the system. By debating the conceptual model, actors can relate the model to real world situations.

Usually a conceptual model is drawn up for each root definition. Creating a conceptual model for each root definition can become an iterative process of debating and modifying until it is agreed upon. The process is meant to draw out different views and conflict to achieve a final version that represents the problem situation. It is important that there should not be a conservative compromise, but a holistic view to consider a complex problem.

e) Comparing conceptual modules with reality.

The problem situation analysed through rich pictures of stage 2 should be compared to the conceptual modules of stage 4. The debate of possible changes should lead to plans to change in order to help the problem situation.

f) Assessing feasible and desirable changes.

By digesting the views of proposed changes of stage 5, the analysts draw up proposals for changes that should be feasible and desirable.

g) Action to improve situation.

The final stage should make recommendations to improve the problem situation. Note that SSM does not describe methods for implementation, only aids in the understanding of problems.

Summary and suitability for DSS development

Lane and Oliva (1998) state that by comparing ideal types of human behaviour and real world problem situations insights can be formed. The analysts should be seen as actors and not as outside onlookers. SSM might be used as a front end before moving on to the hard aspect of system development and methodologies.

By using SSM as a front end, users and owners can feel part of the development from the start. Using SSM as a front end for other methodologies can provide a strong understanding of the problem, making it possible to find the best way to address the problem (Lea *et al.,* 1998).

In an environment where experts might be shy and difficult to communicate with, SSM might help to gain information and knowledge on a problem. Because SSM may use rich picture diagrams, and is based upon discussions between the problem owner and problem solver without conservative compromises, SSM may be the key to really understanding the problem.

The process in which a consensus is reached about the problem situation, instead of a compromise, may create a feeling that each user had a valuable input in the development. This may be especially important when the users or participants include experts.

The drawing of rich pictures and the first two stages concerned with the problem situation may lead to better understanding of these complex and fuzzy problem areas for which DSS are developed. The first two stages might also be very helpful in extracting knowledge from sometimes eccentric and private experts. According to the findings of Lu *et al.* (2001), DSS designers should place more emphasis on making users believe that the decision support system is useful instead of focusing on the easy-to-use interface. They note however that removing the barriers that allow people to believe that using a decision support system is easy would have a direct effect on its perceived usefulness and subsequently indirectly affect people's preferences and willingness to use DSS.

2.2.2.2 Agile methodologies

Rapid Application Development (RAD) was developed as a reaction to the problems of traditional IS development, especially the problems of long development lead times. RAD also addresses the problems associated with changing and evolving requirements during the development process. Changes and the evolution of the problems are characteristic of the problems DSS are developed for to support decision maker. Angioni *et al.* (2006) also claim that Agile Methodologies help in environments where requirements keep on changing and state that the methodologies should overcome problems with small frequent releases.

The author introduces the reader to Extreme Programming as development methodology to address some of the problems associated with developing DSS. An important finding of Meador *et al*. (1984) is that cost-effectiveness is an important factor in DSS project approval and its perceived DSS success. Systems (especially DSS, because it could only be used once, or is developed for one user, etc.) with lower development cost are more likely to be approved for development.

2.2.2.2.1 Extreme Programming (XP)

Why XP?

XP is a methodology that can lead to faster development of software for small and medium organizations. Documentation is not a priority in this methodology and this means the programmers can create code much earlier and faster because they do not have to wait for the rest of the development team to complete detailed documentation. Angioni *et al.* (2006) state that XP is an agile methodology that does not require any specific supporting tool for being successfully applied. This may be very appealing to developers of smaller DSS.

The client continuously participates as part of the team. This means the programmers can easily determine exactly what the client wants because the client is involved and prompted for input throughout the development. Instead of the programmers developing code that they think is important as they interpret the specifications from documents, only to find out on delivery that the client actually had something different in mind, the development can progress much sooner in the direction the client wishes. XP is more of a series of principles to develop software quickly than a step-by-step methodology. The focus is on incremental development with simple solutions, with each release put into operation quickly. A decision support system needs to be built with short, rapid feedback from the users to ensure that development is proceeding correctly. It must be developed to allow for quick and easy changes, Sprague (1980).

The system requirements for DSS are often difficult or impossible to define. Managers that face a difficult or complex situation do not always know what they want or need. The prescription offered is a simple development cycle that is repeated frequently that provides for a shorter feedback loop between designer and developer and user. The decision support system should be built so that it could easily be changed with each cycle. These suggestions correspond with the evolutionary approach (Fick and Sprague, 1980). According to Sprague (1980), the most important four steps in the typical systems development process (analysis, design, construction, and implementation) should be combined into a single step which is iteratively repeated. After a short period of use, the system is evaluated, modified, and incrementally expanded. This suggestion is different from prototyping in that the initial system is real, live and usable, not just a pilot case.

Beck (1999) describes XP as a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. Although using the methodology to develop systems and some of the key practises like paired programming might seem strange at first, Steinberg and Palmer (2004) claim that developers and programmers would prefer this way of development above any other as soon as they feel comfortable using it.

The methodology

The methodology is described according to the work of Avison and Fitzgerald (2003), Mackay *et al.* (2000), Steinberg and Palmer (2004), Layman *et al.* (2006), Hedin *et al.* (2005) and Fitzgerald *et al.* (2002). XP stresses teamwork and efficient communication. Open and honest communication with customers and team members are paramount. Team members, especially in paired programming, should be open to discuss problems, inabilities, or coding errors with each other. Clients should know exactly how the project progress and what is possible to deliver, and if any changes should be made to the project schedule. The client should be honest with feedback after each iteration and release to ensure maximum user satisfaction. The client should feel confident to communicate any problems, doubts and praise to the developers. This feedback will steer the developers in the right direction. It is the developer's responsibility to be completely honest with the client in respect to how the project progress and any setbacks or change in schedule. If any change is needed, the client should make the decision on how the project should change if needed. The project is completed in small steps because small steps are easy to plan, perform, evaluate and, importantly, easy to undo. To ensure the system always compiles, runs and performs correctly, any problem in each small step is resolved completely before moving on to the next step.

The XP team consists of the customer, management and developers. No hierarchies exist and the client is viewed as part of the development team. Any programmer can use or improve any code written by someone else. Steinberg and Palmer (2004) note that after a little designing and building the developers go over the progress with the client to determine if the development still follows the right direction. This approach is repeated to build the system with small increments. This development would lead to a system that is much closer to the client's evolving vision of that portion of the system. The idea would be to make changes where necessary as soon as possible. This prevents changes to code that have been forgotten.

The users define the requirements through user stories. According to Layman *et al.* (2006) user stories are descriptions of systems features written in informal, natural language on small cards. The customer writes these stories and is also responsible for stating the priority. The customer also provides a corresponding customer acceptance test. The developers should be sure to completely understand the user stories. It might be useful to re-read a story to the customer, rephrasing the stories in a process to ask the client if this is exactly what they want. Steinberg and Palmer

(2004) state that the goal is to keep the user stories in a language the client can understand. This prevents the developers of using technical terms or jargon the users don't understand.

Sometimes a spike is needed. Spikes are quick, in-depth investigations or short explorations into specific matters where you feel you don't know enough. Spikes can be used to investigate certain user requirements the developers are not comfortable with. Spikes should, however, only be performed when necessary, not take too long to perform and be completed as fast as possible. An architectural spike is an aid to find answers to difficult technical or design problems. It is a simple program to build a system only to address the problem under examination. These spikes are used to explore possible solutions. The spikes are not high quality codes, and are normally deleted. Spikes can reduce the risk of technical problems and increase the reliability of the estimates on user stories.

Avison and Fitzgerald (2003) describe the following XP phases:

- The development team decide on the scope, the priority of functions, team members, and the content of the increment. This will enable them to estimate the costs and schedules of releases. This planning is performed on the user stories.
- Designing takes place on the principle of coding simplicity, feedback from the user, courage, and enabling incremental change.
- Paired programming is used for development. Tests are developed before codes and extra code can be created to ensure the test work properly. The client gives rapid feedback on questions from the developers and developed parts. New code should be continuously integrated with already-implemented code. Code belongs to everyone and can be changed or improved by anyone.
- The tests at this time ensure that developments fit the whole system. The tests ensure good performances for running the software so that this phase naturally runs into maintenance.

As mentioned earlier, XP is more a set of principles to develop software quickly than a step-by-step methodology. Although the phases indicate what needs to be done, the key practices of XP shows us how. The discussion of the twelve key practices in XP is based upon Hedin *et al.* (2005) and Fitzgerald *et al.* (2002).

The key practises of XP:

a) Planning game

The project scope is determined. There should be a balance between business and technical considerations. The developers give their estimates on what is needed, how long development will take and cost.

The business aspects that need to be decided on include the project scope, the most important functionality that should be developed first and realistic dates for releases of the system from a business point of view.

The technical aspects include estimates of how long it will take to implement the features and how the development process will be organized.

b) On-site customer

An actual client should be available to the development team on a full-time basis. This is to answer questions, resolve disputes and set small scale priorities. The client is part of the development team and should ultimately make decisions on development. Steinberg and Palmer (2004) state that XP clients are not less fickle than other clients but that the XP clients know the cost of changes. A client that is involved in the development process would know the implications of choices in respect to costs and development time.

c) Pair programming

Pair programming involves two people writing code using one computer or workstation. Roles are switched continuously. One person codes whilst the other could be constructing tests or thinking about tests that might not work yet, thinking strategically about the whole approach, or ways to simplify the code. Steinberg and Palmer (2004) state that the programmers are not in a hierarchal relationship although one person leads and the other person supports. At any time the roles can switch. This idea is also common in other parts of our lives and for this reason paired programming is successful.

Paired programming implies that teammates should be open and honest with each other and be aware of their own and the other teammate's strengths and weaknesses. By alternating the roles of teammates and alternating the team members, it is ensured that knowledge and skill spread through the development team. The project manager can resolve conflicts regarding code. Programmers should try to pair with as many other team members as possible as regularly as possible.

Although one might think that development may take twice as long because two programmers are used for every code, Bryant *et al.* (2008) claim that paired-programming, have the following advantages:

- Creates an excellent apprenticeship environment.
- Creates more transparency in the team.
- Several studies claim that teams outperform individuals.
- Leads to lower error rates.
- Creates 'pair pressure', a kind of positive peer pressure.
- Improves communication between programmers.
- Leads to cognitive offloading.
- Less development time.

Bryant *et al.* (2008) compare the role of the support programmer or navigator to that of a foreman on a construction site. During the development, the foreman should think of the overall structure and how newly developed code fits into this structure and how the code will help to solve business problems. The navigator should not be a "back seat driver", only correcting syntax and spelling errors. However, by checking syntax and spelling, it allows the person coding the opportunity to concentrate on the best possible way to code. Switching roles can provide relief whenever a team member feels tired or starts losing concentration.

d) Collective code ownership

The XP programmers should know enough about the system to make necessary changes to improve code. Code is neither owned by anyone nor exclusive for individuals.

e) Continuous integration

As soon as code is working without any problems it is integrated. This is to ensure that it is obvious whose responsibility it is to fix code. Programmers load the latest release, load their changes and run the test until they pass completely.

f) Small releases

Releases should be as small as possible while retaining a coherent set of business requirements. Generally, short time-boxes of one or two months are preferred over six months or a year.

g) Metaphor

The idea behind the metaphor is that each project participant understands the basic elements and their relationships. Each XP project should have one overarching metaphor. Steinberg and Palmer (2004) claim that the metaphor helps you to say "This is kind of like..." and should help everyone get the same fuzzy picture of the system.

h) Simple design

The design should be simple. Extra functionality that might be needed in the future is not added, to avoid an overly complicated design.

i) Coding standard

Programmers should use the same coding standards that are generally accepted. Using the same coding standards enables programmers to use each other's codes and to make necessary changes to any code. Programmers might see an opportunity to make improvements on a piece of code.

j) Testing

In XP, tests are written before code. Tests are constructed from the user stories. All code is tested thoroughly after it is written. XP uses two kinds of tests. Unit tests are written by programmers to test that the code are working. Functional tests ensure the system meets the business requirements. Steinberg and Palmer (2004) state that by writing the test before the code, it forces the programmers into a simple, bottom-up design.

k) Refactoring

Refactoring is the improvements made to a system or code which does not affect its functional behaviour. These changes can improve the quality by making it more simplistic, more flexible, more understandable, or improve the performance.

l) Forty hour work week

Overtime is avoided to ensure that programmers are fresh and productive on a regular basis. Productivity will have an effect on the development time and team members should not burn out.

Steinberg and Palmer (2004), however, mention that the evolution of XP has four changes to these practises:

- Testing now consists of customer tests and test-driven development.
- Refactoring has been renamed to "design improvement" in order to emphasize the efforts input towards system design.
- The 40 hour week is now called "sustainable pace".
- "On-site customer" now includes coaches, developers with different specialities, and a manager and so on. This practice is now called "whole team".

Summary and suitability for DSS development

Bryant *et al.* (2008) claim that the XP methodology values:

- Individuals and interactions over processes and tools. This is because teamwork and honesty is so important in XP. Team mates should be honest with each other with regards to personal strengths and weaknesses when dividing work and teaming up. Interactions between team members can lead to the sharing of knowledge and skills and team members learning from one another. Working together means that tools are available to all.
- Working software over comprehensive documentation. The customer accepting and using code as intended should be the ultimate goal. Requirements change over time and as the system develops. Developers using XP would rather have the client indicating unsatisfactory parts as soon as possible than having to wait for a comprehensive documentation regime to request a change.
- Customer collaboration over contract negotiations. The XP client should know the cost of changing requirements or scope. The client should be the one to choose if some parts of the development can be abandoned to facilitate a change in requirements for other parts of the system.
- Responding to change over following a plan. By immediately responding to change, the developers can focus sooner on the product the client will accept

in the end, instead of developing code that would only be discarded after completion or never used.

XP is especially effective in the quick development of DSS where the requirements change quickly or the long development time is not optional. Because XP is not a rigid methodology focussing on steps, procedures and documentation, developers might be more willing to adopt the methodology. Users and owners form part of the development and small releases and prototyping keeps them involved.

In projects with long development times, the system might lose its relevance in that the problem situation may change. For this reason the XP methodology may be very useful because of the shorter development time. DSS are developed to assist the decision maker to make fast decisions. DSS are also sometimes developed because of rapid changing environments that may affect decisions.

The shorter development time may also lead to less development cost. XP may be prefect in organizations without a need for comprehensive documentation.

XP might however not be useful in the development of large and complex DSS where documentation is required. The idea of pair programming or writing tests before code might be too strange to some to implement.

2.2.2.3 Blended methodologies

Sprague (1980) states that a manager and a computer scientist seldom see a thing in the same way as the other might. It might be that the development of decision support systems need a more structured, detailed development methodology to address the development problems and bridge the gap between the developers, users and owners of a system. Meador *et al*. (1984) find that the least effectively performed tasks were seen to be planning, evaluation and orientation. Planning and evaluation are very important as it is the way the developer assesses the users' needs, how the needs would be met by DSS and what the DSS should look like to address those needs. McKeen (1983) finds that greater time and effort spent on front-end analysis resulted in less overall system development time, less overall cost and greater user satisfaction with the delivered system.

Decision Support Systems analyse data, but need the relevant processes to deliver results. Blended methodologies focus on both data and processes. Although some methodologies may stress data modelling as underlying philosophy, blended methodologies also examine the processes involved in the system. IE also determines if the system is of strategic importance to the organization. Meador *et al.* (1984) state that when a decision support system supports ill-defined problems, the impact of the decision support system on the organization problem-solving is likely to be ill-defined. If the impact on the organization is ill-defined, the benefit would also be ill-defined. Managers may find it difficult to approve DSS if this is the case. For this reason, determining how the planned DSS could help the organization reach its strategic goals is so important. SSADM and IE will be discussed as SDM.

2.2.2.3.1 Structured Systems Analysis and Design Method (SSADM)

Why SSADM?

SSADM is a waterfall method with a rigorous document approach to system design. This contrasts with the more contemporary Rapid Application Development methods. Although SSADM is an older methodology and in contrast with newer contemporary methodologies, the author chose to describe SSADM because the methodology provides more structure for development and has comprehensive documentation. As mentioned, the size of the decision support system also determines the type of DSS that is developed. Complete DSS, as Turban (1993) suggested, that will most likely need a large development team, might benefit more from a comprehensive methodology like SSADM.

According to Avison and Fitzgerald (2003), SSADM was originally developed by Learmonth and Burchett Management Systems and the Central Computing and Telecommunications Agency in the UK. SSADM was used in many government applications since 1981 as well as many Civil Service applications. The discussion is based upon Avison and Fitzgerald (2003) and is based on SSADM Version 4+.

The methodology

The methodology is highly structured and provides very detailed rules and guidelines for project development staff (Avison and Fitzgerald, 2003). Documentation fills every aspect of the project. SSADM is a methodology that can be very useful for developers that need the security associated with in-depth documentation. Developers know from the documentation exactly what is expected from the system and thus what point would indicate a complete system ready for implementation. Documentation protects developers from changes in scope. Everyone on the development team knows exactly what is expected from them and what the sequence of steps would be.

SSADM has seven stages and each stage has its own set of plans, timescales, controls, and monitoring controls, Avison and Fitzgerald (2003). Using a methodology that is so structured gives the development team guidelines on how to tackle big development projects. All the activities and their end products of all the stages are defined. The methodology facilitates project management.

SSADM covers the life cycle from feasibility study to design. SSADM does not, however, cover implementation and maintenance. It is assumed that planning has been completed (Avison and Fitzgerald, 2003).

The stages of SSADM as covered by Avison and Fitzgerald (2003):

a) Feasibility

Firstly it is important to determine if the project, as suggested in the planning phase, is technically possible. This would mean that the project would be worth more than the costs of building it. Davis (1983) states that there are at least three types of feasibility to consider:

- Can the system be implemented using the technology?
- Can the system be implemented in this organization?
- Do the benefits outweigh the costs?

Project management techniques, especially Projects IN Controlled Environments (PRINCE), may be helpful at this point. Davis (1983) states that the project's scope would determine the amount of time and money spent on the feasibility study.

There are four steps in this stage that include:

- Preparing for a feasibility study. This includes determining the scope of the project.
- Defining the problem. This compares requirements with the current position of the organization. By considering the weakness of the present system, it is possible to partly define the requirements of the new system. That is what the system will do, and what the constraints are on the system.

- Selecting feasible option. Alternatives are considered and one is selected. The best option from both the business and technical points of view is recommended.
- Creation of the feasibility report.

Interviews and questionnaires are examples of techniques that can be used to investigate the system. Although the analyst can form a surprisingly complete picture of the system, only people can clarify ambiguities and further questions that may come up (Davis, 1983).

Data flow diagramming is also used in this stage. Gane and Sarson (1977) state that a flowchart can trap the analyst into a commitment. If the flowchart is the only way of communicating, then drawing it would imply that a decision should be made, for example regarding the way to store data. These decisions should, however, be suggested by the designer, taking into consideration all the factors involving the development. The analyst should not become the designer and if the analyst and the designer is the same person, the analysis should be done before the design.

Entity models are drawn. All the techniques are done in outline with the detail only added in later stages. All the information should be published in the feasibility report.

b) Investigation of current environment

This part adds the detail to the work completed in the feasibility stage. The results of the feasibility study are examined. The scope of the project could be changed to ensure management agrees to the overall plan. The requirements of the new system are compared to the processing methods and data of the current system.

To determine the current functionality needed in the new system, the present physical data flow diagram model is mapped onto a logical data flow model. Matrices and catalogues are created.

Customization decisions are possible at this stage. Developers are encouraged to think whether each step is appropriate. Some steps might be dropped or reduced in scope according to the development objective for example rapid application development.

c) Business systems options

In this stage, the functionality of the new system is agreed upon. Only user requirements that justify the cost involved are carried through and then specified in greater detail. Function point analysis is recommended for estimation.

Different business options are outlined. Each of these options should at least satisfy the minimum set of user requirements. Management choose one option from the options presented. Management might choose a hybrid between options. Each option will have an outline of its costs, development timescale, technical constraints, physical organization, volumes, training requirements, benefits, and impacts on the organization.

The option chosen is documented in detail and agreed on as the basis of the system specification. Data flow diagrams and entity models are developed.

d) Definition of requirements

Requirements are specified in full in this stage and are used in the design stages that follow. The following tasks are completed:

- The requirements catalogue is consulted and updated.
- The logical entity model extended and normalized.
- The data flow model is also extended to be used as communication tool with users.
- User roles are defined for new system.
- Documentation forms for all entities and attributes are completed.

Functions are defined and documented in detail in terms of inputs, outputs, events, or enquiry triggers. Documentation also shows the relationship between user roles and functions.

The methodology suggests that prototypes of critical dialogues and menu structures are demonstrated to the users. Prototyping would verify if the analysts understood the requirements, reveal the users' preferences and commit the users to the development. It should not be used as part of incremental development.

Entity life histories show all events that affect an entity type and what the business rules are that is applicable in such an event. This stage also includes the verification of the system objectives, the completeness of definitions and the documents of the full requirements specification.

e) Technical systems options

This stage and the next are carried out in parallel. In this stage, the operating environment is determined in terms of:

- How hardware and software are configured.
- The development strategy.
- The organizational impact.
- The system functionality.

Because of the many alternative hardware, software and implementation strategies, the definition of technical options will depend on the specific implementation. The analyst identifies system constraints in terms of performance, security and minimum service level requirements. Technical system options need to meet all these constraints and this limits choice. Management should agree with the choice of technical option.

f) Logical design

This stage specifies what the system should do rather than how to do it. Users are involved in this stage to define dialogue structures, menu structures and designs. More detail about the system is gained through the following actions:

- The further development of entity life cycles.
- Defining the update processes and operations.
- Processing of enquiries.
- The sequence of processing.
- Specification of the detail such as the rules of validating data entered into the system.

g) Physical design

In this stage, a function component implementation map documents the mapping of the logical design onto a particular physical environment. This stage provides guidelines on how to implement the logic design that can work with most hardware and software configurations.

A database that is appropriate for the database management system is designed using the logical data model. How the database is mapped is very important in the final implementation. This includes how the data and its relationships are held on the database as well as performance and efficiency issues. The way functions are mapped on the components are designed and optimized according to storage and time objectives.

It is at this stage that SSADM stops and detailed software design and testing starts. Gane and Sarson (1977) distinguish the following phases in subsequent development of the system:

- Drawing up an implementation plan that includes testing and acceptance tests.
- The concurrent development of application programs and subsystems.
- The conversion and loading of the data base.
- Testing and accepting each part of the system.
- Exercising the system under realistic loads to see if performance criteria have been met.
- Commitment of the system to live operation.
- Measuring system performance to identify bottlenecks.
- Comparing overall system to original objectives.
- Analyzing and prioritizing of enhancement requests and placing system in maintenance state.

Summary and suitability for DSS development

SSADM has evolved to allow a more ‘pick-and-mix’ approach, and so a rapid application development model or a reuse model, or a SSM front end may be integrated. SSADM is traditionally associated with well defined, large scale systems development projects requiring heavy documentation for use particularly in large bureaucratic organizations. However, using the customization option and dropping or reducing steps to move to a rapid application development project could be perfect for the development of DSS. A large part of the methodology is spent towards the requirement specification (Avison and Fitzgerald, 2003).

Fitzgerald *et al.* (2002) state that the methodology changed over the years and that the methodology specifies guidelines, and not rigid rules, for software development. This, they claim, is to allow for the contingencies of different development situations and that the methodology explicitly identifies particular stages at which the process may be customized to suit the situation at hand. This may be the best solution for developers that may use SSADM to combine it with other methodologies and customize the methodology to fit perfectly with the decision support system development.

Gane and Sarson (1977) state that by using top-down development, the problems of how to make the program parts fit together can be avoided. Instead of dividing the organization into small units and developing code for each unit to be combined later into one big system, the system should be developed top-down with the whole organization in mind from the start. Interface problems can be very nasty according to Gane and Sarson (1977) for the following reasons:

- They tend to involve more changes to existing code. It may be necessary to recode, recompile, and retest all programs that use a piece of code if two subsystems cannot operate correctly together.
- One problem can only be fixed after the previous problem has been fixed. This can stretch out testing for an unknown time.
- Most of the interface bugs in a bottom-up development appear after most of the project budget has been spent and the delivery deadline is near and users expect a product that these bugs delay unpredictably.

SSADM may be very useful in organizations where comprehensive documentation is very important or the impact of the decision is grave. Stakeholders and owners may have the need for structured reports and processes to measure the progress of development of the system. By looking at SSADM as guidelines and adapting the stages to suit the development, the methodology can still be very useful in the development of DSS. The methodology considers data, and processes and prototyping can be used. SSADM ensures that the proposed system is feasible and useful to the organization, and that management supports the development.

2.2.2.3.2 Information Engineering (IE)

Why IE?

Avison and Fitzgerald (2003) claim that IE was developed by Clive Finkelstein in the late 1970's. In 1981, James Martin worked with Finkelstein, and produced a later version. Avison and Fitzgerald (2003) state that different versions of IE exist for different development environments. These different versions evolved from a similar concept, but developed along different lines. Examples of the different environments include a Rapid Application Development approach and an object orientated-based version. IE is said to be applicable in a wide range of industries and environments and according to Davis (1992) Information Engineering Method (IEM) is a methodology designed to simplify the development of information systems. Therefore it may be possible to use an IE version that addresses the specific type of DSS being developed.

Yates (1991) and Young (1991) stated that companies found it difficult to integrate Information Technology with their corporate strategy. This resulted in Information Systems being developed that were not aligned with the needs of the business. Ford *et al.* (1995) states that the Information Engineering Method forces the alignment of the information architecture with the industrial goals, objectives, and critical success factors of the organization. The corporate focus is translated into an information strategic architecture (Hackathorn and Karimi, 1988). This means that the information system should help the organization in issues that is important to the organization.

IE is claimed to be a comprehensive methodology that covers all the aspects of the system development life cycle and can be seen as a project management mechanism (Avison and Fitzgerald, 2003).

IE is built on a number of philosophical beliefs. One belief is that data are more stable than processes or procedures that act upon data. IE still recognizes how important it is to consider processes in detail. IE finds a balance between the modelling of data and processes (Avison and Fitzgerald, 2003).

Another belief is that diagrams are the most appropriate way to communicate in IE. Therefore, each step in IE delivers a standard diagram. Diagrams appeal to end-users and users can understand, participate in and construct IE diagrams for themselves (Avison and Fitzgerald, 2003). Using diagrams in the development of systems like DSS can be very useful where users, experts and analysts can express themselves and communicate with diagrams.

Using an appropriate toolset or automated support, is seen as very important in the use of the IE methodology (Avison and Fitzgerald, 2003).

Jackson (1986) stated that to engineer information, the analyst needs to know and understand the organization, or more specifically the functional activities associated with the organization's business system. IE is seen as top-down. The organization is seen as an unit. Even separate systems that are related are not treated as individual projects. IE uses the 'divide and conquer' approach to manage complex requirements (Avison and Fitzgerald, 2003).

According to Ford *et al*. (1995), Information Engineering Method system development takes place in stages. Information is at first seen at a high level of abstraction. With each stage, data, activities and their interaction are viewed in more detail.

The methodology

The methodology described in the work of Avison and Fitzgerald (2003) is based on the classic IE. The methodology is divided into four levels. The objectives of the four levels are to determine:

- If the system can support the needs and objectives of the organization (Information strategy planning).
- The system requirements of each business area (Business area analysis).
- How users expect the system to behave using the available technology (System planning and design).
- How to build and implement the system (Construction and cutover).

The first two levels do not depend on the available technology. The last two levels are dependent on the proposed technical environment. The following discussion of the methodology is based on the work of Avison and Fitzgerald (2003).

a) Information strategy planning (ISP).

IE first determines the overall corporate objectives. King (1985) states that the purpose of strategic planning for information systems is to provide a process for developing a long-range planning strategy for information systems within an organization that is based upon the overall strategic plan. In other words, the system should help the organization reach its goals and still fits into the strategic plan a few years later. Information systems should be of strategic importance to the organization.

Major business functions, their objectives as well as the organizational structure are looked into. It may be necessary to re-engineer some of the business processes

The plan should state the business requirements and their priorities. Porter (1985) states that the strategic planning for information systems can help an organization to find ways to achieve a competitive advantage by using information as a strategic weapon. It is therefore important to know what to develop and why it is important, or strategically important, to the organization. Developers should keep these business objectives in mind during development. In many other methodologies, these needs can get lost, or maybe never identified at all (Avison and Fitzgerald, 2003).

The ISP involves the following four tasks:

- **Current situation analysis.** The organization, its current position and the strengths and weaknesses of the current system are investigated. The business strategy, the information systems organization and the technical environment are analysed.
- **Executive requirement needs.** At this step managers can state the objectives, needs and perceptions. Business goals can be set and descriptions given on how technology can affect or help them. Critical success factors for the organization are determined and decomposed for each individual part of the organization.
- **Architecture definition.**

  An overview is formed of the following in terms of architecture:
    - Global entity types and the functions that act upon them.
    - The geographical distribution of data of functions.
    - What an ideal system would look like.
    - What technology is needed to support the system in terms of hardware, software and communications facilities.
    - How the system should support the strategy of the organization.
- **Information Strategy plan.**

  This includes:
    - Business areas in logical business groupings. Each group could be an analysis project.
    - Plans on how to achieve the architecture and how to move from the current situation towards it.
    - Priorities for development and work programs for high-priority projects.

b) Business area analysis

Detailed data and functional analysis are performed on each individual business area. Maximum involvement of the end-users is recommended at this stage.

The tasks of business area analysis are:

- **Entity and function analysis.** This is the major task of the stage. It involves analysing the entity types and their relations as well as analysing the processes and their dependencies. The entities, their relations, the processes and their dependencies are represented in diagrams. Attributes are defined.
- **Interaction analysis.** The relationship and interactions between the data and the functions are examined. A CRUD matrix (Create, Read, Update and/or Delete) show these interactions. Entity life cycles and process logic are also analysed.
- **Current system analysis**. Models from the existing system are analysed and compared to ensure a smooth transition. Procedure data flow diagrams are created and data models are created by canonical synthesis. Canonical synthesis pulls together all the data identified in separate parts of the organisation.
- **Confirmation.** The results above are crosschecked. Completeness, correctness and stability are checked. The effect of possible business changes is also examined.
- **Planning for design.** The parts of the models that need to be developed are defined and also how they should be implemented.

This step identifies areas where reusable objects or components can be utilized. Models and code could be reused, generated internally or purchased externally. Reusing objects or purchasing them can speed up the development process. Objects should be flexible and easy to use if they are to be reused later on.

The output for this stage is a business area description that contains the business functions. Each function is broken down into its lower level processes and the process dependencies. Data are described in terms of entity types, relationships and attributes with their properties and usage patterns.

c) System planning and design

This level is divided into business system design and technical design.

The business system designs involve the designing of a system for each business area to fulfil the needs that was identified for each business area. Because the design is developed up to a point where the technical factors become involved, it is called the logical design.

The steps involved for the logical design are as follows:

- **Preliminary data structure design**. This step is done for the whole business and not just for the design area. It is done for the whole business to ensure that systems integrate and are compatible with each other. The entity model is converted to the database management system that was chosen. Also included would be descriptions on how data would be used and preliminary data flow diagrams.
- **System structure design**. Business processes are mapped to procedures. Data flow diagrams show the interactions. In this step procedures are defined and data flow diagrams prepared.
- **Procedure design**. This phase develops data navigation diagrams, dialogue flows and action diagrams. Navigation diagrams determine the type and volume of access required to entity types. Dialogue flows show how user interactions are controlled. Action diagrams show the detail of process logic, business rules and specifies the sequence of actions.
- **Confirmation**. This stage ensures everything is complete, correct and usable.
- **Planning for technical design.** Areas for implementation are defined and the technical design plans are prepared.

Technical designs for each one of the business systems that was identified in the previous stages are developed. These designs are used to plan the cost of the construction of the system. The tasks of this technical design phase are the following:

- **Data design** – data load matrices are created, the database structure refined, the data storage defined, and other files designed.
- **Software design** - programs, models, reuse templates, integration groups, the design of programs/models, and test conditions are defined.

- **Cutover design** - software and procedures for bridging and conversion to the new system are designed. Planning for the fan-out of the system and how users will be trained is performed.
- **Operations design** - includes the design of procedures for: Security, unforeseen events, operating and performance monitoring. Software for operations is also designed.
- **Verification of design** - includes the benchmark testing and performance assessment.
- **System test design** – system tests and acceptance tests are defined.
- **Implementation planning** – the costs and the preparation of the implementation plan are reviewed.

The output for this stage is the technical specification that includes the hardware and software environments, its uses and its conventions. Also in the output is the plan for the next stage: How to construct and deliver?

d) Construction and cutover

This level deals with the construction, cutover and production of the new system. Construction involves:

- **System generation** – building the system. The database and database files, modules, data for testing modules, integration tests and documentation are created in this stage.
- **System verification** – the system should operate and perform as it was intended to. Data to test the system is generated. Performance and acceptance tests are performed to finally obtain approval. The use of test support tools is recommended.

If the system passed the tests and the system is accepted, the cutover begins. This involves a controlled change from the existing systems and procedures to the new system. Cutover include the following tasks:

- **Preparation** – the cutover schedule is prepared, users are trained and the hardware is installed.
- **Installation of new software** - new software is installed and trial runs executed.
- **Final acceptance** - the terms for accepting and moving to the new system is agreed upon.

- **Fan-out** - the system is installed at all locations.
- **System variant development** – if a location needs something different from the norm, the requirements are identified, analysis and design are revised and construction and cutover are performed.

If the systems perform as expected for a predefined time, the cutover is complete.

Production is when the system keeps on operating successfully over its lifetime. To ensure that service is maintained and changes in business requirements are addressed, the following tasks are performed:

- **Evaluate the system** - includes measuring performance, comparing benefits and costs, evaluating the acceptance by the users, and comparing the system with the design objectives.
- **Tune** - involves the monitoring of performance, tuning the software, and reorganizing the database.
- **Maintenance** – includes fixing any bugs and modifying the system if necessary.

Summary and suitability for DSS development

Although the levels, stages and tasks in sequence might suggest a top-down classic waterfall model, it is not necessarily the case. Much of the development after information strategy planning can be performed in parallel (Avison and Fitzgerald, 2003). In this case, IE may be more applicable to the development of DSS. As mentioned, big development projects may need structured methodologies to steer the project. IE can even be helpful in a project management way.

Another way in which IE can be used to develop systems is by starting at the bottom of the framework with an existing implementation and deducing business rules. This is called reverse engineering and is very useful when legacy systems are to be included in an IE framework. Re-engineering is a combination of moving forward and backward through the framework (Avison and Fitzgerald, 2003). In some cases where the expert or users are not always available or understandable, re-engineering might be useful to get the best understanding of the problem and situation.

Hackathorn and Karimi (1988) state that systems development activity has been a bottom-up approach. Various functions and data areas were automated on an application-by-application basis. This way there is not much consideration for integration and optimization at organizational level. For this reason IE with top-down analysis may be better to use depending on the DSS to be developed and how the DSS should be integrated. IE may be useful in understanding the organization completely and how the DSS should fit into the organization's strategic plans.

Inmom (1986), however, warns that the large amount of highly detailed requirement specifications, together with organizational objectives that is not clearly expressed, can make IE activities difficult to perform.

2.2.2.4 People orientated methodologies

Studer, *et al.* (1998) claim that in the 1980's the development of a Knowledge Based System (KBS) was seen as transferring human knowledge into a knowledge base. It was assumed that knowledge existed and had to be collected through interviews and implemented in some kind of production rules that a rule interpreter could execute. There is a consensus that the building of a KBS is nowadays a modelling activity. In other words, it is the building of a computer model that can solve problems like a domain expert (Studer *et al.,* 1998).

In order to extract knowledge from experts and to use this knowledge in DSS, developers need a methodology not only addressing how to develop the system, but also how to develop and implement the intelligence aspect needed to solve these difficult problems. Experts may not always be able to clearly express the way in which they solve problems or could even assume the person that they are describing the way they solve these problems to will have certain knowledge, experience and skill. Studer *et al.* (1998) state that the expert may articulate some part of his or her knowledge but the rest is hidden in his or her skills. This knowledge may not be directly accessible.

2.2.2.4.1 CommonKADS

Why CommonKADS?

CommonKADS is a development methodology under the wider domain of knowledge management. Knowledge is increasingly seen as an extremely important resource for any organization. Knowledge is being able to use information to achieve a goal or purpose (Avison and Fitzgerald, 2003).

CommonKADS is based upon the following principles described in Vollebregt *et al*. (1999):

- A model driven approach.
- Modelling knowledge is not dependent of how to implement it.
- Problem solving through identification of knowledge types.

Every activity in the development results in the construction of different models and form part of a cyclical model similar to Boehm's spiral model (Avison and Fitzgerald, 2003). Studer *et al.* (1998) state that the modelling view implies that:

- The model is not real behaviour but can only approximate behaviour.
- The modelling process is a cyclical process that refines, modifies or completes already built up models. The model itself can lead to new knowledge.
- The model is dependent on the subjective interpretations of the knowledge engineer. The model must therefore be revisable in every stage of the modelling stage of the modelling process.

The methodology

The methodology is discussed by using the framework provided by Avison and Fitzgerald (2003). This structure consists of six models.

a) Organizational model

The model describes the organization in which the KBS should operate and all the functions of each organizational unit are specified. This is a review of the potential knowledge-based sources. It discovers problems, opportunities and potential solutions for knowledge systems. This step establishes how feasible the project is and what the impact will be on the organization.

According to De Hoog *et al*. (1996), the input of the organization will determine the KBS's success and the organizational model helps those involved in developing the KBS to obtain a feeling for the organization in which the system will be deployed.

b) Task model

The task model describes how tasks are performed in the organizational unit and analyzes the global task layout. It includes inputs and outputs, needed resources that include agents, abilities, goals, preconditions and performance criteria. The model specifies which agents are assigned to different tasks.

c) Agent model

The agent model describes how the agent interacts with the system. Agents perform tasks (Allsopp *et al*., 2002). Agents can be humans, information systems, knowledge, culture/power related or a combination of these. The agent model describes the characteristics of agents. These characteristics are abilities, authority to act and constraints.

d) Knowledge model.

The types, roles and structures of knowledge used in a task are modelled. The model investigates the reasoning requirements of the prospective system. The knowledge model defines how problem solving occurs (Allsopp *et al*., 2002).

The knowledge model is independent of implementation. It neither allows representation of, nor gives guidance on, the decisions about which programming techniques to use in order to represent the acquired knowledge. This forms part of the design model (Kingston, 1998).

The following three stages form part of the knowledge model construction:

- Knowledge identification. Useful information sources are identified, like task templates and domain schemas that exist, as well as reusable model components.
- Knowledge specification. The inference knowledge, domain knowledge and task knowledge are identified. Domain knowledge describes objects, properties, relations, concepts, etc. in the real world, independent of a reasoning process. The inference knowledge is atomic tasks which can be simple algorithms that act upon and change domain knowledge using rule sets. Task knowledge defines the order of the inference steps. CommonKADS defines a library of problem solving task models called generic task models that can be used to build up problem solving knowledge specifications for the system in question (Vollebregt *et al*., 1999).

- Knowledge refinement. The knowledge model is validated and the knowledge base completed. Scenarios gathered earlier are simulated to ensure that the knowledge model can generate the problem-solving behaviour that is needed.

e) Communication model.

This model defines how the agents interact and exchange dialogue with other agents. The most important aspect is the transfer of knowledge between the agents. Allsopp *et al*. (2002) describe this model as specifying the interactions between the system and its environment.

f) Design model.

The models mentioned above specify the requirements of the knowledge system. This is transferred into a technical system solution. At this point a specification for the information system is made and it is based on the knowledge and communication model. These specifications may include software components (software architecture, algorithm design and data structure design) and hardware.

According to Kingston (1998) the design model encourages the designer to start with the knowledge contained in a knowledge model and then use a three-stage transformation process to produce design recommendations. The three stages are:

- Application design: The choice is on an overall approach.
- Architectural design: Choosing the ideal knowledge representation and programming techniques.
- Platform design: Choosing how to implement the recommended techniques in the chosen software. It considers how (and whether) the ideal knowledge representation and inference techniques should be implemented in chosen software.

Knowledge is modelled using a variant of the Unified Modeling Language (UML) class diagram. A variant is used as UML is not designed to model inferences and tasks. This is a requirement for CommonKADS. Guidelines help with the list of activities for each stage and techniques like interviews, protocol analysis (how the expert solves the problem), and repertory grids support the user in extracting knowledge from the experts. Knowledge acquisition plays an important role in many artificial intelligence projects (O'Leary, 1998). Knowledge acquisition from multiple experts was proposed to avoid some of the problems involved in relying on only one expert.

Summary and suitability for DSS development

CommonKADS is useful in the development of Knowledge Based Systems and therefore helps in extracting knowledge form the experts. Prototyping is suggested to form the basis for the reasoning system as well as to develop the user interface aspects. Avison and Fitzgerald (2003) feel the methodology takes a middle ground between data and process modelling. This means the development of the components is addressed with emphasis on the data and processes resembling the blended methodologies. Reuse, maintainability, adaptability, explanation, and the support for knowledge elicitation are quality criteria for the design (Avison and Fitzgerald, 2003).

2.3 Fitting SDM to DSS development

The suitability of the SDM to address the development of the different DSS as well as the components and characteristics is summarized in table 2.4.

Table 2.4: SDM addressing DSS issues

		SSM	**XP**	**SSADM**	**IE**	**CommonKADS**
DSS Components	**Users**	Yes	Yes	Yes	Yes	Yes
	Hardware		Yes	Yes	Yes	Yes
	Data	Yes	Yes	Yes	Yes	Yes
	Model	Yes	Yes	Yes	Yes	Yes
	User interface		Yes	Yes	Yes	Yes
	Intelligence component	Yes	Yes		Yes	Yes
DSS Characteristics	**Support decision maker**	Yes	Yes			Yes
	Support semi- or unstructured problems	Yes	Yes			Yes
	Flexible	Yes	Yes			Yes

	Problem evolves rapidly		Yes			Yes
	Combine models with data	Yes	Yes	Yes	Yes	Yes
DSS Development	**Complete**	Yes		Yes	Yes	Yes
	Quick hit		Yes			
	Staged	Yes	Yes			Yes

SSM

As stated earlier, SSM was developed to determine what the problem is, not necessarily how to solve it. This makes SSM suitable to develop systems to find solutions for unstructured problems and bringing the users, developers and experts together in deep discussions. One way to use SSM in DSS development is as a front end for other methodologies. The methodology can be used to develop complete and staged DSS. The users of SSM can start at any step because the steps would most likely be iteratively repeated.

XP

The short iterative cycles of XP might make this methodology a very good option for the development of DSS. Users and stakeholders are involved in the development and know the cost of changing requirements. Prototyping ensures that the users get what they want and continuous honest feedback ensures that the development stays on track. Small changes are made at a time to ensure changes in scope or problem description does not necessitate a major change and recoding in development. This methodology might not be the best option for the development of very big or complete DSS. One of the reasons for this would be that documentation does not play a big role in XP. The methodology does not really have fixed steps to perform and it is more focused on good development practices. The incremental development used in the methodology could be perfect for the development of DSS. Pair programming and writing tests before code may be unusual to some developers.

SSADM

Although the methodology might seem very rigid and unsuited for the development of DSS due to the waterfall approach, the methodology might be successfully used with other methodologies or by adapting it to suit the development of a specific DSS. The methodology takes both data and processes in consideration. The methodology might be best used for the development of large complete DSS because the methodology might not respond well to changes in the problem. The extensive documentation may also be a problem for some developers and will have an effect on the development time. Adaptations to the methodology are possible to suit the type of system developed and development objectives. Prototyping at a stage is recommended.

IE

This methodology also views data and processes as important, addressing it from the start. CASE tools can also be used, and an early part of the methodology deals with positioning the system with the strategic plan of the organization. The methodology uses a top-down approach, reducing problems into smaller manageable chunks. Different variants of IE exist and could fit the development of DSS.

CommonKADS

The methodology is very useful for extracting knowledge from experts and developing the knowledge base, and was developed to manage knowledge. The methodology tries to build systems that reason like humans or experts would. The methodology suggests a cyclical approach that is useful in the development of DSS and the cyclical approach can lead to the creation of new knowledge and make it adaptable to changing requirements, problems, or ways to solve the problems. The methodology addresses data and processes and the specific knowledge or intelligence component needed in DSS. Although the methodology does not focus on implementation and end-users, it involves the experts as a user as specified by the author as a decision support system component.

In a way, each of the SDM described in the previous sections could be useful in the development process of decision support systems. According to Table 2.4, XP and CommonKADS may appear to be the most suitable for DSS development. According to the literature, the incremental or spiral development these methodologies employ would be the most appropriate. The author is however of the opinion that the type of DSS to be developed, the organization for which it is developed and factors like available budget and development time, will dictate which SDM, or combination of SDM is most suited.

2.4 Summary

In Chapter 2, DSS were defined and the increased use of DSS was explained. The author defined the components and characteristics of DSS and stated that the development of DSS should address these components and characteristics. The three DSS development types described were: complete, staged or quick-hit development. The development of IS and the use of SDM were researched to look for the best way to develop DSS. SDM approaches were investigated to address the development of each DSS component. Five SDM were suggested and researched. The five SDM were compared with one another to determine how suitable these SDM are to develop DSS. In the following chapter it is explained how the research was conducted. The philosophy on which the research was based, the type of data, the research strategy, how data was gathered and analyzed will also be explained.

Chapter 3: Research Design

Decision Support Systems existed for a number of years. Yet, like the development of other systems, one might ask if the effort in developing this system was worthwhile? Are DSS used for the purpose they were intended and developed for? The author wanted to find out more on how DSS are developed and if these developments were successful. The aim of this research is to give the reader more insight into the development of DSS and to contribute to the existing knowledge regarding the development of these systems. The research could also assist persons involved in the development of DSS.

Another question the author wanted to address was if developers of DSS used System Development Methodologies (SDM) in the development process. The author wanted to know if the use of SDM had an effect on the success of the development of DSS. The author needed a way to research how DSS are developed.

This chapter deals with the underlying believes and assumptions the research is based upon and the way the research was conducted. Research aims to create new knowledge. This knowledge should be useful to other people. Research is valid if an appropriate process is followed to create the knowledge. Rigor is achieved by doing research in a systematic way. While a number of processes can be viewed as appropriate, the philosophy on which the research is based will dictate the validity (Oates, 2006). This chapter explains the philosophy on which the research was based. The author describes the research strategy and the approach used to collect, analyze, and use data in the research.

3.1 The research paradigm

A paradigm is a way of thinking about the world. Research is built upon an underlying paradigm. These thinking frameworks are based upon the way people view the world we live in, and the different ways we might choose to investigate it.

Mouton (1988) distinguishes between positivistic or anti-positivistic philosophies. If the methodology of the research were based upon the model of natural sciences, it would follow positivistic research. Anti-positivistic research would be based upon the social science model. Mouton (1988) also states that the paradigm of the researcher determines:

- If the research emphasizes the explanation of human behaviour or the description of human behaviour. To explain human behaviour, universal laws should exist to link events and behaviours to other events and behaviours. In other words, cause and effect. Anti-positivists acknowledge that it is not always possible to define all, if any, of these events or behaviours.
- If the researcher places emphasis upon observable or measurable behaviour. This implies that the factors should be observed and measured before scientific claims about the factors can be made. Anti-positivists try to understand meanings, significance, and symbols of human behaviour. According to anti-positivists, human behaviour is not always measurable or observable.
- The objectivity of the research. On the one side, for the natural scientist there is the notion of control typical in classic experimental research. On the other side, the social scientist would be in a sense "neutral" and "distanced" (quotations in original text) to gain the actor's perspective or an insider view to reconstruct the life world as accurately as possible.

Instead of only distinguishing between positivism and anti-positivism, the author will address three paradigms namely positivism, interpretivism, and critical research. This also corresponds with Oates (2006) and Myers and Avison (2002).

3.1.1 Positivism

According to Oates (2006), positivism is probably the oldest and most widely used paradigm of research. Positivism is based on the scientific principle that problems can be reduced into smaller solvable problems. The positivistic researcher wishes to make generalizations that can be shown regardless of the situation or actors involved. The scientific method has two basic assumptions.

The first assumption according to Oates (2006) is that the world is regular and ordered, not random. This would imply that our world has no random events and that all actions are based on rules. If we know the rules, we can predict actions and outcomes. Measurements lead to discoveries and models describe how our world works. Hypothesis testing confirms or refutes these models that describe our world. Hypothesis testing uses quantitative data with mathematical models and statistical analysis. According to Bryman (2001), theory should generate hypothesis that can be tested and facts provide the basis for laws. Bryman (2001) also states that positivism entails that only knowledge confirmed by the senses can genuinely be justified as knowledge.

The second assumption according to Oates (2006) is that we can investigate the world objectively. This implies that actions and events in our world are independent from humans. The rules of our world would exist with or without humans and the world could be investigated without personal feelings in an objective manner. Myers and Avison (2002) state that according to positivists, reality can be observed objectively through measurable properties that are independent of the observers and the researcher's instruments.

Oates (2006) also describes the three basic techniques of scientific method:

- Reductionism implies that complex problems can be broken into smaller, solvable problems. In this way, complex problems can be easier solved and understood.
- Repeatability implies that reliability is achieved through repeatability. Consistent and predicted results contribute to the trustworthiness of the research.
- Refutation implies that the more a claim or hypothesis can withstand tests to prove it wrong, the stronger the claim or hypothesis.

Oates (2006) also lists critique against the techniques mentioned above as:

- Many researchers believe that not all problems can be broken into smaller parts which, solved individually, would lead to success. Many researchers believe, especially in human activities, that the sum of the parts is smaller than the whole. It is necessary to view systems completely in their environment, and some aspects are not observable in a clinical setting.
- Humans are unpredictable, and so are their actions. Humans behave differently in new environments, or when they know they are being studied. Behaviours and actions are seldom repeated as the individual learns and grows.
- It may not always be desirable to generalize. People have different views. They do not always agree on what is the most important aspect or view. In the social world, regular laws and patterns might not exist, or are not easily observed.

Other philosophies like interpretivism and critical research exist because of these criticisms of positivistic research. Each of these philosophies has different views about our world and how to gain knowledge from it.

3.1.2 Interpretivism

Oates (2006) defines interpretivism research in Information Systems as being concerned with the understanding of the social context of an information system. It includes understanding the social processes of people involved in the development and construction of the system, and how the system influences and is influenced by its social setting. This implies that an information system is not isolated and cannot be studied in a controlled environment. The system forms part of an organization and as it is used by people, it is also developed by people, each person involved being unique.

Interpretive researchers believe that social reality cannot be discovered but only be interpreted. Social systems do not exist apart from humans and cannot be understood, characterized, or measured objectively and in a universal way. Researchers aim to understand phenomena by studying the meanings people assign to them (Orlikowski and Baroundi, 1991). Geertz (1973) commented on data collected in an anthropological study: “What we call our data are really our own constructions of other people’s constructions of what they and their compatriots are up to”. According to Bryman (2001), the interpretivist would try to place the interpretations gained through the study into a specific scientific frame. The researcher would interpret the interpretations of the subjects. These interpretations should then be interpreted in terms of the concepts, theories, and literature of the discipline.

Interpretivism does not prove or disprove a hypothesis, but tries to identify, explore, and explain how all the factors in a particular social setting are related and interdependent (Oates, 2006; Orlikowski and Baroundi, 1991).

Oates (2006) lists the characteristics of interprevistic research as:

- Multiple subjective realities. What we perceive as correct, real and knowledge, are constructed by our minds. It is either constructed individually, or in groups. This implies that there may exist more than one version of the truth. Perceptions depend on the group of people and the circumstances.

- Dynamic, socially constructed meaning. Knowledge is accessed and transferred through social constructs such as language, conscience and shared meanings (Myers and Avison, 2002). Different groups may have different understandings, meanings, or interpretations. According to Morgan (1983), subjective meanings, social-political and symbolic actions through which humans construct and reconstruct their reality are important to interpretive researchers. Meanings and intentional descriptions are important because they not only reveal the subjects' states of mind that relate to their behaviour but also create or cause the behaviour.
- Researcher reflexivity. This implies that the researcher cannot be neutral and that the assumptions, beliefs, values and actions of the researcher shape the research process. Researchers must therefore reflect on the way they influenced the research. The researcher's interactions with the participants in the research influence the understanding, meaning and practices of the participants.
- Study of people in their natural social setting. The research should not be carried out in an artificial environment. The research focuses on the people in their own world. The researcher studies the setting from the perspective of the participants' environment without imposing an understanding or expectations.
- Qualitative data analysis. The philosophy lends itself to qualitative data analysis and words, actions, metaphors, and images, to name a few, are gathered and researched.
- Multiple interpretations. It is expected that the research will not lead to only one fixed explanation. Explanations are discussed, as well as the reasons why some seem to have more evidence supporting it.

The aim of interpretive research would be to understand a group in a social setting and convince the reader of certain observations and meanings, beliefs and intentions. Interpretive researchers will rather be looking for plausibility than proof. The researcher should not take on the role of an objective reporter because the collection and analysis of the data involves the researcher's own subjectivity. The research should be trustworthy and findings clear from the data (Oates, 2006; Orlikowski and Baroudi, 1991; Walsham, 1995; Mouton, 1988).

3.1.3 Critical Research

Oates (2006) defines critical research in IS and computing as research into the power relations, conflicts and contradictions of IS. It aims to empower people by eliminating the sources of alienation and domination.

Myers and Avison (2002) and Oates (2006) state that these researchers believe that social reality is created and re-created by people. They state that this social reality have objective properties that dominate our experiences and ways of seeing the world. People can act to change their situation. Domination in the form of social, cultural and political constructs prevents people from changing their situation. Critical researchers feel interpretive research ignores the influence of powers and control that directs and justifies the way we see the world. Critical researchers aim to eliminate or transform the causes of alienation and domination to emancipate their subjects (Orlikowski and Baroudi, 1991).

Oates (2006) states a number of characteristics of critical research:

- Critical research aims to empower people.
- Researchers act more as activists than researchers.
- Assumptions taken for granted and existing patterns are challenged.
- Researchers do not aim to enhance power and control of managers or management efficiency.
- Researchers challenge the idea that technological development should adapt to people and societies. Furthermore, they challenge the idea that research can be done objectively and beliefs can be shaped or influenced by power or vested interests.

3.1.4 The philosophy followed in this research

Systems are developed and used by people. People have different perceptions on what makes a good system or what is a good process to follow in developing a system. DSS are developed for specific users with specific uses and needs. DSS are not generic systems widely applicable for a number of users like in the case of word processors. DSS are developed to aid the decision maker in specific problems. DSS use specific models and data to find solutions to help the decision makers. The users of DSS are normally the decision makers themselves.

It is the opinion of the author that the development of DSS are not executed in a consistent predefined way. Systems are not all developed in the same setting or circumstances or by the same person or teams. People have different backgrounds, training, experiences, and ideas. It is thus not possible to describe a standard setting in which systems are developed.

The author also feels that not all events or behaviour can be explained through other events or behaviour that might be the cause. Generalizations are seldom applicable. The author believes that there is a multitude of factors involved in the development process of DSS. It may not be possible to assign values to these factors or measure them. Neither would it be possible to keep certain factors constant while testing theories on other factors.

The development process is a learning process during which the developers gain insight and experience. This alone would limit the repeatability of experiments.

Studying humans in their environment entails a certain amount of subjectivity. What the researcher finds interesting or focuses upon may become apparent to the subjects being studied. This may cause them to act in a certain way around the researcher. Asking certain or specific questions may lead the subjects to certain answers or behaviour that may not have surfaced if not brought up by the researcher.

By acknowledging that social realities can only be understood through social constructs, the author also acknowledges the limitations regarding it. Not only can language be a barrier, but words or phrases may also easily be misunderstood if it is used in a different context or have a different meaning in another context.

For the reasons mentioned above, the author built the study on the interpretive philosophy. The philosophy on which the research is based determines the research strategy and data used in the research. The author explains in the next part the strategy and data that was used in the research.

3.2 The research strategy

3.2.1. Qualitative and quantitative research

Each research strategy is based upon quantitative or qualitative research. According to Bryman (2001) and Schurink (1988), quantitative research is positivistic in nature and tries to count and measure things (Dabbs, 1982; Mouton, 1988) whilst qualitative research aims to explore and describe (Schurink, 1988). According to Seaman (1999), qualitative research methods were designed to study the complexities of human behaviour and the principal advantage of using qualitative methods is that it forces the researcher to delve into the complexity of the problem and not just abstract the problem away.

Gorman and Clayton (1997) state that qualitative research is a process of enquiry that draws data from the context in which events occur. This is an attempt to describe these occurrences, determine the process in which events are embedded and describe the perspective of those that participate in the events. These descriptions use induction to give possible explanations based on what was observed. The goal would be to understand the subjects being studied from their perspective. It is not always clear what is observed. It might only become known to the researcher by becoming part of the subjects' worlds. It is more important to understand the aspects of reality than to create universal laws (Schurink, 1988).

Contrasts between qualitative and quantitative research are summarized in table 3.1 to illustrate how different qualitative and quantitative research is.

Table 3.1 Contrasts between qualitative and quantitative research Bryman (2001).

Quantitative	**Qualitative**
Numbers	Words
Researcher's point of view	Participants' point of view
Researcher distant and mostly uninvolved.	Researcher close to understand situation better.
Theory testing	Theory evolving
Static. Change and connections between events over time usually do not	Process. Research sometimes looking for change or connections that may

realize.	develop over time.
Structure. Usually highly structured to examine precise concepts and issues.	Unstructured to allow a better understanding of the meanings and concepts.
Generalizations	Contextual understanding
Hard, reliable data	Rich, deep data
Macro. Usually tries to uncover social trends applicable to a large community	Micro. Concerned with small-scale aspects of social reality.
Concerned with the behaviour of the subjects.	Concerned with the meaning of the subjects' actions.
Artificial setting	Natural setting

The author decided to use qualitative data in the research. The author wanted to get as much information as possible about the development of DSS and therefore felt qualitative data would be best to use.

Researchers should be cautious of the some of the most common critiques of qualitative research described by Bryman (2001):

- Being too subjective. Qualitative research may depend heavily on the researcher's own views of what is important. The researcher also builds a personal relationship with the subjects. By beginning in a relatively open-ended way and gradually narrowing down to the research question or topic, little information is sometimes given regarding why one area was focused upon above another.
- Difficult to replicate. The qualitative research is sometimes unstructured and relies on the ingenuity of the researcher. The researcher is the main instrument of data collection. What may seem as important may differ between researchers and can be influenced by the character of the researcher. Age, sex, personality and experience are examples of characteristics that differ in researchers.
- Problems of generalization. When a small number of subjects are investigated, it is difficult to apply the results to a larger group of subjects. Bryman (2001) argues, however, that qualitative research is not meant to be representative of a population but rather that the findings of the research are used to generalize to theory rather than to populations. The quality of the

theoretical inferences that are made out of qualitative data determines the assessment of generalizations.

- Lack of transparency. It is not always clear what and why the researcher has done and how the conclusion was reached.

3.2.2 The case study as research strategy

According to Oates (2006), each research question normally has a research strategy. The possible research strategies and the paradigms they are associated with are summarized in table 3.2.

Table 3.2: Research strategies and paradigms (Oates, 2006)

Research Strategy	**Description**	**Paradigm**
Surveys	The gathering of data from a large group in a standardized and systematic way. Statistics might identify patterns that may lead to generalizations.	Positivistic
Design and creation	Develops new IT products or artefacts.	Positivistic
Experiments	Studies cause and effect relationships, tests hypotheses, and seek to prove or disprove a causal link between a factor and an observed outcome.	Positivistic
Case studies	Focus on one instance of the item under investigation. This item may be an organization, department, system, forums, and so on. It seeks insight into the life of the case and its complex relationships and processes.	Interpretive Positivistic
Ethnography	Studies the culture and ways of a particular group of people and involves spending time in the field, partaking in activities.	Interpretive
Action research	Puts research into practice. An	Critical research

	action is performed and then reflected on before another cycle is started.	

In order to gain a better or deeper understanding of the development and developers of DSS, the author decided upon case studies as research strategy. Taking into consideration the complex nature of the development as well as the changing development environment, the author feels this strategy would yield the most knowledge. Case studies were the strategy that could be used in interpretive research and ethnography is not always possible.

A case study is an in-depth investigation of a discrete entity or case that can be a single setting, subject, social setting, group, collection, or event. The assumption is made that it is possible to derive knowledge of the wider phenomena from intensive investigation of a specific instance or case. The aim is to enable the researcher to effectively understand how the case operates or functions and to make clear the unique features of the case. Case studies are not data gathering techniques but a methodological approach that uses one or more data gathering techniques (Gorman and Clayton, 1997; Berg, 2001; Bryman, 2001).

The author wanted to know as much about the development of DSS as possible. Five developers were contacted for more information regarding the DSS they developed. Each project is regarded as a case study, thus giving five case studies, each as a unit of study on project level.

DSS investigated:

CASE STUDY 1

The first case study was performed on the development of a decision support system that screen plants for their invasive potential. It was described as an expert system designed to help decision makers institute regulations to restrict the import of certain types of plants based on the plant’s invasive potential. The decision support system is also used by students in a learning environment.

CASE STUDY 2

The second case study is built on supporting spatial decisions. It is built on geographical information systems and performs analysis on a lot of spatial data layers. One aim is to help the decision maker identify suitable land for low cost housing.

CASE STUDY 3

The third case study was a decision support system developed to aid decision makers and role players in the great lakes region in Africa during a time of conflict and elections to be held. The decision support system was developed with the inputs of political and military leaders and experts through facilitation sessions.

CASE STUDY 4

The decision support system developed was a system developed for decision makers in national government. It uses geographical information systems and data to help in decisions regarding geographical areas.

CASE STUDY 5

The fifth case study was done on a decision support system in the built and architectural environment. It provides support for investigations and modelling of scenarios in the built environment like traffic flows and the movement of the sun and its effect on buildings.

Most research strategies use one or more methods to collect data. In the following paragraphs the author will discuss which method was chosen to collect data and how it was used in the research.

3.3 Data collection methods

Myers and Avison (2002) list observation and fieldwork, interviews and questionnaires, documents and text, and the researcher's impressions and reactions as data sources for doing qualitative research. Gorman and Clyton (1997) mention that using more than one method on a case may lead to a better understanding of the case. The author used interviews in the case studies to generate data.

3.3.1 The interview as data collection method

A case study, where interviewing leads to the most information, can be called an interview case study. This data is collected through a number of interviews between the researcher and the subject. Focus groups are also popular in settings where subjects may be uncomfortable with or threatened by individual interviews. Interviewing allows the researcher to ask questions about what cannot be seen or observed, or to explore different explanations of what is observed, Gorman and Clyton (1997).

The author distinguishes between three types of interviews. The first type discussed will be the structured or standardized interview. This type of interview is associated with quantitative research (Gorman and Clyton, 1997) but could be used to generate additional information in qualitative research. It is characterized by a formal structured schedule (Berg, 2001), tightly controlled (Gorman and Clyton, 1997) with pre-determined questions (Gorman and Clyton, 1997; Schurink, 1988). The same questions (Berg, 2001) are asked in the same sequence (Schurink, 1988) and using the same wording for each interview (Gorman and Clyton, 1997). Answers are coded (Gorman and Clyton, 1997), and Schurink (1988) claims the answers are the only unstructured part of the interview. Berg (2001) notes that this type of interview is useful in cases where the researchers know exactly what they want to uncover. According to Schurink (1988) the main advantage to this type of interview would be that data are obtained in a systematically way and data comparisons could be done relatively easily. The disadvantage would be that only a small insight is gained into the subject's world. The extreme of a structured interview only yields quantitative data (Seaman, 1999).

Unstructured or semi-standardized interviews have some predetermined questions (Gorman and Clyton, 1997; Berg, 2001). These questions can be used by the researcher as guidelines (Schurink, 1988), to steer the conversation, or as starting point (Gorman and Clyton, 1997). The same sequence of questions asked (Schurink, 1988) or exact wording (Gorman and Clyton, 1997) does not have to be asked in each interview. The list of questions is used to make sure all points are covered (Gorman and Clyton, 1997; Schurink, 1988) and topics the researcher wanted to explore were discussed. The researcher, however, has the freedom to deviate from the questions (Berg, 2001). Schurink (1988) lists the advantages as having a schedule that provides a relatively systematic way to collect data and ensuring important data is not forgotten. According to Schurink (1988), the need for an able or trained interviewer is a disadvantage to this type of interview.

In informal, conversational type interviews, objectivity is very important to the researcher (Schurink, 1988). The researcher does not think of possible questions or topics before the interviews (Berg, 2001; Schurink, 1988), but questions develop and adapt as the discussion progress (Gorman and Clyton, 1997; Berg, 2001). Schurink

(1988) states that the interviewer's contribution to the interview should be limited to a minimum and that only a general theme should be used to start the conversation. Schurink (1988) also states that the objectivity gained would be the biggest advantage. The disadvantages would be that this type of interview is very time consuming and that vast amounts of data are created. Schurink (1988) suggests that the interviews should preferably be done by the researcher.

According to Walsham (1995) and Van Maanen (1979), the interviewee's construction is first-order data and the researchers should not assume that valuable data could be collected as long as they stay in the field. Second-order concepts would be the researcher's constructions. These concepts rely on good theory and insightful analysis and not only through the collection of more data.

In doing interpretive qualitative research with case studies as research strategy, the author decided on interviews as data generating method. The author believed that the most information regarding the developers, systems developed and the process of development could be gained through detailed explanations from the developer.

The author made telephonic appointments with the interviewees. Each developer was interviewed at a place of their choice and the interviews were mostly conducted in their offices.

The author introduced himself to the subjects and thanked them for their willingness to participate. Subjects communicated in the language of their choice. The length of the interviews varied with some interviews only lasting about 30 minutes whilst other 90 minutes.

The author asked permission for the interview to be recorded and explained that the reason was to make the transcription of the interview easier and to make sure valuable information is recorded and not forgotten. Seaman (1999) warns that interviewees might feel that the interview is an evaluation. For this reason, they may even be more nervous being recorded. The author's computer was used to record the interview. The author hoped that the interviewees would be more comfortable with a computer than with a voice recorder. The author used recording software downloaded from the Internet.

The author had an idea of the topics he wanted to research and wanted to keep the interviews on track with certain questions and make sure that all the topics were covered in the interview. The author was however worried about leading the interviewees with specific questions. The author decided on conducting unstructured or semi-standardized interviews. A few questions were drawn up before the interviews. These questions were asked in an informal way and sometimes in different words or contexts. In some interviews, new questions arose as the interview progressed. The author drafted the following questions to be asked in each interview:

- Please describe the last DSS that you worked on.
- Which aspects or characteristics of the system you described distinguishes this system as a decision support system?
- Please describe how this DSS was developed?
- Why did you choose to develop the system in such a manner?
- In your opinion, was the way you developed the decision support system effective?

These questions were used as described by Seaman (1999) as a guide to organize the interview. Notes on any subjects or comments that needed more explanations as well as further questions were written down during the interview. The author had the freedom to concentrate on the interview because all interviews were recorded. This also ensured that the author could re-listen to the interviews if needed. This freed the author as interviewer from trying to keep up with note taking during the interview.

3.4 Data analysis

Lincoln and Guba (1985) describe coding as a process of breaking up the text in the smallest pieces of information that can stand alone as independent thoughts without other information except the broad understanding of the context. Codes or labels that add meaning can be assigned to these small pieces of data that is words or even paragraphs. After the units or small pieces have been described, they can be grouped in new descriptive arrangements like tables, charts, or narratives. Out of this, themes and patterns may arise giving direction to the research.

According to Seaman (1999) coding helps a great deal in organizing and breaking up what is usually a very large amount of data. Although the process is creative, it is not necessarily completely subjective. All of the researcher's propositions should, however, be clearly and strongly supported in the data.

The author played back the recordings of each interview and transcribed each interview. This was done in a word processor. The author also made note of interruptions, pauses, or noises made by the interviewees. This aided in the understanding of the context of some answers and indicated were the subject paused to think of an answer, or was unsure.

The documents were imported into ATLAS.ti. This software aided in the coding of the interviews. Specific codes of interest and quotations was identified and marked or labelled with the software.

The author translated the quotations where it was needed. The author studied the codes and looked for codes with similar themes that could be grouped together. Appendix A contains the tables the author constructed and used.

The codes were pivoted in a table with codes on the vertical and interviews on the horizontal. Codes were compared for each interview and the author looked for similarities, frequent occurrences and contradictions. Content analysis and cross-case analysis provided the author with the basis for the author's propositions. A discussion of the results of these analyses is given in the following chapter.

3.5 Summary

This chapter gives the reader more insight into the way in which the research was conducted. The author explained why an interpretive study was done. By building the research on the interpretive philosophy, the author had the opportunity do a case study as research strategy. The author wanted to know as much about the development of DSS as possible and therefore used qualitative data to prevent loss of information. Interviews were conducted and recorded to be transcribed and coded. The author coded text by looking for themes in the data and comparing these codes to spot similarities and differences. This content and cross-case analysis forms the basis of the author's propositions. Results are discussed in the following chapter.

By describing the way the research was done and explaining why it was done in this manner, the author aims to give the reader a better understanding and background of the reasoning behind the results and conclusions the author reached in the following chapters. The information is useful to persons that would conduct similar studies.

Chapter 4: Results

After the author transcribed the interviews, analysed and coded the content of the interviews, similar codes were grouped together and a cross-case analysis was done. This was done by organizing codes in tabular form and comparing the answers or comments given by the interviewees per code or theme. See appendix A and appendix B to view the data in these tables. The author looked for similarities as well as possible contradictions in the answers of the subjects. The research concludes with these findings as the interpretations of the author. Please note that the author translated interviews for case studies 2, 3, 4, and 5.

4.1 How developers view DSS

Firstly, we need to know how developers see DSS. What do the developers see as the characteristics of DSS? Why do they believe DSS differ from other Information Systems?

4.1.1 Characteristics of DSS

The author summarized certain characteristics that were identified during the interviews. The first five characteristics listed below were discussed in more detail.

4.1.1.1 DSS should assist the decision maker

CASE STUDY 4 states that DSS should aid and not replace the decision maker "it supports decision making, but it doesn't make decisions". This corresponds with the literature that a computer cannot replace the human decision maker, but that the computer can help the decision maker choose the best solution for the specific problem or as CASE STUDY 3 states: it "should support a decision, a specific decision".

The support should be in the form of relevant and useful data and information, the creation of multiple scenarios or solutions, doing what-if analysis (CASE STUDY 3), and discarding non-applicable solutions. The DSS "should be able to very quickly discard not applicable solutions and concentrate on the applicable ones" (CASE STUDY 5), all in a short space of time: "It should be reached quickly, quicker than a human could" (CASE STUDY 5).

CASE STUDY 3 stresses that in strategic decision making "there are many risks, uncertainties, and usually the decision should have been made already... it is loaded with emotional and political issues". CASE STUDY 3 continues in saying "it [strategic decision making] touches on the moral values and integrity and the value system of the group of people that should make the decision". Again, the emphasis is on achieving quick results despite the many factors and uncertainties that should be considered. The impact of the decision should be kept in mind during the development and use of the DSS, as the decision most likely impacts people.

By restructuring the knowledge within a group, new knowledge might arise which can be useful. CASE STUDY 3 summarizes it as: "What we expect or anticipate is shared mental models, an understanding of the problem, a defined solution space, and a tangible take-home model that handles and sensibly performs what-if analysis and supports decision making." In other words, during the design and development phases, new ideas about solving the problem, factors creating the problem, and role players may become apparent through discussions, facilitation of workshops, consultation with clients or users, or in the drafting of documentation or notes. It reminds one of the rich picture diagrams and discussions of the Soft Systems Methodology which also aims to identify stakeholders, sources of conflict, problem areas, controlling bodies, and people involved.

The synergy of people working together in creating a system to solve a problem, possibly with the contributions of experts in the field (CASE STUDY 1, CASE STUDY 2, CASE STUDY 3), may provide new insights, and all inputs will contribute to the best way of solving the problem. Knowledge necessary may come from different fields (CASE STUDY 5).

It is clear that DSS can never replace decision makers. The decision support system should help the decision makers by generating multiple solutions, discarding non-applicable solutions and in a short time, recommend the best solutions. Using and creating DSS can further help the decision maker in finding new information, insights and ways to solve problems. Through discussions, documentation, consultation and design, new or hidden knowledge may emerge. These factors should be addressed in the development of DSS. SDM like SSM or CommonKADS may be useful to discover new or hidden knowledge by encouraging discussion and involvement during development. CASE STUDY 3 states that the problem is sometimes not figured out.

4.1.1.2 DSS operate and are developed in changing environments

DSS operate in a continuously changing environment. Needs or requirements (CASE STUDY 1, CASE STUDY 2), clients (CASE STUDY 2), the environment itself (CASE STUDY 2, CASE STUDY 4), and the problem (CASE STUDY 4) change. Even users' expectations change and according to CASE STUDY 5, things that were special or groundbreaking in the past are minimum requirements today. "Today's delighter is tomorrow's satisfier" (CASE STUDY 5).

Some of the comments made by the developers regarding change in the problem of systems are listed below:

- "Requirements change and if they don't change, in most cases there is something wrong, it means you are not talking to your client properly" (CASE STUDY 1).
- "The requirements and clients differ every time" (CASE STUDY 2).
- "If you developed something for a person, and that person is no longer there, and a new man is appointed that doesn't have that skill or exposure, then he cannot use the system as it was designed" (CASE STUDY 2).
- "Because the environment changes continuously" (CASE STUDY 3).
- "A person suddenly has a problem, and in six months it could be another problem, and that is what makes it so difficult with DSS, because your reports change continuously" (CASE STUDY 4).
- "Because our clients... continuously change the specifications" (CASE STUDY 4).

These changes make the environment for which DSS are developed and operate in complex. This corresponds with the literature stating that traditional system development methods may not be appropriate to develop DSS due to the long development time. Long development time increases the risk that the problem might change or evolve before the decision support system is completed, making the decision support system obsolete.

4.1.1.3 Complex environments

DSS operate in complex environments. The environments change, problems change, needs change, and even stakeholders or actors change (CASE STUDY 1, CASE STUDY 2, CASE STUDY 3, CASE STUDY 4, and CASE STUDY 5).

Managing these changes is part of what makes DSS and the development thereof complex.

In the fast-moving world today, new developments are common in all areas. Technology provides humans with new opportunities. Take the effect of the World Wide Web on commerce and communication for example. Globalization is a reality. These new opportunities are also accompanied by new and greater challenges and threats. Personal information security with high occurrences of fraud is but one example. Multinational companies trade globally. New products are available on the markets. Organizations experience great pressures to produce in greater quantities, faster and with increased quality. Decisions regarding production, marketing and distribution become more complex. Decisions need to be made quickly and the effect of a poor decision becomes greater and greater, and could ruin organizations. Customer needs and expectations are higher with each technological advance. The profile and structure of companies change. Laws and regulations change with governments, and people are becoming more conscious consumers and citizens. The information superhighway creates expectations regarding the availability of data and information. Organizations are in ways held accountable for decisions they make.

CASE STUDY 1 comments: “It is tacit knowledge and sometimes very difficult to write down... human knowledge is something quite unique and difficult to represent in a computer.”

CASE STUDY 3 states that “a problem one has is the complexity of the problem area... it is usually multi-disciplinary.” CASE STUDY 3 also describes strategic decision making with “risks... uncertainty... loaded with emotional and political issues”. There are a number of risks, uncertainties (CASE STUDY 3).

In saying that clients “should have a realistic series of suggestions and then concentrate on applicable solutions”, CASE STUDY 5 reminds us that DSS operate in situations where a number of solutions are possible, from which the decision maker should choose. CASE STUDY 5 also mentions: “My world is hybrid in the sense that you are going to use a combination of things to find solutions.”

A solution CASE STUDY 4 offers to deal with the complexity DSS is to “keep it small, keep it focused”. CASE STUDY 1 also states that the system was compliant in a small domain.

Thus to summarize, working with human knowledge and representing it in a computer program, with a problem area that is multi-disciplinary and which contains a number of possible solutions, makes the DSS environment complex. A suggestion made is to keep the decision support system small and focussed. This corresponds with the literature suggesting iterative or incremental development is more appropriate for the development of DSS.

4.1.1.4 DSS do not solve generic problems

DSS operate in environments or problems that are not generic (CASE STUDY 2, CASE STUDY 3, and CASE STUDY 4).
Comments made include:

- “Decision Support Systems are something you don’t find often... Not usually developed” (CASE STUDY 2).
- “Decision Support Systems... you are not going to generalize it... it is not necessarily that system that is going to be used widely” and DSS are “not necessarily used by other departments or organizations” (CASE STUDY 2).
- “Naturally, decision support is used for supporting a specific problem... So, it is not so generic” (CASE STUDY 3).
- “The information of the Decision Support Systems we develop is more sophisticated, it is not as generic as other Management Information Systems” (CASE STUDY 4).
- Client had specific needs (CASE STUDY 2) or problems (CASE STUDY 2, CASE STUDY 3, and CASE STUDY 4).
- Not mass produced and sold or developed for a wide range of users (CASE STUDY 2).

DSS are developed to support specific problems. Because the problems or environments are not generic, users, experts and information on the problem might be scarce. This can make DSS very difficult to develop. The developers may find it very difficult to understand the problem and therefore not comprehend the entire system unless the development makes provision for it. CommonKADS for example offer some solutions on how to deal with the knowledge base and knowledge models.

CASE STUDY 1 states: Human knowledge is unique and difficult to represent in a computer.

4.1.2 DSS components

The author could identify some of the components the developers mentioned during the interview. These components are listed below.

4.1.2.1 Knowledge used by DSS

It is difficult to gain an understanding of the problem or extract knowledge from experts (CASE STUDY 1, CASE STUDY 3). “Human knowledge is something quite unique and difficult to represent in a computer” (CASE STUDY 1). Knowledge may be tacit (CASE STUDY 1) or implicit (CASE STUDY 3). “Very often the experts don’t know how they make their decisions; it is tacit knowledge” (CASE STUDY 1).
CASE STUDY 3 said: “Every person has their own implicit knowledge about how to solve the problem.”

Repeating the knowledge extraction phase is an opportunity for parties to share and rethink knowledge (CASE STUDY 1, CASE STUDY 3). “It is a learning process for the developer and knowledge engineer and the expert... So this is an opportunity for both parties to engage in a knowledge share process” (CASE STUDY 1).

“And naturally it would be an iterative process, because when people start defining states they realize they really don’t have the right variable. In fact, this is not an important variable; it is a constant, an assumption to be made, in this way people go through an iterative process” (CASE STUDY 3).

CASE STUDY 1 mentions that a combination of informal and formal processes can be used to gain knowledge. “We just used a fairly informal approach just by interviewing and talking to the expert” (CASE STUDY 1).

CASE STUDY 1 was the “knowledge engineer” and said, however, the ideal would be to “have a methodology where the knowledge engineer doesn’t really need to understand the domain structure so automotive [sic] process and system, which facilitates some kind of elicitation, with the expert just answering questions and it builds the rule sets and generates the expert system”.

It may be that, because the knowledge is implicit or tacit, the developer only accesses a certain quantity of the knowledge at a time. Each time a subject is discussed, more knowledge may surface. It also may be that some knowledge is applicable in most problems, but in specific or very difficult problems, extra knowledge is used to solve the problem. CASE STUDY 3 states that the knowledge becomes explicit through the facilitation in the workshops.

Involvement of experts, users and owners are important in the development. The developer gains insight through interaction with them. The insight may be a better understanding of the needs for the system or type of user (CASE STUDY 4, CASE STUDY 2), knowledge, or ideas not previously mentioned (CASE STUDY 1, CASE STUDY 3).

There is a need to extract and then represent the knowledge (CASE STUDY 3). "Usually when the knowledge is explicit, you will have a graphical representation, say on a board" (CASE STUDY 3).

The developers agree that knowledge forms part of DSS. In Chapter 2 the intelligence component was defined as one of the components DSS consists of. The developers use different techniques to extract knowledge from the experts, implying that the knowledge extraction can be a very difficult process. Developers gain insight to the problem from the users, experts and owners. Discussing knowledge may lead to the creation of new knowledge.

4.1.2.2 Data as part of DSS

Data could be managed with a database (CASE STUDY 1, CASE STUDY 4), derived (CASE STUDY 3), or contained in layers (CASE STUDY 2). The developers said the following regarding the use and management of data:

- "So we had sort of a database" (CASE STUDY 1).
- "Data specification is to specify which data the system will need, how we can obtain the data, how we can keep it up to date and what are the processes and the handling of the database design" (CASE STUDY 4).
- "We tried to quantify the expert's qualitative feeling" (CASE STUDY 3).
- "And what it basically does is to look at a number of... data layers... it drills through those layers" (CASE STUDY 2).

Specific data is used (CASE STUDY 2) and most valuable, and should be standardized and reusable or shared (CASE STUDY 5).

Comments made include:

- "It requires specific data" (CASE STUDY 2).
- "Your data is the most valuable thing and (that) you are able to easily exchange the data... one thing that is not negotiable is if you exchange data, to use open standards..." (CASE STUDY 5).

Data may not exist (CASE STUDY 3) or may not be readily available (CASE STUDY 2). CASE STUDY 2 mentions the possibility of buying data from other organizations. This can be very expensive according to CASE STUDY 2. "Often you depend on other people [for the data]... but the moment that you have to purchase from a private practice or data vendor, you know it can become very expensive" (CASE STUDY 2).

"Data was not available, if you look at higher level decision support, strategic decision support systems, there is no data available else there would not have been a need for it [the system] on that level" (CASE STUDY 3).

It is clear that the developers view data or information as very important. In cases where no data is readily available, obtaining some data can be very hard or very expensive.

4.2.1.3 The importance of functions

"You have the information that those functions should be able to use" (CASE STUDY 2). It seems that CASE STUDY 2 is referring to mathematical or statistical models in this context. CASE STUDY 4 uses the term "Functional specification" [translated], although in its context it describes what the system should be able to do: "Functional specification that specifies the type of functions that the system should be able to do" [translated], (CASE STUDY 4).

In Chapter 2, models or model management systems are defined as one of the components of DSS. Although the developers mention functions, the author interpreted this as the models DSS use.

4.2.1.4 Users of DSS

- The expert actually wanted the system, was co-operative and willing to participate in the development (CASE STUDY 1). In other words the system should be seen as useful and able to help the decision maker in performing duties. This corresponds with the literature regarding the acceptance of systems by users.
- Build in transparency so users can understand how it is thinking (CASE STUDY 1).
- A good user interface is a fundamental characteristic (CASE STUDY 5).
- The system should be able to learn (CASE STUDY 5).

It is therefore clear that the developers acknowledge the importance of the user. The user can also be the expert in some cases. The user interface and intelligence aspect were defined as DSS components in Chapter 2. A system that can learn (CASE STUDY 5) probably relates more to Expert systems. It seems as if some developers don't really distinguish between Expert Systems and DSS.

4.2 The development of DSS

How did the subjects go about the development of DSS? What where the obstacles and what were the issues they stressed? The following tries to address these questions.

4.2.1 Difficulties during development

The developers mentioned the following during the interviews that was interpreted by the author as problems they experienced during the development:

- Limited time to develop (CASE STUDY 1).
- It is sometimes necessary to learn more before the development (CASE STUDY 2).
- Clients and needs differ (CASE STUDY 2) from project to project or change during development (CASE STUDY 2).
- Obtaining data can be difficult (CASE STUDY 2).
- You have to rely on others (for data, etc.) (CASE STUDY 2).
- A development plan (recipe) does not really exist (CASE STUDY 2).
- Development depends on the resources (for instance money) available (CASE STUDY 3).

- Users may not always be available (CASE STUDY 3), do not always understand the development or technical side (CASE STUDY 4), have too many choices (CASE STUDY 4), and care about the end results and not the steps needed in the development (CASE STUDY 4).
- Managing the development components becomes more difficult as the development team grows (CASE STUDY 4).
- Skills are not always available (CASE STUDY 5).

As stated in Chapter 2, some DSS should be developed as quickly as possible to prevent situations where the problem changes or evolves before the decision support system is complete. CASE STUDY 1 mentions this limited time. What complicates the matter is that skills, knowledge, information, data and development plans may not always exist. Users, owners and experts may not always be available and could change during the development. These problems can be overcome and might also be part of other development projects. The problem lies in the fact that these problems should be overcome in a project that is complex and have very limited development time. Developers themselves need to learn about the problem and have to rely on other people. Adding people to the development team does not usually solve these problems. Chapter 2 describes three possible types of DSS that can be developed. These are the complete, staged and quick-hit approach. Although the quick-hit may seem as the most appropriate with regards to the short development time, the assumptions of readily available data and clear-cut goals and procedures might not be met.

4.2.2 Approaches to developing DSS

CASE STUDY 1 uses the iterative approach in most projects, "I find with most projects the iterative approach works because the requirements change". CASE STUDY 1 also says: "I have used that same approach in many projects I have worked on". However CASE STUDY 1 still states "It depends on the system you are developing... The decision as to whether to use that process or not, it is not always a foregone conclusion..., if you have a large system which has got a high degree of criticality... the way you would go about that development would be different".

CASE STUDY 2 mentions "We have started looking at... what they call the Decision Enhancement Studio approach... that is basically a process to involve the user...". CASE STUDY 2 also mentions that "it requires you returning to things you completed and making adjustments... and then cyclical further. So the approach of: ”it is finished and you move on to the next and don’t return, is changing”, whilst talking about systems orientated architecture. However, CASE STUDY 2 feels: “The requirements and users differ every time, and you cannot always have a fixed recipe, you can suggest approaches... but there isn’t a recipe.”

CASE STUDY 3 uses a “structured workshop approach” with “facilitation process” in the workshops. The goal of these workshops is to “ask what-if questions”. CASE STUDY 3 also states: “And naturally it is an iterative process”, whilst discussing the process of development.

CASE STUDY 4 does not mention a specific approach used in the development. It was, however, said that “series of specifications go through several iterations”, whilst discussing the development process.

CASE STUDY 5 also does not mention a specific approach. CASE STUDY 5, however, says: “There is not a single technique one could say is the best ...what you should do is to look wider. In other words, you should look for wisdom from other professions, make it your own, and then look into your own profession, in other words, look from the outside in... You will usually find that you can solve much, much more difficult problems sometimes in a unique way”. CASE STUDY 5 suggests that “ we make it a closed world and a bit less ambitious and within those well-defined boundaries we design our systems... we look at our real problem and then make an abstraction to succeed in building a system”.

CASE STUDY 3 also touches on it by saying: “The result [Morphological analysis] is a structured problem. So in other words... the result is not a solution, it is a... structure laboratory... to try different concepts”.

It seems that the developers use the iterative and or incremental approach. This could be because of the fast changing environment. The complexity of the problem may also lead the developers to start building a decision support system that is only applicable in a small domain of the problem. Expansion follows success. CASE STUDY 4 suggests keeping the project small and simple.

According to CASE STUDY 1, CASE STUDY 3, CASE STUDY 4, and CASE STUDY 5, development is iterative. CASE STUDY 2 describes it as a “cyclical process” because of the changing environments or requirements. It keeps the client involved and short cycles promote sign-off: "There are key sign-offs that goes with demonstration" (CASE STUDY 2).

Developers repeat steps with small changes to get better results or performance (CASE STUDY 1, CASE STUDY 5) or to ensure that the clients are happy with the final product (CASE STUDY 4, CASE STUDY 5). "So, you know, it is an iterative approach where you use the phases to inform the development team of what must happen, but also assure the client that what they wanted... is what they will get" (CASE STUDY 4). "Your client is in fact your most valuable source of information. But he tells you... you must interpret it... and then you find at version 2, 3, 4, and 5 that the product only gets better" (CASE STUDY 5).

The developers might mention iterative development, but incremental seems to be more applicable. It seems as if prototyping is also used to ensure users are happy with the final product and promotes sign-off. For example: CASE STUDY 1 and CASE STUDY 5 advocate the idea of building smaller systems only applicable in a part of the problem. The system is fine-tuned until expected results are obtained. The system is then enlarged to cover a bigger part of the problem. One would rather think of incremental development in this case.

The statement of CASE STUDY 2: "Yes I would think years ago one could say this step is completed and then you move on to the next, but it is not longer the case today... you are often required to return to completed tasks to make adjustments after that, sort of cyclical", reaffirms that traditional development strategies is not applicable to the development of DSS.

"It should be a case of co-development and iterative, shorter iterative cycles are good because it keeps you in contact and you build an understanding between your client and yourself about what the issues are, where the system is going and if there are changes needed, what trade-offs or compromises have to be made" (CASE STUDY 1). This statement highlights some of the advantages of iterative development. The involvement of the client or users is similar to that of XP where users take part in the development and understand the implications of changes to the project.

It is clear that the developers view the incremental or iterative development as most appropriate for the development of DSS. The SDM suggested in Chapter 2 could be adapted to fit and to be used with incremental development. A rigid, waterfall approach to the development seems definitely inappropriate.

4.2.3 The development process

CASE STUDY 1

The first step was to interact with the expert and get an understanding and then map the knowledge into a format that could be represented. In other words, knowledge elicitation followed by knowledge representation followed by implementation. Brain storming and testing with the expert yielded a system compliant in a small domain.

A database was constructed and some form of predicate logic to represent the rules. Implementation involved extractive development that involved getting the system to work for a small number of species. Then, adapting the settings to fit more species and the system to behave as an expert would. The knowledge of the expert was crucial in the development.

The development was described as: “quite a lot of iterative development in that, there was analysis, design, implementation, testing over and over”. When asked why use iterative development, the developer replied: “just because it works. It is a learning process”.

The developer also said the expert has the knowledge but does not have the time to sit down and think it through in a structured way. The parties engage in a knowledge sharing process. “It is not like you can do a big design up front and analyse everything... and now you’ve got the specifications and can build the system. It doesn’t work like that because you learn as you go along”.

The same development approach was said to be used in many other projects. He admitted that the approach may not be appropriate for large systems or systems with high degree of criticality.

Some of the aspects of the development of CASE STUDY 1 align with the CommonKADS methodology. The analysis, design, implementation and testing performed over and over, and more, performed in an incremental way is similar to that of the spiral model used in CommonKADS. Furthermore, the developer also viewed the system more as an expert system that includes the involvement of an expert. This makes CommonKADS seem very appropriate.

CASE STUDY 2

The developer mentioned that the development depends on the purpose of the system. Development starts by focusing on the specific client with specific available data. Sometimes a lot of development time is spent on getting the data ready. Consulting people with domain expertise to ensure the correct data is used, was suggested. Mention is made of a “functional component” [translated], using the data. In other words the models used in decision making.

User requirements to involve the user to prevent the system from becoming obsolete or outdated, was said to be important.

The developer also stated that the development approach depends on the system to be developed. The first development steps normally includes: Design, determination of available technology, data, resources and skill. Development of tools might be necessary if it is not available.

According to the developer there is no recipe for development and this is a problem. Long ago, the development process consisted of completing stages before moving to the following stage. The developer states that sometimes you need to return to completed tasks to make adjustments.

A lot of emphasis was placed on getting the right data. Processes are also mentioned and preventing the system from becoming obsolete or outdated. The use of IE might be appropriate in terms of the importance of data, addressing the processes and still ensuring the system is useful to the organization.

CASE STUDY 3

According to the developer of the case study, the starting point in the development would be to constrain the problem area. This means the developers should know exactly what the problem is. By defining the problem, the main drivers or influencing factors can be identified and this leads to the discovery of possible variables. One way to discover the variables would be to gather as much as possible information from the experts regarding the problem. Concepts are clustered and named, and in this way, variables could emerge. It was stated this is an iterative process as variables change through discussions and by defining other states and factors.

Knowledge was said to be implicit and through facilitation, the knowledge becomes explicit. A problem is that experts have limited time available and it can be very difficult to arrange a workshop or facilitation session involving all possible experts at the same time.

The developer mentioned the importance of validation and verification of the models. Ideally another group of experts would validate and verify the constructed model. One problem is that experts in a specific field can be very scarce and therefore there are not enough data to validate the system.

Using groups of people and keeping the process generic, transparent, and very importantly, traceable, it is easy to understand why certain results are given by the decision support system.

Parts of the suggested development reminds of the Soft Systems Methodology. The developer also thought some of the techniques could be used with SSM. The importance of finding out what is the real problem to be solved, was emphasised. The facilitation process that involves the experts is again similar to that of SSM where rich pictures could be drawn and the CATWOE mnemonic to identify influencing factors.

Getting to know the environment in which the problem exists is important. The starting point is the identification and defining of a problem. The rest of the development could follow normal system development. The process involves a design phase with emphasis on the expectations of the users and a phase that specifies what is expected from the system (what functions it should perform). The development includes a technical specification and a data specification. According to the developer, specification goes through iterative cycles because the users continuously change the scope. Any changes in specifications should only be addressed in follow-up versions as new releases according to the developer.

The developer states that design and development should only make up one third of the development time. The other two thirds should be spent on testing and implementation as testing and stabilising the product in the intended environment are the most important component.

Documentation is very important as it is used as a communication tool. Users don't understand and don't care for technical information according to the developer. They want a system that performs as they expect. Because the users changes often during the development, requirements can easily change as well. Thorough documentation can clear ambiguities and keep the project on track.

The developer describes a quality test schedule designed by the developers for the users to ensure quality. This should not be seen as technical back-end test schedule but more of a guide to ensure that the users' expectations are met. This reminds of the testing schedule of the XP methodology where tests are designed before code to ensure developers completely understand what is expected.

The developer emphasised the importance of users signing off certain parts of the product during development. Prototypes were suggested. When users sign-of certain parts, it reduces the risk of a product not being used when users are not satisfied with a product.

The development of this DSS reminds of both a structured process like SSADM or IE with documentation playing a distinct role and possible sign-off stages. The duration of the development and the emphasis on documentation may not be very compatible with methodologies like XP or SSM.

CASE STUDY 5

According to the developer, a starting point would be to confine the problem and develop a system within boundaries. The correct information, structured in a usable way is very important to have. It is extremely important to listen to what the customer wants. This can be done by researching the users' needs and views. Another way is to also look at similar solutions (for example other commercial products) already available. Mapping available and possible technology options to the requirements of the users were suggested.

Although the first version of the product will never be perfect, improved products are achieved by interpreting the clients' feedback on each version of the product. It was stated that it is very difficult to get software error free.

The developer suggested to start with a very capable team, looking from different angles at the problem. The problem is constrained to find solutions in a small domain. User requirements are mapped to possible technology. The product is improved by listening and interpreting the users' comments on the product and releasing improved versions. During the interview a lot of focus was on the user or client.

Once again the idea of iterative or incremental development emerges. Prototyping is suggested in methodologies like XP, CommonKADS and SSADM.

The following comments were mentioned during some of the interviews that the author interpreted as steps in the development process:

- Interact with the expert (CASE STUDY 1).
- Knowledge extraction (CASE STUDY 1, CASE STUDY 3).
- Knowledge representation (CASE STUDY 1, CASE STUDY 3).
- Analysis (CASE STUDY 1).
- Design (CASE STUDY 1, CASE STUDY 2 and CASE STUDY 4).
- Implementation (CASE STUDY 1, CASE STUDY 3 and CASE STUDY 4).
- Testing (CASE STUDY 1, CASE STUDY 3 and CASE STUDY 4).
- Technical (CASE STUDY 2, CASE STUDY 4) and functional (CASE STUDY 4) specifications.
- Technology (CASE STUDY 5), resource, and skills availability (CASE STUDY 2).
- Data specification (CASE STUDY 4, CASE STUDY 5) and availability (CASE STUDY 2).
- Understand (CASE STUDY 1), model (CASE STUDY 3) or identify (CASE STUDY 4, CASE STUDY 5) problem.
- Identify variables (CASE STUDY 3).
- User requirements (CASE STUDY 4, CASE STUDY 5),
- Database design (CASE STUDY 4).
- Sign-off (CASE STUDY 4).
- Find out what is commercially available (CASE STUDY 5).

A number of developers mentioned design, implementation and testing as steps in development. These are also some of the steps of the system development life cycle and the steps Turban (1993) suggested for the construction of a complete DSS.

CASE STUDY 1 and CASE STUDY 3 also mentioned knowledge extraction and representation as steps in the development process. In Chapter 2, DSS were defined as having some intelligence component. It is therefore expected that the construction of DSS should somehow address this component and have ways to incorporate it in the DSS. Soft Systems Methodology and CommonKADS could be methodologies that can be useful in this process. CommonKADS has knowledge modelling as part of the methodology.

The developers do mention that the problem should be identified, modelled or understood. The technical and functional specifications together with looking into the available technology, resources and skill can be seen as part of the planning, analysis and research steps. CASE STUDY 5 makes a valuable point in suggesting doing research in the market place for products. Maybe other commercially available products could be sufficient or supplementary. The SDM suggested in Chapter 2 have steps to ensure the problem area is analysed and understood and that all requirements, if feasible, are addressed. SSM does not really address many development activities like construction and implementation. The methodology is more useful in determining what to do than how to do it. The methodology could however, be extremely helpful in understanding and analysing the problem. Methodologies like Information Engineering align the system with the strategic goals of the organization.

4.2.4 Techniques used in development

The developers are aware of different techniques and acknowledge that it may be useful in parts of the development (CASE STUDY 1, CASE STUDY 2, CASE STUDY 3, CASE STUDY 4, and CASE STUDY 5).

CASE STUDY 1 states that techniques are not “deterministic” and that one could “elicit knowledge” where needed. Interviews, conversations, brainstorming and predicate logic “to represent the rules” were used. It was stated that other techniques such as “flowcharts” could also have been used.

CASE STUDY 2 mentions “System Orientated Architecture”.

CASE STUDY 3 uses “workshops” and “facilitation”. “I think there are a number of those methodologies that could be applied to our techniques... there is definitely a lot of room for research to integrate a better researched process with these techniques that work very well for us.” [Translated], (CASE STUDY 3).

According to CASE STUDY 4 “the test schedule is very important”.

CASE STUDY 5 mentions “voice of the customer”, “rule-based reasoning”, “case-based reasoning”, “neural networks”, and “TREES”. CASE STUDY 5 also mentions that the wider the field of input, the better the solutions will be: “So one should make sure that in the team you assemble for development, that you have enough domain knowledge... What you should do is to look wider. In other words, you should look for wisdom from other professions, make it your own, and then look into your own profession, in other words, look from the outside in... you will usually find that you can solve much, much more difficult problems sometimes in a unique way”.

Due to the fact that DSS may not be developed continuously or regularly, the developers may not use or follow the techniques correctly (CASE STUDY 2). “Something we are not very good at is user requirement specification, because we don’t usually do it, it is often a case of prior learning.” [Translated], (CASE STUDY 2).

CASE STUDY 2 also mentions that “if you do it very seldom you don’t realize how important it is... so if you do it ten times, and then you realize at the tenth time you are really good and you know exactly what to do, to prevent you from forgetting to do it; but at the start, it’s a different story”.

Some of the techniques mentioned are similar to that used in the methodologies described in Chapter 2. Workshops and facilitation sessions could connect with the rich pictures drawn in SSM or the users’ stories in XP. Flowcharts, discussions, interviews, brainstorming are similar to many techniques used in development methodologies. If some techniques prescribed by methodologies are already used in practice, it might be easy for a developer to use these methodologies as methodologies are a combination of techniques and methods, performed in a specific sequence based on a belief about how to develop systems.

4.2.5 Tools used in development

Although some tools are used (Morphological and Bayesian analysis CASE STUDY 3), other developers have a need for more useful tools (CASE STUDY 1 and CASE STUDY 2).

According to the developer of CASE STUDY 1, “the documentation was really just my own documentation as we went along, because as the knowledge engineer, it was a case of I also had to understand the domain. The ideal would be to have a methodology were the knowledge engineer doesn’t really need to understand the domain structure so automotive [sic] process and system which facilitates some kind of elicitation with the expert just answering questions and it builds the rule sets and generates the expert system. But at that time there weren’t tools which we felt were actually going to work the way we wanted them to. I don’t know what the situation is now, but at the time it was a much less vigorous approach.”

CASE STUDY 3 uses their own tools like “post-its” and “smiley faces”.

“The requirements and clients differ every time, and you cannot have a fixed recipe. You can suggest approaches... and look at tools to involve the user in the process, but there is not a recipe”, CASE STUDY 2.

Even though the developer of CASE STUDY 2 felt that no recipe exists or can work every time, SDM may provide some help in prescribing certain tools and techniques. Users of the methodologies should be trained in the use of these tools en techniques. Developers might no know about a tool or technique that could be helpful or perfect for their specific development because so many methodologies exists, and even more tools and techniques.

4.2.6 The development time

DSS development normally lasts between 3 and 12 months (CASE STUDY 1, CASE STUDY 2, CASE STUDY 3, and CASE STUDY 5). CASE STUDY 2 mentions the pressure on the developers for shorter development times: “Nowadays there is pressure to do these things and within a shorter time; every time shorter and shorter times”.

CASE STUDY 4, however, states that it takes at least 6 to 9 months up to 36 months. "I have never developed a decision support system in less than six months" (CASE STUDY 4).

CASE STUDY 2 states that the development time depends on the system: "It depends on the size of the system, the number of functions, the difficulty in building it if you don't have the skills, and how much money you have available."

One gets the idea that the short development time is due to the fast changing environment, needs, problem, and users. Developers may see it necessary to develop the systems before too many changes take place and the new DSS being developed become unneeded or not applicable to the problem it was designed for. The system can become unneeded or not applicable even before the system is implemented. For the sake of shorter development times, developers might choose to limit or exclude certain steps in development, for example documentation, to speed up development.

4.2.7 The role documentation played

Comprehensive documentation is not always developed. Documentation could be personal notes (CASE STUDY 1, CASE STUDY 3), not priority (CASE STUDY 2), or insightful to the users or clients (CASE STUDY 3). There may be a need to follow a more formal process (CASE STUDY 3) or a need to use the documentation as a way to extract knowledge from experts (CASE STUDY 1).

The developers also mentioned the following:

- "Documentation was sort of ongoing but personal" (CASE STUDY 1).
- "It depends, for Morphological analysis, the documentation is very simplistic and we try to keep as much of the documentation in the model itself" (CASE STUDY 3).
- CASE STUDY 2 mentions that documentation is "very important" but also notes that "documentation is usually left till the end and usually neglected".
- CASE STUDY 3 states that "to experiment with the model, you get much more insight than you would with reading a 100 page document".
- According to CASE STUDY 3 "we feel it is a large meaningful system that is going to be developed, and then we have documentation with it".

- "At the time it was a much less vigorous approach... I would have to record that and just understand the reasoning for the whole... process" (CASE STUDY 1).

It seems as if documentation doesn't really get a very high priority. It also seems as if CASE STUDY 3 feels techniques like prototyping can provide more information than a detailed document. This is similar to the XP methodology that places a higher value on working software than on comprehensive documents: Customer collaboration over contract negotiations. CASE STUDY 2 warns against leaving documentation to the end and then neglecting it.

CASE STUDY 4, however, feels that documentation is an important communication tool that facilitates sign-offs on development progress. This seems contradictory to the other interviewees. On the one hand, one might think this might be because of the length of the development process. In other words, the developer might feel that it is necessary to have a well documented process and regular sign-offs due to a lengthy development process. Another explanation could be that this subject's development time is longer because of the detailed documentation.

CASE STUDY 4 further states: "our clients... continuously change the scope and you cannot do system development if you freeze it. What I am saying is that if system development has begun, then we deliver that product. Any changes in the specifications will mean one would have to look at a new release in the future... the client do a sign-off", [translated]. This might mean that the developer views the documentation as a security and blueprint for development to ensure what is agreed upon is delivered.

The size of the decision support system and its impact might determine the importance of documentation in the development. This may affect the choice of methodology to use as methodologies like SSADM places a lot of emphasis on documentation and XP don't.

4.2.8 The involvement of the client

In all the interviews, the client or users are stressed. There might not be definite lines between the client, owner, user or expert but they recognize the importance of involving them in the development. This includes finding out more about the problem, making the product more user friendly, or during testing. There is a need for effective, easy-to-use techniques and tools to involve the users. “It is a process where the user of the system is more closely involved... so it is rather important. I think we see it more and more, that people begin to use those approaches in the development of any DSS” (CASE STUDY 2).

Not all DSS are developed for the same user profile. The users need to make decisions, but differ in the computing experience: “Clients are not IT people” (CASE STUDY 4). CASE STUDY 4 also mentions the cases where systems are developed for clients, then used by the developers to report to the clients. “The best users are the developers” (CASE STUDY 4).

The interviewees describe inexperienced users (CASE STUDY 3, CASE STUDY 4 and CASE STUDY 5), experienced users (CASE STUDY 5), students doing research (CASE STUDY 1), as well as politicians and military personal (CASE STUDY 3) to name a few. According to CASE STUDY 3 it is not the “worker bee” that will use the system, but “more strategic high level people.”

Researchers and developers are increasingly realizing the importance of involving the users, clients and stakeholders in the development process. User acceptance can be improved by involving the users. Developers can have a better understanding of what users expect from the system. CASE STUDY 4 stresses that involving the users can lead to easier sign-off and delivery: “So you use the client in the phases to sort of sign-off that what is completed of the system is according to specifications, else you wait until the end only to hear ‘this is not what we wanted’.”

By keeping the stakeholders up to date, the developers reduce the chance of developers being unaware of changes in the organization, leading to new users with different expectancies. “You run the risk that people that were there in the beginning, and you are not involved for six to nine months, the people [in the organization] involved might change... and you return only to hear [from the new people] that the system is not what they wanted. Then you wasted six to nine months of development time” (CASE STUDY 4). “Clients differ continuously...” (CASE STUDY 2).

CASE STUDY 5 states that the client is the most valuable source of information and could possibly be used in research about the subjects. Comments and inputs from the client regarding the problem and system can lead to great insights. “Your client is actually the most important source of information. The customer might say it in a non-technical way, this you should translate in terms of your product and then you will find that with version 2, 3, 4, and 5, your product only improves." [Translated].

CASE STUDY 2 and CASE STUDY 5 both mention the importance of involving people from other disciplines in the development of DSS. A wider spectrum of knowledge may lead finding the best solution. "It comes from a domain area we call domain expertise, so you have to [get] people that work on these systems, work on these systems from a context that is their expertise" (CASE STUDY 2). "What you should do is to look wider. In other words, you should look for wisdom from other professions, make it your own, and then look into your own profession, in other words, look from the outside in... you will usually find that you can solve much, much more difficult problems sometimes in a unique way" (CASE STUDY 5).

CASE STUDY 3 stresses the importance of including the right people in the development: "A very, very important phase of the whole project is to include the right people. Sometimes it is intuitively easier." [Translated].

According to CASE STUDY 1 the "ultimate" user is someone who "needs to make a decision", though CASE STUDY 1 mentions the system being used by students as well.

4.2.9 The involvement of experts

DSS development needs the involvement and inputs of experts (CASE STUDY 1; CASE STUDY 2; CASE STUDY 3) or could be developed specifically for experts (CASE STUDY 1; CASE STUDY 3).

It seems that CASE STUDY 1 and CASE STUDY 3 also viewed DSS as expert systems. This coincides with the intelligent component we defined as being one of the components of DSS.

CASE STUDY 1 states: "Well, I think when we talk about expert systems, we are trying to capture the knowledge of the experts... Well the first step was to interact with the experts and try to get an understanding of how they think."

CASE STUDY 1 also makes a very important point: "...In that case I was quite fortunate that the expert was available and actually wanted the system...". CASE

STUDY 1 also mentions that the expert was part of the testing and played an important role in the development of the system.

CASE STUDY 2 states: “So you have to get people that work on those systems, work on those systems from a context that is their expertise.”

“It is very difficult to get the calibre experts for five days out of the office” according to CASE STUDY 3. “The user is always part of the workshop... so he is part of the stakeholders and the expert, expertise at the workshop” (CASE STUDY 3).

4.2.10 Factors determining development

The development (CASE STUDY 1; CASE STUDY 2) or type (CASE STUDY 5) of DSS developed depends on situations and environment.

The developers mentioned:

- “Fail tolerance, you know the reliability of that thing [DSS], becomes absolutely critical in navigation systems, airplane control systems, in motor control systems” (CASE STUDY 5).
- “It depends what kind of system you are developing” (CASE STUDY 1).
- “The decision as to whether to use that process or not, it is not always a foregone conclusion” (CASE STUDY 1).
- “There are normally steps but I think it depends on what you are doing... the steps change depending on what you are involved with” (CASE STUDY 2). CASE STUDY 2 also says that the systems depend on “what you want you want to do with it...the domain... [and the] capacity of the user”.

CASE STUDY 5 mentions that the type of system determines the effort. Take for example the difference between life-saving systems and a booking system. CASE STUDY 1 also states: “It is not always a foregone conclusion. For example if you have a large system which has got a high degree of criticality... The way you would go about that development would be different than how you would go about it in another system that is more or less a lower level of technicality” (CASE STUDY 1).

In the same way the size of the decision support system, the impact of the decision it supports or the available development time and budget affects the development process, the use of techniques like documentation and even the type of DSS to be developed. The methodology that is most applicable to the development might be dependant on some of these factors as well.

4.3 The use of In-house developed tools

If a tool that is necessary in the development or functioning of the decision support system is not commercially available, it is also developed (CASE STUDY 2; CASE STUDY 3; CASE STUDY 4).

The comments made include:

- "If you want to try something... and there is not something you can use, then you should do development work" (CASE STUDY 2).
- "Morphological analysis is unfortunately not available commercially, it is sort of in-house developed software that we developed together" (CASE STUDY 3).
- "Actually we use our own Decision Support System tools that we developed ourselves" (CASE STUDY 4).

4.4 Lifespan of DSS

Whilst some developers' work is still in use (CASE STUDY 1; CASE STUDY 3; CASE STUDY 5), other developers feel that the environment changes too fast for a decision support system to stay relevant (CASE STUDY 4). "So we have never implemented DSS that I know of that run longer than 12, 18 months to two years. Thereafter they stop it and start with a new system, because the environment changes the whole time" [Translated], (CASE STUDY 4).

4.5 Testing DSS

The developers do testing with experts (CASE STUDY 1; CASE STUDY 3) or users (CASE STUDY 4) and said the following:

- "There was quite a lot of iterative development in that; there was analysis, design, implementation, and testing over and over" (CASE STUDY 1).
- "After you validated and verified the model, you reach a point where you can implement the model", [translated], (CASE STUDY 3).
- "So it is a test schedule that is drawn up for the client. You tell the client that if the client takes this test schedule and uses it [the system], and this works... and this works, then it means system success. The test schedule is extremely important." [Translated], CASE STUDY 4.

- CASE STUDY 5 states that “to really debug software, is extremely difficult”.

4.6 The use of methods or methodologies

Although there are methodologies and methods available to use, and the developers are aware of them (CASE STUDY 3; CASE STUDY 5), they also express a need for methods or methodologies to use (CASE STUDY 1; CASE STUDY 3).

Comments made include:

- “There is a very good technique that they call Quality Function Deployment, if you are going to develop a new product.” [Translated], (CASE STUDY 5).
- “There is for example the Soft Systems Methodology... I think there are many of those methodologies that could be applied.” [Translated], (CASE STUDY 3).
- “I think something like Soft Systems Methodology could be successfully integrated using these techniques.” [Translated], (CASE STUDY 3).
- “There is definitely a lot of room for research to integrate a better researched process with these techniques that work very well for us.” [Translated], (CASE STUDY 3).
- “The ideal would be to have a methodology were the knowledge engineer doesn’t really need to understand the domain structure so automotive [sic] process and system which facilitates some kind of elicitation with the expert just answering questions and it builds the rule sets and generates the expert system” (CASE STUDY 1).

One developer is not sure if there are any usable methods or methodologies (CASE STUDY 2): “Do you mean, is there a recipe? Why do we follow the process? My opinion of how things worked in the past is that there is not really a recipe. That is the problem.”

CASE STUDY 2 further explains that “the requirements and users differ every time, and you cannot always have a fixed recipe, you can suggest approaches... but there isn’t a recipe.”

4.7 Summary

The author introduces the reader to the results and actual comments developers made regarding DSS, the development of DSS, and SDM. The author explained some of the view points and comments the developers made and summarized it according to appropriate topics. The author will summarize the findings of the study in the next chapter and explain what some of the limitations, contributions and objectives of the study were and recommend how future work could link to this study.

Chapter 5: Conclusion

5.1 Research objectives

The author aimed to determine what the best way would be to develop DSS. Some research indicates that millions of dollars have been spent on DSS that have never been used (Fuerst and Cheney, 1982; Cheney and Dickson, 1982; Robey, 1979). The author wanted to know if there is a way to develop DSS to prevent this from happening.

Better development leads to better products, higher user acceptance and user satisfaction. The right development should lead to lower production costs, and may prevent non-acceptance from clients, which cause great losses, especially for the developers and clients. Quality DSS can improve our lives by helping us make the right decision in a short time. The quality of the decision support system depends on the development of the decision support system.

The research objectives were to:

- Evaluate selections of SDM for their theoretical suitability to develop DSS.
- Describe the use, if any, of SDM during DSS development.
- Explain the reasons for the use or non-use of SDM during DSS development.
- Evaluate the influence of SDM on the quality of DSS.

5.2 Findings

5.2.1 The theoretical suitability of SDM to develop DSS

The author defined DSS as easy-to-use computer programs that use data, models, knowledge and skill of decision makers to aid the user in decision making in complex and ill-structured problems.

Three types of DSS can be developed, depending on the organization, the impact of a decision and size of the problem. The complete, staged, or quick-hit approaches to DSS development were discussed.

The author suggested that the development of DSS addresses the components of DSS. The components of DSS were identified as:

- Users – end-users as well as experts.
- Hardware.
- Data.
- Model.
- User interface.
- Intelligence component.

The components that were mentioned in the interviews were: Knowledge (intelligence component), data, functions (models) and users. The author interpreted some of the decision support system characteristics the interviewees mentioned as: Supporting the decision maker in complex environments and complex decisions, operating in changing environments, and not designed for generic problems.

The author interpreted that the development of DSS are difficult when the necessary building blocks are not available. These building blocks could be insufficient data or data that does not exist, skill, development plans and knowledge about the problem. Limited development time due to the ever changing nature of the problem area worsen this problem. It seems as if the developers use iterative or incremental development to help overcome some of these problems. The literature also suggests different development techniques like iterative for DSS.

SDM were researched in the quest to find the best way of developing systems. A definition of SDM according to Avison and Fitzgerald (2003) is a "recommended means to achieve the development, or part of the development, of information systems based on a set of rationales and an underlying philosophy that supports, justifies and makes coherent such a recommendation for a particular context. The recommended means usually include the identification of phases, procedures, tasks, rules, techniques, guidelines, documentation and tools. They might also include recommendations concerning the management and the organization of the approach and the identification and training of the participants".

Avison and Fitzgerald (2003) organized methodologies according to different development approaches. The author looked for approaches to address the development of each of the components. Methodologies in each approach was identified and discussed.

The suitability of the SDM to address the development of the different DSS as well as the components and characteristics is summarized in the following table:

Table 5.1: SDM addressing DSS issues

		SSM	XP	SSADM	IE	CommonKADS
DSS Components	**Users**	Yes	Yes	Yes	Yes	Yes
	Hardware		Yes	Yes	Yes	Yes
	Data	Yes	Yes	Yes	Yes	Yes
	Model	Yes	Yes	Yes	Yes	Yes
	User interface		Yes	Yes	Yes	Yes
	Intelligence component	Yes	Yes		Yes	Yes
DSS Characteristics	**Support decision maker**	Yes	Yes			Yes
	Support semi- or unstructured problems	Yes	Yes			Yes
	Flexible	Yes	Yes			Yes
	Problem evolves rapidly		Yes			Yes
	Combine models with data	Yes	Yes	Yes	Yes	Yes
DSS development	**Complete**	Yes		Yes	Yes	Yes
	Quick hit		Yes			
	Staged	Yes	Yes			Yes

Theoretically, the five SDM discussed should be helpful in the development of DSS. According to the developers interviewed, there exist a number of factors that determine the type of system as well as the type development best suited for the project. This is confirmed by the literature. Some of these factors may be the size of the development team, the size of the project, and the time or budget available.

Some of the SDM discussed can be adapted, have other variants, or can be used with other SDM to fit better to the development project. Unfortunately, no developer is completely skilled in all SDM that might be useful, or have the tools needed to use the methodology. This would mean that developers most likely would have to choose and empower themselves with one SDM that may still not be applicable to all development projects.

It seems that the developers used some techniques, tools and models in the development of the case studies, which also forms part of some SDM described in Chapter 2. If they already use some of the techniques, it could be possible to use SDM as SDM are tools, techniques and methods used in a certain sequence based on a certain philosophy regarding system development.

It seems that the case studies have a relatively short life span. This, and the changing and evolving nature of the problem area could be the reason behind the limited development time available. Developers might feel that the development time is too short to comply with a methodology and perform seemingly unnecessary tasks, like documentation, prescribed by the methodology.

The author is of the opinion that it is not possible to suggest only a single system development methodology for the development of decision support systems. It is undesirable to suggest only one SDM without considering the complexity of the decision support system, the complex environments they operate in and other development factors like limited development time and budget constraints. However, all of the methodologies described in Chapter 2 can be useful in the development of DSS.

5.2.2 The use, if any, of SDM in the development of DSS

Although some developers knew that SDM exist, none of the developers explicitly admitted to using any to develop DSS. It seems that they believe there is no single methodology that can perfectly address every aspect of the development.

5.2.3 Reasons for the use or non-use of SDM during DSS development

It is the opinion of the author that there are so many SDM that exist that developers might find it difficult to identify and then differentiate between methodologies. First identifying the approach and then selecting possible methodologies by means of elimination might solve the problem of identifying the most appropriate methodology.

Even though the methodology might fit the development perfectly, the lack of training for the developers in the methodology, and the unavailability of applicable tools may prevent developers from using a methodology.

Combining methodologies can possibly also provide a solution to existing methodologies that does not fit perfectly for the development of DSS. Yet again, the developers should know enough about all the methodologies to make sensible suggestions.

The author believes that developers want to do what they do best: Develop systems. Even though SDM can help them do it better, the author believes the developers might see the methodology as restricting them and burdening them with extra work like documentation, which in turn can lead to longer development times.

The developers might also feel that SDM are developed in an academic setting and that the development in practice actually differs completely to what the academics might think.

5.2.4 The influence of SDM on the quality of DSS.

Due to the fact that the developers did not use any SDM, it is not really possible to comment on the influence thereof on the quality of DSS.

5.3 Limitations

- Although only a small number of case studies were investigated, the author is of the opinion that sufficient data was gathered from substantially different case studies to perform a thorough analysis.
- Interpreting the words and phrases, especially in another language, could lead to the formation of wrong ideas.
- Interviewing was a new skill for the author.

5.4 Contributions to the field

The author looked at the development of DSS several decades after it first was introduced. Literature was researched to find the best angle for the development of these systems. The author made suggestions for the development of DSS using SDM, and investigated DSS that was developed to find out how it was developed in practice. This research not only contributes to the academic body of knowledge about using SDM in the development of DSS, but could also be useful to developers embarking on a new decision support system development.

5.5 Future work.

The author would suggest interviewing more DSS developers as well as revisiting those already interviewed for more in-depth discussions. Quantitative surveys could also be informative.

Appendix A: Cross-case analysis of interviews

▪ Code	Case study 1	Case study 2	Case study 3	Case study 4	Case study 5
▪ **AI**					▪ AI is mentioned a number of times during the interview. ▪ I believe you have AI in mind when you talk about decision making.
▪ **Approach**	▪ I find with most projects the iterative approach works because the requirements change. ▪ I have used that same approach in many projects that I have worked on. ▪ The decision as to whether to use that process or not, is not always a foregone conclusion. ▪ I find with most projects the iterative approach works. ▪ System being developed would determine the approach.	▪ Decision Enhancement Studio approach to involve user. ▪ Systems orientated architecture. ▪ Often a case of learning. ▪ Cyclical process.	▪ A structured workshop approach could be used. ▪ Facilitation approach used in workshops. • Goal of approach to do what-if analysis. ▪ Iterative during phases		▪ It is not possible to say one single technique is best. ▪ The developer should look as widely as possible for information and inputs from other professions. ▪ Reduce problem to manageable level.

▪ **Assist decision making**			▪ Should help in decision making. ▪ Strategic decision making touches on the moral values and integrity of the people making the decision. ▪ Strategic decision making involves more than black and white decisions. ▪ Decisions are loaded with emotional and political issues.	▪ Support decision making and not replace decision maker.	▪ Should assist decision maker with relevant and timely suggestions.
▪ **Case based reasoning**					▪ Solve a new problem based upon the solution of a previously solved, similar problem. ▪ Case based reasoning is one of the fields of AI
▪ **Changing environment**	▪ Requirements change.	▪ Our environment changes. ▪ The CSIR are dynamic, changing with needs. ▪ Clients and needs change continuously.		▪ In six months the problem could change. ▪ Reports change continuously. ▪ Our DSS run no longer than 24 months. ▪ Our environment	▪ Today's delighters are tomorrow's satisfiers.

				changes continuously.	
▪ **Characteristics of DSS**	▪ The expert...actually wanted the system ▪ The expert ...was cooperative and willing to spend time. ▪ The system was compliant in a very small domain. ▪ Build in a level of transparency so that users can understand how it's thinking. ▪ We had limited time. ▪ Human knowledge is... unique and difficult to represent in a computer. ▪ Requirements change.	▪ Not usually developed (found). ▪ Client has specific needs. ▪ Not necessarily be used by other departments or organizations. ▪ Not mass produced and sold or developed for a wide range of users. ▪ More focused. ▪ Not done frequently. ▪ Something unique. ▪ We don't do this (requirement spec) regularly. ▪ Addresses specific problems or problem aspects.	▪ Not generic ▪ The problem is multi discipline. ▪ There are a number of risks, uncertainties. The decision should have been made already. ▪ DSS support a specific problem. ▪ DSS support decisions. ▪ The problem is not already figured-out.	▪ Keep it small. ▪ Keep it focussed. ▪ Deliver as quickly as possible. ▪ Next year there'll be a new DSS. ▪ Keep the packaging small and focussed. ▪ Provides user with an answer. ▪ Goal specific. ▪ The DSS we do is sometimes once off. ▪ Not frequently used ▪ Information sophisticated ▪ Not generic ▪ Short term gain. ▪ Addresses a specific problem. ▪ Sometimes used once.	▪ A number of feasible good solutions are available. ▪ Solutions should be reached in a reasonable time. ▪ Decisions should be made faster than humans can. ▪ User interface is another fundamental characteristic. ▪ You can't use it without a user interface. ▪ An infinite number of solutions that can all be good. ▪ It should have a realistic series of proposals. ▪ Not applicable solutions should be quickly discarded. ▪ The system should be able to learn. ▪ Knowledge from different fields. ▪ Solutions structured in a useful way.

▪ **Client** ▪ **Owner** ▪ **Participants**		▪ Client driven ▪ Specific client with specific data needs. ▪ DSS depend on client. ▪ Get involved with people of other disciplines. ▪ Clients are involved in the process.	▪ The user becomes the owner ▪ It is extremely important to include the right people.	▪ Clients are not IT people.	▪ Client is the most valuable source of information. ▪ Voice of the customer important.
▪ **Closed world**			▪ Result of Morphological analysis is a structure for problems that can be used for the solving of problems and could be seen as a structure laboratory to test concepts. ▪ Bayesian network also provides a close laboratory for what-if questions.		▪ It could be described as a closed world. ▪ Within well defined borders we develop systems. ▪ Greater success is achieved in a closed world environment. ▪ You'll develop a small system that represents reality; you'll have a better chance of success if you use good techniques in a situation with definite boundaries.
▪ **Co-development**	▪ It should be co-development.				

▪ **Communication**				▪ Documents and development process helps with communication.	▪ Communication with users or clients important as it leads to new information.
▪ **Complex**	▪ Human knowledge is difficult to represent.		▪ A difficulty is the complexity of the problem. ▪ The problem is multi discipline. ▪ There are a number of risks, uncertainties. The decision should have been made already. ▪ The project is loaded with emotional and political issues. ▪ Subjective. ▪ Not always quantifiable.	▪ Our environment changes continuously. ▪ Ideas change.	▪ You need a wider perspective. ▪ An infinite number of solutions that can all be good. ▪ It should have a realistic series of proposals. ▪ You need combinations of things to find solutions.
▪ **Data** ▪ **Data shortage** ▪ **Database** ▪ **Data Exchange**	▪ We had a sort of database.	▪ Investigate a number of data layers. ▪ It requires specific data. ▪ You often depend on others for data. ▪ Purchasing data can be very expensive. ▪ Preparing the data.	▪ Data is derived. ▪ There are not really data available on that level. ▪ The environment is short on data	▪ We do data specification. ▪ What data we need and the database design and processes go together.	▪ You need information structured in a way that it is useful. ▪ Data is the most valuable thing and should be easily exchanged.
▪ **Depends**	▪ It depends on the	▪ It depends on what			▪ The response time

	system you are developing. ▪ The process to use is not always a foregone conclusion. ▪ The development would be different in other systems ...with lower levels of technicality.	you want to do with it. ▪ It depends on the domain. ▪ The decision support system depends on the client. ▪ The steps depend on what you are doing. ▪ The steps depend on with what you are busy with. ▪ The development time depends on the size of the system.			depends on the system. ▪ Some systems should not be allowed to err or crash. ▪ The type of system determines the effort.
▪ **Development problems** ▪ **Development team** ▪ **Development**	▪ We did not see it as a comprehensive system. ▪ Limited time.	▪ We don't usually do things (requirements spec) this way. ▪ It is sometimes necessary to study beforehand. ▪ Clients are involved in the process. ▪ Clients need differ from end results that need to be fixed when time and money could be depleted. ▪ Obtaining data can be expensive.	▪ Users necessary in development not always available.	▪ Users do not always understand the development or the technical side. ▪ Users have too many choices. ▪ Users care about the end results and not steps needed in between. ▪ Streamlining the development components gets more difficult as the development team grows. ▪ The development team is between 3	▪ The development team should have sufficient domain knowledge. ▪ Skills not always available.

		▪ You have to rely on others. ▪ A development plan (recipe) doesn't really exist. ▪ Changing users. ▪ Development depends on the resources (money) available. ▪ I have not developed a system on my own.		and 5 people.	
▪ **DOB**	▪ About 12 years ago.		▪ Developed in 2001		
▪ **Documentation**	▪ The documentation was ongoing but personal. ▪ It was written up after we finished. ▪ There is a need for a methodology to extract knowledge for expert without understanding the domain. ▪ It was not a vigorous approach.	▪ Unfortunately we are not very good with documentation. ▪ Documentation is very important. ▪ If you don't do it regularly (documentation) you tend to forget how important it is. ▪ Documentation is often left until the end and then neglected.	▪ Sometimes (during development) we follow a more formal documentation process. ▪ You do not get insight from reading a big document. ▪ Documentation at first ongoing, very basic, more personal notes. ▪ Large and significant projects would involve more documentation and a formal	▪ Documentation is an important communication tool.	

			documentation process.		
▪ **Drill**		▪ It drills trough the data layers.			
▪ **Duration**	▪ 3 Months ▪ We had limited time to produce a system.	▪ Normally 12 months depending on the money available. ▪ Market pressure to develop systems each time faster.	▪ 3-6 Months depending on the size although not intensive man hours.	▪ Not less than 6 to 9 months up to 36 months if it should be distributed.	▪ 12 months.
▪ **Expert** ▪ **Expert system**	▪ In expert systems, we try to capture the knowledge of the expert. ▪ The first step was to interact with the experts and try to get an understanding of how they think. ▪ Techniques that are not so deterministic. ▪ The experts don't know how they make their decisions, its tacit knowledge. ▪ We interact with the experts. ▪ The expert was available and actually wanted the system.	▪ Work with people that are experts in their domains. ▪ You need inputs from the expert.	▪ You try to get a group of experts. ▪ It is difficult to get experts involved throughout and out of their normal setting and other obligations.	▪	▪

	▪ Expert was part of the testing. ▪ Expert played an important role in the development. ▪ The expert system was designed. ▪ We try to capture the knowledge of the experts for expert systems. ▪ The first step was to interact with the experts.				
▪ **Functions**		▪ You have the functions and the data that the functions use. ▪ Functional components.		▪ There is a function specification phase to find the functions the system should have.	
▪ **In-house developed tool**		▪ If you want to try something, and the technology is not available, you must develop it yourself.	▪ It is not available commercially and sort of an in-house developed software.	▪ We use our own DSS tools, that we developed ourselves, to provide clients with answers for the clients that only want answers and not systems.	
▪ **Incremental development** ▪ **Iterative development**	▪ Extractive development cycle. ▪ We would adapt settings until it worked pretty much as the expert	▪ Cyclical process. ▪ Nowadays, development requires returning to previous steps in development	▪ Naturally, it is an iterative process.	▪ An iterative process where the developers are briefed and the clients assured that they are	▪ The first version is rarely the perfect version or product. ▪ Developers can get great feedback and comments

▪ **Prototype**	does. ▪ There was quite a lot of iterative development. ▪ There was analysis, design, implementation, testing over and over. ▪ I suppose the process could be defined as iterative. ▪ I find with most projects the iterative approach works. ▪ Shorter iterative cycles are good because it keeps you in contact. ▪ We actually went through several phases. ▪ Iterative is an opportunity for parties to share knowledge.	and making changes.		getting what they want. ▪ Key sign-offs from clients after demonstrations of prototype.	from users regarding previous versions. ▪ Product only improves with each new version.
▪ **Interaction**	▪ The users interact with the system.				
▪ **Knowledge** ▪ **Knowledge engineer**	▪ How they make their decisions is tacit knowledge. ▪ Sometimes very		▪ Probabilities were deduced from expert knowledge. ▪ Predicate logic		▪ The development team should have sufficient domain knowledge.

▪ **Knowledge** ▪ **extraction** ▪ **Knowledge representation** ▪ **Map**	difficult to write down. ▪ Combination (informal and formal process of knowledge extraction). ▪ Knowledge is required to make decisions. ▪ I was the knowledge engineer. ▪ In expert systems we try to capture the knowledge of the experts. ▪ A process to elicit the knowledge and put them together as ruled sense which interacts with each other to generate the knowledge required to make the decisions. ▪ A knowledge elicitation phase. ▪ Iterative is an opportunity for parties to share knowledge. ▪ Knowledge representation phase follows		represent the rules ▪ Each person involved has implicit knowledge about how the problem could be solved. ▪ Facilitation (in workshops) makes this knowledge explicit.		▪ Knowledge should come from as wide as possible to find best solution.

	elicitation. ▪ Human knowledge is difficult to represent in a computer. ▪ We try to somehow map the knowledge in a format we could represent.				
▪ **Learning process**	▪ Development is a learning process for the developer, knowledge engineer and expert. ▪ You learn as you go along and so it just seemed to be the only sensible way to do it.	▪ Often a case of prior learning.			▪ Systems should learn. ▪ One expects successful systems to learn.
▪ **Methodology**	▪ The ideal would be to have a methodology were the knowledge engineer doesn't need to understand the domain structure. ▪ It's not as if you can do just a big design up front… You learn as you go.	▪ The problem is that is that there is not really a recipe. ▪ The needs of the users change continuously and a fixed recipe would not work.	▪ Soft systems methodology ▪ I think there are a number of methodologies that might be applicable.		

▪ **Not generic**		▪ Our environment is different not repetitive. ▪ Not normally widely used. ▪ Specific problems.	▪ DSS does not support generic problems.	▪ DSS use information that is sophisticated and are not generic like other systems. ▪ Ideas change.	
▪ **Outcome of DSS**	▪ Useful. ▪ Educational. ▪ Worthwhile.		▪ Shared mental models, understanding the problem, defined solution space, tangible take-home models that handle and perform what-if analysis and support decision making.		
▪ **Politics**			▪ Hidden agendas. ▪ Hierarchy plays a role.		
▪ **Still in use**	▪ People are still referring to that work.		▪ Users actively use the model.	▪ Our DSS run no longer than 24 months.	▪ Yes.
▪ **Structured** ▪ **Unstructured**	▪ The expert hasn't had the time to really sit down and think through it in a structured way. ▪ Expert system techniques are not deterministic. ▪ In contrast to procedural and	▪ You have to anticipate.	▪ Other systems use processes that are already figured-out. ▪ It's more than just black and white.	▪ Other systems are used frequency wise and produce standard outputs. ▪ The information in DSS is more sophisticated.	

	algorithmic knowledge.				
▪ **Techniques**	▪ Need techniques that are not deterministic. ▪ Need techniques to elicit knowledge. ▪ Interviews. ▪ Talking to the expert. ▪ Brainstorming. ▪ Testing. ▪ Flowchart. ▪ Predicate logic. ▪ A Combination of formal and informal knowledge gathering.	▪ Systems orientated architecture.	▪ Morphological analysis. ▪ Bayesian networks. ▪ Do not have a format process in using the techniques or tools.		▪ Not possible to pronounce one technique the best. ▪ Quality function deployment. ▪ Neural networks. ▪ Rule based reasoning. ▪ Case based reasoning. ▪ Voice of the customer.
▪ **Testing**	▪ Expert is part of the testing.		▪ Verification and validation of model.	▪ Test schedule ▪ Quality testing.	▪ To debug software is extremely difficult.
▪ **Process**	▪ The first step was to interact with the experts. ▪ Understand the problem. ▪ Knowledge elicitation. ▪ Knowledge representation. ▪ We go through different phases. ▪ Analysis. ▪ Design. ▪ Implementation.	▪ There are normally steps but the steps change depending on what you are busy with. ▪ Design. ▪ Technical specifications and determine available technology. ▪ Resource and skills analysis. ▪ Availability of data	▪ Sort of a gut feel. ▪ Naturally it is an iterative process. ▪ Somewhat trial and error. ▪ Model problem. ▪ Identify variables. ▪ Facilitation approach used in workshops. ▪ Facilitation makes implicit knowledge about how the problem could be	▪ An iterative process where the developers are briefed and the clients assured that they are getting what they want. ▪ Firstly learner the environment in which the decision support system should operate. ▪ Identify the	▪ Try not to start from scratch. ▪ It is not possible to say one single technique is best. ▪ Problem identification ▪ Requirement analysis. ▪ Find out what is available. commercially ▪ Technology available.

	▪ Testing ▪ The development would be different in other systems ...with lower level of technicality. ▪ With Expert systems you need to use techniques which are not so deterministic and design expert systems in that whole process to elicit knowledge from the experts and put them together as ruled sense.	▪ The problem is that is that there is not really a recipe. ▪ Cyclical process.	solved, explicit. ▪ Explicit knowledge is represented graphically. ▪ Cluster concepts. ▪ What-if questions about problem possible through simulation. ▪ Validation of model. ▪ Verification of model. ▪ Implementation.	problem that needs to be addressed. ▪ User requirement. ▪ Design. ▪ Functional specification. ▪ Technical specifications. ▪ Data specification. ▪ Database design ▪ Number of iterations. ▪ Test. ▪ Implementation. ▪ Sign-off. ▪ Changes in specifications lead to new releases of system. ▪ Design and development should take up 1/3 of time and budget; testing and implementation should take up the other 2/3. ▪ Steps are generic steps used in all developments.	▪ You'll develop a small system that represents reality; you'll have a better chance of success if you use good techniques in a situation with definite boundaries. ▪ Iterations. ▪ You need information structured in a way that it is useful. ▪ User requirements and "Voice of the customer". ▪ Testing.
▪ **Tools**	▪ There weren't any tools which we felt were going to work. ▪ I don't know what the situation is	▪ Look for tools to involve the user.	▪ Facilitation. ▪ Smiley face. ▪ Post-it.		

	now.				
▪ **User** ▪ **User involvement**	▪ Someone who needs to make a decision. ▪ Students (interact with the system).	▪ Not normally used by other departments or organizations. ▪ User requirement specification. ▪ Users change. ▪ Trend to involve the users more. ▪ Changes are made to the system until users are happy with the system. ▪ The needs of the users change continuously.	▪ User an expert. ▪ Users not worker bees but strategically important people. ▪ Strategic decision making touches on the moral values and integrity of the people making the decision. ▪ A very important phase of the project is to decide how to include in the development. ▪ User always part of the workshop. ▪ User the one how needs to make a decision. ▪ User becomes the owner of the decision support system. ▪ Users were also expert.	▪ Users involved during requirement spec. ▪ User included/involved during development. ▪ Best users are the developers. ▪ Users change scope of system continuously. ▪ Clients are not IT people.	▪ User requirements and “Voice of the customer”. ▪ Satisfiers, dissatisfiers, and delighters. ▪ Communication with users or clients important as it leads to new information.

Appendix B: Summary of questions and answers.

	Case study 1	Case study 2	Case study 3	Case study 4	Case study 5
Please describe the last DSS that you worked on.	▪ Identify invasive plant species. ▪ Expert system. ▪ Data driven. ▪ Ruled sense.	▪ E-land. ▪ Geographical information system. ▪ Identifies usable land. ▪ Analysis on data layers.	▪ Political stability. ▪ Rule. ▪ Data shortage. ▪ What if analysis. ▪ Bayesian networks.	▪ IP Nerve centre. ▪ Reports.	▪ Library – Text book administration. ▪ CRONOS architecture. ▪ Reservations. ▪ Asset management.
Which aspects or characteristics of the system you described, distinguishes this system as a decision support system?	▪ Try to capture the knowledge of experts. ▪ Tacit knowledge. ▪ Experts. ▪ Not procedural or algorithmic knowledge. ▪ Elicit knowledge form experts put them together in a ruled sense. ▪ Make decisions.	▪ Very client driven. ▪ Not used by others. ▪ Not usually developed or found. ▪ Normally depends on the client. ▪ Addresses specific problems or problem aspects. ▪ Need specific data. ▪ Functions that uses the data. ▪ Sometimes learning takes place before development.	▪ Needs to make decisions. ▪ Specific problem. ▪ Not generic. ▪ Doesn't only support processes.	▪ Not frequently used. ▪ Information sophisticated. ▪ Not generic. ▪ Short term gain. ▪ Addresses a specific problem. ▪ Sometimes used once.	▪ DSS makes suggestion. ▪ Learning ▪ Knowledge from different fields. ▪ Information should be structured in a useful way ▪ DSS should provide reasonable suggestions. ▪ DSS should provide suggestions in a reasonable time. ▪ DSS should not do an exhaustive search. ▪ DSS should quickly discard not applicable solutions.

					▪ DSS should have an effective, friendly user interface. ▪ DSS should be able to do some learning. ▪
Please describe how this DSS was developed?	▪ Interact with experts. ▪ Understand problem. ▪ Extract knowledge (formal and informal way). ▪ Map knowledge. ▪ Represent knowledge. ▪ Implement. ▪ Test. ▪ Went through several phases. ▪ Analysis. ▪ Design. ▪ Implementation. ▪ Testing. ▪ Quite a lot of iterative development. ▪ User part of the testing. ▪ 3 months to develop.	▪ Depends on type. ▪ Focus on client and data. ▪ Data analysis. ▪ Inputs from experts ▪ Functions also important. ▪ User requirements. ▪ Designs from experts. ▪ Design. ▪ Available technology. ▪ Resources and skills available. ▪ Cyclical process. ▪ Availability of data ▪ The problem is that is that there is not really a recipe. ▪ Nowadays, development requires returning to previous steps in development and making changes. ▪ Changes to the	▪ Workshop knowledge extraction. ▪ Building of model. ▪ Verification. ▪ Validation. ▪ Implement. ▪ Naturally it is an iterative process. ▪ Somewhat trial and error. ▪ Model problem ▪ Identify variables. ▪ Facilitation approach used in workshops. ▪ Facilitation makes implicit knowledge about how the problem could be solved, explicit. ▪ Explicit knowledge is represented graphically. ▪ Cluster concepts. ▪ What-if questions about problem	▪ Problem statement. ▪ User requirements. ▪ Design. ▪ Functional spec. ▪ Data specs. ▪ Technical specs. ▪ Iterations. ▪ Sign-off. ▪ Test. ▪ Implement. ▪ Handing over. ▪ Design and development should take up 1/3 of time and budget; testing and implementation should take up the other 2/3.	▪ Problem definition. ▪ Customer input. ▪ Iterative development. ▪ Try not to start from scratch. ▪ It is not possible to say one single technique is best. ▪ Problem identification. ▪ Requirement analysis. ▪ Find out what is available commercially. ▪ Technology available. ▪ You'll develop a small system that represents reality; you'll have a better chance of success if you use good techniques in a situation with definite

		system are made until the user is happy.	possible through simulation. ▪ Validation of model. ▪ Verification of model. ▪ Implementation. ▪ Iterative process.		boundaries. ▪ You need information structured in a way that it is useful. ▪ User requirements and "Voice of the customer". ▪ Iterations. ▪ Testing. ▪ Open standards and data sharing/exchange/reuse (if applicable) very important.
Why did you choose to develop the system in such a manner?	▪ It works. ▪ Learning process for developer, knowledge engineer and the expert. ▪ Hasn't had the time to really sit down and think through it in a structured way. ▪ You learn as you go along. ▪ Limited time. ▪ Did not see it as a comprehensive system. ▪ Extract knowledge. ▪ Test.	▪ No recipe available- that is a problem.	▪ Trial and error	▪ For communication purposes. ▪ Environment changes continuously. ▪ Ensuring a successful handing over.	▪

In your opinion, was the way you developed the decision support system effective?	▪ Yes. ▪ Achieved what we wanted to achieve. ▪ Still in use. ▪ I have used that same approach in many projects that I have worked on. ▪ It is not always a foregone conclusion. ▪ The iterative approach works because the requirements change. ▪ Shorter iterative cycles keep you in contact and build an understanding between you and the client.	▪ Changing environment and client needs limits success.	▪ Yes. ▪ Still in use.	▪ Success limited to users. ▪ Keep it small. ▪ Keep it focussed. ▪ Deliver as fast as possible. ▪ Next year there'll be a new DSS. ▪ Keep the packaging small and focussed. ▪ Investment in big systems not worth it.	▪ Yes. ▪ Still in use. ▪ Reduce system until successful. ▪ Success is greater in smaller systems-risk reduced, less time and resources required.
Did SDM play a role in the development of DSS?	▪ No	▪ No	▪ Not in this project but need for a more scientific process or improve scientific credibility. ▪ Methodologies might be successfully integrated with techniques.	▪ No	▪ No ▪ It is not possible to say a single technique is the best, it is essential a continuum of techniques.

References

Adams, R.J., Bloor, I., Collier, M., Meldrum, M., Ward, S. (1993). Decision Support Systems and performance assessment in academic libraries – British Library Research. London: Bowker Saur.

Alavi, M., Henderson, J.C. (1981). An evolutionary strategy for implementing a desicion support system. Management Sciences, Vol. 27, No. 11, pp. 1309-1323.

Alavi, M., Joachimsthaler, E.A. (1992). Revisiting DSS implementation research: A meta-analysis of the literature and suggestions for researchers. MIS quarterly, Vol. 16, No. 1, pp. 95-116.

Allsopp, D.J., Harrison A., Sheppard, C. (2002). A database architecture for reusable CommonKADS agent specification components. Knowledge-Based Systems, 15, pp. 275-283.

Angioni, M., Carboni, D., Pinna, S., Sanna, R., Serra, N., Soro, A. (2006). Integrating XP project management in development environments, Journal of Systems Architecture, 52, pp. 619-626.

Avison, D., Fitzgerald, G. (2003). Information system development – methodologies, techniques and tools. Third edition. McGraw Hill, UK.

Beck, K. (1999). Extreme Programming Explained. Addison-Wesley.

Berg, B.L. (2001). Qualitative research methods for the social sciences. 4th edition. Allyn & Bacon, Needham heights MA, USA.

Berrisford, T., Wetherbe, J. (1979). Heuristic development: A redesign of systems design. MIS quarterly, Vol. 3, No. 2, pp. 11-19.

Bhargava, H.K. (1999). Beyond spreadsheets: Tools for building Decision Support Systems. IEEE Computer Society, Vol. 32, No. 3, pp. 31-39.

Boland, R.J. Jr. (1978). The process and product of system design. Management Sciences, Vol. 24, No. 9, pp. 887-898.

Bryant, S., Romero, P., Du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. International Journal of Human-Computer Studies, Vol. 66, Issue 7, pp. 519-529.

Bryman, A. (2001). Social Research Methods. Oxford University press, New York.

Chae, B., Paradice, D., Courtney, J.F., Cagle, C.J. (2005). Incorporating an ethical perspective into problem formulation: implications for decision support systems design. Decision Support Systems, 40, pp. 197-212.

Checkland, P. (1981). Systems Thinking, Systems Practice. John Wiley & Sons, Chichester, UK

Checkland, P., Scholes, J. (1990). Soft Systems Methodology in Action. John Wiley & sons, Chicester, England.

Cheney, P.H., Dickson, G.W., (1982). Organizational characteristics and information systems: an investigation. Academy of Management Journal 25, pp. 170-184.

Courbon, J., Drageof, J., Jose, T. (1979). L'Approach evolutive. Information El Gestion No. 103, pp. 51-59, Institute d'Administration des enterprises, Grenoble, France.

Dabbs, J.M. Jr, (1982). Making things visible. In J. Van Maanen (Ed.), Varieties if Qualitative research. Beverly Hills, CA: Sage.

Davis, A. (1992). Practical information Engineering (The management challenge). Pitman, London.

Davis, W.S. (1983). Tools and techniques for structured systems analysis and design, Addison-Wesley. Miami University, Oxford, Ohio.

De Hoog, R., Benus, B., Vogler, M., Metselaar, C. (1996). The CommonKADS organizational model: content, usage and computer support. Expert Systems with applications, 11, pp. 29-40.

Duffy, N.M., Hough, P.K. (1985). Decision Support Systems: A view from the top. WITS Business School Research Paper 6.

Eierman, M.A., Niederman, F., Adams, C. (1995). DSS Theory: A model of constructs and relationships. Decision Support Systems, 14, pp. 1-26.

Fazlollahi, B., Parikh, M.A., Verma, S. (1997). Adaptive decision support systems. Decision Support Systems, 20, pp. 297-315.

Fick, G., Sprague, R.H. Jr. (1980). Proceedings of an International Task Force Meeting, Decision support systems: Issues and challenges. International Institute for Applied System Analysis Vol. 11.

Fitzgerald, B., Russo N.L., Stolterman, E. (2002). Information System Development: Methods in action. McGraw-Hill, UK.

Ford, S., Aouad, G., Kirkham, J., Brandon, P., Brown, F., Child, T., Cooper, G., Oxman, R., Young, B. (1995). An information engineering approach to modelling building design. Automation in Construction, Vol. 4, pp. 5-15.

Fuerst, W.L., Cheney, P.H. (1982). Factors affecting the perceived utilization of computer-based decision support systems in the oil industry. Decision Sciences 13, pp. 554-569.

Gane, C., Sarson, T. (1977). Structured Systems Analysis: Tools and Techniques. 2nd impression. Improved System Technologies, New York

Geertz, C. (1973) The interpretation of cultures. Basic Books, New York.

Geraghty, P.J. (1993). Environmental assessment and the application of expert systems: an overview. Journal of Environmental Management 39, pp. 27-38.

Gorman, G.E., Clayton, P. with contributions from Rice-Lively M. L. and Gorman, L. (1997) Qualitative research for the information professional a practical handbook. The Library Association, London.

Hackathorn, R.D., Karimi, J. (1988). A framework for comparing Information Engineering methods. MIS quarterly, Vol. 12, No. 2, pp. 203-220.

Hedin, G., Bendix, L., Magnusson, B. (2005). Teaching extreme programming to large groups of students. The Journal of Systems and software, 74, pp. 133-146.

Huisman, H.M. (2000). The development of systems development methodologies: a South African experience. South Africa.

Inmon, W.H. (1986). Information Systems Architecture. Prentice-Hall, Englewood Cliffs, N.J.

Jackson, I.F. (1986). Corporate Information Management. Prentice-Hall, Englewood Cliffs, N.J.

Janssen, R. (1992). Multiobjective decision support for environmental problems. Dordrecht, The Netherlands: Kluwer Academic publishers.

Keen, P.G.W. (1980). DSS: A research perspective. Working paper number C15R-54, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Keil, M., Beranek, P.M., Konsynski, B.R. (1995). Usefulness and ease of use: field study evidence regarding task considerations. Decision Support Systems 13, pp. 75-91.

King, W.R. (1985). Strategic planning for IS: The state of practice and research. MIS Quarterly, Vol. 9, No.2, Editor’s comment, pp. vi-vii.

Kingston, J.K.C. (1998). Designing knowledge based systems: the CommonKADS design model. Knowledge-Based Systems, 11, pp. 311-319.

Kivijärvi, H., Zmud, R.W. (1993). DSS implementation activities, problem domain characteristics and DSS success. European Journal of Information Systems, Vol. 2, No. 3, pp. 159-168.

Klein, M., Tixier, V. (1971). SCARABEE: a data and model bank for financial engineering and research. In Proceeding IFIO World Congress, North Holland.

Klein, M.R., Methlie, L.B. (1995). Knowledge-based Decision Support Systems with applications in business. Second edition, Chichester: Wiley, England.

Kloditz, C., Rijsberman, F., Werners, S., Westmacott, S. (2000). Development of the user interface: coral-Curacao, coral-Maldivesand COCOMO. In Integrated Coastal Zone Management of Coral Reefs: Decision Support Modeling (K. Gustavson, R. M. Huber and J. Ruitenbeek, eds), pp. 153-158. Washington, DC: The World Bank.

Lane, D.C., Oliva, R. (1998). The greater whole: Towards a synthesis of system dynamics and soft systems methodology. European Journal of operational research, 107, pp. 214-235.

Layman, L., Williams, L., Damian, D., Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. Information and Software Technology, 48, pp. 781-794.

Lea, W., Uttely, P., Vasconcelos, A.C. (1998). Mistakes, misjudgements and mischanches: Using SSM to understand the Hillsborough disaster. International Journal for Information Management, 18, pp. 345-357.

Lincoln, Y.S., Guba, E.G. (1985). Naturalistis inquiry. Newbury Park, CA: Sage Publications.

Lu, H., Yu, H., Lu, S.K. (2001). The effects of cognitive style and model type on DSS acceptance: An empirical study. European journal of operational research 131 (2001), pp. 649-663.

Mackay, H., Carne, C., Beynon-Davies, P., Tudhope, D. (2000). Reconfiguring the user: Using Rapid Application Development. Social Studies of Science, Vol. 30, No. 5, pp. 737-757.

Mann, R.I., Watson, H.T. (1984). A contingency model for user involvement in DSS development. MIS quarterly, Vol. 8, No. 1, pp. 27-38.

McKeen, J.D. (1983). Successful development strategies for business applications systems. MIS quarterly, Vol. 7, No. 3, pp. 47-59.

Meador, C.L., Guyote, M.J., Keen, P.G.W. (1984). Setting priorities for DSS Development. MIS Quarterly, Vol. 8, No. 2, pp. 117-129.

Moreau, E. (2006). The impact of intelligent decision support systems on intellectual task success: An empirical investigation. Decision Support Systems, 42, pp. 593-607.

Morgan, G. (Ed). (1983). Beyond method: Strategies for social research. Sage, Beverly Hills, CA

Mouton, J. (1988). The philosophy of qualitative research. Centre for Research Methodology, Human Sciences Research Council, RSA.

Myers, M.D., Avison, D. (2002). Qualitative Research in Information Systems. A Reader. Sage, London, California, New Delhi.

Newell, A., Simon, H.A. (1972). Human problem solving. Englewood Cliffs, NJ: Prentice-Hall.

O'Leary, D.E. (1998). Knowledge acquisition from multiple experts: An Empirical study. Management Science, Vol. 44, No. 8, pp. 1049-1058.

Oates, B.J. (2006). Researching information systems and computing. Thousand Oaks, UK, pp. 341.

Orlikowski, W.J., Baroudi, J.J. (1991). Studying information technology in organizations: Research approaches and assumptions. Information Systems Research.

Parker, M., Thompson, J.G., Reynolds, R.R., Smith, M.D. (1995). Use and misuse of complex models: examples from water demand management. Water Resources Bulletin 31, pp. 257-263.

Porter, M.E. (1985). Comparative advantage: Creating and sustaining Superior Performance. New York: The Free Press.

Robey, D. (1979). User attitudes and MIS use. Academy of Management Journal, September, pp. 527-538.

Sauter, V.L. (1997). Decision Support Systems: an applied managerial approach. John Wiley and Sons, Canada.

Schurink, E.M. (1988). The methodology of unstructured interviewing. Centre for Research Methodology, Human Sciences Research Council, RSA.

Seaman, C.B. (1999). Qualitative methods in empirical studies of software engineering. IEEE transactions on software engineering, 25, 4, pp. 557-572

Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Shura, R., Carison, C. (2002). Past, present and future of decision support technology. Decision Support Systems, 33, pp. 111-126.

Simon, H.A. (1973). The structure of ill-structured problems. Artificial Intelligence, 4, pp. 181-201.

Sprague, R.H., Jr. (1980). A framework for the development of decision support systems. MIS quarterly, Vol. 6, No. 4, pp. 1-26.

Steinberg, D.H., Palmer, D.W. (2004). Extreme Software Engineering: A hand-on Approach. 1st edition. Upper Saddle River, NJ, Pearson Education, Inc.

Studer, R., Benjamins, V.R., Fensel, D. (1998). Knowledge engineering: Principles and methods. Data and Knowledge Engineering, 25, pp. 161-197.

Turban, E. (1988). Decision support and expert systems. Managerial perspectives. 1st edition. New York, MacMillian.

Turban, E. (1993). Decision support and expert systems – Management support systems. 3rd edition. Macmillan, New York.

Uran, O., Janssen, R. (2003). Why are spatial decision support systems not used? Some experiences from the Netherlands. Computers, Environment and Urban Systems 27, pp. 511-526.

Van Maanen, J. (1979) The fact of fiction in organizational ethnography. Administrative Science Quarterly, 24, 4, pp. 530-550.

Vollebregt, A., Ten Teije, A., Van Harmelen, F., Van der Lei, J., Mosseveld, M. (1999). A study of PROforma, a development methodology for clinical procedures. Artificial intelligence in Medicine, 17, pp. 195-221.

Walsham, G. (1995) Interpretive case studies in IS research: nature and method. European Journal of Information Systems, 4, 2, pp. 74-81, Macmillan Press, and the Operational Research Society.

Watson, H.J., Houdeshel, G., Rainer, R.K. Jr. (1997). Building executive information systems and other decision support applications. John Wiley & sons Canada.

Weick, K. (1969). The social psychology of organizing. Addison-Wesley, Reading, MA.

Westmacott, S. (2001). Developing decision support systems for integrated coastal management in the tropics: Is the ICM decision making environment too complex for the development of a useable and useful DSS? Journal of Environmental Management, 62, pp. 55-74.

Yates, A. (1991). Procurement and Construction Management. Investment, Procurement and Performance in Construction. E. and FN Spon.

Young, B. (1991) Applications of IT and its impact on the education and training needs for professionals in construction, in: Applications of Information Technology in Construction. Institution of Civil Engineers.

Zwass, V. (1998). Foundations of information systems. Irwin McGraw-Hill, New York.